

Stock Price Prediction Using Python for Apple

Capstone Project

Prepared by

Chandra Poudyal

**Table of Content**

1. Introduction
2. Problem Statement
3. Understanding the problem statement
4. Data and Variables
5. Predictive Analysis Methods
6. Moving Average
7. Linear Regression
8. Multivariate analysis
9. Long Short Term Memory (LSTM)
10. ARIMA Model
11. AutoTS
12. Summary and Conclusion
13. Governance Plan
14. CODES
15. Resources

**STOCK PRICE PREDICTION USING PYTHON FOR APPLE**

**1. Introduction**

The stock price refers to the current price of a unit trading in the share markets. The price of the stock going up or down is related to different factors, including change in economy, change in industries, political events, war, and environmental factors.

The stock price tells the company’s current value or market value. So the stock price represents the agreed between the seller and buyer. The increase and decrease of the share price value depends on the how many buyers or sellers in the market. If the sellers are more than buyer than the stock price may climb and if there are more buyer than seller the stock price may drop.

Key facts

* Stock price indicates its current value for buyers and sellers
* The intrinsic value may be higher or lower
* The stock investor always wants to know whether the value of his/her share going up or down.

Using the company’s published financial statement; an investor can investigate and determine the worth of the share and may attract to invest. Some of the online brokerages summaries the results using various factors and attract investor. There are several factors that lead uncertainty in the market; the stock market is highly risky for the investment or not suitable for trading of the stock so only few people are investing. The seasonal variance and steady growth are main characters the investors are focus on and have big role for making decision to invest in the stock or share market.

To address these types of problems, the time series analysis is the best tool for the forecasting the trend. The trend analysis gives adequate guidance to investor and somehow predicts the future value of the stock. The prediction of the stock price using machine learning is most famous and can give concept in details and it is the most difficult things to do. There are so many factors associated with the prediction such as rational and irrational behavior, physical and psychological factors etc. and combining these factors make share price volatile and hard to predict with accuracy. Some features such as latest announcement, quarterly revenue results, earning per share etc. are main factor for the forecasting of stock price in machine learning.

In this analysis I’m using historical data of the stock price found publicly and I am considering Apple in my study. In this analysis machine learning algorithms is used for the prediction of the future stock price. The analysis will start using simple algorithms such as averaging, linear regression and advanced techniques like ARIMA, VARMA, and finally LSTM.

**2. Problem Statement**

The time series forecasting and modeling is an important role in data analysis and extensively used by econometrics and researchers. Due to the volatility of the market the time series forecasting is popular and used by share investors. In this analysis Apple stock price is used for the future price prediction.

**3. Understanding the problem statement**

The stock price prediction is complicated and hard to predict if analysis doesn’t care about other attributes. The analysis must perform both fundamental analysis and technical analysis. The fundamental analysis is done by analyzing the future profitability on the basis of present business environment and financial status of the company. The technical analysis includes analyzing the statistical figures by identifying the trends in the stock market.

**4. Data and Variables**

The required data are downloaded from various sites including yahoo finance including US Bureau of Statistics. There are multiple variables in the dataset such as Date, open, low, high, Close, adjusted close, volume, difference in previous day closing price, different in percent, and earnings per share (EPS) for Apple Inc. Other variables are such as US bank interest rate, unemployment rate.

* The columns Open and Close represent the opening price and final closing price of the stock in particular date.
* High, Low and Adj. Close represent the maximum, minimum and adjusted price of the stock on the particular date.
* Volume is the total sum of traded turnover of the stock on the given date.
* EPS is the earning per share of the company on particular quarter.
* Difference is the share closing price difference from previous day.
* EPS, Bank Int. Rate, and unemployment rate are also taken into consideration and these data are only found in the quarterly.

There are gaps in the daily stock prices which are for the weekends and public holidays. This analysis contains data from the 2012-07-02 to 2021-09-30.

The closing price is the key indicator for the calculation of profit and loss, so closing price is used as target variable. To understand the data we can plot closing price against Date to see how the stock changes day by day.

**5. Predictive Analysis Methods**

The machine learning has potential to ease the process by analyzing large data, spotting significant pattern and producing single output which will navigate investors to make right decision. In this section the daily closing price of the stock will explore using above mentioned variables using different analyzing methods.

1. **Moving Average**

The average is most common and easy to use in our daily life. The calculation of average mark is used to determine the overall performance in past days. In this process we can use in different size of average such as weekly, monthly or based on data to be predicted.

To predict each day closing price will be average of the previously observed values in definite ways. The moving average is generated by using of latest set of values for each prediction, or each subsequent steps predicted new values by leaving old value and taking next day value, as shown in figure below.

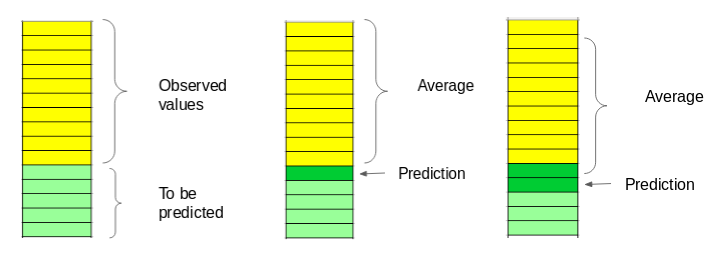


Fig. 1 Moving Average calculation for the prediction

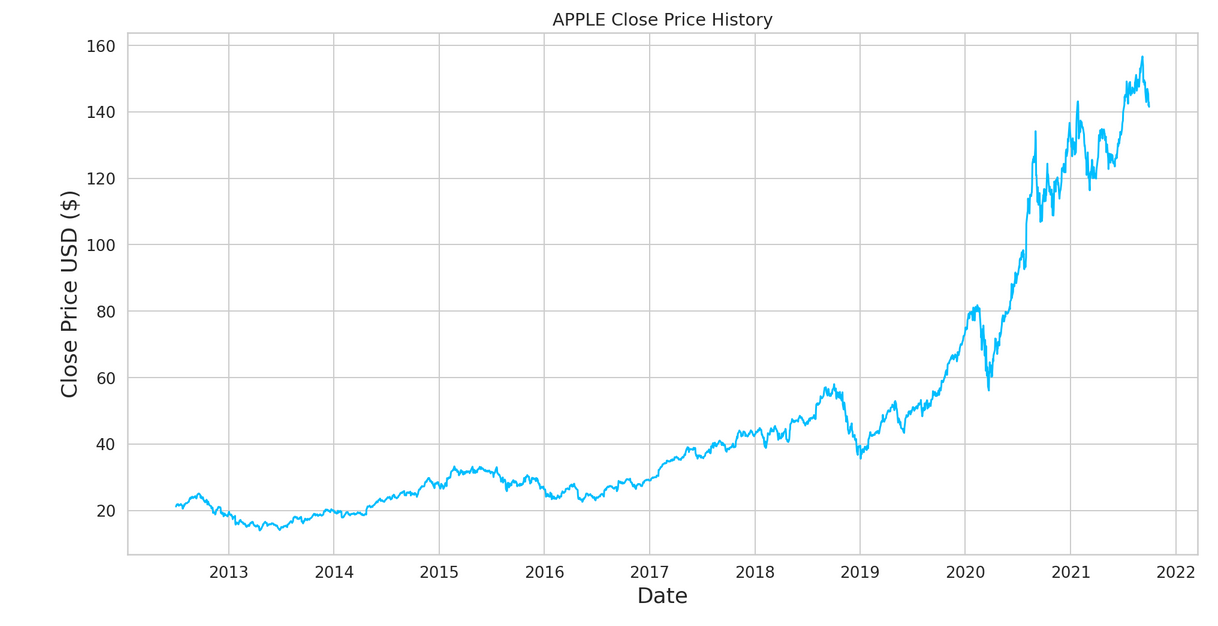


Fig. 2 Apple Price History

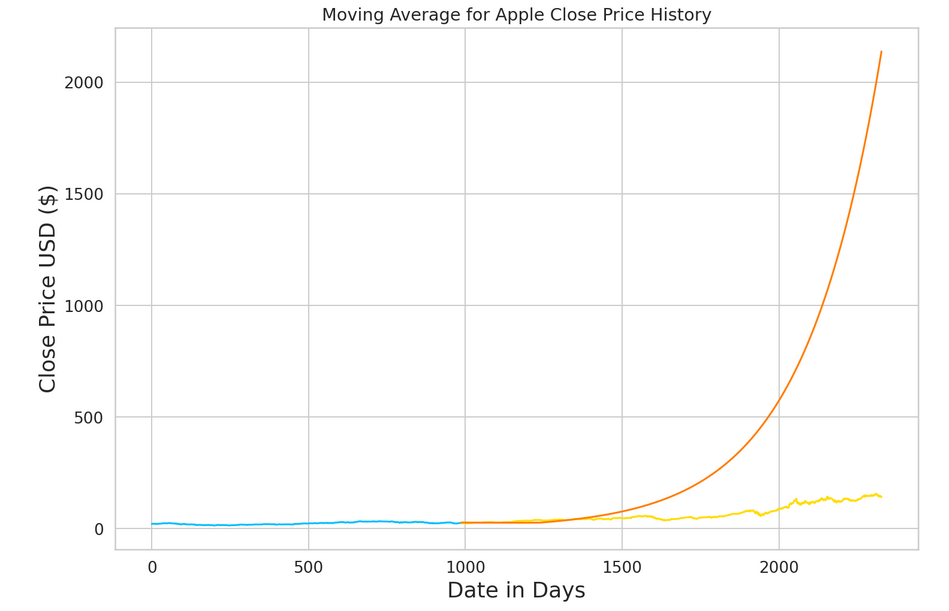


Fig: 3 Moving average and the Apple Closing Price prediction

The root mean square (RMS) value is close to 585 and the result is not promising. The predicted values are same range as the observed values in the train set but not following the trend of the actual closing price.

