VILNIAUS UNIVERSITETAS

EKONOMIKOS VERSLO IR ADMINISTRAVIMO FAKULTETAS

FINANSŲ KATEDRA

Finansų ir bankininkystės magistro programa

II kurso studentas

Lukas Buitkus

MAGISTRO DARBAS

UŽSIENIO VALIUTŲ KURSŲ PROGNOZAVIMAS, PANAUDOJANT DIRBTINIUS NEURONINIUS TINKLUS

Leidžiama ginti _____

(parašas)

Katedros vedėja Doc. Dr. Deimantė Teresienė

Magistrantas _____

(parašas)

Darbo vadovė_____

(parašas)

Doc. Dr. Deimantė Teresienė

Darbo įteikimo data. _____

Registracijos Nr. _____

Vilnius, 2020

VILNIAUS UNIVERSITETAS

EKONOMIKOS VERSLO IR ADMINISTRAVIMO FAKULTETAS

FINANSŲ KATEDRA

Lukas BUITKUS

Finansų ir bankininkystės magistro programa

MAGISTRO DARBAS

UŽSIENIO VALIUTŲ KURSŲ PROGNOZAVIMAS, PANAUDOJANT DIRBTINIUS NEURONINIUS TINKLUS

FORECASTING FOREIGN EXCHANGE RATES USING ARTIFICIAL NEURAL NETWORK

Leidžiama ginti _____ Magistrantas _____

(parašas) (parašas)

Katedros vedėja Darbo vadovė _____

Doc. Dr. Deimantė Teresienė (parašas)

Doc. Dr. Deimantė Teresienė

Darbo įteikimo data _____ Registracijos Nr. _____

Vilnius, 2020

TURINYS

ĮVADAS

Augant tarptautiniui bendradarbiavimui tarp šalių, vis labiau daugėja ekonominių ir finansinių operacijų. Kiekviena tokia operaciją, vykstanti už šalies ribų, dažniausiai reikalauja valiutos konvertavimo. Šios operacijos sukuria pagrindą dėl kurio pradėjo formuotis ir iki šiol veikia valiutų rinka. Valiutų rinka, dar vadinama FOREX rinka, yra įvairios užsienio valiutos pirkimo ir pardavimo sandorių visuma. Ji neapsakomai greitai tapo pati globaliausia ir dinamiškiausia rinka pasaulyje. Globališkumas pasireiškia tuo, kad valiutų rinka fiziškai nėra vienoje vietoje, pasak I. Girsaitės ir G. Žilinskij (2016), tai tinklas, jungiantis visą pasaulį ištisą parą. Šitą teiginį nereiktų suprasti visiškai tiesiogiai, nes jos veikimas nepertraukiamas, jeigu kalbėtumėm apie visus regionus bendrai: Azijos, Europos, Amerikos bei Ramiojo vandenyno. Tam tikras regionas turi savo sesijų laiką, kurios metu galima prekiauti valiutomis. Toks rinkos pasiskirstymas ir geografinis išdėstymas lemia, kad rinka iš tiesų būtų aktyvi dvidešimt keturias valandas per parą. Sandorio metu, tuo pačiu metu perkama viena užsienio valiuta ir parduodama kita. Forex rinka kasdien pateikia labai daug duomenų, susijusių su rinkos pokyčiais, kas turi įtakos vienų ir kitų valiutų kainų mažėjimui arba augimui. Atsižvelgiant į tai, kad ši rinka yra dinamiškiausia, valiutų kursai keičiasi kas sekundę, o dideli kainų pokyčiai gali pakenkti ne tik investuotojams, bet ir tarptautinėms įmonėms, bankams, ir dėl to gali nukentėti net šalies ekonomika pasaulinėje rinkoje. Dėl to išauga svarba laiku sureaguoti į rinkos pokyčius. Norint kuo greičiau ir tiksliau suprognozuoti galimus pokyčius, dėl staigaus technologijos patobulėjimo, imta naudoti mašininio mokymosi algoritmus.

Temos aktualumas. Valiutų rinka yra labai svarbi ekonomikai, kadangi svarus pokytis joje gali sukelti „sniego griūžtės“ efektą, persiduodantį į kitas rinkas. E. Abounoori ir kt. (2012) savo straipsnyje pabrėžė, kad Forex rinkos efektyvumas yra labai svarbus kitoms rinkoms, kadangi valiutų kursų pasikeitimai turi visokeriopą poveikį visoms kainoms. Jeigu valiutos bus įvertintos klaidingai, tai lems pačios rinkos neefektyvumą. Šis netikslumas persiduos į kitas rinkas, taip sukurdamas neefektyvų išteklių paskirstymą, kuris pablogins gyventojų kokybę. Dėl pačios rinkos neefektyvumo, valiutų kursai gali pradėti staigiai šokinėti, kas sumažins investuotojų srautą ir padidins tos valiutos riziką. Pastaruoju metu, finansų rinkai vis labiau robotizuojantis, vis didesnio dėmėsio sulaukia dirbtinis intelektas, kuris yra kuriamas siekiant pagerinti žmogiškąjį gebėjimą numatyti būsimus įvykius, vadovautis nepriekaištingomis taisyklėmis ir svarbiausia, nedaryti žmogiškų klaidų ir visada priimti tik teisingus sprendimus. Pasak, I. Kekytės ir V. Stasytytės (2017), jo naudojimas finansų rinkoje lėmė naujų įdomesnių modelių atsiradimą, grindžiamą netiesiniais ir nestacionariais metodais. Naujieji modeliai sugeba susidoroti su duomenų chaotiškumu ir neapibrėžtumu. Tokie modeliai, kaip dirbtinis

neuronų tinklas ar neraiškioji logika buvo puikiai pritaikyti akcijų rinkai prognozuoti, todėl svarbu ištirti ar tokių modelių algoritmai gali būti pritaikyti ir daug dinamiškesnėje valiutų rinkoje. Dažniausiai valiutų rinkos tyrimuose vertinama pati valiutų rinka, nagrinėjama analizės būdai, tačiau mažai randama mokslinių tyrimų apie įvairių modulių pritaikymą prognozuojant valiutų kursus (Butkus, M. ir Tamašauskas, M., 2016). Šiandieninėje pasaulio ekonomikoje yra labai svarbu tiksliai numatyti užsienio valiutos kursą ar bent jau teisingai prognozuoti pasikeitimų tendenciją, todėl yra naudinga moksliškai ištirti įvairius modelius, kuriais vadovaudamiesi autoriai prognozavo valiutų kursų pasikeitimus, ir autorių moksliniais straipsniais pagrįsti geriausią modelį, kurį pritaikius, galetų būti tiksliausiai prognozuojami valiutų kursų pasikeitimai.

Mokslinė problema. Didėjant informacijos kiekiui, kartu didėja ir poreikis pritaikyti kuo efektyvesnius ir tikslesnius metodus, kurių pagalba būtų galima priimti teisingus modelius bei išvengti neplanuotų drąstiškų valiutų kursų pasikeitimų ateityje. Naujų modelių ieškojimas priveda prie naujausių ir dabartiniame etape didžiausią mokslininkų dėmesio sulaukiančio dirbtinio intelekto panaudojimo. Iš pradžių jie buvo pritaikyti nustatant sukčiavimo atvejus, prognozuojant orus, numatant taršą, tačiau ilgainiui jie pradėti taikyti ir finansinėms rinkoms. Todėl svarbu išnagrinėti, ar galima dirbtinį intelektą pritaikyti prognozavimui valiutų rinkoje ir neuroninių tinklų algoritmų pagalbą sukurti modelį, kuris tiksliausiai prognozuotų ateities valiutų kursus?

Darbo objektas – dirbtinių neuroninių tinklų prognozavimo modelių panaudojimas valiutų rinkoje

Darbo tikslas. Remiantis užsienio ir Lietuvos autorių medžiaga, atlikti valiutų kursų prognozavimui pritaikytų dirbtinių neurononių tinklų metodų analizę bei pasirinkto algoritmo pagalbą sudaryti modelį, kuris padėtų prognozuoti ateities valiutų kursus.

Uždaviniai, išsikelti **darbo tikslui pasiekti**:

1. Išnagrinėti valiutų rinką bei nustatyti svarbiausius veiksnius, darančius įtaką valiutų kursų pasikeitimams;
2. Išanalizuoti neuroninius tinklus, jų koncepciją, klasifikaciją bei tinklo mokymą;
3. Atlikti dirbtinių neuroninių tinklų metodų valiutų rinkoje analizę;
4. Remiantis dirbtinių neuroninių tinklų tyrimais sudaryti modelį, kuriuo bus prognozuojami pasirinkti valiutų kursai;
5. Pritaikant sudarytą modelį, suprognozuoti pasirinktų valiutų porų kursus ir atlikti gautų rezultatų vertinimo analizę.
6. Susimuliuoti pasirinktų valiutų porų kursus pasirinktais laikotarpiais į priekį.

Darbo metodai. Mokslinės literatūros analizė ir sintezė; statistinė duomenų analizė; abstrakcijos metodas bei Python programine įranga atlikti įvairūs matematiniai, ekonometriniai ir statistiniai skaičiavimai bei diagramos

Darbo struktūra. Darbas susideda iš trijų skyrių, įvado bei išvadų ir pasiūlymų.

Pirmo skyriaus pirmoje dalyje trumpai ir sistemiškai pateikiami svarbiausi tarptautinės valiutų rinkos aspektai. Taip pat, remiantis užsienio mokslininkų darbais, išskiriami ir detaliai nagrinėjami pagrindiniai veiksniai bei jų įtakas valiutų kursų pasikeitimams. Antroje pirmo skyriaus dalyje analizuojami neuroniniai tinklai ir jų vaudmuo valiutų rinkoje. Nagrinėjama dirbtinių neuroninių tinklų koncepcija bei išskiriama pagrindinė dirbtinių neuroninių tinklų klasifikaciją, ištiriant bei analizuojant plačiai naudojamus neuroninių tinklų algoritmus. Nagrinėjami modelio mokymo būdai, bei pateikiami svarbiausi jų bruožai.

Antrame skyriuje naudojant literatūros analizės metodą, tiriami Lietuvos ir užsienio autorių moksliniai darbai, kuriuose nagrinėjami valiutų rinkos prognozavimo tyrimai, panaudojus dirbtinius neuroninių tinklų modelius. Galiausiai, remiantis užsienio autorių darbais, sudaromas dirbtinių neuroninių tinklų modelis, pagal kurį bus atliekamas empirinis tyrimas su pasirinktų šalių valiutų kursų duomenimis.

Empirinėje dalyje naudojant Python programavimo kalbą bei gausybę bibliotekų, remiantis antrame skyriuje sudaryta metodologija, kuriamas valiutų kursų prognozavimo modelis bei atliekami eksperimentiniai bandymai su pasirinktų valiutų kursų porų istoriniais duomenimis. Panaudojant ilgos trumpalaikės atminties neuroninio tinklo algoritmą programavimo pagalba atliekami valiutų kursų prognozavimo darbai, bei statistiniais rodikliais patikrinami gauti rezultatai. Patikrinus gautus rezulatus bei modelio tinkamumą, suprognozuojami valiutų porų kursai pasirinktais laikotarpiais į priekį.

Skyriuje „išvados ir siūlymai“ pateikiami svarbiausi pastebėjimai, išsakoma nuomonė dėl neuroninių tinklų naudojimo perspektyvų valiutų kursų prognozavime bei pateikiami pasiūlymai tolimesniems tyrimams.

Darbo pabaigoje pateikiama darbo metu naudota literatūra bei priedai, kuriuose galima rasti medžiagą, skaičiavimus bei informaciją, reikalinga baigiamajam darbui.

1. VALIUTŲ RINKŲ IR DIRBTINIŲ NEURONINIŲ TINKLŲ TEORINIAI ASPEKTAI

Valiutų rinka yra viena iš didžiausių ir likvidžiausių rinkų prekybininkams ir investuotojams, kur informacija visiems yra viešai prieinama (Diego, Ildar ir Oleksiy, 2017). Forex rinkoje dalyvauja labai didelė įvairovė objektų (nuo centrinių bankų iki smulkiųjų investuotojų), kurie turi skirtingus tikslus, taip rinką paverčiant dar labiau patrauklesnę, likvidesnę ir su dar didesniu potencialu vystytis. Įvairios dalyvių grupės turi ne tik savus tikslus, bet ir tam tikrą reikšmę rinkoje. Smulkieji investuotojai valiutų rinkoje siekia asmeninės naudos iš valiutų kursų pasikeitimų. Ir nors kiekvienas smulkus investuotojas valdo mažą kapitalą bei savo veiksmais negali daryti įtakos rinkai, dėl tokių investuotojų gausos, per dieną ši rinkos dalyvių grupė atlieka daugiau nei 90% visų rinkos tranzakcijų. Tuo tarpu rinkos formuotojais (angl. market-makers) valiutų rinkoje vadinami bankai, tarptautinės kompanijos bei stambūs prekiautojai, tarpusavyje glaudžiai bendradarbiauja, kad stabilizuotų, padidintų arba sumažintų savo valiutų vertę (Znaczko, 2013). Didžiausią įtaką valiutų kursams gali daryti centriniai bankai, kurių pagrindinės funkcijos yra valstybinių rezervų valdymas bei konvertuojamo kurso stabilumo užtikrinimas. Šias funkcijas įvykdyti centrinis bankas naudojasi tiesioginėmis (valiutinės intervencijos) arba netiesioginėmis priemonėmis. Tiesioginės valiutinės intervencijos reiškia valstybės obligacijų supirkimą. Atlikdamas šį veiksmą, centrinis bankas į rinką išleidžia daugiau pinigų, o valiutos kiekio padidėjimas automatiškai sumažina valiutos kursą. Netiesiogiai valiutos kiekį rinkoje centrinis bankas gali reguliuoti per palūkanų normą, diskonto normą bei bankuose laikomų privalomųjų valiutos atsargų normą. Kai centrinis bankas nustato žemą palūkanų normą, žmonės nėra suinteresuoti laikyti savo indėlius bankuose, nes už tai jie gauna mažas palūkanas, dėl to didesnis pinigų kiekis cirkuliuoja rinkoje, o tai sumažina valiutos kursą. Centriniai bankai kontroliuoti valiutų kursą taip pat gali per diskonto normą - kaina, už kurią komerciniai bankai gali skolintis iš centrinio banko, bei privalomųjų valiutos atsargų normą.

Prieš pereinant prie valiutų rinkos prognozavimo bei vertinimo kada valiutos gali išaugti arba nuvertėti, svarbu išanalizuoti veiksnius, kurie turi įtakos valiutų kursams. Valiutų kursus lemia daugybė sudėtingų veiksnių, ir nors jų įtaka ne visada lengvai paaiškinama, norint perprasti valiutos rinkos pulsą, svarbu, kad investuotojai suvoktų kokią įtaką šie veiksniai sukelia valiutų vertėms ir kaip veikia valiutų kursus.

1.1. Valiutų kursų svyravimus įtakojantys veiksniai

Užsienio valiutų kursas yra vienas iš svarbiausių rodiklių, kuriuo remiantis nusakomas santykinis šalies ekonomikos lygis. Šalies valiutos kursas suteikia galimybė kontroliuoti ekonomikos stabilumą, todėl jis yra nuolat stebimas ir analizuojamas. Dažniausiai valiutos kursas yra apibrėžiamas kaip kursas, kuriuo vienos šalies valiuta gali būti konvertuojama į kitą. Kadangi daugelio pagrindinių pasaulio valiutų kursų yra laisvai „plaukiojantys“, todėl jų valiutos vertė priklauso nuo pasiūlos ir paklausos. Kursas gali kisti kiekvieną dieną, taip keičiant valiutų pasiūlos ir paklausos jėgas, dėl to yra svarbu suprasti kas daro įtaką valiutų kursų pasikeitimams. Siekiant išsiaiškinti kokie veiksniai turi įtakos valiutų kursų svyravimams, buvo atlikta užsienio autorių tyrimų analizė, kurioje jie išskyrė pagrindinius veiksnius, veikiančius valiutos kursus.

Literatūroje nėra bendro sutarimo dėl veiksnių, turinčių įtakos valiutų kursams ir jų kintamumui. Siekdami ištirti kokie veiksniai turi įtakos valiutų kursų pasikeitimams, daugybė autorių savo tyrimais nagrinėjo įvairius veiksnius (žr. 1 lentelę).

1 lentelė. Autorių išskirti makroekonominiai veiksniai, turintys įtakos valiutų kursų svyravimams

Tyrėjai	Veiksniai, darantys įtaką valiutos kursui
Twarowska, K. (2000)	Infliacija Palūkanų norma Einamosios sąskaitos deficitas Vyriausybės skola
Zada, B (2010)	Infliacija Palūkanų norma Užsienio valiutos atsargos
Perveen, Sh. Khan, A. Q. ir Ismail, M. (2012)	Infliacija Ekonomikos augimas
Mirchandani, A. (2013)	Infliacija Palūkanų norma Einamosios sąskaitos deficitas
Ramasamy, R., Aber K. S. (2015)	Palūkanų norma Mokėjimų balansas Infliacija
Vidyavathi, B. (2016)	Infliacija Palūkanų norma Vyriausybės skola BVP Tiesioginės užsienio investicijos

Šaltinis: sudaryta autoriaus, remiantis užsienių autorių tyrimais

Paprastai šie veiksniai skirstomi į dvi gupes: neekonominius ir ekonominius veiksnius, o pastarieji turi dar platesnį išskirstymą į ilgalaikius ir trumpalaikius. K. Twarowska ir kt. (2000) savo darbe, remdamasi kitų autorių tyrimais prie trumpalaikių ekonominių veiksnių priskiria ekonomikos augimo tempą, infliaciją, palūkanų normą, einamąją sąskaitą, valiutos spekuliaciją, o prie ilgalaikių veiksnių nurodo šiuos rodiklius: šalies ekonominio išsivystymo lygį, konkurencingumą, technologinį vystymąsi, vyriausybės skolos dydį, biudžeto deficitą,

santykines užsienio kainas. Autorė, kaip neekonominius veiksnius, turinčius įtakos valiutų kursų svyravimams išskiria politinę riziką, stichines nelaimes ir psichologinius veiksnius. Vis dėl to, siekdami pagrįsti valiutų kursų pasikeitimus, autoriai dažniausiai naudoja ekonominius kintamuosius, kadangi šie rodikliai gali turėti skaitinę išraišką, juos lengviau pritaikyti ekonometriniuose bei kituose matematiniuose modeliuose. Ta pati autorė K. Twarowska vėliau savo tyrime atlikdama regresinę analizę pasirinko tirti ekonominius veiksnius, lemiančius Lenkijos nacionalinės valiutos – zloto ir euro kurso pasikeitimus. Atlikto tyrimo rezultatai parodė, kad einamosios sąskaitos deficitas ir infliacija buvo svarbiausi veiksniai, turintys įtakos EUR/PLN valiutų kursams. Tarp veiksnių, kurie taip pat turėjo įtakos valiutų kursui buvo išskirta palūkanų norma bei vyriausybės skola.

Vėlesniame laikotarpyje kita autorė B. Zada (2010) analizavo veiksnius, turinčius įtakos Pakistano valiutos kursui 1979 – 2008 m. laikotarpyje. Tyrime buvo naudojamas daugianaris regresijos modelis, kuriame valiutos kursas buvo laikomas priklausomu kintamuoju, o infliacija, palūkanų norma, užsienio valiutos atsargos, mokėjimų balansas, pinigų pasiūla ir bendrasis vidaus produktas – nepriklausomais kintamaisias. Gauti rezultatai parodė, kad infliacija, palūkanų norma ir užsienio valiutos atsargos dar didelę įtaką valiutų kursui, tuo tarpu BVP, pinigų pasiūla ir prekybos deficitas nebuvo reikšmingi. Kiek naujesnis darbas, tyrinėjant Pakistano valiutos kursą bei juos įtakojančius makroekonominius rodiklius, buvo atliktas Sh. Perveen, A. Q. Khan ir M. Ismail (2012). Valiutų kursų kintamumui patikrinti buvo naudojami šie kintamieji: infliacija, ekonomikos augimo tempas, bei eksportas ir importas. Rezultatams tirti buvo naudojamas paprastas tiesinės regresijos modelis su įprastu mažiausiu metodu (angl. Simple Linear Regression model with ordinary least method (OLS)). Atliktas tyrimas parodė, kad pagrindinis Pakistano valiutos kursą veikiantis veiksnys yra infliacijos tempas. Infliacija darė neigiamą poveikį valiutos kursui, kas reiškė, kai infliacija padidėja, valiuta nuvertėja. Kitas veiksnys, kuris irgi turėjo įtakos valiutų kursui buvo šalies ekonomikos augimas.

Kitas autorius, A. Mirchandani (2013) savo darbe taip pat išnagrinėjo įvairius makroekonominius kintamuosius, kurie turi įtakos valiutos kurso kintamumui. Naudojantis statistinėmis priemonėmis buvo nustatytas ryšys tarp Indijos rupijos ir JAV dolerio valiutos kurso ir makroekonominių kintamųjų, tokių kaip palūkanų norma, prekybos balansas, infliacija, TUI, BVP ir kt. Tyrimui buvo imti 1991 – 2010 m. metiniai duomenys. Infliacija, palūkanų norma ir einamosios sąskaitos deficitas buvo išskirti kaip veiksniai, kurie stipriai koreliuoja su užsienio valiutos kurso pokyčiais. Papildomą tyrimą siekiant išnagrinėti pagrindinius makroekonominius rodiklius, kurie turi įtakos Indijos valiutos vertei analizavo ir B. Vidyavathi (2016). Šiame tyrime buvo atliktas koreliacijos testas, kuriame buvo tikrinamas makroekonominių veiksnių ir valiutos kurso sąryšio laipsnis. Buvo nustatytas neigiamas

santykis tarp šių kintamųjų - BVP, infliacijos, palūkanų normos, išorės skolos bei valiutų kurso, ir silpnas teigiamas koreliacinis ryšys tarp tiesioginių užsienio investicijų bei IND/USD valiutos kurso.

2015 m. vasario mėn. R. Ramasamy ir S. K. Abar parašytame straipsnyje (2015) autoriai panaudojo trijų šalių metinius valiutų kursus ir makroekonominius kintamuosius, kad ištirtų jų įtaką valiutoms. Tyrimo modeliui buvo pasirinkta naudoti AUD/USD, EUR/USD ir AUD/EUR valiutų kursus. Šie valiutų kursai buvo pasirinkti, kadangi Jungtinės Amerikos Valstijos, Australija ir Vokietija, kur pastaroji reprezentuoja eurą, yra stiprios ekonomikos, turinčios minimalų nedarbą, korupciją ir mažą biudžeto deficitą. Įtaką valiutų kursams buvo pasirinkti devyni makroekonominiai kintamieji, kurie vienaip ar kitaip veikia kursą. Tyrimo tikslas buvo nustatyti reikšmingiausią iš jų. Gauti rezultatai parodė, kad stipriausias priežastinis ryšys yra tarp valiutos kursų ir palūkanų normos, mokėjimų balanso bei infliacijos.

Taigi, atsižvelgiant į išnagrinėtų autorių tyrimus, galime pastebėti, kad visuose darbuose pagrinde yra išskiriama keletas pagrindinių veiksnių, lemiančių valiutų kursus, ir visi jie susiję su prekybos santykiais tarp dviejų šalių, tai ganėtinai logiška, kadangi valiutų kursai yra santykiniai dydžiai ir išreiškiami kaip dviejų šalių valiutų palyginimas. Remiantis išnagrinėtų autorių darbais pagrindiniai makroekonominiai veiksniai, turintys didžiausią įtaką valiutų kursams yra infliacija, palūkanų normos lygis, mokėjimų balansas arba einamoji sąskaita, prekybos sąlygos ir vyriausybės skola. Taip pat yra išskiriamas vienas veiksnys – politikos stabilumas ir efektyvumas, kuris yra labiau susijęs su valstybės politika, nei su ekonominiais reiškiniais (žr. 1 paveikslą). Toliau, norint susidaryti įgyti platesnį supratimą apie makroekonominių veiksnių įtaką, yra analizuojama išskirtų veiksnių svarba ekonomikai bei jų reikšmė valiutų kursui.

1 paveikslas. Pagrindiniai makroekonominiai veiksniai, turintys įtakos valiutų kursams (Šaltinis – sudaryta autoriaus)

Vienas svarbiausių makroekonominių veiksnių, kuris vaidina svarbų vaidmenį vertinant bet kurios šalies valiutą yra infliacija. Dažniausiai infliacijai apskaičiuoti naudojamas vartotojų kainų indeksas (angl. Consumer Priece Index, CPI), kuris yra išreiškiamas procentais. Jis palygina fiksuoto reprezentatyvaus prekių ir paslaugų krepšelio kainos pokytį per tam tikrą laiką (Jagerson, Hansen, 2006). Infliacijos pasikeitimas turi priešingą priklausomybę valiutų kursams. Paprastai, šalyje, kurioje infliacijos lygis yra mažas palyginti su kitomis valstybėmis, valiutos vertė nuolat auga dėl padidėjusios perkamosios galios. Tai padidina vietinės valiutos paklausą. Tuo pačiu užsienio prekės tampa pigesnės, todėl vietiniai gyventojai moka mažiau, taip sumažinant importą. Tokiu atveju žemesnė infliacija šalyje paprastai lemia padidėjusią šalies valiutos vertę. Tuo tarpu šalys su didesne infliacija, paprastai turi nuvertėjusią valiutą, palyginti su jų prekybos partnerių valiutomis. Tarptautinio valiutos fondo duomenimis, 2019 metais mažiausią infliaciją fiksuoja Vakarų Europos (1,4 %), Australijos (2 %) ir Šiaurės Amerikos (2,2 %) regionai, tuo tarpu dideli infliacijos tempai pastebimi Sub-Saharinėje Afrikos dalyje (9,5 %) ir Centrinės Azijos ir Kaukazo regione (13,7 %). Nors Pietų Amerikos regiono susisteminti duomenys nėra pateikiami, galima nuspėti, jog šis žemynas susiduria su didžiausiomis hyperinfliacijos problemomis: Argentinoje šiais metais fiksuojama 43,7% infliacija, o Venezueloje jau kurį laiką laikosi nesuvokiama 10 mil. % hyperinfliacija.

Jei šalyje vyriauja infliacija, centrinis bankas turi imtis veiksnių, todėl jis greičiausiai padidins bazinę skolinomisi normą (palūkanų normą), kad sumažintų žmonių ir bendrovių pinigų pasiūlą ir padarytų skolinimąsi brangesniu. Įprastai, centrinis bankas vykdydamas ekonominį valdymą, palūkanų normas koreguoja kas ketvirtį (Ramasamy, R., Aber K. S., 2015). Didelės palūkanų normos, palyginti su kitomis šalimis, yra patrauklesnės investuotojams, kadangi ji suteikia didesnę šalies turto grąžą. Kadangi šalis su didelėmis palūkanų normomis yra patraukli užsienio investuotojams, taip pritraukiama daugiau užsienio kapitalo, kas lemia valstybės valiutos vertės padidėjimą. Šį teiginį patvirtina K. Twarkovska (2014), kuri teigia, kad jei šalis išlaiko santykinai aukštą palūkanų normą, paprastai ji trumpalaikiu laikotarpiu pritraukia didelius kapitalo srautus ir šios šalies valiutos vertė kyla. Taigi bazinės palūkanų normos didinimas teigiamai veikia valiutą, kuri tam tikrą laiką brangsta. Vis dėlto, didesnė palūkanos normos turi didelę įtaką ir vietiniam šalies verslui. Aukštesnės palūkanų normos mažina šalies vartotojų perkamąją galią, blogina situaciją žmonėms, kurie yra pasiėmę paskolą, kadangi jiems reikia mokėti didesnes palūkanas.

Dar vienas išskirtas veiksnys, turintis įtakos valiutos kursams yra mokėjimų balansas. Mokėjimų balansą sudaro einamoji ir finansinė saskaitos. Šalies einamoji sąskaita yra prekybos tarp šalies ir jos prekybos partnerių balansas, atspindintis visus mokėjimus tarp šalių už prekes ir paslaugas. Finansinėje sąskaitoje įrašomos tiesioginių užsienio investicijų portfelio įplaukos

ir nutekėjimai. Abi šios sąskaitos kartu sudaro užsienio valiutos atsargas, turimas šalyje (Ramasamy, R., Aber K. S., 2015). Einamosios sąskaitos deficitas susidaro, jei šalis išleidžia daugiau importuojamiems produktams, nei ji uždirba iš eksporto. Šalys, kurios einamojoje sąskaitoje turi perteklių, yra naudingesnės, nei šalys su einamosios sąskaitos deficitu. Teigiamas saldo arba neigiamo mokėjimo balanso mažėjimas yra nacionalinės valiutos vertės augimo veiksnys (Jagerson, Hansen, 2006). Jei užsienio prekybos balansas yra teigiamas, šalis gauna daugiau užsienio valiutos nei išleidžia. Ji keičiama į vidaus valiutą, todėl padidėja užsienio valiutos pasiūla. Dėl to pastarosios kursas krenta, o vidaus valiutos kursas kyla. Vis dėl to, dažniau pastebima atvejų, kai šalys turi deficitą, todėl jos turi skolintis pinigų iš kitų valstybių, kad padengtų tą skirtumą. Perteklinė užsienio valiutos paklausa mažina šalies valiutos kursą, kol vietinės prekės ir paslaugos užsieniečiams atrodo pakankamai pigios. Tai lemia, kad užsienio investuotojai praranda pasitikėjimą šalimi ir pradeda atsiimti savo investicijas. Vietinės valiutos pasiūla išauga, dėl to valiutos vertė krenta.

Prekybos sąlygos yra susijusios su santykiu, kuris lygina eksporto ir importo kainas. Jei šalies eksporto kaina didėja greičiau nei importo, jos prekybos sąlygos pagerės. Gerėjančios prekybos sąlygos rodo didesnę šalies eksporto paklausą, tai savo ruožtu lemia augančias eksporto pajamas, o tai padidina šalies valiutos paklausą ir jos valiutos vertę. Jei eksporto kaina padidės mažiau nei jo importo kaina, valiutos vertė sumažės prekybos partnerių atžvilgiu. Prekybos sąlygos turi labai glaudų ryšį su einamąja sąskaita ir mokėjimų balansu.

Šalys su didele valstybės skola yra dar vienas rodiklis, kuris veikia valiutos veiksnius. Tokios šalys, nesugebėdamos gauti lėšų surenkant mokesčius, yra privestos skolintis iš kitų šalių, kas yra mažiau patrauklu užsienio investuotojams, todėl į šalį bus pritraukta mažiau užsienio kapitalo, investuotojai atsiims savo lėšas, kad išvengtų finansinių problemų, kas sumažins valiutos vertę bei padidins potencialią infliacijos grėsmę. Kas reiškia, kad padidėjus infliacijai, skola turės būt padengiama pigesne šalies valiuta. Užsienio bankai ir investuotojai, kartu su šalies bankais ir finansinėmis institucijomis yra pagrindiniai kreditoriai, perkantys vyriausybės obligacijas ir iždo vekselius, stengdamiesi padengti turimą deficitą. Blogiausiu atveju, vyriausybė, kad sumokėtų skolą, gali pradėti spausdinti pinigus, kas dar labiau padidins pinigų pasiūlą ir sukels dar didesnę infliaciją. Be to, jei šalies vyriausybė nesugebės padengti deficito vidinėmis priemonėmis (pardavinėdama vietines obligacijas, padidindama pinigų pasiūlą), tada ji turės padidinti parduodamų vertybinių popierių pasiūlą užsieniečiams. Tai padidins užsienio valiutos įplaukas vidaus rinkoje. Galiausiai, didelė skola gali sukelti nerimą, kad šalis nesugebės įvykdyti savo įsipareigojimų, ir investuotojai bus linkę mažiau įsigyti vertybinių popierių ta valiuta, jei įsipareigojimų nevykdymo rizika bus didelė. Dėl šios

priežasties kai kurių tarptautinių organizacijų kaip Standard & Poor's ir Moody's šalies skolos reitingas turi labai svarbią reikšmę šalies valiutų kursui.

Šalies politinė situacija ir ekonominiai rezultatai taip pat gali turėti įtakos šalies valiutai. Užsienio investuotojai neišvengiamai ieško stabilių šalių, turinčių gerų ekonominių rezultatų, šalių, kuriose yra mažiau politinių neramumų, kad galėtų jose investuoti savo kapitalą. Valstybė atitinkanti šiuos požymius, nuvilios investicijų fondus nuo kitų šalių, kurioms kyla didesnė politinė ir ekonominė rizika. Nestabili vyriausybė mažina užsienio investuotojų pasitikėjimą šalimi, kas sumažina užsienio kapitalą ir sumažina šalies valiutos vertę. Šalies ekonomikai ištikus nuosmukiui, kris jos palūkanų normos, kas sumažins galimybes įsigyti užsienio kapitalo. Tai reiškia, kad tuo laikotarpiu valiuta susilpnės ir valiutos kursas, palyginti su kitų šalių, sumažės. Prie šios grupės priskiriami ir įvairūs šalies rinkimai, karai, pinigų politikos pokyčiai, kurie daro didelę įtaką šalies valiutai. Pavyzdžiui, karas šaliai gali ne tik padidinti nestabilumą regione, bet ir sukelti didžiulę ekonominę įtampą, kas gali turėti įtakos tos šalies valiutos vertei.

Be išvardintų kintamųjų, valiutų kursams gali turėti įtakos ir kiti autorių išskirti makroekonominiai rodikliai, tokie kaip bendrasis vidaus produktas, nedarbas, gamybos išlaidos, tiesioginės užsienio investicijos, spekuliacija ir kt. Kiekvienas iš šių veiksnių turi savo poveikį valiutos kursų pasikeitimams ir nuo kiekvieno iš jų priklauso valiutos vertės padidėjimas arba nuvertėjimas. Kaip ir daugelyje makroekonomikos dalykų, svarbu atkreipti dėmesį, kad daugelis nagrinėtų veiksnių yra susiję. Pavyzdžiui, didelis infliacijos lygis gali sukelti centrinio banko intervenciją rinkoje, kas padidintų valstybės skolą ar sukeltų kitų veiksnių reakciją. Palūkanų normos, infliacija ir valiutų kursai yra tarpusavyje labai susiję rodikliai. Centriniai bankai gali daryti įtaką tiek infliacijai, tiek valiutų kursui, manipuliuodami palūkanų normomis. Vis dėlto, didesnių palūkanų normų poveikis gali būti sušvelninamas, jei šalyje infliacija yra daug didesnė nei kitose, arba kiti papildomai veiksniai skatina valiutos kurso kritimą.

1.2. Dirbtinių neuronių tinklų vaidmuo valiutų rinkoje

Dėl mokslinės pažangos įvairiose srityse tarp mokslininkų ėmė vis dažniau girdėtis „dirbtinio intelekto“ terminas. 1950 m. L. McCarthy pirmą kartą paminėjo „dirbtinio intelekto“ sąvoką ir nors dabar nėra visuotinai priimto dirbtinio intelekto (angl. artificial intelligence (AI)) apibrėžimo, šiame darbe bus vadovaujamasi Europos Komisijos (2019) pateiktu apibūdinimu, kuriame dirbtinis intelektas (DI) yra apibrėžiamas kaip sistema, kuri analizuodama aplinką ir savarankiškai atlikdama sprendimus, demonstruoja protingą ir sumanų elgesį, siekiant pasiekti užsibrėžtą tikslą. Pasak Europos Komisijos įsteigtos Nepriklausomos aukšto lygio eksperų grupės (angl. Independent High-Level Expert Group on Artificial Intelligence), dirbtinio

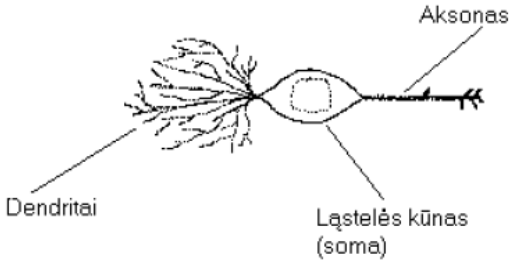
intelekto sistemos gali veikti virtualioje erdvėje vien tik kaip programinė įranga (balso sintezatoriai, paieškos sistemos, kalbos ir vaizdo atpažinimo sistemos ir t.t.) arba gali būti integruota į techninę įrangą (pažangūs robotai, savaeigės transporto priemonės ir t.t). Dirbtinis intelektas, tai toks technologijų rinkinys, kuris pagerina žmogiškąjį gebėjimą numatyti būsimus įvykius, sukurti nepriekaištingas taisykles, priimti reikiamus sprendimus ir atpažinti tinkamus modelius. Dirbtinis intelektas, skirtingai negu paprasti kompiuteriniai algoritmai, sugeba ne tik įsiminti bei atkurti tam tikrą elgesį, bet ir, atsižvelgiant į situaciją, mokytis iš prieš tai atliktų veiksmų, tobulėti, priimti savarankiškus sprendimus bei prognozuoti ateities įvykius. Ne paslaptis, kad laikui bėgant prognozuojama, kad dirbtiniu intelektu veikiamos mašinos pakeis žmonių darbo jėgą. Remiantis naujomis gausiomis mašinų mokymosi tyrėjų apklausos duomenimis, Andy Peart (2017) prognozuoja, kad per ateinančius dešimt metų dirbtinis intelektas pralenks žmonių gebėjimus versti kalbas (prognozuojama, kad tai nutiks 2024 m.), taip pat prognozuojama, kad 2050 m. dirbtinis intelektas galės visiškai atlikti chirurgo profesijos darbus. Tyrėjai, taip pat mano, kad yra 50 procentų tikimybė, kad per ateinančius 45 metus, dirbtinis intelektas sugebės pranokti žmones atliekant visas užduotis, o per 120 metų visiškai jas automatizuoti. Šiuolaikinius žmones mašinos, sugebančios „mąstyti“ bei savarankiškai atlikti įvairiausius veiksmus, koncepcija žavi, bet tuo pačiu gąsdina. Daugelis garsių asmenybių yra išreiškę savo nerimą dėl dirbtinio intelekto: Elonas Muskas viename iš savo interviu teigė, kad „DI yra didžiausia mūsų egzistencinė grėsmė“. Jam pritarė ir Stephen Hawkingas teigdamas, kad „DI kūrimas žmoniją gali paskatinti išnykti, nes žmonės, ribojami lėtos biologinės evoliucijos negalės varžytis su DI, todėl bus išstumti“. Vis dėlto, žmogaus smegenys gali daug greičiau nei bet kuris kompiuteris atlikti tokias užduotis kaip modelio atpažinimas, suvokimas ir motorikos valdymas. Tam pritarė ir Alanas Turing‘as (1950), kai viename iš savo straipsnių teigė, kad nors pavyko sumodeliuoti biologines nervų sistemas, kompiuteris vis dar negali priimti intuityvumu, sąmoningumu bei emocija paremtų sudėtingų problemų sprendimų, kas yra neatsiejama žmogaus intelekto dalis.

Pastaraisiais metais vis svarbesnės tampa intelektinės sistemos, kurios plačiąja prasme remiasi programiniais skaičiavimais (soft computing). Programiniai skaičiavimai imituoja žmogaus suvokimą ir sąmoningumą. Tokios sistemos geba mokytis iš įgautos patirties, todėl gali būti taikomos net tose srityse, apie kurias dar nesukaupta tiesioginių žinių. Be to, pasitelkusios lygiagrečias skaičiavimo architektūras, modeliuojančias biologinius procesus, jos gautus įvesties signalus gali susieti su išvesties signalais daug greičiau, nei taikant nuoseklius analitinius metodus. Stengdamiesi atkartoti žmogaus smegenų veiklą, praėjusio amžiaus ketvirtojo dešimtmečio tyrinėtojai - matematiskas W. Pitts‘as ir neurofiziologas W. McCulloch‘as sukūrė paprastą techninę įrangą biologiniams neuronams ir jų sąveikai

modeliuoti. 1943 m. atsiradus modernesnei technikai, jie sukūrė elektrinę grandinį, imituojančią žmogaus smegenų veiklą. Šeštajame dešimtmetyje, apibendrinus biologinių ir fiziologinių neuronų sampratą, buvo sukurtas pirmasis vieno sluoksnio dirbtinis neuroninis tinklas. Iš pradžių tai buvo elektroninė schema, o vėliau dirbtinis neuroninis tinklas perkeltas į lengviau manipuliuojamą kompiuterinio modeliavimo lygmenį. Pastaraisiais metais neuroniniais tinklais susidomėta dėl to, kad dabartiniai kompiuteriai daug spartesni nei penktajame ar šeštajame dešimtmečiuose. Šiais laikais egzistuoja įvairių rūšių dirbtinių neuroninių tinklų algoritmai ir jie plačiai naudojami visuose sektoriuose – nuo chemijos, fizikos ir medicinos iki finansų sistemų. Tyrėjus neuroniniai tinklai domina gebėjimu pamėgdžioti žmogaus smegenų veiklą ir galimybe mokytis bei reaguoti. Ir nors neuronai yra visiškai paprasti elementai (apsiribojantys tik signalų perdavimu), tačiau dideli tinklai sudaryti iš daugybės neuronų gali greitai ir tiksliai apdoroti didelius kiekius informacijos. Taip pat jie toleruoja klaidas ir gali interpretuoti netikslią situaciją bei prisitaikyti prie naujų situacijų, tad prisitaikymas ir mokymasis yra pagrindiniai neuroninių tinklų tyrimų objektai. Šie tinklai vis labiau populiarėja finansų srityje ir atlieka svarbų vaidmenį vykdant tokias užduotis kaip modelio atpažinimas, klasifikavimas, optimizavimas, robotų valdymas ir laiko eilučių prognozavimo operacijos (K. Chandar, M. Sumathi ir S. N. Sivanandam, 2015). Užduotims atlikti neuroniniai tinklai išmoksta įvesčių ir išvesčių rinkinį, o paskui pritaiko savo žinias aproksimuodami arba prognozuodami įvesčių ir išvesčių priklausomybę.

