UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

CryptoManagement

propusă de

Apușcășiței Silviu-Alexandru

Sesiunea: 02, 2020

Coordonator științific

Drd. Colab. Florin Olariu

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

CryptoManagement

Apușcășiței Silviu-Alexandru

Sesiunea: 02, 2020

Coordonator științific

Drd. Colab. Florin Olariu

Anexa 4

Avizat,

Îndrumător Lucrare de Licenţă

Titlul, Numele şi prenumele _____

Data _____ Semnătura _____

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemntatul(a) ……

domiciliul în ……..

născut(ă) la data de ……..…., identificat prin CNP ……..…., absolvent(a) al(a) Universităţii „Alexandru Ioan Cuza" din Iaşi, Facultatea de …….. specializarea ……., promoţia ……., declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă cu titlul: _____

_____

_____elaborată sub îndrumarea dl. / d-na _____, pe care urmează să o susţină în faţa comisiei este originală, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă să fie verificată prin orice modalitate legală pentru confirmarea originalităţii, consimţind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice in vederea facilitării fasificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diploma sau de disertaţie şi în acest sens, declar pe proprie

răspundere că lucrarea de faţă nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, …… Semnătură student ……

Anexa 5

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Titlul complet al lucrării*", codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, *data*

Absolvent *Prenume Nume*

_____

(semnătura în original)

Cuprins

Introducere

Informații generale

Ce este o criptomonedă?

Criptomoneda sau criptovaluta este un tip de monedă digitală, virtuală, o monedă surogat, nebancară, folosită ca mijloc de plată. Denumirea de criptomonedă indică faptul că acest mijloc de plată utilizează criptografia și este descentralizat pentru a controla tranzacțiile și preveni dubla cheltuială, o problemă curentă pentru valutele digitale. Deseori se face greșeala ca moneda virtuală (criptomoneda) să fie considerată o monedă electronică. Pe scurt, o monedă electronică este varianta electronică a bancnotelor și monedelor, care se poate stoca pe un dispozitiv de plată electronic. Acesta este folosit, de regulă, pentru a face plăți electronice de mică valoare.

Ideea unei monede digitale nu este nouă și a apărut în urmă cu mai bine de 20 de ani. În 1998, Wei Dai a adus în discuție ideea unor bani digitali, iar Nick Szabo a creat Bit Gold. Cele două idei nu au avut succes în acea perioadă, însă 10 ani mai târziu Satoshi Nakamoto a publicat documentația pentru Bitcoin, moneda care avea să transforme lumea.

Criptomonedele cele mai populare sunt Bitcoin, Ethereum, Litecoin sau Ripple. Toate criptomonedele sunt puțin diferite, dar toate împărtășesc următoarele caracteristici:

- **Sunt unidirecționale.** După ce trimiți o criptomonedă și rețeaua a confirmat tranzacția, nu o mai poți lua înapoi, decât dacă acea persoană îți va trimite una înapoi.
- **Nu este necesară o carte de identitate.** Rămâi, în mare parte, anonim. Ai nevoie doar de un portofel digital.

- **Sunt rapide și pot fi obținute de oricine, oriunde în lume.** Tranzacțiile sunt transmise imediat în toată rețeaua și sunt confirmate în câteva minute.
- **Numărul criptomonedelor este limitat.** Cantitatea totală este controlată de rețea.

Tema

Ca și funcționalitate generală, tema proiectului reprezintă o aplicație care automatizează un proces de monitorizare a valorilor criptomonedelor. Aceasta intră frecvent pe un site de cryptocurrency, și, cu ajutorul unui algoritm care are la bază valori mai vechi ale monedei respective, notifică un utilizator atunci când este o oportunitate de a investi în acea monedă. Prin investiție se intelege atât cumpararea cât si vânzarea acelei monede. Astfel, un utilizator își poate alege, prin intermediul unei interfețe, care monede să fie urmărite de către aplicație, notificările fiindu-i ulterior transmise prin email.

Astfel, se reduce timpul petrecut de un utilizator în ținerea manuală a evidenței acestora. Ba mai mult, aplicația vine în ajutorul utilizatorului și cu o predicție a unei valori viitoare ale acelei monede, pentru a-l determina să ia o decizie atunci când se ivește o oportunitate (aceea de a investi sau nu).

Motivație

Am ales această temă ca și proiect de licență deoarece evenimentul cu criptomonede a luat amploare în ultimul timp, fiind unul de actualitate. Acest proiect este menit să vină în ajutorul utilizatorilor deoarece este ușor de folosit, astfel economisindu-se timp prețios de care nu toată lumea dispune. Spre exemplu, pentru cineva care lucrează 8 ore pe zi iar apoi restul timpului și-l petrece cu familia și totuși ar vrea să investească în criptomonede, este greu să țină evidența pieței de valori ale acestora. Aproape jumătate din investitori au renunțat în ultimii ani, unul din principalele motive fiind lipsa timpului.

Deoarece tendința în viitor este de automatizare a proceselor, iar această aplicație automatizează un proces care poate fi extins către investiția în

criptomonede, consider că aplicația poate atrage potențiali clienți pentru folosirea sau menținerea acestui produs. Ba mai mult, consider că investiția în criptomonede va începe din nou să crească odată cu folosirea aplicației de către utilizatori.

Gradul de noutate

Tema proiectului, așa cum am precizat și mai sus, reprezintă automatizarea unui proces. În prezent multe acțiuni care includ un set anumit de pași se preferă a fi automatizate. Aplicația este una de actualitate, incluzând o tematică destul de controversată, aceea a criptomonedelor și proiectarea unui algoritm de machine learning. De-a lungul timpului s-a observat progresul pe care l-au adus algoritmii de machine learning în eficientizarea proceselor.

În timp au existat mai mulți algoritmi de tranzacționare, care se bazează pe strategii simple. Un exemplu foarte ușor și des întâlnit este setarea unui maxim (în cazul în care vrem să vindem) al unei valori pentru o monedă, iar când acel maxim este atins, atunci se face tranzacția. Spre deosebire de acest algoritm, aplicația pe care am implementat-o nu necesită setarea niciunui maxim/minim. Dacă folosim exemplul descris anterior, cu un maxim pentru a vinde acea monedă, algoritmul implementat în proiectul pentru licență funcționează astfel: pornind de la o valoare de start, cât timp valoare monedei este în creștere, algoritmul așteaptă; astfel câștigul obținut în urma unei tranzacții poate fi mult mai mare, deoarece acel maxim poate fi depășit oricând, aplicația asigurându-se că aduce profit maxim utilizatorului, deoarece aceasta va trimite notificări acestuia doar după ce valoarea monedei va începe din nou să scadă.

O altă modalitate de tranzacționare automată cu risc minim este construirea unui algoritm de arbitraj între exchange-uri. În forma lor cea mai simplă, acești algoritmi urmăresc simultan mai multe burse pentru o anumită pereche de monede (de exemplu, ADA și ETH) și realizează simultan operațiunea de a cumpara si de a vinde. Astfel, urmărind prețurile de pe două exchange-uri oarecare (cum ar fi Binance și Bittrex), în momentul în care discrepanța este suficient de mare încât să existe oportunitatea unui profit, un algoritm poate cumpara ADA cu ETH pe Binance,

simultan cu vânzarea de ADA pentru ETH pe Bittrex, încasând diferența. Această diferență trebuie să fie suficient de mare pentru a putea acoperi costurile de tranzacționare, costurile de alimentare a contului, precum și riscul păstrării blocate a unor sume in aceste criptomonede. Și, pentru că viața nu poate fi simplă, trebuie să acopere și riscul de a nu obține prețul teoretic din momentul observației (din cauza mișcărilor care pot avea loc înainte ca acțiunea algoritmului să ajungă pe market) sau de a avea erori de comunicare cu exchange-ul, ce pot lasă algoritmul "descoperit", nereușind să realizeze ambele operațiuni simultan. Revenind la algoritmul din aplicația proprie, deși acesta așteaptă ca valoarea monedei să crească, iar apoi când va scădea, să trimită notificare către utilizator, dacă diferența de procentaj dintre valoare inițială și cea curentă nu este mai mare sau egală cu 5% acesta nu va mai trimite notificarea. Această diferență asigură profit utilizatorului chiar dacă acesta întârzie efectuarea tranzacției, reușind astfel să acopere și costurile de tranzacționare.

Metodologia folosită

Metodologia folosită se bazează pe lucrul segmentat pe anumite componente în intervale de timp. Astfel am început să lucrez la modulul în Java, am finalizat configurările necesare pentru a rula headless o instanță de webdriver pe un site de cryptocurrency. Ulterior am urcat aplicația pe Heroku, unde am fost nevoit să adaug o altă serie de configurări, printre care și adăugarea unui 3-party pentru trimiterea de emailuri, compatibil cu Heroku, și anume SendGrid. Ulterior, pentru un plus de noutate, am generat, folosind ML.NET framework, un algoritm de machine learning. Neavând suport pentru .NET în Heroku, a trebuit să caut un alt server online pentru a pune și API-ul realizat în .NET (și anume Microsoft Azure). Apoi a urmat integrarea dintre modulul de Java cu cel de .NET și crearea unei interfețe prin care utilizatorul să comunice cu aplicația. În partea de cercetare au fost incluse interacțiunea utilizatorilor cu criptomonedele (care sunt cele mai folosite, de unde pot face rost de valori mai vechi pentru a antrena algoritmul?), și generarea algoritmului de machine learning folosind ML.NET framework. Un plus de complexitate l-a reprezentat faptul că odată antrenat algoritmul cu valori până în data

curentă (și implicit generat), acesta își pierde acuratețea cu timpul. Drept pentru care, pentru fiecare predicție pe care acesta o face, ține cont și de ultimele 10 valori ale monedei. Un rol important în automatizarea procesului l-a avut și punerea celor două module (Java si .NET) pe Heroku și Azure.

Descrierea sumară a soluției

Un principal factor pentru care oamenii nu mai investesc în criptomonede este lipsa timpului. Nefiind principala sursă de venit a acestora, ei nu-și mai permit să consume timp pe site-urile de exchange pentru o bună investiție în criptomonede. O altă problemă a acestora este managementul monedelor, în cazul în care se urmăresc mai mult de zece monede acesta devine dificil. Și prin management mă refer la monitorizarea frecventă a monedelor, atât a valorilor curente cât și a valorilor din trecut și luarea deciziei de a investi sau nu în acele monede. Toate acestea puse la un loc duc la o acumulare enormă de timp pe care majoritatea dintre oameni nu și-l permit.

Drept urmare, aplicația este menită să vină în ajutorul acestora. Un utilizator poate alege dintr-o gama de monede cunoscute pentru care ar fi interesat să investească. Sau acesta are posibilitatea pur și simplu de a selecta toate monedele puse la dispoziție. După care procesul este automatizat, managementul monedelor fiind făcut de aplicație. Nu este folosită nicio baza de date, deoarece sunt folosite puține valori, care sunt salvate în timp real, astfel reducându-se costurile. Aplicația intră pe un site de statistici frecvent, la un interval de timp, și verifică, pentru fiecare monedă selectată de utilizator, dacă este o oportunitate pentru acesta de a face o tranzacție, bazat pe valorile din trecut ale acesteia. Astfel utilizatorului revenindu-i doar sarcina de a-și verifica zilnic email-urile. Fapt care oricum face parte din rutina zilnică a fiecăruia dintre noi.