1. **Linear Regression**

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

The linear regression model assumes a linear relationship between the input variables(x) and the single output variable(y). On the other hand the y value can be calculated from the linear combination of the input variable x.

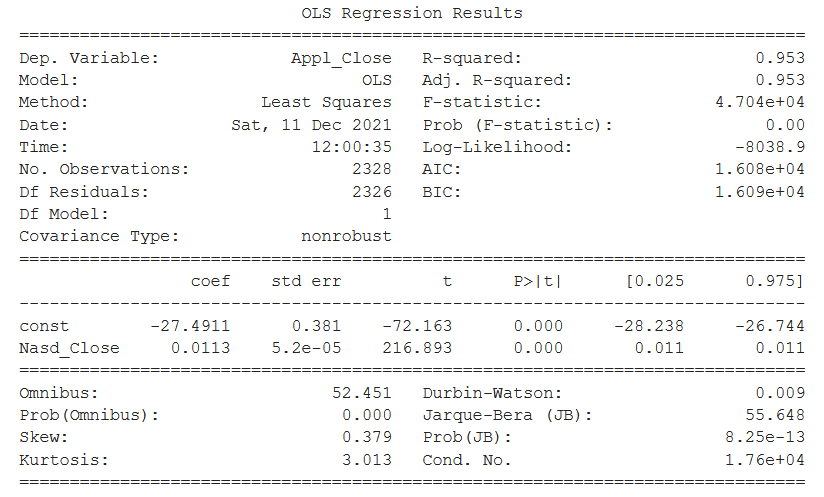


Fig. 4 The linear regression model result for the Apple stock price

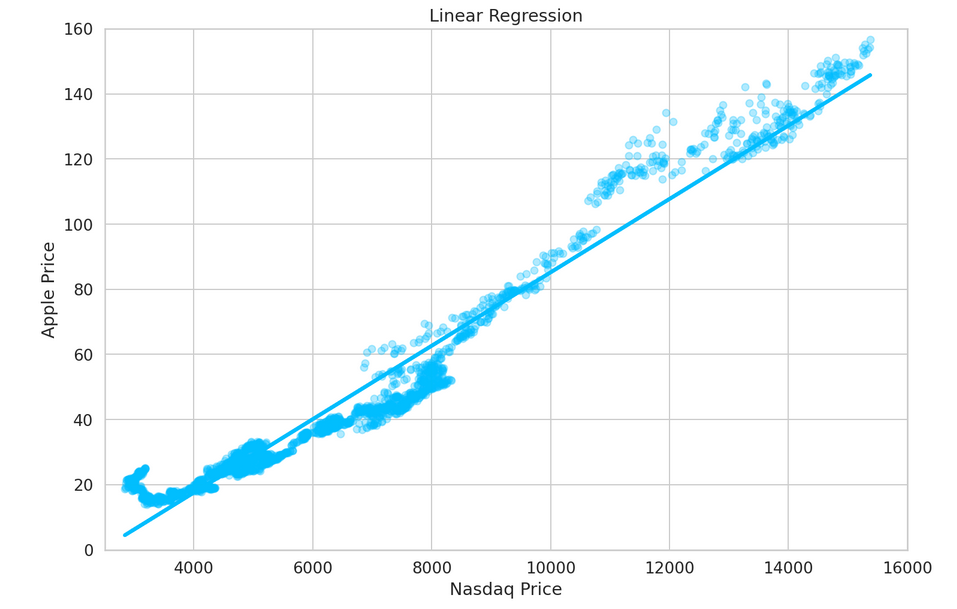


Fig. 5 The linear regression model fit for the Apple stock price and Nasdaq price

The relation between the Apple stock price and NASDAQ price shows that we can predict the Apple stock price while viewing the trend on the NASDAQ stock price.

1. **Multivariate analysis**

The Multivariate time series has more than one time-dependent variable. Some of variables are dependent of its past value but also depends on the other variables. The past dependency and other variable dependency can be used for the forecasting future values. This means our dataset includes stock price which includes opening value, low value, high value and closing value. We are predicting the closing price for the next day based on today’s value. The next day closing value is also dependent with the opening value of that day.

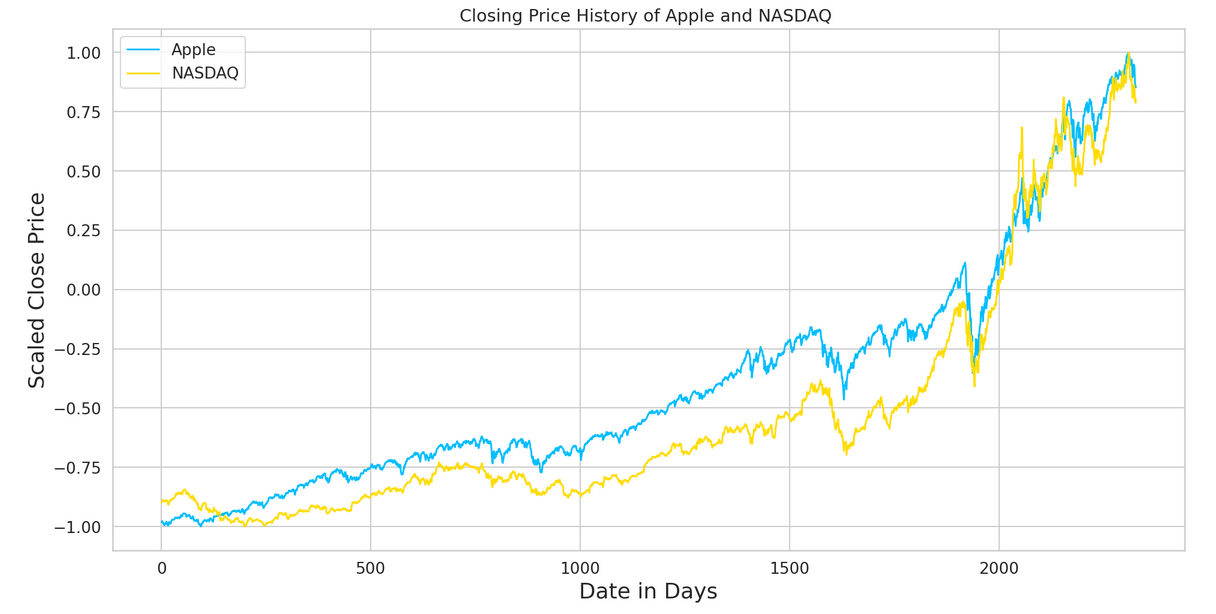


Fig. 6 The closing price increase or decrease relationship between the Apple and NASDAQ

