VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
INFORMATIKOS KATEDRA

Kursinis darbo projektas

Automatizuotas kriptovaliutų prekybos robotas

(Automated cryptocurrency trading bot)

Atliko: 4 kurso 1 grupės studentas
Matas Kaminskas (parašas)

Darbo vadovas:
J. Asist., Dr. Igor Katin (parašas)

Vilnius
2023

Turinys

Įvadas

Pasaulis vis labiau modernėja, technologijos tampa vis išmanesnės ir našesnės, nei bet kada anksčiau. Šiais laikais turime kontraversiškai pagarsėjusią kriptovaliutų rinką, kuri yra kaip skaidresnė atsvara standartinei akcijų biržai. Po 2007-2008 metų įvykusios finansų krizės, visai neužilgo - [Nak08] 2008 metais, atsirado pirmoji kriptovaliuta - Bitcoin. Tai yra decentralizuotą ir nepriklausoma elektroninę valiutą, paremta blokų grandinės technologija. Dabartiniais apskaičiavimais šios rinkos bendra vertė viršija 800 milijardų JAV dolerių [Coi], nors kiek daugiau nei prieš metus ši vertė buvo dvigubai didesnė - 1.6 trilijonų JAV dolerių, o tarp 2018 ir 2020 metų vertė daugmaž buvo stabili - apie 100 milijardų JAV dolerių. Šie duomenys dar kartą pabrėžia kriptovaliutų rinkos nepastovumą, bei ženkliai padidėjusi susidomėjimą, populiarumą visuomenėje.

Pagrindinis skirtumas tarp kriptovaliutų rinkos ir tradicines akcijų biržos yra jog mainai vyksta 24 valandas per parą. Akcijų biržoje yra nustatytos pagrindinės darbo valandos, jos dažniausiai yra tokios pat kaip vietos darbo valandos ir pagrindiniai mainai vyksta šiomis valandomis, todėl stebėti tendencijas ir realizuoti strategijas yra įmanomas darbas asmeniui. Kai rinka yra prieinama visa parą dirbant be sustojimo yra nepraktiška, bei neįmanoma stebėti ir atlikti prasmingus mainus rinkoje. Atsiranda puiki terpė automatizuoti šį procesą - įdarbinti automatizuotą prekybos robotą, kuris gebėtų analizuoti rinkos duomenis ir prognozuoti tolimesnė rinkos eigą.

Pratęsiant kursinio darbo temą, šio kursinio darbo projekto tikslas yra sukurti automatizuotą kriptovaliutų prekybos robotą. Darbe pritaikomas autoregresinių modelių veikimas, pagal kuriuos robotas galės remtis atliekant kriptovaliutų rinkos mainus.

Tikslui įgyvendinti keliami uždaviniai:

- mokslinės literatūros analizė,
- autoregresinių modelių analizė ir pritaikymas,
- sukurti robotą gebanti remtis ištirtų modelių prognozėmis,
- paleisti robotą prekiauti testiniame tinkle,
- ištirti gautus rezultatus.

1. Pagrindinė tiriamoji dalis

Pagrindinė ir pirmoji kriptovaliuta - Bitcoin. Nors pirmieji 50 BTC buvo ”iškasti” 2009 metų pradžioje, Bitcoin susilaukė didesnio dėmesio tik 2013 metais[Mac+14]. Ši kriptovaliuta buvo laikoma ateities valiuta dėl savo naujoviškumo ir decentralizavimo, bet susilaukdavo ir neigiamų atsilipiemų dėl atsiskaitymo nelegaliuose prekybos vietuose internete. Vienas pirmųjų ir itin didelės apyvartos sulaukęs nelegalių prekių tinklalapis ”Silkroad” atsiskaitymui už prekes naudojo Bitcoin. Didelė apyvarta skatino bitcoin augimą, susidomėjimą ir kitų kriptovaliutų progresą. Stebint kriptovaliutų rinką matoma koreliacija tarp Bitcoin kainos ir visų kitų kriptovaliutų kainos, taip dar kartą pabrėžiama, jog ši valiuta yra pagrindinė rinkoje. Kyla bitcoin kaina - kyla kitų valiutų kaina, krenta bitcoin kaina - krenta kitų valiutų kaina. Retesniais atvejais kriptovaliutos sparčiau keičia kainą dėl pažangaus vystymosi ar dėl esančių problemų.

Dauguma kriptovaliutų yra riboto kiekio, tad jeigu prekybos robotas uždirba konkrečios kriptovaliutos mažai, jų kainą gali keleriopai išaugti tikrų valiutų atžvilgiu, todėl į kriptovaliutą galima žiūrėti ir kaip į investiciją. 2018 metais, nepaisant to, kad kriptovaliuta buvo galima atsiskaityti už paslaugas ar prekes, populiauriausia paskirtis buvo investicija. Viena iš priežaščių - tuo metu kaina buvo pakilus nenuspėjamai aukštai[Gar+18]. Akcijų birža, rinka naudojama investiciniais tikslais, jau kurį laiką yra analizuojama dėl savo finansinės naudos ir sudėtingumo. Šios rinkos turi panašumų, todėl galima tirti akcijų biržos analizė literatūroje.

Šiais laikais vis daugiau žmonių naudojasi technologijomis ir nori nepriklausomybės nuo institucijų išleidžiamų ir manipuliuojami valiutų, infliacijos ir kitų dalykų, kurie mažina finansinių instituticijų pasitikėjimą. Kriptovaliutos yra paremtos blokų grandinės technologija, todėl daugelis jų yra decentralizuoti tinklai. Visa blokų informacija yra viešai prieinama publikai, puikiai žinoma tarp kokių šalių vyksta sandoriai, taip suteikiamas skaidresnis ir patikimesnis būdas publikai turėti savo nepriklausomą rinką. Ypatingas kriptovaliutų bruožas yra tas, kad jų paprastai neišleidžia jokia centrinė institucija, todėl teoriškai jos nėra apsaugotos nuo vyriausybės kišimosi. Ankščiau minėtos savybės suteikia anonimiškumo, todėl tai sudaro palankias sąlygas klestėti sukčiavimo atvejams, ”pump-and-dump” ar ”ponzi” schemoms. Dažniausiai sutinkama yra ”pump-and-dump” schema, kuri apibūdina procesą, kai valiutos paklausa yra iš anksto apgalvotai ir trumpam padidinima, taip pakeliant jos kainą, vėliau, kai kainą yra pasiekusi tinkamą tašką, organizatoriai, staigiai parduoda savo turimą kiekį, taip pasipelnydami[XL19].

Pagrindinis tyrimas šiame darbe yra tinkamiausio auto regresinio modelio parinkimas automatizuotam robotui lošėjui. Lošimo procesas tam tikrą laiką gali būti atliekamas žmogaus, stebėti besikeičiančias kainas ir atitinkamai elgtis, tačiau toks darbas yra monotoniškas ir atsiranda natūrali terpė jį automatizuoti. Automatizavimui reikia taikyti pasirinktą strategiją, kuri spręstų kada yra tinkamas metas įsigyti kriptovaliutą, o kada parduoti, bet pirmiausia reikalingas tinkamas prognozavimo modelis. Auto regresiniai modeliai taikomi laiko eilutėmis yra paplitę statistikoje norint prognozuoti tendenciją ateityje. Modelio prognozės tikslumui bei tinkamumui palyginti bus naudojama santykinė ir absoliuti klaidos.

1.1. Darbinė aplinka

Šiam robotui ir statistiniams modeliams teks naudotis jau sukurtais įrankiais, kurie palengvins darbą. Naudojama Python programavimo kalbą, kadangi ši kalba turi patogų API su dauguma kriptovaliutų rinkų ir patogias bibliotekas autoregresyviems modeliams bei kitiems reikalingiems API.

1.1.1. Python

Python yra aukšto lygio programavimo kalba. Joje galima sutikti ne viena programavimo paradigmą: procedurinę, objektinę ir netgi funkcinę. Ši kalba suteikia patogu abstrakcijos lygį šiai užduočiai - automatizuoti robotą lošėją ir panaudoti AR modelius. Šiais laikais python yra viena populiariausių kalbų pasaulyje, dėl savo paprastumo ir paprasto naudojomo naujam vartotojui, bet tikrai yra daugybę savybių kuriomis prireiktų ne vienus metus suprasti ir prasmingai naudoti programuojant. Dėl šios kalbos privalumų dauguma egzistuojančių API bibliotekų palaiko Python programavimo kalbą.

1.1.1.1. Python-binance API

Šiame darbe naudojama ”python-binance” API. Šis API yra ”Wrapper” oficialiam Binance API - binance-connector-python. Oficialus API yra labai paprastas ir lengvasvoris, todėl didelio patogumo nėra, tik paprasčiausios užklausos, kurias visvien reikėtų apdoroti, labai padeda padaryti šis wrapper API - python-binance.

1.1.2. Binance

Binance yra didžiausia kriptovaliutų birža pagal prekybos apimtį, turinti itin didelį valiutų pasiulą ir patogiai prieinamus duomenis. Ši birža taip pat suteikia galimybė ”lošti” kriptovaliutomis testiniame tinkle. Tinklo paskirtis yra lengvai nuspėjama iš pavadinimo, jame galima atlikti tuos pačius veiksmus kaip realioje rinkoje, tik naudojama netikra valiuta kuri tikros vertės neturi.

1.1.2.1. Binance Spot Test Network

Testinis tinklas yra beveik identiškas tikrajam ”Binance” tinklui, kainos yra vienodos kaip ir pagrindiniame tinkle ar pasaulyje, todėl galima analizuoti seniausius rinkos duomenis ir juos taikyti reikiame modelyje. Testinis tinklas nėra toks populiarus, tad jame yra ženkliai mažiau prekiaujančių žmonių. Testiniame tinkle pąskyra sukuriama naudojant ”Github” prisijungimą. Kiekvienas naujas vartotojas turi pradinį balansą susidendantį iš kriptovaliutų matomų 1 lentelėje.

