MARMARA UNIVERSITY
FACULTY OF TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING
BLM3053 INTRODUCTION TO ARTIFICIAL NEURAL NETWORKS
FINAL PROJECT REPORT

STUDENT ID: 170421004
NAME LASTNAME: Kerem Tutumlu
PROJECT TITLE: Stock Prediction with Using LSTM

Prof. Dr. Serhat ÖZEKES
Res. Asst. Abdulsamet AKTAŞ

1. INTRODUCTION

The stock exchange is an investment instrument that allows partial partnership with companies in a country. As a result of this partnership, investors receive a share of the company's profits in proportion to their investment. Of course, there are factors independent of this factor. For example, investors can also make a profit if there is a high demand for a share. The number of investors can be attributed to many factors, which shows the complexity and multi-layeredness of the stock market[1].

İnvestors have difficulty in making transactions. Because the stock market is a transaction that requires technical analysis and knowledge also it has a multiple variety[2]. This can be gained through experience and education. We will train a model for stock prediction to help investors for decision. In this way, investors will be able to get advice when they invest and will be able to invest more confidently by drawing a more accurate path. The focus will be on daily predictions.

We will use historical and current datasets for training. We will access the datatsets through the internet. These datasets will contain the information required for stock market technical analysis and we will train our network with these datas. We will take the daily data of the stocks and add them to the datasets.

2. LITERATURE REVIEW

Research on the use of artificial intelligence in financial transactions dates back to the 1990s. Since then, there have been many approaches to this field [3]. In a research at that time, a neural network model was developed with a rough set theory. There was an approach that the main problem with neural networks was the interpretation of the results [4]. In another research a year later, proposed a probability neural network for stock market prediction and said that looking at the index describes the behavior of a market [5]. Then Artificial Probabilistic Network (APN) design was presented. This design considers historical prices and basic indicators as input [6].

If we look at more recent researches, we see a paper that compares Support Vector Machine (SVM), k-Nearest Neighbour Classifier, Probabilistic Neural Network (PNN), Classification and Regression Tree (CART), boosting (Adaboost) and bagging algorithms aiming to perform binary classification of financial assets. As a result of this research, the most efficient algorithm is said to be the boosting algorithm [7].

None of these researches tried to produce something more effective by also working on social data. Most of them presented a research by training their models on historical, numerical and graphical data. In 2016, researchers proposed a model that trained with twitter data. This paper shows us that we can make a stock price prediction based on the opinions posted on Twitter about a company and show the strong relationship between the two [8].

Then if we look another effective algorithm, we see a LSTM (Long Short Term Memory) Algorithm. This algorithm solves RNN's (Recurrent Neural Network) storing long time memory issue and works efficiently with time series. This network transports historical datas. Each cell has forget gate, memory gate and output gate [9].

3. MATERIALS AND METHODS

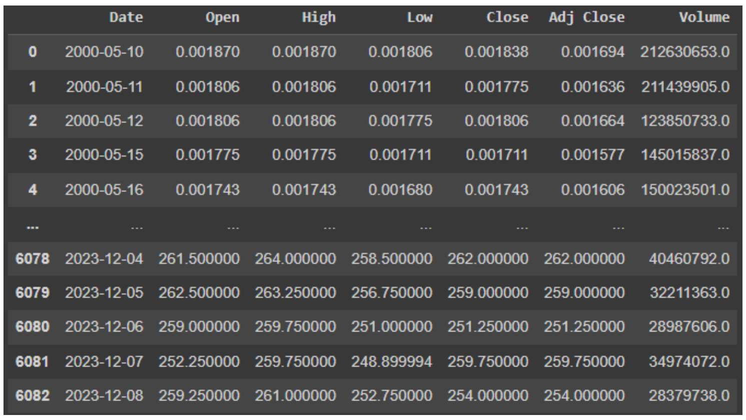
For financial prediction, we need a dataset as input to our neural network. Open source datasets and websites where we can extract the stock data are available on the internet. Some of them require a fee, others we can access for free. We can access existing datasets from https://www.kaggle.com/. But to create a more flexible dataset, we will extract the data ourselves from the internet. One of the websites where we can pull the data ourselves is https://polygon.io/. We can access 2 years historical data and get end of day data with free membership. There is another website we can access the all historical data and real time data with no fee. This website is https://finance.yahoo.com/. This website provides the requirements for our project, so we will extract the data from there.

The data we will extract includes opening, closing, daily max-min, volume and adjusted close price. Adjusted close price is the closing price adjusted for stock splits and dividend distributions. We will use a LSTM cells for learning. We have 6 inputs from stock but we will increase this number 4 times by getting 4 days of data and another 6 values from BIST100 index. Because BIST100's value affects all of the BIST100 stock market. We have 48 input for training model. We will decide the activation function, number of hidden layers and perceptrons by trying different options and comparing their results. The output layer will include future stock closing value for daily.

4. PROPOSED APPROACH

For training the model we need a data first. We will pull the historical data from yahoo finance with "yfinance" python library. Then we should organize data inputs outputs for our model. We will use "numpy" and "pandas" python library for this job. These libraries provides us to manipulate data the shape we want. We need a date data for index, past stock values data for predicting, closing value for output layer.

