Elementi finansijske matematike

PREDVIĐANJE CENA AKCIJA POMOĆU MAŠINSKOG UČENJA

Đorđe Martić, Jovan Samardžić

Matematički fakultet, Univerzitet u Beogradu

Maj 2023.

Sadržaj

1 Uvod

Akcije (eng. *shares, stocks*) [1] su vrednosni papiri, tj. finansijski instrumenti, koji predstavljaju udeo vlasništva neke kompanije. Drugim rečima, vlasnici akcija su u odgovarajućem procentu i vlasnici kompanije.

Oni mogu da ostvaruju prihode na dva načina. Jedan način jeste da zadrže akciju i čekaju da im se povremeno isplaćuju **dividente** (odgovarajući procenat profita koji je firma zaradila). Drugi način jeste da tim akcijama **trguju na berzi**. Svaka akcija ima svoju vrednost i prihod se pravi na promenama te vrednosti. Zbog toga, taj prihod koji trgovina akcijama donosi nije fiksan. Šta više, nije ni siguran, jer ukoliko dođe do greške u predikciji cene, može doći do značajnih gubitaka.

Trgovina akcijama može se izvršiti na dva načina – zauzimanjem jedne od dve pozicije. Ukoliko trgovac veruje da će cena akcije skočiti u budućnosti, onda on zauzima **long poziciju**, tj. kupuje te akcije, a kasnije ih prodaje po višoj ceni. Ukoliko trgovac veruje da će cena akcije pasti u budućnosti, onda on zauzima **short poziciju**, tj. on pozajmi akciju, proda je, a zatim sačeka da cena akcije zaista padne, kako bi je kupio i vratio originalnom vlasniku.

Zarada pomoću akcija može se praviti i kupovinom ili prodajom njihovih finansijskih derivata, npr. **opcija** (eng. *options*).

Dobri trgovci su oni koji najbolje mogu da predvide cenu akcije. Ranije, predviđanja su se u velikoj meri oslanjala na ljudske instinkte i procene. Primera radi, često se koristila tzv. **fundamentalna analiza** [2], u kojoj se odluke donose tako što se prave detaljni izveštaji o poslovanju odgovarajuće kompanije (npr. analiziraju se izjave za javnost koje daju predstavnici kompanije).

Sledeći korak je bila pojava **algoritamskog trgovanja** (eng. *quant trading*) [3], u kojem trgovci zadaju strategiju računaru koji trguje umesto njih. Primeri ovakvih strategija su *trend following*, *mean reversion*, *pairs trading* i mnoge druge.

Ipak, zbog razvoja računara, danas se predviđanja najčešće obavljaju korišćenjem **mašinskog učenja**, koji pruža neograničene mogućnosti. Na ovaj način izvodljivo je učitati ogroman broj podataka i lakše uočavati skrivene paterne.

Naš cilj u ovom radu biće upravo da predstavimo neke od modela za cene akcija koji se dobijaju korišćenjem tehnikama mašinskog učenja, kao i da prikažemo i uporedimo rezultate koje ti modeli daju.

2 Istorijski podaci

Trgovci zarađuju tako što procenjuju kolika će vrednost akcije biti u budućnosti u odnosu na trenutnu vrednost akcije. Oni se služe vrednostima akcija iz prošlosti – predpostavljaju da će se kretanja akcija iz prošlosti na sličan način odvijati i u budućnosti. Na kraju svakog dana, cene akcija se zapisuju i ostaju trajno zapamćene kao **istorijski podaci**.

Istorijski podaci se najčešće izlažu u formi tabele (slika 1) u kojoj svaki red predstavlja jedan dan. Pošto se cene akcija menjaju i u toku dana, u tabele se upisuje barem 5 različitih cena akcija. To su:

- **open** – cena na početku dana
- **close** – cena na kraju dana
- **low** – najniža cena u toku dana
- **high** – najviša cena u toku dana

Peta cena koja se beleži je tzv. **adjusted close** cena, koja je ista kao i obična *close* cena, osim kada dođe do tzv. *splitovanja*. To znači da se svakom trgovcu udvostruči broj akcija koje poseduje, ali se njihova vrednost prepolovi.

Još jedna vrednost koja se zapisuje jeste **volume** – to je broj transakcija u toku jednog dana, odnosno koliko puta je akcija odgovarajuće kompanije prodavana ili kupovana u toku jednog dana.