1 lentelė. Pradinis likutis

Kriptovaliuta	Kiekis
BNB	1,000
BTC	1
BUSD	10,000
ETH	100
LTC	500
TRX	500,000
USDT	10,000
XRP	50,000

Šiuo likučiu galima elgtis kaip norima. Pradinis likutis yra pakankamas atlikti prasmingoms transakcijomis testiniame tinkle. Reikėtų pabrėžti, jog testinis tinklas kas mėnesį laiko yra iš naujo nustatomas ir visas turimas likutis yra konvertuojamas atgal į pradinį likutį, išvalant ankščiau buvusį balansą, tad jeigu nepavyko praturtėti prekiaujant kriptovaliutomis, vėl įgaunama proga pradėti prekyba, na o sėkmingu atveju visas pelnas yra ištrinamas ir reikia vėl pradėti nuo nulio.

2. Laiko eilučių analizė

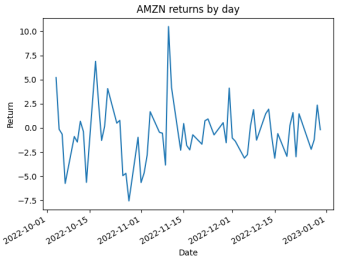
”Visi modeliai yra klaidingi. Kai kurie modeliai yra naudingi” (angl. ”All models are wrong. Some models are useful”) - citata priskiriama George Box, kurios reikėtų nepamiršti analizuojant laiko eilutes. Šis analizės būdas bando pastebėti pasikartojančius modelius bėgant laikui, jog būtų galima atlikti kuo tikslesnes prognozes apie ateitį. Dažnu atveju tai yra vienas ar keli izoliuoti kintamieji, kurie yra stebimi nustatytą laiko tarpą ar surenkant jų istorinius duomenis. Laiko eilučių duomenų rinkimo ir analizavimo pavyzdžiai: sekamas paciento širdies pulsas, prekyboje sandelio prekių kieko priežiūra, konkrečios akcijos kainos prognozavimas akcijų biržoje. Laiko eilučių analizės vienas iš tikslų yra nuspėti kaip kinta reikšmė bėgant laikui, tai ir yra ko prireiks norint prognozuoti kriptovaliutų kainą šiam robotui. Kriptovaliutų kainų duomenys, taip pat yra laiko eilutė, nes kaina yra sekama bėgant laikui, tačiau dažniausiu atveju tai nėra stacionari laiko eilutė. Reikia pabrėžti, kad toks analizės būdas tik bando nuspėti koks yra labiausiai tikėtinas rezultatas remiantis turimais duomeninimis ir naudojamu modeliu.

2.1. Laiko eilutės

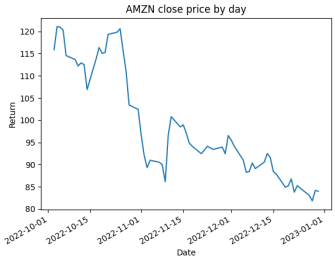
Laiko eilutė yra chronologiškai surikiuotas duomenų rinkinys. Laiko eilučiu analize analitkai gali pastebėti įvairias tendencijas ir taip prognozuoti reikšmes ateityje bei geriau pasiruošti įvykiams. Laiko eilutė gali būti stacionari arba ne stacionari. Šiame darbe norint analizuoti turimus duomenis naudojant autoregresyvius modelius reikia turėti stacionarią laiko eilutę, jog būtų galima pritaikyti autoregresiniui modeliui. Tačiau kriptovaliutų kainą dažnu atveju yra ne stacionari laiko eilutė, todėl prieš dirbant, reikia atitinkamai šiuos duomenis susitvarkyti arba analizuoti kitas eilutes.

2.2. Laiko eilučių stacionarumas

Stacionari laiko eilutė yra laikoma tokia eilutė, kurios statistinės savybės bėgant laikui nekinta [Nas06]. Tai reiškia, jog laiko eilutės vidurkis, variacija ir kovariacija turi pastovias reikšmes. Stacionarios laiko eilutės yra lengviau analizuojamos, todėl autoregresyvieji modeliai yra pritaikyti dirbti su stacionariais duomenimis. Autoregresinių modelių taikymas nestacionariems duomenims gali suteikti nepatikimas prognozes ir prastesnius rezultatus. Prieš dirbant su duomenimis, juos galima pasiversti stacionariais, panaikinant sezoniškumus ar tendencijas imtyje.



(a) Stacionari laiko eilutė



(b) Nestacionari

1 pav. Stacionari ir nestacionari laiko eilutė

Grafai parodantys skirtumą tarp stacionarios ir nestacionarios laiko eilutės. Grafuose naudojama ”Amazon” kompanijos (NASDAQ: AMZN) akcijos duomenys gauti iš ”Yahoo Finance” naudojant Python API[Aro21]. Pirmajame (stacionariame) grafe yra pavaizduota kiekvienos dienos grąža investuojant į AMZN akciją trijų mėnesių laikotarpyje. Stacionarumas matomas ir vizualiai - vidurkis imtyje išlieka panašus, centruojasi ties 0, taip pat variacija ir kovariacija yra panašūs visuose taškuose. Antrame grafe yra pavaizduota tos pačios akcijos uždarymo kaina tokiame pat laikotarpyje. Nestacionarumas matomas ir vizualiai, kadangi yra akivaizdi tendencija žemyn. Šiuos teiginius patvirtina ir ADF testas.

2.3. ADF testas

ADF (Angl. Augmented Dickey-Fuller) testas nustato ar laiko eilutė yra stacionari. Tai yra dažnai naudojamas įrankis analizuojant laiko eilutės ir jų algoritmais, kurie tikisi stacionarių duomenų[Chi18]. Stacionariai laiko eilutėje privaloma atmesti nulinę hipotezę tam tikrame užtikrintumo intervale. Jeigu ADF testo absoliuti statistinė vertė yra daugiau už kritinę vertę, nulinę hipotezę yra atmetama ir ši laiko eilutė yra stacionari. pritaikius ADF testą[SP10] AMZN akcijų duomenims (**1 pav.**) gauti rezultatai:

Time Series	**ADF Statistic**	**p-value**	**1% Critical Value**	**5% Critical Value**	**10% Critical Value**
Daily Returns	-7.087	0.000	-3.542	-2.910	-2.593
Daily Close Price	-1.640	0.461	-3.542	-2.910	-2.593

Pagal šiuos rezultatus

- ADF statistika yra skaičius, kuris lyginamas su kritinėmis vertėmis iš lentelės (skirtingais lygiais), kad būtų nuspręsta, ar atmesti ar priimti nulinę hipotezę apie stacionarumą.
- Pirmu atveju ADF statistika dienos grąžoms yra -7.087, kuri mažesnė, nei kritinė vertė -3.542 1% lygyje. Todėl nulinė hipotezė yra atmesta ir laiko eilutė laikoma stacionari.
- Antru atveju ADF statistika dienos uždarymo kainoms yra -1.640, kuri yra nemažesnė, nei

kritinė vertė -3.542 1% lygyje. Todėl nulio hipotezė nėra atmesta ir laiko eilutė laikoma nestacionari.

3. Autoregresiniai modeliai

Vienas iš būdų analizuoti laiko eilutės yra autoregresija ir ją naudojantis modeliai. Egzistuoja ne vienas autoregresija naudojantis modelis. Populiarusi ir dažniausiai sutinkami modeliai yra ARMA(p, q) ir ARIMA(p, d, q). Šie modeliai taip pat naudojasi anksčiau nepaminėtu MA modeliu kuris yra slankaus vidurkio modelis (angl. MA - moving average). Taip pat yra SARIMA, SARFIMA ir kitų modelių, turinčių savo specifinius panaudjimo atvejus.

3.1. AR modelis

AR (angl. autoregressive) modelis remiasi tik praeities duomenimis, kad nuspėti kintamojo reikšme ateityje, ieškoma ar pastebimas pasikartojantis modelis, kuris padėtų tiksliau nuspėti dydį ateityje[Chi18]. Šis modelis dar dažnai vadinamas ARp modeliu, nes naudojamas kintamasis "p", nusakantis kiek praeiteis reikšmių iš laiko periodo norima naudoti. Laikant, kad kintamasis X yra laiko eilutės kintamasis AR(p) modelio formulė gali atrodyti taip:

$$X_t = \Phi_1 X_{t-1} + \epsilon_t$$

X_t - stacionari laiko eilutė, Φ - AR modelio koeficientas, P - AR modelio laipsnis, ϵ_t - AR modelio paklaida.

3.2. MA modelis

Toliau tyrinėjami autoregresyvieji modeliai susideda iš dar vienos dalies - Slankaus vidurkio - MA (angl. Moving average). Slenkančio vidurkio dabartinė reikšmė tiesiškai priklauso nuo dabartines ir praeitų reikšmių. Žymėjimas MA(q) reiškia q laipsnio slenkamajį vidurkį, kurio formulę atrodo taip:

$$X_t = \epsilon_t - \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

X_t - stacionari laiko eilutė, q - MA modelio laipsnis, ϵ_t - MA modelio paklaida.