Firstly to decide which combination we use, we will work on less stocks data. We get the all historical data for choosen stocks from yahoo finance. This datas has .csv format. We can read the data from there with read_csv() function from pandas library. Then we can see each datasets like figure 1. shown belong.



	Date	Open	High	Low	Close	Adj Close	Volume
0	2000-05-10	0.001870	0.001870	0.001806	0.001838	0.001694	212630653.0
1	2000-05-11	0.001806	0.001806	0.001711	0.001775	0.001636	211439905.0
2	2000-05-12	0.001806	0.001806	0.001775	0.001806	0.001664	123850733.0
3	2000-05-15	0.001775	0.001775	0.001711	0.001711	0.001577	145015837.0
4	2000-05-16	0.001743	0.001743	0.001680	0.001743	0.001606	150023501.0
...	...	...	...	...	...	...	...
6078	2023-12-04	261.500000	264.000000	258.500000	262.000000	262.000000	40460792.0
6079	2023-12-05	262.500000	263.250000	256.750000	259.000000	259.000000	32211363.0
6080	2023-12-06	259.000000	259.750000	251.000000	251.250000	251.250000	28987606.0
6081	2023-12-07	252.250000	259.750000	248.899994	259.750000	259.750000	34974072.0
6082	2023-12-08	259.250000	261.000000	252.750000	254.000000	254.000000	28379738.0

Figure 1. : Turk Hava Yollari's Historical Data

We will use 4 days of data for predicting. That's why we need 'Date', 'Open', 'High', 'Low', 'Volume', 'Close' and 'Adj Close' values from 4 days. And we should add 5th's day 'Close' and 'Adj Close' value for using as output. We can do it with pandas library. And then we need convert this datasets to numpy array.

When we do all of this job our datasets are ready for training. For train we need some of components from tensorflow library. We created our neural network with 1 input layer, 1 LSTM (Long-Short Term Memory) layer with 128 cells, 1 dense layer with 256 cells, 1 dense layer too with 128 cells and then output layer. And we add our main model dropout layer after LSTM layer and dense layer with 256 cell for prevent the overfitting problem but we didn't use this layer for deciding train parameters.

LSTM blocks consists input gate, forget gate, output gate and cell unit.

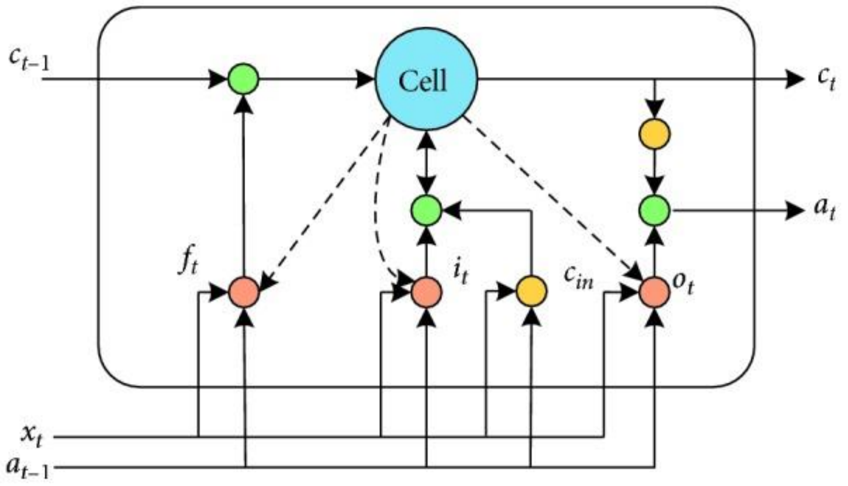


Figure 2. : Long-Short Term Memory Block [10]

f_t is forget gate. This gate decides which data must be retained or discarded to keep past information storage. *i_t* is the input gate which includes neuron and previos memory unit effects. It is used to control if the historical information needs to be updated to the LSTM block. *c_{in}* is the candidate update content which calculated by tanh function. Current time memory cell state *c_t* is calculated from the current candidate cell *c_{in}* , the previous time state *c_{t-1}* , the input gate information *i_t* and the forget gate information *f_t* . *o_t* is the generated at the output gate of the LSTM block at the current time. And *a_t* determines the amount of information for output at the current time[10]. All of this gates can be calculated as follows:

$$
\begin{aligned}
f_t &= \mathrm{sigmoid}(W_f \cdot [a_{t-1}, x_t, c_{t-1}] + b_f), \\
i_t &= \mathrm{sigmoid}(W_i \cdot [a_{t-1}, x_t, c_{t-1}] + b_i), \\
c_{\mathrm{in}} &= \tanh(W_c \cdot [a_{t-1}, x_t, c_{t-1}] + b_c), \\
c_t &= f_t \cdot c_t + i_t \cdot c_{\mathrm{in}}, \\
o_t &= \mathrm{sigmoid}(W_o \cdot [a_{t-1}, x_t, c_{t-1}] + b_o), \\
a_t &= o_t \cdot \tanh(c_t).
\end{aligned}
$$

