VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Magistrinis darbas

Finansinių laiko eilučių kodavimas ranginiais metodais ir prognozavimas

Ordinal patterns and forecast in financial time series

Eitvydė Kabišaitytė

VILNIUS 2020

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

MATEMATINĖS ANALIZĖS KATEDRA

Darbo vadovas doc. dr. Martynas Manstavičius

Darbo recenzentė Eglė Gutauskaitė

Darbas apgintas _____

Darbas įvertintas _____

Registravimo NR. _____

Atidavimo į katedrą data 2020-01-03

Turinys

Finansinių laiko eilučių kodavimas ranginiais metodais ir prognozavimas

Santrauka

Kodavimas ranginiais metodais yra efektyvi technika, kuri buvo pristatyta C. Bandt ir B. Pompe darbe apie keitinių entropiją. Baigiamojo darbo tikslas yra išnagrinėti 1, 5, 10 ir 30 metų trukmės JAV vyriausybinių obligacijų pelningumą, užkoduotą ranginiais metodais ir sukurti du modelius užkoduotų laiko eilučių prognozavimui. Ranginiais metodais užkoduotų finansinių laiko eilučių prognozavimas iki šiol nebuvo nagrinėtas.

Išnagrinėjus ranginiais metodais užkoduotus 1, 5, 10 ir 30 metų trukmės obligacijų duomenis, buvo gauta, kad kiekvienos iš eilučių empirinis rangų blokų skirstinys nežymiai skiriasi nuo atitinkamos Markovo grandinės stacionaraus skirstinio, vertinant tai pagal pilnosios variacijos metriką. Tai reiškia, kad užkoduotų eilučių rangų blokų skirstiniui apibūdinti neblogai tinka atitinkamos Markovo grandinės stacionarus skirstinys. Paskaičiavus keitinių entropiją ir nubrėžus statistinio sudėtingumo entropijos kauzalinę plokštumą buvo nustatyta, kad nė viena iš užkoduotų eilučių nėra artima visiškai nuspėjamai, tačiau ir nėra visiškai atsitiktinė. Tyrimo metu pastebėta stipri koreliacija tarp užkoduotų eilučių ir teigiama priklausomybė, kuri mažėja didėjant skirtumui tarp obligacijų trukmės.

Darbe yra sudaryti du modeliai – vienmatis ir dvimatis, skirti prognozuoti užkoduotoms finansinėms laiko eilutėms. Vienmatis modelis paremtas Markovo grandinėmis ir prognozei naudojami tik prognozuojamos eilutės istoriniai duomenys. Dvimatis modelis paremtas sąlyginėmis tikimybėmis bei pilnos tikimybės formule ir prognozei yra naudojami ne tik prognozuojamos eilutės duomenys, bet taip pat ir kitos, susijusios eilutės duomenys. Prognozes įvertinus trimis kriterijais: vidutine absoliutine procentine paklaida, paklaidų dalimi mažesne už 5% bei paklaidų dalimi didesne už 95%, buvo nustatyta, kad prognozuojant su dvimačiu modeliu galima gauti tikslesnius rezultatus, nei prognozuojant tas pačias eilutes su vienmačiu modeliu.

Raktiniai žodžiai : kodavimas ranginiais metodais, finansinės laiko eilutės, keitinių entropija, statistinio sudėtingumo entropijos kauzalinė plokštuma, priklausomybė, prognozavimas.

Ordinal patterns and forecast in financial time series

Abstract

Ordinal patterns are an effective technique introduced by C. Bandt and B. Pompe in the article about permutation entropy. The aim of this thesis is to investigate 1, 5, 10 and 30 year U.S. treasury rate and to derive two models for ordinal patterns prediction. The prediction of financial time series has not been analysed in this way so far.

It was found that empirical probability distribution of ordinal patterns of different time series is close to stationary distribution of a corresponding Markov chain, measuring it by total variation metric. This means that stationary distribution of a corresponding Markov chain quite accurately describes the empirical probability distribution. Results of permutation entropy and complexity – entropy causality plane showed that none of times series is absolutely predictable, but not completely random too. The analysis also showed that there exists a strong correlation between ordinal patterns of different time series and a positive dependence that decreases with the increase in difference of bonds maturity.

Two models for ordinal patterns prediction are derived in this thesis – one-dimensional and two-dimensional. The one-dimensional model is based on Markov chains and uses only historical data of modelled time series. The two-dimensional model is based on conditional probabilities and law of total probability, this model uses not only relevant time series data, but also data of some other time series, too. The prediction quality is estimated by three criteria: mean absolute percentage error, error rate lower than 5% and error rate higher than 95%. It was found that the two-dimensional prediction model can provide more accurate results than forecasting the same data with one-dimensional model.

Key words : ordinal patterns, financial time series, permutation entropy, statistical complexity – entropy causality plane, dependence, forecast

1 ĮVADAS

Kaip teigiama [1] šaltinyje, ranginė simbolinė analizė yra efektyvi technika laiko eilučių analizėje, kuri atveria įdomias ir plačias perspektyvas. Tam, kad ranginiai metodai būtų naudingi, jų analizė turi būti paprastesnė nei originalių laiko eilučių, tačiau tuo pat metu turi būti išsaugota aktuali laiko eilučių informacija.

Duomenų kodavimas ranginiais metodais buvo pristatytas C. Bandt ir B. Pompe darbe apie keitinių entropiją (angl. permutation entropy) [2] ir toliau nagrinėtas ne viename darbe. Vienas tokių darbų yra JM Amigo, Karsten Keller ir VA Unakafova "Ordinal symbolic analysis and its application to biomedical recordings" [1], kuriame aiškiai ir paprastai nusakoma ranginių metodų esmė: tarkime, turime N dydžio laiko eilutę, tada ranginis metodas yra laiko eilutės suskirstymas į tam tikro ilgio $L \geq 2$ kombinacijas, kuriose skaičiai išdėstomi didėjimo tvarka. Įdomu tai, kad ranginiais metodais užkoduotos kombinacijos nėra tiesiog simboliai, jose slypi turimų duomenų kokybinė informacija.

Laiko eilučių kodavimą ranginiais metodais galima laikyti naudingu metodu praktiniuose skaičiavimuose dėl keleto priežasčių [1]:

- tai yra konceptualiai paprastas ir greitai pritaikomas metodas,
- kombinacijos, sudarytos remiantis nelygybėmis, yra gana "atsparios" stebėjimo triukšmui,
- šis skaičiavimo metodas nereikalauja žinoti turimų duomenų diapazono, o tai yra labai patogu naudojant realius duomenis.

Ranginės kombinacijos yra gaunamos užfiksavus kombinacijos ilgį L ir uždelsimo laiką (angl. embedding delay time) τ. Tam, kad būtų gauti patikimi rezultatai, kombinacijos ilgis privalo tenkinti sąlygą $N \gg L!$, t.y. laiko eilutės dydis turi būti daug didesnis nei visų galimų kombinacijų skaičius. C. Bandt ir B. Pompe [2] rekomenduoja kombinacijos ilgį rinktis $3 \leq L \leq 7$. Uždelsimo laikas – tai laikas, kas kelintą eilutės skaičių įtrauksime į vieną kombinaciją. Skirtingiems duomenų periodiškumams yra naudojami skirtingi uždelsimo laikai.

Prieš pateikiant darbo tikslus, įvykdytus tyrimus bei gautus rezulatatus, tam, kad būtų paprasčiau suprasti apie ką kalbama, paprastu pavyzdžiu iliustruosime, kaip praktiškai ranginiais metodais yra koduojami duomenys. Tarkime, turime laiko eilutę $x = (2, 5, 3, 1, 0, 4, 10)$

ir norime ją užkoduoti su parametrais $L = 3$ ir $\tau = 1$, tada kiekvienai iš reikšmių suteikiame simbolį – $x_0 = 2, x_1 = 5, x_3 = 3$ – pirmoji reikšmė atitiks simbolį 0, antroji – 1, trečioji – 2. Taip užkodavus reikšmes ir jas išrikiavus didėjimo tvarka yra gaunama pirmoji užkoduota kombinacija (0,2,1), už kurios "slepiasi" reikšmės (2,3,5). Antrą kombinaciją konstruojame jau nuo antro laiko eilutės dydžio, dabar $x_0 = 5, x_1 = 3, x_2 = 1$, t.y. dabar jau simboliams priskiriamos naujos reikšmės. Užkoduotos reikšmės analogiškai kaip ir pirmu atveju išrikiuojamos didėjimo tvarka ir dabar, jau antra užkoduota kombinacija yra (2,1,0), taip tęsime toliau iki paskutinės galimos 3 simbolių sekos, šiuo atveju ji užkoduojama (0,1,2). Jei $\tau = 2$, kodavimas atrodys taip: $x_0 = 2, x_1 = 3, x_2 = 0$ su kombinacija (2,0,1), toliau koduosime $x_0 = 5, x_1 = 1, x_2 = 4$. Taip tęsiama iki kol įmanoma sudaryti kombinaciją iš L (mūsų atveju – 3) simbolių. Jei koduojant laiko eilutę nutinka taip, kad du iš eilės einantys skaičiai yra lygūs, t.y. $x_k = x_j, j > k$, tada koduojamoje kombinacijoje juos rašysime tokiu eiliškumu, kaip jie išrašyti eilutėje, t.y. $\ldots, k, j, \ldots$. Šiame pavyzdyje be parametrų τ ir kombinacijos ilgio L, galima paminėti ir tai, jog koduoti eilutę galima pataškiui (kaip mūsų pavyzdžiu) – $(x_0, x_1, x_2), (x_1, x_2, x_3), \ldots$ arba blokais – $(x_0, x_1, x_2), (x_3, x_4, x_5), \ldots$ (kai reikšmės nepersidengia). Kaip teigiama [14] šaltinyje, koduojant pataškiui, gali būti susiduriama su priklausomybės struktūrų pervertinimo problema, tuo tarpu koduojant antru atveju, gali būti prarasta dalis informacijos.

Aptartų parametrų dydžių, kurie naudojami laiko eilutes koduojant ranginiais metodais, praktinis pasirinkimas buvo apžvelgtas [12] šaltinyje. Šiame straipsnyje yra surinkti ir susisteminti duomenys apie tyrimus, nagrinėjančius įvairias sferas, pasitelkiant į pagalbą ranginius metodus. Išskirtos sferos – teorinės, fizinės sistemos, medicinos, aplinkos bei ekonomikos studijos. Šiuo šaltiniu darbe yra remiamasi jau pasirinkus duomenis analizei – pagal duomenų sferos tipą, darbo analizei yra pasirenkami tokie parametrai, kaip rekomenduojama [12] šaltinyje.

Turint baigtinę laiko eilutę $\{x_t, t = 1, \ldots, N\}$ vienas pirmųjų žingsnių yra nuodugniai ją išnagrinėti tam, kad ši eilutė atskleistų svarbią informaciją, savybes, bei kad jos pagrindu būtų galima atlikti prognozes. Vienas iš naudingų įrankių tam atlikti – Shannon entropija (angl. Shannon entropy). Šio įrankio naudingumas vertinant volatilumo reiškinį finansų duomenų srityje buvo įrodytas šaltinyje [3]. Kitas įrankis, kuris gali būti naudingas nagrinėjant laiko eilutes, tai Lamberti pristatytas statistinio sudėtingumo matas (angl. statistical complexity measure, SCM). Jo pagalba galima pastebėti svarbias detales apie duomenų dinamiką [3].

Šio darbo tikslas buvo išnagrinėti 1, 5, 10 bei 30 metų trukmės JAV vyriausybinių

obligacijų pelningumą, užkoduotą ranginiais metodais. Užkoduotas finansines laiko eilutes buvo siekiama išnagrinėti kiekvieną atskirai ir pagal tam tikrus parametrus nustatyti eilučių elgesį. Šiame darbe eilutės buvo nagrinėjamos ne tik kiekviena atskirai, bet ir buvo bandoma išsiaiškinti, ar egzistuoja ryšys tarp šių laiko eilučių. Išnagrinėjus straipsnius apie duomenų kodavimą ranginiais metodais ir pastebėjus, kad iki šiol buvo nagrinėjamos tik įvairios užkoduotų duomenų savybės, tačiau nebuvo bandoma atlikti prognozių iš jau užkoduotų duomenų, šiame darbe buvo nuspręsta pabandyti prognozuoti ranginiais metodais užkoduotus duomenis, t.y. sugalvoti modelius, kuriais remiantis būtų galima prognozuoti užkoduotas finansines laiko eilutes. Galiausiai bus siekiama palyginti sukonstruotų modelių tikslumą pagal tam tikrus kriterijus.

Darbo metu buvo sukurta programa, kuri užkoduoja norimą finansinę laiko eilutę su pasirinktais parametrais. Kiekviena iš tyrimui pasirinktų (1, 5, 10, 30 metų trukmės obligacijų) finansinių laiko eilučių buvo užkoduota, remiantis šia programa ir paskui tiriama – buvo paskaičiuotas rangų blokų skirstinys bei iš užkoduotų duomenų sudarytas Markovo grandinės stacionarusis skirstinys. Šie skirstiniai buvo lyginami, remiantis pilnosios variacijos metrika. Taip pat kiekvienai iš eilučių buvo paskaičiuota keitinių entropiją, nubrėžta statistinio sudėtingumo entropijos kauzalinė plokštuma. Siekiant įvertinti, ar egzistuoja ryšys tarp eilučių, buvo paskaičiuota koreliacija, teigiama ir neigiama priklausomybė tarp kiekvienos iš nagrinėjamų užkoduotų eilučių. Antroje darbo dalyje užkoduotos finansinės laiko eilutės buvo prognozuojamos dviem modeliais – vienmačiu ir dvimačiu. Prognozuojant vienmačiu atveju prognozė yra atliekama tik iš turimos eilutės duomenų ir skaičiavimams naudojamasi Markovo grandinėmis, dvimačiu atveju – eilutė prognozuojama remiantis ne tik jos istoriniais duomenimis, bet ir kitos eilutės duomenimis (pvz. 30 metų trukmės obligacijų pelningumas prognozuojamas remiantis 1, 5 arba 10 metų trukmės obligacijų pelningumu). Pastarasis modelis buvo sukonstruotas, remiantis sąlyginėmis tikimybėmis ir pilnos tikimybes formule. Gauti abiejų modelių rezultatai buvo vertinami pagal kiekvieno iš modelių vidutinę absoliutinė procentinę paklaidą bei paklaidų (ties kiekviens kombinacija) dalį, kuri yra mažesnė arba lygi 5% (nereikšminga paklaida) bei didesnė arba lygi 95% (labai reikšminga paklaida).

Įvykdžius išsikeltus darbo tikslus, buvo nustatyta, kad kiekvienos iš užkoduotų eilučių empirinis rangų blokų skirstinys nežymiai skiriasi nuo atitinkamos Markovo grandinės stacionaraus skirstinio, o tai reiškia, kad Markovo grandinės stacionarus skirstinys neblogai apibūdina užkoduotų eilučių empirinį skirstinį. Taip pat buvo pastebėta, kad nors užkoduotos eilutės ir nėra artimos visiškai nuspėjamoms, tačiau ir nėra visiškai atsitiktinės. Nustatyta, kad egzistuoja koreliacija ir teigiama priklausomybė tarp nagrinėjamų užkoduotų eilučių,

kuri mažėja didėjant skirtumui tarp obligacijų trukmės. Be koreliacijos ir priklausomybės tarp eilučių, buvo pastebėta ir tai, kad prognozuojant eilutes dvimačiu atveju, galima gauti tikslesnius rezultatus, nei prognozuojant tas pačias eilutes vienmačiu atveju. Iš to galima daryti išvadą, kad jei yra turima duomenų apie finansinę laiko eilutę, kurią su prognozuojama eilute sieja priklausomybė bei koreliacija, tai remiantis pagalbinės eilutės duomenimis, kitą eilutę galima nuprognozuoti tiksliau, nei kad ji būtų nuprognozuota be pagalbinės eilutės.

Tolimesni skyriai bus išdėstomi tokia tvarka: 2 skyriuje pristatyta visa teorinė medžiaga, kuri yra reikalinga tolimesniam darbo nagrinėjimui. Taip pat šiame skyriuje yra aprašyta, kaip buvo sukonstruoti tyrime naudojami prognozavimo modeliai ir kriterijai, kuriais remiantis yra vertinami modeliai. 3 skyriuje yra aprašomi tyrimui naudojami duomenys bei tų duomenų užkodavimas ranginiais metodais. Be to, šiame skyriuje kiekvienai iš užkoduotų eilučių yra paskaičiuojama keitinių entropija, nubrėžiama statistinio sudėtingumo entropijos kauzalinė plokštume bei apskaičiuojama teigiama/neigiama priklausomybė ir koreliacija tarp užkoduotų finansinių laiko eilučių. Šio skyriaus 3.3, 3.4 poskyriuose pateikiami rezultatai, gauti prognozuojant užkoduotas eilutes vienmačiu ir dvimačiu atveju, taip pat nagrinėjama, kuris iš modelių ir dėl kokių priežasčių, yra tikslesnis. Paskutiniąjame skyriuje išdėstomos pagrindinės darbo išvados.

2 TEORINĖ DALIS

2.1 Matematinių dydžių apibrėžimai ir formulės

Apibrėžimas 2.1 (Statistinis dažnis, [8]). Tarkime, kad turime sąlygų kompleksą K, kurį galime realizuoti daug kartų. Kiekvieną kartą, jį realizavus, gali įvykti arba neįvykti atsitiktinis įvykis A. Pažymėkime $N_n(A)$ įvykių A skaičių, atlikus n eksperimentų. Santykis

$$\frac{N_n(A)}{n} = \nu_n(A) \tag{1}$$

yra vadinamas įvykio A ***statistiniu dažniu***.

Apibrėžimas 2.2 (Markovo grandinė, [8]). Tarkime, turime atsitiktinį procesą $\{\Omega, A, \mathbb{P}, X(t), t \in T\}$, įgyjantį reikšmes iš mačios erdvės $\{\Gamma, \mathcal{E}\}$. Laikysime $T = \{0, 1, \dots\}$, o būsenų erdvę Γ – baigtine arba skaičia. Būsenas žymėsime tiesiog natūraliaisiais skaičiais. Sakysime, kad procesas yra ***Markovo grandinė*** (tiksliau: diskrečiojo laiko Markovo grandinė), jei bet kuriam natūraliajam skaičiui n ir bet kuriems $k, j_0, j_1, \dots, j_{n-2}, j \in \Gamma$ teisingos lygybės

$$\begin{aligned} &P(X(n) = k | X(0) = j_0, X(1) = j_1, \dots \\ &\dots, X(n-2) = j_{n-2}, X(n-1) = j) = P(x(n) = k | X(n-1) = j). \end{aligned} \tag{2}$$

Remdamiesi sąlyginės tikimybės sąvoka, šias lygybes galime užrašyti šitaip:

$$P(X(n) = k | X(0), \dots, X(n-1)) = P(X(n) = k | X(n-1)).$$

(2) tikimybę vadinsime ***perėjimo iš j-osios būsenos į k-ąją būseną tikimybe*** ir žymėsime $p_{jk}^{(n)}$. Matrica

$$P^{(n)} = \begin{Vmatrix} p_{11}^{(n)} & p_{12}^{(n)} & \dots \\ p_{21}^{(n)} & p_{22}^{(n)} & \dots \\ \dots & \dots & \dots \end{Vmatrix}$$

vadinama ***perėjimo matrica***. Aišku,

$$\sum_k p_{jk}^{(n)} = 1$$

kai sumuojama pagal visas galimas būsenas. Apskritai, kiekviena kvadratinė matrica, sudaryta iš neneigiamų elementų, vadinama ***stochastine***, jei kiekviena jos eilutės elementų suma yra lygi 1.

Pažymėsime

$$P(X(0) = k) = p_k^0 \quad (k = 1, 2, \dots).$$

Šios tikimybės vadinamos ***pradinėmis tikimybėmis***. Ir čia

$$\sum_k p_k^0 = 1.$$

Apibrėžimas 2.3 (Sąlyginė tikimybė, [8])**.** Tarkime, kad turime tikimybinę erdvę $\{\Omega, A, \mathbb{P}\}$. Jei A ir E yra įvykiai ir $P(E) > 0$, tai įvykio A ***sąlygine tikimybe*** su sąlyga, kad E yra įvykęs, vadinsime:

$$\mathbb{P}(A|E) = \frac{P(A \cap E)}{P(E)}. \tag{3}$$

Apibrėžimas 2.4 (Pilnosios tikimybės formulė, [8])**.** Jei H_k yra baigtinė arba skaiti aibė įvykių, kurie kas du nesutaikomi ir

$$\bigcup_k H(k) = \Omega,$$

A – bet koks įvykis, tai ***pilnosios tikimybės formulė*** yra lygi

$$\mathbb{P}(A) = \sum_k P(H_k)P(A|H_k). \tag{4}$$

Apibrėžimas 2.5 (Stacionarusis tikimybių pasiskirstymas, [8, 10])**.** Apibrėžkime ribinę tikimybę (jei ji egzistuoja)

$$p_k^* = \lim_{n\to\infty} p_{jk}^{(n)}.$$

Tada, tikimybių pasiskirstymas $p_k^*(k = 1, 2, \dots)$ su sąlyga

$$p_k^* = \sum_j p_j^* p_{jk},$$

kur $p_{jk} = p_{jk}^{(1)}$, yra vadinamas ***stacionariuoju***. Jo prasmė šitokia: jei kuriam nors n_0 turime $p_k^{(n_0)} = p_k^*$ $(k = 1, 2, \dots)$, tai tikimybės $p_k^{(n)}$, kad laiko momentu n sistema pateks į k-ąją būseną, visiems $n \geq n_0$ yra tos pačios ir lygios $p_k^{(n)} = p_k^*$. Kitais žodžiais, stacionarus skirstinys parodo, kokią laiko dalį kiekvienoje iš būsenų, nesvarbu, kokia buvo pradinė būsena, grandinė praleidžia. Taip pat jis suteikia informacijos apie atsitiktinio proceso stabilumą. Stacionarus skirstinys randamas remiantis šia lygybe:

$$\pi' = \pi' P, \text{ t.y. } \pi'_k = \sum_j \pi'_j p_{jk}, \ \forall k = 1, 2, \dots \tag{5}$$

kitaip dar galima pasakyti, kad π' yra invariantiškas (nekintantis) matricos P atžvilgiu.

Apibrėžimas 2.6 (Teigiamai grįžtamoji Markovo grandinės būsena, [4]). Tegul τ_{ii} žymi laiką, per kurį grandinė pirmąkart grįžta į būseną i, kai $X_0 = i$:

$$\tau_{ii} = \min\{n \geq 1 : X_n = i | X_0 = i\}. \tag{6}$$

$\tau_{ii} = +\infty$, jei $X_n \neq i$ kiekvienam $n \geq 1$ (tada aibė, iš kurios imamas minimumas, yra tuščia). i–oji būsena yra vadinama *grįžtamąja*, jei:

$$\mathbb{P}(\tau_{ii} < \infty) = 1, \tag{7}$$

kitu atveju būsena vadinama *pereinamąja*.
i–oji grįžtamoji būsena vadinama ***teigiamai grįžtamoji***, jei:

$$\mathbb{E}[\tau_{ii}] < \infty, \tag{8}$$

t.y. vidutinis žingsnių skaičius iš i–osios būsenos sugrįžti į i–ąją yra baigtinis.

Apibrėžimas 2.7 (Suskaidoma ir nesuskaidoma Markovo grandinė [8]). Sakoma, kad k–oji būsena yra *pasiekiama* iš j–osios būsenos, jei kuriam nors sveikajam teigiamam n tikimybė iš j–osios būsenos patekti į k–ąją per n laiko tarpų yra teigiama: $p_{jk}^{(n)} > 0$.
Esminę būseną galima apibrėžti taip: j–oji yra esminė, jei ji pasiekiama iš kiekvienos būsenos, kuri yra pasiekiama iš j–osios.
Dvi esminės būsenos vadinamos *susisiekiančiomis*, jei kiekviena iš jų yra pasiekiama iš kitos. Grandinė, sudaryta iš vienos klasės esminių susisiekiančių būsenų, vadinama ***nesuskaidoma***. Jei grandinė yra sudaryta iš daugiau kaip vienos klasės būsenų, tai ji vadinama ***suskaidoma***.