The relationship between the Apple price and the NASDAQ shows similar growth

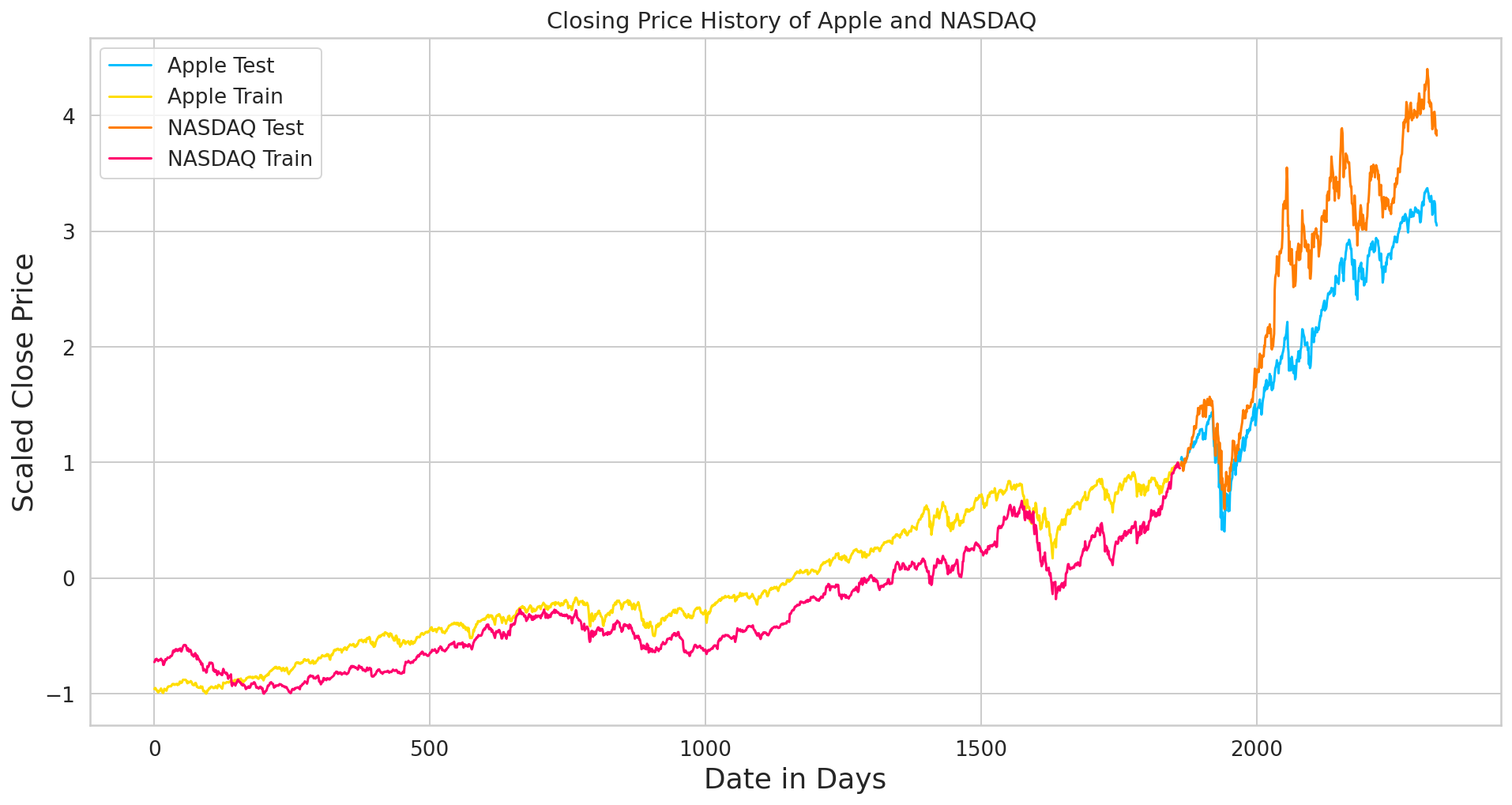


Fig. 7 Scaled test and train data for the Apple and NASDAQ

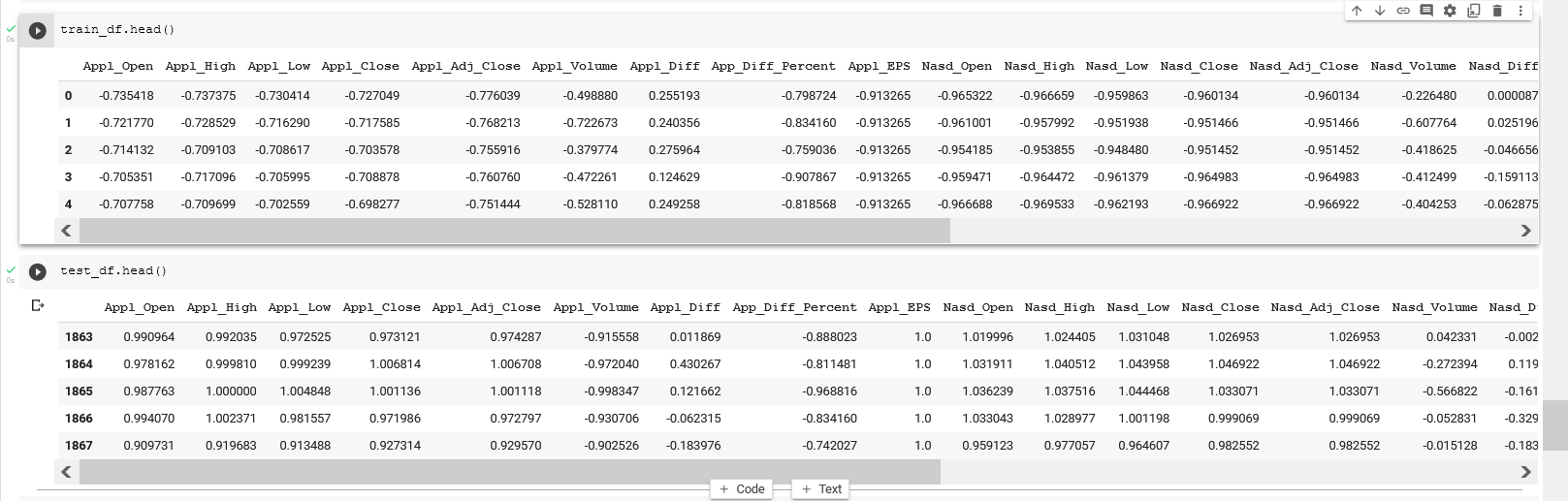


Fig: 8 Scaled data of the train and test sets

Train and test data after the scaling and plotting with the time

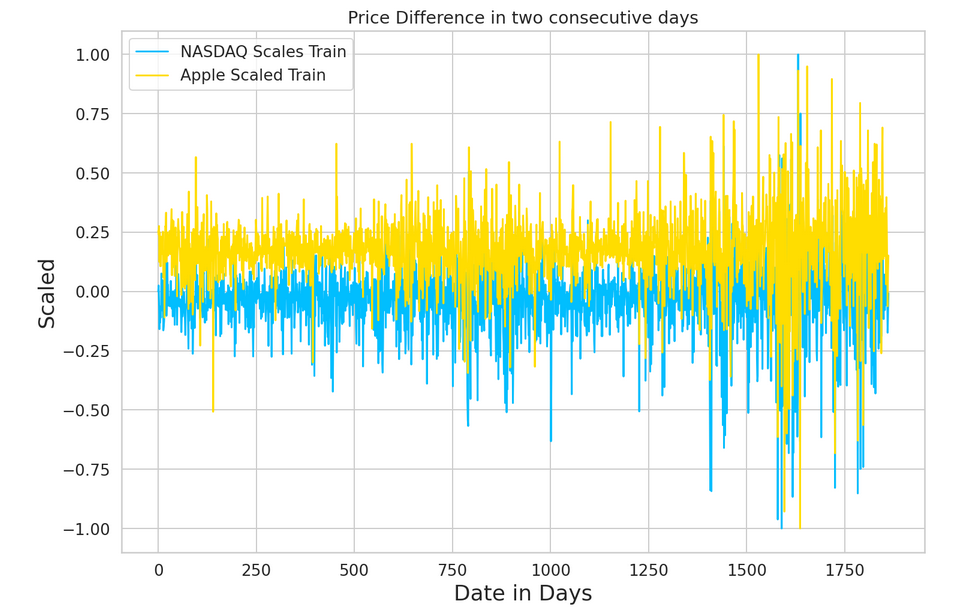


Fig. 9 The closing price difference of two consecutive days of Apple and NASDAQ

The closing price of difference between the Apple and NASDAQ is similar. Above figure shows the plotting the price difference scaled and the increasing number represents the number of days from taken from the taken data start date.

