UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

NAPOVEDOVANJE VREDNOSTI ČASOVNIH SERIJ – OCENJEVANJE RAZLIČNIH METOD

Ljubljana, november 2022 Andrej Bombek

IZJAVA O AVTORSTVU

Podpisani Andrej Bombek, študent Ekonomske fakultete Univerze v Ljubljani, avtor predloženega dela z naslovom Napovedovanje vrednosti časovnih serij – ocenjevanje različnih metod, pripravljenega v sodelovanju s svetovalcem doc. dr. Urošem Godnovom.

I Z J A V L J A M

1. da sem predloženo delo pripravil samostojno;

2. da je tiskana oblika predloženega dela istovetna njegovi elektronski obliki;

3. da je besedilo predloženega dela jezikovno korektno in tehnično pripravljeno v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, kar pomeni, da sem poskrbel, da so dela in mnenja drugih avtorjev oziroma avtoric, ki jih uporabljam oziroma navajam v besedilu, citirana oziroma povzeta v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani;

4. da se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Kazenskem zakoniku Republike Slovenije;

5. da se zavedam posledic, ki bi jih na osnovi predloženega dela dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom;

6. da sem pridobil vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v predloženem delu in jih v njem jasno označil;

7. da sem pri pripravi predloženega dela ravnal v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil soglasje etične komisije;

8. da soglašam, da se elektronska oblika predloženega dela uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;

9. da na Univerzo v Ljubljani neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve predloženega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja predloženega dela na voljo javnosti na svetovnem spletu preko Repozitorija Univerze v Ljubljani;

10. da hkrati z objavo predloženega dela dovoljujem objavo svojih osebnih podatkov, ki so navedeni v njem in v tej izjavi.

V Ljubljani, dne Podpis študenta:

KAZALO

KAZALO TABEL

KAZALO SLIK

KAZALO PRILOG

SEZNAM KRATIC

angl. – angleško
AMD – (angl. Advanced Micro Devices)
API – (angl. Application Programming)
ARIMA – (angl. Autoregressive integrated moving average); Avtoregresivno integrirano drseče povprečje
AutoARIMA – (angl. Auto Autoregressive integrated moving average); Avtomatsko avtoregresivno integrirano drseče povprečje
BATS – (angl. Box-Cox Transformation, ARMA residuals, Trend and Seasonality); Box-Coxova transformacija, ostanki ARMA, trend in sezonskost
BDP – Bruto državni proizvod
CUDA – angl. Compute Unified Device Architecture
CuDNN – angl. CUDA Deep Neural Network library
CV – (angl. Coefficient of variation); Koeficient variacije
GRU – (angl. Gated recurrent unit); Zaporna ponavljajoča se enota
GTX – (angl. Giga Texel Shader eXtreme)
HTML – (angl. HyperText Markup Language)
IDE – (angl. Integrated development environment); Integrirano razvojno okolje
IoT – (angl. Internet of things); Internet stvari
LightGBM – angl. Light Gradient Boosting Machine
LSTM – (angl. Long short-term memory); Dolgi kratkoročni spomin
MAE – (angl. Mean Absolute Error); Popolna absolutna napaka
MAPE – (angl. Mean absolute percentage error); Povprečna absolutna procentualna napaka
MSE – (angl. Mean squared error); Kvadratna napaka
N-BEATS – (angl. Neural basis expansion analysis for interpretable time series forecasting); Razširitvena analiza nevronske osnove za napovedovanje časovnih vrst, ki jo je mogoče interpretirati
N-HiTS – (angl. Neural Hierarchical Interpolation for Time Series Forecasting); Nevronska hierarhična interpolacija za napovedovanje časovnih vrst
NLP – (angl. Natural language processing); Obdelava naravnega jezika
PCA – (angl. Principal component analysis); Analiza glavnih komponent
PDF – (angl. Portable Document Format)
RFR - (angl. Random Forest Regressor); Naključni gozdni regresor
RF - (angl. Random Forest); Naključni gozd
RMSE – (angl. Root mean square error); Celotna napaka
RMSLE – (angl. Root Mean Squared Logarithmic Error); Koren srednja logaritmična napaka na kvadrat

RNN – (angl. Recurrent neural network); Povratna nevronska mreža
RSI – (angl. Relative strength index); Indeks relativne moči
RTX – angl. Ray Tracing Texel eXtreme
sMAPE – (angl. Synthetic Mean absolute percentage error); Simetrična povprečna absolutna odstotna napaka
StatsForecast Auto ARIMA – (angl. StatsForecast Autoregressive integrated moving average); StatsForecast avtoregresivno integrirano drseče povprečje
SULOV – (angl. Searching for Uncorrelated List of Variables)
TBATS – (angl. Trigonometric Seasonal, Box-Cox Transformation, ARMA residuals, Trend and Seasonality); Trigonometrična sezonskost, Box-Coxova transformacija, ostanki ARMA, trend in sezonskost
TCN – (angl. Temporal Convolutional Networks); Časovna konvolucijska omrežja
Deep TCN – (angl. Deep Temporal Convolutional Networks); Globoka časovna konvolucijska omrežja
TFT – (angl. Temporal Fusion Transformer); Časovni fuzijski transformator
VARIMA - (angl. Vector Autoregressive moving average); Vektorsko avtoregresivno drseče povprečje
XGBoost – (angl. eXtreme Gradient Boosting)

UVOD

Podjetja so se v poznem 19. stoletju začela spreminjati. Do takrat so bila podjetja v glavnem v lasti industrialcev, ki so imeli deleže podjetij razdeljene med razmeroma majhno število ljudi. Do neke mere so bile izjema le železnice, katerih gradnja se je financirala z obveznicami. V poznem 19. stoletju se je zgodila velika paradigma. Na tej točki so začeli industrialce zamenjavati kapitalisti. Glavna razlika med njimi je bila, da so za razliko od industrialcev, katerih glavni proizvod je bil izdelek, kapitalisti trgovali z delnicami. Delnice so osnova sodobne kapitalistične ureditve in eden izmed glavnih načinov financiranja sodobnih podjetij (Mitchell, 2008). Glede na navedeno je očitno, da je pomen delnic za podjetja ogromen. Njihova vrednost in nestanovitnost namreč vplivata tudi na odločitve, ki jih podjetja sprejmejo. To je posebej očitno ob potencialnih prevzemih manjših podjetij. Če ima podjetje npr. visoko nestanovitnost, to lahko pomeni, da obstaja veliko dvomov o prihodnosti podjetja in njegovi dejanski vrednosti. Poleg tega lahko podjetje, ki prevzema drugo podjetje, opazuje signale na trgu pred prevzemom. Če bi npr. podjetje z naznanilom namena prevzema drugega podjetja povzročilo, da temu vrednost pade, lahko podjetje vzame padec kot opozorilo trga, ki mu odsvetuje ta prevzem in je na tej točki smiselno razmisliti o prihodnjih odločitvah (Sussman, 2021; Goldstein, 2017).

Iz navedenega je očitna potreba po neki vrsti poročil s predvidevanji o prihodnjih spremembah na trgu. Verjetno si na tej točki mislimo, da bo najlažje, če poročilo oziroma napovedi preprosto kupimo od nekoga drugega, vendar pa tu pridemo do enega izmed glavnih problemov, ki ga je treba razrešiti. Večino poročil namreč napišejo ljudje. Ljudje nismo 100 % racionalni, na vsebino poročil pa vplivajo tudi dejavniki, ki niso tržni. Vsakemu, ki vsaj malo pozna delovanje trgov, je jasno, da na današnje trge vpliva preveč dejavnikov, da bi lahko posamezen udeleženec zbral vse možne podatke in se na osnovi teh odločil. Na odločitve vplivajo tudi etični dejavniki, na primer ali je neka investicija zelena, ali ima podjetje korekten odnos do zaposlenih itd. (Koehn, 2020). To spada v ekonomiko obnašanja. Kot primer bi navedli, da se veliko ljudi prezadolži, četudi se zavedajo, da to zanje ni optimalno. Prav tako nekateri iščejo informacije pri borznih posrednikih, četudi empirične študije ne potrjujejo, da je kakovost njihovih informacij veliko boljša od tistih, ki bi jih lahko dobili drugje (Etzioni, 2011). Običajno imamo enega izmed dveh problemov – premikanje s čredo ali premikanje proti čredi. Premikanje s čredo nastane takrat, ko več strokovnjakov namesto svojih lastnih poročil preprosto sledi poročilom, ki so jih naredili drugi strokovnjaki. K temu načinu se bolj nagibajo strokovnjaki, ki so imeli v prejšnjih poročilih nekaj napačnih predvidevanj, zato se rajši odločijo za »prepisovanje domačih nalog« drugih in s tem obvarujejo svoj sloves in svoj posel (Pierdzioch & Rülke, 2012). Težava običajno nastane, če želijo posamezniki namerno preprečiti premikanje s čredo ali se celo premikati proti njej. To pogosto povzroči neugodne razmere na trgu, saj poročila niso vezana na realnost, temveč na druga poročila. Podobna situacija se je pojavila tudi pri evropski bančni krizi, ko je veliko podjetij izdajalo nekoliko bolj »mila« poročila, saj so menili, da bo sicer prišlo do šoka, niso pa predvideli prihajajoče krize (Tsuchiya, 2021).

Takšen način je veliko pogostejši med akademiki. Na drugi strani je premikanje proti čredi pogostejše v bančnem sektorju. Ta sektor je namreč finančno močnejši in lahko nekomu, ki je nekajkrat pravilno napovedal gibanja, ki jih drugi niso uspeli, prinese ogromne finančne nagrade (Pierdzioch & Rülke, 2012).

Na tej točki se je treba vprašati, kaj je rešitev za podjetja, ki želijo poslovati na čim bolj natančnih podatkih. Osnovi večine sodobne ekonomije sta teorija slučajnega sprehoda in hipoteza učinkovitih trgov (Maloumian, 2022). V resničnem svetu je očitno, da trgi niso 100 % učinkoviti, saj je mogoče izkoristiti posamične neučinkovitosti, ki se dogajajo na trgih. To je bilo najbolj očitno, ko se je leta 2008 zgodila svetovna ekonomska kriza. Če te neučinkovitosti želimo izkoristiti, potrebujemo obsežne analize, posebno trgovalno infrastrukturo in zelo usposobljene ljudi. Prav zaradi potreb po specializaciji in obsežnih stroških je tak pristop rezerviran za zavarovalnice in sklade. Izmed teh so najnaprednejši skladi tveganega kapitala. Ti namreč pogosto uporabljajo tako prodajo na kratko kot prodajo na dolgo, kar jih ločuje od večine drugih udeležencev. Ti si prav tako lahko privoščijo, da kupujejo proti trgu (namreč nekaj, kar že kupujejo vsi, običajno ne raste v nedogled in je že ob nakupu precenjeno). Poleg tega imajo veliko več informacij (včasih celo kakšne interne), kar jim omogoča, da svoje pozicije odprejo že, preden določene informacije pridejo na splet in v časopise. Poleg vsega, kar je navedeno zgoraj, skladi tveganega kapitala upravljajo z veliko količino sredstev, zato lahko s svojimi nakupi in prodajami delnic vplivajo na ceno na trgu (Pedersen, 2015). Logično je, da manjši trgovci zelo težko dosegajo rezultate skladov tveganega kapitala, saj ti preprosto razpolagajo z več viri. Težava nastane predvsem v analizah potencialnih delnic. Neki sklad ima lahko več sto profesionalnih analitikov, ki se razporedijo po različnih industrijah in trgih, medtem ko so običajno navadni trgovci sami. Obdelajo lahko le omejeno količino delnic, s čimer zamudijo veliko zelo dobrih priložnosti. V primeru, da se želi trgovec vsaj približati skladom, mora svoje delo vsaj delno avtomatizirati. Avtomatizira lahko trgovanje ali analizo, v nekaterih primerih celo oboje. Običajno mu največ časa vzame analiza, kar pomeni, da se jo najbolj splača avtomatizirati. Za avtomatizacijo sicer obstaja nekaj programske opreme, vendar je za manjše trgovce predraga in premalo prilagodljiva. Avtomatizacija nam omogoča, da prihranimo dva najredkejša in najcenejša vira. To sta čas in človeški vir (Zhang, 2021).

Ena izmed najočitnejših rešitev so modeli strojnega učenja. Trgovec ima v primeru, da dela sam, razmeroma malo časa in posledično ne more pregledati vseh podatkov, ki jih ima na voljo. Izbirati mora med različnimi tipi analiz. Običajno sta to temeljna in tehnična analiza (včasih se poleg njiju uporabi tudi analiza razpoloženja). Temeljna se bolj osredotoča na širši, bolj makroekonomski pogled, tehnična pa za analizo uporablja pretekle podatke, predvsem pretekla gibanja cen in volumen delnic. Pogostejša je uporaba tehnične analize, saj je osnovana na dnevnih podatkih, kar pomeni, da jo je lažje uporabiti s tehnikami strojnega učenja (Liu, 2017). Pogosto se uporablja tudi analiza razpoloženja, ki jo je prav tako mogoče razdeliti na dneve. Ta se dobro ujema s podatki o vrednosti cen delnic, ki so praviloma dnevni. Težave so pogostejše pri temeljni analizi, saj so njeni kazalniki izračunani

za daljše obdobje (običajno najmanj tri mesece). To posledično pomeni, da je ne moremo neposredno uporabiti z dnevnimi podatki. Zato smo se odločili za uporabo le tehnične analize, saj je najlažja za izvedbo, hkrati pa daje najbolj objektivne rezultate.

V magistrskem delu smo želeli raziskati, kateri tipi delnic so primerni za analizo (različni trgi), katero časovno obdobje je najprimernejše (odvisno je tudi od tipa analize) in katere značilnice je smiselno izbrati.

Namen je raziskati možnosti, kako bi lahko to analizo avtomatizirali in s tem naredili dostopnejšo za potencialne uporabnike. Za dosega tega cilja smo uporabili programski jezik Python, s katerim smo uporabili več modelov strojnega učenja in proučili njihove rezultate. Za zaključek smo te modele še ovrednotili in primerjali med sabo.

Modeli strojnega učenja bi lahko pripomogli tudi k izboljšanju učinkovitosti trgov. Večina skladov in drugih velikih udeležencev svoj denar služi z izkoriščanjem posameznih neučinkovitosti. To posledično pomeni, da manjši udeleženci na trgu izgubijo, saj nimajo dovolj virov, da bi lahko neučinkovitosti pravočasno predvideli. S kakovostnejšimi informacijami bi tudi manjši udeleženci lažje predvideli neučinkovitosti na trgu, njihovi zaslužki bi se povečali, posledično bi se zaslužki velikih udeležencev zmanjšali, kar bi bistveno izboljšalo učinkovitost trgov.

Cilj magistrskega dela je proučitev literature o časovnih serijah, ki bodo predstavljale osnovo za naše delo. Prav tako smo proučili literaturo o predvidevanju prihodnjih cen delnic in modelov, ki jih za to lahko uporabimo. S tem smo poskusili zmanjšati možnost, da bi ponavljali napake drugih. Poleg tega smo določili nekaj najpomembnejših parametrov, kot so primerne delnice, primerno časovno obdobje za učenje modela in primerno časovno obdobje za učenje na predhodnih podatkih. V literaturi smo pregledali tudi potencialne modele, na osnovi katerih smo potem izvedli strojno učenje. Ko smo to opravili, smo več modelov primerjali in na osnovi rezultatov na preteklih in prihodnjih podatkih določili najboljši model ob danih specifikacijah. Za ta namen smo si najprej zastavili tri raziskovalna vprašanja, ki smo jih kasneje dopolnili še z dvema.

V magistrskem delu smo uporabili več analitičnih metod. Delo obsega teoretični in praktični del. V teoretičnem delu smo uporabili metodi deskripcije in povzemanja, saj smo tu povzeli teoretično ozadje dela s časovnimi serijami, predvidevanja cen delnic, modelov strojnega učenja in drugih teoretičnih izhodiščih. Drugi del je vsebinsko bolj praktičen in eksperimentalen. V tem delu smo izvedli uvoz podatkov, njihovo čiščenje in urejanje, izbiro značilnic in uporabo več različnih modelov strojnega učenja, ki smo jih obrazložili v prvem delu. Zatem sledita dve vrsti simulacije teh modelov. Prvi model je namenjen testiranju na preteklih podatkih, s čimer je postavljeno izhodišče za drugi model, ki temelji na testiranju trenutnih podatkov. Ti omogočajo napoved za prihodnje obdobje.

V fazi priprave na magistrsko delo smo pregledali kar nekaj znanstvenih člankov in knjig ter ugotovili pomanjkljivost, ki je bila skupna skoraj vsem. Velika večina se je namreč nanašala

na primerjavo napovedi za le eno delnico, v enem časovnem obdobju in z le dvema ali tremi metodami. To onemogoča primerjavo več sistemov med seboj, saj ima vsak članek drugačne parametre in jih posledično ni mogoče primerjati. Poleg tega je bilo razmeroma malo člankov, kjer so bili uporabljeni novejši modeli, še manj pa je bilo literature o teh modelih. Zaradi teh omejitev smo se odločili, da želimo poskusiti čim več različnih algoritmov in njihovih parametrov, da bomo na osnovi tega lahko dobili boljše rezultate.

Podatki, s katerimi pogosto delajo podatkovni znanstveniki, običajno spadajo v enega izmed naslednjih treh tipov: presečni podatki, časovne serije in panelni podatki. V magistrskem delu so vsi podatki časovne serije. Podatki so preneseni iz Yahoo Finance z uporabo Python knjižice yfinance. Razdeljeni so na stolpce, kjer en stolpec predstavlja datum, drugi pa vsebujejo podatke o ceni od odprtja borze, ceni ob zaprtju borze, najvišji ceni, najnižji ceni in volumnu (včasih pa tudi podatek o izplačani dividendi na tisti dan) (Pal & Prakash, 2017).

Strojno učenje je mogoče uporabiti za več različnih nalog. Med njimi so najpogostejše: klasifikacija, regresija, razvrščanje v skupine, zmanjšanje dimenzij, napake, optimizacija, linearno programiranje, modeli in lastnosti. V magistrskem delu je uporabljena regresija, saj nas ne zanimajo razredi, temveč dejanske vrednosti delnic. Pogosta je linearna regresija, kjer lahko vse podatke združimo v eno črto. Primer je veliko bolj zapleten, saj se cene delnic ne gibljejo linearno. Poleg tega smo morali zmanjšati tudi dimenzije, saj obstaja vsaj 200 različnih lastnosti za vsako delnico. Če bi podatke uporabili na tak način, bi imeli verjetno težave s sistemskimi viri (model bi bil zahtevnejši) in prekomernim prilagajanjem, kar bi model naredilo neuporaben za katero koli drugo delnico (Julian, 2016).

Uporabljali smo več tipov strojnega učenja, med katerimi je bilo najbolj napredno globoko učenje. Njegova glavna prednost za naš namen je, da omogoča analizo multivariatnih podatkov. Ta tehnologija je postala primerna v zadnjih nekaj letih, ko se je znižala cena računalniških virov. Poleg tega se je zvišala tudi količina podatkov, s katerimi lahko delamo. Ogromno literature je bilo napisane v zadnjih nekaj letih, predvsem s področja napovedovanja vrednosti časovnih serij. Pri teh so posebej učinkoviti nekateri modeli nevronskih mrež s povratno zanko (angl. Recurrent neural network, v nadaljevanju RNN), med katerimi je zelo pogost dolgi kratkoročni spomin (angl. Long short-term memory, v nadaljevanju LSTM) (Consoli in drugi, 2021).

Med tehnikami globokega učenja smo se odločili za tri, ki so najprimernejše in omogočajo najboljšo analizo. Izbrali smo take, ki se med sabo razlikujejo po času nastanka in zapletenosti. Začne se z uporabo tipa RNN modela, bolj specifično LSTM. LSTM je zelo primeren za delo s časovnimi serijami, specifično za obdelavo naravnega jezika (angl. Natural language processing, v nadaljevanju NLP), analizo razpoloženja in finančne analize. Ta model je sicer nekoliko starejši, a še vedno zelo pogosto uporabljen (Sezer, Gudelek & Ozbayoglu, 2020). Drugi tip izbranega modela je razširitvena analiza nevronske osnove za napovedovanje časovnih vrst, ki jo je mogoče interpretirati (angl. Neural basis expansion analysis for interpretable time series forecasting, v nadaljevanju N-BEATS). Je dokaj nov

model, saj je bil prvič omenjen leta 2019. Njegova glavna prednost je relativna preprostost v primerjavi s sorodnimi modeli. Navkljub preprosti zasnovi je na M4 tekmovanju dosegel izjemne rezultate (Oreshkin, Carpo, Chapados & Bengio, 2019). Tretji tip modela je Prophet. Je model, ki ga je razvilo podjetje Facebook. Praviloma omogoča kakovostno analizo, a ima vseeno kar nekaj omejitev (Garlapati in drugi, 2021). Četrti in najbolj zapleten tip modela so transformerji. Tako kot prejšnja tipa modelov so tudi transformerji dokaj mlad pristop k delu s časovnimi serijami. Osnova za delo s transformerji je bila postavljena že leta 2017. Za razliko od nekaterih drugih modelov transformerji omogočajo zelo dobro vzporedno izvajanje, ki omogoča učinkovitejšo rabo strojne opreme, predvsem grafičnih kartic. To se odraža v veliko manj časa, potrebnega za učenje. Posebej so učinkoviti za NLP, čeprav jih je mogoče uporabljati tudi pri delu s časovnimi serijami (Vaswani in drugi, 2017).

Poleg teh modelov smo se odločili še za uporabo nekaj drugih modelov, da smo lahko naredili bolj relevantno primerjavo. Dva od teh spadata med modele globokega učenja, in sicer časovna konvolucijska omrežja (angl. Temporal Convolutional Networks, v nadaljevanju TCN) ter časovni fuzijski transformator (angl. Temporal Fusion Transformer, v nadaljevanju TFT). Vsi modeli, ki smo jih izvedli, niso spadali med modele globokega učenja. Mednje spadajo avtoregresivno integrirano drseče povprečje (angl. Autoregressive integrated moving average, v nadaljevanju ARIMA), StatsForecast avtoregresivno integrirano drseče povprečje (angl. StatsForecast Autoregressive integrated moving average, v nadaljevanju StatsForecast Auto ARIMA) in ExponentialSmoothing. S pomočjo teh smo ocenili, ali je uporaba zapletenejših in bolj potratnih modelov globokega učenja sploh potrebna.

1 TEMELJNI POJMI

Pred začetkom praktičnega dela je treba obrazložiti nekaj temeljnih pojmov, s katerimi smo se srečali v tej magistrskem delu. Ti so pomembni, saj smo na njihovi osnovi izbirali programsko opremo, ki je bila uporabljena v praktičnem delu magistrskega dela.

1.1 Časovne serije

V magistrskem delu smo delali s cenami delnic. Te so skoraj vedno zabeležene v obliki časovnih serij. Delniški indeksi so namreč sekvenčne narave (napisane so cene po dnevih). Časovne serije so v bistvu serija preteklih opažanj v konstantnem časovnem intervalu (dnevi, tedni, meseci itd.) po časovnem zaporedju. Zgodovina časovnih serij sega že v deseto stoletje, saj so takrat nastali prvi diagrami, ki so prikazovali razmerje med časom in naklonom več planetov. Več astroloških časovnih serij je sledilo še v šestnajstem in sedemnajstem stoletju. Prve nekoliko sodobnejše časovne serije so sicer nastale komaj v 19. stoletju z deli Benjamina Gompertza (Gompertzova krivulja) in Pierra Francoisa (logistična krivulja). Celotno področje ekonometrije je prav tako nastalo na osnovi dela s časovnimi

serijami. Danes seveda časovne serije ne obsegajo celotne ekonometrije (Guo, Li, Gao & Yang, 2022; Swamidass, 2000; Dodge, 2008).

Časovne serije v primeru delnic niso skoraj nikoli linearne in stacionarne. Stacionarni podatki so tisti, ki se ne spreminjajo s časom. Obstaja več vrst stacionarnosti:

- Stroga stacionarnost – se ne spreminja. Distribucija je ves čas enaka.
- Stacionarnost prve stopnje – serija ima konstantno povprečje, ki se ne spreminja, medtem ko so lahko spreminjajo druge statistične značilnosti.
- Šibka stacionarnost – povprečje, varianca in kovarianca so konstantne čez celotno časovno serijo.
- Stacionarnost trenda – stacionarnost je odvisna od trendov.
- Stacionarnost razlik – ima več razlik v časovnih serijah.