3.3. ARMA modelis

ARMA (angl. autorogressive moving average) modelis yra autoregresyviaus ir slankaus vidurkio modelio junginys. Modelis dažnai žymimas kaip ARMA(p, q), kur p yra AR laipsnis, o q yra MA laispnis. Pirmą kartą sujungtas 1938 mokslininko Herman Wold, jis pastebėjo jog ARMA modelis gal apimti dideles stacionarias laiko eilutes, kai yra tinkamai nurodytas p laipsnis ir q laipsnis[MH97]. Taip pat reikia pabrėžti, jog ARMA(0, q) = MA(q) ir ARMA(p, 0) = AR(p) Reiškia, jog Xt eilutė gali būti modeliuojama kaip kombinacija praeities x_t reikšmių ir/arba praeities e_t

klaidų:

$$X_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + ... + \phi_p x_{t-p} + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - ... - \theta_q e_{t-q}$$

3.4. ARIMA modelis

ARIMA (angl. autoregressive integrated moving average) modelis yra paplitęs nuo 1970-ųjų iki šių dienų. Modelio pavadinimo šifravimas pažodžiui: AR - Autoregresinis modelis, I - Integracija (Diferencijavimas), atsižvelgiama į duomenų tendenciją, MA - (Moving average) slankusis vidurkis. AR ir MA yra atskiri modeliai, kurie gali būti naudojami paprastesnei laiko eilučiu analizei. Šio modelio privalumas, jog jis sujungia AR ir MA naudojimą kartu su diferencijavimu, kuris suteikia galimybė giliau analizuoti laiko eilutes. Diferencijavimo dalis turimus duomenis padeda paversti į stacionarius, kurie yra reikalingi korektiškam modelio veikimui. Diferencijavimas vyksta baigtinį kartų skaičių, jog būtų pasiekta beveik-validi stacionari būseną. Kitais žodžiais ARIMA yra ekvivalentus ARMA modelis tiem patiems MA ir AR laipsniams [Hua20]. Taigi ARIMA modelis yra paremtas ARMA modeliu. Pagrindinis skirtumas tarp šių modelių yra tas, jog ARIMA konvertuoja nestacionarius duomenis į stacionarius prieš dirbant su jais. Modelio tipas yra klasifikuojamas kaip ARIMA(p,d,q), kur p - autoregresyvoji dalis, d - integravimo (diferencijavimo) dalis, q - slenkančio vidurkio dalis. Visos ARIMA modelio reikšmės yra neneigiami sveikieji skaičiai [MSG14].

Dažniausiai yra 3 žingsniai naudoti ARIMA modelį:

1. Duomenų surinkimas ir verifikavimas - tikriname ar gauti duomenys yra stacionarūs ir ieškome sezoniškumo.

2. Diferencijavimas - vertimas laiko eilutę į stacionarią.

3. Prognozavimas naudojant ARIMA modelį - dažniausias taikomas ARIMA modelis yra ARIMA(1,1,0)

3.5. SARIMA modelis

SARIMA (angl. seasonal autoregressive integrated moving average) modelis yra ARIMA modelio išplėtimas turintis vieną papildomą savybę - sezoniškumą[CE20]. Pastebėjus, jog periodiškai kartojasi rezultatai laiko eilutėse, galima taikyti šį modelį. Geras ir paprastas pavyzdys, prekyba šventiniu laikotarpiu - kalėdos. Prekybos centruose, tuo metu padidėja prekybą, tačiau kitą mėnesį ženkliai sumažėja. Spartus sumažėjimas nereiškia, jog prekybai yra didėlės problemos ir reikia pokyčių. Tai dažniausiai yra žmonių poilsis nuo pirkinių po šventinio laikotarpio. šis modelis pastebi panašius sezoniškai atsitinkančius įvykius, juos atpažįsta ir pritaiko naudojant ankščiau minėtą ARIMA modelį.

SARIMA galima išreikšti kaip ARIMA(p,d,q)(P,D,Q)[S], kur p,d,q standartiniai ARIMA modelio parametrai, o P - sezoniškumo autoregresivumo laipsnis, D - sezoniškumo diferencijavimo laipsnis, Q - sezoniškumo slankiojo vidurkio laipsnis ir S - sezoniškumo ciklio ilgis.

4. Prognozių metodika

Skyriuje aprašomi žingsniai, kurių buvo imtasi atliekant tyrimus ir analizę kuriant automatizuotą kriptovaliutų prekybos robotą. Tyrimo metu pagrindinis dėmesys buvo skiriamas tinkamų kriptovaliutų kainų analizės ir prognozavimo įrankiams ir metodams nustatyti bei skirtingų modelių veikimui įvertinti. Duomenys buvo renkami iš Binance testavimo tinklo naudojant python-binance API, o duomenims pritaikyti statistinės ir laiko eilučių analizės metodai. Kainų prognozavimui buvo naudojami autoregresyvūs modeliai, tokie kaip ARIMA ir SARIMA. Šio tyrimo išvados yra vienas iš pagrindų kuriant prekybos robotą.

4.1. Modeliai

Prognozavimui bus naudojami ARIMA ir SARIMA modelis. Kadangi kriptovaliutų kainų duomenys nėra stacionarūs, todėl naudojamas ARIMA modelis, kuris gali taikyti diferencijavimą. ARIMA naudoja anksčiau minėtus AR ir MA modelius, bei pritaiko diferencijavimą, taip galėdamas dirbti ir su nestacionariais duomenimis. Jeigu laiko eilutė yra nestacionari ir bandoma naudoti AR, MA ir ARMA modelius, rezultatai galimai bus klaidingi arba netikslūs. Taip pat naudojamas SARIMA modelis, nes jis yra ARIMA modelio pratęsimas tiriantis sezoniškumą laiko eilutėje, todėl dar bus patikrinama ar duomenyse egzistuoja sezoniškumas galintis padėti prognozuoti.

4.2. Duomenys

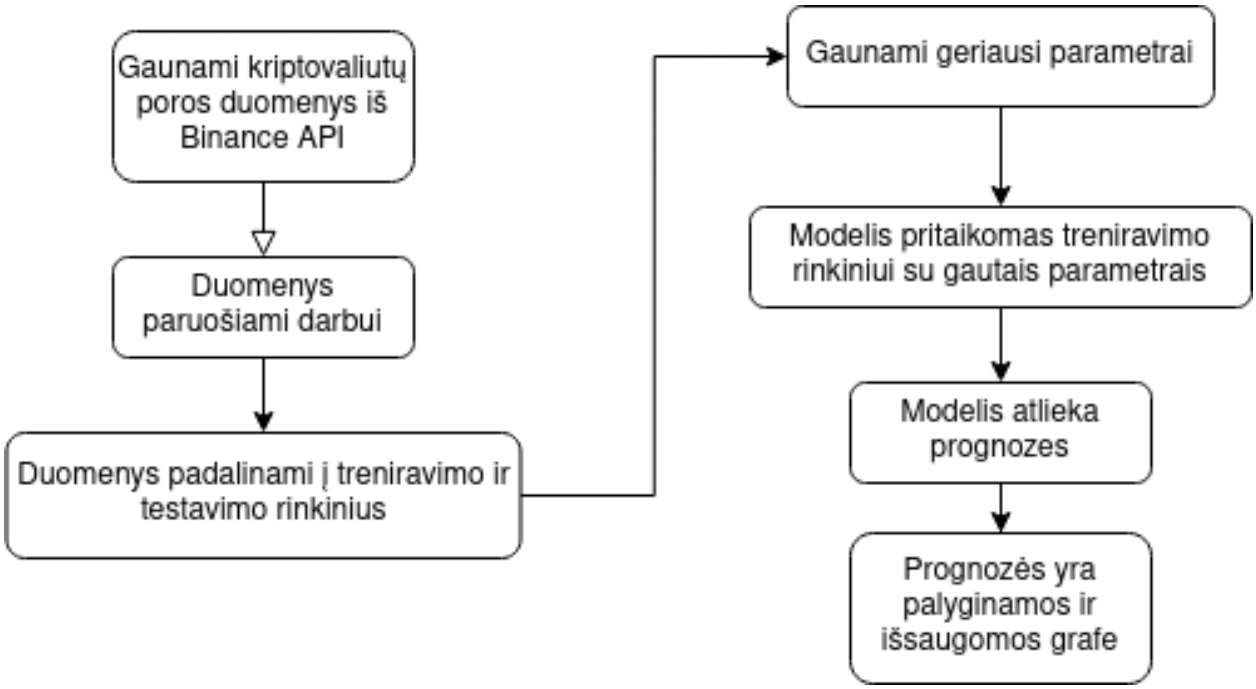
Istoriniai duomenys ”Candlesticks” surinkti iš Binance testavimo tinklo naudojant python-binance API. Kadangi testinis tinklas kiekvieną mėnesį automatiškai išvalo visus duomenis, imami 2023m. sausio 4d. duomenys savaitei į priekį, todėl verta paminėti, kad šie duomenys galioja tik tam tikrą laikotarpį. Kadangi tinklas yra testinis, jame prekiaujama nevisomis galimomis kriptovaliutų poromis. Iš viso yra 20 skirtingų porų tinkle. Pasirinkta tirti šias poras: i) BTC/BUSD, ii) LTC/BUSD, iii) XRP/BUSD. Gaunamuose duomenyse yra daugiau informacijos bet bus naudojama tik datos ir ‘close’ stulpelis, parodantis konkrečios kriptovaliutos uždarymo kainą tam tikru laiku. Tai yra paskutinė kaina, už kurią buvo prekiaujama tam tikru laikotarpiu, pavyzdžiui, valandą ar dieną. Uždarymo kaina yra svarbi, nes ji atspindi galutinę kainą, už kurią pirkėjai ir pardavėjai sutarė prekiauti kriptovaliuta. Duomenys bus naudojami tokie kokie yra ir iškarto perduodami į modelius. Modeliui apmokyti naudojamas 90 procentų dumomenų rinkinio dydis. Kitus 10% bandoma prognozuoti po vieną reikšmę į priekį. Modeliui atlikus prognozė, jis yra iš naujo apmokomas pridėjus tikrąją sekančią vertę prie mokymo rinkinio ir taip vėl yra prognozuojama kitą reikšmė, jau su naujai apmokytu modeliu.