Teorema 2.1. [5] Jei Markovo grandinė yra nesuskaidoma ir teigiamai grįžtamoji, tada ši grandinė turi vienintelį stacionarųjį tikimybių pasiskirstymą.

Apibrėžimas 2.8 (Pilnosios variacijos metrika, [6]). Atstumas tarp dviejų Markovo grandinių yra matas, parodantis skirtumus tarp grandinių. Kraštutiniai atvejai – kai grandinės visiškai lygios, skirtumas yra lygus 0, jei visiškai skirtingos – 1. ***Pilnosios variacijos metrika*** yra standartinis skirtumo įvertinimas tarp dviejų tikimybinių skirstinių. Turint du tikimybinius skirstinius π_1 ir π_2 iš tos pačios skaičios aibės Ω, pilnosios variacijos metriką galima apskaičiuoti taip:

$$2d(\pi_1, \pi_2) = ||\pi_1 - \pi_2||_1 := \sum_{x \in \Omega} |\pi_1(x) - \pi_2(x)|. \tag{9}$$

Apibrėžimas 2.9 (Keitinių entropija, [2,11,16]). Tarkime, turime laiko eilutę $\{x_t\}_{t=1,\dots,T}$. Mes nagrinėsime visas galimas (jų bus $n!$) perstatas π, kurių ilgis n. Kiekvienai unikaliai kombinacijai mes paskaičiuojame santykinį dažnį

$$p(\pi) = \frac{\#\{t|0 \le t \le T-n, (x_{t+1},\dots,x_{t+n}) \text{ yra tipo } \pi\}}{T-n+1}. \tag{10}$$

Šiuo būdu kaip įmanoma geriausiai yra įvertinamas baigtinės aibės reikšmių dažnis.

n – eilės ***keitinių entropija***, kai $n \ge 2$ yra apibrėžiama kaip Shannon entropija:

$$H(n) = -\sum p(\pi)\log p(\pi) \tag{11}$$

čia sumuojama pagal visas užkoduotas unikalias reikšmes. Žinoma, kad $0 \le H(n) \le \log n!$ [2].

Apatinis rėžis (0) apibūdina ranginiais metodais užkoduotą duomenų seką, kurioje su tikimybe 1 visada pasirodo tik viena iš visų galimų kombinacijų. Viršutinis rėžis ($\log n!$) parodo, kad visos užkoduotos kombinacijos atsiranda su tokia pačia tikimybe ($\frac{1}{n!}$, t.y. nė vienos iš kombinacijų tikimybė pasirodyti nėra didesnė nei kitų). Jei duomenys koduojami dvejetainiais skaičiais, logaritmo pagrindas skaičiavimuose paprastai yra lygus 2, kitu atveju skaičiavimams naudojamas natūrinis logaritmas.

Apibrėžimas 2.10 (Statistinio sudėtingumo matas, [16]). ***Statistinio sudėtingumo matas*** yra apibrėžiamas taip:

$$C_{JS}[P] = Q_j[P,P_e]H_s[P] \tag{12}$$

čia $P = \{p_i : i = 1,\dots,M\}$ – diskretus tikimybinis skirstinys, $Q_j[P,P_e] = Q_0 J[P,P_e]$,

$J[P,P_e] = S[(P+P_e)/2] - S[P]/2 - S[P_e]/2$ – Jensen-Shannon divergencija,

$Q_0 -$ normuojanti konstanta, lygi atvirkštinei didžiausiai $J[P,P_e]$ reikšmei,

$P_e = \{\frac{1}{M},\dots,\frac{1}{M}\}, H_s[P] = \frac{S[P]}{S_{\max}}, 0 \le H_s \le 1,$

$S_{\max} = S[P_e] = \ln M, M-$visų galimų kombinacijų skaičius.

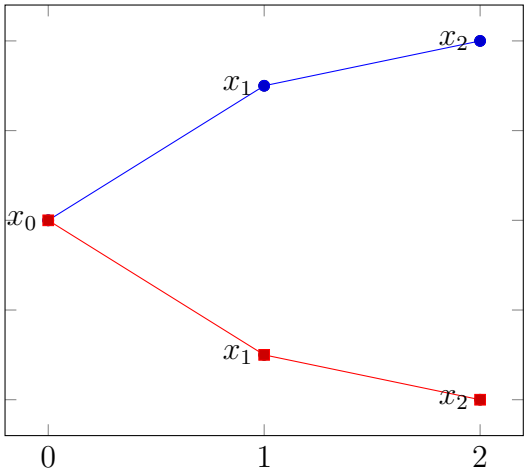
Apibrėžimas 2.11 (Priklausomybė tarp ranginiais metodais užkoduotų laiko eilučių, [14]). Laiko eilutės X ir Y turi ***teigiamą*** $h \in \mathbb{N}$ eilės ir $\alpha > 0$ lygio ranginiais metodais užkoduotų laiko eilučių priklausomybę (ord$\oplus$), jei:

$$\begin{aligned} &P(\Pi(X_n, X_{n+1},\dots,X_{n+h}) = \Pi(Y_n, Y_{n+1},\dots,Y_{n+h})) \\ &\quad > \alpha + \sum_{\pi\in S_h} P(\Pi(X_n, X_{n+1},\dots,X_{n+h}) = \pi)\cdot P(\Pi(Y_n, Y_{n+1},\dots,Y_{n+h}) = \pi) \end{aligned} \tag{13}$$

ir atitinkamai ***neigiamą*** (ord$\ominus$) $h \in \mathbb{N}$ eilės ir $\beta > 0$ lygio priklausomybę, jei

$$\begin{aligned} &P(\Pi(X_n, X_{n+1},\dots,X_{n+h}) = m(\Pi(Y_n, Y_{n+1},\dots,Y_{n+h}))) \\ &\quad > \beta + \sum_{\pi\in S_h} P(\Pi(X_n, X_{n+1},\dots,X_{n+h}) = \pi)\cdot P(m(\Pi(Y_n, Y_{n+1},\dots,Y_{n+h})) = \pi), \end{aligned} \tag{14}$$

čia Π – atvaizdis, duomenų taškams priskiriantis rangų kombinaciją, π – žymi elementus iš visų galimų kombinacijų aibės S_h, $m(\pi)$ – žymi atspindėtą (angl. reflected) kombinaciją. Žemiau pateiktame paveikslėlyje pavaizduotas kombinacijos (0,1,2) atspindys (2,1,0), o šios kombinacijos atspindys yra pradinė kombinacija.



Apibrėžimas 2.12 (Neparametriniai (nuo modelio nepriklausantys) įverčiai teigiamai (ord$\oplus$) α lygio ir neigiamai (ord$\ominus$) β lygio priklausomybei, [14]). Pažymėkime

p_{eq} yra tikimybės $\mathbb{P}(\Pi(X_n, X_{n+1}, \ldots, X_{n+h}) = \Pi(Y_n, Y_{n+1}, \ldots, Y_{n+h}))$ įvertis,

p_{neq} yra tikimybės $\mathbb{P}(\Pi(X_n, X_{n+1}, \ldots, X_{n+h}) = m(\Pi(Y_n, Y_{n+1}, \ldots, Y_{n+h})))$ įvertis,

p_π^X yra tikimybės $\mathbb{P}(\Pi(X_n, X_{n+1}, \ldots, X_{n+h}) = \pi)$ įvertis visiems $\pi \in S_h$,

čia S_h žymi aibę visų galimų kombinacijų, tada

$$\begin{aligned} \widetilde{\alpha} &:= p_{eq} - \sum_{\pi \in S_h} p_\pi^X \cdot p_\pi^Y \\ \widetilde{\beta} &:= p_{neq} - \sum_{\pi \in S_h} p_\pi^X \cdot p_{m(\pi)}^Y \end{aligned} \tag{15}$$

yra ord$\oplus$ α lygio ir ord$\ominus$ β lygio ***neparametriniai įverčiai***. Šie įverčiai parodo, kiek tikimybė yra nutolusi nuo to atvejo, jei būtų priimta nepriklausomumo tarp eilučių hipotezė, t.y. jei įvertis lygus 0 – tai reiškia, kad eilutės nepriklausomos.

Apibrėžimas 2.13 (Statistinio sudėtingumo entropijos kauzalinė plokštuma, [13]). ***Statistinio sudėtingumo entropijos kauzalinė plokštuma*** – tai erdvė, kurios horizontalioji ir vertikalioji ašys yra tikimybių pasiskirstymo funkcijos, atitinkamai keitinių entropija ir statistinio sudėtingumo matas (angl. statistical complexity measure).
Ši diagrama skirta nustatyti deterministinės ir stochastinės kilmės laiko eilutėms. Visiškai atsitiktinis stochastinis procesas plokštumoje bus taške $H_s = 1$ ir $C_{JS} = 0$.

Apibrėžimas 2.14 (Vidutinė absoliutinė procentinė paklaida (MAPE), [15]). ***MAPE*** (angl. mean absolute percentage error) yra statistinis matas, kuris skirtas įvertinti prognozės tikslumui. MAPE yra apskaičiuojamas taip:

$$M = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right| \cdot 100\%, \tag{16}$$

čia A_t – faktiniai duomenys, F_t – prognozuojami duomenys. Kiekvienam laiko momentui yra paskaičiuojama absoliutinė procentinė paklaida, o tada iš visų laiko momentų paskaičiuojamas vidurkis.

2.2 Prognozavimo būdas vienmačiu atveju

1 skyriuje aprašytu metodu užkoduoti duomenys darbo metu yra prognozuojami dviem būdais – šiame skyrelyje yra aprašomas darbe naudojamas vienmatinis prognozavimo būdas, sekančiame skyrelyje bus aprašytas dvimatis prognozavimo būdas.

Užkoduotų eilučių prognozavimui vienmačiu atveju yra naudojamasi Markovo grandinėmis. Iš visos užkoduotos duomenų eilutės yra sudaromas empirinis rangų blokų skirstinys, kuris parodo, su kokia tikimybe kiekviena iš kombinacijų pasirodo faktiškai. Šie faktiniai duomenys (skirstinys sudarytas iš 100% duomenų) yra naudojami palyginimui su prognozuojamomis kiekvienos kombinacijos pasirodymo tikimybėmis. Vienu atveju prognozuojama yra remiantis 80% visų turimų duomenų, kitu atveju – 50% visų turimų duomenų.

Prognozei naudojamo vienmačio modelio principas yra toks: naudojantis matematinės statistikos paketu R (toliau – R paketas) iš tam tikros dalies (atitinkamai 80% arba 50%) užkoduotų duomenų (duomenims užkoduoti buvo sukurta programa, kurios programinį kodą galima rasti A.2 priede) yra sudaroma Markovo grandinė bei tos grandinės tikimybių perėjimo matrica. Taip pat, šiai grandinei (iš tam tikros dalies duomenų) yra paskaičiuotas stacionarus skirstinys, siekiant įvertinti, kurioje iš kombinacijų galimai bus grandinė ilgalaikėje perspektyvoje. Tada, nustačius, ties kuria kombinacija stacionariame skirstinyje yra didžiausia tikimybė, yra tariama, kad pradinių tikimybių vektoriuje tos kombinacijos pasirodymo tikimybė lygi 1.

Turint iki šiol aprašytus skaičius galima prognozuoti: pradinių tikimybių vektorius yra dauginamas iš perėjimo tikimybių matricos (sudarytos iš tam tikros dalies duomenų) ir yra gaunama prognozė, su kokia tikimybe kiekviena iš kombinacijų pasirodys po vieno žingsnio. Norint paskaičiuoti prognozę po dviejų žingsnių, gautas tikimybių vektorius, kurio ilgis toks, kiek yra skirtingų kombinacijų, vėl dauginamas iš perėjimo tikimybių matricos (tos pačios

kaip ir prieš tai) ir vėl yra gaunamas tikimybių vektorius, kuris parodo su kokiomis tikimybėmis pasirodys kiekviena iš kombinacijų po dviejų žingsnių. Taip dauginama tiek kartų, kiek žingsnių į priekį norima prognozuoti. Šiame darbe nagrinėjamu atveju, šis veiksmas bus atliekamas tiek kartų, iki kol bus pasiektas realiai turimų žingsnių skaičių, t.y. pavyzdžiui, jei turime Markovo grandinę sudarytą iš 100 žingsnių, ir prognozavimui naudojame 80% duomenų, tai prieš tai aprašytą pradinių tikimybių vektorių dauginsime iš tikimybių perėjimo matricos (sudarytos iš tų 80 % duomenų) 20 kartų.

2.3 Prognozavimo būdas dvimačiu atveju

Šiame skyrelyje bus aprašytas darbo metu sukurtas dvimatis modelis, skirtas užkoduotų duomenų prognozavimui, kurio esmė turint 100% informacijos apie vieną užkoduotą finansinę laiko eilutę (toliau – pirmoji eilutė) ir turint tam tikrą dalį (80% arba 50%) informacijos apie kitą finansinę laiko eilutę (toliau – antroji eilutė), prognozuoti antrąją laiko eilutę.

Tarkime, dvi skirtingos finansinės laiko eilutės yra užkoduotos ranginiais metodais n – ilgio blokais (sudarytais iš skirtingų simbolių). Visas įmanomas skirtingas kombinacijas pažymėkime kombinacija nr. 1, kombinacija nr. 2, ..., kombinacija nr. $n!$. Tada prognozei naudojamo dvimačio modelio principas yra toks: pirmiausia, siekiant išsiaiškinti, kaip elgiasi antroji eilutė, žinant pirmosios eilutės elgesį, yra skaičiuojamos sąlyginės tikimybės. Jei žinoma, kad pirmojoje eilutėje pasirodys tam tikra kombinacija, pažymėkime ją kombinacija nr. 1, tada yra skaičiuojamos tikimybės, kad tuo pačiu laiko momentu antroje laiko eilutėje pasirodys kombinacijos nr. $1, 2, 3, \ldots, n!$ ir gaunama $n!$ skirtingų tikimybių, kurių suma lygi 1. Toliau tariama, kad pirmoje eilutėje pasirodys kombinacija nr. 2 ir skaičiuojamos tikimybės, kad tuo pačiu laiko momentu, kai pirmoje eilutėje yra kombinacija nr. 2, antroje laiko eilutėje bus kombinacijos $1, 2, 3, \ldots, n!$. Taip tęsiant toliau iš viso bus gauta $(n!)^2$ sąlyginių tikimybių, kurias galima pavaizduoti matrica:

$$\begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n!} \\ p_{21} & p_{22} & \cdots & p_{2n!} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n!1} & p_{n!2} & \cdots & p_{n!n!} \end{pmatrix} \tag{17}$$

čia p_{ij} – tikimybė iš pirmosios eilutės i – osios kombinacijos pereiti į antrosios eilutės j – ąją kombinaciją, kai $i, j = 1, \ldots, n!$.

Sąlyginės tikimybės yra skaičiuojamos siekiant išsiaiškinti, su kokiomis tikimybėmis kiekviena iš kombinacijų (kurių iš viso yra $n!$) pasirodys antrojoje eilutėje, jei yra žinoma, kad

tam tikra kombinacija pasirodė pirmojoje eilutėje. Svarbu paminėti, kad skaičiuojant sąlygines tikimybės, pirmosios eilutės duomenų yra imama ne 100%, o tiek pat, kiek yra turima antrosios eilutės duomenų (80% arba 50%), t.y. sąlyginės tikimybės skaičiuojamos iš dviejų vienodo ilgio laiko eilučių.

Toliau yra pasinaudojama visu 100% pirmosios eilutės duomenų ir suskaičiuojama kiekvienos kombinacijos pasirodymo tikimybė:

$$(p_1^1, p_2^1, \ldots, p_{n!}^1) \tag{18}$$

čia p_i^1 yra i – osios kombinacijos tikimybė pasirodyti pirmojoje eilutėje laiko momentu t, kai $i = 1, \ldots, n!$.

Dabar, kai jau yra žinoma, su kokia tikimybe kiekviena iš kombinacijų pasirodys pirmojoje eilutėje laiko momentu t ir turime matricą, sudarytą iš dalies pirmosios ir antrosios eilutės duomenų, kuri parodo, su kokia tikimybe kiekviena iš kombinacijų pereina iš pirmosios eilutės į antrąją, galima pritaikyti pilnos tikimybės formulę ir tokiu būdu gauti prognozę, su kokiomis tikimybėmis kiekviena iš kombinacijų pasirodys antrojoje eilutėje laiko momentu t. Matematiškai tai atrodys taip:

$$\begin{pmatrix} p_1^1 & p_2^1 & \ldots & p_{n!}^1 \end{pmatrix} \begin{pmatrix} p_{11} & p_{12} & \ldots & p_{1n!} \\ p_{21} & p_{22} & \ldots & p_{2n!} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n!1} & p_{n!2} & \ldots & p_{n!n!} \end{pmatrix} = \begin{pmatrix} p_1^2 & p_2^2 & \ldots & p_{n!}^2 \end{pmatrix} \tag{19}$$

čia p_i^2 yra i – osios kombinacijos tikimybė pasirodyti antrojoje eilutėje laiko momentu t, kai $i = 1, \ldots, n!$.

2.4 Vertinimo metodas

Prognozuojant užkoduotas finansines laiko eilutes tiek vienmačiu (2.2 skyrelyje aprašytu būdu), tiek dvimačiu (2.3 skyrelyje aprašytu būdu) atveju, galiausiai turime du skirstinius, parodančius kokia laiko dalis praleidžiama ties kiekviena kombinacija, – vienas realus, sudarytas iš 100% turimų duomenų (su kokia tikimybe kiekviena iš kombinacijų pasirodys laiko momentu t faktiškai), kitas – gautas prognozuojant iš dalies duomenų – iš 80% arba 50% duomenų (šiuo atveju yra gaunama, su kokia tikimybe kiekviena iš kombinacijų pasirodys laiko momentu t pagal prognozę). Empirinis rangų blokų skirstinys ir nuprognozuotas skirstinys bus lyginami, remiantis tokiais kriterijais:

- apskaičiuojant modifikuotą modelio vidutinę absoliutinę procentinę paklaidą (MAPE),

- įvertinant, kokia dalis visų galimų kombinacijų paskaičiuotų absoliutinių procentinių paklaidų yra mažesnės arba lygios 5% bei kokia dalis paklaidų yra didesnės arba lygios 95%.

Vertinant prognozavimo modelius pirmu iš išvardytų būdų, kaip jau paminėta, MAPE bus šiek tiek modifikuotas pagal darbo specifiką, tai reiškia, kad bus lyginamos atitinkamų kombinacijų tikimybės pasirodyti – kiekvienai iš kombinacijų bus paskaičiuota absoliutinė procentinė paklaida, o kaip bendrą modelio paklaidą laikysime visų kombinacijų absoliutinių procentinių paklaidų vidurkį, t.y.:

$$M = \frac{1}{n!} \sum_{i=1}^{n!} \left| \frac{A_i - F_i}{A_i} \right| \cdot 100\%, \tag{20}$$

čia n – kombinacijų bloko ilgis, čia $n \geq 2$,
i – kombinacijos numeris, $i = 1, 2, \ldots, n!$,
A_i – i – osios kombinacijos pasirodymo tikimybė faktiškai,
F_i – i – osios kombinacijos pasirodymo tikimybė, gauta prognozuojant.

Vertinant prognozavimo modelius antru iš išvardytų būdų, kiekvienai iš kombinacijų bus paskaičiuota absoliutinė procentinė paklaida:

$$m_i = \left| \frac{A_i - F_i}{A_i} \right| \cdot 100\%, \tag{21}$$

i – kombinacijos numeris, $i = 1, 2, \ldots, n!$,
A_i – i – osios kombinacijos pasirodymo tikimybė faktiškai,
F_i – i – osios kombinacijos pasirodymo tikimybė, gauta prognozuojant.

Tada, skaičiuojama, kokia dalis visų kombinacijų absoliutinių procentinių paklaidų yra mažesnės arba lygios 5%:

$$level_{0,05} = \frac{1}{n!} \sum_{i=1}^{n!} \mathbb{1}_{\{m_i \leq 0,05\}}, \tag{22}$$

ir kokia dalis visų kombinacijų absoliutinių procentinių paklaidų yra didesnės arba lygios 95%:

$$level_{0,95} = \frac{1}{n!} \sum_{i=1}^{n!} \mathbb{1}_{\{m_i \geq 0,95\}}. \tag{23}$$

Tarsime, kad jei kombinacijos absoliutinė procentinė paklaida yra mažesnė arba lygi 5%, tai paklaida nereikšminga, kitu atveju, kai paklaida didesnė arba lygi 95%, prognozė visiškai netiksli.

3 PRAKTINĖ DALIS

3.1 Duomenų pasirinkimas

Vienas pirmųjų darbų naudojant ranginių metodų analizę buvo elektroencefalogramos analizė, pacientų, sergančių epilepsija [7]. Nors ranginiai metodai yra plačiai naudojami biologijoje bei medicinoje [1], tačiau jau galima rasti ir darbų, kuriuose ranginiais metodais koduojamos bei analizuojamos finansinės laiko eilutės. Vienas tokių darbų – tai L. Zunino, A. Fernandez Bariviera, M. Bel en Guercio, L.B. Martinez, O.A. Rosso straipsnis [16]. Kaip teigiama pastarajame šaltinyje, kalbant apskritai apie finasines laiko eilutes, akcijų rinkos įvairiais metodais yra žymiai plačiau nagrinėjamos nei fiksuotų pajamų rinkos (pvz. obligacijos). Pasak [16] šaltinio, būtina užpildyti šią spragą literatūroje su tikslu padėti ne tik investuotojams, bet ir obligacijų leidėjams. Šiame šaltinyje yra išnagrinėti 30 išsivysčiusių ir besivystančių ekonomikos valstybių obligacijų indeksai, naudojant inovatyvią statistinę priemonę – statistinio sudėtingumo entropijos kauzalinę plokštumą. Taip pat šaltinyje yra rasta koreliacija tarp keitinių entropijos, ekonomikos vystymosi ir rinkos dydžio, kas galėtų būti įdomu obligacijų leidėjams ir investuotojams.

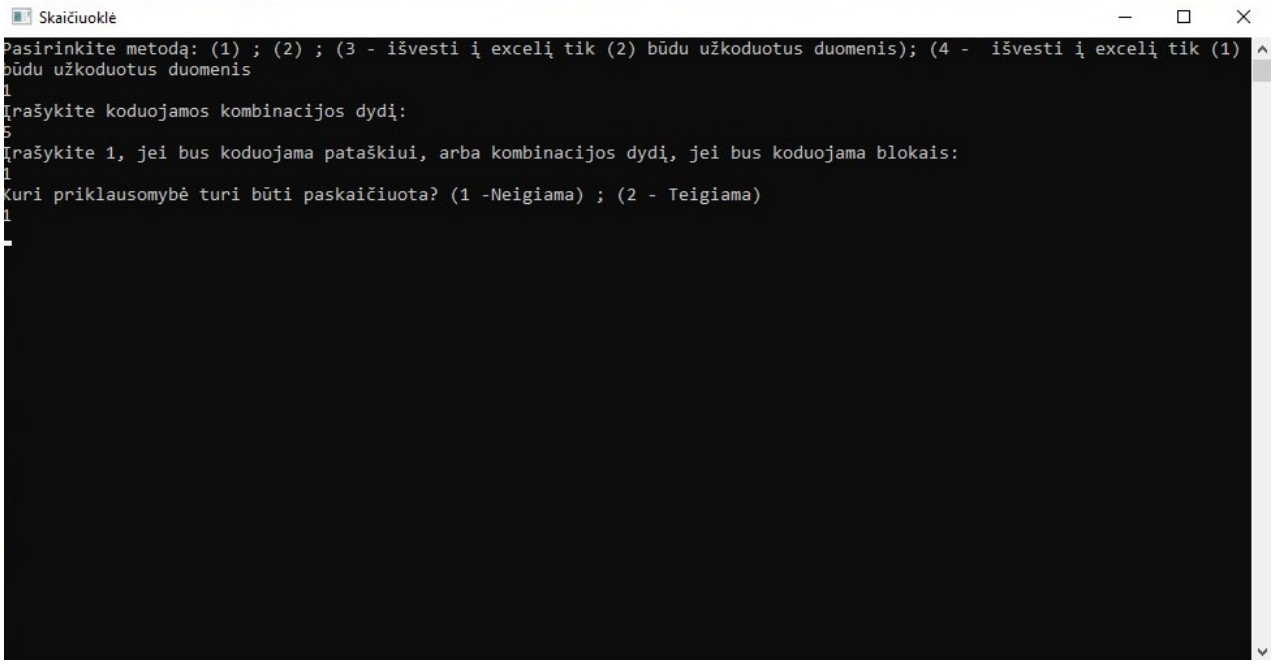
Remiantis argumentais, paminėtais nagrinėtame straipsnyje, kad fiksuotų rinkų obligacijos literatūroje apskritai yra mažiau analizuojamos, šiame darbe buvo nuspręsta pasirinkti 1, 5, 10 ir 30 metų trukmės JAV vyriausybinių obligacijų pelningumą (angl. yield) nuo 2006-02-09 iki 2019-10-31 (dienų tikslumu). Iš [9] šaltinyje pateiktų duomenų buvo sukonstruotos 4 laiko eilutės, kiekviena eilutė – 3454 ilgio. Šios 1, 5, 10 ir 30 metų trukmės obligacijų eilutės, kurios atspindi obligacijų pelningumą minėtoms trukmėms, grafiškai pavaizduotos 1 paveikslėlyje.