Časovne serije so sicer sestavljene iz sekularnega trenda, sezonskih variacij, cikličnih nihanj in iregularnih variacij.

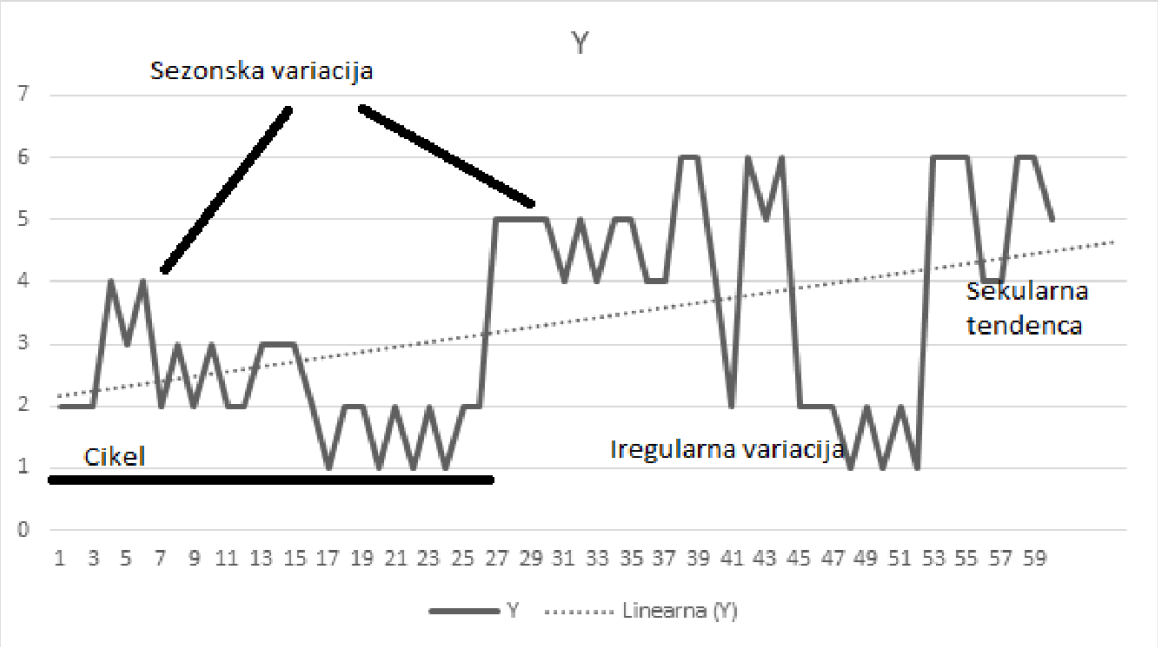
Sekularni trend nam prikaže, ali je trend časovnih serij naraščajoč, padajoč ali stabilen. Ta trend je sicer lahko tudi linearen. V primeru, da ta pogoj ni izpolnjen, moramo izbrati funkcijo, ki je najprimernejša za naše opazovanje.

Iregularne variacije so komponenta, ki je posebej dobro vidna v primeru cen delnic. Iregularne variacije se namreč pojavijo nepričakovano, naključno in jih praviloma ni mogoče predvideti. Vse, česar ne moremo všteti med sekularno tendenco, sezonsko variacijo ali ciklično nihanje, se šteje pod iregularne variacije.

Ciklična nihanja so termin, ki opiše oscilacije, ki se zgodijo skozi daljša časovna obdobja v časovnih serijah.

Zadnja komponenta časovnih serij je sezonska variacija. Ta zaobsega variacije, ki so posledica klimatskih razmer, navad populacije ali religijskih običajev. Med te variacije bi lahko šteli praznike, povečanje porabe goriva pozimi za ogrevanje itd. (Vishwas & Patel, 2020; Dodge, 2008).

Slika 1: Prikaz časovnih serij



Prirejeno po Dodge (2008).

V ekonometriji je veliko različnih načinov uporabe časovnih serij. Razloga za analizo časovnih serij sta običajno dva. Prvi je izboljševanje razumevanja dejavnikov, ki vplivajo na določeno časovno serijo, drugi pa je napovedovanje prihodnjih vrednosti (te so lahko napovedovanje bruto domačega proizvoda (v nadaljevanju BDP), vrednosti prodaje, cen delnic itd.) (Pal & Prakash, 2017).

Časovne serije so pogosto predmet predvidevanja. V bistvu je predvidevanje časovnih serij ena izmed starejših analitičnih tehnik. Osnovna ideja predvidevanja časovnih serij je, da na osnovi preteklih informacij naredimo predvidevanja o verjetnih prihodnjih vrednostih. Področje predvidevanja časovnih serij je sicer še zelo živo, saj so novi koncepti, kot je internet stvari (angl. Internet of things - IoT), bistveno zvišali količine podatkov, ki jih imamo na voljo. Za predvidevanje časovnih serij namreč običajno potrebujemo več sto ali celo več tisoč vrednosti, kar je v marsikateri aplikaciji zelo težko dobiti. V zadnjih letih je bilo razvitih tudi veliko novih algoritmov za napovedovanje časovnih serij. To je področje, ki zanima tudi velika podjetja, saj tudi sama razvijajo svoje algoritme (FB Prophet je razvilo podjetje Facebooka oziroma zdaj META) (Kotu & Deshpande, 2018; Schintler & McNeely, 2019).

1.2 Deljenje serij glede na dimenzije

Časovne serije običajno po dimenzijah delimo na univariatne in multivariatne. Veliko algoritmov je narejenih tako, da nam omogočajo delo z obema tipoma časovnih serij. Skoraj vsi modeli, ki podpirajo multivariatne časovne serije, podpirajo tudi univariatne, obratno pa

redkeje. Delo z univariatnimi časovnimi serijami je preprostejše kot delo z multivariatnimi, saj imamo pri univariatnih le eno opazovanje, ki je večkrat sekvenčno zapisano skozi čas. Delo z univariatnimi časovnimi serijami je lažje kot z multivariatnimi, prav tako zahteva veliko manj procesorske moči. To je zelo pomemben dejavnik, saj so nekateri zapletenejši algoritmi globokega učenja zelo zahtevni. V primeru napovedovanja cen delnic je opazovanje, ki ga uporabljamo in napovedujemo, zaključni tečaj. Primeri algoritmov, ki so izključno univariatni, so: ARIMA, avtomatsko avtoregresivno integrirano drseče povprečje (angl. Auto Autoregressive integrated moving average, v nadaljevanju AutoARIMA), eksponentno glajenje, Box-Coxova transformacija, ostanki ARMA, trend in sezonskost (angl. Box-Cox Transformation, ARMA residuals, Trend and Seasonality, v nadaljevanju BATS), trigonometrična sezonskost, Box-Coxova transformacija, ostanki ARMA, trend in sezonskost (angl. Trigonometric Seasonal, Box-Cox Transformation, ARMA residuals, Trend and Seasonality, v nadaljevanju TBATS), Theta, FourThera, Prophet, Fast Fourier Transform in številni drugi. Multivariatne časovne serije so bistveno bolj zapletene, saj vsebujejo več različnih opazovanj. Zelo uporabne so pri uporabi zapletenejših algoritmov, ki običajno bolje izkoristijo multivariatne časovne serije in posledično večjo količino podatkov, ki nam jih dajejo. Ko govorimo o napovedovanju cen delnic, se moramo zavedati, da zaključni tečaj ni edini dejavnik, ki vpliva na ceno delnice. Vplivajo tudi volumen, drseča povprečja in še veliko drugega. Z uporabo več opazovanj so ti modeli bistveno bolj zapleteni, a velikokrat dajejo boljše rezultate, saj zaobsegajo večje število dejavnikov. Največji problem pri raziskavi so računalniški viri, saj vključevanje dodatnih opazovanj bistveno zviša število računov, ki jih mora na osnovi algoritma računalnik izvesti. Z multivariatnimi serijami lahko vidimo tudi medserijske povezave, ki jih z univariatnimi nismo mogli. S temi lahko npr. opazujemo učinke drugih davkov na rast podjetij v neki državi. Primeri takšnih modelov so: vektorsko avtoregresivno drseče povprečje (angl. Vector Autoregressive moving average, v nadaljevanju VARIMA), naključni gozd (angl. Random Forest, v nadaljevanju RF), RNN, TCN, N-BEATS, TFT, transformer in številni drugi (Unit8, 2022c; Jansen, 2018; Vishwas & Patel, 2020; Schintler & McNeely, 2019; Kuber, Yadov & Yadov, 2022).

Na osnovi zgoraj napisanega je očitno, da se rezultati, dobljeni z uporabo univariatnih in multivariatnih časovnih serij, bistveno razlikujejo, četudi bi pri obeh uporabili isti algoritem. V nekaterih primerih so razlike ogromne, kot na primer pri predvidevanju BDP. Če na primer naredimo avtoregresijo univariatne časovne serije BDP, bo naš rezultat dokaj slab. Če pa uporabimo multivariaten postopek, bo naš rezultat blizu porabi, pomnoženi s povprečjem BDP/poraba (Cochrane, 1990).

1.3 Deljenje serij glede na število vzorcev

Obstaja več vrst časovnih serij, deljenih po številu vzorcev. Delimo jih na deterministične in probabilistične oziroma stohastične. Deterministični prikazi so pri napovedovanju vrednosti časovnih serij redkejši, saj nam prikažejo linearno črto ali krivuljo. To je pri časovnih serijah,

sploh pa pri vrednostih delnic, dokaj neuporabno. Deterministični modeli so sicer uporabni za eksperimentalne podatke, a ne ekonomske, saj ti vsebujejo preveč naključnosti. Zaradi tega smo v vseh primerih uporabili probabilistične oziroma stohastične. Ti podatki so namreč zelo nestanovitni, nanje pa vplivajo zunanji dejavniki. V knjižnici Darts, s katero smo delali, imamo na voljo deterministični prikaz pri skoraj vseh modelih (Unit8, 2021; Bell, 2016; Sharma, Sachdeva & Aggarwal, 2021; Herzen in drugi, 2022; Dodge, 2008).

1.4 Tipi analiz v predvidevanju cen delnic

Ob poskušanju predvidevanja delnic z multivariatnimi časovnimi serijami te večkrat vsebujejo tudi podatke iz analiz. V osnovi obstaja veliko tipov analiz, a se za napovedovanje cen delnic običajno uporabljajo le tri. To so tehnična, temeljna in analiza razpoloženja. Najpogostejši sta tehnična in temeljna, saj sta v tem kontekstu uporabljeni dalj časa kot analiza razpoloženja. Prvi dve sta se namreč pojavili že v šestdesetih letih (Melnick & Everitt, 2008; Gupta & Chen, 2020).

Pri vseh tipih analiz sicer naletimo na problem hipoteze učinkovitega trga, ki predpostavlja, da je trg učinkovit in da z uporabo katere koli vrste analiz ne bi dobili boljših rezultatov, kot če bi uporabili le dnevne podatke. Ta teorija, ki je bila sicer razvita v šestdesetih letih, le delno drži, saj imajo današnji trgi vseeno nekaj neučinkovitosti, ki jih lahko izrabimo za dosego višjih dobičkov. Možnost napovedovanja cen na delniških trgih je v zadnjih letih pokazalo več raziskav. Navkljub tej možnosti je izvedba veliko težje izvedljiva, saj le redko kdo uspe »premagati trg« in doseči višje donose, ki jih ima npr. indeks na izbranem trgu (Mallikarjuna & Rao, 2019; Melnick & Everitt, 2008; Sewell, 2007).

Prva, ki smo jo omenili, je tehnična analiza, ki predpostavlja, da so vsi temeljni dejavniki že vračunani v ceno. Tako se predpostavlja, da je eden izmed glavnih načinov, kako pridobiti prednost na trgu, z uporabo tehničnih kazalcev. Tehnično analizo izvedemo na osnovi analize cene in obsega trgovanja. S prej omenjenimi podatki naredimo kazalnike in svečke. Kazalnikov je ogromno, a praviloma vsi delujejo na osnovi cene (tržne cene ob koncu dneva, cene ob odprtju trga, najvišje cene trgovskega dneva in najnižje cene trgovskega dneva) in obsega trgovanja. Obstaja več različnih tipov kazalnikov, ki so kasneje omenjeni pri knjižnici za dodajanje tehničnih indikatorjev. Vsak kazalnik je izračunan na osnovi prej omenjenih podatkov. Izračunajo se z že prej pripravljenimi formulami.

Svečke za razliko od kazalnikov niso izračunane, temveč so samo grafičen prikaz tržne cene ob koncu dneva, cene ob odprtju trga, najvišje cene trgovskega dneva in najnižje cene trgovskega dneva. Svečke so grafični prikaz gibanja cen, ki izvirajo iz Japonske. Prvič so se pojavile že v osemnajstem stoletju, ko so bile uporabljene za prikaz cen riža. Posebnost sveč je, da s svojo obliko kažejo čustva trgovcev, saj ta lahko bistveno vplivajo na ceno. Praviloma so uporabne predvsem za krajše časovno obdobje. Trenutno so najpogostejši način izrisa na trgovinskih grafih (z izjemo črtnega grafikona), kjer so v glavnem zamenjali stolpčne grafikone. Uporabljajo se na dva različna načina. Prvi je, da pogledamo obliko

svečke in na osnovi tega predvidimo prihodnje trende. To je mogoče predvsem v primeru svečk Hammer oziroma kladivo in Shooting star oziroma utrinkov. Prve imajo ozek trup in zelo velik rep, druge pa imajo zelo ozek trup in zelo veliko glavo. Obe nakazujeta hitre spremembe v trendih. Drugi način je uporaba več svečk v vzorcu. Na osnovi dveh ali več svečk je mogoče predvidevati prihodnja gibanja, a to zahteva veliko znanja in je razmeroma težavno. Ena izmed največjih prednosti tehnične analize je, da jo je mogoče razmeroma preprosto uporabiti v strojnem učenju. Zaradi tega je ta postopek analize postal priljubljen med kvantitativnimi analisti in raziskovalci. Praviloma se uporablja za krajša časovna obdobja kot pri temeljni analizi (Das, Sahu & Janghel, 2021; Liu, 2017; Bell, 2016; Melnick & Everitt, 2008; Petrusheva & Jordanoski, 2016).

Slika 2: Primer svečk

Vir: Jasper (2020).

Drug tip analize je temeljna analiza. Deli se na tri dele, in sicer na analizo gospodarstva, analizo panoge in analizo podjetja. V primeru analize podjetja se osredotočimo na dejavnike, kot so prodaja, dobičkonosnost, kosmati denarni tok, dobiček na delnico in številni drugi. Izdelava tega tipa analize lahko predstavlja velik izziv, saj so podatki, ki so potrebni za ta tip analize, velikokrat težje dostopni. Na voljo imamo sicer veliko količino podatkov na spletu, a ti niso vedno najbolj zanesljivi. Temeljna analiza kot takšna temelji na metodah, ki jih je razmeroma težko modelirati. Prav tako podjetja poročila objavljajo na vsake tri mesece, kar pomeni, da imamo malo podatkov v primerjavi s tehnično analizo, kjer so podatki dnevni, urni ali celo minutni. Zaradi tega je ta tip analize primeren za daljša (običajno večletna) obdobja. Navkljub temu je ta tip analize še kako vreden za investitorje, ki s tem tipom analize odkrivajo precenjene in podcenjene delnice. Prav tako nam ta tip analize

omogoča iskanje dolgoročnih trendov na osnovi ekonomije, demografije, tehnologije in potrošniških trendov. Iz te vrste analize se sicer lahko naučimo veliko o podjetju. Ugotovimo lahko namreč, kaj so glavni ustvarjalci prihodkov in dobičkov v podjetju, kar nam omogoča, da lažje ocenimo tveganja, ki jim je podjetje izpostavljeno. Prav tako nam omogoča iskanje glavnih ustvarjalcev vrednosti v podjetju. Največja prednost pa je, da nas uporaba tega tipa analize prisili v širše raziskovanje, ne le podjetja, temveč tudi celotne industrije. Pri tehnični analizi nam namreč ni treba vedeti česarkoli o podjetju ali industriji, a jo lahko vseeno izvedemo, medtem ko pri temeljni ni tako. Ena izmed glavnih omejitev tega tipa pa je subjektivnost analitika, saj velikega dela pomembnih dejavnikov ni mogoče izmeriti ali izračunati. Zaradi tega razloga smo se tudi odločili, da tega tipa analize ne vključimo v praktični del magistrskega dela (Zhao & Chen, 2021; Liu, 2017; Jardine, 2008; Petrusheva & Jordanoski, 2016).

Analiza razpoloženja se je v predvidevanju cen delnic začela uveljavljati veliko kasneje, saj v veliki meri zahteva več tehnične opreme. V zadnjih letih se je ta metoda začela pogosteje uporabljati, zaradi česar so nastale tudi spletne finančne platforme, ki hranijo te podatke in nam omogočajo njihovo analizo, kot na primer StockTwits. Prav tako se je zvišalo število večjih finančnih institucij, ki si pri svojih investicijskih odločitvah pomagajo z analizo razpoloženja. Metoda je postala uporabna predvsem po uveljavitvi družbenih omrežij, saj na osnovi člankov težko izmerimo mnenje (praviloma so članki dokaj objektivni). Deluje tako, da zajamemo mnenja v obliki besedila in poskušamo prepoznati, ali je odnos pisca do neke teme pozitiven, negativen ali nevtralen. Ta odnos posameznika do teme opredelimo z oceno razpoloženja. Praviloma obstajata dva načina, kako izvesti analizo razpoloženja. Prva metoda je uporaba leksikonov. Ti so statične liste, na osnovi katerih lahko predvidimo razpoloženje na osnovi besedila. Težava je statičnost teh list, kar pomeni, da ima ta metoda bistvene probleme v primeru novih besed ali fraz. Druga, zdaj skoraj pogostejša metoda, je strojno učenje. Ta metoda je sodobnejša in velikokrat daje boljše rezultate, saj je osnovana na modelih strojnega učenja, kar ji daje več fleksibilnosti kot leksikon, ki je statičen. Tako kot temeljna analiza je tudi analiza razpoloženja bolj subjektivna in kot takšna manj primerna za uporabo v magistrskem delu (Gupta & Chen, 2020; Makrehchi, Shah & Liao, 2013; Bapat, 2014; Rao & Srivastava, 2012; Mudinas, Zhang & Levene, 2019).

2 UPORABLJENA ORODJA IN KNJIŽNICE

2.1 Orodja

Kar se tiče tehnične narave praktičnega dela magistrskega dela, smo potrebovali več različnih vrst strojne opreme. Vsa orodja so bila izbrana s predpogojem medsebojne združljive. Vsa so bila narejena za programski jezik Python ali pa so izvedenke tistih, ki so bila narejena za kakšen drug programski jezik in kasneje prilagojena za Python.

Obstaja veliko število orodij, ki omogočajo napovedovanje časovnih serij. Nekaj, kot na primer RapidMiner, je programov, ki nam omogočajo delo z grafičnim vmesnikom (omogoča tudi nekaj malega programiranja v Pythonu) in programske jezike, kot sta R in Python, ki jih uporabljamo v okoljih, kot sta Jupyter Notebook za Python ali R studio za R. Pri izbiri orodij je bilo zastavljenih nekaj ključnih parametrov, ki jih morajo ta orodja izpolnjevati:

- možnost dela z najnovejšimi algoritmi,
- podpora čim več knjižnic,
- možnost uporabe orodja Compute Unified Device Architecture (v nadaljevanju CUDA),
- možnost izvedbe celotnega postopka v tem orodju (od prenosa podatkov do rezultatov),
- čim več možnosti za prilagajanje,
- možnosti za teste na preteklih podatkih in napovedi za prihodnje obdobje.

Zaradi zastavljenih zahtev glede podpore knjižnic, možnosti izvedbe celotnega postopka v orodju in možnosti za prilagajanje je večina programov z grafičnim vmesnikom neprimerna. Tako so nam ostali programski jeziki, predvsem R in Python. Odločili smo se za Python. Razlogov za to odločitev je veliko. Programski jezik Python ima zelo dobro razmerje med zmogljivostjo in preprostostjo uporabe. Nižje-stopenjski programski jeziki so zmogljivejši, a tudi bolj zapleteni, z veliko manj podpornimi knjižnicami (Raschka, 2015).

Omogoča tudi izvedbo vseh podpornih nalog, kot so prenos podatkov, urejanje podatkov, grajenje modela, ugotavljanje veljavnosti modela in upodobitev rezultatov (Pal & Prakash, 2017).

Python in R sta sicer odprtokodna. Uporabljati ju je mogoče na računalnikih z različnimi operacijskimi sistemi (Ozgur, Colliau, Rogers & Hughes, 2017). Če je mogoče prejšnje točke pripisati obema, je v zadnjih dveh točkah Python definitivno v prednosti pred R. Prva točka so knjižnice. Python ima namreč ogromno knjižnic. Te so praviloma bolje podprte kot R-jeve in pogosteje posodobljene. Tako Python omogoča delo z najnovejšimi algoritmi, ki v drugih programskih jezikih zahtevajo lastno uporabo (Schintler & McNeely, 2019).

Druga točka, kjer je Python boljši, je prilagodljivost. R je namreč specializiran za manj različnih nalog, medtem ko Python s svojimi knjižnicami omogoča delo s skoraj katerimi koli podatki, za kakršno koli analizo, na kakršen koli način (lahko nastavljamo obliko posameznih objektov) (Ozgur, Colliau, Rogers & Hughes, 2017).

Po izbranem programskem jeziku je bilo treba izbrati še integrirano razvojno okolje (angl. Integrated development environment - IDE), v katerem se bo programiralo. Že v študijskem okolju smo delali z delovnimi zvezki, zato se je zdela prava izbira okolje, ki delovne zvezke prav tako podpira. Delovni zvezki imajo dobro lastnost, da omogočijo »razbiranje« kode na več manjših delov, kar pomaga pri odpravljanju napak. Poskusili smo več orodij, od katerih so bila nekatera lažja za uporabo kot druga. Prvo orodje, ki smo ga preizkusili, je bil Google

Colab. To je sicer zelo dobro in preprosto orodje, ki nam med drugim omogoča uporabo tudi Googlovih strojnih virov, kot so grafične procesne enote in tenzorske procesne enote. Zanj se nismo odločili, saj smo imeli že v preteklih preizkusih uporabe težave z nekaj knjižnicami, prav tako pa ne omogočajo uporabe lastne strojne opreme, ki je močnejša od opreme, ki jih ponuja Google Colab v brezplačni različici (Google, 2022). Poleg tega smo preizkusili tudi DataSpell (s študentsko licenco) in Visual Studio Code, a nam nobeden od njiju ni bil najbolj všeč (JetBrains, 2022; Microsoft, 2022). Zato smo se odločili za uporabo Jupyter Notebook, s katerim smo v preteklosti že delali (Jupyter, 2022). Najbolj všečno pri tem je, da je bila izvedenka del orodja Anaconda, kar nam je omogočilo tudi veliko lažje nalaganje orodja CUDA, ki smo ga potrebovali za uporabo grafičnega pospeševanja.

CUDA se bistveno razlikuje od vsega drugega, kar je opisano v magistrskem delu, saj je platforma programske in strojne opreme (vse drugo je izključno programsko). Programski del je NVIDIA CUDA Toolkit, strojni pa grafična kartica znamke NVIDIA, ki ima CUDA jedra. CUDA Toolkit je sicer sestavljen še iz več različnih delov, in sicer samega sebe, gonilnikov, 150 primerov z dokumentacijo in vse preostale dokumentacije v PDF (angl. Portable Document Format) in HTML (angl. HyperText Markup Language) formatih. CUDA Toolkit vsebuje vse, kar je potrebno za uporabo CUDE, in sicer glavne datoteke, knjižnice in NVCC predvajalnik. To orodje se praviloma ne posodablja preveč pogosto, razen takrat, ko na trg pride nova strojna oprema in so zato potrebne prilagoditve. CUDA jedra so na voljo pri večini njihovih grafičnih kartic, kar omogoča, da dobimo to tehnologijo tudi pri cenejših GeForce grafičnih kraticah, ki so praviloma narejene za igranje iger, in ne le pri veliko dražjih NVIDIA RTX (angl. Ray Tracing Texel eXtreme, v nadaljevanju RTX) in NVIDIA Quadro (Ansorge, 2022; Storti & Yurtoglu, 2015).