Diagrama următoare reprezintă comunicarea dintre componentele aplicației:

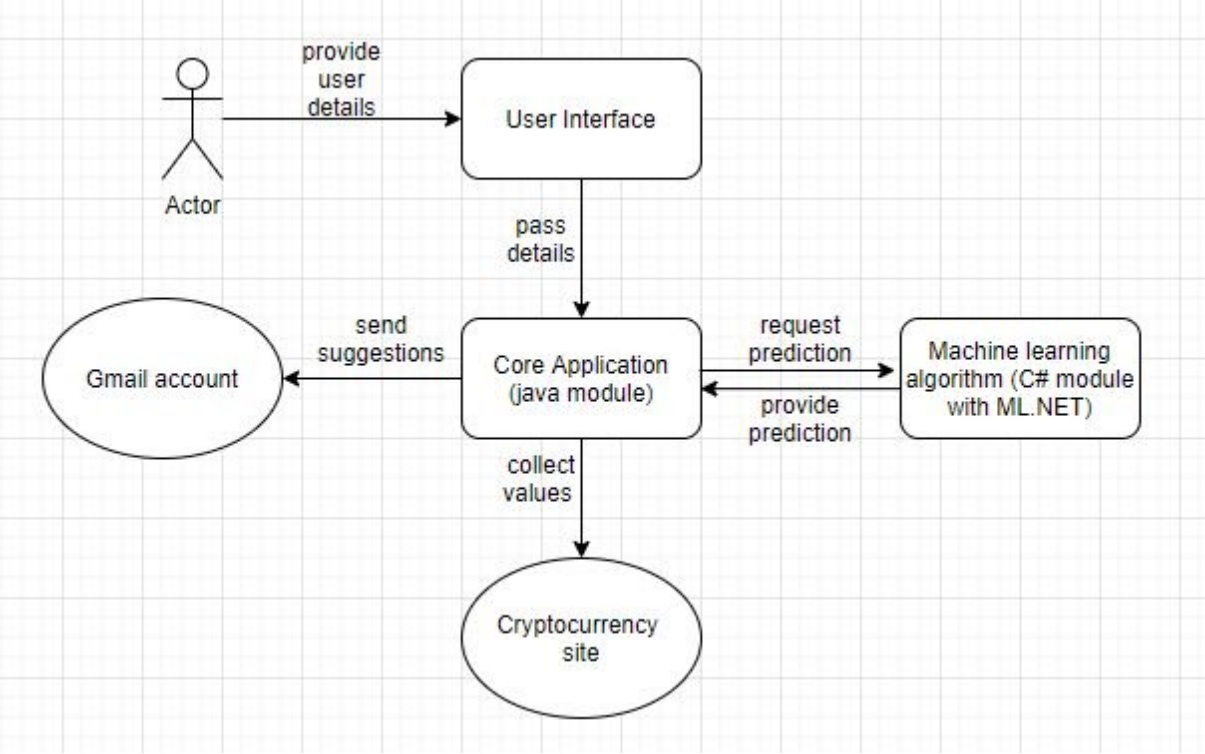


Figura 1 - Descrierea flow-ului end-to-end

Structura lucrării

Primul capitol prezintă proiectarea modulului în Java, care include interacțiunea aplicației cu site-ul de cryptocurrency și dezvoltarea algoritmului de luare a deciziilor.

Capitolul 2 prezintă proiectarea modulului în C# care conține generarea algoritmului de machine learning pentru predicția unei valori viitoare al unei criptomonede și expunerea acesteia într-un API.

Capitolul 3 conține detalii despre punerea celor doua module (Java și C#) pe cele două platforme online, Heroku și Azure.

Ultimul capitol exemplifică interacțiunea utilizatorului cu aplicația.

Contribuții

Pentru modulul de .NET:

- generarea framework-ului de machine learning care prezice următoarea valoare pentru o anumită monedă
- antrenarea algoritmului cu toate valorile monedelor
- menținerea acurateții algoritmului prin compararea valorilor curente ale monedelor cu cele din timpul antrenării
- crearea API-ului care servește valoarea prezisă de algoritm
- expunerea API-ului pe un server online (Microsoft Azure)

Pentru modulul de Java:

- îmbinarea unui tool pentru testarea automată cu un algoritm care oferă sugestii de a face tranzacții cu criptomonede (vânzări/cumpărări)
- sugestiile sunt trimise utilizatorului prin email (folosind SendGrid ca și 3-party)
- apelarea algoritmului de machine learning și atașarea valorii prezisă de acesta in email
- urcarea proiectului pe un server online (Heroku)
- creare unui profil maven care să ruleze metoda principală din proiect (profil ce este utilizat de un worker de pe Heroku)
- adăugarea de configurări necesare webdriver-ului pentru a rula pe server
- integrarea repository-ului de pe Github cu serverul
- crearea unei interfețe pentru ca utilizatorul să comunice cu aplicația

Am ales să nu folosesc un algoritm de machine learning pentru sugestiile de tranzacții ci să implementez de la zero un algoritm care să se plieze pe nevoile mele. Totuși pentru partea în care se prezice ce valoare următoare va avea moneda am folosit ML.NET fiind simplu de generat și de utilizat.

Capitolul 1 - Proiectarea modulului în Java

1. Interacțiunea cu site-ul

Site-ul folosit pentru colectarea periodică a valorilor criptomonedelor este https://www.tradingview.com/ si conține informații despre toate monedele existente, cât și despre exchange-urile disponibile.

Așa cum am precizat mai sus, aplicația are la bază un tool pentru testarea automată, și anume Selenium, iar, cu ajutorul pattern-ului din Java numit Page Object mi-am modelat clasele pe pagini respective de pe site.

Pentru partea de webdriver am ales să folosesc unul pentru Chrome iar configurările acestuia au fost făcute folosind WebDriverManager.

```
public static WebDriver getInstance(){
    if(driver == null){
        WebDriverManager.chromedriver().setup();
```

Fig 2.1 - Instanțierea driver-ului

Acesta se asigură că folosește, pentru fiecare rulare, ultima versiune de Chrome driver.

```
Starting ChromeDriver 79.0.3945.36 (3582db32b33893869b8c1339e8f4d9ed1816f143-refs/branch-heads/3945@{#614}) on port 2531
Only local connections are allowed.
Please protect ports used by ChromeDriver and related test frameworks to prevent access by malicious code.
Jan 08, 2020 2:32:39 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
```

Fig 2.2 - Logguri care descriu driver-ul

Folosind Page Object ca și pattern, clasele reprezintă elementele din paginile site-ului. Ele nu au voie să expună mai departe web elementele, ci doar sa ofere interacțiunea altor obiecte cu acestea. Colectarea web elementelor din pagină se face în momentul instanțierii clasei respective cu ajutorul selectorilor de tip css sau xpath.

```
public class TradingViewHomePage {

    private WebDriver driver;

    @FindBy(css = "span[class='tv-header__dropdown-text']")
    private WebElement signInButton;

    @FindBy(xpath = "//li/a[contains(text(),'Markets')]")
    private WebElement marketButton;

    public TradingViewHomePage(WebDriver driver) { this.driver = driver; }

    public TradingViewLoginPage clicktoSignInButton(){
        Driver.waitForElementToLoad(signInButton,  timeOutInSeconds: 20);
        signInButton.click();
        return PageFactory.initElements(driver, TradingViewLoginPage.class);
    }

    public TradingViewMarketPage clickToMarketButton(){
        Driver.waitForElementToLoad(marketButton,  timeOutInSeconds: 20);
        Driver.waitForElementToBeClickable(marketButton,  timeOutInSeconds: 20);
        marketButton.click();

        return PageFactory.initElements(driver, TradingViewMarketPage.class);
    }
}
```

Fig 2.3 - Structura clasei de tip Page Object

2. Algoritmi folosiți

Așa cum am precizat și mai sus, am ales să proiectez un algoritm care se potrivește pentru nevoile pe care voiam să le rezolv, astfel, deciziile pe care le ia acesta să fie cât mai profitabile. Algoritmul funcționează folosind trei valori pentru fiecare monedă. Și anume: o valoare de start, care este setată în momentul în care este pornită aplicația, o valoare curentă, care este preluată de pe site la fiecare rulare a task-ului, si o valoare precedentă, salvată de la ultima rulare.

Aplicația oferă sugestii de tranzacții atât în momentul în care valoarea unei monede scade foarte mult, cât și atunci cand aceasta crește. Astfel, dacă luăm de exemplu cazul în care moneda crește, algoritmul ia decizia de a vinde acea monedă doar în momentul în care care valoare ei scade, și este mai mare decât valoarea inițială cu un procent de 5%.

Fig 3.1 - Grafic cu valorile monedei Bitcoin

(Sursa: https://www.tradingview.com/symbols/BTCUSD/)

Graficul de mai sus reprezintă valorile monedei Bitcoin pe perioada lunii mai, până la începutul lunii aprilie din 2019. Dacă urmărim evoluția monedei, observăm că aceasta este în creștere. Să presupunem că pornim aplicația la începutul lunii mai, iar valoarea inițială va fi cea din data de 3. În momentul în care valoare ei scade, fapt care se petrece între data de 11 și 12 mai, algoritmul verifică dacă valoarea curentă este mai mică decât ultima valoare înregistrată, și mai mare cu 5% față de valoarea inițială. 5% reprezintă procentul care asigură profit în urma costurilor de tranzacție. Dacă se respectă condiția, atunci acesta va lua decizia de a vinde moneda, iar valoarea inițială va deveni valoare curentă și execuția acestuia va continua. Dacă nu, acesta nu va lua decizia de a vinde și va mai aștepta, continuând execuția, pană în data de 15-16, unde se observă din nou o scădere a valorii monedei. În acel moment se face din nou verificarea, dacă se respectă condiția atunci acesta va lua decizia de a vinde, iar dacă nu, va mai aștepta.

Așa cum nu se poate face profit dintr-o monedă doar prin vânzarea acesteia, algoritmul funcționează și în sens invers, când este nevoie ca utilizatorul să cumpere moneda respectivă la un preț convenabil. Astfel, următorul grafic înregistrează valorile aceleiași monede, și anume Bitcoin pentru perioada lunii august din 2019.

Fig 3.2 - Grafic cu valorile monedei Bitcoin

(Sursa: https://www.tradingview.com/symbols/BTCUSD/)

Dacă urmărim logica algoritmului, atunci acesta cel mai probabil va lua decizia de a vinde moneda pe data de 9 august, când se observă o scădere a valorii acesteia, devenind valoare de start pentru următoarele execuții. Astfel urmează o suită predominantă de scăderi ale monedei. Dacă la prima creștere care urmează a monedei nu sunt îndeplinite condițiile, și anume, valoarea curentă să fie mai mică cu 5% față de valoarea inițială, la a doua creștere, cea din data de 14 august probabil procentajul va fi depășit, algoritmul luând decizia de a cumpara moneda respectivă.