4.3. Modelių implementavimas

Prognozuojant naudojamos ARIMA ir SARIMA pagalbinės bibliotekos[SP10] kuri aprašo šiuos modelius, bei euristins auto_arima algoritmas[Smi+–] padedantis nustatyti geriausius ARIMA ir SARIMA parametrus konkrečiam duomenų rinkiniui. SARIMA modeliui sezoniškumo intervalas naudojamas 4 - bandoma pamatyti ar atsiranda sezoniškumas kiekvieną valandą, nes duomenų intervalas yra 15min. Modelių vertinimui bus naudojama MSE ir MAPE paklaidos.

4.4. Prognozavimas

Kriptovaliutų kainų prognozavimo metodas naudoja anksčiau minėtus modelius ir Binance testinio tinklo duomenis. Žemiau matoma struktūrinę schemą, parodanti bendrą procesą, kurį sudaro keli pagrindiniai žingsniai. Pirmas žingsnis yra duomenų gavimas is binance API, bet jų gaunama daugiau negu reikia, todėl atsirenkami tik reikalingi duomenys, šis žingsnis yra svarbus norint įsitikinti,kad duomenys yra paruošti analizei. Po to pasirenkame modelį (ARIMA/SARIMA). Kitame žingsnyje yra gaunami geriausi parametrai konkrečiam duomenų rinkiniui ir modeliui, kad būtų optimalus veikimas. Tada taikome modelį treniruočių duomenims, kad apmokyti modelį tolimesniam prognozavimui. Po to modelis prognozuoja reikšmes ir jos yra saugomos, kad būtų galima palyginti su tikromis reikšmemis. Paskutinis žingsnis - palyginti gautus rezultatus ir išsaugoti duomenis grafe. Pateikta struktūrinė schema padės geriau suprasti visą procesą.



2 pav. Struktūrinė schema

4.5. Vertinimas

Vidutinė kvadratinė paklaida (MSE) ir vidutinė absoliučioji procentinė paklaida (MAPE) yra dvi dažniausios metrikos, naudojamos prognozavimo modelių našumui įvertinti. MSE yra prognozuotų ir faktinių verčių vidutinio skirtumo kvadrato matas, o MAPE yra vidutinio procentinio skirtumo tarp numatytų ir faktinių verčių matas. Abu šie rodikliai gali būti naudingi lyginant ARIMA ir SARIMA prognozavimo kriptovaliutoms rezultatus. MSE yra gera metrika norint išmatuoti, kiek prognozės nukrypsta nuo faktinių verčių, mažesnė MSE reikšmė rodo tikslesnį modelį. MAPE yra gera metrika prognozavimo paklaidoms procentais matuoti, mažesnė MAPE reikšmė rodo tikslesnį modelį. Abi šios metrikos gali būti naudojamos vertinant modelių veikimą ir lyginant ARIMA ir SARIMA prognozavimo kriptovaliutoms rezultatus.

5. Prognozių realizacija

Šio darbo realizavimo ir rezultatų skyriuje pristatomas metodikos įgyvendinimas ir gauti rezultatai. Skyriuje pateikiamos duomenų ir prognozavimo rezultatų vizualizacijos, taip pat kodo fragmentai, demonstruojantys naudojamų modelių įgyvendinimą. Vizualizacijos suteikia aiškų vaizdą apie modelių veikimą ir prognozių tikslumą. Kodo fragmentai suteikia išsamų supratimą apie modelių įgyvendinimą. Šioje dalyje taip pat palyginami rezultatai, gauti naudojant ARIMA ir SARIMA modelius, leidžiantys įvertinti modelių veikimą.

5.1. Modelių realizacija

Iš pradžių prisijungiama prie testinio tinklo naudojant API ir naudotojo raktus

```
with open(CONFIG_FILE_NAME, 'r') as config_file:
data = json.load(config_file)
validate(instance=data, schema=config_schema)
client = Client(data['api_key'], data['api_secret'], testnet=True)
```

Tuomet gaunami istoriniai duomenys iš API. Kaip minėta, jog Sausio 4d. įvykio atnaujinimas ir eksperimentiniais tikslais imama duomenys savaitė nuo pradžios į priekį. tuomet kviečiami funkcija, jog paverstų gautus duomenis į reikalinga struktūrą.

```
klines = client.get_historical_klines(symbol, Client.KLINE_INTERVAL_15MINUTE
                                      , end_str='11 Jan, 2023')
df = binance_klines_to_df(klines)
model_predict_arima(symbol, df)
model_predict_arima(symbol, df, seasonal=True)

def binance_klines_to_df(klines):
klines = np.array(klines)
df = pd.DataFrame(klines.reshape(-1, 12), dtype=float, columns=('Open Time',
                                      'Open', 'High', 'Low', 'Close', '
                                      Volume', 'Close time', 'Quote asset
                                      volume', 'Number of trades', 'Taker
                                      buy base asset volume', 'Taker buy
                                      quote asset volume', 'Ignore'))
df['Open Time'] = pd.to_datetime(df['Open Time'], unit='ms')
return df
```

Turint duomenis kviečiama model_predict_arima funckija, kuri pagal seasonal parametrą atlieka ARIMA arba SARIMA modelį.

```
df = df[['Open Time', 'Close']]
df.set_index('Open Time', inplace=True)

to_row = int(len(df) * 0.9)
training_data = list(df[0:to_row]['Close'])
```

```
if seasonal:
order, seasonal_order = best_params_arima_seasonal(training_data)
else:
order = best_params_arima(training_data)
seasonal_order = None

testing_data = list(df[to_row:]['Close'])
model_predictions = []
for i in range(len(testing_data)):
model = ARIMA(training_data, order=order, seasonal_order=seasonal_order)
model_fit = model.fit()
output = model_fit.forecast(steps=1)
yhat = output[0]
model_predictions.append(yhat)
actual_test_value = testing_data[i]
training_data.append(actual_test_value)
```

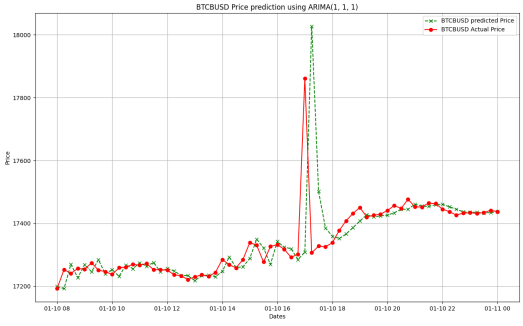
Pagrindinė kodo dalis atliekanti skaičiavimus Pasiemama is duomenų tik data ir uždarymo kainą, tuomet duomenys padalijami 90% apmokymui ir kita dalis prognozavimui. Tuomet modeliui gaunami optimalūs parametrai. Po šito inicijuojamas tuščias sąrašas, vadinamas model_predictions. Kiekvienai iteracijai atliekami šie veiksmus:

- sukuria ARIMA modelį turimais duomenimis ir parametrais ir jis yra pritaikomas naudojant fit().
- gaunama prognozuojama reikšmė yhat
- prie prognozių sąrašo pridedama reikšmė
- į turimus duomenis pridedama tikroji reikšmė ir toliau ciklišKai apmokamas modelis

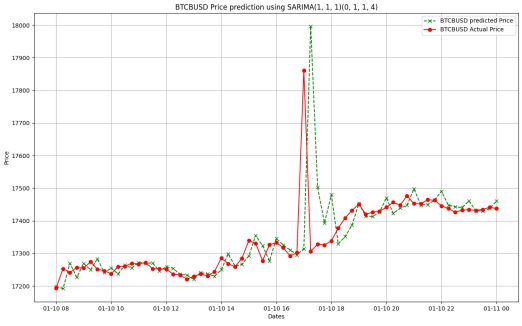
5.2. Prognozės

Šie rezultatai rodo ARIMA ir SARIMA modelių našumą prognozuojant trijų skirtingų simbolių - BTCBUSD, LTCBUSD ir XRPBUSD - kriptovaliutų kainas. Modelių veikimui įvertinti naudojami MSE (Mean Squared Error) ir MAPE (Mean Absolute Percentage Error). MSE matuoja vidutinį skirtumą tarp numatytų ir faktinių verčių kvadratu, o MAPE - vidutinį procentinį skirtumą tarp numatomų ir faktinių verčių. Abu šie rodikliai yra svarbūs vertinant modelių tikslumą.