Namen te arhitekture je izkoristiti eno izmed največjih prednosti grafičnih procesnih enot, in sicer število jeder. Glavne procesorske enote, ki jih sicer uporabljamo za izvedbo podobnih nalog, imajo praviloma veliko manj jeder kot grafične procesne enote. V našem primeru imamo dokaj močan računalnik, ki ima glavno procesno enoto Advanced Micro Devices (v nadaljevanju AMD) Ryzen 9 5950x. Četudi je to eden izmed močnejših procesorjev za osebno rabo, vseeno premore le 16 fizičnih jeder oziroma 32 virtualnih jeder (mogoča večnitnost). Naša grafična procesna enota NVIDIA RTX 3090 ima 10496 CUDA jeder (AMD, 2022; Nvidia, 2022b; Ansorge, 2022).

Veliko sodobnih algoritmov za globoko učenje (sploh Transformerji) je prilagojenih za vzporedno izvajanje in kot taki omogočajo zelo učinkovito uporabo grafičnih procesnih enot. Poleg CUDE uporabili tudi NVIDIA CUDA Deep Neural Network library (v nadaljevanju cuDNN). Namen te knjižice je, da je narejena izključna za uporabo pri globokem učenju in je tako veliko bolj učinkovita kot navadna CUDA. Knjižnica Darts, s katero smo izvedli praktični del magistrskega dela, deluje na osnovi knjižnice PyTorch. PyTorch je sicer odprtokodna knjižnica, ki jo je naredilo podjetje Facebook (zdaj META). Različica za Python je izšla leta 2017 in kmalu postala ena izmed najpogosteje uporabljenih knjižnic za namen globokega učenja. Razlogov za priljubljenost je sicer več, a so glavni preprostost

uporabe, hitrost, fleksibilnost, imperativnost (vsaka vrstica se izvede zaporedoma, kar nam omogoča lažje reševanje težav) ter že vnaprej trenirani modeli, ki nam lahko prihranijo veliko časa. Glavna slabost pa je predvsem majhna skupnost uporabnikov v primerjavi s knjižnico TensorFlow, ki je njena glavna konkurenca, zaradi česar ima slabšo dokumentacijo. cuDNN ima v globokem učenju med dva- in petkrat večjo zmogljivost kot CUDA (Saleh, 2019; Nvidia, 2022c).

PyTorch je bil izbran kot okvir za strojno učenje. S tem okvirjem nismo delali neposredno, a so njegove funkcije integrirane v knjižnico Darts. PyTorch je v zadnjem obdobju zelo priljubljen, saj ponuja zelo dobro podporo za izvajanje na grafičnih procesnih enotah, prav tako pa ima podporo za uporabo več grafičnih procesnih enot hkrati. Prednost je tudi razmeroma preprosta inštalacija in vzpostavitev delovanja. Njegova priljubljenost med drugih izhaja tudi iz dejstva, da je optimiziran za uporabo z modeli globokega učenja, ki so v zadnjih letih čedalje pogostejši. To je predvsem zato, ker se je število CUDA jeder na grafičnih karticah zelo hitro zvišalo (GTX 1080 Ti (angl. Giga Texel Shader eXtreme) je imela 3584 CUDA jeder, RTX 2080 Ti 4352 CUDA jeder, RTX 3080 Ti pa kar 10240 CUDA jeder) (Nvidia, 2021; Nvidia, 2022a; Den Bakker, 2017; Saleh, 2019; Subramanian, 2018; Mathew, 2020).

2.2 Knjižnice

Knjižnico za branje podatkov o cenah delnic s spleta smo se odločili integrirati v naš Jupyter Notebook. To pomeni, da smo morali najti vmesnik za namensko programiranje (angl. Application Programming Interface, v nadaljevanju API), ki je bil primeren za uporabo s programskim jezikom Python. Na voljo je veliko različnih knjižnic, a so marsikatere plačljive in posledično dokaj drage. Te knjižice ponujajo veliko možnosti, med drugim tudi izračune različnih finančnih kazalnikov, ki smo jih mi raje izračunali sami.

Po pregledu in testiranju več različnih rešitev je bilo ugotovljeno, da sta najboljši izbiri knjižnici yfinance in Alpha Vantage. Na koncu smo izbrali yfinance, saj je z njo lažje delati in ne potrebuje API ključa (Aroussi, 2022; Vantage, 2022).

Yfinance (knjižnica, ki smo jo uporabljali za branje podatkov s spleta) omogoča izvoz neposredno v Pandas podatkovni okvir, zato smo se za delo s podatki odločili za uporabo Pandasa. Knjižnica je namenjena branju, pisanju in procesiranju podatkov. Med glavnimi razlogi, da smo izbrali ravno to knjižico, je sicer tudi dejstvo, da je ta knjižica neuradni standard, kar nam omogoča premikanje podatkov iz ene knjižnice v drugo (npr. iz yfinance v Darts ne moremo neposredno prenesti podatkov), prav tako pa jih brez te knjižice razmeroma težko urejamo (Hagedorn, Kläbe & Sattler, 2021; Raschka, Julian & Hearty, 2016).

Ko imamo na voljo Pandas podatkovni okvir s podatki o delnicah, imamo na voljo le datum (Date), ceno ob odprtju borze (Open), najvišjo ceno dneva (High), najnižjo ceno dneva

(Low), ceno ob zaprtju borze (Close), količino delnic, ki so bile prodane v tistem dnevu (Volume) in ali je bila na tisti dan izplačana dividenda (Dividends). To so vsi podatki, ki jih imamo na voljo. Iz teh podatkov je mogoče izračunati več novih kazalnikov, ki so ključni del magistrskega dela. Ker bi ročna izvedba (vpis formule za vsakega posebej) trajala zelo dolgo, smo se odločili za uporabo knjižnice TA, katere namen je izračun in zapis večjega števila kazalnikov na osnovi prej omenjenih podatkov (Darío, 2022; Padial, 2018).

Pri izrisu grafov smo imeli na voljo veliko manj izbire med knjižnicami. Knjižnica Darts namreč predpostavlja uporabo knjižnice Matplotlib. Ker smo potrebovali le preproste dvodimenzijske upodobitve, smo se odločili, da bi bilo najbolj smiselno uporabiti kar to knjižnico. Glavna prednost je, da je zgrajena na osnovi SciPy okolja, s čimer ima zelo dobro povezljivost z drugimi knjižnicami, ki smo jih uporabili. Najpomembnejša je podpora za NumPy, saj nam tako olajšuje del s podatki, predvsem pri uvozu in izvozu podatkov. Skoraj vse upodobitve, ki smo jih naredili, namreč vsebujejo iste podatke (zadnja cena dneva in napovedana cena) in ne zahtevajo bistvenih prilagajanj ali zapletenejših prikazov (Raschka, Julian & Hearty, 2016; Ari & Ustazhanov, 2014; Barrett, Hunter, Miller, Hsu & Greenfield, 2005).

Knjižnica Darts, na osnovi katere smo naredili magistrsko delo, ponuja skoraj vse, kar je potrebno za uporabo algoritmov globokega učenja in delo s podatki. Težava nastane, ker nekatera v knjižnici prisotna orodja niso nujno najboljša. Takšen primer so orodja za razdelitev podatkov na testne in podatke za ugotavljanje veljavnosti ter pripisovanje manjkajočih vrednosti. V teh primeru smo potrebovali knjižnico, ki nam bo omogočala izvajanje teh dveh nalog. Tu smo se odločili za najpogosteje uporabljeno Python knjižnico za strojno učenje, in sicer Scikit-Learn. Njena glavna prednost je predvsem zelo veliko število uporabnikov in posledično tudi več podpore ter boljša dokumentacija. Knjižnica je sicer veliko bolj splošna in je primerna za delo z različnimi težavami, ne le s časovnimi serijami in regresijo (Hao & Ho, 2019; Pedregosa in drugi, 2011).

Velik del magistrskega dela je bil narejen na osnovi knjižnice Darts. Knjižnica nam omogoča, da skoraj celoten postopek od dela s podatki do uporabe algoritma in testiranja na preteklih podatkih lahko izvedeno le z orodji, ki jih premore ta knjižnica. Vsebuje namreč orodja za procesiranje podatkov, lastne podatkovne sete, metrike, modele, orodja za časovne serije in druga orodja. Vse modele lahko uporabljamo na isti način s funkcijami, kot sta fit() in predict(). Ideja tega poenotenja je bila v veliki meri povzeta Scikit-learn. Pri algoritmih iz knjižnic, ki podpirajo grafično pospeševanje, je to zelo dobro pripravljeno in preprosto za uporabo. Algoritmi, ki so na voljo, so zelo raznoliki in podpirajo univariatne serije, multivariatne serije, probabilistične prikaze, več časovnih serij skupaj, pretekle sospremenljivke (covariates), prihodnje sospremenljivke in statične sospremenljivke. Seveda vsak model ne podpira vsega, kar je bilo prej navedeno, z izjemo TFT, ki podpira vse. Zadnja velika prednost, ki je bila za nas odločilna pri izbiri te knjižnice, pa je podpora najnovejših in zapletenejših modelov. To nam omogoča, da bo vsebina magistrskega dela

relevantna vsaj še nekaj let in ne bo zastarela že ob oddaji (Herzen in drugi, 2022; Unit8, 2022c; Unit8, 2022a).

Izbor podatkov je mogoče narediti na veliko različnih načinov. Pri knjižnicah za izbor značilnic je izjemno malo izbire, zato smo izbrali FeatureWiz, saj je imel še najboljšo dokumentacijo, hkrati pa za izbiro uporablja umetno inteligenco, na kateri sloni celotno magistrsko delo. Ta knjižnica ima sicer več različnih možnosti za izbor primernih značilnic, a smo se po pregledu odločili za privzet način izbora, ki je zasnovan tako, da se izvede v dveh stopnjah. V prvi uporabi metodo SULOV (angl. Searching for Uncorrelated List of Variables), ki v več izvedbah poišče značilnice, ki imajo najvišje informacijske vrednosti in najmanj sorazmerja med seboj. V drugi uporabi rekurzivno XGBoost (angl. eXtreme Gradient Boosting) metodo, da med preostalimi značilnicami najde tiste, ki so najboljše. Uporaba te metode tudi zmanjša možnosti za prekomerno oprijemanje (AutoViML, 2022).
Druga knjižnica, ki smo jo uporabili, je SKLearn, pri kateri smo uporabili Naključni gozdni regresor (Random Forest Regressor, v nadaljevnaju RFR), ki temelji na RF. Ta pristop bo uporabljen kot kontrolna primerjava izboru s FeatureWiz, saj smo v preteklosti RFR že uporabili in je dal zelo dobre rezultate (Scikit-learn developers, 2022).

3 PRIPRAVA PODATKOV IN IZBOR TEHNIK STROJNEGA UČENJA

3.1 Urejanje podatkov

Kot že prej omenjeno, nam viri podatkov dajejo le šest stolpcev podatkov. Štirje od teh stolpcev so med seboj izjemno povezani in v primeru zelo nizke nestanovitnosti skoraj identični (tržne cene ob koncu dneva, cene ob odprtju trga, najvišje cene trgovskega dneva in najnižje cene trgovskega dneva). Če torej želimo uporabiti multivariatne podatke, bomo morali dodati še kakšno lastnost oziroma vrstico. Inženiring lastnosti je eden izmed najpomembnejših del pri procesu strojnega učenja in ima na rezultat pogosto večji vpliv kot izbrani algoritem ali nastavitve hiperparametrov algoritma. Namenjen je zmanjševanju napak modela in izpeljavi novih lastnosti iz starih. Praviloma je to zelo dolgotrajen postopek, saj je število kombinacij izjemno veliko. Zahteva tudi ogromno znanja s področja, na katerem delamo. V marsikaterem primeru se inženiring lastnosti uporablja tudi za zmanjševanje dimenzionalnosti, saj pogosto tako dobimo manjše število bolj relativnih lastnosti (Xie in drugi, 2020; Jukes, 2018; Richert, 2013; Soloviev in drugi, 2020).

Napovedovanje prihodnjih vrednosti delnic je eden izmed zapletenejših problemov, s katerimi se soočajo profesionalci na tem področju in akademiki. V večini primerov se dodatne lastnosti dodajo na osnovi tehnične analize. Ta isti postopek smo uporabili tudi mi. Lastnosti smo namreč izračunali iz obstoječih šestih vrstic s pomočjo knjižnice TA. Treba je omeniti, da sicer obstajajo še bolj zapletene rešitve. Mogoča je tudi izvedba inženiringa

lastnosti z globokim učenjem, ki pa je bolj zapletena in bi v primeru uporabe presegala cilje magistrskega dela. V magistrskem delu smo se namreč osredotočili na primerjavo med univariatnimi in multivariatnimi časovnimi serijami ter med različnimi modeli (Long, Lu & Chui, 2019).

3.2 Nadomeščanje manjkajočih vrednosti

Podatki, ki nam ostanejo po inženiringu lastnosti, imajo nekaj težav. Najbolj očitne so manjkajoče vrednosti pri večini kazalnikov, ki smo jih izračunali. Manjkajoče vrednosti delimo na tiste, ki manjkajo čisto naključno (ne moremo obrazložiti razloga), manjkajo naključno (primanjkljaj lahko razložimo na osnovi ostalih podatkov) in na tiste, ki ne manjkajo naključno (manjkajoče vrednosti so odvisne od vrednosti ali spremenljivk, ki jih ne vidimo). Manjkajoče vrednosti so namreč eden izmed najpogostejših problemov, s katerimi se soočamo pri preverbi kakovosti podatkov. Na srečo v našem primeru vemo, kaj je razlog. Prvih 31 dni je praznih zato, ker marsikateri kazalnik lahko izračunamo le za en mesec po tistem, ko dobimo podatke. Tako je bilo glede na razmeroma veliko količino podatkov najboljše odstraniti prvih 31 vrstic. Podatki, ki so ostali, so imeli majhno število manjkajočih vrednosti, tako da smo zlahka zapolnili še preostale s pripisovanjem manjkajočih vrednosti. Pri pripisovanju manjkajočih vrednosti imamo na voljo več možnosti. Odločili smo se, da uporabimo eno izmed metod iz knjižnice SKLearn. Na voljo smo imeli SimpleImputer, IterativeInputer in KNNImputer. Odločili smo se za uporabo KNNImputer, saj smo z njim že pri drugih projektih dobili dobre rezultate. Uporabili smo pet najbližjih sosedov (Curley, Krause, Feiock & Hawkins, 2019; Liu, 2017; Kotu & Deshpande, 2018).

Soroden problem so tudi »luknje« med datumi. Ker govorimo o delniških podatkih, so ti na voljo le za dneve, ko je borza odprta. Tako nastane težava, saj knjižnica Darts zahteva neprekinjen indeks. Treba je bilo dodati manjkajoče datume. Knjižnica Darts ima sicer lastne funkcije za zapolnjevanje manjkajočih dni, a v našem primeru funkcija ni delovala pravilno in smo se zato odločili za uporabo knjižnice Pandas. Uporabili smo funkcijo asfreq, ki nam je omogočila dodajanje manjkajočih dni. Ko smo imeli na voljo prej manjkajoče dni, smo njihovo vrednost dodali s funkcijo fillna (isto iz knjižnice Pandas), ki nam je omogočila, da smo manjkajoče vrednosti zamenjali z zadnjo polno vrednostjo. To je logična rešitev, saj se vrednost delnic na dneve brez trgovanja ne spreminja in ostane ista, kot je bila pri zadnjem trgovalnem dnevu (Unit8, 2022k; Pandas, 2022a; Pandas, 2022b).

Tabela 1: Podatki pred nadomeščanjem manjkajočih vrednosti

Close	Volume	Dividends	ABER_ZG_5_15	ABER_SG_5_15
64.462.509	11337200	0.0	NaN	NaN
64.727.478	10220400	0.0	NaN	NaN
62.956.425	18502400	0.0	NaN	NaN
62.621.719	16670700	0.0	NaN	NaN
63.256.260	13590700	0.0	63.682.043	NaN

...	...	...	...	...
65.083.824	10161400	0.0	64.521.100	65.119.014
65.091.560	4777200	0.0	64.661.648	65.243.470
65.029.564	7000600	0.0	64.858.005	65.419.640
65.122.559	7495300	0.0	65.030.592	65.567.703
64.828.041	8523400	0.0	65.051.261	65.581.501

Vir: lastno delo.

Tabela 2: Podatki po nadomeščanju manjkajočih vrednosti

Close	Volume	Dividends	ABER_ZG_5_15	ABER_SG_5_15
65.369.232	11758700.0	0.0	64.529.971	65.708.190
63.936.108	19177900.0	0.0	64.545.894	65.775.831
62.833.141	19350600.0	0.0	64.415.691	65.681.079
63.170.372	15959900.0	0.0	64.071.928	65.300.478
63.170.372	15959900.0	0.0	64.071.928	65.300.478
...	...	...	...	...
65.083.824	10161400.0	0.0	64.521.100	65.119.014
65.091.560	4777200.0	0.0	64.661.648	65.243.470
65.029.564	7000600.0	0.0	64.858.005	65.419.640
65.122.559	7495300.0	0.0	65.030.592	65.567.703
64.828.041	8523400.0	0.0	65.051.261	65.581.501

Vir: lastno delo.

3.3 Transformacija podatkov

Pred uporabo smo morali naše podatke še transformirati. Podatki, s katerimi smo delali, so imeli zelo velike razlike med vrednostmi glede na lastnost. Marsikateri indeksi so imeli vrednosti pod ena, medtem ko je bila vrednost delnic razmeroma visoka. To bi v praksi pomenilo, da bi imele vrednosti, kot je npr. vrednost delnic, disproporcionalno velik vpliv na naš rezultat. Tega si zagotovo ne želimo, zato je nujno, da te razdalje med različnimi lastnosti bistveno zmanjšamo. Praviloma imamo na voljo dve možnosti. To sta normalizacija in standardizacija. V našem primeru smo uporabili normalizacijo, saj je že zelo dobro integrirana v knjižnico Darts. Obstaja sicer veliko različnih metod normalizacije podatkov. Med pogostejše spadajo Min-Max (v našem primeru je uporabljena izpeljanka te metode), Z-Score, decimalno skaliranje in še nekaj drugih. V primeru Min-Max normalizacije, ki smo jo uporabili, ta vse podatke spremeni v vrednosti med nič in ena. To je razmeroma preprosta rešitev, četudi ne daje vedno najbolj optimalnih podatkov. V marsikaterem primeru bi sicer dobili boljše podatke z uporabo standardizacije. Ta je sicer nekoliko bolj zapletena, a mogoče daje boljše rezultate. Praviloma ta deluje tako, da centriramo stolpce z lastnostmi pri srednji vrednosti nič, s standardnim odklonom ena. Stolpci imajo tako normalno porazdelitev. Prednost standardizacije pred normalizacijo je tudi boljše ohranjanje informacij o izstopajočih vrednostih (Ali, Faraj, Koya, Ali & Faraj, 2014; Jukes, 2018; Kotu & Deshpande, 2018; Raschka; Julian & Hearty, 2016; Swamynathan, 2019; Jo, 2019; Raschka, 2015).

Tabela 3: Podatkovni niz pred transformacijo

WCP	HA_close	HLC3	MEDIAN_30	SQZPRO_OFF
160.910.779	161.511.102	161.064.708	154.397.148	2.0
160.910.779	161.511.102	161.064.708	154.397.148	2.0
160.910.779	161.511.102	161.064.708	154.397.148	2.0
157.527.579	157.252.936	157.448.184	154.572.144	2.0
160.349.362	159.941.044	160.180.039	155.121.422	2.0
...	...	...	...	...
253.947.502	252.647.503	253.590.001	241.364.792	2.0
253.947.502	252.647.503	253.590.001	241.364.792	2.0
253.947.502	252.647.503	253.590.001	241.364.792	2.0
250.570.000	251.022.499	250.693.334	241.485.001	2.0
246.469.997	247.895.000	246.919.998	241.485.001	2.0

Vir: lastno delo.

Tabela 4: Podatkovni niz po transformaciji

WCP	HA_close	HLC3	MEDIAN_30	SQZPRO_OFF
0.135403	0.137229	0.135704	0.014567	1.0
0.135403	0.137229	0.135704	0.014567	1.0
0.135403	0.137229	0.135704	0.014567	1.0
0.119087	0.116672	0.118263	0.015537	1.0
0.132695	0.129649	0.131438	0.018579	1.0
...	...	...	...	...
0.584079	0.577194	0.581919	0.496355	1.0
0.584079	0.577194	0.581919	0.496355	1.0
0.584079	0.577194	0.581919	0.496355	1.0
0.567790	0.569349	0.567950	0.497021	1.0
0.548018	0.554251	0.549752	0.497021	1.0

Vir: lastno delo.

3.4 Izbor značilnic

Naš podatkovni set ima približno 1000 vrstic in malo več kot 200 stolpcev. Stolpcev je 200, saj smo iz začetnih 6 stolpcev z uporabo knjižnice TA izračunali več indikatorjev. To je vsekakor problem, saj imamo zelo visoko dimenzionalnost. Prav tako imamo razmeroma majhno število vzorcev. Ta dva dejavnika nas pripeljeta do »prekletstva dimenzionalnosti«. Če ne bi sprejeli ukrepov za znižanje dimenzionalnosti, bi imeli najverjetneje resne težave s

prekomernim prileganjem. Ta je sicer najpogosteje prisoten v ravno takšnih podatkovnih nizih, kot je naš. Poleg tega vsi vzorci ne prispevajo enako k napovedovalni moči našega modela. Nekateri pa so med seboj tako sorazmerni, da je mogoče enega spustiti, ne da bi izgubili napovedno moč. Na voljo imamo dva pristopa, s katerima lahko zmanjšamo dimenzionalnost. Prva možnost je izbor značilnic, druga pa je projekcija lastnosti. Odločili se bomo sicer za izbor značilnic, a je vseeno smiselno omeniti projekcijo lastnosti. S to metodo del lastnosti, ki jih imamo na voljo v podatkovnem nizu, prestavimo na druge dimenzije ali osi. Med najpriljubljenejšimi metodami tega tipa je analiza glavnih komponent (angl. Principal component analysis - PCA). Ta poskuša poiskati komponente, na katere je mogoče projicirati podatke, ne da bi izgubili prejšnje sorazmerje med dimenzijami. Ta metoda uporablja evklidsko razdaljo za merjenje razmika med različnimi objekti. Druga možnost, ki je zanimivejša, je izbor značilnic. Ne glede na to, za katero možnost se odločimo, moramo ohraniti obstoječa razmerja med spremenljivkami. Pri izboru lastnosti je naš cilj zmanjšati število stolpcev (vzorcev) v našem podatkovnem nizu, da dobimo podatkovni set, ki bo imel večjo napovedno moč. Obstaja več metod za izbiro značilnic. V večini primerov se delijo na filtrske metode, ovojne metode in vgrajene metode. Filtrske metode so hitre, razširljive in neodvisne od klasifikatorja, a posledično ignorirajo vse interakcije s klasifikatorjem. Ovojne metode so preproste, imajo interakcije s klasifikatorjem in modelirajo odvisnost lastnosti, a imajo večje možnosti za prekomerno prilagajanje (angl. overfitting), so odvisne od klasifikatorja in so računsko zahtevne. Tretje so vgrajene oziroma hibridne metode. Te so manj zapletene kot ovojne metode, imajo interakcije s klasifikatorjem in modelirajo odvisnost lastnosti, a so tako kot ovojne odvisne od klasifikatorja (Jukes, 2018; Kotu & Deshpande, 2018; Julian, 2016; Li, Li & Liu, 2017; Liu in drugi, 2005; Zocca, Spacagna, Slater & Roelants, 2017; Li in drugi, 2016).

Poleg prej navedenega obstajajo še druge možnosti za izbor značilnic. Nekateri modeli globokega učenja so sposobni postopek izbora lastnosti v postopku povratnega širjenja opraviti sami. Obstajajo tudi metode za izbor značilnic z uporabo strojnega učenja. Eno izmed takšnih smo uporabili tudi sami. Uporabili smo knjižnico FeatureWiz, ki izvede izbor značilnic na osnovi funkcij SULOV in XGBoost. Pomembno je omeniti, da v primeru nekaterih algoritmov, ki smo jih uporabili, ločen izbor značilnic ni potreben, saj ga algoritem opravi sam. Dober primer tega je algoritem TFT, ki z algoritmom variabilne izbire sam izbere primerne lastnosti. V določenih primerih, kot je delo s podatki v medicini (kjer bo kliničen rezultat), je sicer mogoč tudi ročni izbor značilnic, a bi bil ta v našem primeru nepotreben, zamuden in bi verjetno dal slabše rezultate od avtomatiziranih pristopov (Jan, 2020; Liu in drugi, 2005; Zocca, Spacagna, Slater & Roelants, 22017).