Așa cum se observă din al doilea graf, algoritmul are și unele slăbiciuni, nefiind unul perfect, deoarece momentul oportun de a cumpăra Bitcoin ar fi fost la ultima creștere, cea din data de 16 august. Însă, niciun algoritm nu este perfect, iar investiția în criptomonede vine la pachet și cu riscuri.

Detaliile de implementare pot fi urmărite în imaginea următoare.

```
public List<Object> takeDecision(Double startValue, Double lastValue, Double currentValue, String coin){

    List<Object> arrayDecision = new ArrayList<>();
    String message = "";

   if (currentValue > lastValue) {
        if ((100 * (startValue - currentValue)) / startValue > 5) {
            Double percentage = (100 * (startValue - currentValue)) / startValue;
            message = "Buy " + coin + ", the current price is " + currentValue + " USD. It is lower than " + startValue + " with " + percentage + "%.";
            startValue = currentValue;
        }
   }

   if (currentValue < lastValue) {
        if ((100 * (currentValue - startValue)) / currentValue > 5) {
            Double percentage = (100 * (currentValue - startValue)) / currentValue;
            message = "Sell " + coin + ", the current price is " + currentValue + " USD. It is higher than " + startValue + " with " + percentage + "%.";
            startValue = currentValue;
        }
    }

    lastValue = currentValue;
```

Fig 4.1 - Detalii de implementare ale algoritmului

3. Rularea periodică a taskului

Programul pornește din clasa “Main”, care are ca scop inițializarea valorilor criptomonedelor și rularea taskului periodic pentru monitorizarea acestora, luând decizii pe baza algoritmului prezentat mai sus.

Acesta folosește interfața Runnable din Java pentru execuția periodică a taskului, fiind setat la intervalul de o oră.

```
public static void main(String args[]) throws Exception {

    /* ------------------------------------------------------------------------ */
    applicationManager.initializeValues();

    Runnable runnable = new Runnable() {
        int count = 0;
        public void run() {
            System.out.println("************************************ run ***************************************");
            applicationManager.manage();
            if(count % 23 == 0) {
                applicationManager.sendLifeServerCheckEmail();
            }
            System.out.println("Waiting for the next run...");
            count++;
        }
    };
    ScheduledExecutorService service = Executors.newSingleThreadScheduledExecutor();
    service.scheduleAtFixedRate(runnable,  initialDelay: 0,  period: 1, TimeUnit.HOURS);
}
```

Fig 4.2 - Rularea periodica a task-ului

Corul aplicației îl reprezintă funcția “manage()” care:

- interacționează cu site-ul de cryptocurrency prin logarea unui user pe acesta

```
public void manage(){
    tradingViewCommand = new TradingViewCommand();
    tradingViewCommand.login();
```

Fig 4.3 - Interacțiunea cu site-ul de cryptocurrency

- prin colectarea valorii curente pentru fiecare monedă

```
Integer index = 0;
while (index < coinList.size()){
   tradingViewCommand.goToCurrency(CryptoCoinMapping.getAppValue(coinList.get(index)));
```

Fig 4.4 - Accesarea valorii curente pentru o moneda

- apelarea algoritmului de luare a unei decizii și updatarea valorilor

```
List<Object> list =  mathCommand.takeDecision(startValue.get(index), lastValue.get(index), I
startValue.set(index, Double.valueOf(String.valueOf(list.get(0))));
lastValue.set(index, Double.valueOf(String.valueOf(list.get(1))));
```

Fig 4.5 - Interacțiunea cu algoritmul de luare a deciziilor

- și trimiterea email-ului utilizatorului, în funcție de răspunsul dat de algoritm

```
if (list.get(2) != ""){
    System.out.println(list.get(2));
    try {
        MailUtil.sendMail( userReadPr
    } catch (IOException e) {
        e.printStackTrace();
    }
}
 index++;
```

Fig 4.6 - Trimiterea de email către utilizator

4. Descrierea flow-ului pe site

1. Se inițializează driver-ului de Chrome.

```
private Driver driver =  new Driver();
public static WebDriver webDriver;

public WebDriverBuilder(){
    webDriver = driver.getInstance();
}

public static void set(String site){
    AppReadProperties appReadProperties = new AppReadProperties();
    if ("tradingview".equals(site)){
        webDriver.get(appReadProperties.getTradingViewURL());
    }
```

Fig 5.1 - Inițializarea driver-ului de chrome

2. Care deschide pagina de home a site-ului.

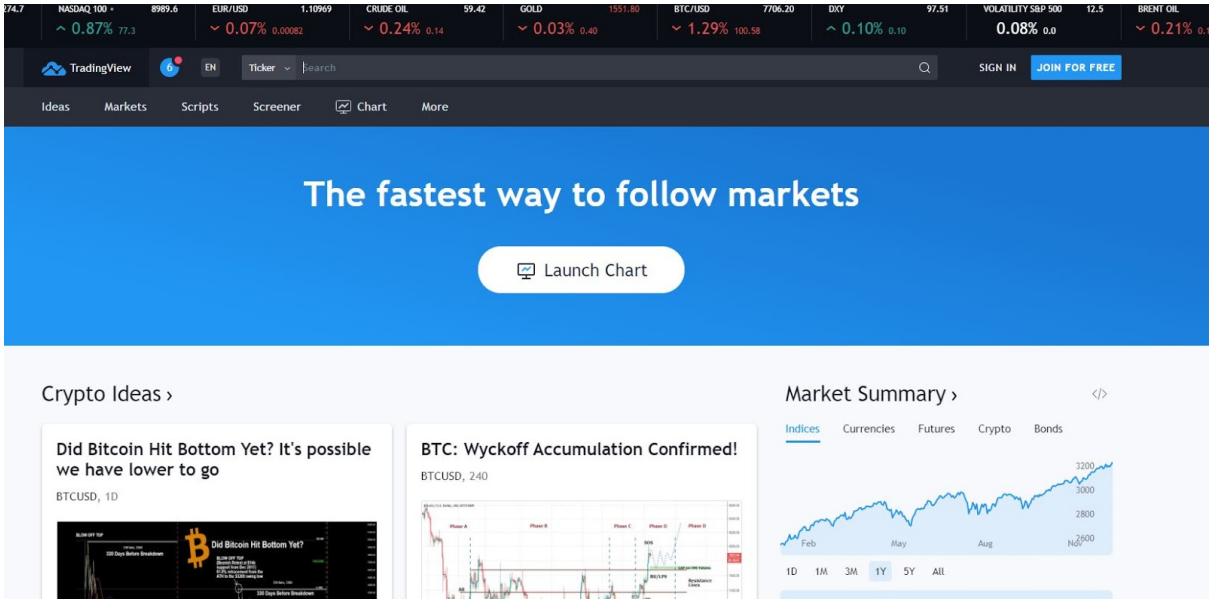


Fig 5.2 - Pagina de home a site-ului de cryptocurrency

(Sursa: https://www.tradingview.com/)

3. Folosind butonul de “SIGN IN” ajungem pe pagina de login, unde este introdus contul de gmail al aplicației și se face autentificarea.

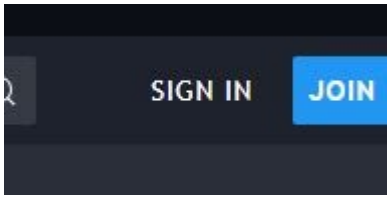


Fig 5.3 - Butonul de sign in

(Sursa: https://www.tradingview.com/)

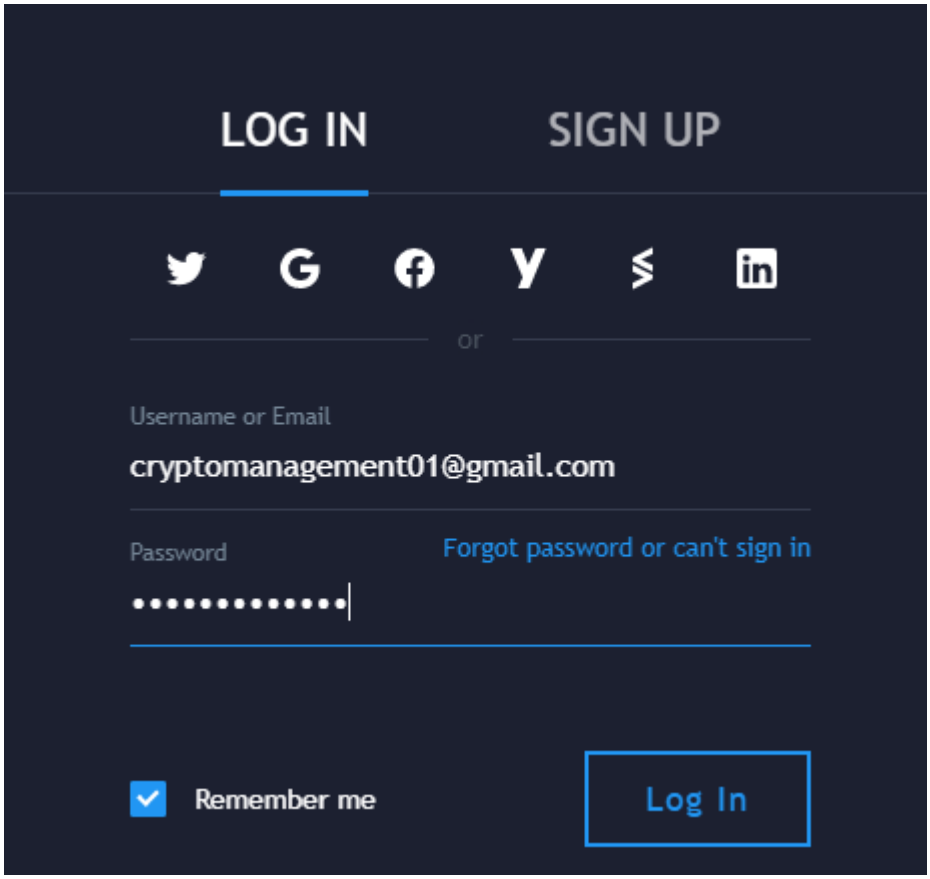


Fig 5.4 - Formularul de login

(Sursa: https://www.tradingview.com/)

4. Odată logat, navighează până la moneda respectivă, de unde este preluată valoarea curentă.

Fig 5.5 - Accesarea valorii curente pentru o moneda

(Sursa: https://www.tradingview.com/symbols/BTCUSD/)

Capitolul 2 - Proiectarea algoritmului de machine learning

1. Motivația alegerii unui algoritm de machine learning

Algoritmul pe care l-am ales pentru această aplicație este unul de predicție. Consider că aduce un plus în luarea deciziei de către utilizator faptul că acesta are o informație în plus, și anume, o valoare viitoare a monedei.

Algoritmul proiectat este unul nesupravegheat deoarece utilizează o abordare mai independentă, în care acesta învață să identifice procese și modele complexe, fără ajutorul unui om care să ofere îndeaproape orientări în mod constant. Deoarece variația pe care o au valorile criptomonedelor este foarte vastă, și depinde de cum investesc oamenii în ele, nici nu avea cum să ofere orientări, fiind dificil pentru o persoană să prezică următoarea valoare.

2. Antrenarea algoritmului și menținerea acurateții acestuia

Pentru a aduna datele necesare antrenării modelului, am descărcat date de pe una dintre cele mai mari platforme care oferă gratuit date despre istoricul criptomonedelor (https://www.investing.com/crypto/currencies).

