**Kauno technologijos universitetas**
Matematikos ir gamtos mokslų fakultetas

# Nekilnojamojo turto vertės nustatymas pasitelkiant mašininio mokymosi technikas

Baigiamasis magistro studijų projektas

**Simonas Adomavičius**
Projekto autorius

Prof. Dr. Evaldas Vaičiukynas
Vadovas

Prof. Dr. Robertas Alzbutas
Vadovas

**Kaunas, 2022**

**Kauno technologijos universitetas**
Matematikos ir gamtos mokslų fakultetas

# Nekilnojamojo turto vertės nustatymas pasitelkiant mašininio mokymosi technikas

Baigiamasis magistro studijų projektas
Didžiųjų verslo duomenų analitika (6213AX001)

**Simonas Adomavičius**
Projekto autorius

**Prof. Dr. Evaldas Vaičiukynas**
Vadovas

**Prof. Dr. Robertas Alzbutas**
Vadovas

**Doc. Dr. Valentas Gružauskas**
Recenzentas

**Dr. Paulius Danėnas**
Recenzentas

**Kaunas, 2022**

**Kauno technologijos universitetas**

Matematikos ir gamtos mokslų fakultetas

Simonas Adomavičius

# Nekilnojamojo turto vertės nustatymas pasitelkiant mašininio mokymosi technikas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Simonas Adomavičius

*Patvirtinta elektroniniu būdu*

Simonas Adomavičius. Nekilnojamojo turto vertės nustatymas pasitelkiant mašininio mokymosi technikas Magistro studijų baigiamasis projektas vadovai Prof. Dr. Evaldas Vaičiukynas ir Prof. Dr. Robertas Alzbutas; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Taikomoji matematika.

Reikšminiai žodžiai: mašininis mokymasis, regresija, būsto vertės nustatymas, teksto analizė, vaizdų analizė, klasterizavimas, aruodas.lt.

Kaunas, 2022. 92 p.

**Santrauka**

Šiame darbe yra nagrinėjami dirbtinio intelekto metodai, siekiant atlikti tikslesnį Vilniaus mieste ir rajone parduodamų butų vertinimą. Darbe yra naudojama viešai prieinama informacija apie parduodamus butus iš Aruodas.lt kuri yra surenkama automatizuotu būdu. Informacija kuri yra renkama susideda iš tekstinės – parduodamo buto skelbimo aprašymas, nuotraukų – skelbime patalpintos nuotraukos, bei bendrinė informacija pateikiama skelbime – kaina, vietovė, buto plotas, buto ypatumai ir kita.

NT vertės nustatymo uždaviniuose nuolat pasitaikanti problema yra mažai vertės turinčių objektų pervertinimas ir / ar didelę vertę turinčių objektų nepakankamas vertinimas. Sprendžiant regresijos uždavinius, mes dažnai turime duomenų apie daugumą objektų, tačiau visuomet per mažai itin pigių, bei itin prabangių. Dėl šios priežasties vertinti daugumos objektų vertę yra lengviau, nei pigių ar brangių. Vis dėl to, tobulėjančių dirbtinio intelekto metodų kontekste bei informacijai tampant vis lengviau pasiekiamai, mūsų galimybės geriau įvertinti šio tipo būstus tampa vis didesnės. Darbe tikimasi, jog informacija esanti nuotraukose ir tekste leis atlikti geresnę būsto vertės prognozė geriau vertinant tiek pigius, tiek brangius butus.

Pirmojoje darbo dalyje atliekama literatūros apžvalga, kitų autorių darbų nagrinėjusių dirbtinio intelekto panaudojimo galimybes būsto vertės prognozavimui. Antroje dalyje aprašomi tyrimo metodai, kurie bus taikomi darbe ir pristatoma informacijos rinkimo strategija. Trečiojoje dalyje yra atliekamas tyrimas, kurio metu iš pradžių yra atliekama požymių inžinerija, o vėliau modelių apmokymas bei optimizavimas. Galiausiai yra pristatomi geriausio modelio su 13.74 MAPE, ir 33,307 RMSE rezultatai, bei pateikiamos išvados.

Simonas Adomavičius. Real estate valuation using machine learning techniques. Master's Final Degree Project supervisors Prof. Dr. Evaldas Vaičiukynas and Prof. Dr. Robertas Alzbutas; Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Study field and area (study field group): Applied Mathematics.

Keywords: machine learning, regression, housing price prediction, text analysis, vision analysis, clustering, aruodas.lt.

Kaunas, 2022. 92 p.

**Summary**

In this work, the methods of artificial intelligence are analyzed in order to perform a more accurate value prediction of the apartments sold in Vilnius city and district. The work uses publicly available information about apartments put for sale on Aruodas.lt, which is collected in an automated way. The information that is collected consists of a text – description of the apartment for sale, photos – photos placed in the ad, and general information provided in the ad – price, location, apartment size, various features of the apartment and more.

A constant problem in real estate valuation is the overvaluation of low-value objects and / or underestimation of high-value objects. When dealing with regression problems, we often have data on most average objects, but never enough of cheap and luxurious ones. For this reason, estimating the value of most properties is easier than cheap or expensive. However, as the methods of artificial intelligence evolve and information becomes more and more available, our ability to better value this type of housing increases. It is hoped that the information contained in the photos and text will allow for a better forecast of the value of housing by better valuing both cheap and expensive apartments.

In the first part of the work, a review of the literature is performed where other authors have examined the possibilities of using artificial intelligence to predict the value of housing. The second part describes the research methods that will be applied in the work and presents the information gathering strategy. In the third part, a study is conducted, during which feature engineering is performed first, followed by model training and optimization. Finally, the results of the best model with 13.74 MAPE and 33,307 RMSE are presented along with the conclusions of the work.

**Turinys**

## Lentelių sąrašas

**Paveikslų sąrašas**

## Santrumpų ir terminų sąrašas

**Santrumpos:**

**ML** – mašininis mokymasis (angl. *Machine Learning*).

**MTL** – daugiaužduotinis mokymasis (angl. *Multi-Task Learning)*.

**NN** – neuroninis tinklas (angl. *Neural Network*).

**NT** – nekilnojamasis turtas.

**Terminai:**

**Regresantas** – Priklausomas arba aiškinamasis kintamasis yra regresijos lygties kairėje pusėje esantis kintamasis (Y), kurio vidutinių reikšmių pokyčius stengiamasi paaiškinti kitų – dešinėje esančių – veiksnių pokyčiais. Regresinėje analizėje, bei šiame darbe bus toliau vartojami šie sinonimai: priklausomas – aiškinamasis – stimulas.

**Regresorius** – Nepriklausomi arba aiškinantieji kintamieji ($X_1$, $X_2$ ir taip iki $X_k$ ) – tai dešinėje lygties pusėje esantys kintamieji, kurie veikia priklausomąjį kintamąjį (Y). Regresinėje analizėje, bei šiame darbe bus toliau vartojami šie sinonimai: nepriklausomas – aiškinantysis – atsakas.

# Įvadas

Vertės nustatymas bei kainos prognozavimas yra nuo seno sprendžiami uždaviniai. Vieni uždaviniai keliami siekiant numatyti, kaip keisis valiutų, akcijų ar prekių kainos per tam tikrą laiką, kiti – keliami tam, kad pasitelkus atitinkamus požymius apie objektą būtų apskaičiuota galima jo vertė. Nors pirmieji yra laiko eilutės uždaviniai, o antrieji – regresiniai, kurie anksčiau buvo sprendžiami atskirai, dabar, kai darosi vis lengviau surinkti daugiau informacijos bei besivystant naujiems ir pažangesniems metodams – gali būti atlikti kartu siekiant pasiekti geresnius rezultatus. Negana to, atsiranda vis daugiau galimybių įtraukti vis daugiau informacijos į modelių sudarymą bei atsirinkti požymius, geriausiai išsprendžiančius šiuos uždavinius.

Gerai sudaryti modeliai gali būti naudojami prognozuoti įvairią vertę ar kainą, pavyzdžiui, automobilių, nekilnojamojo turto, žaliavų, valiutų ir kt. Šie modeliai ypač naudingi bankams, kadangi už kiekvienos paskolos egzistuoja įkeistas turtas, o finansinės įstaigos privalo prisiimti riziką klientui bankrutavus ir tapus nemokiam. Bankai, kurie patys vykdo savo kredito rizikos valdymą A-IRB (angl. *Advanced Internal Rating-Based),* privalo stebėti pagal Bazelio (angl. *Basel*) reikalavimus įkeisto turto vertę, jog atitiktų visus rizikos standartus. Taip pat dideli bankai turi klientų ne tik vienoje šalyje, o keliose. Todėl gerai įvairius NT objektus galintis vertinti modelis tampa ypač paklausus. Vienintelė problema, jog modelio paaiškinamumo stoka (dažnai neatsiejama dalis sudėtingesnių modelių) taip pat yra rizika bankui, todėl modelio paaiškinamumas ir galimybė lengvai jį interpretuoti taip pat yra itin svarbūs modelio elementai.

Tikslus NT bei automobilių kainų prognozavimas atneštų naudos ne tik bankams, bet ir kitoms finansinėms įstaigoms, pavyzdžiui, draudimo. Žinant turto vertę galima tikslingai atsidėti priimamai rizikai kapitalą. Nekilnojamojo turto sektorius taip pat turi daug potencialo, nes gerai veikiantis modelis galėtų pakeisti būsto vertintojus ir tapti skaidri ir nešališka alternatyva vertinti NT vertę.

Šiame darbe bus koncentruojamasi tik į NT vertės prognozavimą, nes manoma, jog Lietuvoje gerai veikiantis modelis galėtų būti plačiai taikomas. Taip pat darbo autoriui žinoma, jog automobilių kainos vertinimui modelio kurti neverta – praktikoje šio tipo modeliai yra dažnai sudaromi taisyklių principu.

Būsto vertės kainos prognozavimas buvo nagrinėtas labai plačiai, tačiau vis dar yra taip pat išsamiai nagrinėjamas dėl nuolat atsirandančių efektyvesnių modelių. Dabar į modelio sudarymą galime įtraukti ne tik būsto skelbime išvardintus pokyčius, tačiau (įtraukti) ir jo vietovę, susietą su mieste esančiais objektais, įvertinti būsto kokybę iš nuotraukų, įvertinti pridėtinę vertę, aprašytą skelbimo aprašyme, ir t.t. Darbe toliau bus nagrinėjami modeliai ir technologijos, siekiantys kuo tiksliau įvertinti būsto kainą. Vėliau, naudojant Aruodas.lt duomenis, bus surinkti parduodami būstai Vilniuje ir atliekama jų kainos prognozė, pasitelkus geriausius modelius.

**Darbo problema:** įprastai NT rinkoje naudojami modeliai puikiai prognozuoja būstus, apie kuriuos yra daug informacijos ir kurie yra vidutinės kainos rėžiuose. Tačiau modeliai dažnai pervertina ar nepakankamai įvertina NT, kuris yra užmiesčiuose prastuose rajonuose, tačiau aukščiausios įrengimo kokybės ar prabangiame rajone ir puikioje vietoje, tačiau avarinės būsenos. Vartotojų kuriamame turinyje ( nuotraukose, skelbimo aprašymuose) ši informacija yra prieinama ir tinkamai ją panaudojus galime atsižvelgti ne tik į skaitinę informaciją sudarydami kainos prognozę, tačiau ir į tekstinę bei vaizdinę. Tai leistų tiksliau prognozuoti brangesnius ar pigesnius nei vidutiniai būstus.

**Darbo tikslas**: tikslesnis NT turto vertės prognozavimas, pasitelkiant vartotojų kuriamą bei viešai prieinamą informaciją ir dirbtinį intelektą.

**Darbo objektas**: viešai prieinami Aruodas.lt Vilniaus mieste parduodamų butų skelbimai.

**Darbo uždaviniai**:
1. atlikti mokslinių darbų, nagrinėjančių pasirinktą temą, paiešką ir apžvalgą;
2. surinkti skelbimų informaciją iš Aruodas.lt ir paruošti duomenis;
3. įtraukti atviro teksto informaciją į galutinį regresijos modelį;
4. įtraukti vaizdo ir lokacijos informaciją į galutinį regresijos modelį;
5. optimizuoti požymių inžinerijos procesą siekiant padidinti jų prognostinę vertę;
6. parinkti ir suderinti modelio architektūrą, galinčią tiksliausiai vertinti būsto kainą;
7. apibendrinti rezultatus ir pateikti išvadas.

# 1. Literatūros apžvalga

## 1.1. Būsto kaina kaip regresinis uždavinys

Nekilnojamojo turto kainos yra veikiamos daugybės veiksnių bei juos siejančių ryšių. Ne apie visus šiuos ryšius vertindami būsto kainą, galime žinoti. Todėl A. Baldominos'as [1] savo darbe iškelia problemą, jog tradiciniai nekilnojamojo turto vertinimo modeliai remiasi hedonistine regresija[1], kuri turtą išskaido į pagrindines sudedamąsias savybes (atrinktą požymių matricą) tam, kad galėtų nustatyti kiekvienos iš jų santykį tarp viena kitos ir kainos. Dėl šios priežasties nėra įvertinama tikroji būsto kaina, kurią veikia, kaip pasakytų Adamas Smitas, „nematoma ranka“[2]. Baldominos'o visas darbas grindžiamas idėja, jog būsto kaina gali būti didesnė ar mažesnė nei rinkos, todėl darbe siekiama atrasti modelį, galintį atrinkti (atrasti) būstą, parduodamą už mažesnę nei rinkos kainą.

Duomenyse yra naudojami kokybiniai, binariniai ir tęstiniai kintamieji. Taip pat pastebėtina, jog Naive Bayes ar Baysesian tinklų funkcionavimas nėra tinkamas, kad būtų sėkmingai sprendžiami regresiniai uždaviniai. [2], tuo tarpu regresijos medžių taip pat buvo atsisakyta, nes ansambliai taip pat veikia geriau sprendžiant regresinius uždavinius. Darbe buvo naudoti 4 kategorijų modeliai: branduolio modeliai, geometriniai modeliai, taisyklėmis pagrįsti modeliai (regresijos medžiai) ir neuronų tinklų (universaliųjų funkcijų aproksimatorių) modeliai.

Branduolio modeliui tirti buvo pasirinktas atraminių vektorių metodas (SVM) [3]. Įprastai SVM transformuoja duomenis į aukštesnio matmens būseną, kur duomenis galima atskirti pagal tam tikrą funkciją (šiuo atveju branduolio funkciją, bet nebūtinai), tada išmokstama tokios funkcijos atskirti egzempliorius. Toks taikymas įprastai yra tinkamas klasifikacijos uždaviniams, tačiau pasirinkus naują „*epsilon*“ (e) parametrą galima šį metodą taikyti ir regresijos uždaviniams, kur (e) atlieka kontrolės funkciją, kuri užtikrina, kad stebėjimai nesiektų konkrečios funkcijos reikšmės.

Geometriniam modeliui buvo naudotas k-artimiausių kaimynų modelis. Šis modelis kiekvienam stebėjimui apskaičiuoja atstumą nuo kitų stebėjimų naudodamas tam tikrą atstumo skaičiavimo metriką (pvz., Euklido, Čebyševo, Kosinus panašumo, Manhatano ir t.t.). Toliau kiekvienas stebėjimas yra priskiriamas prie artimiausių esančių kaimynų ir išvedama tam tikra metrika (kuri darbo atveju yra turto vidutinė kaina). Šis metodas užima daug laiko, kai duomenų imtyje yra daug požymių ir turi nemažai metodų, kurių taikymas priklausys nuo stebėjimų išsidėstymo daugiamatėje erdvėje. Pastebėtina, jog darbo autorius laikosi nuomonės, kad šis modelis yra sunkiau pritaikomas esant požymiams su dvejetainėmis reikšmėmis. Vis dėlto šia problemą galima išspręsti naudojant apibendrintus skaičiavimus [4], kur konkrečiai tokiems požymiams yra skaičiuojama paprasta dažnių lentelė, kuri tinkama ir kategoriniams kintamiesiems, o gautas rezultatas yra pridedamas prie tolydžių ir / ar ranginių kintamųjų apskaičiuotų artumo matų ir suvidurkinamas. Pažymėtina, kad šiems apibendrintiems skaičiavimams galima suteikti ir svorius.

Taisyklėmis pagrįstiems metodams buvo naudotas regresijos medžių modelis. Minėti metodai gali išmokti tam tikrų taisyklių rinkinį, kad apskaičiuotų duomenų rinkinio ryšį su tikslo atributu. Šiam modeliui perprasti galime naudoti atitinkamą kriterijų (entropijos, statistinės dispersijos, kt.), siekiant skirstyti duomenų rinkinį į požymius, paaiškinančius daugiausia informacijos naudojant

[1] https://en.wikipedia.org/wiki/Hedonic_regression

[2] https://en.wikipedia.org/wiki/Invisible_hand

atitinkamą kriterijų (nuo daugiausiai paaiškinančio požymio iki mažiausiai suteikiančio informacijos), kol yra suformuojamas „medis“. Nagrinėjamame darbe yra naudojamas medžių ansamblio – atsitiktinio miško – alternatyvą pristatyta [5]: itin atsitiktiniai miškai (angl. *extremely randomized trees)*.

Galiausiai naudotas neuroninis tinklas, kuris yra visiškai sujungtas, taip pat naudotas gradientinis nusileidimas su atgaliniu mokymu (angl. *backpropagation*) svoriams atrinkti. Čia svarbu paminėti, jog naudotas neurorininis tinklas yra sukurtas ant Sklearn[3] bibliotekos, kuri pasirūpina daug su neuroniniais tinklais susijusių veiksmų, tačiau tai atima ir dalį funkcionalumo prieinamo naudojat Tensorflow, Keras ar Pytorch paketus. Toliau yra pateikiami modeliai, naudoti parametrai ir jų reikšmės, duomenys, kurie buvo suskaidyti į treniravimosi ir mokymosi imtis naudojant 5-fold kryžminį patikrinimą (angl. *cross-validation*). Pritaikius hiperparametrų paiešką, buvo atrinkti šie modelių parametrai:

**1 lentelė.** Autoriaus A. Baldominos optimalūs modelių parametrai

| ML algorithm | Parameter | Values |
|---|---|---|
| Support vector regression<br>`svm.SVR` | Kernel type (`kernel`)<br>Penalty (`C`)<br>Kernel coefficient (`gamma`) | Radial basis function kernel (`rbf`)<br>1.0<br>Inverse of the number of features |
| *k*-nearest neighbors<br>`neighbors.KNeighborsRegressor` | Number of neighbors (`n_neighbors`)<br>Distance metric (`metric`)<br>Weight function (`weights`) | 5, 10, 20, 50<br>Minkowski, cosine<br>Uniform, inverse to distance (`distance`) |
| Ensembles of regression trees<br>`ensemble.ExtraTreesRegressor` | Number of trees in the forest (`n_estimators`)<br>Criterion for split quality (`criterion`)<br>Whether bootstrap samples are used (`bootstrap`) | 10, 20, 50<br>Mean absolute error (`mae`), mean squared error (`mse`)<br>True, false |
| Multi-layer perceptron<br>`neural_network.MLPRegressor` | Network architecture (`hidden_layer_sizes`)<br>Activation function (`activation`)<br>Learning rate (`learning_rate_init`)<br>Optimizer (`solver`)<br>Batch size (`batch_size`) | 1024, 256–128, 128–64–32<br>Rectified linear unit (`relu`)<br>0.001<br>Adam<br>200 |

Rezultatai buvo gauti iš kiekvieno modelio parametrų varianto atrinkus geriausią. Paminėtini šie parametrai :

- K-artimiausių kaimynų: 50 kaimynų, naudojant Minkowskio atstumą;
- 2 sluoksnių neuroniniai tinklai: 2 sluoksniai (256,128);
- itin atsitiktiniai miškai: 50 medžių naudojant MAE kaip atrankos ir karpymo kriterijų;
- SVM: naudojant radiantinę branduolio funkciją (RBF).

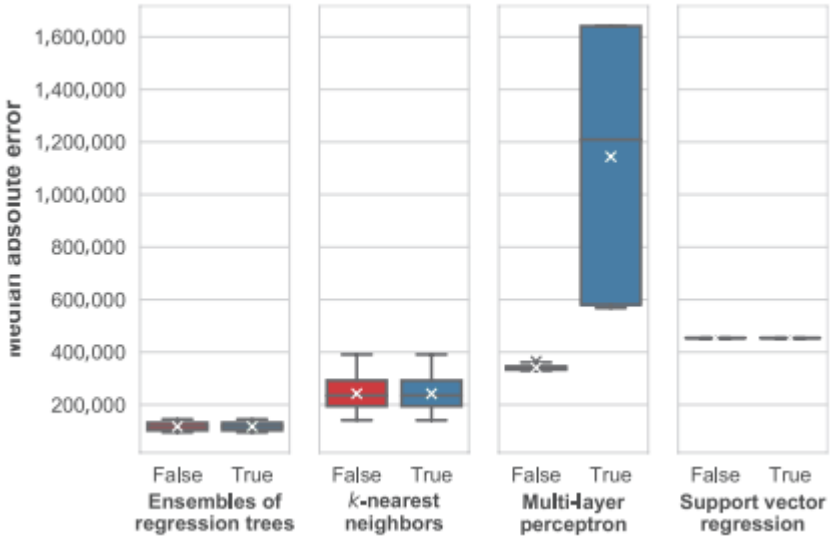
**2 lentelė.** Autoriaus A. Baldominos modelių rezultatai

| ML Algorithm | $E_{var}$ | *MAE* | *MedAE* | *MSE* | $R^2$ |
|---|---|---|---|---|---|
| *k*-nearest neighbors | 0.3625 (–) | 0.4404 (–) | 0.2068 (–) | 4.044 (–) | 0.3598 (–) |
| Multi-layer perceptron | 0.3113 (0.0020) | 0.5637 (0.0026) | 0.3355 (0.0041) | 4.2262 (0.0029) | 0.3067 (0.0027) |
| Ensembles of regression trees | 0.1303 (0.1381) | 0.3714 (0.0075) | 0.1319 (0.0038) | 4.3468 (0.1548) | 0.1253 (0.1386) |
| Support vector regression | 1.73E-5 (–) | 0.7384 (–) | 0.4540 (–) | 4.9015 (–) | −0.0664 (–) |

Papildomai yra apžvelgta ir standartizavimo įtaka visiems modeliams. Paaiškėjo, jog neuroniniams tinklams itin prastai sekasi apdoroti standartizuotus duomenimis, dėl to modelių liekanų vidurkio

[3] https://scikit-learn.org/

mediana tapo itin didelė. Likusiems modeliams ši transformacija neturi įtakos dėl tiesinės transformacijos[4] ypatybių.




**1 pav.** Autoriaus A. Baldominos MAE rezultatai naudojant standartizavimą skirtingiems modeliams

[6] K. Truong'as savo darbe siekia nustatyti būsto kainos indeksą remdamasis, pasak autoriaus, sudėtingesniais, todėl rečiau naudojamais metodais. Darbe buvo naudoti šie modeliai:

- atsitiktiniai miškai;
- ekstremalus gradientinis pastiprinimas (XGBoost);
- švelni gradientinio didinimo mašina (angl. *Light Gradient Boosting Machine (LightGBM)*);
- hibridinė regresija (angl. *Hybrid regression*);
- sudedamųjų ansamblis (angl. *Stacked Generalisation / Ensemble*).

Optimizavus hiperparametrus ir atrinkus geriausius parametrus kiekvienam iš modelių, pasirodė, jog efektyviausias sudedamųjų ansamblio modelis. Šį modelį sudarė visi prieš tai bandyti modeliai, tačiau autorius nepamini kokio tipo modelį naudojo kaip meta modelį. Kita vertus, dažniausiai šis modeliais yra paprasta tiesinė regresija. Pasak autoriaus, tam buvo 2 priežastys: 1) modelis remiasi K-fold kryžmine patikra, kuri yra tinkama atrasti geriausią balansą tarp dispersijos ir šališkumo; 2) pats modelių sudėjimas lėmė tai, jog skirtingi baziniai modeliai papildo vienas kitą ir tai leidžia meta modeliui geriau prognozuoti galutinį sprendimą.

**3 lentelė.** Autoriaus Q. Truong modelių rezultatai

[4] https://en.wikipedia.org/wiki/Linear_map

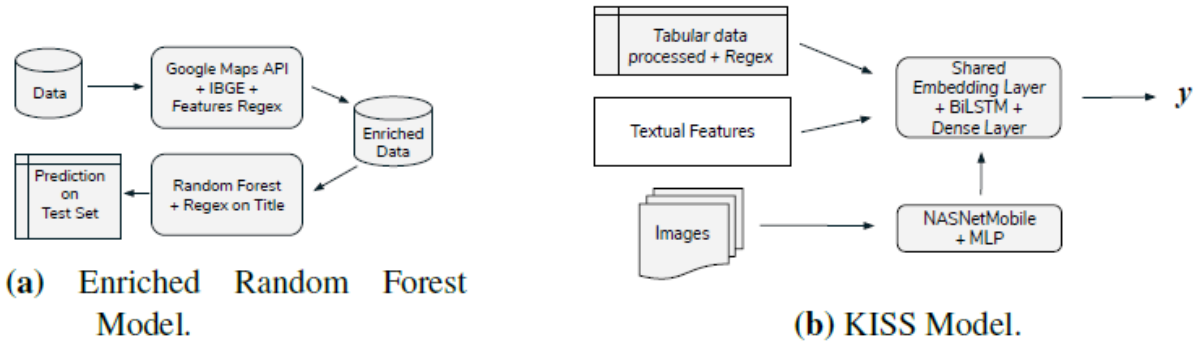
| Model | RMSLE | |
|---|---|---|
| | *Train Set* | *Test Set* |
| Random Forest | **0.12980** | 0.16568 |
| Extreme Gradient Boosting | 0.16118 | 0.16603 |
| Light Gradient Boosting Machine | 0.16687 | 0.16944 |
| Hybrid Regression | **0.14969** | 0.16372 |
| Stacked Generalization Regression | 0.16404 | **0.16350** |

Autorius taip pat akcentuoja, jog tolimesniuose darbuose būtų pravartu apžvelgti šias temas:

- kelių regresijos modelių susiejimo galimybes (angl. *coupling effect*);
- nuolatinio mokymosi galimybes (angl. *continious learning*);
- kombinacijas tarp mašininio mokymosi ir giliojo mokymosi;
- greitesnių sudėtingų modelių apmokymo metodus.

[7] su kolegomis savo darbe panaudojo ne tik nuotraukas, tačiau ir skelbimų aprašymus, kad sukurtų 2 modelių architektūras. Viena architektūra remiasi atsitiktinių miškų regresija, o antroji – LSTM, kuri įprastai būna tiek laiko eilučių, tiek teksto apdorojimo uždaviniuose. Tačiau autoriai į modelio architektūrą įtraukė daugiau dedamųjų, siekdami pagerinti modelio tikslumą praturtino turimus duomenis.

Pirmoji architektūra nuotraukoje (a) (*Enriched Random Forest Model*), kurioje panaudoję Google API lokaciją susiejo su informacija, būdinga tai teritorijai (vidutinis atlyginimas ir BVP dalis, tenkanti vienam gyventojui). Taip pat buvo naudotas REGEX metodas papildomoms būsto ypatybėms nustatyti (ištraukti papildomus atributus apie būstą). Antras modelis akronimu KISS (angl. *Keep it simple, stupid*) nuotraukoje (b) (*KISS model*) paremtas giliuoju mokymusi, leidžiančiu naudotis papildytais lentelės duomenimis, atributais, išrinktais iš teksto, bei nuotraukomis. Tekstui apdoroti buvo naudoti 2 BiLSTM tinklai, o nuotraukoms – NASNet Mobile tinklas [8] kartu su Multilayer Perceptron (MLP). Galime matyti, jog iš architektūros pusės turime gan sudėtingą modelį, kuris jau galėtų būti suprantamas kaip būsto kainos kaip giliojo mokymosi uždaviniui, apie kurį bus kalbama toliau. Vis dėl to, svarbu paminėti, jog būtent šio darbo kontekste išryškėja vienas aktualus dalykas būdingas regresijos uždaviniams – dauguma jų apjungia įvairius sudėtingus (arba nebūtinai) metodus į ansamblį, kuris sugeba pasiekti geresnius rezultatus, nei vienas individualus modelis.




**2 pav.** Autoriaus B. K. Alfonso Modelių architektūra

Grįžtant prie rezultatų, apmokyti modeliai pasiekė prastus RMSLE rezultatus, tačiau autoriai pasiūlė, jog rezultatą galėtų pagerinti iš dviejų modelių sukurdami ansamblį. (Taip pat, kaip paaiškėjo, jog šis sprendimas) Pastarasis pelnė pirmąją vietą EEE 2019[5] Kaggle konkurse.

**4 lentelė.** Autoriaus B. K. Alfonso modelių RMSLE rezultatai

| | *Enriched RF* | *KISS Model* | *Ensemble Model* |
|---|---|---|---|
| RMSLE | 0.27967 | 0.26135 | **0.23847** |

[9] savo darbe naudojo mašininį mokymąsi panaudodamas 3 skirtingus modelius: SVM, Atsitiktinius miškus ir gradientinio pastiprinimo mašinos (angl. *gradient boosting machine* (GBM)). Pastarasis metodas remiasi paprastu pastiprinimo (angl. *boosting*) principu, tačiau šiame modelyje silpniems modeliams (medžiams) yra suteikiami svoriai naudojant tikslo funkciją, kuria siekiama minimizuoti paklaidas naudojant gradientinį nusileidimą.

[10] savo darbe pasiūlė hibridinį Lasso ir gradientinį regresinius modelius, kurie Kaggle iššūkyje „Būsto kainos: "House Prices: pažangūs regresijos metodai" (angl. *Advanced Regression Techniques)*[6], buvo tarp 1% geriausių modelių. Darbe autoriai papildomai apdoroja turimus duomenis, siekdami gauti daugiau požymių apie skelbimus ir sudarydami iš viso 260 požymių, kurie naudojant Lasso reguliarizaciją buvo atrinkti iš 490 požymių, sąrašą. Darbe sukuriami Lasso, Ridge ir Gradient regresijos modeliai, iš kurių yra sudaromas hibridas tarp Lasso ir Gradient naudojant susietumą (angl. *coupling effect*) atitinkamai 65% ir 35%.

[11] savo darbe naudoja 4 mašininio mokymosi modelius: mažiausių kvadratų SVM modelį (LSSVM), klasifikacijos ir regresijos medžius (CART), apibendrintos regresijos neuroninius tinklus (GRNN) ir atgalinės atrankos neuroninį tinklą (BPNN). Hiperparametrų atrankai buvo pasirinktas genetinis algoritmas (GA), kurį naudodamas geriausias LSSVM modelis pagerino savo MAPE rezultatus iš 1.676 į 0.228 po atrankos.

[12] taip pat daug dėmesio skyrė mašininio mokinimosi modeliams: C4.5, RIPPER (angl. *Repeated Incremental Pruning to Produce Error Reduction*), Naive Bayesian ir AdaBoost. Geriausias

[5] https://www.kaggle.com/c/data-science-challenge-at-eef-2019/

[6] https://www.kaggle.com/c/house-prices-advanced-regression-techniques

rezultatas buvo pasiektas naudojant RIPPER modelį. Darbo autoriai taip pat integravo būsto paskolų pokyčius per tam tikrą laiką bei aplink esančių mokyklų reitingus, tačiau darbo išvadose pateikia rekomendacijas įtraukti ir valiutų, naftos bei kitus makroekonominius rodiklius siekiant įtraukti daugiau išorinių duomenų į galutinio modelio sudarymą. Svarbu paminėti, jog darbe autoriai sprendė klasifikacijos uždavinį, siekdami įvertinti, ar būstas buvo parduotas už mažesnę nei rinkos kainą, ar ne, naudojant mokymosi, testavimo ir tikrinimo imtį su 10-Fold kryžminimu.

[13] savo darbe išbando 8 mašininio mokymosi modelius, daugiausia dėmesio skirdamas SVM modeliui, orientuotam į MSE. Darbe yra naudojami pažingsninės atrankos (angl. *Stepwise*), PCA ir pastiprinimo metodai atrenkant svarbiausius su būsto kainomis susijusius požymius. Darbe taip pat buvo palyginti ir modelių greičiai. Greičiausias buvo neuroninių tinklų modelis, o PCA su SVM – lėčiausi.

**5 lentelė.** Autoriaus Phan modelių MSE rezultatai ir apmokymo trukmė

| Model | Train MSE | Eval. MSE | Eval. Ratio |
|---|---|---|---|
| Linear regression | 0.0948 | 0.0994 | **1.00** |
| Polynomial regression | 0.0773 | 0.0832 | **0.84** |
| Regression tree | 0.0925 | 0.0985 | **0.99** |
| Neural Network | 0.2657 | 0.2749 | **2.77** |
| Stepwise & SVM | 0.0558 | 0.0615 | **0.62** |
| Stepwise & tuned SVM | 0.0480 | 0.0561 | **0.56** |
| PCA & SVM | 0.0721 | 0.0810 | **0.82** |
| PCA & tuned SVM | 0.0474 | 0.0728 | **0.74** |

| Model | Time (min.) |
|---|---|
| Regression tree | 0.033 |
| Neural Network | 0.033 |
| Stepwise & SVM | 1.583 |
| Stepwise & tuned SVM | 1.400 |
| PCA & SVM | 2.317 |
| PCA & tuned SVM | 2.733 |

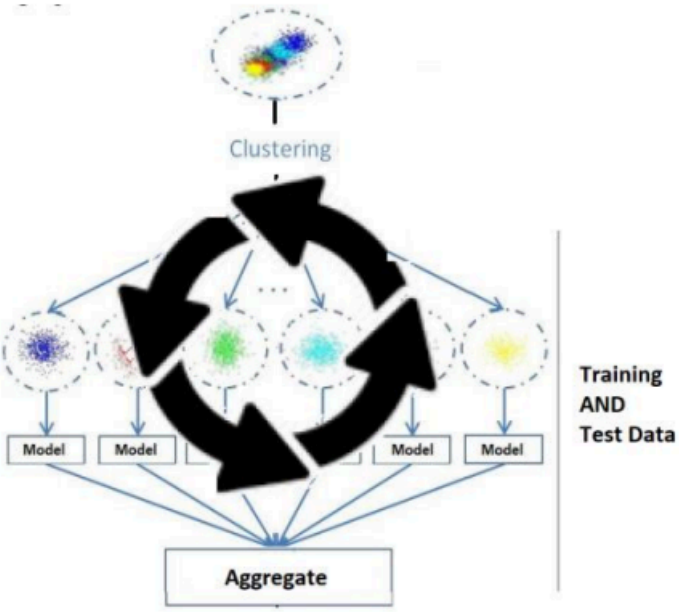
## 1.2. Būsto kaina kaip artimiausių kaimynų uždavinys

Be regresijos, kita dažnai pasitaikanti metodika praktikoje yra naudoti artimiausių kaimynų metodus. Ši strategija puiki ir veiksminga net už mašininio mokymosi ar statistinių modelių ribų. Nepriklausomiems turto vertintojams atliekant turto vertinimą yra keletas metodikų, kuriomis jie gali remtis, kad paskaičiuotų preliminarią būsto vertę. Populiariausias metodas yra naudojant artimiausius objektus parduodamo būsto, įvertinti kiek panašus būstas parduotas ankščiau ir įvertinus bendrą kainų kilimą, galėtų būti vertas dabar. Pavyzdžiui, jeigu butas yra parduodamas Žemaitės gatvėje, Vilniaus mieste, tuomet mes žiūrėtume į visus panašius butus kurie buvo parduoti būtent toje gatvėje per atitinkamą laiką. Tuomet paėmę kvadrato kainą ir įvertinę kokios ypatybės galėjo lemti tokią kainą, atsižvelgę į NT kainų indeksą – galėtume apytiksliai prognozuoti, kiek parduodamas butas galėtų būti vertas dabar. Tuo labiau, šis metodas itin gerai veikia butams – jeigu yra parduodamas butas tame pačiame daugiabutyje, telieka atsižvelgti į buto dydį, jo įrengimą ir sandorio datą. Taigi, artimiausių kaimynų metodas geras sprendimas bandyti prognozuoti būsto vertę.

[14] savo darbe nagrinėjo metodus, kurie būtent išnaudoja koordinačių ir požymių panaudojimą siekiant optimalaus absoliučios procentinės paklaidos medianos (angl. *Median Absolute Percent Error* (MDAPE)). Šiam uždaviniui jis rinkosi tarp dviejų metodų k-artimiausių kaimynų modelio (KNN) ir neuroninių tinklų (NN). Vis dėl to, autoriaus idėja dėl artimiausių kaimynų metodo taikymo buvo kiek kitokia – jo tikslas buvo klasteriuoti duomenis į mažesnius rinkinius, tam kad

pagreitinti modelio greitaveiką, tačiau tuo pačiu metu pagerinti bendrus rezultatus būtent dėl atskyrimo – kiekvienam klasteriui būtų skirtas vienas „silpnas“ modelis, kuris jį optimaliai prognozuotų [15].

Autorius darbe išskyrė dvejas mokymo kryptis. Pirmoji yra kai kiekviename modelyje esantis klasteris naudoja ir mokymosi ir testavimo duomenis esančius tame pačiame klasteryje, iteraciniu būdu išmėginant įvairius parametrus, taip jog paklaida klasterio viduje esančių testavimo duomenų būtų kiek įmanoma mažesnė. Antruoju atveju, optimizavimas vyksta nebe klasterio lygiu, o visos architektūros – kiekvienas modelis veikiantis klasterio aplinkoje stengiasi iteraciniu būdu rasti tokius modelio parametrus, kurie leistų geriausiai prisidėti prie bendro rezultato – žemos architektūros paklaidos.




**3 pav.** Autoriaus Oxenstierna architektūros planas

Abejais atvejais iš pradžių buvo atliekamas klasterizavimas, kuomet buvo bandyti skirtingi parametrai, klasterių skaičius bei metodai naudojat tinklelio paiešką. Pirmosios architektūros atveju, buvo įvestas papildomas ribojimas, jog galimas klasterių skaičius yra 2-10 ir vienas klasteris negali būti mažesnis nei 30 kaimynų, tam kad būtų galima išskirti mokomąją ir testavimo imtį. Antrosios architektūros atveju jokių ribojimų nebuvo ir klasterių skaičius buvo neribojama, tačiau autorius vis vien laikė jog mažiausias narių skaičius turėtų būti 13. Galiausiai, pirmosios architektūros atveju buvo naudojamas tiek KNN regresorius, tiek NN, o antruoju atveju tik KNN. Taip pat verta pridėti, jog pirmuoju atveju naudojant KNN modelį, buvo išbandyti įvairūs atstumo skaičiavimo metodai, tačiau antruoju atveju, naudojant tą patį modelį, papildomam modelio lankstumui, buvo įterpta požymių atranka su svoriais – kiekvienas požymis turėjo savo svorį [0;1] skaičiuojant galutines prognozes. Tokiu būdu buvo atrinkti tik tie požymiai, kurie geriausiai prisidėjo prie prognozių sudarymo.

Autorius eksperimento metu naudojo 6-12 mėnesių duomenis, o testavimui 3 naujausius mėnesius apie įvykdytus pardavimus. Kadangi praeities informacija apie paskutinį atliktą pardavimą buvo naudota, požymių inžinerijos metu ši informacija buvo ištraukta mėnesių pavidalu. Darbe autorius

siekė išsiaiškinti kiek duomenų klasterizavimas pagerina prognozes, lyginant su paprastu tiesiniu modeliu, bei naudojant KNN ir NN algoritmus neatlikus duomenų padalijimo į klasterius.