3.2 Duomenų užkodavimas ranginiais metodais ir analizė

Pirmiausia, darbui palengvinti, buvo parašyta programa (su *C sharp*), kuriai yra paduodamos dvi laiko eilutės. Tada šioje programoje galima pasirinkti, kokio rezultato norima: apskaičiuotos priklausomybės tarp užkoduotų laiko eilučių 1 būdu (kai ranginiais metodais koduojama su h skirtingų simbolių), 2 būdu (kai ranginiais metodais koduojama dviem simboliais – 0 (kai kaina leidosi) ar 1 (kai kaina kilo)), 3 pasirinkimas – tai 2 būdu užkoduotų laiko eilučių išvedimas į *Microsoft Office Excel* (toliau – Excel) bei visų galimų tokio ilgio kombinacijų išvedimas, 4 pasirinkimas – analogiškas 3, tik viskas išvedama naudojant 1 kodavimo metodą. Programoje taip pat reikia įrašyti, kokio dydžio vektoriais koduosime (2, 3,

1 pav.: Skirtingos trukmės JAV vyriausybinių obligacijų pelningumo kitimas

..., n) ir kiek duomenų peršoksime po kiekvieno užkodavimo (koduosime persidengiančius duomenis, ar koduosime blokais). Galiausiai reikia pasirinkti, kurią priklausomybę norima skaičiuoti – teigiamą ar neigiamą. Konsolinis programos langas pavaizduotas 2 paveikslėlyje.



2 pav.: Konsolinis programos langas

Parametras τ sąmoningai nėra įvedamas, kaip pasirenkamas parametras, todėl, nes išnagrinėjus [12] šaltinį, kuriame remiamasi ekspertų, iki šiol dirbusių su ranginiais metodais ir panašaus tipo duomenimis, patirtimi, parametras τ buvo pasirinktas 1 – kaip nekintantis parametras. Tai darbai, kuriuose buvo nagrinėti pasaulio akcijų indeksai, 32 šalių akcijų rinkų indeksai, kasdienės obligacijų indeksų vertės. Šiuose darbuose laiko eilučių dydžiai svyravo nuo 2047 iki maždaug 3138, nagrinėjamų rangų dydis buvo pasirinktas nuo 4 iki 6, o parametras $\tau = 1$. Kadangi mūsų nagrinėjami duomenys – įvairios trukmės kasdienis obligacijų pelningumas – glaudžiai susiję su nagrinėtomis laiko eilutėmis, tai būtent dėl šios priežasties parametras τ pasirinktas kaip fiksuotas dydis, lygus 1. Kaip jau buvo paminėta, [2] šaltinyje koduojamų kombinacijų ilgį siūloma rinktis nuo 3 iki 7, [12] šaltinyje, jau atsižvelgus į duomenų pobūdį (finansiniai duomenys) bei laiko eilutės dydį ($\approx$ 2000 – 3000 duomenų), panašiuose darbuose nagrinėjamų kombinacijų dydis sumažintas iki 4–6, vietoje 3–7, todėl šiame darbe iš pradžių taip pat bus nagrinėjamos 4–6 ilgio kombinacijos, be to, koduojama bus pataškiui.

Pirmiausia, 1, 5, 10 ir 30 metų trukmės obligacijų pelningumas padieniui buvo užkoduotos 1 būdu pataškiui ir gauta Markovo grandinė su $n!$ būsenų (čia n – kombinacijų, kuriomis koduosime, ilgis). Taip buvo koduojama su 4, 5, 6 dydžio kombinacijomis, kai 0123 reiškia

vis didėjančią grąžą, o 3210 visas keturias iš eilės dienas mažėjančias grąžas (atitinkamai 01234, 43210 ir 012345, 543210). Kiekvienai iš užkoduotų eilučių buvo paskaičiuotas empirinis rangų blokų skirstinys, t.y. buvo apskaičiuota, kokią dalį laiko yra būnama konkrečioje būsenoje. Taip pat R paketo pagalba kiekvienai iš užkoduotų eilučių buvo sudarytos Markovo grandinės ir perėjimo tikimybių matricos. Remiantis 2.5 apibrėžimu buvo paskaičiuoti ir sudarytų Markovo grandinių stacionarūs skirstiniai. Kad kiekvienai iš Markovo grandinių egzistuoja vienintelis ir unikalus stacionarusis skirstinys, buvo užtikrinta, pritaikius 2.1 teoremą, t.y. kiekvienai iš sudarytų grandinių su R paketu buvo patikrintos savybės – ar grandinė nesuskaidoma (2.7 apibrėžimas) ir ar ji teigiamai grįžtamoji (2.6 apibrėžimas – R paketu patikrinta, ar visos būsenos yra grįžtamosios ir tada, ar vidutinis žingsnių skaičius iš i–osios būsenos sugrįžti į i–ąją būseną yra baigtinis). Kadangi visos grandinės tenkina abi savybes (nesuskaidomumo ir teigiamo grįžtamumo), tai remiantis 2.1 teorema, visos nagrinėjamos grandinės turi unikalius stacionarius skirstinius. Šie skirstiniai grafiškai atvaizduoti 3, 4, 5 paveikslėliuose.

Tarp atitinkamų empirinių rangų blokų skirstinių ir sudarytų Markovo grandinių stacionarių skirstinių buvo paskaičiuota pilnosios variacijos metrika. Gauti rezultatai pateikti lentelėje 1.

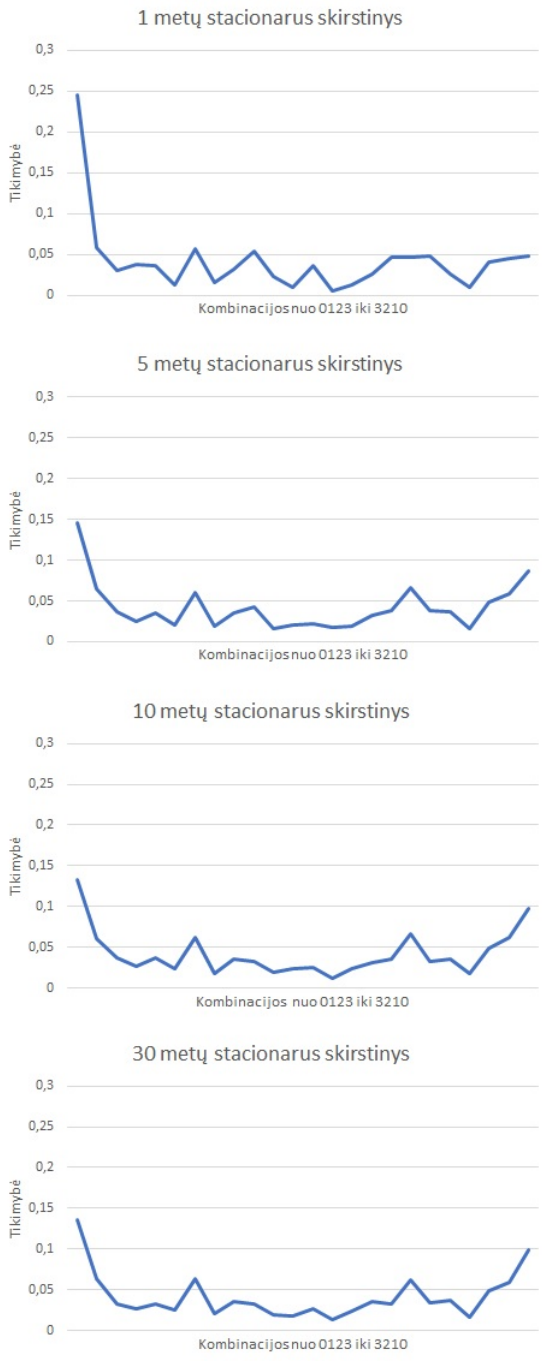
	Obligacijų trukmė			
Bloko dydis	1	5	10	30
4	0,0007	0,0006	0,0006	0,0007
5	0,0008	0,0008	0,0009	0,0008
6	0,0014	0,0012	0,0013	0,0011

1 lentelė: Pilnosios variacijos metrika tarp pradinio tikimybinio (užkoduotų eilučių rangų blokų) ir sudarytos Markovo grandinės stacionaraus skirstinio, kai bloko dydis 4–6

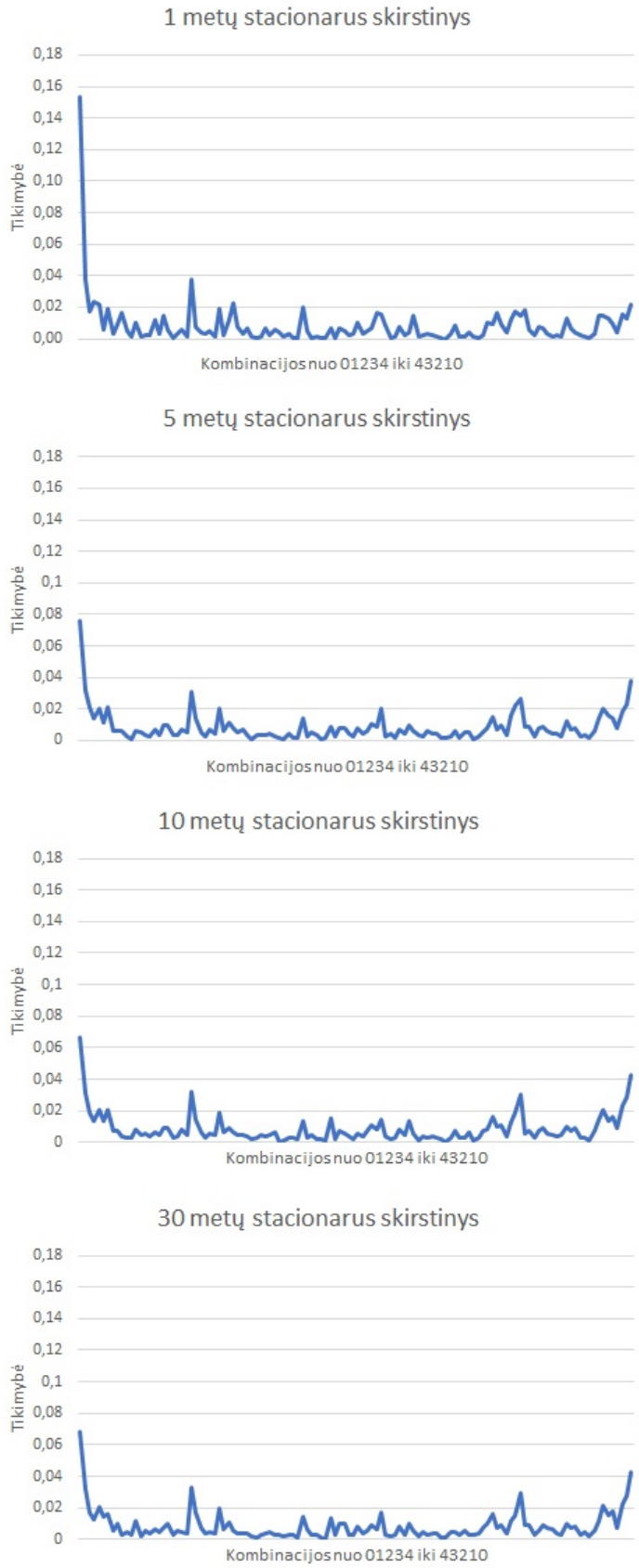
1 lentelėje pateikti rezultatai parodo, kad pilnosios variacijos metrika visais atvejais yra artima 0, o tai reiškia, kad kiekvienos trukmės obligacijų empiriniai ir ribiniai stacionarūs skirstiniai yra beveik lygūs, kitaip sakant, atitinkamos rangų blokų tikimybės, kai jas skaičiuojame empiriškai, iš esmės atitinka gautos Markovo grandinės stacionaraus skirstinio tikimybes. Šiuo požiūriu Markovo grandinė gerai aproksimuoja gautas empirines rangų blokų tikimybes. Toliau panagrinėsime koreliaciją tarp neužkoduotų įvairios trukmės obligacijų pelningumo duomenų ir koreliaciją tarp užkoduotų eilučių.

Paskaičiavus Pearson koreliacijos koeficientus tarp pasirinktų (neužkoduotų) laiko eilučių

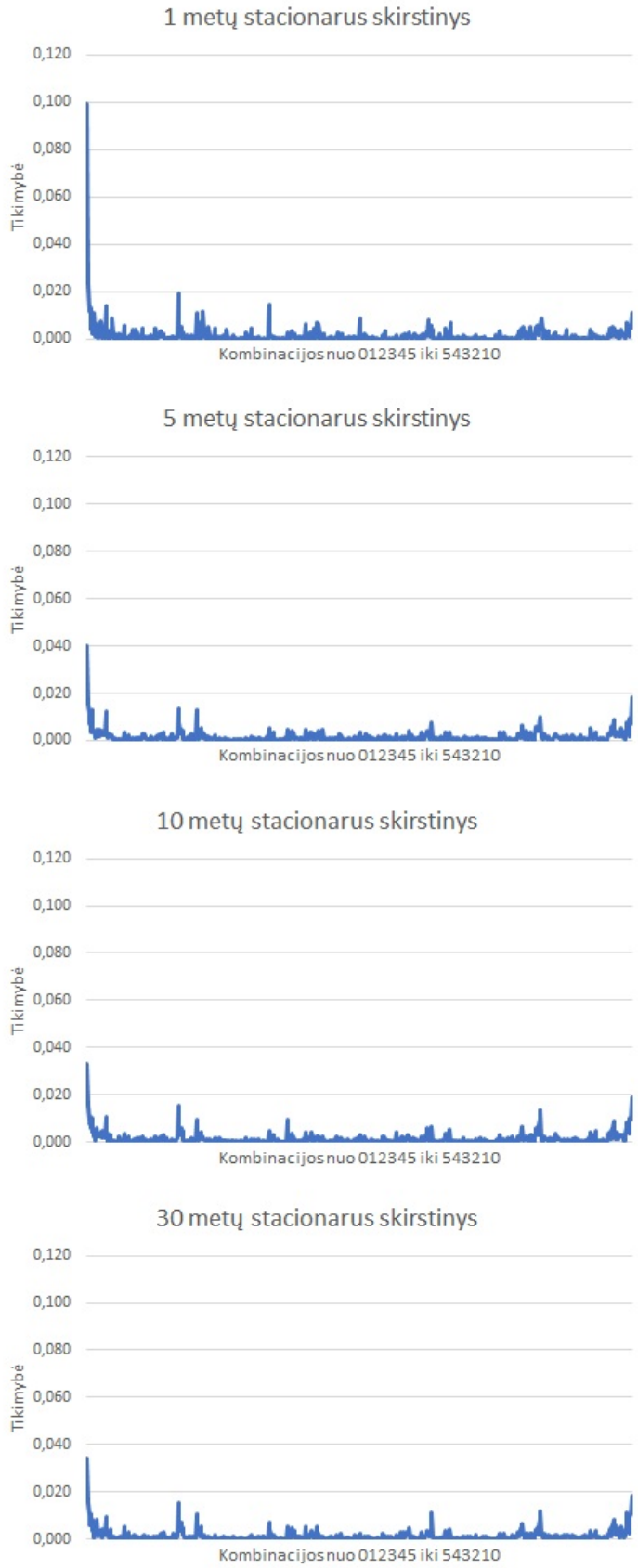
(žr. 2 lentelę), galima pastebėti, kad didžiausia priklausomybė yra tarp 1 ir 5, 5 ir 10 bei 10 ir 30 metų trukmės obligacijų pelningumo. Didėjant skirtumui tarp obligacijų trukmės, koreliacija tarp duomenų mažėja. Pažiūrėkime, ar tą patį galima pasakyti apie koreliaciją tarp užkoduotų eilučių rangų blokų skirstinių (žr. 3, 4, 5 lenteles).



3 pav.: Markovo grandinės, sudarytos iš užkoduotų 1, 5, 10, 30 metų trukmės obligacijų pelningumo duomenų, stacionarus skirstinys, užkodavus duomenis 4 simbolių vektoriais. X ašyje – visos galimos $4! = 24$ kombinacijos, Y ašyje – jų pasirodymo tikimybės.



4 pav.: Markovo grandinės, sudarytos iš užkoduotų 1, 5, 10, 30 metų trukmės obligacijų pelningumo duomenų, stacionarus skirstinys, užkodavus duomenis 5 simbolių vektoriais. X ašyje – visos galimos $5! = 120$ kombinacijos, Y ašyje – jų pasirodymo tikimybės.



5 pav.: Markovo grandinės, sudarytos iš užkoduotų 1, 5, 10, 30 metų trukmės obligacijų pelningumo duomenų, stacionarus skirstinys, užkodavus duomenis 6 simbolių vektoriais. X ašyje – visos galimos $6! = 720$ kombinacijos, Y ašyje – jų pasirodymo tikimybės.

Obligacijų trukmė	1	5	10	30
1	1	0,923	0,759	0,525
5	0,923	1	0,930	0,747
10	0,759	0,930	1	0,934
30	0,525	0,747	0,934	1

2 lentelė: Koreliacija tarp neužkoduotų duomenų laiko eilučių

Obligacijų trukmė	1	5	10	30
1	1	0,8925	0,834	0,844
5	0,8925	1	0,986	0,985
10	0,834	0,986	1	0,996
30	0,844	0,985	0,996	1

3 lentelė: Koreliacija tarp empirinių rangų blokų skirstinių, kai bloko dydis 4

Obligacijų trukmė	1	5	10	30
1	1	0,875	0,816	0,822
5	0,875	1	0,977	0,972
10	0,816	0,977	1	0,985
30	0,822	0,972	0,985	1

4 lentelė: Koreliacija tarp empirinių rangų blokų skirstinių, kai bloko dydis 5

Obligacijų trukmė	1	5	10	30
1	1	0,831	0,763	0,773
5	0,831	1	0,940	0,932
10	0,763	0,940	1	0,945
30	0,773	0,932	0,945	1

5 lentelė: Koreliacija tarp empirinių rangų blokų skirstinių, kai bloko dydis 6

Nors koreliacija, paskaičiuota 3 atvejais (kai bloko dydis 4, 5, 6) tarpusavyje skiriasi nežymiai, t.y. koreliacijos dėsningumas išlieka tas pats, tačiau šiais atvejais koreliacija nesielgia taip, kaip koreliacija, apskaičiuota tarp neužkoduotų duomenų. Užkoduotų duomenų

koreliacija neturi dėsningumo – vienu momentu padidėjus skirtumui tarp obligacijų trukmės (pavyzdžiui, jei lyginam 1 ir 5 bei 1 ir 10 metų trukmės obligacijas) koreliacija sumažėja, kitu momentu – vėlgi padidėjus skirtumui tarp obligacijų trukmės (pavyzdžiu, jei lyginam 1 ir 10 bei 1 ir 30 metų trukmės obligacijas) – koreliacija tarp duomenų padidėja. Pabandykime rasti atitikmenį užkoduotų duomenų sąryšiui nustatyti ir paskaičiuoti 2.12 apibrėžime aprašytus teigiamos bei neigiamos priklausomybės įverčius, taip pat panagrinėti, kaip jie elgiasi, kintant obligacijų trukmei. Skaičiavimų rezultatai pateikti 6, 7 lentelėse.

	Lyginamų trukmių obligacijos					
Bloko ilgis	1 ir 5	1 ir 10	1 ir 30	5 ir 10	5 ir 30	10 ir 30
4	0,172	0,133	0,099	0,508	0,321	0,504
5	0,099	0,076	0,054	0,396	0,215	0,389
6	0,054	0,036	0,026	0,284	0,129	0,280

6 lentelė: Teigiama priklausomybė tarp skirtingos trukmės obligacijų užkoduotų eilučių, kai bloko dydis 4–6

	Lyginamų trukmių obligacijos					
Bloko ilgis	1 ir 5	1 ir 10	1 ir 30	5 ir 10	5 ir 30	10 ir 30
4	–0,038	–0,032	–0,057	–0,058	–0,056	–0,057
5	–0,013	–0,007	–0,017	–0,017	–0,016	–0,017
6	–0,004	–0,000	–0,005	–0,005	–0,005	–0,005

7 lentelė: Neigiama priklausomybė tarp skirtingos trukmės obligacijų užkoduotų eilučių, kai bloko dydis 4–6

Dėl gautų labai mažų neigiamos priklausomybės įverčių, sunku būtų pastebėti tam tikrus dėsningumus, tačiau pažvelgus į 6 lentelę (kurioje pateikta teigiama priklausomybė) yra pastebėtinas dėsningumas, kuris buvo pastebėtas jau skaičiuojant koreliaciją tarp dar neužkoduotų duomenų – 2 lentelėje. Nepriklausomai nuo kombinacijos bloko ilgio – visais atvejais didinant skirtumą tarp obligacijų trukmės – priklausomybė mažėja (analogiškai mažėjo skaičiuojant koreliaciją), t.y. pavyzdžiui nagrinėjant 1 ir 5, 1 ir 10, 1 ir 30 metų priklausomybes (analogiškai nežkoduotų duomenų koreliacijas), jos vis mažės. Po šių paskaičiavimų galime daryti išvadą, kad teigiama priklausomybė, paskaičiuota mūsų nagrinėjamiems užkoduotiems duomenims pagal 2.12 apibrėžimą yra tarsi analogija koreliacijai, paskaičiuotai

tarp neužkoduotų duomenų, jei mus domina, kaip elgiasi (didėja, mažėja) priklausomybė tarp dviejų laiko eilučių.

Kaip jau buvo paminėta anksčiau, tolesniam darbui su laiko eilutėmis, pirmiausia reikia nuodugniai jas ištirti. Kadangi šiame darbe yra nagrinėjami ir lyginami 1, 5, 10 ir 30 metų trukmės obligacijų pelningumai, tai visoms šioms eilutėms paskaičiuosime keitinių entropiją pagal 2.9 apibrėžimą. Gauti rezultatai pateikti lentelėje 8 lentelėje.