Figure 3. : Calculations of LSTM cell gates [8]

Firstly we can decided our optimization function. We tried ‘Adam’, ‘AdamW’, ‘Adagrad’ and ‘RMSprop’ functions with 0.001 learning rate, relu activation function on denses and 75 epoch. We used different stock’s 1 year historical value for testing. The results are as follows ;

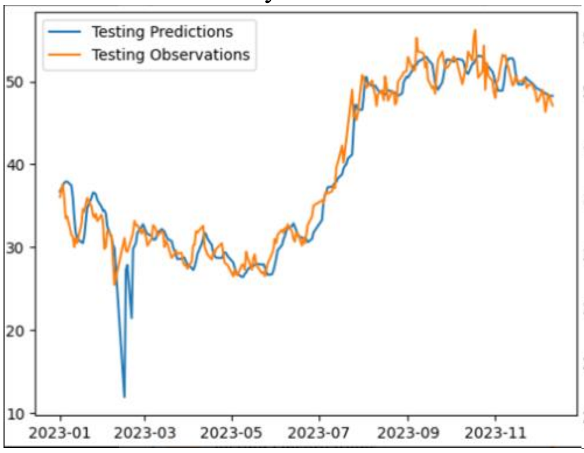


Figure 3. : Adam prediction result

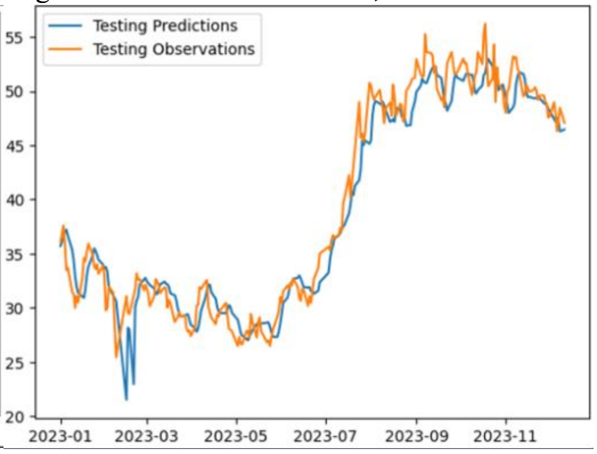


Figure 4. : AdamW prediction result

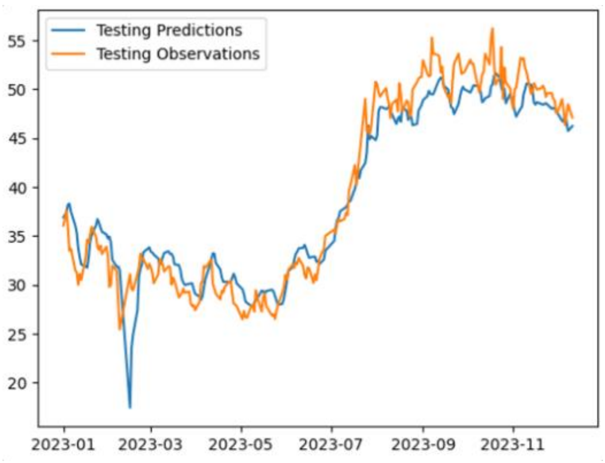


Figure 5. : RMSprop prediction result

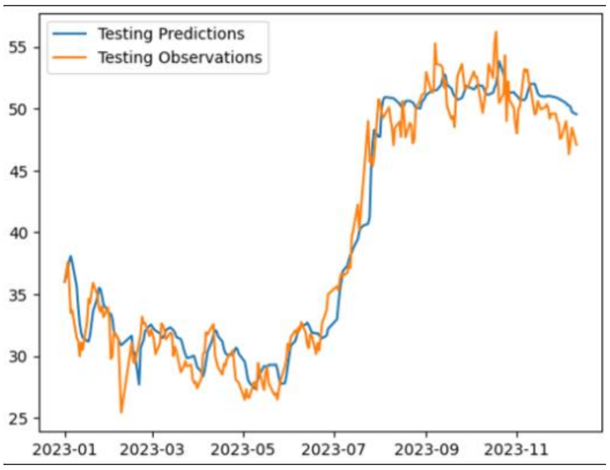


Figure 6. : Adagrad prediction result

For looking at the results we used matplotlib library in python. When we look at the results we saw 'AdamW' optimization function has the best results. That's why we will use this optimization for our model. Then we decided our denses activation function from 'Relu', 'Leaky Relu', 'Tanh' and 'Selu' functions. For testing we used AdamW optimization, 75 epoch, 0.001 learning rate and tried different combination of activation functions. The results are as follows ;