AAPL

Date	Open	High	Low	Close	Adj Close	Volume
2023-01-03	130.279999	130.899994	124.169998	125.070000	124.879326	112117500
2023-01-04	126.889999	128.660004	125.080002	126.360001	126.167366	89113600
2023-01-05	127.129997	127.769997	124.760002	125.019997	124.829399	80962700
2023-01-06	126.010002	130.289993	124.889999	129.619995	129.422394	87754700
2023-01-09	130.470001	133.410004	129.889999	130.149994	129.951584	70790800
2023-01-10	130.259995	131.259995	128.119995	130.729996	130.530701	63896200
2023-01-11	131.250000	133.509995	130.460007	133.490005	133.286499	69458900
2023-01-12	133.880005	134.259995	131.440002	133.410004	133.206619	71379600
2023-01-13	132.029999	134.919998	131.660004	134.759995	134.554550	57809700
2023-01-17	134.830002	137.289993	134.130005	135.940002	135.732758	63646600
2023-01-18	136.820007	138.610001	135.029999	135.210007	135.003876	69672800

Slika 1: Istorijski podaci za kompaniju Apple iz 2005. godine

Za potrebe našeg rada, odlučili smo se da koristimo podatke kompanije *Apple*. Podatke smo besplatno preuzeli sa sajta `Yahoo Finance` [4].

Na slici 2 možemo videti kretanje *close* cene tih akcija.

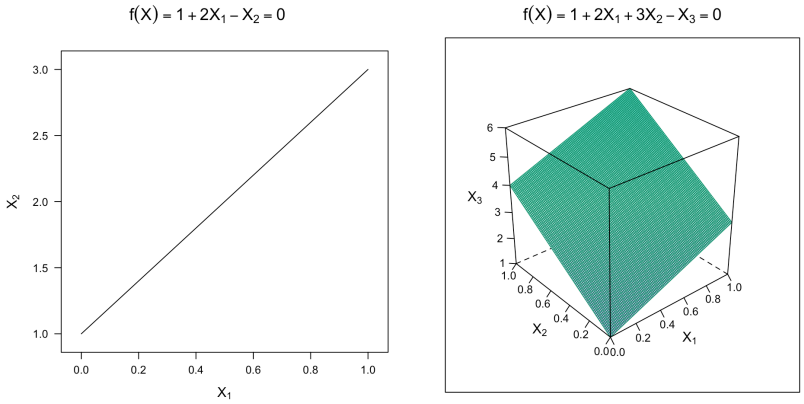
Slika 2: Close cene Apple akcija

Kako bi naši modeli bili bolji, odlučili smo se da koristimo samo podatke počevši od 2005, pošto je otprilike tada cena počela značajnije da se menja.

3 Support Vector Machine

3.1 Hiperravni

Kako bismo razumeli metod potpornih vektora (eng. *support vector machine*), prvo se podsetimo pojma **hiperravni** [5]. U n-dimenzionom vektorskom prostoru, hiperravan je njegov $(n-1)$-dimenzioni potprostor. Na primer, hiperravan euklidske ravni je prava, dok je hiperravan euklidskog prostora ravan.



Slika 3: Primeri hiperravni u 2 i 3 dimenzionom euklidskom prostoru

3.2 Support Vector Classifier

Prvo ćemo pogledati kako funkcioniše klasifikacija metodom potpornih vektora. Postmatrajmo skup tačaka u ravni koje su podeljene u dve klase tako da postoji prava koja razdvaja tačke jedne klase od tačaka druge klase. Primetimo da ova prava gotovo nikada nije jedinstvena, pa mi želimo da nađemo pravu tako da ima najveće moguće rastojanje do najbliže tačke. Te najbliže tačke nazivamo **potpornim vektorima**.

Jednačina ove optimalne hiperravni je

$$\omega \cdot x + \omega_0 = 0,$$

gde je $x, \omega \in \mathbb{R}^n$ i $\omega_0 \in \mathbb{R}$ slobodan član.

Hiperravni paralelne optimalnoj koje sadrže potporne vektore imaju jednačine imaju jednačine:

$$\omega \cdot x + \omega_0 = c,$$

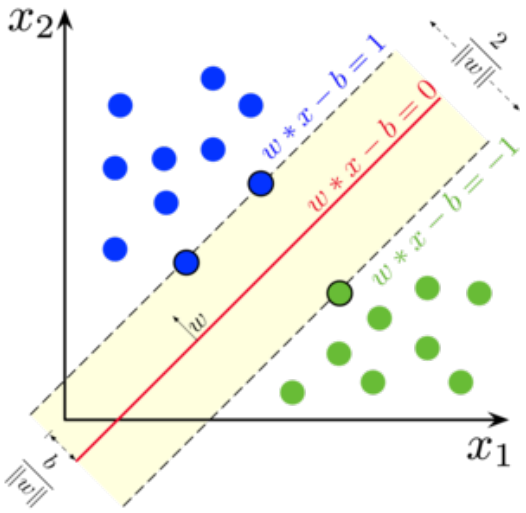
$$\omega \cdot x + \omega_0 = -c$$

Deljenjem ovih jednačina sa c dobijamo nove jednačine:

$$\omega \cdot x + b = 0,$$

$$\omega \cdot x + b = 1,$$

$$\omega \cdot x + b = -1.$$



Slika 4: Optimalna ravan i njoj paralelne hiperravni na potpornim vektorima

Naš zadatak je da nađemo optimalnu ravan tako da je njeno rastojanje od potpornih vektora najveće moguće. Drugim rečima, tražimo:

$$\max_{\omega} \frac{|\omega \cdot x + b|}{\|\omega\|_2}.$$

Primetimo da je $|\omega \cdot x + b| = 1$ za svako x na paralelnoj pravi koja sadrži potporni vektor, pa se naš problem svodi na pronalaženje

$$\min_{\omega,\omega_0} \frac{\|\omega\|_2^2}{2}$$

pod uslovom

$$y_i(\omega \cdot x_i + b) \geq 1 \quad i = 1, ..., N,$$

gde su x_i za $i \in \{1, 2, .., N\}$, tačke na osnovu kojih pravimo model, a $y_i \in \{-1, 1\}$.

Kako je često slučaj da u podacima možemo naći autlajere koji onemogućavaju pronalaženje poptpornog vektora pa možemo koristiti i takozvani **metod potpornih vektora sa mekim pojasom**, a problem koji rešavamo u ovom slučaju je:

$$\min_{\omega,\omega_0} \frac{\|\omega\|_2^2}{2} + c\sum_{i=1}^{N} \epsilon_i$$
$$y_i(\omega \cdot x_i + \omega) \geq 1 - \epsilon_i$$
$$\epsilon_i \geq 0 \quad i = 1, 2, ..., N.$$

Ovi problemi minimizacije se rešavaju pomoću Lagranžovih množioca i numeričkih metoda. Ovome nećemo detaljno posvećivati pažnju ali je važno napomenuti da minimalni vetor ω zadovoljava:

$$\omega = \sum_{i=1}^{N} \alpha_i y_i x_i$$

i time u stvari tražimo:

$$\max_{\alpha_1,..,\alpha_N} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) + \sum_{i=1}^{N} \alpha_i$$
$$\alpha_i \geq 0$$
$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

3.3 Support Vector Regressor

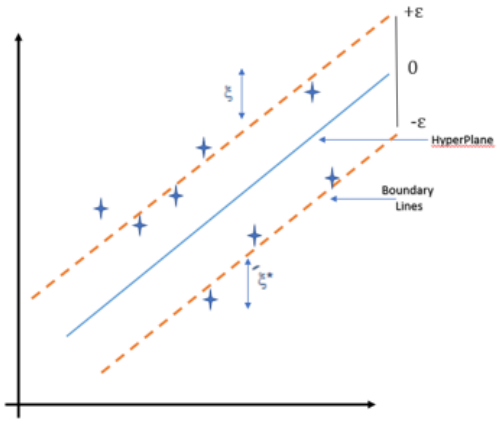
U slučaju regresije metodom potpornih vektora, koristimo istu ideju sa hiperravni, samo sada tražimo takvu hiperravan da zadovoljava:

$$\min_{\omega} \|\omega\|_2^2$$

$$|\omega \cdot x + b - y| = 0.$$

Naravno, ovo je nemoguće osim ukoliko su sve tačke u jednoj hiperravni, što nam nije interesantan slučaj, tako da ćemo oslabiti uslov minimizacije

$$|\omega \cdot x + b - y| < \epsilon.$$



Slika 5: Optimalna ravan i njoj paralelne hiperravni na potpornim vektorima

U ovom slučaju zahtevamo da sve tačke budu na rastojanju najviše od optimalne ravni, što je retko zadovoljeno, pa ćemo kao u slučaju SVM klasifikacije sa mekim pojasom, oslabiti uslov blizine tačaka od optimalne hiperravni.

$$\min_{\omega,\omega_0} \|\omega\|_2^2 + c\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

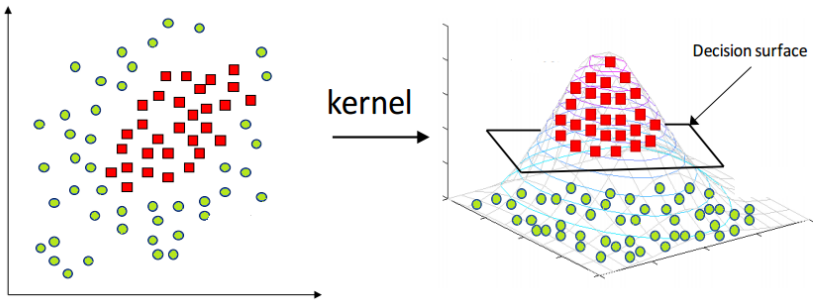
$$\omega \cdot x_i - y < \epsilon + \xi_i \quad i = 1, 2, ..., N.$$

$$y - \omega \cdot x_i < \epsilon + \xi_i^* \quad i = 1, 2, ..., N.$$

Do sada nismo razmotrili značenje konstante c u problemu minimizacije. Nju određujemo samo u zavisnosti od toga kakav model želimo, sa većim vrednostima c dobijamo bolje uklapanje trening podacima ali i mogućnost overfitovanja, dok premale vrednosti c ne dovode do dovoljno dobrog uklapanja podataka da bi model bio koristan.

