Универзитет „Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Истражувачки проект по предметот Управување со ИКТ проекти на тема:

„Model for stock market prediction using historical data (USD/EUR)“

Изработил:

Ефрем Кулаков, индекс број: 211227

Ментори:

проф. д-р Димитар Трајанов

асс. Милена Трајаноска

Вовед.....3
Објаснување на кодот.....4
Резултати.....9

Вовед

Во овој истражувачки проект користиме податочно множество од историски податоци од берза за валутите долар и евро. Потоа користејќи повеќе-слојна LSTM невронска мрежа ги предвидуваме највисоката вредност на денот и вредноста на отварање на берзата.

Објаснување на кодот

Во овој дел ќе го објаснам целиот напишан код.

Најпрво со помош на библиотеката yfinance (Yahoo Finance) го симнуваме податочното множество потребно за овој проект. Симнатото податочно множество има податоци од 2019 до 2025.

```
import datetime
import numpy as np
import pandas as pd
import yfinance as yf
from yahoofinancials import YahooFinancials
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.core.display import display
%matplotlib inline

# time period
d1 = '2019-01-01'
d2 = '2025-05-16'

# Fetching data from yahoo finance
# List of Symbols : http://finance.yahoo.com/webservice/v1/symbols/allcurrencies/quote

eurusd_df = yf.download('EURUSD=X',
                        start=d1,
                        end=d2,
                        progress=False,
)
```

Податочното множество изгледа вака:

```
eurusd_df
```

Price	Close	High	Low	Open	Volume
Ticker	EURUSD=X	EURUSD=X	EURUSD=X	EURUSD=X	EURUSD=X
Date					
2019-01-01	1.149306	1.155001	1.146500	1.149425	0
2019-01-02	1.146171	1.149700	1.134572	1.146132	0
2019-01-03	1.131811	1.140914	1.131734	1.131734	0
2019-01-04	1.139108	1.141774	1.134816	1.139095	0
2019-01-07	1.141044	1.147447	1.140524	1.141292	0
...	...	...	...	...	...
2025-05-09	1.122423	1.129114	1.119984	1.122423	0
2025-05-12	1.122965	1.124290	1.107506	1.122965	0
2025-05-13	1.109558	1.117743	1.109668	1.109558	0
2025-05-14	1.118556	1.126456	1.118243	1.118556	0
2025-05-15	1.118193	1.122776	1.117481	1.118193	0

1659 rows × 5 columns

Ги дефинираме колоните да имаат поедноставни имиња, со цел полесна манипулација со нив. Исто така ја тргаме колоната „volume“ бидејќи станува збор за берза со пари, а не со акции, и поради тоа не ни треба таа информација.

```
eurusd_df.columns = ['close','high','low','open','volume']
```

```
eurusd_df.drop(columns=['volume'], inplace=True)
```

```
eurusd_df
```

Date	close	high	low	open
2019-01-01	1.149306	1.155001	1.146500	1.149425
2019-01-02	1.146171	1.149700	1.134572	1.146132
2019-01-03	1.131811	1.140914	1.131734	1.131734
2019-01-04	1.139108	1.141774	1.134816	1.139095
2019-01-07	1.141044	1.147447	1.140524	1.141292
...	...	...	...	...
2025-05-09	1.122423	1.129114	1.119984	1.122423
2025-05-12	1.122965	1.124290	1.107506	1.122965
2025-05-13	1.109558	1.117743	1.109668	1.109558
2025-05-14	1.118556	1.126456	1.118243	1.118556
2025-05-15	1.118193	1.122776	1.117481	1.118193

1659 rows × 4 columns

Правиме lags за седум денови наназад, за сите 4 атрибути.

```
lags = range(7, 0, -1)
list(lags)
```

```
[7, 6, 5, 4, 3, 2, 1]
```

```
features = ['close', 'high', 'low', 'open']
```

```
for lag in lags:
    for column in features:
        eurusd_df[f"{column}_{lag}"] = eurusd_df[column].shift(lag)
```

Ја извршуваме оваа линија код, со цел да ги тргнеме сите непотребни NaN вредности.

```
eurusd_df.dropna(axis=0, inplace=True)
```

Ги тргаме колоните со првичните вредности за атрибутите „close“, „high“ и „low“. Исто така ја доделуваме колоната „open“ за таргет вредност.

```
eurusd_df.drop(columns=['close', 'high', 'low'], inplace=True)
```

```
target = 'open'
```