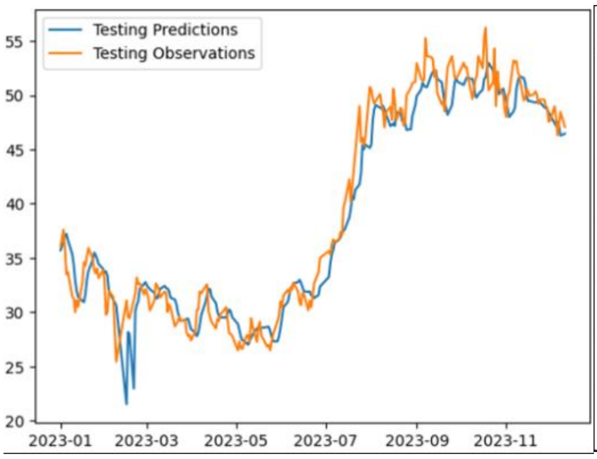


Figure 7. : 'Relu' prediction result

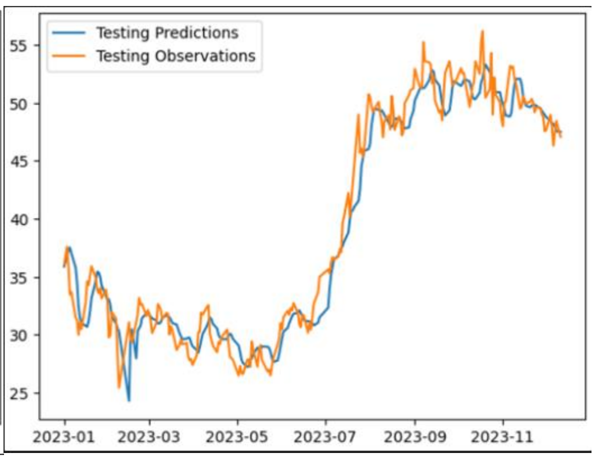


Figure 8. : 'Leaky Relu' prediction result

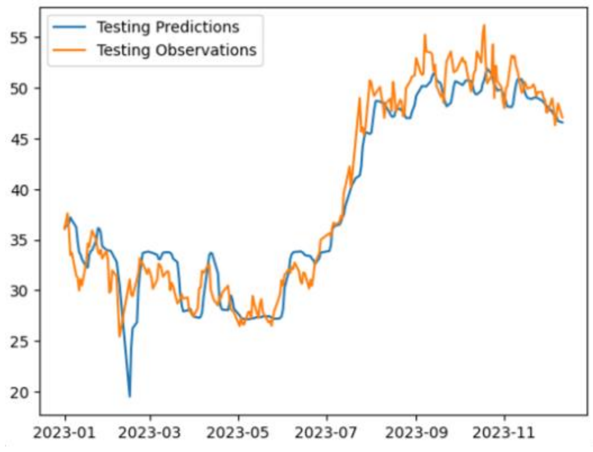


Figure 9. : 'Tanh' prediction result

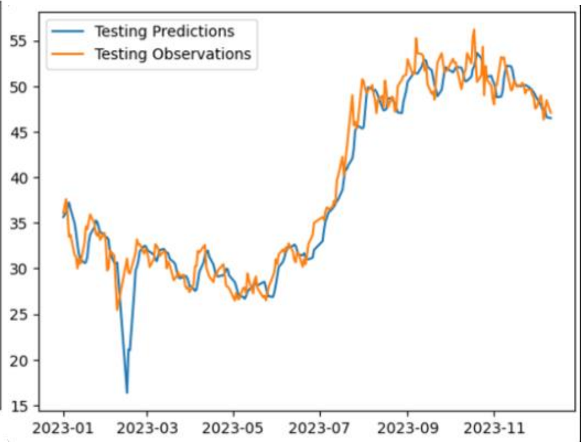


Figure 10. : 'Selu' prediction result

For this results 'Leaky Relu' is the best option for our model. Final combination is 'AdamW' optimization function and 'Leaky Relu' activation function. For the main training we will extract all stock’s data with python and we will split them for training. We will make 75 epoch training. The cell numbers will stay as previously described and we will not change them.

5. TOOL

We used PyCharm IDE for train and google collab for test.

6. CODES

We add our csv file to project that contains stock codes and stock names. Then we read this .csv file;

```
stocksDf = pd.read_csv('C:/Users/kerem/Desktop/Masaustu/Dersler/Pdfler/5.
Yarıyıl/Yapay Sinir Ağları/Proje/stockPredict/stocksList.csv')
```

We setted this dataframe’s index;

```
stocksDf.index = stocksDf.pop('Sıra No')
```

For pull the historical data we need stock codes. That’s why we get stocks code;

```
stocksCode = stocksDf['Pay Kodu'].to_numpy()
```

Then we define an array which name historicalDataStocks and pull all stock’s historical data with use for loop;

```
historicalDataStocks = []
for i in range(0, len(stocksCode)):
    dfTemp = yf.download(stocksCode[i] + '.IS')
    historicalDataStocks.append(dfTemp)
```