1.2.1. Neuroninių tinklų koncepcija

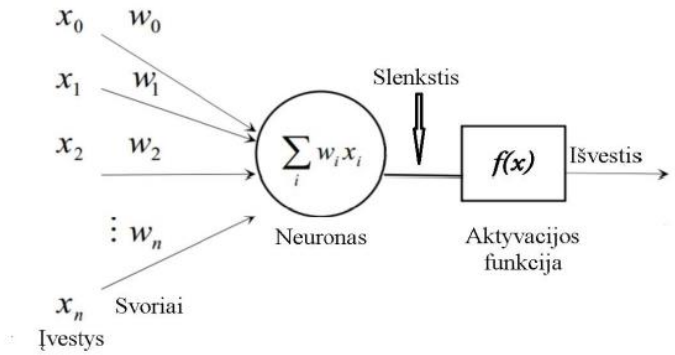
Siekiant supaprastinti dirbtinio intelekto tyrimus, buvo kuriami teoriniai modeliai paremti gyvų organizmų nervų sistemos pagrindu. Biologiniame neurono modelyje (2 pav.) kiekvienas neuronas yra sudarytas iš ląstelės kūno (somos), aksono ir daugybės dendritų (Maknickienė, 2012; Zakaria ir kt., 2014). Aksonas ir dendritai dalyvauja procese, kurio metu nervinis impulsas yra perduodamas iš vieno neurono į kitą. Nervinio impulso perdavimas vyksta per mažus tarpelius tarp išsišakojusių aksono galų ir dendritų, kurie vadinami sinapsėmis. Dendritai priima signalus iš kitų neuronų per įvesties sinapses, o aksonas, sudarydamas sinapsinius ryšius su daugeliu kitų neuronų, per išvesties sinapses perduoda signalus į kitus neuronus.



2 paveikslas. Biologinis neuronas

Gavęs pakankamai įvesties signalų, stimuliuojančių neuroną iki slenkstinio (angl. threshold) lygio, neuronas yra sužadinimamas ir išsiunčia elektrinį signalą, kitaip vadinamą impulsą, savo aksonui. Sužadintas neuronas toliau siunčia signalą aksonu kitiems prie jo prisijungusiems neuronams. Vienas neuronas vienu metu gali jungtis su daugybę neuronų, ir iš viso jis gali sudaryti iki 100 000 sinapsinių jungčių vienu momentu. Jeigu įvesties signalai nepasiekia reikiamo slenkstinio lygio, impulsas greitai nuslopsta, taip ir nesukėlęs jokių matomų padarinių. Taip atskiri neuronai tarpusavyje jungdamiesi sukuria neuroninį tinklą.

Dirbtiniai neuroniniai tinklai (***angl. artificial neural networks, toliau – DNT***) apima informacijos apdorojimo sistemas, paremtas biologinio neurono veikimo principu (Krenker ir kt. 2011; Maknickienė, 2015). Kaip ir biologiniame modelyje, taip ir matematiniame, dirbtinį neuroną sudaro trys pagrindiniai komponentai (3 pav.): svoriai, slenksčiai ir viena aktyvavimo funkcija.



3 paveikslas. Dirbtinis neuronas

Kiekvienas neuronas gauna kelias įvesties (angl. Input) reikšmes (žymima - x =[x_1, x_2, ..., x_n]). Kiekviena įėjimo jungtis turi individualų perdavimo koeficientą – vadinamąjį svorį (angl. Weight) (žymima – W =[W_1, W_2, ..., W_n].). Paprastai įvesčių ir svorių reikšmės yra realieji skaičiai (W $\in$ R). Skirtingas svorio koeficientas, rodo atskirų įvesčių stiprumą, jis atitinka biologinio neurono sinapsių efektyvumą. DNT mokymo algoritmo tikslas yra nustatyti geriausią svorių reikšmių aibę. Kiekvienas įvesties signalas yra dauginamas iš svorio koeficiento bei taip gaunama atskira neuroninė jungtis (žymima – XW arba a_n), kuri formuoja neuronų sinapses bei skaičiuojama pagal formulę:

$$a_n = w_1x_1 + w_2x_2 + \cdots + w_nx_n = \sum_{k=1}^{n} w_kx_k \qquad (1)$$

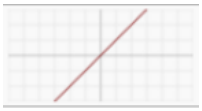
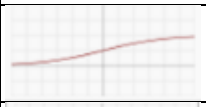
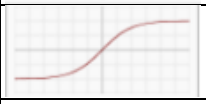
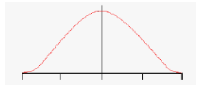
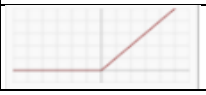
Jei svorio koeficientas neigiamas, tai reiškia, jog jungtis a_n turi slopinantį poveikį išvestyje y, o jei teigiamas, tai reiškia, kad a_n sužadina išvesties signalą. Sinapsių formuojamos reikšmės yra sumuojamos, o galutinę sumą koreguoja aktyvacijos slenkstis (angl. Threshold) (žymimas -w_{n0}). Kiekvienas neuronas turi individualią sužadinimo slenksčio reikšmę (angl. Threshold). Slenksčio sužadinimo reikšmė gaunama naudojant aktyvavimo funkciją ir

skaičiuojant įėjimo signalų svorių sumą atimant iš nustatytos slenksčio reikšmės. Kartais, kai slenksčio reikšmė vadinama poslinkio (angl. Bias) ir ši reikšmė yra pridedama, o ne atimama:

$$y_n = f\left(\sum_{k=1}^{n} w_{nk}x_n(t) + w_{n0}\right) \qquad (2)$$

Funkcijos argumentu tampa suma a_n, vadinama aktyvacijos (perdavimo) funkcija. Pagal sužadinimo signalą, naudojant perdavimo funkciją f(a) yra skaičiuojama neurono išėjimo (angl. Output) reikšmė y_n. Aktyvacijos funkcijos reikšmė atitinka neurono išėjimo reikšmę. Ši funkcija reikalinga tam, kad duomenis patenkantys į neuroninį tinklą būtų tam tikrame intervale. Šie intervalai gali būti (0,1) ir (-1,1). Kokio sudėtingumo aktyvavimo funkcija taikoma, priklauso nuo neuroninio tinklo sprendžiamo uždavinio ir mokymo algoritmo. Bendru atveju rezultatas mažai skiriasi, bet pataikius pasirinkti funkciją gali mažėti apmokymo laikas ir didėti tikslumas. Populiariausios aktyvavimo funkcijos yra šios – tiesinė, slenksčio, sigmoidinė ir hiberbolinė tangento ir ReLu (žr. 2 lentelę).

2 lentelė. Dirbtinių neuroninių tinklų aktyvacijos funkcijų pavyzdžiai *Šaltinis: sudaryta autoriaus*

Pavadinimas	**Formulė**	**Kreivės**	**Reikšmių sritis**
Tiesinė	$f(a) = ka$		$(-\infty, \infty)$
Pusiautiesinė (slenksčio)	$f(a) = \begin{cases} ka, a > 0, \\ 0, \quad a \leq 0 \end{cases}$		$(0, \infty)$
Sigmoidinė (loginė)	$f(a) = \frac{1}{1 + e^{-a}}$		$(0, \infty)$
Hiperbolinis tangentas	$f(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} = \frac{e^{2a} - 1}{e^{2a} + 1}$		$(-1,1)$
Gauso kreivė	$f(a) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(a-\mu)^2}{2\sigma^2}}$		$(-\infty, \infty)$
ReLU	$f(a) = \begin{cases} a, \quad a \geq 0, \\ 0, \quad a < 0 \end{cases} = \max(0, x)$		$(0, \infty)$

Bene labiausiai paplitusi yra sigmoidinė, kitaip – loginė funkcija. Nepaisant funkcijos privalumų, kad ši funkcija yra netiesinė, jos išvestinė yra lengvai skaičiuojama bei dėl reikšmių intervalo neturi rizikos neurono persimokymu (angl. over-fitting), ji turi ir trūkumų, kadangi formulė įgauna tik teigiamas reikšmes - intervale (0; 1). Visgi tiriant dažnai reikalingos ir neigiamos reikšmes. Tokiam atvejui galima naudoti hiperbolinio tangento funkciją, kuri įgauna tiek teigiamas, tiek neigiamas reikšmes intervale (-1; 1). Šiuo metu dažniausiai pasaulyje yra naudojama ReLU aktyvavimo funkcija, nes ji gali įgauti reikšmes nuo nulio iki begalybės. Ji įvardijama kaip tinkamiausia efektyviam neuroninio tinklo mokymui. Iš funkcijos atrodo, kad

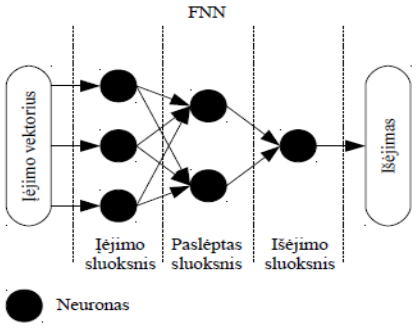
ReLU funkcija yra tiesinė, tačiau tai netiesa. Dar vienas privalumas, kad ReLU aktyvacijos funkcijoje, kitaip nei sigmoidinei, nėra nykstančio gradiento problemos (angl. vanishing gradient), nes gradientas yra konstanta. Vis dėlto, su tuo susijęs ir pagrindinis ReLU funkcijos minustas, kad funkcijai įgijus labai dideles reikšmes yra galimybė persimokyti.

Visos kitos naudojamos aktyvacijos funkcijos nėra tinkamos spręsti netiesiniams uždaviniams, todėl apibendrinant visas plačiau aprašytas aktyvacijos funkcijas galime sakyti, kad jų pasirinkimas labiausiai priklauso nuo dirbtinio neuroninio tinklo pasirinkimo, tačiau įvertinus visus funkcijų privalumus ir trūkumus, daugelyje atvejų praktikoje naudojama ReLU funkcija.

1.2.2. Neuroninių tinklų klasifikacija

Dviejų ar daugiau dirbtinių neuronų sujungimas suformuoja dirbtinį neuroninį tinklą. Jei vienas dirbtinis neuronas yra bevertis sprendžiant realaus gyvenimo problemas, dirbtiniai neuroniniai tinklai netiesiniu (angl. non-linear), paskirstančiuoju (angl. distributed), lygiagrečiu (angl. parallel) ar vietiniu (angl. local) būdu gali išspręsti net sudėtingus uždavinius, apdorodami informaciją pagrindiniuose jų blokuose. Atskirų neuronų apjungimas į bendrą neuronų tinklą yra vadinamas topologija arba architektūra (Krenker, A. ir kt., 2011). Per visą neuroninių tinklų plėtojimo laikotarpį neuronai buvo sujungiami įvairiais būdais (1 priede pateikiamos beveik visos galimos neuroninių tinklų architektūros (Vjodor van Veen, 2016)) ir priklausomai nuo sujungimo būdo ir kokia kryptimi siunčiami signalai, dirbtiniai neuroniniai tinklai yra suskirstomi į dvi pagrindines grupes:

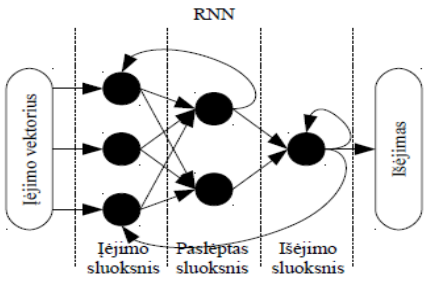
- **tiesioginio sklidimo (angl. feedforward) neuroninį tinklą.** Ši neuroninių tinklų grupė yra viena iš paprasčiausių dirbtinio neuronio tinklo formų. Tiesioginio sklidimo neuroninių tinklų neuronai sujungti taip, kad neuronais sklindantis signalas niekada nesudarytų uždarų grandinių. Tai reiškia, kad vieno sluoksnio įvestys gali jungtis tik su kito (po jo einančio) sluoksnio išvestimis – informacija keliauja tik viena kryptimi, nuo įvesčių iki išvesčių ir negali būti jokių grįžtamųjų ryšių (4 pav.). Šie tinklai neturi jokių sluoksnių skaičiaus, aktyvacijos funkcijos tipo ir sąryšių ribojimų.



4 paveikslas. Tiesioginio sklidimo neuroninis tinklas

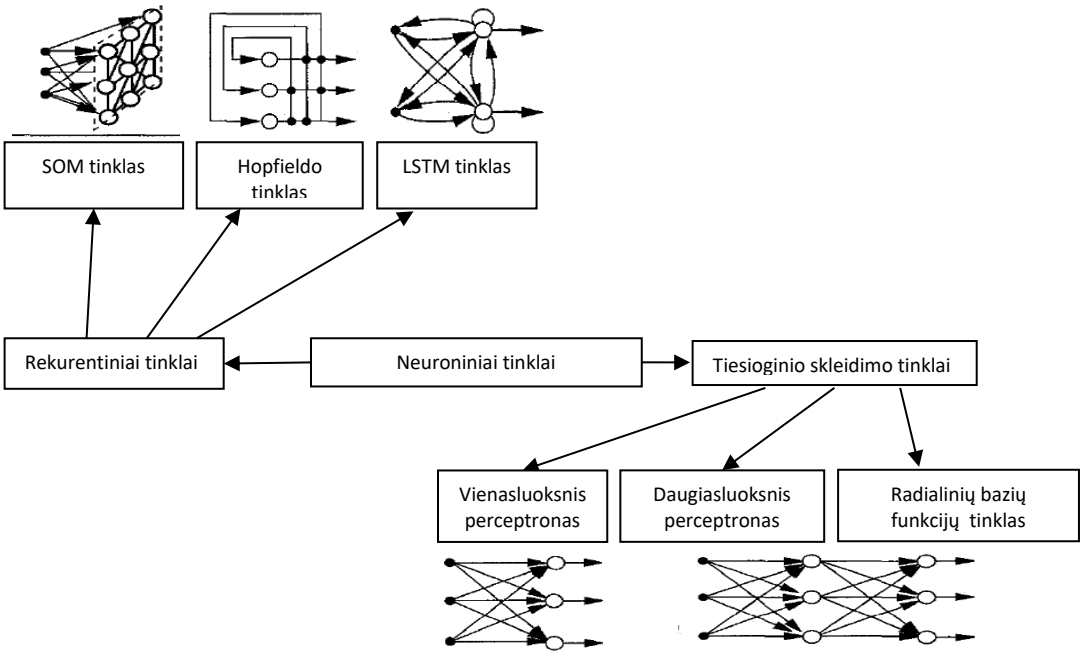
Tiesioginio sklidimo neuroniniai tinklai yra pritaikomi kompiuteriniame matyme (angl. computer vision) – mokslo srityje, kuri nagrinėja, kaip kompuateriai gali įgyti aukšto lygio supratimą iš skaitmeninių vaizdų ar vaizdo įrašų, bei balso atpažinime (angl. speech recognition), kur sudėtinga klasifikuoti tikslines grupes. Tokie neuroniniai tinklai reaguoja į triukšmingus duomenis ir yra lengvai prižiūrimi.

- **grįžtamojo ryšio, kitaip atbulinio skleidimo (angl. feedback) neuroninį tinklą.** Grįžtamojo ryšio neuroniniai tinklai (toliau - GRNT), arba rekurentiniai tinklai (toliau - RNN) gali būti sujungti bet kokia tvarka. Tokie neuroniniai tinklai gali turėti ne tik neribotą skaičių sluoksnių, bet ir neribotą skaičių grįžtamųjų ryšių. Informacija keliauja ne tik iš įvesčių, bet galima ir priešinga kryptis (5 pav.). Tinklai gali naudoti savo vidinę atmintį bet kuriai įvesties sekai apdoroti. Žmonės kiekvieną sekundę nepradeda galvoti nuo nulio, o remiasi anksčiau įgytomis žiniomis. Tradiciniai neuroniniai tinklai to negali padaryti ir tai yra didelis trūkumas, tuo tarpu pasikartojantys neuroniniai tinklai išsprendžią šią problemą. Šiuose tinkluose egzistuoja tam tikra struktūra, “kilpos” (angl. loop), kurios perduoda informaciją iš vieno žingsnio į kitą ir taip leidžia praėjusiai informacijai išlikti tinkle. Pagrindinė pasikartojančio (rekurentinio) dirbtinio neuronų tinklo topologija yra visiškai rekurentinis dirbtinis tinklas, kuriame kiekvienas dirbtinis neuronas yra tiesiogiai sujungtas su kitu neuronų visomis kryptimis. Praktikoje GRNT nėra plačiai taikomi dėl savo sudėtingumo.



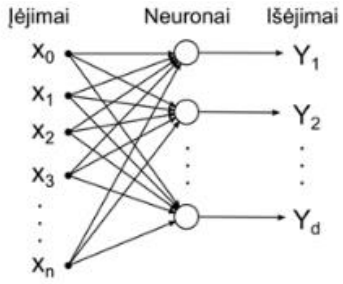
5 paveikslas. Grįžtamojo ryšio arba rekurentinis neuroninis tinklas

Tiek tiesioginio sklidimo neuroniniai tinklai, tiek atbulinio skleidimo neuroniniai tinklai dar plačiau skirstomi į algoritmus (6 pav.). Tiesioginio sklidimo tinklams priskiriami vienasluoksniai ir daugiasluoksniai perceptronai bei radialinių bazių funkcijų tinklas (angl. Radial basis functions network (RBF)). GRNT priskiriami savarankiškai organizuojamo žemėlapio (angl. Self-organizing map, toliau – SOM) neuroniniai tinklai, Hopfieldo (angl. Hop-field) tinklai, Ilgos trumpalaikės atminties (angl. Long Short Term Memory, toliau - LSTM) neuroniniai tinklai ir kt. Siekiant apžvelgti kuo šie algoritmai skiriasi viens nuo kito, toliau bus plačiau analizuojami kiekvieni iš jų. Iš pradžių bus tyrinėjami tiesioginio skleidimo tinklų algoritmai, po jų – rekurentinių tinklų algoritmai.



6 paveikslas. Neuroninių tinklų algoritmai (Jain, 1996)

Pirmuoju dirbtiniu neuroniniu tinklu laikomas vienasluoksnis neuroninis tinklas, kurį po gausybės modeliavimo bandymų 1958 m., tyrinėjant vabzdžių akis, pavyko sukurti neurobiologui F. Rosenblatt, kuris vėliau šį tinklą pavadino perceptronu. Perceptrono struktūra yra pati paprasčiausia ir primityviausia iš DNT. Joje galimos tik vienos krypties jungtys, iš vieno įėjimo, į vieną išėjimą ir yra sudaryta iš vieno sluoksnio d neuronų su n išėjimų. Įvesties sluoksnis (angl. input layer) projektuojamas tiesiai į išvesties sluoksnį, bet ne atvirkščiai (7 paveikslas).



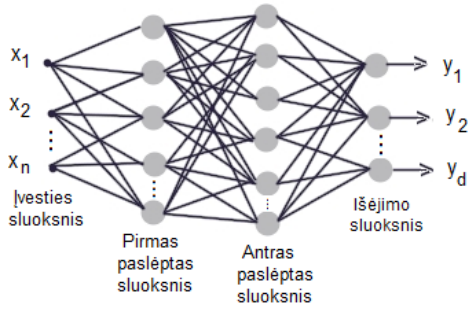
7 paveikslas. Tiesioginio neurono sklidimo tinklas

Pavaizduotas vienasluoksnis neuroninis tinklas, turi po keturis neuronus iėjimų ir keturis neuronus išėjimų sluoksnyje. Toks neuroninis tinklas vadinamas vienasluoksniu, nes išvesties (išėjimų) sluoksnis nėra įtraukiamas, kadangi jame nėra atliekami jokie skaičiavimai. Visgi, vienasluoksniai neuroniniai tinklai, dėl savo paprastumo, leidžia spręsti tik paprastus klasifikavimo uždavinius ir praktikoje jo pritaikymas yra labai ribotas.

Galimybę spręsti daug sudėtingesnius uždavinius, su žymiai sudėtingesne struktūrą, suteikė tokio tinklo pradininkai Rumelhart ir McClelland (1986 m.), kai sukūrė daugiasluoksnį neuroninio tinklo (angl. Multilayer Neural Network) modelį. Daugiasluoksnis neuroninis tinklas gaunamas, jungiant vieną ar daugiau paslėptų sluoksnių (8 pav.). Paslėptame sluoksnyje

esantys neuronai yra tarpininkai tarp įvesties ir išvesies sluoksnių neuronų. Taigi, daugiasluoksnį neuroninį tinklą sudaro trys besijungiantys sluoksniai, kurie atlieka tam tikras funkcijas (Zakaria ir kt., 2014):

- Įvesties sluoksnyje neuronai priima informaciją iš išorinių šaltinių ir sinapsėmis siunčia ją apdoroti tinklui. Tai gali būti arba kito sluoksnio išėjimai, arba tinklo išorėje esančių sistemų siunčiami signalai.
- Paslėptajame sluoksnyje neuronai priima informaciją iš įvesties sluoksnio arba kito sluoksnio išėjimų ir apdoroja ją. Šis sluoksnis jungiasi tik su kitais neuronio tinklo sluoksniais.
- Išėjimo sluoksnyje neuronai gauna apdorotą informaciją ir siunčia ją iš neuroninio tinklo.



8 pav. Dviejų paslėptų sluoksnių neuroninis tinklas

Įprastai daugiasluoksnis DNT modeliuojamas su vienu ar keliais paslėptais sluoksniais. Vieną paslėptą sluoksnį turintis neuroninis tinklas gali spręsti bet kokio sudėtingumo uždavinius ir gali sudaryti visas funkcines priklausomybes, vis dėl to, labiau naudojamas yra du paslėptus sluoksnius turintis neuroninis tinklas. Daugiau nei du paslėptus sluoksnius apmokyti DNT gali užtrukti labai ilgai, o rezultatų skirtumas gali būti ne visada toks akivaizdus, koks planuojamas. Reikalingas sluoksnių skaičius ir neuronų skaičius nėra nusakomas vienareikšmiškai. Neuroninis tinklas bendru atveju gali būti kokio tik norimo dydžio. Didinant sluoksnių skaičių juose, didėja tinklo sudėtingumas. Jau dviejų sluoksnių neuroninis tinklas gali aproksimuoti bet kokią tolydinę funkciją. Tačiau dažnai didinant sluoksnių skaičių tinklo apimtys mažėja, t. y. neuronų skaičius mažėja.

Radialinių bazių funkcijų (toliau – RBF) neuroniniai tinklai iš tikrųjų yra tiesinio sklidimo neuroniniai tinklai, kurie kaip aktyvavimo funkciją naudoja ne logistinę, o radialinės bazės funkciją. Skirtingai negu logistinė funkcija, radialinės bazės funkcijos atsako į klausimą, kiek toli jis yra nuo tikslo. Tai reiškia, kad radialinių bazių funkcijos atsižvelgia į taško atstumą nuo centro. RBF funkcijos yra dviejų sluoksnių – iš pradžių funkcijos sujungiamos radialiniės bazės funkcija vidiniame sluoksnyje, o vėliau, atsižvelgiant į šių funkcijų išvestis, apskaičiuojama ta pati išvestis kitame laiko žingsnyje, kas iš esmės yra atmintis. Šie tinklai naudojami funkcijos

tikslumui nustatyti, laiko eilutėms numanyti, klasifikuoti bei sistemai valdyti. Radialinių bazių funkcijų neuroninio tinklo struktūra yra labai panaši kaip daugiasluoksnio tinklo ir taip pat turi tris besijungiančius sluoksnius – įvesties, paslėptąjį ir išvesties su netiesine RBF aktyvavimo funkciją.

Savarankiškai organizuojamo žemėlapio (SOM) neuroninis tinklas yra dirbtinio neuroninio tinklo tipas, kuris treniruojamas naudojant mokymą be mokytojo, kad būtų sukurtas žemo matmens (paprastai dvimatis) diskretiškas vaizdavimas. Tokia ištreniruotų pavyzdžių įvesčių erdvė vadinama žemėlapiu (angl. maps). Savarankiškai organizuojami žemėlapiai skiriasi nuo kitų dirbtinių neuroninių tinklų, kadangi jie naudoja kaimynines funkcijas, kad išsaugotų įvesties vietos topologines savybes. SOFM idėjos kartu su elastingu tinklo algoritmu naudojamos išspręsti Euklido problemas, tokias kaip keliaujančio pirklio problema. Keliaujančio pirklio uždavinys, arba problema yra ta, kad pirklys žino kokius miestus jis turi aplankyti, tačiau jam reikia rasti trumpiausią maršrutą, aplankant kiekvieną miestą po vieną kartą ir grįžtant į pradinį tašką

Hopfieldo tinklai naudojami optimizavimo problemoms spręsti. Šiuose tinkluose nėra sluoksnių struktūros, o svoriai yra konstantos ir simetriški. Vis dėlto, Hopfieldo tinklai yra visiškai sujungti neuronų sistema. Hopfieldo dirbtinis neuroninis tinklas yra naudojamas saugoti vieną ar daugiau stabilių taikinių vektorių. Į šiuos vektorius galima žiūrėti kaip į prisiminimus, kuriuos tinklas atsimena, kai jie yra aprūpinti panašiais vektoriais. Taigi, tinklo svoriai kaupia informaciją apie tinklo atsiminimus ir stabilias būsenas. Kiekviena ląstelė tarnauja kaip įvestis prieš mokymą, kaip paslėpta ląstelė per apmokymą ir kaip išvestis, kai yra naudojami. Neuronai atnaujinami pagal diferencialinę lygtį, o dėl stabilių būsenų ir svorių simetriškumo, energijos funkcija laikui bėgant yra optimizuojama (sumažėja). Jei bus naudojami nesimetriniai svoriai, tinklas gali elgtis periodiškai arba chaotiškai (A. Krenker ir kt. 2011).

Ilgos trumpalaikės atminties neuroniniai tinklai (angl. Long Short Term Memory, LSTM) – tai rekurentiniai neuroniniai tinklai galintys išmokti ilgalaikes priklausomybes. Pagrindinis LSTM privalumas lyginant su kitais rekurentiniais neuroniniais tinklais, kad jie gali selektyviai „prisiminti“ arba „pamiršti“ informaciją. LSTM neuroninis tinklas yra sudarytas iš įvairių LSTM blokų, galinčių įsiminti vertę bet kuriuo momentu. Taigi informacijos įsimenamumas ilgą laikotarpį yra ne tai, ko tinklas stengiasi išmokti, o tai tiesiog pagal nutylėjimą yra šių tinklų savybė. Pasak, K. Greff ir kt. (2017), dažniausiai pasitaikanti bloko sudėtis būna sudaryta iš keturių dalių: įvesties vartų (angl. input gate), išvesties vartų (angl. output gate) ir užmaršties vartų (angl. forget gate) bei atminties ląstelės (angl. memory cell). Atminties ląstelė yra svarbiausia LSTM architektūros dalis. Ji atsakinga už informacijos „atsiminimą“ neapibrėžtą

laiką ir yra valdoma per visus 3 vartus. Įvesties dalis kontroliuoja naujų reikšmių patekimą į ląstelę (į ląstelę turi patekti tik svarbi informacija). Analogiškai, išvesties dalis kontroliuoja reikšmių išvestis iš atminties ląstelės. Užmaršties vartų paskirtis - kontroliuoti kiek informacijos bus „pamiršta“ t.y. informacija, kuri nėra reikalinga arba nereikšminga LSTM modeliui suprasti sąryšius, yra pašalinama. Per šiuos kanalus LSTM tinklai geba klasifikuoti, apdoroti ir numatyti laiko eilučių duomenis su labai ilgais nežinomo dydžio laiko tarpais, bei tokiu būdu yra optimizuojamas LSTM modelio veikimas. Šiuo metu LSTM neuroniniai tinklai laikomi vieni sėkmingiausiai praktikoje pritaikomų dirbtinių neuroninių tinklų. Savo technologijose LSTM naudoja tokios didžiulės koorporacijos kaip „Apple“, „Amazon“ ar „Google“.

Apibendrinant galime teigti, kad neuroninių tinklų spektras nuo pirmojo dirbtinio neuroninio tinklo atsiradimo sparčiai išsiplėtė. Priklausomai kaip neuronai tinkluose yra sujungiami (kokia jų neuroninių tinklų architektūra) ir kokia kryptimi jie gali siųsti signalus jie skirstomi į pagrindines dvi grupes: tiesioginio sklidimo ir rekurentinius neuroninius tinklus. Šie tinklai yra dar plačiau skirstomi į įvairius algoritmus, kurie yra naudojami ir pritaikomi priklausomai nuo sprendžiamo uždavinio tikslo bei jo sudėtingumo.

1.2.3. Neuroninių tinklų mokymas

Pasirinkdami ir sudarydami dirbtinio neuroninio tinklo topologiją, yra atliekama tik pusę darbo, kad būtų galima šį neuroninį tinklą pritaikyti nurodytai problemai spręsti. Kaip ir biologiniai neuronai tinklai turi išmokti tinkamai reaguoti į duotas aplinkos įvestis, taip ir dirbtiniai neuroniniai tinklai turi daryti tą patį. Taigi, kitas būtinas žingsnis dirbtiniui neuroniniui tinklui yra išmokti suteikti tinkamą atsaką duotoms įvestims ir tą jis gali išmokti atliekant tam tikrą neuroninių tinklų mokymą. Apmokymo metu neuroninis tinklas išmoksta įvesties ir išvesties duomenų priklausomybę. Priklausomybė tarp įvesties ir išvesties duomenų nėra tiesinė ir ji nėra žinoma, tai reiškia, kad pats algoritmas nusako priklausomybes funkcinę formą. Vadinasi, neuroninis tinklas gali savarankiškai suformuoti bet kokį modelį iš pavyzdžių. Dirbtinio neuronų tinklo mokymas – tai jungčių svorių keitimo uždavinys, siekiant, kad tinklas galėtų atlikti jam skirtą užduotį. Skirtingos tinklų architektūros reikalauja skirtingų mokymo algoritmų. Skiriamos trys neuroninių tinklų mokymosi paradigmos, kurios dažniausiai gali naudoti bet kurios rūšies dirbtinio neuroninio tinklo archtektūrą ir kurios pačios turi daugybę mokymo algoritmų:

1. Mokymas su mokytoju, arba prižiūrimas mokymas *(angl. suprivised learning)*. Tai yra plačiausiai paplitęs neuroninio tinklo apmokymo būdas. Prižiūrimam mokymui reikalingas išorinis mokytojas, valdantis mokymosi procesą ir teikiantis informaciją. Tai gali būti duomenų rinkinys, iš kurio galima mokytis arba stebėtojas, vertinantis neuroninio tinklo

našumą. Mokymo su mokytoju algoritmuose vartojama sąvoka „norimos išėjimo reikšmės“. Tai reiškia, kad žinome, pavyzdžiui, kuriai klasei priklauso objektas, arba yra žinoma prognozuojama reikšmė. Prižiūrimo mokymo tikslas – priversti neuroninį tinklą pakeisti neuroninių jungčių svorius pagal pavyzdines įvestis ir išvestis (Lison, 2012). Tinklas koreguojamas keičiant svorių reikšmes. Ieškoma tokių svorių, kad skirtumas tarp norimų išėjimo reikšmių ir reikšmių, gautų išmokius tinklą, būtų kuo mažesnis. Mokymas baigiamas tinklui išmokus (su galima minimali paklaida) sieti įvestis su išvestimis. Atsiradus naujiems duomenims, numatyti galimą išvestį bandoma remiantis ankstesniais mokymais ir apytiksliu žemėlapiu (angl. mapping) tarp įvesčių ir išvesčių. Svarbus veiksnys – mokymo duomenų aibė, kuri turi būti suprantama ir privalo aprėpti visas praktines tinklo taikymo sritis. Taigi tinklas veiks gerai tik parinkus tinkamą mokymo aibę.

Šiuo metodu neuroninis tinklas mokomas naudojant imtį, kurią sudaro įvesties ir išvesties reikšmės. Tinklas nežino priklausomybės tarp jų. Tinklas yra gerai apmokytas tik tada, jeigu jis modeliuoja priklausomybės funkciją, siejančią įvesties ir išvesties kintamuosius. Pagal šią nustatytą funkciją vėliau tinklas pats gali prognozuoti naujas išvestis. Tai reiškia, kad apmokymas su mokytoju dažniausiai sprendžia klasifikavimo problemą, kadangi yra turimas duomenų rinkinys ir žinomas rezultatas, kurį duoda kiekvienas šio rinkinio elementas, tad apmokymo tikslas yra apmokyti neuroninį tinklą taip, kad jis galėtų klasifikuoti naujus ir dar nematytus duomenis. Dažniausiai gavus rezultatą būna aptinkama neuroninio tinklo klaida, kuri apskaičiuojama iš duotos įvesties rezultato atimant gautą rezultatą. Neuroninį tinklą taip pat galima išmokyti atpažinti veidus, naudojant tam tikrų žmonių veidų pavyzdžius ir apmokant tinklą juos atpažinti. Pasak A. Krenker ir kt. (2011) mokymas su mokytojų dar gali būti skirstomas pagal klasifikacijas. Nors šiam modeliui turime didelį spektrą klasifikacijų, kurios turi savų privalumų ir trūkumų. Vis dėlto, pasirinkti tam tikrą klasifikaciją tam tikrai problemai spręsti dar laikoma atsitiktinumu ar sėkme, o ne mokslu.

2. Mokymas be mokytojo *(angl. reinforcement (unsupervised) learning),* arba neprižiūrimas mokymas (angl. unsupervised learning). Neprižiūrimas mokymas vyksta be išorinio įsiterpimo. Čia nėra „norimų išėjimo reikšmių“ – nėra žinomos tinklo išvesčių reikšmės ir nėra mokytojo, kuris ištaisytų klaidas. Remdamasi vidiniais kriterijais ir tinklo informacija, sistema turi pati save suderinti bei suprasti duomenų ypatybes. Tokiems neuroniniams tinklams pateikiami tik įvesčių pavyzdžiai, o sistema pati pagal požymius turi suklasifikuoti įvestis (Lison, 2012). Tinklas mokomas ieškoti panašumų (pvz.: saviorganizuojantis neuroninis tinklas (SOM), Kohoneno tinklai). Viena iš bendrų neprižiūrimo mokymosi formų yra grupavimas, kai pagal duomenų panašumą yra bandoma suskirstyti į skirtingas grupes. Pirmiausia duomenys yra suskirstomi į grupes, pavyzdžiui, sugrupuojant klientus pagal jų pirkimo elgseną. Tai

vadinamas grupavimas pagal klasterį (angl. clustering). Vėliau yra grupuojama pagal asociacijas (angl. association), tai reiškia, kad algoritmai bando suprasti taisykles, kurios paaiškina didelės apimties duomenų elgseną, pavyzdžiui, kad pirkėjai, kurie perka marškinėlius, taip pat yra linkę nusipirkti ir kelnias.

3. Sustiprintas mokymas *(angl. reinforcement learnin*g). Čia dalis tinklo svorių nustatomi pagal mokymą su mokytoju, kita dalis gaunama iš mokymo be mokytojo. Sustiprintas mokymasis yra mašininio mokymosi technika, kai duomenys paprastai nėra duoti, bet generuojami sąveikaujant su aplinka (Lison, 2012). Sustiprintas mokymasis iš esmės veikia taip pat, kaip vaikas ankstyvajame savo gyvenimo etape. Kai vaikas padaro gerą darbą, ar elgiasi gerai, jis yra skatinamas kartoti tą veiksmą, o kai elgiasi netinkamai ir daro blogą darbą, už tai yra baudžiamas ir atgrąsomas nuo tokio pat veiksmo kartojimo. Taip pat ir dirbtinis neuroninis tinklas sąveikaudamas su aplinka, gauna atlygį už teisingą užduočių atlikimą ir baudą už netinkamą darbą. Šis mokymas yra pritaikomas aptikti kaip dirbtinis neuroninis tinklas imasi veiksmų aplinkoje, kad būtų maksimizuota kažkieno ilgalaikė grąža (didinamas atlygis ir mažinamos bausmės). Apibrėžus maksimalios grąžos funkciją (angl. return function), sustiprintas mokymas naudoja kelis algoritmus, kad būtų galima rasti strategiją (angl. policy), kuri duoda geriausią rezultatą. Naivus brutalios jėgos algoritmas (angl. naive brute force algorithm) pirmiausia kiekvienai strategijai apskaičiuoja grąžinimo funkciją, ir parenka tokią politiką, kurios grąža būtų didžiausia. Akivaizdus šio algoritmo trūkumas yra ypač didelis ar net begalinis galimų strategijų skaičius. Šis trūkumas gali būti pašalintas taikant vertės funkcijos (angl. value function) metodus arba tiesioginį politikos vertinimą (angl. direct policy estimation). Stiprinamasis apmokymas ypač gerai tinka problemoms, kurios yra koncentruotos į ilgalaikius, o ne į trumpalaikius tikslus (Krenker ir kt, 2011). Jis sėkmingai pritaikytas įvairioms problemoms spręsti, įskaitant robotų valdymą, telekomunikacijas ir žaidimus, tokius kaip šachmatai ir kitos nuoseklios sprendimų priėmimo užduotys.

Taigi, apibendrinant galime teigti, kad dirbtinis neuroninis tinklas yra galingas progromavimo metodas, įkvėptas žmogaus smegenų mokymosi savybėmis. Jį sudaro neuronai, kurie atlieka įkėlimo, mokymosi ir įsiminimo funkcijas identištas kaip žmogaus smegenų ląstelės. Šios savybės leidžia modeliui veiksmingai analizuoti ir spręsti problemas įvairiose srityse ir plačiai naudojamas paslaugų, produkcijos sektoriuose bei įvairiose programėlėse. Nors šiuolaikinė programinė įranga leidžia nesunkiai kurti, pritaikyti ir valdyti dirbtinius neuroninius tinklus, optimizuoti ir naudoti juos realiose situacijose, būtina suprasti už to slypinčią teoriją, kaip šie tinklai veikia ne tik praktiškai, bet ir teoriškai. Dėl to šitas skyrius buvo skirtas sukaupti bendrinę informaciją apie dirbtinius neuroninius tinklus, bei duoti pradžią šios temos gilesnio pažinimo link. Pirmiausia šio skyriaus dalyje buvo aprašyta pagrindinė

neuroninio tinklo koncepcija – jo sudedamosios dalys bei pats neurono virsmas tinklu. Taip pat buvo pateikta pagrindinė informacija apie skirtingus, dažniausiai naudojamus dirbtinių neuroninių tinklų tipus bei išanalizuota susisteminta medžiaga apie jų klasifikavimą. Galiausiai buvo aprašytos skirtingos mokymosi paradigmos bei jų derinimas prie pasirinktų architektūrų.

2. METODOLOGIJA

Pasaulyje didėjant informacijos kiekiui, kartu didėja ir poreikis pritaikyti kuo efektyvesnius ir tikslesnius metodus, kurių pagalba būtų galima sukurti modelį, kuris leistų numatyti staigius valiutų kursų pasikeitimų ateityje.