**6 lentelė.** Autoriaus Oxenstierna rezultatai

| | Linear | kNN All | ANN All | kNN SC | ANN SC | kNN MC |
|---|---|---|---|---|---|---|
| MDAPE (%) | 11 | 9.02 | 8.35 | 8.47 | 8.35 | 8.31 |
| MAPE (%) | 18.5 | 14.9 | 13.5 | 14.2 | 13.7 | 13.6 |
| CPU-time training (min) | 0.1 | 17 | 26 | 11 | 18 | 14 |
| CPU-time predicting (min) | 0.1 | 1.3 | 0.1 | 1.5 | 0.1 | 7.3 |

Rezultatų lentelėje pateikiami visi atlikti testai, kur „All“ yra visi duomenys, neatlikus padalijimo, „SC“, kuomet kiekvienas modelis per klasterį buvo optimizuotas jo paties viduje, o „MC“ – kai visi modeliai buvo optimizuoti galutiniam visos architektūros spėjimo paklaidai sumažinti. Galime matyti, jog pastaroji architektūra, nors ir nežymiai, buvo geriausia. Vis dėl to, lyginant su neuroniniais tinklais rezultatas pagerėjo nedaug, tačiau architektūra tapo sudėtingesnė ir lėtesnė – prognozės truko 7 minutes, vietoj 1/10 minutės. Kita vertus, NN modeliai yra „juodoji dėže“ ir juos paaiškinti todėl yra sunku, todėl net ir sudėtingesnis, tačiau lengviau paaiškinamas modelis gali būti geresnė alternatyva naudoti praktikoje, versle.

### 1.3. Būsto kaina kaip BKI prognozės uždavinys

Šio tipo uždaviniai yra įgyvendinami per būsto kainos indeksą (BKI) (angl. *Housing Price Index* (HPI)). Šis indeksas, tai laike kintanti vidutinė būsto vertė, kuriai nustatyti baziniai metai (pavyzdžiui 2007=100). Turint indekso reikšmę ir žinodami pardavimo kainą, mes galime susieti pardavimo vertę su tuometiniu indeksu ir paprastai išsiskaičiuoti dabartinę ar būsimą vertę žinant dabartini BKI ar prognozuojamą. Tuomet dažnai BKI yra švelninamas siekiant pašalinti sezoniškumą, tokiu būdu indeksas dar geriau apibendrina būstą. Taip pat BKI indeksas, priklausomai nuo jų sudarinėtojų, pateikia indeksą pagal būsto tipą bei vietovę. Vietovė gali būti skirstoma nuo rajonų, iki pašto kodų. Bankams dažnai yra taikomas reikalavimas, jei yra prognozuojama užstato vertė, taikyti kuo mažesnį vietovės mastą. Nors šis principas yra gana paprastas, jis dažniau yra naudojamas esamai būsto vertei nustatyti, kai yra žinoma praeities vertė. Vis dėl to pritaikius, laiko eilučių uždavinį, BKI mes galime prognozuoti ir tokiu būdu turėti būsimas BKI reikšmes leidžiančias mums prognozuoti būstą į ateitį. Pagrindinis šio modelio trūkumas yra tas, jog praktikoje ne visi duomenys pašto kodų / gatvių yra lengvai prieinami, todėl dažniau yra naudojami mikrorajonų / savivaldybių lygio indeksai išskaidyti pagal būsto tipus. Dėl šios priežastis tai kenkia modelio prognozuojamajai galiai, kadangi, pavyzdžiui, ne visi butai Naujamiestyje, Vilniuje, yra verti vienodai – naujamiestis skirstosi į dalį artimesnę centrui, bei dalį vadinama „krasnūha“, kurioje žmonės nenori gyventi. Negana to, dažnai BKI yra ketvirtinis skaičius ir daug rečiau mėnesinis – tai tik apsunkiną vertės prognozavimą pamėnesiui. Taip pat kita itin svarbi problema, jog šis modelis yra itin paprastas ir gali būti tinkamas ir dažniausio išplanavimo, įrengimo, kokybės ir ypatybių būstui. Todėl dažnai galutinėse modelio architektūrose, šis modelis yra taikomas kaip galutinės architektūros sudedamoji dalis, siekiant atsižvelgti į kylančias būsto kainos tendencijas. Taip pat galime atvejai, kai turint itin plačia duomenų bazę, galima patiems sukurti savo BKI naudojant mašininio mokymosi modelius, kuomet BKI tampa

pritaikytas kiekvienam NT objektui individualiai. Tačiau toks sprendimas reikalauja itin plačios duomenų bazės, o tai nėra lengvai prieinama.

[16] savo darbe taip pat siekė atlikti BKI prognozę, tačiau vietoj laiko eilutės uždavinio, viską atliko per regresiją. Uždaviniui išspręsti duomenų rinkinį pateiktą Kaggle – Būsto Kainos Indeksas (angl. *Home Price Index*)[7]. Taip pat naudojo visus požymius išrašytus duomenų rinkinyje, kurį sudarė 99326 įrašai. Darbe buvo išmėginti 3 modeliai – Lasso, Ridge ir atgalinio mokymosi neuroninis tinklas.

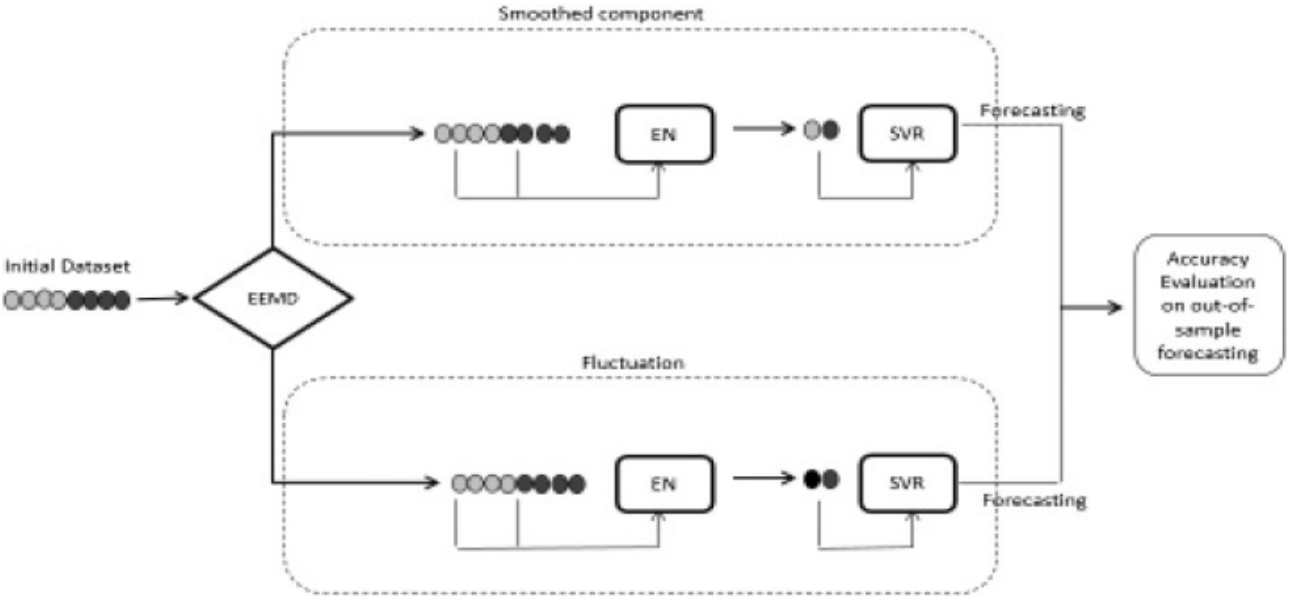
**7 lentelė.** Autoriaus Li modelio rezultatai

| Name of set | Final Results | | |
|---|---|---|---|
| | ***Lasso Regression*** | ***Ridge Regression*** | ***BP NN*** |
| Training set | 0.99925 | 0.99925 | 0.99949 |
| Testing set | 0.99925 | 0.99925 | 0.99949 |

Autorius pateikia R2 rezultatus, kurie šio tipo uždaviniui tikėtina yra pakankami. Galime matyti, jog net pritaikius paprastus modelius kaip Lasso ir Ridge, tačiau prieš tai atlikus parametrų optimizaciją – buvo gauti gana geri rezultatai prognozuojant ateities BKI reikšmes. Vis dėl to, darbe autoriai tiksliai nenusako, kokio ilgio yra prognozės langas. Tačiau kaip sprendimo būdas tai be abejo įdomus ir su stebėtinai gerais rezultatais.

Tuo tarpu [17] BKI prognozavimo kaip regresijos ir laiko eilutės uždavinį, įtraukdamas įvairius makroekonominius veiksnius: GDP per gyventoją, palūkanų normas, populiacijos dydį, akcijų kainas, statybų kainas, nedarbo lygį, infliacijos lygį, naftos kainą ir valstybės biudžeto dydį. Darbe siūloma iš pradžių glodinti turimus duomenis naudojant EEMD (angl. *Ensemble Empirical Mode Decomposition*), kadangi ne visi duomenys yra mėnesiniai. Tai leidžia dekomponuoti laiko eilutes į žemo ir aukšto variacijos lygiu laiko eilutes. Tuomet kiekvienai iš jų yra atrinkami svarbiausi požymiai su ElasticNet modeliu, o naudojant galutinius požymius panaudoti SVR modelį prognozėms, kur viena prognozė yra atliekama žemo dažnio laiko eilutei, o kita aukšto ir galiausiai šie rezultatai yra susumuojami.

[7] https://www.kaggle.com/datasets/PythonforSASUsers/hpindex




**4 pav.** Autoriaus V. Plakandaras modelio architektūra

Taip pat buvo išmėginti alternatyvūs laiko eilutės metodai: atsitiktinis vaikščiojimas (angl. *random walk),* BAR, BVAR. Taip pat galutinė architektūra buvo išmėginta pakeitus ElasticNet požymių atrankos dalį su autoregresijos elementu (EEMD-AR-SVR).

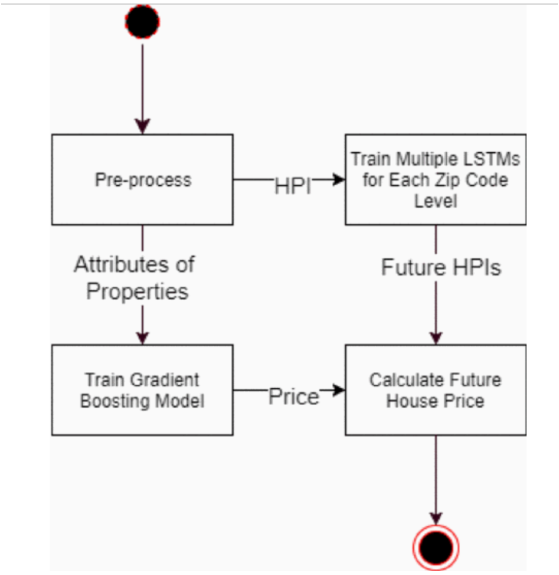
**8 lentelė.** Autoriaus V. Plakandaras rezultatai

| | In-sample accuracy | | Out-of-sample accuracy | |
|---|---|---|---|---|
| Model | MAPE (%) | DS (%) | MAPE (%) | DS (%) |
| RW | 4.732 | 56.701 | 5.352 | 70.833 |
| RW with a drift | 6.588 | 52.577 | 5.387 | 70.833 |
| BAR | 4.952 | 55.670 | 5.422 | 70.833 |
| BVAR | 4.809 | 56.701 | 11.931 | 75.000 |
| EEMD-AR-SVR(linear) | 1.985* | 84.615* | 2.151* | 87.500* |
| EEMD-EN-SVR(linear) | 2.337 | 78.495 | 5.990 | 87.500* |

Galime matyti, jog EEMD-AR-SVR architektūra tapo geriausia. Autoriaus prognozės langas yra 1 – tai itin mažas langas, todėl rezultatai tokie geri. Tačiau toliau autorius pateikia prognozės rezultatus iki 4 metų į priekį, kuomet EEMD-AR-SVR vis tiek pirmauja rezultatu. Apžvelgus šį bei ankstesnį darbą įdomų pažvelgti, kaip iš pažiūros laiko eilutės uždavinys yra sprendžiamas kaip regresijos ir sugeba pasiekti gerus rezultatus.

Kita vertus, yra autorių kurie BKI prognozavimą sprendžia griežtai kaip laiko eilutės uždavinį. Štai [18] turėdamas duomenis apie parduodamus būstus ir BKI iš statistikos departamento, jis regresijos daliai atlieka svarbių požymių atranką su Boruta, o BKI daliai – išskaidžius BKI pašto kodo lygmeniu apmoko LSTM modelius. Pagrindiniame duomenų rinkinyje yra pardavimų informacija 2013-2015, tačiau tik 2014-2015 pardavimai suskirstyti atitinkamai pagal BKI dažnumą buvo naudoti LSTM modelių tikrinimui. Galiausiai, apmokius laiko eilučių dalį ant 1975-2013 duomenų

ir regresijos dalį ant 2013 – abiejų modelių prognozės buvo apjungtos. Gautas rezultatas modelis – MAPE rezultatas 24.03% galutiniai būsto vertei.




**5 pav.** Autoriaus J. Bin architektūra

### 1.4. Būsto kaina kaip giliojo mokymosi uždavinys

Yra pastebėta, jog dažnai būsto kainos analizės susideda iš dviejų dalių: su būstu susijusių požymių analizės ir lokacijos, kuri yra siejama su tuo būstu. Dauguma klasikinių modelių, kaip minėta darbo pradžioje, yra hedonistiniai, atspindintys ekonominį požiūrį į kainą. Tačiau atsirandant vis geresniems dirbtinio intelekto modeliams, mes vis dažniau galime įtraukti ir su būsto lokacija susijusią informaciją. Ši informacija dažnai būna nestruktūrinė – teksto ir / ar vaizdo pavidalu. Šią papildomą informaciją galime panaudoti ne vienu būdu, tačiau literatūros apžvalgoje dažnai sutinkama giliojo mokymosi architektūra, kuri įgalina vaizdo / teksto rezultatus įtraukti kaip papildomas įvestis į naują neuroninio tinklo architektūrą ir tapti jo dalimi, arba išvesti rezultatus vektorine forma ir prijungti prie pagrindinių duomenų lentelės formoje.

[19], įkvėptas Zillow‘o įšūkio,[8] savo darbe nagrinėjo galimybę naudoti 640x640 raiškos satelitų nuotraukas, kuriose parduodamas objektas yra nuotraukos centre, ir duomenų rinkinį apie būstą. Darbas remiasi idėja, jog nuotraukų atpažinimu grįstas modelis geba prie duomenų rinkinio priderinti būstui būdingas savybes: atstumus iki parko, stotelės, parduotuvių ir t.t., taip patobulindamas bendrą modelio tikslumą 10% lyginant su modeliais, kuriuose nebuvo naudojamas nuotraukų atpažinimas.

Darbe iš pradžių yra naudojamas iš anksto apmokytas Inception-v3 [20] konvoliucinis neuroninis tinklas (CNN) ImageNet [21] duomenims, kuris gali atpažinti nuotraukose kampus ir smulkius požymius. Tuomet modelis buvo tikslinamas imties duomenimis (angl. „*fine-tunning*“). Tikslui įgyvendinti buvo papildomai naudojamas Google statinių žemėlapių API[9], kur naudojant būsto ilgumos ir platumos koordinates buvo gautos satelito nuotraukos.

[8] https://www.kaggle.com/c/zillow-prize-1

[9] https://developers.google.com/maps/documentation/maps-static/overview

Naudotame duomenų rinkinyje buvo 40 požymių, iš kurių 23 buvo kategoriniai, tačiau, kaip rašoma darbe, visi požymiai buvo normalizuoti ir standartizuoti. Iš pradžių buvo sudaryti 2 pradiniai modeliai be nuotraukų atpažinimo: medžių ir neuroninių tinklų modeliai, apie kurių parametrus darbe nėra kalbama. Geriausiai pasirodė medžių modelis, todėl jis kartu su patikslintu CNN modeliu buvo sujungti į vieną. Tai buvo atlikta paimant kiekvieno iš modelių požymių vektorius į atskirus tankio tinklus ir rezultatus pateikiant paskutiniame sluoksnyje.

Toliau pateikiami bandytų modelių rezultatai, kur (F) žymimi modeliai buvo duomenų rinkinys, o (I) satelito nuotraukos. Paskutinio ir geriausio modelio – „Neural Net F+I“ – hiperparametrai pateikiami kitoje lentelėje.

**9 lentelė.** Autoriaus S. J. Semnani modelių rezultatai, kur F - požymiai, o I - satelito nuotraukos

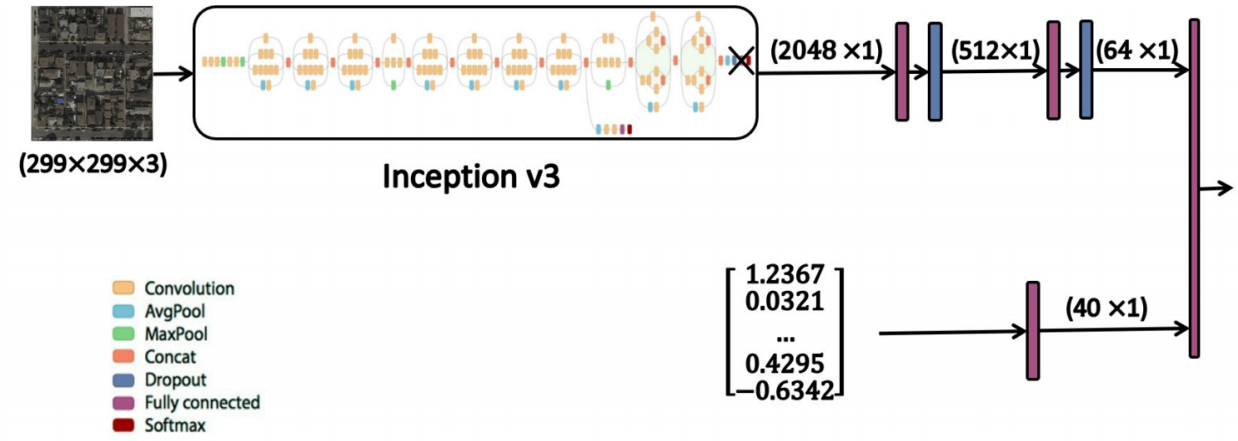
| Model | Train $R^2$ | Dev $R^2$ | Test $R^2$ | Train MSE | Dev MSE | Test MSE |
|---|---|---|---|---|---|---|
| Extra Tree (baseline) | 0.99 | 0.86 | 0.71 | 0.000 | 0.112 | 0.002 |
| Neural Net F (baseline) | 0.96 | 0.84 | 0.85 | 0.037 | 0.136 | 0.001 |
| Neural Net I | 0.000 | -0.0001 | -0.038 | 1.062 | 0.859 | 0.010 |
| Neural Net F+I | 0.98 | 0.94 | 0.93 | 0.011 | 0.044 | 0.001 |

Kur F yra požymiai, o I satelito nuotraukos. Moksliniame darbe autorius pastebi jo siekiant išvengti modelio persimokymo buvo naudota Ridge reguliarizacija ir atmetimo slenkstis.

**10 lentelė.** Autoriaus S. J. Semnani F+I modelio hiperparametrai

| | |
|---|---|
| L2 Regularization Parameter | 0.1 |
| First Dropout (drop probability) | 0.3 |
| First Dropout (drop probability) | 0.2 |
| Activation Function | ReLU |
| Batch Size | 1024 |
| Optimizer | Adam(learning rate=0.0005, $\beta_1 = 0.9$, $\beta_2 = 0.999$) |
| Epochs | 200 |
| Learning Rate Decay | $\alpha = 0.0001$ |

Toliau pateikiama modelio architektūra. Kaip ir buvo minėta iš vienos pusės yra pateikiama sluoksniuota (angl. *pooling*) 640x640, kuri yra išskaidoma į 3 pilnai sujungtus užslėptus sluoksnius vektoriais 2048x512x64. Taip pat greta yra 1 paslėpto sluoksnio neuroninis tinklas iš 40 neuronų, kuris paima iš medžių gautus požymių koeficientus, kurie yra paimami kaip pradinis sluoksnis. Paskutiniame modelio sluoksnyje, rezultatai iš paslėptų sluoksnių yra sujungiami į galutinį rezultatą.




**6 pav.** Autoriaus S. J. Semnani F+I modelio architektūra

[22] savo darbe lokaciją panaudojo naudojant geografinę erdvę naudojant „Geo-Spatial Network Embedding“ (GSNE). Pasak autoriaus, šis metodas gerai užfiksuoja aplink nekilnojamąjį turtą esančias mokyklas, viešojo transporto stoteles ir t.t. Šiam tikslui pasiekti yra pasitelkiami grafai, kurie naudoja Gauso principus, kurie kituose darbuose buvo įrodyti jog efektyviau ir patikimiau veikia esant išskirtims, tokioms kaip prabangūs butai. Modelyje namai ir kaimynystėje esantys objektai vaizduojami kaip daugiapartinis grafas (angl. *multipartite graph*) kuriame skirtingų tipų mazgai vaizduoja šiuos objektus, o kraštai erdvinį atstumą tarp jų. Pagrindinė siūlomo modelio idėja yra erdvinio tinklo mazgų projektavimas į Gauso funkcijų erdvę.

### 1.5. Būsto kaina kaip daugiaužduotinis uždavinys

[23] savo darbe atkreipia dėmesį, jog įvairūs moksliniai darbai [24], [25], [26], [27] jau yra parodę kaip stipriai vietovė įtakoja būsto kainą ir siekiant geresnių rezultatų būstai yra skaidomi į geografinius vienetus, kuriems yra sudaromi atskiri modeliai. Tačiau tokio požiūrio problema yra ta, jog šie atskiri modeliai neįvertina to, jog rajonai yra tarpusavy susiję. Taip pat problema atsiranda ir dėl to, jog vienuose rajonuose duomenų yra daugiau, nei kituose. Šiai problemai spręsti autorius pasiūlo MTL modelį kaip sprendimą aukščiau minėtoms problemoms, bei kuris pagerina geriausius tuometinius modelius šiam uždaviniui spręsti.

MTL yra modelis galintis dalintis informacija tarp skirtingų užduočių, tokiu būdu pagerinantis rezultatus. Šis būdas taip pat išsprendžia ir duomenų imties dydžio problema mažesniuose rajonuose. MTL yra svarbu nuspręsti 2 dalykus: kaip aprašyti uždavinius ir sąryšius tarp jų. Taip pat Guangliang, sudaro itin išsamų vietovės profilį, į kurį įtraukia įvairią informaciją apie geografinę padėtį kurioje bus nagrinėjami skelbimai. Į ją įeina informacija apie artimiausias mokyklas, viešojo transporto stoteles, atstumus iki darboviečių, atstumus iki parduotuvių ir ligoninių. Iš 14 nagrinėtų darbų, jie pirmieji sugalvojo sutalpinti tiek daug informacijos į konkrečią lokaciją.

**11 lentelė.** Autoriaus G. Gao turimų požymių palyginimas su kitais autoriais

| | Data Examples | References | | | | | | | | | | | | | | Ours |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | [5] | [6] | [7] | [8] | [11] | [12] | [13] | [15] | [27] | [28] | [29] | [30] | [31] | [32] | |
| Scale of data | 100 | ✔ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✘ | ✔ | ✘ | ✘ |
| | 1, 000 | ✘ | ✘ | ✔ | ✔ | ✘ | ✘ | ✘ | ✔ | ✘ | ✔ | ✘ | ✔ | ✘ | ✔ | ✘ |
| | 10, 000 | ✘ | ✔ | ✘ | ✘ | ✔ | ✔ | ✔ | ✘ | ✔ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ |
| | 100, 000 | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ |
| House profile | floor area, number of bedrooms | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| | geo-information, address, suburb | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ | ✘ | ✔ | ✔ | ✘ | ✔ | ✔ | ✔ |
| | air condition, water, heating views | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ | ✘ | ✔ | ✘ | ✔ | ✔ |
| Education profile | nearby schools | ✘ | ✘ | ✔ | ✘ | ✔ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✘ | ✔ |
| | school districts | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✘ | ✘ | ✔ |
| | school rankings | ✘ | ✘ | ✘ | ✘ | ✔ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✘ | ✘ | ✔ |
| Transportation profile | nearby public transport | ✘ | ✘ | ✔ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✘ | ✔ | ✔ | ✔ |
| | travel time to work | ✘ | ✔ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✘ | ✔ |
| Facility profile | hospitals, shops | ✘ | ✘ | ✔ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ | ✔ |
| | distance to nearest hospitals | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✔ | ✘ | ✔ | ✔ | ✔ | ✘ | ✔ |

Modelio įgyvendinimui yra daug architektūrinių sprendimų: Cross-stitch Networks [28] ir SemifreddoNets [29]. Tačiau svarbu paminėti, jog ne visus uždavinius galime apjungti [30] nes vieni uždaviniai padeda vieni kitiems išgauti papildomos informacijos, tuo tarpu kiti – tik trukdo. Turint architektūra svarbu sukurti nuostolio ir optimizavimo funkcijas. Nuostolio funkcija dažniausiai susideda iš kitų nuostolio funkcijų per atitinkamą uždavinį, tačiau konkretus šių funkcijų apjungimas labai priklausys nuo nuostolių dydžių tarp užduočių, kurie nuostoliai yra aktualesni pačiam modeliui, bei kurie lengviau yra apmokomi, tačiau visa tai galima atlikti automatiškai [31], [32]. Vis dėl to, šio tipo modeliai gauna ir kritikos jog lyginant su STL (angl. *Single-Task Learning*) šie modeliai yra sunkiau prižiūrimi kai reikia daryti pakitimus, taip pat dažnai būna sunku suderinti užduotis tarpusavy, galiausiai MTL modeliai ne visuomet garantuoja geresnius rezultatus dėl negatyvaus perkėlimo (angl. *negative transfer*). Įrankiai kurie buvo naudoti:

- Google Maps API[10]: atstumams ir laikui matuoti tarp jų
- GTFS[11] - viešojo transporto atstumams ir laikui matuoti.

Taip pat į būsto kainų uždavinį buvo pažiūrėta ir iš laiko eilučių pusės, kuomet yra taikomi LSTM modeliai ar LSTM modelių savotiškas ansamblis (angl. *stacked LSTM*) [33]. Vis dėl to, šiam uždaviniui spręsti reikėtų daug duomenų už ilgą laikotarpį, kurių Lietuvos mastu būtų sudėtinga gauti ir reikėtų pakankamai daug laiko surinkti pakankamą kiekį duomenų. Todėl būstų kainų prognozavimo, kaip laiko eilučių uždavinio darbai nebuvo nagrinėti.

## 1.6. Išvados

Darbe buvo stengtasi į būsto kainos prognozavimo uždavinį pažiūrėti iš įvairių pusių: kaip regresinį uždavinį, BKI ir kaimynų uždavinius, giliojo mokinimosi uždavinį, daugiaužduotinį uždavinį ar kaip laiko eilutes. Apžvelgtų darbų rezultatai yra pateikiami žemiau. Svarbu paminėti, jog darbuose skyrėsi modelio vertinimo kriterijai, todėl buvo pateiktos įvairios metrikos. Taip pat darbuose kuriuose buvo naudoti specifiniai požymiai kaip naudota lokacija, nuotraukos ar tekstas, žymimi „x“.

**12 lentelė.** Rezultatų suvestinė

[10] https://developers.google.com/maps
[11] https://developers.google.com/transit/gtfs

| Rezultatų suvestinė | | |
|---|---|---|
| **Šaltinis** | **Algoritmas** | **Rezultatas** |
| [1] | Ensembles of regression trees | MAE=0.371(0.0075), MSE=4.346(0.1548) |
| [6] | Stacked Generalization Regression | RNSKE=0.16350 |
| [7] | RF+LSTM Ensemble | RMSLE=0.23847 |
| [19] | NN | R2=93, MSE=0.001[12] |
| [14] | KNN MC | MDAPE=0.0831 |
| [22] | Geo-spatial network embeddings (GSNE) | RMSE=0.181, MAE=0.125 |
| [23] | MTL | RMSE=0.184-0.262, MAE=0.135-0.187 |
| [9] | GBM | R2=90.4, RMSE=0.08903, MSE=0.00793, MAPE=0.32251 |
| [10] | 0.65Lasso+ 0.365Xgb | RMSE=0.11260 |
| [11] | LSSVM | MAPE=0.228 |
| [12] | RIPPER | MAE=0.2488 |
| [13] | Stepwise and tuned SVM, | MSE=0.0561 |
| [18] | GB+LSTM | MAPE=0.2403 |

Nors buvo naudoti įvairūs metodai, galime pastebėti tendenciją jog darbuose nebėra naudojami hedonistiniai modeliai, o vietoj jų vyrauja mašininio mokinimosi modeliai. Nors rezultatai buvo vertinti skirtingais kriterijais, taip pat galime matyti jog geriausi rezultatai buvo pasiekti naudojant apie būsto skelbimo susijusią lokaciją. Tam dabar yra daug galimybių pasitelkiant Google žemėlapių API: galime ne tik naudodami būsto koordinates gauti satelito nuotraukas, tačiau ir pasiskaičiuoti atstumus iki viešojo transporto stotelių, traukinio stočių.

Vis dėl to, buvo rasti ir keletas modelių, kurie stengdamiesi sukurti kiek įmanoma daugiau požymių naudojo teksto ir nuotraukų atpažinimo modelius. Šie modeliai pasiekė gan aukštus rezultatus, todėl derėtų nepamiršti jog nereikėtų apsieiti tik su turimais duomenimis, tačiau ir pagalvoti kaip būtų galima sukurti naujų požymių, kurie galėtų būti prasmingi galutinio modelio sudarymui.

Vienas įdomesnių sprendimų buvo naudoti daugiaužduotinį mokymąsi (MTL), kuris plačiai yra naudojamas Tesla automobiliuose. Pagal šį modelį buvo sukurti įvairūs uždaviniai, kurie sprendė būsto kainas skirtinguose rajonuose, reitingavo vietines mokyklas ir t.t. Apjungiant įvairius modelius būtų galima pagalvoti apie MTL architektūra kuri ne tik bando nustatyti kainą skirtinguose rajonuose, tačiau ir įvertinti būsto būklę iš nuotraukų ar pridėtinę vertę iš skelbimo aprašymo. Taip pat galime įkomponuoti ir „Atviras Vilnius“[13] atvirus duomenis apie darželius, mokyklas ir kt.

Taip pat keliuose darbų buvo pastebėta, jog pašalinus išskirtis – itin aukštos kainos, ar itin žemos kainos būstus – modelio rezultatai pagerėjo. Mūsų atveju, būtų galima išskaidyti būtus į papildomas kategorijas prabangos (kaina didesnė nei 0.3 mln. Eur.), vidutinės kainos, dažniausiai perkami (kaina 0.05-0.3 mln. Eur.), bei žemos kainos, bendrabučio tipo, prastos vietovės butus (kaina mažesnė nei 0.05 mln. Eur.).

Kitas svarbus dalykas, jog pastebima tendencija, jog vieno modelio, kaip sprendimo – nebepakanka ir yra pradedama stengtis apjungti įvairius modelius į vieną. Šiame tikslui pasiekti yra įvairių architektūrų tačiau tos, kurios buvo pastebėtos apžvalgoje buvo ansambliai, sluoksniavimas (angl.

[12] Paskaičiuota ant normalizuotų duomenų

[13] https://atviras.vilnius.lt/

*Stack*), kuriam yra atskira Python programavimo kalbos paketas ***vecstack***[14], MTL ar kaip daugybinės įvesties neuroninius tinklus (angl. *Multi-Input Neural Networ*). Tačiau pastebima ir tai, jog įvairūs mašininio ir giliojo mokymosi modelių gali būti pritaikyti proceso eigoje kuriant naujus požymius galutiniam duomenų rinkiniui.

Taip pat pastebėta, jog nėra itin daug darbų, kurie nagrinėti kitokias architektūras nei regresija per mašininį ar gilųjį mokymąsi. Dauguma sprendimų apsiriboja kurdami sudėtingesnius sprendimus bandydami įtraukti vaizdą, tekstą, bei papildomą informaciją į galutinę regresijos prognozę. Vis dėl to, [7] ir [18] savo darbuose sugebėjo įkomponuoti ir laiko eilutes, įtraukdami makroekonominius veiksnius į modelių prognozavimą. Tačiau kaip matyti per rezultatus, vieno galutinio sprendimo nėra, tačiau kaip jau buvo galima matyti – įvairių sprendimų kombinacija dažnai lemia geresnius sprendimus, nei individualūs sprendimai modelio kompleksiškumo kaina.

Galiausiai, pastebėta jog daug lemia ir tinkami hiperparametrų atrinkimo metodai, požymių atrankos / reguliarizacijos modeliai, bei tinkami kriterijai modelių kokybei vertinti. Hiperparametrų paieškai pastebėta jog buvo naudoti tinklelio paieška (angl. *GridSearch*), gradientiniai metodai, reguliarizacijai Ridge, Lasso, ElasticNet bei pažingsninė atranka. Modelių kokybei vertinti daugiausiai buvo naudoti RMSE, MAE ir MSE.

**Darbo tikslas**: tikslesnis NT turto vertės prognozavimas, pasitelkiant vartotojų kuriamą bei viešai prieinamą informaciją ir dirbtinį intelektą.

**Darbo objektas**: viešai prieinami Aruodas.lt Vilniaus mieste parduodamų butų skelbimai.

**Darbo uždaviniai**:
1. atlikti mokslinių darbų, nagrinėjančių pasirinktą temą, paiešką ir apžvalgą;
2. surinkti skelbimų informaciją iš Aruodas.lt ir paruošti duomenis;
3. įtraukti atviro teksto informaciją į galutinį regresijos modelį;
4. įtraukti vaizdo ir lokacijos informaciją į galutinį regresijos modelį;
5. optimizuoti požymių inžinerijos procesą siekiant padidinti jų prognostinę vertę;
6. parinkti ir suderinti modelio architektūrą, galinčią tiksliausiai vertinti būsto kainą;
7. apibendrinti rezultatus ir pateikti išvadas.

[14] https://github.com/vecxoz/vecstack

## 2. Tyrimo metodai

### 2.1. Duomenų rinkimas

Duomenų rinkimui buvo naudoti 2 pagrindiniai Python paketai: *BeautifulSoup* ir *Selenium*. Pirmoji biblioteka yra skirta HTML ir XML apdorojimui, kurios dėka visas HTML turinys esantys puslapyje gali būti skaitomas ir randamas naudojant atitinkamus paieškos metodus. Vis dėl to, tam tikri puslapiai gali būti sukurti taip jog naudotų dinaminį tekstą naudojant JavaScript. Tokio teksto su anksčiau minėta biblioteka rasti nepavyktų, kadangi pastaroji nuskaito tik HTML ir XML informacija. Dėl šios priežasties buvo pasitelkta kita biblioteka – *Selenium*.

Ši biblioteka yra lankstesnė, kadangi kartu su išorine programine įranga *WebDriver*[15], leidžia imituoti bet kurią žiniatinklio naršyklę ir per ją leisti automatinius veiksmus, tokius kaip paspaudimus ar teksto įvedimas į atitinkamus laukus. Tačiau svarbiausia, jog minėtą išorinę programinę įrangą galime užfiksuoti dinamiškai sugeneruotą tekstą paverčiant jį į HTML ir pasiekti jį per *BeautifulSoup*. Kartu, šie įrankiai leidžia iš svetainės ištraukti visą informaciją, bei tuo pačiu valdyti tam tikrus veiksmus, pavyzdžiui, išskleisti teksto aprašymą skelbimo aprašyme ar uždaryti iššokančius langus.

Dažnai gali kilti klausimas, dėl duomenų rinkimo iš interneto teisėtumo. Tai yra teisėta, jei nukrauname tik viešai internete pasiekiamus duomenis. Į šiuos duomenis neįeina asmens duomenys ar intelektinė nuosavybė. Vis dėl to, ši sritis, kaip ir dirbtinis intelektas, dar nėra gan tiksliai apibrėžti, todėl iš teisinės pusės svarbiausia atsižvelgti į GDPR ir Intelektualinės nuosavybės įstatymus.

Pagrindiniai principai kurių derėtų laikytis:

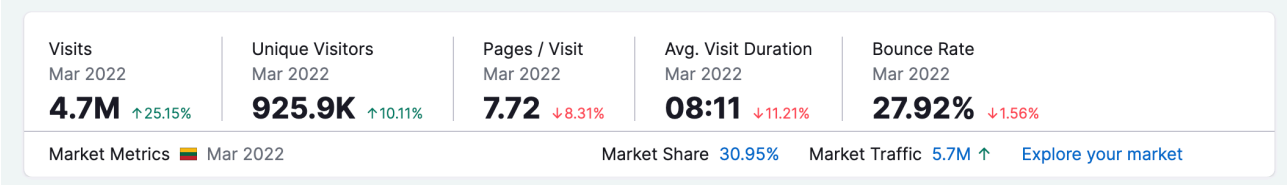
- siekti kuo labiau sumažinti naštą interneto svetainės savininkams;
- patenkinti svetainių savininkų pateiktus prašymus dėl duomenų rinkimo, aprašyto „robots.txt“ faile;
- saugoti visus asmeninius duomenis visose statistikos ir tyrimų rezultatuose;
- taikyti mokslinius principus rengiant statistiką ir tyrimus, pagrįstus nuasmenintais / užšifruotais duomenimis;
- laikytis visų galiojančių teisės aktų ir stebėti besikeičiančią teisinę situaciją.

Aruodas.lt neturi „robots.txt“ failo, kurį galime patikrinti per įvairius įrankius, pavyzdžiui, https://seositecheckup.com/analysis. Tai reiškia, jog svetainė neturi ribojimų robotų veiklai pačioje svetainėje. Vis dėl to, visas svetainės turinys nėra reikalingas, todėl svetainės medyje, pagrindinis informacijos rinkimas vyko tik šioje svetainės dalyje: https://www.aruodas.lt/butai/vilniuje/.

Sekantis sprendimas buvo nuspręsti duomenų rinkimo strategiją, jog duomenų rinkimas neaprautų svetainės resursų. Parduodamų būtų skelbimų Vilniuje nuolat galima rasti 1800-2300, taip pat galime teigti jog naujų butų kasdien atsiranda 5-50 (tačiau nebūtinai kasdien). Todėl tik pirminis duomenų rinkimas reikalautų daugiau svetainės užklausų pateikimo, o visi tolimesni būtų tik palaikomieji. Aruodas per mėnesį sulaukia ~4.7 mln. Lankytojų per mėnesį SemRush[16]

[15] https://www.selenium.dev/documentation/webdriver/
[16] https://www.semrush.com/

duomenimis. Todėl papildomos <2000 užklausos (daroma prielaida jog kas mėnesį maksimalus skelbimų skaičius gali siekti 2000) neturės įtakos svetainės veikimui.




**7 pav.** SemRush pateikta informacija. žiūrėta 2022-04-28

Vis dėl to, derėtų atsižvelgti į užklausų skaičių per sekundę skaičių. Kadangi robotizuotas sprendimas gali siųsti po užklausą kas sekundę ar net greičiau, derėtų riboti užklausų siuntimo kiekį per konkretų intervalą, priešingu atveju tai gali būti interpretuojama kaip DDoS[17] ataka. Dėl šios priežasties, užklausos bus siunčiamos atsitiktinai kas 2-7 sekundes. Tokiu būdu bus siekiama neapkrauti serverio, tačiau turint omeny užklausų skaičių, net ir siunčiant visus 2000 užklausų vienu metu, Aruodas.lt puslapiui pakenkti neturėtų pavykti.