3.5 Modeli

V magistrskem delu smo uporabili več različnih modelov. Modeli, ki smo jih uporabili, so bili del knjižnice Darts. Ta vsebuje večje število modelov, kar nam ponuja nekoliko več prilagodljivosti pri izbiri. Na voljo imamo naslednje: ARIMA, VARIMA, AutoARIMA,

StatsForecast Auto ARIMA, eksponentno glajenje, BATS, TBATS, Theta, FourTheta, Prophet, Fast Fourier Transform, KalmanForecaster, Croston metoda, RegressionModel, RF, LinearRegressionModel, LightGBM (angl. Light Gradient Boosting Machine), RNN, LSTM, zaporna ponavljajoča se enota (angl. Gated recurrent unit, v nadaljevanju GRU), N-BEATS, nevronska hierarhična interpolacija za napovedovanje časovnih vrst (angl. Neural Hierarchical Interpolation for Time Series Forecasting, v nadaljevanju N-HiTS), Globoka časovna konvolucijska omrežja (angl. Deep Temporal Convolutional Networks v nadaljevanju Deep TCN), transformer, TFT in Naive Baselines. Iz logičnih razlogov ne moremo uporabiti vseh modelov, zato smo se odločili, da uporabimo le 10 modelov. Te smo razdelili v štiri kategorije.
Prvi so klasični modeli, kjer smo uporabili ARIMA, RNN in LSTM. Med novejšimi modeli smo uporabili Prophet, N-BEATS in Deep TCN. Za preproste modele smo uporabili StatsForecast Auto ARIMA in eksponentno glajenje. Zaključili pa smo z zapletenejšimi modeli, kjer smo uporabili dve različni vrsti transformerjev, in sicer klasični transformer in zelo zapleteni TFT.

3.6 Klasični modeli

3.6.1 ARIMA

Prednosti modela ARIMA so razmeroma nizka procesna zahtevnost (posledično ne potrebuje grafičnega pospeševanja), hitrost, je zelo učinkovit pri napovedovanju časovnih serij, podpira probabilistični prikaz in prihodnje sospremenljivke. Glavna slabost modela je, da v večini uporab podpira le univariatne časovne serije. Poleg tega je bistveno bolj preprost kot novejši modeli (Kotu & Deshpande, 2018; Pal, 2017; Unit8, 2022b; Ning, Kazemi & Tahmasebi, 2022).

3.6.2 RNN

RNN ima veliko prednosti pred drugimi modeli. Podpira namreč tako univariatne kot multivariatne časovne serije (podpora multivariatnih nam v našem primeru omogoča uporabo tehnične analize za napovedovanje vrednosti), prav tako nam omogoča uporabo prihodnjih sospremenljivk. Velika prednost je možnost uporabe grafičnega pospeševanja, saj nam to bistveno zniža potreben čas izvajanja. To sicer ni vedno potrebno, ker je ta model med manj zahtevnimi modeli globokega učenja in nevronskih mrež. Glavna slabost tega modela je nerazumevanje konteksta, kar onemogoča nekatere zapletenejše aplikacije, kot je prevajanje jezikov. Model sicer tudi ni enako zapleten kot nekatere njegove izpeljanke (GRU, LSTM). (Unit8, 2022g; Pal & Prakash, 2017; Raschka, Julian & Hearty, 2016; Zaccone & Karim, 2018; Kumar, 2019).

3.6.3 LSTM

LSTM je »zlati standard« modelov za napovedovanje časovnih serij in do let 2015 in 2016, ko so začeli prihajati boljši modeli, skoraj ni imel konkurence. To med drugim pomeni tudi, da so količine literature zanj ogromne, prav tako pa je na voljo v skoraj vseh knjižnicah za napovedovanje časovnih serij. Skoraj vse prednosti si deli z RNN modelom, na osnovi katerega je narejen (možnost uporabe univariatnih in multivariatnih časovnih serij, možnost grafičnega pospeševanja, možnost uporabe prihodnjih sospremenljivk). Poleg teh je njegova glavna prednost dolgoročni kratkoročni spomin, ki nam omogoča uporabo večje količine podatkov. Praviloma daje boljše rezultate kot RNN, prav tako pa ga je mogoče uporabiti za zapletenejše aplikacije, kot je prevajanje jezikov. Model nima veliko slabosti, z izjemo razmeroma velikih zahtev po računalniških virih ter nižje stopnje zapletenosti v primerjavi z nekaterimi novejšimi modeli (Bonaccorso, 2018; Vishwas & Patel, 2020; Zaccone & Karim, 2018).

3.7 Novejši modeli

3.7.1 Prophet

Prophet je univariaten model in eden izmed najnovejših, saj je bil razvit komaj leta 2017. Ima podporo podjetja META, ki ga je razvilo, kar pomeni, da je med bolje podprtimi. Je tudi odprtokoden in zelo odporen na skoke v trendih, manjše količine manjkajočih podatkov ter omogoča delo z več različnimi sezonskostmi. Spada med preprostejše modele (tudi ne zahteva veliko računalniških virov), a kljub temu daje enega izmed najboljših rezultatov za napovedovanje univariatnih časovnih serij. Podpira tudi uporabo prihodnjih sospremenljivk. Model nima veliko slabosti z izjemo omejenosti na univariatne časovne serije in nezmožnosti uporabe grafičnega pospeševanja (to tako ali tako ni potrebno, saj je model dokaj nezahteven in hiter) (Facebook, 2022; Unit8, 2022e; Garlapati in drugi, 2021; Vishwas & Patel, 2020; Letham Ben, 2017; Soloviev in drugi, 2020).

3.7.2 N-BEATS

N-BEATS model je eden izmed najnovejših modelov (leto 2019). Praviloma je narejen le za univariatne časovne serije, a v knjižnici Darts omogoča tudi uporabo pri multivariatnih časovnih serijah. Uporablja lahko pretekle sospremenljivke. V večini primerov daje zelo dobre rezultate in je srednje zahteven. Omogoča uporabo grafičnega pospeševanja. Glavna slabost je, da v večini knjižic podpira le univariatne časovne serije in da je iz njega izpeljan N-HiTS model, ki pogosto daje nekoliko boljše rezultate (Unit8, 2022f).

3.7.3 Deep TCN

Deep TCN je vrsta konvolucijskega omrežja. Je nov model (2020), ki omogoča uporabo univariatnih in multivariatnih časovnih serij. Uporaba preteklih sospremenljivk je mogoča. Posebnost tega modela je, da lahko predvidi latentno sorazmerje med serijami. Omogoča uporabo grafičnega pospeševanja. Model je sicer razmeroma zahteven, a kljub temu v primeru časovnih serij daje razmeroma povprečne rezultate (Bai, Kotler & Koltun, 2018; Chen, Kang, Chen & Wang, 2020; Kotu & Deshpande, 2018; Pal & Prakash, 2017; Or, 2020).

3.8 Preprosti modeli

3.8.1 StatsForecast Auto ARIMA

Prednosti in slabosti tega modela so sorodne modelu ARIMA in seveda Auto ARIMA. Ker je izpeljan iz modela Auto ARIMA, prav tako omogoča avtomatsko nastavitev parametrov. Podpira prihodnje sospremenljivke. Namenjen pa je le univariatnim časovnim serijam. Glavni prednosti pred drugimi vrstami ARIMA in Auto ARIMA sta možnost probabilističnega prikaza in večja hitrost (Unit8, 2022i; Unit8, 2022h; Taylor, 2022).

3.8.2 Eksponentno glajenje

Eksponentno glajenje je eden izmed najpreprostejših modelov. Model je neke vrste drseče povprečje tako kot ARIMA. Deluje le v univariatni obliki in nima možnosti za nobene nastavitve. Je manj zahteven od modelov globokega učenja. Omogoča večjo napovedovalno moč zadnjih točk na račun starejših (prvih točk). Uporaben je predvsem tam, kjer imamo časovne serije, ki niso pretirano volitivne in se gibajo linearno. V našem primeru ni najbolj uporaben in dobri rezultati niso predvideni (Kotu & Deshpande, 2018; Jansen, 2018; Josef, Skipper & Jonathan, 2019; Unit8, 2022d).

3.9 Zahtevnejši modeli

3.9.1 Transformer

Transformerji so model, ki omogoča skoraj vse možnosti (univariatna napoved, multivariatna napoved, probabilistični model in pretekle sospremenljivke). Model je prilagojen za vzporedno izvajanje, kar pomeni, da je veliko učinkovitejši, če se izvaja na grafični kartici (model omogoča grafično pospeševanje). Posebej učinkovit je pri delu z jeziki, prav tako pa tudi pri časovnih serijah. Je eden izmed najbolj zapletenih modelov. Slabosti tega modela ni veliko. Največja je požrešnost računalniških virov, saj tega modela skorajda ni mogoče izvajati na procesorjih, ker je zanje prezahteven (traja veliko časa). Ko

izvajamo model na grafični kartici, potrebujemo ogromno grafičnega pomnilnika, česar pa marsikatera grafična kartica ne premore. Večji modeli tako potrebujejo veliko drage strojne opreme (Kamath, Graham & Emara, 2022; Vaswani in drugi, 2017; Unit8, 2022j; Ekman, 2021; Wu, Green, Ben & O'Brien, 2020).

3.9.2 TFT

TFT je edini model, ki podpira vse možnosti, torej univariatni model, multivariatni model, probabilistični prikaz, uporabo prihodnjih in preteklih kovariat. Med vsemi modeli za napovedovanje časovnih serij, ki jih podpirajo najpogostejše knjižnice (TensforFlow, Keras, Pytorch itd.), je ta najbolj zapleten in tudi najbolj zmogljiv. Posebnost tega modela je tudi, da avtomatsko izvede svoj lasten izbor spremenljivk ter tako optimizira rezultate napovedi. Model podpira uporabo grafičnega pospeševanja, ki pa je tu na žalost nujna, saj se bo na procesorju izvajal ekstremno dolgo. S tem smo prišli do glavnega problema tega sicer izjemno dobrega modela. Količina računalniških virov, ki jih porabi, je ogromna. Izvaja se tudi do desetkrat dlje kot drugi zapleteni modeli, njegova uporaba pa zahteva najboljšo strojno opremo, ki je na voljo na igričarskem trgu in trgu delovnih postaj. Večji modeli, narejeni na njegovi arhitekturi, pa so rezervirani za najmočnejše delovne postaje in superračunalnike (Unit8, 2022j; Jan, 2020; Wu, Green, Ben & O'Brien, 2022; Hu, 2021; Phetrittikun in drugi, 2021; Wen in drugi, 2022).

4 SIMULACIJA IN OVREDNOTENJE REZULTATOV

V tem delu smo izvedli naše simulacije. Izvedene so bile vse simulacije, ki so bile omenjene med modeli. Modeli, ki so to omogočali, so bili izvedeni s pomočjo orodja CUDA preko grafične kartice, modeli, ki tega niso omogočali, pa so bili izvedeni na procesorju. Izvedba je bila razdeljena na več Jupyter Notebook dokumentov. En dokument je bil namenjen univariatni izvedbi, drugi multivariatni izvedbi s FeatureWiz in tretji multivariatni izvedbi z RFR. Vsak je bil izveden šestkrat, in sicer dvakrat na delnici podjetja Microsoft (MSFT), dvakrat na delnici podjetja Exxon (XOM) in dvakrat na delnici podjetja J.P. Morgan (JPM). Razlog za dvojno izvedbo je v časovnih obdobjih, saj je bilo prvič izvedeno za časovno obdobje med 1. 1. 2015 in 1. 1. 2018, drugič pa za časovno obdobje med 7. 8. 2019 in 15. 9. 2022.

4.1 Postavitev raziskovalnih vprašanj

Prvotno so bila v magistrskem delu zamišljena tri raziskovalna vprašanja:

- Kateri način izbora značilnic nam omogoča najboljše rezultate? V osnovi jih imamo okoli 200 za vsako izmed delnic. Če v model vstavimo vse, dobimo ekstremen primer

prekomernega prileganja modelov. Pomembno je, da naredimo strategijo, ki nam bo omogočala, da izbor bistveno skrčimo.
- Ali bo uporaba tehničnih kazalnikov omogočala boljše rezultate, kot če bi uporabili le zadnje dnevne cene?
- Kateri izmed izbranih modelov nam omogoča najboljše rezultate? Uporabili bi različne modele strojnega učenja (transformerje, LSTM, N-BEATS, Facebook Prophet). Namen je narediti model, ki bo omogočal čim boljše rezultate v primeru testiranja na preteklih podatkih in prihodnjih podatkih.

V postopku izdelave magistrskega dela smo ugotovili, da obstaja še kar nekaj pomembnih vprašanj, na katera je treba odgovoriti. Na osnovi specifičnih razmer na mednarodnih trgih (covid-19 in ukrajinsko-ruska vojna) smo ugotovili, da bomo v trenutnem časovnem obdobju zelo težko dobili primerne podatke, zato smo izbrali dve časovni obdobji, ki smo ju sicer že prej omenili. To nam bo omogočalo primerjavo rezultatov pred trenutnimi krizami in v času njihovega trajanja.

Poleg tega je bilo tu še vprašanje dolžine napovedi. Ko smo začeli izdelavo magistrskega dela, smo izbrali dolžino napovedi 260 dni. To je razmeroma dolgo obdobje, zato smo se odločili poskusiti še s 105-dnevno napovedjo, da bomo videli, če bo dala bistveno boljše rezultate.

Tretji dejavnik, ki ga ob postavljanju prvih treh raziskovalnih vprašanj še nismo upoštevali, pa so sospremenljivke. Nekateri modeli sicer delujejo brez njih (preprostejši), drugi delujejo samo z njimi (TFT), obstaja pa tudi kar nekaj modelov, ki lahko delujejo z njimi ali brez njih. Pri teh modelih bi bilo smiselno primerjati rezultate.

Na osnovi teh treh dejavnikov smo razvili še dodatni dve raziskovalni vprašanji:

- Ali se rezultati bistveno razlikujejo glede na izbiro delnice?
- Ali bi modeli, narejeni na osnovi podatkov za obdobje med letoma 2015 in 2018, dali boljše rezultate kot tisti, ki so narejeni na podatkih med letoma 2019 in 2022?

4.2 Način ocenjevanja raziskovalnih vprašanj

Za ocenjevanje bomo uporabili sedem različnih metrik in sicer Popolno absolutno napako (angl. Mean Absolute Error, v nadaljevanju MAE), Povprečno absolutno procentualno napako (angl. Mean absolute percentage error; v nadaljevanju MAPE), Kvadratno napako (angl. Mean squared error, v nadaljevanju MSE), Celotno napako (angl. Root mean square error, v nadaljevanju RMSE), Koren srednje logaritmične napake na kvadrat (angl. Root Mean Squared Logarithmic Error, v nadaljevanju RMSLE), Koeficient variacije (angl. Coefficient of variation, v nadaljevanju CV) in Simetrično povprečno absolutno odstotno napako (angl. Synthetic Mean absolute percentage error, v nadaljevanju sMAPE). Pri vseh danih modelih je optimalno, če je vrednost čim bližje 0.

Postopek izračuna vrednosti, ki jih bomo uporabljali, je dokaj preprost. Začnemo z vrednostjo metrike. To kvadriramo (s kvadriranjem odstranimo negativne predznake). Nato damo vrednost na koren. Zatem zaokrožimo na 0,5 ter to vrednost pomnožimo z 2. Od 100 odštejemo dobljeno število.

Bistvenih napak pri podatkih ni bilo, z izjemo dveh manjkajočih vrednosti kazalnika RMSE pri eni izmed izvedb LSTM in RNN. V tem primeru smo ju nadomestili s povprečno vrednostjo istih algoritmov z drugimi specifikacijami.

4.3 Raziskovalna vprašanja

4.3.1 Metoda za izbor značilnic

Pri prvem vprašanju o načinu izbiranja značilnic, ki nam omogočajo najboljše rezultate, smo se odločili, da se osredotočimo na načina izbora značilnic, ki sta zasnovana na osnovi modelov umetne inteligence. Zanju smo se odločili predvsem zaradi velikega števila parametrov, ki jih imamo na voljo v našem podatkovnem okvirju. Najprej smo se odločili za RFR, s katerim smo imeli zelo pozitivne izkušnje iz prejšnjih projektov in FeatureWiz, ki je še bistveno bolj zapleten in deluje dvostopenjsko. Izbor značilnic smo seveda opravili le pri multivariatnih modelih, saj imamo pri univariatnih le eno značilnico, in sicer zadnjo ceno dneva. Vrednost tega parametra sicer poskušamo napovedati pri univariatnih in multivariatnih izvedbah.

Pri prvem (RFR) ni veliko nastavitev, ki bi jih bilo vredno omeniti. Uporabljenih je bilo 100 ocenjevalcev, izbor pa je bil narejen na osnovi indeksa pomembnosti posameznih parametrov. Tako smo izbrali vse, katerih vrednost je bila višja od 0,05. Ta meja je bila določena na osnovi izvedenih eksperimentov. Želeli smo jo določiti tako, da bomo dobili manj kot 10 vrednosti. Z vrednostjo 0,05 je ta pogoj skoraj vedno izpolnjen.

Pri drugem pristopu za izbiro značilnic smo uporabili knjižnico FeatureWiz. Nastavitve so bile v veliki meri privzete. Tako je bila izbrana na osnovi več poskusov v pripravah na praktični del magistrskega dela. Za razumljivost prikaza je tudi tokrat izbran isti nabor parametrov kot pri prejšnjem primeru, kar pomeni multivariatne podatke pri podjetju Microsoft med letoma 2015 in 2018 z napovedjo za 260 dni. Na prvi ravni izvede predizbor parametrov s pomočjo metode SULOV.

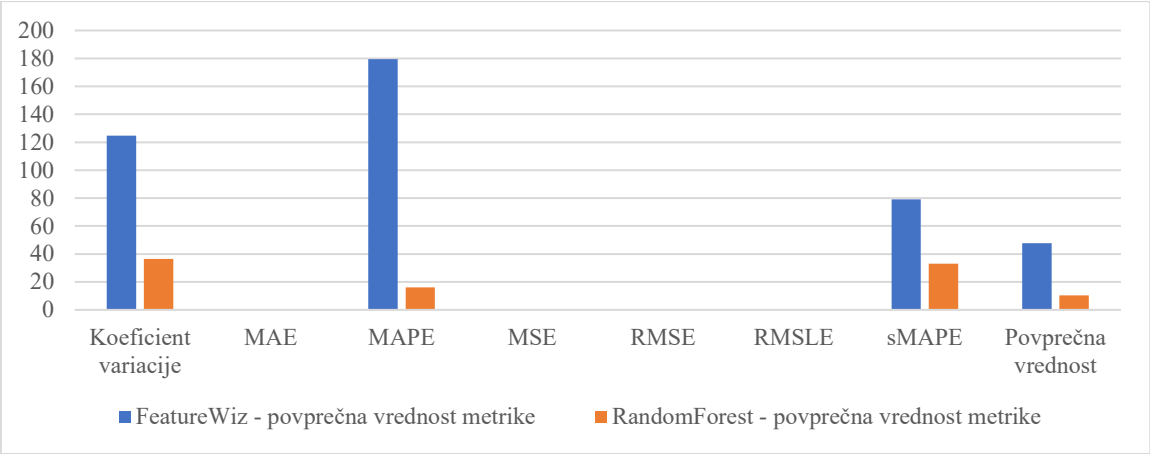
Za tem sledi še daljši izbor z metodo XGBoost, ki nam med prejšnjimi 48 značilnicami (48 jih je samo v tem specifičnem primeru in se razlikuje od modela do modela) poišče 10 najboljših. To izvede s petimi ponovitvami, v katerih izloča in izbira spremenljivke, ki imajo najboljše F-razporeditve. S tem lahko določimo, katere spremenljivke je najbolj smiselno uporabiti za napoved vrednosti ciljne spremenljivke. V našem primeru smo v teh dobili 17 spremenljivk: 'LDECAY_5', 'HLC3', 'TOS_STDEVALL_L_3', 'ABER_ATR_5_15',

'J_9_3', 'HWM', 'CFO_9', 'MACD_12_26_9', 'PPOh_12_26_9', 'VHF_28', 'KURT_30', 'CDL_INSIDE', 'PSARr_002_02', 'CDL_DOJI_10_01', 'THERMOs_20_2_05', 'SQZPRO_ON_NORMAL', 'Dividends'. Če odštejemo LDECAY_5, ki je med najboljšimi pri obeh načinih izbiranja značilnic, lahko hitro vidimo, da so si parametri, ki so bili izbrani, med seboj zelo različni, kar bi moralo teoretično privesti do drastično različnih rezultatov. Večino zgornjih spremenljivk predstavljajo tehnični kazalniki, ki so bili dodani takoj po prenosu podatkov. Številke poleg imen kazalnikov pa nakazujejo na dolžino časovnih obdobij. Primer tega je MACD kazalnik, ki ima zapisane vrednosti 12, 26 in 9. Ta števila nakazujejo tri različno dolga eksponentna drseča povprečja, in sicer 12-, 26- in 9-dnevno. Za ostale tehnične kazalnike nismo zapisali obrazložitev, saj bi to zavzelo preveč prostora. Izjema so sicer dividende, ki spadajo že med osnovne podatke.

Pri vseh vprašanjih smo naredili tri tabele, na osnovi katerih smo ocenjevali rezultate. Prva skupina je bila namenjena ovrednotenju, druga najboljšemu ovrednotenju, tretja pa za povratni test. Ovrednotenje je primerjava dobljenih rezultatov (napoved) z verifikacijskimi vrednostmi. Najboljše je, če so razlike med tema dvema vrednostnima čim manjše. Da bi razumeli, kako deluje najboljše ovrednotenje, moramo razumeti način izvedbe napovedi. Vsaka napoved je izvedena večkrat, vsakokrat v obliki enega epoha (teh smo uporabili od 100 do 300, odvisno od primera). Teoretično naj bi bila uporaba čim večjega števila teh boljša. Pri evalvaciji nam običajno izpiše vrednost zadnjega epoha (odvisno od knjižnice in modela), četudi ta vrednost ni nujno najboljša. Najboljše ovrednotenje deluje tako, da na osnovi vseh epohov poišče tistega, ki je najboljši na osnovi nekega ali nekaj kriterijev. V praksi bi lahko to pomenilo, da bi najboljša evalvacija dala vrednosti, ki bi bile pridobljene pri epohu z najmanjšo vrednostjo metrike MAPE. Povratni test je praviloma najmanj zanesljiv in najbolj zamuden. Ta primerja potencialne napovedi z verifikacijskimi vrednostmi. Njegovi rezultati so praviloma boljši od drugih in velikokrat preveč optimistični.

Ovrednotenje je prva skupina, s katero smo delali. Slika 11 nakazuje, da so povprečne vrednosti metrik v primeru uporabe RFR za izbor značilnic veliko boljše kot uporaba knjižice FeatureWiz. Kot je bilo prej zapisano v tabeli metrike, so idealne vrednosti vseh teh indikatorjev čim bližje 0. Med nekaterimi metrikami sicer ni veliko razlik, a so pri drugih ogromne. Dobili smo izjemno visoke vrednosti CV, MAPE in v manjši meri tudi sMAPE, to nakazuje, da so vrednosti, ki smo jih dobili, bistveno različne od verifikacijskih vrednosti. Povprečna vrednost metrike (povprečje vseh metrik) je tako pri FeatureWiz 47,66, pri RFR pa 10,3.