3.4 Kerneli i SVM

Nekada, nije moguće pronaći odgovarajuću hiperravan tako da zadovolji uslove koje smo postavili. Tada koristimo funkcije koje pretprocesiraju podatke u više dimenzione tako da postoji zadovoljavajuća razdvajajuća hiperravan, kao na primeri koji možemo videti na slici ispod.



Slika 6: Transformacija podataka u više dimenzija

Tada naš problem postaje traženje:

$$\max_{\alpha_1,..,\alpha_N} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (\Phi(x_i)\cdot\Phi(x_j)) + \sum_{i=1}^{N}\alpha_i$$

$$\alpha_i \geq 0$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0.$$

Neka je X neprazan skup i neka je data simetrična funkcija $k : X \times X \to R$. Ako je za svako $n \in N$ i svako $x_1, , , x_n \in X$ matrica dimenzija $n \times n$ sa elementima $k(x_i, x_j)$ pozitivno semidefinitna, funkcija k je pozitivno semidefinitan **kernel**

ili **Mercerov kernel**. Važan podatak je da za svaki kernel k postoji preslikavanje Φ_k iz X u neki vektorski prostor sa skalarnim proizvodom, tako da važi

$$k(x, y) = \Phi_k(x) \cdot \Phi_k(y),$$

odnosno kernel se može posmatrati kao skalarni proizvod u nekom vektorskom prostoru.

Kerneli su nam bolji za korišćenje od funkcija za pretprocesiranje jer dosta često smanjuju kompleksnost računanja što dosta ubrzava algoritam.

Neki od najpopularnijih kernela su:

- **Gausov kernel** $k(x, y) = exp(-\frac{\|x-y\|^2}{\gamma})$
- **Polinomijalni kernel** $k(x, y) = (x \cdot y + c)^d$
- **Linearni kernel** $k(x, y) = x \cdot y$

3.5 Implementacija i rezultati

Model ćemo implementirati u Python-u, koristeći paket `scikitk-learn`.

Prvo učitavamo sve potrebne biblioteke.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import datetime as dt

from sklearn.svm import SVR
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
```

Nakon toga učitavamo podatke vrednosti AAPL deonica od 2005. godine i proveravamo da li ima NA vrednosti i podešavamo tip kolone *Date* na `datetime`.

```
df = pd.read_csv('efm_Projekat/AAPL.csv', sep = ',')
df.head()
df.isnull().sum()
df['Date'] = pd.to_datetime(df['Date'])
```

Sada možemo pripremiti podatke za primenu SVM tako što ćemo ih skalirati i podeliti na trening i test skup.

```
close = df_close['Close']
scaler = MinMaxScaler(feature_range=(0,1))
closedf=scaler.fit_transform(np.array(close).reshape(-1,1))

training_size=int(len(closedf)*0.90)
test_size=len(closedf)-training_size
train_data,test_data=closedf[:training_size,:],closedf[
    training_size:,:1]
```

Nakon toga pravimo funkciju koja nam omogućava da primenimo algoritam na njih

```
def create_dataset(dataset, time_step=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99
       100
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return np.array(dataX), np.array(dataY)
```

Koristili smo vremenski korak od 15 dana.
Da bismo odredili optimalne hiperparametre, koristimo grid search.

```
param_grid = {'C':[0.01,1,10,50,100],
              'kernel':['linear','rbf','poly'],
               'gamma':['scale','auto'],
               'degree':[2,3,4],
               'epsilon':[0.01,0.1,0.5,1]}

svr = SVR()
grid = GridSearchCV(svr,param_grid=param_grid)

grid.fit(X_train,y_train)
grid.best_params_
```

Dobili smo da su optimalni hiperparametri linearni kernel, $C = 50, epsilon = 0.001, degree = 2$ i gamma = 'scale'.

Pravimo SVR model sa ovim parametrima i vršimo predikciju na test skupu.

```
svr_best_model = SVR(kernel = 'linear', C = 50, epsilon = 0.001,
    gamma = 'scale', degree = 2)
svr_best_model
svr_best_model.fit(X_train, y_train)

train_predict=svr_best_model.predict(X_train)
test_predict=svr_best_model.predict(X_test)

train_predict = train_predict.reshape(-1,1)
test_predict = test_predict.reshape(-1,1)
```

Nakon ovoga moramo vratiti podatke u prvobitni oblik, nakon čega crtamo možemo izračunati MSE i RSME na trening i test skupu i grafčki uporediti dobijene vrednosti.