Taip pat, visi surinkti duomenys yra naudojami tik akademiniams tikslams, todėl surinkta informacija nebus dalinama ar parduodama. Galiausiai, skelbimuose esanti jautri informacija tokia kaip brokerių vardai, pavardės, jų telefono numeriai ir el. adresai, būdami teksto dalimi yra vektorizuojami į skaitinę išraišką, todėl tiesiogiai naudojami nėra. Visa nuotraukose esanti jautri informacija taip pat yra nuasmeninama per vektorizavimo procesą, todėl tiesiogiai nuotraukos taip nėra naudojamos. Galiausiai buto numeris (jei yra nurodytas), kaip požymis nėra naudojamas regresijos uždavinyje.

Su visais aukščiau minėtais veiksmais siekiama, jog duomenų rinkimas iš Aruodas.lt būtų skaidrus ir etiškas.

## 2.2. Sent2Vec

[7] su kolegomis savo darbe panaudojo skelbimų aprašymus, naudojant LSTM modelį. Savo darbe, siekiant supaprastinti modelio architektūrą, bus naudojamas teksto vektorizavimas (angl. *text embeddings*). Paties metodo esmė labai paprasta ir artima NLP uždaviniams, kuomet yra sudaromas TF-IDF ar BoW žodžių rinkinys, kur vėliau žodžiai/frazės yra susiejami pagal savo panašumą. Tačiau vietoj teksto prognozavimo, žodžiai paversti skaitine išraiška ir susietini panašumu yra naudojamas kaip vektorius modeliams naudojantiems skaitinio tipo kintamuosius. Mūsų atveju, tai regresijos uždavinys.

Žodžių ir frazių pasiskirstymo ir sudėties nustatymui galimi 2 metodai [34]: tęstinis žodžių maišas (angl. *Continuous Bag of Words*) ir n-gramų įterpinių (angl. *The skip gram model*). Pirmasis apima konteksto žodžių nuspėjimą naudojant centrinį žodį, o kitas apima žodžio nuspėjimą naudojant kontekstinius žodžius. Tą pačią idėją galima išplėsti sakiniams ir dokumentams. Word/Sent/Doc2Vec remiasi n – gramų įterpiniais. Ši savybė ir yra šių modelio tipo išskirtinumas lyginant su kitais modeliais.

[17] https://en.wikipedia.org/wiki/Denial-of-service_attack

Pats modelis yra sukurtas neuroninių tinklų pagrindu, kuriame yra vienas sluoksnis. Pirmiausia, į paieškos sluoksnį įkeliami dokumentai žodžio n-gramų lygiu, kur kiekviena n-grama vektorizuojama iki žodžio. Toliau, visi žodžiui gauti n-gramų vektoriai yra suvidurkinami, kad būtų gaunamas vienas vektorius identifikuojantis žodį o vėliau - dokumentą. Paslėptame sluoksnyje yra įterpiami pasirinktieji parametrai, tokie kaip vektoriaus dimensionalumas ar žodyno dydis.

N – gramais vadiname šalia esančių elementų seką iš n elementų, svyruojančią nuo 1 (vienas žodis) iki n-tojo elementų skaičiaus. Šiuo metodu galima aptikti itin retus ir didelę informacinę vertę turinčius žodžius, tuo pačiu, modelis sugeba ir toliau veikti ir vektorizuoti žodžius kurių nėra sudarytame žodyne bei yra atsparus įvairioms žodžio kalbinėms formoms. Metodas pasižymi efektyvumu, gerais rezultatais, tinkamas dideliems duomenų kiekiams [35].

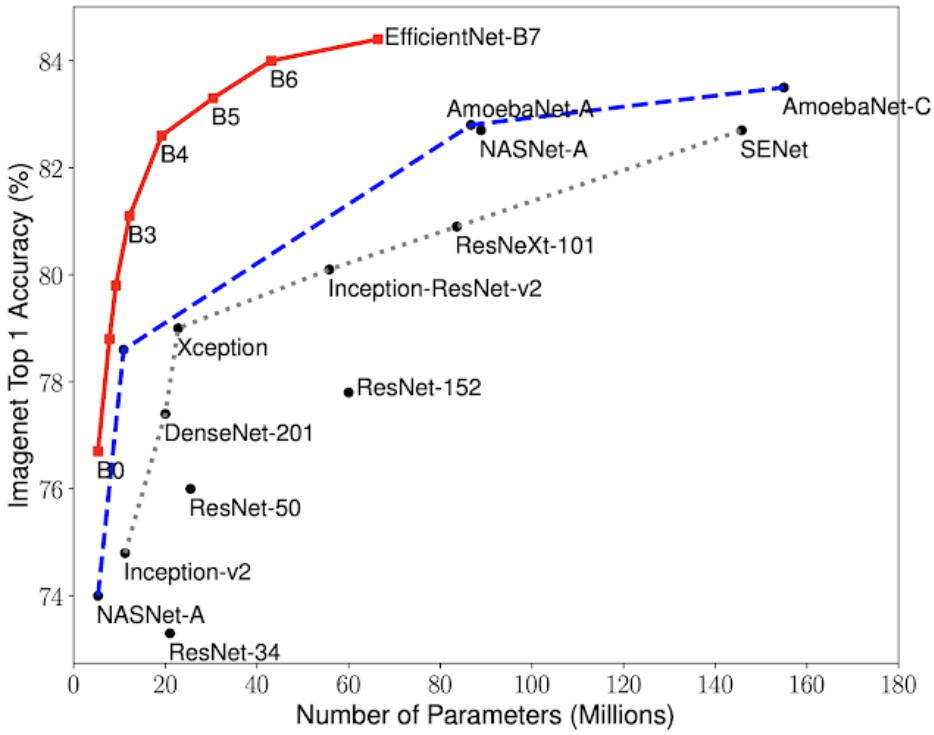
Kaip ir buvo minėta, egzistuoja modelio alternatyvos žodžio, sakinio ir dokumento lygmeniu. Žodžių atveju (Word2Vec), modelis veiksmingai fiksuoja semantinius ryšius tarp žodžių, todėl gali būti naudojamas žodžių panašumui apskaičiuoti arba naudojamas kaip įvairių NLP užduočių kaip kad nuotaikų/semantinei analizei vykdyti. Vis dėl to, gretimi žodžiai negali užfiksuoti daug esmės, tam reikia ryšių tarp sakinių ar dokumentų, o ne pavienių žodžių ir jų n-gramų rinkinių.

Kadangi mūsų darbe teksto nebus daug – NT būstų skelbimų aprašymai – savo darbe naudosime sakiniams skirtą modelį Sent2Vec.

### 2.3. EfficientNet

Kaip negalime įtraukti teksto į modelius naudojančius skaitinius kintamuosius, taip pat negalime įtraukti ir paveikslėlių. Literatūros apžvalgoje autoriai naudojo konvuliucinius neuroninius tinklus, tam kad galėtų išmokti atpažinti atitinkamas savybes esančias nuotraukose siejamomis su vienokia ar kitokia kaina. Šiame darbe, mes vis dėl to žengsime kita kryptimi ir paveikslėlius, kaip ir tekstą, vektorizuosime. Tam yra įvairių modelių, tačiau darbe naudosime *EfficientNet* [36].

„EfficientNet“ yra ypač įspūdingas, nes jame yra automatiškai sugeneruotų modelių klasė su kompromisu tarp parametrų skaičiaus ir tikslumo. Modelių klasės skirstomos nuo B0 iki B7, kur parametrų skaičius atitinkamai 5.3 mln parametrų, o didžiausiame B7 – 66 milijonai. Iš nuotraukos pateiktos žemiau galime matyti jog atitinkamai ir modelio tikslumas kinta priklausomai nuo modelio klasės, tačiau pastebima tai jog su mažiau parametrų EfficientNet sugeba pasiekti geresnius rezultatus nei kiti modeliai.




**8 pav.** EfficientNet ir kitų modelių palyginimas per parametrų skaičių ir tikslumą Šaltinis: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

Taip pat modelis yra ir efektyvesnis nei kiti modeliai: mažiausias modelis atlieka 0.39 milijardų slankiojo kablelių operacijų per sekundė (FLOP‘s), o didysis – 37 milijardų. Modelio konkurentai atlieka nuo 4.1 iki 19 kartų daugiau operacijų ir vis vien pasiekia prastesnius rezultatus.

Taip pat pastebėtina, jog modeliai jau yra apmokyti ir juos galėsime naudoti tik šiek tiek pakoregavę svorius (angl. *fine tuning*).

### 2.4. Tiesinė regresija

Bet kuriame regresiniame uždavinyje, neapsiesime be tiesinės regresijos – šis modelis nepaisant savo paprastumo ir trūkumų vis dar yra plačiai naudojamas. Paprasta tiesinė regresija, yra vieno priklausomojo intervalinio kintamojo priklausomybė nuo vieno ar daugiau kitų kintamųjų, kurie nebūtinai visi taip pat yra intervaliniai. Jų priklausomybė yra išreiškiama tiesiniu modeliu ir užrašoma:

$$Y = \alpha + \sum_{i=1}^{n} \mathrm{B}_i X_i + \epsilon$$

Nors darbo atveju kainos prognozuoti mes tiksliai su šiuo modeliu negalėsime, tačiau naudojant „stacked“ modelių metodą, tikrai galėsime šį modelį panaudoti galutiniam rezultatui gauti. Vis dėl to, pritaikius šio modelio reguliarizaciją jo pritaikomumas tampa aukštesnis.

Reguliarizacija yra regresijos rūšis, kai įverčių koeficientai yra traukiami link nulio. Ji naudojama tam, kad modeliai sumažintų persimokymo riziką bei tam, kad modelis būtų supaprastintas. Reguliarizacija paremta tuo, kad prie paklaidų kvadratų sumos funkcijos RSS yra pridedama tam

tikra nuobauda. Ir gautą funkciją bandoma minimizuoti. Regresijai be reguliarizacijos taikome šią nuostolių funkciją:

$$RSS = \sum_{i=1}^{n} \left( y_1 - \mathrm{B}_0 - \sum_{j=1}^{p} \mathrm{B}_j x_{ij} \right)^2$$

Ridge regresija – minimizuoja β koeficientų poveikį modeliui. Parametras λ nurodo reguliarizacijos stiprumą. Verta naudoti, kai tam tikri parametrai daro per didelę įtaką modeliui, versdami jį persimokyti.

$$RSS_{Ridge} = RSS + \lambda \sum_{j=1}^{p} \mathrm{B}_j^2$$

Lasso regresija – ne tik minimizuoja β koeficientų poveikį modeliui, tačiau gali ir visiškai juos pašalinti, jei β tampa lygus 0. Parametras λ nurodo reguliarizacijos stiprumą. Verta naudoti, kai yra naudojama daug kintamųjų, kai dalies kintamųjų svarba modeliui yra nedidelė.

$$RSS_{Lasso} = RSS + \lambda \sum_{j=1}^{p} |\mathrm{B}_j|$$

Elastic net regresija – naudojamos baudos tiek iš Lasso, tiek iš Ridge regresijos. Koeficientai β yra ir mažinami, ir atsisakoma nesvarbių kintamųjų.

$$RSS_{Elastic\ net} = RSS + \lambda_1 \sum_{j=1}^{p} \left|\mathrm{B}_j\right| + \lambda_2 \sum_{j=1}^{p} \mathrm{B}_j^2$$

Elastic net reguliarizacija sujungia tiek Ridge, tiek Lasso reguliarizacijos funkcionalumus. Šio reguliarizacijos tipo privalumai yra tai, kad šis metodas galimas naudoti net tais atvejais, kai kintamųjų skaičius yra didesnis nei stebėjimų bei šis metodas skatina kintamųjų grupavimo efektą, kur stipriai koreliuojantys kintamieji būna kartu arba naudojami, arba nenaudojami modelyje [37]. Reguliarizacija yra pagrindiniai tiesinės regresijos įrankiai, tačiau ne vieninteliai.

Be reguliarizcijos, skirtinguose tiesinės regresijos modeliuose galime rasti ir skirtingus metodus apskaičiuojančius požymių koeficientus. Įprastai, tai yra daroma pasitelkiant mažiausių kvadratų metodą, tačiau yra modelių kurie gali naudoti ir kitus metodus, pvz.: Bajeso[18] tikimybę. Taikant šį regresijos metodą, vietoj mažiausių kvadratų nustatymo nustatomas aposteriorinis požymių pasiskirstymas. Bajeso tiesinė regresija yra panaši į tiesinę, tačiau yra stabilesnė nei paprasta tiesinė regresija. Kita alternatyva galėtų būti stochastinis gradientinis nusileidimas, kuris taip pat yra gan plačiai naudojamas.

Kitas metodas taip pat dažnai naudojamas tiesinėje regresijoje yra atraminių mašinų modelis (angl. *Support Vector Machine* / *Support Vector Regressor*). Skirtingai nuo įprastų regresijos modelių, kurie bando atrasti minimalią paklaidą tarp tikrosios ir numatomos vertės, šis modelis stengiasi

[18] https://lt.wikipedia.org/wiki/Bajeso_teorema

pritaikyti geriausią liniją slenkstinės vertės ribose – atstumo tarp hiperplokštumos ir ribos linijos. Taip pat šis metodas gali būti nebūtinai tiesinis, dėl branduolio transformacijų, kurios leidžia transformuoti duomenis neturinčius tiesinės priklausomybės.

Galiausiai, kadangi tiesinė regresija yra statistinis modelis, pažeidus atitinkamas taisykles galima bandyti kitus modelius pvz.: kvantilių tiesinę regresiją, kai yra pažeidžiama heteroskedastiškumo sąlyga, polinomų regresiją, kai ryšys nėra tiesinis, alternatyvius metodus kai yra stebimos įtakingos išskirtys. Darbe todėl bus išmėgintos šios dvi papildomos technikos: stochastiniu gradientiniu nusileidimu grįsta regresija (angl. *SDGRegressor*) ir Bajeso teorema grįsta Ridge regresija (angl. *BayesianRidge*).

Pirmasis modelis, artimas ElasticNet tuo, jog turi L1 ir L2 reguliarizaciją, tačiau nuostolio funkcijai skaičiuoti naudoja SDG metodą (angl. *Stochastic Gradient Decent)*. Taip pat, nuostolio funkcijos įverčiai yra skaičiuojami iteraciniu būdu, todėl šis modelis yra tinkamas naudoti esant dideliam duomenų rinkiniui. Tuo tarpu antrasis modelis tinkamas esant nepakankamai reprezentatyviems ar prastai paskirsčiusiems duomenims, kuomet tiesinė regresija yra skaičiuojama ne iš taškinių įverčių, tačiau iš skirstinio, naudojant Bajeso teoremą.

### 2.5. Medžių modeliai

Kita svarbi modelių šeima mašininiame mokymąsi yra medžiai. Pats sprendimų medis kaip metodas yra labai paprastas ir linkęs į persimokymą, todėl stengiantis labiau apibendrinti šio modelio tipą yra kuriami medžių ansambliai, kurių idėja remiasi tuo, jog daugybė silpnų modelių – medžių – kartu gali veikti kaip vienas stiprus modelis. Vis dėl to, yra keletas ansamblių metodų bei pačių medžių „auginimo“ metodų, kuriuos aptarsime darbe.

Nors ansamblių metodas yra būdingas medžiams, tačiau pati idėja gali būti taikoma bet kokiems modeliams. Plačiąja prasme, ansamblio tikslas yra sujungti kelių bazinių modelių prognozes į vieną, pagerinant modelių naudojamų atskirai tikslumą ir gebėjimą apibendrinti. Šie baziniai modeliai gali būti tiek medžiai, tiek tiesiniai, neuroniniai tinklai ar jų kombinacija. Regresijos uždaviniuose ir jų konkursuose, ansamblių dėka dažnai yra gaunami geresni rezultatai, nei individualių modelių. Įprastai, yra išskiriamos 2 ansamblių šeimos:

1. Ansambliai naudojantys vidurkinimo metodą. Šie ansambliai, kaip ir išduoda pavadinimas, naudoja atskirų modelių rezultatų vidurkį, kuris paprastai būną geresnis rezultatas nei vieno iš bazinių. Šių modelių pavyzdžiai yra: visi savirankos (angl. *Bagging*) metodai, atsitiktiniai miškai ir jų skirtingos adaptacijos ir kt. Šiame darbe bus naudojami šie modeliai: atstiktiniai miškai (angl. *Random Forest*) papildomų regresijos medžių regresija (angl. *Extra-Tree*) ir regresijos medžių saviranka (angl. *Regression Tree Bagging*).
2. Ansambliai naudojantys pastiprinimo (angl. *Boosting*) metodą. Šiuose ansambliuose, baziniai įverčiai kuriami nuosekliai ir bandoma sumažinti kombinuoto įverčio paklaidą. Motyvacija yra sujungti kelis silpnus modelius, kad sukurti vieną stiprų. Modeliai priklausantys šiai šeimai, bei kuriuose naudosime šiame darbe yra: Ada pastiprinimas (angl. *Ada Boosting*), regresijos medžių gradientinis pastiprinimas (angl. *Gradient Tree Boost*), histograma grįsto regresijos medžių gradientinis pastiprinimas (angl. *Histogram Gradient Boosting),* ekstremalus gradientinis pastiprinimas (angl. *XGBoost*), „CatBoost“, švelni gradientinio didinimo mašina (angl. *Light Gradient Boosted machine / Light GBM*).

Tačiau, kaip matysime vėliau, tai nėra vieninteliai metodai, apjungti kelių bazinių modelių rezultatus vienam modeliui gauti.

### 2.6. Stacking metodas

Tai yra mašininio mokinimosi modelių ansamblis, kuris apima kelių modelių prognozių derinimą tame pačiame duomenų rinkinyje. Sudėjimas (angl. *Stacking*) kaip sprendimas, bando atsakyti į klausimą „Atsižvelgiant į kelis mašininio mokymosi modelius, kurie puikiai išsprendžia problemą, bet skirtingais būdais, kaip pasirinkti, kurį modelį naudoti (pasitikėti)?“. Šis klausimas sprendžiamas naudojant kitą mašininio mokymosi modelį, kuris išmoksta naudoti kiekvieną ansamblyje esantį modelį arba juo pasitikėti.

Priešingai nuo savirankos (angl. *Bagging*) metodo, sudėjimo (angl. *Stacking*) visi modeliai nėra sprendimų medžiai ir mokinasi ant tų pačių duomenų. Priešingai nuo pastiprinimo (angl. *Boosting*) metodo, čia naudojamas vienas modelis, siekiant sužinoti, kaip geriausiai derinti modelių rezultatus, vietoj modelių sekos koreguojančios ankstesnius rezultatus.

Kūrimo modelio architektūra apima du ar daugiau bazinių modelių, dažnai vadinamų 0 lygio modeliais, ir meta modelį, kuris sujungia bazinių modelių prognozes, vadinamą 1 lygio modeliu. Šio tipo ansamblio architektūra taip pat dažnas sprendimas įvairiuose Kaggle konkursuose, todėl darbe taip pat bus panaudotas. Taip pat svarbu paminėti, jog šis metodas nėra įgyvendinamas per atitinkamą modelį, kaip kad su „Bagging“ ir „Boosting“ metodais.

### 2.7. Papildomi modeliai

Darbe papildomai taip pat bus panaudoti 2 modeliai: artimiausių kaimynų modelis, skirtas regresijos uždaviniams spręsti, bei neuroninio tinklo architektūros pritaikymas per Sklearn[19] biblioteką kuriant daugiasluoksnį perceptroną (angl. *Multi-layer perceptron*) regresijai.

Pirmuoju modeliu bus siekiama atsižvelgti į kaimynystėje esančius skelbimus ir panašius būstus per centroidų giminės artimiausių kaimynų modelį, paimant tik keletą svarbiausių požymių – vietovę (ilguma, platuma), plotas, statybos metai. Antru modeliu siekiama išbandyti paprastą neuroninio tinklo architektūrą, siekiant įvertinti ar neskiriant daug dėmesio į neuroninio tinklo kūrimą, galime pasiekti dažniausiai naudojamų mašininio mokymosi modelių rezultatus. Iš literatūros apžvalgos, neuroniniai tinklai buvo taikomi, tačiau jie nepasiekdavo geriausių rezultatų, todėl ir darbe, bus bandoma paprasčiausias neuroninio tinklo architektūros įgyvenimo sprendimas.

### 2.8. Požymių inžinerija

Požymių inžinerija ML yra pats svarbiausias dalykas ir tuo pačiu vienas sudėtingiausių. Šis etapas gali būti nuolat tobulinamas ir plečiamas.

[38] savo darbe ištyrė 16 skirtingų požymių inžinerijos metodų poveikį 4 populiariems modeliams – neuroniniams tinklams, atsitiktiniui miškui, atraminių vektorių mašinai ir GBM. Darbe paaiškėja, jog skirtingi metodai generuojantys įvairias pradinių požymių kombinacijas ir jų transformacijas veikia skirtingai skirtingus modelius: medžių algoritmus GBM ir atsitiktiniams miškams naudingiausios buvo vienokios transformacijos, o SVM ir NN – kitokios. Todėl darbe atliekant

[19] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

požymių transformacijas bus atsižvelgta į 2 modelių šeimas kurios bus naudojamos – tiesinius ir medžių. Šioms šeimoms bus sukurti atskiri požymių inžinerijos grandinės (angl. *pipelines*), siekiant gauti optimalius rezultatus.

Aukščiau minėtame darbe taip pat buvo apžvelgtos ir skirtingos požymių inžinerijos transformacijos, tokios kaip skirtumai, daugyba, santykis, polinominės transformacijos, kvadratai, šaknys, logaritmai, atstumas, bei keletas kitų. Vis dėl to, šiame darbe bus naudojamos ne visos technikos minėtos darbe, siekiant sumažinti „juodosios dėžes“[20] efektą darbe.

## 2.9. Reikšmingų požymių atranka

Reikšmingų požymių atranka yra vienas svarbiausių mašininio mokymosi užduočių, duomenų parengime. Tai yra procesas, kurio metu yra parenkami požymiai naudingiausi tyrinėjamam uždaviniui (t.y. konkrečiam regresijos ar klasifikacijos uždaviniui). Ne visi požymiai esantys duomenų rinkinyje yra vienodi, tam tikra dalis jų yra triukšmingi ir neteikiantys jokios vertės modeliui – lėtindami skaičiavimus, užimdami daugiau vietos atmintyje. Tačiau taip pat šie požymiai gali ir pabloginti modelio rezultatus per persimokymą.

Darbe reikšmingų požymių atrankai naudosime „Boruta“[21] paketą. Tai yra požymių reitingavimo ir jų atrankos algoritmas sukurtas Varšuvos universitete. Šis algoritmas yra pagrįstas atsitiktiniais miškais (angl. *Random Forest*), tačiau gali būti naudojamas ir su kitais medžių algoritmais – XGBoost, CatBoost ir t.t. Priešingai nei pažingsninis požymių rinkimas (angl. *step-wise selection*) randamas tiesiniuose modeliuose, šis metodas atsižvelgia ir į priklausomybes tarp skirtingų kintamųjų, vietoj regresoriaus statistinio reikšmingumo regresantui.

Šio algoritmo veikimo principas yra toks, jog kiekvienam požymiui yra sukuriamas sintetinis jo atitikmuo, vadinamas šešėliniu požymiu (angl. *shadow feature*). Kiekvienas iš šių sintetinių požymių yra sudaryti iš atsitiktinių duomenų. Algoritmo idėja yra pamatuoti tikrų požymių svarbą, lyginant juos su sintetiniais požymiais, t.y. algoritmas atrenka tik tuos požymius kurie yra daro daugiau įtakos regresoriui, nei atsitiktinumas (per sintetinius požymius). Tai jis daro iteraciniu būdu primenančiu pažingsninę atranką.

## 2.10. Modelių parametrų optimizavimas

Hiperparametrų optimizavimas yra kitas itin svarbus žingsnis mokant mašininio mokymosi modelius. Turint daug optimizuotinų parametrų, bei didelę duomenų imtį su aukštomis dimensijomis – šis procesas gali tapti ne tik itin sudėtingas (dėl lokalių minimumų), bet ir ilgas, ypač jei paties modelio apsimokymo procesas yra gan ilgas (pvz.: neuroniniai tinklai). Negana to, taip pat reikia atsižvelgti ir į tokius dalykus kaip duomenų nutekėjimas apmokinant modelius kryžminiu metodu. Įprastai, paprastiems ir/ar mažiems projektams/uždaviniams dažnas pasirinkimas yra tinklelio paieška (angl. *Grid Search)* ar atsitiktinė paieška (angl. *Random Search*).

Tinklelio metodas itin paprastas – jo metu yra išbandomos visos nurodytos hiperparametrų kombinacijos ir parenkama ta, kuriuos įvertis yra geriausias. Šiame darbe šis metodas bus naudojamas ieškant tinkamiausios modelių vamzdžio architektūros sprendimo – nes alternatyvų yra

[20] https://en.wikipedia.org/wiki/Black_box
[21] https://github.com/scikit-learn-contrib/boruta_py

gan nedaug ir mums yra svarbu pamatyti kaip skirtingos architektūros skiriasi tarpusavyje ir kaip galime architektūrą supaprastinti neprarandant daug rezultato.

Tinklelio paieškos pagrindinis trūkumas yra laikas – šis metodas tampa itin lėtas dėl to, jog jis išbando visas galimas parametrų kombinacijas – jei šių kombinacijų yra daug tai tampa ilgu procesu. Ypač jei yra tenkinamos sąlygos minėtos aukščiau – sudėtingas modelis, didelis duomenų rinkinys. Siekiant pagreitinti skaičiavimus galime naudoti atsitiktinę paiešką, kuri iš visų kombinacijų pabandys atsitiktines. Šis sprendimas gali būti naudingas kai po tam tikros ribos hiperparametrų koregavimas pradeda tik nežymiai gerinti/bloginti rezultatus ir mums nerūpi žinoti apie visų kombinacijų įverčius. Vis dėl to, šio sprendimo trūkumas yra tas, jog jį reikia kartoti keletą kartų, kad būtume tikri dėl rezultatų ir neužstriktume lokaliame minimume.

Efektyvesnis parametrų parinkimo metodas yra naudojant Bajeso teoremą, kuri seka ankstesnio vertinimo rezultatus, kuriuos jie naudoja sudarydami tikimybinį modelį, susiejantį hiperparametrus su tikslo funkcijos balo tikimybe:

$$P(\text{įvertis} \mid hiperparametrai)$$

Ši tikslo funkcija vadinama surogatine/pakaitalu (angl. *Surrogate*). Šį pakaitalą yra lengviau optimizuoti nei tikrąją tikslo funkciją. Radus geriausius parametrus surogatinei tikslo funkcijai – tie patys parametrai yra panaudojami tikrojoje tikslo funkcijoje, o gauti rezultatai atnaujina surogato funkciją. Galiausiai kartojant šią operaciją kelis kartus yra pasiekiami optimalūs modelio hiperparametrai. Iš esmės galime sakyti, jog šio metodo tikslas yra tarp „mažiau klaidingu“ naudojant daugiau duomenų.

Bajeso paieškos metodų implementacijų Python programavimo kalboje yra ne viena. Įvairiuose ML konkursuose galime sutikti 2 populiariausias bibliotekas Hyperopt[22] ir Optuna[23]. Bibliotekos yra gan panašios tiek savo veikimo esme, tiek pritaikomumu, vis dėl to, darbe buvo pasirinkta naudoti Optuna, dėl geresnio rezultatų vizualizavimo ir paprastesnio paieškos lauko bei tikslo funkcijos apibrėžimo.

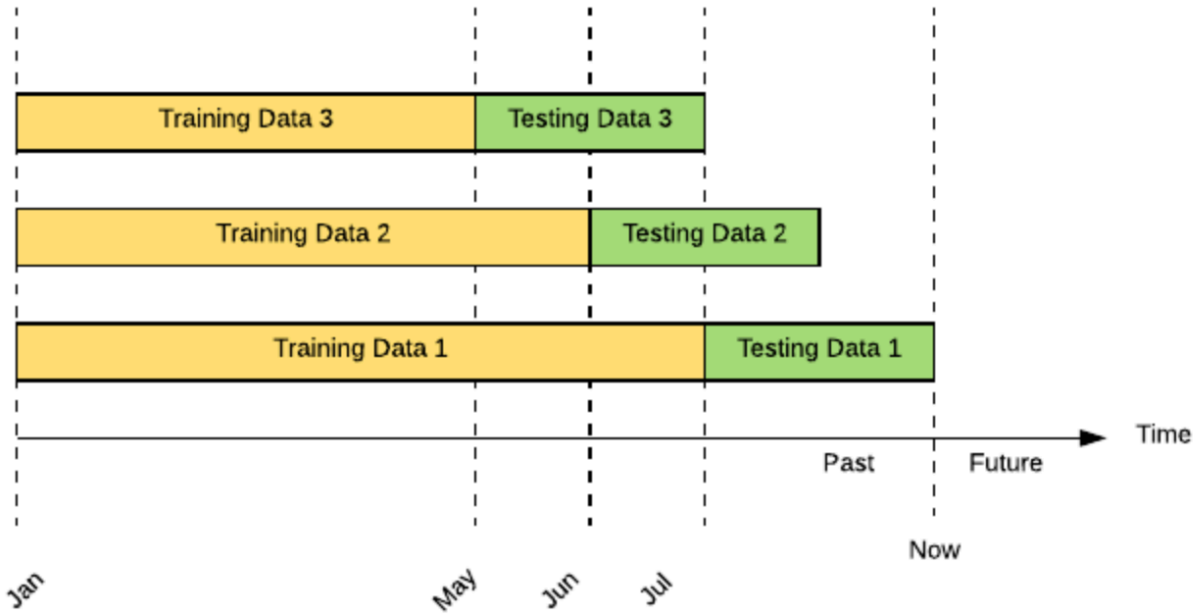
Optuna [39] yra automatinio hiperparametrų optimizavimo programinės įrangos sistema, specialiai sukurta mašininiam mokymuisi. Optuna ypatinga tuo, jog dinamiškai leidžia dinamiškai kurti parametrų paieškos lauką, atlieką parametrų paiešką ir jų genėjimą itin greitai dėl paralelizmo bei kitų techninių sprendimų ir itin lanksti architektūriniams sprendimas nuo paprastų iki komercinių.

## 2.11. Modelių patikra ir gerumo vertinimas

Modelio mokinimas ir jo kokybės vertinimas bus skaičiuojamas naudojant kryžminę patikrą. Ši kryžminė patikra bus ypatinga tuo, jog kiekvieno mėnesio turimi duomenys, bus naudojami prognozuoti sekančio mėnesio būstų kainas. Kitas svarbus dalykas, jog prie kiekvieno mėnesio bus pridedami visi prieš tai buvę mėnesiai nuo pat duomenų rinkimo pradžios – 2022 Sausio mėnesio. Ši besiplečiančio lango strategija (angl. *expanding window*) buvo priimta dėl mažos duomenų imties – 4354 įrašų, o sekančio mėnesio prognozavimas naudingas tuo jog leidžia netiesiogiai

[22] http://hyperopt.github.io/hyperopt/
[23] https://optuna.org/

atsižvelgti į makroekonominius kainos augimo veiksnius. Apie šį metodą sužinota ir pritaikyta naudojant Uber[24] įmonės tinklaraščio įrašą apie atgalinį testavimą[25] (angl. *backtesting*).




**9 pav.** Slankiojo lango atgalinio testavimo iliustracija

Šis Uber sprendimas kartu su slenkančio lango atgaliniu testavimu, kaip rašo straipsnyje, yra du pagrindiniai testavimo metodai, kuriais siekiama įvertinti prognozuojančių modelių kokybę. Vis dėl to, svarbu paminėti, jog šie metodai yra naudojami laiko eilutės uždaviniams, o ne regresijos, tačiau pati idėja leidžia juos taikyti ir regresijai. Taip pat straipsnyje minima, jog jie rėmėsi MAPE rezultatais vertindami modelio tikslumą. Mes savo darbe naudosime įvairias metrikas, tačiau daugiausia dėmesio skirsime RMSE ir MAPE.

[24] https://en.wikipedia.org/wiki/Uber
[25] https://eng.uber.com/backtesting-at-scale/

## 3. Mokslinis tyrimas

### 3.1. Teksto vektorizavimas

Teksto vektorizavimas buvo greitas ir paprastas procesas, kadangi Sent2Vec[26] autoriai yra sukūrę patogų būdą vektorizuoti tekstą – per terminalą. Aplamai, visas sprendimas yra patalpintas GitHub ir paruoštas kaip aplikacija, todėl naudojimas yra gan paprastas. Svarbiausi dalykai kuriuos reikėjo atlikti tai sutvarkyti tekstą, bei nuspręsti kokius vektorizavimo parametrus naudoti.

Prieš teksto vektorizavimą iš pradžių atliksime teksto tvarkymą:

- Iš teksto išimame el. paštus ir svetaines
- Pašaliname įvairius trumpinius susijusius su buto skelbimu: m2, eur, m (metai), kv (kvadratai), k (kambariai) ir keletą kitų;
- Kadangi dažnai yra vardijamas kambarių skaičius ar aukštų skaičius skaitine išraišką, skaitmenis 1-10 keičiame į žodines išraiškas (vienas, du, trys...);
- Taip pat tam tikrus trumpinius keičiame į pradines reikšmes: g. į gatvę, pr. į prospektą ir dar keletas kitų;
- Tuomet pašalinami visi kiti simboliai ir skaičiai nepriklausantys lotyniškoms raidėms išskyrus taškus, kadangi mums svarbu žinoti, kada baigiasi sakinys;
- Galiausiai visas tekstas paverčiamas tik mažosiomis raidėmis.

```
Parduodame 1k. buta Rinktines g. Zirmunuose.  Butas pilnai paruostas remontui. Yra galimybe sutarti remonto baigtuma. - sienos
dazymui; - lubos nudazytos; - yra supirkta dalis statybiniu medziagu (plyteles, laminatas ir t.t.); Yra galimybe sutarti
remonto baigtuma.  Buto isplanavimas: - Butas nekampinis, 9a is 9. - Bendras buto plotas - 32,73 kv.m, - Virtuve, erdvus
koridorius. - Vonia ir WC kartu. - Balkonas istiklintas. - Butui priklauso rusys. Irengimas: - Langai pakeisti, plastikiniai;
- Durys sarvo. - Naujas liftas. Aplinka: - Saugi, rami ir aplinka; - Namas toliau nuo gatves, todel nera triuksmo; - Kodine
laiptines spyna; - Tvarkinga laiptine; VIETA: - Netoli Kalvariju turgus, parduotuves Lidl, Maxima, IKI. Per 10min automobiliu
pasiekiami Prekybos centrai- PC Ozas, PC Akropolis. Iki visuomeninio transporto sustojimo 100m. - Prie pat yra darzelis,
mokykla. - Iki cmiesto centro 15 min pestute.  Skambinkite Jums patogiu metu, galima skambinti ir savaitgaliais, sutarsime
laika apziurai. Tel.
```

**10 pav.** Nesutvarkytas skelbimo tekstas

```
'parduodame k. buta rinktines gatve zirmunuose. butas pilnai paruostas remontui. yra galimybe sutarti remonto baigtuma. sienos
dazymui lubos nudazytos yra supirkta dalis statybiniu medziagu plyteles laminatas ir t.t. yra galimybe sutarti remonto
baigtuma. buto isplanavimas butas nekampinis a is . bendras buto plotas  m virtuve erdvus koridorius. vonia ir wc kartu.
balkonas istiklintas. butui priklauso rusys. irengimas langai pakeisti plastikiniai durys sarvo. naujas liftas. aplinka saugi
rami ir aplinka namas toliau nuo gatves todel nera triuksmo kodine laiptines spyna tvarkinga laiptine vieta netoli kalvariju
turgus parduotuves lidl maxima iki. per min automobiliu pasiekiami prekybos centrai pc ozas pc akropolis. iki visuomeninio
transporto sustojimo prie pat yra darzelis mokykla. iki cmiesto centro min pestute. skambinkite jums patogiu metu galima
skambinti ir savaitgaliais sutarsime laika apziurai.'
```

**11 pav.** Sutvarkytas skelbimo tekstas

Taip pat galime matyti jog butų aprašymuose dažnai yra akcentuojama susisiekimas su kitomis vietomis (darželiai, prekybos centrai, mieto centras). Taip pat dažnai minima pagrindinė informacija apie būtą: kambarių skaičius, plotas, statybos metai, šildymo tipas, ar butas yra parduodamas su buitine technika, ar butui priklauso stovėjimo vieta. Taip pat pastebima jog yra informacijos susijusios ir su buto kokybe: pakeista elektros instaliacija, dažytos sienos.

[26] https://github.com/epfml/sent2vec


**12 pav.** Skelbimų teksto žodžių žemėlapis

Sutvarkius turime 625821 sakinių duomenų rinkinį kurį praleidžiame per Sent2Vec modelį naudodami šiuos parametrus:

- minimalus žodžių pasikartojimų skaičius [minCount] = 8;
- rezultato dimensijos [dim] = 16;
- epochų skaičius [epoch] = 9;
- mokymosi dydis (angl. „learning rate“) [lr] = 0.2;
- maksimalus ngram žodžių kiekis [wordNGrams] = 2;
- nuostolio funkcija [loss] = ns;
- atrinktų negatyvų skaičius [neg] = 10;
- gijų skaičius naudojamas skaičiavimams kompiuteryje [thread] = 2;
- atrankos slenkstis (angl. „sampling threshold“) [t] = 0.000005;
- išmetamų ngramų skaičius mokinant modelį [dropoutk] = 4;
- minimalus Y(label) atvejų skaičius [minCountLabel] = 20;
- „hash bucket“ skaičius duomenų rinkiniui [bucket] = 4000000;
- maksimalus duomenų rinkinio dydis sakiniais [maxVocabSize] = 750000;
- tarpinių rezultatų išsaugojimo dažnumas [numCheckPoints] = 10.

Rezultate turime kiekvienam skelbimui asocijuojamą 16 narių dydžio vektorių, kuris bus naudojamas kartu su regresijos uždaviniu. Taip pat šalia pridedame papildomus požymius susijusius su tekstu, tokius kaip žodžių skaičius, sakinių skaičius, per skelbimą.

### 3.2. Paveikslėlių vektorizavimas

Su vaizdu buvo daugiau problemų, nei su tekstu. Pagrindinė to priežastis – didelis kiekis apdorojamos informacijos pavertus ją matrica. Kadangi turime ~69000 paveikslėlių, tačiau kiekvieną paveikslą transformuojame į 224,224 tuomet turime (~69000, 224, 224, 3) matricą. Tačiau mums nereikia visų paveikslėlių iš karto – mes galime užkrauti į laikinąją atmintį, pavyzdžiui, 1000 paveikslėlių, atlikti skaičiavimus ir baigus atlaisvinti atmintį ir užkrauti iš naujo

sekančius 1000. Vis dėl to, taip pat reikėtų atsižvelgti ir į tai, kaip patys paveikslėliai yra užkraunami – mums aktualu kad vyktų ir greitas jų užkrovimas į atmintį.