Pirmame šios dalies poskyryje bus išnagrinėti autorių moksliniai darbai, kuriuose pritaikydami skirtingus dirbtinio intelekto prognozavimo metodus jie stengėsi gauti tiksliausius valiutų kursų rezultatus. Antroje dalyje, remiantis šių tyrimų analizę bus deliojamas modelis, kurio pagalba bus atliktas trijų valiutų kursų porų prognozavimas tiriamąjame laikotarpyje bei statistiškai įvertinti gauti rezultatai.

Svarbu paminėti, kad šiame tyrime bus atliekama tik techninė analizė, neįvertinant fundamentaliųjų reiškinių. Taip pasirinkta dėl to, kad buvo norimą ištirti, kiek tiksliai ir patikimai galima suprognozuoti pasirinktus valiutų kursus, remiantis tik techninėmis priemonėmis. Be to, pasak M. H. Eng bei kitų autorių (2008) atlikto tyrimo, kuriame ekonominiai rodikliai buvo pridėti prie dirbtinio neuroninio tinklo, gauti rezultatai parodė, jog jų įvertinimas nepagerino prognozuotų valiutų kursų našumo. Viena iš priežasčių autoriai išskyrė tai, jog dažniausiai ekonominiai rodikliai yra atnaujinami kas ketvirtį, todėl, jei šie rodikliai būtų atnaujinami dažniau, galbūt jie turėtų didesnį indėlį į prognozuojamą rezultatą. Taip pat, remiantis anksčiau išnagrinėtais darbais, daugybė prognozavimo strategijų yra vykdoma būtent atliekant techninę analizę.

Tyrimo tikslas – atlikus dirbtinių neuroninių tinklų metodų analizę, bei išnagrinėjus autorių atliktus tyrimus, sudaryti bei pritaikyti dirbtinio neuronio tinklo modelį, kuris gebėtų suprognozuoti pasirinktų valiutų porų kursus.

Tyrimo uždaviniai:

1. Išnagrinėti autorių darbus, kurie analizavo valiutų kursus bei pritaikė skirtingus dirbtinius neuroninio tinklo metodus;
2. Remiantis išanalizuota valiutų rinkos ir neuroninių tinklų tema bei autorių atliktų mokslinių darbų literatūra, sudaryti modelį, kuris padėtų suprognozuoti valiutų kursus;
3. Pritaikant sudarytą modelį, suprognozuoti pasirinktų valiutų porų kursus bei patikrinti gautų rezultatų reikšmingumą ir modelio tinkamumą;
4. Sutikrinus modelio tinkamumą, suprognozuoti pasirinktų valiutų kursų uždarymo kainas laikotarpiais į priekį.
5. Apibendrinti gautus rezultatus bei pasidalinti pasiūlymais tolimesniems darbams šia tema.

2.1. Dirbtinių neuroninių tinklų prognozavimo metodų analizė

Valiutų kursų prognozavimas šiuolaikiniais ekonomikos laikais nėra nauja tyrimų kryptis, siekiant investuotojui rasti geriausią rinkos strategiją, naujus planavimo būdus į užsienio projektus ar tiesiog geidžiant didesnio pelno. Valiutų kursų prognozavimui pasiūlyta ir taikoma daugybę metodų. Pasak L.Liu ir W. Wang (2008), šiuos metodus galima suskirstyti į dvi kategorijas: tiesinio (angl. linear prediction method) ir netiesinio prognozavimo metodus. Pirmoji kategorija apima tiesinės regresijos modelį, autoregresinį integruoto slenkančio vidurkio (angl. Auto Regressive Integrated Moving Averages - ARIMA) modelį, apibendrintą autoregresinį sąlyginį heteroskedastiškumą (angl. Generalized linear auto-regression - GARCH), paslėptą Markovo modelį (angl. The Hidden Markov Model – HMM), vektorinį autoregresijos (VAR) modelį ir kitus. Nors šiuos tiesinius metodus lengva įgyvendinti ir jie yra gana lengvai suprantami, jie nesugeba fiksuoti netiesinių duomenų priklausomybės (O. Erdogan, A. Goksu, 2014). Analizuoti netiesinių ryšių laiko eilučių duomenus gali netiesinės prognozavimo metodų kategorijai priskirti netiesinio chaosinio prognozavimo (angl. nonlinear chaos prediction) modeliai, dirbtinio neuroninio tinklo (angl. Artificial Neural network) modeliai, palaikymo vektorių mašinos (SVM) ir kiti. Netiesinio prognozavimo metodai turi pranašumą prieš klasikinius modelius dėl to, kad gali spręsti bet kokias netiesines funkcijas, neturint jokios informacijos apie duomenis. Taip pat, jie sugeba mokytis iš duomenų. Pritaikyti šiuos netiesinės prognozavimo metodus nėra lengva, norint gerai suprognozuoti finansinę laiko eilutę, reikalingos ne tik stiprios žinios, bet ir geri programavimo įgūdžiai. Vis dėlto pritaikyti netiesinio prognozavimo būdai leidžia tikėtis geresnių rezultatų ir tikslesnių prognozių, tačiau pats prognozavimo procesas nėra idealus. Pagrindinė to priežastis galimai susiję su valiutų rinkos nepastovumu ir netikrumu (Ellen ir kt. 2013). Prognozavimo tikslumui taip pat turi įtakos tai, jog kiekvienas prognozavimo metodas turi savų trūkumų, todėl svarbu išnagrinėjus skirtingus prognozavimo metodus rasti tokį modelį ar modelių grupę, kurie galėtų duoti tiksliausius prognozavimo rezultatus.

Valiutų kursų prognozavimo tema yra plačiai nagrinėjama užsienio autorių tyrimuose. Visuose nagrinėtuose tyrimuose autoriai, pasirinkę tam tikras valiutų poras bei tyrimo laikotarpį, pritaiko skirtingus prognozavimo metodus bei siekia rasti geriausią modelį, kuris fiksuotų tiksliausius suprognozuotus valiutų kursus. Žemiau, 3 lentelėje, pateikta susisteminta informacija apie naujausius išnagrinėtus darbus, kuriuose autoriai tyrė skirtingų valiutų porų kursus.

3 lentelė: Susisteminta informaciją apie autorių prognozavimo tyrimus, panaudojant dirbtinį intelektą.

Grupės	Algoritmai	Tyrėjai	Gauti rezultatai- išvados
Feedforward neural network	Multi-layer perceptron	Galeshchuk S. (2015)	Autorių darbe suprognozuoti valiutų kursus buvo pritaikytas daugiasluoksnis neuroninio tinklo algortimas. Rezultatai parodė, kad tiksliausi suprognozuoti valiutų kursai buvo gauti su dieniniais duomenimis.
	Atvirkštinis skleidimas (angl. Backpropagation algorithm)	Nanayakkara S., Chandrasekara V., Jayasundara D. D. M. (2014)	Tyrime buvo lyginamas dviejų modelių tikslumas: laiko eilučių modelis (GARCH) ir dirbtinio neuroninio tinklo (Backpropogation algortimas) modelis. Rezultatai parodė, kad ANN modelis nuspėti nagrinėjamos valiutos kursą yra tinkamesnis.
	Adaptive Neuro-Fuzzy Inference System (ANFIS)	Mondal S., Dakshinakabat P. Swain K. S. (2015)	Tyrime buvo naudota keletą laiko eilučių prognozavimo metodų bei adaptyvi neuro-fuzzy išvadų sistema, kuri yra tam tikras dirbtinis nervų tinklas, paremtas Takagi - Sugeno fuzzy išvadų sistema. Gauti rezultatai parodė, kad tikslesnį prognozavimą suteikia būtent tiesioginio sklidimo metodas.
Recurrent neural network	Least squares support vector machine (LS-SVM)	Liu L., Wang W. (2008)	Realieji ir prognozuojami valiutų kursai yra labai panašus, kas patvirtina, kad LS-SVM metodas gali būti pritaikytas valiutų kursų prognozavimui.
	Elman-Jordan recurrent network	Kondratenko V.V., Kuprin Y. (2003); Oancea B., Ciuciu C. S. (2014)	Abu tyrimai pagrindė, kad šis neuroninio tinklo algoritmas yra tinkamas prognozuoti valiutų kursus ir duoda geresnius prognozavimo rezultatus negus laiko eilučių metodai.
	Singular spectrum analysis (SSA); Multivariate SSA (MSSA)	Mahmoudvand R., Rodrigues C. P., Yarmohammadi M. (2018)	MSSA metodo rezultatai buvo pranašesni, lyginant su SSA.
Hybrid neural network	VAR-VECM-ANN model	Parot A., Michell K., Kristjanpoller D. W. (2019)	Gauti rezultatai rodo, kad siūloma skirtingų modelių integraciją savo rezultatais pralenkia etaloninius modelius ir padidina prognozavimo tikslumą.

Šaltinis: sudaryta autoriaus, remiantis nagrinėtais tyrėjais.

Daugybė tyrimų, nagrinėjusių dirbtinius neuronius tinklus ir jų galimybę prognozuoti valiutų kursus buvo atlikta 2015 m. Tais pačiais metais Svitlana Galeshchuk savo darbe „Neural networks performance in exchange rate prediction“ tyrė tris valiutų poras (EUR/USD, GBP/US ir USD/JPY) bei nagrinėjo tiesioginio sklidimo neuroninių tinklų pritaikymą, prognozuojant šių valiutų porų kursus dienos, mėnesio ir ketvirčio žingsniais. Tyrime iš viso buvo naudotos 83 dieninės reikšmės (nuo 2014 m. sausio 1 dienos iki 2014 m. balandžio 25 d.), 60 mėnesinių reikšmių (nuo 2009 m. gegužės mėn. iki 2014 m. gegužės mėn.) ir 59 pusmetinės reikšmės (nuo 1999 m. gegužės mėn. iki 2014 m. gegužės mėn.). Panaudojant daugiasluoksnį neuroninio tinklo algortimą, prognozavimo rezultatų analizė parodė, kad vidutinės ir didžiausios santykinės prognozės paklaidos trumpalaikėje prognozėje su dieniniais duomenimis yra 0,2-0,4% EUR/USD kursui, 0,2-0,9% GBP/USD kursui ir 0,3-1,3% USD/JPY kursui, atitinkamai su mėnesiniais duomenimis paklaidos buvo: 1,3-3,3% EUR/USD kursui, 2,2-4,5% GBP/USD kursui ir 0,3-1,3% USD/JPY kursui. Prognozavimą atlikus su pusmetiniais žingsniais, gautos vidutinės ir didžiausios paklaidos buvo dar didesnės: 2,3–5,1% EUR/USD kursui, 1,9–5,0% GBP/USD kursui bei 3,5–10,2% USD/JPY kursui. Taigi, šie rezultatai rodo, kad prognozavimo metodas yra tinkamas ir gali būti naudojamas praktinėse sistemose, o norint gauti kuo tikslesnę

prognozę vertėtų naudoti dieninius duomenis. 2015 metais prognozavimo metodų palyginimą savo tyrime atliko ir S. Mondal su kolegomis (2015), kuriame laiko eilučių metodais ir neuroninio tinklo metodu tyrė Indijos rupijos ir JAV dolerio valiutų porą. Laiko eilučių metodu buvo atliekamas matematinis Indijos rupijos (INR) ir JAV dolerio (USD) prognozės apskaičiavimas panaudojant 12 metų vidutinį metinį duomenų rinkinį ir apskaičiuojant prognozavimo paklaidas MSE ir RMSE rodikliais. Prognozuojant valiutų kursus neuroninio tinklo adaptyvi neuro-fuzzy išvadų sistema taip pat buvo skaičiuojami MSE ir RMSE parametrai. Neuroniniu tinklu gauti rezultatai buvo ženkliai geresni negu skaičiuojant laiko eilutės metodais. Taip pat tyrime buvo padaryta išvada, kad didesnis neuronų skaičius sumažina MSE ir RMSE rodiklių reikšmes ir teikia tikslingesnę prognozę.

S. Nanayakkara, V. Chandrasekara ir D. D. M. Jayasundara 2014 metais pritaikė kitą tiesioginio skleidimo neuroninio tinklo algortimą – „Backpropagation“. Ištirti ir suprognozuoti JAV dolerio ir Šri Lankos rupijos dienos valiutos kursą, atlikti palyginamąją analizę buvo panaudotas ir autoregresinio sąlyginio heteroskedaziškumo (GARCH) laiko eilučių modelis. Abiejuose modeliuose kaip aiškinamieji kintamieji buvo naudojami praeities duomenų eilučių stebėjimai ir slenkamieji vidurkiai, o numatomasis efektyvumas buvo vertinamas naudojant daugelį plačiai taikomus statistinius rodiklius. Rezultatai parodė, kad pritaikant laiko eilučių GARCH modelį, buvo gauta, kad modelis būsimus valiutų kurso rodiklius numatė 69% tikslumu, tuo tarpu geriausias grįžtamojo neuroninio tinkle modelio su atgalinio dauginimo (angl. Backpropogation) algortimu valiutų kursų numatymo tikslumas buvo 82%. Remiantis dviejų modelių rezultatais, galima daryti išvadą, kad tyrime ANN pagrįstas modelis yra geresnis, palyginti su GARCH modeliu, kad būtų galima numatyti USD / LKR kursą.

Daugybė autorių nagrinėdami valiutų kursų prognozes naudojo rekurentinių neuroninių tinklų algoritmus. Vienas iš jų buvo Liu ir Wang (2008), kuris savo tyrime pritaikydamas mažiausių kvadratų palaikymo vektorių mašiną (angl. least squares support vector machine - LS-SVM) nustatė, jog tikslumo ir prognozavimo rezultatai buvo labai aukšti, kas rodo, kad LS-SVM modelis yra tinkamas ir pagrįstas metodas prognozuoti valiutos kurso laiko eilutes. Šiame darbe buvo tiriamos keturios valiutų poros – JPY/USAD, CD/USD, USD/GBP, USD/EUR. Per laikotarpį buvo atlikti 1196 stebėjimai iš kurių pirmieji 1006 naudoti kaip treniravimo duomenys, paskutiniai 190 skirti prognozavimui. Visai kitą rekurentinio tinklo algortimą naudojo V. V. Kondratenko ir Y. Kuprin (2003) savo tyrime prognozuojant keturių pagrindinių Forex rinkos valiutų kursus su JAV doleriu: Japonijos jenos, Šveicarijos franko, Didžiosios Britanijos svaro. Autoriai iš anksto apdorojo surinktus duomenis, kad būtų pašalintos visos galimos koreliacijos. Gauti rezultatai parodė, kad neuroninį tinklą galimą sėkmingai įdiegti ir pritaikyti, o tyrimui atlikti buvo naudojamas Elman-Jordan algoritmas, kuris buvo paskirtas

prognozuoti vienos dienos į priekį grąžos vertes. Šis algoritmas anksčiau buvo sėkmingai naudojamas prognozuojant finansines rinkas, nes rekurentinis neuroninis tinklas išmoksta laiko eilučių taisykles, būtinas dirbant su tokiomis. Vis dėlto, šių neuroninių tinklų trūkumas yra ilgas mokymosi laikas. Gauti rezultatai parodė, jog šis tinklo algortimas gali numatyti apytiksliai 80% valiutų kursų pasikeitimą, ko visiškai pakanka praktiniam naudojimui. Valiutų prognozavimo tyrimą, panaudojant grįžtamojo ryšio neuroninių tinklų Elman-Jordan algoritmą atliko ir B. Oancea, S. C. Ciuciu (2014). Tyrimo metu buvo lyginami tiesioginio sklidimo ir rekurentinio neuroninio tinklo metodai bei ištreniruotų algoritmų pagalba buvo supronozuoti EUR/RON ir USD/RON valiutų kursai bei tikrinama, kurio modelio rezultatai yra tinkamesni prognozavimui. Pirmuoju modeliu buvo gauta 0,1% prognozavimo paklaida, o rekurentinio neuroninio tinklo rezultatai parodė 0,01% prognozavimo paklaidą, kas ir parodo, kad šiuo atveju rekurentinis tinklas veikia geriau nei klasikinis. Vienas naujausių rastų tyrimų, kuriame pritaikomi grįžtamojo ryšio neuroninių tinklų prognozavimo metodai buvo atliktas R. Mahmoudvand, P. C. Rodrigues ir M. Yarmohammadi (2018). Jame autoriai tyrinėjo keturių BRICS kylančių ekonomikų (Brazilijos, Indijos, Kinijos ir Pietų Afrikos) dienos valiutų kursus 2001 – 2015 m. laikotarpyje. Rusija teko pašalinti iš tyrimo, kadangi, pasak autorių, nepavyko rasti išsamių to meto duomenų. Norint suprognozuoti šių šalių valiutų kursus buvo naudoti tiek daugiamačiai, tiek vienamačiai laiko eilučių metodai. Iš daugybės laiko eilučių metodų buvo pasirinkta naudoti pavienio spektro analizę (angl. singular spectrum analysis, toliau SSA), kaip vieną galingiausių neparametrinių technikų, ir daugialypę SSA versiją – MSSA (angl. multivariate SSA). Įvertinus metodus vidutinio kvadrato paklaidos rodikliu, visose valiutų porose buvo pastebėtas MSSA modelio pranašumas, išskyrus tiriant Pietų Afrikos ir JAV dolerio valiutų porą, kuri fiksavo geresnius rezultatus su SSA modeliu. Todėl galime daryti išvadą, kad šiuo atveju patikimesnis ir labiau naudingas būdas prognozuoti valiutų kursus yra MSSA.

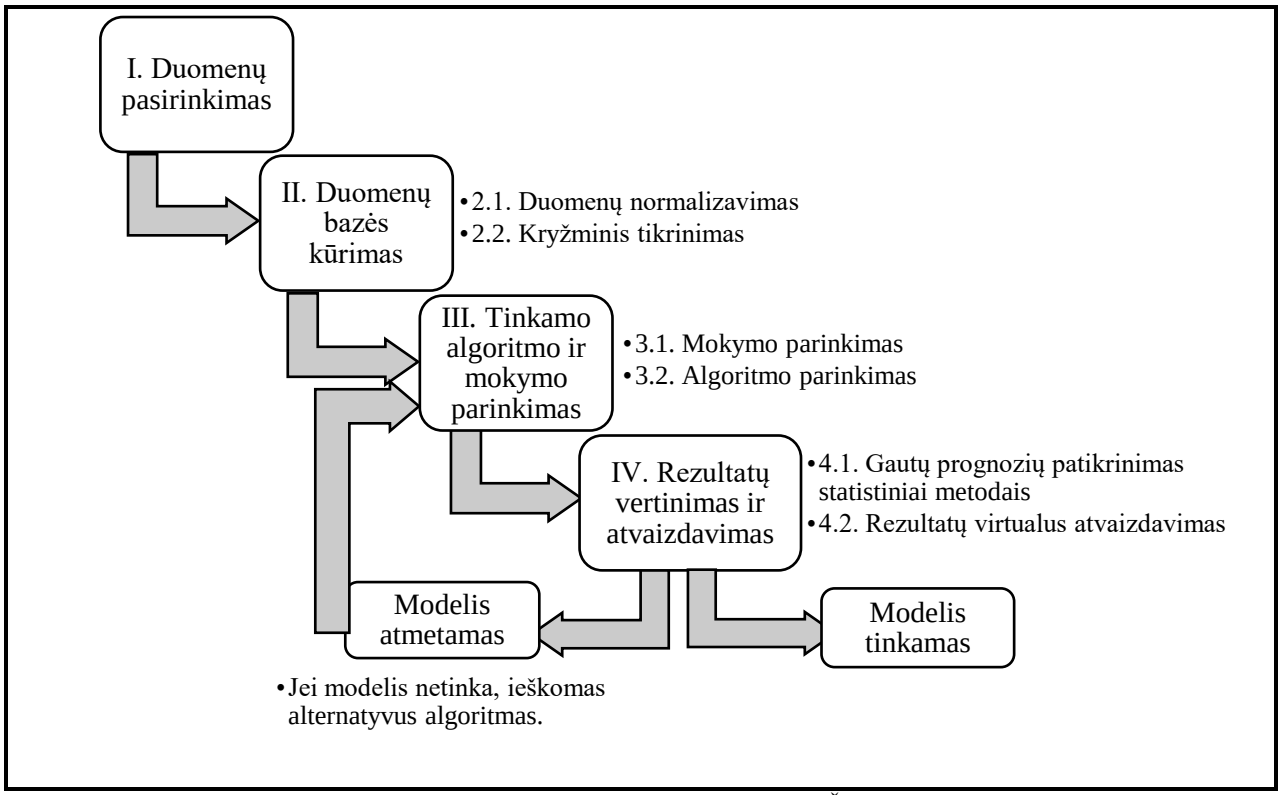
Bėgant laikui, susiformavo nauja grupė, kuri sujungia skirtingas dirbtinio neuroninio tinklo grupes ir ji sukuria hibridinį neuroninį tinklą. Naujausias nagrinėtas tyrimas, susijęs su dirbtinio neuroninio tinklo pritaikymu valiutų kursams atliktas A. Parot, K. Michell ir W.D. Kristjanpoller (2019). Jame autoriai ėmėsi nemenko iššūkio ir pasiūlė naują sistemą, skirtą euro ir Jav dolerio valiutų kurso prognozavimo tikslumui pagerinti, kurioje dirbtinį neuroninį tinklą panaudotų su vektoriniu auto regresiniu (angl. Vector Auto Regressive, toliau – VAR) ir vektorinių klaidų taisymo (angl. Vector Error Corrective, toliau – VECM) modeliais. Pasak autorių, skirtingų modelių integravimas turėtų pagerinti prognozavimo galimybes, lyginant su atskiru modelių pritaikymu. Šią hipotezę argumenduodami tuo, kad dirbtiniai neuroniai tinklai gali fiksuoti trumpalaikius ir ilgalaikius netiesinius laiko eilučių komponentus, ko VECM ir

VAR modeliai padaryti nesugeba. Gauti rezultatai patvirtino išsikeltą hipotezę ir parodė, jog siūloma sistema pralenkia etaloninius modelius, sumažindama prognozavimo paklaidas 32,5%, lyginant su geriausiu ekonometriniu modeliu. Prognozavimo paklaidos buvo skaičiuojamos remdamasi RMSE parametru.

Taigi apibendrinant visus nagrinėtus autorių darbus galime teigti, kad valiutų kursų prognozavimo tema per pastariuosius dešimtmečius yra plačiai nagrinėjama užsienio autorių, tačiau mažai paliesta Lietuvos tyrėjų. Užsienio autorių darbuose yra nagrinėjamos skirtingos valiutų poros bei pritaikomi skirtingi dirbtinių neuroninių tinklų prognozavimo algortimai. Darbuose, kuriuose buvo lyginami klasikiniai metodai ir netiesiniai prognozavimo metodai, tikslesnius rezultatus parodė būtent pastarieji. Iš nagrinėtų darbų galime susidaryti įspūdį, kad dirbtiniu intelektu paremti modeliai yra tinkamesni prognozuoti valiutų kursus. Taip pat, išnagrinėti autorių darbai pagrindė hipotezę, kad geriausia prognozavimo procesą atlikti su dieniniais duomenimis, kadangi jie parodo tiksliausius rezultatus.

2.2. Prognozavimo modelio kūrimas valiutų rinkoje

Šiame skyriuje analizuojama dirbtinio neuroninio tinklo modelio, skirto prognozavimo uždavinių sprendimui ir pritaikyto konkrečiai užduočiai – valiutų kursų prognozavimui, sukūrimo metodika. Šio modelio kūrimo tikslas yra kuo tiksliau numatyti užsienio valiutos kursą FOREX rinkoje, panaudojant dirbtinių neuroninių tinklų algoritmą. Sėkmingai įgyvendinti šį tikslą yra vykdomi keli etapai, pagal kuriuos yra sukūriamas prognozavimo modelis (9 paveikslas).



9 pav. Dirbtinio neuroninio tinklo prognozavimo modelio struktūra (Šaltinis – sukurta autoriaus)

Duomenų rinkimo etape kaip įvesties kintamieji pasirenkami rinkos valiutos kursai. Duomenys renkami vieno laikotarpio, patartina juos traukti iš vieno patikimo šaltinio duomenų bazės.

Norint ištreniruoti dirbtinį neuroninį tinklą ir parengti algortimą mašinų mokymuisi, reikia paruošti ir išvalyti duomenis, kad jie būtų tinkami ir galėtų būti naudojami mokymui. Tam yra konstruojama duomenų bazė. Duomenų bazė plačiau skirstoma į šiuos etapus:

- Duomenų tvarkymas ir valymas (normalizavimas). Atrinkus kintamuosius, reikia patikrinti, kad duomenyse neegzistuotų nulinių verčių. Savaitgaliais (šeštadienį ir sekmadienį) valiutų vertės nėra aiškios, todėl tokie stebėjimai turi būti pašalinti. Atlikus duomenų valymą, paprastai, gera praktika yra normalizuoti duomenis tam tikru diapazonu, kuris dažniausiai svyruoja nuo 0 iki 1, arba nuo -1 iki 1, nes duomenys negali būti labai skirtinguose diapazonuose, o tai gali pakenkti modelio treniravimo metu (Kao ir kt., 2013). Rėžiai priklauso nuo to, kokią aktyvavimo funkcija bus panaudota tyrime. Labiausiai paplitusi yra sigmoidinė funkcija, kurios reikšmes yra nuo 0 iki vieno. Vis dėlto, kadangi žinome, jog valiutų kursai įgija ne tik teigiamas, bet ir neigiamas reikšmes, svarstytina patikrinti ir kitas alternatyvias funkcijas, tokias kaip hiperbolinio tangento funkciją, kurios reikšmių intervalas yra nuo -1 iki 1, ar ReLU aktyvavimo funkciją. Iš esmės, reikšmių intervale esanti didžiausia reikšmė simbolizuoja maksimalią stulpelio vertę, tuo tarpu mažiausia – minimalią, o visos kitos vertės bus tarp šio rėžio. Šis duomenų normalizavimas bus atliekamas pagal šią formulę:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \qquad (3)$$

  Kur:

  z_i – normalizuota įvesties ar išvesties vertė

  x_i – orginali įvestis

  $\max(x)$ – maksimali pradinė įvesties vertė

  $\min(x)$ – minimali pradinė įvesties vertė

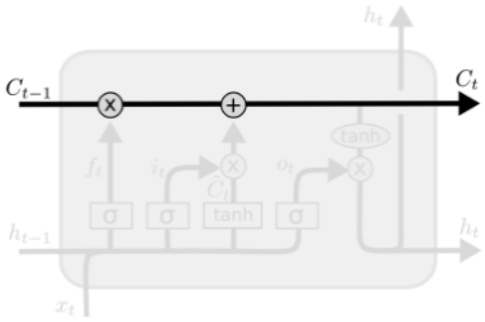
- Atlikus duomenų normalizavimą, duomenys padalijami į dvi dalis – vykdomas taip vadinamas kryžminis tikrinimas (angl. Cross Validation). Pasirenkama procentinė dalis kiek duomenų bus naudojami mokymui, ir kiek testavimui. Mokymo rinkinys naudojamas mokymosi procesui, kuriuo, pagal mokymo pavyzdžius, algoritmas išmoksta signalus bei koreliaciją tarp įvesčių, jų svorių bei išvesčių. Testavimas atliekams po modelio apmokymo, kad patikrinti modelio tikslumą. Paprastai, mokymui skiriama 80% visų duomenų, o testavimui lieka 20%.

Pasirinkę kintamuosius, atlikus aukščiau aptartus veiksmus, gaunamas galutinis įvesties kintamųjų rinkinys bei laikome, kad duomenų bazė parengta neuroniniam tinklui treniruoti.

Parengus duomenų bazę, dirbtiniam neuroniniam tinklui ištreniruoti reikia pasirinkti mokymo rūšį – su mokytoju, be mokytojo ar mišrų. Kadangi šiame darbe kuriamas valiutų kursų prognozavimo modelis, remiantis surinktais aiškiais apibrėžtais įvesties ir tikslo išvesties duomenimis, panaudotas prognozavimo užduotims labiau tinka mokymo su mokytoju neuroninio tinklo modelis, kurį ir naudosime savo sukurtame modelyje.

Kadangi darbe analizuojama kompleksinė ir didelė duomenų imtis, be to, sprendžiamas prognozavimo uždavinys, vieno sluoksnio tiesioginio sklidimo neuroninis tinklas, dar vadinamas parceptronu, nėra tinkamas šiam modeliui. Todėl prognozuojant valiutų kursus pasirinkta pritaikyti rekurentinių neuroninių tinklų algoritmą - ilgos trumpalaikės atminties neuroninius tinklus (angl. Long Short Term Memomy Networks). Sepp‘o Hochreeichter‘io ir Jurgen‘o Shmidhuber‘io sukurtas ilgos trumpalaikės atminties tinklas (toliau – LSTM) iš esmės bando „atsiminti“ visas ankstesnes sukauptas žinias ir siekia „pamiršti“ nesusijusius duomenis (Shmidhuber ir Hochreichter, 1997). Tai atliekama įvedant skirtingas aktyvavimo funkcijas. LSTM tinklas turi visiems rekurentiniams neuroniniams tinklams būdingą modulių grandininę struktūrą, tačiau skirtingai negu RNN, LSTM turi ne vieną, o keturius neuroninio tinklo sluoksnius, kurie sąveikauja ypatingai. Ilgos trumpalaikės atminties tinklą – LSTM modelį sudaro penki pagrindiniai komponentai, skirti įvairiems tikslams ir leidžiantys modeliuoti tiek ilgalaikius, tiek trumpalaikius duomenis (Ganegedara, 2018):

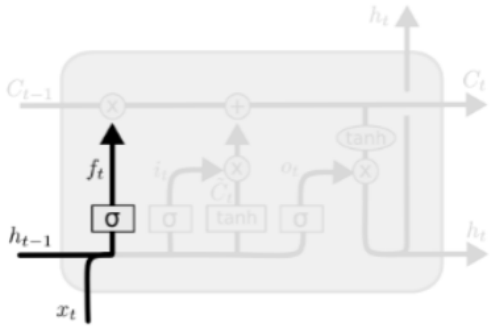
- Ląstelės būsena (angl. Cell state) – tai vidinė ląstelės atmintis, kurioje saugoma trumpalaikė ir ilgalaikė atmintis. Ląstelės būsena vaizduojama horizontalia linija (žr. 10 paveikslą), kuri schemoje viršuje eina tiesiai per visa grandinę turėdama tik nedidelę tiesinę sąveiką, todėl nepakeistą informaciją yra labai lengva perduoti. Ląstelės būsena paprastai žymima “c_t” raide.



10 paveikslas. Ląstelės būsenos scheminis atvaizdavimas LSTM struktūroje

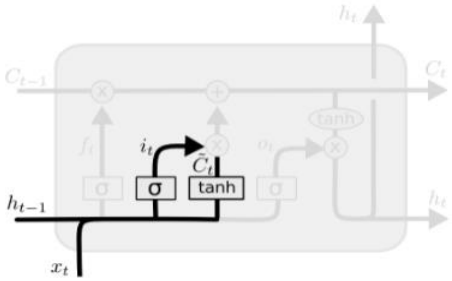
- Užmaršties vartai (angl. Forget gate) – nustato, kokiu mastu bus pamiršti ankstesni duomenys. Gavę informacijos iš ankstesnės būsenos (žymimos “h_{t-1}“) užmaršties vartai padeda priimti sprendimus kiek informacijos iš įvesties vartų ir ankstesnės ląstelės

būsenos reikia pašalinti ir atrinkti informaciją, kuri patenka į dabartinę ląstelės būseną (žr. 11 paveikslą). Užmaršties vartai žymimi raide "f_t".



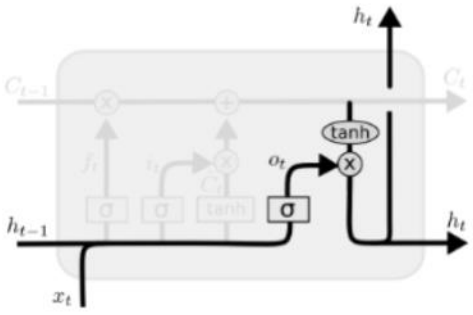
11 paveikslas. Užmaršties vartų scheminis atvaizdavimas LSTM struktūroje

- Įvesties vartai (angl. Input gate) – nustato kiek naujos informacijos reikia iš dabartinio įvesties srauto įrašyti į vidinę ląstelės būseną. Sigmoidinis įvesties sluoksnis nusprendžia kurias vertes reikia atnaujinti, o hiperbolinis tangentinis sluoksnis sukuria vektorių naujiems kandidatams įtraukti į esamą ląstelės būseną (žr. 12 paveikslą). Įvesties vartai žymimi "i_t" raide, tuo tarpu atnaujintas vektorius žymimas "$\widetilde{C_t}$".



12 paveikslas. Įvesties vartų scheminis atvaizdavimas LSTM struktūroje

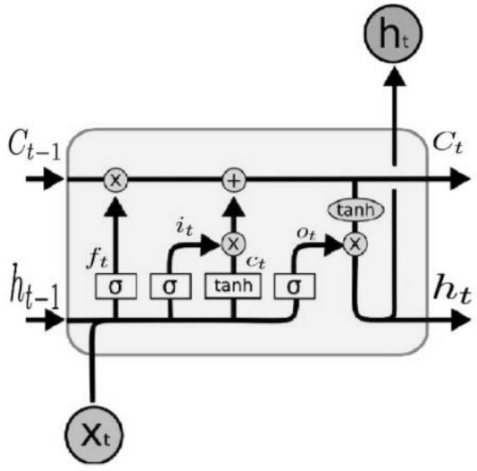
- Išvesties vartai (angl. Output gate) – nustato, kurią išvestį (kitą paslėptą būseną) sugeneruoti iš esamos vidinės ląstelės būsenos. Jie nusprendžia kiek informacijos iš dabartinės ląstelės būsenos patenka į paslėptą būseną, kad prireikus LSTM galėtų pasirinkti tik ilgalaikius ar ilgalaikius ir trumpalaikius prisiminimus (žr. 13 paveikslą). Žymima – "o_t" raide.



13 paveikslas. Įvesties vartų scheminis atvaizdavimas LSTM struktūroje

- Paslėpta būsena (angl. Hidden state) – tai išvesties būsenos informacija, apskaičiuota iš dabartinės išvesties ir ankstesnės paslėptos būsenos sandaugos. Ši informacija galiausiai naudojama numatyti būsimas kainas rinkoje.

Visa ilgos trumpalaikės atminties tinklo struktūra pavaizduota 14 paveiksle (O. Christopher, 2015). Diagramoje viršutine horizontalia linija vaizduojama ląstelės būsena. C_{t-1} žymimas perduotos informacijos srautas iš ankstesnės būsenos. Gavę informacijos iš ankstesnės būsenos (žymimos "h_{t-1}"), panaudojant LSTM modelį reikia priimti sprendimą, kokią informaciją yra nereikalinga ląstelės būsenai. Šį sprendimą priima užmaršties vartai, kurie atrenka ir išvalo informaciją iš ankstesnės langelio būsenos į dabartinę. Tinklas nusprendžia kokią naują informaciją kaupsime ląstelės būsenoje – kiekvienam skaičiui yra suteikiama reikšmė nuo 0 iki vieno, kur vienetas reiškia, kad informacija turi būti visiškai apsaugota, o nulinė reikšmė žymi, kad šios informacijos reiktų visiškai atsikratyti (B. Zhang, 2018). Taip yra atrenkama informacija ir pasiliekama tik reikšminga, kuri patenka į dabartinę ląstelės būseną. Vėliau yra sukuriamas vektorius, kuriame saugojama naujų kandidatų vertės ir kurios yra pridedamos prie dabartinės ląstelės būsenos. Atnaujiname seną tinklo būseną C_{t-1} į naują C_t, prie jo pridedame naujas kandidato vertes $i_t * C_t$, paskirstytas pagal tai, kiek yra nuspręsta atnaujinti kiekvienos būsenos vertę. Galiausiai pagal dabartinę būseną yra nusprendžiama kokia bus tinklo išvestis (Handa, R., Shrivas, A. K., Hota, H.S., 2019).



14 paveikslas. Ilgos trumpalaikės atminties tinklo struktūra
Šaltinis: O. Christopher, 2015

Pagrindinės LSTM lygčių rodikliai apskaičiuojami taip:

$$f_t = \sigma_s\left(W_{fx_t} + U_f h_{t-1} + b_f\right) \quad (4)$$

$$i_t = \sigma_s\left(W_{ix_t} + U_i h_{t-1} + b_i\right) \quad (5)$$

$$o_t = \sigma_s\left(W_{ox_t} + U_o h_{t-1} + b_o\right) \quad (6)$$

Kur:
W_q ir U_q yra matricos, kurios kaupia svorius ir pasikartojančias jungtis.
σ_s – sigmoidinė funkcija, σ_t – hiperbolinė tangento funkcija. Iš esmės σ vertė aktyvacijos funkcijos skaičiavimo metu gali būti skirtinga kiekvienose vartuose ir net atminties ląstelėse.

Norėdami surasti esamą vidinę ląstelės būseną c_t, pirmiausia reikia apskaičiuoti įvesties ir išvesties moduliacijos vartų dauginimo vektorius. Paskui apskaičiuoti užmaršties vartų ir ankstesnės vidinės ląstelės būsenos elementinį koeficientą bei pridėti du vektorius.

$$C_t = i_t * \sigma_t\left(W_{cx_t} + U_c h_{t-1} + b_c\right) + f_t * c_{t-1} = i_t * g + f_t * c_{t-1} \quad (7)$$

Norint apskaičiuoti esamą paslėptą būseną, pirmiausia reikia paimti esamo vidinės ląstelės būsenos vektoriaus hiperbolinį tangentą ir padauginti jį iš išvesties vartų reikšmės.

$$h_t = o_t * \sigma_t(c_t) \quad (8)$$

Pritaikę modelį ir suradę prognozuojamus valiutos kursus savo tyrime gautų rezultatų įvertinimui bus pritaikyti šie statistinės analizės metodai:

- Absoliučių paklaidų vidurkį (angl. Mean absolute deviation, MAD), kuris padeda įvertinti, kaip toli tikroji vertė yra nuo prognozuotosios.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|R_t - P_t| \quad (9)$$

  Kur:

  P_t – prognozuota kurso vertė

  R_t – tikroji valiutos kurso vertė

  n – prognozių skaičius

- Vidutinės kvadratinės paklaidos (angl. Mean Squared Error, MSE) bei šaknies iš vidutinės kvadratinės paklaidos (angl. Root Mean Squared Error, RMSE) metrikas, kurios dažnai naudojamos kokybės įvertinimui.

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(P_t - R_t)^2 \quad (10)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(P_t - R_t)^2} \quad (11)$$

Statistiniais metodais įvertinę prognozuojamus valiutos kursus, modelis yra priimamas arba atmetamas. Jeigu modelis fiksuoja nedideles paklaidas, bei suprognozuotos valiutų porų reikšmės mažai skiriasi nuo realių valiutos kursų, modelis yra taikomas suprognozuoti ateities valiutos kursus, tačiau jeigu prognozuojami valiutos kursai ženkliai nukrypę nuo realių reikšmių yra galimybė bandyti taikyti kitą rekurentinio neuroninio tinklo algoritmą.

3. VALIUTŲ KURSŲ PROGNOZAVIMAS, PANAUDOJANT DIRBTINIUS NEURONINIUS TINKLUS

Kaip ir kiekviename nagrinėtų autorių darbe, taip ir šiame tyrime buvo naudojamos kai kurios priemonės eksperimentams atlikti ir prognozės modeliams kurti. Kadangi diegti mašinų mokymosi algoritmus yra naudojamos tik kelios programavimo kalbos – Matlab, Python ir R, dėl gausių bibliotekų pasirinkimo ir prieinamumo buvo pasirinkta naudoti Python programinę įrangą kaip modelio kūrimo programavimo kalbą ištirti skirtingų valiutų istorinius duomenis bei išnagrinėti jų signalus pasinaudojant įvairiomis skirtingomis bibliotekomis (angl. libraries), kurios suteikia galimybė panaudoti jau įdiegtus algoritmus skirtingais būdais. Siekiant geriau perteikti tyrimo eigą, trumpai pateikiama informaciją apie pasirinktų bibliotekų ir programavimo įrankių niuansus.

Python yra bendrosios paskirties programavimo kalba, naudojama per įvairias platformas: internetą, matematines ir mokslines programas. Python programavimo erdvė yra gerai žinoma dėl savo paprastumo, kadangi ji iš pradžių buvo sukurta vaikams. Pastaruoju metu Python išpopuliarėjo dirbtinio intelekto ir mašinų mokymosi srityje dėl gausios bibliotekų apimties, kurios kaskart auga tarp šios srities kūrėjų ir ekspertų.