Taikymo rezultatai trims skirtingoms kriptovaliutų poroms: BTCBUSD, LTCBUSD ir XRPBUSD. Grafikuose matosi prognozuota ir tikra kaina laikotarpiu. BTCBUSD



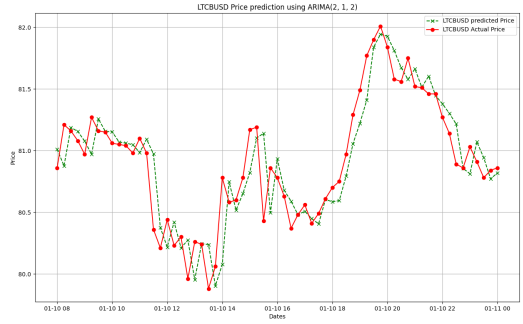
(a) ARIMA modelis



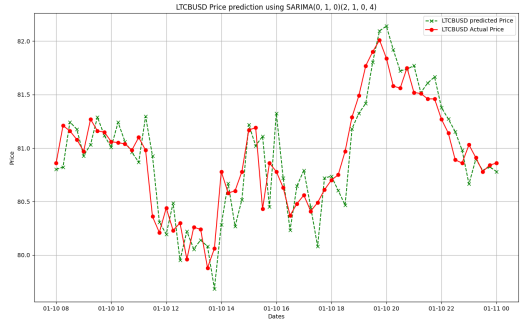
(b) SARIMA modelis

3 pav. BTCUSD prognozavimas

LTCBUSD



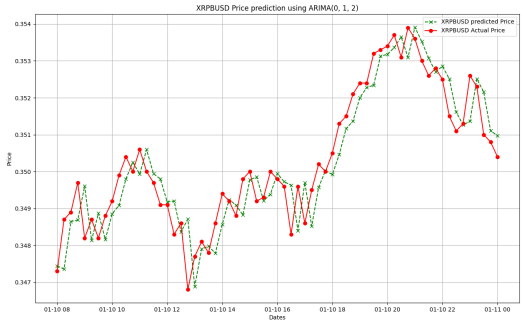
(a) ARIMA modelis



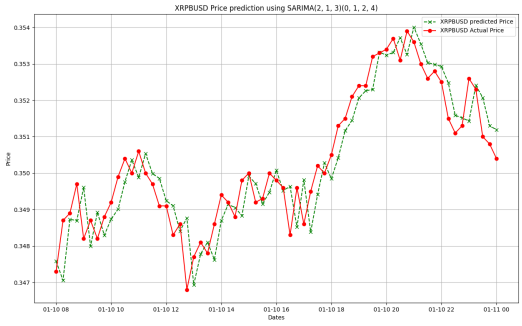
(b) SARIMA modelis

4 pav. LTCBUSD prognozavimas

XRPBUSD



(a) ARIMA modelis



(b) SARIMA modelis

5 pav. XRPBUSD prognozavimas

Symbol	Model	MSE	MAPE
BTCBUSD	ARIMA	13702.25	0.22%
BTCBUSD	SARIMA	13349.98	0.24%
LTCBUSD	ARIMA	0.05	0.2%
LTCBUSD	SARIMA	0.06	0.24%
XRPBUSD	ARIMA	4.92	0.16%
XRPBUSD	SARIMA	5.24	0.17%

2 lentelė. Modelių taikymo rezultatai

Iš lentelės matome, kad BTCBUSD poros MSE yra šiek tiek mažesnė SARIMA modeliui nei ARIMA modeliui, tai rodo, kad SARIMA modelis gali šiek tiek geriau atitikti šį duomenų rinkinį, bet MAPE yra šiek tiek mažesnė ARIMA modeliui.

LTCBUSD ir XRPBUSD poroje abiejų modelių MSE ir MAPE yra panašiai žemi, o tai rodo, kad abiejų modelių šių duomenų rinkinių našumas yra panašus. Visų porų MSE ir MAPE yra palyginti žemos, o tai rodo, kad modeliai gerai tinka duomenims, o prognozės yra gana tikslios.

Verta paminėti, kad šie rezultatai yra naudojami konkrečiams duomenų rinkiniui, taigi modeliai gali duoti skirtingus rezultatus kitiems rinkiniams.

6. Prekybos robotas

Skyriuje siekiama nuodugniai pažvelgti į automatizuoto roboto metodiką bei roboto kūrimo ir taikytų strategijų rezultatus. Pagrindinis dėmesys bus skiriamas paaiškinimui, kaip kuriamas robotas, kaip atliekami sandoriai ir kokie rezultatai gaunami naudojant skirtingas strategijas.

6.1. Metodika

Robotas parašytas ”Python” programavimo kalbą naudojant ”python-binance” API bendrauti su Binance testiniu tinklu. Yra keli pagrindiniai žingsniai roboto veikime.

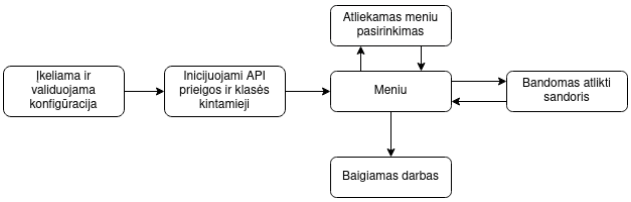
1. Įkeliama konfigūracija ir simbolių duomenys iš failų, duomenys patikrinami pagal validacijos schemas, inicijuojami API prieigos ir klasės kintamieji.
2. Gaunami istoriniai simbolių ”klines” duomenys, kurie yra saugojami naudojimui.
3. Siūlomas vartotojui sąveikos su prekybos robotu parinkčių meniu, pvz., likučių, pavedimų, pozicijų, simbolių informacijos spausdinimas ir pavedimų atšaukimas.

4a. Vykdomas meniu pasirinkimas

4b. Skaičiuojamos taikomos strategijos prognozės ir robotas bando atlikti sandorius, jei sąlygos ir tenkinamos.

4c. Išsaugo kriptovaliutų porų pozicijas, kai vartotojas išeina iš programos.

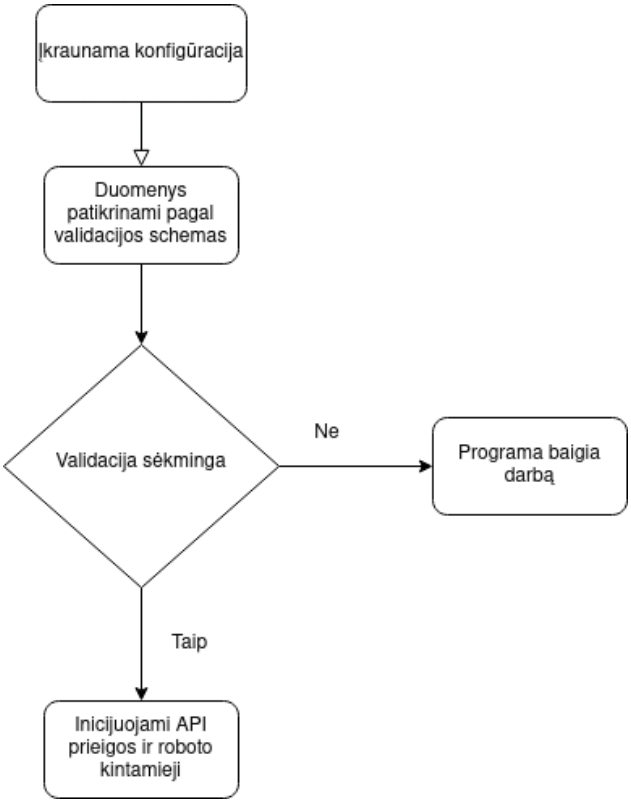
Pagrindinės būsena yra 3, kai robotas laukia įvesties iš vartotojo arba laukia kol pasibaigs nustatytas prekiavimo laikas, tuomet pereis į vieną iš 4 būsenų, arba bus vykdoma, meniu pasirinkimas, pvz. spausdinti dabartinį balansą arba bus bandoma atlikti prekybą arba tiesiog baigti darbą. Toliau apie visą eigą plačiau poskyriuose.



6 pav. Pagrindinės būsenos

6.1.1. Validacija

Darbo pradžia - vykdoma validacija, kuri patikrina ar konfigūracija yra teisinga ir galima toliau prekiauti robotui - t.y. porų duomenis yra tvarkingi, pasirinkta egzistuojanti strategija ir pateikti jai reikalingi kintamieji, pasirinktas prekiavimo intervalas bei kiti kintamieji.



7 pav. Validacijos struktūrinė schema

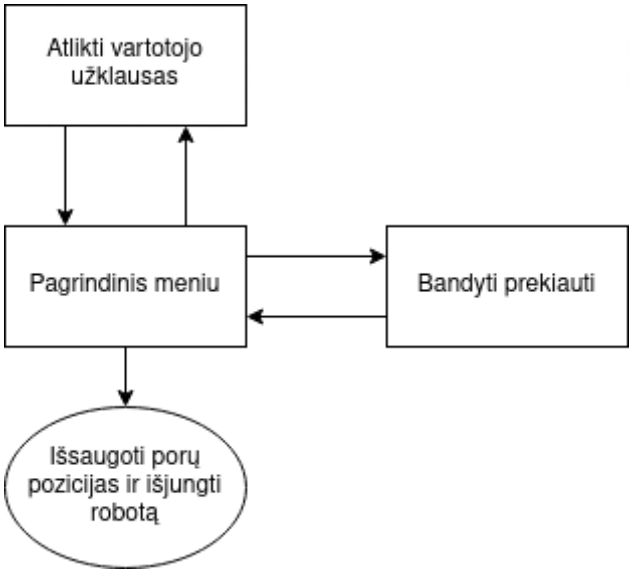
6.1.2. Meniu

Robotui sėkmingai įvykdžius validaciją sutinkamas meniu, kuriame galima atlikti vartotojui reikalingas operacijas, tokias kaip pasižiūrėti valiutų balansą, pažiūrėti įvykdytus sandorius ar tiesiog baigti darbą.

Galimi pasirinkimai pradėjus roboto darbą.

1. Spausdinti paskyros balansą.
2. Spausdinti simbolių poros sandorius.
3. Spausdinti dabartines kriptovaliutų porų pozicijas.
4. Spausdinti sandorį pagal ID.
5. Atšaukti sandorį rankiniu būdu.
6. Išsaugoti simbolio sandorių grafiką.
9. Spausdinti meniu.
0. Baigti darbą.
-1. Bandyti atlikti sandorius (Numatyta meniu parinktis pasibaigus laukimo laikui).