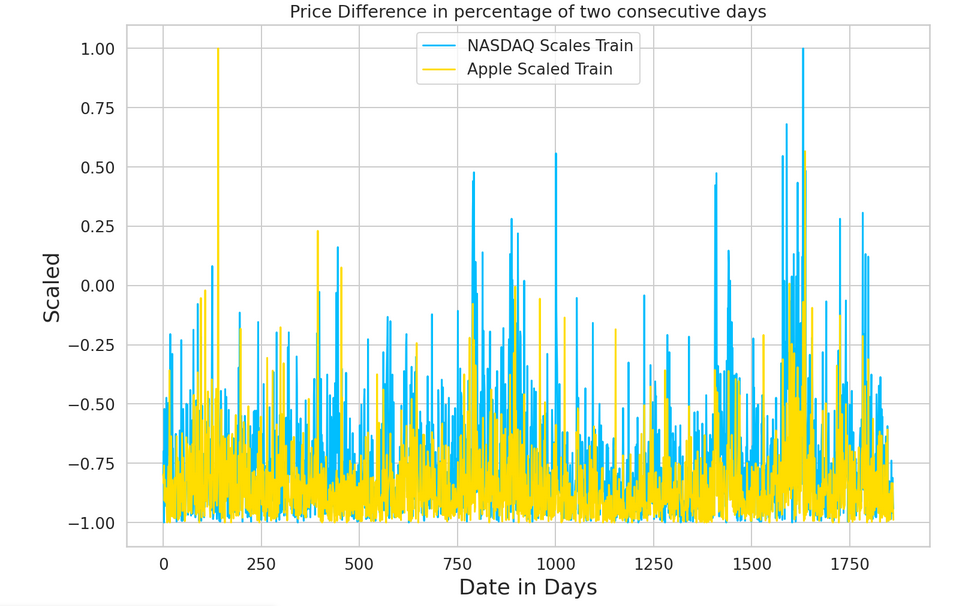


Fig. 10 The price difference in percentage

Above figure shows the trend of Apple and NASDAQ looks similar the percentage of price difference in two consecutive days.

1. **Long Short Term Memory (LSTM)**

The long short term memory (LSTM) is a powerful model for the time series data analysis. It can predict arbitrary number of steps in the future.

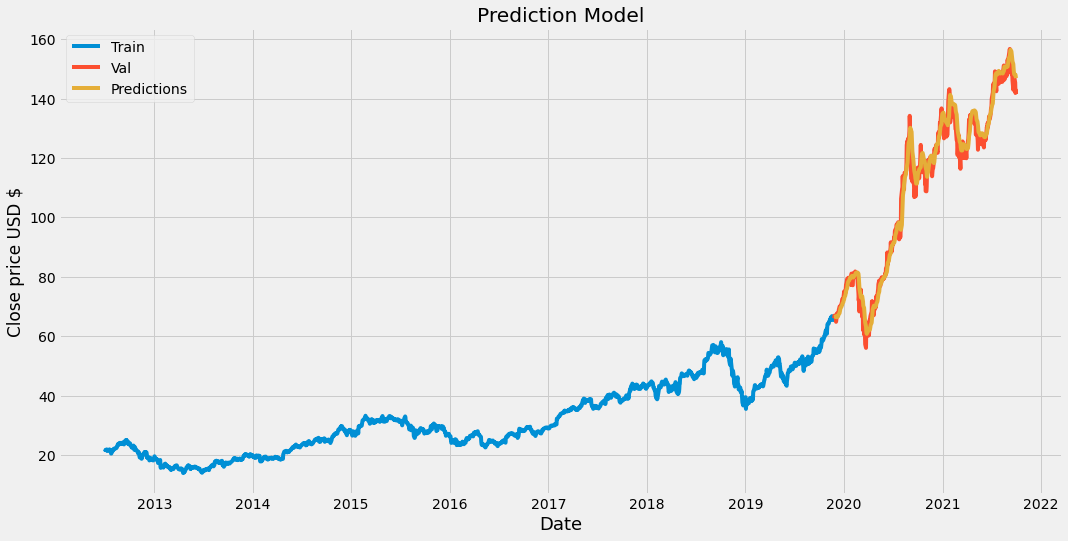
****

Fig. 11 Apple price prediction using LSTM the figure shows train data validation data and prediction data.

The root mean square value is 0.83 and looks predicted more or less correctly.

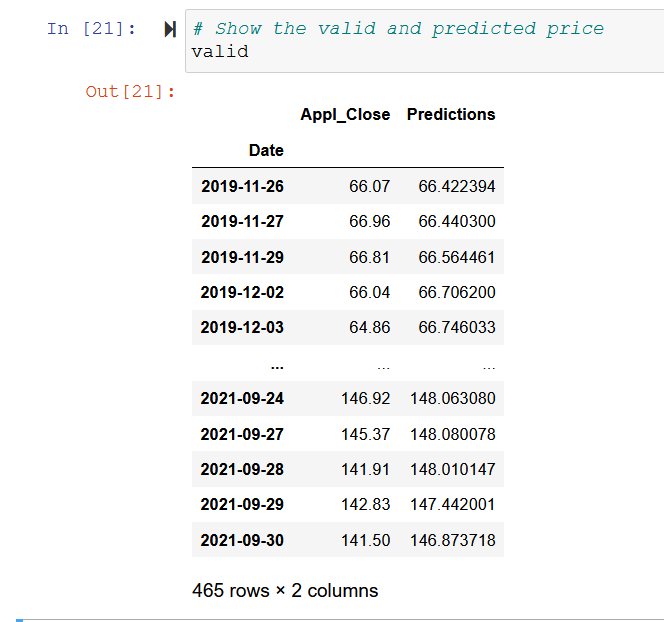


Fig. 12 the actual value and predicted value

1. **ARIMA Model**

ARIMA is also a popular statistical method forecasting of time series data. The ARIMA model take the past values to predict the future values. There are three important parameters in ARIMA:

* p (past values used for forecasting the next value)
* q (past forecast errors used to predict the future values)
* d (order of differencing)

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

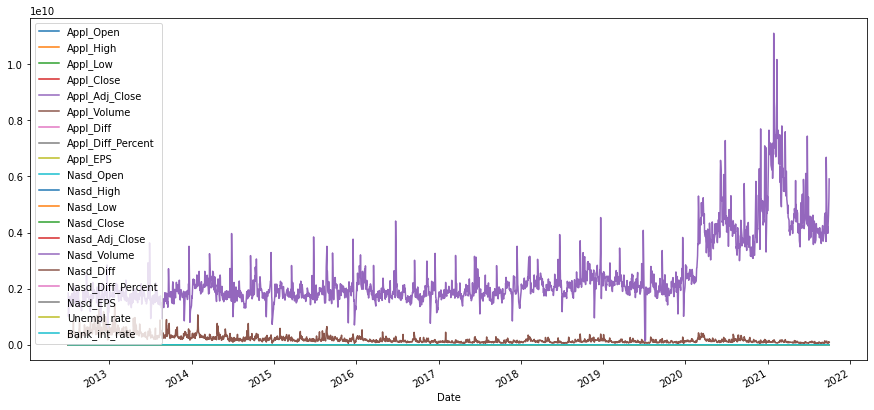


Fig. 13 Plotting of all variables

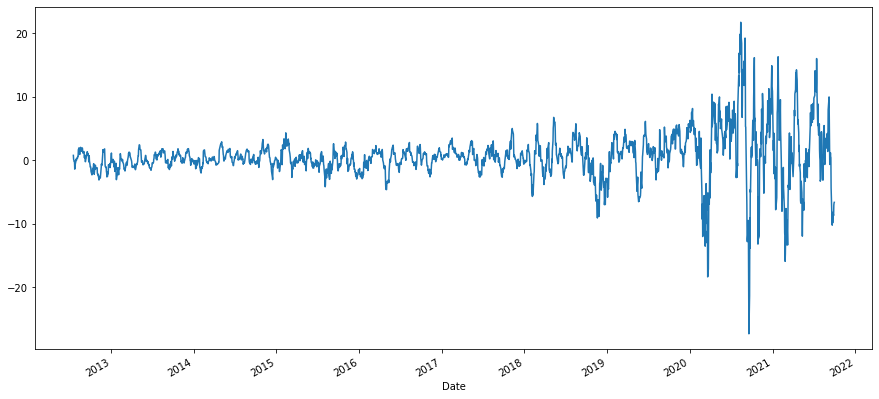


Fig. 14 The difference in stock price of Apple for two consecutive days

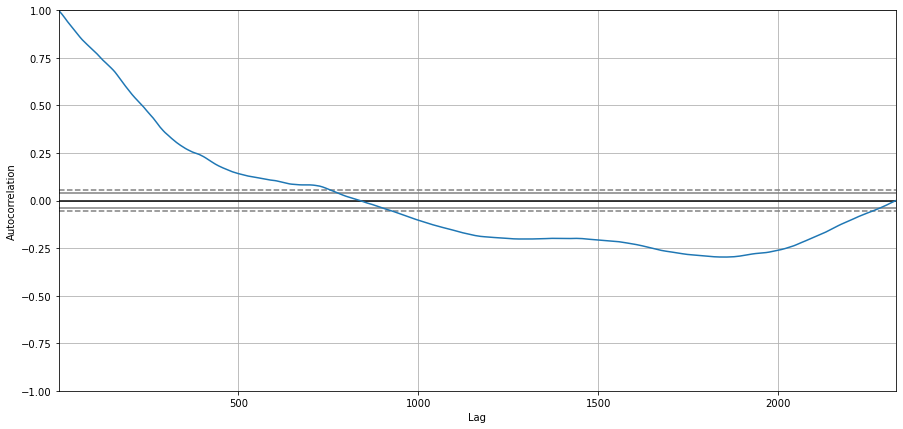


Fig. 15 The plot of Apple closing price for the autocorrelation using Lag of 1 day