Pentru primele 30 de monede cele mai cunoscute, am descărcat pe zile, toate datele necesare din ultimii 10 ani.

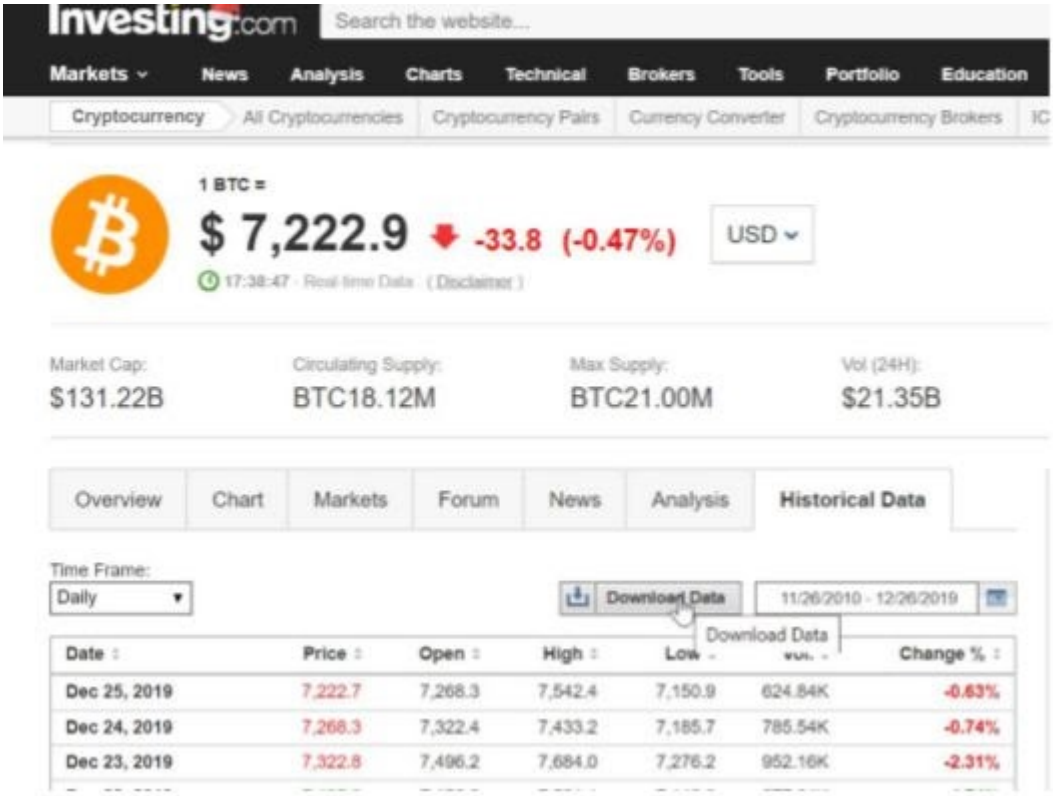
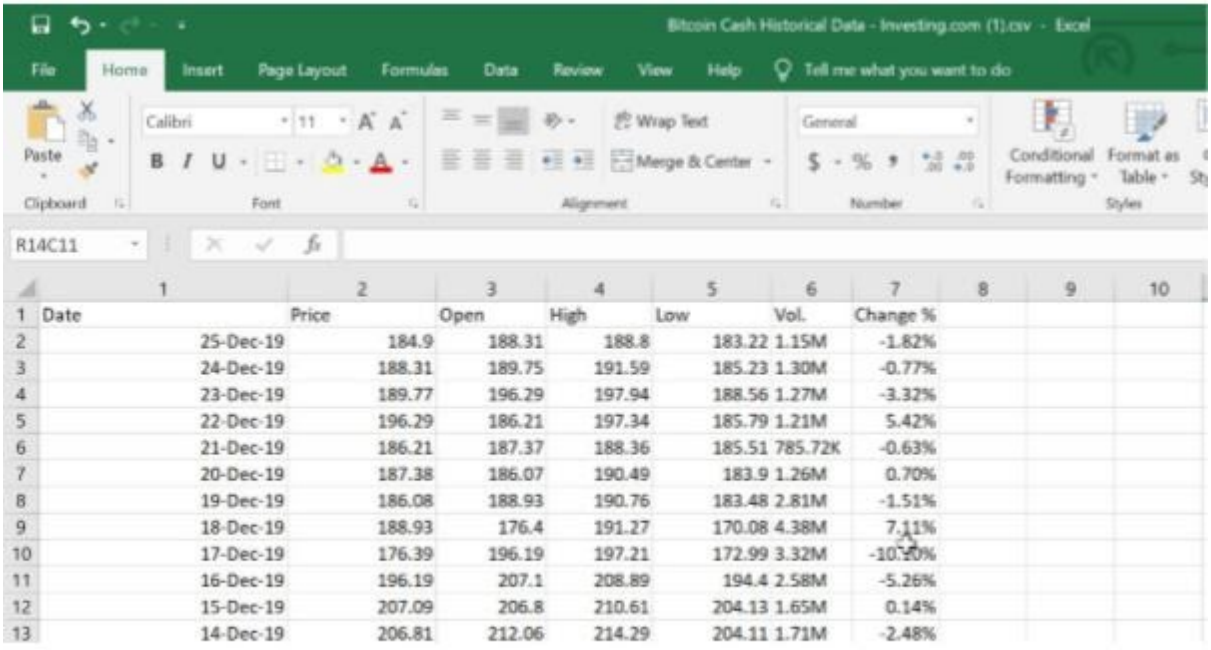


Fig 6.1 - Pagina de detalii a unei monede

(Sursa: https://www.investing.com/crypto/bitcoin)

Apoi, pentru fiecare document, am adăugat o coloană de tip discriminator care este chiar denumirea internațională a monedei (ex: pentru Bitcoin este BTC)

Exemplu de fișier:



	1	2	3	4	5	6	7
1	Date	Price	Open	High	Low	Vol.	Change %
2	25-Dec-19	184.9	188.31	188.8	183.22	1.15M	-1.82%
3	24-Dec-19	188.31	189.75	191.59	185.23	1.30M	-0.77%
4	23-Dec-19	189.77	196.29	197.94	188.56	1.27M	-3.32%
5	22-Dec-19	196.29	186.21	197.34	185.79	1.21M	5.42%
6	21-Dec-19	186.21	187.37	188.36	185.51	785.72K	-0.63%
7	20-Dec-19	187.38	186.07	190.49	183.9	1.26M	0.70%
8	19-Dec-19	186.08	188.93	190.76	183.48	2.81M	-1.51%
9	18-Dec-19	188.93	176.4	191.27	170.08	4.38M	7.11%
10	17-Dec-19	176.39	196.19	197.21	172.99	3.32M	-10.50%
11	16-Dec-19	196.19	207.1	208.89	194.4	2.58M	-5.26%
12	15-Dec-19	207.09	206.8	210.61	204.13	1.65M	0.14%
13	14-Dec-19	206.81	212.06	214.29	204.11	1.71M	-2.48%

Fig 6.2 - Fișier csv cu valorile monedei

După adaugarea discriminărilor în fiecare dintre cele 30 de fișiere, le-am combinat folosind un tool online și am reușit să fac un fișier csv. destul de mare încât algoritmul să poată învăța destule informații încât să dea predicții utile.

Algoritmul a fost antrenat cu toate valorile celor 30 de criptomonede până în ziua generării. Astfel, predicția pentru valoarea pe care o va lua o monedă în ziua următoare fiind de o acuratețe foarte bună. O problemă cu care m-am confruntat în schimb este aceea a timpului. Cum mențin această acuratețe? Cât de utile vor mai fi predicțiile peste 10 zile, o lună, sau poate un an?

Soluția la care m-am gândit în această situație este aceea de al “alimenta” cu viitoarele valori ale monedelor care vor avea să vină. Astfel, pentru fiecare predicție pe care vreau să o primesc de la algoritm, acesta va folosi, în plus pe lângă valorile cu care a fost antrenat, ultimele 10 valori pe care le-a avut moneda respectivă. Valorile sunt descărcate de pe același site, care pune la dispoziție un endpoint care poate fi apelat. Un exemplu de url care poate fi apelat pentru moneda Bitcoin este acesta:
https://min-api.cryptocompare.com/data/v2/histoday?fsym=BTC&tsym=USD&limit=1

0&api_key=409b2bce1e8c9f05d32323cf70ba4f1114cfdebc9106f4d596d5c05bf34617b9, care returnează un json cu informațiile necesare.

Apelul este de tip GET prin HttpClient

```
using var client = new HttpClient(new HttpClientHandler { AutomaticDecompression = DecompressionMethods.GZip
{
    BaseAddress = uri
};
HttpResponseMessage response = client.GetAsync(uri).Result;
response.EnsureSuccessStatusCode();
return response.Content.ReadAsStringAsync().Result;
```

Fig 6.3 - Apelul pentru valorile monedei

Iar modelul de input dat algoritmului pentru predicție conține valorile din json-ul returnat.

```
var input = new ModelInput
{
    Date = DateTime.Today.ToString(DATE_TIME_FORMAT),
    Open = lastTenOpen,
    Low = lastTenLow,
    High = lastTenHigh,
    Coin = coinName

};
ModelOutput result = ConsumeModel.Predict(input);
return result.Score;
```

Fig 6.4 - Model de input dat algoritmului pentru predicție

3. Generarea algoritmului

Deși aplicația este scrisă în Java, modulul care conține algoritmul de machine learning, după cum se observă, este scris în C#. Pentru a genera algoritmul propriu-zis am folosit framework-ul ML.NET fiind o platformă open source care permite antrenarea, construirea și modelarea lor, având ca și limbaj de programare de bază, C#.