Šiame tyrime taip pat buvo naudojama daugybė bibliotekų, be kurių pagalbos pasirinktų valiutų prognozavimas būtų neįmanomas. Toliau įvardintos ir analizuojamos darbe naudotos bibliotekos bei pagrindinės jų pritaikymo naudos.

Numpy yra atvirojo kodo (angl. open source) biblioteka, kuri pagerina ir sutrumpina mokymo ir prognozavimo laiką. Ši biblioteka kaupia ir vykdo įvairius skaičiavimus daugialypiuose masyvuose bei matricose, kurios turi labai daug funkcijų, taip palengvindama darbą su jais bei padidindama duomenų analizės užduočių atlikimo spartą ir efektyvumą.

Kita atvirojo kodo biblioteka naudota darbe yra **Pandas**, kuri teikia įvairias duomenų struktūras ir duomenų analizės įrankius. Šia biblioteka yra ypatingai lengva naudotis atliekant įvairius veiksmus susijusius su skaitmeninėmis lentelėmis bei laiko eilučių duomenimis. Jei kalbėt apie „pandos“ svarbą, tai užtenka pasakyt, kad jos pagalba yra sukeliami bei importuojami pradiniai duomenys, be kurių joks tolesnis procesas nebūtų įmanomas.

Matplotlib yra braižymo biblioteka, kuri yra naudojama įvairiems grafikams kurti ir pavaizduoti. Matplotlib kaip ir kitos pasirinktos bibliotekos yra labai paprastai panaudojamos, o kelių kodų eilutėmis galima sukurti didelės kokybės erdvę, kurioje yra tiksliai parodyti skirtumai tarp numatytos ir relios išvesties.

Scikit-Learn yra dar viena Python biblioteka, kuri orientuota į mašininį mokymąsi. Biblioteka palaiko skirtingus klasifikavimo, regresijos ir grupavimo algoritmus. Ji palengvina algoritmų kūrimą tiems, kas juos diegia, taip pat Scikit-Learn atlieka svarbias užduotis, tokias

kaip duomenų normalizavimas, bei dinamiškai padalina duomenis į treniravimo ir testavimo dalis. Taip pat „Scikit-Learn“ gali būti naudojamas statistinių rodiklių, tokių kaip MSE ar MAPE pritaikymui.

Vis dėlto, pagrindinės bibliotekos, kurios buvo naudojamos šiame tyrime yra **TensorFlow** ir **Keras**. TensorFlow yra „Google“ kompanijos sukurta biblioteka, kuria iš pradžių buvo naudojamasi tik įmonės viduje, tačiau dabar ji yra viešai prieinama visuomenei. Iš esmės ji yra skirta matematinėms operacijoms atlikti ir šiais laikais yra plačiai naudojama mašininiam mokymuisi. Keras taip pat yra atvirojo kodo biblioteka, kurią sukūrė „Google“ inžinierius Francois Chollet. Specialios paskirties neuroninių tinklų biblioteka, skirta kurti gilaus mokymosi modelius yra patogi ir greita erdvė vykdyti eksperimentus neuroniniuose tinkluose. Keras bibliotekos naudojimas auga, kadangi šios bibliotekos pagalba yra sutaupoma laiko, ji naudoja tinkamas funkcijas bei yra puiki vieta kurti įvairius neuroninio tinklo modelius, tokius kaip atgalinio sklidimo, LSTM ir RBF.

Jupiter Notebook yra internetinė programa, teikianti tiesioginių kodų, vizualizacijų, lygčių ir paprastųjų tekstų kūrimą bei modifikavimą. Ši atvyro kodo „užrašų knygutė“ naudoja kelias programavimo kalbas, o šiame tyrime šios internetinės programos pagalba buvo rašomi kodai, kuriamas modelis bei vizualizuojami gauti rezultatai.

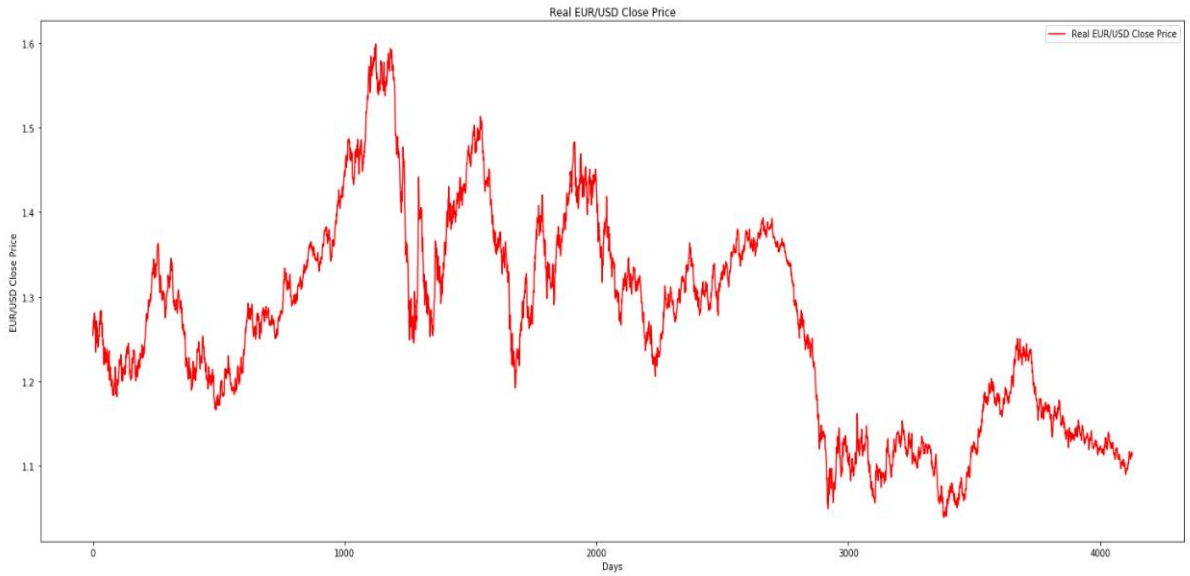
Trumpai aptarus šiam tyrimui naudojamą programinę įrangą bei pasirinktų bibliotekų aspektus pagal dirbtinio neuroninio tinklo modelio, skirto valiutų kursų prognozavimui, sukurtą metodiką yra pradedama kurti duomenų bazė. Pirminiame etape yra renkami pasirinkti valiutų kursų duomenys. Duomenų rinkiniai yra traukiami iš Šveicarijos banko internetinės svetainės „dukascopy.com“. Atlikdami atranką ir darydami paiešką daugelyje aktyvių pasaulinių valiutų keitimo bankų ir tarptautinių organizacijų duombazėse, padarėme išvadą, kad pasirinktas puslapis pateikia išsamius ir aiškius pasirinktų tikslinių valiutų duomenis. Prognozavimo modelio veikimui įvertinti buvo pasirinktos EUR/USD, EUR/GDP bei EUR/JPY porų dieninės valiutų kursų vertės. Valiutų porų duomenys buvo nagrinėjami nuo 2004 m. sausio 1 d. iki 2019 m. spalio-31 d. laikotarpio diapazone, kas sudaro iš viso po 4128 stebėjimus kiekvienos valiutų poros pradiniuose duomenų rinkiniuose. Pateikiami dieniniai duomenys, tai reiškia, kad kiekvieną dieną egzistuoja tik vienos dienos išrašas. Šių trijų duomenų rinkiniai turi šiuos atributus, kurie yra naudojami prognozavimui įvykdyti:

- Data (angl. date)
- Atidarymo kaina (angl. open)
- Aukščiausia kaina (angl. high)
- Žemiausia kaina (angl. low)
- Uždarymo kaina (angl. close)

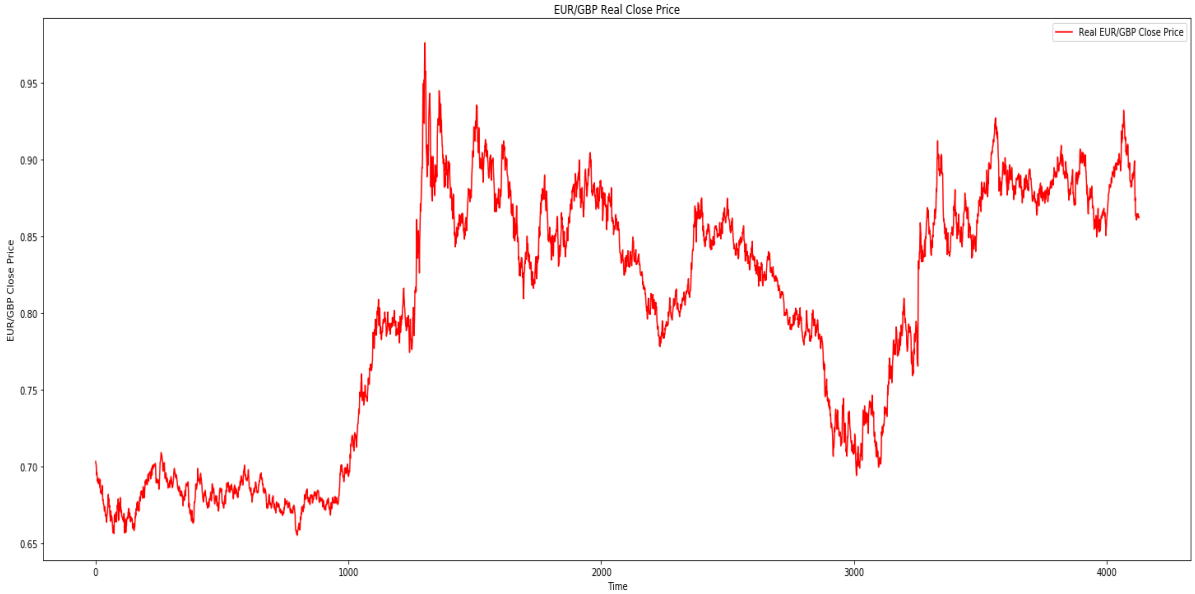
- Kiekis (angl. volume)

Keturi atributai (atidarymo kaina, aukščiausia kaina, žemiausia kaina ir kiekis), naudojami treniruoti algoritmų įvestis, o paskutinis atributas – uždarymo kaina („close“) naudojama išvesčiai ir kiekvienos dienos uždarymo kainos prognozavimui.

Pradedant tirti nagrinėjamus duomenų rinkinius, kiekviena valiutų pora yra pavaizduojamas taip, kad būtų galima susidaryti bendrą vaizdą apie pasirinktos valiutos judėjimą euro atžvilgiu tiriamuoju laikotarpiu. Grafikai sukurti pasinaudojant Matplotlib bibliotekos pagalba su paprasčiausia funkcija *#data.plot()*. Turint grafinį vaizdą nesunku pamatyti tiriamųjų valiutų porų kainų nepastovumą ir kintamumą. Žemiau pavaizduoti pasirinktų valiutų porų uždarymo kainos nagrinėjamu laikotarpius (1, 2, 3 grafikai).



1 grafikas. EUR/USD kurso pokyčiai nagrinėjamu laikotarpiu 2004.01.01 – 2019.10.31



2 grafikas. EUR/GBP kurso pokyčiai nagrinėjamu laikotarpiu 2004.01.01 – 2019.10.31

3 grafikas. EUR/JPY kurso pokyčiai nagrinėjamu laikotarpiu 2004.01.01 – 2019.10.31

Norint ištreniruoti dirbtinį neuroninį tinklą ir parengti algoritmą mašinų mokymuisi, kuris gebėtų suprognozuoti valiutų kursus pritaikius ilgos trumpalaikės atminties neurininius tinklus – LSTM, reikalinga paruošti ir išvalyti duomenis, kad jie būtų tinkami ir galėtų būti naudojami mokymui. Šis procesas vadinamas duomenų apdorojimu („data processing“).

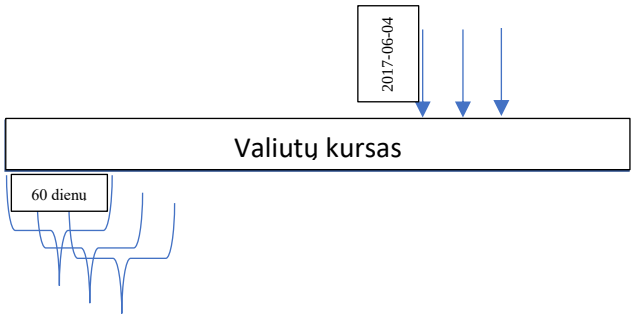
Norint naudoti Python programinę įrangą bei visas jos bibliotekas svarbu tinkamai paruošti duomenis, kad tinklas gebėtų skaityti duomenis, taigi pati pirma užduotis išsitraukus duomenis iš dukascopy.com puslapio buvo tinkamai parengti datos stulpelį, kadangi iš svetainės ištraukti duomenys buvo pateikti netinkama struktūra. Duomenyse buvo pateikta ne tik valiutos stebėjimo data, bet ir laikas, kuris nėra reikalingas prognozavimo procesui, todėl informaciją buvo apdorota taip, kad duomenyse matytųsi tik stebėjimo data. Taip pat, remiantis metodologija, reikėjo patikrinti, kad duomenyse neegzistuotų nulinių verčių. Ištrauktuose duomenyse nulinės vertės buvo „apimties“ stulpeliuose, kadangi šeštadieniais ir sekmadieniais prekyba neegzistuoja. Savaitgaliais rinkos vertė nėra aiški (kiti stulpeliai buvo užpildyti ankstesniais stebėjimais), todėl šiuos duomenis reikėjo pašalinti iš nagrinėjamos duomenų bazės.

Normalizuoti duomenis, kad jie nebūtų labai skirtinguose diapazonuose ir neiškreiptų modelio buvo pasinaudota *#MinMaxScaler* funkcija naudojantis „Scikit-Learn“ biblioteką. Ji leido duomenis normalizuoti taip, kad didžiausia reikšmė būtų arti vieneto, o mažiausia – nulio, o kitos vertės svyruotų tarp šio diapazono.

Visų valiutų porų duomenys buvo apdorojami ir valomi vienodai, tad pabaigus pagrindinį duomenų apdorojimą buvo pradedamas duomenų mokymo ir testavimo procesas. Atlikus duomenų normalizavimą, duomenys buvo padalyti į dvi dalis. Treniravimo ar mokymo rinkinys yra naudojamas mokymosi procesui, kuriuo algortimas moko tinklą kaip atpažinti tam tikrus signalus ir koreliaciją tarp įvesčių, jų svorių ir išvesčių. Testavimas atliekams po modelio

apmokymo, kad patikrinti modelio tikslumą. Nors dauguma tyrėjų yra pratę paskirstyti, kad 80% būtų skirta mokymuisi, o likę procentai tinklo testavimui, atlikę daugybę bandymų su įvairiu santykiu, pasirinkome, kad duomenys nuo 2004 m. sausio 1 d. iki 2017 m. birželio 1 dienos bus naudojami neuroninio tinklo mokymui, o likusieji duomenys – testavimui (nuo 2017 m. birželio 4 d. iki 2019 spalio 31 dienos). Pagal šią informaciją beveik 85 procentai visų duomenų bus naudojami mokymuisi (3500 eilutės), o likę maždaug 15% - tinkle testavimui (628 eilutės).

Gauti testavimo duomenis buvo pasirinkta, kad 2017 m. birželio 4 d., kuri yra pirma testavimo duomenų diena, bus prognozuojama pagal pirmąsias 60 treniravimo dienų (nuo 1 dienos iki 60), tuo tarpu antrosios testavimo dienos rezultatą (2017 m. birželio 5 d.), gauname pagal sekančias 60 dienų (nuo 2 d. iki 61 d.). Paprasčiau šitas procesas pavaizduotas 15 paveiksle.



15 paveikslas. Suprognozuotų testavimo dienų duomenų gavimo schema
Šaltinis: sudaryta autoriaus

Atlikę duomenų treniravimo ir testavimo procesus pereiname prie LSTM modelio kūrimo, naudojant „Keras“ ir „Tenserflow“ bibliotekomis. Šiame etape buvo dirbama su įvesties, išvesties ir keturiais paslėptais sluoksniais. Gauti kuo tikslesnį rezultatą buvo išbandytos dvi aktyvavimo funkcijos „ReLU“ ir „sigmoid“. Kadangi „ReLU“ yra įvardijama, kaip geriausiai tinkanti neuroninio tinklo mokymo efektyvumui bei pritaikyta ji davė geresnį rezultatą, ją naudosime kaip tinklo aktyvavimo funkciją.

Tyrime buvo atlikti bandymai su skirtingu epochų skaičiumi bei paketo dydžiu. Remiantis literatūrą, buvo pritaikyti dažniausiai naudojamas epochų skaičius: 10, 50 ir 100 epochų (angl. epochs), bei skirtingas paketo dydis (angl. batch-size), kur literatūroje dažniausiai minima, kad prognozavimo darbai atliekami su 32, 64 ir 128 paketo dydžiais, siekiant surasti kuo įmanoma geresnes šių parametrų reikšmes. Epochos skaičius yra tarsi ciklų skaičius, per kurį neuroninis tinklas yra išmokomas. Pasirinktas epochų skaičius vaidina svarbų vaidmenį prognozuojant valiutų kursus (S. S. Saini ir kt., 2016). Tyrimo metu buvo pastebėta, kad mokant tinklą su daugiau epochų, tinklas labai gerai išmoksta duomenų modelį ir pateikia tikslesnius rezultatus, tačiau naudojant per didelį epochų skaičių galima neišvengti persimokymo (angl. overfitting) problemos. Paketo dydis leidžia nustatyti mišrų rėžimą, kuriuo tinklas mokosi. Kadangi šiuo

tyrimu nėra siekiama palyginti architektūrų naudojimo apmokymui efektyvumą, nes yra naudojama tik viena architektūra, skirtingiems valiutų kursų apmokymams buvo naudojamas geriausiais paketo dydis, neatsižvelgiant, jei skirtingų porose jie nesutaptų. Visi išbandyti epochų ir paketo dydžio rinkiniai naudoti pasirinktoms valiutų poroms pateikiami 4 lentelėje, kur atsižvelgiant į vidutinės standartinės paklaidos rodiklio reikšmes buvo pasirinkti geriausi parametrų rinkiniai.

4 lentelė: Epochų skaičiaus ir paketo dydžio rinkinių pasirinkimai kiekvienai valiutų porai pagal MSE rodiklį.

Epochų skaičius/ Paketo dydis	EUR/USD	EUR/GBP	EUR/JPY
10/32	0,0003636	0,0006893	2,1702723
10/64	0,0001786	0,0003918	2,3562256
10/128	0,0002332	0,0007274	3,2036526
50/32	3,8653868	0,0003724	0,7596632
50/64	0,0002097	0,0005582	1,6542547
50/128	0,0001776	0,0005719	3,1599250
100/32	0,0003941	0,0004909	1,4980599
100/64	0,0002004	0,0007973	1,5120251
100/128	0,0001133	0,0013182	3,4580787

4 lentelėje žalia spalva pažymėti mažiausio vidutinės standartinės paklaidos reikšmės, kurios parodo geriausių epochų skaičiaus ir paketo dydžio derinius. Apmokant kiekvienos valiutos porą atskirai, geriausias visų porų gaunamas rezultatas buvo su 50 epochų, tačiau skirtingoms poroms buvo parenkamas skirtingas paketo dydis (EUR/USD – 128, EUR/GBP ir EUR/JPY – 32).

Šiam modeliui buvo taip pat pritaikytas Adam optimizavimo algoritmas, kuris padeda gauti geresnius prognozuojamus rezultatus. Adam optimizavimo algoritmas naudojamas vietoj klasikinės stochastinio gradiento nusileidimo algortimo, siekiant atnaujinti tinklo svorio iteraciją atsižvelgiant į treniravimo duomenis. Pastaruoju metu Adam algoritmas yra plačiai pritaikomas daugybėje giluminio mokymosi programose.

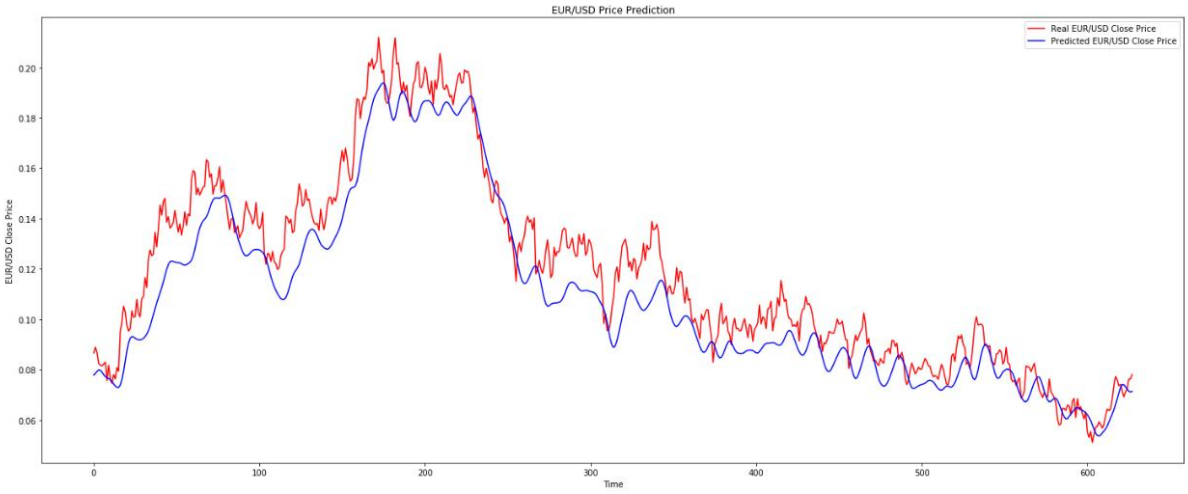
Išmetimo (angl. dropout) technika yra naudojama, kad apsaugotų paslėptus sluoksnius nuo perderinimo, tinklo mokymo metu. Pagrindinė šios technikos idėja yra atsitiktinai iš neuroninio tinklo išmetinėti neuronus (N. Srivastava ir kt., 2014). Naudojant šią techniką, buvo pasirinkta, kad neuronai iš tinklo bus išmetami su tikimybe 0,2.

Visa susisteminta, su visais naudotais parametrais ir pasirinktomis skirtingų valiutų kursų porų reikšmėmis pateikiama 5 lentelėje apačioje, o pilni valiutų kursų prognozavimo python kodai pateikiami 2, 3 ir 4 prieduose.

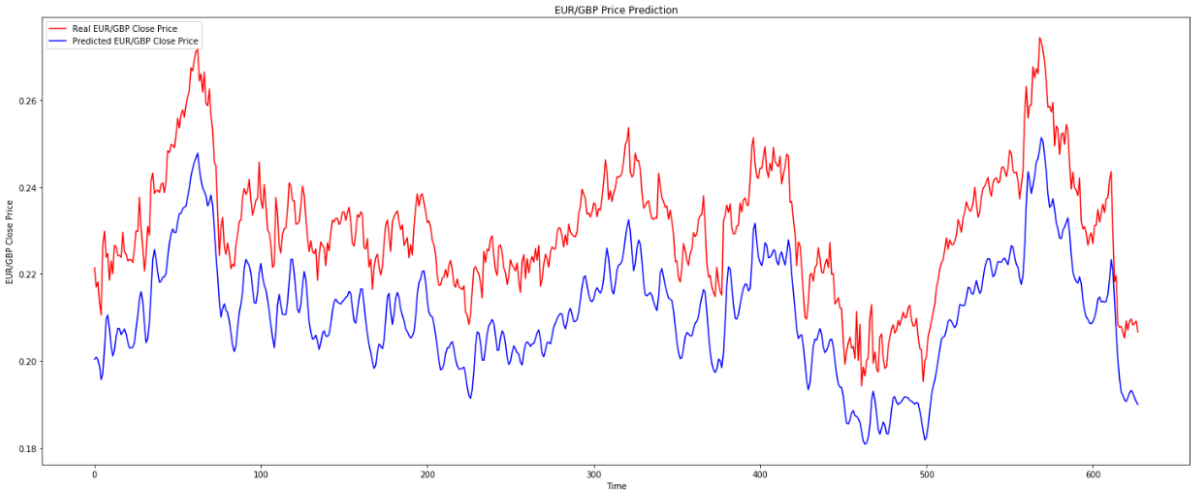
5 lentelė: Susisteminta tyrime naudoti parametrai, reikalingi suprognozuoti pasirinktus valiutų kursus

Valiutos kursai	EUR/USD	EUR/GBP	EUR/JPY
Treniravimo stebėjimų skaičius	3500	3500	3500
Testavimo stebėjimų skaičius	628	628	628
Epochų skaičius (angl. epochs)	50	50	50
Paketo dydis (angl. batch_size)	128	32	32
Optimizavimo funkcija (angl. optimizer)	Adam	Adam	Adam
Paslėptų sluoksnių aktyvavimo funkcija	ReLU	ReLU	ReLU
Išmetimo tikimybė (angl. dropout)	0,2	0,2	0,2
Praradimo reikšmė (angl. loss)	0,00170000	0,00083576	0,00078196

Kadangi patogiausia ir suprantamiausia gautas reikšmes atvaizduoti grafikuose, pasinaudodami Matplotlib bibliotekos pagalbą su funkcija *#plt.plot* galime palyginti tiek realius valiutos kursus, tiek suprognozuotus, kad aiškiai matytumėme skirtumą tarp jų. Žemiau pateikti grafikai, kuriuose atvaziduoti EUR/USD, EUR/GBP ir EUR/JPY valiutų kursų realios kainos ir suprognozuotos reikšmės, kad būtų aiškiau nustatyti, kiek arti ar kiek nutolus yra prognozė nuo tikrosios vertės (4, 5, 6 grafikai).



4 grafikas. EUR/USD realios ir suprognozuotos reikšmės tiriamuoju laikotarpiu 2017.06.01 – 2019.10.31



5 grafikas. EUR/GBP realios ir suprognozuotos reikšmės tiriamuoju laikotarpiu 2017.06.01 – 2019.10.31

6 grafikas. EUR/JPY realios ir suprognozuotos reikšmės tiriamuoju laikotarpiu 2017.06.01 – 2019.10.31

Modelio tinkamumo ir reikšmingumo įvertinimui, kiekvienai valiutų porai atskirai *#sklearn.metrics* funkcijos pagalba buvo apskaičiuoti du statistiniai matematiniai rodikliai: absolučių paklaidų vidurkis MAE (angl. mean absolute error) ir vidutinės kvadratinės paklaidos rodiklis MSE (angl. mean squared error) (žr. 5 lentelę). Tiek MAE, tiek MSE statistiniai metodai padeda įvertinti kiek tikroji vertė yra nutolusi nuo prognozuojamos. Susisteminta gautų rezultatų skirtingų vertinimo metodų lentelė pateikiama žemiau (žr. 5 lentelę).

5 lentelė: Susisteminti gauti modelio tikrinimo rodiklių rezultatai.

Statistiniai rodikliai	EUR/USD	EUR/GBP	EUR/JPY
MSE	0,0001776	0,0003724	0,7596633
MAE	0,0108038	0,0189934	0,8675761

Iš pateiktos lentelės matome, kad įvertinus visų suprognozuotų valiutų kursų rezultatus, sunku išskirti geriausios valiutų poros, pagal šiuos įvertinimo rodiklius, kuri labiausiai tinka sudarytam modeliui. Iš šių rezultatų galime tik pagrįsti modelio tinkamumą, kadangi šiandieniniuose tyrimuose, naudojant dirbtiniu nuroninius tinklus ir pritaikant juos laiko eilučių prognozavime, puikūs vidutinės kvadratinės paklaidos rezultatai svyruoja nuo 0,02 klasikiniame atgalinio sklidimo (angl. back-propagation) neuroniniuose tinkluose iki 0,002 vertės daugialogiškų daugiareikšmių (angl. multi-logic multi-valued) neuronų tinkluose. Tad atsižvelgiant į rezultatus, gautus vertinant pagal vidutinės kvadratinės paklaidos metodą (MSE = 0,0001776 EUR/USD valiutų poroje, MSE = 0,0003724 EUR/GBP valiutų poroje bei MSE=0,7596633 EUR/JPY poroje), galime teigti, kad LSTM algoritmu suprognozuotos EUR/USD ir EUR/GBP valiutų kursų reikšmės yra labai patikimos, nes gautos reikšmė pakliūna į šiuos rėžius, o EUR/JPY rezultatas yra neblogas.

Galiausiai, patikrinus sukurto modelio tinkamumą, pagrinde *#shift* funkcijos pagalba buvo suprognozuoti numanomos valiutų kursų uždarymo kainos 5 dienų, 20 dienų ir vienų metų laikotarpiais į priekį. Tokie laikotarpiai pasirinkti ne atsitiktinai - jie atspindi vienos savaitės, vieno mėnesio ir vienų metų laikotarpius, kadangi mūsų modelis neprognozuoja savaitgalinių

reikšmių, kuriuose uždarymo kaina būna tokia pati, kaip ir savaitės paskutinės biržos darbo dienos, kada vyksta prekyba. Taigi, vienoje savaitėje yra 5 darbo dienos, todėl norint sužinoti vienos savaitės prognozę, suprognozavome 5 dienų į priekį valiutų kursus. Savaitinės prognozės reikšmės pateiktos 6 lentelėje. Iš savaitinių duomenų matome, kad skirtingų valiutų porų uždarymo kainos neturi trumpalaikės nuoseklios tendencijos augti ar mažėti, o minimalus kainų pasikeitimas vyksta kiekvieną dieną. Modelio tikslumas (angl. accuracy) šiai trumpalaikei prognozei visų valiutų porų kursams yra labai aukštas (EUR/USD kurso prognozės tikslumas siekia 98,10%, EUR/JPY – 98,49%, o EUR/GBP net 98,61%).

6 lentelė: Suprognozuotų nagrinėjamų valiutų porų kursų uždarymo kainos 5 dienas į ateitį

Dienos į priekį	EUR/USD	EUR/GBP	EUR/JPY
1	1,115041	0,860361	120,840914
2	1,115822	0,864364	121,181140
3	1,111705	0,862242	120,619752
4	1,109845	0,864201	120,477651
5	1,112711	0,863820	121,099193

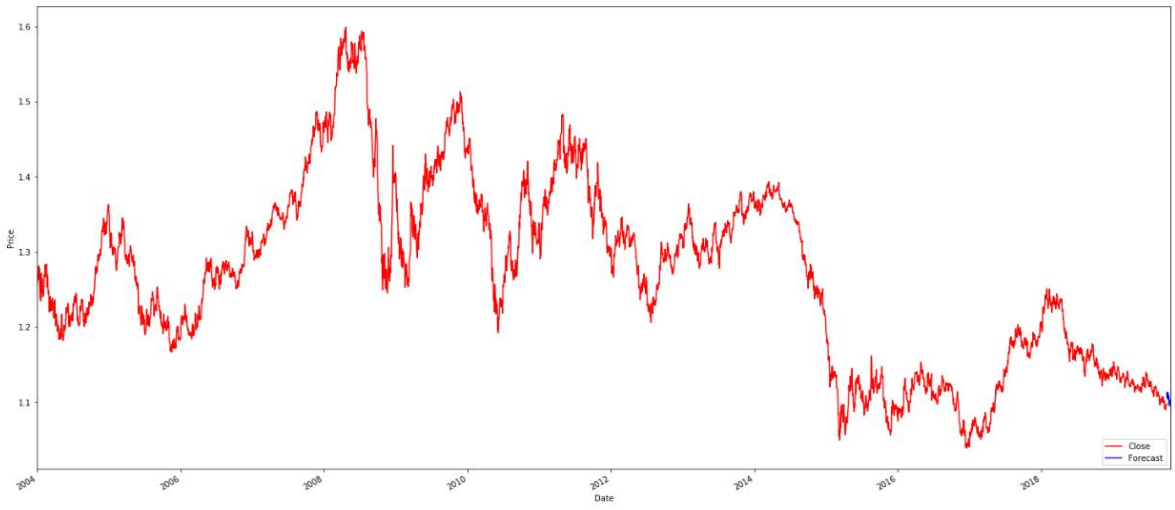
Skirtingų laikotarpių pasirinkimo diferencijavimas leidžia patikrinti modelio tinkamumą pagal gaunamą prognozės tikslumą, todėl patikrinti modelį buvo atliktos ir mėnesio (20 dienų) prognozės į priekį. Suprognozuoti kursai neatsižvelgia į nedarbo dienas, tad reikėtų suprasti, kad šios uždarymo kainos prognozuoja tik dienas, kada vyksta prekyba šiomis valiutomis. Dėl šios priežasties mes prognozuojame ne 30 dienų į priekį (tiek iš viso dienų turi lapkričio mėnesis), bet 20 dienų, kadangi tiek dienų lapkričio mėnesį vyko. 7 pateiktos dvidešimt dienų į priekį suprognozuotos pasirinktų valiutų kursų reikšmės.

7 lentelė: Suprognozuotų nagrinėjamų valiutų porų kursų uždarymo kainos 20 dienas į ateitį

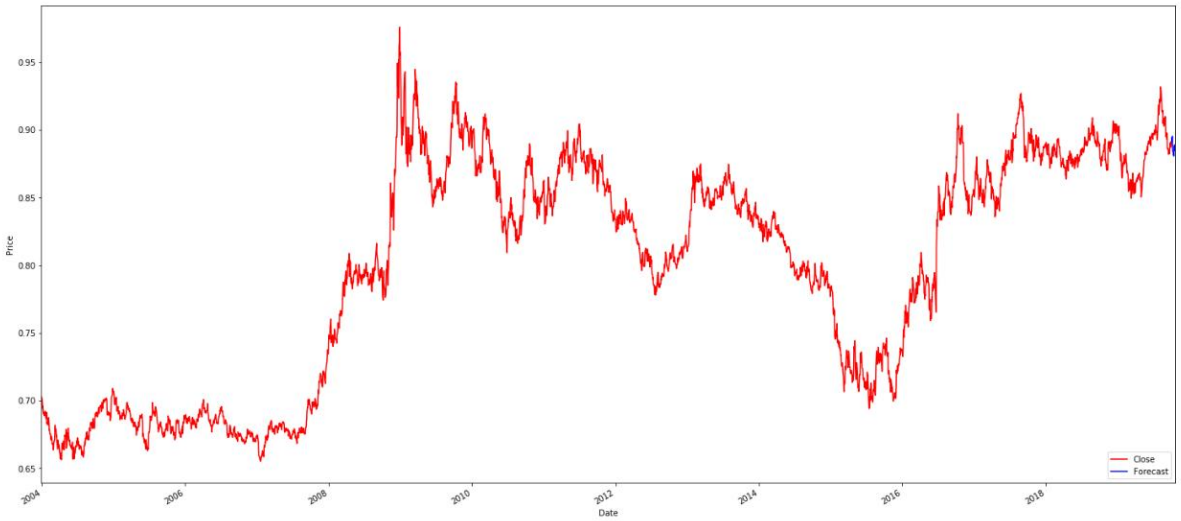
Dienos į priekį	EUR/USD	EUR/GBP	EUR/JPY
1	1,112390	0,894153	119,488667
2	1,106502	0,892252	119,067719
3	1,112406	0,891043	120,046255
4	1,113032	0,895247	120,020693
5	1,104356	0,886367	119,381164
6	1,112762	0,884650	120,125255
7	1,108352	0,884500	120,243001
8	1,108676	0,883368	119,629365
9	1,107648	0,882229	118,907187
10	1,104989	0,881559	118,731631
11	1,108522	0,882639	118,224302
12	1,097497	0,880697	118,547559
13	1,098242	0,884620	118,203172
14	1,101037	0,884860	118,379503
15	1,095270	0,888421	118,250230
16	1,100930	0,885087	118,332996
17	1,102466	0,888042	117,999032
18	1,102769	0,888973	117,576481
19	1,103159	0,887174	118,069145
20	1,102866	0,888928	118,074254

Iš pateiktos lentelės galime matyti, kad EUR/USD valiutos kurso uždarymo kainos dvidešimt dienų prognozės intervalas yra [1.095270 iki 1.113032], EUR/GBP suprognozuotos ateities uždarymo kainos svyruoja nuo 0,880697 iki 0,895247, o Euro ir Japonijos jenos uždarymo kainos prognozuojama, kad svyruos nuo 117.576481 iki 120,243001. Atsižvelgus į gautas prognozės galime matyti, kad trumpajame laikotarpyje nei vienos nagrinėtos valiutų poros atveju nepastebima valiutų poros uždarymo kainų didėjimo ar mažėjimo tendencija. Klaidų matavimo metodikos, bei valiutų porose aptinkamų klaidų skaičiaus, palyginus su kitomis tiriamomis valiutų poromis, neužtenka įvertinti modelio tinkamumo. Reikia atsižvelgti į šiandienines aktualijas, vyraujančias tendencijas. Euro ir Didžiosios Britanijos svaro kurso atveju, atsižvelgus, kad rinkoje vyrauja netikrumas dėl Brexit be susitarimo, tikėtina, kad kursas svyruos, kol nebus priimtas sprendimas, kuris ir turės lemiamą įtaką šiam kursui. Šiandieninėmis nuotaikomis, įsigaliojus naujam Europos Sąjungos ir Jungtinės Karalystės susitarimui ir nusprendus vykdyti tiesioginius rinkimus gruodžio viduryje, rizika, kad bus pasitraukta iš Europos Sąjungos be susitarimo labai sumažėjo. Tai šiek tiek sustiprino svarą ir tikėtina, kad EUR/GBP kursas ilguoju laikotarpiu šiek tiek augs, bet trumpuoju laikotarpiu galimai nedidelis svyravimas išliks. Vis dėlto neatmestina galimybė, kad prekybos derybos gali žlugti, kas turėtų visiškai kitokią įtaką šiam valiutos kursui. Euro ir JAV dolerio kursas šiuo metu nežymiai, bet krenta, kadangi doleris ir toliau pelnosi iš pasaulinio ekonomikos letėjimo. Tačiau prognozuojamas valiutos susilpnėjimas, kadangi JAV Federalinis rezervų bankas pirmą kartą nuo praėjusios finansų krizės sumažino savo bazines palūkanas nuo dabartinių 2,25 -2,5% iki 2-2,25%, atsižvelgiant į tai manome, kad valiutos kursas palaipsniui ilguoju laikotarpiu turetų didėti.

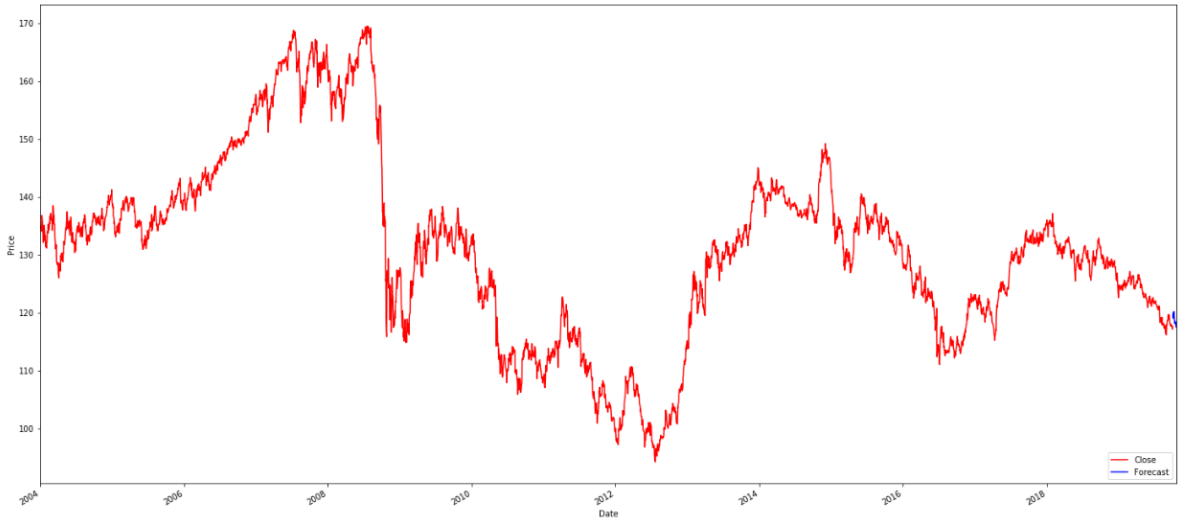
Visų valiutų kursų uždarymo kainų prognozių reikšmės mėnesiui į priekį atspindėtos grafikuose (7, 8, 9 grafikai). Raudona spalva pažymėtos realios nagrinėjamo valiutos kurso uždarymo kainos, tuo tarpu suprognozuotos reikšmės žymimos mėlyna spalva.