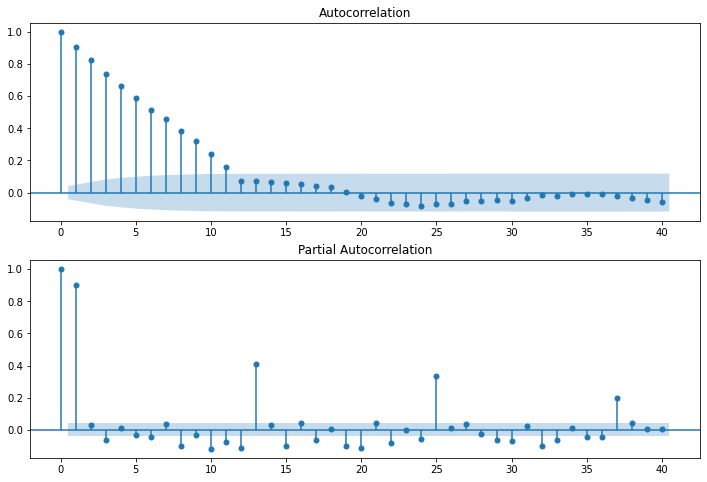


Fig. 16 Autocorrelation and partial autocorrelation for the apple closing price

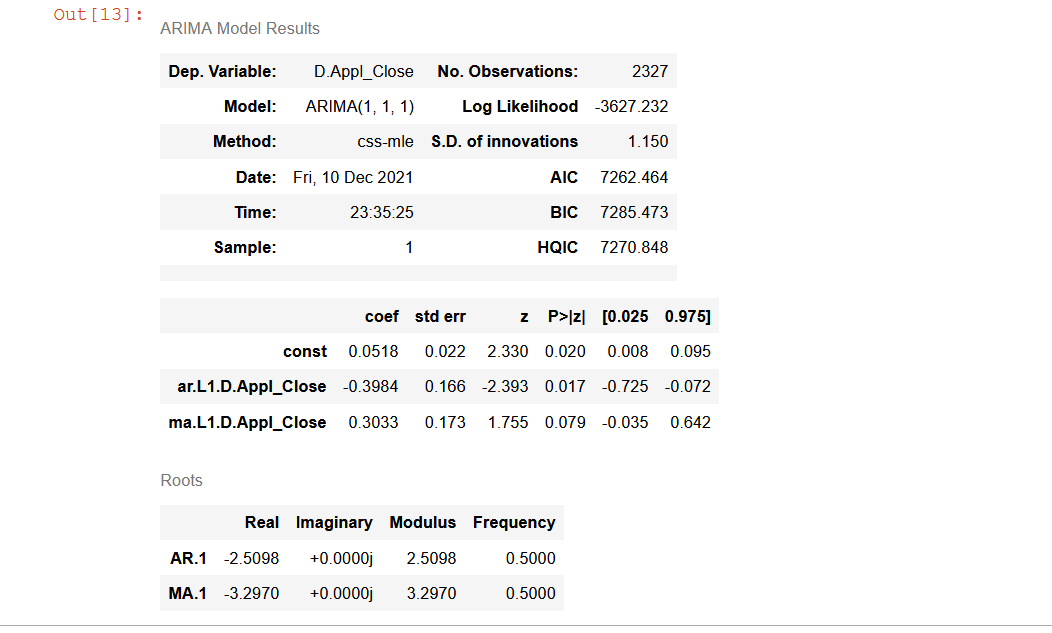


Fig. 17 Summary of ARIMA Model

1. **AutoTS**

AutoTS is a python package programming for the time series forecasting designed with deploying high-accuracy for forecasting. There are many models in use and functions directly on Pandas Dataframes. Most of the model uses user-defined exogenous regressors.

This AutoTS model is designed to find automatically the best models, preprocessing and ensembling for the given dataset through the genetic algorithms. The AutoTS process is a combination of metrics and cross-validation options and can apply regressor and simulation for forecasting.