Pentru crearea modelului de ML.NET am urmat pașii standard din Visual Studio:

a) Selectarea tipului de predicție

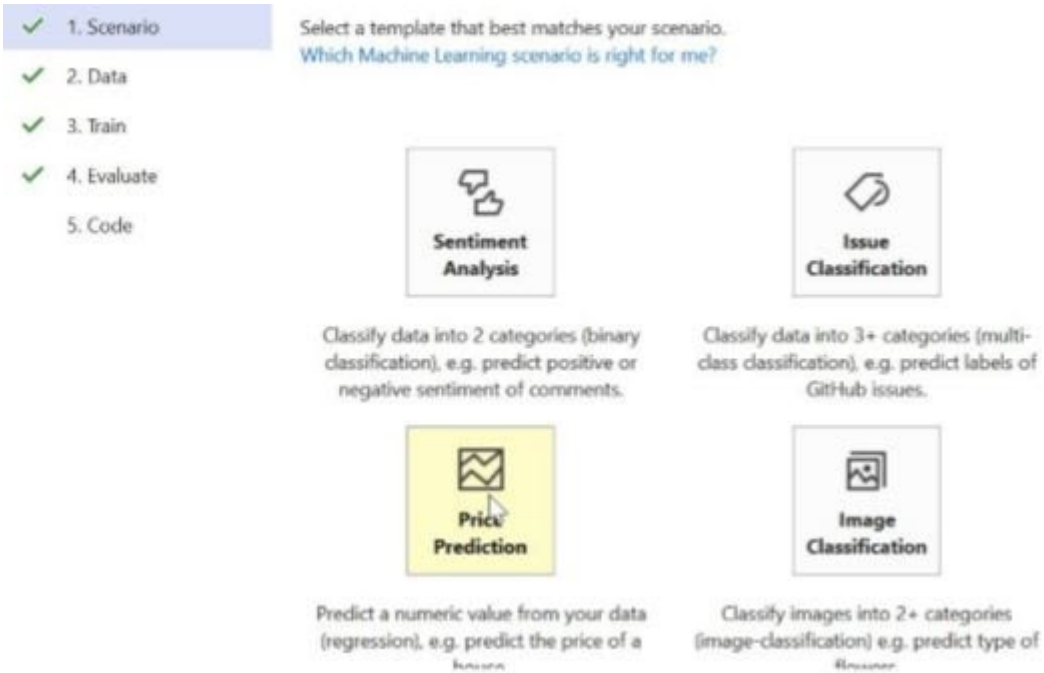


Fig 7.1 - Selectarea tipului de predicție (VS Code)

b) Selectarea fișierului de intrare și coloana pe care vrem să o prezicem

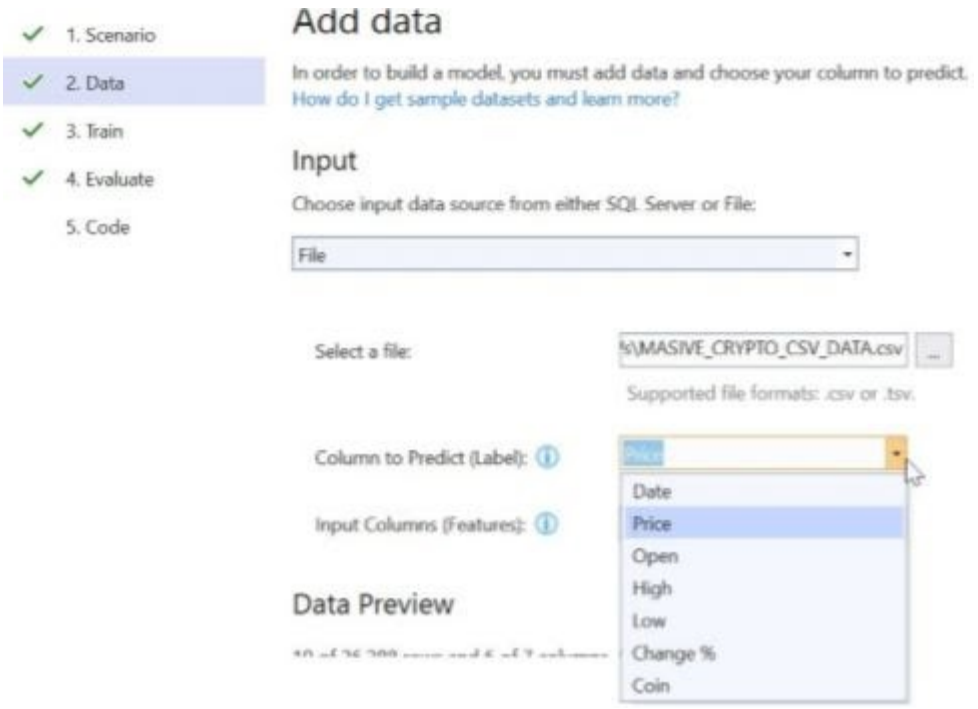


Fig 7.2 - Datele de intrare pentru algoritm (VS Code)

c) Antrenarea modelului

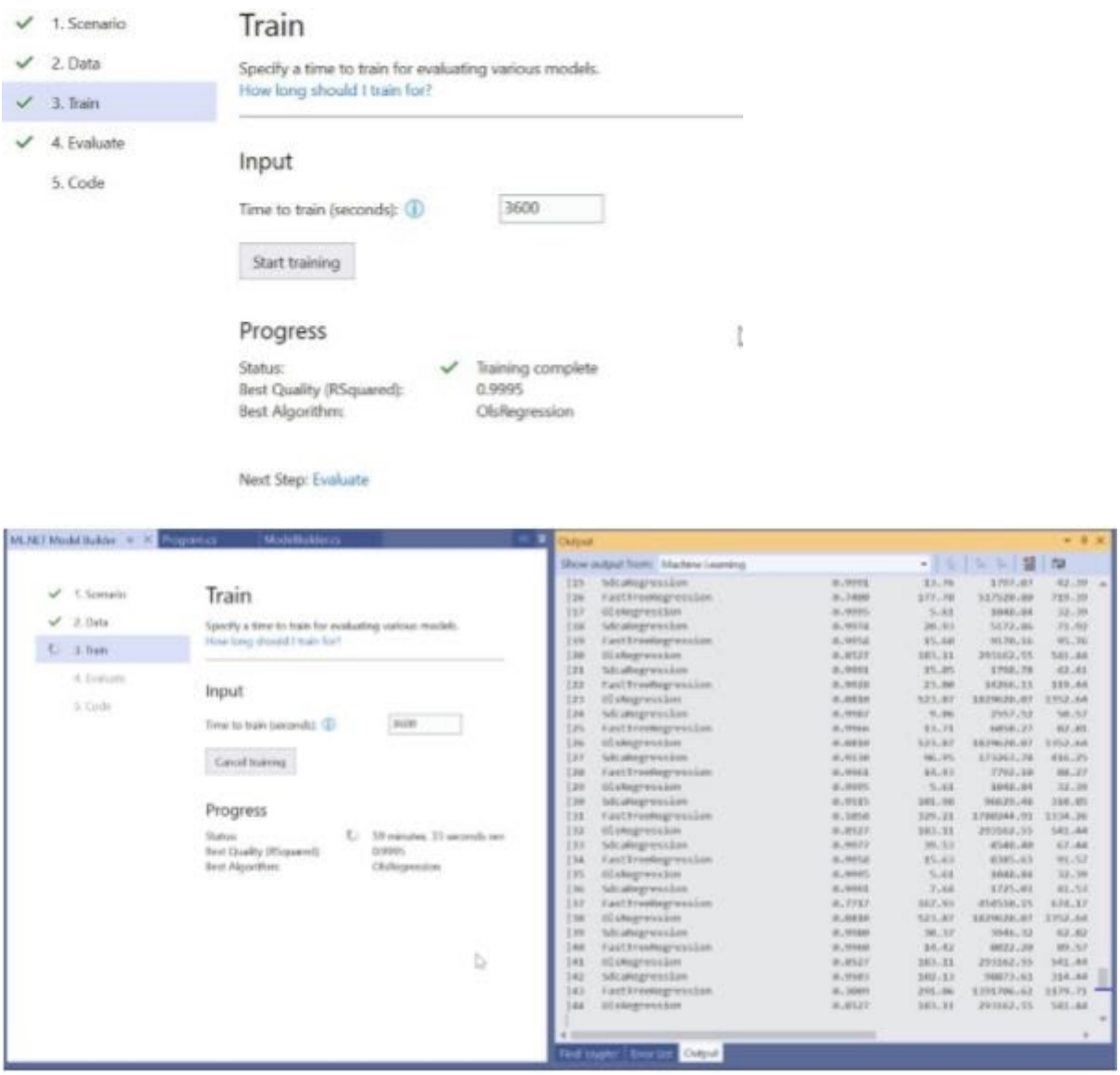


Fig 7.3 - Antrenarea algoritmului

d) Evaluarea modelului

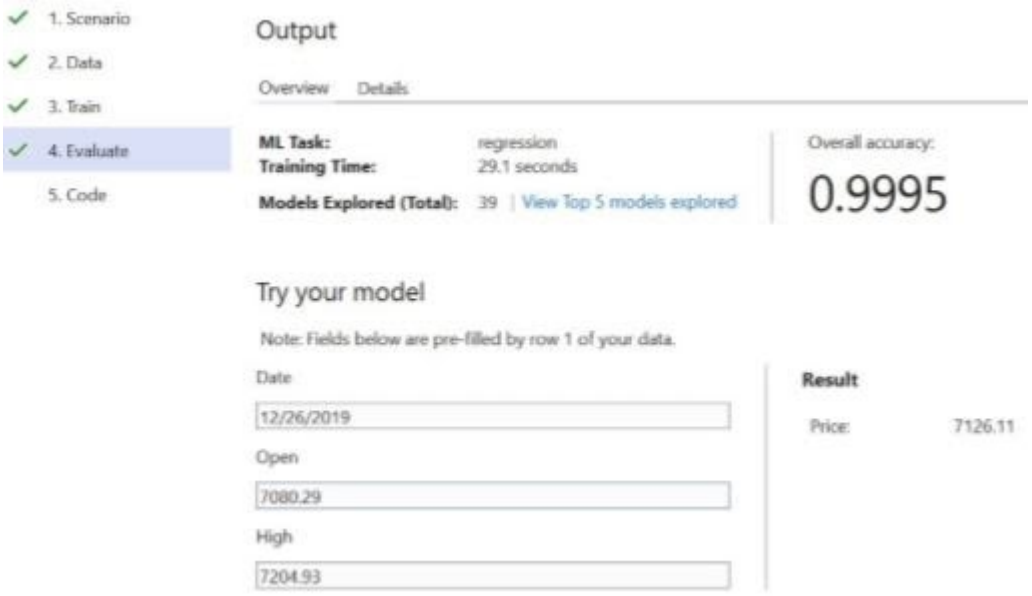


Fig 7.4 - Evaluarea modelului

e) Exportarea modelului în cod

Din modelul final se creaza 2 proiecte noi care expun o metodă unde se introduce un input și rezultă o predicție.

4. Expunerea predicției printr-un API

Odată generat algoritmul de machine learning a trebuit să-l expun printr-un API către modulul de Java. Astfel am creat un web service folosind ASP.NET framwork format din model, controler si serviciu.

API-ul are expus un singur endpoint care poate fi accesat la: /crypto/tomorrowPrice/{coinName} unde coinName este numele monedei de care utilizatorul este interesat (exp: BTC, ETH)

Răspunsul va fi de forma:

```
{
    "coinPrice": 0.0000 (float),
    "date": "MM/dd/yyy" (string),
    "currency": "USD" (string),
    "detailsAboutModel": {
        "meanAbsoluteError": 0.0000 (float),
        "meanSquaredError": 0.0000 (float),
        "rootMeanSquaredError": 0.0000 (float),
        "lossFunction": 0.0000 (float),
        "rSquared": 0.0000 (float)
    }
}
```


Fig 8.1 - Model de răspuns dat de algoritm

Un exemplu de răspuns pentru /crypto/tomorrowPrice/BTC:


```
{
    "coinPrice": 7169.864,
    "date": "12/27/2019",
    "currency": "USD",
    "detailsAboutModel": {
        "meanAbsoluteError": 5.417313291692829,
        "meanSquaredError": 970.0583125340038,
        "rootMeanSquaredError": 31.14575914204057,
        "lossFunction": 970.0583132333834,
        "rSquared": 0.9995350843534974
    }
}
```


Fig 8.2 - Exemplu de răspuns pentru moneda Bitcoin

“detailsAboutModel” reprezintă metricile algoritmului, unde, se poate observa că acuratețea acestuia este aproape de 1, avand valoare 0.9995..

