PREDICTION OF THE BEST BUY-SELL TIME

FOR A CERTAIN STOCK-MARKET SHARE

YUNUS EMRE KÜTÜK

ALİ ERKUT COŞKUN

ABİDİN ZEYNEL ÇETİN

ORHAN BATURAY YAĞINLI

Graduation Project

Department of Electrical and Electronics Engineering

May 2020

ABSTRACT

Graduation Thesis

PREDICTION OF THE BEST BUY-SELL TIME

FOR A CERTAIN STOCK-MARKET SHARE

Yunus Emre KÜTÜK

Ali Erkut COŞKUN

Abidin Zeynel ÇETİN

Orhan Baturay YAĞINLI

Eskişehir Technical University

Engineering Faculty

Electrical and Electronics Engineering

Supervisor: Assist. Prof. Dr. MEHMET FİDAN

2020, 56 pages

We researched and analyzed the ways of best prediction of the buy-sell time for a certain stock-market share. We took different approaches at the problem. Artificial intelligence could guess the future predicts better than us. So we thought that forecasting decomposed data may show strong and more accurate predictions than using whole data. However better mathematical modeling such as regression makes it easier. We decided to use LSTM that would be useful for predicting trend which is component of the data. By the way it is possible to train our data for long term and see the possible results with low error rate. Then we goal to reach the best working system. Based on our work, we combined together residual forecast with Machine Learning and trend forecast with LSTM to achieve prediction of whole time series. We have added all the details of our thesis to our report.

CONTENTS

Pages

LIST OF FIGURES

1.INTRODUCTION

We tried to use forecasting about the future with the data that we received from the stocks by making use of machine learning. Forecasting has fascinated people for thousands of years. In the world of finance, stock trading is one of the most important issues. The stock market is a complex nonlinear system. Stock market is affected by numerous external assets such as political, social and economical factors. Thus prices in the stock market rise and decrease randomly due to external and internal factors. As this changes affected by human behavior and psychology, it is hard job to predict correctly. Over the years professional traders have developed different analysis methods so there are many various techniques in the literature. In recent years machine learning has gained a great importance.

2.PRIMARY INFORMATION

2.1. Machine Learning

Machine learning follows a process of preparing data, training an algorithm and generating a machine learning model, and then making and refining predictions.

2.1.1. Preparing the data

Machine learning requires data that is analyzed, formatted and conditioned to build a machine learning model. Data preparation typically involves these tasks:

Select a sample subset of data. Make and track assumptions about the data to select attributes germane to the problem you want the algorithm to train for or solve.

Merge or join data sets to aggregate records. Merging simplifies the data and makes it easier to manage.

Format and sort the data for modeling. Choose the format: flat file or relational database for example. Certain algorithms may require data to be sorted in a specific way.

Clean the data by removing or replacing any blank or missing values. There are statistical analysis tools that can help inspect the data for errors and deviations. The goal is to ensure that data is exact, complete and relevant.

Normalize the data or adjust values that are measured on different scales to a common scale. [1]

2.1.2. Training the algorithm

Machine learning uses the prepared data to train a machine learning algorithm. An algorithm is a computerized procedure or recipe. When the algorithm is trained on the data, a machine learning model is generated. Selecting the right algorithm is essential to applying machine learning successfully. Selection is largely influenced by the application and the data available. But there are some commonly used algorithms and applications:

Regression algorithms

Linear and logistic regression are examples of regression algorithms used to understand relationships in data. Linear regression is used to predict the value of a dependent variable based on the value of an independent variable. Logistic regression can be used when the dependent variable is binary in nature, A or B.

Decision trees

Decision trees use classification to make recommendations based on a set of decision rules.

Instance-based algorithms

A good example of an instance-based algorithm is K-Nearest Neighbor or k-nn. It uses classification to estimate how likely a data point is to be a member of one group or another based on its proximity to other data points

Clustering algorithms

Think of clusters as groups. Clustering focuses on identifying groups of similar records and labeling the records according to the group to which they belong. This is done without prior knowledge about the groups and their characteristics. Types of clustering algorithms include the K-means, TwoStep and Kohonen clustering.

2.1.3. Predicting and refining

Once the data is prepared and the algorithm trained, the machine learning model can make determinations or predictions about the data on its own.

2.1.4. Types of Machine Learning

- Supervised machine learning

Supervised machine learning uses sample data that is well classified and labeled. It's supervised because it involves a set of feedback data that indicates whether the predictions based on the sample data are correct or incorrect.

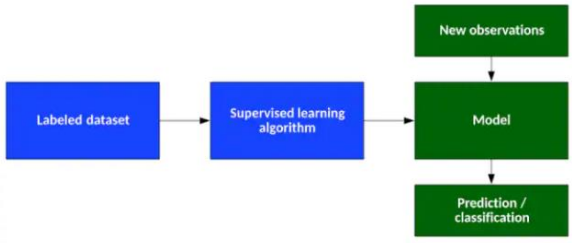

Figure 2.1: Modelling Supervised ML

- Unsupervised machine learning

Unsupervised machine learning algorithms infer patterns from a dataset without reference to known, or labeled, outcomes.

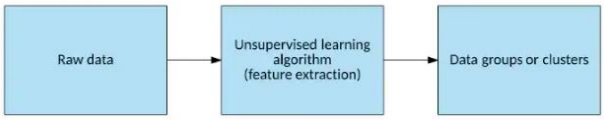

Figure 2.2: Modelling Unsupervised ML

- Reinforcement machine learning

Reinforcement machine learning is a behavioral learning model that is similar to supervised learning, but the algorithm isn't trained using sample data.

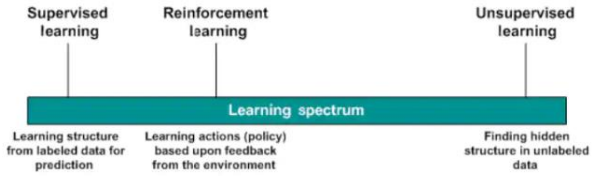

Figure 2.3: Modelling Reinforcement ML

- Deep learning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. [2]

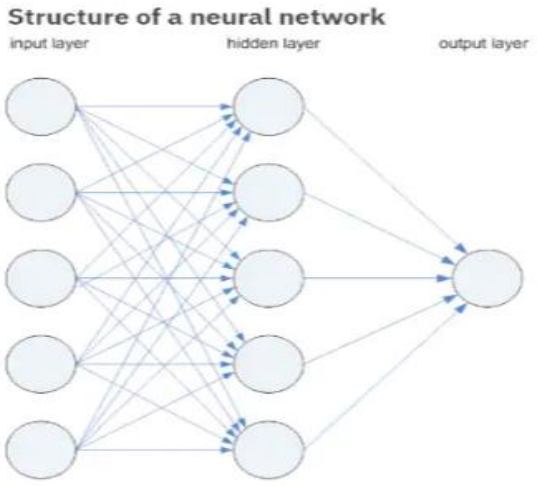

Figure 2.4: Modelling Deep Learning

2.1.5. Machine learning applications

One reason that machine learning is important is its growing prevalence in society and everyday life. Machine learning's ability to learn from data helps human analysts and decision makers: i.) Consume and better understand big data. İi.) Discover relationships in data to inspire insights and create opportunities. iii.) Identify anomalies and solve problems. iv.) Anticipate outcomes and make better decisions.
Machine learning uses three basic capabilities:

- **Classification** — dividing objects into two or more classes
- **Regression** — discovering relationships between variables
- **Clustering** — grouping objects by similar characteristics [3]

Applying these capabilities is often the domain of data science and data scientists. As data explodes, the need for professionals focused on harnessing and gaining value from vast volumes of data has become critical. Data scientists are working with other business professionals, from software developers to marketing specialists, to apply machine learning techniques in innovative ways.

2.2 Stock Market

The stock market refers to the collection of markets and exchanges where regular activities of buying, selling, and issuance of shares of publicly-held companies take place. Such financial activities are conducted through institutionalized formal exchanges or over-the-counter (OTC) marketplaces which operate under a defined set of regulations. There can be multiple stock trading venues in a country or a region which allow transactions in stocks and other forms of securities.

2.2.1. Working Principle of Stock Market

In a nutshell, stock markets provide a secure and regulated environment where market participants can transact in shares and other eligible financial instruments with confidence with zero- to low-operational risk. Operating under the defined rules as stated by the regulator, the stock markets act as primary markets and as secondary markets.

As a primary market, the stock market allows companies to issue and sell their shares to the common public for the first time through the process of initial public offerings (IPO). This activity helps companies raise necessary capital from investors. It essentially means that a company divides itself into a number of shares (say, 20 million shares) and sells a part of those shares (say, 5 million shares) to common public at a price (say, $ 10 per share). [4]

To facilitate this process, a company needs a marketplace where these shares can be sold. This marketplace is provided by the stock market. If everything goes as per the plans, the company will

successfully sell the 5 million shares at a price of $ 10 per share and collect $ 50 million worth of funds. Investors will get the company shares which they can expect to hold for their preferred duration, in anticipation of rising in share price and any potential income in the form of dividend payments. The stock exchange acts as a facilitator for this capital raising process and receives a fee for its services from the company and its financial partners.

Following the first-time share issuance IPO exercise called the listing process, the stock exchange also serves as the trading platform that facilitates regular buying and selling of the listed shares. This constitutes the secondary market. The stock exchange earns a fee for every trade that occurs on its platform during the secondary market activity.

2.3. Forecasting

2.3.1. Trends And Seasonality In Time Series

Time series datasets may contain trends and seasonality, which may need to be removed prior to modeling. Trends can result in a varying mean over time and seasonality can result in a changing variance over time, thus it makes time series non-stationary. Stationary datasets are those that have a stable mean and variance.

Time series are stationary if they do not have any seasonal effect or trend. If time series is stationary, it is easier to model.

Non-stationary time series show seasonal effects, trends, and other structures that depend on the time index. Time series analysis and forecasting methods are working when non- stationary time series data becomes stationary by removing trends and seasonal effects.

If we fit a stationary model to data, we assume our data are a realization of a stationary process. So our first step in an analysis should be to check whether there is any evidence of a trend or seasonal effects and, if there is, remove them.[5]

To make a series stationary we may use classical multiplicative decomposition model, which is;

$$(t)=\mu(t)+S(t)+Z(t) \quad (2.3.1.1)$$

$$t=1,2,3,....,n \quad (2.3.1.2)$$

μ(t): Slowly changing function known as a trend
S(t): Function with known period d referred to as a seasonal component
Z(t): Random noise component (remainder component)

Our aim will be to identify and extract the deterministic components μt and St in the hope that the residual or noise component Zt turns out to be a stationary random process. Then we will use the theory of stationary processes to find a satisfactory probabilistic model for the process Z(t), to analyze its properties and to use it in conjunction with μt and St for the purpose of prediction and control of X(t). An alternative approach to make a series stationary, is to apply difference operators repeatedly to the data X(t). We will go over several approaches to seasonality and trend removal. Such as estimation of μt and St and by differencing the data X(t).

The next method is calculates an L-step moving average centered at the time period, t, where L is the length of the seasonality (e.g., L would be 12 for a monthly series). Since the moving average gives the mean of a year's data, the seasonality factor is removed. Usually, the averaging removes the randomness component as well. Symbolically, this step is represented as;

$$m_t = \Sigma_{t=1}^{\infty} Y_t \qquad (2.3.1.3)$$

The seasonality is computed by dividing the Y series by the moving averages. Symbolically, this step is represented as;

$$Kt = {}^{Yt}/_{Mt} \qquad (2.3.1.4)$$

K series is composed of both the seasonality and the randomness. To calculate the seasonal component for each season, we simple average all like seasons. For example the average of all Februarys gives the seasonal value for February, and so on. Another complexity that must be dealt with is what to do at the ends of the series. Because the average is centered, the first and last L/2 averages cannot be computed (because of the lack of data). Many different end-effect techniques have been proposed. Our end-effect strategy can best be explained by considering an example. Suppose we have a monthly series that runs from January of 1985 to December of 1990. To compute the moving average centered at January, 1985, we will need estimated data back through July, 1984. The estimate of July 1984 is obtained by subtracting the difference of July, 1985 and July, 1986 from July, 1985.

2.3.2. Seasonal Differencing Method for Time Series

In time series, some trending behavior may be seen. In addition to this trend, cyclic or seasonal effect may occur as well. This seasonal effect should occur at specific regular intervals less than a year, such as weekly, monthly or quarterly. Various factors such as weather, vacation, holidays etc. cause this seasonal effect.

A seasonal difference is the difference between an observation and the previous observation from the same season. So;

$$Y_t^1 = Y_t - Y_{t-m} \qquad (2.3.2.1)$$

Where m the number of seasons. These are also called ―lag-m differences‖, as we subtract the observation after a lag of m periods. For monthly data, there are 12 periods in a season. The seasonal difference of Y at period t is,

$$Y_t - Y_{t-12} \qquad (2.3.2.2)$$

If seasonally differenced data appear to be white noise, then an appropriate model for the original data is

$$Y_t = Y_{t-m} + E_t \; [6] \qquad (2.3.2.3)$$

An example of a model for seasonal data is,

$$X_t = S_t + Z_t \qquad (2.3.2.4)$$

Where S_t is seasonal component that varies slowly one year to the next according to a random walk.

$$S_t = S_{t-12} + W_t \qquad (2.3.2.5)$$

This kind of seasonality is known as additive seasonality. Show that the seasonal difference operator ∇_{12} of order 1 acts on X_t, to produce a stationary series. A seasonal difference operator of order D is defined as;

$$\nabla s^D X_t = (1 - B^s)^D X_t \qquad (2.3.2.6)$$

Where D takes integer values. [7]

2.3.3.Forecast Errors

Forecasting methods needs to be evaluated for prediction accuracy. There are some methods to check the errors between the forecasted and actual values. Mean absolute percentage error (MAPE) and Root mean square error (RMSE) measures are commonly used ones to measure the accuracy of forecast.