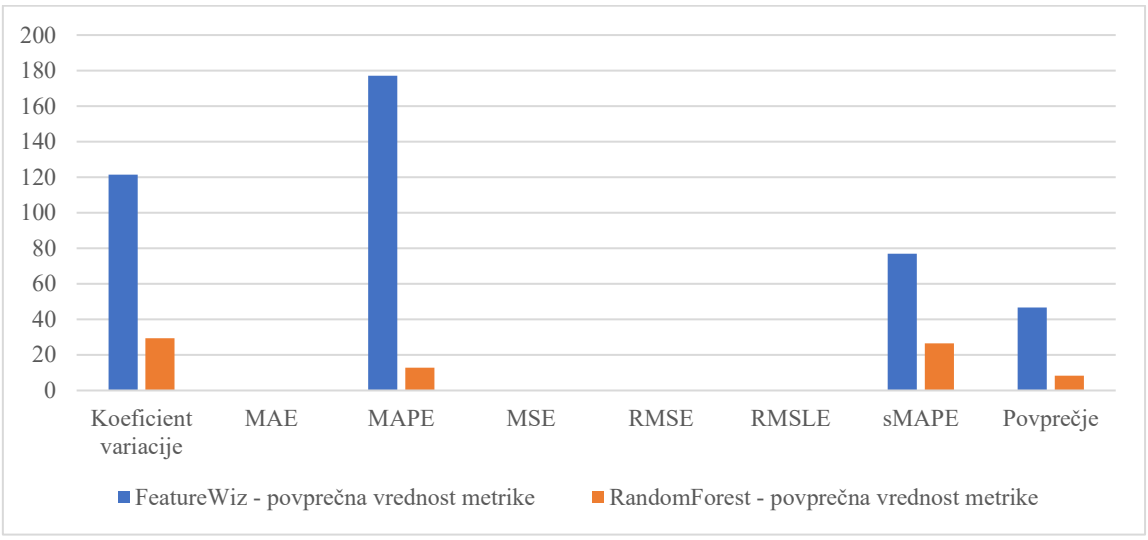
Slika 3: Najboljša metoda za izbor značilnic – ovrednotenje



Vir: lastno delo.

Pri najboljšem ovrednotenju se prej viden vzorec nadaljuje, le da je še nekoliko izrazitejši. Povprečna vrednost metrike je tako pri FeatureWiz 46,62, pri RFR pa 8,4.

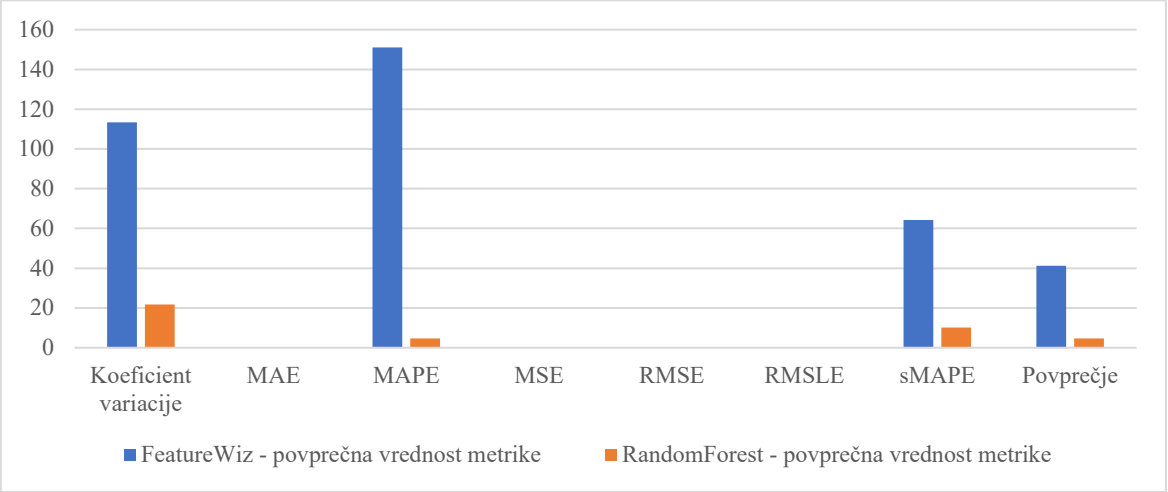
Slika 4: Najboljša metoda za izbor značilnic – najboljše ovrednotenje



Vir: lastno delo.

Iz slike 13 je očitno, da se prej videni vzorec nadaljuje tudi pri povratnem testu. Za obsežne razlike med tema dvema za izbor značilnic obstaja več razlag. Praviloma FeatureWiz dela dobro in ne daje slabih rezultatov, zato bi težko trdili, da so rezultati takšni, kot so, zaradi algoritma samega.

Slika 5: Najboljša metoda za izbor značilnic – povratni test



Vir: lastno delo.

Za rezultate pa obstajata dve zelo dobri in med seboj povezani razlagi. FeatureWiz sicer uspešno najde primerne značilnice, a jih ne ovrednoti po vrsti. Kot takšen ne omogoča izpisa relevantnosti za posamezno značilnico. Če ga zaženemo na naših podatkih, nam da približno 50 značilnic, kar pa je veliko preveč za naš model. Število teh lahko zmanjšamo le, če za oceno pomembnosti posameznih značilnic uporabimo drugo orodje ali če izmed teh značilnic izberemo manjše število čisto naključno. Ker smo se želeli temu izogniti, smo se odločili, da število značilnic zmanjšamo, zato smo nastavili, da morajo imeti izbrane metode nižjo korelacijo s ciljno spremenljivko. S tem smo zmanjšali število značilnic na med 15 in 17 (odvisno od podatkov), a tudi izpustili nekaj dobrih značilnic. Kljub temu je bilo teh vseeno več kot pri RFR, kjer smo jih uporabili manj kot deset. Ta dva dejavnika sta verjetno vplivala na slabe rezultate pri FeatureWiz.

4.3.2 Primerjava uporabe univariatnih ter multivariatnih časovnih serij

Drugo vprašanje, ali lahko z uporabo tehničnih kazalnikov dobimo boljše rezultate kot pri uporabi zadnjih dnevnih cen, je najpomembnejše izmed vseh, ki smo si jih zastavili. Če bi na tej točki namreč ugotovili, da modeli, narejeni na univariatnih časovnih serijah, dajejo boljše rezultate kot modeli, narejeni na osnovi multivariatnih časovnih serij, so nekatera druga vprašanja, predvsem pa prvo, brezpredmetna. Tudi ideja o temi magistrskega dela se je porodila prav na osnovi dela z multivariatnimi serijami, ki smo ga opravili za druge akademske projekte.

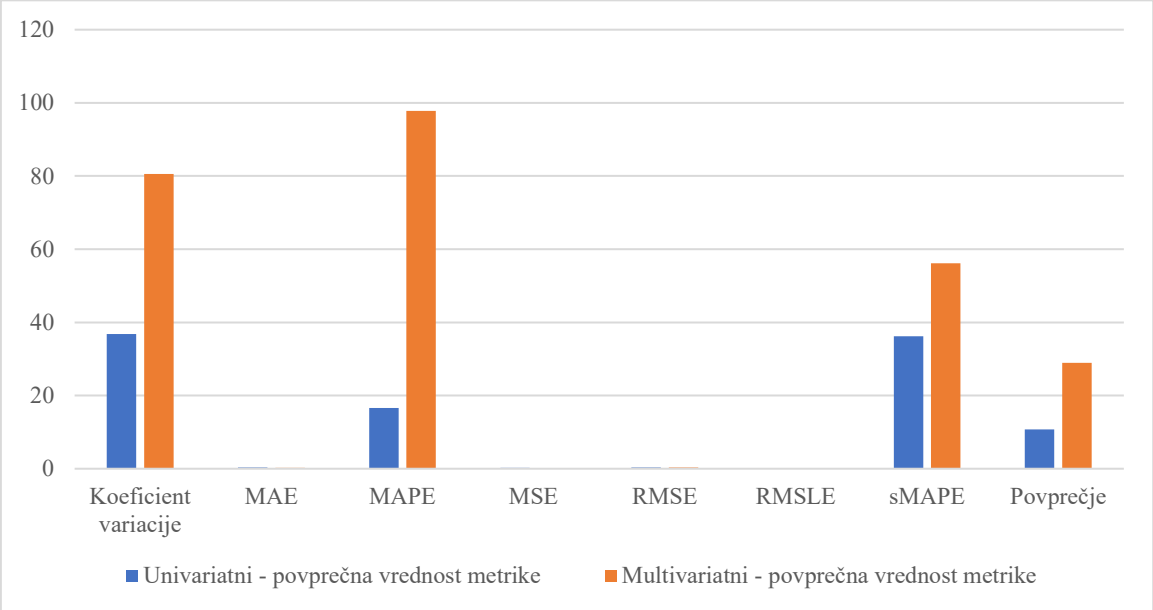
Na rezultate tega vprašanja zelo negativno vplivajo porazni rezultati FeatureWiz kot metode za izbor značilnic. Ker smo uporabili le dve različni metodi za izbor značilnic, ena izmed njiju pa je imela slabe rezultate, so tako rezultati dokaj nerealni. Četudi bi imeli multivariatni primeri, ki uporabljajo RFR, primerljive ali celo boljše rezultate, to ne bo dobro videno. Rezultati so sicer pričakovani. Multivariatne časovne serije se najslabše odrežejo ravno pri

istih metrikah, kot smo dobili slabe rezultate s FeatureWizom. Zanimivo je sicer, da so razlike med CV, MAPE, sMAPE in povprečjem manjše, kot so bile pri primerjavi metod za izbor značilnic. Na osnovi tega lahko sklepamo, da so rezultati RFR vsaj nekoliko boljši kot uporaba univariatnih serij. Prej je bila razlika med povprečjema FeatureWiz in RFR 47,66 proti 10,3, zdaj pa je med univariatnimi in multivariatnimi serijami le 10,73 proti 28,98. To je razvidno na sliki 6.

Rezultati v primeru najboljšega ovrednotenja so dokaj sorodni tistim iz ovrednotenja. Tu je povprečna vrednost metrike univariatnih časovnih serij 9,23, pri multivariatni pa je 27,51. To je razvidno na sliki 7.

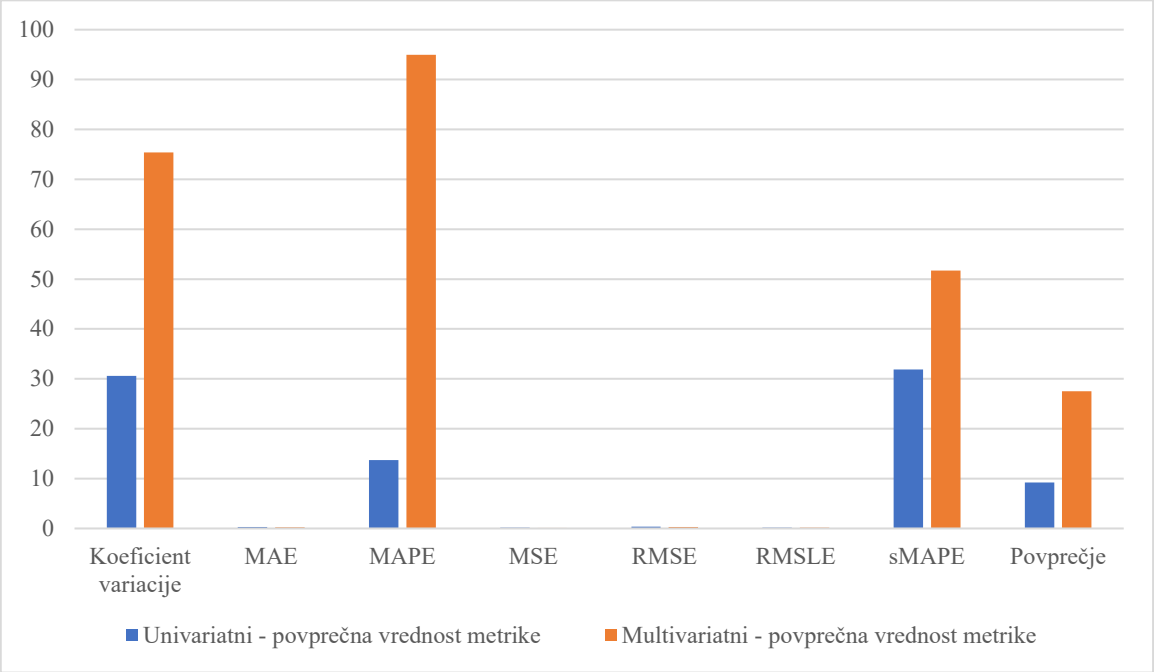
Rezultati v primeru povratnega testa so dokaj primerljivi s prejšnjimi, saj je povprečna vrednost metrike pri univariatnih serijah 6,51, pri multivariatnih pa 22,93. Kot smo zapisali pri ovrednotenju, bomo prave rezultate videli šele pri ovrednotenju najboljših modelov, saj je FeatureWiz pokvaril povprečje. Na osnovi dobljenih podatkov pa lahko vseeno sklepamo, da so rezultati multivariatnih časovnih serij, kjer je bil izbor značilnic narejen z RFR, nekoliko boljši kot rezultati univariatnih. To je razvidno na sliki 8.

Slika 6: Univariatne in multivariatne časovne serije – ovrednotenje



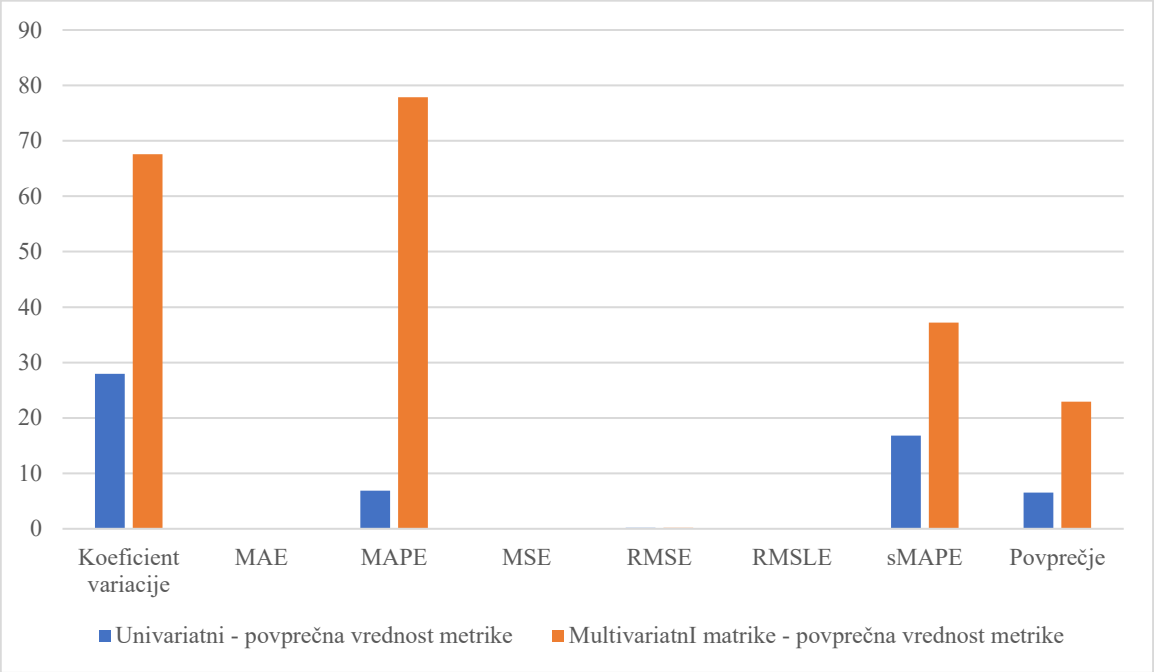
Vir: lastno delo.

Slika 7: Univariatne in multivariatne časovne serije – najboljše ovrednotenje



Vir: lastno delo.

Slika 8: Univariatne in multivariatne časovne serije – povratni test



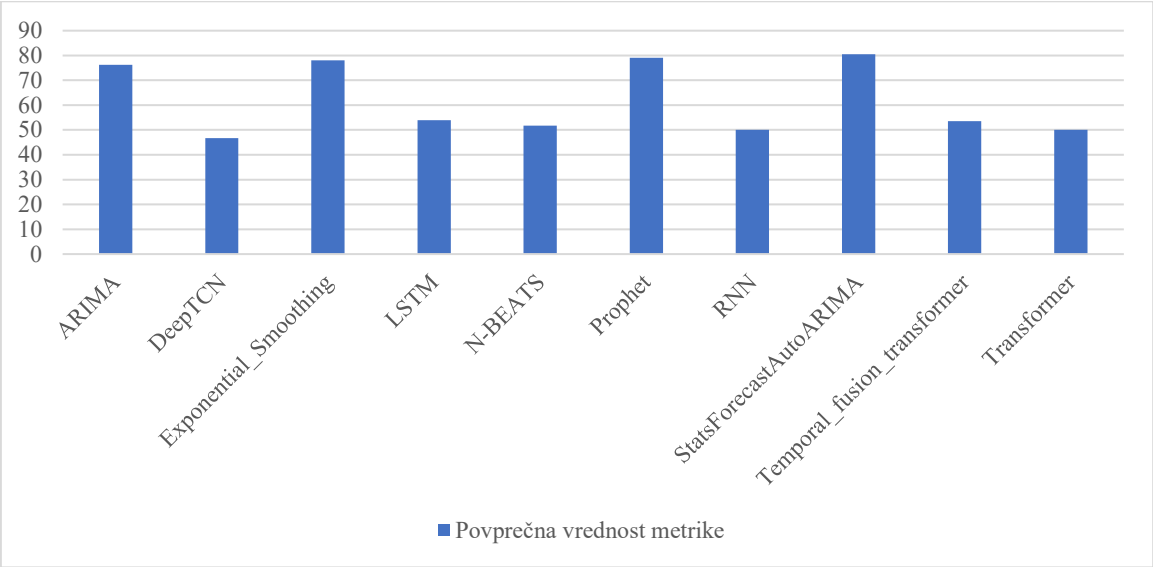
Vir: lastno delo.

4.3.3 Primerjava modelov

Najboljši model je eno izmed »originalnih« treh vprašanj, ki smo jih zastavili že v dispoziciji. S tem vprašanjem smo želeli ugotoviti, kateri izmed izbranih modelov nam omogoči najboljši rezultat. Tukaj smo jih ocenili tako, da smo skupaj (pri modelih, ki to sicer podpirajo) šteli tako univariatne in multivariatne primere istih modelov. To sicer ni najbolje, saj nekateri modeli dajejo veliko boljše rezultate v univariatni ali multivariatni obliki, a bi bila v primeru upoštevanja tudi tega parametra količina podatkov prevelika in nepregledna. Modele smo ocenjevali po isti lestvici kot prej.

Pri izbiri najboljšega modela se pojavi problem rezultatov, pridobljenih s FeatureWiz. Ko takšnih modelov ne moremo v celoti pravilno oceniti, saj nekateri modeli omogočajo izključno univariatno izvedbo, medtem ko drugi omogočajo uporabo univariatnih in multivariatnih časovnih serij. Kljub temu je očitno, da so v povprečju modeli, ki omogočajo uporabo univariatnih in multivariatnih časovnih serij, boljši od drugih. Tako so vsi štirje modeli, ki omogočajo samo uporabo univariatnih časovnih serij, bistveno slabši in imajo višje povprečne vrednosti. Tako je povprečna vrednost metrike pri najboljšem univariatnem modelu 76,23, pri tistih, ki omogočajo uporabo univariatnih in multivariatnih časovnih serij, pa je najslabši model LSTM z vrednostjo 53,95.

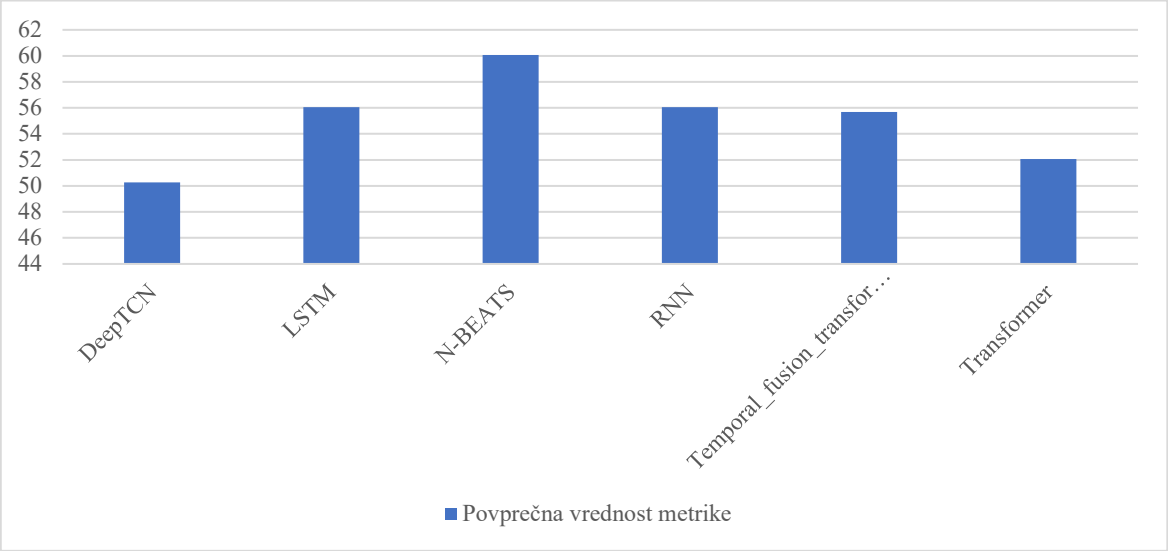
Slika 9: Najboljši model – ovrednotenje



Vir: lastno delo.

Pri najboljšem ovrednotenju imamo na voljo le modele, ki omogočajo uporabo univariatnih in multivariatnih časovnih serij. Tu ima najslabšo vrednost model N-BEATS (60,06), medtem ko ima najboljšo vrednost model Deep TCN (50,28). Večina modelov ima vrednosti slabše kot pri navadnem ovrednotenju, kar bi lahko kazalo na dokaj dobro predhodno optimizacijo.

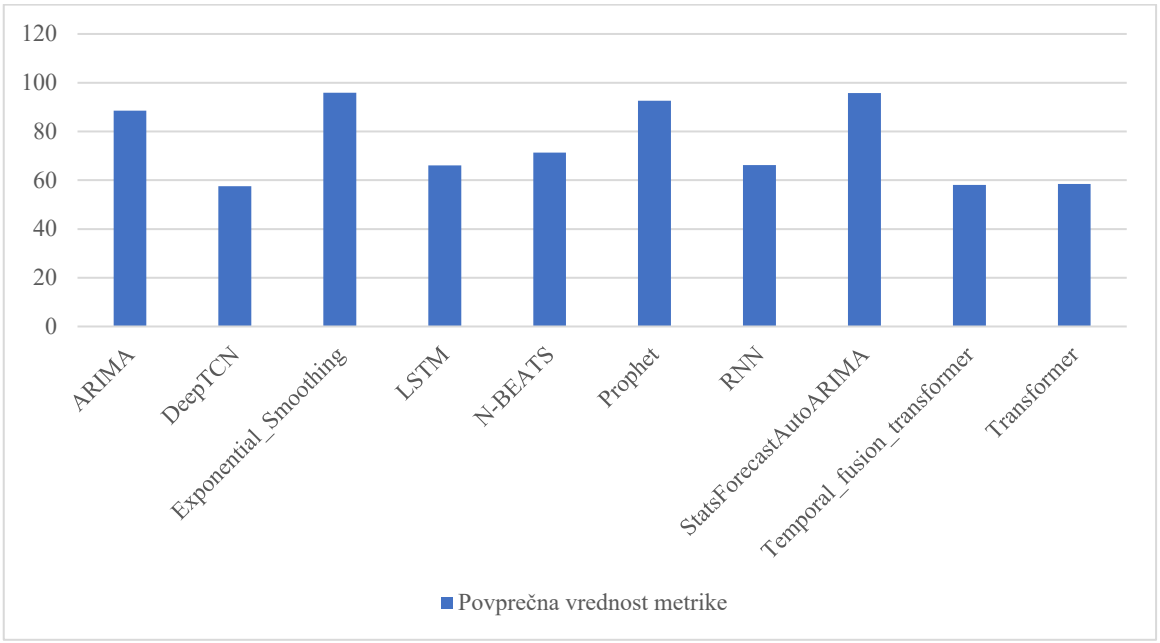
Slika 10: Najboljši model – najboljše ovrednotenje



Vir: lastno delo.

Pri povratnem testiranju se rezultati iz ovrednotenja v veliki meri ponovijo (razmerja med modeli), a so povprečne vrednosti vseeno višje kot pri ovrednotenju.

Slika 11: Najboljši model – povratni test



Vir: lastno delo.