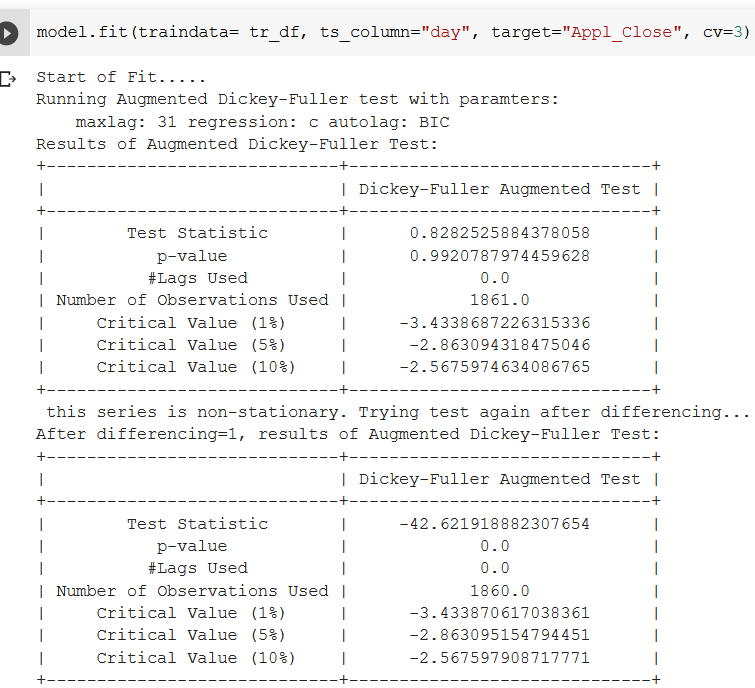


Fig. 18 Summary result for the Dickey-Fuller Test

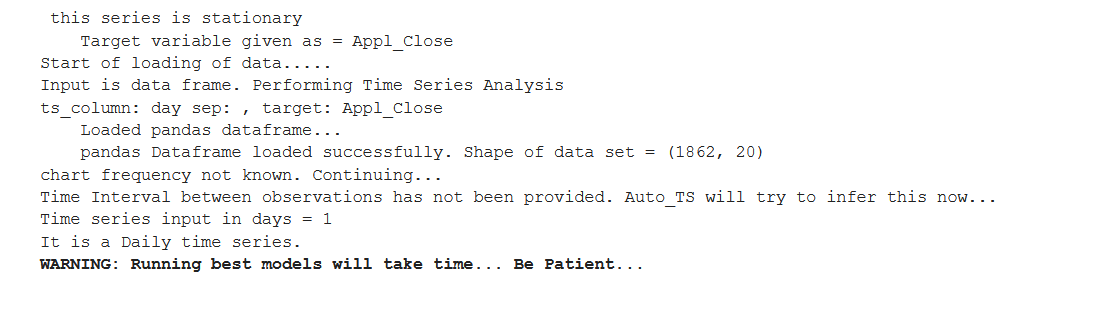


Fig. 19 Choosing Target Variable

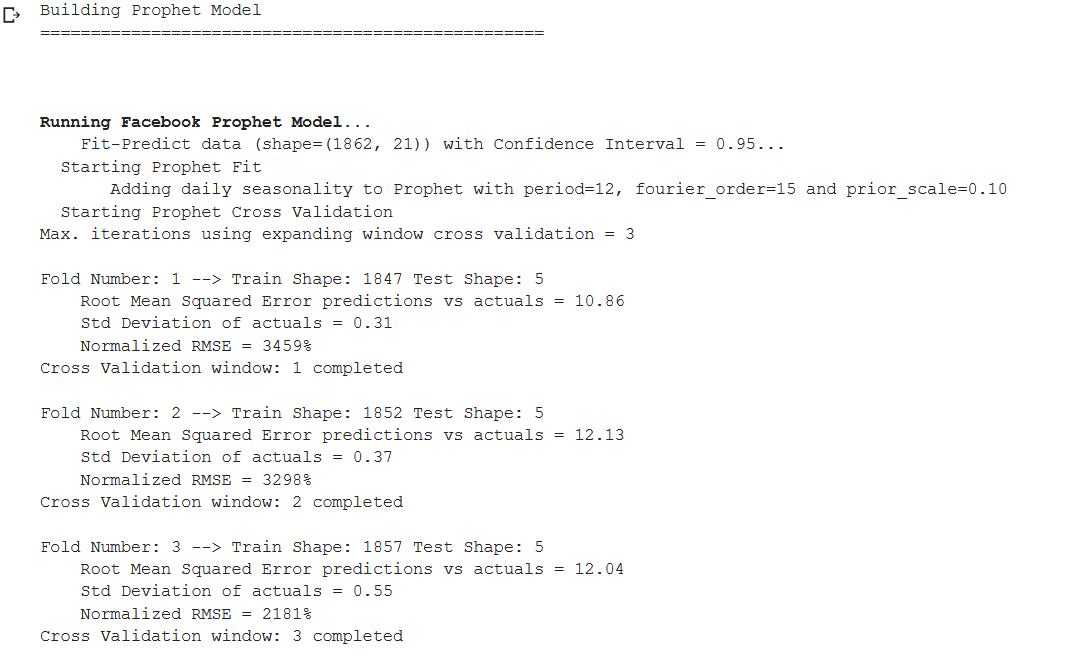


Fig. 20 The Prophet Model running

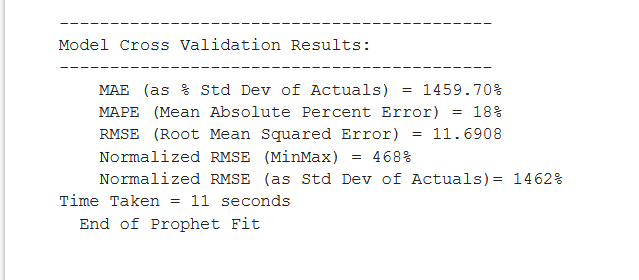


Fig. 21 Model Validation Results

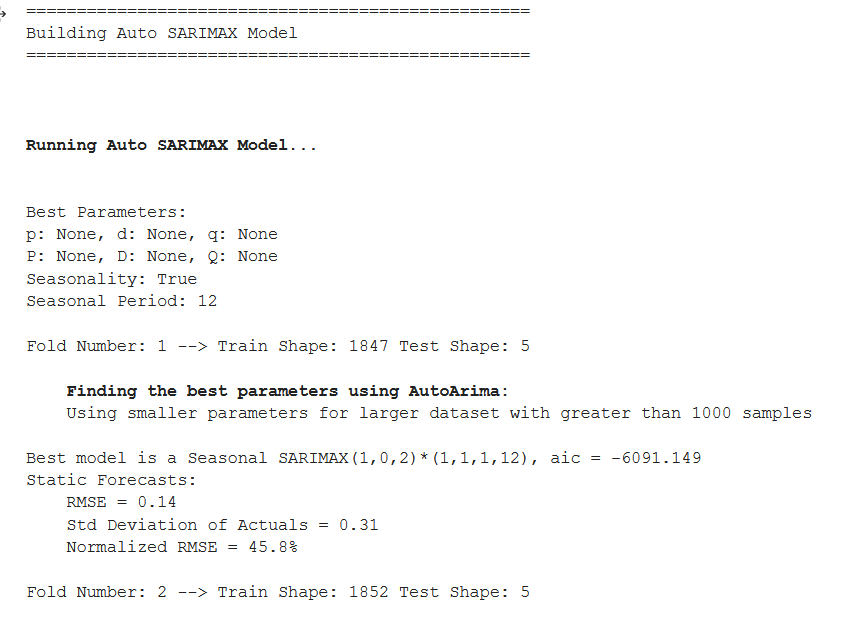


Fig. 22 Sarimax model for finding AutoArima

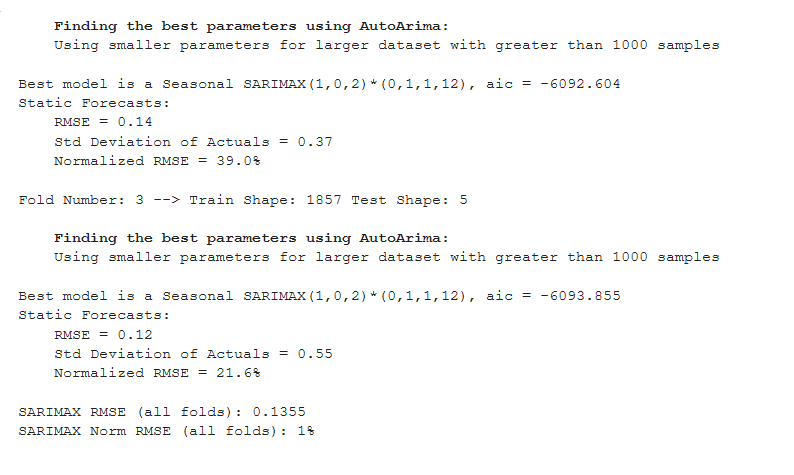


Fig. 23 Finding best parameter for AutoArima

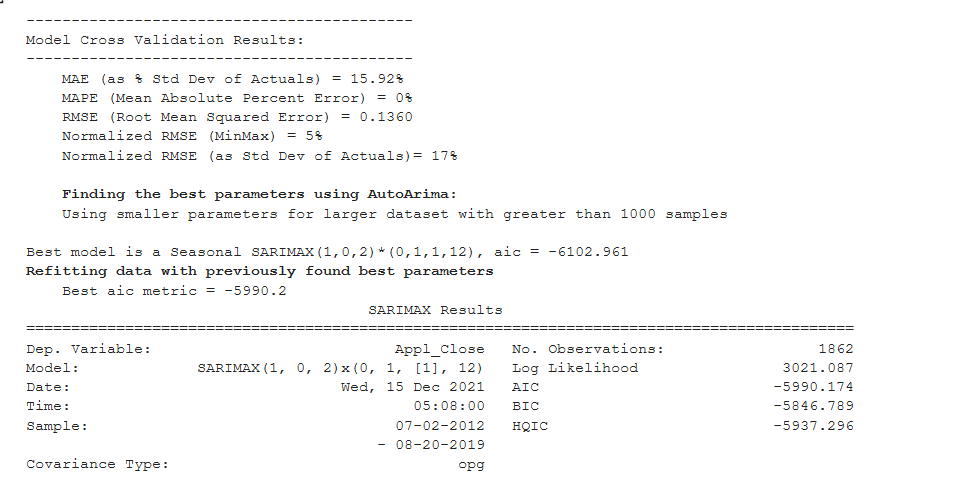