MAPE or Mean absolute percentage error measures the magnitude of the error in percentage. It is calculated as the average of the absolute difference between forecast and actual value.

$$\text{MAPE} = \frac{\sum \frac{|A-F|}{A} * 100}{N} \qquad (2.3.3.1)$$

A = Actual Value
F = Forecast Value
N = Number of Observations

Month	Actual	Forecast	**Absolute Percentage Error**
1	112,3	124,7	%11
2	108,4	103,7	%4,3
3	148,9	116,6	%21,7
4	117,4	78,5	%33,1
MAPE			%17,6

Most people are MAPE is sensitive and should not be used with low-volume data. When Actual is zero MAPE is undefined due to Actual is in the denominator.If the actual value was very high (e.g. 1m), then the resulting error will be very low. These two situations should be taken into consideration. comfortable thinking in percent term, making MAPE is easy to understand and interpret.

RMSE (Root mean square error) or RMSD (Root mean square deviation) represents the square root of the second sample moment of the differences between predicted values and observed values.

$$\text{MAPE} = \sqrt{\frac{\sum_{i=1}^{N}(\text{Predicted}_i - \text{Actual}_i)^2}{N}} \quad (2.3.3.2)$$

2.3.4. Forecasted Time Error

Crystal Ball calculates three different error measures for the fit of each time-series forecast. uses one of these error measures to determine which time-series forecasting method is the best:

- RMSE
- MAD
- MAPE

RMSE

Root mean squared error is an absolute error measure that squares the deviations to keep the positive and negative deviations from canceling one another out. This measure also tends to exaggerate large errors, which can help when comparing methods.

The formula for calculating RMSE:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2} \quad (2.3.4.1)$$

where (y_i) is the actual value of a point for a given time period t, n is the total number of fitted points, and (y'_i) is the fitted forecast value for the time period t.

MAD

Mean absolute deviation is an error statistic that averages the distance between each pair of actual and fitted data points.

The formula for calculating the MAD:

$$\text{MAD} = \frac{\sum_{t=1}^{n}|y_i - y'_i|}{n} \quad (2.3.4.2)$$

where (y_i) is the actual value of a point for a given time period t, n is the total number of fitted points, and (y'_i) is the fitted forecast value for the time period t.

MAPE

Mean absolute percentage error is a relative error measure that uses absolute values to keep the positive and negative errors from canceling one another out and uses relative errors to enable you to compare forecast accuracy between time-series models.

The formula for calculating the MAPE:

$$\text{MAPE} = \sqrt{\frac{\sum_{i=1}^{N}(\text{Predicted}_i - \text{Actual}_i)^2}{N}} \qquad (2.3.4.3)$$

where (y_i) is the actual value of a point for a given time period t, n is the total number of fitted points, and (y'_i) is the fitted forecast value for the time period t. [8]

2.3.5. Block Diagram

As we applied all the steps of the project we present the progress in a schematic figure. Following the way lead us to the result of our project as it's shown above.

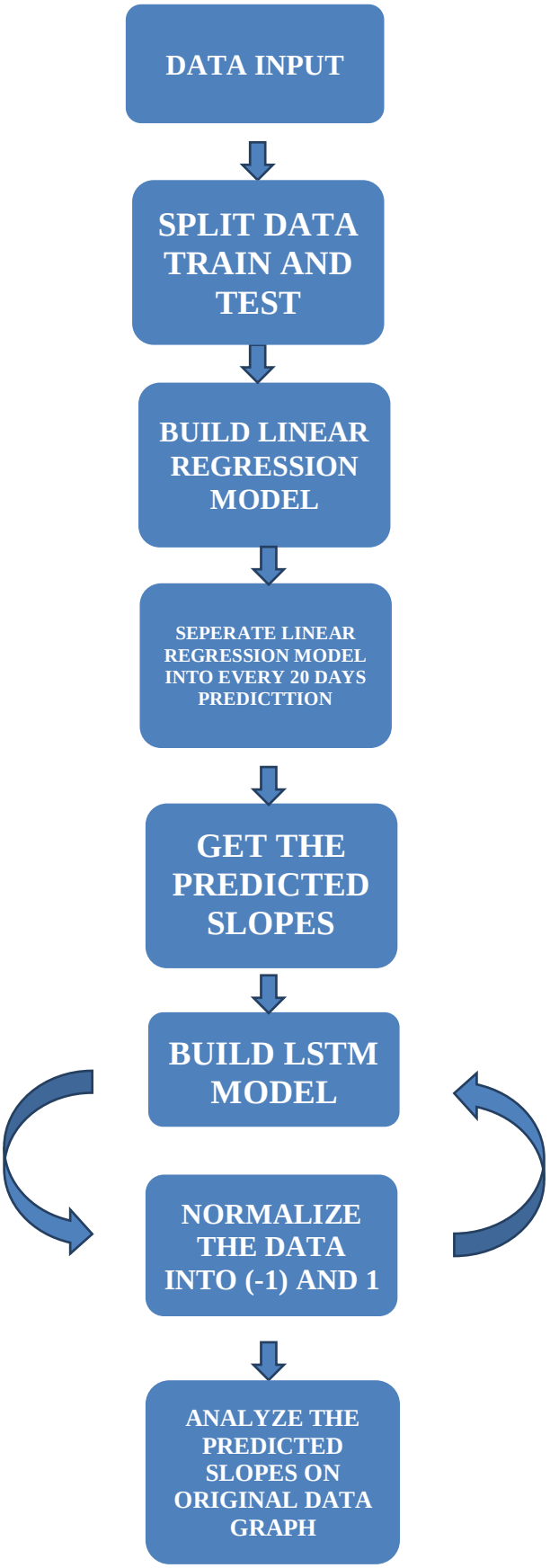
DATA INPUT
SPLIT DATA TRAIN AND TEST
BUILD LINEAR REGRESSION MODEL
SEPERATE LINEAR REGRESSION MODEL INTO EVERY 20 DAYS PREDICTTION
GET THE PREDICTED SLOPES
BUILD LSTM MODEL
NORMALIZE THE DATA INTO (-1) AND 1
ANALYZE THE PREDICTED SLOPES ON ORIGINAL DATA GRAPH

Figure 2.5: Block Diagram

3.REGRESSION ANALYSIS

Regression analysis is a powerful statistical method that allows you to examine the relationship between two or more variables of interest.[9]

200 years ago Regression analysis was published by scientist and today Regression Analysis continue to be an area of active research. Scientists and economists are trying to create new models for Regression Analysis.[10]

Regression models involve the following parameters and variables:

Unknown parameters : ß (3.1)
Independent parameters : X (3.2)
Dependent parameters : Y (3.3)

So our function for Regression Model relates as;

$$Y = f(X, B) \tag{3.4}$$

This Y is our predicted values not observed value.

There are lots of type of Regression models. Some of them are;

- Linear Regression
- Nonlinear Regression
- Polynomial Regression
- ElasticNet Regression
- Logistic Regression
- Lasso Regression
- Stepwise Regression

and more[11].

In our thesis we researched on Linear Regression in Long Term Forecasting model. Because we decided the best way to reach least error is provided by Linear Regression.

3.1.Linear Regression

This method establishes a relationship between dependent variable(Y) and one independent variable or more than one independent variables(X) using a best fit straight line. This regression type is the most widely known modeling technique. In this technique dependent variables are continuous but independent variables can be continues or discrete.

In mathematics we need a linear model formula to describe. For example;

Defined dataset is $=> \{Yi, Xi1, Xi2,...., Xip\}ni=1$ so linear formula is:

$$Y_i = B_0 + B_1 * X_{i^1} + \cdots + B_p * X_{ip} + E_i \tag{3.1.1}$$

$$Y_i = X_i^T * ß + E_i \quad (3.1.2)$$

$$i = 1,2,3,....,n \quad (3.1.3)$$

To get this formula we are using matrices and our simply definition function formula becomes;

$$Y = ß * X + E \quad (3.1.4)$$

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ . \\ . \\ Y_n \end{bmatrix} \quad X = \begin{bmatrix} X_1^T \\ X_2^T \\ . \\ . \\ X_n^T \end{bmatrix} = \begin{bmatrix} 1 & X_{11} & .... & X_{1p} \\ 1 & X_{21} & .... & X_{2p} \\ 1 & . & .... & .... \\ 1 & . & .... & .... \\ 1 & X_{n1} & .... & X_{np} \end{bmatrix} \quad ß = \begin{bmatrix} ß_0 \\ ß_1 \\ . \\ . \\ ß_p \end{bmatrix} \quad E = \begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ E_n \end{bmatrix}$$

Figure 3.1: Matrices for defined formula

Here Y is observed value, X is our input value(Independent Value), ß is coefficient (Unknown parameter) and E is an error. [12]

Above of formula $ß_0 + ß_1 * X_{i1} + ... + ß_p X_{ip}$ part is predicted value which equals to Y = f(X, ß) so to find error[13];

$$Yobserved - Ypredict = E_i \quad (3.1.5)$$

Implementation of Lineer Regression in our code and results of our experiment are shown below.

Code-1:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression


data=pd.read_csv("2016dolaralis.csv")

print(data)
print(data.describe())

x=data["Gun"]
y=data["Fiyat"]

x=x.values.reshape(-1,1)
y=y.values.reshape(-1,1)
```

```
plt.scatter(x,y)

lr=LinearRegression()
lr.fit(x,y)
y_head=lr.predict(x)
print(lr.score(x,y))
plt.plot(x,y_head,color="red")
plt.show()
```

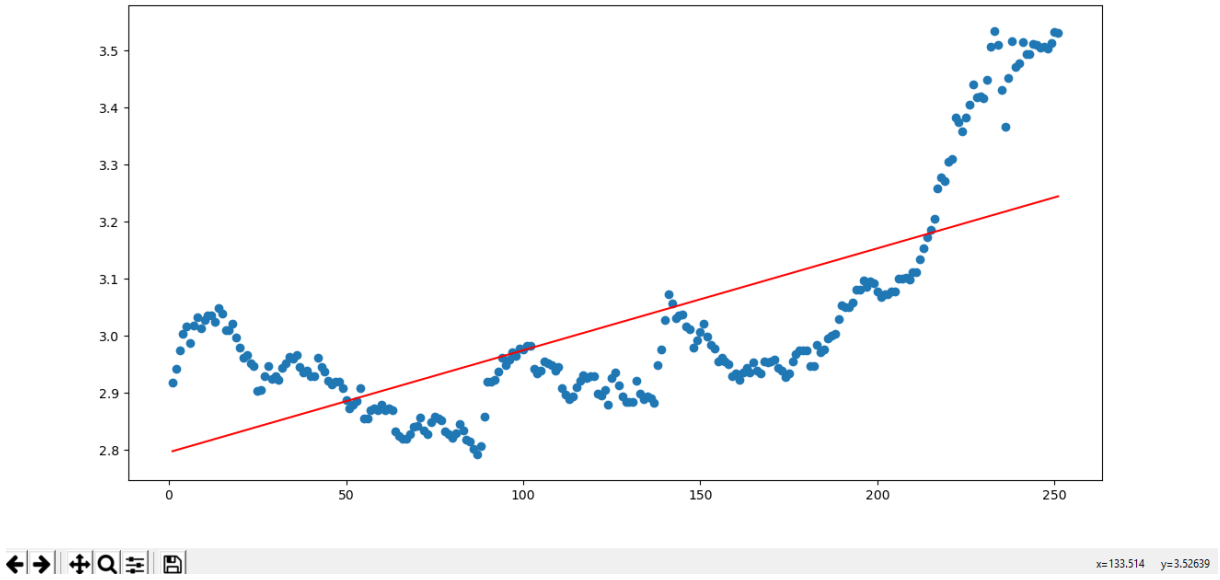

Figure 3.2: Application of Linear Regression on Various Power

Error Ratio is %48 of The Linear Regression Method, thus that wiil not be our choosen method for now.

Secondly we saw that datas foollow the same path as exponancial graphics by the way the decided new method is Polynomial Regression.

3.2.Nonlinear Regression

Nonlinear regression is a form of regression analysis in which observational datas are modeled by a function which is a nonlinear combination of the model parameters and depends on one or more independent variables. This technique is useful if only our observed datas are not linear.[14]
As mentioned above to apply nonlinear regression model we need to create a formula for our dataset these formulas can be exponentian, sinusoidial, logarithmic etc. but these formulas gives us $Y_{predicted}$, not $Y_{observed}$.

3.3. Polynomial Regression

Polynomial regression is a special case of linear regression where we fit a polynomial equation on the data with a curvilinear relationship between the target variable and the independent variables.

In a curvilinear relationship, the value of the target variable changes in a non-uniform manner with respect to the predictor (s).

In Linear Regression, with a single predictor, we have the following equation:

$$Y = \theta_0 + \theta_1 X \tag{3.3.1}$$

where,

Y is the target,

x is the predictor,

$\theta 0$ is the bias,

and $\theta 1$ is the weight in the regression equation

This linear equation can be used to represent a linear relationship. But, in polynomial regression, we have a polynomial equation of degree n represented as:

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \cdots + \theta_n x^n \tag{3.3.2}$$

Here:

$\theta 0$ is the bias,

$\theta 1$, $\theta 2$, …, θn are the weights in the equation of the polynomial regression,

and n is the degree of the polynomial

The number of higher-order terms increases with the increasing value of n, and hence the equation becomes more complicated.

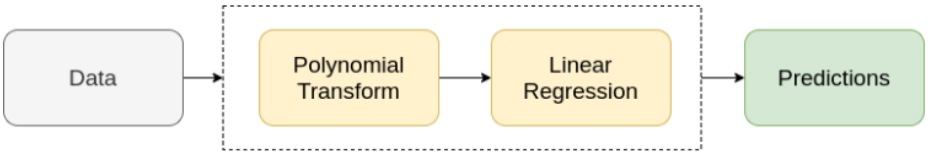

Figure 3.3: Pipeline of Polynomial Regression

Uses of Polynomial Regression:

These are basically used to define or describe non-linear phenomenon such as:

- Growth rate of tissues.
- Progression of disease epidemics
- Distribution of carbon isotopes in lake sediments

Implementation of Polynomial and Linear Regression in our code and results of our experiments are shown simultaneously below. [15]

CODE-2:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression



data=pd.read_csv("2016dolaralis.csv")

print(data)
print(data.describe())

x=data["Gun"]
y=data["Fiyat"]

x=x.values.reshape(-1,1)
y=y.values.reshape(-1,1)

plt.scatter(x,y)


lr=LinearRegression()
lr.fit(x,y)
y_head=lr.predict(x)
print(lr.score(x,y))
plt.plot(x,y_head,color="red")


for i in range(2,20):

    pl=PolynomialFeatures(degree=i)
    x_new=pl.fit_transform(x)
    plmodel=LinearRegression()
    plmodel.fit(x_new,y)
    y_head_2=plmodel.predict(x_new)
    plt.plot(x,y_head_2)
```

```
    print(i,plmodel.score(x_new,y))
plt.show ()
```