	Obligacijų trukmė				
Bloko ilgis	1	5	10	30	$\ln(n!)$
4	2,820	2,992	3,004	2,999	**3,178**
5	4,106	4,387	4,395	4,397	**4,787**
6	5,409	5,810	5,817	5,823	**6,579**

8 lentelė: Keitinių entropija, kai bloko dydis 4–6

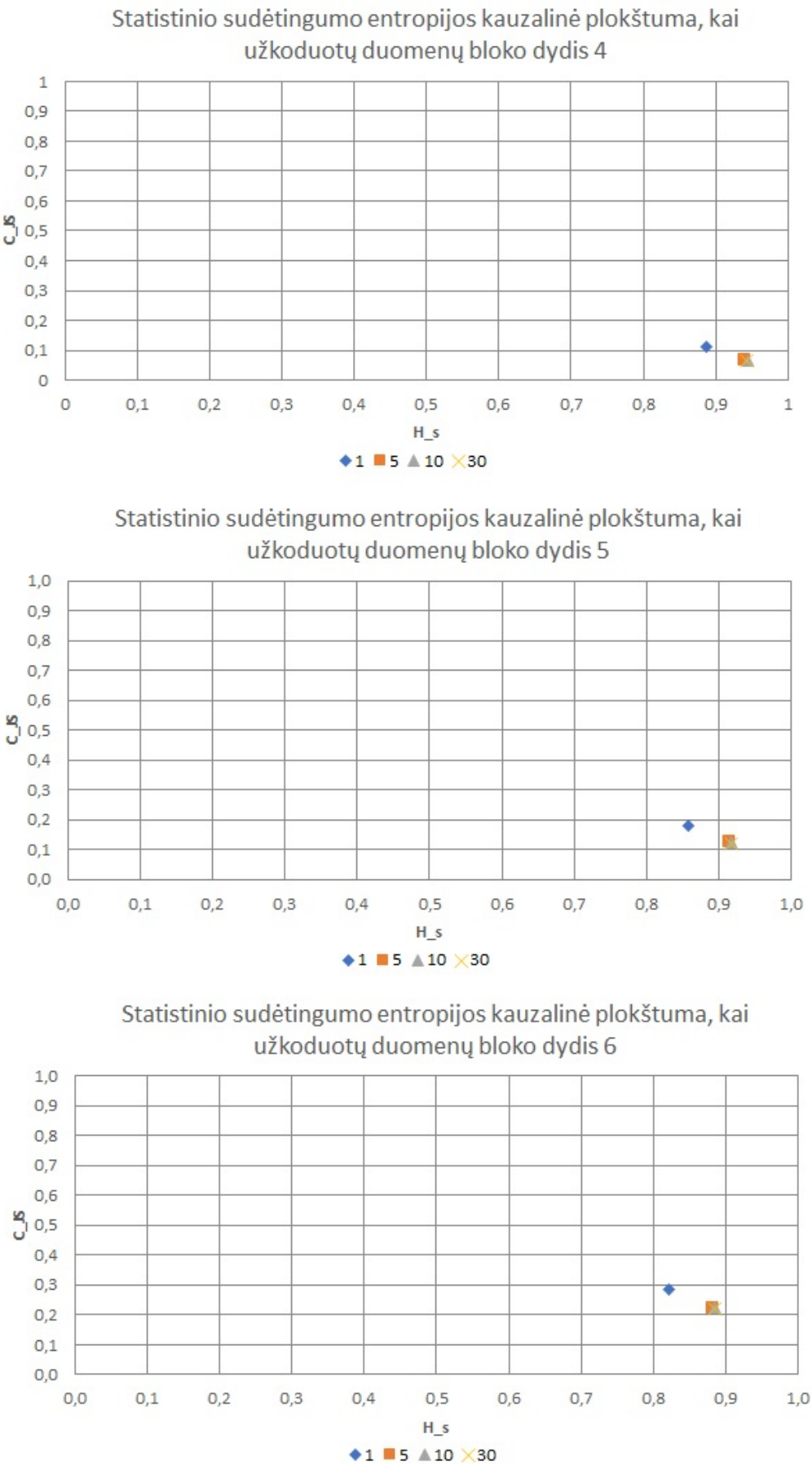
Pažvelgus į 8 lentelėje gautus rezultatus, pirmoji išvada yra tokia, kad nė viena iš užkoduotų laiko eilučių nėra visiškai nuspėjama, t.y. nė vienu iš atvejų apskaičiuotas parametras $H(n)$ nėra lygus nuliui. Taip pat, galima pasakyti, kad sekos nėra visiškai atsitiktinės – lygindami lentelės stulpelį $\ln(n!)$ (kuris parodo, kokį gautumėme $H(n)$, jei visos kombinacijos būtų vienodai tikėtinos – tai yra būtų visiškai nenuspėjamos) su $H(n)$, gautu kiekvienai sekai su skirtingais rango dydžiais, matome, kad visi $H(n)$ yra mažesni už $\ln(n!)$. Pastebėtina ir tai, kad visais atvejais (pasirinkus bet kurį bloko dydį) 1 metų trukmės obligacijų užkoduotų duomenų keitinių entropija yra mažiausia, palyginus ją su kitų trukmių obligacijomis. Tai galima susieti su 3, 4, 5 paveikslėliuose matoma tendencija. Nagrinėkime 3 paveikslėlį – jame yra matoma, kad kombinacija 0123, palyginus su kitų galimų kombinacijų įgyjamomis tikimybėmis, turi didžiausią tikimybę pasirodyti, nepaisant obligacijų trukmės. Nors ši kombinacija pasirodo su didžiausia tikimybe visais 4 atvejais (1, 5, 10, 30 metų trukmės obligacijų užkoduotose duomenyse), tačiau 1 metų trukmės obligacijų stacionariame skirstinyje šios kombinacijos pasirodymo tikimybė (lygi 0,245) yra didesnė nei kitos trukmės obligacijų stacionariuose skirstiniuose (atitinkamai 5 metų – 0,145, 10 metų – 0,132, 30 metų – 0,137). Analogiška tendencija yra matoma ir 4, 5 paveikslėliuose. Tai reiškia, kad gavome pagrįstus rezultatus – kaip ir buvo paaiškinta 2.9 apibrėžime, mažesnė keitinių entropija indikuoja apie labiau koncentruotą viename taške (labiau nuspėjamą) tikimybių pasiskirstymą tarp kombinacijų. Nors nagrinėjamu atveju gavome, kad kad keitinių entropija yra mažiausia nagrinėjant būtent 1 metų trukmės obligacijų užkoduotus duomenis, tačiau ji taip pat nėra

artima 0, taigi galime tik bendrai apibūdinti visas 4 užkoduotas eilutes ir pasakyti, kad jų keitinių entropija nėra artima 0 (tikimybės tarp kombinacijų nėra susikoncentravusios ties kažkuria viena kombinacija), tačiau ji taip pat nėra lygi dydžiui $\ln(n!)$, kuris indikuotų apie tai, jog užkoduota eilutė yra visiškai atsitiktinė.

Tęsdami toliau, pabandykime ištirti ir pagal 2.13 apibrėžimą nubrėžti statistinio sudėtingumo entropijos kauzalinę plokštumą 1, 5, 10 ir 30 metų trukmės užkoduotoms laiko eilutėms, kai bloko dydis vėlgi 4, 5, 6.

6 paveikslėlyje galima pastebėti, kad 5, 10, 30 metų trukmės obligacijų užkoduotų duomenų parametrai susikoncentravę tarsi viename taške, o 1 metų trukmės – pasislinkęs šiek tiek kairiau ir x ašies atžvilgiu ir aukščiau y ašies atžvilgiu, t.y. 5, 10 ir 30 metų trukmės obligacijų pelningumas turi didesnę keitinių entropiją ir mažesnį statistinio sudėtingumo matą, kai tuo tarpu 1 metų trukmės užkoduotas obligacijų pelningumas turi mažesnę keitinių entropiją ir didesnį statistinio sudėtingumo matą. Vėlgi, tokius rezultatus galėjo lemti jau prieš tai (nagrinėjant keitinių entropiją) aptartos priežastys.

Iš nagrinėtų užkoduotų duomenų parametrų su nedideliais nuokrypiai buvo pastebėtos bendros 1, 5, 10 ir 30 metų trukmės obligacijų pelningumo savybės, kitaip tariant, nepastebėta, kad kažkuri viena iš užkoduotų eilučių stipriai išsiskirtų iš kitų savo savybėmis – kiekviena iš užkoduotų eilučių nėra visiškai atsitiktinė, tačiau taip pat nepasižymi ir stipraus susikoncentravimo ties kažkuria iš kombinacijų savybe. Užkodavus nagrinėjamus duomenis ranginiais metodais nė vienu iš atvejų (kai bloko dydis 4, 5 ar 6) keitinių entropija nebuvo artima nuliui (kas galbūt padėtų prognozuojant), statistinio sudėtingumo kauzalinė plokštuma taip pat neatskleidė tokių savybių, kurios padėtų išskirti kažkurios trukmės obligacijų duomenis, ar pasirinkti bloko dydį, kuris būtų parankiausias prognozavimui, tačiau buvo aptiktas ryšys tarp 1, 5, 10 ir 30 metų trukmės obligacijų pelningumo – šį ryšį atskleidė koreliacija ir teigiama priklausomybė. Didžiausia teigiama priklausomybė bei koreliacija tarp užkoduotų laiko eilučių buvo gauta, kai bloko dydis 4. Kadangi prognozavimas bus atliekamas ne tik vienmačiu, bet ir dvimačiu atveju, kuriam gali turėti įtakos sąryšiai tarp užkoduotų eilučių, tad toliau ir bus nagrinėjami tik duomenys, užkoduoti 4 skirtingais simboliais.



6 pav.: Statistinio sudėtingumo entropijos kauzalinė plokštuma

3.3 Prognozavimas vienmačiu atveju

Šiame skyrelyje bus pateikti ir detalizuoti prognozavimo vienmačiu atveju rezultatai bei įvertinti 2.4 skyrelyje aprašytais būdais, sekančiame skyrelyje – analogiški veiksmai bus atlikti prognozuojant dvimačiu atveju.

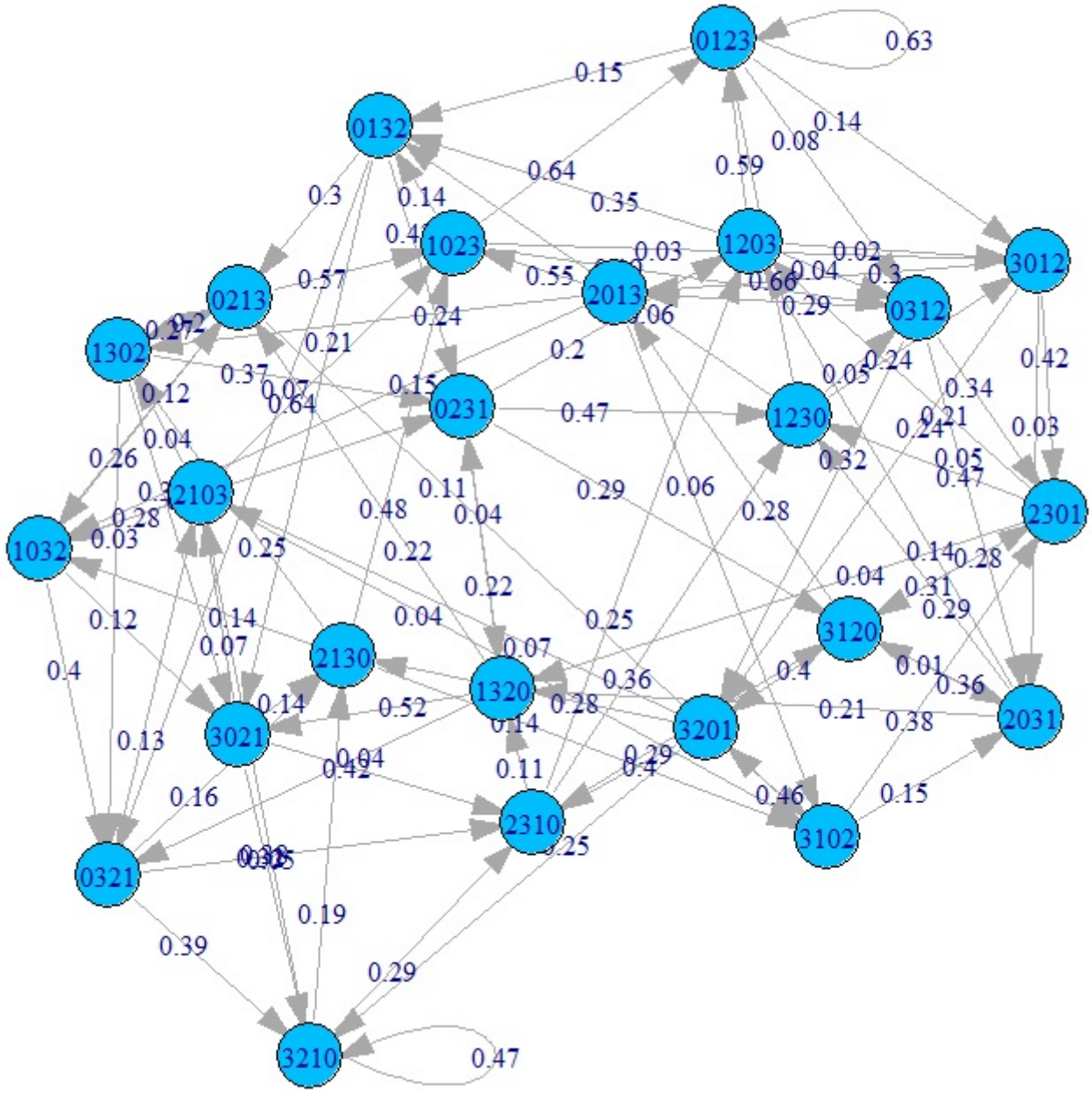
Prognozuojant užkoduotas laiko eilutes, kaip ir aprašyta 2.2 skyrelyje, kiekvienai iš užkoduotų laiko eilučių (1, 5, 10 ir 30 metų trukmės) buvo sudarytos Markovo grandinės su 24 būsenomis: 0123, 0132, 0213, 0231, 0312, 0321, 1023, 1032, 1203, 1230, 1302, 1320, 2013, 2031, 2103, 2130, 2301, 2310, 3012, 3021, 3102, 3120, 3201, 3210. Šios būsenos parodo, kaip per keturias iš eilės einančias dienas kito obligacijų pelningumas, pavyzdžiui, 0123 – parodo, kad visas dienas iš eilės obligacijų pelningumas vis kilo, 3012 – atvaizduoja, kad 3–os dienos (simbolis 2) pelningumas buvo didžiausias, po jo pagal didumą buvo 2–os dienos (simbolis 1) pelningumas, 1–os dienos (simbolis 0) pelningumas ir mažiausias pelningumas pastebėtas ties 4–ąja diena (simbolis 3). Kiekvienai iš nagrinėjamų užkoduotų finansinių laiko eilučių buvo paskaičiuoti galimų kombinacijų santykiniai (statistiniai) dažniai (empirinis rangų blokų skirstinys), taip gaunant tikimybes, su kuriomis kombinacijos pasirodys 2019-10-31 dieną. Rezultatai pateikti 9 lentelėje.

Prognozavimui naudojama kiekvienos trukmės 80% ir 50% turimų užkoduotų duomenų (atitinkamai laikotarpiai nuo 2006-02-09 iki 2017-02-08 ir nuo 2006-02-09 iki 2012-12-28) ir prognozuojama yra datai 2019-10-31. Iš atitinkamų dalių duomenų, naudojantis R paketu, kaip jau buvo paminėta anksčiau, sudaromos Markovo grandinės su perėjimo tikimybių matrica. Tam, kad būtų paprasčiau įsivaizduoti sudarytas perėjimo tikimybių matricas, viena iš jų (1 metų trukmės obligacijų), kuri sudaryta iš 80% turimų apie tą eilutę duomenų, pavaizduota 7 paveikslėlyje.

Iš tų pačių duomenų (80% bei 50%) kiekvienai iš eilučių buvo paskaičiuoti stacionarieji skirstiniai, kurie parodė, jog visais atvejais didžiausia pasirodymo tikimybė išlieka ties kombinacija 0123. Tai reiškia, kad prognozuojant užkoduotas eilutes pradinių tikimybių vektorius (vektorius, iš kurio pirmą kartą dauginsime sudarytą tikimybių perėjimo matricą) visada bus (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), kitaip sakant, labiausiai tikėtina, kad pirmasis žingsnis bus į kombinaciją 0123.

	Kombinacijų pasirodymo tikimybės			
Kombinacija	1 metų	5 metų	10 metų	30 metų
0123	0,245	0,145	0,132	0,137
0132	0,059	0,064	0,061	0,063
0213	0,031	0,037	0,037	0,032
0231	0,037	0,024	0,026	0,027
0312	0,036	0,035	0,037	0,033
0321	0,013	0,021	0,023	0,025
1023	0,057	0,060	0,063	0,063
1032	0,015	0,019	0,018	0,021
1203	0,032	0,035	0,036	0,035
1230	0,053	0,042	0,032	0,033
1302	0,023	0,015	0,020	0,019
1320	0,009	0,021	0,023	0,018
2013	0,037	0,022	0,025	0,027
2031	0,005	0,018	0,012	0,013
2103	0,012	0,019	0,024	0,024
2130	0,026	0,032	0,031	0,036
2301	0,047	0,039	0,036	0,033
2310	0,046	0,066	0,066	0,062
3012	0,048	0,038	0,033	0,035
3021	0,025	0,037	0,036	0,038
3102	0,010	0,016	0,017	0,016
3120	0,041	0,049	0,049	0,049
3201	0,045	0,059	0,062	0,060
3210	0,048	0,086	0,097	0,099

9 lentelė: Skirtingų trukmių obligacijų užkoduotų duomenų pasirodymo tikimybės 2019-10-31 dieną



7 pav.: 1 metų trukmės obligacijų užkoduotų duomenų Markovo grandinės perėjimų diagrama (sudaryta iš 80% turimų duomenų)

Dabar, kai turime pradinių tikimybių vektorių ir perėjimo tikimybių matricas (kiekvienai iš eilučių po dvi tikimybių perėjimo matricas – viena sudaryta naudojantis 80% duomenų, kita – 50% duomenų), galima atlikti prognozavimą.

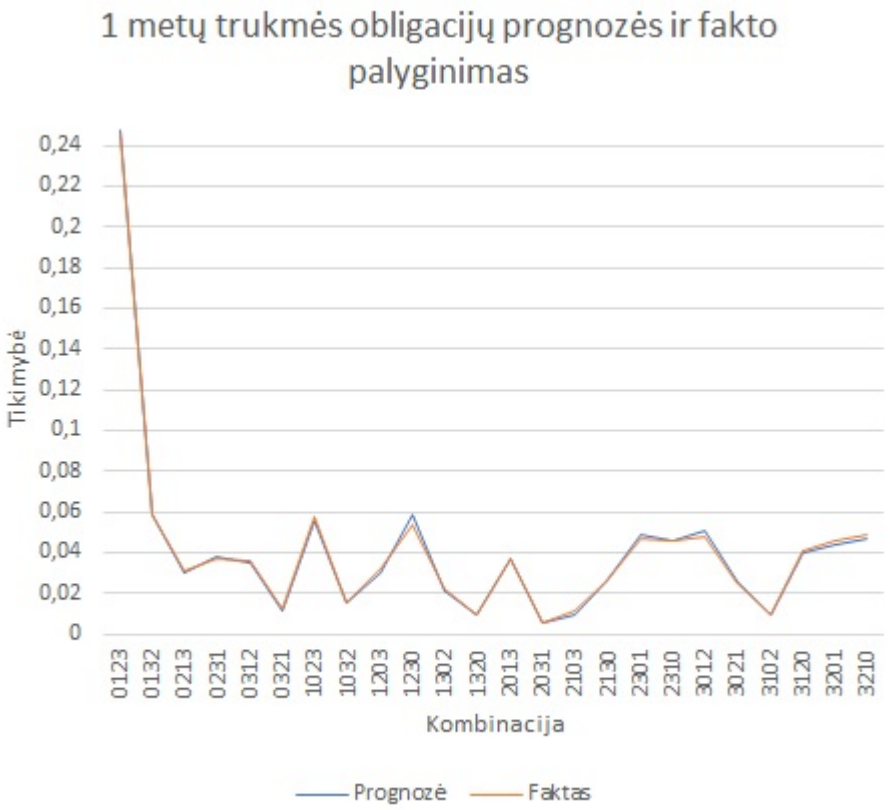
Kadangi kiekviena iš užkoduotų duomenų eilučių sudaryta iš 3451 kombinacijų, tai atliekant prognozavimą iš 80% duomenų (2761 žingsnių Markovo grandinės), tikimybių vektorių (su kokia tikimybe kombinacijos pasirodys po kiekvieno žingsnio) iš perėjimo tikimybių matricos dauginame 690 kartų (tada gausime prognozę 2019-10-31 dienai). Kitu atveju, atliekant prognozavimą iš 50% duomenų (1725 žingsnių Markovo grandinės), tikimybių vektorių iš perėjimo tikimybių matricos dauginame 1726 kartus (vėlgi bus gauta prognozė 2019-10-31 dienai). Tokiu būdu kiekvienai iš užkoduotų duomenų eilučių kombinacijai gauname po dvi pasirodymo tikimybes (viena gauta prognozuojant iš 80% duomenų, kita iš 50%), t.y. nuprognozuojame, su kokia tikimybe kiekviena iš kombinacijų pasirodys po 3451 žingsnių. Idealiu atveju duomenys pateikti 9 lentelėje ir mūsų gauti duomenys turėtų visiškai sutapti. Toliau pateikiami gauti rezultatai.

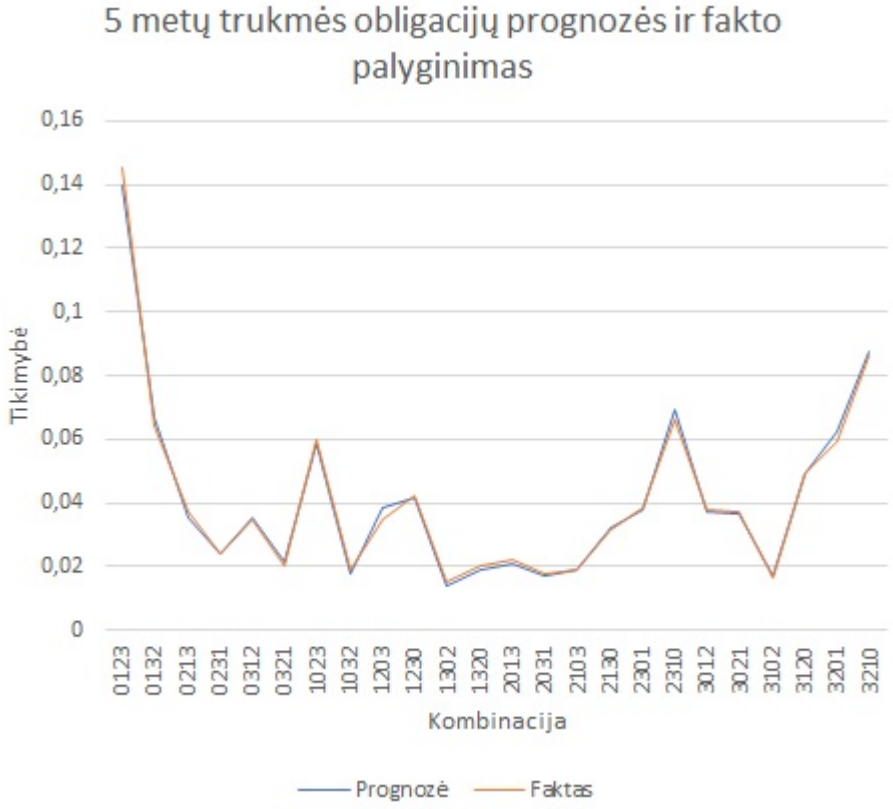
Obligacijų trukmė	1 metų	5 metų	10 metų	30 metų
$level_{0,05}$	66,67%	58,33%	70,83%	**75,00%**
$level_{0,95}$	0%	0%	0%	0%
MAPE	4,52%	3,88%	3,60%	**2,97%**