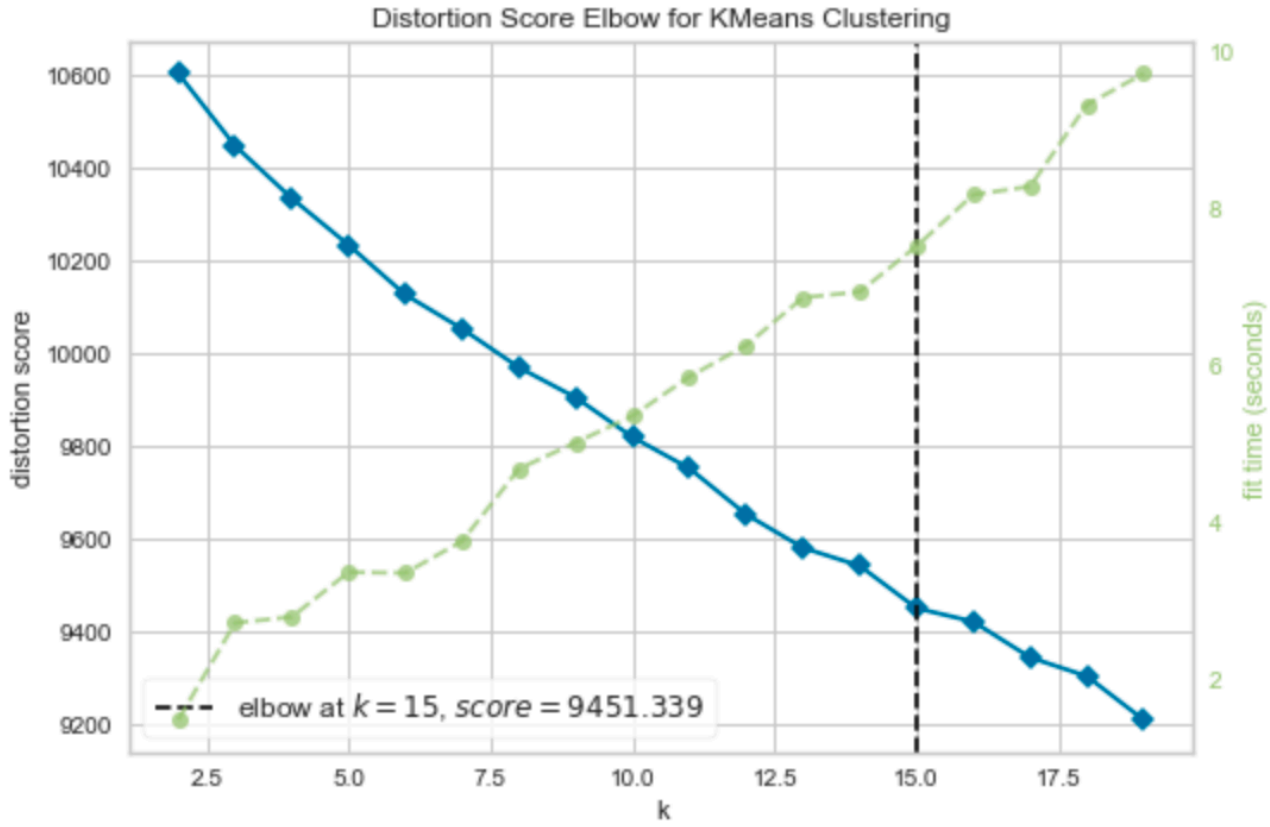
Dirbant su dideliais duomenų rinkiniais, optimaliausia failus naudoti dvejetaine forma. Dvejetainiai duomenys užima mažiau vietos diske, trumpiau kopijuojami ir juos galima daug efektyviau nuskaityti iš disko.

Mokinimasis truko 2 valandas ir 10 minučių. EfficientNetB0 mums kiekvienam paveikslėliui sugrąžina 1280 dydžio vektorių. Iš viso turime (~69000, 1280) dydžio matricą, kuriai toliau atliekame dimensijų mažinimą naudojant PCA. Pasiliekame tik x požymių paaiškinančių 95% informacijos ir liekame su 166 požymiais. Dimensijų mažinimas mums yra labai svarbus, kadangi toliau ieškosime panašių nuotraukų naudodami artimiausių kaimynų modelį, o šis modelis veikia gan ilgai su esant didelėms duomenų rinkinio dimensijoms. Su šia operacija siekiame suskirstyti panašius paveikslėlius į klasterius, tikėdamiesi jog tam tikrose grupėse nuotraukų yra esminė informacija leidžianti nustatyti kada butas iš tiesų ir pervertinamas ar nesuvertintas. Taip pat gautus vektorius dar normalizuojame.

Uždaviniui atlikti naudosime centroidų giminės modelį „KMeans“, kuris klasterizuoja kaimynus pagal atstumą nuo klasterio vidurkio. Šis metodas yra plačiai naudojamas dėl paprastumo, bei greito skaičiavimo. Šiam modeliui yra reikalingas vienas parametras – išankstinis klasterių skaičius. Šiam parametrui suskaičiuoti yra daug metodų [40], tačiau darbe panaudosime keletą, siekiant suderinti skirtingų metodų rezultatus bei įžvalgas apie nuotraukas iš skelbimų.

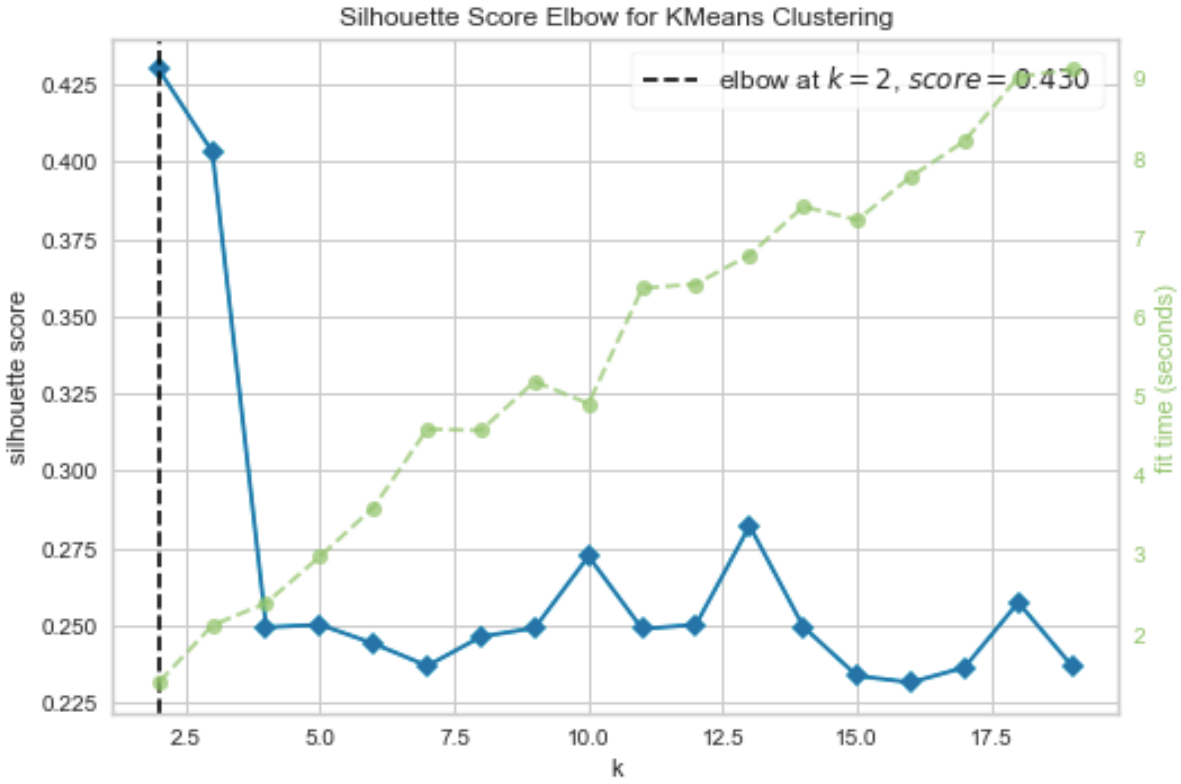
Pirmasis metodas kuris yra naudojamas – alkūnės. Šis metodas yra vienas populiariausių ir plačiausiai naudojamų metodų siekiant nustatyti klasterių skaičių. Šis metodas grįstas klasterių narių paklaidų kvadratais (WSS, angl. *Within-Cluster-Sum of Squared Errors*). Metodo idėja yra ta, kad paaiškinta variacija greitai keičiasi nedideliam skaičiui grupių, o vėliau sulėtėja, todėl kreivėje susidaro alkūnė. Alkūnės taškas yra klasterių skaičius, kuris optimaliausiai paaiškinta sklaidą klasterių viduje. Šiam metodui įgyvendinti ir vizualizuoti, naudosime „YellowBricks“[27] paketą, kurį taip pat naudosime ir visuose kituose grafikuose vizualizuoti. Matome, jog optimaliausias klasterių skaičius yra 15.

[27] https://www.scikit-yb.org/en/latest/



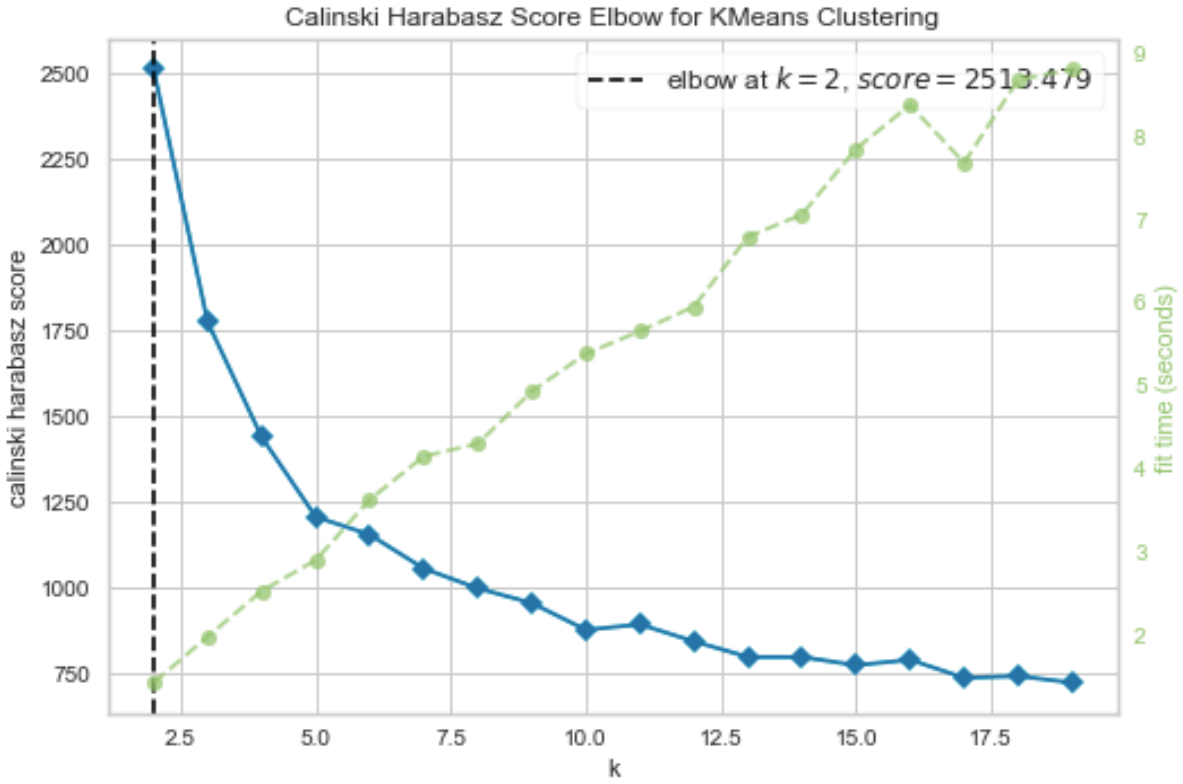

**13 pav.** Alkūnės metodo rezultatai paveikslėlių klasterizavimui

Praktikoje, dažnai alkūnės metodas atliekamas kartu su siluetų, kurį taip pat apskaičiuosime. Siluetų metodas remiasi idėja, jog tinkamam klasterių skaičiui esant, klasteriai yra gerai atsiskyrę vienas nuo kito, o klasterio nariai yra išsidėstę aplink klasterio centrą. Dėl šios priežasties, atstumas tarp klasterių narių yra mažesnis, nei atstumas iki narių esančių kitame klasteryje. Silueto koeficientas parodo ar atskiri taškai teisingai priskirti klasteriams ir turi 3 orientacines reikšmes: koeficientui esant artimam 0 – taškai vidutiniškai yra tarp kelių klasterių, kai koeficientas yra artimas (-1) – taškai geriau turėtų būti priskirti kitam, nei dabartiniam klasteriui, galiausiai kai turime koeficiento reikšmę artimą 1 – taškai yra gerai atsiskyrę ir tinkamuose klasteriuose. Atlikę silueto skaičiavimus, gauname jog optimaliausias klasterių skaičius yra 2, tačiau taip pat iš rezultatų matome, jog esant 15 klasterių skaičiui, kaip kad buvo paskaičiuota alkūnės metodu, silueto reikšmė yra labiau artimesnė 0, nei 1.




**14 pav.** Silueto koeficiento rezultatai paveikslėlių klasterizavimui

Taip pat išmėginsime Calisnki-Harabsz indeksą, kuris remiasi ta pačia idėja, kaip ir silueto koeficientas. Indeksas apskaičiuojamas atskirų objektų atstumų iki jų klasterio centro kvadratų sumų dispersiją padalijus iš atstumo tarp klasterio centrų kvadratų sumos. Kuo didesnė Calinski-Harabasz indekso vertė, tuo geresnis klasterizavimo modelis. Atlikę skaičiavimus taip pat gauname, jog optimalus klasterių skaičius turėtų būti 2.




**15 pav.** Calisnki-Harabsz indekso rezultatai paveikslėlių klasterizavimui

Nors 2 metodai sutaria, kad klasterių skaičius turėtų būti 2, jų turėtų būti daugiau, nes nuotraukų grupių yra daugiau nei 2: yra nuotraukos apie būsto išplanavimą, žemėlapiai (su atstumais iki centro ar kitų objektų), nuotraukos kiemo ar lauke esančios aplinkos, kambarių nuotraukos, kurios dar gali būti skirstomos ne tik pagal kambarius, tačiau pagal išstatymą (apstatyta baldais ar ne). Dėl šios priežasties, tikėdamiesi jog klasterizavimas vis dėl to sugebėjo apibendrinti šias grupes, naudosime 15 klasterių. Tuomet:

1. Kiekvienai nuotraukai priskiriame klasterį;
2. kiekvieną nuotrauką suspaudžiame į 5 dimensijų, siekdami neužgošti kitų duomenų vaizdo vektoriais;
3. kiekvienai nuotraukai priskiriame jos atitinkamą skelbimą;
4. kiekvieno skelbimo nuotraukas, priklausančias tam pačiam klasteriui, suvidurkiname;
5. papildomai kiekvienam skelbimui paskaičiuojame: kiek nuotraukų iš viso buvo rasta, kiek nuotraukų rasta kiekviename iš klasterių ir kokia procentinė išraiška nuo visų nuotraukų yra pasiskirsčiusi per klasterius.

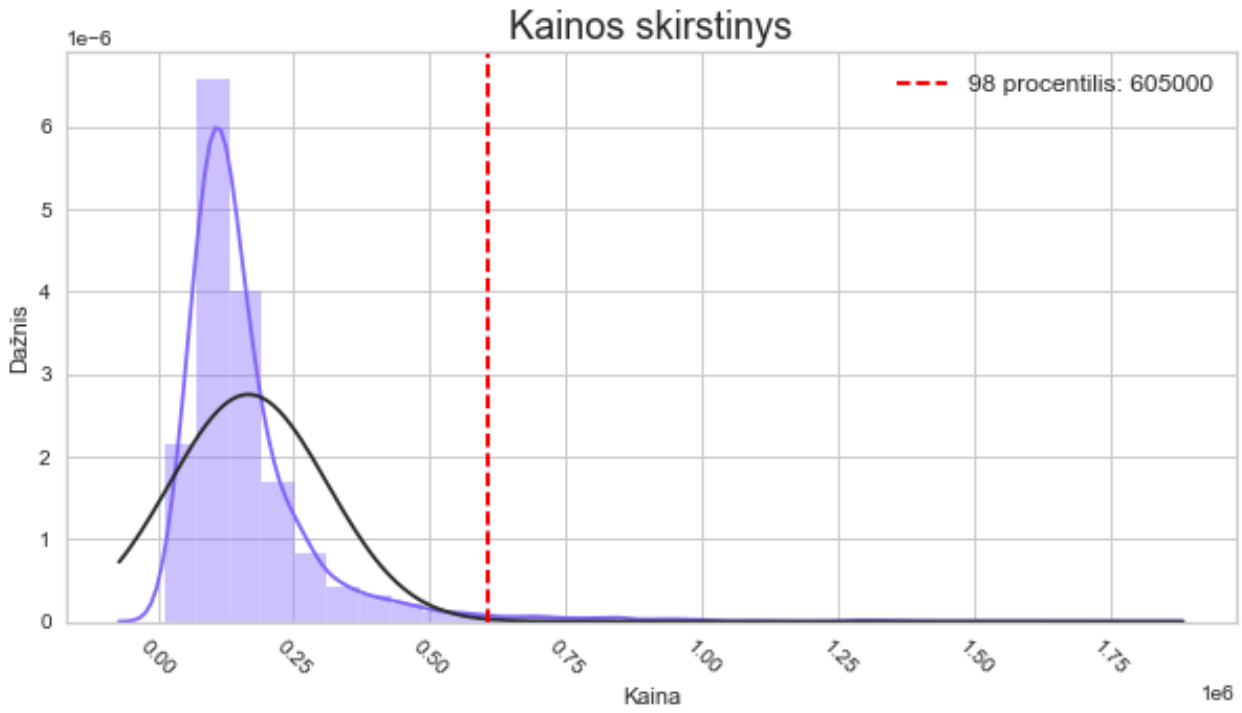
Rezultate turime vektorizuotus vaizdus su papildomais požymiais paruoštus regresijos uždaviniui spręsti.

### 3.3. Požymių inžinerija

Požymių inžinerija mašininio mokymosi kontekste yra apibrėžiamas kaip pats svarbiausias uždavinys, kuris užtrunka ilgiausiai, bei yra svarbiausias geriems rezultatams gauti. Šiame etape taip pat svarbus tampa nagrinėjamo dalyko supratingumas ir „ekspertinės žinios“ apie nagrinėjamo objekto ypatybes ir / ar procesus, todėl požymių inžinerija prasideda nuo duomenų apžvalgos ir

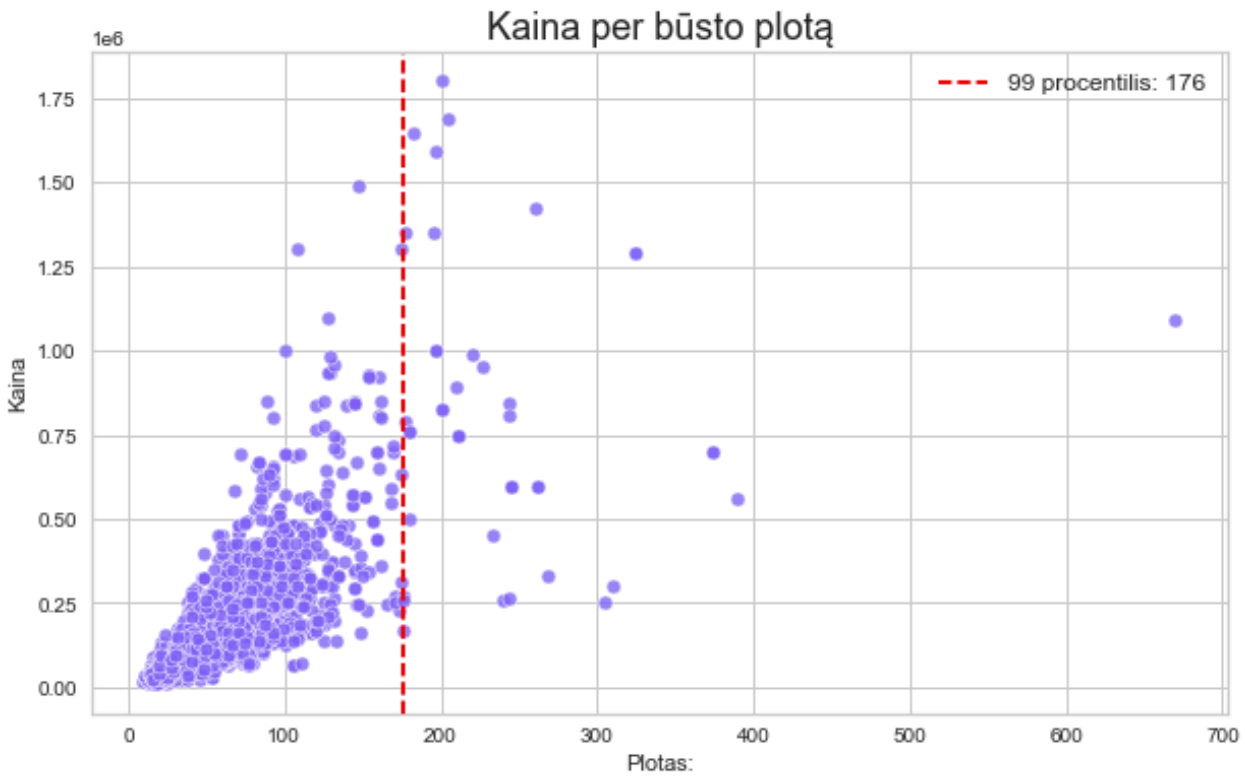
turimo duomenų rinkinio įsivertinimo. Tai leidžia įsivertinti kokius sprendimus mums gali tekti priimti atliekant modelių kūrimą ir kokias transformacijas mes galėtume išmėginti.

Apžvalgą pradedami su regresanto tyrimu. Galime matyti jog būsto kaina nėra normaliai pasiskirsčiusi ir turi uodegą dešinėje. To ir galime tikėtis, kadangi Vilniuje yra nemažai prabangos butų, kurie kainuoja virš rinkos vidurkio dėl tam tikrų specifinių ypatybių. Žvelgiant į grafiką taip pat matome, jog dėl jo ypatybių tiesiniams modeliams bus sunkiau atlikti tikslius skaičiavimus todėl kainą reikės lognormuoti. Taip pat kadangi yra nemažai itin prabangių būstų, kurių vertė siekia >0.75 mln. Eur. Todėl apskaičiavome 98 procentilį, kurį pasirinkome dėl itin ilgos uodegos, ir pašalinome reikšmes didesnes nei gautas rezultatas kuris yra 605 tūkst. Eur.



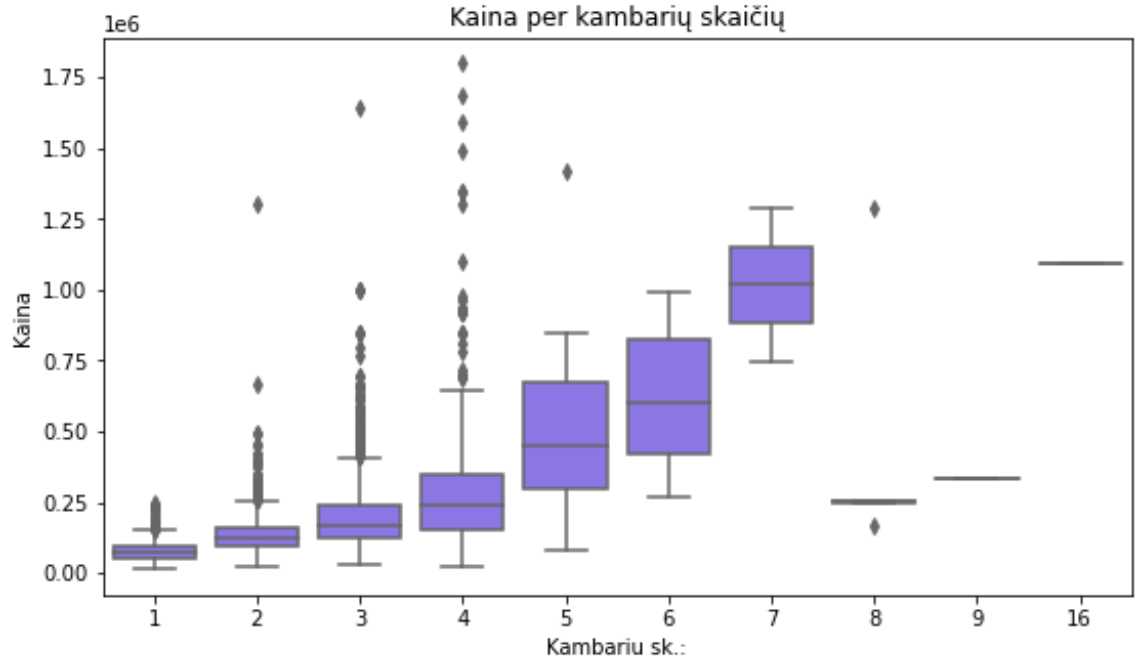

**16 pav.** Butų kainos skirstinys

Apžvelgdami kainą ir būsto plotą, taip pat matome išskirčių į kurias reikės atsižvelgti. Vidutinis būsto plotas yra 51 $m^2$, o 3 kvartilio reikšmė siekia – 67 $m^2$. Matome, jog yra nemažai butų kurie yra virš 200 $m^2$, todėl šias išskirtis taip pat reikės pašalinti. Apskaičiavome 99 procentilį ir pašalinome visus skelbimus kurių dydis virš 176 $m^2$.



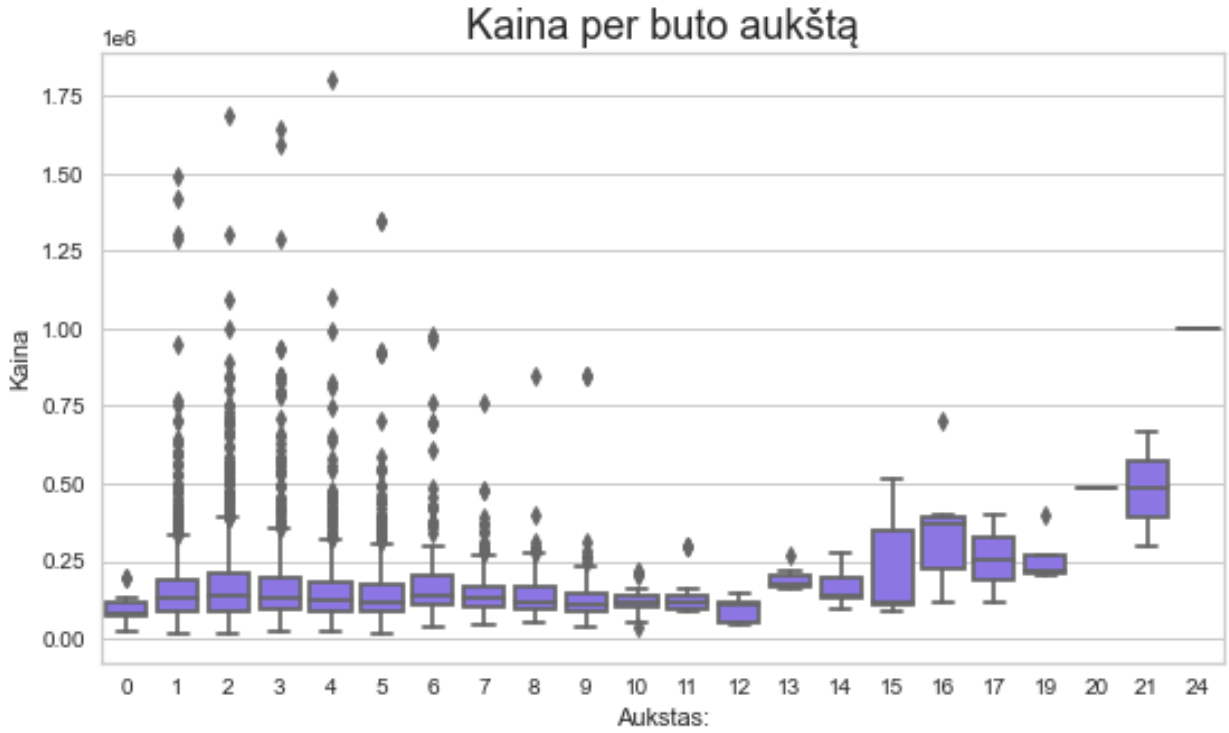

**17 pav.** Butų kaina per buto plotą

Lygindami kainą su kambarių skaičiumi, matome jog akivaizdžiai turime išskirčių ar blogų duomenų – yra įrašas su 16 kambarių! Tačiau taip pat matome jog daugiau 7 kambarių yra vienetai skelbimų. Tačiau kadangi filtruosime objektus kurių kaina didesnė nei 600 tūkst., išfiltruosime ir butus turinčius 7 ir daugiau kambarių. Taip pat matome, jog populiariausi dydžiai yra 1-4 kambariai, čia taip pat matome daugiausia įvairovės kainos dydyje, kuri šiek tiek pasikoreguos atfiltravus itin prabangius butus.



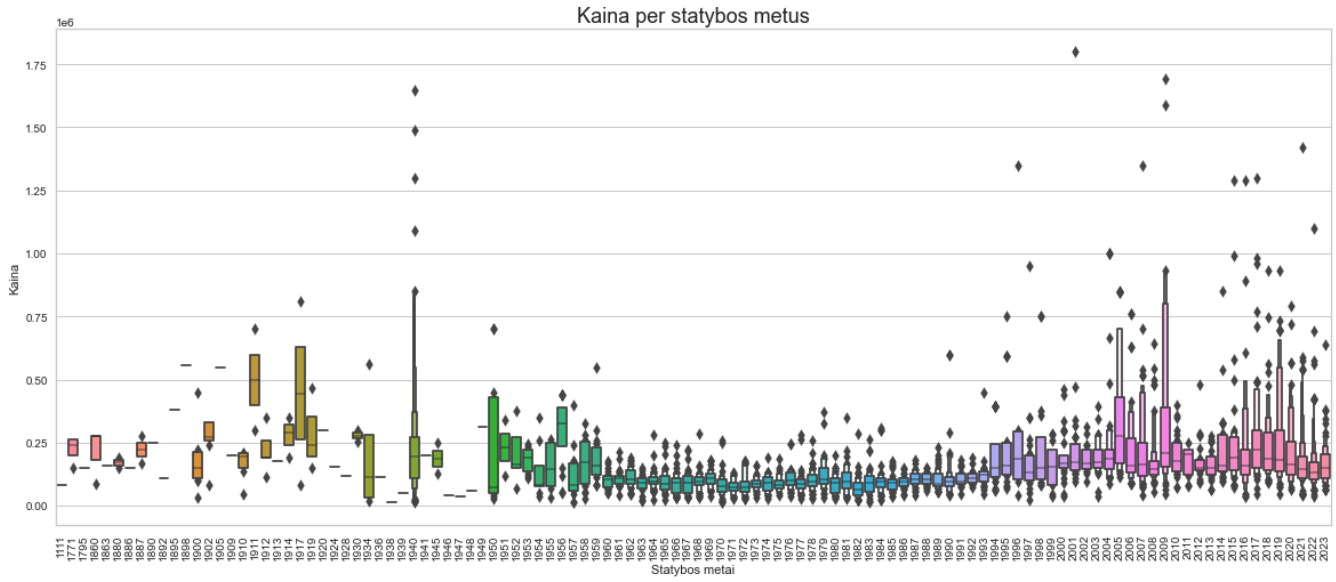
**18 pav.** Butų kaina per kambarių skaičių

Žvelgiant į kainą bei aukštą, kuriame yra parduodamas būstas, galime matyti keletą dalykų. Visų pirma, sulig 13 aukštu būsto vertė tampa šiek tiek aukštesnė, lyginant su kitais aukštais. Tai yra natūralu dėl dviejų priežasčių. Visų pirma tokiuose butuose atsiveria nuostabi Vilniaus panorama, o suderinus šią su atitinkama lokacija „ant delno“ gali matytis senamiestis ir visa sostinės „širdis“. Kita priežastis yra ta, jog dangoraižių ir /ar itin aukštų daugiabučių bet kur mieste statyti negalima, todėl vietovės su itin aukštais daugiabučiais turėtų būti patogioje lokacijoje pridedančioje vertės būstui.



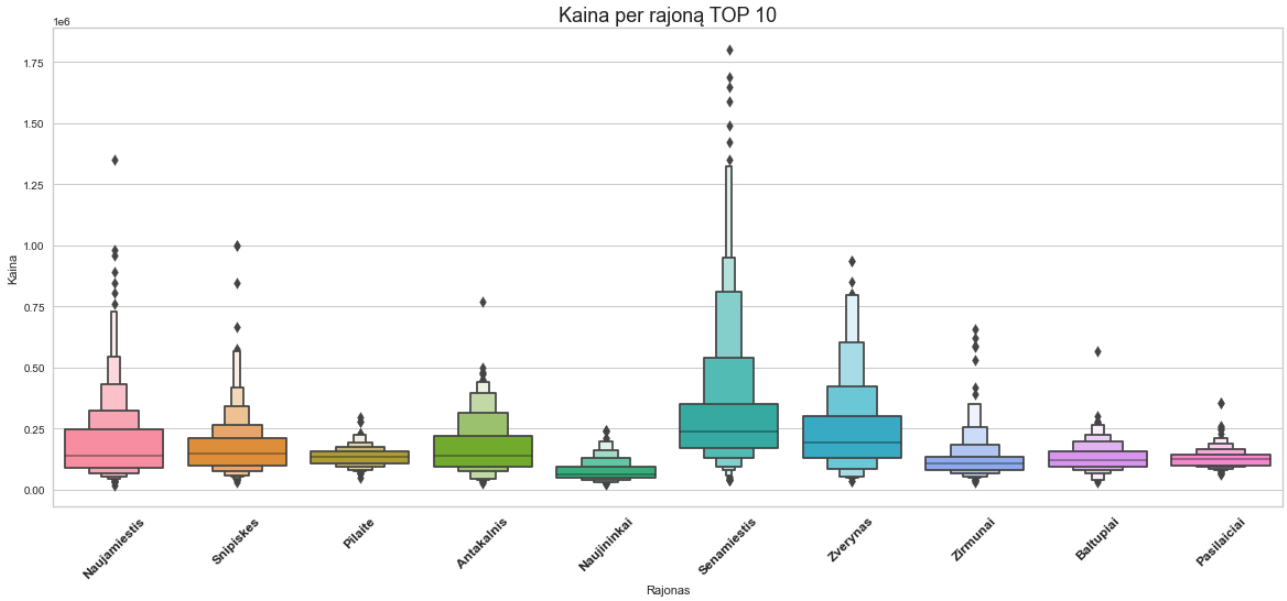

**19 pav.** Butų kaina per buto aukštą

Lyginant statybos metus su kaina, visų pirma galime pastebėti jog yra daug įvairovės statybos metuose. Seniausias parduodamas būstas yra 1771 metų statybos, taip pat yra ir nemažai projektų kurie dar nėra pastatyti, tačiau yra numatomi 2023. Iš šio grafiko galime matyti keletą dalykų. Pirma, būstai iki 1959 metų yra brangesni nei 1960-1993 metų statybos. Tikėtina to priežastis, jog daugiau senesnių būstų yra centrinėje Vilniaus dalyje, kuri taip pat yra ir brangesnė. Daug būstų sovietmečiu buvo statyta Lazdynuose ir kituose miegamuosiuose rajonuose, kurie yra toliau nuo centro ir todėl labiau pigesni. Nuo 1994 statyti daugiabučiai vėl pradeda kilti kaina, o 2005 ir 2009 statyti daugiabučiai turi daug kainos įvairovės. Taip pat svarbu paminėti, jog ateityje „pastatyti“ butai matosi jog yra pigesni, tačiau tai yra todėl, jog jie parduodami, dažnai, neįrengti ir iš statytojų, tačiau įrengus ir pardavus būstą ši kaina tampa didesnė.



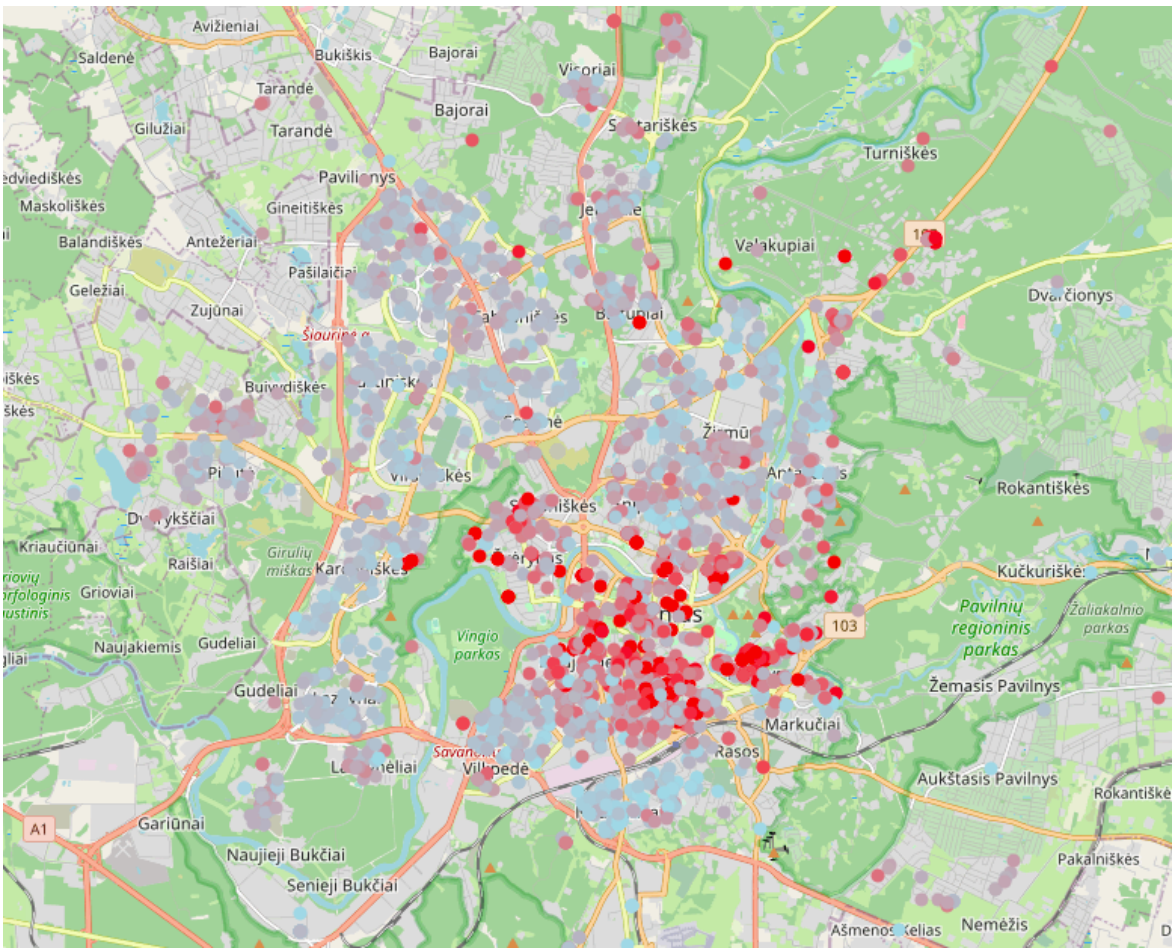

**20 pav.** Butų kaina per statybos metus

Taip pat svarbu apžvelgti ir vietovės įtaką būsto kaina, tačiau iš statybos ir kainos ryšio jau galime matyti, kad vietovė būsto kainai yra itin svarbi, todėl nenuostabu jog daugiausia kainos įvairovės yra būtent Senamiestyje. Tačiau taip pat geri rajonai yra Žvėrynas, Naujamiestis ir Antakalnis, kurie priklausomai nuo vietos mikrorajone taip pat yra itin patogioje vietoje ir geru susiekimu. Galime matyti, jog Naujininkai yra pigiausias iš 10 populiariausių mikrorajonų, tačiau šiame rajone vyksta daug naujos statybos projektų, o rajono vardas „gerėja“ gyventojų akyse, juolab kad tai gan gera susisiekimą turintis rajonas, todėl ateityje jo vertė tikrai gali kilti.