Then we delete stokcs with less than 5 days of data from the array. We define length variable which equals length of historicalDataStocks array for loop and i variable for loop. If we delete element we decrease the length and i number. Every loop controls condition and if the conditions are met, the element is deleted from the array. After this control block we organise the columns of our dataset;

```
length = len(historicalDataStocks)
i = 0
while i < length:
    if len(historicalDataStocks[i]) <= 5:
        del historicalDataStocks[i]
        stocksCode = np.delete(stocksCode, i)
        i = i - 1
        length = length - 1
    historicalDataStocks[i] = historicalDataStocks[i][['Open', 'High',
'Low', 'Volume', 'Close', 'Adj Close']]
    i = i + 1
```

Then we extract BIST100’s historical value;

```
historicalDataIndex = yf.download('XU100.IS')
```

And then we use a function which name set_df for setting dataframe for training. This function returns 4 days of stock and BIST100 index value which combined in a single line. It takes 5 parameters;

```
def set_df(dataframe, historicalDataIndex, strFirstDate, strLastDate, n):
```

dataframe parameter for stock’s data, historicalDataIndexDataIndex for BIST100’s value, strFirstDate for first target close value’s date, strLastDate for last target close value’s date and n value for days;

```
firstDate = convert_datetime(strFirstDate)
lastDate = convert_datetime(strLastDate)
```

convert_datetime function is;

```
def convert_datetime(d):
    split = d.split('-')
    year, month, day = int(split[0]), int(split[1]), int(split[2])
    return datetime.datetime(year = year, month = month, day = day)
```
It takes datetimes as string and split for '-' character and take year, month and day from this splits and returns datetime object.

We get firstDate as targetDate and makes array definition for stocks and BIST100 value;
```
targetDate = firstDate

dates = []
X1, X2, X3, X4, X5, X6, iX1, iX2, iX3, iX4, iX5, iX6, Y = [], [], [], [],
[], [], [], [], [], [], [], [], []
```

And then we start a loop that set dataframes. First we get values until target date but not includes target date;
```
while True:
    df_subset = dataframe.loc[:targetDate].tail(n + 1)
```

And then if this target date is not appropriate for start we add days to targetDate variable;
```
while len(df_subset) != n + 1 or targetDate.weekday() == 5 or
targetDate.weekday() == 6:
    if targetDate.weekday() == 4:
        targetDate = targetDate + datetime.timedelta(days=3)
        df_subset = dataframe.loc[:targetDate].tail(n + 1)

    else:
        targetDate = targetDate + datetime.timedelta(days=1)
        df_subset = dataframe.loc[:targetDate].tail(n + 1)

    if (targetDate - lastDate).days >= 0:
        break
```

Then we get all datas to array until the target date;
```
dfIndex = historicalDataIndex.loc[:targetDate].tail(n + 1)

val1 = df_subset['Close'].to_numpy()
val2 = df_subset['Adj Close'].to_numpy()
val3 = df_subset['Open'].to_numpy()
val4 = df_subset['High'].to_numpy()
val5 = df_subset['Low'].to_numpy()
val6 = df_subset['Volume'].to_numpy()

ival1 = dfIndex['Close'].to_numpy()
ival2 = dfIndex['Adj Close'].to_numpy()
ival3 = dfIndex['Open'].to_numpy()
ival4 = dfIndex['High'].to_numpy()
ival5 = dfIndex['Low'].to_numpy()
ival6 = dfIndex['Volume'].to_numpy()

y = val1[-1]
x1, x2, x3, x4, x5, x6 = val1[:-1], val2[:-1], val3[:-1], val4[:-1],
val5[:-1], val6[:-1]
ix1, ix2, ix3, ix4, ix5, ix6 = ival1[:-1], ival2[:-1], ival3[:-1], ival4[:-
1], ival5[:-1], ival6[:-1]

dates.append(targetDate)
```

And then we get this temporary arrays to main arrays;
```
X1.append(x1)
X2.append(x2)
X3.append(x3)
```

```
X4.append(x4)
X5.append(x5)
X6.append(x6)

iX1.append(ix1)
iX2.append(ix2)
iX3.append(ix3)
iX4.append(ix4)
iX5.append(ix5)
iX6.append(ix6)

Y.append(y)
```

Then we increase he targetDate. Loop is break if targetDate equals lastDate;

```
if targetDate.weekday() == 4:
    targetDate = targetDate + datetime.timedelta(days=3)
else:
    targetDate = targetDate + datetime.timedelta(days=1)

if (targetDate - lastDate).days >= 0:
    break
```

Then we create a dataframe which return valu of this function. We add dates to ‘Target Date’ column;

```
ret_df = pd.DataFrame({})
ret_df['Target Date'] = dates
```

Then we convert the other values to numpy array;

```
X1 = np.array(X1)
X2 = np.array(X2)
X3 = np.array(X3)
X4 = np.array(X4)
X5 = np.array(X5)
X6 = np.array(X6)
iX1 = np.array(iX1)
iX2 = np.array(iX2)
iX3 = np.array(iX3)
iX4 = np.array(iX4)
iX5 = np.array(iX5)
iX6 = np.array(iX6)
```