Robotui prireikia minimalios vartotojo sąsajos, kad būtų galima sekti progresą ir atliktus darbus. Galima pasižiūrėti paskyros balansą ar atšaukti rankiniu būdu sandorį. Šios komandos praverčia atliekant sandorius, kurie gali būti atliekami ne iš karto. Taip pat patogu gauti duomenys iš API, kad sužinoti kokie sandoriai kokiu laiku būdu buvo vykdomi, taip nereikia saugoti žurnale šios informacijos. Vienintelis minusas, jog nebus aišku, kokia strategija buvo taikyta, kadangi tai yra šios programos implementacijos reikalas. Šis meniu yra visada matomas vartotojo. Jeigu nepasirinkta jokia operacija, po nustayto laiko, robotas automatiškai pasirenka paskutinį variantą ir bando atlikti sandorius.



8 pav. Meniu schema

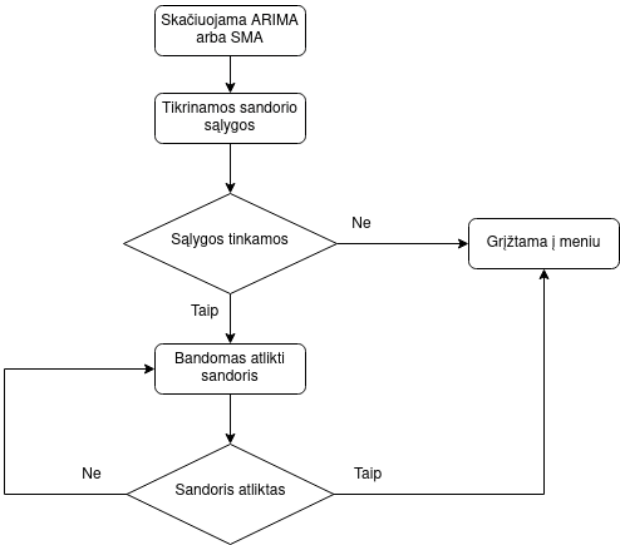
6.1.3. Prekyba

Robotui sėkmingai pradėjus darbą ir praėjus laiko tarpui po kurio robotas pradeda bandyti prekiauti, bandoma atlikti pagrindinį žingsnį - prekybą. Pagrindiniai žingsniai yra tokie:

- Skaičiuojama ARIMA prognozė arba skaičiuojamos trumpojo ir ilgojo vidurkio vertes, pagal pasirinktą strategiją.
- Tikrinamos sąlygos pagal apskaičiuotas vertes.
- Jeigu sąlygos tinkamos bandoma atlikti sandorį, kitu atveju grįžtama į meniu ir vėl laukiama kada reikės bandyti atlikti sandorį su naujesniais duomenimis.
- Jeigu reikia atlikti sandorį, jį bandoma atlikti. Bet sandoris nevisada gali pavykti, ypač jei norima pirkti nustatytą kainą, sandoris gali būti neįvykdomas, todėl dar bandoma jį kartoti.
- Jei sandoris pavyko, grįžtama į meniu būseną kur vėl robotas yra laukimo būsenoje.

Taigi procesas gana paprastas - atnaujinami duomenis ir bandomas atlikti sandoris, jei sąlygos yra tenkinamos. Taip pat jeigu sandoris nepavyksta, sandorį dar bandoma po nustatyto laukimo

laiko pakartoti kelis kartus pagal nustatymus. Tai praverčia jeigu norima prekiauti nustatytą kainą, o ne tokia kokią Siūlo rinka, bet kitais atvejais sandoris taip ir gali nepavykti, nes niekas nepriima tokios kainos ir sandoris bus neįvykdytas pagal turimas prognozes.



9 pav. Prekybos struktūrinė schema

6.2. Strategijos

6.2.1. Tendencijos sekimas

Šiuo atveju vykdoma strategija yra standartinė pirkimo ir laikymo strategija. Robotas naudoja ARIMA, kad sudarytų pasirinktų kriptovaliutų porų kainos prognozes. Kai prognozė numato kylančią kainą, robotas įvykdys pirkimo pavedimą, o prognozei numatant mažėjančią kainą robotas vykdys pardavimo pavedimą. Šio tipo strategija paremta idėja, kad galima pasipelnyti iš rinkos tendencijų perkant pigiai ir parduodant brangiai. Robotas naudoja ARIMA, kad analizuotų istorinius kainų duomenis ir prognozuotų būsimus kainų pokyčius, kuriuos jis naudoja priimdamas pirkimo ir pardavimo sprendimus. Vadovaudamasis šia strategija, robotas siekia pasinaudoti rinkos tendencijomis.

- ARIMA prognozuoja kriptovaliutos kainą.
- Remiantis prognoze, jei modelis numato didėjimo tendenciją, bandoma nusipirkti kriptovaliutą, o jei prognozuoja mažėjimo tendenciją, bandoma kriptovaliutą parduoti.
- Nuolat stebima kriptovaliutos kainą ir atitinkamai atnaujinamas ARIMA modelis, kad būtų atnaujintos prognozes.

6.2.2. Grįžimas prie vidurkio

Grįžimas prie vidurkio yra prekybos strategija, pagrįsta idėja, kad kainos laikui bėgant grįžta į savo vidurkį. Tai apima dabartinės kainos palyginimą su istorine vidutine kaina. Sandoris yra

sudaromas pagal tai, ar dabartinė kaina yra didesnė ar mažesnė už vidutinę. Pagal šią strategiją, jei dabartinė kaina yra didesnė už istorinį vidurkį, prekiautojas parduotų turtą, o jei ji yra mažesnė, prekiautojas pirktų. Tikslas - pasinaudoti rinkos tendencija grįžti prie vidurkio.

- Apskaičiuojama kriptovaliutos pastarųjų trumpojo ir ilgojo intervalų paprastą slenkantį vidurkį (SMA).
- Jei trumpasis SMA yra didesnis nei ilgasis SMA - bandoma parduoti kriptovaliutą.
- Jei trumpasis SMA yra mažesnis nei ilgasis SMA - bandoma pirkti kriptovaliutą.
- Nuolat stebima kriptovaliutos kaina ir atitinkamai atnaujinami SMA.

Strategija grindžiama idėja, kad kai kriptovaliutos kaina nukrypsta nuo vidutinės vertės, ji ilgainiui grįš į vidutinę. Šiuo atveju vidurkį atstoja kainos paprastasis slankusis vidurkis (SMA), o nuokrypis nuo vidurkio nustatomas lyginant trumpalaikį SMA ir ilgalaikį SMA. Jei trumpalaikis SMA yra didesnis nei ilgalaikis SMA, tada daroma prielaida, kad kaina nukrypsta nuo vidurkio ir galiausiai grįš, todėl bandoma atlikti pardavimą. Ir atvirkščiai, jei trumpalaikis SMA yra mažesnis už ilgalaikį SMA, tada pateikiamas pirkimo pavedimas, numatant, kad kaina grįš į vidutinę. Šioje programoje, galima pasirinkti kiek pastarųjų verčių kainų vidurkį skaičiuoti trumpajam ir ilgajam SMA.

6.3. Realizacija

Toliau robotui sandoriams atlikti pasirinktos šios kriptovaliutų poros

- BTC/BUSD
- ETH/BUSD
- LTC/BUSD

Šios poros yra populiarios ir dažnai prekiaujamos tinkle, todėl jomis bus paprasčiau naudotis. Kadangi tinklas yra testinis, didžiausios prekybos apimties sulaukia tik populiariausios kriptovaliutos.

6.3.1. Konfigūracija

Naudojamų porų konfigūracija prekiauti kriptovaliutų poromis. Pasirenkama kiekis po kiek prekiauti, pirkimo ar pardavimo pozicija, koks yra užsakymo tipas (”LIMIT” arba ”MARKET”) ir jeigu tipas yra ”LIMIT” pasirinkti užsakymo reikšmę: ”FOK”, ”IOC” arba ”GTC”. Žemiau pateikta konfigūracija naudojama prekiauti abiejomis strategijomis.

```
{
"BTCBUSD": {
```

```
    "trade_quantity": 0.01, # Kiekis kuriuo prekiaujama
    "position": "BUY", # Dabartinė pozicija
    "order_type": "LIMIT", # Užsakymo tipas (LIMIT arba MARKET)
    "time_in_force": "FOK" # LIMIT nustatymas (FOK, GTC arba IOC)
  },
  "ETHBUSD": {
    "trade_quantity": 0.1,
    "position": "BUY",
    "order_type": "LIMIT",
    "time_in_force": "FOK"
  },
  "LTCBUSD": {
    "trade_quantity": 1,
    "position": "BUY",
    "order_type": "LIMIT",
    "time_in_force": "FOK"
  }
```

Taip pat yra dar vienas konfigūracijos failas kuriame įrašoma API prieigos raktai, strategija, prekybos laikas ir kokio intervalo duomenis naudoti. Žemiau pateika strategija buvo naudojama prekiauti ”Grįžimo prie vidurkio” strategijai. ”Tendencijos sekimo” strategijai viskas tas pats, tik nebelieka ”long_term” ir ”short_term” kintamųjų ir strategija pakeičiama į ”TENDENCY_ARIMA”

```
{
  "api_key": "key", # API prieigos raktas
  "api_secret": "s3cr3t", # API prieigos slaptas raktas
  "timeout": 900, # Kas kiek laiko bandyti prekiauti
  "interval": "15m", # Kokio intervalo duomenis naudoti
  "strategy": "MEAN_SMA", # Strategija (MEAN_SMA arba TENDENCY_ARIMA)
  "long_term": 15, # MEAN_SMA strategijos ilgojo slankaus vidurkio parametras
  "short_term": 5 # MEAN_SMA strategijos trumpojo slankaus vidurkio parametras
}
```

6.3.2. Prekybos realizacija

Robotas skirtas analizuoti rinkos tendencijas ir priimti pagrįstus sprendimus, kada pirkti ir parduoti kriptovaliutas. Kad tai pasiektų, robotas naudoja ARIMA analizę ir paprastą slenkantį vidurkį (SMA). Šios technikos leidžia robotui nuolat stebėti rinkos sąlygas ir sudaryti sandorius remiantis realaus laiko duomenimis. Ištirsime, kaip robotas integruoja šias strategijas ir procesą, kuriuo jis pateikia pavedimus ir vykdo sandorius.