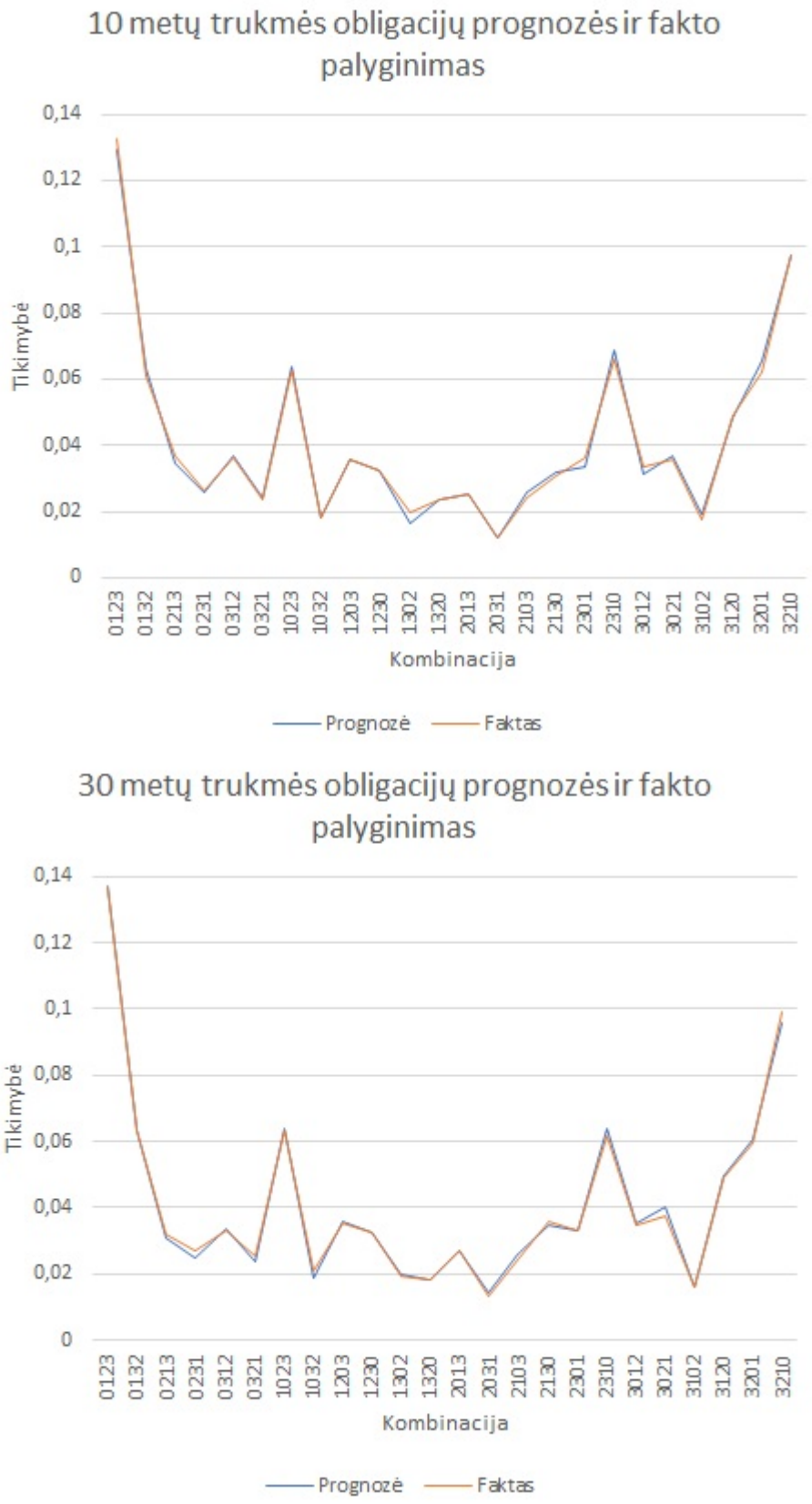
10 lentelė: Skirtingos trukmės obligacijų užkoduotų duomenų prognozės vertinimas, kai prognozei naudojama 80% duomenų

Obligacijų trukmė	1 metų	5 metų	10 metų	30 metų
$level_{0,05}$	20,83%	37,50%	41,67%	**54,17%**
$level_{0,95}$	0%	0%	0%	0%
MAPE	12,08%	6,38%	**6,22%**	7,17%

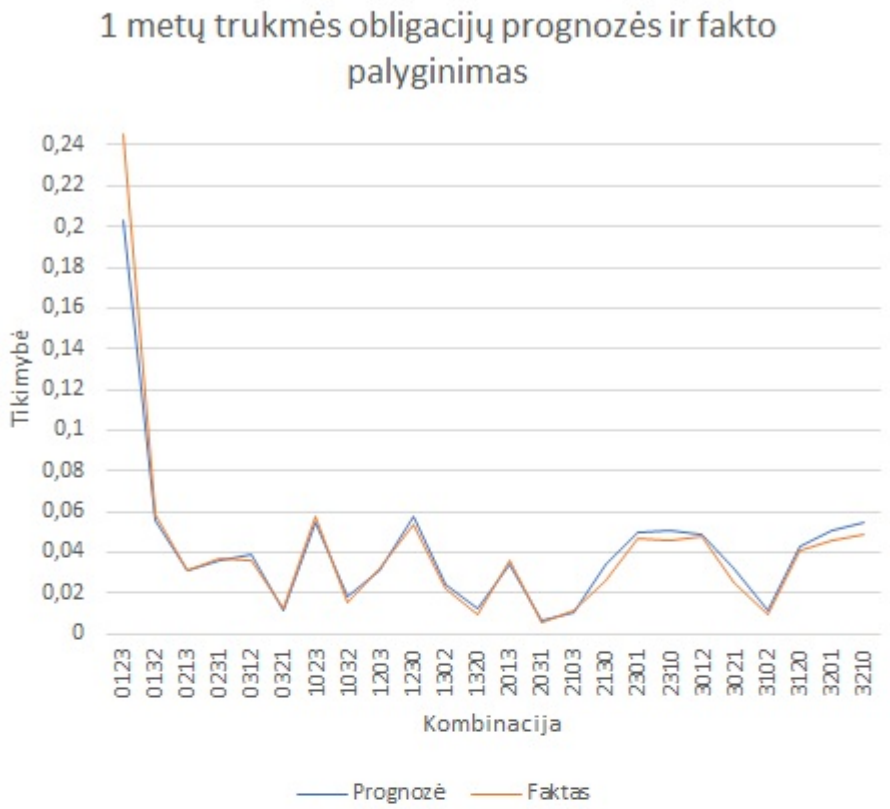
11 lentelė: Skirtingos trukmės obligacijų užkoduotų duomenų prognozės vertinimas, kai prognozei naudojama 50% duomenų

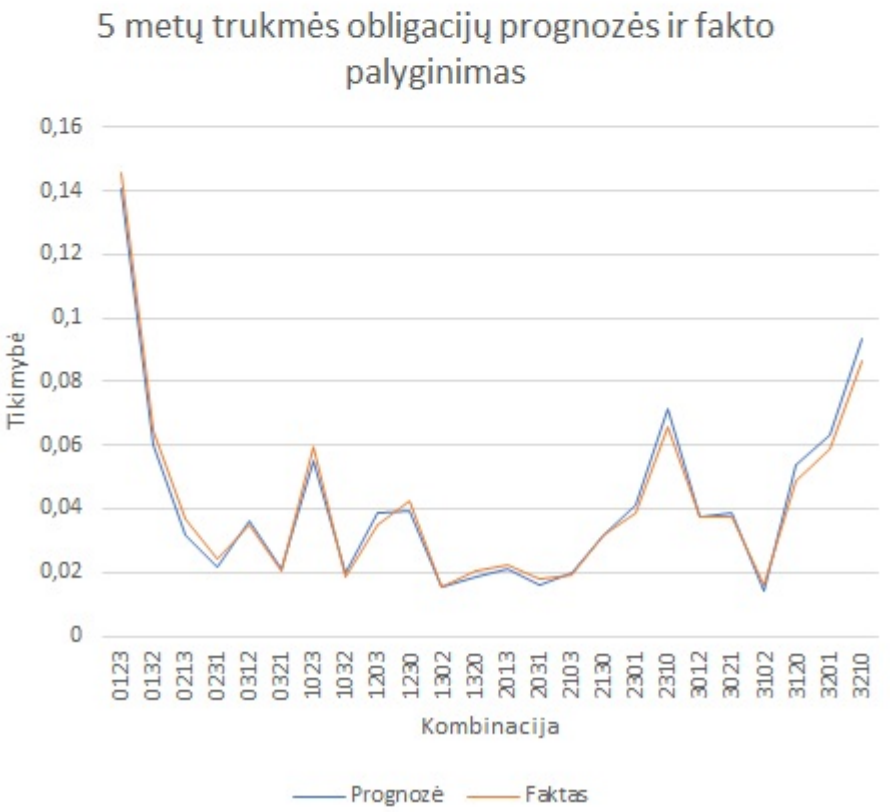

1 metų trukmės obligacijų prognozės ir fakto palyginimas
Tikimybė
0,24
0,22
0,2
0,18
0,16
0,14
0,12
0,1
0,08
0,06
0,04
0,02
0
0123 0132 0213 0231 0312 0321 1023 1032 1203 1230 1302 1320 2013 2031 2103 2130 2301 2310 3012 3021 3102 3120 3201 3210
Kombinacija
Prognozė
Faktas

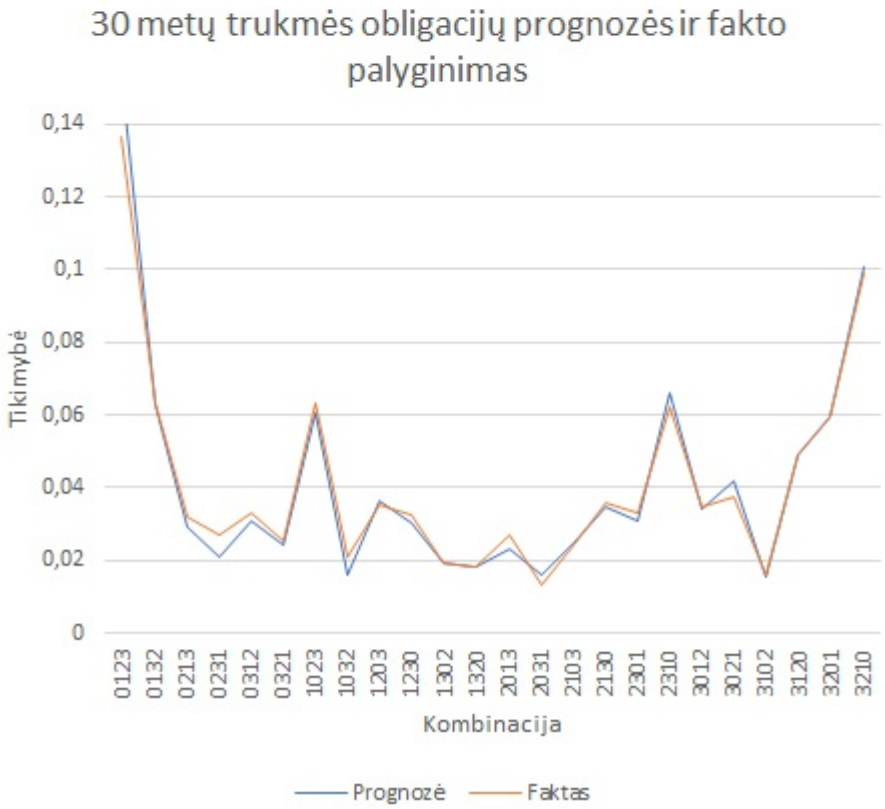

5 metų trukmės obligacijų prognozės ir fakto palyginimas
Tikimybė
0,16
0,14
0,12
0,1
0,08
0,06
0,04
0,02
0
0123 0132 0213 0231 0312 0321 1023 1032 1203 1230 1302 1320 2013 2031 2103 2130 2301 2310 3012 3021 3102 3120 3201 3210
Kombinacija
Prognozė
Faktas



8 pav.: Skirtingos trukmės obligacijų užkoduotų duomenų prognozės palyginimas su faktu, kai prognozei naudojama 80% duomenų


1 metų trukmės obligacijų prognozės ir fakto palyginimas
Tikimybė
0,24
0,22
0,2
0,18
0,16
0,14
0,12
0,1
0,08
0,06
0,04
0,02
0
0123 0132 0213 0231 0312 0321 1023 1032 1203 1230 1302 1320 2013 2031 2103 2130 2301 2310 3012 3021 3102 3120 3201 3210
Kombinacija
Prognozė
Faktas


5 metų trukmės obligacijų prognozės ir fakto palyginimas
Tikimybė
0,16
0,14
0,12
0,1
0,08
0,06
0,04
0,02
0
0123 0132 0213 0231 0312 0321 1023 1032 1203 1230 1302 1320 2013 2031 2103 2130 2301 2310 3012 3021 3102 3120 3201 3210
Kombinacija
Prognozė
Faktas



9 pav.: Skirtingos trukmės obligacijų užkoduotų duomenų prognozės palyginimas su faktu, kai prognozei naudojama 50% duomenų

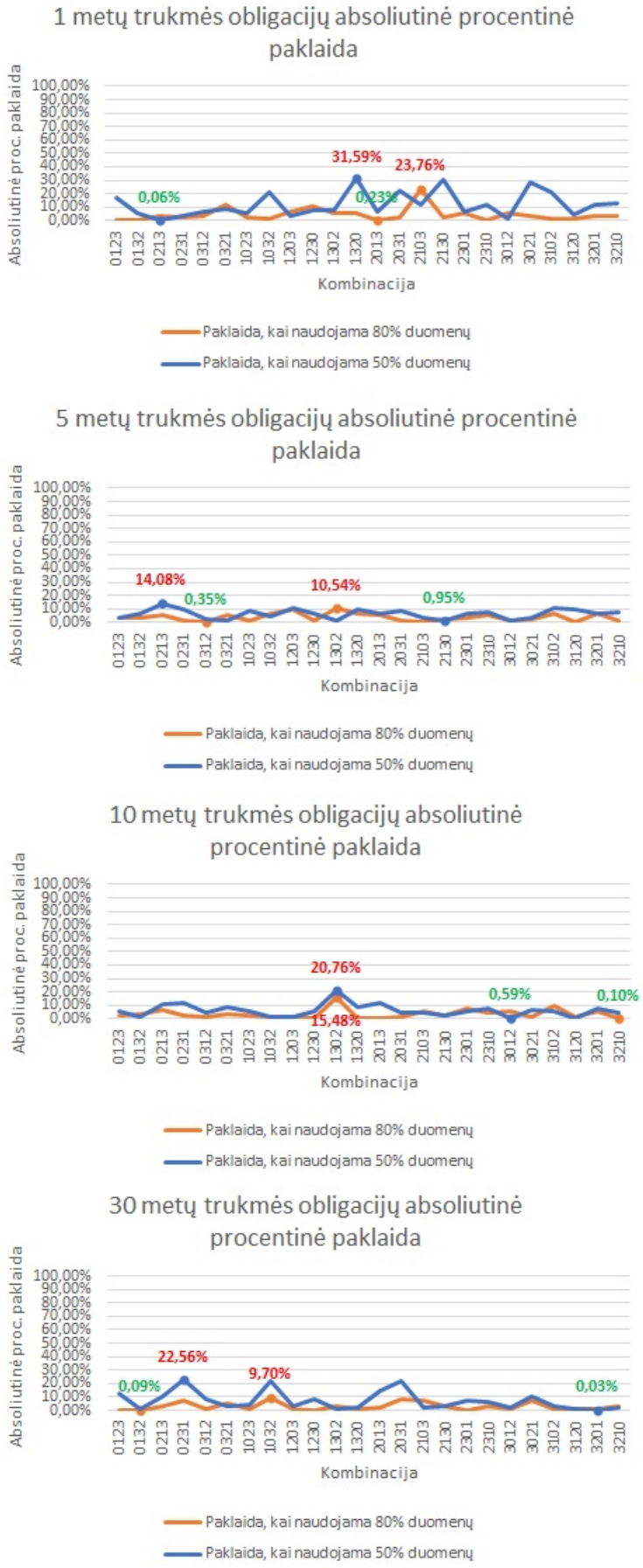
Kaip ir galima buvo tikėtis, prognozė, naudojantis tik 50% duomenų, yra prastesnė, nei prognozė, naudojantis 80% duomenų – nors nė vienu iš atvejų nebuvo aptikta paklaidos, kuri būtų didesnė arba lygi 95% procentams, tačiau prognozuojant, remiantis didesniu kiekiu istorinių duomenų, paklaidų, kurios mažesnės arba lygios 5%, yra daugiau visoms nagrinėjamoms eilutėms. Taip pat svarbu paminėti, kad vidutinė absoliutinė procentinė paklaida, nagrinėjant visų laikotarpių obligacijų užkoduotus duomenis, yra gauta geresnė taip pat prognozuojant, naudojantis 80% procentų duomenų.

Pastebėtina, kad prognozuojant, remiantis didesniu kiekiu istorinių duomenų, geriausiai pavyko nuprognozuoti 30 metų trukmės obligacijų užkoduotus duomenis. Taip galima teigti remiantis abiem nagrinėjamais vertinimo kriterijais – net 75% visų kombinacijų pasirodymo tikimybių buvo nuprognozuotos su mažesnėmis arba lygiomis 5% absoliutinėmis procentinėmis paklaidomis, o vidutinė absoliutinė procentinė paklaida lygi 2,97%.

Remiantis mažesniu kiekiu duomenų negalima vienareikšmiškai nusakyti, kurios trukmės obligacijų duomenis pavyko nusakyti tiksliausiai, todėl, nes pagal vieną iš kriterijų – paklaidų dalį, kuri mažesnė arba lygi 5% – vėlgi būtų galima išskirti 30 metų trukmės obligacijas, tačiau pagal vidutinę absoliutinę procentinę paklaidą – tiksliausiai pavyko nuprognozuoti 10 metų trukmės užkoduotus obligacijų duomenis.

10 paveiksėlyje pavaizduota, su kokia absoliutine procentine paklaida kiekviena iš kombinacijų buvo nuprognozuota. Maksimalios paklaidos aptinkamos prognozuojant 1 metų trukmės obligacijų užkoduotus duomenis – 23,76% (prognozuojant, remiantis 80% istorinių duomenų) ties kombinacija 2103 ir 31,59% (prognozuojant, remiantis 50% istorinių duomenų) ties kombinacija 1320. Panagrinėjus šių kombinacijų paklaidas buvo pastebėta, kad kombinacija 2103, remiantis 80% istorinių duomenų, pasirodė 25 kartus iš galimų 2761, tuo tarpu per likusį laikotarpį (kuris buvo prognozuojamas) – kombinacija pasirodė jau 16 kartų iš galimų 690. Tai reiškia, kad remiantis istoriniais duomenimis, buvo tariama, jog kombinacija 2103 pasirodys su tikimybe 0,0091, tačiau per likusius žingsnius ji realiai pasirodė su daugiau nei du kartus didesne tikimybe – 0,0232, todėl logiška, kad vertinant modelį ties šia kombinacija yra pastebima didesnė paklaida. Analogiška situacija su kombinacija 1320, prognozuojant, remiantis 50% duomenų. Ši kombinacija per pirmus 1726 žingsnių pasirodo 21 kartą (su tikimybe 0,0122), per likusius 1725 ta pati kombinacija pasirodo jau beveik du kartus mažiau – tik 11 kartų iš galimų 1725 (t.y. su tikimybe 0,64), todėl vėlgi tokią kombinaciją gana sunku tiksliai nuprognozuoti.

Toliau nagrinėkime prognozavimą dvimačiu atveju, palygindami prognozės tikslumą su vienmačiu atveju.



10 pav.: Skirtingos trukmės obligacijų užkoduotų duomenų prognozės absoliutinė procentinė paklaida, kai prognozei naudojama 80% ir 50% duomenų

3.4 Prognozavimas dvimačiu atveju

Šiame skyrelyje bus trumpai aprašyti veiksmai, kurie buvo atlikti skaičiuojant prognozę, pagal 2.3 skyrelyje pateiktą metodą. Gauti duomenys, prognozuojant dvimačiu atveju, bus pateikti ir įvertinti tokiais pačiais kriterijais, kaip ir vienmačiu atveju. Šie veiksmai padės palyginti, kuris iš modelių, ir kokioms sąlygoms esant, yra tikslesnis.

Pirmiausia, kaip jau buvo aprašyta teorinėje dalyje, siekiant prognozuoti dvimačiu atveju, reikalingos dvi užkoduotos laiko eilutės – buvo prognozuojami 1 metų trukmės obligacijų duomenys, remiantis 5, 10, 30 metų trukmės obligacijų duomenimis, 5 metų trukmės obligacijų duomenys, remiantis 1, 10, 30 metų trukmės obligacijų duomenimis, 10 metų trukmės obligacijų duomenys, remiantis 1, 5, 30 metų trukmės obligacijų duomenimis ir 30 metų trukmės obligacijų duomenys, remiantis 1, 5, 10 metų trukmės obligacijų duomenimis. Tam, kad būtų aiškus 2.3 skyrelyje aprašytas dvimatis prognozavimo modelis, išnagrinėsime vieną iš atvejų, tarkime, kaip 5 metų trukmės obligacijų duomenys buvo prognozuojami, remiantis 1 metų trukmės obligacijų duomenimis, kai naudojamasi 80% prognozuojamų duomenų. Vėliau bus pateikti visais atvejais gauti rezultatai.

11 paveikslėlyje yra pavaizduotos visos paskaičiuotos sąlyginės tikimybės (iš 80% visų duomenų – t.y. iš 2761 žingsnių), reikalingos prognozuojant 5 metų trukmės obligacijų duomenis, remiantis 1 metų trukmės obligacijų duomenimis. Šią matricą reikėtų skaityti taip – pirmame stulpelyje ir pirmoje eilutėje yra išrašytos visos įmanomos 24 kombinacijos. Jei yra priimama sąlyga, kad 1 metų trukmės obligacijų užkoduota eilutė (kuri naudojama tik kaip pagalbinė) yra ties tam tikra kombinacija (viena iš 11 paveikslėlyje pavaizduoto pirmojo stulpelio), tai pereiti į 5 metų trukmės obligacijų užkoduotos eilutės (kuri bus prognozuojama) tam tikrą kombinaciją (vieną iš 11 paveikslėlyje pavaizduotos pirmos eilutės) tikimybė yra lygi reikšmei, nurodytai susikertančiame tarp kombinacijų stulpelyje. Pavyzdžiui, tikimybė pereiti iš pirmosios (1 metų trukmės obligacijų) užkoduotos eilutės 0123 kombinacijos į antrosios (5 metų trukmės obligacijų) užkoduotos eilutės 3021 kombinaciją yra lygi reikšmei, kuri 11 paveikslėlyje išryškinta raudonai.

	0123	0132	0213	0312	0231	0321	1023	1032	2013	3012	2031	3021	1203	1302	2103	3102	2301	3201	1230	1320	2130	3120	2310	3210
0123	0,286	0,1	0,063	0,038	0,028	0,022	0,088	0,022	0,025	0,023	0,018	0,009	0,032	0,01	0,019	0,01	0,015	0,041	0,026	0,012	0,016	0,022	0,037	0,04
0132	0,166	0,27	0,012	0,123	0,012	0,006	0,037	0,031	0	0,086	0	0,031	0,025	0,025	0	0,025	0,012	0,031	0,006	0,018	0,018	0,025	0,012	0,031
0213	0,22	0,049	0,195	0,012	0,073	0,049	0,037	0	0,122	0	0,049	0,061	0,024	0	0,024	0	0,024	0	0	0	0	0,024	0,024	0,012
0312	0,155	0,103	0,031	0,196	0,031	0,093	0,01	0,01	0	0,052	0,01	0,093	0,01	0	0	0,021	0,01	0,052	0,01	0,01	0	0,041	0	0,062
0231	0,132	0,047	0,047	0,028	0,123	0,066	0,019	0	0,009	0,028	0,038	0,047	0	0	0	0,009	0,104	0,113	0	0	0,028	0,038	0,066	0,057
0321	0,032	0,161	0,032	0,097	0,129	0,129	0	0	0,032	0	0	0,161	0	0	0	0,032	0	0,129	0	0,032	0	0	0,032	0
1023	0,195	0,032	0,013	0,019	0,006	0	0,26	0,052	0,019	0,019	0	0,006	0,078	0,032	0,026	0,013	0,019	0,013	0,045	0,026	0,045	0,013	0,045	0,019
1032	0,071	0,119	0	0	0	0	0,143	0,19	0	0	0	0	0	0,167	0	0,048	0	0,024	0,048	0,071	0	0,024	0,024	0,071
2013	0,139	0,01	0,069	0,01	0,01	0	0,04	0	0,079	0,03	0,05	0,01	0,04	0	0,069	0	0,05	0,089	0,04	0,01	0,079	0,04	0,069	0,069
3012	0,058	0,072	0,029	0,036	0	0,022	0,043	0,007	0	0,144	0,029	0,094	0,022	0,007	0	0,05	0,029	0,072	0,036	0,022	0,014	0,072	0,029	0,115
2031	0	0	0	0,071	0,071	0,071	0	0	0	0	0,071	0	0	0	0	0	0,357	0,143	0	0	0,071	0	0,143	0
3021	0,028	0,042	0	0,042	0,042	0,042	0,014	0	0,014	0,097	0,028	0,333	0	0	0	0	0,014	0,208	0,014	0	0	0,014	0,028	0,042
1203	0,134	0,024	0,024	0	0	0,012	0,134	0,012	0,024	0	0,012	0	0,207	0	0,049	0,012	0,012	0,037	0,146	0	0,098	0,037	0,012	0,012
1302	0,102	0,085	0	0,017	0	0	0,085	0,051	0,034	0,034	0,017	0,017	0,017	0,068	0	0,017	0,017	0,017	0,085	0,119	0	0,136	0,017	0,068
2103	0,12	0,04	0,12	0	0,04	0	0	0	0,08	0	0	0	0,04	0	0,24	0	0,08	0	0,04	0	0,08	0	0,12	0
3102	0	0,077	0	0	0	0,038	0	0,038	0	0,154	0	0,038	0	0,038	0	0,154	0	0,038	0,038	0,038	0	0,077	0,115	0,154
2301	0,074	0,022	0,029	0,015	0,007	0,029	0,007	0,007	0,037	0,037	0,044	0,059	0,029	0	0,007	0,022	0,169	0,096	0,015	0,007	0,044	0,007	0,176	0,059
3201	0,041	0,008	0	0,008	0,05	0,025	0,017	0	0	0,066	0,008	0,074	0	0	0,008	0,017	0,058	0,264	0	0,025	0	0,041	0,083	0,207
1230	0,061	0,025	0,012	0,006	0,006	0,012	0,055	0,006	0	0,025	0,006	0,006	0,123	0,025	0,025	0,018	0,049	0,031	0,178	0,025	0,031	0,08	0,08	0,117
1320	0,111	0	0	0	0	0	0,037	0,037	0	0	0	0,037	0	0,074	0	0,111	0	0	0,074	0,148	0,037	0,296	0,037	0
2130	0,027	0	0,027	0	0,014	0,014	0,027	0	0,014	0,014	0,014	0	0,096	0	0,096	0	0,014	0,041	0,055	0	0,219	0,041	0,205	0,082
3120	0,009	0,036	0	0,036	0,009	0	0,009	0,018	0	0,045	0	0	0,027	0,018	0	0,036	0,027	0,027	0,082	0,055	0,036	0,245	0,045	0,236
2310	0,024	0	0,008	0,008	0,016	0,008	0,008	0	0,04	0,008	0,024	0,016	0,016	0,008	0,024	0	0,087	0,024	0,079	0	0,087	0,016	0,341	0,159
3210	0,039	0,016	0,008	0,016	0	0	0,008	0,008	0	0,008	0,008	0,031	0,023	0	0,008	0,008	0,023	0,124	0,008	0,023	0,008	0,132	0,101	0,403

11 pav.: Sąlyginės tikimybės, paskaičiuotos remiantis 80% 1 ir 5 metų trukmės obligacijų duomenimis

Kadangi yra turima 100% duomenų apie pagalbinę eilutę, t.y. apie 1 metų trukmės obligacijų užkoduotus duomenis, tai remiantis jais, yra žinoma, su kokia tikimybe kiekviena iš pagalbinės eilutės kombinacijų pasirodys po visų 3451 žingsnių. Šiuos duomenis jau turime iš praeitų skyrelių skaičiavimų – 9 lentelė. Padauginus pastarąjį vektorių iš sąlyginių tikimybių matricos, pavaizduotos 11 paveikslėlyje, yra gaunamos prognozės, su kokia tikimybe kiekviena iš 5 metų trukmės obligacijų užkoduotų duomenų kombinacijų bus po 3451 žingsnių (turint omenyje, kad pradedant prognozuoti duomenys buvo ties 2761 žingsniu – nes naudojamasi 80% visų duomenų).