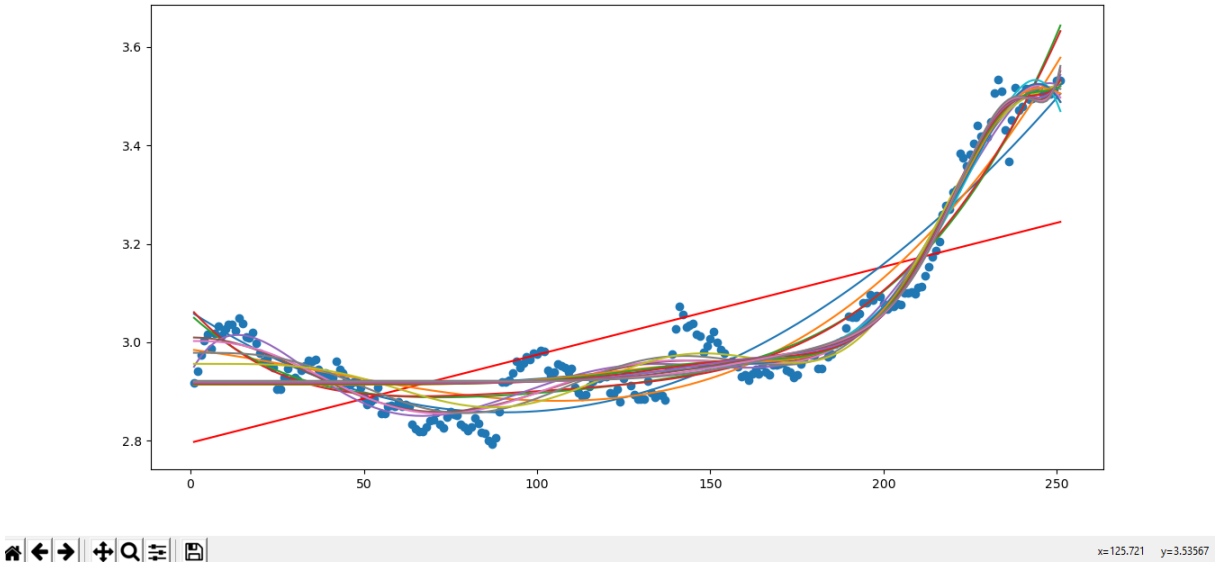

Figure 3.4: Application of Polynomial Regression on Various Power

The truth is accuracy which has the best ratio at eight degree as shown above;

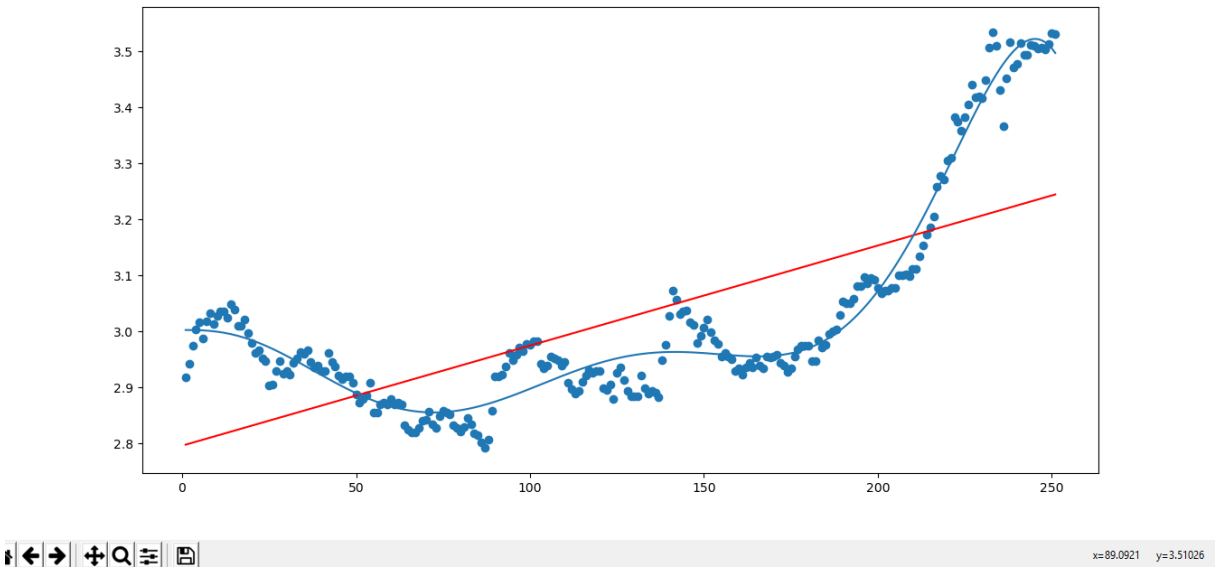

Figure 3.5: Application of Polynomial Regression on Best Predicted Power

3.4.Curve Fitting

There is a method which calls least squares method. This method looks sum of squares of errors and takes minimum result of it. Minimum sum value gives the best fitted line.[16]
Y_i is our observed value and S is sum of squares of errors, so;

$$E_i = f(X_{i,}ß) \quad (3.4.1)$$

$$S = \sum_{i=1}^{n} E_i \quad (3.4.2)$$

Minimum value of S gives us optimal parameter values.[17]

4.LSTM

Long Short Term Memory networks - usually just called “LSTMs” - are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997) and were refined and popularized by many people in following work. They work tremendously well on a large variety of problems, and are now widely used.
All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

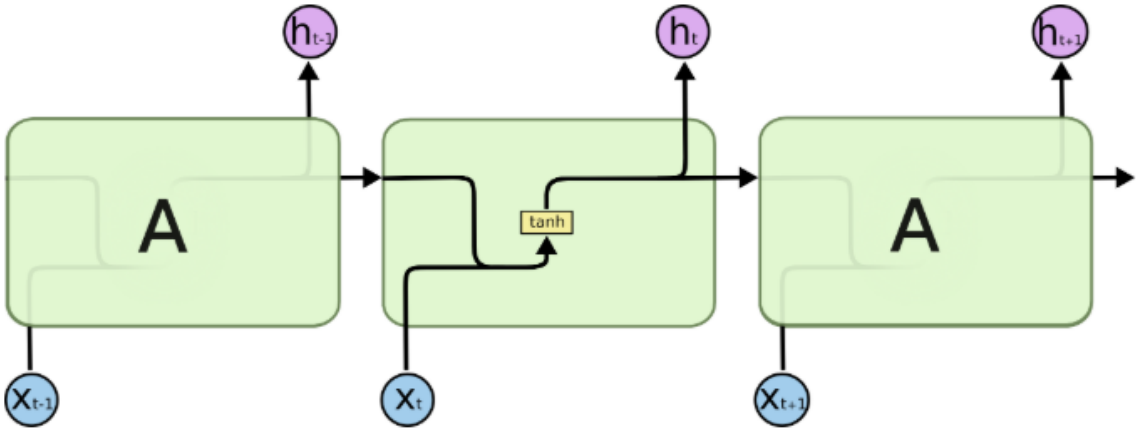

Figure 4.1: LSTM memory cell 1

The repeating module in a standard RNN contains a single layer.

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

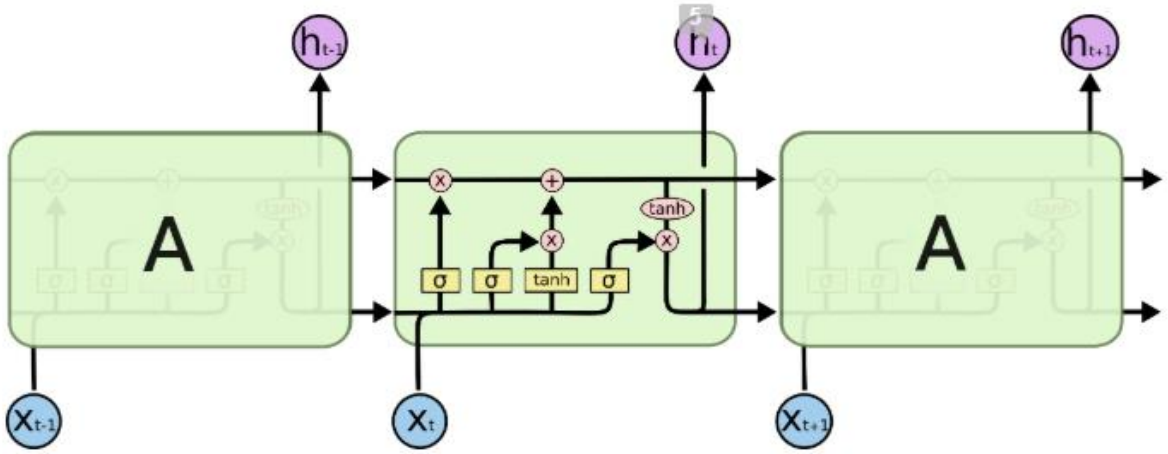

Figure 4.2: LSTM memory cell 2

The repeating module in an LSTM contains four interacting layers.

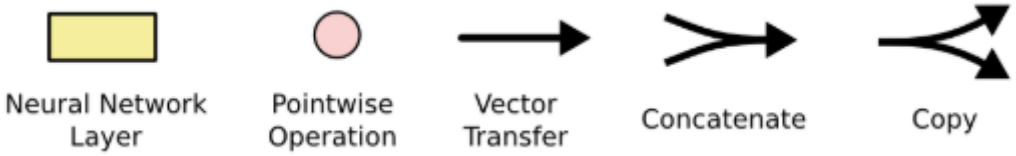

Figure 4.3: LSTM operations

In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations. [18]

4.1.The Core Idea Behind LSTMs

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

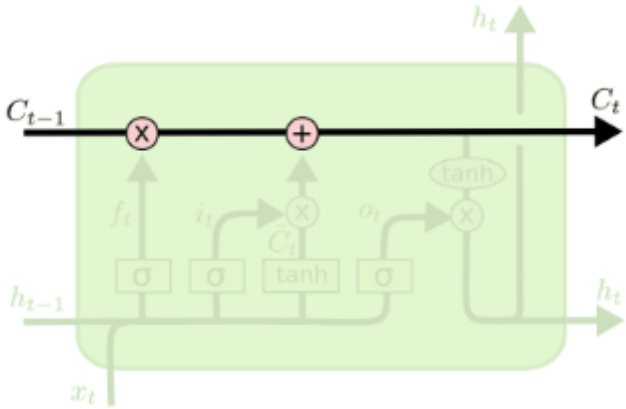

Figure 4.4: LSTM memory cell 3

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

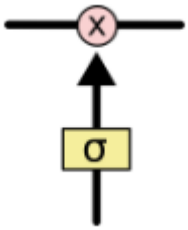

Figure 4.5: LSTM multiplication operation

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”

An LSTM has three of these gates, to protect and control the cell state.

4.2. Step-by-Step LSTM Walk Through

The first step in our LSTM is to decide what information we’re going to throw away from the cell state. This decision is made by a sigmoid layer called the “forget gate layer.” It looks at ht − 1 and xt, and outputs a number between 0 and 1 for each number in the cell state Ct − 1. A 1 represents “completely keep this” while a 0 represents “completely get rid of this.”

Let’s go back to our example of a language model trying to predict the next word based on all the previous ones. In such a problem, the cell state might include the gender of the present subject, so that the correct pronouns can be used. When we see a new subject, we want to forget the gender of the old subject.

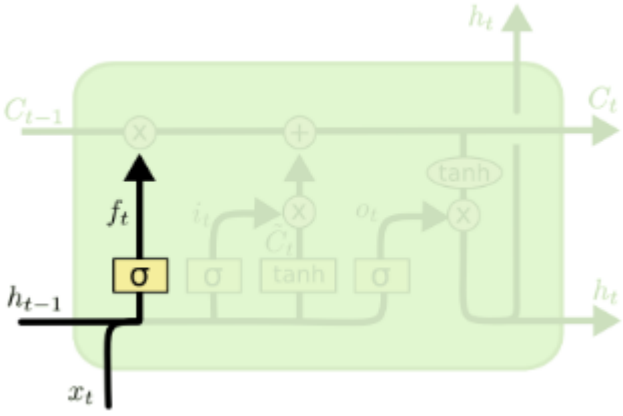

Figure 4.6: LSTM memory cell 4

$$f_t = \sigma\left(W_f . [h_{t-1}, x_{t]} + b_f\right) \quad (4.2.1)$$

The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, C ~ t, that could be added to the state. In the next step, we'll combine these two to create an update to the state.

In the example of our language model, we'd want to add the gender of the new subject to the cell state, to replace the old one we're forgetting.