Programos pradžioje, praėjus validacijai ir inicializacijai, kodas patikrina, kokia prekybos strategija pasirinkta ir pagal tai atlieka reikiamus skaičiavimus. Jei pasirenkama MEAN_STRATEGY, nuskaitoma istorines kriptovaliutos kainas (tiek pastarųjų verčių koks ilgojo slankaus vidurkio parametras) ir apskaičiuoja trumpąjį ir ilgąjį paprastąjį slankųjį vidurkį (SMA). Jei pasirinkta TENDNCY_ARIMA, ji nuskaito pastarųjų 1000 intervalo "candlesticks" eilučių duomenis ir apskaičiuoja ARIMA prognozę.

```
if self._strategy == MEAN_STRATEGY:
    self._get_historic_prices(limit=self._long_term)
    self._calculate_sma()
elif self._strategy == TENDENCY_STRATEGY:
    self._get_klines_as_df(limit=1000)
    self._calculate_arima()
```

Toliau, kai praeina nustatytas "timeout" laikas, programa kviečia _try_trade funkciją , kuri vykdo atitinkamą prekybos strategiją priklausomai nuo strategijos. Abi strategijos atnaujina savo duomenis, išmesdama seniausia verte ir pridedama naujausia kainos vertę, tuomet pagal strategija atliekami SMA arba ARIMA skaičiavimai ir bandoma atitinkamai atlikti užsakymą.

```
def _try_trade(self) -> None:
  klines = 1
  if self._strategy == MEAN_STRATEGY:
    self._get_historic_prices(klines)
    self._calculate_sma()
    self._trade_sma()
  elif self._strategy == TENDENCY_STRATEGY:
    self._get_klines_as_df(klines)
    self._calculate_arima()
    self._trade_arima()
```

Šis kodo blokas gauna vidutinę konkretaus simbolio kainą rinkoje jeigu yra nustatytas "LIMIT" užsakymo tipas ir atlieka sandorius pagal strategiją. Patikrinama, ar trumpalaikis SMA didesnis už ilgalaikį SMA ir ar esama pozicija yra pirki, o jei tenkinamos abi sąlygos, pateikia užsakymą parduoti nurodytą simbolio kiekį už apskaičiuotą vidutinę kainą. Jei trumpalaikis SMA yra mažesnis už ilgalaikį SMA, o dabartinė pozicija yra parduoti, pateikiamas užsakymas pirkti nurodytą simbolio kiekį. Atlikus sandorį, padėtis atnaujinama, kad atspindėtų naują būseną. Analogiškai vyksta ir tendencijos strategijoje - tikrinama ar ARIMA prognozuojama kainą yra didesnę ar mažesnę už dabartinę vidutinę kainą ir atitinkamai atliekamas užsakymas.

```
    price = None
    if config['order_type'] == Client.ORDER_TYPE_LIMIT:
        price = self._get_symbol_avg_price(symbol)
    if short_sma > long_sma and position == 'BUY':
        if self._make_order(symbol, position, config['trade_quantity'], price)
                                                :
            config['position'] = 'SELL'
```

```
    elif short_sma < long_sma and position == 'SELL':
        if self._make_order(symbol, position, config['trade_quantity'], price)
                                                                        :
            config['position'] = 'BUY'
```

Funkcija _make_order naudojama norint atlikti konkretaus simbolio užsakymą su nurodyta puse (pirkti arba parduoti), kiekiu ir kaina. Naudojama „create_order“ metodą iš „Binance“ API ir nustato simbolio parametrus remiantis konfigūracija.

Funkcija turi pakartotinio bandymo mechanizmą, kad kelis kartus būtų bandoma pateikti užsakymą, jei atsiranda klaidų arba užsakymas būtų atmestas. Pakartotinių bandymų kintamasis nustatomas į 0 ir didėja 1 po kiekvieno nesėkmingo bandymo. Jei pakartojimų skaičius viršija ORDER_MAX_RETRIES (prekybos atveju šis kintamasis yra - 3), funkcija grąžins False, nurodydama, kad užsakymas nebuvo baigtas.

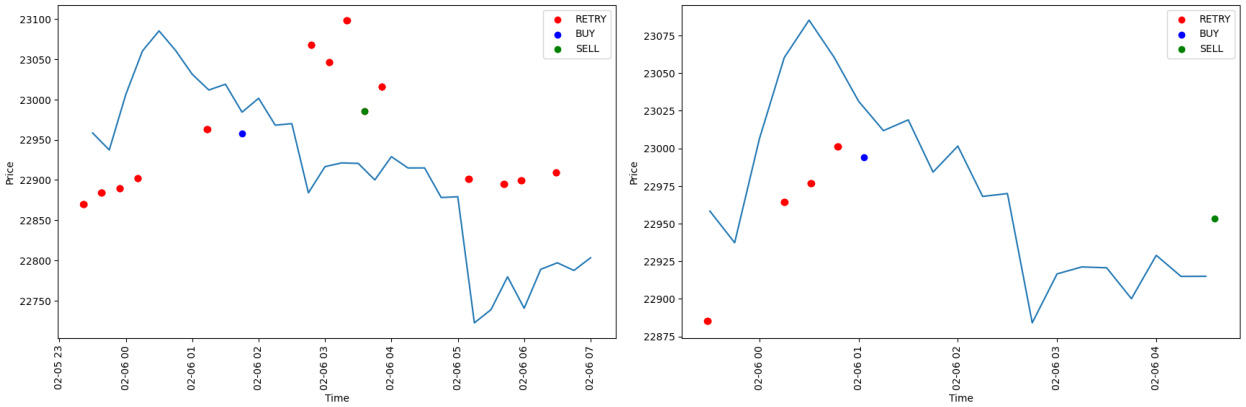
Po kiekvieno bandymo funkcija laukia tam tikrą laiką, apibrėžtą ORDER_RETRY_WAIT_TIME (prekybos atveju šis kintamasis yra - 10), prieš bandydama dar kartą. Jei užsakymas bus užpildytas, atsakymo būsena bus „FILLED“, o funkcija grąžins True, nurodydama, kad užsakymas įvykdytas sėkmingai.

```
def _make_order(self, symbol: str, side: str, qty: float, price: float = None)
                                        -> bool:
        # Retry is useful if order type is LIMIT
        retries = 0
        order_completed = False
        while not order_completed and retries < ORDER_MAX_RETRIES:
            try:
                response = self._client.create_order(
                    symbol=symbol,
                    side=side,
                    type=self._pairs_config[symbol]['order_type'],
                    quantity=qty,
                    price=price,
                    timeInForce=self._pairs_config[symbol]['time_in_force'],
                )
                print(f"{response['side']} {response['type']} {response['
                                                    symbol']} {response['
                                                    status']}")
                if response['status'] == Client.ORDER_STATUS_FILLED:
                    order_completed = True
                    break
            except exceptions.BinanceOrderException as e:
                print(e.message)
            retries += 1
            time.sleep(ORDER_RETRY_WAIT_TIME)
        return order_completed
```

6.3.3. Prekybos rezultatai

Prekybos rezultatų poskyris yra automatizuoto kriptovaliutų prekybos roboto veikimo įvertinimas. Šioje skiltyje bus vaizdžiai pavaizduoti prekybos strategijos, įgyvendintos naudojant grafikus ir diagramas, rezultatai. Roboto krepšelio pokytis bus lyginamas su pradiniu krepšeliu prieš pradedant prekybą. Šios dalies tikslas – įvertinti bendrą roboto pelningumą ir pateikti jo prekybos rezultatų analizę.

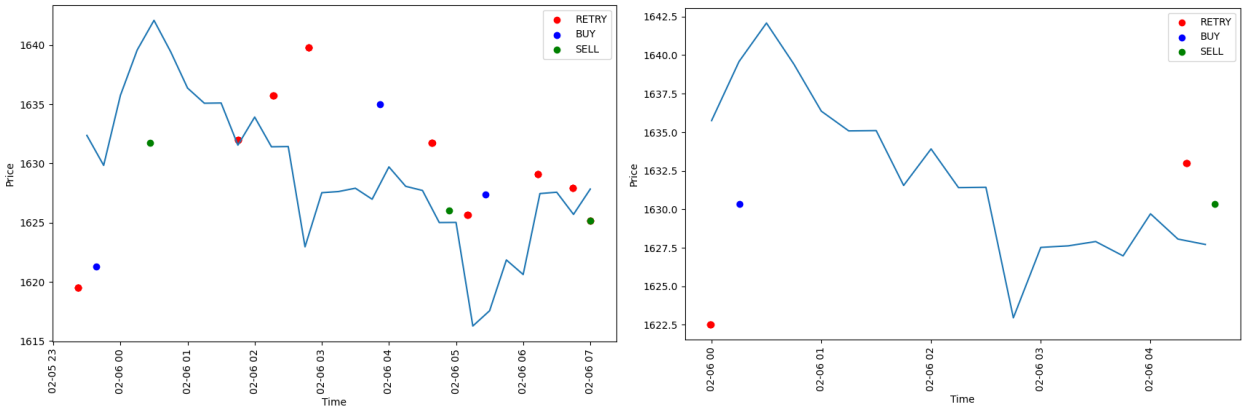
Prekyba buvo atlikta naudojant abi aprašytas strategijas, ir palikus robotą dirbti nakties metu, bandant atlikti prekyba kas 15 minučių.