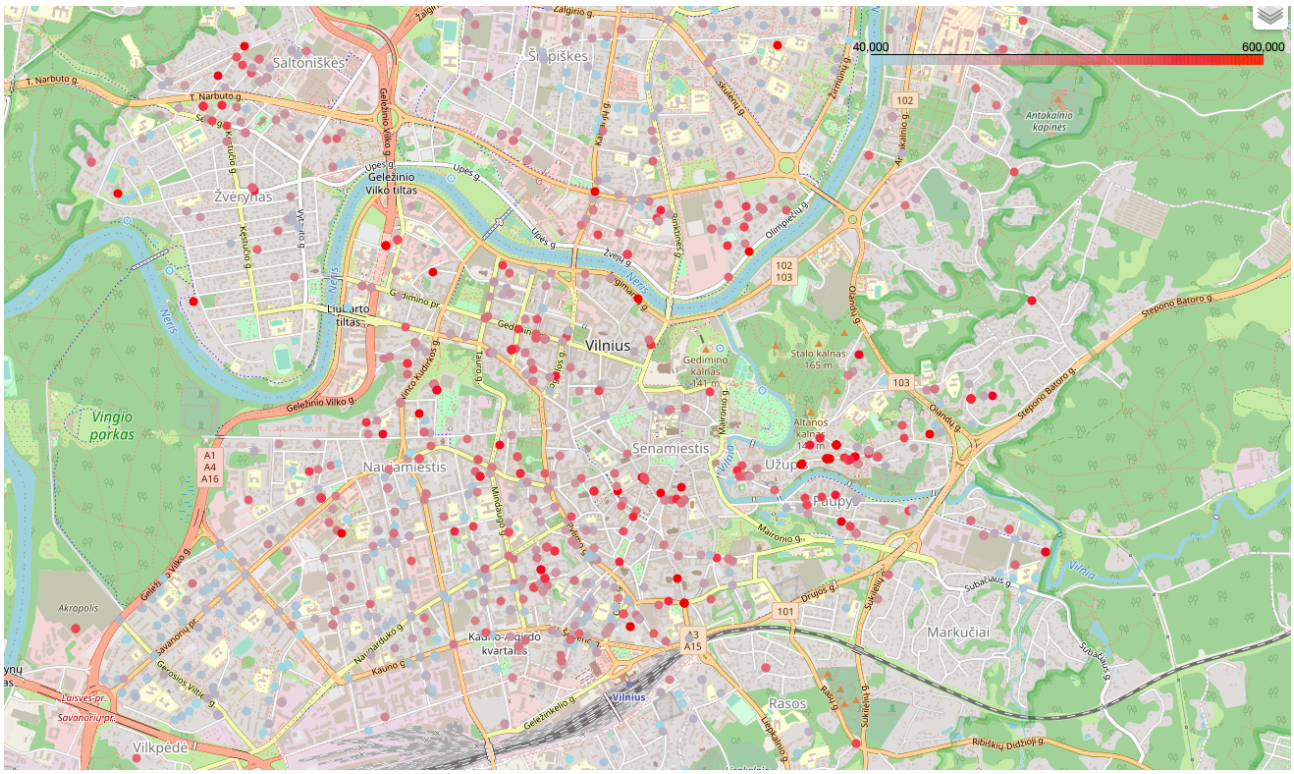

**21 pav.** Butų kaina per TOP 10 rajonų

Tikslios vietovės atveju, matome jog iš tiesų daugiausia itin brangių būstų yra Senamiestyje, Naujamiestyje, Žvėryne, Antakalnio pradžioje, Užupyje / Paupyje ir Markučiuose. Vis dėl to, matome ir tolimesniuose rajonuose brangių būstų, kur kai kurie matome jog yra arti miškų ir / ar parkų.



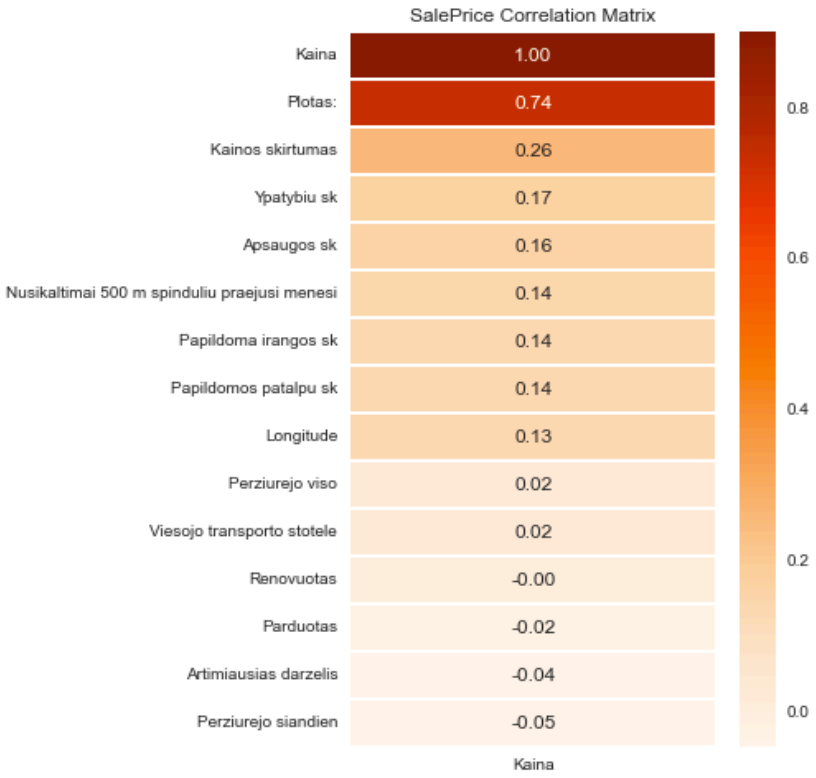

**22 pav.** Butų kainos žemėlapyje - Vilniaus miestas

Pažvelgus iš arčiau galime matyti ir įsivaizduoti kaip atrodo Vilniaus miesto centras, kuriame būsto kaina yra aukščiausia.



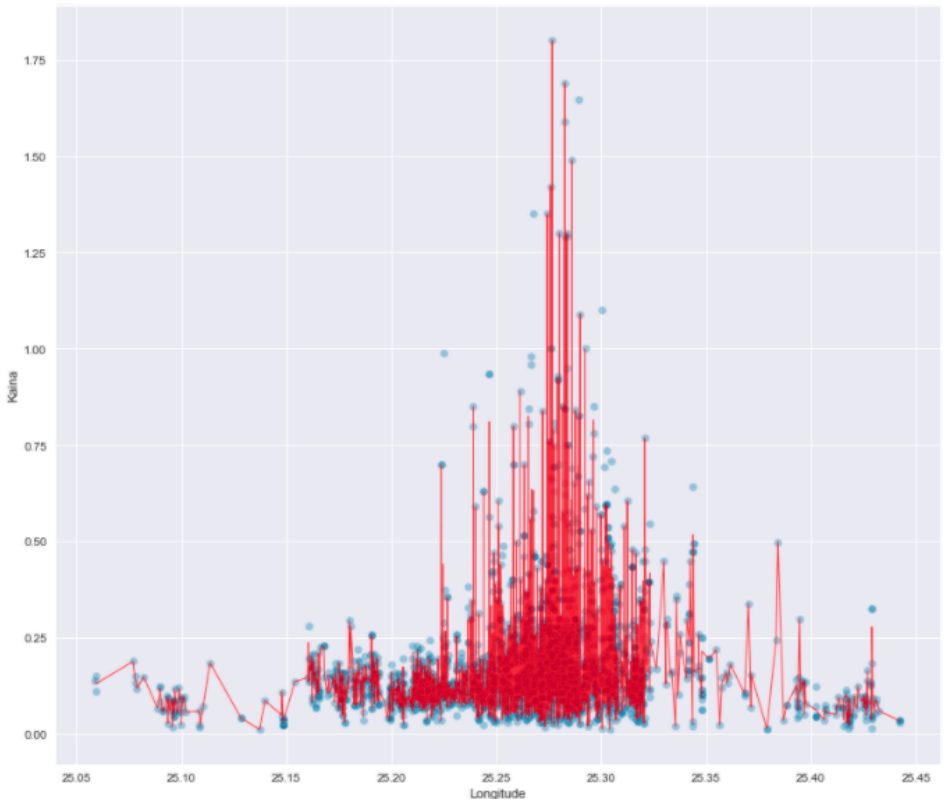

**23 pav.** Butų kainos žemėlapyje - centrinė Vilniaus miesto dalis

Žvelgiant į žemėlapį matome jog šis centras tikrai kelia būsto kainą, tačiau jei paimtume TOP 15 požymių koreliuojančių su kaina, matytume jog tik ilguma turi tiesiška priklausomybę (nors ir labai silpną) su kaina. Plokštuma turi neigiamą 0.06 koreliaciją. Taip pat iš šio grafiko galime matyti jog tiesiniams modeliams gali sunkiau sektis, jei netransformuosime duomenų, nes be būsto ploto, nėra pastebimos tiesinės priklausomybės.




**24 pav.** TOP 15 koreliuojantys požymiai su kaina

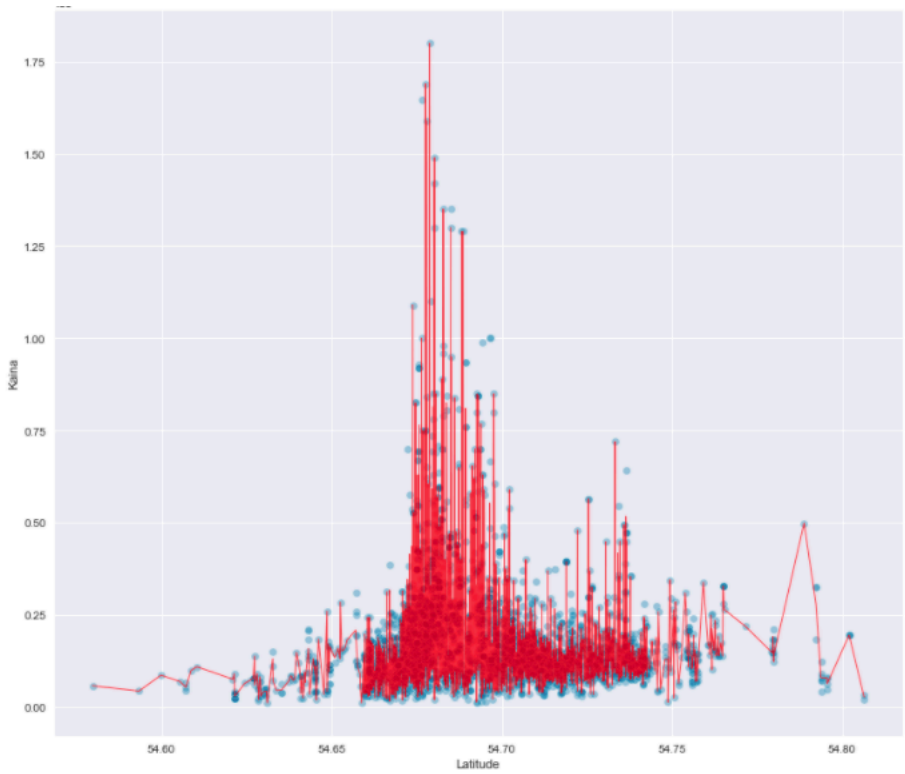
Vis dėl to, ryšys tarp ilgumos ir platumos su kaina tikrai yra ir jį mes išgausime per požymių inžineriją. Visų pirma, jei paimtume ilgumą kartu su kaina, bei ant viršaus uždėtume vidutinę būsto kainą tame taške (raudona linija), tuomet pastebėtume jog tarp 25.23 – 25.32 ilgumos yra vidutiniškai daugiau brangesnių būstų, nei kitur. Tai žinodami, mes galime patikrinti šioje atkarpoje įvairias ilgumos reikšmes bei joms paskaičiuoti skirtumą su kiekvieno skelbimo ilguma. Tokiu būdu turėtume rasti „centro“ ilgumą, o kuo skirtumas tarp šios ir skelbimo ilgumos bus mažesnis – tuo kaina turėtų būti didesnė.




**25 pav.** Ilgumos ir kainos ryšys

Atlikę skaičiavimus randame, jog „centro“ ilgumą turėtų būti 25.29, o paskaičiavus skirtumą tarp šios reikšmės ir kiekvieno skelbimo, gauname koreliaciją lygią -0.24. Matome, jog informacija apie centrą tikrai svarbi kainai, todėl tą patį procesą pakartojame ir su platuma. Platumoje, vidutiniškai kainos didesnės yra tarp 54.66 – 54.73, todėl šioje atkarpoje ieškosime „centro“. Iteraciniu būdu atlikus skaičiavimus, čia koordinatę randame lygią 54.679, o skirtumo ir kainos koreliacija yra -0.22. Taigi, tiek „centro“ platumos, tiek ilgumos skirtumai su būstų skelbimais yra reikšmingesni, nei pačios parduodamo būsto koordinatės. Tačiau jei naudotume abu šiuos požymius – koreliacija tampa dar didesnė ir pradeda siekti -0.3. Visa tai reiškia, jog kuo atstumas iki „centro“ yra maženis, tuo kaina yra didesnė, nes būstas priklauso centrui. Todėl šią informaciją įtrauksime kaip požymį duomenų rinkinyje, bei atstumą iki centro paskaičiuosime naudodami Euklido atstumą[28].

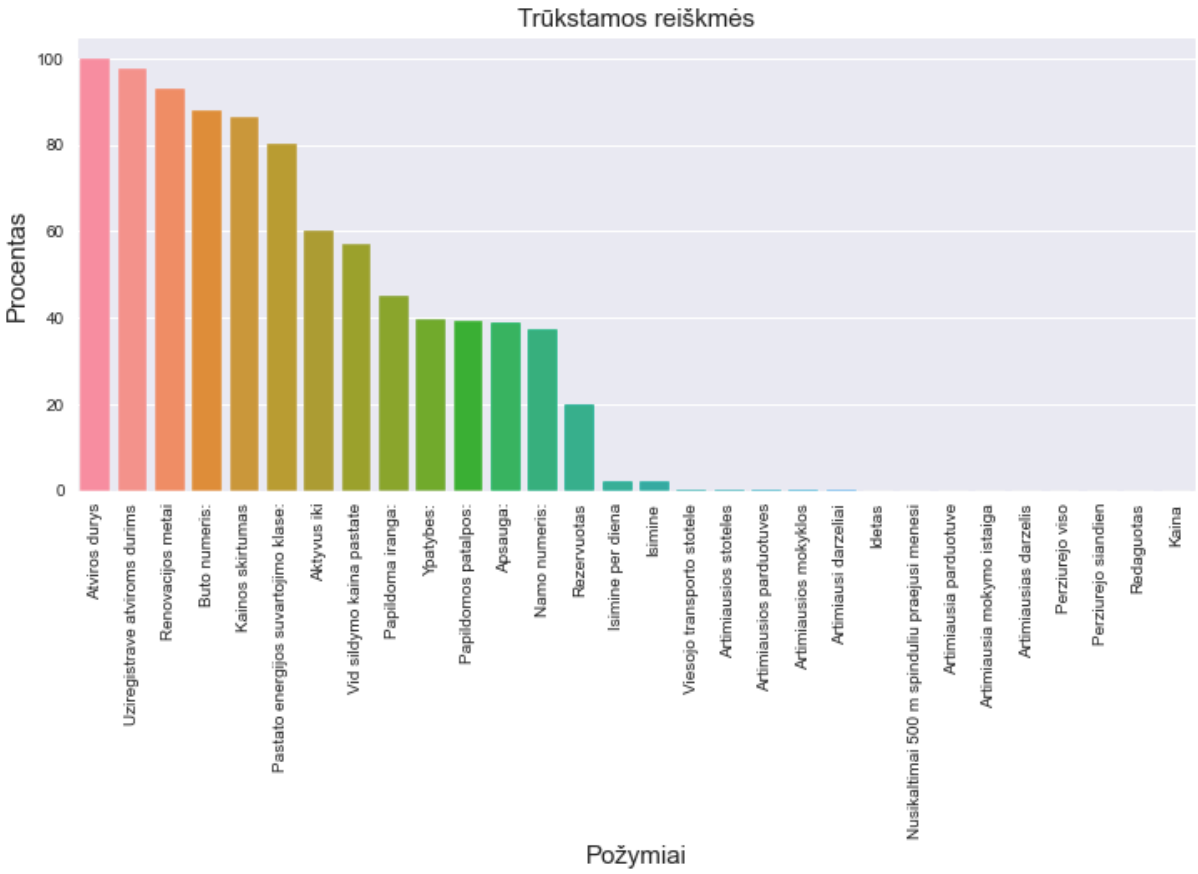
[28] https://en.wikipedia.org/wiki/Euclidean_distance




**26 pav.** Platumos ir kainos ryšys

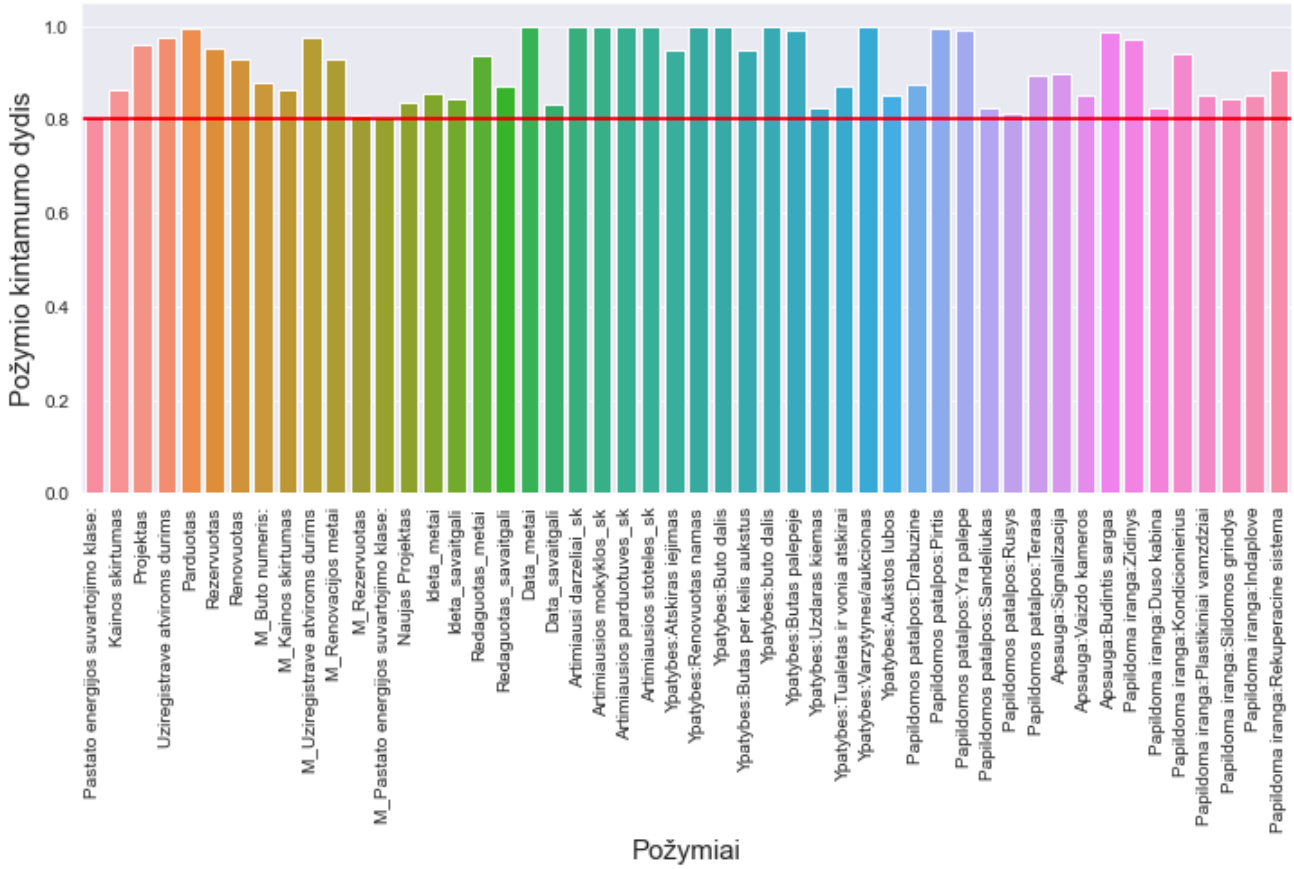
Sekančiu žingsniu tvarkome trūkstamas reikšmes. Turimame duomenų rinkinyje, yra požymių kurie turi daug trūkstamų reikšmių. Pavyzdžiui, požymis „atviros durys“, panašu jog buvo blogai paruoštas ir nors skelbime šios informacijos kartais yra, tačiau ji nebuvo surinkta, tai galima matyti iš sekančio požymio „uzsiregistrave atviroms durims“ – jei atvirų durų nebūtų, nebūtų ir jos užsiregistravusių asmenų. „Renovacijos metai“ atrodo kaip itin prasminga ir naudinga informacija apie būstą, kadangi žinome, jog ji pakelia būsto kaina, vis dėl to, iš 4354 skelbimų tik 7%, tačiau jei renovacijos nėra – galime ją užkoduoti kaip statybos metus. Panašiai ir su „kainos skirtumu“ – tai informacija parodanti kaip kito parduodamo būsto kaina, jei pardavėjas sugalvojo ją padidinti ar sumažinti. Ši informacija taip pat atrodo prasminga, tačiau tik 14% skelbimų ją turi, o visais kitais atvejais laikysime, kad kaina nesikeitė ir buvo ta pati – skirtumas buvo 0. Tokie požymiai kaip „papildoma įranga“, „apsauga“, „ypatybės“ ir „papildomos patalpos“, neturėjo daug tuščių reikšmių todėl visas rastas kodavome kaip „nėra“. Šie požymiai nusako buto ypatybes tokias kaip rakinama laiptinė, šarvo durys, ar yra „tamsus kambariukas“, parkavimo vietą ar įrengti atitinkama įranga kuri paliekama bute, pavyzdžiui, kondicionierius. Taip pat užkoduojame ir kitus panašius požymius.

Taip pat buvo keletas požymių susijusių su lokacija: artimiausi darželiai, stotelė, parduotuvė ir mokykla. Kadangi tuščių reikšmių buvo labai nedaug, jos buvo užpildytos naudojant kaimynus. Jei informacijos trūko, ji buvo paimama iš kitų skelbimų kurie yra toje pačioje gatvėje, paimant patį dažniausią rezultatą. Jeigu gatvės lygiu informacijos nėra, tuomet yra naudojamas rajono lygmuo. Tokiu būdu mes rankiniu būdu atlikome trūkstamų reikšmių užpildymą, kaip kad naudodami artimiausių kaimynų techniką.




**27 pav.** Trūkstamų reikšmių dydis duomenyse

Sutvarkę trūkstamas reikšmes toliau kūrėme naujus požymius, turinčius prasmę ir galinčius padėti mašininiam mokymuisi lengviau spręsti regresijos uždavinį. Visų pirma paskaičiavime akivaizdesnius požymius, tokius kaip plotą per kambarį, ar butas yra pirmutiniame ar paskutiniame aukšte, santykį tarp esamo aukšto ir nuo visų galimų, praėjęs dienų skaičius tarp įdėto skelbimo ir duomenų surinkimo. Taip pat iš visų datos požymių buvo ištraukti metai, mėnuo, savaitė, diena ir ar tai yra savaitgalis. Taip pat atrinkome TOP 30 populiariausių darželių, mokyklų, parduotuvių ir stotelių ir kiekvienam skelbimui, kuris nepapuola tarp populiariausių, jo artimiausias darželis ar kitas objektas yra keičiamas į „Kita“. Tas pats yra atliekame su gatvių pavadinimais. Daug skirtingų kategorinių kintamųjų reikšmių nėra naudingos, kadangi jos stipriai išplečia duomenų rinkinį ir jį „užkemša“ mažai informacijos turimais požymiais. Todėl remiantis maksimaliu požymių skaičiumi naudojamu medžio algoritme (šis yra 32) kadangi daugiausia jų naudosime, buvo pasirinkta naudoti TOP 30, nes 31-a reikšmė bus “Kita”. Kita vertus, kadangi sukūrėme ir šiek tiek naujų kategorinių kintamųjų, ne visi jie yra prasmingi, jeigu daugiau nei 80% reikšmių yra tos pačios, todėl visą duomenų rinkinį patikriname ir pašaliname visus tokius požymius jei vienodų reikšmių yra daugiau nei 80%.




**28 pav.** Kategorinių kintamųjų įvairovės dydis duomenų imtyje

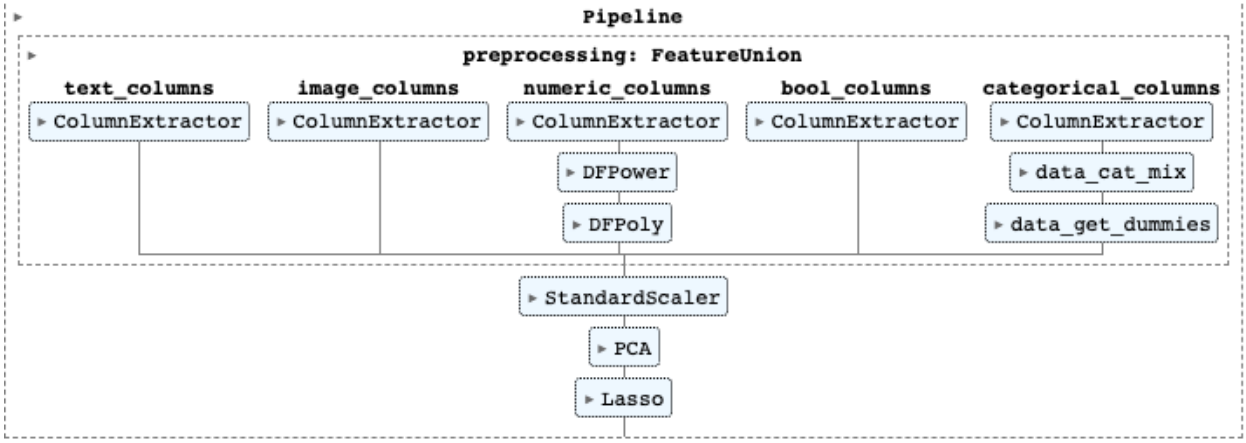
Galiausiai po teksto valymo ir požymių inžinerijos, prie galutinių duomenų pridedame vektorizuotą tekstą ir vaizdą. Šio žingsnio pabaigoje liekame su: 4354 stebėjimais ir 209 požymiais, iš kurių 29 yra kategoriniai.

### 3.4. Grandinės kūrimas

Praeitame žingsnyje mes atlikome požymių inžinerijos, tačiau atlikome ne viską, ką galime dar galime padaryti. Pavyzdžiui, vienas iš dalykų yra kategoriniai kintamieji – bet kuris modelis kurį naudotume nesupranta pačių kategorinių kintamųjų ir tam kad juos galėtume naudoti mes juos turime atitinkamai transformuoti, kad šie būtų suprantami modeliui. Dažniausiai naudojama transformacijos technika yra naudojant fiktyvius kintamuosius, kuomet kiekviena kategorinio kintamojo reikšmė įgauna dvinarę reikšmę 0 arba 1, jei konkrečiame stebėjime pasitaiko atitinkama transformuoto kategorinio kintamojo reikšmė. Šią transformaciją altiekame žingsniu „data_get_dummies“. Kita mažiau populiari, tačiau naudinga transformacija yra kategorinių kintamųjų vektorizacija (angl. „*categorical embedding*“), kuri šiame darbe nėra bandoma, tačiau ateityje gali būti panaudota kaip alternatyva.

Papildomai apie kiekvieną kategorinį kintamąjį ištrauksime jo dažnio charakteristiką ir išsaugome kaip papildomus požymius toliau naudojamus modelyje. Šis žingsnis yra išsaugotas pavadinimu „data_cat_mix“. Taip pat visoje grandinėje mes neliečiame 3 tipo duomenų: vektorizuoto teksto, vektorizuotų paveikslėlių ir dvinarių kintamųjų, kurie buvo sukurti požymių inžinerijos metu ir turinčių reikšmes 0 / 1. Galiausiai mums lieka tolydieji kintamieji, kuriems yra atliekamos 2 transformacijos. Pirmoji „DFPower“ yra transformacija skirta transformuoti visus tolydžiuosius

kintamuosius į turinčius Gauso skirstinį, tam kad tiesiniams modeliams palengvintų skaičiavimus. Transformacijai atlikti yra naudojamas „Yeo-Johnson“ [41] metodas, kaip alternatyva „Box-Cox“ transformacijai. Šis transformacijos būdas buvo pasirinktas todėl, jog jis leidžia kintamiesiems turėti neigiamas ar lygias nuliui reikšmes, kai tuo tarpu „Box-Cox“ to daryti neleidžia. Galiausiai, po transformacijos visi kintamieji yra transformuojami naudojant antros eiles polinominę transformacija - [1, a, b, a^2, ab, b^2]. Šioje transformacijoje yra panaudojami visi tolydieji kintamieji taip išplečiant požymių skaičių. Ši transformacija padeda ML modeliams geriau apčiuopti netiesinius ryšius ir palengvinti jų aptikimą. Žinoma, medžiams tai nėra problema, nes jie gali „atsipjauti“ įvairius ryšius. Antros eilės polinomai yra patogus sprendimas, kadangi tai įtraukia ir kintamųjų tarpusavio sandaugas ir jų kvadratus. Vis dėl to, daugiau transformacijų nėra atliekama, siekiant supaprastinti modelio architektūrą. Galiausiai, atskirų žingsnių rezultatai tekstui, vaizdui, dvinariams, tolydiems ir kategoriniams kintamiesiems yra sujungiami „preprocessing“ žingsniu, o galutiniai rezultatai yra standartizuojami. Vėliau pritaikoma PCA išlaikant 95% informacijos ir pritaikomas tiesinius modelius „atstovaujantis“ Lasso modelis. Svarbu paminėti, jog PCA nebūtinai bus pritaikytas, nes tolimesniame žingsnyje mes tikrinsime ar dimensijų mažinimas iš tiesų padeda modeliui geriau prognozuoti, taip pat patikrinsime ir kitas transformacijas.




**29 pav.** Tiesinių modelių grandinės struktūra

Siekiant patikrinti ar vaizduojama grandinės struktūra yra optimali, iteraciniu būdu tikrinsime skaičiuojant kiek kiekviena transformacija pagerinta modelio rezultatus. Skaičiavimams naudosime 4 dalių kryžminę patikrą, paremtą RMSE rezultatais naudodami tinklelio paiešką (angl. *grid-search*). Iš pradžių patikrinsime išorines grandinės dalis, o vėliau – vidines. Darbo kontekste išorinės grandinės dalys yra PCA, standartizavimas, bei teksto, paveikslėlių, dvinarių, kategorinių ir tolydžiųjų kintamųjų naudojimas. Vidinės grandinės dalys darbo kontekste yra transformacijos atliekamos konkrečiai tolydiesiems ir kategoriniams kintamiesiems. Šis atskyrimas naudingas tuo, jog jei paaiškėtų, kad kategoriniai kintamieji tik blogina modelio rezultatus – galėtume paimti visas jiems atliekamas transformacijas ir patikrinti ar kažkuri transformacija esanti viduje viską gadina ar vis dėl to patys požymiai nesuteikia naudingos informacijos modeliui. Išorinė paieška truko 8 minutes (1 Priedas). Toliau pateikiami pirmi 7 rezultatai.

Rezultate gauname, jog visų pirma PCA su 95% nepagerina modelio rezultatų, todėl žinome jog dimensijų mažinimui naudosime kitus metodus. Taip pat matome, jog nėra itin akivaizdaus skirtumo tarp teksto, paveikslėlių ar dvinarių naudojimo modelyje – RMSE skiriasi tik 200. Vis dėl to, priimamas sprendimas naudoti visus turimus duomenis ir vėliau naudojant požymių svarbos

atrinkimo metodus sumažinti požymių skaičių iki paties svarbiausių. Galiausiai prieiname prie normavimo metodų, kur buvo naudoti 4 variantai: standartizavimas „*StandardScaler*“ (vidurkis 0), normalizavimas „*MinMaxScaler*“ (reikšmės tarp 0 ir 1), standartizavimas atsparus išskirtimis naudojant „*RobustScaler*“ (tas pats standartizavimo procesas tik vietoj vidurkio yra naudojama mediana, o vietoj standartinio nuokrypio – naudojamas atitinkamas kvartilių diapazonas) arba jokio „*passthrough*“. Matome, jog tarp geriausių rezultatų nėra tik standartizavimo, vis dėl to, rezultatai atrodo gan panašus taikant likusius variantus. Kadangi nusprendėme naudoti visus turimus duomenis, pasirinksime ir metodą, su geriausiu rezultatu, kuris šiuo atveju yra paprastas normalizavimas.

**13 lentelė.** Tiesinių modelių grandinės išorinės struktūros tinklelio paieškos rezultatai

| RMSE | PCA | USE TEXT | USE IMAGE | USE BOOL | SCALER |
|---|---|---|---|---|---|
| 38704 | No | No | No | Yes | MinMaxScaler |
| 38771 | No | Yes | Yes | Yes | MinMaxScaler |
| 38817 | No | Yes | Yes | No | MinMaxScaler |
| 38874 | No | No | No | Yes | passthrough |
| 38899 | No | No | No | Yes | RobustScaler |
| 38945 | No | Yes | Yes | Yes | passthrough |
| 38983 | No | Yes | Yes | Yes | RobustScaler |

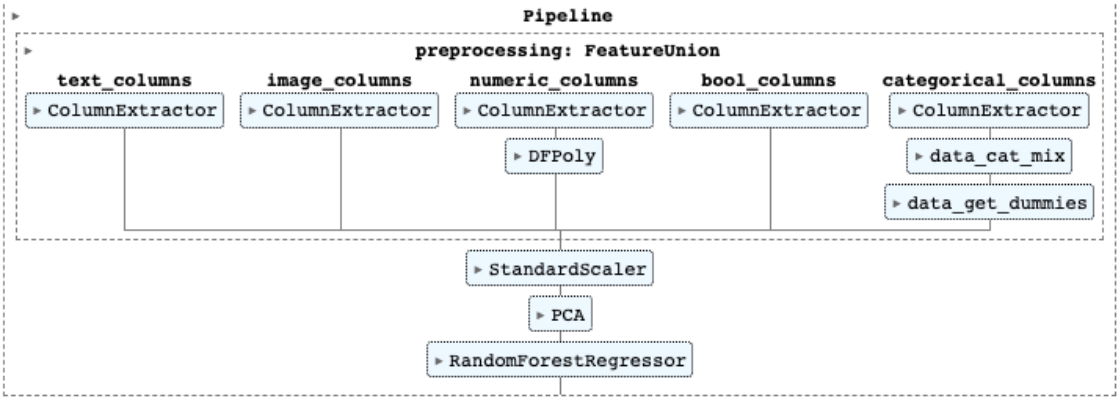
Toliau patikriname vidines grandinės dalis, pritaikius iš aukščiau gautus geriausius rezultatus. Paieška truko 1 minutę. Čia galime pastebėti šiek daug skirtumų, nei aukščiau. Visų pirma, matome jog polinominė transformacija padeda Lasso modeliui „įveikti“ netiesines priklausomybes tarp kintamųjų. Taip pat pastebime jog tolydžiųjų kintamųjų transformavimas jog šie taptų normaliai pasiskirstę neteikia pastebimos naudos, tačiau to priežastis yra ta, jog pats regresantas vis dar neturi Gauso skirstinio, todėl šį transformavus rezultatas turėtų pagerėti. Galiausiai matome jog iš kategorinių kintamųjų dažnių informacijos ištraukimas taip pat nedaro pastebimos įtakos. Kadangi dar darysime svarbių požymių atranką, pasiliksime daugiau požymių todėl liekame su visomis transformacijomis, kurias tikrinome.

**14 lentelė.** Tiesinių modelių grandinės vidinės struktūros tinklelio paieškos rezultatai

| RMSE | TRANSFORM CATEGORICALS | USE POLYNOMIALS | USE POWER TRANSFORM |
|---|---|---|---|
| 38704 | No | Yes | Yes |
| 38771 | Yes | Yes | Yes |
| 39941 | No | Yes | No |
| 40005 | Yes | Yes | No |
| 42154 | No | No | No |
| 42201 | Yes | No | No |
| 43562 | No | No | Yes |
| 43591 | Yes | No | Yes |

Toliau tikriname medžiams skirtą grandinę. Priešingai nuo ankstesnės, vienintelis skirtumas yra tas jog neįtrauksime tolydžiųjų skaičių transformacijos siekiant juos paversti Gauso skirstinį turinčius.

Medžiams yra nesvarbu ar kintamieji yra normaliai pasiskirstę ar ne, todėl neapsukinsime papildomai architektūros. Taip pat medžius „atstovaus“ Atsitiktiniai miškai. Kaip ir su Lasso modeliu, šiame žingsnyje hiperparametrų optimizavimo neįtrauksime.




**30 pav.** Medžių modelių grandinės struktūra

Atlikę paiešką, kuri truko 65 minutes gauname gan panašius rezultatus (2 priedas). Atvaizdavus keletą geriausių rezultatų galime matyti, jog PCA neteikia naudos. Taip pat pastebima, nepanašu jog yra didelis skirtumas tarp to ar yra naudojami tekstiniai, vaizdiniai ar dvinariai požymiai, todėl kaip ir su tiesinių modelių grandine – šiuos žingsnius mes paliksime ir vėliau atrinksime, kurie iš jų yra svarbiausi. Vienintelis pastebimas skirtumas jog čia duomenų normalizavimo metodas nėra svarbus ir to priežastis yra tai kaip yra skaičiuojami medžiai. Vis dėl to, pasiliksime išskirtims atsparų standartizavimą, tam kad turėtume vienodas dimensijas ir palengvintume skaičiavimus.

**15 lentelė.** Medžių modelių grandinės išorinės struktūros tinklelio paieškos rezultatai

| RMSE | PCA | USE TEXT | USE IMAGE | USE BOOL | SCALER |
|---|---|---|---|---|---|
| 38292 | No | No | No | Yes | StandardScaler |
| 38550 | No | No | No | Yes | passthrough |
| 38584 | No | No | No | Yes | MinMaxScaler |
| 38694 | No | No | No | Yes | RobustScaler |
| 38706 | No | Yes | Yes | Yes | RobustScaler |
| 38790 | No | Yes | Yes | Yes | MinMaxScaler |
| 38968 | No | Yes | Yes | No | StandardScaler |

Taip pat patikriname ir keletą viduje atliekamų transformacijų, kurios truko 7 minutes. Gauname jog akivaizdaus skirtumo tarp transformacijų nėra. Kadangi medžiai sugeba „pagauti“ netiesines priklausomybes – polinominės transformacijos neturi tokios įtakos rezultatams kaip tiesiniams modeliams. Tačiau, kadangi atliksime svarbių požymių atranką, pasiliksime su šiomis transformacijos.