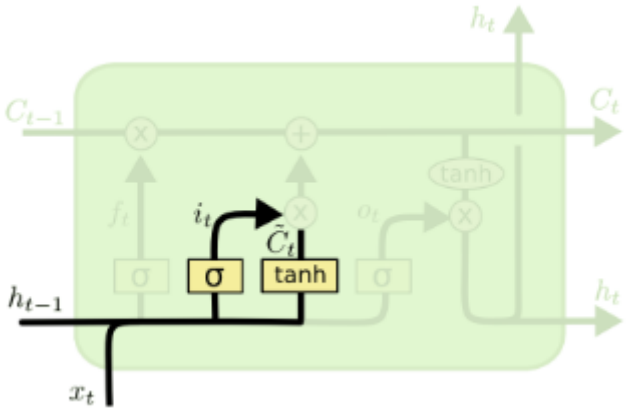

Figure 4.7: LSTM memory cell 5

$$i_t = \sigma(w_i . (h_{t-1}, x_t) + b_i) \quad (4.2.2)$$

$$\tilde{c}_t = \tan h(w_c \cdot (h_{t-1} x_t) + b_{c)} \quad (4.2.3)$$

Update the old cell state, Ct − 1, into the new cell state Ct. The previous steps already decided what to do, we just need to actually do it.

We multiply the old state by ft, forgetting the things we decided to forget earlier. Then we add it ∗ C ~ t. This is the new candidate values, scaled by how much we decided to update each state value.

In the case of the language model, this is where we'd actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps.

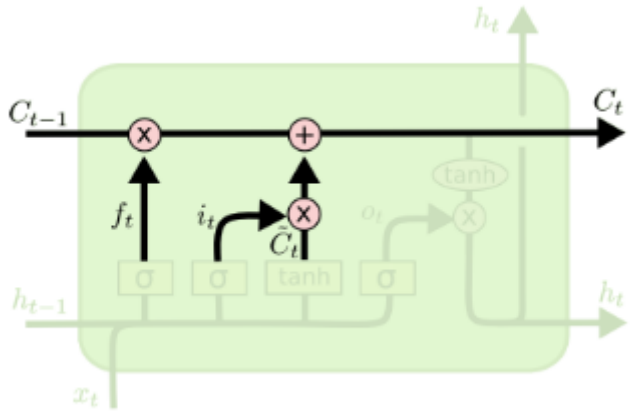

Figure 4.8: LSTM memory cell 6

$$C_t = f_t * C_{t-1} + i * \tilde{C}_t \quad (4.2.4)$$

We need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

For the language model example, since it just saw a subject, it might want to output information relevant to a verb, in case that's what is coming next. For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that what what follows next.

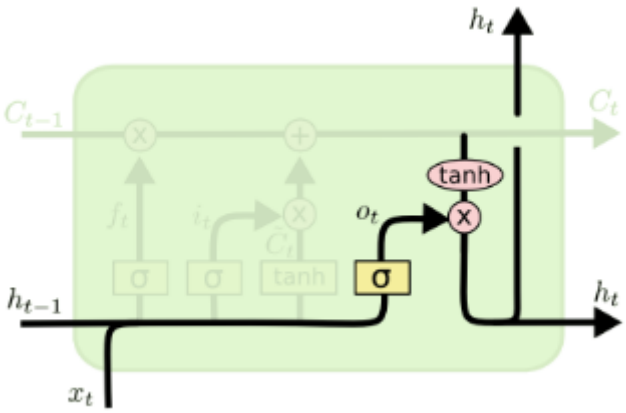

Figure 4.9: LSTM memory cell 7

$$o_t = \sigma(W_0 \cdot [h_{t-1} x_t] + b_0) \quad (4.2.5)$$

$$h_t = o_t * \tanh(C_t) \quad (4.2.6)$$

4.3. Variants on Long Short Term Memory

What I've described so far is a pretty normal LSTM. But not all LSTMs are the same as the above. In fact, it seems like almost every paper involving LSTMs uses a slightly different version. The differences are minor, but it's worth mentioning some of them. [19]

One popular LSTM variant, introduced by Gers & Schmidhuber (2000), is adding "peephole connections."This means that we let the gate layers look at the cell state.

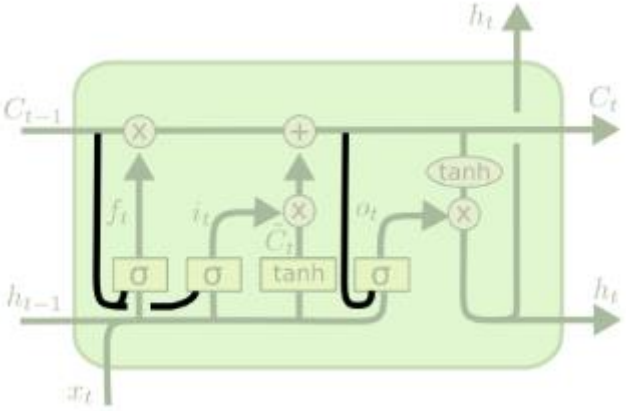

Figure 4.10: LSTM memory cell 8

$$f_t = \sigma(W_f.[C_{t-1}, h_{t-1}, x_t] + b_f) \quad (4.3.1)$$

$$i_t = \sigma(W_i.[C_{t-1}, h_{t-1}, x_t] + b_i) \quad (4.3.2)$$

$$o_t = \sigma(W_o.[C_t, h_{t-1}, x_t] + b_o) \quad (4.3.3)$$

The above diagram adds peepholes to all the gates, but many papers will give some peepholes and not others.

Another variation is to use coupled forget and input gates. Instead of separately deciding what to forget and what we should add new information to, we make those decisions together. We only forget when we're going to input something in its place. We only input new values to the state when we forget something older.

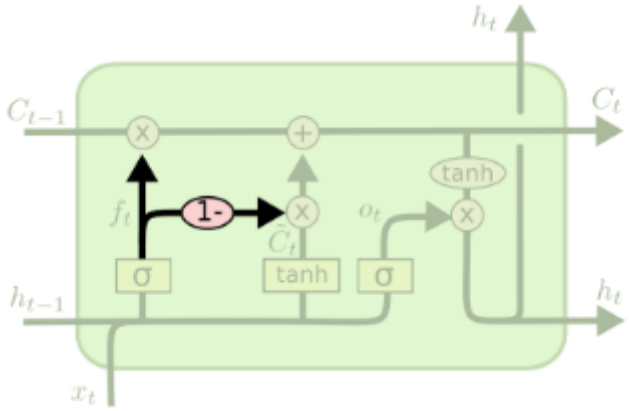

Figure 4.11: LSTM memory cell 9

$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t \quad (4.3.4)$$

A slightly more dramatic variation on the LSTM is the Gated Recurrent Unit, or GRU, introduced by Cho, et al. (2014). It combines the forget and input gates into a single "update gate." It also merges the cell state and hidden state, and makes some other changes. The resulting model is simpler than standard LSTM models, and has been growing increasingly popular.

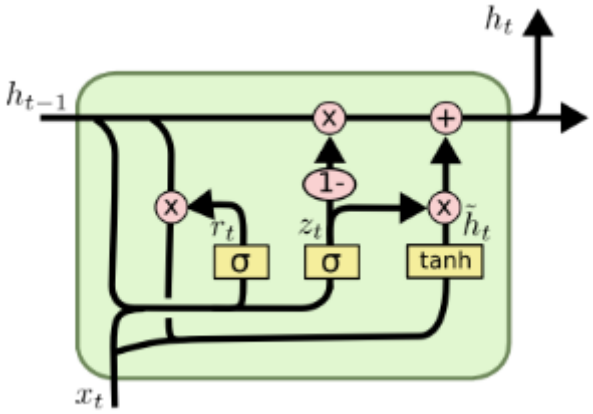

Figure 4.12: LSTM memory cell 10

$$z_t = \sigma(W_z.[h_{t-1}, x_t]) \quad (4.3.5)$$

$$r_t = \sigma(W_r.[h_{t-1}, x_t]) \quad (4.3.6)$$

$$\tilde{h}_t = \tanh(W.[r_t * h_{t-1}, x_t]) \quad (4.3.7)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \quad (4.3.8)$$

These are only a few of the most notable LSTM variants. There are lots of others.

4.4.Mean Squared Error (MSE)

Mean squared error is a single values that provides information about the goodness of fit of the regression line. The smaller the MSE values, the better the fit, as smaller values imply smaller magnitudes of error. Mean squared error can be calculated by differencing the forecasted value from the original one then taking the square of it.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2 \quad (4.4.1)$$

Mean squared error of the forecasted data is ~0.15 for trend derivative which shows that forecast is fairly accurate. We need to return back to the trend graph from the derivative graph. In order to do this, we sum up the derivative graph with the trend graph. And it gives us the forecasted trend. This forecasted trend will later multiply with the forecasted residual to achieve forecasted final value.

5.NEURAL NETWORK

5.1.Introduction to Artificial Neural Networks

Artificial Neural Networks (ANN) is an information processing technology inspired by the information processing technique of the human brain. With ANN, working of simple biological neural system is simulated. As in biological neural systems, ANN also contains neurons. These neurons form a neural network by connecting each other in various ways. These connections are called as synapses. These networks do the task of learning, memorizing and revealing the relationship between data.

In biological systems, learning is occured by adjusting the synaptic connections between neurons. This also applies to ANN. Learning is carried out using examples through education; in other words, the realization takes place by processing the input / output data, that is, by re-setting the training algorithms until a convergence of the weights of the synapses using this data is achieved.

ANNs are mathematical systems consisting of several processing units (neurons) connected in a weighted manner. A processing unit is actually an equation commonly referred to as the transfer function. This processing unit receives signals from other neurons; combines them, converts them and creates a numerical result. In general, processing units roughly correspond to real neurons and are connected to one another in a network; this structure also forms neural networks.

There are distributed, adaptive and nonlinear process concepts at the center of neural calculus. ANNs operate in a different way than traditional processors. In conventional processors, a single central processing unit performs each movement in turn. ANNs are composed of several simple process units, each of which is interested in a part of a major problem. In its simplest form, a processing unit weighs an input with a weight set, converts it non-linearly, and generates an output value. At first glance, the operation of the processing units is deceptively simple. The power of the neural computation comes from the intense connection structure between the process units that share the total processing load. In these systems, healthy learning is provided by back propagation method.

In most ANNs, neurons with similar characteristics are structured as layers and the transfer functions are run simultaneously. Almost all networks have data-receiving neurons and data-transferring neurons.

The mathematical function, which is the main element of ANN, is shaped by the architecture of the network. To be more precise, the basic structure of the function determines the size of the weights and the operation of the processing elements. The behavior of ANNs, ie how they relate the input data to the output data, is first influenced by the transfer functions of the neurons, how they are connected to each other and the weights of these connections.

5.2.Components of an Artificial Neural Network

5.2.1.Neuron

A neuron with label j receiving an input $p(t)$ from predecessor neurons consists of the following components:

- An activation $a_j(t)$, the neuron's state, depending on a discrete parameter,
- Possibly a threshold θ_j, which stays fixed unless changed by a learning function,
- An activation function f that computes the new activation at a given time $t+1$ from $a_j(t)$, θ_j and $p_j(t)$ giving rise to relation $a_j(t+1)=f(a_j(t),p_j(t),\theta_j)$,
- And an output function f_{out} computing the output from the activation $a(t)=f_{out}(a_j(t))$.

5.2.2.Synapses

The network consists of connections, each connection transferring the output of a neuron i to the input of a neuron j. In this sense i is the predecessor of j and j is the successor of i. Each connection is assigned a weight w_{ij}. Sometimes a bias term is added to the total weighted sum of inputs to serve as a threshold to shift the activation function.[20]

5.2.3. Layers

Networks generally organized into three main layers: the input layer, the hidden layer, and the output layer. Quantity of hidden layer may be more than one. But it is not true that quantity of hidden layers or number of neurons of these hidden layers directly proportional to the performance of system. There are a few theorems to determine the number and size of hidden layers. According to one of these theorems:

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
- The number of hidden neurons should be less than twice the size of the input layer.[21]

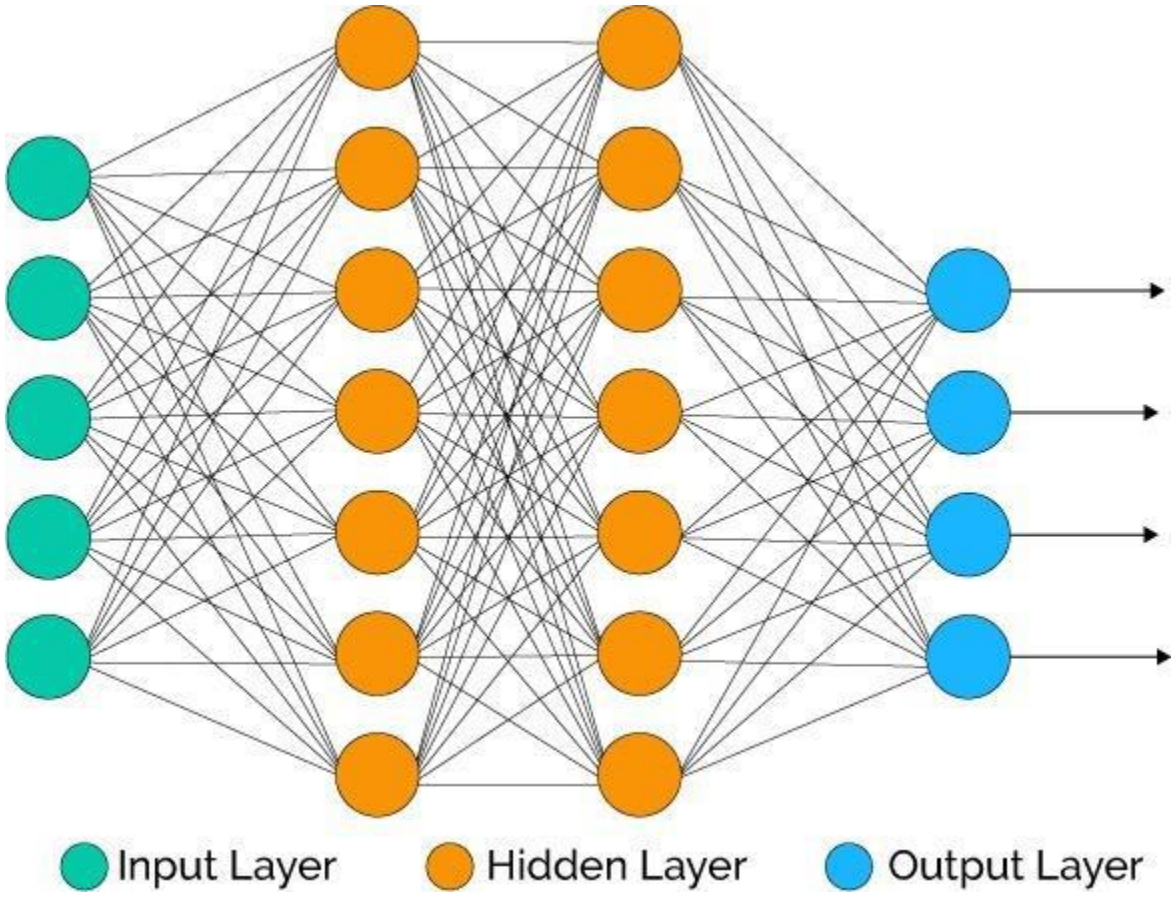

Figure 5.1: A Simple Representation Of A Neural Network

In the Figure 5.1, circles represent neurons and lines represent synapses. The role of a synapse is to take the multiply the inputs and weights. These weights can be called as strength of the connection between neurons. To do arithmetical operations, a matrix (2-D array) is created between two layers due to the assigned weights.