Toliau bus pateikti visi rezultatai, gauti remiantis dvimačiu modeliu. Tokių atvejų, kad $level_{0,95}$ būtų ne nulis vėlgi nebuvo aptikta, kaip ir vienmačiu atveju, todėl šis vertinimas apačioje pateiktose lentelėse nebekartojamas.

	Prognozuojama finansinė eilutė			
Pagalbinė obligacijų eilutė	1 metų	5 metų	10 metų	30 metų
1 metų	-	3,96%	3,89%	3,19%
5 metų	4,65%	-	**2,97%**	**2,89%**
10 metų	**4,59%**	**2,73%**	-	3,25%
30 metų	4,69%	3,72%	3,65%	-

12 lentelė: Skirtingos trukmės obligacijų užkoduotų duomenų prognozės vidutinė absoliutinė procentinė paklaida, kai prognozei naudojama pagalbinė laiko eilutė ir 80% prognozuojamos laiko eilutės duomenų

	Prognozuojama finansinė eilutė			
Pagalbinė obligacijų eilutė	1 metų	5 metų	10 metų	30 metų
1 metų	-	70,83%	75,00%	**79,17%**
5 metų	**70,83%**	-	**79,17%**	70,83%
10 metų	**70,83%**	**83,33%**	-	75%
30 metų	66,67%	75,00%	**79,17%**	-

13 lentelė: Skirtingos trukmės obligacijų užkoduotų duomenų prognozės $level_{0,05}$, kai prognozei naudojama papildoma laiko eilutė ir 80% prognozuojamos laiko eilutės duomenų

	Prognozuojama finansinė eilutė			
Pagalbinė obligacijų eilutė	1 metų	5 metų	10 metų	30 metų
1 metų	-	**5,35%**	5,86%	8,00%
5 metų	**10,89%**	-	5,21%	6,72%
10 metų	12,19%	**5,35%**	-	**6,18%**
30 metų	12,70%	5,75%	**4,83%**	-

14 lentelė: Skirtingos trukmės obligacijų užkoduotų duomenų prognozės vidutinė absoliutinė procentinė paklaida, kai prognozei naudojama pagalbinė laiko eilutė ir 50% prognozuojamos laiko eilutės duomenų

	Prognozuojama finansinė eilutė			
Pagalbinė obligacijų eilutė	1 metų	5 metų	10 metų	30 metų
1 metų	-	**62,50%**	58,33%	37,50%
5 metų	**29,17%**	-	58,33%	**62,50%**
10 metų	**29,17%**	50,00%	-	**62,50%**
30 metų	20,83%	33,33%	**66,67%**	-

15 lentelė: Skirtingos trukmės obligacijų užkoduotų duomenų prognozės $level_{0,05}$, kai prognozei naudojama papildoma laiko eilutė ir 50% prognozuojamos laiko eilutės duomenų

Remiantis pateiktais rezultatais, pastebėtina, kad prognozuojant su pastaruoju modeliu, yra gaunami keli skirtingi rezultatai tos pačios trukmės obligacijoms. Tai priklauso nuo finansinės laiko eilutės, kuria yra remiamasi, skaičiuojant prognozę. Taip pat pažvelgus į 12 lentelę galima pastebėti įdomią asimetriją – jei prognozuojant 1 metų trukmės obligacijas

naudingiausia remtis 10 metų trukmės obligacijų eilute, tai nebūtinai reiškia, kad ir prognozuojant 10 metų trukmės obligacijų pelningumą naudingiausia yra remtis 1 metų trukmės obligacijų duomenimis, nes gauti rezultatai rodo, kad prognozuojant 10 metų trukmės obligacijas, tiklsiausi rezultatai gaunami, remiantis 5 metų trukmės obligacijų duomenimis (o ne 1 metų trukmės).

Kadangi mus domina, ar įmanoma dvimačiu atveju gauti tikslesnę prognozę nei vienmačiu atveju, išrinkime tiksliausius rezultatus (prognozuojant, remiantis 80% ir 50% duomenų) bei palyginkime rezultatus su tais, kurie buvo gauti prognozuojant vienmačiu atveju.

Obligacijų trukmė	1 metų	5 metų	10 metų	30 metų
Prognozė vienmačiu atveju				
$level_{0,05}$	66,67%	58,33%	70,83%	75,00%
MAPE	4,52%	3,88%	3,60%	2,97%
Prognozė dvimačiu atveju				
$level_{0,05}$	70,83%	83,33%	79,17%	79,17%
MAPE	4,59%	2,73%	2,97%	2,89%

16 lentelė: Skirtingos trukmės obligacijų užkoduotų duomenų prognozės vertinimas, taikant skirtingus modelius, kai prognozei naudojama 80% duomenų

Obligacijų trukmė	1 metų	5 metų	10 metų	30 metų
Prognozė vienmačiu atveju				
$level_{0,05}$	20,83%	37,50%	41,67%	54,17%
MAPE	12,08%	6,38%	6,22%	7,17%
Prognozė dvimačiu atveju				
$level_{0,05}$	29,17%	62,50%	66,67%	62,50%
MAPE	10,89%	5,35%	4,83%	6,18%

17 lentelė: Skirtingos trukmės obligacijų užkoduotų duomenų prognozės vertinimas, taikant skirtingus modelius, kai prognozei naudojama 50% duomenų

Lentelėse pateikti duomenys parodo, kad remiantis viena iš pagalbinių eilučių, prognozuojant skirtingų trukmių obligacijų pelningumą, galima gauti tikslesnes prognozes – tai įrodo visi trys vertinimo kriterijai.

Pirmiausia, prognozuojant dvimačiu modeliu, nebuvo aptikta nė viena paklaida, kuri

būtų didesnė už 95% (taip pat kaip ir vienmačiu atveju). Tai reiškia, kad remiantis šiuo vertinimo kriterijumi, dvimatis modelis yra ne prastesnis negu vienmatis.

Nagrinėjant kitą vertinimo kriterijų – absoliutinių procentinių paklaidų, kurios yra mažesnės arba lygios 5%, dalį (kitaip sakant paklaidų dalį, kuri yra nereikšminga), yra matoma, kad prognozuojant visų keturių trukmių obligacijų duomenis, $level_{0,05}$ yra gaunamas didesnis, nei vienmačiu atveju. Tai reiškia, kad prognozuojant dvimačiu atveju didesnė dalis paklaidų yra nereikšmingos.

Trečiasis vertinamas kriterijus – vidutinė absoliutinė procentinė paklaida – tik vienu iš nagrinėjamų atvejų buvo gauta šiek tiek prastesnė – tai 1 metų trukmės obligacijų prognozė, remiantis 80% duomenų. Šiuo atveju MAPE, prognozuojant su dvimačiu modeliu, buvo gauta 0,07% didesnė nei vienmačiu atveju, t.y. paklaida buvo šiek tiek didesnė, tačiau įvertinus tai, kad kitas kriterijus – $level_{0,05}$ dvimačiu atveju yra gaunamas 4,16% geresnis, negalima vienareikšmiškai teigti, kad prognozuojant 1 metų trukmės obligacijų duomenis dvimačiu atveju yra gaunami prastesni rezultatai negu vienmačiu atveju. Visų likusių – 5, 10, 30 metų trukmės obligacijų prognozės bei 1 metų trukmės obligacijų (kai prognozei naudojama 50% istorinių duomenų), remiantis MAPE vertinimo kriterijumi, tiksliau buvo nuprognozuotos, su dvimačiu modeliu.

Nustačius, jog dvimačio modelio atveju įmanoma gauti tikslesnius rezultatus, pabandykime panagrinėti ir išskirti ne tik pačius tiksliausius dvimačiu atveju gautus rezultatus, bet apskritai visus atvejus, kai prognozuojant dvimačiu atveju buvo gauti tikslesni rezultatai, nei vienmačiu atveju. 12 paveikslėlyje grafiškai pateikta informacija apie sąryšius tarp finansinių laiko eilučių, kurie buvo nagrinėjami 3.2 skyrelyje.

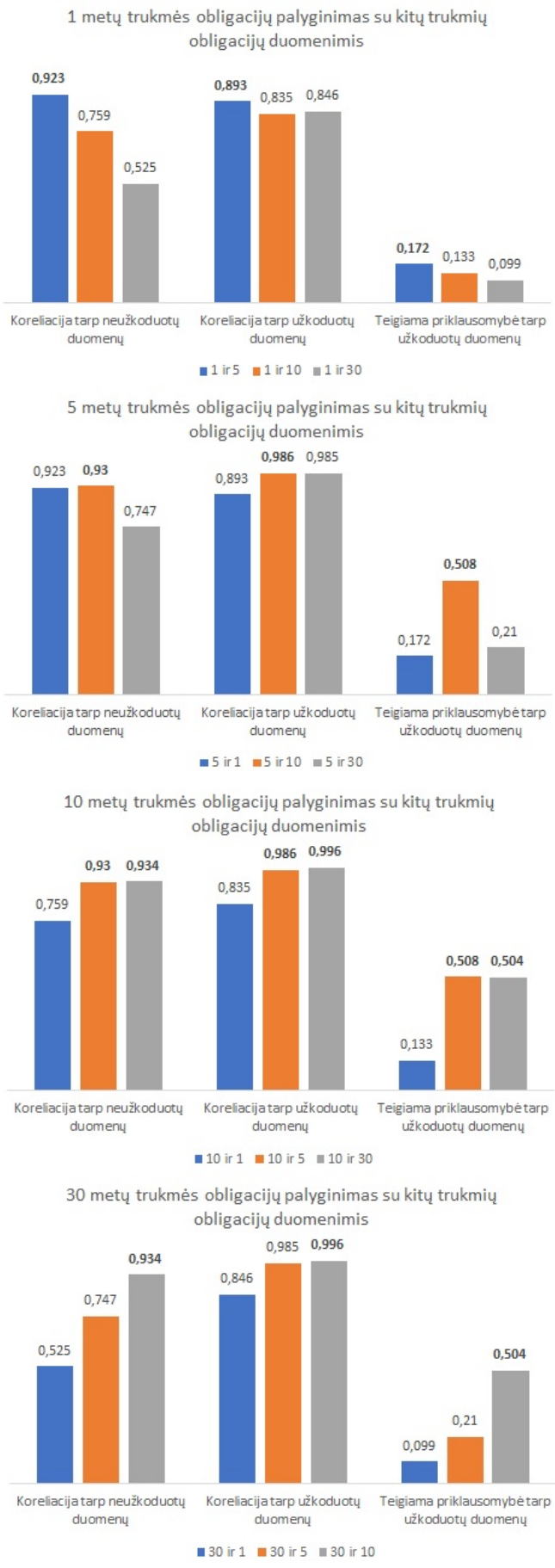
Paanalizuokime kiekvienų metų trukmės prognozės rezultatą dvimačiu atveju, atsižvelgiant į tos trukmės obligacijų sąryšį su kitų trukmių obligacijomis. Nagrinėjimui imkime kiek įdomesnį atvejį – kai prognozė remiasi mažesniu kiekiu istorinių duomenų, t.y. 50% duomenų. Prognozuojant 1 metų trukmės obligacijų duomenis dvimačiu atveju, tikslesnė prognozė (t.y. geresnė tiek $level_{0,05}$, tiek MAPE atžvilgiu) buvo gauta tik vienu iš galimų atvejų – remiantis 5 metų trukmės obligacijų duomenimis. Pažvelgus į 12 paveikslėlį, matome, jog nors ir negalima įvardinti konkrečios tendencijos tarp 1 metų ir 5, 10, 30 metų trukmės obligacijų (nes, pavyzdžiui, priklausomybė tarp duomenų ar koreliacija tarp užkoduotų duomenų skiriasi tik šimtosiomis skaičiaus dalimis), tačiau matome egzistuojančią teigiamą priklausomybę tarp laiko eilučių, kuri yra didžiausia būtent tarp 1 ir 5 metų trukmės obligacijų – 0,172, bei stiprią koreliaciją. Visais kitais atvejais (remiantis 10, 30 metų trukmės obligacijų duomenimis) tikslesnių prognozių nei vienmačiu atveju neaptikta.

Prognozuojant 5 metų trukmės obligacijų duomenis dvimačiu modeliu, geresni rezultatai nei su vienmačiu modeliu, remiantis MAPE vertinimu, buvo gauti visais atvejais, remiantis $level_{0,05}$ kriterijumi – prognozuojant su 1 ir 10 metų trukmės obligacijų duomenimis.

Prognozuojant 10 metų trukmės obligacijų duomenis su dvimačiu modeliu, geresni rezultatai nei su vienmačiu modeliu, remiantis abiem vertinimais, buvo gauti visais atvejais, nepaisant to, kad 12 paveikslėlyje tarp 10 ir 1 metų trukmės obligacijų teigiama priklausomybė yra stipriai mažesnė nei kitais dviem atvejais.

Prognozuojant 30 metų trukmės obligacijų duomenis dvimačiu atveju, geresni rezultatai nei vienmačiu atveju, remiantis abiem vertinimais, buvo gauti prognozuojant su 10 ir 5 metų trukmės obligacijų duomenimis. Žvelgiant į 12 paveikslėlio 4 grafiką, būtent ties šiais 30 ir 10 bei 30 ir 5 trukmės obligacijų duomenimis pastebima didesnė priklausomybė ir koreliacija nei tarp 30 ir 1 metų trukmės obligacijų.

Pastarosios įžvalgos tiksliai nenusako sąlygų, kurioms esant, dvimatis prognozavimo modelis yra tikslesnis negu vienmatis, tačiau galima daryti išvadas, kad esant tam tikram priklausomybės lygiui bei koreliacijai tarp užkoduotų finansinių laiko eilučių, galima gauti tikslesnes prognozes, taikant dvimatį modelį.



12 pav.: Koreliacija bei teigiama priklausomybė tarp skirtingų trukmių obligacijų duomenų

4 IŠVADOS

Šiame darbe buvo nagrinėjamas 1, 5, 10 ir 30 metų trukmės JAV vyriausybinių obligacijų pelningumas, užkoduotas ranginiais metodais. Kiekvienai iš užkoduotų eilučių buvo paskaičiuotas empirinis rangų blokų skirstinys bei rastas unikalus Markovo grandinės, sudarytos iš užkoduotų duomenų, stacionarusis skirstinys. Skirstiniai buvo lyginami, remiantis pilnosios variacijos metrika. Pilnosios variacijos metrikos rezultatai visais atvejais buvo gauti artimi 0, o tai reiškia, kad apskaičiuoti atitinkami empiriniai ir Markovo grandinės stacionarūs skirstiniai yra beveik lygūs arba, kitaip sakant, atitinkamos rangų blokų tikimybės, kai jas skaičiuojame empiriškai, iš esmės atitinka gautos Markovo grandinės stacionaraus skirstinio tikimybes. Taip pat, kiekvienai iš užkoduotų eilučių buvo paskaičiuota keitinių entropija, kuri parodė, kad nė vienos iš eilučių keitinių entropija nėra artima 0 (nė viena iš eilučių nėra visiškai nuspėjama), tačiau nėra ir visai atsitiktinė – keitinių entropija nėra lygi keitinių entropijai, kuri būtų, jei tikimybės pasirodytų ties kiekviena kombinacija su tokia pat tikimybe. Nubrėžus statistinio sudėtingumo entropijos kauzalinę plokštumą vėlgi nustatyta, kad eilutės nėra visiškai atsitiktinės, tačiau konkrečių išskirtinių savybių apie kažkurią iš eilučių nebuvo aptikta. Išnagrinėjus užkoduotas eilutes kiekvieną atskirai, taip pat buvo išnagrinėtas sąryšis tarp kiekvienos iš jų ir aptikta, kad egzistuoja teigiama priklausomybė, kuri mažėja didėjant skirtumui tarp obligacijų trukmės, bei stipri koreliacija.

Išnagrinėjus straipsnius, kuriuose dirbama su ranginiais metodais ir nustačius, kad iki šiol buvo nagrinėjamos tik tam tikros užkoduotų eilučių savybės, šiame darbe buvo nuspręsta ne tik panagrinėti savybes, bet ir prognozuoti užkoduotas eilutes. Prognozavimui buvo sukonstruoti du modeliai – vienmatis ir dvimatis. Vienmatis modelis remiasi tik prognozuojamos eilutės užkoduotais duomenimis ir prognozuojant veiksmai buvo atliekami su Markovo grandinėmis. Dvimatis modelis remiasi ne tik prognozuojamos eilutės istoriniais duomenimis, bet ir kitos – pagalbinės eilutės – duomenimis, šis modelis sukonstruotas naudojantis sąlyginėmis tikimybėmis ir pilnos tikimybės formule. Prognozes įvertinus trimis kriterijais: vidutine absoliutine procentine paklaida, paklaidų dalimi mažesne už 5% bei paklaidų dalimi didesne už 95%, buvo nustatyta, kad prognozuojant su dvimačiu modeliu galima gauti tikslesnius rezultatus, nei prognozuojant tas pačias eilutes su vienmačiu modeliu.

Pastarieji rezultatai leidžia daryti išvadą, kad turint duomenų apie kitą užkoduotą laiko eilutę, kuri turi tam tikrų sąryšių su prognozuojama eilute, galima gauti tikslesnę tos eilutės prognozę.

Tolimesniame darbe įdomu būtų išnagrinėti kuo daugiau galimų kriterijų apie ryšius

tarp užkoduotų finansinių laiko eilučių ir rasti tikslias sąlygas, kuriomis remiantis dvimatis modelis visada duoda tikslesnius rezultatus, nei vienmatis.

5 LITERATŪRA

[1] JM Amigo, Karsten Keller, and VA Unakafova. Ordinal symbolic analysis and its application to biomedical recordings. *Phil. Trans. R. Soc*, 373:20140091, 2015.

[2] Christoph Bandt and Bernd Pompe. Permutation entropy: A natural complexity measure for time series. *Physical review letters*, 88:174102, 05 2002.

[3] Sónia Bentes, Rui Menezes, and Diana Mendes. Long memory and volatility clustering: is the empirical evidence consistent across stock markets? *arXiv.org, Quantitative Finance Papers*, 387, 09 2007.

[4] Pierre Bremaud. Basic markov chains. `https://www.di.ens.fr/~busic/cours/M2amis/markov.pdf`.

[5] Lothar Breuer. Introduction to stochastic processes. *Lecture Notes, Univ. Kent*, 2014.

[6] Taolue Chen and Stefan Kiefer. On the total variation distance of labelled markov chains. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, page 33. ACM, 2014.

[7] Karsten Keller and Heinz Lauffer. Symbolic analysis of high-dimensional time series. *International Journal of Bifurcation and Chaos*, 13:2657–2668, 09 2003.

[8] Jonas Kubilius. *Tikimybių teorija ir matematinė statistika. Antrasis leidimas.* Vilniaus universiteto leidykla, 1996.

[9] Macrotrends. U.s. treasury rate. `https://www.macrotrends.net/`.

[10] Henry Maltby, Samir Khan, and Jimin Khim. Stationary distributions of markov chains. `https://www.statisticshowto.datasciencecentral.com/mean-absolute-percentage-error-mape/`.

[11] Omar Olvera-Guerrero, Alfonso Prieto-Guerrero, and Gilberto Espinosa-Paredes. A non-linear stability monitor for boiling water reactors. 08 2019.

[12] Maik Riedl, Andreas Müller, and Niels Wessel. Practical considerations of permutation entropy: A tutorial review. *The European Physical Journal Special Topics*, 222, 06 2013.

[13] Osvaldo Rosso, Hilda Larrondo, M.T. Martin, A. Plastino, and Miguel Fuentes. Distinguishing noise from chaos. *Physical review letters*, 99:154102, 11 2007.

[14] Alexander Schnurr. An ordinal pattern approach to detect and to model leverage effects and dependence structures between financial time series. 07 2012.

[15] Statistics How To. Mean absolute percentage error (mape). `https://www.statisticshowto.datasciencecentral.com/mean-absolute-percentage-error-mape/`.

[16] Luciano Zunino, Aurelio F. Bariviera, M. Belén Guercio, Lisana Martinez, and Osvaldo Rosso. On the efficiency of sovereign bond markets. *Physica A Statistical and Theoretical Physics*, 391:4342–4349, 10 2012.