Na osnovi videnega lahko trdimo, da so praviloma modeli, ki omogočajo uporabo univariatnih in multivariatnih časovnih serij, boljši od tistih, ki omogočajo le uporabo univariatnih. Med glavnimi razlogi je verjetno dejstvo, da so ti modeli novejši, veliko bolj

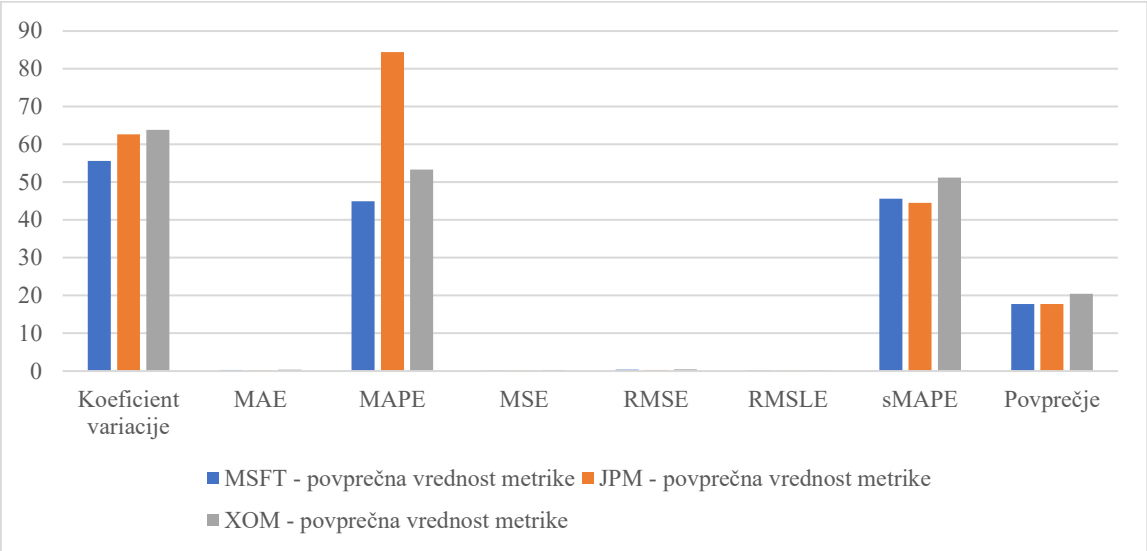
sofisticirani, prav tako pa porabijo več časa za napoved. Kljub temu so lahko razmeroma uporabni v dveh primerih, če delamo izključno z univariatnimi časovnimi serijami ali če imamo bistvene omejitve glede računalniških resorov, ki so nam na voljo.

4.3.4 Izbor delnic in vpliv na napoved

Pri četrtem vprašanju smo želeli ugotoviti, kakšen vpliv ima izbira delnice na rezultate. V magistrskem delu smo imeli tri različne delnice, ki smo jih uporabili za modeliranje. To so bile MSFT (Microsoft), JPM (JP Morgan) in XOM (Exxon). Predvidevali smo, da bo delnica JPM dala nekoliko boljše rezultate, saj ima manjšo nestanovitnost od MSFT. To seveda ni nujno vezano na ti dve specifični delnici, je pa vezano na njuni panogi. Tehnološka podjetja so splošno znana po razmeroma visoki nestanovitnosti cen delnic, medtem ko so vrednosti delnic bank veliko bolj stabilne. XOM smo izbrali, saj je podjetje dokaj dobro raslo, trenutna kriza v Ukrajini pa je zvišala cene fosilnih goriv, ki predstavljajo temelj poslovanja podjetja XOM.

Pri ovrednotenju so bile povprečne vrednosti metrik razmeroma podobne. Pri MSFT in JPM smo sicer dobili nekoliko manjši vrednosti (pri obeh 17,7), medtem ko je bila vrednost pri XOM nekoliko višja, in sicer 20,49. Iz Slike 20 izstopa vrednost MAPE pri delnici JPM. Ta je bistveno drugačna od drugih. Visoka vrednost MAPE pri JPM bi sicer lahko nakazovala na potencialne probleme, a bi morala biti potem višja tudi CV in sMAPE.

Slika 12: Najboljša delnica za točno napoved – ovrednotenje



Vir: lastno delo.

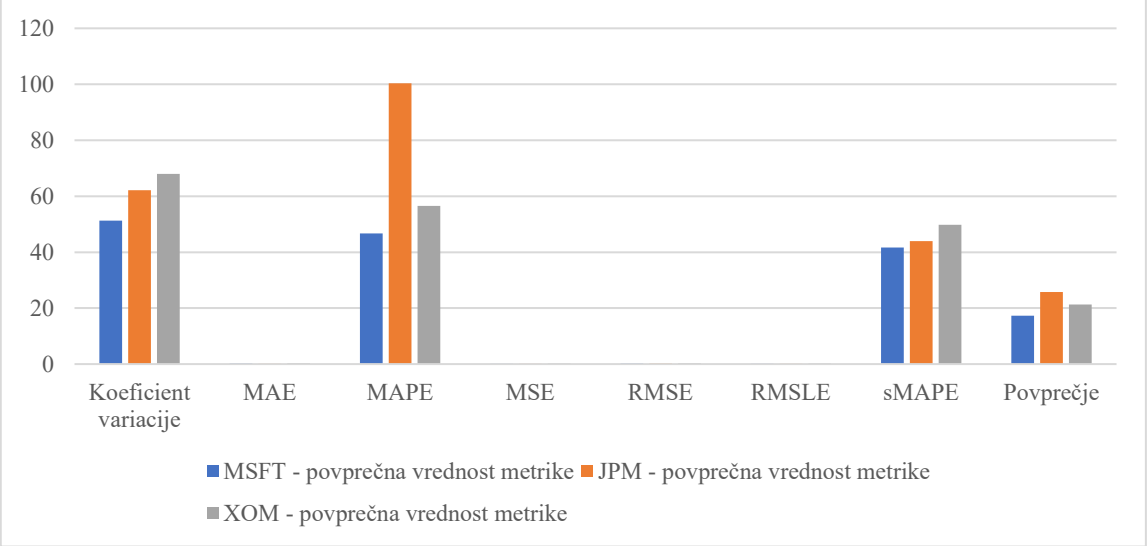
Prej omenjeni problem je veliko bolje viden pri najboljšem ovrednotenju. MAPE je pri JPM še višji, kot je bil prej in niti druge metrike ne morejo v celoti kompenzirati za njegovo

vrednost. Tako je najslabši JPM s povprečno vrednostjo metrike 25,72, sledita mu XOM z 21,26 in MSFT z 17,27. Razliko med najboljšim ovrednotenjem in navadnim ovrednotenjem je mogoče obrazložiti s tem, da grafa ne vsebujeta istih modelov. Najboljše ovrednotenje je namreč mogoče le pri modelih, ki omogočajo uporabo univariatnih in multivariatnih časovnih serij. To je razvidno na sliki 13.

Pri povratnem testu se začnejo pojavljati nekatere večje anomalije. Vrednosti CV in MAPE so tokrat veliko bližje skupaj, kot so bile prej in JPM ne izstopa več toliko, kot je prej. Tokrat je povprečje MSFT 16,29, JPM 18,37, pri XOM pa 13,22. To je razvidno na sliki 14.

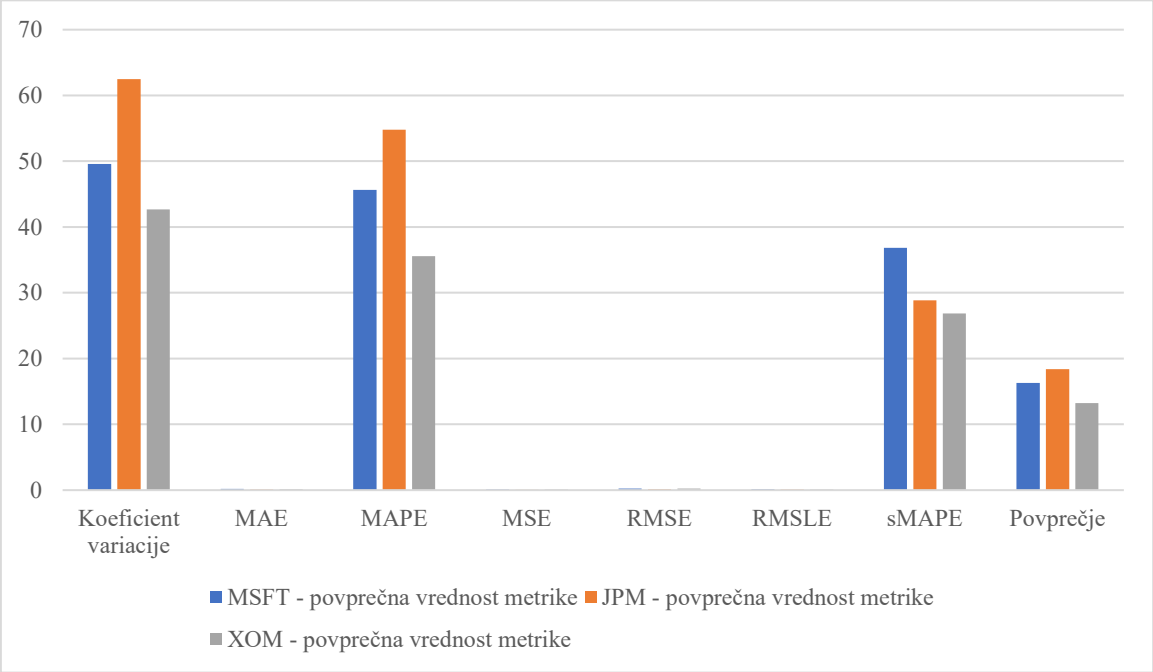
Na osnovi vsega, kar je bilo napisano, je težko narediti pomembnejše zaključke. Očitno so napovedi za JPM nekoliko slabše, a je kljub temu očitno, da je kakovost napovedi razmeroma sprejemljiva v vseh treh primerih. Slabše vrednosti JPM je mogoče v veliki meri pripisati panogi, v kateri deluje, saj ta (bančništvo) praviloma ni tako odporna proti nezaželenim dogodkom. To je razvidno na sliki 15.

Slika 13: Najboljša delnica za točno napoved – najboljše ovrednotenje



Vir: lastno delo.

Slika 14: Najboljša delnica za točno napoved – povratni test



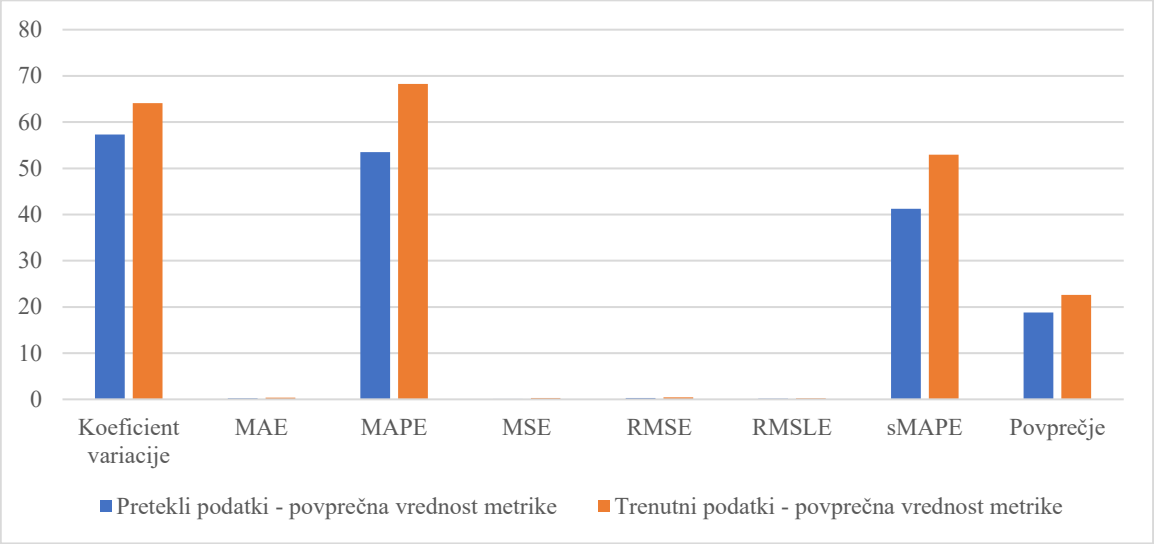
Vir: lastno delo.

4.3.5 Vpliv izbranega časovnega obdobja na rezultate

Peto vprašanje se je navezovalo na napovedovanje modelov z istimi podatki v različnih obdobjih. Primarno časovno obdobje med letoma 2019 in 2022 predstavlja kar velik izziv. V tem času smo imeli oziroma še imamo dve veliki mednarodni krizi, in sicer epidemijo covida-19 in ukrajinsko-rusko vojno. Obe imata zelo močne negativne učinke za svetovno gospodarstvo, zato smo se odločili, da izberemo še eno obdobje, ki je bilo nekoliko manj nestanovitno. Odločili smo se za obdobje med letoma 2015 in 2018, saj takrat z izjemo sirske vojne in begunske krize ni bilo tako obsežnih kriz.

Po kratkem pregledu podatkov ugotovimo, da so pretekli podatki dali nekoliko boljše rezultate kot trenutni podatki. To je vidno tako pri CV, MAPE in sMAPE. Kljub temu razlika med njimi ni tako velika, saj je povprečna vrednost metrike pri preteklih podatkih 18,78, pri trenutnih pa 22,59. To nakazuje, da lahko izvedemo dokaj uspešne napovedi cen delnic tudi v bolj turbulentnih gospodarskih razmerah, ne da bi imeli prevelik izpad natančnosti naših napovedi.

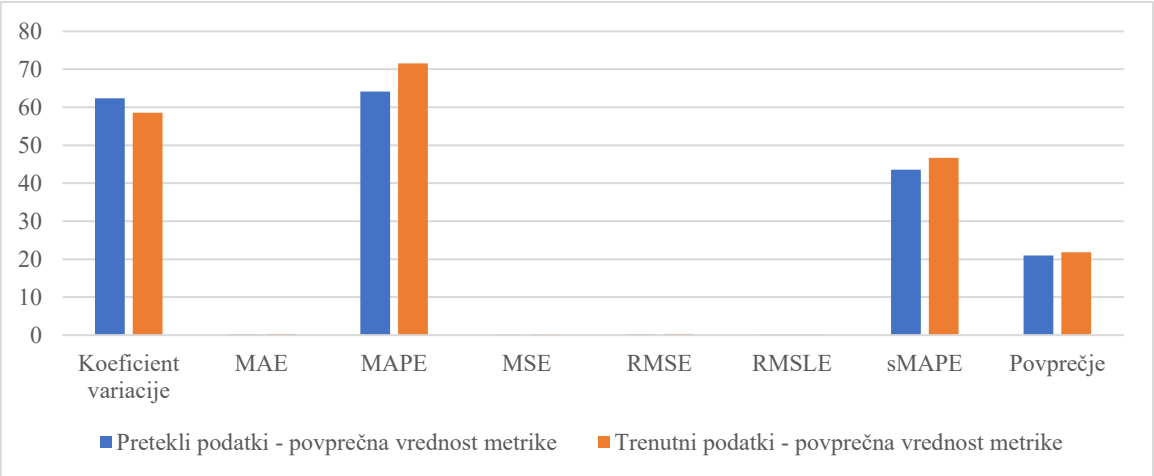
Slika 15: Najboljše časovno obdobje – ovrednotenje



Vir: lastno delo.

Pri najboljšem ovrednotenju lahko vidimo, da je tu razlika med preteklimi podatki in trenutnimi podatki še veliko manjša, kot je bila prej. V primeru CV lahko vidimo, da je povprečna vrednost te matrike še celo boljša pri trenutnih podatkih kot pri preteklih (58,6 proti 62,33). Razlika med skupnimi vrednostmi je prav tako manjša, in sicer 20,01 pri preteklih podatkih in 21,82 pri trenutkih podatkih. Razlike v vrednostih med ovrednotenjem in najboljšim ovrednotenjem bi najlažje obrazložili s tem, da najboljše ovrednotenje vsebuje le modele, ki omogočajo uporabo univariatnih in multivariatnih časovnih serij. Izključno univariatni modeli pa so očitno primernejši za pretekle podatke.

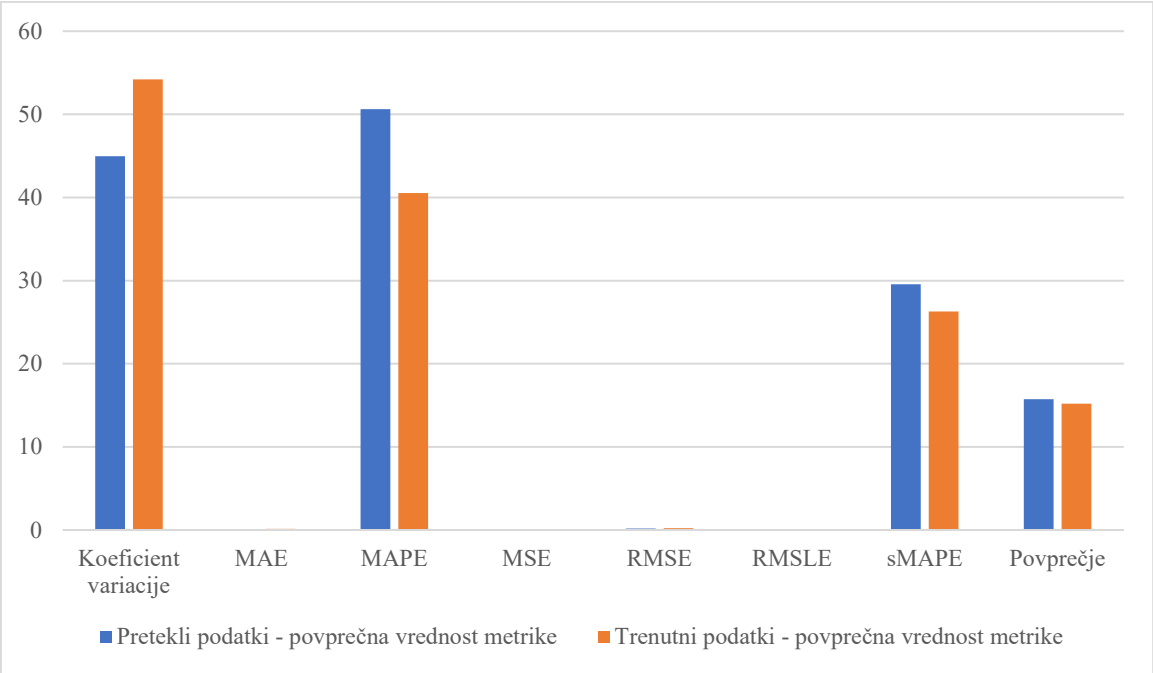
Slika 16: Najboljše časovno obdobje – najboljše ovrednotenje



Vir: lastno delo.

Rezultati povratnega testa so dokaj podobni prejšnjima dvema z izjemo MAPE in sMAPE, ki sta v primeru preteklih podatkov višja – slabša. Povprečna vrednost metrik je pri obeh skoraj enaka pri nekaj čez 15 (pretekli 15,72 in trenutni 15,21).

Slika 17: Najboljše časovno obdobje – povratni test



Vir: lastno delo.

Zaključek, ki ga je mogoče narediti, je, da so rezultati sprejemljivi tako na preteklih kot na trenutnih podatkih. Modeli očitno niso tako občutljivi na poslabšane ekonomske razmere, kot je bilo najprej predpostavljeno.

4.4 Najboljši modeli

V procesu napovedovanja cen delnic smo naredili več napovedi z različnimi parametri in seveda z uporabo različnih modelov. Naredili smo tri različne tabele, kjer je vsaka izmed njih predstavljala eno izmed prej izbranih delnic. Spodaj so vidne vse tri tabele.

Tabela 5: MSFT – najboljši model

Model	Tip_casovne_serije	Tip_izbire_parametrov	Podatki	Vsota_indeksov
DeepTCN	Multivariatni	RFR	Pretekli_podatki	2295
N-BEATS	Multivariatni	RFR	Trenutni_podatki	2179
RNN	Multivariatni	RFR	Trenutni_podatki	2157

Model	Tip_casovne_serije	Tip_izbire_parametrov	Podatki	Vsota_indeksov
RNN	Univariatni	Brez	Pretekli_podatki	2137
N-BEATS	Multivariatni	RFR	Pretekli_podatki	2132
N-BEATS	Univariatni	Brez	Trenutni_podatki	2124
N-BEATS	Univariatni	Brez	Pretekli_podatki	2117
RNN	Multivariatni	RFR	Pretekli_podatki	2114
TFT	Multivariatni	RFR	Trenutni_podatki	2090
TFT	Univariatni	Brez	Trenutni_podatki	2086
LSTM	Univariatni	Brez	Trenutni_podatki	2054
RNN	Univariatni	Brez	Trenutni_podatki	2051
LSTM	Multivariatni	RFR	Trenutni_podatki	2046
LSTM	Univariatni	Brez	Pretekli_podatki	2045
LSTM	Multivariatni	RFR	Pretekli_podatki	2037
Transformer	Multivariatni	RFR	Trenutni_podatki	2029
TFT	Multivariatni	RFR	Pretekli_podatki	2017
TFT	Univariatni	Brez	Pretekli_podatki	1971
Transformer	Multivariatni	RFR	Pretekli_podatki	1953
DeepTCN	Univariatni	Brez	Trenutni_podatki	1802

Vir: lastno delo.

Tabela 6: JPM – najboljši model

Model	Tip_casovne_serije	Tip_izbire_parametrov	Podatki	Vsota_indeksov
DeepTCN	Univariatni	Brez	Pretekli_podatki	2265
RNN	Multivariatni	RFR	Pretekli_podatki	2217
N-BEATS	Multivariatni	RFR	Pretekli_podatki	2211

RNN	Univariatni	Brez	Pretekli_podatki	2207
N-BEATS	Univariatni	Brez	Pretekli_podatki	2195
TFT	Univariatni	Brez	Pretekli_podatki	2171
DeepTCN	Multivariatni	RFR	Pretekli_podatki	2156
LSTM	Univariatni	Brez	Pretekli_podatki	2129
LSTM	Multivariatni	RFR	Pretekli_podatki	2127
Transformer	Multivariatni	RFR	Pretekli_podatki	2124
TFT	Multivariatni	RFR	Pretekli_podatki	2110
N-BEATS	Multivariatni	RFR	Trenutni_podatki	2109
DeepTCN	Univariatni	Brez	Trenutni_podatki	2032
Transformer	Univariatni	Brez	Trenutni_podatki	1967
N-BEATS	Univariatni	Brez	Trenutni_podatki	1944
RNN	Univariatni	Brez	Trenutni_podatki	1940
LSTM	Univariatni	Brez	Trenutni_podatki	1928
TFT	Multivariatni	RFR	Trenutni_podatki	1905
TFT	Univariatni	Brez	Trenutni_podatki	1892
DeepTCN	Multivariatni	RFR	Trenutni_podatki	1890

Vir: lastno delo.

Tabela 7: XOM – najboljši model

Model	Tip_casovne_serije	Tip_izbire_parametrov	Podatki	Vsota_indeksov
LSTM	Univariatni	Brez	Pretekli_podatki	2240
Transformer	Univariatni	Brez	Pretekli_podatki	2234
N-BEATS	Multivariatni	RFR	Pretekli_podatki	2226
N-BEATS	Univariatni	Brez	Pretekli_podatki	2225

DeepTCN	Multivariatni	RFR	Pretekli_podatki	2217
RNN	Multivariatni	RFR	Pretekli_podatki	2217
TFT	Univariatni	Brez	Pretekli_podatki	2210
Transformer	Multivariatni	RFR	Pretekli_podatki	2185
DeepTCN	Univariatni	Brez	Pretekli_podatki	2176
LSTM	Multivariatni	RFR	Pretekli_podatki	2170
TFT	Multivariatni	RFR	Pretekli_podatki	2147
RNN	Univariatni	Brez	Pretekli_podatki	2140
RNN	Multivariatni	RFR	Trenutni_podatki	1758
DeepTCN	Multivariatni	RFR	Trenutni_podatki	1757
RNN	Univariatni	Brez	Trenutni_podatki	1754
LSTM	Univariatni	Brez	Trenutni_podatki	1739
LSTM	Multivariatni	RFR	Trenutni_podatki	1676
N-BEATS	Univariatni	Brez	Trenutni_podatki	1610
N-BEATS	Multivariatni	RFR	Trenutni_podatki	1571
TFT	Univariatni	Brez	Trenutni_podatki	1556

Vir: lastno delo.

Ocenjevanje modelov in iskanje najboljših kombinacij sta bistva magistrskega dela. Ko smo zasnovali magistrsko delo, nas je zanimalo, ali obstaja kakšen model, ki bo v večini primerov dal bistveno boljše rezultate kot drugi. Pri vsaki delnici smo izbrali 20 najboljših modelov. Najboljši modeli imajo v našem primeru vsoto indeksov čez 2000. Ti modeli so tudi najbolj natančni. Zaradi tega bomo upoštevali le modele, ki imajo takšno ali večjo vsoto indeksov. Vsi modeli, ki so se uvrstili, omogočajo delo tako z univariatnimi kot multivariatnimi časovnimi serijami in spadajo med bolj kompleksne, prav tako pa so, z izjemo RNN in LSTM, tudi vsi novejši.