- $W1_{i,j}$, between input layer and hidden layer-1, where i is the number of neurons of input layer and j is the number of neurons of hidden layer-1,
- $W2_{j,k}$, between hidden layer-1 and hidden layer-2, where j is the number of neurons of hidden layer-1 and k is the number of neurons of hidden layer-2,
- $W3_{k,}$, between hidden layer-2 and output layer, where k is the number of neurons of hidden layer-2 and o is the number of neurons of output layer.

5.3.Forward Propagation

The propagation function computes the input $p(t)$ to the neuron j from the outputs $o_i(t)$ of predecessor neurons and typically has the form:

$$p_j(t) = \sum_i O_i(t) w_{ij} \quad (5.3.1)$$

When bias value is added with the function, the above form changes to following:

$$p_j(t) = \sum i O i(t) w_{ij} + w\theta j \quad (5.3.2)$$

where $w_{\theta j}$ is the bias.

After the summation, sum is applied to a activation function. Result of the activation function will be relevant neuron's value. In this project, we shall use the Sigmoid Function as are activation function.

A sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve. Often, sigmoid function refers to the special case of the logistic function shown in the first figure and defined by the formula

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \quad (5.3.3)$$

Special cases of the sigmoid function include the Gompertz curve (used in modeling systems that saturate at large values of x) and the ogee curve (used in the spillway of some dams). Sigmoid functions have domain of all real numbers, with return value monotonically increasing most often from 0 to 1 or alternatively from −1 to 1, depending on convention.
A wide variety of sigmoid functions including the logistic and hyperbolic tangent functions have been used as the activation function of artificial neurons. Sigmoid curves are also common in statistics as cumulative distribution functions (which go from 0 to 1), such as the integrals of the logistic distribution, the normal distribution, and Student's t probability density functions.[22]

To do this arithmetical operations, dot product is needed between the weight matrix which are created before and layer which is prior of the matrix. At the end of the first dot product between input layer's neuron's values and weights which are relevant, hidden layer-1's neurons will have some values. When the Sigmoid function is applied to these values, hidden layer-1's neuron's are reached. With hidden layer-1's neuron's values and W2 matrix is dot producted and the application of Sigmoid function, hidden layer-2's will have values, too. When the same implementation is done between hidden layer-2's neurons and W3 weight matrix output layer's neuron's values are reached. With the output layer's values, prediction of the output is completed.

5.4.Error Calculation

An error will occur between predicted output and the actual output as long as the prediction will not have the same values with the desired output values, which is very unlikely. It is aimed to decrease this value of loss as close as can get to 0 so the predicted output is more like the actual output. One way to represent this loss is using the Mean Sum Squared Loss Function which is mentioned before.

$$\text{Loss} = \sum (0.5)(o - y)^2 \quad (5.4.1)$$

In this function, o is predicted output, and y is actual output. As the Neural Network is trained, purpose is minimizing the loss.

5.5.Backpropagation

To figure out which direction to weights will be altered, the rate of change of the loss with respect to the weights should be found. In other words, derivative of the loss function to understand how the weights affect the input should be used. This method is known as Gradient Descent. Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. If, instead, one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as gradient ascent.[23]

By knowing which way to alter the weights, predicted outputs can only get more accurate.

Here is how the incremental change of the weights are calculated:

- Find the margin of error of the output layer (o) by taking the difference of the predicted output and the actual output (y)
- Apply the derivative of sigmoid activation function to the output layer error. this result is called as the delta output sum.
- Use the delta output sum of the output layer error to figure out how much hidden layer-2 contributed to the output error by performing a dot product with the second weight matrix. This is called as the z4 error .
- Use the delta output sum of the hidden layer-2 error to figure out how much hidden layer-1 contributed to the z4 error by performing a dot product with the first weight matrix. This is called as the z2 error .
- Calculate the delta output sum for the hidden layer-1 by applying the derivative of sigmoid activation function.
- Adjust the weights for the first layer by performing a dot product of the input layer with the hidden (z2) delta output sum. For the second weight, perform a dot product of the hidden layer-1 and the hidden (z4) delta output sum. For the third weight, perform a dot product of the hidden layer-2 and the output (o) delta output sum.

Calculating the delta output sum and then applying the derivative of the sigmoid function are very important to backpropagation. The derivative of the sigmoid, also known as sigmoid prime, will give the rate of change, or slope, of the activation function at output sum.

As doing these steps hundreds or thousands of times, loss will be very decremented and predicted output will be more close to actual output. This process is called as Training of Neural Network.