```
train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)
original_ytrain = scaler.inverse_transform(y_train.reshape(-1,1)
    )
original_ytest = scaler.inverse_transform(y_test.reshape(-1,1))

print("Train  RMSE: ", math.sqrt(mean_squared_error(
    original_ytrain,train_predict)))
print("Train  MSE: ", mean_squared_error(original_ytrain,
    train_predict))

print("Test  RMSE: ", math.sqrt(mean_squared_error(
    original_ytest,test_predict)))
print("Test  MSE: ", mean_squared_error(original_ytest,
    test_predict))
```

Dobijamo vrednosti MSE (RMSE) 0.83 (0.69) na trening skupu, odnosno 8.76 (2.96) na test skupu.

Na grafiku ispod možemo videti predviđene vrednosti na test skupu obeležene plavom bojom, dok su stvarne vrednosti obeležene crvenom bojom.

```
plt.figure(figsize = (10,6))
plt.grid(True)
plt.xlabel = 'Date'
plt.ylabel = 'AAPL Close price'
plt.plot(original_ytest, c = 'red')
plt.plot(test_predict, c = 'blue')
plt.show()
```

Slika 7: Predviđene i stvarne cene na test skupu

4 Long Short Term Memory

4.1 Rekurentne neuralne mreže

Ukoliko želimo da modelujemo podatke koji predstavljaju nizove, poput prirodnog jezika ili vremenskih serija, koriste su **rekurentne neuronske mreže** (eng. *recurrent neural network* ili skraćeno *RNN*) [6].

Glavna ideja koja stoji iza ove vrste neuronskih mreža jeste da se modeluju zavisnosti između instanci. U našem slučaju, vrednost neke vremenske serije u ovom trenutku zavisi od njenih vrednosti u prošlosti, pa upravo zato RNN predstavlja dobru osnovu za njihovo modelovanje.

Dužina vremenske serije nije unapred poznata. Za jedan ulaz može biti jedna, a za drugi ulaz može biti druga dužina serije. Mi očekujemo da naš model pravi predikcije u oba slučaja. Ovaj problem se rešava tako što se elementi serije obrađuju u koracima. Mreža ima *skriveno stanje* (eng. *hidden state*) koje čuva informaciju o elementima serije koji su već obrađeni. To stanje se menja iz koraka u korak, a način na koji se ono menja određuju upravo parametri modela.

Formalno [7], ovaj model možemo zapisati na sledeći način:

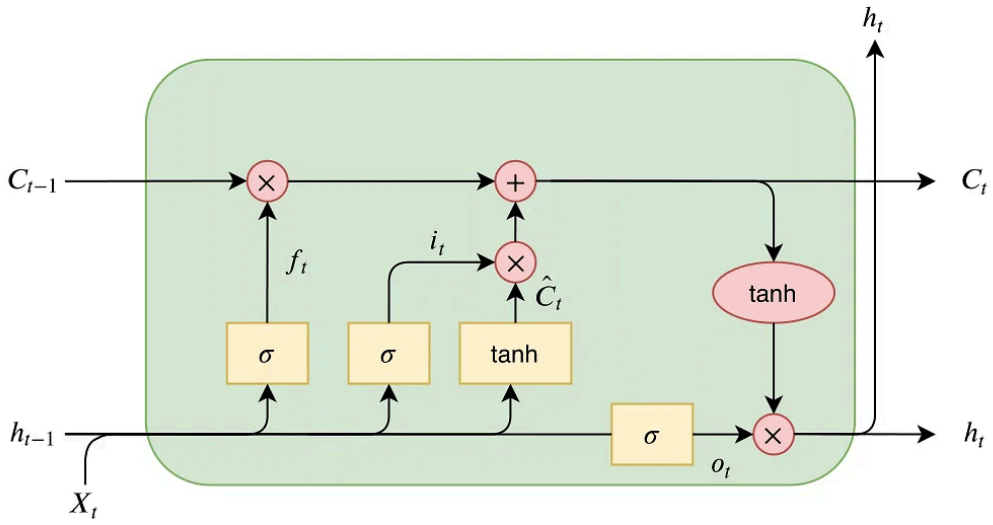
$$\begin{aligned} h^{(0)} &= 0 \\ h^{(t)} &= g\left(b_h + W \cdot h^{(t-1)} + U \cdot x^{(t)}\right) \\ o^{(t)} &= g\left(b_0 + V \cdot h^{(t)}\right), \end{aligned}$$

za $t = 1, 2, ..., T$, gde je T dužina serije, vektor $x(t)$ predstavlja ulaz u trenutku t, $h(t)$ predstavlja vektor vrednosti skrivenog sloja, g je aktivaciona funkcija, a $o(t)$ je vektor izlaza mreže (predviđanje prave vrednosti $y(t)$).

Parametri opšteg modela su: matrica transformacije stanja u stanje W, matrica ulaza u stanje U , matrica stanja u izlaz V i vektori slobodnih članova b_h i b_0.

4.2 Opis LSTM modela

Long Short Term Memory, skraćeno **LSTM**, [8] je tip rekurentne neuralne mreže koja se često koristi u predviđanju cena akcija. Razlog za to je jer je pogodna za uočavanje dugoročnih zavisnosti u podacima. To radi pomoću niza *ćelija*, koje se čuvaju informacije o tim zavisnostima. One po potrebi čuvaju i prosleđuju te informacije, ali istovremeno zaboravljaju i ignorišu podatke koji nisu relevantni.



Slika 8: Arhitektura LSTM modela

Kako bismo bolje opisali ovaj model, opisaćemo njegovu arhitekturu i način na koji radi [9]. Pre toga, još jednom ćemo da istaknemo ključne pojmove i pojmove koje idu uz njih:

- X_t = **ulazni podaci** = ulaz u trenutnom koraku
- C_t = **ćelija** = dugoročna memorija u trenutnom koraku
- h_t = **skriveno stanje** = izlaz iz trenutnog koraka

Dakle, kada čitamo neku od ovih reči, cilj je da u glavi istovremeno imamo i pojam na koji se odnosi. Sada možemo preći na konkretniji deo.