7 grafikas. EUR/USD realių bei 20 dienų į ateitį suprognozuotų uždarymo kreivė

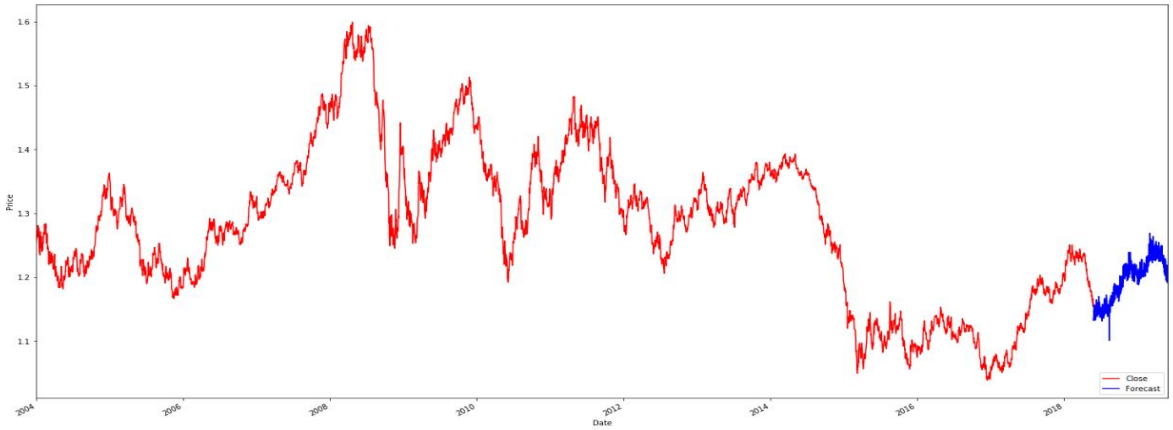


8 grafikas. EUR/GBP realių bei 20 dienų į ateitį suprognozuotų uždarymo kreivė

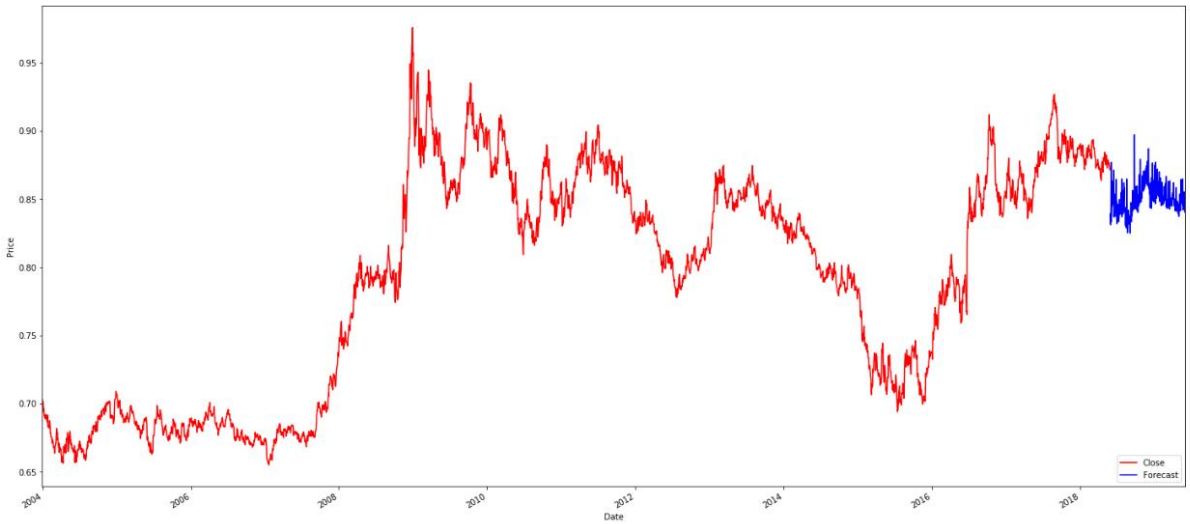


9 grafikas. EUR/JPY realių bei 20 dienų į ateitį suprognozuotų uždarymo kreivė

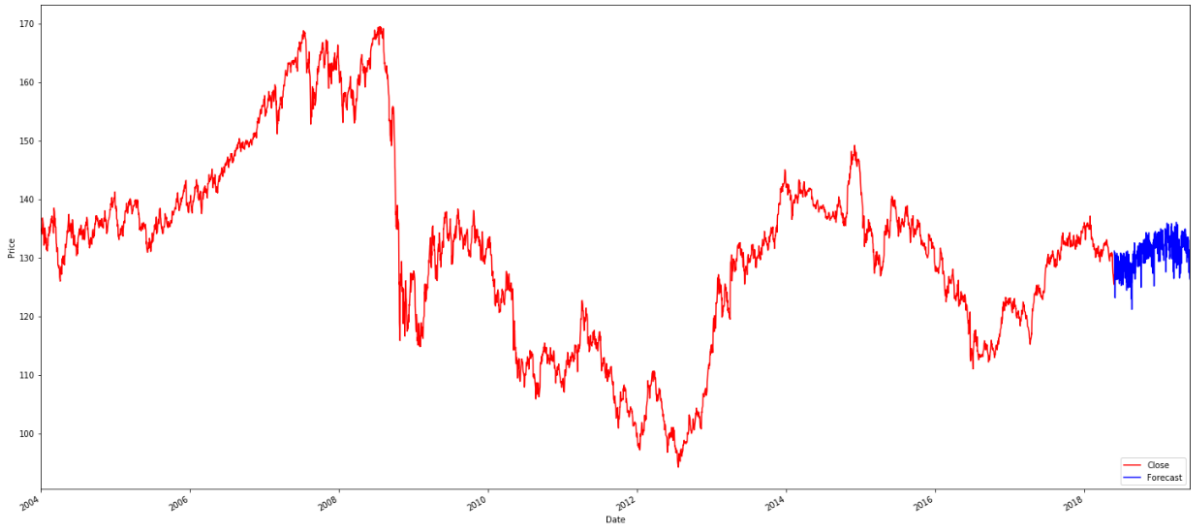
Kadangi šiame darbe sukurtas modelis yra tinkamas prognozuoti pasirinktą laikotarpį į priekį, pasirinkome suprognozuoti valiutų kursų uždarymo kainas 365 dienas į ateitį (žr. 10, 11, 12 grafikai). Juose galime įžvelgti, kad EUR/USD bei EUR/JPY kursų uždarymo kainos ilgame laikotarpyje turi tendenciją didėti, tuo tarpu EUR/GBP kursas turėtų nedaug sumažėti, bet kaip ir minėjome prieš tai, jam didelės įtakos turės sprendimas dėl Brexito.



10 grafikas. EUR/USD realių bei 365 dienų į ateitį suprognozuotų uždarymo kreivė



11 grafikas. EUR/GBP realių bei 365 dienų į ateitį suprognozuotų uždarymo kreivė



12 grafikas. EUR/JPY realių bei 365 dienų į ateitį suprognozuotų uždarymo kreivė

Vis dėlto, ilgesnio laikotarpio prognozavimas į priekį turi įtakos modelio tikslumui. Prognozuojant 365 dienas, lyginant su 20 į priekį, visais atvejais modelio tikslumas akivaizdžiai krenta. Tiriant euro ir JAV dolerio kursą, prognozės tikslumas krito nuo 91,76% iki 32,22%, EUR/GBP kurso tikslumas krito nuo 95,81% prognozuojant 20 dienų į priekį iki 29,67% prognozuojant metus į ateitį, tuo tarpu euro ir Japonijos jenos kursas nuo 92,23% krito net iki 12,83%, todėl vadovautis tokia metine prognoze, švelniai pasakius būtų neprotinga. Visi prognozavimo į ateitį kodai pateikiami 5-13 prieduose.

Apibendrinant galime teigti, kad sukurtas modelis yra tinkamas prognozuoti valiutų kursus bei nuspėti galimas ateities reikšmes. Pasinaudojant LSTM modeliu buvo suprognozuotos trijų valiutų kursų (EUR/USD, EUR/GBP ir EUR/JPY) uždarymo kainos, bei panaudojant statistinius rodiklius, gauta, kad puikius rezultatus parodė visos tiriamos valiutų poros. Šio tyrimo rezultatai įrodė, kad investuotojai ir spekuliantai galėtų naudotį sukurtą modelį, norėdami ištirti finansų rinką bei suprognozuoti pasirinktų valiutų porų kursus.

Suprognozavus uždarymo kainas 5, 20 ir 365 dienų į priekį, gautus rezultatus galime lyginti su šių dienų realiosiomis kainomis bei žiūrėti kaip modelis veikia. Sukurtas modelis yra labai universalus, nes tinkamai išsitraukus ir apdorojus duomenis, galima nesunkiai suprognozuoti bet kokius valiutų kursus. Taip pat, galima prognozuoti pasirinktą laikotarpį į priekį, vis dėlto, nuo to nukenčia modelio tikslumas – prognozuojant metus į priekį, lyginant su savaitės į ateitį prognoze, modelio tikslumas ženkliai krenta (tiriant EUR/GBP kursą tikslumas krito nuo 98,61% iki 29,67%, EUR/USD atveju – nuo 98,10% iki 32,22%, o dirbant su EUR/JPY kursu, modelio patikimumas kristų iki 12,83%. Taigi, nors modeliu ir galima diferencijuoti prognozės laikotarpį, reiktų tai daryti atsargiai, kadangi pati prognozė gali tapti ne tokia tiksli.

IŠVADOS IR PASIŪLYMAI

Išvados:

1. Auganti ekonomika, kartu su sparčiai tobulėjančiomis technologijomis reikalauja prognozuoti beveik kiekvieną pramonės šaką, nesvarbu ar tai būtų kliento rinkodara, automobilių, telefonų pasiūla rinkoje ar valiutos kursai. Valiutos kurso prognozavimas tapo iššūkiu žmonėms, kadangi valiutos kursas kinta chaotiškai, jį prognozuoti neatlikus kokių skaičiavimų, ar nepadarius fundamentaliosios analizės yra praktiškai neįmanoma. Vis dėlto, žmonėms valiutos keitimas yra gyvybiškai svarbus, kadangi beveik kiekvieno žmogaus ekonomika yra tiesiogiai ar netiesiogiai susijusi su užsienio valiutos kainomis. Todėl prognozuoti valiutų kursus žmonės pasitelkė mašinų mokymąsi ir paprastas dirbtinio intelekto technologijas.

2. Išanalizavus skirtingų autorių darbus, išskiriama, kad pagrindiniai veiksniai, turintys didžiausios įtakos valiutų kursams yra tokie makroekonominiai veiksniai, kaip infliacija, palūkanų normos lygis, mokėjimų balansas, prekybos sąlygos ir vyriausybės skola bei vienas veiksnys, labiau susijęs su valstybes politika, nei su ekonomika – tai politikos stabilumas ir efektyvumas. Šių veiksnių pokytis turi didelį impulsą valiutų kursų pasikeitimams, todėl nagrinėjant valiutų poras, svarbu atkreipti dėmesį ir įvertinti šiuos abiejų šalių rodiklius, kadangi valiutų kursai yra santykiniai dydžiai ir jie yra išreiškiami kaip dviejų šalių valiutų palyginimas.

3. Neuroninių tinklai turi aibę išraiškų. Priklausomai kaip neuronai tinkluose yra sujungiami (kokia jų neuroninių tinklų architektūra) ir kokia kryptimi jie gali siųsti signalus jie skirstomi į tiesioginio sklidimo ir rekurentinius neuroninius tinklus. Tiesioginio sklidimo neuroniniuose tinkluose informacija keliauja tik viena kryptimi, ir skirtingai negu rekurentiniuose tinkluose, jose nėra jokių grįžtamųjų ryšių. Tuo tarpu Grįžtamojo ryšio neuroniuose (arba rekurentiniuose) tinkluose, informacija keliauja ne tik iš įvesčių, bet gali ir priešinga krytimi. Taip pat rekurentiniai tinklai pasižymi savybe, kad gali saugoti informaciją vidinėje atmintyje ir panaudoti ją bet kuriai įvesties sekai apdoroti. Šių tinklų grupės yra dar plačiau skirstomos į įvairius algoritmus, kurie yra naudojami ir pritaikomi priklausomai nuo sprendžiamo uždavinio tikslo bei jo sudėtingumo.
4. Remdamiesi rekurentinio tinklo algoritmu - LSTM neuroniniu tinklu, šiame tyrime buvo sukurtas modelis, pagal kurį palygintos trijų valiutų porų kainos Forex rinkoje. Ištyrus valiutų poras ir pritaikius įvairius statistinius parametrus buvo nustatyta, kad

modelis yra gerai veikiantis bei patikimas, todėl yra tinkamas prognozuoti valiutų kursus į ateitį. Vis dėlto, nepaisant to, kad modelis yra pritaikytas prognozuoti įvairaus ilgumo laikotarpiu į priekį, reikėtų koncentruotis į prognozes trumpuoju laikotarpiu, kadangi šių prognozių patikimumas šiame modelyje yra didžiausias.

Pasiūlymai:

Išanalizavus valiutų kursų bei dirbtinių neuroninių tinklų teorines dalis bei stebint šių dienų aktualijas yra nenuginčyjama, kad šiandieniniame pasaulyje yra populiaru ir būdinga naujoms, besikuriančioms įmonėms – startuoliams taikyti įvairias dirbtinio intelekto strategijas, taip išvengiant žmonėms būdingų klaidų. Todėl įvairiems spekuliantams ir Forex rinkos brokeriams bei įmonėms dirbantiems su valiutų kursų analizę galėčiau pasiūlyti savo sukurtą modelį trumpalaikių valiutų kursų kintamumo analizei bei prognozavimo procesams.

LITERATŪROS SĄRAŠAS

1. A. M. Turing (1950) *Computing Machinery and Intelligence. Mind 49: 433-460.* Prieiga per internetą: https://www.csee.umbc.edu/courses/471/papers/turing.pdf

2. Abounoori, E., Shahrazi, M., Rasekhi, (2012) S. *An investigation of Forex market efficiency based on detrended fluctuation analysis: A case study for Iran*. // Physica a: Statistical Mechanics and its Applications. Vol. 391, No. 11, pp. 3170–3179. doi:10.1016/j.physa.2011.12.045.

3. Adhikari, R. ir Agrawal, R. K. (2013). *An introductory study on time series modeling and forecasting.* Cornell University Library. Prieiga per internetą: https://www.researchgate.net/publication/235219651_An_Introductory_Study_on_Time_series_Modeling_and_Forecasting

4. Akincilar A., Temiz I., Şahin E. (2011) *An Application Of Exchange Rate Forecasting In Turke.* Gazi University Journal of Science GU J Sci 24(4):817-828 Prieiga per internetą: https://www.researchgate.net/publication/220024751_An_Application_Of_Exchange_Rates_Forecasting_In_Turkey

5. Bartkus , Č. *Finansų rinkos*. Panevėžys: Panevėžio kolegija, 2014. ISBN 9789955722816. Prieiga per internetą: http://www.leda.lt/Studijos_kintancioje_verslo_aplinkoje_2015.pdf

6. Butkus, M., Tamašauskas, M. (2016) *Formation of trading strategy based on technical analysis and application in the FOREX market/Technine analize grįstos prekybos strategijos FOREX rinkoje formavimas ir taikymas.* ISSN 2335-8742 Prieiga per internetą: https://www.vdu.lt/cris/bitstream/20.500.12259/31552/1/ISSN2335-8742_2016_V_10.N_1.PG_65-84.pdf

7. Chandar, K. S., Sumathi, M., Sivanandam, N. S. (2015). *Forecasting of Foreign Currency Exchange Rate Using Neural Network.* International Journal of Engineering and Technology (IJET). 0975-4024 Vol 7 No 1 Feb-Mar 2015. Prieiga per internetą: https://pdfs.semanticscholar.org/2dd4/a00151564471d6b7a8f317ed7fc614fa1965.pdf

8. Christopher, O. (2015). *Understanding LSTM Networks – colah's blog*. Prieiga per internetą: http://colah.github.io/posts/2015-08-Understanding-LSTMs/#fn1

9. Eng H. M., Li. Y., Wang, Q., Lee H. T. (2008). *Forecast Forex With ANN Using Fundamental Data.* International Conference on Information Management, Innovation Management and Industrial Engineering, Prieiga per internetą: https://ieeexplore.ieee.org/abstract/document/4737544

10. Erdogan, O., Goksu, A. (2014) *Forecasting Euro and Turkish Lira Exchange Rates with Artificial Neural Networks (ANN)* International Journal of Academic Research in Accounting, Finance and Management Sciences Vol. 4, No.4, October 2014, pp. 307–316. Prieiga per internetą: http://hrmars.com/hrmars_papers/Article_29_Forecasting_Euro_and_Turkish_Lira_Exchange.pdf

11. Galeshchuk S. (2015) *Neural networks performance in exchange rate prediction.* Neurocomputing 172 (2016) 446–452. Prieiga per internetą: https://www.sciencedirect.com/science/article/pii/S0925231215010528

12. Ganegedara, T. (2018) *LSTM in Python: Stock Market Predictions (article) - DataCamp.* Prieiga per internetą: https://www.datacamp.com/community/tutorials/lstm-python-stock-market

13. Girsaitė, I.; Žilinskij, G. (2016). Fundamentaliosios analizės panaudojimo investavimo sprendimų priėmimui valiutų rinkoje galimybių analizė, iš 19-osios Lietuvos jaunųjų mokslininkų konferencijos „Mokslas – Lietuvos ateitis“ teminė konferencija „Verslas XXI amžiuje“, 2016 m. vasario 11 d., Vilnius, Lietuva Vilnius: Technika, 1–9. eISSN 2029–7149. Prieiga per internetą: https://www.researchgate.net/publication/311555234_FUNDAMENTALIOSIOS_ANALIZES_PANAUDOJIMO_INVESTAVIMO_SPRENDIMU_PRIEMIMUI_VALIUTU_RINKOJE_GALIMYBIU_ANALIZE

14. Investing.com - stock market quotes & financial news. Investing.com. Prieiga per internetą: https://www.investing.com/

15. Jagerson, J; Hansen S. W. (2006) *Profiting with Forex.* New York: „McGraw-Hill. 267 p. ISBN-13: 978-0071464659, ISBN-10: 0071464654. Prieiga per internetą: https://epdf.pub/queue/profiting-with-forex-the-most-effective-tools-and-techniques-for-trading-currenc.html

16. Kancerevyčius , G. *Finansai ir investicijos*. Kaunas: Smaltijos leidykla, 2009. ISBN 978-9955707646.

17. Kartašova, J. Ir Venclauskienė D. (2014) *Valuation Of Fundamental Analysis Reliability In Stock Pricing: Theoretical Approach.* Prieiga per internetą: https://www.researchgate.net/publication/300487640_Valuation_Of_Fundamental_Analysis_Reliability_In_Stock_Pricing_Theoretical_Approach

18. Kekytė, I., Stasytytė V. (2017) *Comparative Analysis of Investment Decision Models / Investavimo sprendimų priėmimo modelių lyginamoji analizė.* MLA 2017, 9, 197-208.. Prieiga per internetą: https://vb.vgtu.lt/object/elaba:22921719/

19. Klaus Greff, Rupesh K. Srivastava, Jan Koutn´ık, Bas R. Steunebrink, and Jurgen Schmidhuber. (2017) *Lstm: A search space odyssey*. arXiv, 2017. Prieiga per internetą: https://arxiv.org/pdf/1503.04069.pdf

20. Krenker, A., Bešter, J. ir Kos, A. (2011) *Introduction to the artificial neural networks.* InTech. Prieiga per internetą: http://cdn.intechweb.org/pdfs/14881.pdf

21. Lietuvos Ekonomikos ir inovacijų ministerijos ir ekspertų grupės leidinys (2018) *Lietuvos dirbtinio intelekto strategija. Ateities vizija.* Prieiga per internetą: http://eimin.lrv.lt/uploads/eimin/documents/files/DI_strategija_LT_koreguota.pdf

22. Lison, P. (2012). *An introduction to machine learning.* HiOA. Prieiga per internetą: http://folk.uio.no/plison/pdfs/talks/machinelearning.pdf

23. Lixia Liu, Wenjing Wang (2008) *Exchange Rates Forecasting with Least Squares Support Vector Machine.* International Conference on Computer Science and Software Engineering. Prieiga per internetą: https://ieeexplore.ieee.org/document/4723077

24. Mahmoudvand R., Rodrigues C. P., Yarmohammadi M. (2018) *FORECASTING DAILY EXCHANGE RATES: A COMPARISON BETWEEN SSA AND MSSA.* Prieiga per internetą:https://www.ine.pt/revstat/pdf/PREDICTIONOFBRICSDAILYEXCHANGERATEACOMPARISONBETWEENSSAANDMSSA.pdf

25. Maknickienė N. (2012). *Evaluation of the portfolio performance indicators, using Evolino RNN trading model // Contemporary issues in business, management and education* - Vilnius : Technika, 2012, p. 158-169. - ISSN 2029-7963; ISBN 9786094573231. Prieiga per internetą: http://old.konferencijos.vgtu.lt/cbme.vgtu.lt/public_html/index.php/cbme/cbme_2012/paper/view/90

26. Maknickiene, N. (2014). *Selection of orthogonal investment portfolio using evolino RNN trading model*, in Procedia – Social and Behavioral Sciences. The 2-dn

International Scientific conference „Contemporary Issues in Business, Management and Education 2013". Amsterdam: Elsevier Science Ltd., vol. 110, 1158–1165. Prieiga per internetą: https://www.researchgate.net/publication/263468909_Selection_of_Orthogonal_Investment_Portfolio_Using_Evolino_RNN_Trading_Model

27. Maknickiene, N. (2015). *Paramos sistema investuotojui valiutų rinkoje*. Daktaro disertacija. ISBN 978-609-457-780-2. VGTU leidykla TECHNIKA, 2015. Prieiga per internetą: http://dspace.vgtu.lt/handle/1/1822

28. McCarthy, J.; Lifschitz, V. 1991/1950. Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy. Academic Press.

29. Mirchandani, A. (2013). *Analysis of Macroeconomic Determinants of Exchange Rate Volatility in India*. International Journal of Economics and Financial Issues, 3(1), 172–179. Prieiga per internetą: https://www.researchgate.net/publication/282606310_Analysis_of_Macroeconomic_Determinants_of_Exchange_Rate_volatility_in_India

30. Mondal S., Dakshinakabat P., Swain S. K. (2015) *Comparative Study between Time Series and Neural Network for Exchange Rate Forecasting*. IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661 Prieiga per internetą: http://www.iosrjournals.org/iosr-jce/papers/Vol17-issue2/Version-6/A017260110.pdf

31. N. Srivastava, G. E. Hinton, A. Krizhevsky ir kt., (2014) *Dropout: a simple way to prevent neural networks from overfitting*. Journal of Machine Learning Research.

32. Nanayakkara, S., Chandrasekara V. ir Jayasundara, D.D.M. (2014). *Forecast Exchage rates using Time Series and Neural Network Approach*. European International Journal of Science and Technology, Vol. 3, No. 2, 2014. Prieiga per internetą: https://pdfs.semanticscholar.org/89ec/2080ef2761f405c2af5b9b9913a148c8c0e3.pdf?_ga=2.211445589.1589733522.1577641091-954728550.1567504501

33. Oancea B., Ciuciu S. C. (2014) *TIME SERIES FORECASTING USING NEURAL NETWORKS.* Prieiga per internetą: https://arxiv.org/abs/1401.1333

34. Ovsianikas, V. (2011). *Forex 101: Paprastai ir suprantamai apie valiutų rinką. II atnaujintas leidimas*. – Kaunas: Smaltija

35. Parot A, Michell K, Kristjanpoller WD. (2019) *Using Artificial Neural Networks to forecast Exchange Rate, including VAR-VECM residual analysis and prediction linear*

combination. Intell Sys Acc Fin Mgmt. ;26:3–15. Prieiga per internetą: https://doi.org/10.1002/isaf.1440

36. Patel, P. J., Patel, N. J., & Patel, A. R. (2014). *Factors affecting currency exchange rate, economical formulas and prediction models.* International Journal of Application or Innovation in Engineering & Management (IJAIEM), 3(3), 53-56. Prieiga per internetą: https://www.ijaiem.org/volume3issue3/IJAIEM-2014-03-05-013.pdf

37. Peart, A., (2017). *Homage to John McCarthu, the Father of Artificial Intelligence (AI).* Prieiga per internetą: https://www.artificial-solutions.com/blog/homage-to-john-mccarthy-the-father-of-artificial-intelligence

38. Perveen, Sh. Khan, A. Q. ir Ismail, M. (2012) *Analysis of the factors affecting exchange rate variability in Pakistan.* Academic Research International ISSN-L: 2223-9553, ISSN: 2223-9944 Vol. 2, No. 3, May 2012. Prieiga per internetą: http://www.savap.org.pk/journals/ARInt./Vol.2(3)/2012(2.3-81).pdf

39. Ramasamy, R. ir Abar, K. S. (2015) *Influence of Macroeconomic Variables on Exchange Rates.* Journal of Economics, Business and Management, Vol. 3, No. 2, February 2015. Prieiga per internetą: https://www.researchgate.net/publication/272908793_Influence_of_Macroeconomic_Variables_on_Exchange_Rates

40. Ratnadip Adhikari and R. K. Agrawal. (2013) *An introductory study on time series modeling and forecasting.* Cornell University Library, Prieiga per internetą: https://arxiv.org/ftp/arxiv/papers/1302/1302.6613.pdf

41. Rutkauskas A. V. (2010) *Modelling of the history and predictions of financial market time series using Evolino* // The 6th International Scientific Conference Business and Management 2010: selected papers. – Vilnius: Vilniaus Gedimino technikos universitetas, Vol. 1, p. 170–175. - ISSN 2029-4441. Prieiga per internetą: http://dspace.vgtu.lt/bitstream/1/588/1/170-175_Rutkauskas_Maknickiene_Maknickas.pdf

42. Saini, S. S., Parkhe, O., Khadtare, T.D. (2016). *Analysis of Feedforward and Recurrent Neural Network in Forecasting Foreign Exchange Rate.* Imperial Journal of Interdisciplinary Research (IJIR) Vol-2, Issue-6. Prieiga per internetą: http://www.onlinejournal.in/IJIRV2I6/155.pdf

43. Schmidhuber J., Wierstra D., Gomez F. (2005) *Evolino: Hybrid Neuroevolution / Optimal Linear Search for Sequence Learning. Proceedings of the 19th International*

Joint Conference on Artificial Intelligence, 466–477. Priega per internetą: https://www.ijcai.org/Proceedings/05/Papers/1452.pdf

44. Schmidhuber, J.; Gagliolo, M.; Wierstra, D.; Gomez, F. (2006). *Evolino for recurrent support vector machines.* isbn 2-930307-06-4. ESANN'2006 proceedings – European Symposium on Artificial Neural Networks, 593–598. Prieiga per internetą: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2006-117.pdf
45. Shalzy M. R. ir Shalzy H. E. (1999) *Forecasting currency prices using a genetically evolved neural network architecture* // International Review of financial analysis. – Vol. 8, Nr. 1, p.67. Prieiga per internetą: https://www.sciencedirect.com/science/article/pii/S105752199900006X?via%3Dihub
46. Shmidhuber J., Hochreiter, S. (1997). *Long Short-Term Memory*. Neural Computation, 9(8), 1735-1780. Prieiga per internetą: https://www.mitpressjournals.org/doi/pdf/10.1162/neco.1997.9.8.1735
47. Skorupa P.; Dubovičienė T. (2009) Idiomaticity of English business terms and their equivalents in Lithuanian. Santalka. Filologija. Edukologija. Vilnius: Technika. ISSN 1822-430X. T. 17, nr. 4 Priega per internetą: https://www.researchgate.net/publication/237342767_Idiomaticity_of_English_Business_Terms_and_Their_Equivalents_in_Lithuanian
48. Tlegenova D. (2015) *Forecasting Exchange Rates Using Time Series Analysis: The sample of the currency of Kazakhstan.* Prieiga per internetą: https://arxiv.org/abs/1508.07534
49. Twarowska, K. (2000). *Analysis of factors affectng fluctuations in the exchange rate of Polish Zloty against Euro*, Prieiga per internetą: www.toknowpress.net/ISBN/978-961-6914-09-3/papers/ML14-652.pdf
50. Valiulis, D. (2010). *Analizės metodai pasaulinėje valiutų rinkoje* // Mechanika, medžiagų inžinerija, pramonės inžinerija ir vadyba. T. 2, Nr. 4, p. 69–72. Prieiga per internetą: https://webcache.googleusercontent.com/search?q=cache:C_SRIQb3aCgJ:https://journals.vgtu.lt/index.php/MLA/article/download/10143/8815+&cd=1&hl=lt&ct=clnk&gl=lt
51. Vidyavathi, B., Keerti, K., & Pooja, A. (2016). *A Study on macro economic indicators and their impact on exchange rates*. International Journal of Engineering and

Management Studies, 7(3), 160–169. Prieiga per internetą: http://scienceandnature.org/IJEMS-Vol7(3)-July2016/IJEMS%20Vol7(3)-1.pdf

52. Zada, B. (2010), *An Analysis of Factors Affecting Exchange Rate of Pakistan 1979-2008*. .Hazara university, .Pakistan.

53. Zhang, B. (2018) *Foreign exchange rates forecasting with an EMD-LSTM neural networks model.* Journal of Physics: Conference Series 1053 012005. Priega per internetą: https://iopscience.iop.org/article/10.1088/1742-6596/1053/1/012005/pdf

FORECASTING FOREIGN EXCHANGE RATE USING ARTIFICIAL NEURAL NETWORK

Lukas BUITKUS

Paper of the Master‘s degree

Finance and Banking Master‘s Program

Vilnius University, Faculty of Economics and Business Administration

Department of Finance

Supervisor – Assoc. Prof. Dr. Deimantė Tereseinė

Vilnius, 2020

SUMMARY

61 pages, 12 charts, 6 tables, 15 pictures, 53 references.

The main purpose of this paper is to develop a model to predict future exchange rates using the LSTM algorithm of an artificial neural network, after reviewing previous related studies and evaluating their methods and results.

Without introduction and counclusion the thesis consists of three main parts: theoretical aspects of foreigh markets and artificial neural networks, methodology, practical application of the developed exchange rates prediction model.

In the first chapter of the thesis the most important aspects of the international exchange rates market presented and the the key factors influencing exchange rate changes indentified. It also neural networks main concepts, classification, network training analized and previous related studies and their using methods reviewed and investigated.

In the methodology based on the research of artificial neural networks, a model that can predict the selected exchange rates in the futured created. Algorithm and parametres being used in this thesis was explained in details with their backend mathematical formulas in order to explain the detailed plan for further research.

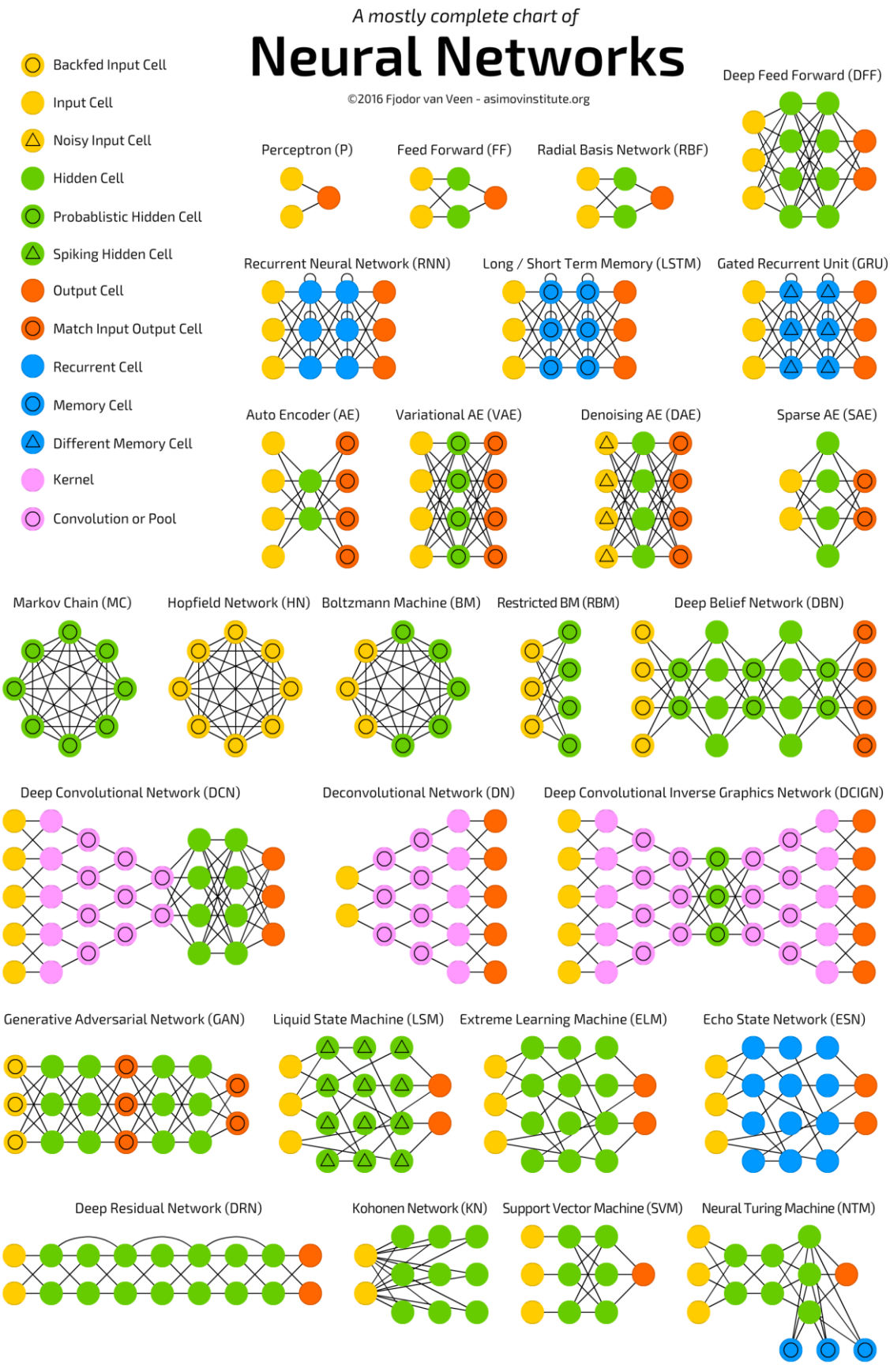
In the empirical part, using the Python programming language and an abundance of libraries, the currency rate prediction model created in methodology is used to perform experimental tests on historical data for selected exchange rates pairs. Using long-term short-term memory (LSTM) neural network algorithm, to perform exchange rate forecasting and statistical results to verify the obtained results programming is used. After checking the results

obtained and the suitability of the model, the exchange rates of the currency pairs for the selected periods are forecasted.

In conclusion the results of the thesis presented, main concepts of the research summarized and recommendation for future improvements provided.