6. APPLICATION

6.1. Long Term Forecasting Application

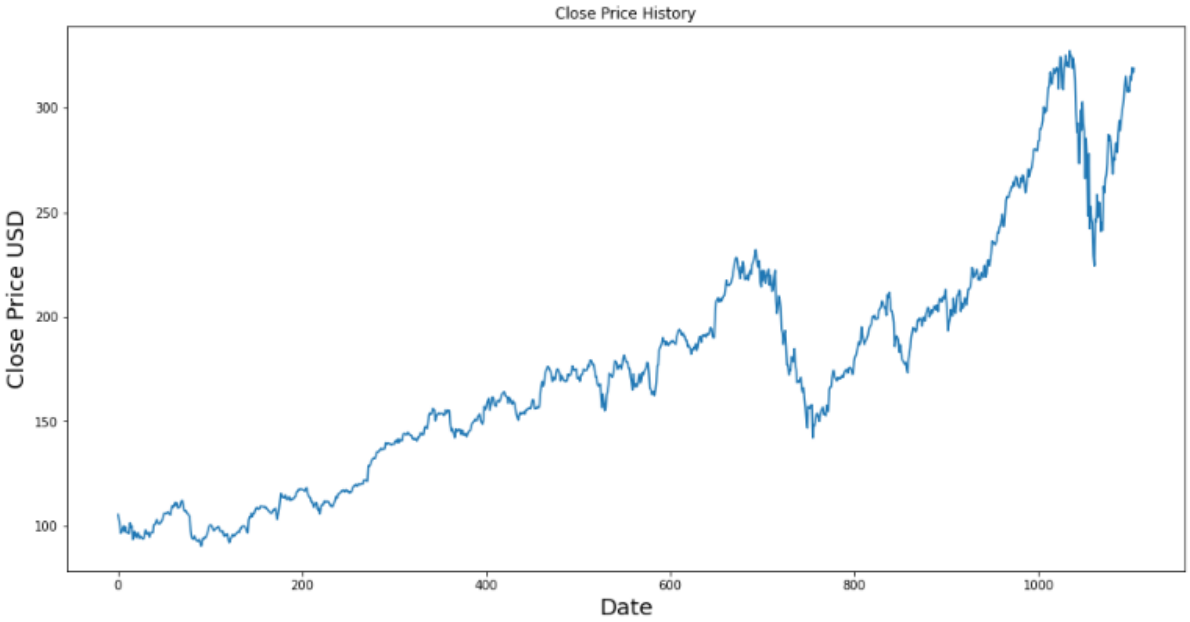

Figure 6.1: Close Price History in USD

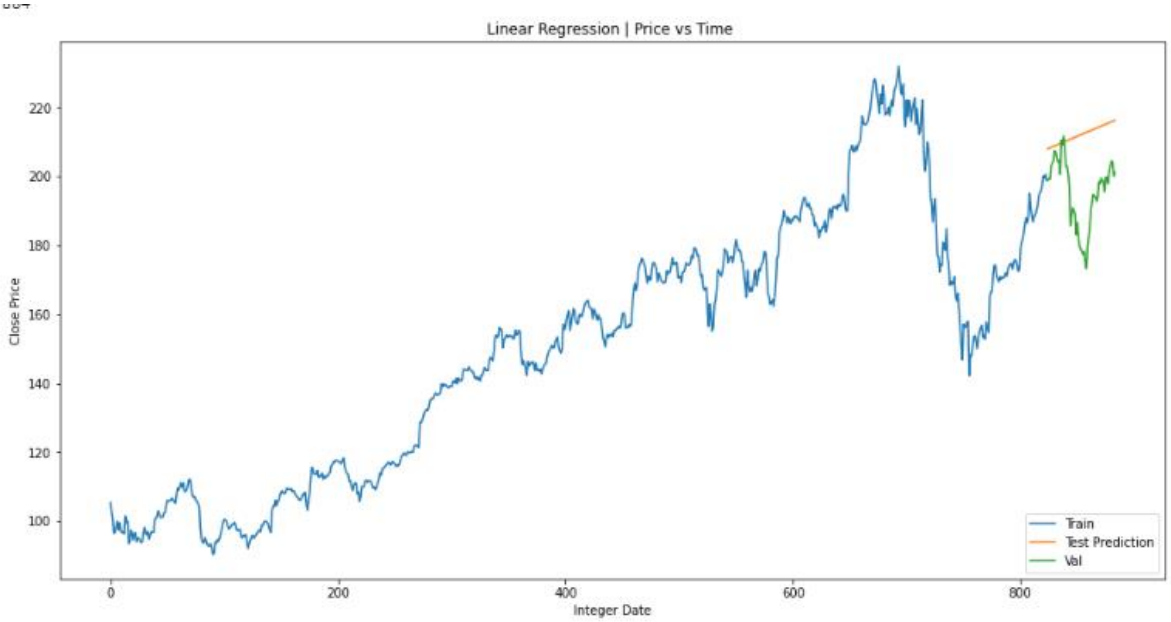

Figure 6.2: Linear Regression Application in Thesis

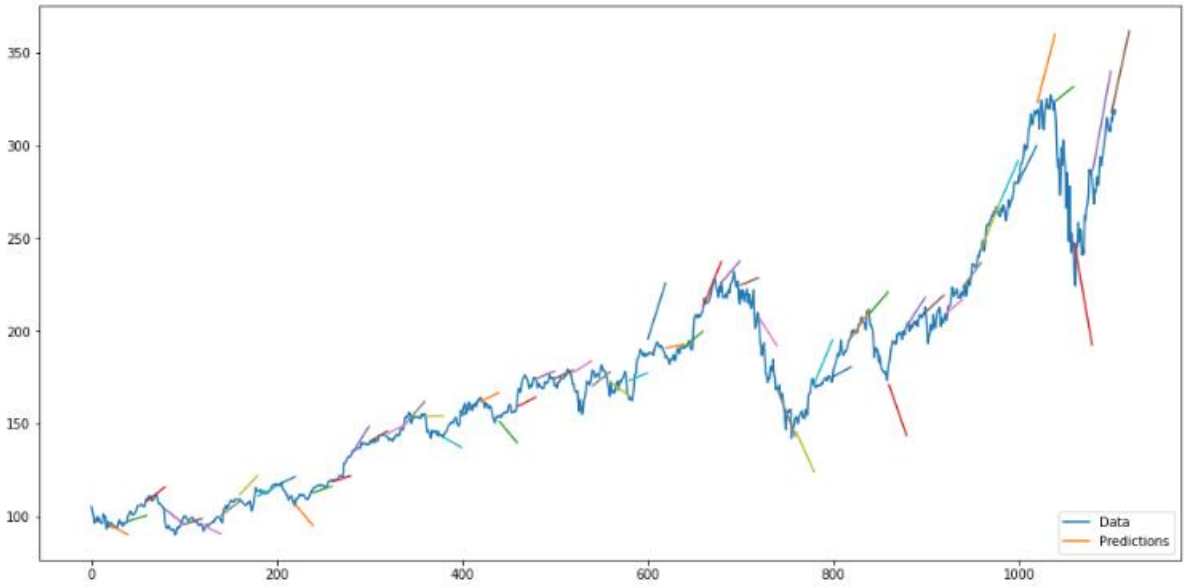

Figure 6.3: Linear Regression for Long Term

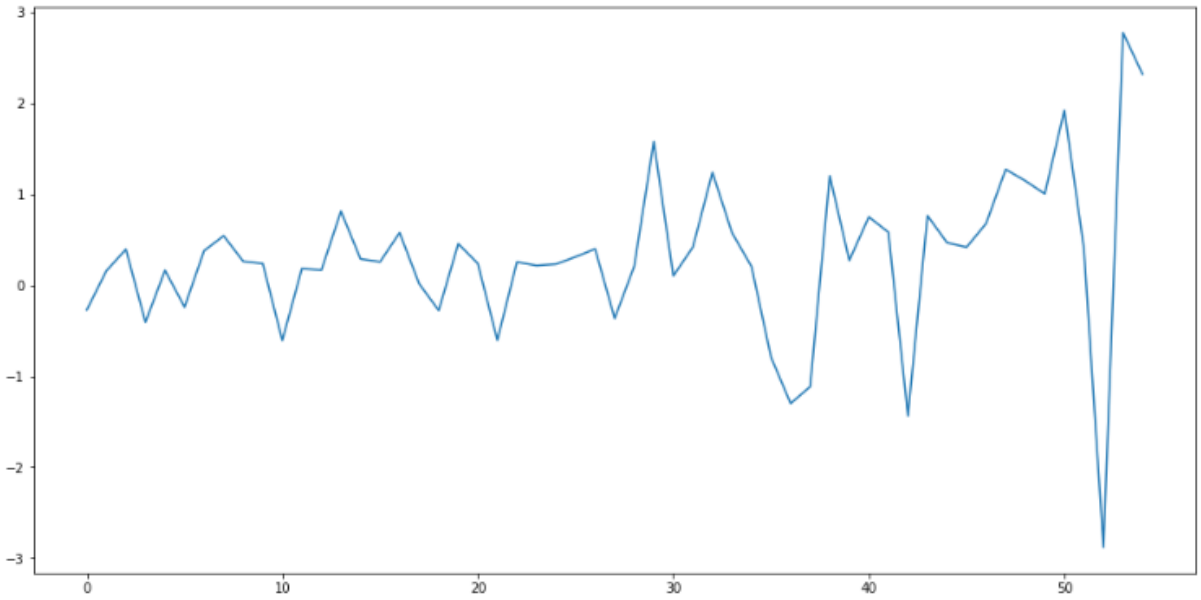

Figure 6.4: The results of Linear Regression

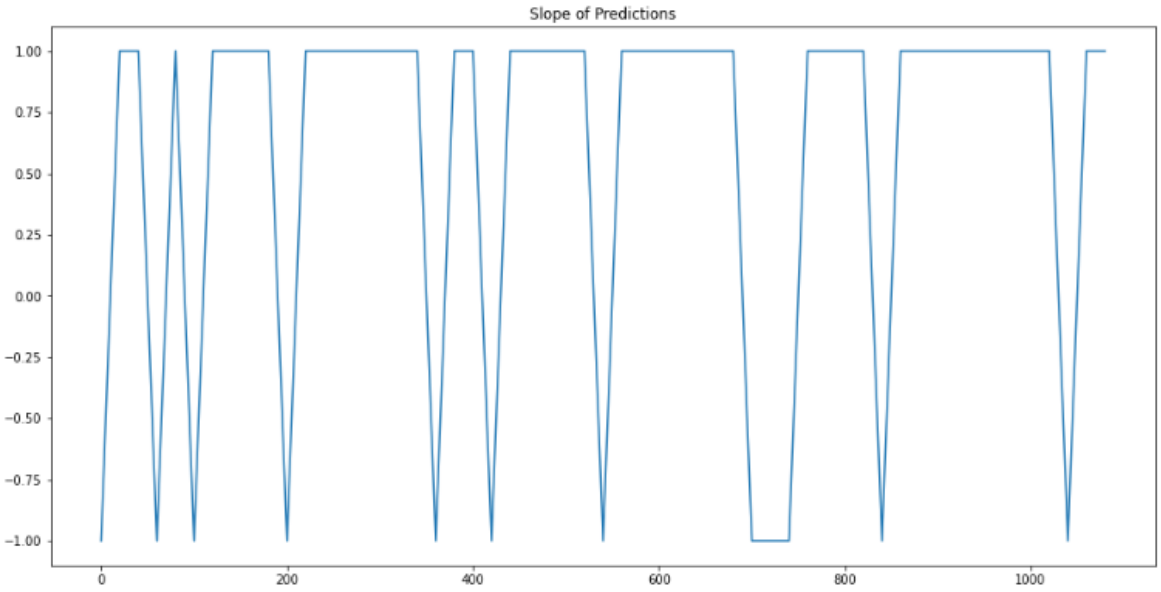

Figure 6.5: Turning Slope Values Into (-1) and 1

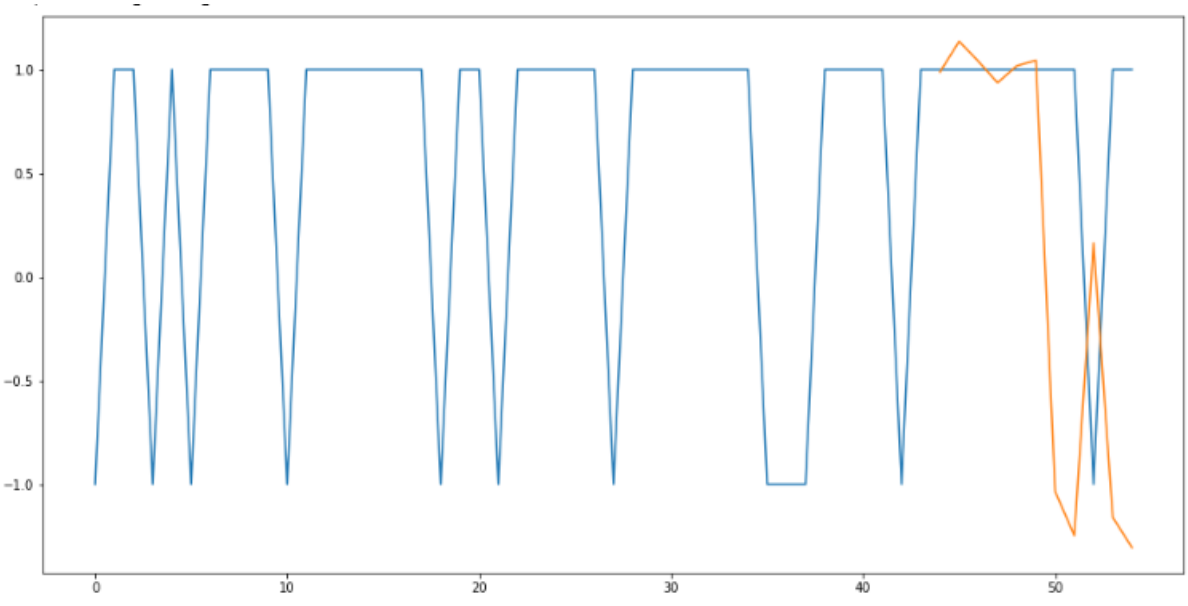

Figure 6.5: The Result of prediction with LSTM

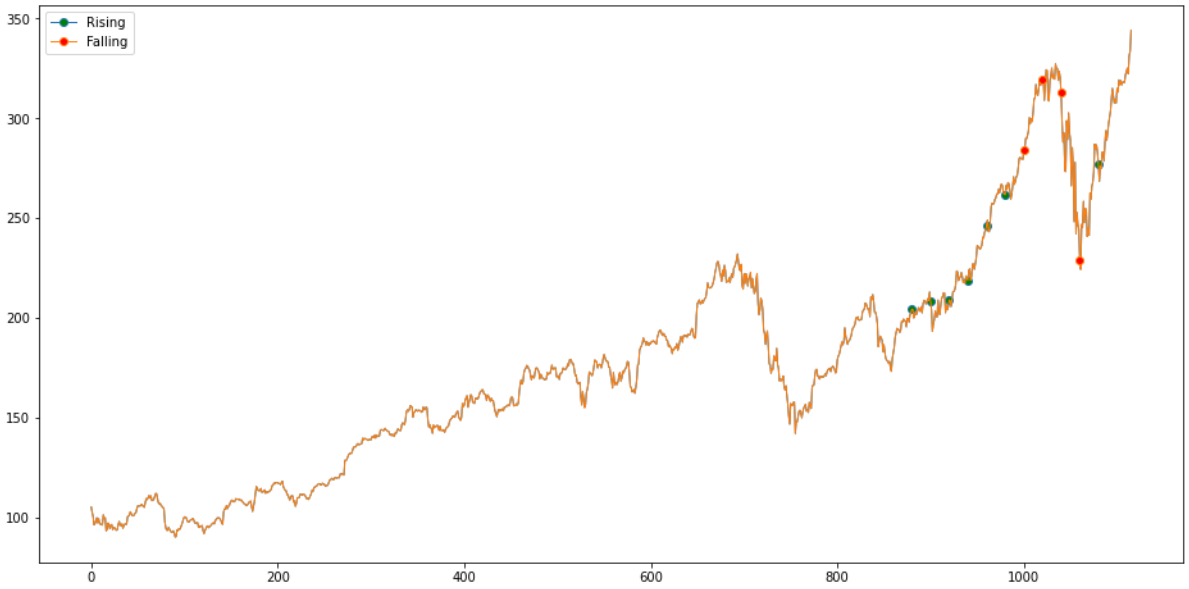

Figure 6.7: Forecast Result on Long Term

6.2. Short Term Forecasting Application

Figure 6.8: Close Price History in USD

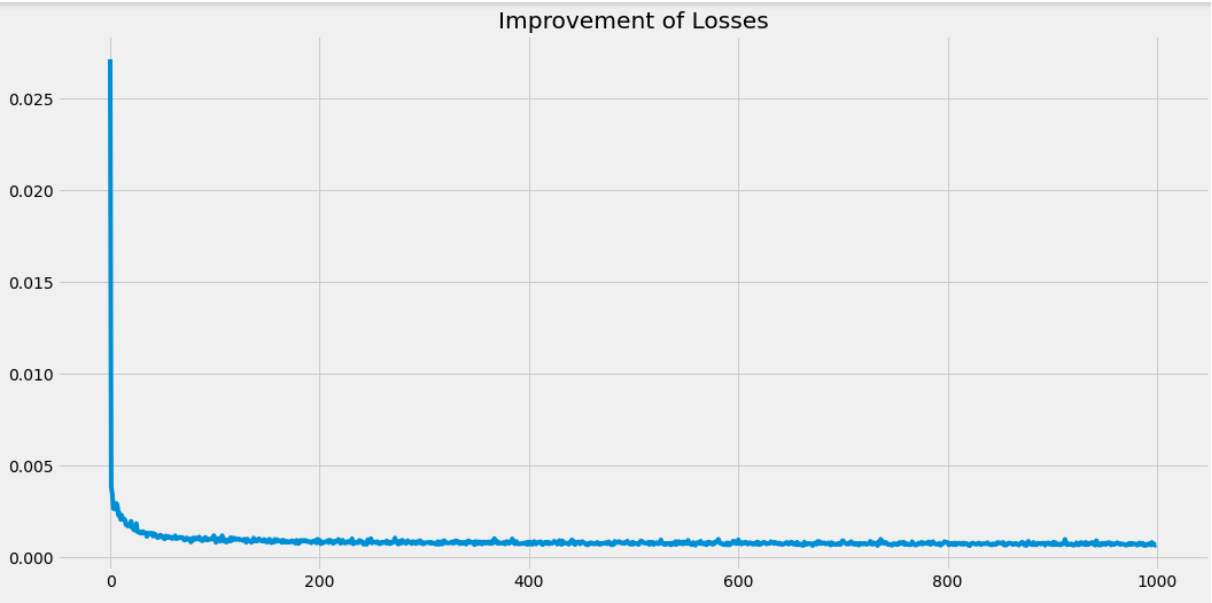

Figure 6.9: Improvment of Losses of LSTM Model

Figure 6.10: Comparing Real Data & Predicted Values

Figure 6.11: Rise & Fall -Compared to Open and Close Prices

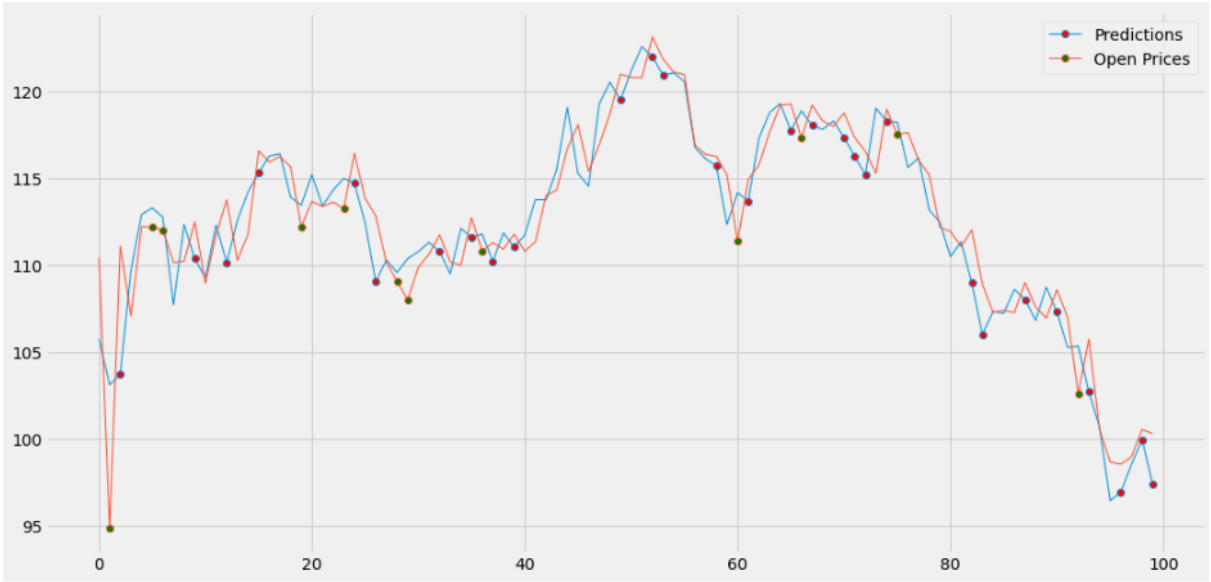

Figure 6.12: Predictions & Open Prices in Short Term

7. CONCLUSION

This study has presented the extensive procedure of LSTM model for long term and short term stock price prediction based on close prices. LSTM neural network was used for both long term prediction model and sort term prediction model. In this study, APPLE stock share was selected to train and get high accuracy rate for long term prediction and BITCOIN stock share was selected to train and get high accuracy rate for short term prediction by various modeling.