And lastly we add our values to ret_df dataframe and return;

```
for i in range(0, n):
    ret_df[f'Close-{n - i}'] = X1[:, i]
    ret_df[f'Adj Close-{n - i}'] = X2[:, i]
    ret_df[f'Open-{n - i}'] = X3[:, i]
    ret_df[f'High-{n - i}'] = X4[:, i]
    ret_df[f'Low-{n - i}'] = X5[:, i]
    ret_df[f'Volume-{n - i}'] = X6[:, i]

    ret_df[f'I-Close-{n - i}'] = iX1[:, i]
    ret_df[f'I-Adj Close-{n - i}'] = iX2[:, i]
    ret_df[f'I-Open-{n - i}'] = iX3[:, i]
    ret_df[f'I-High-{n - i}'] = iX4[:, i]
    ret_df[f'I-Low-{n - i}'] = iX5[:, i]
    ret_df[f'I-Volume-{n - i}'] = iX6[:, i]

ret_df['Daily Target'] = Y

return ret_df
```

Then we use this function for getting setted dfs;

```
settedDfs = []

for i in range(0, len(historicalDataStocks)):
    settedDfs.append(set_df(
        historicalDataStocks[i],
        historicalDataIndex,
        (historicalDataStocks[i].index[0] +
datetime.timedelta(days=4)).strftime("%Y-%m-%d"),
        (historicalDataStocks[i].index[-1]).strftime("%Y-%m-%d"),
        n=4
    ))
```

Then we use another function for getting numpy arrays to each dataframes. This functions takes set_df's function return dataframes and returns numpy array;

```
def windowed_df_to_date(windowed_dataframe):
  df_as_np = windowed_dataframe.to_numpy()
  dates = df_as_np[:, 0]
  return dates

def windowed_df_to_X(windowed_dataframe):
  df_as_np = windowed_dataframe.to_numpy()
  dates = df_as_np[:, 0]
  middle_matrix = df_as_np[:, 1:-1]
  X = middle_matrix.reshape((len(dates), middle_matrix.shape[1], 1))
  return X.astype(np.float32)

def windowed_df_to_Y(windowed_dataframe):
  df_as_np = windowed_dataframe.to_numpy()
  Y = df_as_np[:, -1]
  return Y.astype(np.float32)
```

Then we use it for all dataframes;

```
dates, X, Y = [], [], []

for i in range(0, len(settedDfs)):
    dates.append(windowed_df_to_date(settedDfs[i]))
    X.append(windowed_df_to_X(settedDfs[i]))
    Y.append(windowed_df_to_Y(settedDfs[i]))
```

We concatenate all datas for train;

```
X_train = np.concatenate(X)
Y_train = np.concatenate(Y)
dates_train = np.concatenate(dates)
```

Then we create our model;

```
model = Sequential([
    layers.Input((48,1)),
    layers.LSTM(128),
    layers.Dropout(0.2),
    layers.Dense(256, activation='LeakyReLU'),
    layers.Dropout(0.2),
    layers.Dense(128, activation='LeakyReLU'),
    layers.Dense(1)
])
```

And then we started our train with this codes;

```
kFold = sklearn.model_selection.KFold(n_splits=5)
```

```
model.compile(loss='mse',
              optimizer=AdamW(learning_rate=0.001),
              metrics=['mean_absolute_error'])
for train, test in kFold.split(X_train, Y_train):
    model.fit(X_train[train], Y_train[train], epochs=75)
```

We saved our model;

```
model.save('model7.h5')
```

7. RESULTS

We trained our model with ASGYO.IS, ASELS.IS, ATAKP.IS, BIMAS.IS, BRLSM.IS, BUCIM.IS and CLEBI.IS stocks historical datas. This training took 5 hours and 21 minutes. The results of prediction that have never been used for training are as follows ;