5. Apelarea API -ului din Java

Pentru interacțiunea dintre modulul de Java cu API-ul creat in C# am folosit RestTemplate, cu ajutorul căruia se pot face call-uri de tip HTTP.

Astfel se efectuează un request de tip GET pentru o monedă anume:

```java
public Prediction getCoinPrediction(String cryptoCoin){

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);

    HttpEntity<Prediction> entity = new HttpEntity<>(headers);

    return restTemplate.exchange( url: url + cryptoCoin, HttpMethod.GET, entity, Prediction.class).getBody();
}
```

Fig 9.1 - Apelarea endpointului din modulul de Java

Răspunsul este modelat sub forma clasei “Prediction” având ca și variabile coinPrice, date și currency.

```java
public class Prediction {

    private Float coinPrice;
    private String date;
    private String currency;
```

Fig 9.2 - Tipul de clasă care conține răspunsul dat de algoritm

Urmând ca acest răspuns să fie atașat în email-ul care va fi trimis către utilizatorul aplicației.

```java
MailUtil.sendMail( userReadProperties.getGmailAccount(),
        appReadProperties.getApplicationGmailAccountName(),
         subject: "It's time to make a trade" ,
         message: list.get(2) + " Tomorrow, this value could be "
                + prediction.getCoinPrice()
                + " "
                + prediction.getCurrency());
```

Fig 9.3 - Atașarea răspunsului în email-ul adresat utilizatorului

Un exemplu de email primit de utilizator este acesta:

Fig 9.4 - Exemplu de email primit de un utilizator

(Sursa: https://mail.google.com/mail/u/0/#inbox)

Capitolul 3 - Deploy-ul aplicației pe un server online

Deoarece este nevoie de rularea continuă a aplicației pentru a da sugestii utilizatorilor, iar acest lucru nu se poate face rulând o mașină local, am ales să urc aplicația pe un server online care este gratuit.

Pentru modulul de Java am ales să folosesc Heroku, fiind o platformă care suportă aplicațiile scrise in Java care folosesc Selenium. Astfel, pentru a rula aplicația a fost nevoie de adăugarea a 3 pachete în platforma pusă la dispoziție de către Heroku: unul pentru Java, unul pentru a putea deschide pagina de Chrome și unul pentru a putea folosi driver-ul de Chrome.

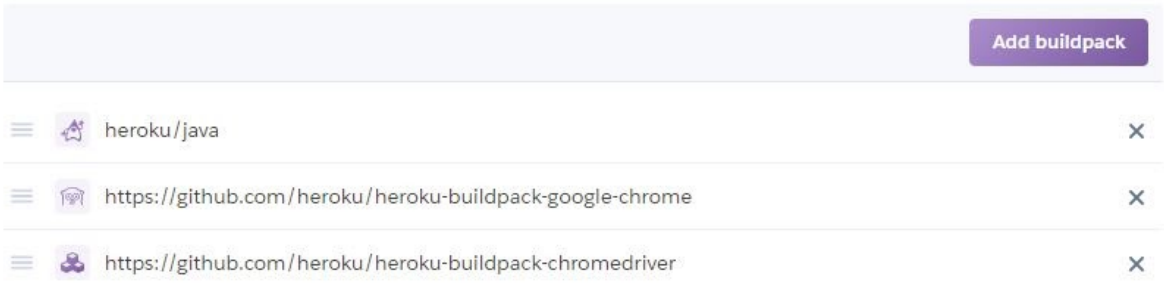


Fig 10.1 - Pachete din Heroku pentru configurarea aplicației
(Sursa: https://dashboard.heroku.com/apps/cryptomanagement/settings)

Iar, în cod, a fost nevoie de adăugarea unor opțiuni pentru driver-ul de Chrome, pentru a putea rula pe server, deoarece acesta nu poate deschide în același mod paginile web, cum se deschid pe local.

```
ChromeOptions options = new ChromeOptions();
options.addArguments("--headless");
options.addArguments("--window-size=1920,1080");
options.addArguments("--disable-gpu");
options.setExperimentalOption( name: "useAutomationExtension",  value: false);
options.addArguments("--proxy-server='direct://'");
options.addArguments("--proxy-bypass-list=*");
options.addArguments("--start-maximized");
options.addArguments("--disable-dev-shm-usage");
DesiredCapabilities SSLCertificate = DesiredCapabilities.chrome();
SSLCertificate.setCapability(CapabilityType.ACCEPT_SSL_CERTS,  value: true);
SSLCertificate.setCapability(CapabilityType.ACCEPT_INSECURE_CERTS,  value: true);
SSLCertificate.setCapability(ChromeOptions.CAPABILITY, options);
driver = new ChromeDriver(SSLCertificate);
```

Fig 10.2 - Configurări adăugate în cod pentru rularea pe server

Aplicația rulează pe server prin intermediul unui free dyno care pornește un worker. Un dyno reprezintă corul platformei Heroku, el având grijă de procese, care sunt sub denumirea de workers. Pentru ca worker-ul să știe cum să ruleze aplicația am creat un profil în maven care specifică ce clasă se vrea a fi rulată

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>

            <configuration>
                <source>8</source>
                <target>8</target>
            </configuration>
        </plugin>

        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>appassembler-maven-plugin</artifactId>
            <version>1.1.1</version>
            <configuration>
                <assembleDirectory>target</assembleDirectory>
                <programs>
                    <program>
                        <mainClass>Main</mainClass>
                        <name>worker</name>
                    </program>
                </programs>
            </configuration>
            <executions>
                <execution>
                    <phase>package</phase><goals><goal>assemble</goal></goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

Fig 10.3 - Profil Maven

Pentru trimiterea email-urilor am folosit SendGrid, un plugin compatibil cu Heroku.

Fig 10.4 - Plugin pentru trimiterea de email-uri

(Sursa: https://dashboard.heroku.com/apps/cryptomanagement)

Care oferă și o platformă web cu detaliile și stările prin care au trecut email-urile.

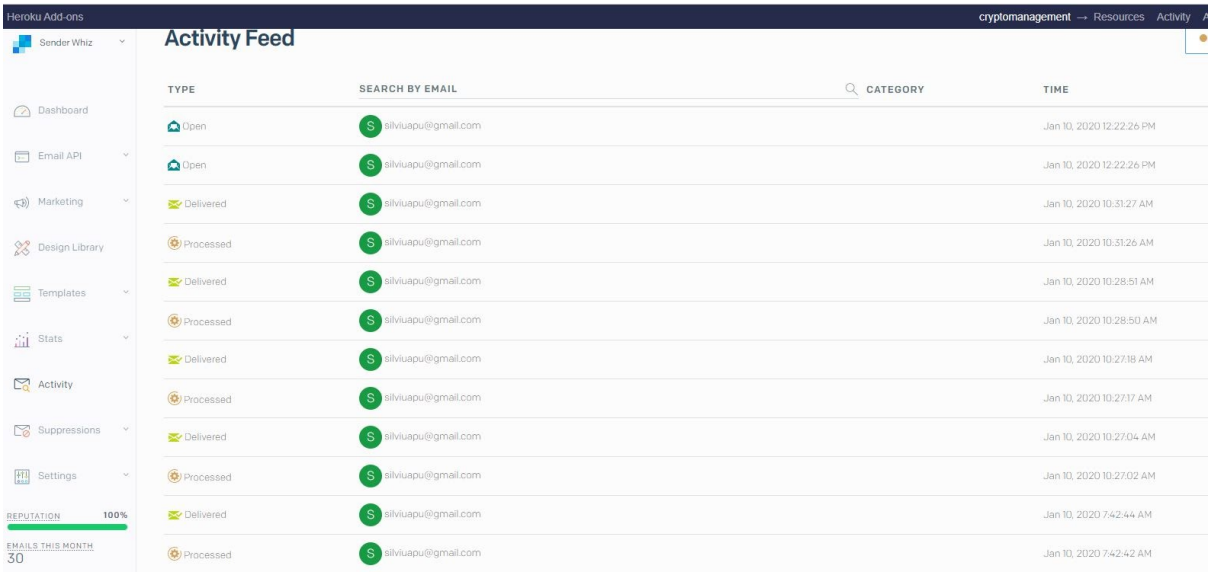


Fig 10.5 - Platformă online de la SendGrid cu activitățile aplicației

(Sursa: https://app.sendgrid.com/email_activity?)

Urmând ca în cod să adaug următoarele configurări:

```
public class MailUtil {

    public static void sendMail(String recipient, String fromAccount, String subject, String message
        Email from = new Email(fromAccount);
        Email to = new Email(recipient);
        Content content = new Content( type: "text/plain", message);
        Mail mail = new Mail(from, subject, to, content);

        SendGrid sg = new SendGrid(System.getenv( name: "SENDGRID_API_KEY"));
        Request request = new Request();
        try {
            request.setMethod(Method.POST);
            request.setEndpoint("mail/send");
            request.setBody(mail.build());
            Response response = sg.api(request);
            System.out.println("status code: " + response.getStatusCode());
            System.out.println("response body: " + response.getBody());
            System.out.println(response.getHeaders());
            System.out.println("Message sent successfully");
        } catch (IOException ex) {
            throw ex;
        }
    }
```

Fig 10.6 - Porțiune de cod care se ocupă de configurările pentru a trimite email

și un fișier Procfile care conține comanda "`worker: sh target/bin/worker`"

Platforma Heroku mai pune la dispoziție și o consolă prin intermediul căreia se pot rula diferite comenzi. Una dintre comenzile utile fiind aceea care afișează

log-urile aplicației din timpul rulării. Astfel se poate observa dacă aplicația funcționează, și dacă nu, care sunt motivele.

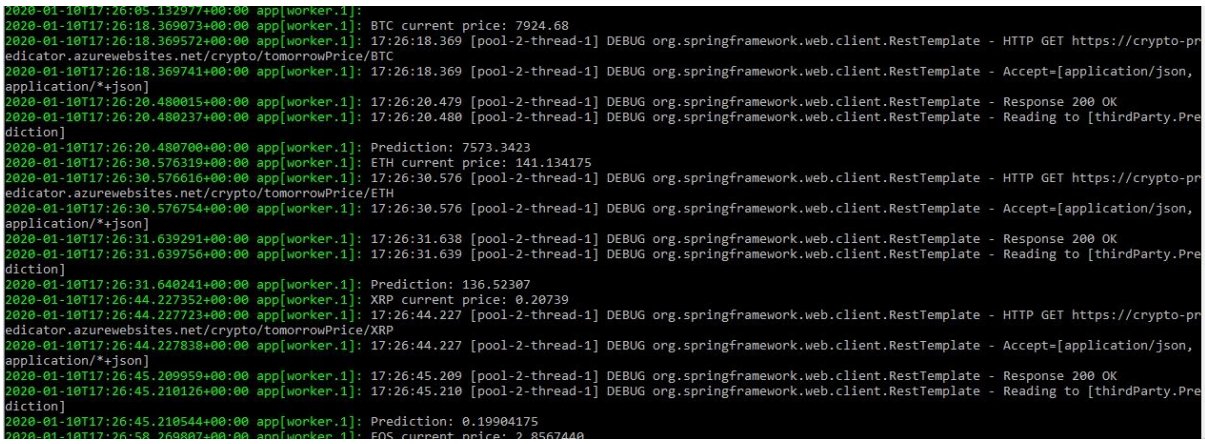


Fig 10.7 - Loggurile oferite de Heroku despre rularea aplicației

Pentru modulul de C# care conține algoritmul de machine learning am folosit, în schimb, altă platformă online, și anume Microsoft Azure, fiind mai potrivită pentru aplicațiile de tip web application scrise folosind ASP.NET. Totodată este și ușor de configurat. Pe partea de cod, singurele modificări pe care le-am adus au fost acelea de a transforma path-urile care conțineau adresa absolută a fișierelor în adresă relativă. Din varietatea de servicii pe care le oferă Azure am ales ca aplicația să fie de tip App Services, am urcat codul pe platformă și am creat o subscripție având o versiune free.