A Priedas

A.1 R kodas

```
library(diagram)
library(magrittr)
library(shapes)
library(Gmisc)
library(MmgraphR)
library(markovchain)
library(CRANsearcher)
library(backports)
library(ggplot2)
library(Gmisc)
library(matlab)
library(markovchain)
library(msm)
library(mstate)
library(shape)
library(shapes)
library(MmgraphR)
library(arrow)
library(Matrix)
library(igraph)
library(matlab)
library(readxl)
library("readr")

#dabar bus dirbama su 1, 5, 10, 30 metų trukmės obligacijų
    pelningumo duomenimis, užkoduotais
#ranginiais metodais (su C sharp parašyta programa), kai bloko
    dydis 4, 5, 6
#one_4 reiškia, kad nagrinėjami 1 metų trukmės obligacijų už
    koduoti duomenys, kai bloko dydis 4
```

```
#analogiškai su kitų trukmių ir bloko dydžių duomenimis (pavyzdž
    iui, five_6 - 5 metų trukmės užkoduoti duomenys, kai bloko
    dydis 6)
one_4<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R_
    project/1_4.csv")
#nuskaitomos užkoduotos 1 metų trukmės obligacijų kainos, kai
    rango dydis 4
one_4 <- markovchainFit(data=one_4)
one_4<-one_4[["estimate"]]@transitionMatrix
#iš nuskaitytų užkoduotų duomenų yra sukuriama Markovo grandinė ir
     kintamajam one_4 priskiriama grandinės perėjimo tikimybių
    matrica
one_4_markov<-new("markovchain",transitionMatrix = one_4)
#dabar tikrinsime tyrimui reikalingas Markovo grandinės savybes
is.irreducible(one_4_markov)
#patikrinama, ar Markovo grandinė nesuskaidoma
recurrentStates(one_4_markov)
#pažiūrima, ar visos būsenos rekurentinės
meanRecurrenceTime(one_4_markov)
#pažiūrima, ar vidutinis žingsnių skaičius iš i-osios į i-ąją bū
    seną yra baigtinis
graphics.off()
par("mar")
par(mar=c(1,1,1,1))
plot(one_4_markov, vertex.size = 15, edge.arrow.size=0.05)
#nubrėžiama Markovo grandinės perėjimų diagrama
vienetai<-c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
one_4_solve<-data.matrix(rbind(t(one_4)-diag(24),vienetai))
b<- c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1)
stat_1_4<- data.matrix(drop(solve(t(one_4_solve) %*% one_4_solve,
    t(one_4_solve) %*% b)))
#apskaičiuojamas stacionarusis Markovo grandinės skirstinys
    skirstinys
```

```
five_4<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R_
    project/5_4.csv")
five_4 <- markovchainFit(data=five_4)
five_4<-five_4[["estimate"]]@transitionMatrix
five_4_markov<-new("markovchain",transitionMatrix = five_4)
is.irreducible(five_4_markov)
recurrentStates(five_4_markov)
meanRecurrenceTime(five_4_markov)
graphics.off()
par("mar")
par(mar=c(1,1,1,1))
plot(five_4_markov, vertex.size = 15, edge.arrow.size=0.05)
five_4_solve<-data.matrix(rbind(t(five_4)-diag(24),vienetai))
stat_5_4<- data.matrix(drop(solve(t(five_4_solve) %*% five_4_solve
    , t(five_4_solve) %*% b)))
#analogiški veiksmai atliekami su 5 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 4


ten_4<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R_
    project/10_4.csv")
ten_4 <- markovchainFit(data=ten_4)
ten_4<-ten_4[["estimate"]]@transitionMatrix
ten_4_markov<-new("markovchain",transitionMatrix = ten_4)
is.irreducible(ten_4_markov)
recurrentStates(ten_4_markov)
meanRecurrenceTime(ten_4_markov)
ten_4_solve<-data.matrix(rbind(t(ten_4)-diag(24),vienetai))
stat_10_4<- data.matrix(drop(solve(t(ten_4_solve) %*% ten_4_solve,
     t(ten_4_solve) %*% b)))
#analogiški veiksmai atliekami su 10 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 4
```

```
thirty_4<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R_
    project/30_4.csv")
thirty_4 <- markovchainFit(data=thirty_4)
thirty_4<-thirty_4["estimate"]]@transitionMatrix
thirty_4_markov<-new("markovchain",transitionMatrix = thirty_4)
is.irreducible(thirty_4_markov)
recurrentStates(thirty_4_markov)
meanRecurrenceTime(thirty_4_markov)
thirty_4_solve<-data.matrix(rbind(t(thirty_4)-diag(24),vienetai))
stat_30_4<- data.matrix(drop(solve(t(thirty_4_solve) %*% thirty_4_
    solve, t(thirty_4_solve) %*% b)))
#analogiški veiksmai atliekami su 30 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 4


one_5<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R_
    project/1_5.csv")
one_5_markov<- markovchainFit(data=one_5)
is.irreducible(new("markovchain",transitionMatrix = one_5_markov[[
    "estimate"]]@transitionMatrix))
recurrentStates(new("markovchain",transitionMatrix = one_5_markov
    [["estimate"]]@transitionMatrix))
meanRecurrenceTime(new("markovchain",transitionMatrix = one_5_
    markov[["estimate"]]@transitionMatrix))
one_5<- markovchainFit(data=one_5,possibleStates = "31042")
#paprastumo dėlei (kad sudaryta perėjimo tikimybių matrica būtų
    120x120), veiksmams atlikti pridedame būseną, kuri nepasirodo š
    ioje grandinėje, nors yra viena iš galimų 120 kombinacijų)
one_5<-one_5["estimate"]]@transitionMatrix
vienetai_5=c()
for (i in 1:120) (vienetai_5[i]=1)
one_5_solve<-data.matrix(rbind(t(one_5)-diag(120),vienetai_5))
b_5=c()
for (i in 1:121) (b_5[i]=0)
```

```
b_5[121]=1
stat_1_5<- data.matrix(drop(solve(t(one_5_solve) %*% one_5_solve,
    t(one_5_solve) %*% b_5)))
#analogiški veiksmai atliekami su 1 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 5

five_5<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R_
    project/5_5.csv")
five_5 <- markovchainFit(data=five_5)
five_5<-five_5["estimate"]]@transitionMatrix
five_5_markov<-new("markovchain",transitionMatrix = five_5)
is.irreducible(five_5_markov)
recurrentStates(five_5_markov)
meanRecurrenceTime(five_5_markov)
five_5_solve<-data.matrix(rbind(t(five_5)-diag(120),vienetai_5))
stat_5_5<- data.matrix(drop(solve(t(five_5_solve) %*% five_5_solve
    , t(five_5_solve) %*% b_5)))
#analogiški veiksmai atliekami su 5 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 5

ten_5<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R_
    project/10_5.csv")
ten_5 <- markovchainFit(data=ten_5)
ten_5<-ten_5["estimate"]]@transitionMatrix
ten_5_markov<-new("markovchain",transitionMatrix = ten_5)
is.irreducible(ten_5_markov)
recurrentStates(ten_5_markov)
meanRecurrenceTime(ten_5_markov)
ten_5_solve<-data.matrix(rbind(t(ten_5)-diag(120),vienetai_5))
stat_10_5<- data.matrix(drop(solve(t(ten_5_solve) %*% ten_5_solve,
     t(ten_5_solve) %*% b_5)))
#analogiški veiksmai atliekami su 10 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 5
```

```
thirty_5<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R_
    project/30_5.csv")
thirty_5 <- markovchainFit(data=thirty_5)
thirty_5<-thirty_5[["estimate"]]@transitionMatrix
thirty_5_markov<-new("markovchain",transitionMatrix = thirty_5)
is.irreducible(thirty_5_markov)
recurrentStates(thirty_5_markov)
meanRecurrenceTime(thirty_5_markov)
thirty_5_solve<-data.matrix(rbind(t(thirty_5)-diag(120),vienetai_
    5))
stat_30_5<- data.matrix(drop(solve(t(thirty_5_solve) %*% thirty_5_
    solve, t(thirty_5_solve) %*% b_5)))
#analogiški veiksmai atliekami su 30 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 5

busenu_pildymas_1<-readLines("C:/Users/ekabi/OneDrive/Desktop/
    Magistras/R_project/Trukstamos_busenos_1.csv")
one_6<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R_
    project/1_6.csv")
one_6_markov<- markovchainFit(data=one_6)
is.irreducible(new("markovchain",transitionMatrix = one_6_markov[[
    "estimate"]]@transitionMatrix))
recurrentStates(new("markovchain",transitionMatrix = one_6_markov
    [["estimate"]]@transitionMatrix))
meanRecurrenceTime(new("markovchain",transitionMatrix = one_6_
    markov[["estimate"]]@transitionMatrix))
one_6<- markovchainFit(data=one_6,possibleStates =data.matrix(
    busenu_pildymas_1))
one_6<-one_6[["estimate"]]@transitionMatrix
vienetai_6=c()
for (i in 1:720) (vienetai_6[i]=1)
one_6_solve<-data.matrix(rbind(t(one_6)-diag(720),vienetai_6))
```

```
b_6=c()
for (i in 1:721) (b_6[i]=0)
b_6[721]=1
stat_1_6<- data.matrix(drop(solve(t(one_6_solve) %*% one_6_solve,
    t(one_6_solve) %*% b_6)))
write.table(stat_1_6, file = "stat_1_6.txt",sep=",")
#analogiški veiksmai atliekami su 1 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 6

busenu_pildymas_2<-readLines("C:/Users/ekabi/OneDrive/Desktop/
    Magistras/R project/Trukstamos_busenos_5.csv")
five_6<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R
    project/5_6.csv")
five_6_markov<- markovchainFit(data=five_6)
is.irreducible(new("markovchain",transitionMatrix = five_6_markov
    [["estimate"]]@transitionMatrix))
recurrentStates(new("markovchain",transitionMatrix = five_6_markov
    [["estimate"]]@transitionMatrix))
meanRecurrenceTime(new("markovchain",transitionMatrix = five_6_
    markov[["estimate"]]@transitionMatrix))
five_6<- markovchainFit(data=five_6,possibleStates =data.matrix(
    busenu_pildymas_2))
five_6<-five_6[["estimate"]]@transitionMatrix
five_6_solve<-data.matrix(rbind(t(five_6)-diag(720),vienetai_6))
stat_5_6<- data.matrix(drop(solve(t(five_6_solve) %*% five_6_solve
    , t(five_6_solve) %*% b_6)))
write.table(stat_5_6, file = "stat_5_6.txt",sep=",")
#analogiški veiksmai atliekami su 5 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 6

busenu_pildymas_3<-readLines("C:/Users/ekabi/OneDrive/Desktop/
    Magistras/R project/Trukstamos_busenos_10.csv")
ten_6<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R
```

```
    project/10_6.csv")
ten_6_markov<- markovchainFit(data=ten_6)
is.irreducible(new("markovchain",transitionMatrix = ten_6_markov[[
    "estimate"]]@transitionMatrix))
recurrentStates(new("markovchain",transitionMatrix = ten_6_markov
    [["estimate"]]@transitionMatrix))
meanRecurrenceTime(new("markovchain",transitionMatrix = ten_6_
    markov[["estimate"]]@transitionMatrix))
ten_6<- markovchainFit(data=ten_6,possibleStates =data.matrix(
    busenu_pildymas_3))
ten_6<-ten_6[["estimate"]]@transitionMatrix
ten_6_solve<-data.matrix(rbind(t(ten_6)-diag(720),vienetai_6))
stat_10_6<- data.matrix(drop(solve(t(ten_6_solve) %*% ten_6_solve,
     t(ten_6_solve) %*% b_6)))
write.table(stat_10_6, file = "stat_10_6.txt",sep=",")
#analogiški veiksmai atliekami su 10 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 6

busenu_pildymas_4<-readLines("C:/Users/ekabi/OneDrive/Desktop/
    Magistras/R_project/Trukstamos_busenos_30.csv")
thirty_6<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R_
    project/30_6.csv")
thirty_6_markov<- markovchainFit(data=thirty_6)
is.irreducible(new("markovchain",transitionMatrix = thirty_6_
    markov[["estimate"]]@transitionMatrix))
recurrentStates(new("markovchain",transitionMatrix = thirty_6_
    markov[["estimate"]]@transitionMatrix))
meanRecurrenceTime(new("markovchain",transitionMatrix = thirty_6_
    markov[["estimate"]]@transitionMatrix))
thirty_6<- markovchainFit(data=thirty_6,possibleStates =data.
    matrix(busenu_pildymas_4))
thirty_6<-thirty_6[["estimate"]]@transitionMatrix
thirty_6_solve<-data.matrix(rbind(t(thirty_6)-diag(720),vienetai_
```

```
    6))
stat_30_6<- data.matrix(drop(solve(t(thirty_6_solve) %*% thirty_6_
    solve, t(thirty_6_solve) %*% b_6)))
write.table(stat_30_6, file = "stat_30_6.txt",sep=",")
#analogiški veiksmai atliekami su 30 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 6


'PROGNOZAVIMAS VIENMAČIU ATVEJU SU 80 PROC. DUOMENŲ'
'1 metų'
one_4_80<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R
    project/1_4_80.csv")
one_4_80<- markovchainFit(data=one_4_80)
one_4_80<-one_4_80[["estimate"]]@transitionMatrix
one_4_markov_80<-new("markovchain",transitionMatrix = one_4_80,
    states = c("0123", "0132", "0213", "0231", "0312", "0321", "
    1023", "1032", "1203", "1230", "1302", "1320", "2013", "2031",
    "2103", "2130", "2301", "2310", "3012", "3021", "3102", "3120",
     "3201", "3210"), name="MarkovChain A")
graphics.off()
par("mar")
par(mar=c(1,1,1,1))
plot(one_4_markov_80, vertex.size = 13,edge.arrow.size=0.08)
#sudaroma 1 metų trukmės obligacijų Markovo grandinė sudaryta iš
    80% visų turimų duomenų ir nubrėžiama tos Markovo grandinės per
    ėjimų diagrama
vienetai_4=c()
for (i in 1:24) (vienetai_4[i]=1)
one_4_80_solve<-data.matrix(rbind(t(one_4_80)-diag(24),vienetai_4)
    )
b_4=c()
for (i in 1:25) (b_4[i]=0)
b_4[25]=1
stat_1_4_80<- data.matrix(drop(solve(t(one_4_80_solve) %*% one_4_
```

```
   80_solve, t(one_4_80_solve) %*% b_4)))
write.table(stat_1_4_80, file = "stat_1_4_80.txt",sep=",")
b_4_stat_1_80=c()
for (i in 1:24) (b_4_stat_1_80[i]=0)
b_4_stat_1_80[1]=1
#iš turimų duomenų apskaičiuojame stacionarųjį skirstinį, tam, kad
    nustatytumėme ties kuria kombinacija yra didžiausia tikimybė
   atsidurti sekančiame žingsnyje
forecast_1_80_4<-b_4_stat_1_80
for (i in 1:690)
  (forecast_1_80_4<-forecast_1_80_4%*%one_4_80)
write.table(forecast_1_80_4, file = "forecast_1_80_4.txt",sep=",")
#tikimybių vektorius (su kokia tikimybe kiekviena iš kombinacijų
   pasirodys - pirmą kartą dauginant šiame vektoriuje ties
#kombinacija, kuriai pasirodyti yra didžiaudžia tikimybė pagal
   stacionarųjį skirstinį, tikimybė įrašoma 1, o toliau dauginama
   vis iš kito
#tikimybių vektoriaus forecast_1_80_4, gauto po kiekvieno žingsnio
    prognozės) yra dauginamas iš perėjimo matricos tiek kartų,
   kiek žingsnių į priekį
#norima nuprognozuoti
'5 metų'

five_4_80<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R
   project/5_4_80.csv")
five_4_80<- markovchainFit(data=five_4_80)
five_4_80<-five_4_80[["estimate"]]@transitionMatrix
five_4_80_solve<-data.matrix(rbind(t(five_4_80)-diag(24),vienetai_
   4))
stat_5_4_80<- data.matrix(drop(solve(t(five_4_80_solve) %*% five_4
   _80_solve, t(five_4_80_solve) %*% b_4)))
write.table(stat_5_4_80, file = "stat_5_4_80.txt",sep=",")
b_4_stat_5_80=c()
```

```
for (i in 1:24) (b_4_stat_5_80[i]=0)
b_4_stat_5_80[1]=1
forecast_5_80_4<-b_4_stat_5_80
for (i in 1:690)
  (forecast_5_80_4<-forecast_5_80_4%*%five_4_80)
write.table(forecast_5_80_4, file = "forecast_5_80_4.txt",sep=",")
#analogiški veiksmai atliekami su 5 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 4

'10 metų'
ten_4_80<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R
    project/10_4_80.csv")
ten_4_80<- markovchainFit(data=ten_4_80)
ten_4_80<-ten_4_80[["estimate"]]@transitionMatrix
ten_4_80_solve<-data.matrix(rbind(t(ten_4_80)-diag(24),vienetai_4)
    )
stat_10_4_80<- data.matrix(drop(solve(t(ten_4_80_solve) %*% ten_4_
    80_solve, t(ten_4_80_solve) %*% b_4)))
write.table(stat_10_4_80, file = "stat_10_4_80.txt",sep=",")
b_4_stat_10_80=c()
for (i in 1:24) (b_4_stat_10_80[i]=0)
b_4_stat_10_80[1]=1
forecast_10_80_4<-b_4_stat_10_80
for (i in 1:690)
  (forecast_10_80_4<-forecast_10_80_4%*%ten_4_80)
write.table(forecast_10_80_4, file = "forecast_10_80_4.txt",sep=",
    ")
#analogiški veiksmai atliekami su 10 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 4

'30 metų'
thirty_4_80<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/
    R project/30_4_80.csv")
```

```
thirty_4_80<- markovchainFit(data=thirty_4_80)
thirty_4_80<-thirty_4_80[["estimate"]]@transitionMatrix
thirty_4_80_solve<-data.matrix(rbind(t(thirty_4_80)-diag(24),
    vienetai_4))
stat_30_4_80<- data.matrix(drop(solve(t(thirty_4_80_solve) %*%
    thirty_4_80_solve, t(thirty_4_80_solve) %*% b_4)))
write.table(stat_30_4_80, file = "stat_30_4_80.txt",sep=",")
b_4_stat_30_80=c()
for (i in 1:24) (b_4_stat_30_80[i]=0)
b_4_stat_30_80[1]=1
forecast_30_80_4<-b_4_stat_30_80
for (i in 1:690)
  (forecast_30_80_4<-forecast_30_80_4%*%thirty_4_80)
write.table(forecast_30_80_4, file = "forecast_30_80_4.txt",sep=",
    ")
#analogiški veiksmai atliekami su 30 metų trukmės obligacijų už
    koduotais duomenimis, kai bloko dydis 4

'PROGNOZAVIMAS VIENMAČIU ATVEJU SU 50 PROC. DUOMENŲ'
#toliau bus analogiškai, kaip iki šiol, prognozuojami duomenys,
    tik dabar jau naudojantis 50% duomenų
'1 metų'
one_4_50<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R_
    project/1_4_50.csv")
one_4_50<- markovchainFit(data=one_4_50)
one_4_50<-one_4_50[["estimate"]]@transitionMatrix
vienetai_4=c()
for (i in 1:24) (vienetai_4[i]=1)
one_4_50_solve<-data.matrix(rbind(t(one_4_50)-diag(24),vienetai_4)
    )
b_4=c()
for (i in 1:25) (b_4[i]=0)
b_4[25]=1
```

```
stat_1_4_50<- data.matrix(drop(solve(t(one_4_50_solve) %*% one_4_
   50_solve, t(one_4_50_solve) %*% b_4)))
write.table(stat_1_4_50, file = "stat_1_4_50.txt",sep=",")
b_4_stat_1_50=c()
for (i in 1:24) (b_4_stat_1_50[i]=0)
b_4_stat_1_50[1]=1
forecast_1_50_4<-b_4_stat_1_50
for (i in 1:1725)
  (forecast_1_50_4<-forecast_1_50_4%*%one_4_50)
write.table(forecast_1_50_4, file = "forecast_1_50_4.txt",sep=",")

'5 metų'
five_4_50<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R
   project/5_4_50.csv")
five_4_50<- markovchainFit(data=five_4_50)
five_4_50<-five_4_50[["estimate"]]@transitionMatrix
five_4_50_solve<-data.matrix(rbind(t(five_4_50)-diag(24),vienetai_
   4))
stat_5_4_50<- data.matrix(drop(solve(t(five_4_50_solve) %*% five_4
   _50_solve, t(five_4_50_solve) %*% b_4)))
write.table(stat_5_4_50, file = "stat_5_4_50.txt",sep=",")
b_4_stat_5_50=c()
for (i in 1:24) (b_4_stat_5_50[i]=0)
b_4_stat_5_50[1]=1
forecast_5_50_4<-b_4_stat_5_50
for (i in 1:1725)
  (forecast_5_50_4<-forecast_5_50_4%*%five_4_50)
write.table(forecast_5_50_4, file = "forecast_5_50_4.txt",sep=",")

'10 metų'
ten_4_50<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/R
   project/10_4_50.csv")
ten_4_50<- markovchainFit(data=ten_4_50)
```

```
ten_4_50<-ten_4_50[["estimate"]]@transitionMatrix
ten_4_50_solve<-data.matrix(rbind(t(ten_4_50)-diag(24),vienetai_4)
    )
stat_10_4_50<- data.matrix(drop(solve(t(ten_4_50_solve) %*% ten_4_
    50_solve, t(ten_4_50_solve) %*% b_4)))
write.table(stat_10_4_50, file = "stat_10_4_50.txt",sep=",")
b_4_stat_10_50=c()
for (i in 1:24) (b_4_stat_10_50[i]=0)
b_4_stat_10_50[1]=1
forecast_10_50_4<-b_4_stat_10_50
for (i in 1:1725)
  (forecast_10_50_4<-forecast_10_50_4%*%ten_4_50)
write.table(forecast_10_50_4, file = "forecast_10_50_4.txt",sep=",
    ")


'30␣metų'
thirty_4_50<-readLines("C:/Users/ekabi/OneDrive/Desktop/Magistras/
    R␣project/30_4_50.csv")
thirty_4_50<- markovchainFit(data=thirty_4_50)
thirty_4_50<-thirty_4_50[["estimate"]]@transitionMatrix
thirty_4_50_solve<-data.matrix(rbind(t(thirty_4_50)-diag(24),
    vienetai_4))
stat_30_4_50<- data.matrix(drop(solve(t(thirty_4_50_solve) %*%
    thirty_4_50_solve, t(thirty_4_50_solve) %*% b_4)))
write.table(stat_30_4_50, file = "stat_30_4_50.txt",sep=",")
b_4_stat_30_50=c()
for (i in 1:24) (b_4_stat_30_50[i]=0)
b_4_stat_30_50[1]=1
forecast_30_50_4<-b_4_stat_30_50
for (i in 1:1725)
  (forecast_30_50_4<-forecast_30_50_4%*%thirty_4_50)
write.table(forecast_30_50_4, file = "forecast_30_50_4.txt",sep=",
    ")
```