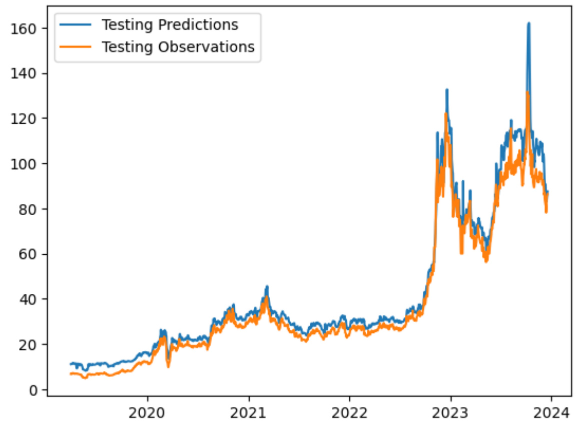


Figure 11. :ALCTL.IS Prediction Results

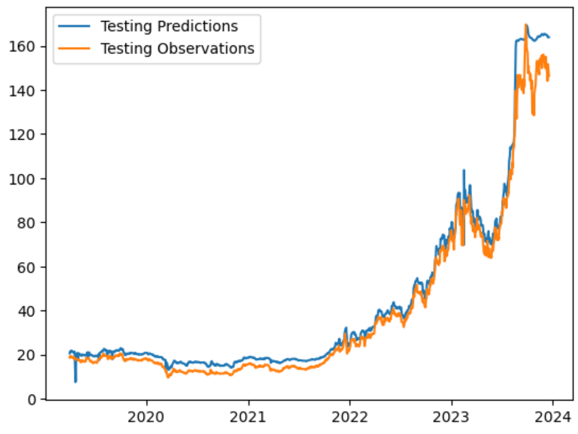


Figure 12. : TUPRS.IS Prediction Results

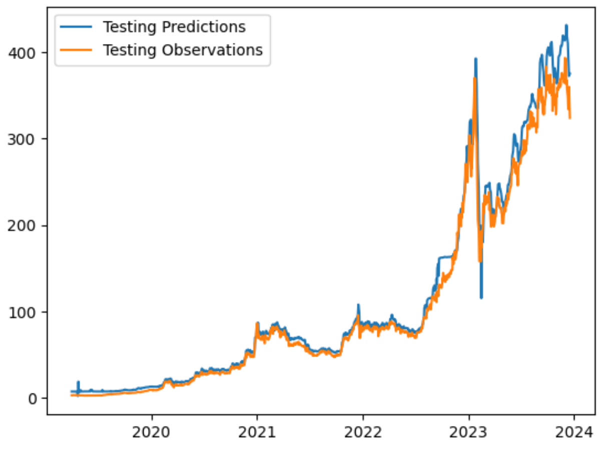


Figure 13. : GUBRF.IS Prediction Results

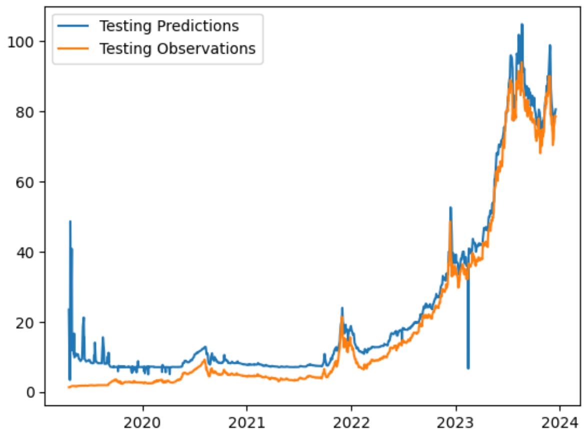


Figure 14. : RALYH.IS Prediction Results

When we look at those results we can see our model can predict general movements of stocks. Then we decided to increase training data. We merge all stock's historical data. And we decided to use k-fold cross validation for prevent overfitting. This training took 33 hours and 43 minutes. The results of prediction for ALCTL.IS, TUPRS.IS, GUBRF.IS and RALYH.IS are as follows ;