PRIEDAI

1 PRIEDAS. NEURONINIŲ TINKLŲ ARCHITEKTŪROS/TOPOLOGIJOS



In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
```

In [2]:
```
df = pd.read_csv('EURUSD.csv', date_parser = True)
df
```

Out[2]:

	Date	Open	High	Low	Close	Volume
0	2003-12-31	1.25915	1.26035	1.24679	1.25434	5.120000e+04
1	2004-01-01	1.25420	1.26263	1.25198	1.25806	1.340000e+11
2	2004-01-04	1.25816	1.26924	1.25785	1.26727	1.350000e+05
3	2004-01-05	1.26720	1.28083	1.26650	1.27230	1.349585e+05
4	2004-01-06	1.27230	1.27404	1.26221	1.26296	1.350000e+11
...	...	...	...	...	...	...
4123	2019-10-27	1.10990	1.11183	1.10735	1.11118	1.240000e+04
4124	2019-10-28	1.11117	1.11516	1.10801	1.11501	1.362668e+04
4125	2019-10-29	1.11499	1.11755	1.11314	1.11513	1.500000e+04
4126	2019-10-30	1.11511	1.11718	1.11281	1.11641	1.299288e+04
4127	2019-10-31	1.11684	1.11753	1.11247	1.11272	1.070000e+11

4128 rows × 6 columns

In [3]:
```
df_training = df[df['Date']<='2017-06-01'].copy()
df_training
```

Out[3]:

	Date	Open	High	Low	Close	Volume
0	2003-12-31	1.25915	1.26035	1.24679	1.25434	5.120000e+04
1	2004-01-01	1.25420	1.26263	1.25198	1.25806	1.340000e+11
2	2004-01-04	1.25816	1.26924	1.25785	1.26727	1.350000e+05
3	2004-01-05	1.26720	1.28083	1.26650	1.27230	1.349585e+05
4	2004-01-06	1.27230	1.27404	1.26221	1.26296	1.350000e+11
...	...	...	...	...	...	...
3495	2017-05-28	1.11623	1.12054	1.11094	1.11856	1.910000e+04
3496	2017-05-29	1.11854	1.12521	1.11645	1.12431	1.870000e+04
3497	2017-05-30	1.12430	1.12566	1.12021	1.12122	1.676239e+04
3498	2017-05-31	1.12124	1.12850	1.12049	1.12825	1.630000e+11
3499	2017-06-01	1.12703	1.12837	1.12342	1.12541	1.209625e+04

3500 rows × 6 columns

In [4]:
```
df_test = df[df['Date']>'2017-06-01'].copy()
df_test
```

Out[4]:

	Date	Open	High	Low	Close	Volume
3500	2017-06-04	1.12543	1.12842	1.12404	1.12771	1.660000e+11
3501	2017-06-05	1.12771	1.12823	1.12039	1.12554	1.877830e+04
3502	2017-06-06	1.12552	1.12690	1.11947	1.12122	1.940000e+11
3503	2017-06-07	1.12123	1.12366	1.11663	1.11941	2.190000e+11
3504	2017-06-08	1.12061	1.12321	1.11918	1.12029	1.630000e+11
...	...	...	...	...	...	...
4123	2019-10-27	1.10990	1.11183	1.10735	1.11118	1.240000e+04
4124	2019-10-28	1.11117	1.11516	1.10801	1.11501	1.362668e+04
4125	2019-10-29	1.11499	1.11755	1.11314	1.11513	1.500000e+04
4126	2019-10-30	1.11511	1.11718	1.11281	1.11641	1.299288e+04
4127	2019-10-31	1.11684	1.11753	1.11247	1.11272	1.070000e+11

628 rows × 6 columns

In [5]:
```
training_df = df_training.drop(['Date'], axis = 1)
training_df
```

Out[5]:

	Open	High	Low	Close	Volume
0	1.25915	1.26035	1.24679	1.25434	5.120000e+04
1	1.25420	1.26263	1.25198	1.25806	1.340000e+11
2	1.25816	1.26924	1.25785	1.26727	1.350000e+05
3	1.26720	1.28083	1.26650	1.27230	1.349585e+05
4	1.27230	1.27404	1.26221	1.26296	1.350000e+11
...	...	...	...	...	...
3495	1.11623	1.12054	1.11094	1.11856	1.910000e+04
3496	1.11854	1.12521	1.11645	1.12431	1.870000e+04
3497	1.12430	1.12566	1.12021	1.12122	1.676239e+04
3498	1.12124	1.12850	1.12049	1.12825	1.630000e+11
3499	1.12703	1.12837	1.12342	1.12541	1.209625e+04

3500 rows × 5 columns

In [6]:
```
scaler = MinMaxScaler()
training_df = scaler.fit_transform(training_df)
training_df
```

Out[6]:
```
array([[3.93168799e-01, 3.88838484e-01, 3.85155684e-01, 3.84815913e-01,
        8.31168629e-08],
       [3.84335350e-01, 3.92894629e-01, 3.94551050e-01, 3.91458058e-01,
        2.17532468e-01],
       [3.91402109e-01, 4.04653893e-01, 4.05177408e-01, 4.07902725e-01,
        2.19155824e-07],
       ...,
       [1.52524225e-01, 1.49223462e-01, 1.56010138e-01, 1.47127094e-01,
        2.72116551e-08],
       [1.47063547e-01, 1.54275853e-01, 1.56517017e-01, 1.59679320e-01,
        2.64610390e-01],
       [1.57396006e-01, 1.54044582e-01, 1.61821144e-01, 1.54608435e-01,
        1.96367443e-08]])
```

In [7]:
```
X_train = []
Y_train = []
```

In [8]:
```
training_df.shape[0]
```

Out[8]:
```
3500
```

In [9]:
```
for i in range(60, training_df.shape[0]):
    X_train.append(training_df[i-60:i])
    Y_train.append(training_df[i, 0])
```

In [10]:
```
X_train, Y_train = np.array(X_train), np.array(Y_train)
```

In [11]:
```
X_train.shape, Y_train.shape
```

Out[11]:
```
((3440, 60, 5), (3440,))
```

In [12]:
```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
```

In [13]:
```
regressior = Sequential()

regressior.add(LSTM(units = 60, activation = 'relu', return_sequences = Tru
e, input_shape = (X_train.shape[1], 5)))
regressior.add(Dropout(0.2))

regressior.add(LSTM(units = 60, activation = 'relu', return_sequences = Tru
e))
regressior.add(Dropout(0.2))

regressior.add(LSTM(units = 80, activation = 'relu', return_sequences = Tru
e))
regressior.add(Dropout(0.2))

regressior.add(LSTM(units = 120, activation = 'relu'))
regressior.add(Dropout(0.2))

regressior.add(Dense(units = 1))
```

In [14]:
```
regressior.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 60, 60)            15840
_________________________________________________________________
dropout (Dropout)            (None, 60, 60)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 60, 60)            29040
_________________________________________________________________
dropout_1 (Dropout)          (None, 60, 60)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 60, 80)            45120
_________________________________________________________________
dropout_2 (Dropout)          (None, 60, 80)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 120)               96480
_________________________________________________________________
dropout_3 (Dropout)          (None, 120)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 121
=================================================================
```

```
Total params: 186,601
Trainable params: 186,601
Non-trainable params: 0
_________________________________________________________________
```

In [15]:
```
regressior.compile(optimizer='adam', loss = 'mean_squared_error')
```

In [16]:
```
regressior.fit(X_train, Y_train, epochs=50, batch_size=128)
```

```
Train on 3440 samples
Epoch 1/50
3440/3440 [==============================] - 43s 12ms/sample - loss: 0.0666
Epoch 2/50
3440/3440 [==============================] - 42s 12ms/sample - loss: 0.0085
Epoch 3/50
3440/3440 [==============================] - 43s 12ms/sample - loss: 0.0051
Epoch 4/50
3440/3440 [==============================] - 44s 13ms/sample - loss: 0.0045
Epoch 5/50
3440/3440 [==============================] - 45s 13ms/sample - loss: 0.0045
Epoch 6/50
3440/3440 [==============================] - 45s 13ms/sample - loss: 0.0042
Epoch 7/50
3440/3440 [==============================] - 45s 13ms/sample - loss: 0.0041
Epoch 8/50
3440/3440 [==============================] - 46s 13ms/sample - loss: 0.0036
Epoch 9/50
3440/3440 [==============================] - 46s 13ms/sample - loss: 0.0035
Epoch 10/50
3440/3440 [==============================] - 45s 13ms/sample - loss: 0.0034
Epoch 11/50
3440/3440 [==============================] - 46s 13ms/sample - loss: 0.0032
Epoch 12/50
3440/3440 [==============================] - 46s 13ms/sample - loss: 0.0033
Epoch 13/50
3440/3440 [==============================] - 46s 13ms/sample - loss: 0.0033
Epoch 14/50
3440/3440 [==============================] - 46s 14ms/sample - loss: 0.0031
Epoch 15/50
3440/3440 [==============================] - 46s 13ms/sample - loss: 0.0030
Epoch 16/50
3440/3440 [==============================] - 46s 13ms/sample - loss: 0.0030
Epoch 17/50
3440/3440 [==============================] - 46s 14ms/sample - loss: 0.0028
Epoch 18/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0029
Epoch 19/50
3440/3440 [==============================] - 46s 13ms/sample - loss: 0.0027
Epoch 20/50
3440/3440 [==============================] - 45s 13ms/sample - loss: 0.0028
Epoch 21/50
3440/3440 [==============================] - 46s 13ms/sample - loss: 0.0026
Epoch 22/50
3440/3440 [==============================] - 46s 13ms/sample - loss: 0.0026
Epoch 23/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0028
Epoch 24/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0027
Epoch 25/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0028
```

```
Epoch 26/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0024
Epoch 27/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0025
Epoch 28/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0024
Epoch 29/50
3440/3440 [==============================] - 48s 14ms/sample - loss: 0.0023
Epoch 30/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0022
Epoch 31/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0023
Epoch 32/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0021
Epoch 33/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0023
Epoch 34/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0022
Epoch 35/50
3440/3440 [==============================] - 46s 13ms/sample - loss: 0.0021
Epoch 36/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0020
Epoch 37/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0021
Epoch 38/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0020
Epoch 39/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0020
Epoch 40/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0019
Epoch 41/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0018
Epoch 42/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0019
Epoch 43/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0019
Epoch 44/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0018
Epoch 45/50
3440/3440 [==============================] - 42s 12ms/sample - loss: 0.0019
Epoch 46/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0018
Epoch 47/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0018
Epoch 48/50
3440/3440 [==============================] - 47s 14ms/sample - loss: 0.0017
Epoch 49/50
3440/3440 [==============================] - 48s 14ms/sample - loss: 0.0017
Epoch 50/50
3440/3440 [==============================] - 48s 14ms/sample - loss: 0.0017
```

Out[16]: `<tensorflow.python.keras.callbacks.History at 0x2d664e864e0>`

In [17]:
```
df_test.head()
```

Out[18]:

	Date	Open	High	Low	Close	Volume
3440	2017-03-12	1.06531	1.06624	1.06003	1.06035	1.820000e+11
3441	2017-03-13	1.06038	1.07399	1.06032	1.07339	2.060000e+04
3442	2017-03-14	1.07330	1.07702	1.07057	1.07652	2.390000e+04
3443	2017-03-15	1.07652	1.07823	1.07274	1.07382	1.960000e+04
3444	2017-03-16	1.07318	1.07772	1.07251	1.07393	1.677617e+04
3445	2017-03-19	1.07390	1.08191	1.07190	1.08097	2.240000e+11
3446	2017-03-20	1.08092	1.08245	1.07758	1.07963	2.220000e+11
3447	2017-03-21	1.07962	1.08048	1.07680	1.07830	2.060000e+11
3448	2017-03-22	1.07830	1.08179	1.07602	1.07969	1.970000e+11
3449	2017-03-23	1.08350	1.09058	1.08264	1.08638	1.920000e+04
3450	2017-03-26	1.08640	1.08725	1.07988	1.08125	1.660000e+03
3451	2017-03-27	1.08114	1.08264	1.07398	1.07655	1.780000e+04
3452	2017-03-28	1.07654	1.07696	1.06718	1.06744	1.630000e+11
3453	2017-03-29	1.06743	1.07018	1.06508	1.06528	1.910000e+11
3454	2017-03-30	1.06594	1.06809	1.06423	1.06691	1.737102e+00
3455	2017-04-02	1.06688	1.06770	1.06355	1.06729	1.630000e+11
3456	2017-04-03	1.06729	1.06889	1.06349	1.06627	1.950000e+04
3457	2017-04-04	1.06625	1.06840	1.06289	1.06438	1.830000e+11
3458	2017-04-05	1.06438	1.06659	1.05806	1.05889	2.046163e+04
3459	2017-04-06	1.05866	1.06064	1.05698	1.05957	1.464474e+04
3460	2017-04-09	1.05954	1.06299	1.05785	1.06042	1.580000e+04
3461	2017-04-10	1.06042	1.06751	1.05890	1.06651	1.820000e+11
3462	2017-04-11	1.06651	1.06775	1.06094	1.06131	1.660000e+11
3463	2017-04-12	1.06131	1.06294	1.06060	1.06105	6.160000e+11
3464	2017-04-13	1.06129	1.06703	1.06028	1.06424	1.080000e+11
3465	2017-04-16	1.06418	1.07359	1.06372	1.07296	1.890000e+04
3466	2017-04-17	1.07296	1.07367	1.06998	1.07101	1.570000e+04
3467	2017-04-18	1.07092	1.07775	1.07086	1.07165	1.631009e+04
3468	2017-04-19	1.07164	1.07379	1.06822	1.07237	1.540000e+11
3469	2017-04-20	1.09164	1.09164	1.08207	1.08672	2.180000e+11
3470	2017-04-23	1.08672	1.09499	1.08512	1.09258	1.790000e+11
3471	2017-04-24	1.09247	1.09507	1.08557	1.09037	2.020000e+11
3472	2017-04-25	1.09037	1.09327	1.08517	1.08718	1.795251e+04
3473	2017-04-26	1.08716	1.09474	1.08571	1.08961	1.710000e+11
3474	2017-04-27	1.09099	1.09239	1.08840	1.08982	8.460000e+10
3475	2017-04-30	1.08985	1.09330	1.08881	1.09293	1.348495e+04
3476	2017-05-01	1.09294	1.09367	1.08827	1.08856	1.530000e+04
3477	2017-05-02	1.08845	1.09870	1.08749	1.09845	1.860000e+11
3478	2017-05-03	1.09843	1.09995	1.09494	1.09950	1.670000e+11
3479	2017-05-04	1.10188	1.10216	1.09164	1.09235	1.859446e+00

3480	2017-05-07	1.09237	1.09333	1.08633	1.08730	1.750000e+11
3481	2017-05-08	1.08729	1.08980	1.08532	1.08673	1.595730e+04
3482	2017-05-09	1.08672	1.08929	1.08390	1.08612	1.430000e+04
3483	2017-05-10	1.08612	1.09342	1.08556	1.09310	1.530000e+11
3484	2017-05-11	1.09288	1.09893	1.09225	1.09749	1.419059e+04
3485	2017-05-14	1.09748	1.10971	1.09731	1.10826	1.667582e+04
3486	2017-05-15	1.10824	1.11620	1.10794	1.11584	2.137709e+04
3487	2017-05-16	1.11584	1.11719	1.10757	1.11028	2.590000e+11
3488	2017-05-17	1.11026	1.12117	1.10968	1.12065	2.130000e+11
3489	2017-05-18	1.11917	1.12634	1.11614	1.12372	2.043141e+04
3490	2017-05-21	1.12372	1.12682	1.11750	1.11826	2.400000e+11
3491	2017-05-22	1.11822	1.12199	1.11685	1.12185	2.290000e+11
3492	2017-05-23	1.12182	1.12503	1.11937	1.12102	1.960000e+11
3493	2017-05-24	1.12101	1.12346	1.11605	1.11799	1.920000e+11
3494	2017-05-25	1.11701	1.11898	1.11618	1.11624	1.110000e+11
3495	2017-05-28	1.11623	1.12054	1.11094	1.11856	1.910000e+04
3496	2017-05-29	1.11854	1.12521	1.11645	1.12431	1.870000e+04
3497	2017-05-30	1.12430	1.12566	1.12021	1.12122	1.676239e+04
3498	2017-05-31	1.12124	1.12850	1.12049	1.12825	1.630000e+11
3499	2017-06-01	1.12703	1.12837	1.12342	1.12541	1.209625e+04

In [19]:
```
past_60_days = df_training.tail(60)
```

In [20]:
```
df = past_60_days.append(df_test, ignore_index = True)
df = df.drop(['Date'], axis = 1)
df.head()
```

Out[20]:

	Open	High	Low	Close	Volume
0	1.06531	1.06624	1.06003	1.06035	1.820000e+11
1	1.06038	1.07399	1.06032	1.07339	2.060000e+04
2	1.07330	1.07702	1.07057	1.07652	2.390000e+04
3	1.07652	1.07823	1.07274	1.07382	1.960000e+04
4	1.07318	1.07772	1.07251	1.07393	1.677617e+04

In [21]:
```
inputs = scaler.transform(df)
inputs
```

Out[21]:
```
array([[4.72544926e-02, 4.35146146e-02, 4.70673425e-02, 3.84423098e-02,
        2.95454545e-01],
       [3.84567339e-02, 5.73019516e-02, 4.75923244e-02, 6.17255294e-02,
        3.34415382e-08],
       [6.15129290e-02, 6.26923556e-02, 6.61477190e-02, 6.73142163e-02,
        3.87986810e-08],
       ...,
       [1.35910202e-01, 1.34795681e-01, 1.43211441e-01, 1.36253259e-01,
        2.43506291e-08],
       [1.36124346e-01, 1.34137446e-01, 1.42614048e-01, 1.38538728e-01,
        2.10923186e-08],
       [1.39211592e-01, 1.34760100e-01, 1.41998552e-01, 1.31950148e-01,
        1.73701299e-01]])
```

In [22]:
```
X_test = []
Y_test = []

for i in range(60, inputs.shape[0]):
    X_test.append(inputs[i-60:i])
    Y_test.append(inputs[i, 0])
```

In [23]:
```
X_test, Y_test = np.array(X_test), np.array(Y_test)
X_test.shape, Y_test.shape
```

Out[23]:
```
((628, 60, 5), (628,))
```

In [24]:
```
y_pred = regressior.predict(X_test)
```

In [25]:
```
scaler.scale_
```

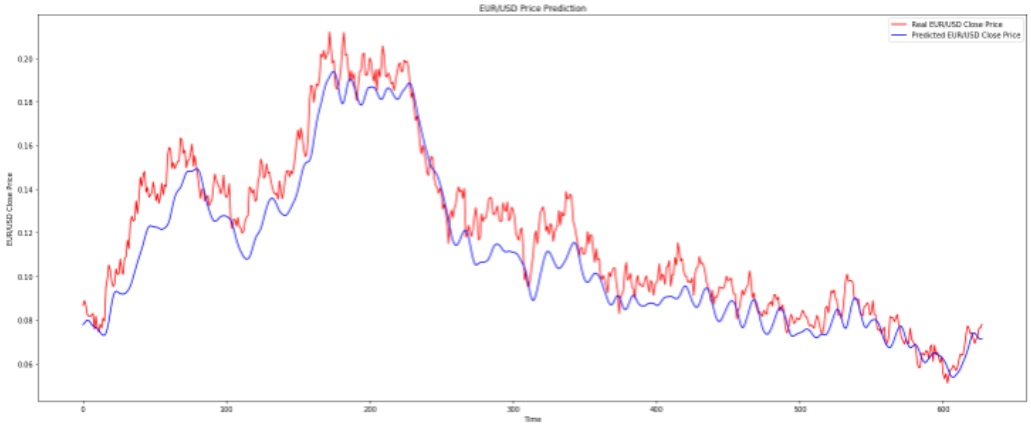
Out[25]:
```
array([1.78453522e+00, 1.77901123e+00, 1.81028240e+00, 1.78552298e+00,
       1.62337662e-12])
```

In [26]:
```
scale = 1/1.78453522e+00
scale
```

Out[26]:
```
0.5603699993099603
```

In [27]:
```
y_pred = y_pred*scale
Y_test = Y_test*scale
```

In [29]:
```
plt.figure(figsize=(25,10))
plt.plot(Y_test, color ='red', label = 'Real EUR/USD Close Price')
plt.plot(y_pred, color ='blue', label = 'Predicted EUR/USD Close Price')
plt.title('EUR/USD Price Prediction')
plt.xlabel('Time')
plt.ylabel('EUR/USD Close Price')
plt.legend()
plt.show()
```



In [30]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [31]:
```
mean_squared_error(Y_test, y_pred), mean_absolute_error(Y_test, y_pred)
```

Out[31]:
```
(0.0001776300037631571, 0.010803778023131152)
```

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
```

In [2]:
```
df = pd.read_csv('EURGBP.csv', date_parser = True)
df
```

Out[2]:

	Date	Open	High	Low	Close	Volume
0	2003-12-31	0.70531	0.70720	0.69826	0.70326	9.271690e+08
1	2004-01-01	0.70332	0.70606	0.70122	0.70186	2.480000e+16
2	2004-01-04	0.70194	0.70582	0.70017	0.70152	2.410000e+16
3	2004-01-05	0.70160	0.70213	0.69808	0.69846	2.370000e+16
4	2004-01-06	0.69846	0.69969	0.69432	0.69463	2.460000e+16
...	...	...	...	...	...	...
4123	2019-10-27	0.86494	0.86525	0.86160	0.86347	1.145169e+09
4124	2019-10-28	0.86350	0.86511	0.86081	0.86391	1.280000e+16
4125	2019-10-29	0.86391	0.86460	0.86120	0.86442	1.300570e+09
4126	2019-10-30	0.86442	0.86474	0.86008	0.86190	1.496236e+09
4127	2019-10-31	0.86192	0.86288	0.86154	0.86225	8.190000e+15

4128 rows × 6 columns

In [3]:
```
df_training = df[df['Date']<='2017-06-01'].copy()
df_training
```

Out[3]:

	Date	Open	High	Low	Close	Volume
0	2003-12-31	0.70531	0.70720	0.69826	0.70326	9.271690e+08
1	2004-01-01	0.70332	0.70606	0.70122	0.70186	2.480000e+16
2	2004-01-04	0.70194	0.70582	0.70017	0.70152	2.410000e+16
3	2004-01-05	0.70160	0.70213	0.69808	0.69846	2.370000e+16
4	2004-01-06	0.69846	0.69969	0.69432	0.69463	2.460000e+16
...	...	...	...	...	...	...
3495	2017-05-28	0.87290	0.87290	0.86937	0.86963	8.700000e+15
3496	2017-05-29	0.86963	0.87069	0.86561	0.87001	1.276549e+09
3497	2017-05-30	0.87011	0.87488	0.86983	0.87238	1.351909e+09
3498	2017-05-31	0.87238	0.87554	0.86841	0.87062	1.219060e+09
3499	2017-06-01	0.87061	0.87681	0.86957	0.87547	1.171520e+09

3500 rows × 6 columns

In [4]:
```
df_test = df[df['Date']>'2017-06-01'].copy()
df_test
```

Out[4]:

	Date	Open	High	Low	Close	Volume
3500	2017-06-04	0.87668	0.87687	0.86949	0.87220	9.680000e+15
3501	2017-06-05	0.87228	0.87559	0.86925	0.87379	1.192342e+09
3502	2017-06-06	0.87342	0.87410	0.86773	0.86869	1.280000e+16
3503	2017-06-07	0.86869	0.87001	0.86530	0.86603	1.290000e+16
3504	2017-06-08	0.86582	0.88599	0.86582	0.87879	2.122814e+05
...	...	...	...	...	...	...
4123	2019-10-27	0.86494	0.86525	0.86160	0.86347	1.145169e+09
4124	2019-10-28	0.86350	0.86511	0.86081	0.86391	1.280000e+16
4125	2019-10-29	0.86391	0.86460	0.86120	0.86442	1.300570e+09
4126	2019-10-30	0.86442	0.86474	0.86008	0.86190	1.496236e+09
4127	2019-10-31	0.86192	0.86288	0.86154	0.86225	8.190000e+15

628 rows × 6 columns

In [5]:
```
training_df = df_training.drop(['Date'], axis = 1)
training_df
```

Out[5]:

	Open	High	Low	Close	Volume
0	0.70531	0.70720	0.69826	0.70326	9.271690e+08
1	0.70332	0.70606	0.70122	0.70186	2.480000e+16
2	0.70194	0.70582	0.70017	0.70152	2.410000e+16
3	0.70160	0.70213	0.69808	0.69846	2.370000e+16
4	0.69846	0.69969	0.69432	0.69463	2.460000e+16
...	...	...	...	...	...
3495	0.87290	0.87290	0.86937	0.86963	8.700000e+15
3496	0.86963	0.87069	0.86561	0.87001	1.276549e+09
3497	0.87011	0.87488	0.86983	0.87238	1.351909e+09
3498	0.87238	0.87554	0.86841	0.87062	1.219060e+09
3499	0.87061	0.87681	0.86957	0.87547	1.171520e+09

3500 rows × 5 columns

In [6]:
```
scaler = MinMaxScaler()
training_df = scaler.fit_transform(training_df)
training_df
```

In [7]:
```
X_train = []
Y_train = []
```

In [8]:
```
training_df.shape[0]
```

Out[8]:
```
3500
```

In [9]:
```
for i in range(60, training_df.shape[0]):
    X_train.append(training_df[i-60:i])
    Y_train.append(training_df[i, 0])
```

In [10]:
```
X_train, Y_train = np.array(X_train), np.array(Y_train)
```

In [11]:
```
X_train.shape, Y_train.shape
```

Out[11]:
```
((3440, 60, 5), (3440,))
```

In [12]:
```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
```

In [13]:
```
regressior = Sequential()

regressior.add(LSTM(units = 60, activation = 'relu', return_sequences = Tru
e, input_shape = (X_train.shape[1], 5)))
regressior.add(Dropout(0.2))

regressior.add(LSTM(units = 60, activation = 'relu', return_sequences = Tru
e))
regressior.add(Dropout(0.2))

regressior.add(LSTM(units = 80, activation = 'relu', return_sequences = Tru
e))
regressior.add(Dropout(0.2))

regressior.add(LSTM(units = 120, activation = 'relu'))
regressior.add(Dropout(0.2))

regressior.add(Dense(units = 1))
```

In [14]:
```
regressior.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 60, 60)            15840
_________________________________________________________________
dropout (Dropout)            (None, 60, 60)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 60, 60)            29040
_________________________________________________________________
dropout_1 (Dropout)          (None, 60, 60)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 60, 80)            45120
_________________________________________________________________
dropout_2 (Dropout)          (None, 60, 80)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 120)               96480
_________________________________________________________________
dropout_3 (Dropout)          (None, 120)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 121
=================================================================
```

```
Total params: 186,601
Trainable params: 186,601
Non-trainable params: 0
_________________________________________________________________
```

In [15]:
```
regressior.compile(optimizer='adam', loss='mse')
```

In [16]:
```
regressior.fit(X_train, Y_train, epochs=50, batch_size=32)
```

```
Train on 3440 samples
Epoch 1/50
3440/3440 [==============================] - 31s 9ms/sample - loss: 0.0213
Epoch 2/50
3440/3440 [==============================] - 23s 7ms/sample - loss: 0.0040
Epoch 3/50
3440/3440 [==============================] - 23s 7ms/sample - loss: 0.0034
Epoch 4/50
3440/3440 [==============================] - 23s 7ms/sample - loss: 0.0034
Epoch 5/50
3440/3440 [==============================] - 23s 7ms/sample - loss: 0.0033
Epoch 6/50
3440/3440 [==============================] - 23s 7ms/sample - loss: 0.0027
Epoch 7/50
3440/3440 [==============================] - 24s 7ms/sample - loss: 0.0032
Epoch 8/50
3440/3440 [==============================] - 25s 7ms/sample - loss: 0.0025
Epoch 9/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 0.0025
Epoch 10/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 0.0022
Epoch 11/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 0.0021
Epoch 12/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0023
Epoch 13/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0018
Epoch 14/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0019
Epoch 15/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0018
Epoch 16/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0018
Epoch 17/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0017
Epoch 18/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0015
Epoch 19/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.00150
s - loss: 0.0
Epoch 20/50
3440/3440 [==============================] - 19s 5ms/sample - loss: 0.0018
Epoch 21/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 0.0014
```

```
Epoch 22/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 0.0014
Epoch 23/50
3440/3440 [==============================] - 19s 5ms/sample - loss: 0.0013
Epoch 24/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 0.0014
Epoch 25/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0012
Epoch 26/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0013
Epoch 27/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 0.0011
Epoch 28/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.00121
s - lo
Epoch 29/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 0.0012
Epoch 30/50
3440/3440 [==============================] - 19s 6ms/sample - loss: 0.0012
Epoch 31/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0012
Epoch 32/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0011
Epoch 33/50
3440/3440 [==============================] - 19s 6ms/sample - loss: 0.0010T
A: 0s - loss: 0.001
Epoch 34/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0010
Epoch 35/50
3440/3440 [==============================] - 19s 5ms/sample - loss: 0.0010
Epoch 36/50
3440/3440 [==============================] - 19s 5ms/sample - loss: 0.0010
Epoch 37/50
3440/3440 [==============================] - 19s 5ms/sample - loss: 9.5123e
-04
Epoch 38/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 0.0011
 Epoch 39/50
 3440/3440 [==============================] - 20s 6ms/sample - loss: 9.5404e
 -04
 Epoch 40/50
 3440/3440 [==============================] - 21s 6ms/sample - loss: 8.8457e
 -04
 Epoch 41/50
 3440/3440 [==============================] - 20s 6ms/sample - loss: 9.2022e
 -04
 Epoch 42/50
 3440/3440 [==============================] - 20s 6ms/sample - loss: 8.7617e
 -04
 Epoch 43/50
 3440/3440 [==============================] - 19s 6ms/sample - loss: 9.0350e
 -04
 Epoch 44/50
 3440/3440 [==============================] - 20s 6ms/sample - loss: 9.2110e
 -04
 Epoch 45/50
 3440/3440 [==============================] - 20s 6ms/sample - loss: 9.6360e
 -04
 Epoch 46/50
 3440/3440 [==============================] - 20s 6ms/sample - loss: 9.8539e
 -04
```

```
Epoch 47/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 8.8790e
-04
Epoch 48/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 8.8512e
-04
Epoch 49/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 8.4128e
-04
Epoch 50/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 8.3576e
-040s - loss: 8.3735e-
```

Out[16]: `<tensorflow.python.keras.callbacks.History at 0x1983215c898>`

In [17]:
```
df_test.head()
```

Out[17]:

	Date	Open	High	Low	Close	Volume
3500	2017-06-04	0.87668	0.87687	0.86949	0.87220	9.680000e+15
3501	2017-06-05	0.87228	0.87559	0.86925	0.87379	1.192342e+09
3502	2017-06-06	0.87342	0.87410	0.86773	0.86869	1.280000e+16
3503	2017-06-07	0.86869	0.87001	0.86530	0.86603	1.290000e+16
3504	2017-06-08	0.86582	0.88599	0.86582	0.87879	2.122814e+05

In [18]:
```
df_training.tail(60)
```

Out[18]:

	Date	Open	High	Low	Close	Volume
3440	2017-03-12	0.87846	0.87879	0.87095	0.87192	1.430000e+16
3441	2017-03-13	0.87192	0.87865	0.87165	0.87271	1.341675e+09
3442	2017-03-14	0.87276	0.87376	0.86669	0.87339	1.530000e+16
3443	2017-03-15	0.87339	0.87598	0.86694	0.87108	1.650000e+16
3444	2017-03-16	0.87110	0.87375	0.86611	0.86641	1.427641e+08
3445	2017-03-19	0.86724	0.87085	0.86598	0.86905	1.190000e+16
3446	2017-03-20	0.86904	0.87269	0.86503	0.86636	1.500000e+16
3447	2017-03-21	0.86645	0.87008	0.86439	0.86493	1.350000e+16
3448	2017-03-22	0.86507	0.86562	0.86055	0.86132	1.460000e+16
3449	2017-03-23	0.86140	0.86621	0.86107	0.86596	1.194913e+09
3450	2017-03-26	0.86770	0.86770	0.86250	0.86510	1.346267e+09
3451	2017-03-27	0.86513	0.86889	0.86182	0.86860	1.270000e+16
3452	2017-03-28	0.86858	0.87361	0.86246	0.86601	1.448474e+09
3453	2017-03-29	0.86601	0.86630	0.85573	0.85616	1.250000e+16
3454	2017-03-30	0.85616	0.85972	0.84873	0.84889	1.425153e+09
3455	2017-04-02	0.85023	0.85579	0.84957	0.85464	1.200000e+16
3456	2017-04-03	0.85462	0.85882	0.85376	0.85804	1.158776e+09
3457	2017-04-04	0.85821	0.85917	0.85321	0.85434	1.406375e+09
3458	2017-04-05	0.85434	0.85678	0.85114	0.85375	1.246094e+09
3459	2017-04-06	0.85378	0.85803	0.85316	0.85635	1.538474e+09
3460	2017-04-09	0.85523	0.85545	0.85244	0.85384	1.110000e+16
3461	2017-04-10	0.85384	0.85490	0.84885	0.84898	1.270434e+08

3462	2017-04-11	0.84898	0.85123	0.84776	0.85080	1.386836e+09
3463	2017-04-12	0.85083	0.85124	0.84754	0.84904	1.292600e+09
3464	2017-04-13	0.84918	0.84958	0.84692	0.84764	1.100000e+16
3465	2017-04-16	0.84690	0.84904	0.84614	0.84715	9.046420e+08
3466	2017-04-17	0.84703	0.85128	0.83139	0.83573	1.683250e+09
3467	2017-04-18	0.83573	0.83895	0.83377	0.83857	1.446682e+09
3468	2017-04-19	0.83860	0.84143	0.83598	0.83656	1.313382e+09
3469	2017-04-20	0.83652	0.83876	0.83522	0.83793	1.300000e+16
3470	2017-04-23	0.85071	0.85096	0.84515	0.84950	1.590000e+16
3471	2017-04-24	0.84949	0.85310	0.84781	0.85089	1.440000e+16
3472	2017-04-25	0.85090	0.85315	0.84589	0.84879	1.530000e+16
3473	2017-04-26	0.84874	0.84962	0.84160	0.84274	1.500000e+16
3474	2017-04-27	0.84280	0.84626	0.84054	0.84163	1.503339e+09
3475	2017-04-30	0.84383	0.84600	0.84205	0.84584	8.690733e+08
3476	2017-05-01	0.84588	0.84850	0.84230	0.84488	1.190485e+09
3477	2017-05-02	0.84500	0.84771	0.84381	0.84607	1.273649e+08
3478	2017-05-03	0.84638	0.85010	0.84565	0.85002	1.380000e+16
3479	2017-05-04	0.85003	0.85094	0.84662	0.84755	1.351442e+09
3480	2017-05-07	0.84895	0.84908	0.84343	0.84442	1.362778e+09
3481	2017-05-08	0.84428	0.84475	0.84016	0.84092	1.390000e+16
3482	2017-05-09	0.84099	0.84174	0.83841	0.84003	1.310000e+16
3483	2017-05-10	0.84003	0.84525	0.83964	0.84283	1.320000e+16
3484	2017-05-11	0.84282	0.84891	0.84261	0.84817	1.290000e+16
3485	2017-05-14	0.84890	0.85149	0.84584	0.85127	1.187830e+09
3486	2017-05-15	0.85126	0.85958	0.85042	0.85829	1.460000e+16
3487	2017-05-16	0.85809	0.86157	0.85605	0.86044	1.470000e+16
3488	2017-05-17	0.86048	0.86136	0.85244	0.85829	1.770000e+16
3489	2017-05-18	0.85846	0.86032	0.85692	0.86002	1.290000e+16
3490	2017-05-21	0.86185	0.86516	0.86003	0.86450	1.280000e+16
3491	2017-05-22	0.86448	0.86755	0.86193	0.86283	1.640000e+16
3492	2017-05-23	0.86285	0.86565	0.86031	0.86480	1.570000e+16
3493	2017-05-24	0.86467	0.86678	0.86406	0.86634	1.308689e+09
3494	2017-05-25	0.86628	0.87511	0.86601	0.87329	1.370000e+16
3495	2017-05-28	0.87290	0.87290	0.86937	0.86963	8.700000e+15
3496	2017-05-29	0.86963	0.87069	0.86561	0.87001	1.276549e+09
3497	2017-05-30	0.87011	0.87488	0.86983	0.87238	1.351909e+09
3498	2017-05-31	0.87238	0.87554	0.86841	0.87062	1.219060e+09
3499	2017-06-01	0.87061	0.87681	0.86957	0.87547	1.171520e+09

In [19]:
```
past_60_days = df_training.tail(60)
```

In [20]:
```
df = past_60_days.append(df_test, ignore_index = True)
df = df.drop(['Date'], axis = 1)
df.head()
```

Out[20]:

	Open	High	Low	Close	Volume
0	0.87846	0.87879	0.87095	0.87192	1.430000e+16
1	0.87192	0.87865	0.87165	0.87271	1.341675e+09
2	0.87276	0.87376	0.86669	0.87339	1.530000e+16
3	0.87339	0.87598	0.86694	0.87108	1.650000e+16
4	0.87110	0.87375	0.86611	0.86641	1.427641e+08

In [21]:
```
inputs = scaler.transform(df)
inputs
```

Out[21]:
```
array([[6.96185159e-01, 6.85535758e-01, 6.96500673e-01, 6.75826575e-01,
        2.20338983e-01],
       [6.75785271e-01, 6.85102697e-01, 6.98743831e-01, 6.78290705e-01,
        2.06729569e-08],
       [6.78405440e-01, 6.69976491e-01, 6.82849452e-01, 6.80411728e-01,
        2.35747304e-01],
       ...,
       [6.50800087e-01, 6.41641920e-01, 6.65256681e-01, 6.52432938e-01,
        2.00396038e-08],
       [6.52390904e-01, 6.42074981e-01, 6.61667628e-01, 6.44572676e-01,
        2.30544856e-08],
       [6.44592782e-01, 6.36321455e-01, 6.66346215e-01, 6.45664379e-01,
        1.26194145e-01]])
```

In [22]:
```
X_test = []
Y_test = []

for i in range(60, inputs.shape[0]):
    X_test.append(inputs[i-60:i])
    Y_test.append(inputs[i, 0])
```

In [23]:
```
X_test, Y_test = np.array(X_test), np.array(Y_test)
X_test.shape, Y_test.shape
```

Out[23]:
```
((628, 60, 5), (628,))
```

In [24]:
```
y_pred = regressior.predict(X_test)
```

In [25]:
```
scaler.scale_
```

Out[25]:
```
array([3.11924888e+00, 3.09329374e+00, 3.20451195e+00, 3.11915159e+00,
       1.54083205e-17])
```

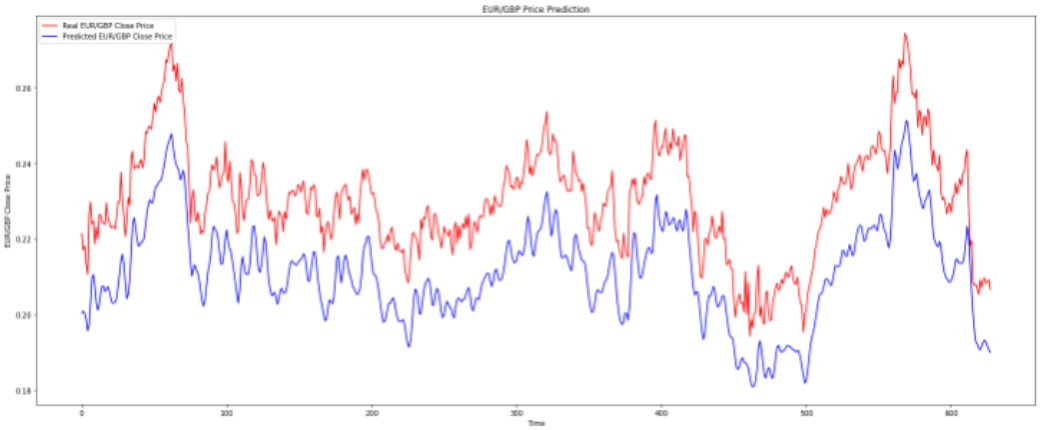
In [26]:
```
scale = 1/3.11924888e+00
scale
```

Out[26]:
```
0.3205900005003769
```

In [27]:
```
y_pred = y_pred*scale
Y_test = Y_test*scale
```

In [29]:
```
### Visualization
```

In [30]:
```
plt.figure(figsize=(25,10))
plt.plot(Y_test, color ='red', label = 'Real EUR/GBP Close Price')
plt.plot(y_pred, color ='blue', label = 'Predicted EUR/GBP Close Price')
plt.title('EUR/GBP Price Prediction')
plt.xlabel('Time')
plt.ylabel('EUR/GBP Close Price')
plt.legend()
plt.show()
```



In [31]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [32]:
```
mean_squared_error(Y_test, y_pred), mean_absolute_error(Y_test, y_pred)
```

Out[32]:
```
(0.00037241124321477775, 0.01899343134172093)
```

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
```

In [2]:
```
df = pd.read_csv('EURJPY.csv', date_parser = True)
df
```

Out[2]:

	Date	Open	High	Low	Close	Volume
0	2003-12-31	134.970	135.351	133.983	134.793	1.447987e+06
1	2004-01-01	134.717	135.094	134.098	134.685	2.730000e+06
2	2004-01-04	134.642	135.717	134.278	134.582	2.750000e+06
3	2004-01-05	134.589	135.980	134.440	135.094	2.790000e+06
4	2004-01-06	135.100	135.366	134.019	134.086	2.789103e+06
...	...	...	...	...	...	...
4123	2019-10-27	120.908	121.063	120.575	120.990	2.030000e+13
4124	2019-10-28	120.984	121.397	120.875	121.353	2.330000e+13
4125	2019-10-29	121.342	121.468	120.284	120.470	5.390000e+12
4126	2019-10-30	120.461	120.941	120.360	120.766	2.100000e+13
4127	2019-10-31	120.873	121.106	120.771	120.807	1.770000e+13

4128 rows × 6 columns

In [3]:
```
df_training = df[df['Date']<='2017-06-01'].copy()
df_training
```

Out[3]:

	Date	Open	High	Low	Close	Volume
0	2003-12-31	134.970	135.351	133.983	134.793	1.447987e+06
1	2004-01-01	134.717	135.094	134.098	134.685	2.730000e+06
2	2004-01-04	134.642	135.717	134.278	134.582	2.750000e+06
3	2004-01-05	134.589	135.980	134.440	135.094	2.790000e+06
4	2004-01-06	135.100	135.366	134.019	134.086	2.789103e+06
...	...	...	...	...	...	...
3495	2017-05-28	124.190	124.393	123.156	123.971	1.890000e+13
3496	2017-05-29	123.965	124.553	123.729	124.553	1.920000e+13
3497	2017-05-30	124.550	125.090	124.468	124.853	1.830000e+06
3498	2017-05-31	124.856	125.310	124.428	124.554	1.820000e+13
3499	2017-06-01	124.435	124.712	124.193	124.309	1.309485e+06

3500 rows × 6 columns

In [4]:
```
df_test = df[df['Date']>'2017-06-01'].copy()
df_test
```

Out[4]:

	Date	Open	High	Low	Close	Volume
3500	2017-06-04	124.309	124.392	122.927	123.384	1.828061e+06
3501	2017-06-05	123.384	123.735	122.631	123.607	1.797753e+06
3502	2017-06-06	123.591	124.032	123.162	123.351	1.900000e+13
3503	2017-06-07	123.351	123.864	122.816	123.474	2.119815e+06
3504	2017-06-08	123.628	123.737	122.794	123.167	1.690000e+13
...	...	...	...	...	...	...
4123	2019-10-27	120.908	121.063	120.575	120.990	2.030000e+13
4124	2019-10-28	120.984	121.397	120.875	121.353	2.330000e+13
4125	2019-10-29	121.342	121.468	120.284	120.470	5.390000e+12
4126	2019-10-30	120.461	120.941	120.360	120.766	2.100000e+13
4127	2019-10-31	120.873	121.106	120.771	120.807	1.770000e+13

628 rows × 6 columns

In [5]:
```
training_df = df_training.drop(['Date'], axis = 1)
training_df
```

Out[5]:

	Open	High	Low	Close	Volume
0	134.970	135.351	133.983	134.793	1.447987e+06
1	134.717	135.094	134.098	134.685	2.730000e+06
2	134.642	135.717	134.278	134.582	2.750000e+06
3	134.589	135.980	134.440	135.094	2.790000e+06
4	135.100	135.366	134.019	134.086	2.789103e+06
...	...	...	...	...	...
3495	124.190	124.393	123.156	123.971	1.890000e+13
3496	123.965	124.553	123.729	124.553	1.920000e+13
3497	124.550	125.090	124.468	124.853	1.830000e+06
3498	124.856	125.310	124.428	124.554	1.820000e+13
3499	124.435	124.712	124.193	124.309	1.309485e+06

3500 rows × 5 columns

In [6]:
```
scaler = MinMaxScaler()
training_df = scaler.fit_transform(training_df)
training_df
```

Out[6]:
```
array([[5.41083684e-01, 5.38220342e-01, 5.31604560e-01, 5.38729585e-01,
        1.86115001e-08],
       [5.37718785e-01, 5.34791252e-01, 5.33137791e-01, 5.37293185e-01,
        3.50898162e-08],
       [5.36721285e-01, 5.43103793e-01, 5.35537631e-01, 5.35923286e-01,
        3.53468856e-08],
       ...,
       [4.02497739e-01, 4.01310259e-01, 4.04746350e-01, 4.06527637e-01,
        2.35216928e-08],
       [4.06567537e-01, 4.04245667e-01, 4.04213052e-01, 4.02550939e-01,
        2.33933162e-01],
       [4.00968240e-01, 3.96266695e-01, 4.01079928e-01, 3.99292440e-01,
        1.68312687e-08]])
```

In [7]:
```
X_train = []
Y_train = []
```

In [8]:
```
training_df.shape[0]
```

Out[8]:
```
3500
```

In [9]:
```
for i in range(60, training_df.shape[0]):
    X_train.append(training_df[i-60:i])
    Y_train.append(training_df[i, 0])
```

In [10]:
```
X_train, Y_train = np.array(X_train), np.array(Y_train)
```

In [11]:
```
X_train.shape, Y_train.shape
```

Out[11]:
```
((3440, 60, 5), (3440,))
```

In [12]:
```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
```

In [13]:
```
regressior = Sequential()

regressior.add(LSTM(units = 60, activation = 'relu', return_sequences = Tru
e, input_shape = (X_train.shape[1], 5)))
regressior.add(Dropout(0.2))

regressior.add(LSTM(units = 60, activation = 'relu', return_sequences = Tru
e))
regressior.add(Dropout(0.2))

regressior.add(LSTM(units = 80, activation = 'relu', return_sequences = Tru
e))
regressior.add(Dropout(0.2))

regressior.add(LSTM(units = 120, activation = 'relu'))
regressior.add(Dropout(0.2))

regressior.add(Dense(units = 1))
```

In [14]:
```
regressior.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 60, 60)            15840
_________________________________________________________________
dropout (Dropout)            (None, 60, 60)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 60, 60)            29040
_________________________________________________________________
dropout_1 (Dropout)          (None, 60, 60)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 60, 80)            45120
_________________________________________________________________
dropout_2 (Dropout)          (None, 60, 80)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 120)               96480
_________________________________________________________________
dropout_3 (Dropout)          (None, 120)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 121
=================================================================
```