Со помош на функцијата train_test_split, го делиме множеството на тренирачко и тестирачко.

```
from sklearn.model_selection import train_test_split
```

```
X, y = eurusd_df.drop(columns=[target]), eurusd_df[target]
```

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.20, shuffle=False)
```

Користејќи ја библиотеката MinMaxScaler го скалираме множеството.

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()
train_X = scaler.fit_transform(train_X)
test_X = scaler.transform(test_X)
```

```
scaler = MinMaxScaler()
train_y = scaler.fit_transform(train_y.to_numpy().reshape(-1, 1))
```

Го менуваме обликот на податоците.

```
# (samples, lags, features)
lag = 7
(train_X.shape[0], lag, (train_X.shape[1] // lag))
```

```
(1321, 7, 4)
```

```
train_X = train_X.reshape((train_X.shape[0], lag, (train_X.shape[1] // lag)))
test_X = test_X.reshape((test_X.shape[0], lag, (test_X.shape[1] // lag)))
```

```
from keras.models import Sequential
from keras.layers import Input, LSTM, Dense
```

```
train_X.shape
```

```
(1321, 7, 4)
```

Трениране невронска мрежа со три различни слоеви од кои два се LSTM.

```
model = Sequential([
    Input((train_X.shape[1], train_X.shape[2],)),
    LSTM(64, activation="relu", return_sequences=True),
    LSTM(32, activation="relu"),
    Dense(1, activation="linear")
])
```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 7, 64)	17,664
lstm_1 (LSTM)	(None, 32)	12,416
dense (Dense)	(None, 1)	33

Total params: 30,113 (117.63 KB)
Trainable params: 30,113 (117.63 KB)
Non-trainable params: 0 (0.00 B)

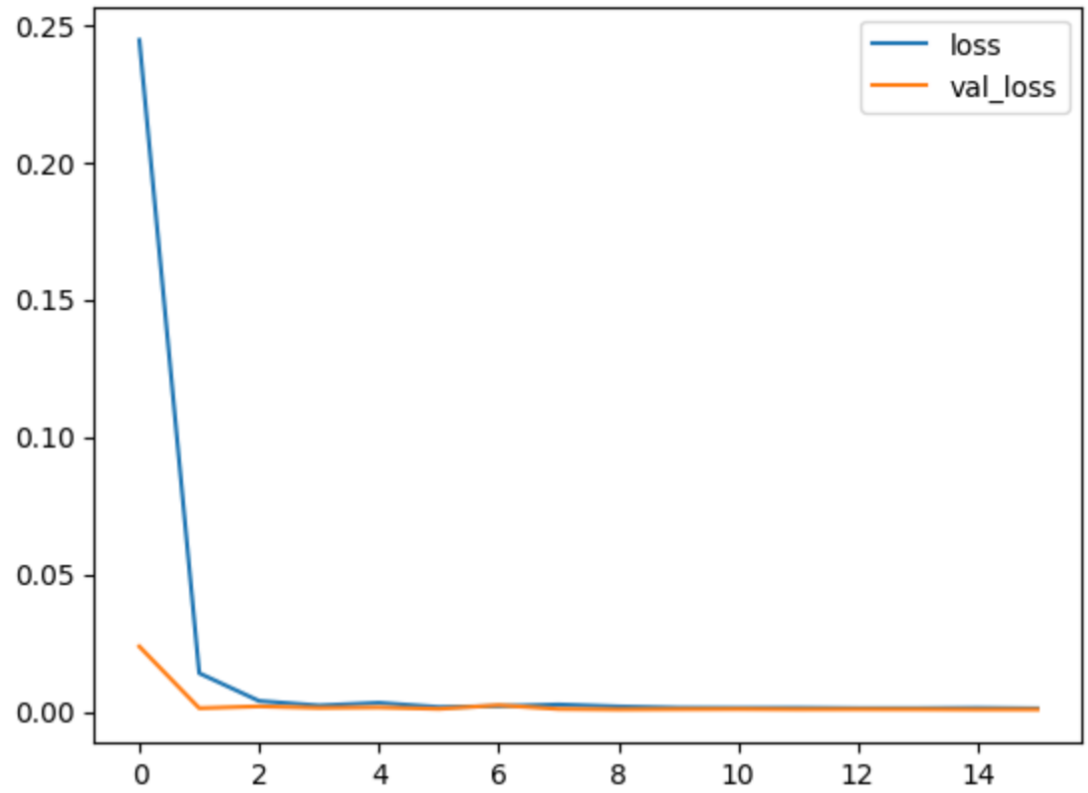
```
model.compile(
    loss="mean_squared_error",
    optimizer="adam",
    metrics=["mean_squared_error"],
)
```

```
history = model.fit(train_X, train_y, validation_split=0.20, epochs=16, batch_size=64, shuffle=False)
```