Long term prediction of stock price requires large data sets. Because the main idea for training the dataset is examine the trends of limited dataset if it is increasing or decreasing. Linear regression is a good technical way to calculate the slopes of specific range of dataset. So this long term prediction model calculates all slopes for specific range of dataset and train the data by LSTM neural network and it gives us a great opportunity to see if the trend of the data will increase or decrease.

Sort term prediction of stock price doesn't need very large data as long term prediction. The main idea is similar to long term prediction but this model trains the close prices of stock shares by LSTM neural network. After training of the dataset the objective way to control the accuracy rate is compare it to next day's open prices. This models accuracy rate is not as much as the other model but it satisfies the most of the predictions.

We can say this study shows us predicting the stock shares is possible but there are too many factors that cause fluctuation of the stock market. Most of them can not predictable at all. For now our models just based on the numbers and mathematical calculations but in the future projects the economical parameters can be include the neural system.

8.REFERENCES

[1] Derrick Mwiti, Data and Notebook for the Stock Price Prediction Tutorial(2018), Github

[2] Page 122, Introductory Time Series with R.

[3] https://medium.com/auquan/https-medium-com-auquan-machine-learning-techniques-trading-b7120cee4f05

[4] https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/?

[5]: https://www.investopedia.com/ask/answers/122314/what-exponential-moving-average-ema-formula-and-how-ema-calculated.asp

[6] https://www.surveygizmo.com/resources/blog/regression-analysis/

[7]http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvUmVncmVzc2lvbl9hbmFseXNpcw

[8] https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/

[9]http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvTGluZWFyX3JlZ3Jlc3Npb24

[10]http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvTm9ubGluZWFyX3JlZ3Jlc3Npb24

[11] https://www.mathsisfun.com/data/least-squares-regression.html

[12]http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvTGVhc3Rfc3F1YXJlcw

[13]http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQXJ0aWZpY2lhbF9uZXVyYWxfbmV0d29yaw

[14] https://www.heatonresearch.com/2017/06/01/hidden-layers.html

[15]http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvU2lnbW9pZF9mdW5jdGlvbg

[16]http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvR3JhZGllbnRfZGVzY2VudA

9.APPENDICES

Long Term Forecasting

```
# -*- coding: utf-8 -*-
"""TRYLinearRegressionPrediction.ipynb

Automatically generated by Colaboratory.

Original file is located at
```

```
    https://colab.research.google.com/drive/1cooD3l7XNEwaz-
zOWbemUeUD9uvIUMoZ
"""

import pandas as pd
import numpy as np
import math
import pandas_datareader as web
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
from sklearn import metrics
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler
#########################################################################
###
#Get the stock quote
df = web.DataReader('AAPL',data_source ='yahoo',start = '2016-01-01')
print(df)

data = pd.DataFrame(df, columns=['Close'])
print(data.head())
#########################################################################
###
y = data['Close']
data_len_org = len(y)
x=[]
for i in range(0,data_len_org):
  x.append(i)

print(len(x))
xtmp =np.array(x).reshape(-1,1)
ytmp =np.array(y).reshape(-1,1)

xtmp=pd.DataFrame(xtmp)
ytmp=pd.DataFrame(ytmp)
#########################################################################
###
#Original Data
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(xtmp,ytmp)
plt.xlabel("Date",fontsize = 18)
plt.ylabel("Close Price USD", fontsize=18)
plt.show()
#########################################################################
###
#Predict the next days
print(len(y))
```

```
training_data_len = math.ceil(len(y)*.8)
print(training_data_len)

X_train =xtmp[0:training_data_len-60]
Y_train =ytmp[0:training_data_len-60]

X_test =xtmp[training_data_len-60:training_data_len]
Y_test =ytmp[training_data_len-60:training_data_len]
X_train.shape
##########################################################################
###
#Building Model
model = LinearRegression()
model.fit(X_train, Y_train)

predictions = model.predict(X_test)
##########################################################################
###
#Plot the predicted data
plt.figure(figsize = (16,8))
plt.title('Linear Regression | Price vs Time ')
plt.plot(X_train,Y_train)
plt.plot(X_test,predictions,Y_test, label='Predicted Price')
plt.legend(['Train','Test Prediction','Val'],loc='lower right')
plt.xlabel('Integer Date')
plt.ylabel('Close Price')
plt.show()
##########################################################################
###
# Split the data to test , train
data_len = len(x)
data_len_fiftydays = data_len
sample_times = 20          #The Day how often the data will be predicte
d
data_len = data_len/sample_times
data_len = int(data_len)
print(data_len,"asd")
#Adding extra 50 days for predicting last 50 days
dayarray=[]
for i in range(0,data_len):
  dayarray.append(i*sample_times)

lastfiftydays=[]
for i in range(0,data_len_org+sample_times):
  lastfiftydays.append(i)
print(len(lastfiftydays))
lastfiftydays = pd.DataFrame(lastfiftydays)

count =1
```

```
#########################################################################
###
#Predict for every 50 days.
lnmodel = LinearRegression()
plt.figure(figsize=(16,8))
plt.plot(xtmp,ytmp)
for i in dayarray:
  lnmodel.fit(xtmp[i:i+sample_times],ytmp[i:i+sample_times])
  lnprediction = lnmodel.predict(lastfiftydays[i+sample_times:i+sample_
times+sample_times])
  plt.plot(lastfiftydays[i+sample_times:i+sample_times+sample_times],ln
prediction)

  R2 = lnmodel.score(xtmp[i:i+sample_times],ytmp[i:i+sample_times])
  Slope =lnmodel.coef_
  print("R2 for ", count,". Prediction:",R2,"Slope :" , Slope)
  count=count+1

plt.legend(['Data','Predictions'],loc='lower right')
#########################################################################
###
#Get the slopes
slopemodel = LinearRegression()
slopearray = []
for i in dayarray:
  slopemodel.fit(xtmp[i:i+sample_times],ytmp[i:i+sample_times])
  lnprediction = slopemodel.predict(lastfiftydays[i+sample_times:i+samp
le_times+sample_times])
  new_slope = np.asscalar(slopemodel.coef_)
  slopearray.append(new_slope)
#########################################################################
###
#Slopes
plt.figure(figsize=(16,8))
plt.plot(slopearray)
#########################################################################
###
#Configure slopes -1 and 1
for i in range (0,len(dayarray)):
  if slopearray[i]>0:
    slopearray[i]=1
  elif slopearray[i]<0:
    slopearray[i]=-1
  elif slopearray[i]==0:
    slopearray[i]=0
#########################################################################
###
#Plot the slopes
plt.figure(figsize=(16,8))
```

```
plt.title("Slope of Predictions")
plt.plot(dayarray,slopearray)
plt.show()
##########################################################################
###
#Slope training LSTM

a_train_len =math.ceil(len(slopearray)*.8)
a_slopearray=pd.DataFrame(slopearray)
a_slopearray=a_slopearray.values
a_slopearray
##########################################################################
###
#Train and Test
a_train_data=a_slopearray[0:a_train_len,:]
a_xtrain=[]
a_ytrain=[]

for i in range(10,len(a_train_data)):
  a_xtrain.append(a_train_data[i-10:i,0])
  a_ytrain.append(a_train_data[i,0])
  if i <=11:
    print(a_xtrain)
    print(a_ytrain)

a_xtrain, a_ytrain= np.array(a_xtrain),np.array(a_ytrain)
a_xtrain=np.reshape(a_xtrain,(a_xtrain.shape[0],a_xtrain.shape[1],1))
a_xtrain.shape
##########################################################################
###
#Libraries for LSTM Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,LSTM,Dropout

#Build the LSTM Model
model=Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(a_xtrain.shape[1],
1)))
model.add(LSTM(50,return_sequences=False))
model.add(Dense(25)) #25 Neuron
model.add(Dense(1))
##########################################################################
###
model.compile(optimizer='adam',loss='mean_squared_error')
##########################################################################
###
model.fit(a_xtrain,a_ytrain,batch_size=1,epochs=100) #This epoch is num
ber of steps
```

```
#########################################################################
###
test_data = a_slopearray[a_train_len-10:,:]
a_xtest =[]
a_ytest =a_slopearray[a_train_len:,:]
for i in range(10,len(test_data)):
  a_xtest.append(test_data[i-10:i,0])

a_xtest = np.array(a_xtest)
a_xtest = np.reshape(a_xtest,(a_xtest.shape[0],a_xtest.shape[1],1))

predictions = model.predict(a_xtest)
print(predictions," , ",a_ytest)
#########################################################################
###
rmse = np.sqrt(np.mean(predictions-a_ytest)**2)
rmse
#########################################################################
###
length = xtmp[a_train_len:len(a_slopearray)]

plt.figure(figsize=(16,8))
plt.plot(a_slopearray)
plt.plot(length,predictions)
#########################################################################
###
new_samples = sample_times*length
new_samples

rising=[]
falling=[]


for i in range(0,len(predictions)):
  if predictions[i]>0:
    rising.append((i+44)*sample_times)
  elif predictions[i]<0:
    falling.append((i+44)*sample_times)
#########################################################################
###
plt.figure(figsize=(16,8))
plt.plot(xtmp,ytmp,marker='o',markevery=rising,markerfacecolor='green',
linewidth=1)
plt.plot(xtmp,ytmp,marker='o',markevery=falling,markerfacecolor='red',l
inewidth=1)
plt.legend(['Rising',"Falling"])
```

Short Term Forecasting

```
# -*- coding: utf-8 -*-
"""LSTM-FULL.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1c_P0ZFUHrx8uf4sYDjSfJQMhWtIK1vKv
"""

import math
import pandas_datareader as web
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,LSTM,Dropout
import matplotlib.pyplot as plt

plt.style.use('fivethirtyeight')
# ================================================================================
# ================================================================================
#Get the stock quote
df = web.DataReader('AAPL',data_source ='yahoo',start = '2015-01-01',end = '2019-01-01')
df.head
df.shape
# ================================================================================
# ================================================================================
#Visualize the closing price history
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date',fontsize=18)
plt.ylabel('Close Price USD' ,fontsize=18)
plt.show()
df
# ================================================================================
# ================================================================================
```

```
#Create a new data frame with only the Close column
data = df.filter(['Close'])
#Convert the dataframe to a numpy array
dataset = data.values
#Get the number of rows to train the model on
training_data_len = math.ceil(len(dataset)*.8)
training_data_len
# ================================================================================
========
# ================================================================================
========
#Scale the data
scaler=MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)
scaled_data
# ================================================================================
========
# ================================================================================
========
train_data = scaled_data
x_train = []
y_train =[]
# ================================================================================
========
# ================================================================================
========
for i in range(60,len(train_data)):
  x_train.append(train_data[i-60:i,0])
  y_train.append(train_data[i,0])
# ================================================================================
========
# ================================================================================
========
#Convert the x_train and y_train to numpy arrays
x_train , y_train = np.array(x_train),np.array(y_train)
#Reshape
x_train = np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))
x_train.shape
# ================================================================================
========
# ================================================================================
========
#Build the LSTM model
model= Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(x_train.shape[1],1
)))
model.add(Dropout(0.2))
model.add(LSTM(50,return_sequences=False))
model.add(Dropout(0.2))
```

```
model.add(Dense(25))
model.add(Dense(1))
# ================================================================================
========
# ================================================================================
========
#Compile the model
model.compile(optimizer='adam',loss='mean_squared_error')
# ================================================================================
========
# ================================================================================
========
#Train the model
history = model.fit(x_train,y_train,batch_size=32,epochs=1000)
# ================================================================================
========
# ================================================================================
========
#Plotting the losses of each epoch

print(history.history['loss'])
loss_history = history.history['loss']

plt.figure(figsize=(16,8))
plt.title("Improvement of Losses")
plt.plot(loss_history)
# ================================================================================
========
# ================================================================================
========
#Create the testing data set
#Create a new array containing scaled values from index 1348 to 1808
test_data = scaled_data
#Create the data sets x_test and y_test
x_test = []
y_test = dataset[60:len(test_data)]
for i in range(60,len(test_data)):
  x_test.append(test_data[i-60:i,0])
len(x_test)
# ================================================================================
========
# ================================================================================
========
#Convert the data to a numpy array -LSTM Models want 3D shape
x_test = np.array(x_test)
x_test = np.reshape(x_test,(x_test.shape[0] , x_test.shape[1] , 1 ))
# ================================================================================
========
```

```
# ================================================================================
========
#Get the models predicted price values (x_test)
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions) # Unscale the value
s
# ================================================================================
========
# ================================================================================
========

plt.figure(figsize=(32,24))
plt.plot(predictions,linewidth=2)
plt.plot(y_test,linewidth =2)
plt.legend(['Predictions','Real'])
plt.show()
# ================================================================================
========
# ================================================================================
========
open_prices = df.filter(['Open'])
open_prices = open_prices[60:]
len(open_prices)

open_prices = np.array(open_prices)
# ================================================================================
========
# ================================================================================
========#Total counts
count_fall=0
count_rise=0
rising=[]
falling=[]
diff_th =0.5
for i in range (0,len(y_test)):
  if (open_prices[i]-y_test[i])>diff_th:
    falling.append(i)

for i in range(0,len(y_test)):
  if (open_prices[i]-y_test[i])<-diff_th:
    rising.append(i)
#Copmparing the prices between predicted close price and next days open
 prices

rising_pred =[]
falling_pred=[]
point_rise=[]
point_fall=[]
```

```
diff = 0

for i in falling:
  if (open_prices[i]-predictions[i])>diff:
    count_fall=count_fall+1
    point_fall.append(i)

for i in rising:
 if (open_prices[i]-predictions[i])<diff:
    count_rise=count_rise+1
    point_rise.append(i)
print(len(rising))
print(len(falling))

print(count_fall,count_rise)
percent_fall = count_fall/len(falling)*100
percent_rise = count_rise/len(rising)*100
print(percent_fall)
print(percent_rise)
# ================================================================================
========
# ================================================================================
========
plt.figure(figsize=(32,24))
plt.plot(y_test,linewidth=1,marker='o',markevery=point_fall,markerfacec
olor='Red')
plt.plot(open_prices,linewidth=1,marker='o',markevery=point_rise,marker
facecolor='Green')
#plt.scatter(rising,rising_pred,color='green')
#plt.scatter(falling,falling_pred,color='red')
plt.legend(['Predictions','Open Prices','Rising','Falling'])
# ================================================================================
========
# ================================================================================
========
zoom_ytest = y_test[100:200]
zoom_open = open_prices[100:200]
zoom_mark_f = [2, 9, 12, 15, 24, 26, 32, 35, 37, 39, 49, 52, 53, 58, 61
, 65, 67, 70, 71, 72, 74, 82, 83, 87, 90, 93, 96, 98, 99]
zoom_mark_r = [1,5,6,19,23,28,29,36,60,66,75,92]

plt.figure(figsize=(16,8))
plt.plot(zoom_ytest,linewidth=1,marker='o',markevery=zoom_mark_f,marker
facecolor='Red')
plt.plot(zoom_open,linewidth=1,marker='o',markevery=zoom_mark_r,markerf
acecolor='Green')
plt.legend(['Predictions','Open Prices','Rising','Falling'])
```