Tabela 8: Najboljši modeli pri različnih delnicah

Model	MSFT	JPM	XOM	Skupaj
DeepTCN	1	3	2	6
RNN	4	2	2	8
LSTM	4	2	2	8
N-BEATS	4	3	2	9

Transformer	1	2	2	5
TFT	3	2	2	7

Vir: lastno delo.

SKLEP

V magistrskem delu smo obravnavali napovedovanje časovnih serij. V našem primeru smo uporabili cene delnic v obliki časovnih serij. Ocenili smo, da so cene delnic ali kateri koli drugi ekonomski podatki dokaj dobri za napovedovanje, saj je njihova kakovost podatkov razmeroma visoka. Napovedovanje cen delnic je ena izmed ključnih težav, s katero se sooča dober del poslovnega sveta. Je tudi vprašanje, povezano z enakostjo med podjetji, saj so za napoved optimalnih rezultatov potrebna obsežna tehnična sredstva in pravilno izobražen kader, oboje pa ceno napovedovanja dviga v višave. Če je bilo pred nekaj leti kakovostno napovedovanje cen delnic s strani posameznikov ali manjših podjetij redko, so danes na voljo dodatne možnosti, in sicer novejši modeli, ki so nastali v zadnjih letih, široka uporaba grafičnih kartic za strojno in globoko učenje, s čimer se je cena napovedi bistveno znižala (prej so bili za marsikateri izračun potrebni superračunalniki ali vsaj namenske delovne postaje). Zmogljivost te strojne opreme se je v zadnjih letih eksponentno povečevala, kar je omogočilo tudi razvoj novih algoritmov, kot so Deep TCN, N-BEATS, več vrst transformerjev in številni drugi. Novi algoritmi pa ne predstavljajo le prednosti, temveč tudi slabosti. Pred desetletjem so za multivariatne časovne serije v glavnem lahko izbirali le med RNN in LSTM, danes pa imamo ogromno izbire. Izbiranje med večjim številom algoritmov pa seveda prinaša še dodatne nevšečnosti. V ta namen smo v teoretičnem delu pregledali večje število algoritmov in si jih med njimi izbrali deset, za katere smo na osnovi lastnosti predvidevali najboljše rezultate v posameznih kategorijah. Prav tako smo izbrali dve časovni obdobji, saj smo želeli preveriti, ali bodo trenutne geopolitične in gospodarske razmere bistveno vplivale na natančnost napovedovanja prihodnjih vrednosti delnic.

Pred začetkom izdelave magistrskega dela smo morali tudi izbrati vrsto analize (tehnična, temeljna in razpoloženjska). Želeli smo uporabiti vse tri vrste analize hkrati, a bi to zahtevalo preveč časa in dela, magistrsko delo pa bi po dolžini naredilo primerljivo doktorski disertaciji. Kot je bilo že prej napisano, podatki o cenah delnic ne predstavljajo resne omejitve in smo jih z lahkoto prenesli s spleta z uporabo knjižnice Yfinance, ki bere podatke s storitve Yahoo Finance. Prav tako je bila razmeroma lahka izbira programskega jezika, v katerem smo naredili praktični del magistrskega dela. Tu smo imeli v bistvu na izbiro le R in Python, za katerega smo se na koncu tudi odločili. V našem primeru so bili podatki dokaj primerni že v privzeti obliki, zato posledično nismo potrebovali obsežnejših popravkov (manjkajoče vrednosti itd.). Za dobro primerjavo različnih parametrov smo uporabili tako univariatne in multivariatne serije, pri katerih pa smo celo uporabili dve različni metodi za izbor značilnic, in sicer FeatureWiz in RFR. Uporabili smo kar deset različnih algoritmov, in sicer ARIMA, RNN, LSTM, Prophet, N-BEATS, StatsForecast Auto ARIMA, DeepTCN eksponentno glajenje, klasični transformer in TFT. Na tej točki se je pojavil problem

ocenjevanja rezultatov. Logično je, da rezultatov nismo mogli oceniti na osnovi dobljenih grafikonov, zato smo morali uporabiti več različnih metrik. Vsi modeli, ki smo jih uporabili, so bili sicer del knjižnice Darts, ki je namenjena napovedovanju časovnih serij. Iz te knjižnice smo uporabili tudi večje število metrik, ki so bile prej omenjene, in sicer: CV, MAE, MAPE, MSE, RMSE, RMSLE in sMAPE. Nastavitve posameznih algoritmov smo v veliki meri osnovali na predlogih iz dokumentacije knjižnice Darts in testov, narejenih v orodju Ray.

V dispoziciji magistrskega dela smo zastavili tri raziskovalna vprašanja, ki smo jim skozi postopek dodali še dve.

V primeru prvega, torej izbora značilnic za omogočanje najboljših rezultatov, smo primerjali dva modela, in sicer FeatureWiz in RFR. Poudariti je treba, da imajo rezultati lahko majhna odstopanja, ker način izbora značilnic ni bil identičen, kar je posledica različnega ovrednotenja značilnic. Ob predpostavki, da so rezultati, ki smo jih dobili, dokaj primerni, je očitno, da smo z RFR dobili bistveno boljše rezultate. Med najboljšimi modeli tako tudi ni bilo nobenega, pri katerem bi bil uporabljen FeatureWiz. To sicer ne pomeni, da orodje ni dobro, temveč samo, da je njegova uporaba v našem specifičnem primeru dokaj nesmiselna.

Pri drugem vprašanju smo raziskali prednost uporabe tehničnih kazalnikov pred uporabo le zadnje dnevne cene. To je bilo eno izmed najbolj ključnih vprašanj, saj bi bilo, če bi univariatne časovne serije dale boljše rezultate kot multivariatne (ki seveda vsebujejo tehnične kazalnike), sploh nesmiselno odgovarjati na prejšnje vprašanje. Sami smo predpostavljali, da bo večje število značilnic omogočalo boljše napovedi. Ta predpostavka je bila le deloma pravilna, saj je natančnost posameznih napovedi v veliki meri odvisna od metode za izbor značilnic. Če je ta izbrana in implementirana pravilno, bodo rezultati nekoliko boljši od univariatnih, kar pa še ne pomeni, da so univariatne napovedi bistveno slabše, sploh glede na to, da se pri njih izognemo izboru značilnic, ki pogosto predstavlja velik problem.

Pri tretjem vprašanju smo predvidevali, da bomo najboljše rezultate dobili iz novejših ter zapletenejših modelov. Mednje smo šteli predvsem Prophet, N-BEATS, Deep TCN, Transformer in TFT. V veliki meri lahko to predpostavko potrdimo, saj je večina teh modelov (z izjemo Propheta) dala zelo dobre rezultate. Prophet je bil sicer konkurenčen drugim metodam za univariatne časovne serije, a ni dal bistveno boljših rezultatov. Verjetno bi bila njegova učinkovitost boljša, če bi uporabili kakšne druge podatke. Med modeli je bil izjemno impresiven N-BEATS, četudi je bil edini model, kjer smo morali parametre, ki jih je predlagalo orodje Ray Tune, nekoliko zmanjšati, saj bi bil pri polnih parametrih čas učenja vsakega modela več kot dve uri, z zmanjšanjem števila epohov (iz 300 na 200) in števila nizov (iz 90 na 60) pa smo zmanjšali čas delovanja na približno 80 do 90 minut. Težava naše predpostavke je izključno v novejših modelih, saj so nekateri nekoliko starejši, a vseeno kompleksnejši modeli dali prav tako dobre oziroma velikokrat celo boljše rezultate. Takšen

primer sta modela RNN in LSTM. Iz našega raziskovanja lahko z gotovostjo trdimo, da je na naših podatkih dobre rezultate lažje doseči s kompleksnejšimi modeli.

Četrto vprašanje je eno izmed dodanih vprašanj. Z njim smo želeli izvedeti, kako se rezultati razlikujejo glede na izbiro delnice. Prvotno se nam ni zdelo smiselno, da bi ga vključili, saj se nam je zdel odgovor logičen. Rezultati se bodo gotovo razlikovali v ekstremnih primerih, kot je Tesla, kjer en Tweet Elona Muska požene vrednost delnic v višave Olimpa ali v globine Tartarusa. Odločili smo se za delnici treh velikih multinacionalk, in sicer Microsoft, JP Morgan in Exxon. Vse tri so ogromna podjetja in vsaka od njih je med največjimi na svojem področju. Ko smo izvedli simulacije, smo ugotovili, da je v veliki večini primerov rezultat zelo odločilen. Najboljše povprečne rezultate smo dobili pri delnicah podjetja Exxon in Microsoft, kar je v skladu s trenutnimi globalnimi trendi. Rezultati pri delnici podjetja JP Morgan so bili sicer nekoliko slabši, a še vedno sprejemljivi. Lahko trdimo, da izbira delnice v primeru primernega izbora modela in njegovih parametrov ni tako pomembna in da lahko razmeroma dobro napovemo veliko večino delnic multinacionalk, če njihove delnice niso bistveno nestanovitnejše od povprečja (primer Tesla).

Peto vprašanje, ki se je navezovalo na napoved modelov v različnih obdobjih, vendar z istimi podatki, praviloma ne bi bilo tako pomembno, če v zadnjih nekaj mesecih ne bi videli veliko napetosti na področju vzhodne Evrope in Azije. To je bilo predvsem vidno v senci ukrajinskega konflikta, ki je imela dokaj velik vpliv na mednarodne trge. Iz tega bi bilo mogoče predpostavljati, da bodo rezultati, ki bodo obsegali podatke v nekoliko mirnejšem obdobju, bistveno boljši. Zavedamo se sicer, da so se tudi med letoma 2015 in 2018 dogajali dogodki, ki so imeli širše ekonomske in geopolitične razsežnosti, a ocenjujemo, da so dogodki zadnjih treh let prodornejši. Ko smo izvedli simulacije, smo to teorijo delno potrdili, saj so bili rezultati za prvo obdobje nekoliko boljši od rezultatov za trenutno obdobje. Razlika je sicer majhna, kar pomeni, da je mogoče razmeroma dobro napovedovati ceno delnic tudi v manj stabilnem gospodarskem okolju.

V procesu izdelave magistrskega dela in predvsem v času izvajanja simulacij smo odkrili, da je v tem postopku skoraj neomejeno število omejitev, na osnovi katerih bi bilo mogoče razširiti raziskavo. Najočitnejši omejitvi sta malo število sistemov za izbor značilnic in uporaba le enega izmed tipov analize (tehnična analiza). Ko smo začeli z izdelavo magistrskega dela, smo predpostavljali, da imamo na voljo veliko manjše število parametrov, ki jih je treba vzeti v obzir, poleg tega pa smo naivno pričakovali, da bomo skozi postopek simulacij odkrili »srebrni naboj«, ki bo v večini primerov »zadel tarčo«. Tega na žalost ni. Kljub temu pa so nekateri modeli z nekaterimi parametri vedno boljši od drugih. To sicer še ne pomeni, da bodo njihovi rezultati vedno zelo dobri, bodo pa vsaj sprejemljivi v veliki večini primerov. Oceniti optimalno rešitev za specifičen primer je skoraj nemogoče in zahteva ogromno število poskusov z ogromnim številom različnih parametrov. Tako je natančnejše napovedovanje časovnih serij še vedno v domeni profesionalcev ali vsaj tistih, ki so za napoved le ene delnice pripravljeni porabiti več ur.

LITERATURA IN VIRI

1. AMD. (2022). *AMD Ryzen™ 9 5950X Desktop Processors*. AMD. Pridobljeno 1. avgusta 2022 iz https://www.amd.com/en/products/cpu/amd-ryzen-9-5950x
2. Ansorge, R. (2022). *Programming in Parallel with CUDA: A Practical Guide* (1. izd.). Cambridge University Press. https://doi.org/DOI: 10.1017/9781108855273
3. Ari, N. & Ustazhanov, M. (2014). Matplotlib in python. *2014 11th International Conference on Electronics, Computer and Computation* (str. 1-6). Abuja: IEEE.
4. AutoViML. (2022). *AutoViML featurewiz*. AutoViML Pridobljeno 1. avgusta 2022 iz https://github.com/AutoViML/featurewiz
5. Bai, S., Kolter, J. Z. & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 14.
6. Bapat, P. (2014). A comprehensive review of sentiment analysis of stocks. *International Journal of Computer Applications*, *106*(18), 1-3.
7. Barrett, P., Hunter, J., Miller, J. T., Hsu, J. C. & Greenfield, P. (2005). Matplotlib--A Portable Python Plotting Package. *ASP ConferenceSeries, Vol. 347*.
8. Bell, S. (2016). *Quantitative Finance For Dummies* (1. izd.). Hoboken: Wiley.
9. Bonaccorso, G. (2018). *Mastering machine learning algorithms: expert techniques to implement popular machine learning algorithms and fine-tune your models* (2. izd.). Birmingham: Packt Publishing.
10. Chen, Y., Kang, Y., Chen, Y. & Wang, Z. (2020). Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing*, *399*, 491-501.
11. Cochrane, J. H. (1990). *Univariate vs. Multivariate Forecasts of GNP Growth and Stock Returns: Evidence and Implications for the Persistence of Shocks, Detrending Methods* (3427). Massachusetts: National Bureau of Economic Research Cambridge.
12. Consoli, S., Reforgiato, R. D. & Saisana, M. (2021). *Data science for economics and finance : methodologies and applications* (1. izd.). Berlin: Springer Nature.
13. Cui, Z., Ke, R., Pu, Z. & Wang, Y. (2020). Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values. *Transportation Research Part C: Emerging Technologies*, *118*, 1-11.
14. Curley, C., Krause, R. M., Feiock, R. & Hawkins, C. V. (2019). Dealing with missing data: A comparative exploration of approaches using the integrated city sustainability database. *Urban affairs review*, *55*(2), 591-615.
15. Dario, P. L. (2018). *Welcome to Technical Analysis Library in Python's documentation!* Pridobljeno 1. avgusta 2022 iz https://technical-analysis-library-in-python.readthedocs.io/en/latest/index.html
16. Darío, P. L. (2022). *Technical Analysis*. Pridobljeno 1. avgusta 2022 iz https://github.com/bukosabino/ta
17. Das, S., Sahu, T. P. & Janghel, R. R. (2021). Stock market forecasting using intrinsic time-scale decomposition in fusion with cluster based modified CSA optimized ELM. *Journal of King Saud University-Computer and Information Sciences*, 1-17.

18. Den Bakker, I. (2017). *Python Deep Learning Cookbook: Over 75 practical recipes on neural network modeling, reinforcement learning, and transfer learning using Python* (1. izd.). Birmingham: Packt Publishing.
19. Scikit-learn developers. (2022). *Sklearn.ensemble.RandomForestRegressor*. Pridobljeno 1. avgusta 2022 iz https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
20. Dodge, Y. (2008). *The Concise Encyclopedia of Statistics* (1. izd.). New York: Springer.
21. Ekman, M. (2021). *Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision, Natural Language Processing, and Transformers Using TensorFlow* (1. izd.). Boston: Addison-Wesley Professional.
22. Etzioni, A. (2011). Behavioral Economics: Toward a New Paradigm [Article]. *American Behavioral Scientist*, *55*(8), 1099-1119. https://doi.org/10.1177/0002764211412355
23. Facebook. (2022). *Prophet: Automatic Forecasting Procedure*. Pridobljeno 1. avgusta 2022 iz https://github.com/facebook/prophet
24. Garlapati, A., Krishna, D. R., Garlapati, K., Yaswanth, N. M. S., Rahul, U. & Narayanan, G. (2021). Stock Price Prediction Using Facebook Prophet and Arima Models. *6th International Conference for Convergence in Technology* (I2CT). Pune: IEEE.
25. Goldstein, I. (2017). Wharton. *How the Stock Market Affects Corporate Decision-making.* Pridobljeno 10. februarja 2022 iz https://knowledge.wharton.upenn.edu/article/how-the-stock-market-affects-corporate-decision-making/
26. Google. (2022). *Welcome to Colab!* Pridobljeno 1. avgusta 2022 iz https://colab.research.google.com/?utm_source=scs-index
27. Guo, W., Li, Z., Gao, C. & Yang, Y. (2022). Stock price forecasting based on improved time convolution network. *Computational Intelligence*, 1-18. https://doi.org/https://doi.org/10.1111/coin.12519
28. Gupta, R. & Chen, M. (2020). Sentiment Analysis for Stock Price Prediction. *2020 IEEE Conference on Multimedia Information Processing and Retrieval* (MIPR). Guangdong: IEEE.
29. Hagedorn, S., Kläbe, S. & Sattler, K.-U. (2021). *Putting Pandas in a Box*. California: CIDR.
30. Hao, J. & Ho, T. K. (2019). Machine learning made easy: a review of scikit-learn package in python programming language. *Journal of Educational and Behavioral Statistics*, *44*(3), 348-361.
31. Herzen, J., Lässig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G. . . . Huguenin, N. (2021). Darts: User-friendly modern machine learning for time series. *arXiv preprint arXiv:2110.03224*.

32. Hu, X. (2021). Stock Price Prediction Based on Temporal Fusion Transformer. *3rd International Conference on Machine Learning, Big Data and Business Intelligence* (MLBDBI). Taiyuan: IEEE.
33. Jan, B. (2020). *TemporalFusionTransformer*. Pridobljeno 1. avgusta 2022 iz https://pytorch-forecasting.readthedocs.io/en/latest/api/pytorch_forecasting.models.temporal_fusion_transformer.TemporalFusionTransformer.html#pytorch_forecasting.models.temporal_fusion_transformer.TemporalFusionTransforme
34. Jansen, S. (2018). *Hands-On Machine Learning for Algorithmic Trading: Design and implement investment strategies based on smart algorithms that learn from data using Python*. Birmingham: Packt Publishing.
35. Jasper, L. (2020). Price action CFD trading strategy: rejection candles. *Flowbank*. Retrieved 28.9. from https://www.flowbank.com/en/research/price-action-cfd-trading-strategy-rejection-candles
36. JetBrains. (2022). *DataSpell: The IDE for Professional Data Scientists*. Pridobljeno 1. avgusta 2022 iz https://www.jetbrains.com/dataspell/
37. Jo, J.-M. (2019). Effectiveness of Normalization Pre-Processing of Big Data to the Machine Learning Performance. *The Journal of the Korea institute of electronic communication sciences*, *14*, 547-552. https://doi.org/10.13067/JKIECS.2019.14.3.547
38. Josef, P., Skipper, S. & Jonathan, T. (2019). *Statsmodels.tsa.holtwinters.ExponentialSmoothing*. statsmodels. Pridobljeno 1. avgusta 2022 iz https://www.statsmodels.org/stable/generated/statsmodels.tsa.holtwinters.ExponentialSmoothing.html
39. Jukes, E. (2018). *Encyclopedia of machine learning and data mining* (2. izd.). Berlin: Springer Nature.
40. Julian, D. (2016). *Designing machine learning systems with Python : design efficient machine learning systems that give you more accurate results* (1. izd.). Birmingham: Packt Publishing.
41. Jupyter. (2022). *Project Jupyter*. Pridobljeno 1. avgusta 2022 iz https://jupyter.org/
42. Kamath, U., Graham, K. L. & Emara, W. (2022). *Transformers for Machine Learning: A Deep Dive* (1. izd.). Boca raton: CRC Press.
43. Koehn, D. (2020). A virtue ethics critique of ethical dimensions of behavioral economics. *Business and Society Review*, *125*, 241-260. https://doi.org/10.1111/basr.12208
44. Kotu, V. & Deshpande, B. (2018). *Data science: concepts and practice* (2. izd.). Burlington: Morgan Kaufmann.
45. Kuber, V., Yadav, D. & Yadav, A. K. (2022). Univariate and Multivariate LSTM Model for Short-Term Stock Market Prediction. *arXiv preprint arXiv:2205.06673*, 1-24.

46. Kumar, R. (2019). *Machine Learning Quick Reference: Quick and essential machine learning hacks for training smart data models*. Birmingham: Packt Publishing.
47. Letham Ben, T. J. S. (2017). *Prophet: forecasting at scale*. Pridobljeno 1. avgusta 2022 iz https://research.facebook.com/blog/2017/02/prophet-forecasting-at-scale/
48. Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R., Tang, J. & Liu, H. (2016). Feature Selection: A Data Perspective. *ACM Computing Surveys*, *50*, 1-73. https://doi.org/10.1145/3136625
49. Li, Y., Li, T. & Liu, H. (2017). Recent advances in feature selection and its applications. *Knowledge and Information Systems*, *53*(3), 551-577.
50. Lim, B., Arık, S. Ö., Loeff, N. & Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, *37*(4), 1748-1764.
51. Liu, H., Dougherty, E. R., Dy, J. G., Torkkola, K., Tuv, E., Peng, H. ... Parsons, L. (2005). Evolving feature selection. *IEEE Intelligent systems*, *20*(6), 64-76.
52. Liu, Y. (2017). *Python Machine Learning By Example*. Birmingham: Packt Publishing.
53. Long, W., Lu, Z. & Cui, L. (2019). Deep learning-based feature engineering for stock price movement prediction. *Knowledge-Based Systems*, *164*, 163-173.
54. Makrehchi, M., Shah, S. & Liao, W. (2013). Stock Prediction Using Event-Based Sentiment Analysis. *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies* (IAT), Atlanta: IEEE.
55. Mallikarjuna, M. & Rao, R. P. (2019). Evaluation of forecasting methods from selected stock market returns. *Financial Innovation*, *5*(1), 40. https://doi.org/10.1186/s40854-019-0157-x
56. Maloumian, N. (2022). Unaccounted forms of complexity: A path away from the efficient market hypothesis paradigm. *Social Sciences & Humanities Open*, *5*(1), 1-5. https://doi.org/https://doi.org/10.1016/j.ssaho.2021.100244
57. Mathew, J. (2020). *PyTorch Artificial Intelligence Fundamentals: A recipe-based approach to design, build and deploy your own AI models with PyTorch 1.x*. Birmingham: Packt Publishing.
58. Melnick, E. L. & Everitt, B. S. (2008). *Encyclopedia of Quantitative Risk Analysis and Assessment*. Hoboken: Wiley.
59. Microsoft. (2022). *Visual Studio Code - Code editing. Redefined.* Pridobljeno 1. avgusta 2022 iz https://code.visualstudio.com/
60. Mitchell, L. E. (2008). *The Speculation Economy : How Finance Triumphed Over Industry*. Oakland: Berrett-Koehler Publishers.
61. Mudinas, A., Zhang, D. & Levene, M. (2019). Market trend prediction using sentiment analysis: lessons learned and paths forward. *arXiv preprint arXiv:1903.05440*, 1-10.
62. Muhammad Ali, P. & Faraj, R. (2014). Data Normalization and Standardization: A Technical Report. *Machine Learning Technical Reports (2014)*, 1-6. https://doi.org/10.13140/RG.2.2.28948.04489

63. Ning, Y., Kazemi, H. & Tahmasebi, P. (2022). A comparative machine learning study for time series oil production forecasting: ARIMA, LSTM, and Prophet. *Computers & Geosciences*, *164*(164), 1-11.
64. Nvidia. (2021). *Specifications*. Pridobljeno 1. avgusta 2022 iz https://www.nvidia.com/en-gb/geforce/graphics-cards/geforce-gtx-1080-ti/specifications/
65. Nvidia. (2022a). *COMPARE 20 SERIES SPECS*. Pridobljeno 1. avgusta 2022 iz https://www.nvidia.com/en-gb/geforce/graphics-cards/compare/?section=compare-20
66. Nvidia. (2022b). *GEFORCE RTX 3090 FAMILY*. Pridobljeno 1. avgusta 2022 iz https://www.nvidia.com/en-eu/geforce/graphics-cards/30-series/rtx-3090-3090ti/
67. Nvidia. (2022c). *NVIDIA cuDNN*. Pridobljeno 1. avgusta 2022 iz https://developer.nvidia.com/cudnn
68. Or, B. (2020). *Temporal Convolutional Networks, The Next Revolution for Time-Series?* Pridobljeno 1. avgusta 2022 iz https://towardsdatascience.com/temporal-convolutional-networks-the-next-revolution-for-time-series-8990af826567
69. Oreshkin, B., Carpo, D., Chapados, N. & Bengio, Y. (2019). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting ICLR 2020, Virtual.
70. Ozgur, C., Colliau, T., Rogers, G. & Hughes, Z. (2017). MatLab vs. Python vs. R. *Journal of data Science*, *15*(3), 355-371.
71. Pal, A. & Prakash, P. (2017). *Practical time series analysis: master time series data processing, visualization, and modeling using Python*. Birmingham: Packt Publishing.
72. Pedersen, L. H. (2015). *Efficiently inefficient : how smart money invests and market prices are determined* (1. izd.). Princeton University Press.
73. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. ... Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *The Journal of machine Learning research*, *12*, 2825-2830.
74. Petrusheva, N. & Jordanoski, I. (2016). Comparative analysis between the fundamental and technical analysis of stocks. *Journal of Process Management and New Technologies*, *4*(2), 26-31.
75. Phetrittikun, R., Suvirat, K., Pattalung, T. N., Kongkamol, C., Ingviya, T. & Chaichulee, S. (2021). Temporal Fusion Transformer for forecasting vital sign trajectories in intensive care patients. The 2021 Biomedical Engineering International Conference (BMEiCON-2021), Ayutthaya, Tajska.
76. Pierdzioch, C. & Rülke, J.-C. (2012). Forecasting stock prices: Do forecasters herd? [Article]. *Economics Letters*, *116*(3), 326-329. https://doi.org/10.1016/j.econlet.2012.03.019
77. Ran, A. (2022). Yfinance. *Pypi*. Pridobljeno 1. avgusta 2022 iz https://pypi.org/project/yfinance/
78. Rao, T. & Srivastava, S. (2012). Analyzing stock market movements using twitter sentiment analysis. *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 1-5.