Fig. 24 Model Cross Validation Results

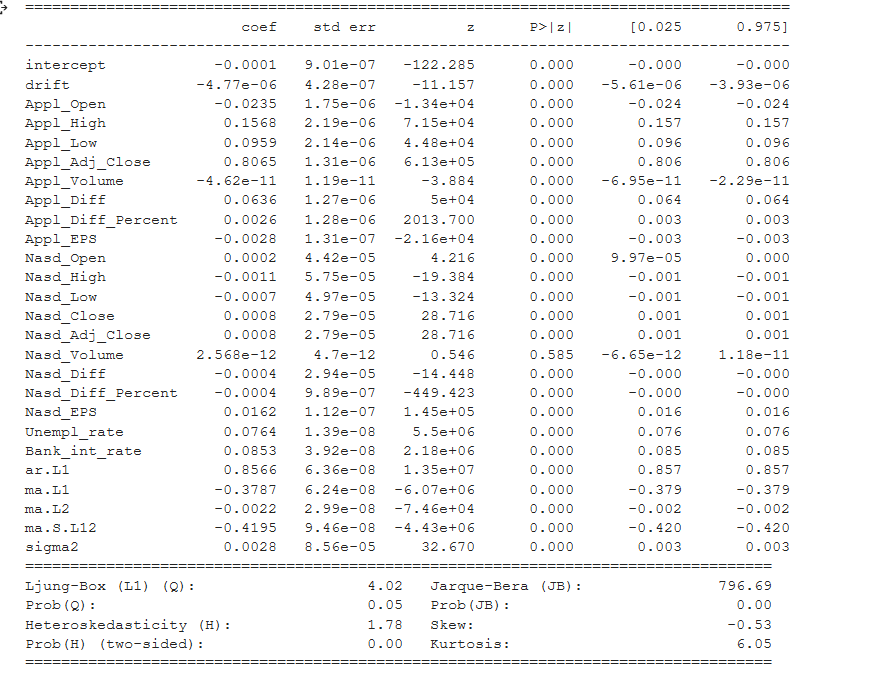


Fig. 25 Final Result of the model AutoTS

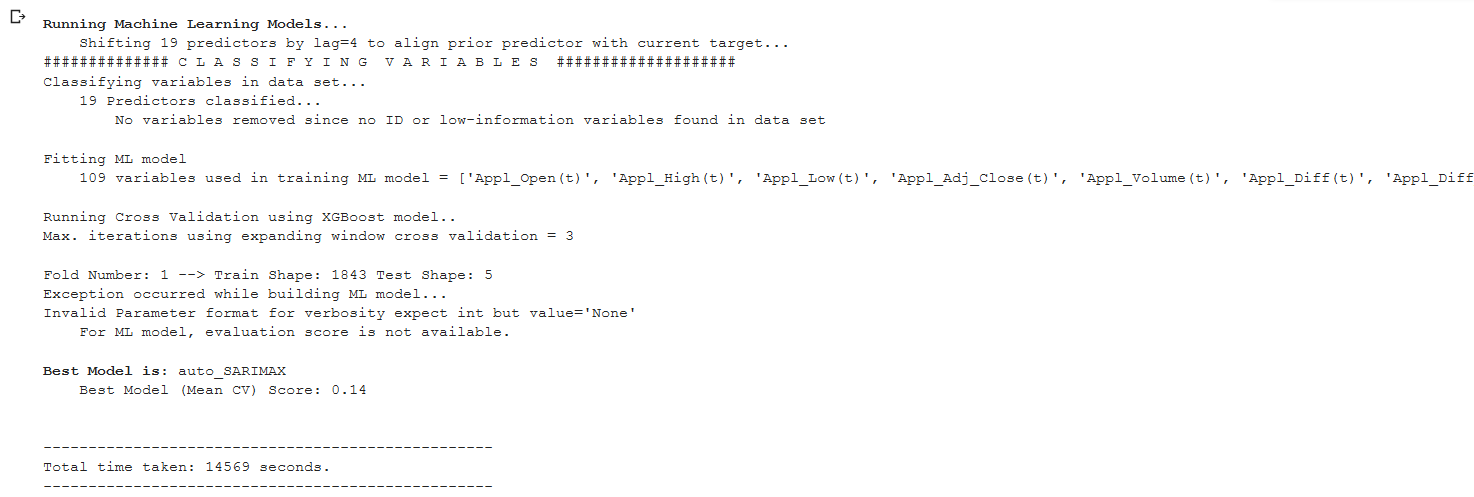


Fig. 26 Choosing best model

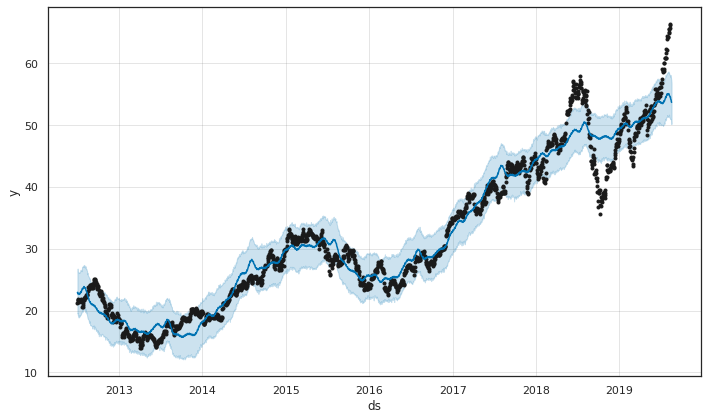


Fig. 27 Machine Learning Model using AutoTS

The machine learning model shows no proper correlation.

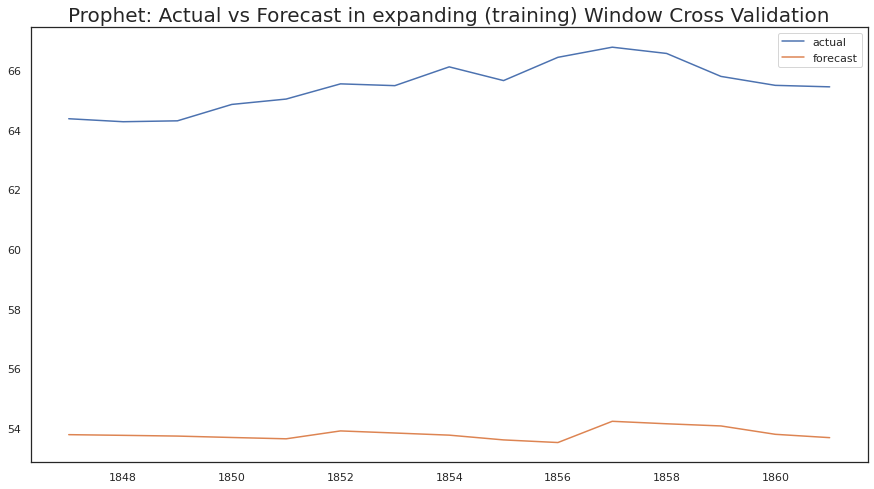


Fig. 28 The prophet model using AutoTS

This graph shows that model is not suitable for the prediction of the stock price