EEM 413 / 414

PROJECT PROPOSAL FORM

Project Title:	PREDICTION OF THE BEST BUY-SELL TIME FOR A CERTAIN STOCK-MARKET SHARE
Thematic Area:	DESIGN AND IMPLEMENTATION OF MACHINE LEARNING AND SIGNAL PROCESSING
Applying Institution:	
Project Team:	Ali Erkut COŞKUN Abidin Zeynel ÇETİN Yunus Emre KÜTÜK Baturay YAĞINLI
Supporting Industry Partner:	
Advisors:	ASSIST.PROF.DR. Mehmet FİDAN

1. Abstract

Purpose of this project is predicting stock market.The subjects were allocated each group members.Our project base on software programming and we are going to use Python and Conda Environment for this project. At the beginning we started the learning Python language after then we are going to work on appropriate algorithms and methods.Last step is learning Machine Learning algorithms and implement it to the our code.

2. Motivation and Objectives

Predicting how the stock market will perform is one of the most difficult things to do. There are so many factors involved in the prediction – physical factors vs. physhological, rational and irrational behaviour, etc. All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy.. machine learning techniques have the potential to unearth patterns and insights we didn't see before, and these can be used to make unerringly accurate predictions.

In this article, we will work with historical data about the stock prices of a publicly listed company. We will implement a mix of machine learning algorithms to predict the future stock price of this company, starting with simple algorithms like averaging and linear regression, and then move on to advanced techniques like Auto ARIMA and LSTM.

3. Novelty

The projects that have made before provide the best prediction just like we do. In addition to this our target about this project is getting more accuracy and less error rate .We want to reach at least %75 accuracy .

4. Proposed Method

1. Understanding the Problem Statement
2. Moving Average
3. Linear Regression
4. k-Nearest Neighbors
5. Auto ARIMA
6. Prophet
7. Long Short Term Memory (LSTM)

Understanding the Problem Statement

We'll dive into the implementation part of this article soon, but first it's important to

establish what we're aiming to solve. Broadly, stock market analysis is divided into two parts – Fundamental Analysis and Technical Analysis.

a)Fundamental Analysis involves analyzing the company's future profitability on the basis of its current business environment and financial performance.

b)Technical Analysis, on the other hand, includes reading the charts and using statistical figures to identify the trends in the stock market.

Moving Avarage

The predicted closing price for each day will be the average of a set of previously observed values. Instead of using the simple average, we will be using the moving average technique which uses the latest set of values for each prediction. In other words, for each subsequent step, the predicted values are taken into consideration while removing the oldest observed value from the set.

Linear Regression

The most basic machine learning algorithm that can be implemented on this data is linear regression. The linear regression model returns an equation that determines the relationship between the independent variables and the dependent variable.

k-Nearest Neihgbors

Machine Learning algorithm that one can use here is k-Nearest Neihgbors . Based on the independent variables, k-Nearest Neigbors finds the similarity between new data points and old data points.

Auto ARIMA

ARIMA is a very popular statistical method for time series forecasting. ARIMA models take into account the past values to predict the future values. There are three important parameters in ARIMA:

a) p (past values used for forecasting the next value)

b) q (past forecast errors used to predict the future values)

c) d (order of differencing)

Parameter tuning for ARIMA consumes a lot of time. So we will use auto ARIMA which automatically selects the best combination of (p,q,d) that provides the least error.

Probhet

There are a number of time series techniques that can be implemented on the stock prediction dataset, but most of these techniques require a lot of data preprocessing before fitting the model. Prophet, designed and pioneered by Facebook, is a time series forecasting library that requires no data preprocessing and is extremely simple to implement. The input for Prophet is a dataframe with two columns: date and target

Long Short Term Memory

LSTMs are widely used for sequence prediction problems and have proven to be extremely effective. The reason they work so well is because LSTM is able to store past information that is important, and forget the information that is not. LSTM has three gates:

a) The input gate: The input gate adds information to the cell state

b) The forget gate: It removes the information that is no longer required by the model

c) The output gate: Output Gate at LSTM selects the information to be shown as output

5. Design Issues

Project Requirements
Functional Requirements (hardware and software specifications, etc.) 1. Program should make forecast within 3 different time periods such as short, medium , long term. 2. Program can evaluate forecasting errors with ARIMA and LTSM 3. Problem can make predictions via Spyder and PyCharm
Usability Requirements (human factors, esthetics, ergonomic, etc.) 1. Program can use machine learning algorithms to restore the forecast data 2. Program can make predictions by eliminating trends and seasonalities
Performance Requirements (speed, efficiency, source management, scalability, etc.)
1. Program should make predictions at least %75 accuracy.

Evaluate design of the project with respect to the following aspects:
Economics:Outcome of the project will increase reliability and safety for investment.
Environment:There is not any effect in environmental view.

Sustainability:Investments can be done according to the accuracy of results
Producibility:There is no effect in productibility.
Ethics:There is no effect in ethical view.
Healthcare:There is no effect in healthacare
Security:There is no effect in security.
Social and Politics:Society can also benefit from these results when the investments are done . In terms of corporations and investors, forecasted data is important as well.

6. Project Timeline

Gantt Chart of The Project (can be attached in a separate file)

List of Work Packages		
No.	Name of The Work Package	Starting – Ending Dates
1.	Literature survey of forecasting	03/10/2019 – 16/10/2019
2.	Analysing the Stock and Market	17/10/2019 – 6/11/2019
3.	Research and Coding of Linear Regression Method	07/11/2019 – 20/11/2019
4.	Research and Coding of K-Neareset Neighbor Method	21/11/2019 – 04/12/2019
5.	Factors effecting Forecast	05/12/2019 – 18/12/2019
6.	Trends and Seasonality and how to remove it	19/12/2019 – 29/01/2019
7.	Error evoluation of forecasting	30/01/2019 – 12/02/2019
8.	Application of trends and seasonality in PyCharm and Spyder	13/02/2019 – 26/02/2019
9.	Designing the forecast algorithm for Short/Medium/Long Terms	27/02/2019 – 22/03/2020
10.	Checking the correlation of extra data (unemployement/inflation/currency rate etc.)	23/03/2020 – 31/03/2020
11.	Restoration of forecasted data with previous step’s data	23/03/2020 – 31/03/2020
12.	Testing and Comparing forecasted data and actual data	01/04/2020 – 06/04/2020

List of Intermediate Outputs		
Output	Related Work Package	Expected Date

Work Package Description Form (will be prepared for each work package)	
Work Package No:	1
Work Package Name:	Literature survey of forecasting
Starting Date:	03/10/2019
Ending Date:	16/10/2019
Work Package Manager (Supervisor?):	Mehmet Fidan
Work Package Tasks:	
Terms that will be used in the project is researched.	

Work Package Description Form (will be prepared for each work package)	
Work Package No:	2
Work Package Name:	Analysing the Stock and Market
Starting Date:	17/10/2019
Ending Date:	6/11/2019
Work Package Manager (Supervisor?):	Mehmet FİDAN
Work Package Tasks:	

Work Package Description Form (will be prepared for each work package)	
Work Package No:	3
Work Package Name:	Research and Coding of Linear Regression Method
Starting Date:	17/10/2019
Ending Date:	6/11/2019
Work Package Manager (Supervisor?):	Mehmet FİDAN
Work Package Tasks:	

Work Package Description Form (will be prepared for each work package)	
Work Package No:	4
Work Package Name:	Research and Coding of K-Neareset Neighbor Method
Starting Date:	17/10/2019
Ending Date:	6/11/2019
Work Package Manager (Supervisor?):	Mehmet FİDAN

Work Package Tasks:	

Work Package Description Form (will be prepared for each work package)	
Work Package No:	5
Work Package Name:	Factors effecting Forecast
Starting Date:	07/11/2019
Ending Date:	20/11/2019
Work Package Manager (Supervisor?):	Mehmet FİDAN
Work Package Tasks:	
Some factors that affect the data is researched such as inflation, unemployement etc. and according to the correlations, the appropriate one is chosen to be used at forecasting	

Work Package Description Form (will be prepared for each work package)	
Work Package No:	6
Work Package Name:	Trends and Seasonality and how to remove it
Starting Date:	21/11/2019
Ending Date:	04/12/2019
Work Package Manager (Supervisor?):	Mehmet FİDAN
Work Package Tasks:	
Trends and seasonal effects are researched and extraction from the data is learnt.	

Work Package Description Form (will be prepared for each work package)	
Work Package No:	7
Work Package Name:	Error evoluation of forecasting
Starting Date:	21/11/2019
Ending Date:	04/12/2019
Work Package Manager (Supervisor?):	Mehmet FİDAN
Work Package Tasks:	
Two different error types were learnt ARIMA and LTSM. Two different forecasting methods will be done with calculated error rates. The one has less error will be chosen	

Work Package Description Form (will be prepared for each work package)	
Work Package No:	8
Work Package Name:	Application of trends and seasonality in PyCharm and Spyder
Starting Date:	05/12/2019
Ending Date:	18/12/2019
Work Package Manager (Supervisor?):	Mehmet FİDAN
Work Package Tasks:	
At Several periods, testings are done. Through the results, best period is chosen at	

forecasting.

Work Package Description Form (will be prepared for each work package)	
Work Package No:	9
Work Package Name:	Designing the forecast algorithm for Short/Medium/Long Terms
Starting Date:	19/12/2019
Ending Date:	29/01/2020
Work Package Manager (Supervisor?):	Mehmet FİDAN
Work Package Tasks:	

Work Package Description Form (will be prepared for each work package)	
Work Package No:	10
Work Package Name:	Checking the correlation of extra data (unemployement/inflation/currency rate etc.)
Starting Date:	30/01/2020
Ending Date:	12/02/2020
Work Package Manager (Supervisor?):	Mehmet Fidan
Work Package Tasks:	

Work Package Description Form (will be prepared for each work package)	
Work Package No:	11
Work Package Name:	Restoration of forecasted data with previous step's data
Starting Date:	13/02/2020 –
Ending Date:	26/02/2020
Work Package Manager (Supervisor?):	Mehmet Fidan
Work Package Tasks:	

Work Package Description Form (will be prepared for each work package)	
Work Package No:	12
Work Package Name:	Testing and Comparing forecasted data and actual data
Starting Date:	27/02/2020
Ending Date:	22/03/2020
Work Package Manager (Supervisor?):	Mehmet Fidan
Work Package Tasks:	

Department Infrastructure (How will department infrastructure be used during the project timeline?)

7. Risk Management

Issues that can be encountered during the project studies

Risk	Precautions	Despite the precautions, what is the probability of risk to occur and how much can it effect?		Back Up Plan (If the risk occurs)
		Probability (High/Med/Low)	Effect (High/Med/Low)	
Low accuracy	Comparing two different forecasting methods	Medium	High	Changing forecast method/algorithm
High error rate	Checking correlation of extra factors	High	High	Changing error algorithm

8. Conclusion

9. References

Scientific Publications
1. https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/ 2. https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/?utm_source=blog&utm_medium=stockmarketpredictionarticle 3. https://medium.com/@randerson112358/predict-stock-prices-using-python-machine-learning-53aa024da20a 4. https://becominghuman.ai/predicting-buying-behavior-using-machine-learning-a-case-study-on-sales-prospecting-part-i-3bf455486e5d 5.https://medium.com/auquan/https-medium-com-auquan-machine-learning-techniques-trading-b7120cee4f05 6.https://towardsdatascience.com/how-to-use-machine-learning-to-possibly-become-a-millionaire-predicting-the-stock-market-33861916e9c5 7. https://towardsdatascience.com/in-12-minutes-stocks-analysis-with-pandas-and-scikit-learn-a8d8a7b50ee7 8. https://towardsdatascience.com/predicting-stock-prices-with-python-ec1d0c9bece1

Registered Models and Patents
Standards and Specifications