**16 lentelė.** Medžių modelių grandinės vidinės struktūros tinklelio paieškos rezultatai

| RMSE | TRANSFORM CATEGORICALS | USE POLYNOMIALS |
|---|---|---|
| 38778 | No | Yes |
| 39043 | Yes | Yes |
| 39184 | No | No |
| 39375 | Yes | No |

### 3.5. Svarbių požymių atranka

Kaip buvo minėta antroje dalyje, svarbių požymių atrankai naudosime Boruta biblioteką. Pritaikius šį metodą, iš 1230 požymių buvo atrinkti 61 iš kurių 55 buvo priimti, o 6 dar iki galo nepatvirtinti per 100 iteracijas. Atfiltravus rezultatus, buvo sudarytas galutinis medžiams skirtas vamzdis. Jis buvo panaudotas su XGBoost modeliu naudojant gamyklinius parametrais. Pirmasis bandymas buvo su visais požymiais, o antrasis – su Boruta atrinktais. Rezultate matome, jog visi rezultatai pagerėjo, o svarbiausia, skaičiavimas pagreitėjo 4 kartais! Kita vertus, atranka užtruko ~102 minutes

| Model_name | R2 Score | MAE | MAPE | MSE | RMSE | Time | PARAMS |
|---|---|---|---|---|---|---|---|
| XGBoost | 0.74 | 27,269.57 | 19.54 | 1,804,639,753.32 | 42,481.05 | 23.68 | None |
| XGBoost | 0.76 | 26,416.55 | 18.73 | 1,639,986,870.72 | 40,496.75 | 5.55 | None |

**31 pav.** Xgboost modelio rezultatai prieš ir po požymių atrankos

Panašus rezultatas pastebimas ir su kitais medžių modeliais – modelių rezultatai tapo nežymiai geresni, o skaičiavimo laikas tapo kelis kartus greitesnis. Greitesnis skaičiavimo laikas tapo itin svarbiu veiksniu optimizuojant tokius modelius kaip Catboost, kurie su daugybe požymių regresiją apskaičiuoja itin lėtai esant ne gamykliniams nustatymams. Vis dėl to, dėl galimai egzistuoja tokia modelio parametrų kombinacija, kuriai naudingiau bus turėti visus požymius, nei tik svarbiausius, todėl Boruta atrinkus požymius tikrinsime kartu su parametrais ir įvertinsime, ar verta taikyti požymių atranką ar ne.

Boruta buvo paleista naudojant atsitiktinių miškų algoritmą su medžiams sukurta grandine. Baigus, buvo atrinkti šie požymiai:

- teksto 16-tas vektorius;
- plotas;
- platuma;
- atstumas iki centro;
- skaičius skelbimą įsiminusių žmonių * atstumas iki artimiausios mokymo įstaigos;
- skaičius skelbimą įsiminusių žmonių * aukštų skaičius daugiabutyje;
- skaičius skelbimą įsiminusių žmonių * atstumas iki centro;
- skaičius skelbimą įsiminusių žmonių * atstumas iki Vingio parko;
- atstumas iki artimiausio darželio * plotas;
- atstumas iki artimiausio darželio * papildoma įrangos skaičius;
- atstumas iki artimiausio darželio * (esamas aukštas/iš viso aukštų);
- atstumas iki artimiausios mokymo įstaigos * skaičius skelbimą įsiminusių žmonių;
- atstumas iki artimiausios mokymo įstaigos * atstumas iki centro;

- atstumas iki artimiausios parduotuvės * plotas;
- atstumas iki artimiausios parduotuvės * papildoma įrangos skaičius;
- nusikaltimų skaičius 500 m spinduliu praėjusi mėnesi * atstumas iki centro;
- plotas^2;
- plotas * kambarių skaičius;
- plotas * platuma;
- plotas * ilguma;
- plotas * papildomos įrangos skaičius;
- plotas * apsaugų skaičius;
- plotas * atstumas iki centro;
- plotas * plotas per kambarį;
- plotas * papildomų ypatybių skaičius;
- kambarių skaičius * buto aukštas;
- kambarių skaičius * platuma;
- kambarių skaičius * ilguma;
- kambarių skaičius * atstumas iki centro;
- kambarių skaičius * plotas per kambarį;
- buto aukštas * papildoma įrangos skaičius;
- buto aukštas * atstumas iki centro;
- aukštų skaičius daugiabutyje * atstumas iki centro;
- platuma^2;
- platuma * ilguma;
- platuma * atstumas iki centro;
- ilguma * apsaugos skaičius;
- ilguma * atstumas iki centro;
- skelbimo peržiūrų skaičius * skelbimo peržiūrų skaičius informacijos rinkimo dieną;
- skelbimo peržiūrų skaičius * vidutinis skelbimą įsiminusių žmonių skaičius per dieną;
- skelbimo peržiūrų skaičius * atstumas iki centro;
- skelbimo peržiūrų skaičius informacijos rinkimo dieną * atstumas iki centro;
- papildoma įrangos skaičius * atstumas iki Vingio parko;
- apsaugų skaičius * plotas per kambarį;
- vidutinis skelbimą įsiminusių žmonių skaičius per dieną * atstumas iki centro;
- atstumas iki centro^2;
- atstumas iki centro * atstumas iki Vingio parko;
- atstumas iki centro * plotas per kambarį;
- atstumas iki centro  * (esamas aukštas/iš viso aukštų);
- šildymas: centrinis;
- šildymas: centrinis kolektorinis;
- rajonas: Senamiestis;
- šildymo tipo dažnio charakteristika;
- statybos metų dažnio charakteristika;
- renovacijos metų dažnio charakteristika.

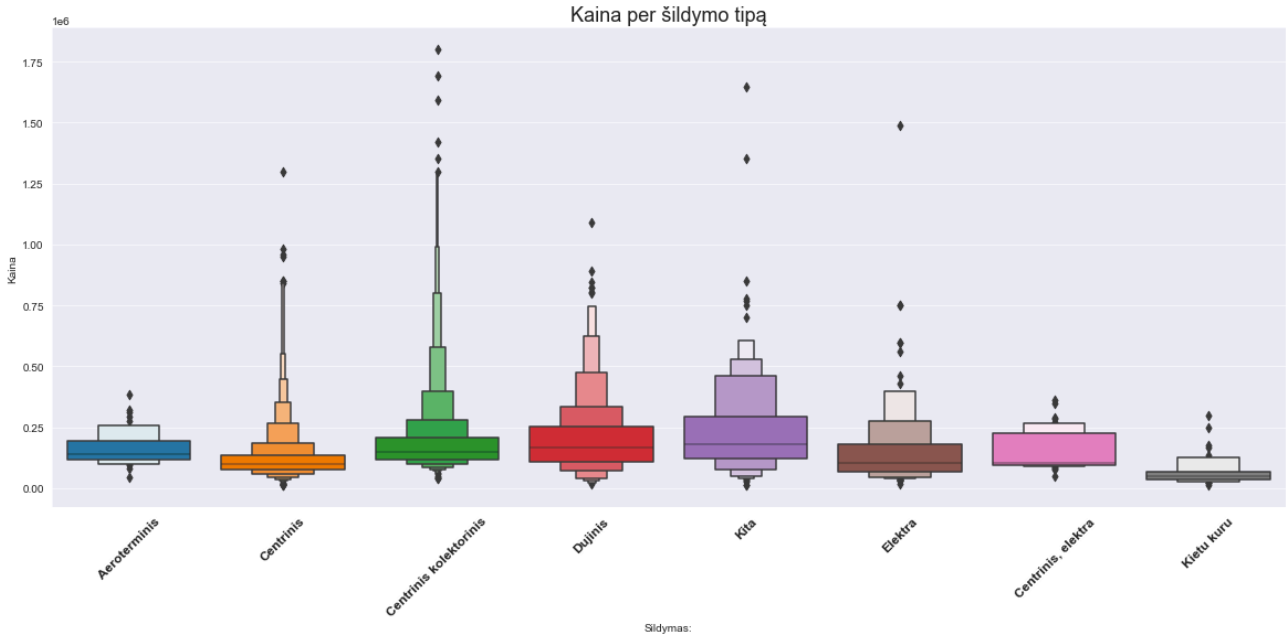
Taip pat buvo atrinkti 6 požymiai dėl kurių Boruta nebuvo tikra:

- teksto 2-tas vektorius;

- teksto 4-tas vektorius;
- teksto 10-tas vektorius;
- skelbimą įsiminusių žmonių skaičius * atstumas iki artimiausio darželio;
- plotas * atstumas iki Vingio parko;
- ilguma * papildoma įrangos skaičius.

Iš atrinktų požymių visų pirma galime pastebėti jog dauguma jų visų pirma tolydieji kintamieji ir taip pat dauguma jų yra kombinacijos tarpusavyje. Tačiau nors pradinių kintamųjų buvo nemažai, galime pastebėti jog yra keli pagrindiniai požymiai besikartojantys per kombinacijas: plotas, platuma ir ilguma, atstumas iki centro, artimiausias atstumas iki darželio ir mokymo įstaigos, skelbimo įsiminimų ir peržiūrų skaičius. Iš to galime galvoti jog šiuo metu daugiausia įtakojantys veiksniai butams Vilniuje yra buto dydis tiek ploto, tiek kambarių prasme, atstumas iki centro, bei geras susisiekimas su mokyklomis ir darželiais. Iš šių požymių galima galvoti, jog vilniečiai nori geros lokacijos auginti vaikams, bei kad centre esančios pramogos ir / ar darbovietės būtų ranka pasiekiamos ir už tai jie yra pasiryžę mokėti daugiau. Kita vertus aplink miesto centrą ir jame kuriasi vis daugiau verslo centrų, o didėjantis Vilniaus rajono gyventojų skaičius verčia gyventojus trauktis į miesto miegamuosius rajonus ir periferijas. Dėl šių tendencijų natūralu jog geroje vietovėje esantys būstai turi didelę paklausą, o todėl ir didesnę kainą. Taip pat galime pastebėti jog dalis prieš tai minėtų požymių – plotas, ilguma ir plokštuma, taip pat yra pakliuvę atskirai, bei pakelti kvadratu. Tai parodo jog šie požymiai itin svarbūs regresijos uždaviniui.

Iš kategorinių kintamųjų, matome jog atrinkta buvo nedaug. Jau žinome, kad atstumas iki centro / senamiesčio yra svarbus, todėl nenuostabu jog būtų skelbimai senamiestyje yra reikšmingas kainai. Taip pat papuolė ir informacija apie šildymą. Šildymo tipas šiek nustebino, kadangi iš vienos pusės kolektorinis šildymas yra patogus, nes leidžia kontroliuoti šildymo kiekį, ir, žinoma, mokamų mokesčių už šildymą kainą. Tačiau palyginus kainą per šildymo tipą, akivaizdaus skirtumo nėra, juolab kad ~70% visų skelbimų turi centrinį arba centrinį kolektorinį šildymą, todėl šiems tipams matome daugiausia kainų įvairovės. Kadangi kolektorinis šildymas būna įrengtas naujesnės statybos arba renovuotuose daugiabučiuose – mediana yra didesnė kolektoriniui šildymui. Nepaisant to, matome ir gan populiarų dujinį šildymą, kuris yra pigesnis nei kolektorinis. Iš to galime manyti, kad papildoma informacija apie šildymo tipą modeliui naudingai padeda identifikuoti naujesnius ar renovuotus butus, asocijuojant šiuolaikiškesnes šildymo technikas su aukštesne kaina. Tai dar kartą parodo tai, jog tarp kitų svarbių požymių papuolė informacija apie statybos metus ir renovacijos metus dažnio forma.




**32 pav.** Butų kaina per šildymo tipą

Dveji darbo tikslai yra panaudoti tekstinius ir vaizdinius duomenis apie būstą siekiant pagerinti kainos prognozavimo kokybę. Šie du tikslai remiasi idėja, jog šiuose duomenų tipuose slepiasi informacija leidžianti geriau prognozuoti itin pigius ir itin brangius būstus. Vis dėl to, galime matyti jog tik tekstas tapo svarbus, tuo tarpu niekas iš vaizdinės informacijos nebuvo atrinkta. Nors ir ne visi teksto vektoriai buvo atrinkti, tačiau 4 iš 16 buvo svarbūs. Nors ir nežinome kas konkrečiai slepiasi šiuose vektoriuose, galime tik manyti, jog kadangi tekste dažnai būdavo rašoma informacija apie atstumus iki darželių, mokyklų, rotušės ir katedros ar Aušros vartų, bei bendra informacija apie butą. Galime manyti, jog ta pati informaciją, kurią apžvelgėme aukščiau, susijusi su vietove, susisiekimu su mokymų įstaigomis ir centru, buto dydžiu ir statybos metais – taip pat galėjo būti atrinkta, tačiau „suspausta“ į vektorinę formą.

Nors vaizdinė informacija nebuvo reikšminga požymių atrankoje, tai dar nereiškia jog ji nereikšminga aplamai. Vaizduose yra daug naudingos informacijos, tačiau tą informaciją pasiekti yra daug įvairių būdų. Šiame darbe mes neskyrėme daug dėmesio tinkamiausios architektūros / sprendimo paieškai siekiant ištraukti prasmingą informaciją iš nuotraukų. Darbe buvo naudotas itin paprastas sprendimas, kuris iš gautų rezultatų pasirodė esąs per daug bendrinis. Todėl tolimesniuose darbuose būtų pravartu išmėginti kitas vaizdų apdorojimo technikas bei išgautos informacijos segmentavimo technikas.

Galiausiai, kaip alternatyva Boruta atrinktiems rezultatams taip pat reikšmingiems požymiams atrinkti buvo naudotas Lasso modelis. Šis modelis tai dažnai praktikoje taikomas būdas atrinkti reikšmingus požymius tiesiniams modeliams. Vis dėl to, šis modelis prastas tuo, jog jis vertina tik regresanto ir regresoriaus ryšį, neatsižvelgdamas į ryšius tarp regresorių, kurie netiesiogiai įtakoja regresantą. Tačiau, šį metodą vis vien naudosime. Jo paskirtis bus lyginamoji – turėdami požymius atrinktus su Lasso modeliu ir Boruta, galėsime palyginti kurie požymiai pasiekia geresnius rezultatus. Taip pat patikrinsime ir atvejus, kada yra naudojami visi požymiai.

Lasso požymių atranka remiasi tiesinių modelių koeficientų ir požymių sandauga:

$$\hat{y} = f(\vec{w} \cdot \vec{x}) = f(\sum_i w_i x_i)$$

Po reguliarizacijos, nereikšmingų požymių koeficientai tampa lygūs 0, todėl atrinkus visus požymius, kurių koeficientai nėra lygus nuliui, mes galime atlikti svarbių požymių atranką. Jeigu esame požymius standartizavę, visų požymių koeficientai tampa vienoje skalėje, kas reiškia jog kuo didesnis koeficientas – tuo daugiau įtakos daro regresantui. Todėl atliekant požymių svarbą su Lasso iš pradžių buvo atlikta optimalaus *alpha* parametro paieška, o tuomet atrinkti požymiai, kurių įtaka yra didesnė nei 50000. Šis dydis buvo parinktas atsitiktinai siekiant gauti 100-200 požymių. Taip buvo atrinkti 181 požymiai. Šie požymiai bus naudojami tiesinių modelių grandinėje kuriant pradinius modelius.

Taigi, šiame etape mes atlikome dvi požymių atrankas: Boruta ir Lasso. Boruta atrinkti požymiai bus naudojami medžių grandinėję, tuo tarpu Lasso – tiesinių modelių. Vis dėl to, atliekant hiperparametrų paiešką patikrinsime ar tikrai Boruta atrinkti požymiai pagerina rezultatus medžiams ir ar Lasso iš tiesų padeda tiesiniams modeliams. Taip pat patikrinsime kiek iš tikro požymių atranka pagerina rezultatus.

### 3.6. Hiperparametrų rinkimas

Nors turime atsirinkę svarbiausius požymius – tai dar negarantuoja aukšto rezultato. Problema, jog kai kurie modeliai patys savyje turi reguliarizacijos mechanizmus, kaip kad Lasso, Ridge ar ElasticNet. Šių modelių atveju, Borutos atrinkti požymiai gali ne tik kad nepagerinti rezultatų, tačiau priešingai – juos pabloginti.

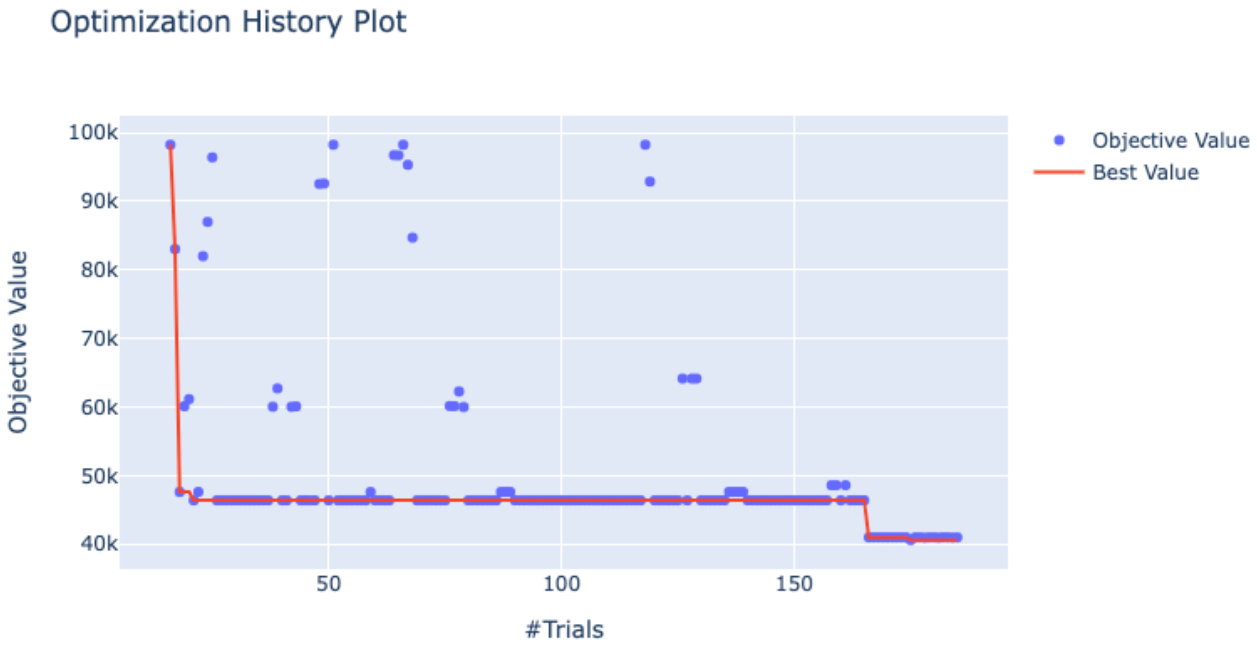
Štai pavyzdys su Lasso modelio parametrų patikra, kur yra ieškoma: ar lognormuoti regresantą, kokius požymius pasirinkti – visus, atrinktus Lasso ar Borutos, bei kokią alpha reikšmę pasirinkti (3 priedas). Iš tiesų galime matyti, jog esant alpha=0.001, geriausias rezultatas buvo pasiektas naudojant visus duomenis, bei lognormuojant regresantą. Taip pat galime rasti atvejų kuomet pritaikius panašius kriterijus, tačiau atrinkus iš anksto kintamuosius – rezultatas suprastėja.

**17 lentelė.** Lasso modelio parametrų paieškos rezultatai naudojant tinklelio paiešką

| RMSE | INVERSE Y | ALPHA | FEATURES |
|---|---|---|---|
| 38900 | Yes | 0,001 | passthrough |
| 40081 | Yes | 0,0014 | passthrough |
| 44997 | No | 0,0058 | Lasso selected |
| 49557 | Yes | 0,0038 | Boruta selected |
| 50084 | Yes | 0,0046 | Boruta selected |
| 50156 | Yes | 0,001 | Lasso selected |
| 52243 | Yes | 0,0034 | Lasso selected |
| 53570 | Yes | 0,0058 | Lasso selected |

Dėl šios priežasties, turime tikrinti ne tik modelių hiperparametrus, tačiau ir kitus du galimus sprendimus susijusius su modelio rezultatais: ar naudoti atrinktus svarbiausius požymius ir jei taip kuriuos, bei jei tai yra tiesinis modelis ar naudoti regresanto transformacija pašalinančią netiesiškumus. Paieškai galime pasitelksime Optuna biblioteką.

Svarbu paminėti, jog Optuna biblioteka nors ir yra paremta Bajeso teorema siekiant atrinkti optimalius parametrus – ji vis vien gali užstrigti vietiniame minimume, jei parametrų paieškos laukas (angl. *search space*) yra gan platus, o gaunami rezultatai tam tikrose parametrų ribose yra vienodi. Iliustracijai galime panaudoti tą pačia Lasso parametrų paiešką. Naudojant Optuna buvo 160 iteracijų buvo bandyta surasti optimalius parametrus, tačiau kaip matome iš grafiko, kad beveik su 20-ta iteracija „geriausias“ rezultatas nejudėjo gan ilgai. Iš ankstesnio bandymo mes žinome, jog galime gauti RMSE kuri yra ~39000, esant itin mažam alpha (0.001). Vis dėl to, ilgą laiką šis alpha dydis buvo neatrastas, nes alpha paieškos laukas buvo gan platus [0.0001; 0.01].

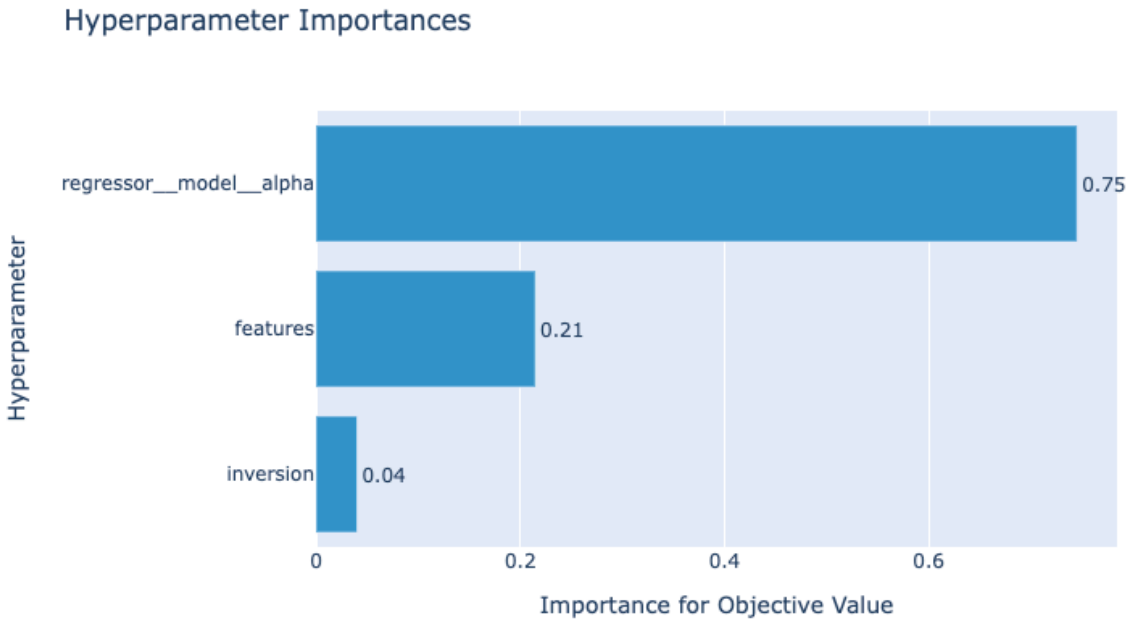



**33 pav.** Lasso modelio parametrų paieškos rezultatai naudojant Optuna

Galime matyti jog Optuna daugiau bandė didesnes alpha reikšmes, nei mažesnes, o kadangi daugumai alpha reikšmių rezultatas buvo panašus, optimalus alpha taip ir nebuvo pasiektas. Viskas pasikeitė, kai sumažinome alpha paieškos lauką ir padarėme jį [0.001; 0.0015]. Šis sprendimas leido greitai atrasti optimalią alpha reikšmę ir priartėti prie RMSE reikšmės kurią mes buvome radę ankščiau. Dėl šios priežasties yra itin svarbu panaudoti ir ekspertines žinias apie modelių veikimo principus ir taip susimažinti paieškos lauką. Tai leidžia taupyti kompiuterio resursus ir ieškoti optimalių parametrų kurie tikrai gerina rezultatus, o ne juos blogina.


**34 pav.** Lasso modelio parametrų paieškos rezultatai per parametrus naudojant Optuna

Optuna taip pat leidžia pažiūrėti kurie iš parametrų kurie buvo ieškoti yra svarbiausi. Iš nagrinėto pavyzdžio galime matyti, jog *alpha* yra svarbiausias parametras, tačiau taip pat pastebima ir požymių atrankos svarba.




**35 pav.** Lasso modelio parametrų paieškos rezultatai per parametrų svarbą naudojant Optuna

Po parametrų paieškos su Optuna, pavyko pasiekti tas pačias pradines įžvalgas: Lasso modelis savyje jau turi reguliarizacijos mechanizmą apsaugantį modelį nuo prasto kokybės požymių, todėl modeliui geriau naudoti visus turimus požymius, o ne atfiltruotus. Taip pat regresanto lognormavimas yra naudingas ir padeda pasiekti geresnius rezultatus. Galiausiai alpha buvo atrinktas taip pat 0.001.

Kaip matome, požymiai įtakoja modelio rezultatus, tačiau tai nėra vienintelis dalykas kai kalbame apie tiesinius modelius. Kitas dalykas kurį jau buvome ne kartą palietę darbe tai yra regresanto lognormavimas, siekiant paversti jo skirstinį Gauso. Todėl kiekvienam modeliui tikrinsime ne tik požymius (visi, atrinktus Boruta, atrinktus Lasso), bet ir regresanto lognormavimą. Tai leis įvertinti požymių ir modelio parametrų sąveiką tarpusavyje, bei tiesinių modelių atveju – įtraukti ir regresanto skirstinį.

Beveik visi modeliai, kurie bus naudojami yra iš Sklearn[29] bibliotekos, todėl modelių angliški pavadinimai pateikiami taip, jog būtų galima rasti bibliotekoje. Darbe bus naudojami šie tiesiniai modeliai:

1. Tiesinė regresija;
2. Lasso regresija;
3. Ridge regresija;
4. ElasticNet regresija;
5. atraminių vektorių mašina regresijai (angl. *SVR*);
6. stochastiniu gradientiniu nusileidimu grįsta regresija (angl. *SDGRegressor*);
7. Bajeso teorema grįsta Ridge regresija (angl. *BayesianRidge*).

Darbe bus naudojami šie medžių modeliai:

8. Atsitiktinis miškas (angl. *Random Forest*);
9. Papildomų medžių regresija (angl. *Extra Tree Regressor*)
10. gradientinio pastiprinimo regresija (angl. *Gradient Boosting Regressor);*
11. Ada pastiprinimo regresija (angl. *Ada Boost Regressor*);
12. savirankos regresija (angl. *Bagging Regressor*);
13. histograma grįsto gradientinio pastiprinimo regresija (angl. *HistGradientBoostingRegressor*);
14. ekstremalus gradientinis pastiprinimas (angl. *XGBoost*);
15. švelni gradientinio didinimo mašina (angl. *Light Gradient Boosted machine / Light GBM*);
16. CatBoost.

Taip pat darbe bus naudojami po vieną neuroninį modelį bei artimiausių kaimynų iš Sklearn bibliotekos:

17. Daugiasluoksnis perceptronas regresijai (angl. *MLPRegressor*)
18. Artimiausi kaimynai regresijai (angl. *KNeighborsRegressor*)

Kiekvienam iš šių modelių buvo ieškoti parametrai minimizuojant RMSE klaidą. Atliekant paieška su Optuna buvo laikomasi jog kiekvienam modeliui minimaliai yra skiriama arba 60 minučių paieškos lako arba 100 paieškų – priklausomai nuo to kas nutinka greičiau. Kaip matoma iš rezultatų ne visiems modeliams buvo skirta vienodai laiko, bei ne visiems modeliams buvo atliktas vienodas paieškų skaičius. Iš viso Optuna paieškoje praleista ~19 valandų ant macbook pro kompiuterio ant CPU su 2,3 GHz Dual-Core Intel Core i5 procesoriumi ir 8 GB RAM. Kompiuteris turi 4 branduolius, kuriuos Optuna utilizavo atliekant skaičiavimus paraleliai. Todėl galutinėje rezultatų lentelėje pateikiamas visas vartotojo laikas, bendras laikas sudėjus visus procesorius, vidutinis paieškos laikas. Taip pat be modelio parametrų buvo ieškoma kokius požymius naudoti ir jei tai yra tiesinis modelis ar naudoti regresanto lognormavimą.

[29] https://scikit-learn.org/stable/index.html

Toliau pateikiami paieškos rezultatai. Kalbant apie tiesinius modelius, galime matyti, jog beveik visi modeliai „atrado“ jog lognormuojant regresantą yra pasiekiami geresni rezultatai. Vienintelis ElasticNet buvo kuriam šios transformacijos neprireikė. Šis modelis neturi didelio parametrų paieškos laiko ir buvo atliktas sąlyginai didelis paieškų skaičius. Vis dėl to, tikėtina jog po daugiau paieškų, šis rezultatas turėtų pasikeisti, kadangi žvelgiant į RMSE rezultatus – modelio rezultatai dar nėra pasiekę Lasso ar Ridge rezultatų, todėl galima manyti, jog 300 bandymų šiam modeliui nepakako.

Iš požymių atrankos pusės, buvo arba visi požymiai, arba atrinkti Boruta. Galime matyti jog MLPRegressor vienintelis pasirinko požymius atrinktus Lasso, tačiau kadangi šio modelio apmokymo laikas yra gana ilgas ir buvo atliktos tik 8 paieškos – šis rezultatas buvo atrinktas atsitiktinai bandant įvairias kombinacijas ir modelis dar nespėjo atrasti tinkamų parametrų kombinacijos kaip kad galime manyti jog nutiko su ElasticNet modeliu. Bandant įžvelgti Boruta atrinktų požymių naudą, galime matyti, jog ne visi modeliai pasirinko naudoti šiuos požymius. Iš vienos pusės, požymių atranka nėra svarbi modeliams kurie savyje turi požymių atrankos mechanizmus – t.y. visi medžių modeliai, tiesiniai modeliai su reguliarizacija ir neuroniniai tinklai. Medžių modeliai tuo tarpu yra itin lankstūs požymių atrankoje, kadangi sprendimų ir regresijos medžių sudarymo principas yra informacijos išlošis (angl. *information gain*), kurio dėka neinformatyvus požymiai yra neatrenkami [42]. Vis dėl to, požymių atranka vis tiek yra naudinga, nes trumpina skaičiavimo laiką, o atliekant parametrų paiešką – tai tampa itin svarbu. Vis dėl to, požymių atranka gali pagerinti rezultatus net ir tuomet, kai pačiame modelyje šis mechanizmas jau yra, tačiau galiausiai viską nulemia paties modelio parametrai bei, žinoma, kaip ir kokie požymiai buvo atrinkti.

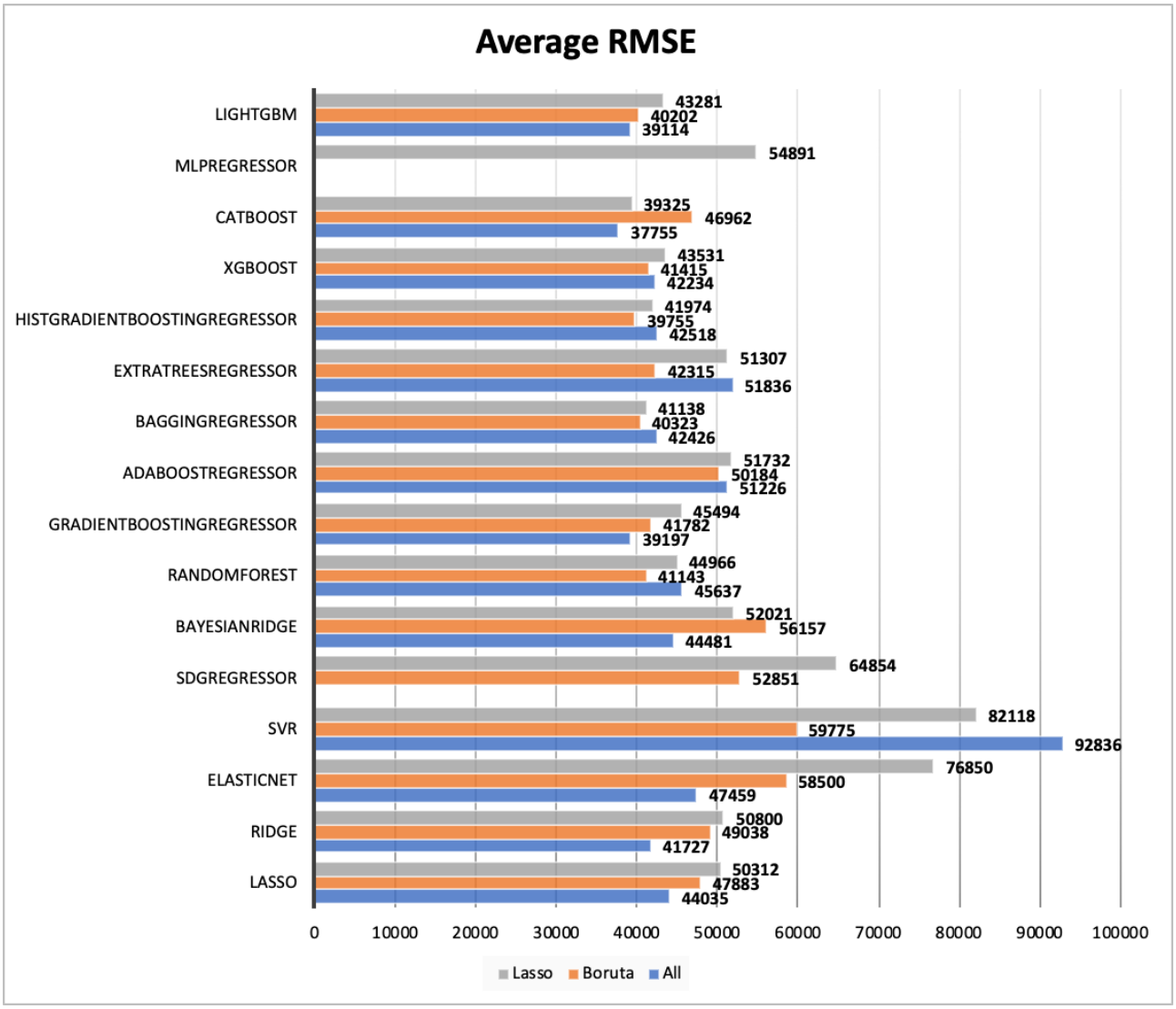
**18 lentelė.** Optuna rezultatų suvestinę

| MODEL | OPTUNA ROUNDS | RMSE | FEATURES | INVERSE | TOTAL USER TIME (minutes) | TOTAL CPU TIME (minutes) | AVG CPU TIME (seconds) |
|---|---|---|---|---|---|---|---|
| Lasso | 100 | 40589 | ALL | TRUE | 7 | 29 | 0,28 |
| Ridge | 100 | 41082 | ALL | TRUE | 6 | 25 | 0,25 |
| ElasticNet | 300 | 43474 | ALL | FALSE | 16 | 62 | 0,2 |
| SVR | 391 | 43728 | BORUTA | TRUE | 40 | 160 | 0,4 |
| SDGRegressor | 250 | 49432 | BORUTA | TRUE | 12 | 46 | 0,2 |
| BayesianRidge | 97 | 42144 | ALL | TRUE | 82 | 328 | 3,38 |
| MLPRegressor | 8 | 46840 | LASSO | #N/A | 402 | #N/A | #N/A |
| RandomForest | 334 | 40091 | BORUTA | #N/A | 68 | 270 | 0,8 |
| GradientBoostingRegressor | 71 | 34547 | ALL | #N/A | 75 | 299 | 4,2 |
| XGBoost | 56 | 35148 | ALL | #N/A | 158 | 631 | 11,27 |
| Catboost | 8 | 37755 | ALL | #N/A | 88 | 353 | 224,16 |
| AdaBoostRegressor | 100 | 48874 | BORUTA | #N/A | 39 | 156 | 1,55 |
| BaggingRegressor | 100 | 39532 | BORUTA | #N/A | 37 | 147 | 1,47 |
| ExtraTreesRegressor | 122 | 39209 | BORUTA | #N/A | 17 | 67 | 0,55 |
| HistGradientBoostingRegressor | 39 | 35940 | ALL | #N/A | 71 | 282 | 2,57 |
| LightGBM | 250 | 37450 | ALL | #N/A | 30 | 120 | 0,5 |

Tęsiant požymių atrankos įtaką rezultatams toliau buvo nagrinėti rezultatai per naudotus požymius – visus, atrinkus Boruta arba atrinkus Lasso. Iš RSME pusės, buvo paimti visi Optuna paieškos

rezultatai ir per naudotų požymių tipą išvesti vidurkiai. Taip pat, dėl konvergavimo klaidų bei itin didelių reikšmių keliems modeliams atitinkami rezultatai buvo pašalinti. Iš šių rezultatų galime matyti, jog daugumai medžių modelių didelių skirtumų tarp naudotų požymių nebuvo, kita vertus galime matyti, jog vidutiniškai geriausi rezultatai buvo pasiekti su Boruta požymiais, tačiau ne visi modeliai su jais apsistojo, pavyzdžiui XGBoost. CatBoost modelį sunku vertinti, kadangi jam buvo įvykdytos tik 8 paieškos, tuo tarpu LightGBM panašu jog nuo pat pradžių labiau vertino didesnį požymių skaičių.

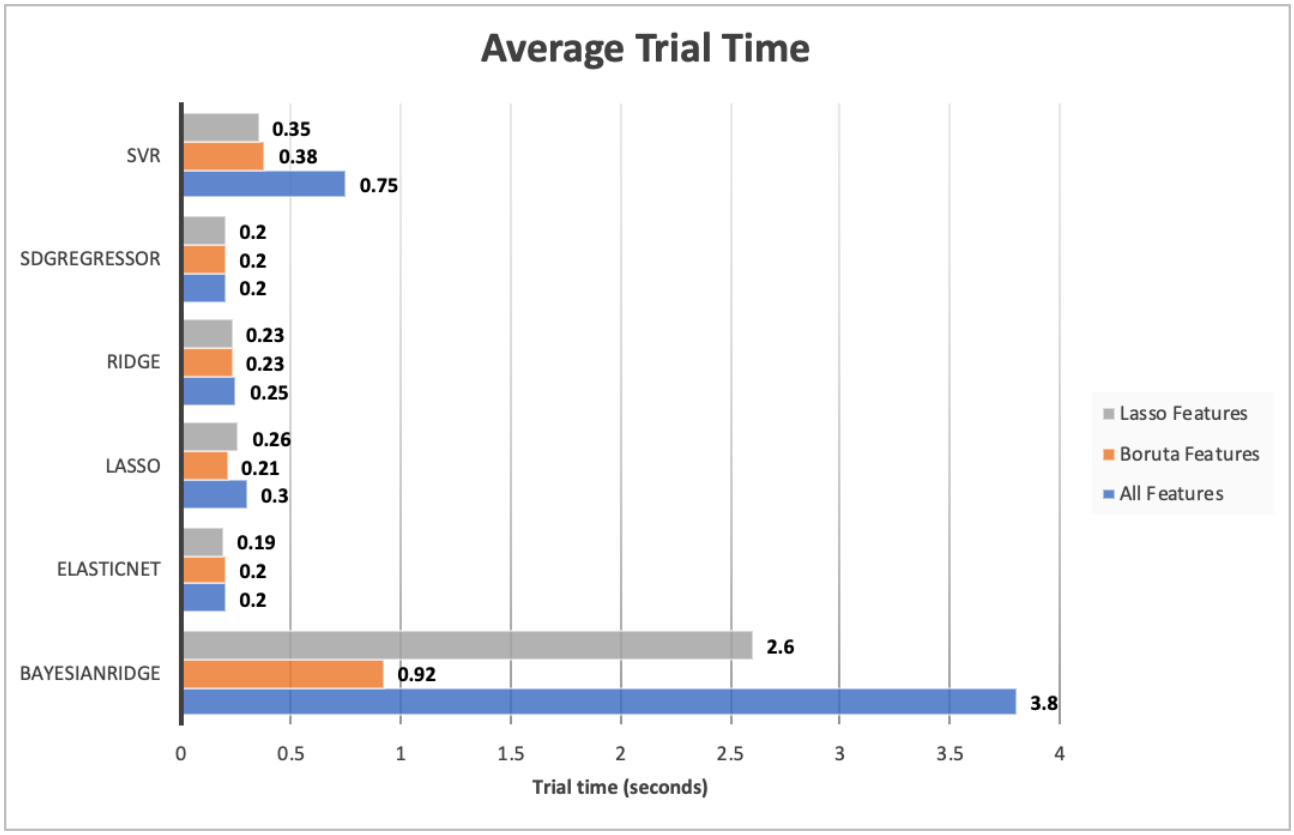
Visi darbe naudoti tiesiniai modeliai turi reguliarizacijos mechanizmus, tačiau ne visuose ji yra išreikšta vienodai. Lasso, Ridge ir ElasticNet reguliarizacija yra tarpusavy labai panašūs modeliai, su paprastai išreikšta reguliarizacija, todėl nenuostabu, jog šiems modeliams nuo pat pradžių sekėsi geriausiai su visais požymiais. Tuo tarpu SVR ir SDGRegressor modeliams aplamai sekėsi sunkiau su rezultatais – jie buvo prasčiausi, net ir su ~300 Optuna paieškų.