Naime, osnovni LSTM model prolazi kroz više faza u kojima su redom aktivne sledeće jedinice:

- **Kapija za zaboravljanje** (eng. *forget gate*) – odlučuje koji elementi ćelije su i dalje relevantni
- **Ulazna kapija** (eng. *input gate*) i **nova memorijska mreža** (eng. *new memory network*) – kontrolišu koje informacije treba uneti u ćeliju;
- **Izlazna kapija** (eng. *output gate*) – kontroliše koje informacije se koriste za pravljenje prognoza.

Svaka od ovih jedinica predstavlja zasebnu standardnu rekurentnu mrežu i na slici 8 je predstavljena žutom bojom. Operacije predstavljene crvenom bojom.

1. U prvoj fazi, kapija za zaboravljanje kao ulaz uzima prethodno skriveno stanje h_{t-1} i trenutne ulazne podatke X_t i na osnovu njih računa vektor f_t koji je određen sledećom formulom:

   $$f_t = \sigma\left(W_f \cdot [h_{t-1}, X_t] + b_f\right),$$

   gde je σ sigmoidna aktivaciona funkcija (podsetimo se, ona vraća), W_f je matrica težina (kao *weights*), a b_f vektor pristrasnosti (kao *bias*). Ova matrica i ovaj vektor uče, tj. promenljivi su u svakom koraku, u zavisnosti od tipa optimizacije koji smo izabrali. Ove oznake će biti analogne i u sledećim fazama.

   Kao što znamo, sigmoidna funkcija vraća vrednosti iz segmenta $[0, 1]$, tako da vektor f_t uzima vrednosti upravo odatle. Vrednosti bliske nuli nam nagoveštavaju da odgovarajući element u ćeliji nema veliki značaj.

   Zato je naš sledeći korak da izračunamo $f_t \cdot C_{t-1}$, tj. skalarni proizvod prethodnog stanja ćelije i dobijenog vektora. Taj rezultat prosleđujemo u drugu fazu.

2. U drugoj fazi određujemo koje nove informacije treba upisati u ćeliju, tj. u dugoročnu memoriju.

   Ulazna kapija radi sličan posao kao kapija za zaboravljanje, samo u ovom slučaju ona određuje *koje* vrednosti ćelije treba ažurirati. To se opisuje sledećim izrazom:

   $$f_i = \sigma\left(W_i \cdot [h_{t-1}, X_t] + b_i\right).$$

   Nova memorijska mreža određuje *količinu* u kojoj se te vrednosti treba ažurirati:

   $$\hat{C}_t = tanh\left(W_C \cdot [h_{t-1}, X_t] + b_C\right),$$

   gde je $tanh$ hiperbolički tangens – ta funkcija uzima vrednosti u segmentu $[-1, 1]$.

   Konačno, računaju se nove vrednosti koje se upisuju u dugoročnu memoriju, tj. ćeliju:

   $$C_t = i_t \cdot \hat{C}_t + f_t \cdot C_{t-1}$$

3. Ostalo je još da ažuriramo skriveno stanje. Prvo računamo vektor koji nam govori koji elementi iz nove ćelije su bitni:

   $$o_t = \sigma\left(W_o \cdot [h_{t-1}, X_t] + b_o\right).$$

   Konačno, na osnovu svega prethodnog, dobijamo izlaz iz ovog koraka, tj. vrednost koja nam služi za predikciju.:

   $$h_t = o_t \times tanh(C_t)$$

Ovim smo opisali samo jedan korak našeg modela. Ovaj korak se ponavlja odgovarajući broj puta, sve dok se ne dobije konačni izlaz u skrivenom stanju, koji će nam služiti kao vrednost predikcije.

4.3 Implementacija i rezultati

Model ćemo implementirati u Python-u, koristeći paket `keras`, koji je deo TensorFlow-a. Program smo pokretali na računaru *MacBook Pro 2017*, koji koristi *2.3GHz dual-core Intel Core i5* procesor.

Prvo učitavamo sve potrebne biblioteke. To su uglavnom poznate biblioteke koje se često koriste, tako da se nećemo zadržavati na detaljnom opisivanju svake od njih.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

Zatim učitavamo *AAPL* podatke koje smo opisali ranije. Ovom prilikom ih skaliramo na segment $[-1, 1]$, jer je opšte poznato da neuralne mreže daju bolje rezultate kada se izvrši skaliranje u fazi pretprocesiranja podataka. Ono može poboljšati konvergenciju algoritma optimizacije i ubrzati proces učenja.