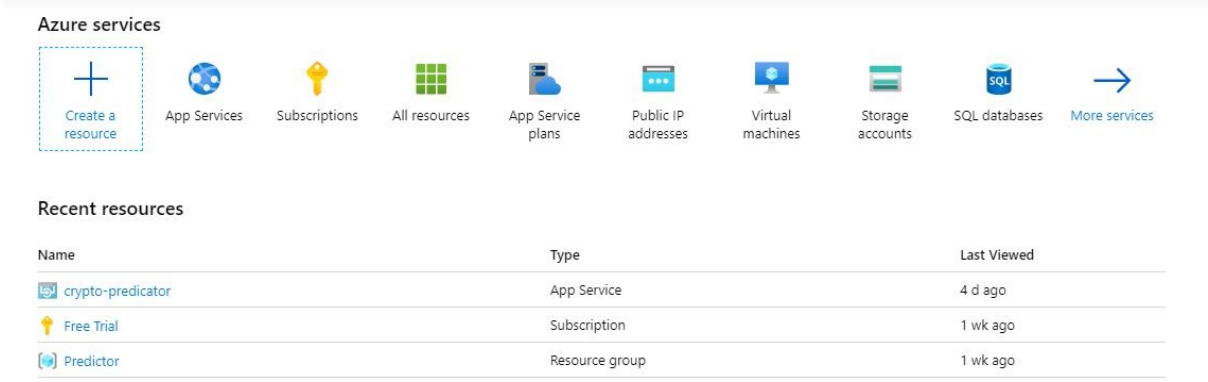


Fig 11.1 - Azure services

(Sursa: https://portal.azure.com/#home)

Și această platformă oferă o mulțime de informații, o parte din ele fiind acelea care spun dacă API-ul este up, sau, spre exemplu, de câte ori a fost apelat endpoint-ul într-un interval de timp

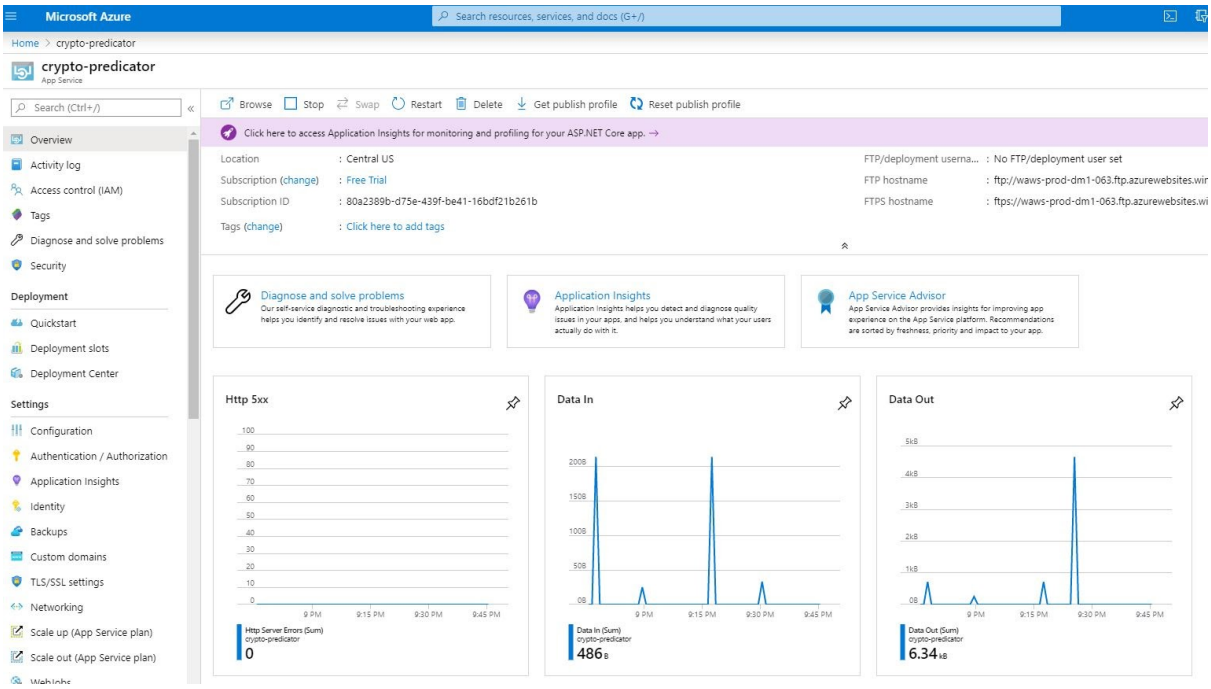


Fig 11.2 - Monitorizarea algoritmului de machine learning pe platforma Azure

(Sursa:https://portal.azure.com/#@silviualex95yahoo.onmicrosoft.com/resource/subscriptions/80a2389b-d75e-439f-be41-16bdf21b261b/resourceGroups/crypto-predicator/providers/Microsoft.Web/sites/crypto-predicator/appServices)

Enpointul aplicației poate fi accesat folosind urmatorul URL: https://crypto-predicator.azurewebsites.net/crypto/tomorrowPrice/%7BcoinName%7D , unde coinName reprezintă numele monedei pentru care se cere predicția.

Capitolul 4 - Interacțiunea utilizatorului cu aplicația

Pentru interacțiunea utilizatorului cu aplicația am creat o interfața GUI folosind JFrame din Swing.

Astfel, utilizatorul trebuie să își introducă adresa de gmail pe care vrea să primească notificările

Fig 12.1 - TextField pentru adresa de gmail

și are opțiunea de a-și alege, dintr-o listă cu cele mai populare criptomonede, pe care vrea ca aplicația să le urmărească, putând chiar să le bifeze pe toate folosind butonul de “Select all”.

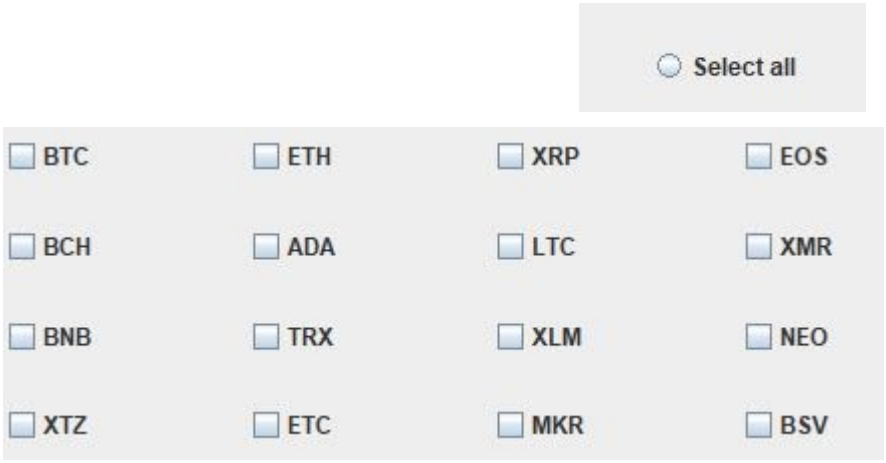


Fig 12.2 - Criptomonedele cele mai populare

În cazul în care acesta nu este interesat de nicio monedă din cele oferite ca și exemplu, sau vrea să adauge mai multe decât cele oferite, acesta poate insera, respectând aceeași denumire internațională, moneda sau monedele, separate prin virgulă, pe care acesta și le dorește în plus

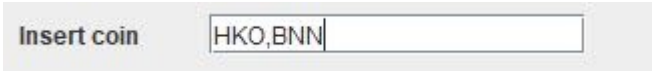


Fig 12.3 - TextField pentru alte monede

După care utilizatorul poate să facă submit la detalii apăsând butonul de “save”.

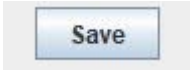


Fig 12.4 - Buton de submit

Urmând ca un pop-up cu un mesaj de confirmare să apară.

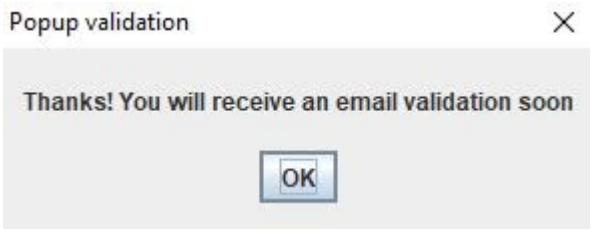


Fig 12.5 - Popup de confirmare

Așa cum scrie și în mesajul de confirmare, un email de înregistrarea va fi trimis către utilizator la adresa oferită de acesta, urmând ca singura sarcină care îi revine să fie verificarea periodică a email-urilor.

Fig 12.6 - Exemplu de email de înregistrare
(Sursa: https://mail.google.com/mail/u/0/#inbox)

Detaliile oferite de către utilizator sunt urcate ulterior pe Github, iar pe platforma Heroku se face un nou deploy cu noile modificări, astfel ca la următoarea rulare a aplicației să fie pentru noul utilizator cu noile monede pe care acesta și le dorește.

Concluziile lucrării

Aplicația CryptoManagement vine în ajutorul utilizatorilor, automatizând un proces care poate fi costisitor în privința timpului și oferă acestora o experiență plăcută, atât datorită sugestiilor pe care acesta le oferă, cât și datorită predicțiilor pentru următoarele valori ale monedei.

Aplicația înglobează funcționalități ca:

- împărțirea pe componente și decuplarea acestora (un modul în Java și unul în C#)
- proiectarea și generarea de cod pentru un algoritm de machine learning

- expunerea predicțiilor date de către algoritm printr-un API
- lucrul cu servere online
- crearea unei interfețe pentru utilizatorii acesteia.

Astfel, folosind toate aceste funcționalități, se poate urmări un flow end-to-end care pleacă de la interacțiunea utilizatorilor cu aplicația până la componente care includ servere online și 3-party-uri pentru trimiterea de emailuri.

Dezvoltări ulterioare

Ideea proiectului a fost aceea de a da sugestii cu oportunități de a face tranzacții folosind criptomonede. Ulterior acesta se poate extinde în două faze:
- una ar fi aceea de a atașa in email link către site-urile de exchange cu cele mai convenabile prețuri pentru acea monedă. Asta ar include un alt algoritm care să decidă care este site-ul cel mai potrivit dintr-o gamă cu cele mai populare
- a doua ar fi aceea de a permite aplicației să facă singură tranzacții atunci când este o oportunitate. Asta ar include în primă instanță algoritmul de decizie de la prima fază, conectarea unui utilizator la aplicație, permiterea acestui de a insera anumite credențiale de care este nevoie pentru a face tranzacțiile, printre care și portofelul online al acestuia. Asta ar crește profitul câștigat, deoarece nu se mai pierde timp în momentul în care apare o oportunitate de a face o tranzacție, astfel sarcinile utilizatorului devenind mult mai mici.