**36 pav.** Optuna rezultatai pateikti per naudotus požymius

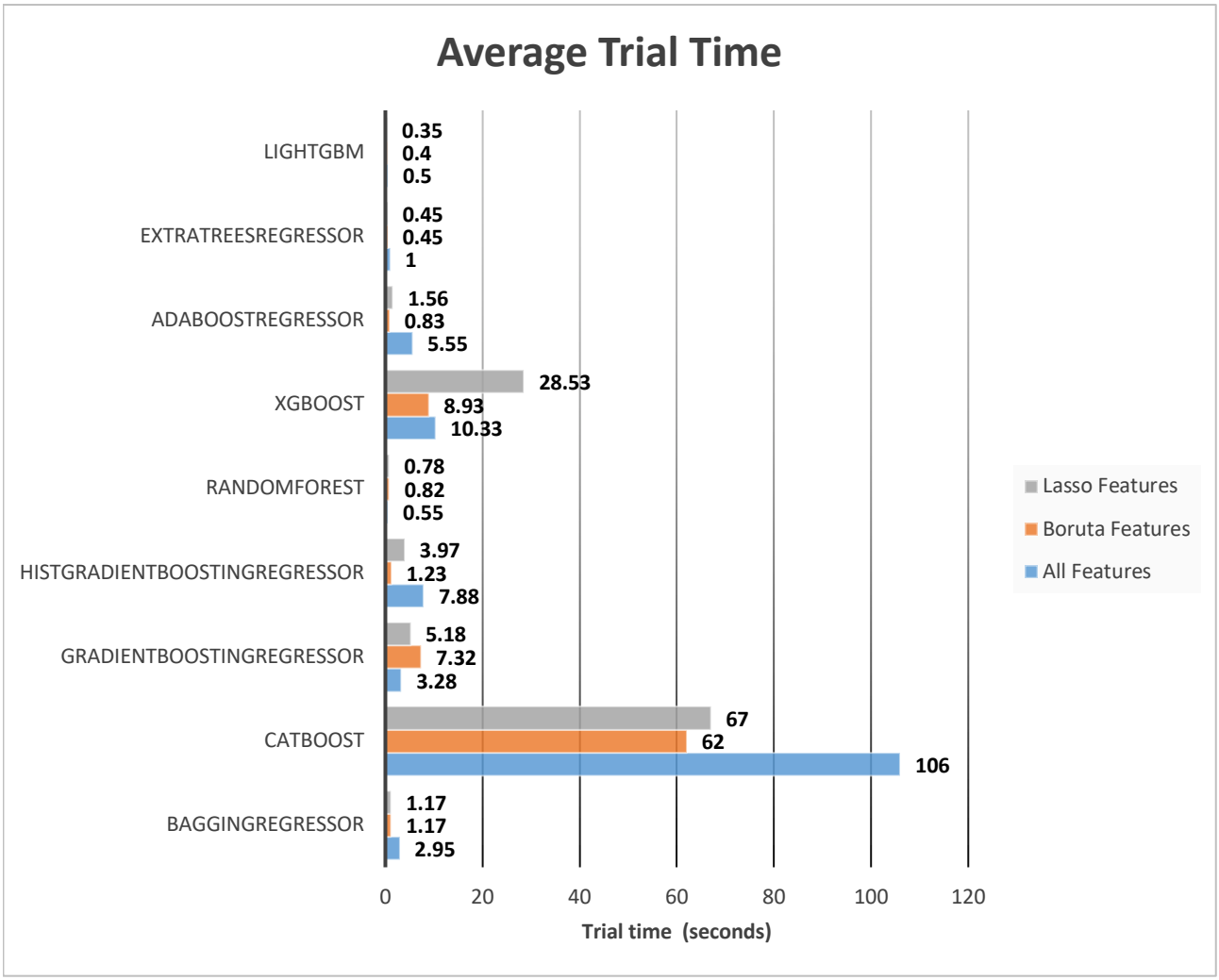
Taip pat rezultatai palyginti per skaičiavimo laiką. Tiesinių modelių rezultatai yra labai panašūs – daugumai modelių požymių sumažinimas nedarė įtakos skaičiavimo greičiui. Vienintelės dvi

išimtys yra SVR ir BayesianRidge modeliai, kuriems požymių mažinimas vidutiniškai pagreitino paieškos skaičiavimą 2-3 kartais.




**37 pav.** Optuna rezultatai pateikti per tiesinių modelių vidutinį paieškų laiką

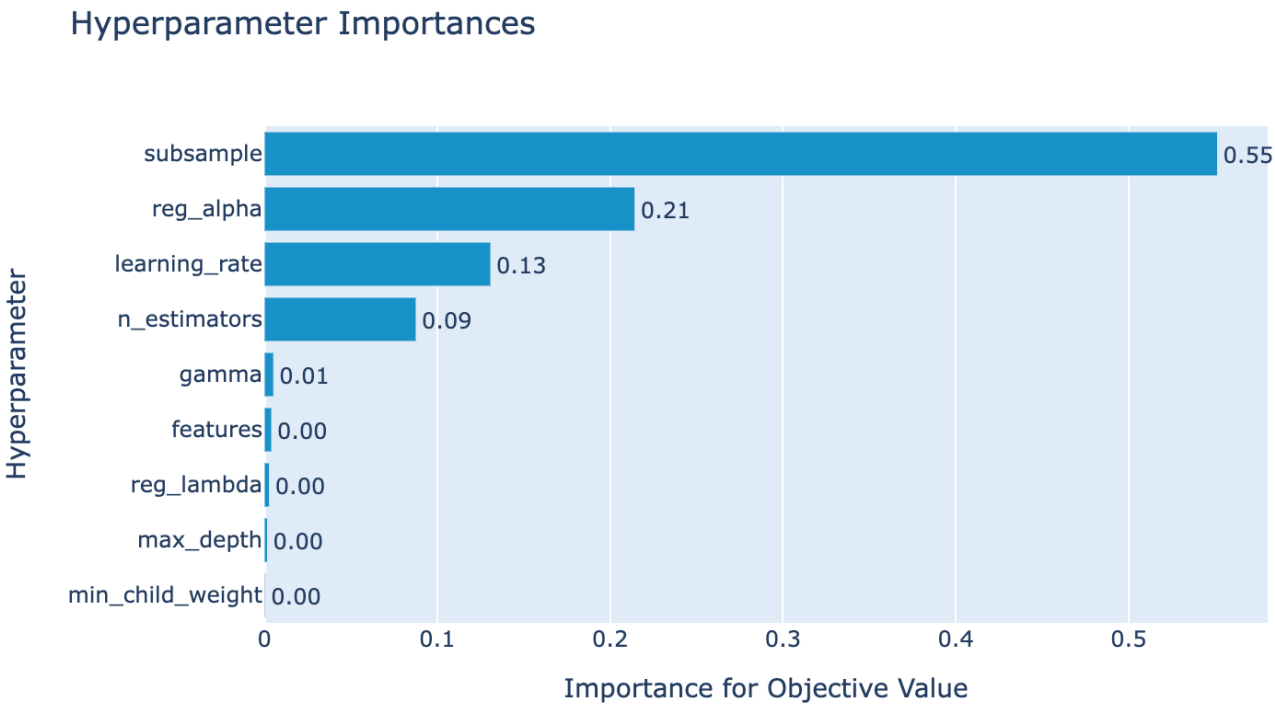
Tuo tarpu likusiems kitiems modeliams situacija buvo gana panaši – požymių mažinimas trumpina skaičiavimo laiką. Tai itin aktualu modeliams kurių vidutinis skaičiavimo laikas yra ilgesnis ir užtrunka ilgiau nei vieną minutę. Kaip matome iš pavyzdžio CatBoost paieška buvo itin ilga, vidutiniškai trunkanti ilgiau nei valandą per vieną bandymą. Vis dėl to, nors vidutiniai skaičiavimai su Boruta atrinktais požymiais buvo trumpesni, ne visiems modeliams pavyko pasiekti geriausius rezultatus naudojant būtent Boruta atrinktus požymius.




**38 pav.** Optuna rezultatai pateikti per netiesinių modelių vidutinį paieškų laiką

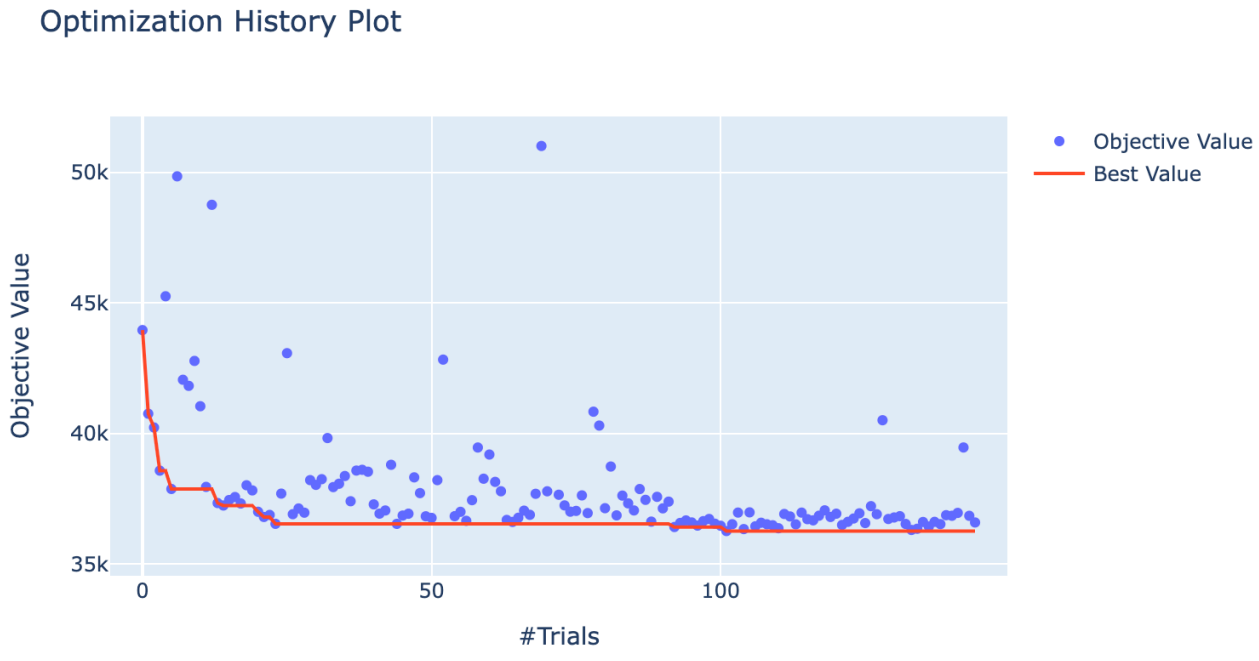
Iš atliktos paieškos galime pastebėti, jog tiesiniams modeliams regresanto transformavimas į Gauso skirstinį turinčio yra svarbus, tačiau dėl požymių mažinimo ir atrankos situacija šiek tiek sudėtingesnė. Visų pirma, akivaizdu, jog atrinkti požymiai mažina skaičiavimo laiką. Darbe nagrinėjame pavyzdyje, tai matoma ne visuose modeliuose, tačiau verta atkreipti dėmesį, jog mūsų turimas duomenų rinkinys yra itin mažas. Esant milijonams įrašų, skirtumą pastebėtume ir tarp tiesinių modelių.

Kitas svarbus dalykas yra tai, jog daug ką lemia modelių parametrai. Štai pavyzdyje pateikiamas XGBoost modelio, kuriam buvo atliktos 56 Optuna paieškos ieškotų parametrų svarba. Galime matyti, jog požymis „features“ po kuriuo tikrinome naudoti visus ar atrinktus, visiškai nebuvo svarbus modelio rezultatams. Nors su Boruta atrinktais požymiais modelis veikė greičiausiai, o vidutiniškai rezultatai buvo geriausiai esant optimaliai atrinktiems parametrams – visa tai tampa nesvarbu.



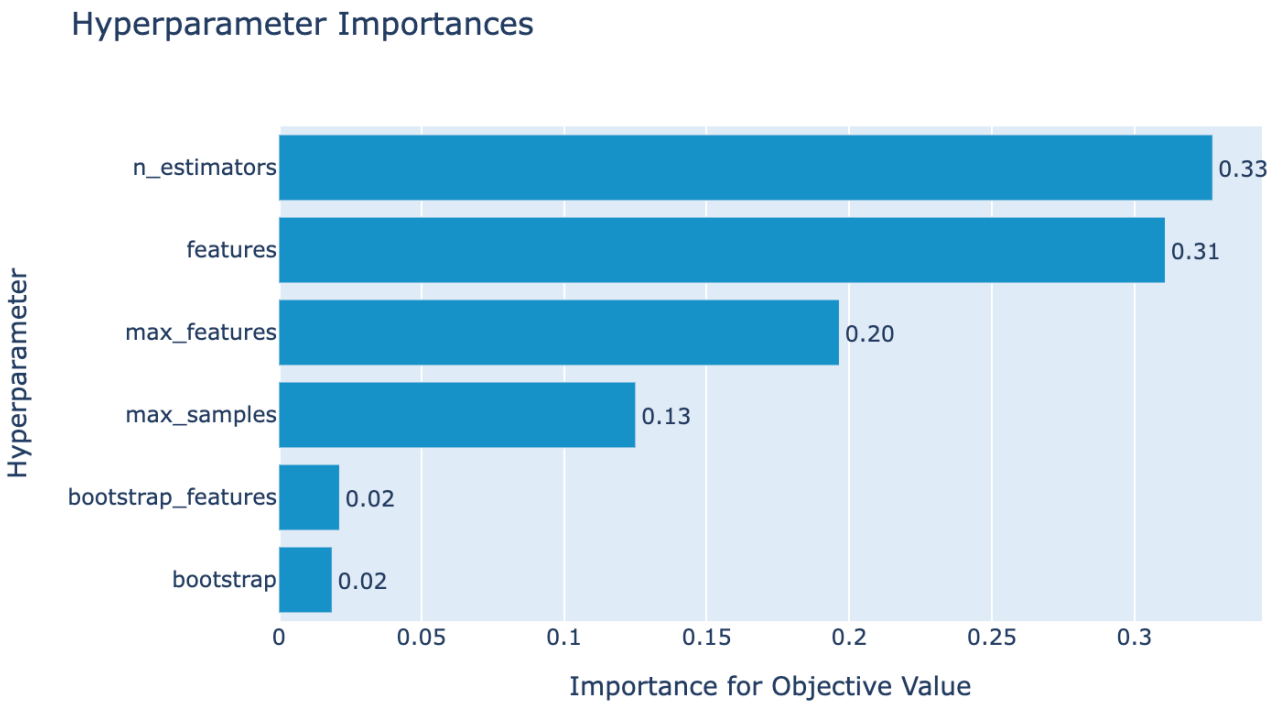

**39 pav.** XGBoost modelio parametrų svarba optimaliam rezultatui

Vis dėl to, gali kilti mintis, jog paprasčiausiai rezultatai nekonvergavo ir esant daugiau paieškos bandymų, galėtume gauti geresnius rezultatus naudojant tik Boruta atrinktus požymius. Taip pat, kadangi kiekviena iteracija būtų greitesnė, per ta patį laiką galėtume atlikti ir daugiau bandymų. Būtent tai ir buvo pabandyta atlikti – iš paieškos išėmėme naudojamų požymių parametrą ir atlikome 145 bandymus. Tai užtruko vartotojo laiku pusantros valandos, kas yra pusantro karto trumpiau, tačiau 3 kartus daugiau bandymų. Vis dėl to, gauti rezultatai parodė, jog geriausias rezultatas kurį pavyko pasiekti yra 36268 (geriausias 35148). Nors skirtumas nėra žymus, tačiau net ir po daugiau bandymų rezultato pagerinti nepavyko.



**40 pav.** XGBoost modelio parametrų paieška naudojant Optuna ir Boruta atrinkus požymius

Kita vertus, BaggingRegressor modelio atveju, požymiai kurie buvo naudoti modelyje yra antras svarbiausių dalykų.




**41 pav.** BaggingRegressor modelio parametrų svarba optimaliam rezultatui

Taigi, iš rezultatų galime matyti, jog nors ir ne visiems modeliams požymių atranka yra naudinga, tačiau net ir tuomet, kai yra nedidelis duomenų rinkinys, požymių atranka / dimensijų mažinimas gali būti naudingas atitinkamiems modeliams, todėl atliekant parametrų atranką derėtų atsižvelgti ir modelyje naudojamus požymius.

### 3.7. Galutinių rezultatų skaičiavimas

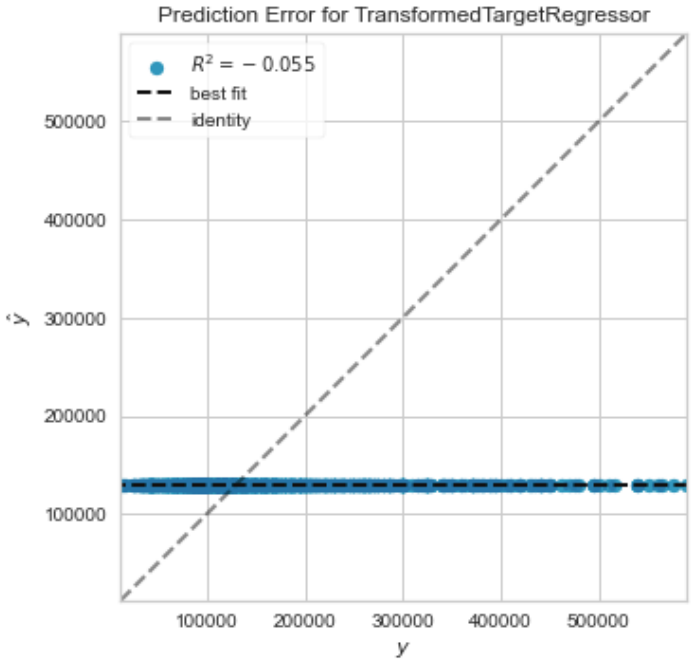
Iš pradžių paskaičiuosime visus modelius su gamykliniais parametrais bei pradinėmis prielaidomis:

- Visiems tiesiniams modeliams taikomi Lasso atrinkti požymiai, o regresantas yra lognormuojamas;
- visiems medžių modeliams yra taikomi Boruta atrinkti rezultatai.

Kad tinkamai įvertinti modelių parametrų ir grandinėje atliekamų transformacijų naudą, rezultatus skaičiuosime pradiniams (angl. *baseline*) modeliams ir modeliams su atrinktais parametrais. Pradiniai modeliai naudos atitinkamą grandinę su joje esančiomis transformacijomis ir gamyklinius modelio parametrus. Šių modelių rezultatai bus lyginami su parametrais atrinktais naudojant Optuna.

Toliau pateikiami pradinių modelių rezultatai. Čia pateikiami visi sekami modelio kokybės įverčiai, bei laikas (sekundėmis) kiek užtruko paskaičiuoti rezultatus. Visų pirma, matome, jog visi modeliai skaičiavimo itin greitai – dauguma skaičiavimus (įskaitant ir grandinės transformacijas) atliko per greičiau nei 10 sekundžių. Vis dėl to, neuroninis tinklas ir atsitiktiniai miškai čia buvo lėčiausi –

užtrukdami ~ 24 sekundes. Taip pat iš karto galime matyti, jog yra keletas modelių, kurių R2 įvertis yra neigiamas. Tiesiniai modeliai gali turėti šį neigiamą įvertį jeigu yra neapskaičiuota konstanta (kas kartais gali būti renkamasi, kai nenatūralu turėti laisvąjį narį) arba kai turimas modelis itin prastai prognozuoja regresantą. Jeigu R2 yra artimas nuliui – modelio prognozuojamoji galia yra tokia pati, kaip nubrėžus tiesią horizontalią liniją, o jei R2 yra neigiamas – horizontali linija prognozuoja geriau.




**42 pav.** Pradinio Lasso modelio paklaidų vizualizacija

Pažvelgę į Lasso atvejį, galime matyti, jog visi būstai buvo suvertinti vienodai – nepaisant to kokie jie. Kadangi buvo naudoti gamykliniai parametrai, kur Lasso gamyklinis *alpha* yra 1, Lasso modelis per daug "nubaudė" požymius. Iš Optuna parametrų paieškos žinome, jog *alpha* turėtų būti mažas dydis, jog pasiektų gerus rezultatus. Todėl šiuo atveju prasti rezultatai buvo netinkami gamykliniai parametrai. Tą patį matome ir su ElasticNet modeliu.

Taip pat galime matyti jog vieninteliam turimam neuroniniam modeliui sekasi itin sunkiai – jo rezultatai patys prasčiausi. Sklearn biblioteka suteikia jų infrastruktūrinį sprendimą naudoti neuroninius tinklus. Tai padeda vartotoju greičiau ir paprasčiau naudoti šio tipo modelius, kai šis yra susipažinęs su bendra Sklearn bibliotekos modelių struktūra. Vis dėl to, šio modelio gamykliniai parametrai visiškai klaidinga atlieka regresanto prognozę – net paprastai tiesiniai regresijai sekasi geriau.

Medžių modeliams sekasi geriausiai. Itin aukšti rezultatai yra pasiekiami su atsitiktiniais miškais, XGboost, LightGBM, CatBoost ir histograma grįsto gradientinio pastiprinimo regresijos medžio. Pastarasis modelis tapo geriausias ir pasiekė 37,869 RMSE rezultatą.

**19 lentelė.** Visų pradinių modelių rezultatai

| Model_name | R2 Score | MAE | MAPE | MSE | RMSE | Time |
|---|---|---|---|---|---|---|
| Linear Regression | 0.76 | 29,666.74 | 19.82 | 2,130,373,854.54 | 46,155.97 | 3.96 |
| Lasso Regression | -0.10 | 64,049.28 | 45.84 | 9,653,388,443.77 | 98,251.66 | 3.49 |
| Ridge Regression | 0.75 | 29,770.79 | 19.72 | 2,212,686,677.18 | 47,039.20 | 3.32 |
| Elastic Net Regression | -0.10 | 64,049.28 | 45.84 | 9,653,388,443.77 | 98,251.66 | 3.55 |
| SVM | 0.77 | 26,788.79 | 17.21 | 2,005,839,986.62 | 44,786.60 | 5.15 |
| Stochstic Gradient Descent | 0.23 | 49,199.98 | 31.28 | 6,738,209,064.30 | 82,086.59 | 3.38 |
| Bayesian Ridge Regression | 0.75 | 29,923.37 | 19.98 | 2,196,754,017.27 | 46,869.54 | 3.22 |
| KNN Reggressor | 0.78 | 27,375.03 | 18.46 | 1,920,219,266.76 | 43,820.31 | 0.19 |
| Random Forest | 0.82 | 24,760.49 | 16.79 | 1,608,721,361.36 | 40,108.87 | 23.60 |
| Ada Boost Regressor | 0.71 | 40,018.76 | 38.02 | 2,577,240,643.41 | 50,766.53 | 6.00 |
| Bagging Regressor | 0.79 | 26,893.82 | 18.02 | 1,844,926,496.68 | 42,952.61 | 5.36 |
| Gradient Boosting Regressor | 0.82 | 25,335.82 | 17.63 | 1,598,453,663.97 | 39,980.67 | 12.30 |
| Extra Trees Regressor | 0.82 | 23,147.06 | 15.44 | 1,541,282,213.48 | 39,259.17 | 12.75 |
| Histogram-based Gradient Boosting Regression Tree | 0.84 | 23,850.49 | 16.26 | 1,434,128,969.11 | 37,869.90 | 6.43 |
| XGBoost | 0.82 | 24,818.96 | 16.97 | 1,612,855,837.34 | 40,160.38 | 8.76 |
| Catboost | 0.83 | 24,527.00 | 17.02 | 1,506,759,198.93 | 38,817.00 | 14.21 |
| LightGBM | 0.82 | 24,551.70 | 16.73 | 1,557,970,060.02 | 39,471.13 | 7.86 |
| Multi-layer Perceptron regressor | -1.05 | 108,538.39 | 72.34 | 17,958,195,150.50 | 134,008.19 | 23.85 |

Toliau yra pritaikomi visi parametrai atrinkti naudojant Optuna. Mūsų tikslas yra pagerinti pradinių modelių rezultatus. Galime matyti, jog visų modelių rezultatai pagerėjo. Tačiau galime pastebėti, jog tam tikriems modeliams pastebimai išaugo skaičiavimo laikas. Ypač CatBoost – paskaičiuoti rezultatus jam užtruko ~14 minučių, nors rezultatai pagerėjo nežymiai. Geriausias rezultatas vis dėl to buvo pasiektas su gradientinio pastiprinimo (angl. *Gradient Boosting Regressor*) modeliu, jo RMSE yra 34,547. Vis dėl to, mažiausia MAPE yra histograma grįsto gradientinio pastiprinimo regresijos medžio. Pastarajam, tikėtina, geriau sekasi prognozuoti brangesnius būstus, vidutinių būstų kainos sąskaita, todėl bendra procentinė paklaida yra mažesnę.

**20 lentelė.** Visų modelių su optimaliais parametrais rezultatai

| Model_name | R2 Score | MAE | MAPE | MSE | RMSE | Time |
|---|---|---|---|---|---|---|
| Linear Regression | 0.76 | 29,666.74 | 19.82 | 2,130,373,854.54 | 46,155.97 | 3.84 |
| Lasso Regression | 0.81 | 24,541.26 | 15.64 | 1,647,487,889.29 | 40,589.26 | 4.64 |
| Ridge Regression | 0.81 | 24,301.01 | 15.81 | 1,687,804,585.25 | 41,082.90 | 4.60 |
| Elastic Net Regression | 0.78 | 30,204.53 | 24.00 | 1,890,051,754.95 | 43,474.73 | 3.68 |
| SVM | 0.78 | 27,030.20 | 17.70 | 1,912,208,203.77 | 43,728.80 | 5.57 |
| Stochstic Gradient Descent | 0.72 | 31,115.70 | 20.08 | 2,443,529,088.49 | 49,432.07 | 3.24 |
| Bayesian Ridge Regression | 0.80 | 24,918.84 | 16.22 | 1,776,128,369.63 | 42,144.14 | 7.95 |
| KNN Reggressor | 0.80 | 24,322.38 | 16.16 | 1,740,854,803.47 | 41,723.55 | 0.13 |
| Random Forest | 0.81 | 24,835.88 | 16.76 | 1,629,803,544.33 | 40,370.83 | 23.69 |
| Ada Boost Regressor | 0.73 | 36,893.88 | 34.02 | 2,388,709,495.00 | 48,874.43 | 15.39 |
| Bagging Regressor | 0.82 | 24,161.79 | 16.68 | 1,562,846,277.17 | 39,532.85 | 18.68 |
| Gradient Boosting Regressor | 0.86 | 21,850.95 | 15.27 | 1,193,510,824.54 | 34,547.23 | 59.03 |
| Extra Trees Regressor | 0.82 | 23,300.57 | 15.42 | 1,537,407,808.12 | 39,209.79 | 5.62 |
| Histogram-based Gradient Boosting Regression Tree | 0.85 | 21,994.76 | 14.59 | 1,291,711,487.63 | 35,940.39 | 202.42 |
| XGBoost | 0.86 | 22,256.48 | 15.45 | 1,235,424,544.62 | 35,148.61 | 184.35 |
| Catboost | 0.84 | 23,385.94 | 16.11 | 1,433,786,874.95 | 37,865.38 | 850.42 |
| LightGBM | 0.84 | 23,949.97 | 16.64 | 1,408,119,259.42 | 37,524.92 | 8.88 |

Atlikus palyginimą, galime matyti, jog pradinių modelių geriausias rezultatas ne taip daug atsilieka nuo geriausio modelio po optimalių parametrų atrankos. Iš viso modelių parametrų paieškai buvo

praleista ~19 valandų, tačiau net ir toks laikas nebuvo pakankamas visiems modeliams atrasti tinkamus parametrus. Nors parametrų atranka ir padėjo pagerinti rezultatus nežymiai, laikas per kurį modeliai atlieka skaičiavimus išaugo.

Sekančiam žingsniui atrinksime TOP 5 geriausius modelius ir naudosime ansamblių kūrime. Renkant modelius atsižvelgta ne tik į modelio kokybę, tačiau ir laiką, per kurį buvo apskaičiuoti rezultatai. Dėl šios priežasties CatBoost buvo naudotas su gamykliniais parametrais, nes rezultatai skiriasi minimaliai, tačiau skaičiavimo laikas greitesnis 85 kartais. Buvo pasirinkti šie modeliai:

1. Gradientinio pastiprinimo modelis su optimaliais parametrais;
2. XGBoost su optimaliais parametrais;
3. Histograma grįsto gradientinio pastiprinimo regresijos medžio su optimaliais parametrais;
4. LightGBM su optimaliais parametrais;
5. CatBoost su gamykliniais parametrais.

Atrinkę modelius toliau išbandysime šias ansamblių technikas:

- Visų modelių prognozių rezultatus su meta modeliu – paprasta tiesine regresija (angl. *Stacking*);
- visų modelių prognozėms paprasčiausiai paskaičiuoti vidurkį;
- atlikti visų modelių balsavimą, kur kiekvieno modelio balso svoris yra nustatomas taip, jog paklaida būtų mažiausia, o visų modelių bendras balsas būtų lygus 1;
- parinkti geriausią ansamblio architektūrą ir iš jo liekanų sukurti antrą modelį, kuris pakoreguotų pirmojo paklaidas, kaip dar vieną mėginimą pagerinti modelio rezultatus.

Iš rezultatų galime matyti, jog jie dar kartą pagerėjo. Visi ansamblio metodai, atliko geresnius skaičiavimus, nei individualūs modeliai. Iš visų modelių, prasčiausiai sekėsi balsų ansambliui. Geriausiai pasirodė liekanų modelis. Šis modelis pradinėms prognozėms naudojo vidurkių ansamblį, o liekanoms skaičiuoti TOP 1 individualų modelį. Šio modelio rezultatai mažiausi per visus parametrus, o skaičiavimo laikas nėra itin ilgas, lyginant su „*stacking*“.

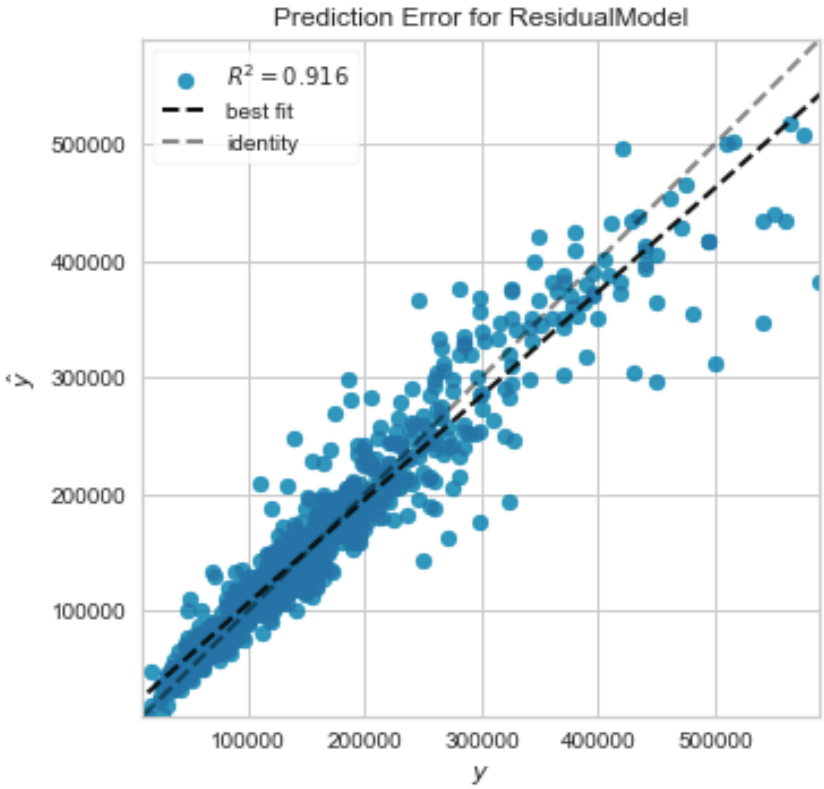
**21 lentelė.** TOP 5 modelių ansamblių rezultatai

| Model_name | R2 Score | MAE | MAPE | MSE | RMSE | Time |
|---|---|---|---|---|---|---|
| Stacking Models | 0.87 | 20,695.76 | 13.99 | 1,125,785,632.75 | 33,552.73 | 1,907.69 |
| Averaging Models | 0.87 | 20,663.98 | 14.02 | 1,156,017,310.87 | 34,000.25 | 359.63 |
| Ensemble Prediction | 0.85 | 22,203.91 | 14.69 | 1,335,156,497.66 | 36,539.79 | 340.07 |
| Residual Model | 0.87 | 20,224.45 | 13.74 | 1,109,365,262.59 | 33,307.14 | 439.82 |

Taigi, geriausias modelis ir šio projekto finalinis modelis yra liekanų modelis sudarytas iš TOP 5 geriausių individualių modelių prognozės vidurkio, bei ant viršaus sukurto papildomo liekanų modelio naudojantis geriausio individualaus modelio architektūrą – gradientinio pastiprinimo modelis su optimaliais parametrais. Toliau pateikiami geriausio modelio rezultatai.

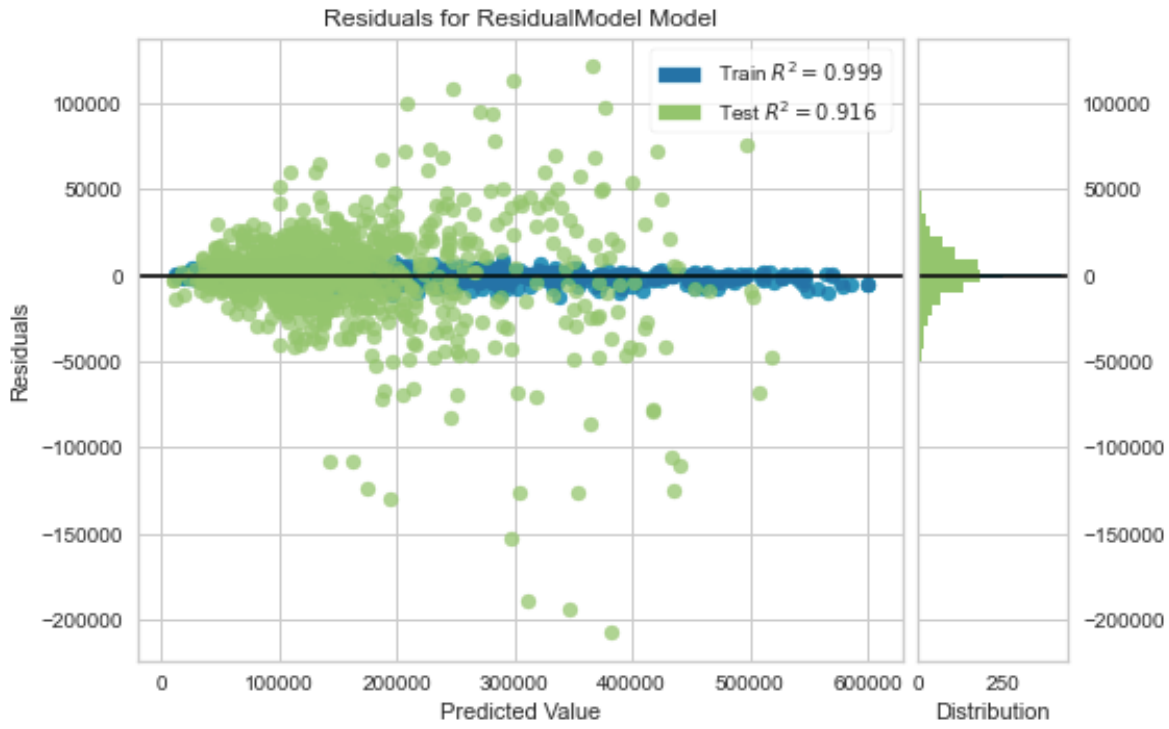
Visų pirma, galime matyti, jog modeliui iš itin gerai sekasi prognozuoti būstus, kurie yra verti ~100 – 200 tūkst. Eurų. Čia galime įžvelgti itin mažas paklaidas tarp prognozuojamos ir realios kainos. Taip pat matome, jog su pigesniais butais prognozė nėra tokia tiksli, tačiau kainų skirtumai stipriai nesiskiria. Panašu, jog iš grafiko 200 – 300 tūkst. Eur. Riba jau yra prognozuojama sunkiau, tačiau

dauguma būstų dar nėra itin blogai vertinami. Su dar brangesniais butais prognozuojamos vertės skirtumai didėja, vizualiai panašu, jog modelis yra linkęs labiau nesuvertinti tikrosios vertės, nei pervertinti, tačiau kad galėtume pilnai atsakyti į šį klausimą, toliau vertinsime liekanas.




**43 pav.** Geriausio modelio paklaidos

Liekanų grafike, galime pastebėti, jog modelis puikiai įsimena mokymo imtį, tačiau nepersimoko ir gali tinkamai prognozuoti objektus kurių nėra matęs. Taip pat galime matyti, jog modelis panašu jog daugiau objektų yra pakankamai nesuvertinęs (didelės paklaidos iki 200 tūkst. Eur.), tačiau ne tiek daug pervertina (paklaidos iki ~130 tūkst. Eur.). Vis dėl to, vertinant testavimo imties paklaidų skirstinį, galime įžvelgti jog skirstinys yra pasislinkęs į kairę nuo nulio, kas reiškia, jog testavimo imtyje modelis linkęs bendrai pervertinti būstus, nei nesuvertinti pakankamai. Taip pat galime matyti, jog mokymosi imtyje buvo suvertinti ir vidutinės vertės būstai ir brangesni, tačiau testavimo duomenyse sunkiau sekėsi įvairios vertės būstams, ne tik itin prabangiems.