Исцртуваме график на кој прикажуваме „loss“ и „val_loss“.

```
sns.lineplot(history.history["loss"], label="loss")
sns.lineplot(history.history["val_loss"], label="val_loss")
```

```
<Axes: >
```



Правиме предвидување на вредностите и потоа ги пресметуваме метриките „Mean absolute error“, „Mean squared error“ и „R2“.

```
pred_y = model.predict(test_X)
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```

```
pred_y = scaler.inverse_transform(pred_y)
```

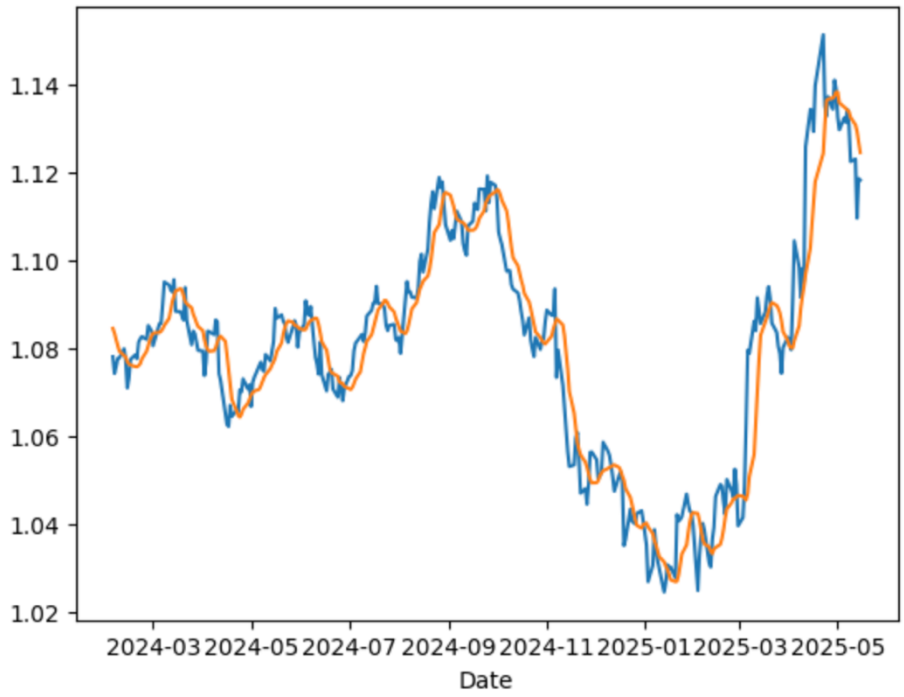
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
(mean_absolute_error(test_y, pred_y), mean_squared_error(test_y, pred_y), r2_score(test_y, pred_y))
```

```
(0.0059934454742155046, 6.60566126819248e-05, 0.9016105606166411)
```

Исцртуваме график со вистински вредности и предвидени вредности.

```
sns.lineplot(x=test_y.index, y=test_y.values)
sns.lineplot(x=test_y.index, y=pred_y.flatten())
```

```
<Axes: xlabel='Date'>
```



Истото го правиме и со „target = high“ за да ги споредиме резултатите.

Резултати

Добиените резултати се следните:

- Кога за таргет земаме „open“:
  - Mean absolute error: 0.0059934454742155046
  - Mean squared error: 6.60566126819248e-05
  - R2 score: 0.9016105606166411
- Кога за таргет земаме „high“:
  - Mean absolute error: 0.006001901050348656
  - Mean squared error: 7.514509164140933e-05
  - R2 score: 0.8896001402695107

Забележуваме дека резултатите се разликуваат за многу малку. Најголемата разлика е за метриката Mean squared error.