Bibliografie

- https://en.bitcoinwiki.org/wiki/BitConnect
- https://www.todaysoftmag.ro/article/2684/tranzactionarea-automata-criptomonedele-si-programatorii
- https://ro.wikipedia.org/wiki/Criptomoned%C4%83
- https://www.transfergo.com/ro/blog/ce-sunt-criptomonedele/
- https://www.antena3.ro/economic/istoria-criptomonedelor-cum-a-evoluat-ideea-de-monede-decentralizate-536901.html
- https://www.tradingview.com/
- https://www.oracle.com/ro/artificial-intelligence/what-is-machine-learning.html
- https://www.investing.com/crypto/currencies

- https://min-api.cryptocompare.com/data/v2/histoday?fsym=BTC&tsym=USD&limit=10&api_key=409b2bce1e8c9f05d32323cf70ba4f1114cfdebc9106f4d596d5c05bf34617b9
- https://dotnet.microsoft.com/learn/ml-dotnet/what-is-mldotnet
- https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet
- https://spring.io/blog/2009/03/27/rest-in-spring-3-resttemplate
- https://dashboard.heroku.com
- https://www.heroku.com/dynos
- https://azure.microsoft.com/en-us/free/search/?&ef_id=CjwKCAiA3uDwBRBFEiwA1VsajMdGscwHkYRN3LVeIOoAI6nrvGyObAAI9faTUCfiYm7wu9b0EvtwrhoCXfcQAvD_BwE:G:s&OCID=AID2000602_SEM_pgC4hutD&MarinID=pgC4hutD_363245430689_microsoft%20azure_e_c__76558203032_kwd-11177476540&lnkd=Google_Azure_Brand&dclid=CjkKEQiA3uDwBRDu7KaO-uaDqaMBEiQACSLPoBbvnJjH5yd0YYrtkFxFVdf38Cud1-Q3nDefhnAe79zw_wcB
- https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html
- https://en.wikipedia.org/wiki/Selenium
- https://economictimes.indiatimes.com/definition/selenium-web-driver
- https://en.wikipedia.org/wiki/Swing_(Java)
- http://www.ls-infomat.ro/user/content/e9efcsharp.pdf
- https://searchcloudcomputing.techtarget.com/definition/Windows-Azure

Anexe

Anexa 1 - Java

Java este un limbaj de programare orientat-obiect, puternic tipizat, conceput de către James Gosling la Sun Microsystems (acum filială Oracle) la începutul anilor ’90, fiind lansat în 1995. Cele mai multe aplicații distribuite sunt scrise în Java, iar noile evoluții tehnologice permit utilizarea sa și pe dispozitive mobile gen telefon, agendă electronică, palmtop etc. În felul acesta se creează o platformă unică, la nivelul programatorului, deasupra unui mediu eterogen extrem de diversificat. Acesta este utilizat în prezent cu succes și pentru programarea aplicațiilor destinate intraneturilor.

Limbajul împrumută o mare parte din sintaxă de la C și C++, dar are un model al obiectelor mai simplu și prezintă mai puține facilități de nivel jos. Un program Java compilat, corect scris, poate fi rulat fără modificări pe orice platformă care are instalată o mașină virtuală Java (engleză *Java Virtual Machine*, prescurtat JVM). Acest nivel de portabilitate (inexistent pentru limbaje mai vechi cum ar fi C) este posibil deoarece sursele Java sunt compilate într-un format *standard* numit cod de octeți (engleză *byte-code*) care este intermediar între codul mașină (dependent de tipul calculatorului) și codul sursă.

Anexa 2 - Selenium

Selenium este un framework pentru testarea aplicațiilor web. Acesta oferă un tool pentru automatizarea testelor fără a fi necesară învățarea limbajelor de scripting. Oferă un test domain-specific language pentru a scrie teste într-un număr mare de limbaje de programare, incluzând C#, Groovy, Java, Python, Ruby, Scala, etc. Apoi testele pot rula pe diferite browsere atât pe Windows, cât și pe Linux și macOS.

Selenium WebDriver tool este folosit pentru automatizarea testelor pe aplicații web pentru a verifica dacă acestea funcționează corect. Suportă multe tipuri de browsere, cum ar fi Chrome, Firefox, IE și Safari.

Anexa 3 - Swing

Swing este un set de instrumente pentru widget GUI pentru Java. Face parte din Oracle’s Java Foundation Class (JFC) - un API pentru furnizarea unei interfețe grafice de utilizator (GUI) pentru programele Java.

Swing a fost dezvoltat pentru a furniza un set mai sofisticat de componente GUI decât anterior Abstract Window Toolkit (AWT). Swing oferă un aspect care imită structura mai multor platforme și, de asemenea, suportă un aspect și o conectare, care permite aplicațiilor să aibă un aspect și să nu se relaționeze cu platforma de bază. Are componente mai puternice și flexibile decât AWT. În plus față de componente cunoscute, cum ar fi butoane, casete de selectare și etichete, Swing oferă mai multe componente avansate, cum ar fi panou cu file, panouri de defilare, copaci, tabele și liste.

Spre deosebire de componentele AWT, componentele Swing nu sunt implementate prin cod specific platformei. În schimb, sunt scrise în întregime în Java și, prin urmare, sunt independente de platformă. Termenul "ușor" este folosit pentru a descrie un astfel de element.

Anexa 4 - C#

Numele limbajului C# a fost inspirat din notaţia ♯ (diez) din muzică, care indică faptul că nota muzicală urmată de ♯ este mai înaltă cu un semiton. Este o similitudine cu numele limbajului C++, unde ++ reprezintă atât incrementarea unei variabile cu valoarea 1, dar şi faptul că C++ este mai mult decât limbajul C.

Limbajul C# a fost dezvoltat în cadrul Microsoft. Principalii creatori ai limbajului sunt Anders Hejlsberg, Scott Wiltamuth şi Peter Golde. Prima implementare C# larg distribuită a fost lansată de către Microsoft ca parte a iniţiativei .NET în iulie 2000. Din acel moment, se poate vorbi despre o evoluţie spectaculoasă. Mii de programatori de C, C++ şi Java, au migrat cu uşurinţă spre C#, graţie asemănării acestor limbaje, dar mai ales calităţilor noului limbaj. La acest moment, C# şi-a câştigat şi atrage în continuare numeroşi adepţi, devenind unul dintre cele mai utilizate limbaje din lume.

Creatorii C# au intenţionat să înzestreze limbajul cu mai multe facilităţi. Succesul de care se bucură în prezent, confirmă calităţile sale:

- Este un limbaj de programare simplu, modern, de utilitate generală, cu productivitate mare în programare.
- Este un limbaj orientat pe obiecte.
- Permite dezvoltarea de aplicaţii industriale robuste, durabile.
- Oferă suport complet pentru dezvoltarea de componente software, foarte necesare de pildă în medii distribuite. De altfel, se poate caracteriza C# ca fiind nu numai orientat obiect, ci şi orientat spre componente.

Anexa 5 - ML.NET

ML.NET este o platformă gratis si open source care oferă un framework de machine learning pentru developerii de pe platforma .NET

ML.NET permite antrenarea, construirea și customizarea modelelor machine learning folosind C# sau F# pentru o varietate de scenarii ML. Acesta include funcționalități cum ar fi automated machine learning (AutoML) si tool-uri precum ML.NET CLI și ML.NET Model Builder, care fac integrarea dintre machine learning și aplicații mult mai simplă.

Anexa 6 - ASP.NET

.NET este o platformă pentru developeri, care oferă tool-uri, limbaje de programare și librării pentru crearea diferitelor tipuri de aplicații. Platforma de bază oferă componente care se aplică pe toate tipurile de aplicații. Framework-uri adiționale, ca ASP.NET extind .NET care includ componente pentru construirea de aplicații specifice.

Platforma .NET include:

- Limbajul de programare C# și compilările acestuia
- Librării de bază pentru lucrul cu stringuri, date, fișiere și multe altele
- Tool-uri și editoare pentru Windows, Linux, macOS și Doker

ASP.NET extinde platforma .NET cu tool-uri și librării specifice pentru construirea aplicațiilor web.

ASP.NET include:

- Framework de bază pentru procesarea requesturilor web
- Web Page templating syntax, cunoscut ca și Razor, pentru construirea dinamică a paginilor web folosind C#
- Librării comune pentru patternurile web, cum ar fi Model View Controller (MVC)
- Sistem de autentificare care include librării, baze de date, și page template-uri, incluzând multi-factor authentication și external authentication cu Google, Twitter și altele
- Extensii de editoare care oferă syntax highlighting, code completion, și alte funcționalități specifice dezvoltării paginilor web

Anexa 7 - Heroku

Heroku este o platformă Cloud bazată pe containere ca serviciu (PaaS). Dezvoltatorii folosesc Heroku pentru a implementa, gestiona și scala aplicații moderne. Platforma este elegantă, flexibilă și ușor de utilizat, oferind dezvoltatorilor calea cea mai simplă de a scoate aplicațiile pe piață.

Heroku este complet gestionat, oferind dezvoltatorilor libertatea de a se concentra asupra produsului lor principal, fără distragerea întreținerii serverelor, hardware-ului sau infrastructurii. Experiența Heroku oferă servicii, instrumente, fluxuri de lucru și suport poliglot - toate concepute pentru a spori productivitatea dezvoltatorilor.

Heroku lucrează cu o mare varietate de clienți și parteneri.

Anexa 8 - SendGrid

SendGrid este un furnizor SMTP bazat pe Cloud care permite trimiterea de e-mailuri fără a fi necesară menținerea unor servere de e-mail. SendGrid gestionează toate detaliile tehnice, de la extinderea infrastructurii până la informarea ISP și monitorizarea reputației până la lista albă de servicii și analize în timp real.

SendGrid oferă două modalități de a trimite email: prin intermediul releului de SMTP sau prin API-ul Web. SendGrid oferă biblioteci pentru clienți în multe limbi. Aceasta este modalitatea preferată de integrare cu SendGrid. Dacă se folosește SendGrid fără o bibliotecă client, API-ul Web este recomandat, în majoritatea cazurilor, deoarece este mai rapid, oferă o serie de codificări și are tendința de a fi mai ușor de utilizat. SMTP oferă în mod implicit multe funcții, dar este mai greu de configurat.

Anexa 9 - Microsoft Azure

Microsoft Azure, cunoscută anterior drept Windows Azure, este platforma publică de Cloud computing Microsoft. Oferă o serie de servicii Cloud, inclusiv cele pentru calcul, analitică, stocare și rețea. Utilizatorii pot alege dintre aceste servicii pentru a dezvolta și scala noi aplicații sau pentru a rula aplicații existente în Cloud public.

Microsoft Azure este considerat pe scară largă atât o platformă ca un serviciu (PaaS) cât și o infrastructură ca un serviciu (IaaS).