```
Total params: 186,601
Trainable params: 186,601
Non-trainable params: 0
_________________________________________________________________
```

In [15]:
```
regressior.compile(optimizer='adam', loss = 'mean_squared_error')
```

In [16]:
```
regressior.fit(X_train, Y_train, epochs=50, batch_size=32)
```

```
Train on 3440 samples
Epoch 1/50
3440/3440 [==============================] - 14s 4ms/sample - loss: 0.0301
Epoch 2/50
3440/3440 [==============================] - 13s 4ms/sample - loss: 0.0046
Epoch 3/50
3440/3440 [==============================] - 15s 4ms/sample - loss: 0.0041
Epoch 4/50
3440/3440 [==============================] - 15s 4ms/sample - loss: 0.0038
Epoch 5/50
3440/3440 [==============================] - 15s 4ms/sample - loss: 0.0039
Epoch 6/50
3440/3440 [==============================] - 15s 4ms/sample - loss: 0.0034
Epoch 7/50
3440/3440 [==============================] - 15s 4ms/sample - loss: 0.0029
Epoch 8/50
3440/3440 [==============================] - 16s 5ms/sample - loss: 0.0029
Epoch 9/50
3440/3440 [==============================] - 16s 5ms/sample - loss: 0.0030
Epoch 10/50
3440/3440 [==============================] - 16s 5ms/sample - loss: 0.0025
Epoch 11/50
3440/3440 [==============================] - 16s 5ms/sample - loss: 0.0023
Epoch 12/50
3440/3440 [==============================] - 16s 5ms/sample - loss: 0.0022
Epoch 13/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 0.0023
Epoch 14/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 0.0021
Epoch 15/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0020
Epoch 16/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 0.0022
Epoch 17/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 0.0020
Epoch 18/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0017
Epoch 19/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 0.0019
Epoch 20/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0020
Epoch 21/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0017
Epoch 22/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0015
Epoch 23/50
3440/3440 [==============================] - 19s 5ms/sample - loss: 0.0016
Epoch 24/50
3440/3440 [==============================] - 19s 5ms/sample - loss: 0.0015
Epoch 25/50
3440/3440 [==============================] - 19s 5ms/sample - loss: 0.0015
```

```
Epoch 26/50
3440/3440 [==============================] - 19s 5ms/sample - loss: 0.0014
Epoch 27/50
3440/3440 [==============================] - 19s 6ms/sample - loss: 0.0015
Epoch 28/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 0.0014
Epoch 29/50
3440/3440 [==============================] - 19s 6ms/sample - loss: 0.0013
Epoch 30/50
3440/3440 [==============================] - 19s 6ms/sample - loss: 0.0012
Epoch 31/50
3440/3440 [==============================] - 19s 6ms/sample - loss: 0.0014
Epoch 32/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 0.0012
Epoch 33/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 0.0012
Epoch 34/50
3440/3440 [==============================] - 19s 6ms/sample - loss: 0.0011
Epoch 35/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 0.0011
Epoch 36/50
3440/3440 [==============================] - 16s 5ms/sample - loss: 0.0010
Epoch 37/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 0.0010
Epoch 38/50
3440/3440 [==============================] - 19s 6ms/sample - loss: 0.0010
Epoch 39/50
3440/3440 [==============================] - 19s 6ms/sample - loss: 0.0011
Epoch 40/50
3440/3440 [==============================] - 19s 6ms/sample - loss: 0.0011
Epoch 41/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 9.5515e
-04
Epoch 42/50
3440/3440 [==============================] - 16s 5ms/sample - loss: 0.0010
Epoch 43/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 8.9021e
-04
Epoch 44/50
3440/3440 [==============================] - 19s 5ms/sample - loss: 8.6638e
-04
Epoch 45/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 9.0622e
-04
Epoch 46/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 8.8133e
-04
Epoch 47/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 8.7223e
-04
Epoch 48/50
3440/3440 [==============================] - 17s 5ms/sample - loss: 7.9535e
-04
Epoch 49/50
3440/3440 [==============================] - 18s 5ms/sample - loss: 8.9568e
-04
Epoch 50/50
3440/3440 [==============================] - 20s 6ms/sample - loss: 7.8196e
-04
```

Out[16]: <tensorflow.python.keras.callbacks.History at 0x1c2de626358>

In [17]:
```
df_test.head()
```

Out[17]:

	Date	Open	High	Low	Close	Volume
3500	2017-06-04	124.309	124.392	122.927	123.384	1.828061e+06
3501	2017-06-05	123.384	123.735	122.631	123.607	1.797753e+06
3502	2017-06-06	123.591	124.032	123.162	123.351	1.900000e+13
3503	2017-06-07	123.351	123.864	122.816	123.474	2.119815e+06
3504	2017-06-08	123.628	123.737	122.794	123.167	1.690000e+13

In [18]:
```
df_training.tail(60)
```

Out[18]:

	Date	Open	High	Low	Close	Volume
3440	2017-03-12	122.376	122.629	121.652	121.671	1.920000e+13
3441	2017-03-13	121.674	122.059	121.204	121.689	2.010000e+05
3442	2017-03-14	121.689	122.042	121.130	121.999	2.310000e+13
3443	2017-03-15	121.997	122.253	120.818	120.988	1.840000e+13
3444	2017-03-16	120.927	121.432	120.714	120.870	1.640000e+13
3445	2017-03-19	120.872	121.838	120.622	120.756	2.320000e+06
3446	2017-03-20	120.741	120.850	119.684	120.023	2.300000e+13
3447	2017-03-21	120.022	120.313	119.323	119.612	2.330000e+06
3448	2017-03-22	119.613	120.245	119.524	120.207	2.150000e+13
3449	2017-03-23	120.182	120.317	119.544	120.224	2.150000e+13
3450	2017-03-26	120.219	120.386	119.697	120.170	1.920000e+13
3451	2017-03-27	120.166	120.442	119.015	119.541	2.000000e+13
3452	2017-03-28	119.541	119.848	119.087	119.469	2.025730e+06
3453	2017-03-29	119.448	119.821	118.624	118.647	2.080000e+13
3454	2017-03-30	118.762	119.048	118.130	118.304	1.830000e+13
3455	2017-04-02	118.304	118.360	117.429	118.184	1.920000e+13
3456	2017-04-03	118.179	118.794	118.020	118.038	1.940000e+06
3457	2017-04-04	118.040	118.426	117.360	117.942	2.200000e+06
3458	2017-04-05	117.942	118.164	117.318	117.613	2.450000e+06
3459	2017-04-06	117.554	118.062	117.458	117.544	1.960000e+06
3460	2017-04-09	117.541	117.541	116.223	116.241	2.120000e+13
3461	2017-04-10	116.233	116.544	115.933	116.258	2.391757e+06
3462	2017-04-11	116.257	116.456	115.719	115.760	2.210000e+13
3463	2017-04-12	115.742	116.017	115.189	115.268	1.110000e+06
3464	2017-04-13	115.523	116.030	114.852	115.891	1.700000e+13
3465	2017-04-16	115.880	116.492	115.759	116.352	2.280000e+05
3466	2017-04-17	116.352	116.977	116.269	116.586	2.080000e+05
3467	2017-04-18	116.588	117.818	116.531	117.151	2.080000e+13

3468	2017-04-19	117.149	117.306	116.466	116.964	1.990000e+13
3469	2017-04-20	120.629	120.636	118.920	119.282	2.380000e+13
3470	2017-04-23	119.276	121.645	119.064	121.328	2.140000e+13
3471	2017-04-24	121.328	121.975	120.913	121.088	2.360000e+13
3472	2017-04-25	121.083	121.889	120.606	120.951	2.199574e+06
3473	2017-04-26	120.947	121.999	120.693	121.439	1.940000e+05
3474	2017-04-27	121.424	122.089	121.304	121.901	1.560000e+13
3475	2017-04-30	121.898	122.600	121.869	122.418	1.840000e+13
3476	2017-05-01	122.416	122.866	122.318	122.733	1.740000e+13
3477	2017-05-02	122.732	123.647	122.599	123.543	2.000000e+06
3478	2017-05-03	123.527	124.054	122.927	123.924	1.950000e+13
3479	2017-05-04	124.467	124.500	122.984	123.718	2.250000e+13
3480	2017-05-07	123.706	124.539	123.652	123.937	2.140000e+13
3481	2017-05-08	123.925	124.323	123.495	124.186	2.050000e+13
3482	2017-05-09	124.184	124.427	123.323	123.658	1.990000e+06
3483	2017-05-10	123.658	123.954	123.311	123.874	2.200000e+06
3484	2017-05-11	123.639	124.900	123.628	124.869	1.870000e+13
3485	2017-05-14	124.876	125.815	124.592	125.355	2.100087e+06
3486	2017-05-15	125.353	125.393	123.638	123.653	2.680000e+06
3487	2017-05-16	123.653	124.111	122.558	123.769	3.090000e+13
3488	2017-05-17	123.762	124.941	123.382	124.673	2.390000e+13
3489	2017-05-18	124.275	125.297	124.118	125.076	2.120000e+13
3490	2017-05-21	125.076	125.389	124.563	124.988	2.370000e+13
3491	2017-05-22	124.986	125.406	124.905	125.080	2.170000e+13
3492	2017-05-23	125.078	125.806	125.063	125.389	2.010000e+06
3493	2017-05-24	125.389	125.421	124.167	124.433	2.013097e+06
3494	2017-05-25	124.273	124.641	124.184	124.194	1.130000e+13
3495	2017-05-28	124.190	124.393	123.156	123.971	1.890000e+13
3496	2017-05-29	123.965	124.553	123.729	124.553	1.920000e+13
3497	2017-05-30	124.550	125.090	124.468	124.853	1.830000e+06
3498	2017-05-31	124.856	125.310	124.428	124.554	1.820000e+13
3499	2017-06-01	124.435	124.712	124.193	124.309	1.309485e+06

In [19]:
```
past_60_days = df_training.tail(60)
```

In [20]:
```
df = past_60_days.append(df_test, ignore_index = True)
df = df.drop(['Date'], axis = 1)
df.head()
```

Out[20]:

	Open	High	Low	Close	Volume
0	122.376	122.629	121.652	121.671	1.920000e+13
1	121.674	122.059	121.204	121.689	2.010000e+05
2	121.689	122.042	121.130	121.999	2.310000e+13
3	121.997	122.253	120.818	120.988	1.840000e+13
4	120.927	121.432	120.714	120.870	1.640000e+13

In [21]:
```
inputs = scaler.transform(df)
inputs
```

Out[21]:
```
array([[3.73583551e-01, 3.68473721e-01, 3.67202187e-01, 3.64207054e-01,
        2.46786632e-01],
       [3.64246954e-01, 3.60868347e-01, 3.61229251e-01, 3.64446454e-01,
        2.58338946e-09],
       [3.64446454e-01, 3.60641520e-01, 3.60242650e-01, 3.68569453e-01,
        2.96915167e-01],
       ...,
       [3.59831356e-01, 3.52982774e-01, 3.48963402e-01, 3.48233761e-01,
        6.92802057e-02],
       [3.48114061e-01, 3.45951139e-01, 3.49976668e-01, 3.52170559e-01,
        2.69922879e-01],
       [3.53593659e-01, 3.48152695e-01, 3.55456303e-01, 3.52715859e-01,
        2.27506427e-01]])
```

In [22]:
```
X_test = []
Y_test = []

for i in range(60, inputs.shape[0]):
    X_test.append(inputs[i-60:i])
    Y_test.append(inputs[i, 0])
```

In [23]:
```
X_test, Y_test = np.array(X_test), np.array(Y_test)
X_test.shape, Y_test.shape
```

Out[23]:
```
((628, 60, 5), (628,))
```

In [24]:
```
y_pred = regressior.predict(X_test)
```

In [25]:
```
scaler.scale_
```

Out[25]:
```
array([1.32999947e-02, 1.33427622e-02, 1.33324445e-02, 1.32999947e-02,
       1.28534704e-14])
```

In [26]:
```
scale = 1/1.32999947e-02
scale
```

Out[26]:
```
75.18799988694732
```

In [27]:
```
y_pred = y_pred*scale
Y_test = Y_test*scale
```

In [29]:
```
plt.figure(figsize=(25,10))
plt.plot(Y_test, color ='red', label = 'Real EUR/JPY Close Price')
plt.plot(y_pred, color ='blue', label = 'Predicted EUR/JPY Close Price')
plt.title('EUR/JPY Price Prediction')
plt.xlabel('Time')
plt.ylabel('EUR/JPY Close Price')
plt.legend()
plt.show()
```

In [30]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

In [31]:
```
mean_absolute_error(Y_test, y_pred), mean_squared_error(Y_test, y_pred)
```

Out[31]:
```
(0.7596632656019777, 0.8675761417143208)
```

5 PRIEDAS. EUR/USD KURSO PROGNOZAVIMO 5 DIENŲ Į ATEITĮ KODAS

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pandas import Timestamp

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import math, datetime
```

In [2]:
```
df = pd.read_csv('EURUSD.csv',
                 header=0,
                  index_col='Date',
                  parse_dates=True)
df
```

Out[2]:

Date	Open	High	Low	Close	Volume
2003-12-31	1.25915	1.26035	1.24679	1.25434	5.120000e+04
2004-01-01	1.25420	1.26263	1.25198	1.25806	1.340000e+11
2004-01-04	1.25816	1.26924	1.25785	1.26727	1.350000e+05
2004-01-05	1.26720	1.28083	1.26650	1.27230	1.349585e+05
2004-01-06	1.27230	1.27404	1.26221	1.26296	1.350000e+11
...	...	...	...	...	...
2019-10-27	1.10990	1.11183	1.10735	1.11118	1.240000e+04
2019-10-28	1.11117	1.11516	1.10801	1.11501	1.362668e+04
2019-10-29	1.11499	1.11755	1.11314	1.11513	1.500000e+04
2019-10-30	1.11511	1.11718	1.11281	1.11641	1.299288e+04
2019-10-31	1.11684	1.11753	1.11247	1.11272	1.070000e+11

4128 rows × 5 columns

In [3]:
```
df = df[['Open','High','Low','Close','Volume',]]
```

In [4]:
```
df['HL_PCT'] = (df['High']-df['Close'])/df['Close']*100
df['PCT_change'] = (df['Close']-df['Open'])/df['Open']*100
```

In [5]:
```
df = df[['Close','HL_PCT','PCT_change','Volume']]
```

In [6]:
```
forecast_col = 'Close'
df.fillna(-99999, inplace=True)
```

In [7]:
```
forecast_out = int(math.ceil(0.0012*len(df)))
```

In [8]:
```
df['label'] = df[forecast_col].shift(-forecast_out)

print(df.tail(6))
```

```
              Close    HL_PCT  PCT_change        Volume     label
Date
2019-10-24  1.10991  0.066672    0.172383  1.048124e+04  1.11272
2019-10-27  1.11118  0.058496    0.115326  1.240000e+04       NaN
2019-10-28  1.11501  0.013453    0.345582  1.362668e+04       NaN
2019-10-29  1.11513  0.217015    0.012556  1.500000e+04       NaN
2019-10-30  1.11641  0.068971    0.116580  1.299288e+04       NaN
2019-10-31  1.11272  0.432274   -0.368898  1.070000e+11       NaN
```

In [9]:
```
X = np.array(df.drop(['label'],1))
X = preprocessing.scale(X)
X = X[:-forecast_out]
X_lately = X[-forecast_out:]

df.dropna(inplace=True)

y = np.array(df['label'])
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15)
```

In [11]:
```
clf = LinearRegression()
clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)

forecast_set = clf.predict(X_lately)

print(forecast_set, accuracy, forecast_out)
df['Forecast'] = np.nan
```

```
[1.11504068 1.11582248 1.1117049  1.10984466 1.11271143] 0.9810224549709547
5
```

In [12]:
```
last_date = df.iloc[-1].name
last_unix  = last_date.timestamp()
one_day = 86400
next_unix = last_unix+one_day
```

In [13]:
```
for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]

print(df.tail(6))

plt.figure(figsize=(25,12))
df['Close'].plot(color ='red')
df['Forecast'].plot(color ='blue')
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

```
                      Close    HL_PCT  PCT_change       Volume    label  \
Date
2019-10-24 00:00:00  1.10991  0.066672    0.172383  10481.24006  1.11272
2019-10-25 03:00:00      NaN       NaN         NaN          NaN      NaN
2019-10-26 03:00:00      NaN       NaN         NaN          NaN      NaN
2019-10-27 03:00:00      NaN       NaN         NaN          NaN      NaN
2019-10-28 02:00:00      NaN       NaN         NaN          NaN      NaN
2019-10-29 02:00:00      NaN       NaN         NaN          NaN      NaN

                     Forecast
Date
2019-10-24 00:00:00       NaN
2019-10-25 03:00:00  1.115041
2019-10-26 03:00:00  1.115822
2019-10-27 03:00:00  1.111705
2019-10-28 02:00:00  1.109845
2019-10-29 02:00:00  1.112711
```

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pandas import Timestamp

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import math, datetime
```

In [2]:
```
df = pd.read_csv('EURUSD.csv',
                 header=0,
                  index_col='Date',
                  parse_dates=True)
df
```

Out[2]:

Date	Open	High	Low	Close	Volume
2003-12-31	1.25915	1.26035	1.24679	1.25434	5.120000e+04
2004-01-01	1.25420	1.26263	1.25198	1.25806	1.340000e+11
2004-01-04	1.25816	1.26924	1.25785	1.26727	1.350000e+05
2004-01-05	1.26720	1.28083	1.26650	1.27230	1.349585e+05
2004-01-06	1.27230	1.27404	1.26221	1.26296	1.350000e+11
...	...	...	...	...	...
2019-10-27	1.10990	1.11183	1.10735	1.11118	1.240000e+04
2019-10-28	1.11117	1.11516	1.10801	1.11501	1.362668e+04
2019-10-29	1.11499	1.11755	1.11314	1.11513	1.500000e+04
2019-10-30	1.11511	1.11718	1.11281	1.11641	1.299288e+04
2019-10-31	1.11684	1.11753	1.11247	1.11272	1.070000e+11

4128 rows × 5 columns

In [3]:
```
df = df[['Open','High','Low','Close','Volume',]]
```

In [4]:
```
df['HL_PCT'] = (df['High']-df['Close'])/df['Close']*100
df['PCT_change'] = (df['Close']-df['Open'])/df['Open']*100
```

In [5]:
```
df = df[['Close','HL_PCT','PCT_change','Volume']]
```

In [6]:
```
forecast_col = 'Close'
df.fillna(-99999, inplace=True)
```

In [7]:
```
forecast_out = int(math.ceil(0.0048*len(df)))
```

In [8]:
```
df['label'] = df[forecast_col].shift(-forecast_out)

print(df.tail(21))
```

```
              Close    HL_PCT  PCT_change        Volume     label
Date
2019-10-03  1.09703  0.274377   -0.091072  1.670000e+11  1.11272
2019-10-06  1.09560  0.363271   -0.130352  2.170000e+04       NaN
2019-10-07  1.09710  0.173184    0.136911  1.800000e+11       NaN
2019-10-08  1.10050  0.260791    0.315394  2.350000e+11       NaN
2019-10-09  1.10329  0.269195    0.254432  2.510000e+11       NaN
2019-10-10  1.10248  0.161454   -0.061641  1.550000e+04       NaN
2019-10-13  1.10320  0.125997    0.063492  2.260000e+11       NaN
```

```
2019-10-14  1.10710  0.130973    0.354427  2.330000e+11      NaN
2019-10-15  1.11234  0.147437    0.473309  2.220000e+04      NaN
2019-10-16  1.11703  0.014324    0.419828  1.460000e+11      NaN
2019-10-17  1.11487  0.274471   -0.107520  1.270000e+11      NaN
2019-10-20  1.11243  0.292153   -0.218860  1.248444e+04      NaN
2019-10-21  1.11297  0.097038    0.049442  1.140329e+04      NaN
2019-10-22  1.11037  0.531354   -0.232713  1.480000e+11      NaN
2019-10-23  1.10790  0.396245   -0.221550  1.050000e+11      NaN
2019-10-24  1.10991  0.066672    0.172383  1.048124e+04      NaN
2019-10-27  1.11118  0.058496    0.115326  1.240000e+04      NaN
2019-10-28  1.11501  0.013453    0.345582  1.362668e+04      NaN
2019-10-29  1.11513  0.217015    0.012556  1.500000e+04      NaN
2019-10-30  1.11641  0.068971    0.116580  1.299288e+04      NaN
2019-10-31  1.11272  0.432274   -0.368898  1.070000e+11      NaN
```

In [9]:
```
X = np.array(df.drop(['label'],1))
X = preprocessing.scale(X)
X = X[:-forecast_out]
X_lately = X[-forecast_out:]

df.dropna(inplace=True)

y = np.array(df['label'])
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15)
```

In [11]:
```
clf = LinearRegression()
clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)

forecast_set = clf.predict(X_lately)

print(forecast_set, accuracy, forecast_out)
df['Forecast'] = np.nan
```

```
[1.11239026 1.10650219 1.11240643 1.11303209 1.10435596 1.11276243
 1.1083521  1.10867558 1.10764843 1.10498887 1.10852155 1.09749697
 1.09824201 1.10103698 1.09526965 1.10093024 1.10246584 1.1027691
 1.10315907 1.1028665 ] 0.9175536747000508 20
```

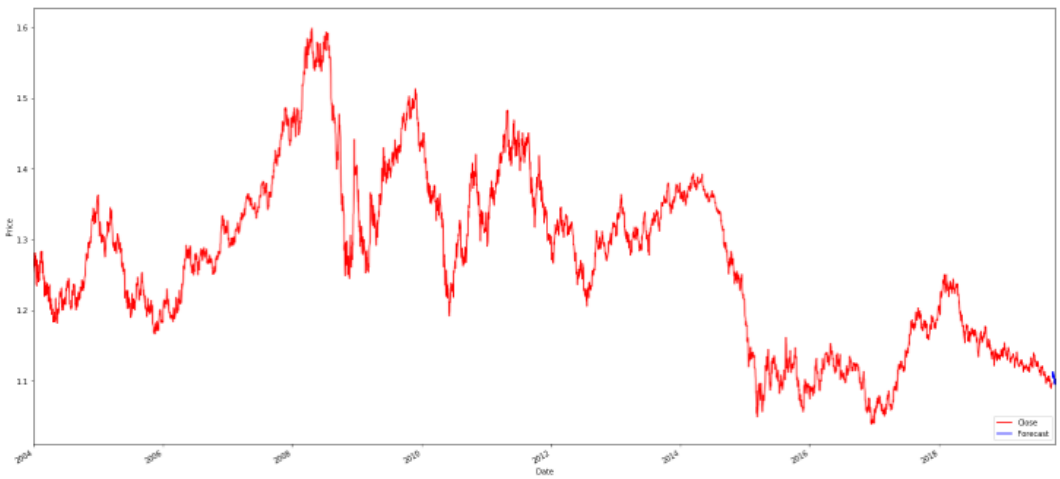
In [12]:
```
last_date = df.iloc[-1].name
last_unix  = last_date.timestamp()
one_day = 86400
next_unix = last_unix+one_day
```

In [13]:
```
for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]

print(df.tail(21))

plt.figure(figsize=(25,12))
df['Close'].plot(color ='red')
df['Forecast'].plot(color ='blue')
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

6 PRIEDAS. EUR/USD KURSO PROGNOZAVIMO 20 D. Į ATEITĮ KODAS. TĘSINYS

Date	Close	HL_PCT	PCT_change	Volume	label
2019-10-03 00:00:00	1.09703	0.274377	-0.091072	1.670000e+11	1.11272
2019-10-04 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-05 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-06 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-07 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-08 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-09 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-10 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-11 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-12 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-13 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-14 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-15 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-16 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-17 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-18 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-19 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-20 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-21 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-22 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-23 03:00:00	NaN	NaN	NaN	NaN	NaN

Date	Forecast
2019-10-03 00:00:00	NaN
2019-10-04 03:00:00	1.112390
2019-10-05 03:00:00	1.106502
2019-10-06 03:00:00	1.112406
2019-10-07 03:00:00	1.113032
2019-10-08 03:00:00	1.104356
2019-10-09 03:00:00	1.112762
2019-10-10 03:00:00	1.108352
2019-10-11 03:00:00	1.108676
2019-10-12 03:00:00	1.107648
2019-10-13 03:00:00	1.104989
2019-10-14 03:00:00	1.108522
2019-10-15 03:00:00	1.097497
2019-10-16 03:00:00	1.098242
2019-10-17 03:00:00	1.101037
2019-10-18 03:00:00	1.095270
2019-10-19 03:00:00	1.100930
2019-10-20 03:00:00	1.102466
2019-10-21 03:00:00	1.102769
2019-10-22 03:00:00	1.103159
2019-10-23 03:00:00	1.102866



7 **PRIEDAS.** EUR/USD KURSO PROGNOZAVIMO METUS Į ATEITĮ KODAS

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pandas import Timestamp

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import math, datetime
```

In [2]:
```
df = pd.read_csv('EURUSD.csv',
                 header=0,
                  index_col='Date',
                  parse_dates=True)
df
```

Out[2]:

Date	Open	High	Low	Close	Volume
2003-12-31	1.25915	1.26035	1.24679	1.25434	5.120000e+04
2004-01-01	1.25420	1.26263	1.25198	1.25806	1.340000e+11
2004-01-04	1.25816	1.26924	1.25785	1.26727	1.350000e+05
2004-01-05	1.26720	1.28083	1.26650	1.27230	1.349585e+05
2004-01-06	1.27230	1.27404	1.26221	1.26296	1.350000e+11
...	...	...	...	...	...
2019-10-27	1.10990	1.11183	1.10735	1.11118	1.240000e+04
2019-10-28	1.11117	1.11516	1.10801	1.11501	1.362668e+04
2019-10-29	1.11499	1.11755	1.11314	1.11513	1.500000e+04
2019-10-30	1.11511	1.11718	1.11281	1.11641	1.299288e+04
2019-10-31	1.11684	1.11753	1.11247	1.11272	1.070000e+11

4128 rows × 5 columns

In [3]:
```
df = df[['Open','High','Low','Close','Volume',]]
```

In [4]:
```
df['HL_PCT'] = (df['High']-df['Close'])/df['Close']*100
df['PCT_change'] = (df['Close']-df['Open'])/df['Open']*100
```

In [5]:
```
df = df[['Close','HL_PCT','PCT_change','Volume']]
```

In [6]:
```
forecast_col = 'Close'
df.fillna(-99999, inplace=True)
```

In [7]:
```
forecast_out = int(math.ceil(0.09*len(df)))
```

In [8]:
```
df['label'] = df[forecast_col].shift(-forecast_out)

print(df.tail(365))
```

```
              Close    HL_PCT  PCT_change        Volume  label
Date
2018-06-07  1.17833  0.315701    0.082388  2.030000e+04    NaN
2018-06-10  1.17446  0.546634   -0.328431  2.380000e+11    NaN
2018-06-11  1.17911  0.082265    0.394217  2.360000e+04    NaN
2018-06-12  1.15679  2.452476   -1.892121  2.883495e+04    NaN
2018-06-13  1.16059  0.180942    0.328495  2.636285e+04    NaN
...             ...       ...         ...           ...    ...
2019-10-27  1.11118  0.058496    0.115326  1.240000e+04    NaN
2019-10-28  1.11501  0.013453    0.345582  1.362668e+04    NaN
2019-10-29  1.11513  0.217015    0.012556  1.500000e+04    NaN
2019-10-30  1.11641  0.068971    0.116580  1.299288e+04    NaN
2019-10-31  1.11272  0.432274   -0.368898  1.070000e+11    NaN

[365 rows x 5 columns]
```

In [9]:
```
X = np.array(df.drop(['label'],1))
X = preprocessing.scale(X)
X = X[:-forecast_out]
X_lately = X[-forecast_out:]

df.dropna(inplace=True)

y = np.array(df['label'])
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15)
```

In [11]:
```
clf = LinearRegression()
clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)

forecast_set = clf.predict(X_lately)

print(forecast_set, accuracy, forecast_out)
df['Forecast'] = np.nan
```

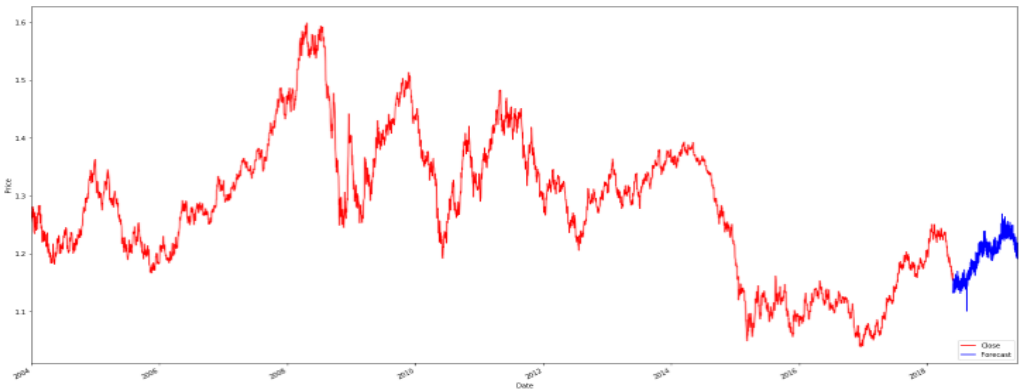
```
[1.13518645 1.13236827 1.14757528 1.13525166 1.15697425 1.14307482
 1.13250728 1.15106735 1.13785379 1.15605863 1.15154244 1.13958187
 1.13380563 1.14269256 1.13947961 1.13999724 1.16474684 1.13958895
 1.15625651 1.13872325 1.1445928  1.16031198 1.14201162 1.14324387
 1.14174677 1.14176229 1.16998675 1.1616162  1.14945189 1.16223099
 1.16022927 1.15962981 1.13747201 1.1384738  1.15366441 1.1405927
 1.13782625 1.15096314 1.13881582 1.1386595  1.1525112  1.13505472
 1.1500907  1.13152215 1.15517771 1.15772403 1.15618228 1.15044701
 1.14772372 1.13913103 1.15585398 1.13585785 1.14862862 1.15725429
 1.16521186 1.14375803 1.14027542 1.16731507 1.1598154  1.16123246
 1.16309582 1.14831936 1.14415338 1.14338115 1.14712761 1.17248509
 1.1663394  1.16452068 1.14956633 1.14261196 1.15404117 1.13880288
 1.1549344  1.14117525 1.15772917 1.14980683 1.15342591 1.14326896
 1.14304492 1.10098768 1.14900907 1.16325322 1.15672225 1.16672357
 1.14469147 1.14942235 1.1601295  1.1547168  1.17142213 1.16177833
 1.16042497 1.1682357  1.16907432 1.16194776 1.1565201  1.17629226
 1.15541266 1.16725322 1.16709866 1.15922158 1.17424494 1.18548867
 1.18449257 1.16217774 1.17244548 1.19057807 1.17139042 1.16510264
 1.17129662 1.17103239 1.17314352 1.18460604 1.18850371 1.18630608
 1.17708108 1.18683448 1.17337034 1.1858916  1.17191489 1.16816185
 1.17182159 1.17062864 1.17864148 1.18704693 1.1697276  1.18469357
 1.16231674 1.16459892 1.18064088 1.17178537 1.1699558  1.20256757
 1.17367956 1.19874179 1.19424749 1.1821542  1.19208689 1.17484211
 1.18140674 1.17963428 1.18110717 1.2020436  1.18316437 1.18334484
 1.200609   1.19684629 1.1902631  1.18484409 1.21775259 1.19187757
 1.20969757 1.19831625 1.21670837 1.21988938 1.19777659 1.20369253
 1.19770973 1.21035331 1.22120619 1.20607469 1.20208844 1.21950808
 1.21122561 1.21397005 1.22207967 1.20218141 1.21585253 1.19439762
 1.19687577 1.21553231 1.21984485 1.21738732 1.20300505 1.1966127
 1.21275242 1.20905513 1.23907152 1.20808756 1.19989439 1.21435284
 1.20786424 1.22446315 1.20482429 1.23934609 1.21536952 1.22965799
 1.22417371 1.21256722 1.22162382 1.21079439 1.22582917 1.20868954
 1.19867763 1.22084383 1.21184454 1.20861157 1.200584   1.19428811
 1.21794182 1.20142153 1.21943335 1.19873971 1.21642765 1.21610979
 1.19288437 1.21272102 1.22117597 1.20372153 1.20369752 1.20550082
 1.21492432 1.21405418 1.21633971 1.20312303 1.21967703 1.21329036
 1.2012169  1.19986693 1.20876929 1.1891311  1.19346949 1.206125
 1.19141548 1.21315959 1.19880301 1.18890029 1.18813416 1.19061873
 1.19231738 1.20905244 1.19353137 1.20936698 1.22408073 1.19989227
 1.20322732 1.2183793  1.19778703 1.20564685 1.21803167 1.22823334
 1.22682493 1.22454787 1.22187993 1.22734386 1.20316492 1.20050161
 1.20225831 1.20200443 1.20021406 1.2111034  1.20323711 1.20074061
 1.22130456 1.222957   1.2016631  1.20758543 1.22051884 1.20813321
 1.21909211 1.2017096  1.21868045 1.21781065 1.20883425 1.22715313
 1.23216767 1.21767619 1.21379478 1.21723925 1.21509918 1.21708496
```

```
 1.20739908 1.23400513 1.21621259 1.22722557 1.22964704 1.2191338
 1.25229226 1.22379465 1.22610177 1.23812552 1.24312201 1.25459346
 1.26903943 1.23362947 1.22584673 1.23077278 1.2570581  1.26020533
 1.25455832 1.25710657 1.25432374 1.2284626  1.24057958 1.24053326
 1.22265426 1.22884442 1.25518453 1.25515947 1.26409856 1.23440208
 1.22727903 1.24774483 1.22696452 1.24362565 1.23034852 1.24950202
 1.23876097 1.24399238 1.24717158 1.22576836 1.25274246 1.22890126
 1.25640997 1.24409747 1.22807529 1.23096217 1.24900719 1.24869294
 1.24570851 1.23089965 1.22950699 1.22952147 1.22956175 1.24949837
 1.23596375 1.25468206 1.23059614 1.22481476 1.23619367 1.23280875
 1.22797885 1.22596008 1.24191308 1.24295303 1.22476825 1.24863351
 1.24876023 1.22664546 1.22627034 1.2325719  1.24995058 1.24744627
 1.2490825  1.22825173 1.22391663 1.22105052 1.22154471 1.22243378
 1.23237352 1.23462546 1.21538006 1.23188914 1.20760656 1.20745565
 1.22625116 1.22479244 1.20295446 1.22925662 1.22668386 1.21073867
 1.22585748 1.19558979 1.21874698 1.21833607 1.19515883 1.21941129
 1.19250326 1.21299315 1.21427149 1.21736072 1.21019776 1.19108426] 0.32222
26377278391 372
```

In [12]:
```
last_date = df.iloc[-1].name
last_unix  = last_date.timestamp()
one_day = 86400
next_unix = last_unix+one_day
```

In [13]:
```
for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]

print(df.tail(365))

plt.figure(figsize=(25,12))
df['Close'].plot(color ='red')
df['Forecast'].plot(color ='blue')
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

```
                     Close  HL_PCT  PCT_change  Volume  label  Forecast
Date
2018-06-05 03:00:00    NaN     NaN         NaN     NaN    NaN  1.151067
2018-06-06 03:00:00    NaN     NaN         NaN     NaN    NaN  1.137854
2018-06-07 03:00:00    NaN     NaN         NaN     NaN    NaN  1.156059
2018-06-08 03:00:00    NaN     NaN         NaN     NaN    NaN  1.151542
2018-06-09 03:00:00    NaN     NaN         NaN     NaN    NaN  1.139582
...                    ...     ...         ...     ...    ...       ...
2019-05-31 03:00:00    NaN     NaN         NaN     NaN    NaN  1.212993
2019-06-01 03:00:00    NaN     NaN         NaN     NaN    NaN  1.214271
2019-06-02 03:00:00    NaN     NaN         NaN     NaN    NaN  1.217361
2019-06-03 03:00:00    NaN     NaN         NaN     NaN    NaN  1.210198
2019-06-04 03:00:00    NaN     NaN         NaN     NaN    NaN  1.191084

[365 rows x 6 columns]
```



8 PRIEDAS. EUR/GBP KURSO PROGNOZAVIMO 5 DIENŲ Į ATEITĮ KODAS

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pandas import Timestamp

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import math, datetime
```

In [2]:
```
df = pd.read_csv('EURGBP.csv',
                 header=0,
                  index_col='Date',
                  parse_dates=True)
```

In [3]:
```
df = df[['Open','High','Low','Close','Volume',]]
```

In [4]:
```
df['HL_PCT'] = (df['High']-df['Close'])/df['Close']*100
df['PCT_change'] = (df['Close']-df['Open'])/df['Open']*100
```

In [5]:
```
df = df[['Close','HL_PCT','PCT_change','Volume']]
```

In [6]:
```
forecast_col = 'Close'
df.fillna(-99999, inplace=True)
```

In [7]:
```
forecast_out = int(math.ceil(0.0012*len(df)))
```

In [8]:
```
df['label'] = df[forecast_col].shift(-forecast_out)

print(df.tail(6))
```

```
              Close    HL_PCT  PCT_change        Volume    label
Date
2019-10-24  0.86395  0.365762   -0.039339  1.150000e+16  0.86225
2019-10-27  0.86347  0.206145   -0.169954  1.145169e+09      NaN
2019-10-28  0.86391  0.138903    0.047481  1.280000e+16      NaN
2019-10-29  0.86442  0.020823    0.059034  1.300570e+09      NaN
2019-10-30  0.86190  0.329505   -0.291525  1.496236e+09      NaN
2019-10-31  0.86225  0.073065    0.038287  8.190000e+15      NaN
```

In [9]:
```
X = np.array(df.drop(['label'],1))
X = preprocessing.scale(X)
X = X[:-forecast_out]
X_lately = X[-forecast_out:]

df.dropna(inplace=True)

y = np.array(df['label'])
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15)
```

In [11]:
```
clf = LinearRegression()
clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)

forecast_set = clf.predict(X_lately)

print(forecast_set, accuracy, forecast_out)
df['Forecast'] = np.nan
```

```
[0.86036144 0.8643635  0.86224172 0.86420063 0.86381972] 0.9861411550737572
5
```

In [12]:
```
last_date = df.iloc[-1].name
last_unix  = last_date.timestamp()
one_day = 86400
next_unix = last_unix+one_day
```

In [13]:
```
for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]

print(df.tail(6))

plt.figure(figsize=(25,12))
df['Close'].plot(color ='red')
df['Forecast'].plot(color ='blue')
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