(a) Tendencijos sekimas

(b) Grįžimas prie vidurkio

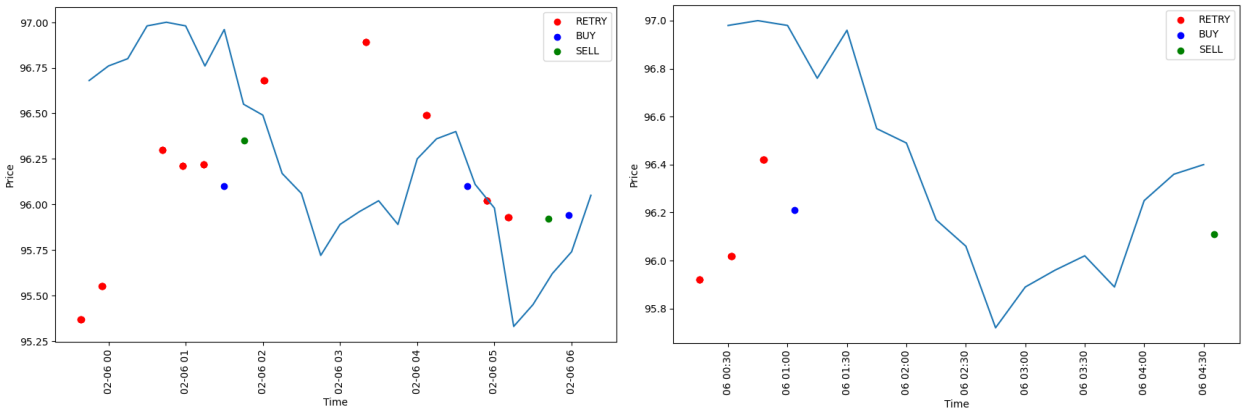
10 pav. BTCBUSD prekyba



(a) Tendencijos sekimas

(b) Grįžimas prie vidurkio

11 pav. ETHBUSD prekyba



(a) Tendencijos sekimas

(b) SMA strategija

12 pav. LTCBUSD prekyba

Gauti rezultatai robotui prekiaujant pritaikius anksčiau minėtas strategijas. Kaip matome robotas dažnai bando atlikti pakartotinus užsakymus dėl nustatyto ”LIMIT” užsakymo tipo ir galiausiai ne visada užsakymas pavyksta. Pastebima, kad grįžimo prie vidurkio strategija, atlieka žymiai mažiau bandymų prekiauti lyginant su tendencijos sekimu. Bendrai paėmus visos su visomis poros buvo atlikta nors vienas pirkimas ir pardavimas. Taip pat pastebima, jog šios kriptovaliutos turi labai panašią grafiko tendenciją

Strategija	Krepšelio pokytis
Tendencijos sekimas	+5.54 BUSD
Grįžimas prie vidurkio	-0.32 BUSD

3 lentelė. Prekybos rezultatai

Tendencijos sekimo strategija atneše šiek tiek pelno po nakties prekybos. Tai galimai įtakoja didesnis atliktų prekybos kiekis, nes bandoma tiesiog pasipelnyti iš tendencijos į kurią kryptį keliaujama. Grįžimo prie vidurkio strategija atliko mažai užsakymų ir atnešė labai maža nuostolį. Ši strategija galbūt nėra pats geriausias pasirinkimas trumpalaikiui prekiavimui tokioje aplinkoje.

7. Išvados

Taigi apibendrinant šiame darbe atlikti keli dalykai

- atlikta autoregresinių modelių analizė,
- parinkta naudoti ARIMA modelį prognozėmis,
- sukurtas robotas gebantis prekiauti remiantis strategijomis testiniame tinkle

Neišvengiama, jog tokiame trumpame darbe būtų aptartos visos įmanomos sritys ir atvejai, taigi tolimesnei tyrimo eigai pasiūlymas nagrinėti kitus modelius ir duomenų rinkinius, kad būtų galima palyginti prognozavimo būdus aptartais šiame darbe. Taip pat galima analizuoti alternatyvias strategijas kurias galima pritaikyti robotui ir jas palyginti.

Sąvokų apibrėžimai

API - Aplikacijų programavimo sąsaja (angl. Application programming interface), tai sistemos suteikiama sąsaja, kuria galima naudotis norint pasiekti tos sistemos funckionalumą ar apsikeisti duomenimis.

Autoregresinis modelis - Statistinis modelis yra autoregresinis, jei jis numato būsimas reikšmes pagal praeities reikšmes. Pavyzdžiui, autoregresinis modelis gali siekti numatyti būsimas akcijų kainas, remiantis ankstesniais rezultatais.

Euristinis algoritmas - euristiniai algoritmai yra tokios intelektualios optimizavimo uždavinių sprendimo priemonės, kuriomis siekiama rasti aukštos kokybės (bet nebūtinai optimalius) sprendinius per priimtiną laiką. Algoritmas negarantuoja, kad rastas sprendimas bus optimalus. [MBB09]

FOK - ”Fill or Kill” - limitinis sandorio tipas kai už norimą kainą, atlikti sandorį tik jei jis yra pilnai įvykdomas.

GTC - ”Good til cancelled” - limitinis sandorio tipas kai už norimą kainą, sandoris yra paliekamas rinkoje iki tol kol pilnai įvykdomas arba rankiniu būdu atšaukiamas.

IOC - ”Immediate or cancel” - limitinis sandorio tipas kai už norimą kainą, sandoris yra atliekamas tik tada jei nors dalis sandorio yra iškarto įvykdoma.

Kriptovaliuta - Kriptovaliuta yra skaitmeninė arba virtuali valiuta, kuri yra apsaugota kriptografija, todėl beveik neįmanoma jos padirbti ar išleisti dvigubai.

Kriptovaliutų pora - Kriptovaliutų pora rinkoje yra naudojama prekiaujant. Kriptovaliutos yra susietos poromis, norint nusipirkt BTC valiutos pirmiausia reikia surasti galimus keitimo variantus, jei egzistuoja pora BTC/BUSD, galima nusipirkti BTC kriptovaliutos uz turimas BUSD valiutas. Dažnu atveju rinkoje pasidėjus (”FIAT”) valiuta prekybos rinką ją konvertuoją i panašią valiuta kaip BUSD, USDT ar BNB. prekybos rinką ją konvertuoją i panašių token valiuta kaip BUSD(us -regulated stablecoin), USDT (usd tether) ar BNB (binance coin)

LIMIT užsakymas - Prekybos rinkoje bandomas atlikti užsakymas su vartotojo nustatyta kaina.

MARKET užsakymas - Prekybos rinkoje atliekamas užsakymas automatiškai gaunant geriausia siūloma kainą rinkoje tuo metu.

MSE - Vidutinė kvadratinė paklaida, angl. (Mean squared error)

MAPE - Vidutinė bsoliučioji paklaida išreikšta procentais, angl. (Mean absolute percentage error)

Literatūra

[Aro21] Ran Aroussi. Yfinance, 2021. URL: https://github.com/ranaroussi/yfinance (tikrinta 2023-01-07).

[CE20] David Carl ir Christian Ewerhart. Ethereum gas price statistics. *University of Zurich, Department of Economics, Working Paper*, (373), 2020.

[Chi18] Wan Le Chi. Stock price forecasting based on time series analysis. *AIP Conference Proceedings*, tom. 1967 numeris 1, p. 040032. AIP Publishing LLC, 2018.

[Coi] CoinMarketCap. Global cryptocurrency charts. Total cryptocurrency market cap. the graph shows the total market cap of all cryptoassets, including stablecoins and tokens. URL: https://coinmarketcap.com/charts/ (tikrinta 2022-12-28).

[Gar+18] Shruti Garg ir k.t. Autoregressive integrated moving average model based prediction of bitcoin close price. *2018 international conference on smart systems and inventive technology (ICSSIT)*, p.p. 473–478. IEEE, 2018.

[Hua20] Yiqing Hua. Bitcoin price prediction using arima and lstm. *E3S Web of Conferences*, tom. 218, p. 01050. EDP Sciences, 2020.

[Mac+14] Alec MacDonell ir k.t. Popping the bitcoin bubble: an application of log-periodic power law modeling to digital currency. *University of Notre Dame working paper*:1–33, 2014.

[MBB09] Alfonsas Misevičius, Vytautas Bukšnaitis ir Jonas Blonskis. Euristinių algoritmų klasifikavimas. *Information & Media*, 48:117–126, 2009.

[MH97] Spyros Makridakis ir Michele Hibon. Arma models and the box–jenkins methodology. *Journal of forecasting*, 16(3):147–163, 1997.

[MSG14] Prapanna Mondal, Labani Shit ir Saptarsi Goswami. Study of effectiveness of time series modeling (arima) in forecasting stock prices. *International Journal of Computer Science, Engineering and Applications*, 4(2):13, 2014.

[Nak08] Satoshi Nakamoto. Bitcoin: a peer-to-peer electronic cash system [white paper], 2008. URL: https://bitcoin.org/bitcoin.pdf (tikrinta 2022-12-28).

[Nas06] Guy P Nason. Stationary and non-stationary time series. *Statistics in volcanology*, 60, 2006.

[Smi+–] Taylor G. Smith ir k.t. pmdarima: arima estimators for Python, 2017–. URL: http://www.alkaline-ml.com/pmdarima.

[SP10] Skipper Seabold ir Josef Perktold. Statsmodels: econometric and statistical modeling with python. *9th Python in Science Conference*, 2010. (Tikrinta 2023-01-12).

[XL19] Jiahua Xu ir Benjamin Livshits. The anatomy of a cryptocurrency pump-and-dump scheme. *28th USENIX Security Symposium (USENIX Security 19)*, p.p. 1609–1625, 2019.