A.2 *C sharp* programinis kodas

```
namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Pasirinkite metodą: (1) ; (2) ; (3
                - išvesti į excelį tik (2) būdu užkoduotus duomenis
                ); (4 - - išvesti į excelį tik (1) būdu užkoduotus
                duomenis");
            var methodType = Console.ReadLine();
            Console.WriteLine("Įrašykite koduojamos kombinacijos
                dydį:");
            var paternSize = Console.ReadLine();
            Console.WriteLine("Įrašykite 1, jei bus koduojama pata
                škiui, arba kombinacijos dydį, jei bus koduojama
                blokais: ");
            var delay = Console.ReadLine();
            Console.WriteLine("Kuri priklausomybė turi būti paskai
                čiuota? (1 - Neigiama) ; (2 - Teigiama)");
            var reverse = Console.ReadLine();
            //Sukuriame langą, į kurį vedame komandas, ką programa
                turi skaičiuoti ir su kokiais parametrais.
            var stopwatch = new Stopwatch();//Inicijuojame objektą
                , kuris naudojamas suskaičiuoti, kiek laiko
                programa sugaišta atliekant skaičiavimus.
            var excelHandler = new ExcelHandler();//Inicijuojame
                objektą, per kurį bus pasiekiamos skaičiavimų
                funkcijos.
            excelHandler.ReadData();//Nuskaitome pirmuosius ir
                antruosius duomenis iš excel failo.
```

```
switch (methodType) //Pagal įvestą metodo numerį
    nurodome, ką programa turėtų daryti.
{
     case "1":
          stopwatch.Start();
          if (reverse == "1")
          {
               var results = new Dictionary<int, decimal
                  >();
               for (int i = 2; i <= int.Parse(paternSize)
                  ; i++)
               {
                    excelHandler.ProcessDataOneFirst(i,
                       int.Parse(delay));
                    excelHandler.
                       ProcessDataOneSecondReversed(i, int
                       .Parse(delay));
                    var calculatedValue = excelHandler.
                       CalculateDependenceLevel();
                    results.Add(i, calculatedValue);
               }
               excelHandler.WriteCalculatedData(results);
          }
          else
          {
               var results = new Dictionary<int, decimal
                  >();
               for (int i = 2; i <= int.Parse(paternSize)
                  ; i++)
               {
                    excelHandler.ProcessDataOneFirst(i,
                       int.Parse(delay));
```

```
                excelHandler.ProcessDataOneSecond(i,
                    int.Parse(delay));
                var calculatedValue = excelHandler.
                    CalculateDependenceLevel();
                results.Add(i, calculatedValue);
            }
            excelHandler.WriteCalculatedData(results);
        }

        excelHandler.WriteProcessedData();
        break;
        /*Pirmuoju metodu duomenys yra užkoduojami
            tokiu skaičiu skirtingų simbolių, koks yra
            rango dydis.
        Pasirinkus pirmąjį metodą yra patikrinama,
            kuri priklausomybė - teigiama ar neigiama
            turi būti suskaičiuota.
        Nuo minimalaus rango dydžio (2) iki įvesto
            rango dydžio yra užkoduojami pagal kitus į
            vestus parametrus ir tada
        paskaičiuojamos priklausomybės tarp dviejų už
            koduotų duomenų eilučių, kai rango dydis
            2,3,...,įvestas rango dydis.
        */
    case "2":
        stopwatch.Start();
        if(reverse == "1")
        {
            var results = new Dictionary<int, decimal
                >();
            for (int i = 2; i <= int.Parse(paternSize)
                ; i++)
            {
```

```
                excelHandler.ProcessDataTwoFirst(i,
                    int.Parse(delay));
                excelHandler.
                    ProcessDataTwoSecondReversed(i, int
                    .Parse(delay));
                var calculatedValue = excelHandler.
                    CalculateDependenceLevel();
                results.Add(i, calculatedValue);
            }
            excelHandler.WriteCalculatedData(results);
        }
        else
        {
            var results = new Dictionary<int, decimal
                >();
            for (int i = 2; i <= int.Parse(paternSize)
                ; i++)
            {
                excelHandler.ProcessDataTwoFirst(i,
                    int.Parse(delay));
                excelHandler.ProcessDataTwoSecond(i,
                    int.Parse(delay));
                var calculatedValue = excelHandler.
                    CalculateDependenceLevel();
                results.Add(i, calculatedValue);
            }
            excelHandler.WriteCalculatedData(results);
        }

        break;
    /* Pasirinkus 2 metodą - vykdomi analogiški
       veiksmai, kaip ir 1 - uoju metodu, tačiau dabar
    duomenys yra koduojami į įvesto rango dydžio
```

```
            vektorius dvejetainiais skaičiais - 0 ir 1.
        */
    case "3":
        stopwatch.Start();
        if (reverse == "1")
        {
            excelHandler.ProcessDataTwoFirst(int.Parse
                (paternSize), int.Parse(delay));
            excelHandler.ProcessDataTwoSecondReversed(
                int.Parse(paternSize), int.Parse(delay)
                );
        }
        else
        {
            excelHandler.ProcessDataTwoFirst(int.Parse
                (paternSize), int.Parse(delay));
            excelHandler.ProcessDataTwoSecond(int.
                Parse(paternSize), int.Parse(delay));
        }
        excelHandler.WriteProcessedData();
        break;
      /* Šiuo atveju priklausomybė nėra skaičiuojama -
          duomenys yra tik užkoduojami (2) būdu ir iš
         vedami į excel failą.
       */
    case "4":
        stopwatch.Start();
        if (reverse == "1")
        {
            excelHandler.ProcessDataOneFirst(int.Parse
                (paternSize), int.Parse(delay));
            excelHandler.ProcessDataOneSecondReversed(
                int.Parse(paternSize), int.Parse(delay)
```

```
                );
            }
            else
            {
                excelHandler.ProcessDataOneFirst(int.Parse
                    (paternSize), int.Parse(delay));
                excelHandler.ProcessDataOneSecond(int.
                    Parse(paternSize), int.Parse(delay));
            }
            string inputLine = "";
            for (int i = 0; i < int.Parse(paternSize); i
                ++)
            {
                inputLine += i;
            }
            Combinations combinationsObj = new
                Combinations();
            combinationsObj.InputSet = combinationsObj.
                MakeCharArray(inputLine);
            combinationsObj.CalcPermutation(0);
            excelHandler.AllPossibleCombinations =
                combinationsObj.CombinationValues;
            excelHandler.WriteProcessedData();
            break;
        /* Šiuo atveju priklausomybė nėra skaičiuojama
            taip pat - duomenys yra tik užkoduojami (1) bū
            du ir išvedami į excel failą.
        Taip pat išvedamos visos galimos įvesto rango dydž
            io kombinacijos (t.y. n! kombinacijų).
        */
        default:
            Console.WriteLine("Įvestas metodas nerastas");
            break;
```

```
            }

            stopwatch.Stop();
            Console.WriteLine("Vykdymo laikas: " + stopwatch.
                ElapsedMilliseconds/1000m);
            //Išvedame paskaičiuotą programos vykdymo laiką.
            Console.ReadKey();
        }
    }
}


namespace ConsoleApp
{
    public class ExcelHandler
    {
        public List<string> FirstData { get; set;}
        //Kintamasis, į kurį nusiskaitoma pirmoji duomenų eilutė.
        public List<string> ProcessedFirstData { get; set; }
        //Kintamasis, į kurį dedami užkoduoti pirmi duomenys.
        public List<string> SecondData { get; set; }
        //Kintamasis, į kurį nuskaitoma antroji duomenų eilutė.
        public List<string> ProcessedSecondData { get; set; }
        //Kintamasis, į kurį dedami užkoduoti antri duomenys.
        public List<string> AllPossibleCombinations { get; set; }
        //Kintamasis, į kurį sudedamos visos galimas kombinacijos
            (n!).

        public ExcelHandler()
        {
            if (FirstData == null)
                FirstData = new List<string>();
```

```
        if (SecondData == null)
            SecondData = new List<string>();

        if (ProcessedFirstData == null)
            ProcessedFirstData = new List<string>();

        if (ProcessedSecondData == null)
            ProcessedSecondData = new List<string>();

        if (AllPossibleCombinations == null)
            AllPossibleCombinations = new List<string>();
    }

    public void ReadData()
    {
        var excelFile = new ExcelQueryFactory(@"C:\Users\ekabi
            \OneDrive\Desktop\Paduodami_duomenys.xlsx");
        var excelFirstData = from c in excelFile.Worksheet<
            ExcelColumn>("Duomenys") select c.
            FirstDataStringValue;
        var excelSecondData = from c in excelFile.Worksheet<
            ExcelColumn>("Duomenys") select c.
            SecondDataStringValue;
        //Iš excel failo "Paduodami_duomenys.xlsx" nuskaitomos
            duomenų eilutės.
        FirstData.Clear();
        SecondData.Clear();
        ProcessedFirstData.Clear();
        ProcessedSecondData.Clear();
        //Kiekvieną kartą vykdant naują iteraciją - išvalome
            senos iteracijos duomenis.
        FirstData = excelFirstData.ToList();
        SecondData = excelSecondData.ToList();
```

```
}

public decimal CalculateDependenceLevel()
{
    var processedFirstDataRepeatingCounts = new Dictionary
       <string, int>();
    var processedSecondDataRepeatingCounts = new
       Dictionary<string, int>();
    int itemsCount;
    foreach (var item in ProcessedFirstData)
    {
        if (processedFirstDataRepeatingCounts.TryGetValue(
           item, out itemsCount))
        {
            processedFirstDataRepeatingCounts[item] = ++
               itemsCount;
        }
        else
            processedFirstDataRepeatingCounts[item] = 1;
    }

    foreach (var item in ProcessedSecondData)
    {
        if (processedSecondDataRepeatingCounts.TryGetValue
           (item, out itemsCount))
        {
            processedSecondDataRepeatingCounts[item] = ++
               itemsCount;
        }
        else
            processedSecondDataRepeatingCounts[item] = 1;
    }
```

```
        var repeatingQuantityFirst = new Dictionary<string,
            decimal>();
        foreach(var item in processedFirstDataRepeatingCounts)
        {
            repeatingQuantityFirst.Add(item.Key, item.Value /
                (decimal)ProcessedFirstData.Count());
        }

        var repeatingQuantitySecond = new Dictionary<string,
            decimal>();
        foreach (var item in
            processedSecondDataRepeatingCounts)
        {
            repeatingQuantitySecond.Add(item.Key, item.Value /
                (decimal)ProcessedSecondData.Count());
        }

        decimal resultCalculation = 0m;
        foreach (var item in repeatingQuantityFirst)
        {
            resultCalculation += repeatingQuantitySecond.
                ContainsKey(item.Key) ? item.Value *
                repeatingQuantitySecond[item.Key] : 0;
        }

        decimal result = (decimal)(MatchingData() -
            resultCalculation);

        return result;
    }
    /* Tai metodas, kuris paskaičiuoja teigiamą ir neigiamą
       priklausomybę tarp duomenų eilučių. Pirmiausia, yra
```

```
    paimama
  pirmi ir antri duomenys ir suskaičiuojama, kiek kartų
     kiekviena unikali reikšmė pasirodo šiuose duomenyse (
     atskirai
  pirmuosiuose ir antruosiuose duomenyse), tada gauti skaič
     iai padalinami iš viso kombinacijų kiekio skaičiaus (t
     .y. gaunamas
  santykinis dažnis kiekvienais iš duomenų kombinacijų). Gr
     ąžinamas rezultatas yra tikimybė, kiek vienos ir kitos
      duomenų eilutės
  kombinacijų atitinkamose vietose sutampa, minus vienuose
     ir kituose duomenyse pasikartojančių kombinacijų
     tikimybių sandaugų suma.
  Taip yra padaryta todėl, nes, jei kažkuri kombinacija
     vienuose iš duomenų pasirodo su tikimybe 0, tai
     nesvarbu, su kokia
  tikimybe ta kombinacija gali pasirodyti kituose duomenyse
     , šių tikimybių sandauga vis tiek bus 0.
  Teigiama ar neigiama priklausomybė yra paskaičiuota -
     priklauso nuo to, kokios reikšmės priskirtos
     ProcessedSecondData - jei šiai
  eilutei priskiriamos atspindžio principu užkoduotos
     kombinacijos, tai gaunama neigiama priklausomybė, kitu
      atveju - teigiama priklausomybė.
  */
public HashSet<string> GetUniqueValues(List<string> data)
{
    var result = new HashSet<string>();
    foreach(var item in data)
    {
        if (!result.Contains(item))
            result.Add(item);
    }
```

```
        return result;
    }
    //Metodas, kuriam padavus užkoduotą duomenų sarašą - jis
       grąžina unikalias to saraso kombinacijas.
    private decimal MatchingData()
    {
        decimal result = 0m;

        for(int i = 0; i < ProcessedFirstData.Count; i++)
        {
            if (ProcessedFirstData.ElementAt(i) ==
               ProcessedSecondData.ElementAt(i))
                ++result;
        }

        var final = result / ProcessedFirstData.Count();

        return final;
    }
    /*Metodas, kuris sulygina paduotas eilutes ir suskaičiuoja
       , kiek kombinacijų, esančių abiejuose eilutėse i-ojoje
       vietoje,
    sutampa. Šis metodas grąžina sutampančių kombinacijų skaič
       ių padalintą iš visų vienos eilutes kombinacijų skaič
       iaus - nesvarbu,
    kurios eilutės - nes abiejų eilučių dydžiai vienodi.*/
    public void ProcessDataTwoFirst(int paternSize, int delay)
    {
        var result = new List<string>();
        var temp = new List<int>();
        for (int i = 0; i < FirstData.Count; i += delay)
        {
            if (FirstData.ElementAtOrDefault(i + paternSize -
```

```
                1) != null)
            {
                for (int j = 0; j < paternSize; j++)
                {
                    var elemValue = decimal.Parse(FirstData.
                        ElementAt(i + j));
                    temp.Add(elemValue < 0 ? 0 : 1);
                }

                var codedValue = "";
                foreach (var item in temp)
                {
                    codedValue += item;
                }

                result.Add(codedValue);
                temp.Clear();
            }
        }
        ProcessedFirstData = result;
    }
    /* Metodas, kuris pirmus duomenis užkoduoja su
       dvejetainiais skaičiais - 0 ir 1.
     Metodo veikimas: iteruojama nuskaityta duomenų eilutė iki
         tol, kol pakanka likusių duomenų numatyto rango dydž
        io sudarymui.
     Kai randama tiek duomenų, kiek reikalinga (pagal rango
        dydį), tuos duomenis užkoduojame pagal tai, ar jie yra
         daugiau arba lygu 0,
     ar mažiau už 0. Pirmu atveju skaičius užkoduojamas 1 (
        kaina kilo), kitu atveju - 0. Užkodavus pirmąjį
        vektorių, nuo pirmojo duomenų
     eilutės nario, kita kombinacija pradedama koduoti nuo
```

```
        sekančio skaičiaus, jei koduojama pataškiui (t.y. jei
        įvestas parametras 1).
     Kitu atveju, kombinacija pradedama koduoti nuo vėlesnio
        skaičiaus (priklausomai nuo įvesto parametro dydžio,
        jei parametras
     2 - peršokamas 1 narys, jei parametras 3 - peršokami 2
        nariai ir t.t.).
     Kodavimas baigiamas tada, kai nebeužtenka duomenų įvesto
        rango dydžio kombinacijai. Metodas grąžina užkoduotų
        kombinacijų sąrašą.
     */
    public void ProcessDataTwoSecond(int paternSize, int delay
        )
    {
        var result = new List<string>();
        var temp = new List<int>();
        for (int i = 0; i < SecondData.Count; i += delay)
        {
            if (SecondData.ElementAtOrDefault(i + paternSize -
                1) != null)
            {
                for (int j = 0; j < paternSize; j++)
                {
                    var elemValue = decimal.Parse(SecondData.
                        ElementAt(i + j));
                    temp.Add(elemValue < 0 ? 0 : 1);
                }

                var codedValue = "";
                foreach (var item in temp)
                {
                    codedValue += item;
                }
```

```
                result.Add(codedValue);
                temp.Clear();
            }
        }
        ProcessedSecondData = result;
    }
    //Šiame metode atliekami analogiški veiksmai, kaip ir
       ProcessDataTwoFirst metode, tik su antrais duomenimis.
    public void WriteProcessedData()
    {
        string file = @"C:\Users\ekabi\OneDrive\Desktop\Už
           koduoti_duomenys.xls";
        Workbook workbook = new Workbook();
        Worksheet worksheet = new Worksheet("Results");
        worksheet.Cells[0, 0] = new ExcelLibrary.SpreadSheet.
           Cell("Pirma duomenų eilutė");
        worksheet.Cells[0, 1] = new ExcelLibrary.SpreadSheet.
           Cell("Antra duomenų eilutė");
        worksheet.Cells[0, 2] = new ExcelLibrary.SpreadSheet.
           Cell("Visos galimos kombinacijos");
        int counter = 1;
        foreach (var item in ProcessedFirstData)
        {
            worksheet.Cells[counter, 0] = new ExcelLibrary.
               SpreadSheet.Cell(item);
            counter++;
        }
        counter = 1;
        foreach (var item in ProcessedSecondData)
        {
            worksheet.Cells[counter, 1] = new ExcelLibrary.
               SpreadSheet.Cell(item);
```

```
            counter++;
        }
        counter = 1;
        foreach (var item in AllPossibleCombinations)
        {
            worksheet.Cells[counter, 2] = new ExcelLibrary.
                SpreadSheet.Cell(item);
            counter++;
        }
        workbook.Worksheets.Add(worksheet);
        workbook.Save(file);
    }
    //Metodas, kuris į excel failą "Užkoduoti duomenys",
        naudodamas prieš tai aprašytus metodus, išveda už
        koduotus duomenis.
    public void WriteCalculatedData(Dictionary<int, decimal>
        columnData)
    {
        string file = @"C:\Users\ekabi\OneDrive\Desktop\
            Rezultatai.xls";
        Workbook workbook = new Workbook();
        Worksheet worksheet = new Worksheet("Results");
        worksheet.Cells[0, 0] = new ExcelLibrary.SpreadSheet.
            Cell("Rango dydis");
        worksheet.Cells[0, 1] = new ExcelLibrary.SpreadSheet.
            Cell("Priklausomybės lygis");
        int counter = 1;
        foreach (var item in columnData)
        {
            worksheet.Cells[counter, 0] = new ExcelLibrary.
                SpreadSheet.Cell(item.Key);
            worksheet.Cells[counter, 1] = new ExcelLibrary.
                SpreadSheet.Cell(item.Value.ToString());
```

```
            counter++;
        }
        workbook.Worksheets.Add(worksheet);
        workbook.Save(file);
    }
    /*Metodas, kuris naudojamas įrašyti į excel failą jau
       apskaičiuotus priklausomybės lygius pagal rango dydį.
    Kitaip sakant, šis metodas į excel išspausdina jam
       paduotus duomenis.*/

    public void ProcessDataTwoFirstReversed(int paternSize,
       int delay)
    {
        var result = new List<string>();
        var temp = new List<int>();
        for (int i = 0; i < SecondData.Count; i += delay)
        {
            if (FirstData.ElementAtOrDefault(i + paternSize -
               1) != null)
            {
                for (int j = 0; j < paternSize; j++)
                {
                    var elemValue = decimal.Parse(FirstData.
                       ElementAt(i + j));
                    temp.Add(elemValue < 0 ? 0 : 1);
                }

                var codedValue = "";
                foreach (var item in temp)
                {
                    codedValue += item;
                }
```

```
                result.Add(Reverse(codedValue));
                temp.Clear();
            }
        }
        ProcessedFirstData = result;
    }
    /*Tai metodas, kuris veikia analogiškai , kaip
       ProcessDataTwoFirst metodas, tik šio metodo pabaigoje
       dar papildomai yra
     panaudojamas Reverse metodas, kuris jau užkoduotas
       kombinacijas perstato naudojant atspindžio principą.*/
    public void ProcessDataTwoSecondReversed(int paternSize,
       int delay)
    {
        var result = new List<string>();
        var temp = new List<int>();
        for (int i = 0; i < SecondData.Count; i += delay)
        {
            if (SecondData.ElementAtOrDefault(i + paternSize -
                1) != null)
            {
                for (int j = 0; j < paternSize; j++)
                {
                    var elemValue = decimal.Parse(SecondData.
                        ElementAt(i + j));
                    temp.Add(elemValue < 0 ? 0 : 1);
                }

                var codedValue = "";
                foreach (var item in temp)
                {
                    codedValue += item;
                }
```

```
                result.Add(Reverse(codedValue));
                temp.Clear();
            }
        }
        ProcessedSecondData = result;
    }
    /*Tai metodas, kuris veikia analogiškai , kaip
       ProcessDataTwoSecond metodas, tik šio metodo pabaigoje
       dar papildomai yra
     panaudojamas Reverse metodas, kuris jau užkoduotas
       kombinacijas perstato naudojant atspindžio principą.*/
    private string Reverse(string s)
    {
        char[] charArray = s.ToCharArray();
        Array.Reverse(charArray);
        return new string(charArray);
    }
    //Tai metodas, kuris paima kombinaciją ir gražiną ją
       perstatytą atspindžio principu.
    public void ProcessDataOneFirst(int paternSize, int delay)
    {
        var result = new List<string>();
        var temp = new Dictionary<int, decimal>();
        for (int i = 0; i < FirstData.Count; i += delay)
        {
            if (FirstData.ElementAtOrDefault(i + paternSize -
               1) != null)
            {
                for (int j = 0; j < paternSize; j++)
                {
                    temp.Add(j, decimal.Parse(FirstData.
                       ElementAt(i + j)));
```

```
                }

                temp = temp.OrderBy(x => x.Value).ToDictionary
                    (x => x.Key, x => x.Value);
                var codedValue = "";
                foreach (var item in temp)
                {
                    codedValue += item.Key;
                }

                result.Add(codedValue);
                temp = new Dictionary<int, decimal>();
            }
        }
        ProcessedFirstData = result;
    }
/* Metodas, kuris pirmus duomenis užkoduoja su tiek simboli
   ų, koks yra rango dydis.
Metodo veikimas: iteruojama nuskaityta duomenų eilutė iki
   tol, kol pakanka likusių duomenų numatyto rango dydžio
   sudarymui.
Kai randama tiek duomenų, kiek reikalinga (pagal rango dydį
   ), tuos duomenis užkoduojame – kiekvienai iš reikšmių
   priskiriamas
simbolis nuo 0 iki įvesto rango dyžio minus vienas. Tada, į
    rango dydžio vektorių sudėtas reikšmes išrikiuojama did
   ėjimo
tvarka – atitinkamai pagal tai išsirikiuoja reikšmėms
   priskirti simboliai.
Užkodavus pirmąjį vektorių, nuo pirmojo duomenų eilutės
   nario, kita kombinacija pradedama koduoti nuo sekančio
   skaičiaus,
jei koduojama pataškiui (t.y. jei įvestas parametras 1).
```

```
        Kitu atveju, kombinacija pradedama koduoti nuo vėlesnio
        skaičiaus
    (priklausomai nuo įvesto parametro dydžio, jei parametras 2
        - peršokamas 1 narys, jei parametras 3 - peršokami 2
        nariai ir t.t.).
    Kodavimas baigiamas tada, kai nebeužtenka duomenų įvesto
        rango dydžio kombinacijai. Metodas grąžina užkoduotų
        kombinacijų sąrašą.*/
    public void ProcessDataOneSecond(int paternSize, int delay
        )
    {
        var result = new List<string>();
        var temp = new Dictionary<int, decimal>();
        for (int i = 0; i < SecondData.Count; i += delay)
        {
            if (SecondData.ElementAtOrDefault(i + paternSize -
                1) != null)
            {
                for (int j = 0; j < paternSize; j++)
                {
                    temp.Add(j, decimal.Parse(SecondData.
                        ElementAt(i + j)));
                }

                temp = temp.OrderBy(x => x.Value).ToDictionary
                    (x => x.Key, x => x.Value);
                var codedValue = "";
                foreach (var item in temp)
                {
                    codedValue += item.Key;
                }

                result.Add(codedValue);
```

```
                temp = new Dictionary<int, decimal>();
            }
        }
        ProcessedSecondData = result;
    }
    //Šiame metode atliekami analogiški veiksmai, kaip ir
       ProcessDataOneFirst metode, tik su antrais duomenimis.
    public void ProcessDataOneSecondReversed(int paternSize,
       int delay)
    {
        var result = new List<string>();
        var temp = new Dictionary<int, decimal>();
        for (int i = 0; i < SecondData.Count; i += delay)
        {
            if (SecondData.ElementAtOrDefault(i + paternSize -
                1) != null)
            {
                for (int j = 0; j < paternSize; j++)
                {
                    temp.Add(j, decimal.Parse(SecondData.
                        ElementAt(i + j)));
                }

                temp = temp.OrderBy(x => x.Value).ToDictionary
                    (x => x.Key, x => x.Value);
                var codedValue = "";
                foreach (var item in temp)
                {
                    codedValue += item.Key;
                }

                result.Add(Reverse(codedValue));
                temp = new Dictionary<int, decimal>();
```

```
                }
            }
            ProcessedSecondData = result;
        }
    }
}
/* Tai metodas, kuris veikia tokiu pačiu principu kaip
   ProcessDataOneSecond, tik šis metodas grąžinant duomenų sarašą
  panaudoja jau prieš tai aprašytą Reverse metodą - tai yra, grąž
     ina tokias užkoduotas kombinacijas kaip ProcessDataOneSecond
  metode, tik perstatytas atspindžio principu.*/


namespace ConsoleApp
{
    public class Combinations
    {
        private int elementLevel = -1;
        private int numberOfElements;
        private int[] permutationValue = new int[0];
        public List<string> CombinationValues = new List<string>()
            ;

        public char[] InputSet { get; set; }
        public char[] MakeCharArray(string InputString)
        {
            char[] charString = InputString.ToCharArray();
            Array.Resize(ref permutationValue, charString.Length);
            numberOfElements = charString.Length;
            return charString;
        }

        public void CalcPermutation(int k)
```

```
        {
            elementLevel++;
            permutationValue.SetValue(elementLevel, k);

            if (elementLevel == numberOfElements)
                SavePermutation(permutationValue);
            else
            {
                for (int i = 0; i < numberOfElements; i++)
                {
                    if (permutationValue[i] == 0)
                        CalcPermutation(i);
                }
            }
            elementLevel--;
            permutationValue.SetValue(0, k);
        }

        private void SavePermutation(int[] value)
        {
            string combinationValue = "";
            foreach (int i in value)
            {
                combinationValue += InputSet.GetValue(i - 1);
            }
            CombinationValues.Add(combinationValue);
        }
    }
}
/* Tai klasė, kurioje 3 metodų pagalba yra paskaičiuojamos visos (
   n!) galimos įvesto rango dydžio kombinacijos.
 Veikimo principas: pagal Alexander Bogomolyn rekursijos algortitm
   ą yra išrenkamos ir grąžinamos visos galimos nurodyto
```

```
ilgio kombinacijos.
*/
```