```
df = pd.read_csv('AAPL.csv')
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(df['Close'].values.reshape
    (-1, 1))
```

Postavljamo vrednosti hiperparametara. Jasno je šta `epoch` i `batch_size` predstavljaju, dok je `prediction_days` broj dana za koje pravimo predikciju (tj. veličina test skupa).

```
prediction_days = 60
epochs = 100
batch_size = 64
```

Iz skaliranih podataka izdvajamo trening i test podatke, koristeći prethodno zadat hiperparametar `prediction_days`.

```
training_data = scaled_data[:len(scaled_data)-prediction_days]
testing_data = scaled_data[len(scaled_data)-prediction_days:]
```

Model koji pravimo je sekvencijalan, tj. sastoji se od više LSTM modela. Svaki od njih ima 50 ćelija, što znači da se opisani postupak ponavlja 50 puta. Takođe, nakon svakog sloja dodali smo tzv. *dropout* sa parametrom 0.2. To je metoda regularizacije koja za cilj ima sprečavanje preprilagođavanje modela trening podacima. U našem slučaju, nakon izvršavanja svakog LSTM sloja, na slučajan način se izbaci 20% dostupnih podataka.

```
model = Sequential()

model.add(LSTM(units=50, return_sequences=True, input_shape=(
    training_data.shape[1], 1)))
model.add(Dropout(0.2))

model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=50))
model.add(Dropout(0.2))

model.add(Dense(units=1))
```

Konačno, model možemo da kompajliramo – tj. pripremamo ga za traniranje tako što mu zadajemo optimizator adam (o kom smo više govorili na drugim kursevima), a osim toga zadajemo i srednjekvadratnu grešku kao funkciju gubitaka.

Nakon toga, model treniramo na ranije izdvojenim trening podacima i koristimo ranije zadate hiperparametre.

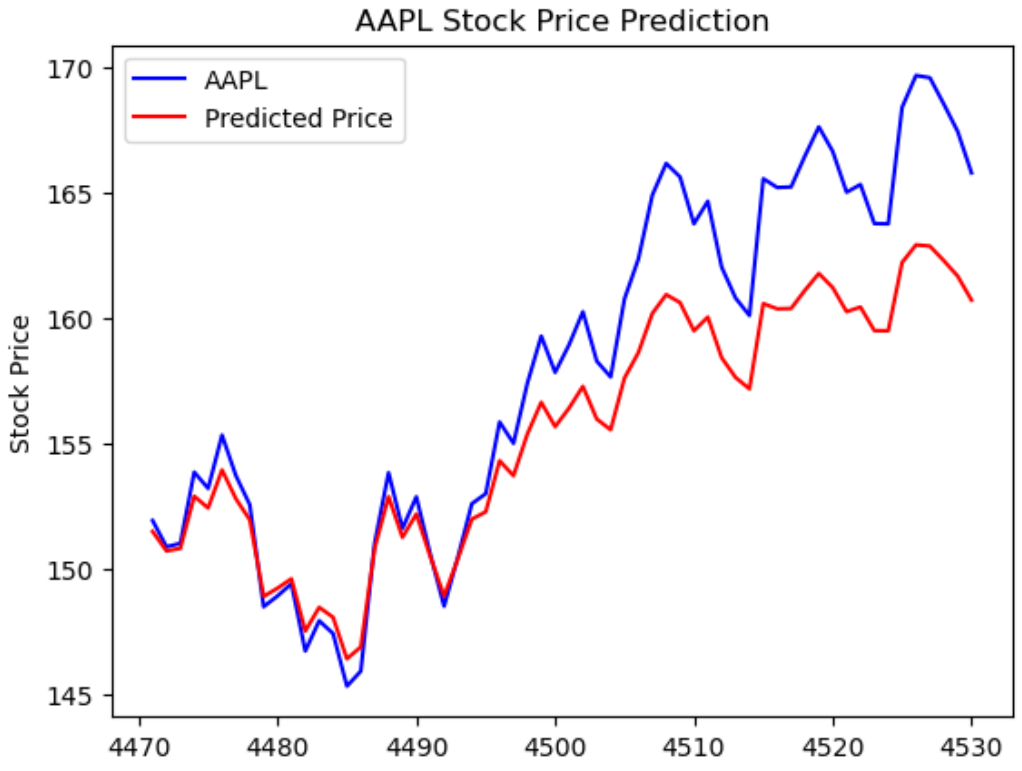
```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(training_data, training_data, epochs=epochs,
    batch_size=batch_size)
```

Sada je vreme da napravimo predikcije na test skupu. Osim toga, inverzno skaliramo dobijene vrednosti, kako bismo mogli da ih tumačimo i upoređujemo sa pravim vrednostima.