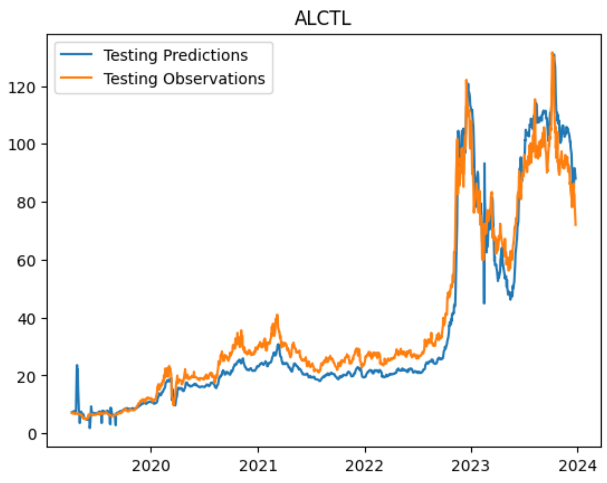


Figure 15. :ALCTL.IS Other Prediction Results

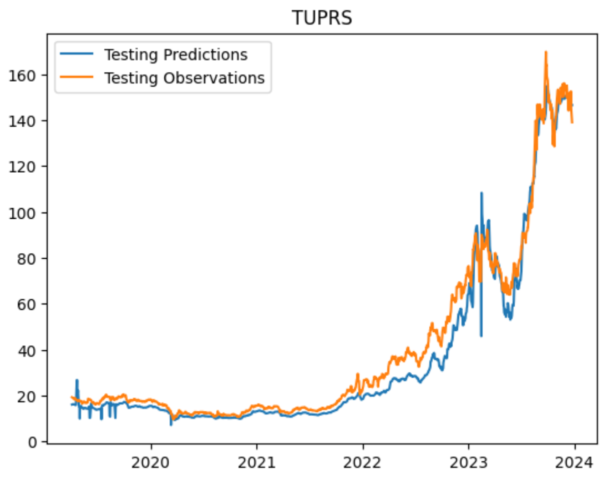


Figure 16. : TUPRS.IS Other Prediction Results

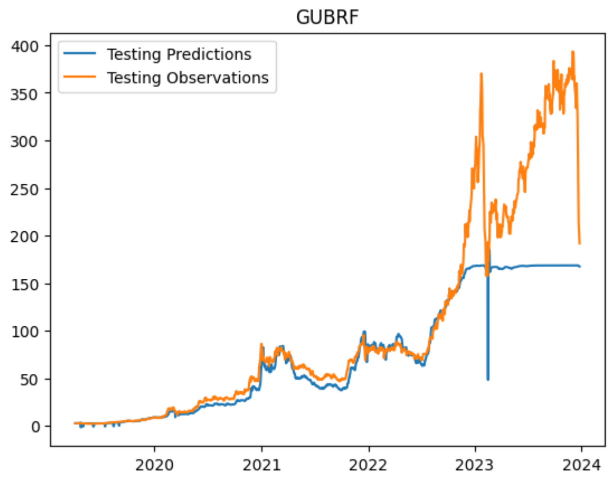


Figure 17. : GUBRF.IS Other Prediction Results

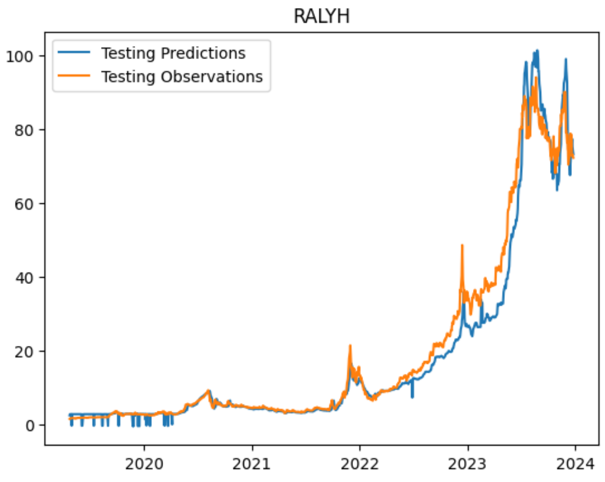


Figure 18. : RALYH.IS Other Prediction Results

8. DISCUSSION AND CONCLUSION

Predicting future stock value is hard because it depends on too many parameters. When we look at the results our model can predict general movements of stocks. And our dataset is larger than the other papers (etc. [9]). We can see some predictions are too far the real close value. It could depend on diversity of variables, model depth, number of epochs and number of days taken for input. Our model takes 4 days variable but if we increase that number we can make more accurative prediction. We used basic parameters for our model that’s why failing to predict unusual movements. We should add more parameters for better predict future value. We can use the social media data like [8]’s did. We can add other technical values.

If we do all of that implementations we believe we can develop more accurative model.

9. REFERENCES

[1]. Aduda, Josiah, Jacinta Mwelu Masila, and Erick Nyakundi Onsongo, "The determinants of stock market development: The case for the Nairobi Stock Exchange." International journal of humanities and social science 2.9 (2012): 214-230
[2]. Y. Gour,”A study on efficacy of trading strategies in stock market in India” Neuro Quantology, 2022, vol.20, pp.6386-6392, doi: 10.14704/nq.2022.20.9.NQ44748

[3]. F.G.D.C Ferreira, A.H. Gandomi and R.T.N. Cardoso,“Artificial Intelligence Applied to Stock Market Trading: A Review,” in IEEE Access, vol. 9, pp. 30898-30917, 2021, doi:10.1109/ACCESS.2021.3058133
[4]. R.H Golan and W. Ziarko, “A methodology for stock market analysis utilizing rough set theory,” Proceedings of 1995 Conference on Computational Intelligence for Financial Engineering (CIFEr), New York, NY, USAİ 1995İ pp. 32-40, doi: 10.1109/CIFER.1995.495230
[5]. K. Schierholt and C. H. Dagli, “Stock market prediction using different neural network classification architectures” IEEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering (CIFEr), New York, NY, USA, 1996, pp. 72-78, doi: 10.1109/CIFER.1996.501826
[6]. S. H. Kim and S. H. Chun, “Graded forecasting using an array of bipolar predictions; Application of probabilistic neural networks to a stock market index,” Int. J. Forecasting, vol.14, no. 3, pp.323-337, Sep.1998
[7]. Z. Zhanggui, H. Yau and A. M. N. Fu, “A new stock price prediction method based on pattern classification,” IJCNN’99. International Joint Conference on Neural Networks. Proceedings (Car. No. 99CH36339), Washington, DC, USA, 1999, pp. 3866-3870 vol.6, doi: 10.1109/IJCNN. 1999.830772
[8]. V.S. Pagolu, K.N. Reddy, G. Panda and B. Majhi, “Sentiment analysis of Twitter data for predicting stock market movements,” 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi, India, 2016, pp. 1345-1350, doi: 10.1109/SCOPES.2016.7955659.
[9]. A. Moghar, M. Hamicher, “Stock Market Prediction Using LSTM Recurrent Neural Network”, Procedia Computer Science, 2020, vol.170, pp. 1168-1173, doi: 10.1016/ISSN 1877-0509
[10]. Gao J, Zhang H, Lu P, Wang Z. An Effective LSTM Recurrent Network to Detect Arrhythmia on Imbalanced ECG Dataset. J Healthc Eng. 2019 Oct 13;2019:6320651. doi: 10.1155/2019/6320651. PMID: 31737240; PMCID: PMC6815557.