79. Raschka, S. (2015). *Python Machine Learning: Unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics*. Birmingham: Packt Publishing.
80. Raschka, S., Julian, D. & Hearty, J. (2016). *Python: deeper insights into machine learning*. Birmingham: Packt Publishing.
81. Renaud, O. & Victoria-Feser, M.-P. (2010). A robust coefficient of determination for regression. *Journal of Statistical Planning and Inference*, *140*(7), 1852-1862.
82. Richert, W. & Coelho, L. P. (2013). *Building machine learning systems with Python: Master the art of machine learning with Python and build effective machine learning systems with this intensive hands-on guide*. Birmingham: Packt Publishing.
83. Saleh, H. (2019). *Applied Deep Learning with PyTorch: Demystify Neural Networks with PyTorch*. Birmingham: Packt Publishing.
84. Schintler, L. A. & McNeely, C. L. (2019). *Encyclopedia of Big Data Technologies* (1 izd.). Berlin: Springer Nature.
85. Sewell, M. (2007). *Technical analysis* (str. 1-8). London: Department of Computer Science University College London.
86. Sezer, O., Gudelek, U. & Ozbayoglu, M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, *90*, 106181. https://doi.org/10.1016/j.asoc.2020.106181
87. Sharma, A., Sachdeva, S. N. & Aggarwal, P. (2021). Predicting IRI Using Machine Learning Techniques. *International Journal of Pavement Research and Technology*, 10. https://doi.org/10.1007/s42947-021-00119-w
88. Soloviev, V., Titov, N., & Smirnova, E. (2020, 2020). Coking coal railway transportation forecasting using ensembles of ElasticNet, LightGBM, and Facebook prophet. In G. Goos, J. Hartmanis, E. Bertino, W. Gao, B. Steffen, G. Woeginger, & M. Yung, *Lecture Notes in Computer Science* 6th International Conference, LOD 2020, Siena, Italija.
89. Storti, D. & Yurtoglu, M. (2015). *CUDA for engineers: an introduction to high-performance parallel computing* (1. izd.). Addison-Wesley Professional.
90. Subramanian, V. (2018). *Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch* (1. izd.). Birmingham: Packt Publishing.
91. Sussman, A. L. (2021). UCLA Anderson. *How Bond and Stock Prices Combine to Influence Corporate Investment*. Pridobljeno 12. februarja 2022 iz https://anderson-review.ucla.edu/how-bond-and-stock-prices-combine-to-influence-corporate-investment/
92. Swamidass, P. (2000). *Encyclopedia of Production and Manufacturing Management* (1. izd.). Kluwer Academic Publishers. https://doi.org/10.1007/1-4020-0612-8_1037
93. Swamynathan, M. (2019). *Mastering machine learning with python in six steps: A practical implementation guide to predictive data analytics using python* (2. izd.). Apress.

94. Taylor, S. G. (2022). *Pmdarima.arima .AutoARIMA*. Pridobljeno 1. avgusta 2022 iz https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.AutoARIMA.html
95. Pandas. (2022a). *pandas.DataFrame.asfreq*. Pridobljeno 1. avgusta 2022 iz https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.asfreq.html
96. Pandas. (2022b). *Pandas.DataFrame.fillna*. Pridobljeno 1. avgusta 2022 iz https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html
97. Tsuchiya, Y. (2021). Crises, market shocks, and herding behavior in stock price forecasts. *Empirical Economics*, *61*(2), 919-945. https://doi.org/10.1007/s00181-020-01894-4
98. Unit8. (2021). *Darts Unifying time series forecasting models from ARIMA to Deep Learning*. Europython. Pridobljeno 1. avgusta 2022 iz https://ep2021.europython.eu/media/conference/slides/5KySKcS-darts-unifying-time-series-forecasting-models-from-arima-to-de_MYlSmg0.pdf
99. Unit8. (2022a). *About us*. Pridobljeno 1. avgusta 2022 iz https://unit8.com/about/
100. Unit8. (2022b). *ARIMA*. Pridobljeno 1. avgusta 2022 iz https://unit8co.github.io/darts/generated_api/darts.models.forecasting.arima.html
101. Unit8. (2022c). *Darts: Time Series Made Easy in Python*. Pridobljeno 1. avgusta 2022 iz https://unit8.com/resources/darts-time-series-made-easy-in-python/
102. Unit8. (2022d). *Exponential Smoothing*. Pridobljeno 1. avgusta 2022 iz https://unit8co.github.io/darts/generated_api/darts.models.forecasting.exponential_smoothing.html
103. Unit8. (2022e). *Facebook Prophet*. Pridobljeno 1. avgusta 2022 iz https://unit8co.github.io/darts/generated_api/darts.models.forecasting.prophet_model.html
104. Unit8. (2022f). *N-BEATS*. Pridobljeno 1. avgusta 2022 iz https://unit8co.github.io/darts/generated_api/darts.models.forecasting.nbeats.html
105. Unit8. (2022g). *Recurrent Neural Networks*. Pridobljeno 1. avgusta 2022 iz https://unit8co.github.io/darts/generated_api/darts.models.forecasting.rnn_model.html#recurrent-neural-networks
106. Unit8. (2022h). *StatsForecastAutoARIMA*. Pridobljeno 1. avgusta 2022 iz https://unit8co.github.io/darts/generated_api/darts.models.forecasting.sf_auto_arima.html
107. Unit8. (2022i). *Temporal Fusion Transformer (TFT)*. Pridobljeno 1. avgusta 2022 iz https://unit8co.github.io/darts/generated_api/darts.models.forecasting.tft_model.html
108. Unit8. (2022j). *Time Series Made Easy in Python.* Pridobljeno 1. avgusta 2022 iz https://unit8co.github.io/darts/
109. Unit8. (2022k). *Timeseries*. Pridobljeno 1. avgusta 2022 iz https://unit8co.github.io/darts/generated_api/darts.timeseries.html
110. Unit8. (2022l). *Transformer Model*. Pridobljeno 1. avgusta 2022 iz https://unit8co.github.io/darts/generated_api/darts.models.forecasting.transformer_model.html

111. Vantage, A. (2022). *Stock API, Reimagined*. Alpha Vantage. Pridobljeno 1. avgusta 2022 iz https://www.alphavantage.co/
112. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., ... Polosukhin, I. (2017). Attention Is All You Need. 31st Conference on Neural Information Processing Systems (str. 6000-6010). Long Beach: Association for Computing Machinery.
113. Vishwas, B. V. & Patel, A. (2020). *Hands-on Time Series Analysis with Python: From Basics to Bleeding Edge Techniques*. Apress California. https://doi.org/10.1007/978-1-4842-5992-4
114. Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J. & Sun, L. (2022). Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 1-8.
115. Wu, B., Wang, L. & Zeng, Y.-R. (2022). Interpretable wind speed prediction with multivariate time series and temporal fusion transformers. *Energy*, *252*, 1-24.
116. Wu, N., Green, B., Ben, X. & O'Banion, S. (2020). Deep transformer models for time series forecasting: The influenza prevalence case. *37 th International Conference on Machine Learning*. Dunaj.
117. Xie, Z., Zhang, W., Ding, H., & Ma, L. (2020a, 2020). MsFcNET: Multi-scale Feature-Crossing Attention Network for Multi-field Sparse Data. In R. Goebel, Y. Tanaka, & W. Wahlster, *Lecture Notes in Artificial Intelligence* 24th Pacific-Asia Conference, Singapur.
118. Zaccone, G. & Karim, M. R. (2018). *Deep learning with tensorFlow: Explore neural networks and build intelligent systems with Python* (2. izd.). Birmingham: Packt Publishing.
119. Zhang, M. (2021). Design of simulated stock forecasting trading system based on time series. *IEEE 4th International Conference on Information Systems and Computer Aided Education* (ICISCAE). Dalian: IEEE.
120. Zhao, Y. & Chen, Z. (2021). Forecasting stock price movement: New evidence from a novel hybrid deep learning model. *Journal of Asian Business and Economic Studies*, *29*, 91-104.
121. Zocca, V., Spacagna, G., Slater, D. & Roelants, P. (2017). *Python Deep Learning: Next generation techniques to revolutionize computer vision, AI, speech and data analysis* (1. izd). Birmingham: Packt Publishing.

PRILOGE

Priloga 1: Generiranje sospremenljivk.

```
stock_year = datetime_attribute_timeseries(series_transformed, attribute="year")
stock_month = datetime_attribute_timeseries(series_transformed, attribute="month")

future_stock_covariates = stock_year.stack(stock_month)

scaler_dt_stock = Scaler()
future_stock_covariates = scaler_dt_stock.fit_transform(future_stock_covariates)

future_stock_train_covariates, future_stock_val_covariates = future_stock_covariates.split_after(0.75)

# add the day as a covariate
past_stock_covariates = datetime_attribute_timeseries(
    series_transformed, attribute="day", one_hot=True
)
scaler_day = Scaler()
past_stock_covariates = scaler_day.fit_transform(past_stock_covariates)
past_stock_train_covariates, past_stock_val_day = past_stock_covariates.split_after(0.75)
```

Vir: lastno delo.

Priloga 2: RNN univariatni.

```
my_model_RNN_with_covariates1 = RNNModel(
    model="RNN",
    dropout=0.2,
    batch_size=16,
    n_epochs=300,
    model_name="RNN_model_with_covariates1",
    log_tensorboard=True,
    random_state=0,
    optimizer_kwargs={"lr": 1e-3},
    training_length=20,
    input_chunk_length=30,
    output_chunk_length=15,
     likelihood=QuantileRegression(
        quantiles=quantiles
    ),
    force_reset=True,
    save_checkpoints=True,
    pl_trainer_kwargs={
      "accelerator": "gpu",
      "gpus": [0]
    },
)

my_model_RNN_with_covariates1.fit(
    train_transformed,
    future_covariates=future_stock_covariates,
    val_series=val_transformed,
    val_future_covariates=future_stock_covariates,
    verbose=True,
)

pred_series = my_model_RNN_with_covariates1.predict(n=260,
future_covariates=future_stock_covariates, num_samples=100)
plt.figure(figsize=(8, 5))
transformer.inverse_transform(series_transformed).plot(label="actual")
transformer.inverse_transform(pred_series).plot(label="forecast")
plt.title("Forecast")
plt.legend()

print(str(my_model_RNN_with_covariates1) + " Coefficient of variation:
{:.2f}%".format(coefficient_of_variation(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " DTW metric:
{:.2f}%".format(dtw_metric(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " MAE:
{:.2f}%".format(mae(val_transformed, pred_series)))
print((str(my_model_RNN_with_covariates1) + " MAPE: {:.2f}%".format(
mape(
        transformer.inverse_transform(val_transformed),
```

```
            transformer.inverse_transform(pred_series),
        )
    )
))
print(str(my_model_RNN_with_covariates1) + " MARRE: {:.2f}%".format(marre(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " MSE: {:.2f}%".format(mse(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " OPE: {:.2f}%".format(ope(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " R2 score: {:.2f}%".format(r2_score(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " RMSE: {:.2f}%".format(rmse(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " RMSLE: {:.2f}%".format(rmsle(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " SAMPE: {:.2f}%".format(smape(val_transformed, pred_series)))

row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija","Coefficient_of_variation",coefficient_of_variation(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija","DTW_metric",dtw_metric(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija","MAE",mae(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija","MAPE",mape(transformer.inverse_transform(val_transformed), transformer.inverse_transform(pred_series),)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija","MARRE",marre(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija","MSE",mse(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija","OPE",ope(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
```

```
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija","R2_score",r2_score(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija","RMSE",rmse(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija","RMSLE",rmsle(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija","SAMPE",smape(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])

my_model_RNN_with_covariates_best1 = my_model_RNN_with_covariates1.load_from_checkpoint(model_name="RNN_model_with_covariates1", best=True)
pred_series = my_model_RNN_with_covariates_best1.predict(n=260, future_covariates=future_stock_covariates, num_samples=100)
plt.figure(figsize=(8, 5))
transformer.inverse_transform(series_transformed).plot(label="actual")
transformer.inverse_transform(pred_series).plot(label="forecast")
plt.title("Forecast")
plt.legend()

print(str(my_model_RNN_with_covariates_best1) + " Coefficient of variation: {:.2f}%".format(coefficient_of_variation(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " DTW metric: {:.2f}%".format(dtw_metric(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " MAE: {:.2f}%".format(mae(val_transformed, pred_series)))
print((str(my_model_RNN_with_covariates_best1) + " MAPE: {:.2f}%".format(
mape(
        transformer.inverse_transform(val_transformed),
        transformer.inverse_transform(pred_series),
    )
  )
))
print(str(my_model_RNN_with_covariates_best1) + " MARRE: {:.2f}%".format(marre(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " MSE: {:.2f}%".format(mse(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " OPE: {:.2f}%".format(ope(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " R2 score: {:.2f}%".format(r2_score(val_transformed, pred_series)))
```

```
print(str(my_model_RNN_with_covariates_best1) + " RMSE: {:.2f}%".format(rmse(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " RMSLE: {:.2f}%".format(rmsle(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " SAMPE: {:.2f}%".format(smape(val_transformed, pred_series)))

row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","Coefficient_of_variation",coefficient_of_variation(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","DTW_metric",dtw_metric(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","MAE",mae(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","MAPE",mape(transformer.inverse_transform(val_transformed), transformer.inverse_transform(pred_series),)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","MARRE",marre(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","MSE",mse(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","OPE",ope(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","R2_score",r2_score(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","RMSE",rmse(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","RMSLE",rmsle(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
```

```
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","SAMPE",smape(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])

backtest_series = my_model_RNN_with_covariates1.historical_forecasts(
    series_transformed,
    future_covariates=future_stock_covariates,
    start=0.5,
    forecast_horizon=6,
    num_samples=100,
    retrain=False,
    verbose=True,
)
plt.figure(figsize=(8, 5))
transformer.inverse_transform(series_transformed).plot(label="actual")
transformer.inverse_transform(backtest_series).plot(label="backtest")
plt.legend()
plt.title("Backtest")

print(str(my_model_RNN_with_covariates1) + " Coefficient of variation: {:.2f}%".format(coefficient_of_variation(series_transformed, backtest_series )))
print(str(my_model_RNN_with_covariates1) + " DTW metric: {:.2f}%".format(dtw_metric(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " MAE: {:.2f}%".format(mae(series_transformed, backtest_series)))
print((str(my_model_RNN_with_covariates1) +
   " MAPE: {:.2f}%".format(
      mape(
         transformer.inverse_transform(series_transformed),
         transformer.inverse_transform(backtest_series),
      )
   )
))
print(str(my_model_RNN_with_covariates1) + " MARRE: {:.2f}%".format(marre(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " MSE: {:.2f}%".format(mse(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " OPE: {:.2f}%".format(ope(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " R2 score: {:.2f}%".format(r2_score(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " RMSE: {:.2f}%".format(rmse(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " RMSLE: {:.2f}%".format(rmsle(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " SAMPE: {:.2f}%".format(smape(series_transformed, backtest_series)))
```

```
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Backtest","Coefficient_of_variation",coefficient_of_variation(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Backtest","DTW_metric",dtw_metric(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Backtest","MAE",mae(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Backtest","MAPE",mape(transformer.inverse_transform(series_transformed), transformer.inverse_transform(backtest_series))]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Backtest","MARRE",marre(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Backtest","MSE",mse(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Backtest","OPE",ope(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Backtest","R2_score",r2_score(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Backtest","RMSE",rmse(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Backtest","RMSLE",rmsle(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"Univariatni","Trenutni_podatki","260dni","RNN","Z_kovariatami","Backtest","SAMPE",smape(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
```

Vir: lastno delo.

Priloga 3: RNN multivariatni.

```
my_model_RNN_with_covariates1 = RNNModel(
    model="RNN",
    dropout=0.12,
    batch_size=16,
    n_epochs=300,
    model_name="RNN_model_with_covariates1",
    log_tensorboard=True,
    random_state=0,
    optimizer_kwargs={"lr": 1e-3},
    training_length=20,
    input_chunk_length=40,
    output_chunk_length=10,
     likelihood=QuantileRegression(
        quantiles=quantiles
    ),
    force_reset=True,
    save_checkpoints=True,
    pl_trainer_kwargs={
      "accelerator": "gpu",
      "gpus": [0]
    },
)

my_model_RNN_with_covariates1.fit(
    train_transformed,
    future_covariates=future_stock_covariates,
    val_series=val_transformed,
    val_future_covariates=future_stock_covariates,
    verbose=True,
)

pred_series = my_model_RNN_with_covariates1.predict(n=260,
future_covariates=future_stock_covariates, num_samples=100)
plt.figure(figsize=(8, 5))
transformer.inverse_transform(series_transformed)["Close"].plot(label="actual")
transformer.inverse_transform(pred_series)["Close"].plot(label="forecast")
plt.title("Forecast")
plt.legend()

print(str(my_model_RNN_with_covariates1) + " Coefficient of variation:
{:.2f}%".format(coefficient_of_variation(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " DTW metric:
{:.2f}%".format(dtw_metric(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " MAE:
{:.2f}%".format(mae(val_transformed, pred_series)))
print((str(my_model_RNN_with_covariates1) + " MAPE: {:.2f}%".format(
mape(
        transformer.inverse_transform(val_transformed),
```

```
            transformer.inverse_transform(pred_series),
        )
    )
))
print(str(my_model_RNN_with_covariates1) + " MARRE: {:.2f}%".format(marre(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " MSE: {:.2f}%".format(mse(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " OPE: {:.2f}%".format(ope(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " R2 score: {:.2f}%".format(r2_score(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " RMSE: {:.2f}%".format(rmse(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " RMSLE: {:.2f}%".format(rmsle(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates1) + " SAMPE: {:.2f}%".format(smape(val_transformed, pred_series)))

row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija","Coefficient_of_variation",coefficient_of_variation(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija","DTW_metric",dtw_metric(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija","MAE",mae(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija","MAPE",mape(transformer.inverse_transform(val_transformed), transformer.inverse_transform(pred_series),)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija","MARRE",marre(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija","MSE",mse(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija","OPE",ope(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
```

```
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija","R2_score",r2_score(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija","RMSE",rmse(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija","RMSLE",rmsle(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija","SAMPE",smape(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])

my_model_RNN_with_covariates_best1 = my_model_RNN_with_covariates1.load_from_checkpoint(model_name="RNN_model_with_covariates1", best=True)
pred_series = my_model_RNN_with_covariates_best1.predict(n=260, future_covariates=future_stock_covariates, num_samples=100)
plt.figure(figsize=(8, 5))
transformer.inverse_transform(series_transformed)["Close"].plot(label="actual")
transformer.inverse_transform(pred_series)["Close"].plot(label="forecast")
plt.title("Forecast")
plt.legend()

print(str(my_model_RNN_with_covariates_best1) + " Coefficient of variation: {:.2f}%".format(coefficient_of_variation(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " DTW metric: {:.2f}%".format(dtw_metric(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " MAE: {:.2f}%".format(mae(val_transformed, pred_series)))
print((str(my_model_RNN_with_covariates_best1) + " MAPE: {:.2f}%".format(
mape(
        transformer.inverse_transform(val_transformed),
        transformer.inverse_transform(pred_series),
    )
  )
))
print(str(my_model_RNN_with_covariates_best1) + " MARRE: {:.2f}%".format(marre(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " MSE: {:.2f}%".format(mse(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " OPE: {:.2f}%".format(ope(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " R2 score: {:.2f}%".format(r2_score(val_transformed, pred_series)))
```

```
print(str(my_model_RNN_with_covariates_best1) + " RMSE: {:.2f}%".format(rmse(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " RMSLE: {:.2f}%".format(rmsle(val_transformed, pred_series)))
print(str(my_model_RNN_with_covariates_best1) + " SAMPE: {:.2f}%".format(smape(val_transformed, pred_series)))

row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","Coefficient_of_variation",coefficient_of_variation(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","DTW_metric",dtw_metric(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","MAE",mae(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","MAPE",mape(transformer.inverse_transform(val_transformed), transformer.inverse_transform(pred_series),)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","MARRE",marre(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","MSE",mse(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","OPE",ope(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","R2_score",r2_score(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","RMSE",rmse(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row = [stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovariatami","Evalvacija_najboljsa","RMSLE",rmsle(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
```

```
row                                                                                                =
[stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovar
iatami","Evalvacija_najboljsa","SAMPE",smape(val_transformed, pred_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])

backtest_series = my_model_RNN_with_covariates1.historical_forecasts(
        series_transformed,
        future_covariates=future_stock_covariates,
        start=0.5,
        forecast_horizon=6,
        num_samples=100,
        retrain=False,
        verbose=True,
)
plt.figure(figsize=(8, 5))
transformer.inverse_transform(series_transformed)["Close"].plot(label="actual")
transformer.inverse_transform(backtest_series)["Close"].plot(label="backtest")
plt.legend()
plt.title("Backtest")

print(str(my_model_RNN_with_covariates1) + " Coefficient of variation:
{:.2f}%".format(coefficient_of_variation(series_transformed, backtest_series )))
print(str(my_model_RNN_with_covariates1) + " DTW metric:
{:.2f}%".format(dtw_metric(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " MAE:
{:.2f}%".format(mae(series_transformed, backtest_series)))
print((str(my_model_RNN_with_covariates1) +
    " MAPE: {:.2f}%".format(
        mape(
            transformer.inverse_transform(series_transformed),
            transformer.inverse_transform(backtest_series),
        )
    )
))
print(str(my_model_RNN_with_covariates1) + " MARRE:
{:.2f}%".format(marre(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " MSE:
{:.2f}%".format(mse(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " OPE:
{:.2f}%".format(ope(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " R2 score:
{:.2f}%".format(r2_score(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " RMSE:
{:.2f}%".format(rmse(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " RMSLE:
{:.2f}%".format(rmsle(series_transformed, backtest_series)))
print(str(my_model_RNN_with_covariates1) + " SAMPE:
{:.2f}%".format(smape(series_transformed, backtest_series)))
```

```
row =
[stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovar
iatami","Backtest","Coefficient_of_variation",coefficient_of_variation(series_transform
ed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row =
[stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovar
iatami","Backtest","DTW_metric",dtw_metric(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row =
[stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovar
iatami","Backtest","MAE",mae(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row =
[stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovar
iatami","Backtest","MAPE",mape(transformer.inverse_transform(series_transformed),
transformer.inverse_transform(backtest_series))]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row =
[stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovar
iatami","Backtest","MARRE",marre(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row =
[stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovar
iatami","Backtest","MSE",mse(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row =
[stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovar
iatami","Backtest","OPE",ope(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row =
[stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovar
iatami","Backtest","R2_score",r2_score(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row =
[stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovar
iatami","Backtest","RMSE",rmse(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row =
[stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovar
iatami","Backtest","RMSLE",rmsle(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
row =
[stockName,"FeatureWiz","Multivariatni","Pretekli_podatki","260dni","RNN","Z_kovar
iatami","Backtest","SAMPE",smape(series_transformed, backtest_series)]
PodatkiMatrike = np.vstack([PodatkiMatrike, row])
```

Vir: lastno delo.