```
predictions = model.predict(testing_data)
predictions = scaler.inverse_transform(predictions)
```

Za kraj ćemo grafički da predstavimo rezultate. Plava linija predstavlja stvarne cene, dok su crvenom linijom prikazane predikcije ovog modela.

```
plt.plot(df[len(df)-prediction_days:].index, df[len(df)-
    prediction_days:]['Close'], color='blue', label='AAPL')
plt.plot(df[len(df)-prediction_days:].index, predictions, color=
    'red', label='Predicted Price')
plt.title('AAPL Stock Price Prediction')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```



Slika 9: Grafički prikaz rezultata LSTM modela

Preciznost modela je u početku izuzetno visoka, dok onda u daljim vremenskim trenucima opada.

Izračunali smo i srednje kvadratnu grešku koju daje naš model i dobili smo rezultat 11.683. Ovaj rezultat nije toliko stabilan, zato što za više pokretanja dobijaju se različiti modeli, pa samim tim i različite srednjekvadratne greške, koje su uglavnom u opsegu između 4 i 30.

Ovakva konfiguracija sekvencijalnog modela je inspirisana radom [10]. Dijagram tog modela, prikazan je na slici 10.

2.2 Long Short Term Memory (LSTM) Network Based Model

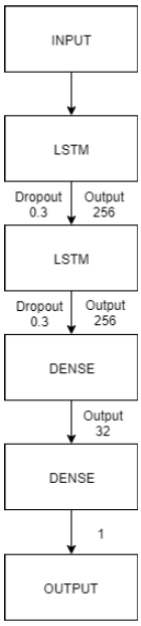


Figure 2. LSTM Layers

Slika 10: Sekvencijalni model iz rada [10]

Ipak, uz male modifikacije, daje dosta verodostojnije rezultate, što se lako može zaključiti samo posmatranjem grafičkih prikaza rezultata u tom i našem radu.

5 Zaključak

Želimo ovde da napomenemo da je inicijalna ideja bila da se implementira i **ARIMA**, pa bi rad bio bogatiji za još jedan model. Iako smo tome posvetili dosta vremena , na kraju je to poglavlje u potpunosti izbačeno iz rada, zato što su paketi koji se uglavnom koriste zastareli i nismo uspeli da se snađemo sa novijim verzijama. Tačnije, deo koda u kom se grafički prikazuju rezultati je uvek prikazivao predikcije kao horizontalnu liniju, što svakako nije tačno.

Rezultati koje smo dobili imaju dosta malu grešku, ali ona i dalje nije dovoljno mala kako bismo mogli da ih primenimo za berzanske spekulacije. Rezultati dobijeni SVM metodom su bolji od rezultata dobijenih LSTM, pre svega jer nema povećanja odstupanja sa vremenom.

Svakako, treba ispitati mogućnost da li drugi modeli mogu dovesti do još boljih rezultata, npr. pomenuti ARIMA ili ANN modeli. Takođe, mi smo prilikom predikcije posmatrali jedino close cenu deonica, a moguće je da bismo videli poboljšanje modela ubacivanjem drugih preditkora, kao npr razlike open i close cene.

Još jedan način na koji se ovaj rad može unaprediti jeste da se tesira i na drugim podacima, osim samo na akcijama kompanije *Apple*.

Literatura

[1] S. Jankovic and B. Milosevic, *Elementi Finansijske Matematike*. 2017.

[2] C. Abad, S. A. Thore, and J. Laffarga, "Fundamental analysis of stocks by two-stage dea," *Managerial and Decision Economics*, vol. 25, no. 5, pp. 231–241, 2004.

[3] E. P. Chan, *Quantitative Trading: How to Build Your Own Algorithmic Trading Business*. Hoboken, NJ: Wiley, 2008.

[4] https://finance.yahoo.com.

[5] S. Ding, X. Hua, and J. Yu, "An overview on nonparallel hyperplane support vector machine algorithms," *Neural computing and applications*, vol. 25, pp. 975–982, 2014.

[6] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, pp. 64–67, 2001.

[7] M. Nikolić and A. Zečević, "Mašinsko učenje," *Beograd: Matematički fakultet*, 2019.

[8] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[9] S. Aggarwal, "The ultimate guide to building your own lstm models." https://www.projectpro.io/article/lstm-model/832#mcetoc_1gsj4do3jv, 2023.

[10] I. Parmar, N. Agarwal, S. Saxena, R. Arora, S. Gupta, H. Dhiman, and L. Chouhan, "Stock market prediction using machine learning," in *2018 first international conference on secure cyber computing and communication (ICSCCC)*, pp. 574–576, IEEE, 2018.