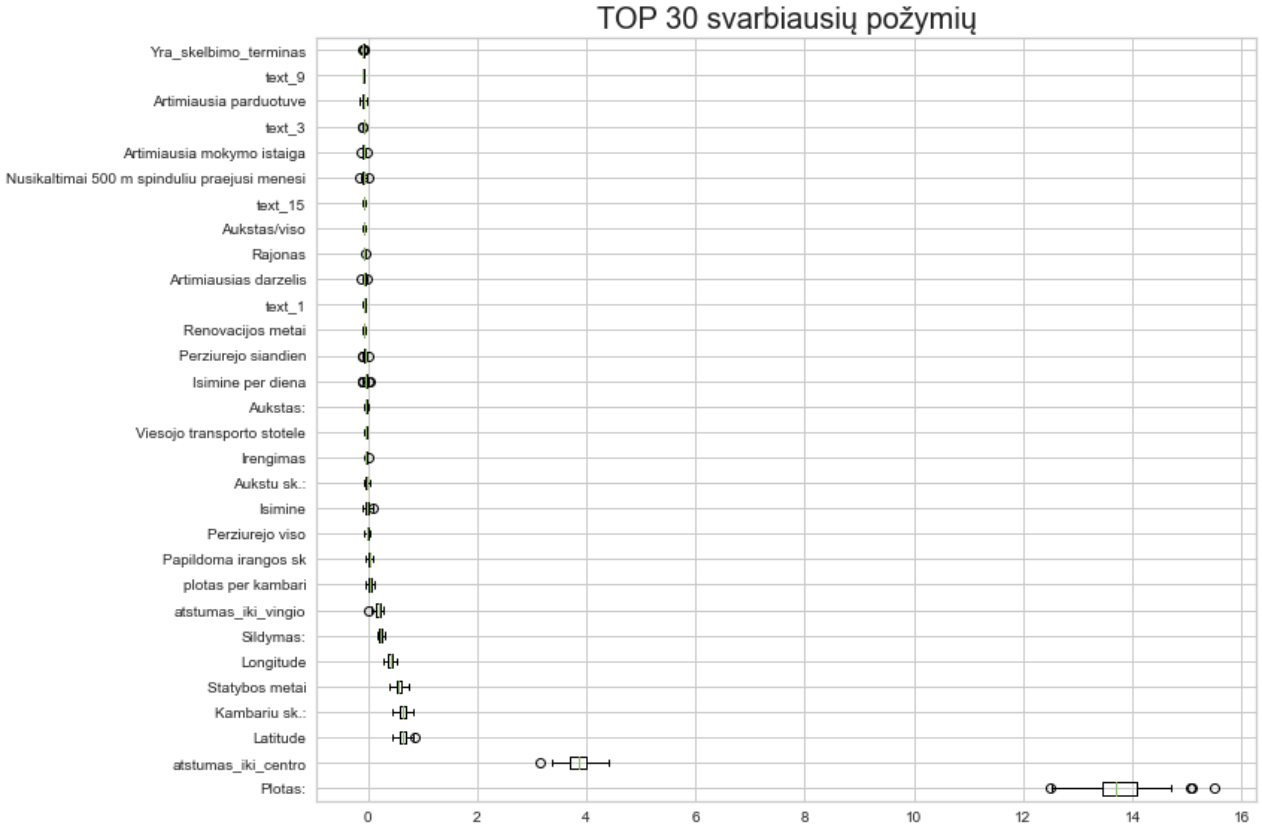

**44 pav.** Geriausio modelio liekanos mokymosi ir treniravimosi imtyse

Iš pateiktų rezultatų galime matyti, jog siūloma modelio architektūra ir jos kūrimas, leido pasiekti neblogus rezultatus, turint sąlyginai mažą duomenų rinkinį. Pasiūlytas modelis remiasi TOP 5 geriausiais individualiais modeliais, kuriems buvo atrinkti optimalūs parametrai, požymiai ir požymių inžinerija. Visų modelių prognozėms tuomet buvo paskaičiuotas vidurkis, kaip galutinė prognozė. Ši architektūra jau siekė gerus rezultatus, tačiau siekdami dar kartą pagerinti modelio prognozę ant viršaus buvo „uždėtas“ liekanų modelis, kuris naudojo tuos pačius požymius, tačiau vietoj kainos, naudojo liekanas gautas iš vidurkio prognozės. Tai prailgino skaičiavimo laiką, tačiau taip pat pagerino MAPE(%) iš 14.02 į 13.74.

Taip pat, geriausiam modeliui buvo patikrinti svarbiausi požymiai. Tikrinimas vyko naudojant kitą, nei Boruta metodą – kadangi modelio architektūra yra sudėtinga, apskaičiuoti požymių svarbą pavyko su Sklearn biblioteka, naudojant permutacijos (angl. *permutation*) metodą. Šis metodas, leidžia išmatuoti požymių svarbą, pagal tai kaip pasikeičia modelio prognostinė kokybė, kai konkretaus požymio reikšmės yra atsitiktinai sumaišomos. Taip pat, kadangi modelio grandinėje atliekame įvairių papildomų transformacijų, požymių svarbą tikrinsime pradiniams duomenims, o ne jų transformacijoms. Tokiu būdu, įvertinsime požymio svarbą platesniame kontekste – įtraukdami ir naujus požymius gimstančius iš transformacijų, bei tų transformacijų įtaką galutiniam rezultatui.

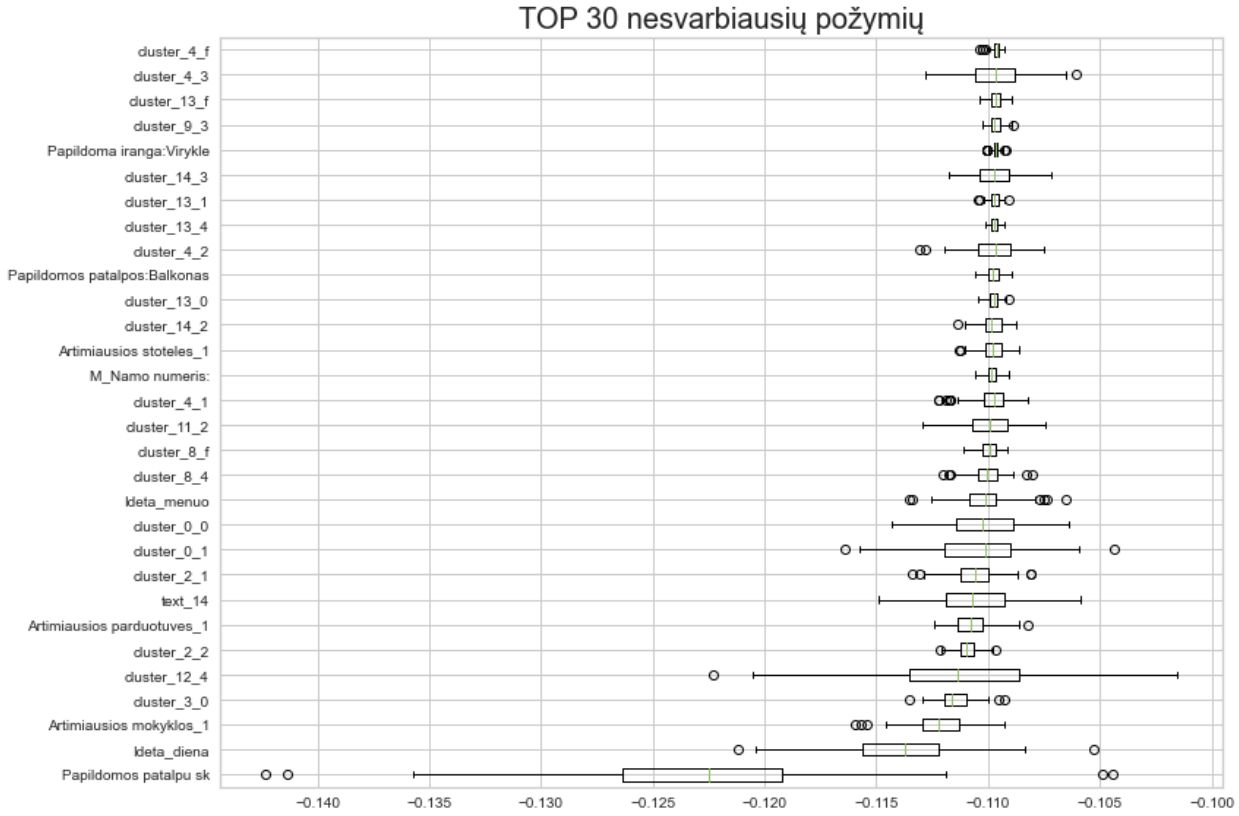
Iš viso patikra vyko 100 kartų, kas užtruko 11 valandų. Rezultate gauname panašius rezultatus kaip kad naudojant Boruta. Svarbiausias požymis yra plotas, kuris kaip matėme ankščiau taip pat tampa svarbus su kitomis tolydžiųjų kintamųjų kombinacijomis. Čia taip pat pastebime atstumą iki centro, bei platumą ir ilgumą. Iš kategorinių kintamųjų, matome, jog tarp svarbių požymių papuola šildymo tipas. Prisimindami Boruta rezultatus, centrinis kolektorinis šildymas galėtų būti asocijuojamas su naujesnės statybos butai, kurių, kaina yra didesnė. Taip pat matome, jog rajono kategorinis

kintamasis, nors ir svarbus, tačiau labai nežymiai – to priežastis galėtų būti jog tai ar butas yra senamiestyje ar ne, yra naudinga, o visi kiti rajonai tokie svarbūs nėra. Vis dėl to, nors matome jog tarp TOP 30 požymių papuola ir keletas teksto vektorių, tačiau jų įtaka bent jau geriausio modelio architektūroje yra labai minimali. Tačiau turint didesnę duomenų imtį, bei atlikus daugiau paieškos kartų – šie rezultatai mažai įtakingiems požymiams galėtų pasikeisti.




**45 pav.** TOP 30 svarbiausių požymių geriausio modelio architektūroje

Taip pat buvo atrinkti ir TOP 30 mažiausiai svarbių požymių. Visų pirma, galime pastebėti, jog daugiausia požymių yra susijusių su vaizdu „*cluster_...*“. Tai atskleidžia, jog pasirinktas metodas įtraukti vaizde esančia informaciją buvo netinkamas. Gautas neigiamas rezultatas parodo, jog atsitiktinai sumaišius konkretaus požymio reikšmes, mes galime gauti geresnius rezultatus, nei tikrosios reikšmės. Taip pat matome, artimiausios parduotuvės („Maxima“, „Iki“, ‚Lidl“ ir t.t.), mokyklos ar stotelės pavadinimas nėra svarbus, tačiau kaip žinome iš Boruta rezultatų – svarbu atstumas iki bet kurios mokyklos ar parduotuvės.




**46 pav.** TOP 30 nesvarbiausių požymių geriausio modelio architektūroje

Lyginant su literatūroje apžvelgtais sprendimais, tai nebuvo geriausiai pasiektas rezultatas vertinant pagal MAPE rezultatą, tačiau daug lemia tai, kaip pasiruošiame duomenų rinkinį ir kaip renkamės šalinti išskirtis jei jų yra, jei iš viso renkamės jas šalinti. Taip pat lyginant su kitais darbais, rezultatus pateikėme netransformuotomis regresanto vertėmis, kai tuo tarpu literatūroje, autoriai regresantą normalizuodavo. Iš vienos pusės, galime matyti jog literatūroje toks rezultatų pateikimas atrodo panašus į standartą, tačiau kita vertus, pateikiant rezultatus tikromis reikšmėmis galime lengviau suprasti tikrus modelio rezultatus.

## Išvados

1. Literatūros apžvalgoje buvo rasti įvairūs metodai, siekiantys atlikti tikslesnę NT kainos prognozę, tačiau esminiai dalykai buvo šie: vietovės informacijos išnaudojimas skaitine ar paveikslėlio išraiška, būsto nuotraukos, duomenų praturtinimas įvairia papildoma informacija. Buvo naudoti įvairūs modeliai, tačiau modelių ansambliams pavykdavo pasiekti geriausius rezultatus.
2. Informacijos iš Aruodas.lt rinkimas buvo atliktas naudojant *Selenium* įrankį. Atliekant informacijos rinkimą, svarbu atsižvelgti į svetainės, iš kurios bus renkama informacija resursus ir pagal tai sukurti etišką duomenų rinkimo strategiją. Verta paminėti, jog informacijos rinkimas Europoje vis dar yra „pilkoji zona“, todėl svarbu žinoti, ką ir kaip galima rinkti.
3. Teksto įtraukimui į regresijos modelį buvo naudotas Sent2Vec modelis, kurį naudojant sutvarkytas tekstas buvo transformuotas į 16 dydžio vektorių. Atlikus požymių atranką paaiškėjo, jog tekstas yra vienas iš požymių, kuris buvo svarbus atliekant kainos prognozę. Šiuos rezultatus patvirtino požymių atranka naudojant Boruta, bei permutacijos būdu įvertinta požymių svarba iš geriausio modelio.
4. Vaizdui įtraukimo į regresijos modelį buvo naudotas EfficientNetB0 modelis. Šis mažiausias modelis buvo pasirinktas todėl, jog modelio greitaveika (angl. *inference*) buvo vykdoma lokaliai, esant ribotiems kompiuterio resursams. Su šiuo modeliu, vaizdai buvo suspausti į 1280 dydžio vektorių, tuomet naudojant artimiausių kaimynų modelį, buvo suskirstyti į klasterius. Visos nuotraukos tuomet buvo suvidurkintos savo klasteryje, per parduodamo būsto skelbimą. Atlikus požymių atranką paaiškėjo, jog šiuo būdu įtraukti paveikslėliai nebuvo naudingi atliekant prognozes, o permutacijos būdu įvertinus požymių svarbą – šie požymiai turėjo žemiausius (neigiamus) įverčius.
5. Modelių parametrų optimizacija buvo atlikta naudojant Optuna biblioteką. Šalia modelių parametrų atrankos, taip pat buvo vertinama, ar informatyvių požymių atranka pagerina modelio prognozuojamąją galią, esant optimaliems parametrams. Analizė parodė, jog tai priklauso nuo paties modelio – vieniems modeliams tai ne tik pagreitina greitaveiką, tačiau ir pagerina rezultatus, net ir tuomet kai patys modeliai turi požymių atrankos mechanizmus patys savyje. Kitiems modeliams priešingai, geresni rezultatai buvo pasiekti naudojant visus požymius. Kita vertus, verta paminėti, jog naudojant gamyklinius / bazinius modelio parametrus – požymių atranka gali ne tik padidinti greitaveiką, tačiau ir rezultatus (modelio tikslumą). Dėl šios priežasties, siekiant greitų rezultatų praktikoje, vertėtų pradėti nuo greitų sprendimų, kurie nereikalauja eikvoti išteklių ieškant optimalių parametrų ir požymių kombinacijų, naudojant bazinius parametrus ir atrinktus požymius. Taip pat darbe pastebėta jog regresanto normalumas leidžia pasiekti geresnius rezultatus tiesiniams modeliams. Galiausiai, Optuna buvo itin naudingas įrankis, dėl Bajeso teorijos pritaikymo parametrų paieškoje, bei pateikiantis naudingos su paieška susijusios informacijos kurią galima vizualiai atvaizduoti geriau suvokiant įvairių modelio parametrų svarbą bendriems rezultatams ir visą paieškos eigą.
6. Lyginant atskirus modelius su geriausių modelių ansambliais, pasiteisino literatūroje pastebėta tendencija dėl ansamblio rezultatų pagerinimo. Dauguma ansamblio architektūrų parodė geresnius rezultatus, nei TOP 5 modeliai atskirai, tačiau visų jų rezultatai buvo beveik identiški. Vis dėl to, rezultatai nepagerėjo itin daug, tačiau greitaveikos laikas – prailgėjo pastebimai. Teoriniai / eksperimentiniai modeliai yra puikūs dėl savo rezultatų, tačiau taikant juos produkcijoje, verta atsižvelgti į jų greitaveikos laiką, ypač kai turime milijonus duomenų, bei į architektūros sudėtingumą, kai reikia paaiškinti kodėl yra prognozuojama vienaip, o ne kitaip.

Geriausias modelis prognozuojantis būsto vertę buvo modelių ansamblis atliekantis prognozių vidurkio skaičiavimą, bei ant viršaus papildomas liekanų modelis naudojantis geriausio individualaus modelio architektūrą - gradientinio pastiprinimo modelis su optimaliais parametrais. Šis sprendimas sugebėjo pasiekti mažiausias prognozavimo klaidas (MAPE=13.74, RMSE=33,307).

## Rekomendacijos

1. Požymių inžinerija yra viskas, tai yra svarbiausia veikla mašininio mokymosi procese. Verta pabandyti prijungti prie esamo duomenų rinkinio daugiau informacijos, susijusios su aplink esančiomis vietovėmis. Literatūros apžvalgoje autoriai papildomai naudojo informaciją apie darželius ir mokyklas, bei jų reitingus per miestą. Taip pat naudojo informaciją apie viešojo transporto stoteles, atstumus iki darboviečių, atstumus iki parduotuvių ir ligoninių. Verta išnagrinėti atvirus duomenis prieinamus Lietuvoje ir pagalvoti ką iš jų būtų galime atrinkti ir panaudoti.
2. Išmėginti kitas klasterizavimo ar paveikslėlių požymių segmentavimo technikas. Panašios nuotraukos buvo atrinktos naudojant centroidų giminės kaimynų modelį, tačiau nebuvo išmėgintos kitos kaimynų technikos, bei kiti metodai leidžiantys atlikti semantinį segmentavimą. Žinome, jog nuotraukose yra tokia informacija apie būsto išplanavimą, buto vidų, išorę / kiemą bei kt. Sėkmingas šių nuotraukų tipų atskyrimas, tikėtina, turėtų teikti naudos prognozėms.
3. Vaizdo vektorizavimui išmėginti kitus modelius ar aukštesnius EfficientNet variacijos. Didesni modeliai, turi daugiau parametrų, todėl jų naudojimas lokaliai, negali būti nesėkmingas turint mažai operatyviosios atminties RAM, o procesas gan lėtas turint mažai branduolių ar silpną vaizdo plokštę. Todėl pravartu būtų išmėginti debesijos sprendimus naudojant didesnius modelius, kurie ne tik leistų sėkmingai išmėginti sudėtingesnes architektūras, tačiau ir sutaupyti laiko.
4. Sekanti artima rekomendacija trečiajai būtų pasitelkti mokymosi perleidimo (angl. *transfer-learning*) bei koreguotą mokymąsi (angl. *fine-tuning*) ant jau apmokytų modelių. Kadangi modeliai įprastai yra apmokinti ant bendrinių duomenų, jie gali sunkiai susidoroti su specifine užduotimi. Kadangi modelio svoriai jau visi yra nustatyti, mes galime arba prie jau esamos architektūros pridėti naujų paslėptų sluoksnių arba koreguoti esamą architektūrą bei pridėti papildomų sluoksnių. Nuotraukas mes galime susieti su būsto kaina, arba įsiminimų skaičiumi. Tai leistų paruošti modelius leidžiančius tiksliau asocijuoti konkrečius elementus nuotraukose su būsto kokybę ir / ar verte.
5. Teksto įtraukimui į regresijos modelį yra daugiau nei vienas metodas. Darbe buvo naudotas teksto vektorizavimas, kuriam taip pat yra daugiau nei vienas būdas atlikti. Ateityje būtų galima pamėginti panaudoti transfomerius bei susieti tekstą su kaina.
6. Svarbių požymių atranka atskleidė, jog ilguma ir plokštuma yra itin svarbūs požymiai kainos prognozavimui. Dėl šios priežasties, būtų galima panaudoti lokaciją kaip vaizdo uždavinį paimant įvairias kiekvieno būsto lokacijos nuotraukas skirtingais masteliais, bei satelito ir gatvės vaizdais pažiūrint ar vaizdo apdorojimo modeliai sugebėtų susieti paveikslėliuose matomus tokius objektus kaip parduotuvės, aplinkkeliai, verslo centrai ir kita su kaina.

## Literatūros sąrašas


1. Alejandro Baldominos, I. B. (2018). Identifying Real Estate Opportunities using Machine Learning. Appl. Sci., 8(11: 2321), DOI 10.3390/app8112321.
2. Alex Kendall, Y. G. (2018). Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. CVF Conference on Computer Vision and Pattern Recognition (CVPR), (p. DOI 7482-7491. 10.1109/CVPR.2018.00781).
3. B. Case, J. C. (2004). Modeling spatial and temporal house price patterns: A comparison of four models. The Journal of Real Estate Finance and Economics, vol. 29, no. 2, 167– 191. DOI https://doi.org/10.1023/B:REAL.0000035309.60607.53.
4. Bruno Klaus de Aquino Afonso, L. C. (2019). Housing Prices Prediction with a Deep Learning and RandomForest Ensemble.
5. Byeonghwa Park, J. K. (2015). Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data. Expert Systems with Applications, 2928-2934. doi: https://doi.org/10.1016/j.eswa.2014.11.040.
6. C. Szegedy, V. V. (2016). Rethinking the Inception Architecture for Computer Vision. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (p. 2818-2826. doi: 10.1109/CVPR.2016.308.).
7. Cartmell, J. (2010). Methods to Pre-Process Training Data for K-Nearest Neighbors Algorithm,. InterDigital Communications LLC.
8. Chunhui Yuan, H. Y. (2019). Research on K-Value Selection Method of K-Means Clustering Algorithm. MDPI, 226-235. doi: https://doi.org/10.3390/j2020016.
9. Frank, E., Trigg, L., Holmes, G., & Witten, I. (2000). Technical Note: Naive Bayes for Regression. Machine Learning, 5–25. doi: https://doi.org/10.1023/A:1007670802811.
10. Gerek, I. H. (2014). House selling price assessment using two different adaptive neuro-fuzzy techniques. Automation in Construction, vol. 41, 33–39. doi: https://doi.org/10.1016/j.autcon.2014.02.002.
11. Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely Randomized Trees. Machine Learning 63, 3–42. doi: https://doi.org/10.1007/s10994-006-6226-1.
12. Guangliang Gao, Z. B. (2019). Location-Centered House Price Prediction: A Multi-Task Learning Approach. doi: https://doi.org/10.48550/arXiv.1901.01774.
13. Heaton, J. (2016). An Empirical Analysis of Feature Engineering for Predictive Modeling. SoutheastCon, 1-6. doi: 10.1109/SECON.2016.7506650.
14. Hui Zou, T. H. (2005). Regularization and Variable Selection via the Elastic Net. Journal of the Royal Statistical Society. Series B (Statistical Methodology) Vol. 67, No. 2, 301-320. doi: https://www.jstor.org/stable/3647580.
15. Ishan Misra, A. S. (2016). Cross-stitch Networks for Multi-task Learning. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (p. 3994-4003. doi: 10.1109/CVPR.2016.433.).
16. J. Han, M. K. (2013). Data Mining: Concepts and Techniques.
17. J.-M. Montero, R. M.-A. (2018). Housing price ´ prediction: parametric versus semi-parametric spatial hedonic models. Journal of Geographical Systems, vol. 20, no. 1, 27–55. doi: https://doi.org/10.1007/s10109-017-0257-y.

18. Junchi Bin, S. T. (2017). Regression model for appraisal of real estate using recurrent neural network and boosting tree. 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA), (p. 209-213. doi: 10.1109/CIAPP.2017.8167209.).
19. Leo F Isikdogan, B. V.-T. (2020). SemifreddoNets: Partially Frozen Neural Networks for Efficient Computer Vision Systems. Computer Science, doi: https://doi.org/10.1007/978-3-030-58583-9_12.
20. Li, Z. (2021). Prediction of House Price Index Based on Machine Learning Methods. 2nd International Conference on Computing and Data Science (CDS), (p. 472-476. doi: 10.1109/CDS52072.2021.00087.).
21. Matteo Pagliardini, P. G. (2018). Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (p. 528-540. doi: 10.18653/v1/N18-1049). Association for Computational Linguistics.
22. Mingxing Tan, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. International Conference on Machine Learning,, (p. doi: https://doi.org/10.48550/arXiv.1905.11946).
23. O. Russakovsky, J. D.-F. (2015). Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115, 211-252. doi: https://doi.org/10.1007/s11263-015-0816-y.
24. Oxenstierna, J. (2017). Predicting house prices using Ensemble Learning with Cluster Aggregations. ICMLC '19: Proceedings of the 2019 11th International Conference on Machine Learning and Computing, (p. 350-356. doi: 10.1145/3318299.3318377).
25. Pai, P.-F., & Wang, W.-C. (2020). Using Machine Learning Models and Actual Transaction Data for Predicting Real Estate Prices. Applied Sciences. 2020; 10(17):5832, doi: https://doi.org/10.3390/app10175832.
26. Phan, T. D. (2018). Housing Price Prediction Using Machine Learning Algorithms: The Case of Melbourne City, Australia. International Conference on Machine Learning and Data Engineering (iCMLDE), (p. 35-42. doi: 10.1109/iCMLDE.2018.00017.).
27. Quang Truong, M. N. (2020). Housing Price Prediction via Improved Machine Learning Techniques. Procedia Computer Science, 174, 433-442. doi: https://doi.org/10.1016/j.procs.2020.06.111.
28. S. Bourassa, E. C. (2010). Predicting house prices with spatial dependence: a comparison of alternative methods. Journal of Real Estate Research, vol. 32, no. 2,, 139–159. doi: 10.1080/10835547.2010.12091276.
29. S. Lu, Z. L. (2017). A hybrid regression technique for house prices prediction. IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), (p. 319-323. doi: 10.1109/IEEM.2017.8289904.).
30. Sarkar Snigdha Sarathi Das, M. E.-F.-B. (2020). Boosting House Price Predictions using Geo-Spatial Network Embedding. Data Min Knowl Disc 35, 2221–2250. doi: https://doi.org/10.1007/s10618-021-00789-x.
31. Sina Jandaghi Semnani, H. R. (2021). House Price Prediction using Satellite Imagery. Intelligent Systems with Applications, 14, doi: https://doi.org/10.1016/j.iswa.2022.200081.

32. Smola, A., & Schölkopf, B. A. (2004). Tutorial on Support Vector Regression. Statistics and Computing 14, 199–222. doi: https://doi.org/10.1023/B:STCO.0000035301.49549.88.
33. Takuya Akiba, S. S. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (p. 2623–2631. doi: https://doi.org/10.1145/3292500.3330701).
34. Tan, P.-N. S. (2005). Introduction to Data Mining.
35. Tomas Mikolov, I. S. (2013). Distributed Representations of Words and Phrases and their Compositionality. 26th International Conference on Neural Information Processing Systems - Volume 2, (p. 3111–3119).
36. Trevor Standley, A. R. (2020). Which Tasks Should Be Learned Together in Multi-task Learning? 37th International Conference on Machine Learning, 9120-9132.
37. Vasilios Plakandaras, R. G. (2011). Forecasting the U.S. Real House Price Index. Economic Modelling, Volume 45, 259-267. doi: https://doi.org/10.1016/j.econmod.2014.10.050.
38. Weisberg, S. (2001). Yeo-Johnson Power Transformations. Prieiga per: https://www.stat.umn.edu/arc/yjpower.pdf.
39. Winky K.O. Ho, B.-s. T. (2020). Predicting property prices with machine learning algorithms. Journal of Property Research, 38:1, 48-70. doi: 10.1080/09599916.2020.1832558.
40. Xiaochen Chen, L. W. (2017). House Price Prediction Using LSTM. Computer Science, Prieiga per: https://arxiv.org/pdf/1709.08432.pdf.
41. Zhao Chen, V. B.-Y. (2018). GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. Computer Science, doi: https://doi.org/10.48550/arXiv.1711.02257.
42. Zoph, B. V. (2018). Learning transferable architectures for scalable image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 8697–8710. doi: 10.1109/CVPR.2018.00907.

## Priedai

### 1 priedas. Tiesinio modelio grandinės struktūros rezultatai

| RMSE | PCA | USE TEXT | USE IMAGE | USE BOOL | SCALER |
|---|---|---|---|---|---|
| 38704 | No | No | No | Yes | MinMaxScaler |
| 38771 | No | Yes | Yes | Yes | MinMaxScaler |
| 38817 | No | Yes | Yes | No | MinMaxScaler |
| 38874 | No | No | No | Yes | passthrough |
| 38899 | No | No | No | Yes | RobustScaler |
| 38945 | No | Yes | Yes | Yes | passthrough |
| 38983 | No | Yes | Yes | Yes | RobustScaler |
| 38992 | No | No | Yes | Yes | MinMaxScaler |
| 39001 | No | Yes | Yes | No | passthrough |
| 39021 | No | No | No | Yes | StandardScaler |
| 39069 | No | Yes | Yes | No | RobustScaler |
| 39129 | No | Yes | Yes | Yes | StandardScaler |
| 39184 | No | No | Yes | Yes | passthrough |
| 39200 | No | No | Yes | Yes | RobustScaler |
| 39225 | No | Yes | Yes | No | StandardScaler |
| 39355 | No | No | Yes | Yes | StandardScaler |
| 43072 | Yes | Yes | Yes | Yes | StandardScaler |
| 43264 | Yes | No | Yes | Yes | StandardScaler |
| 43460 | Yes | Yes | Yes | No | StandardScaler |
| 43588 | Yes | Yes | Yes | Yes | MinMaxScaler |
| 43662 | Yes | No | No | Yes | StandardScaler |
| 43665 | Yes | No | Yes | Yes | MinMaxScaler |
| 43816 | Yes | Yes | Yes | No | MinMaxScaler |
| 44025 | Yes | No | No | Yes | MinMaxScaler |
| 48010 | Yes | No | No | Yes | RobustScaler |
| 63509 | Yes | No | No | Yes | passthrough |
| 93326 | Yes | Yes | Yes | Yes | passthrough |
| 93326 | Yes | Yes | Yes | No | passthrough |
| 93365 | Yes | No | Yes | Yes | passthrough |
| 93833 | Yes | No | Yes | Yes | RobustScaler |
| 93833 | Yes | Yes | Yes | Yes | RobustScaler |
| 93833 | Yes | Yes | Yes | No | RobustScaler |

## 2 priedas. Medžių modelio grandinės struktūros rezultatai

| RMSE | PCA | USE TEXT | USE IMAGE | USE BOOL | SCALER |
|---|---|---|---|---|---|
| 38292 | No | No | No | Yes | StandardScaler |
| 38550 | No | No | No | Yes | passthrough |
| 38584 | No | No | No | Yes | MinMaxScaler |
| 38694 | No | No | No | Yes | RobustScaler |
| 38706 | No | Yes | Yes | Yes | RobustScaler |
| 38790 | No | Yes | Yes | Yes | MinMaxScaler |
| 38968 | No | Yes | Yes | No | StandardScaler |
| 38970 | No | Yes | Yes | Yes | StandardScaler |
| 38974 | No | No | Yes | Yes | MinMaxScaler |
| 38984 | No | Yes | Yes | No | RobustScaler |
| 38994 | No | Yes | Yes | No | MinMaxScaler |
| 39015 | No | Yes | Yes | Yes | passthrough |
| 39081 | No | Yes | Yes | No | passthrough |
| 39088 | No | No | Yes | Yes | RobustScaler |
| 39100 | No | No | Yes | Yes | passthrough |
| 39193 | No | No | Yes | Yes | StandardScaler |
| 56682 | Yes | No | No | Yes | StandardScaler |
| 57422 | Yes | No | Yes | Yes | StandardScaler |
| 57489 | Yes | Yes | Yes | Yes | StandardScaler |
| 57644 | Yes | Yes | Yes | No | StandardScaler |
| 63384 | Yes | No | No | Yes | MinMaxScaler |
| 63998 | Yes | No | Yes | Yes | MinMaxScaler |
| 64011 | Yes | Yes | Yes | Yes | MinMaxScaler |
| 64995 | Yes | Yes | Yes | No | MinMaxScaler |
| 90307 | Yes | No | No | Yes | RobustScaler |
| 93691 | Yes | Yes | Yes | Yes | RobustScaler |
| 94078 | Yes | Yes | Yes | No | RobustScaler |
| 94102 | Yes | No | Yes | Yes | RobustScaler |
| 113939 | Yes | No | Yes | Yes | passthrough |
| 114012 | Yes | Yes | Yes | Yes | passthrough |
| 114042 | Yes | No | No | Yes | passthrough |
| 114120 | Yes | Yes | Yes | No | passthrough |

## 3 priedas. Lasso hiperparametrų atranka su GridSearch

| RMSE | INVERSE Y | ALPHA | FEATURES |
|---|---|---|---|
| 38900 | Yes | 0,001 | passthrough |
| 38933 | No | 0,0058 | passthrough |
| 38933 | No | 0,0054 | passthrough |
| 38933 | No | 0,005 | passthrough |
| 38933 | No | 0,0046 | passthrough |
| 38934 | No | 0,0042 | passthrough |
| 38934 | No | 0,0038 | passthrough |
| 38934 | No | 0,0034 | passthrough |
| 38934 | No | 0,003 | passthrough |
| 38934 | No | 0,0026 | passthrough |
| 38934 | No | 0,0022 | passthrough |
| 38934 | No | 0,0018 | passthrough |
| 38935 | No | 0,0014 | passthrough |
| 38935 | No | 0,001 | passthrough |
| 40081 | Yes | 0,0014 | passthrough |
| 40985 | Yes | 0,0018 | passthrough |
| 41767 | Yes | 0,0022 | passthrough |
| 42490 | Yes | 0,0026 | passthrough |
| 43227 | Yes | 0,003 | passthrough |
| 44000 | Yes | 0,0034 | passthrough |
| 44771 | Yes | 0,0038 | passthrough |
| 44997 | No | 0,0058 | Lasso selected |
| 44998 | No | 0,0054 | Lasso selected |
| 44998 | No | 0,005 | Lasso selected |
| 44998 | No | 0,0046 | Lasso selected |
| 44998 | No | 0,0042 | Lasso selected |
| 44998 | No | 0,0038 | Lasso selected |
| 44998 | No | 0,0034 | Lasso selected |
| 44998 | No | 0,003 | Lasso selected |
| 44998 | No | 0,0026 | Lasso selected |
| 44998 | No | 0,0022 | Lasso selected |
| 44998 | No | 0,0018 | Lasso selected |
| 44998 | No | 0,0014 | Lasso selected |
| 44998 | No | 0,001 | Lasso selected |
| 45404 | No | 0,0058 | Boruta selected |
| 45404 | No | 0,0054 | Boruta selected |
| 45404 | No | 0,005 | Boruta selected |
| 45404 | No | 0,0046 | Boruta selected |
| 45404 | No | 0,0042 | Boruta selected |
| 45404 | No | 0,0038 | Boruta selected |
| 45404 | No | 0,0034 | Boruta selected |
| 45404 | No | 0,003 | Boruta selected |

| | | | |
|---|---|---|---|
| 45404 | No | 0,0026 | Boruta selected |
| 45405 | No | 0,0022 | Boruta selected |
| 45405 | No | 0,0018 | Boruta selected |
| 45405 | No | 0,0014 | Boruta selected |
| 45405 | No | 0,001 | Boruta selected |
| 45516 | Yes | 0,0042 | passthrough |
| 46221 | Yes | 0,0046 | passthrough |
| 46914 | Yes | 0,005 | passthrough |
| 47584 | Yes | 0,0054 | passthrough |
| 48013 | Yes | 0,001 | Boruta selected |
| 48217 | Yes | 0,0014 | Boruta selected |
| 48220 | Yes | 0,0058 | passthrough |
| 48436 | Yes | 0,0018 | Boruta selected |
| 48676 | Yes | 0,0022 | Boruta selected |
| 48930 | Yes | 0,0026 | Boruta selected |
| 49109 | Yes | 0,003 | Boruta selected |
| 49319 | Yes | 0,0034 | Boruta selected |
| 49557 | Yes | 0,0038 | Boruta selected |
| 49812 | Yes | 0,0042 | Boruta selected |
| 50084 | Yes | 0,0046 | Boruta selected |
| 50156 | Yes | 0,001 | Lasso selected |
| 50350 | Yes | 0,005 | Boruta selected |
| 50560 | Yes | 0,0014 | Lasso selected |
| 50634 | Yes | 0,0054 | Boruta selected |
| 50917 | Yes | 0,0058 | Boruta selected |
| 50944 | Yes | 0,0018 | Lasso selected |
| 51310 | Yes | 0,0022 | Lasso selected |
| 51663 | Yes | 0,0026 | Lasso selected |
| 51976 | Yes | 0,003 | Lasso selected |
| 52243 | Yes | 0,0034 | Lasso selected |
| 52473 | Yes | 0,0038 | Lasso selected |
| 52718 | Yes | 0,0042 | Lasso selected |
| 52956 | Yes | 0,0046 | Lasso selected |
| 53167 | Yes | 0,005 | Lasso selected |
| 53369 | Yes | 0,0054 | Lasso selected |
| 53570 | Yes | 0,0058 | Lasso selected |

## 4 priedas. Naudotų modelių geriausi atrinkti parametrai

```
Lasso = {
    'alpha': 0.0010367002387601818,
    'random_state' : 2022}

Ridge = {
    'alpha': 19.998413489040594,
    'random_state' : 2022}

ElasticNnet = {
    'alpha': 0.20360073714615895,
    'l1_ratio': 0.8955689634607604,
    'max_iter': 10,
    'random_state' : 2022}

SVR = {
    'C': 1.7502441450129442,
    'degree': 11,
    'gamma': 0.04340306074603101,
    'kernel': 'rbf'}

SDGRegressor = {
    'alpha': 0.0036046804647723693,
    'epsilon': 0.1099863957053521,
    'learning_rate': 'adaptive',
    'penalty': 'l2',
    'random_state' : 2022}

BayesianRidge = {
    'alpha_1': 9.69262253084425e-10,
    'alpha_2': 9.76546675452083,
    'lambda_1': 2.565398857372389e-07,
    'lambda_2': 5.521044155562931e-05}

RandomForest = {
    'max_depth': 19,
    'max_features': 31,
    'min_samples_leaf': 2,
    'min_samples_split': 4,
    'n_estimators': 190}

GradientBoostingRegressor = {
    'learning_rate': 0.04778916946886645,
    'max_depth': 4,
    'max_features': 48,
    'min_samples_leaf': 12,
    'min_samples_split': 14,
    'n_estimators': 1429,
    'random_state' : 2022}

AdaBoostRegressor = {
    'learning_rate': 0.19096772966677428,
    'loss': 'exponential',
```

```
    'n_estimators': 168,
    'random_state' : 2022}

BaggingRegressor = {
    'bootstrap': False,
    'bootstrap_features': True,
    'max_features': 0.30268924575762435,
    'max_samples': 0.9121715030580635,
    'n_estimators': 107,
    'random_state' : 2022,
    'n_jobs' : -1}

ExtraTreesRegressor = {
    'bootstrap': False,
    'max_depth': 18,
    'max_features': 'auto',
    'min_samples_leaf': 2,
    'min_samples_split': 4,
    'n_estimators': 88,
    'warm_start': False,
    'random_state' : 2022,
    'n_jobs' : -1}

HistGradientBoostingRegressor = {
    'l2_regularization': 0.08828074608780355,
    'learning_rate': 0.06693946637003684,
    'max_depth': 11,
    'max_iter': 662,
    'random_state' : 2022}

XGBoost = {
    'gamma': 15.466491045811638,
    'learning_rate': 0.05472899298754795,
    'max_depth': 3,
    'min_child_weight': 9,
    'n_estimators': 892,
    'reg_alpha': 68.69391125997467,
    'reg_lambda': 0.6940127914191733,
    'subsample': 0.8287292480136365,
    'booster': 'gbtree',
    'random_state' : 2022}

CatBoost = {
    'depth': 14,
    'l2_leaf_reg': 1.5,
    'learning_rate': 0.014000000000000002,
    'min_child_samples': 32,
    'grow_policy' : 'Depthwise',
    'iterations' : 10000,
    'use_best_model' : True,
    'eval_metric' : 'RMSE',
    'od_type' : 'Iter',
    'od_wait' : 20,
```

```
    'logging_level' : 'Silent'}

LightGBM = {
    'objective' : 'regression',
    'learning_rate': 0.08116570908936283,
    'max_depth': 3,
    'min_child_weight': 1.3795097847212774,
    'n_estimators': 438,
    'num_leaves': 80,
    'reg_alpha': 9.982436436600059,
    'reg_lambda': 0.034873555232074416,
    'subsample': 0.5115940643235746,
    'random_state' : 2022}

MLPRegressor = {
    'activation': 'relu',
    'alpha': 0.06724076627787699,
    'hidden_layer_sizes': [50, 50, 50],
    'learning_rate': 'constant',
    'model__solver' : 'adam',
    'model__max_iter' : 1000}
```