```
                       Close    HL_PCT  PCT_change        Volume    label
\
Date
2019-10-24 00:00:00  0.86395  0.365762   -0.039339  1.150000e+16  0.86225
2019-10-25 03:00:00      NaN       NaN         NaN           NaN      NaN
2019-10-26 03:00:00      NaN       NaN         NaN           NaN      NaN
2019-10-27 03:00:00      NaN       NaN         NaN           NaN      NaN
2019-10-28 02:00:00      NaN       NaN         NaN           NaN      NaN
2019-10-29 02:00:00      NaN       NaN         NaN           NaN      NaN

                     Forecast
Date
2019-10-24 00:00:00       NaN
2019-10-25 03:00:00  0.860361
2019-10-26 03:00:00  0.864364
2019-10-27 03:00:00  0.862242
2019-10-28 02:00:00  0.864201
2019-10-29 02:00:00  0.863820
```

9 PRIEDAS. EUR/GBP KURSO PROGNOZAVIMO 20 DIENŲ Į ATEITĮ KODAS

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pandas import Timestamp

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import math, datetime
```

In [2]:
```
df = pd.read_csv('EURGBP.csv',
                 header=0,
                  index_col='Date',
                  parse_dates=True)
```

In [3]:
```
df = df[['Open','High','Low','Close','Volume',]]
```

In [4]:
```
df['HL_PCT'] = (df['High']-df['Close'])/df['Close']*100
df['PCT_change'] = (df['Close']-df['Open'])/df['Open']*100
```

In [5]:
```
df = df[['Close','HL_PCT','PCT_change','Volume']]
```

In [6]:
```
forecast_col = 'Close'
df.fillna(-99999, inplace=True)
```

In [7]:
```
forecast_out = int(math.ceil(0.0048*len(df)))
```

In [8]:
```
df['label'] = df[forecast_col].shift(-forecast_out)

print(df.tail(21))
```

Date	Close	HL_PCT	PCT_change	Volume	label
2019-10-03	0.89039	0.378486	0.116939	1.490000e+16	0.86225
2019-10-06	0.89273	0.015682	0.019046	1.300000e+16	NaN
2019-10-07	0.89698	0.340030	0.468190	1.763495e+09	NaN
2019-10-08	0.89897	0.062294	0.224090	1.518883e+09	NaN
2019-10-09	0.88495	1.926663	-1.539848	2.003528e+09	NaN
2019-10-10	0.87303	1.601320	-1.381515	2.320000e+16	NaN
2019-10-13	0.87495	0.714327	0.163705	1.530000e+16	NaN
2019-10-14	0.86330	1.396965	-1.326994	2.281602e+09	NaN
2019-10-15	0.86309	1.001054	-0.068312	2.250000e+16	NaN
2019-10-16	0.86321	0.836413	0.023175	2.216365e+09	NaN
2019-10-17	0.86227	0.485927	-0.108896	1.810000e+16	NaN
2019-10-20	0.86053	0.660058	-0.165901	1.740000e+16	NaN
2019-10-21	0.86448	0.012724	0.457852	1.933610e+09	NaN
2019-10-22	0.86218	0.452342	-0.266056	1.701274e+09	NaN
2019-10-23	0.86429	0.393386	0.219156	1.632808e+09	NaN
2019-10-24	0.86395	0.365762	-0.039339	1.150000e+16	NaN
2019-10-27	0.86347	0.206145	-0.169954	1.145169e+09	NaN
2019-10-28	0.86391	0.138903	0.047481	1.280000e+16	NaN
2019-10-29	0.86442	0.020823	0.059034	1.300570e+09	NaN
2019-10-30	0.86190	0.329505	-0.291525	1.496236e+09	NaN
2019-10-31	0.86225	0.073065	0.038287	8.190000e+15	NaN

9 PRIEDAS. EUR/GBP KURSO PROGNOZAVIMO 20 D. Į ATEITĮ KODAS. TĘSINYS

In [9]:
```
X = np.array(df.drop(['label'],1))
X = preprocessing.scale(X)
X = X[:-forecast_out]
X_lately = X[-forecast_out:]

df.dropna(inplace=True)

y = np.array(df['label'])
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15)
```

In [11]:
```
clf = LinearRegression()
clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)

forecast_set = clf.predict(X_lately)

print(forecast_set, accuracy, forecast_out)
df['Forecast'] = np.nan
```

```
[0.89415284 0.8922525  0.89104332 0.89524697 0.8863668  0.88464955
 0.88449984 0.88336791 0.88222887 0.88155937 0.88263907 0.88069678
 0.88462013 0.88486009 0.88842066 0.88508672 0.88804167 0.8889726
 0.88717428 0.88892752] 0.9581568587087219 20
```

In [12]:
```
last_date = df.iloc[-1].name
last_unix  = last_date.timestamp()
one_day = 86400
next_unix = last_unix+one_day
```

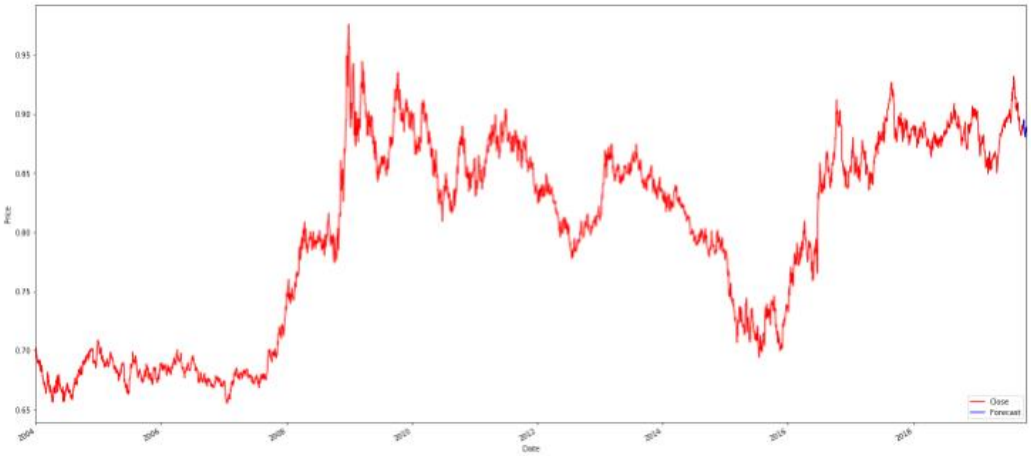
In [13]:
```
for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]

print(df.tail(21))

plt.figure(figsize=(25,12))
df['Close'].plot(color ='red')
df['Forecast'].plot(color ='blue')
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

Date	Close	HL_PCT	PCT_change	Volume	label
2019-10-03 00:00:00	0.89039	0.378486	0.116939	1.490000e+16	0.86225
2019-10-04 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-05 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-06 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-07 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-08 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-09 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-10 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-11 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-12 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-13 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-14 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-15 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-16 03:00:00	NaN	NaN	NaN	NaN	NaN

9 PRIEDAS. EUR/GBP KURSO PROGNOZAVIMO 20 D. Į ATEITĮ KODAS. TĘSINYS

```
2019-10-17 03:00:00      NaN      NaN      NaN      NaN      NaN
2019-10-18 03:00:00      NaN      NaN      NaN      NaN      NaN
2019-10-19 03:00:00      NaN      NaN      NaN      NaN      NaN
2019-10-20 03:00:00      NaN      NaN      NaN      NaN      NaN
2019-10-21 03:00:00      NaN      NaN      NaN      NaN      NaN
2019-10-22 03:00:00      NaN      NaN      NaN      NaN      NaN
2019-10-23 03:00:00      NaN      NaN      NaN      NaN      NaN
```

Date	Forecast
2019-10-03 00:00:00	NaN
2019-10-04 03:00:00	0.894153
2019-10-05 03:00:00	0.892252
2019-10-06 03:00:00	0.891043
2019-10-07 03:00:00	0.895247
2019-10-08 03:00:00	0.886367
2019-10-09 03:00:00	0.884650
2019-10-10 03:00:00	0.884500
2019-10-11 03:00:00	0.883368
2019-10-12 03:00:00	0.882229
2019-10-13 03:00:00	0.881559
2019-10-14 03:00:00	0.882639
2019-10-15 03:00:00	0.880697
2019-10-16 03:00:00	0.884620
2019-10-17 03:00:00	0.884860
2019-10-18 03:00:00	0.888421
2019-10-19 03:00:00	0.885087
2019-10-20 03:00:00	0.888042
2019-10-21 03:00:00	0.888973
2019-10-22 03:00:00	0.887174
2019-10-23 03:00:00	0.888928



10 PRIEDAS. EUR/GBP KURSO PROGNOZAVIMO METUS Į ATEITĮ KODAS

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pandas import Timestamp

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import math, datetime
```

In [2]:
```
df = pd.read_csv('EURGBP.csv',
                 header=0,
                  index_col='Date',
                  parse_dates=True)
```

In [3]:
```
df = df[['Open','High','Low','Close','Volume',]]
```

In [4]:
```
df['HL_PCT'] = (df['High']-df['Close'])/df['Close']*100
df['PCT_change'] = (df['Close']-df['Open'])/df['Open']*100
```

In [5]:
```
df = df[['Close','HL_PCT','PCT_change','Volume']]
```

In [6]:
```
forecast_col = 'Close'
df.fillna(-99999, inplace=True)
```

In [7]:
```
forecast_out = int(math.ceil(0.09*len(df)))
```

In [9]:
```
df['label'] = df[forecast_col].shift(-forecast_out)

print(df.tail(365))
```

```
              Close    HL_PCT  PCT_change        Volume  label
Date
2018-06-07  0.87798  0.228935   -0.150119  1.430000e+16    NaN
2018-06-10  0.88110  0.238338    0.357647  1.167803e+09    NaN
2018-06-11  0.87842  0.548712   -0.310954  1.505663e+09    NaN
2018-06-12  0.88180  0.089589    0.383639  1.370000e+16    NaN
2018-06-13  0.87239  1.121058   -1.067135  1.526877e+09    NaN
...             ...       ...         ...           ...    ...
2019-10-27  0.86347  0.206145   -0.169954  1.145169e+09    NaN
2019-10-28  0.86391  0.138903    0.047481  1.280000e+16    NaN
2019-10-29  0.86442  0.020823    0.059034  1.300570e+09    NaN
2019-10-30  0.86190  0.329505   -0.291525  1.496236e+09    NaN
2019-10-31  0.86225  0.073065    0.038287  8.190000e+15    NaN

[365 rows x 5 columns]
```

In [10]:
```
X = np.array(df.drop(['label'],1))
X = preprocessing.scale(X)
X = X[:-forecast_out]
X_lately = X[-forecast_out:]

df.dropna(inplace=True)

y = np.array(df['label'])
```

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15)
```

In [12]:
```
clf = LinearRegression()
clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)

forecast_set = clf.predict(X_lately)

print(forecast_set, accuracy, forecast_out)
df['Forecast'] = np.nan
```

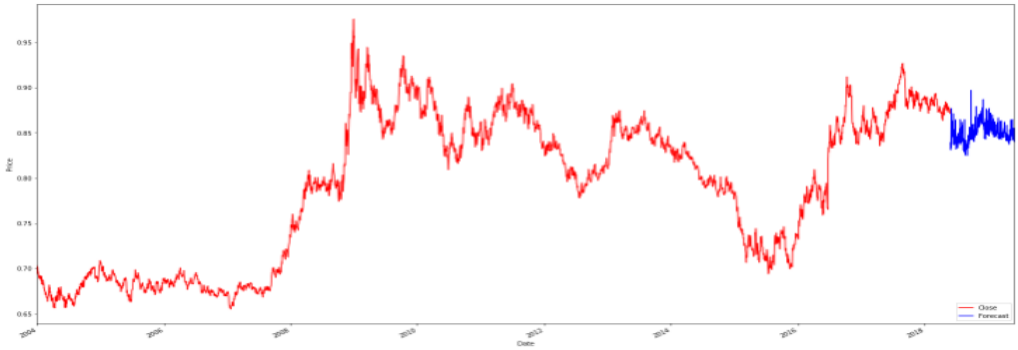
```
[0.83913012 0.83289568 0.83658307 0.83105443 0.84434235 0.87706642
 0.83342045 0.83812658 0.84131222 0.85249823 0.84482198 0.85338287
 0.86325882 0.8518434  0.85154733 0.84638117 0.84797455 0.87125301
 0.85270158 0.83704628 0.83915256 0.84435943 0.85369764 0.8454036
 0.8370839  0.83814803 0.84617734 0.84954175 0.84956929 0.86454193
 0.84052654 0.84005501 0.84881681 0.83198815 0.83691215 0.84188305
 0.83613237 0.84632351 0.83360094 0.84090534 0.84224958 0.83471701
 0.84203985 0.83344219 0.83603367 0.84395594 0.84972695 0.84001936
 0.84265009 0.83876192 0.84488855 0.8465543  0.84264456 0.84507599
 0.84809297 0.85476402 0.84890519 0.86481335 0.83710605 0.84820862
 0.85680758 0.84447567 0.85173433 0.84910457 0.83914852 0.84627657
 0.84099248 0.84121194 0.86195406 0.84944913 0.85853225 0.84068847
 0.84257943 0.84280067 0.84175514 0.84380191 0.83059451 0.84295138
 0.83456814 0.83371748 0.82888317 0.83577703 0.86474958 0.83096959
 0.83887823 0.82529416 0.82678346 0.83641564 0.83693221 0.83695324
 0.84189882 0.83158747 0.83943378 0.83594502 0.83198046 0.83660637
 0.83632112 0.82908742 0.8249785  0.83703897 0.83777859 0.83504332
 0.84744812 0.83805375 0.83484984 0.8334169  0.83918121 0.8451969
 0.8377131  0.84133542 0.85639858 0.84156672 0.84137148 0.85366194
 0.85463039 0.85555347 0.84568896 0.84991104 0.84561421 0.84269368
 0.89719266 0.85785706 0.85253419 0.85405601 0.85049569 0.84308488
 0.84662246 0.85928799 0.8512338  0.85197771 0.84231808 0.84480481
 0.85778382 0.85955721 0.841334   0.84804299 0.84459309 0.84029767
 0.84843537 0.8528857  0.85349967 0.84492524 0.86185059 0.86768081
 0.85899741 0.85011911 0.84711173 0.86827627 0.85004494 0.87923556
 0.85541577 0.85428668 0.85222411 0.84798091 0.85434676 0.85383827
 0.85139541 0.85081674 0.86410223 0.867066   0.85736083 0.85818294
 0.86494443 0.85589133 0.86204838 0.87057893 0.8568507  0.86277447
 0.86464859 0.85795382 0.86744572 0.85931667 0.85931136 0.8707551
 0.86095923 0.8690259  0.86371346 0.8726692  0.86928035 0.87443047
 0.8671585  0.87179317 0.86597543 0.86063376 0.87701479 0.87802341
 0.86435886 0.86133686 0.85941222 0.88696366 0.86735959 0.85977372
 0.85757198 0.85978607 0.85381116 0.8609178  0.85369175 0.84365083
 0.84690096 0.84954018 0.86194656 0.85371851 0.84980677 0.84313106
 0.86063839 0.86570587 0.86288428 0.85505544 0.85761691 0.87655317
 0.84881967 0.85322079 0.85501268 0.85219519 0.86372088 0.86304322
 0.84795365 0.86264734 0.85662976 0.87478234 0.85971396 0.84952332
 0.85295803 0.84658012 0.87711971 0.85875727 0.85831443 0.85298313
 0.86052836 0.85049958 0.85397134 0.86487245 0.86330894 0.8720238
 0.85494829 0.85558597 0.85316824 0.85287327 0.85433682 0.85187721
 0.86270927 0.85277817 0.86998304 0.85355453 0.85063344 0.84824267
 0.85051344 0.86350478 0.85695363 0.86552896 0.85786435 0.85303979
 0.84782825 0.84525374 0.85820289 0.8623945  0.85341798 0.86097893
 0.84786358 0.85222284 0.84583752 0.84955086 0.85005713 0.85232756
 0.84735497 0.84704234 0.85925217 0.85320637 0.85737365 0.8506899
```

```
 0.85584865 0.84908395 0.86094567 0.85812566 0.86213465 0.85504702
 0.8443748  0.86658926 0.84706369 0.8489974  0.84975466 0.84313648
 0.84628827 0.84964202 0.84644194 0.84401209 0.85306309 0.86296664
 0.84691057 0.8547876  0.85810967 0.862567   0.85353463 0.84269957
 0.86336958 0.851908   0.8488863  0.8494989  0.84534797 0.84985006
 0.84939306 0.84896132 0.85094349 0.84995462 0.84469503 0.85058943
 0.85061119 0.85359476 0.85001441 0.85464266 0.85237706 0.85780941
 0.86203095 0.85996084 0.84941478 0.84632011 0.84516326 0.85027229
 0.85004861 0.84471809 0.84144075 0.84653376 0.84304033 0.84515141
 0.84178762 0.84303766 0.85851583 0.84117252 0.84322105 0.84849279
 0.8438915  0.84800942 0.84835705 0.84297507 0.84710873 0.83944344
 0.84268506 0.84256506 0.84616663 0.83731579 0.84412086 0.84138902
 0.84904982 0.85370751 0.84989085 0.84628147 0.84460223 0.8416371
 0.85286949 0.86437788 0.8523571  0.85443183 0.84719026 0.84711483
 0.84713389 0.84789614 0.85401951 0.84538329 0.86484612 0.84195605
 0.84613451 0.85523956 0.84409653 0.84596332 0.84309758 0.84555662
 0.84542909 0.84657776 0.84071588 0.84857027 0.85544572 0.83985029] 0.29673
390485716644 372
```

In [13]:
```
last_date = df.iloc[-1].name
last_unix  = last_date.timestamp()
one_day = 86400
next_unix = last_unix+one_day
```

In [14]:
```
for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]

print(df.tail(365))

plt.figure(figsize=(25,12))
df['Close'].plot(color ='red')
df['Forecast'].plot(color ='blue')
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

```
                     Close  HL_PCT  PCT_change  Volume  label  Forecast
Date
2018-06-05 03:00:00    NaN     NaN         NaN     NaN    NaN  0.838127
2018-06-06 03:00:00    NaN     NaN         NaN     NaN    NaN  0.841312
2018-06-07 03:00:00    NaN     NaN         NaN     NaN    NaN  0.852498
2018-06-08 03:00:00    NaN     NaN         NaN     NaN    NaN  0.844822
2018-06-09 03:00:00    NaN     NaN         NaN     NaN    NaN  0.853383
...                    ...     ...         ...     ...    ...       ...
2019-05-31 03:00:00    NaN     NaN         NaN     NaN    NaN  0.846578
2019-06-01 03:00:00    NaN     NaN         NaN     NaN    NaN  0.840716
2019-06-02 03:00:00    NaN     NaN         NaN     NaN    NaN  0.848570
2019-06-03 03:00:00    NaN     NaN         NaN     NaN    NaN  0.855446
2019-06-04 03:00:00    NaN     NaN         NaN     NaN    NaN  0.839850

[365 rows x 6 columns]
```



11 PRIEDAS. EUR/JPY KURSO PROGNOZAVIMO 5 DIENŲ Į ATEITĮ KODAS

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pandas import Timestamp

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import math, datetime
```

In [2]:
```
df = pd.read_csv('EURJPY.csv',
                 header=0,
                  index_col='Date',
                  parse_dates=True)
df
```

Out[2]:

Date	Open	High	Low	Close	Volume
2003-12-31	134.970	135.351	133.983	134.793	1.447987e+06
2004-01-01	134.717	135.094	134.098	134.685	2.730000e+06
2004-01-04	134.642	135.717	134.278	134.582	2.750000e+06
2004-01-05	134.589	135.980	134.440	135.094	2.790000e+06
2004-01-06	135.100	135.366	134.019	134.086	2.789103e+06
...	...	...	...	...	...
2019-10-27	120.908	121.063	120.575	120.990	2.030000e+13
2019-10-28	120.984	121.397	120.875	121.353	2.330000e+13
2019-10-29	121.342	121.468	120.284	120.470	5.390000e+12
2019-10-30	120.461	120.941	120.360	120.766	2.100000e+13
2019-10-31	120.873	121.106	120.771	120.807	1.770000e+13

4128 rows × 5 columns

In [3]:
```
df = df[['Open','High','Low','Close','Volume',]]
```

In [4]:
```
df['HL_PCT'] = (df['High']-df['Close'])/df['Close']*100
df['PCT_change'] = (df['Close']-df['Open'])/df['Open']*100
```

In [5]:
```
df = df[['Close','HL_PCT','PCT_change','Volume']]
```

In [6]:
```
forecast_col = 'Close'
df.fillna(-99999, inplace=True)
```

In [7]:
```
forecast_out = int(math.ceil(0.0012*len(df)))
```

In [8]:
```
df['label'] = df[forecast_col].shift(-forecast_out)

print(df.tail(6))
```

```
              Close    HL_PCT  PCT_change        Volume    label
Date
2019-10-24  120.908  0.066993    0.341920  1.880000e+12  120.807
2019-10-27  120.990  0.060336    0.067820  2.030000e+13      NaN
2019-10-28  121.353  0.036258    0.304999  2.330000e+13      NaN
2019-10-29  120.470  0.828422   -0.718630  5.390000e+12      NaN
2019-10-30  120.766  0.144908    0.253194  2.100000e+13      NaN
2019-10-31  120.807  0.247502   -0.054603  1.770000e+13      NaN
```

In [9]:
```
X = np.array(df.drop(['label'],1))
X = preprocessing.scale(X)
X = X[:-forecast_out]
X_lately = X[-forecast_out:]

df.dropna(inplace=True)

y = np.array(df['label'])
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15)
```

In [11]:
```
clf = LinearRegression()
clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)

forecast_set = clf.predict(X_lately)

print(forecast_set, accuracy, forecast_out)
df['Forecast'] = np.nan
```

```
[120.84091431 121.1811405  120.619752   120.47765073 121.09919301] 0.9849912461
849442 5
```

In [12]:
```
last_date = df.iloc[-1].name
last_unix  = last_date.timestamp()
one_day = 86400
next_unix = last_unix+one_day
```

In [13]:
```
for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]

print(df.tail(6))

plt.figure(figsize=(25,12))
df['Close'].plot(color ='red')
df['Forecast'].plot(color ='blue')
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

```
                      Close    HL_PCT  PCT_change        Volume    label  \
Date
2019-10-24 00:00:00  120.908  0.066993     0.34192  1.880000e+12  120.807
2019-10-25 03:00:00      NaN       NaN         NaN           NaN      NaN
2019-10-26 03:00:00      NaN       NaN         NaN           NaN      NaN
2019-10-27 03:00:00      NaN       NaN         NaN           NaN      NaN
2019-10-28 02:00:00      NaN       NaN         NaN           NaN      NaN
2019-10-29 02:00:00      NaN       NaN         NaN           NaN      NaN

                      Forecast
Date
2019-10-24 00:00:00        NaN
2019-10-25 03:00:00  120.840914
2019-10-26 03:00:00  121.181140
2019-10-27 03:00:00  120.619752
2019-10-28 02:00:00  120.477651
2019-10-29 02:00:00  121.099193
```

12 PRIEDAS. EUR/JPY KURSO PROGNOZAVIMO 20 DIENŲ Į ATEITĮ KODAS

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pandas import Timestamp

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import math, datetime
```

In [2]:
```
df = pd.read_csv('EURJPY.csv',
                 header=0,
                  index_col='Date',
                  parse_dates=True)
df
```

Out[2]:

Date	Open	High	Low	Close	Volume
2003-12-31	134.970	135.351	133.983	134.793	1.447987e+06
2004-01-01	134.717	135.094	134.098	134.685	2.730000e+06
2004-01-04	134.642	135.717	134.278	134.582	2.750000e+06
2004-01-05	134.589	135.980	134.440	135.094	2.790000e+06
2004-01-06	135.100	135.366	134.019	134.086	2.789103e+06
...	...	...	...	...	...
2019-10-27	120.908	121.063	120.575	120.990	2.030000e+13
2019-10-28	120.984	121.397	120.875	121.353	2.330000e+13
2019-10-29	121.342	121.468	120.284	120.470	5.390000e+12
2019-10-30	120.461	120.941	120.360	120.766	2.100000e+13
2019-10-31	120.873	121.106	120.771	120.807	1.770000e+13

4128 rows × 5 columns

In [3]:
```
df = df[['Open','High','Low','Close','Volume',]]
```

In [4]:
```
df['HL_PCT'] = (df['High']-df['Close'])/df['Close']*100
df['PCT_change'] = (df['Close']-df['Open'])/df['Open']*100
```

In [5]:
```
df = df[['Close','HL_PCT','PCT_change','Volume']]
```

In [6]:
```
forecast_col = 'Close'
df.fillna(-99999, inplace=True)
```

In [7]:
```
forecast_out = int(math.ceil(0.0048*len(df)))
```

In [8]:
```
df['label'] = df[forecast_col].shift(-forecast_out)

print(df.tail(21))
```

Date	Close	HL_PCT	PCT_change	Volume	label
2019-10-03	117.657	0.209082	0.469656	2.390000e+13	120.807
2019-10-06	117.303	0.565203	-0.277990	2.970000e+13	NaN
2019-10-07	117.913	0.153503	0.523449	2.450000e+12	NaN
2019-10-08	118.815	0.106889	0.784630	3.590000e+12	NaN
2019-10-09	119.630	0.311795	0.685093	3.460000e+13	NaN
2019-10-10	119.516	0.163995	-0.006693	2.270000e+13	NaN
2019-10-13	120.086	0.114918	0.480286	2.890000e+13	NaN
2019-10-14	120.417	0.042353	0.275636	2.970000e+13	NaN
2019-10-15	120.843	0.422863	0.353771	2.730000e+13	NaN
2019-10-16	121.074	0.036341	0.185354	2.570000e+13	NaN
2019-10-17	121.093	0.312983	0.051227	2.270000e+13	NaN
2019-10-20	120.680	0.501326	-0.339414	2.230000e+12	NaN
2019-10-21	120.975	0.062823	0.244448	2.520000e+12	NaN
2019-10-22	120.602	0.649243	-0.308328	2.470000e+13	NaN
2019-10-23	120.366	0.361398	-0.195685	2.140000e+13	NaN
2019-10-24	120.908	0.066993	0.341920	1.880000e+12	NaN
2019-10-27	120.990	0.060336	0.067820	2.030000e+13	NaN
2019-10-28	121.353	0.036258	0.304999	2.330000e+13	NaN
2019-10-29	120.470	0.828422	-0.718630	5.390000e+12	NaN
2019-10-30	120.766	0.144908	0.253194	2.100000e+13	NaN
2019-10-31	120.807	0.247502	-0.054603	1.770000e+13	NaN

In [9]:
```
X = np.array(df.drop(['label'],1))
X = preprocessing.scale(X)
X = X[:-forecast_out]
X_lately = X[-forecast_out:]

df.dropna(inplace=True)

y = np.array(df['label'])
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15)
```

In [11]:
```
clf = LinearRegression()
clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)

forecast_set = clf.predict(X_lately)

print(forecast_set, accuracy, forecast_out)
df['Forecast'] = np.nan
```

```
[119.48866663 119.06771936 120.04625479 120.02069298 119.38116363
 120.12525471 120.24300121 119.62936519 118.90718689 118.73163111
 118.22430243 118.5475593  118.20317174 118.37950343 118.25022964
 118.33299608 117.99903193 117.57648142 118.06914528 118.07425392] 0.922586
7441480918 20
```

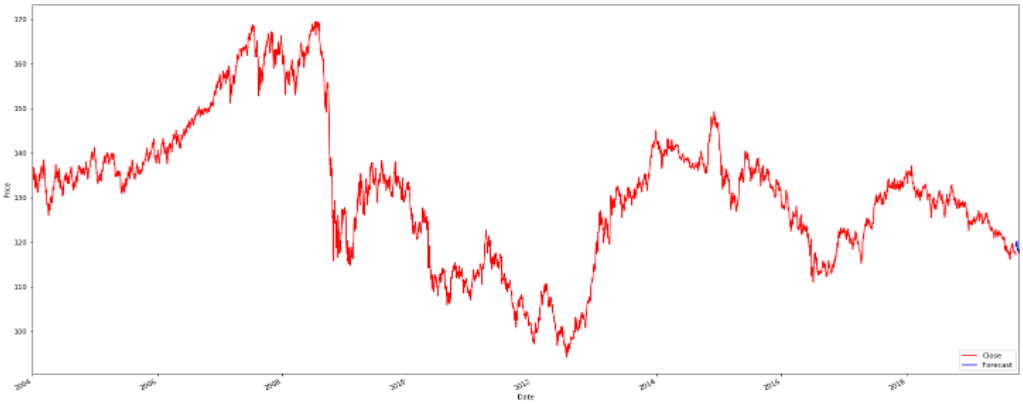
In [12]:
```
last_date = df.iloc[-1].name
last_unix  = last_date.timestamp()
one_day = 86400
next_unix = last_unix+one_day
```

In [13]:
```
for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]

print(df.tail(21))
```

```
plt.figure(figsize=(25,12))
df['Close'].plot(color ='red')
df['Forecast'].plot(color ='blue')
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

Date	Close	HL_PCT	PCT_change	Volume	label
2019-10-03 00:00:00	117.657	0.209082	0.469656	2.390000e+13	120.807
2019-10-04 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-05 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-06 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-07 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-08 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-09 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-10 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-11 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-12 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-13 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-14 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-15 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-16 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-17 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-18 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-19 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-20 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-21 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-22 03:00:00	NaN	NaN	NaN	NaN	NaN
2019-10-23 03:00:00	NaN	NaN	NaN	NaN	NaN

Date	Forecast
2019-10-03 00:00:00	NaN
2019-10-04 03:00:00	119.488667
2019-10-05 03:00:00	119.067719
2019-10-06 03:00:00	120.046255
2019-10-07 03:00:00	120.020693
2019-10-08 03:00:00	119.381164
2019-10-09 03:00:00	120.125255
2019-10-10 03:00:00	120.243001
2019-10-11 03:00:00	119.629365
2019-10-12 03:00:00	118.907187
2019-10-13 03:00:00	118.731631
2019-10-14 03:00:00	118.224302
2019-10-15 03:00:00	118.547559
2019-10-16 03:00:00	118.203172
2019-10-17 03:00:00	118.379503
2019-10-18 03:00:00	118.250230
2019-10-19 03:00:00	118.332996
2019-10-20 03:00:00	117.999032
2019-10-21 03:00:00	117.576481
2019-10-22 03:00:00	118.069145
2019-10-23 03:00:00	118.074254



13 PRIEDAS. EUR/JPY KURSO PROGNOZAVIMO METUS Į ATEITĮ KODAS

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pandas import Timestamp

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import math, datetime
```

In [2]:
```
df = pd.read_csv('EURJPY.csv',
                 header=0,
                  index_col='Date',
                  parse_dates=True)
df
```

Out[2]:

Date	Open	High	Low	Close	Volume
2003-12-31	134.970	135.351	133.983	134.793	1.447987e+06
2004-01-01	134.717	135.094	134.098	134.685	2.730000e+06
2004-01-04	134.642	135.717	134.278	134.582	2.750000e+06
2004-01-05	134.589	135.980	134.440	135.094	2.790000e+06
2004-01-06	135.100	135.366	134.019	134.086	2.789103e+06
...	...	...	...	...	...
2019-10-27	120.908	121.063	120.575	120.990	2.030000e+13
2019-10-28	120.984	121.397	120.875	121.353	2.330000e+13
2019-10-29	121.342	121.468	120.284	120.470	5.390000e+12
2019-10-30	120.461	120.941	120.360	120.766	2.100000e+13
2019-10-31	120.873	121.106	120.771	120.807	1.770000e+13

4128 rows × 5 columns

In [3]:
```
df = df[['Open','High','Low','Close','Volume',]]
```

In [4]:
```
df['HL_PCT'] = (df['High']-df['Close'])/df['Close']*100
df['PCT_change'] = (df['Close']-df['Open'])/df['Open']*100
```

In [5]:
```
df = df[['Close','HL_PCT','PCT_change','Volume']]
```

In [6]:
```
forecast_col = 'Close'
df.fillna(-99999, inplace=True)
```

In [7]:
```
forecast_out = int(math.ceil(0.09*len(df)))
```

In [8]:
```
df['label'] = df[forecast_col].shift(-forecast_out)

print(df.tail(365))
```

```
              Close    HL_PCT  PCT_change        Volume  label
Date
2018-06-07  129.662  0.305409    0.766266  2.540000e+06    NaN
2018-06-10  129.629  0.498345   -0.025451  2.730132e+06    NaN
2018-06-11  130.104  0.184468    0.364882  2.410000e+13    NaN
2018-06-12  127.976  1.856598   -1.635615  2.650000e+06    NaN
2018-06-13  128.440  0.045936    0.382178  2.640000e+13    NaN
...             ...       ...         ...           ...    ...
2019-10-27  120.990  0.060336    0.067820  2.030000e+13    NaN
2019-10-28  121.353  0.036258    0.304999  2.330000e+13    NaN
2019-10-29  120.470  0.828422   -0.718630  5.390000e+12    NaN
2019-10-30  120.766  0.144908    0.253194  2.100000e+13    NaN
2019-10-31  120.807  0.247502   -0.054603  1.770000e+13    NaN

[365 rows x 5 columns]
```

In [9]:
```
X = np.array(df.drop(['label'],1))
X = preprocessing.scale(X)
X = X[:-forecast_out]
X_lately = X[-forecast_out:]

df.dropna(inplace=True)

y = np.array(df['label'])
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15)
```

In [11]:
```
clf = LinearRegression()
clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)

forecast_set = clf.predict(X_lately)

print(forecast_set, accuracy, forecast_out)
df['Forecast'] = np.nan
```

```
[131.19478934 130.33841577 125.71584805 130.91462736 123.21074741
 128.07944848 127.22767119 130.51285336 128.46249077 128.17354957
 126.29062427 128.16899879 129.66043474 130.79948768 128.16196057
 127.76565356 127.98864259 126.34547277 127.66896304 130.29993606
 127.42123605 128.43192279 129.52249668 128.36499873 127.98352392
 125.62272546 129.15829783 126.97818344 128.84102719 129.02137186
 126.21854342 130.85794751 130.10656454 128.24276761 125.27827558
 128.19654072 130.69346227 130.30195249 130.32552703 125.7538429
 128.71116084 128.41019966 130.41899963 129.84797088 125.44323585
 127.71158375 130.80667003 127.77821695 128.61120439 128.76044588
 128.25467274 130.19257037 126.93534447 126.54044513 127.03989786
 129.95578668 127.49512018 130.28217277 131.00124661 125.92917024
 128.03261182 124.47533162 127.95365719 128.53870047 129.36061279
 131.20147258 130.60505306 126.89400743 129.84757823 124.92956149
 127.20148696 130.63208962 126.40190835 128.55134377 128.83560523
 127.74988313 125.04895679 129.21280353 126.78444965 126.92968917
 128.2365622  128.60327016 127.49757879 123.03262303 129.05154621
 125.73321681 121.23429347 125.82003436 125.83408173 126.62775045
 129.36116938 129.2922975  130.67612918 129.85167793 130.67134861
 129.69424515 127.36726633 131.20717154 129.78659255 132.57819844
 128.44370887 128.87640305 126.38958353 129.55018023 127.52068673
 128.59768092 130.563234   130.56070385 129.9128282  129.94454946
 130.38528959 130.77956844 131.18277643 131.24340611 128.57530273
 131.42366116 129.30505368 132.06877683 128.50953639 130.25690573
 130.34598475 130.11564899 128.87409593 130.35099027 128.68122169
 132.57618708 129.88355609 132.542969   132.03957153 130.69271972
 131.22922907 124.98861822 132.01641427 128.22872835 132.47175891
 131.46189628 132.18996669 130.99685816 131.14766049 130.74526763
 131.96347043 132.56966082 129.13640115 130.50940748 131.39751437
 133.1101604  131.8347391  131.97735042 129.12441614 130.62258307
 132.5671804  129.12947543 133.26018045 131.3510071  131.36673274
 132.68053498 134.34930011 131.13879752 132.90330517 129.80936433
 132.27539234 130.09412699 133.94878307 130.84180946 131.7474838
 130.1742141  130.65616008 129.88989122 130.04154221 132.54285592
 134.33260906 131.71807221 132.98603414 132.91761981 130.27540459
 132.72031146 132.38801804 131.65812194 131.49462386 131.57734784
 132.35391836 130.18010129 130.96621523 128.49155998 130.14280288
 131.13453098 130.74828817 131.85420491 132.18865291 127.37272734
```

```
132.6271458  132.63503447 131.79018152 133.14215994 133.19865293
125.20929202 133.81099319 134.23557723 133.26725923 132.28186553
131.67674452 132.16298868 134.52068315 133.15716865 131.71026889
133.45058705 133.38443881 132.97113023 132.88519044 133.14400669
134.21974406 134.19412663 130.75068991 133.97796115 133.03533119
132.02021224 132.50800769 134.44416273 129.38063748 130.93493664
132.85867748 132.23192411 134.16877445 134.36747317 133.42223618
129.43414475 132.90323706 134.82766347 134.98247269 133.42752038
134.56887922 132.15580702 132.67464248 133.15962636 131.41260026
134.06016098 134.77690257 132.66866961 133.94746506 130.50304355
131.47982098 131.64411943 131.81208278 129.58297814 131.89575306
132.07357856 132.28266759 132.42421406 132.18294036 134.99667505
134.15290378 134.01612571 134.78159504 127.60775947 132.6703042
131.8315722  130.09277227 131.67085165 134.94405466 135.93522326
135.67639892 134.96715834 133.82669778 135.05652712 132.63945182
134.28625028 135.72010167 131.32332438 132.22036508 132.74276089
132.77520166 129.84619083 130.55140604 130.16308991 133.77886325
131.87663876 133.018786   132.68686255 133.55622093 135.40352578
133.8098359  135.8527104  131.93235434 129.85758913 134.61309882
134.14581092 131.33684434 131.35375267 133.3936204  127.66983331
133.38542376 129.08386772 126.50013016 129.57129843 134.39233702
132.39184265 135.37082507 133.09840457 129.18845376 134.16460276
135.21731898 131.49564299 133.23369429 133.61380423 136.09733215
129.81373166 128.22907996 131.34223329 134.47036202 131.55176232
130.02989642 135.66303708 128.82606208 129.09450887 132.1634074
130.54759425 131.28080044 133.23065931 132.08727916 131.61751956
126.59535106 131.91559652 128.96482301 131.46308966 127.36100476
128.84108579 131.42363573 132.01220704 133.83848872 129.06168396
132.24830361 134.4196922  134.61709822 132.96730204 132.13911437
131.79637228 133.79169372 134.51090372 132.12909942 133.43417653
133.46550997 133.82908215 132.03046421 131.85266079 133.79975237
131.45037474 134.89204241 131.28079552 134.91365817 132.50275528
133.97402942 131.79354033 130.81084494 133.08753574 133.28262055
131.99756267 129.28215232 131.57478313 133.54780504 130.25706804
131.8581158  133.59173036 132.48276434 130.07266999 133.08534074
130.73304912 127.51870503 131.39261101 127.61692586 128.37332366
126.62819202 126.39493897] 0.12827871372290844 372
```

In [12]:
```
last_date = df.iloc[-1].name
last_unix  = last_date.timestamp()
one_day = 86400
next_unix = last_unix+one_day
```

In [13]:
```
for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]

print(df.tail(365))

plt.figure(figsize=(25,12))
df['Close'].plot(color ='red')
df['Forecast'].plot(color ='blue')
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

Date	Close	HL_PCT	PCT_change	Volume	label	Forecast
2018-06-05 03:00:00	NaN	NaN	NaN	NaN	NaN	130.512853
2018-06-06 03:00:00	NaN	NaN	NaN	NaN	NaN	128.462491
2018-06-07 03:00:00	NaN	NaN	NaN	NaN	NaN	128.173550
2018-06-08 03:00:00	NaN	NaN	NaN	NaN	NaN	126.290624
2018-06-09 03:00:00	NaN	NaN	NaN	NaN	NaN	128.168999
...	...	...	...	...	...	...
2019-05-31 03:00:00	NaN	NaN	NaN	NaN	NaN	131.392611
2019-06-01 03:00:00	NaN	NaN	NaN	NaN	NaN	127.616926
2019-06-02 03:00:00	NaN	NaN	NaN	NaN	NaN	128.373324
2019-06-03 03:00:00	NaN	NaN	NaN	NaN	NaN	126.628192
2019-06-04 03:00:00	NaN	NaN	NaN	NaN	NaN	126.394939

[365 rows x 6 columns]