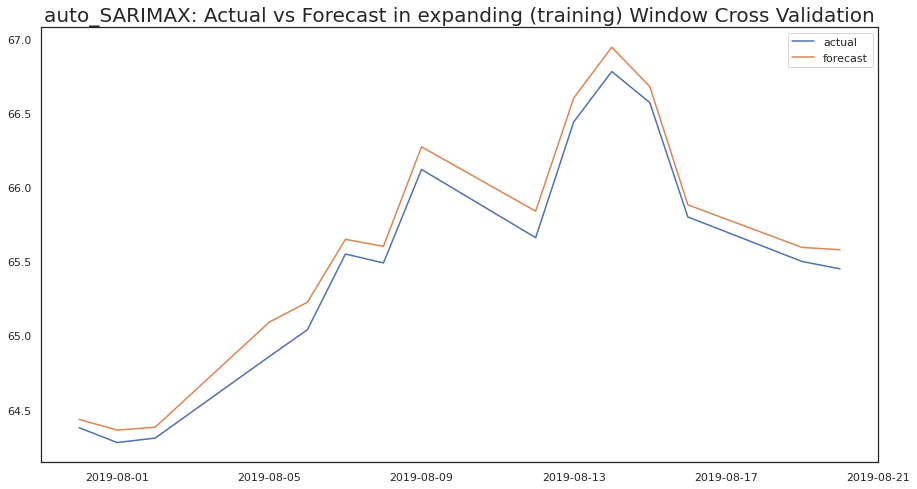


Fig. 29 Auto-SATIMAX model using AutoTS

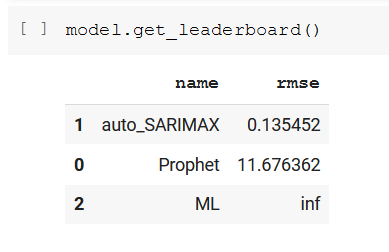


Fig. 30 Root mean square value of these models

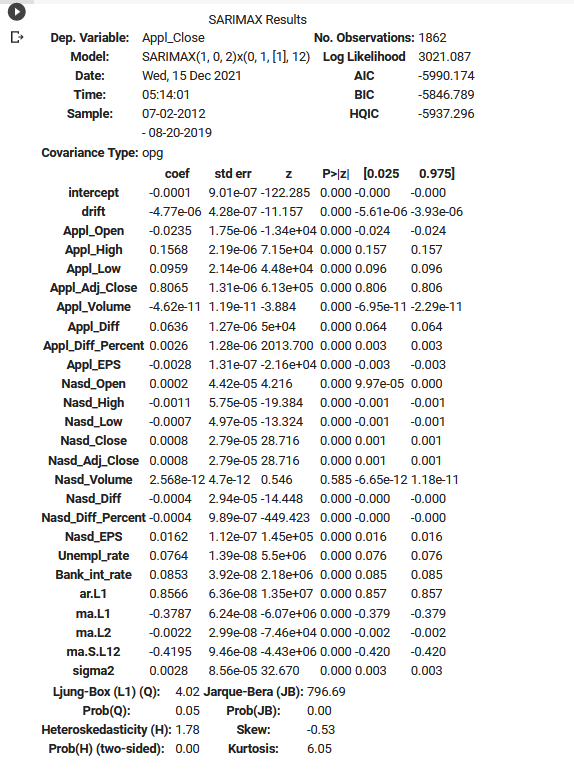


Fig. 31 Sarimax Model results

**Other plots:**

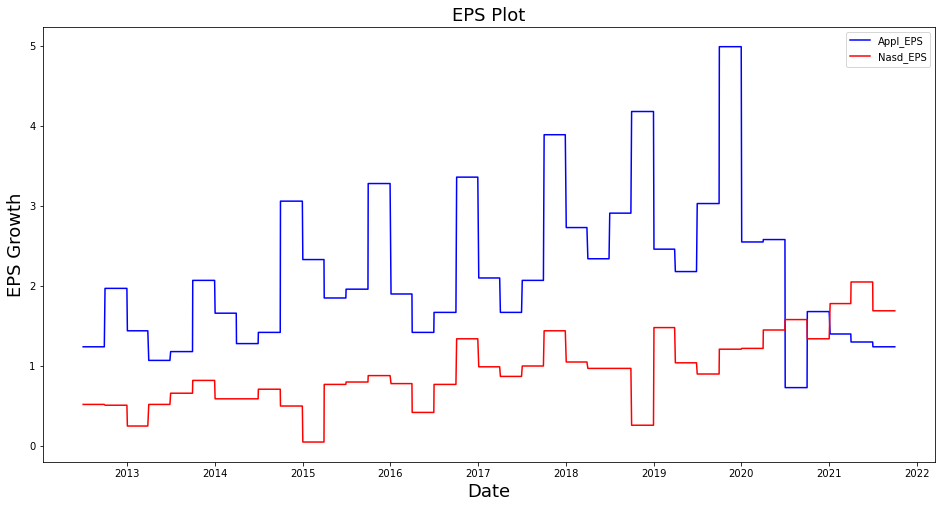


Fig. 18 Earnings per share (EPS) for Apple and NASDAQ

The plot shows there is no correlation between the EPS of apple and NASDAQ.

1. **Summary and Conclusion**

In this whole process there are several methods used to analyze the time series data of Apple stock price. For the prediction of the future price of the stock initial closed price was taken into consideration. The predictive modeling analysis includes moving average, linear regression, multivariate analysis, long short term memory (LSTM), ARIMA and using AutoTS model.

During the analysis, it is found that there is no correlation between the EPS gain of Apple and NASDAQ. Regarding the change in share price of these two company’s gives high correlation while analyzing with the linear regression model. In terms of univariate model of data prediction method for the future price of the stock the LSTM model is predicting more close to the closing price of particular date.

In case of multivariate analysis the Auto-SARIMAX is best model and has rmse 0.13, here the Auto-SARIMAX method used all the variables while calculating the prediction.

1. **Governance Plan**

Financial market analysis and prediction continues to be a fascinating and challenging problem. Nowadays, data access is becoming easier, but difficulties are increasing acquisition and processing of data to exact valuable insights and analyze their impact on the stock prices. The features that related with the financial data are extracted and used for the prediction. The quality of data is most important to predict properly. The prediction of stock price is challenging when the market condition is volatile. Last year due to COVID-19 the market down in all stocks and these types of factors can create volatility in the market situation.

The data can be affected due to various factors, such as fake news and bot data published by various sources. The challenging thing is to identify the quality of data and draw valuable insights from it.

By considering different types of problems the time series analysis is best tool for forecasting the trend of the sales. Now a day, trend analysis is performed using machine learning and can give concept in details. The market volatility makes difficult in prediction. Other features include the quarterly revenue results, earning per share etc. and these factors are major influence in the stock price.

The use of historical data of the stock price found publicly is base for the prediction and for this analysis Apple stock price is considered as an example. For the prediction of Apple stock price various methods are used and they are simple algorithms such as averaging, linear regression and advanced techniques like ARIMA, VARMA, LSTM, and AutoTS.

**Data acquisition and Variables**

The data are collected from various sources which are easily available in the internet. The stock price data downloaded from yahoo finance and some are downloaded from different websites including the US Bureau of Statistics. The stock price data includes opening price, closing price, and low for the day, high for the day, adjusted closing price and transaction volume. The earnings per share, bank interest rate and unemployment rate are also downloaded.

* The columns Open and Close represent the opening price and final closing price of the stock in particular date.
* High, Low and Adj. Close represent the maximum, minimum and adjusted price of the stock on the particular date.
* Volume is the total sum of traded turnover of the stock on the given date.
* EPS is the earning per share of the company on particular quarter.
* Difference is the share closing price difference from previous day.
* EPS, Bank Int. Rate, and unemployment rate are also taken into consideration and these data are only found in the quarterly.

The stocks are only traded in the week days so there are gaps in the daily stock. The public holidays and weekends, there is no any entry available as there were no trading on that day. This analysis contains data from the 2012-07-02 to 2021-09-30.

Due to volatility of the market, the closing price is considered as key indicator for the next day opening price and used as target variable. To understand the data we can plot closing price against Date to see how the stock changes day to day.

**There are several methods are used in this analysis:**

**Moving Average:** The averaging is most common and easy way to assume the market changes. In this process we can use in different size of average such as weekly, monthly or based on data to be predicted.

For the closing price prediction the average of previously observed values are taken. The moving average is generated by using of latest set of values for each prediction, or using lag for the previous data and each subsequent steps predicted new values by leaving old value and taking next day value just as shown in figure below.

After taking moving averaging the root mean square (RMS) value is calculated and predicted through the value how it is close to 1. If the value is close to 1 then the prediction is considered as towards accuracy or it follows the trend of the actual closing price.

**Linear Regression**

The linear regression is also common method of machine learning algorithm that can be implemented on this case. The linear regression model determines the relationship between the independent variables and the dependent variable.

The linear regression model assumes a linear relationship between the input variables(x) and the single output variable(y). On the other hand the y value can be calculated from the linear combination of the input variable x.

**Multivariate analysis**

The multivariate analysis has more than one time-dependent variable which are related to the single independent variable, in our case the closing price is an independent variable. The dependent variable is used for the prediction of independent variable or the target variable. In this case of analysis the stock price includes opening value, low value, high value and closing value. We are predicting the closing price for the next day based on today’s value.

**Long Short Term Memory (LSTM)**

The long short term memory (LSTM) is a powerful model of neural network and is capable of learning order dependence in sequence prediction problems. LSTM is a complex area of deep learning. In this method each memory cell’s internal architecture guarantees constant error and analyzing of these errors through networking which have connected blocks and called memory blocks. Each of memory contains one or more recurrently connected memory cells and three multiplicable units, the input, output and forget gates. These three units provides continuous analogous of wire, read and reset operations for the cells and can only be interact with the cells via the gates. The LSTM prediction is close to the stock closing price and promising 0.83 of rmse value and result.

**ARIMA Model**

ARIMA is an acronym of different models and is called Auto Regressive Integrated Moving Average. It can be generalize the simple combination of Auto-Regression, Moving Average and their integration.

This acronym is descriptive, capturing the key aspects of the model itself. Briefly, they are:

* **AR**: *Autoregression*. The dependent relationship between an observation and some number of lagged observations are used for calculation.
* **I**: *Integrated*. To make the time series stationary the observations are subtracted from the observations of the previous time span
* **MA**: *Moving Average*. The moving average is the dependency between an observation and a residual error from a moving average model applied to lagged observations.

The ARIMA is also one of the popular statistical methods used by many authors for the forecasting of time series data. The ARIMA model takes the past values to predict the future values. There are three important parameters in ARIMA:

* p (past values used for forecasting the next value)
* q (past forecast errors used to predict the future values)
* d (order of differencing)

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

**AutoTS**

The AutoTS is a new package of programming for the time series forecasting and used multi-variable with deploying high-accuracy for forecasting. This is complex model and used all the dependent variables for the future prediction.

This AutoTS model automatically finds the best models through the preprocessing and ensembling for the given dataset with the genetic algorithms. The AutoTS process is a combination of metrics and cross-validation options and can apply to simulate for forecasting.

**End Notes**

The time series forecasting is a very intriguing work and stock price prediction is complex and difficult to predict accurately. In this analysis the univariate and multivariate analysis have been performed. In the univariate analysis the moving average is not giving promising result whereas the linear regression is close to the trend of NASDAQ closing price and the R-square value is 0.95 which shows clear promising result. Another method used in this analysis is LSTM and the LSTM measures its root mean square error is 0.83, which means it slightly predicting the values for the prediction. The ARIMA model doesn’t show any promising result for the prediction of the stock price.

The final model is AutoTS which calculates automatically and shows the best fit model for the given dataset using multivariate analysis. It calculates three different models automatically and the best model is Auto-SARIMAX which has rmse value of 0.13 which means the prediction is more promising, whereas the Prophet and ML are not that much promising.

1. **CODES:**

#import packages

import pandas as pd

import numpy as np

#to plot within notebook

import matplotlib.pyplot as plt

%matplotlib inline

#setting figure size

from matplotlib.pylab import rcParams

rcParams['figure.figsize'] = 20,10

#for normalizing data

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature\_range=(0, 1))

#read the file

df = pd.read\_csv('Apple\_Nasd.csv')

#print the head

df.head()

#setting index as date

df['Date'] = pd.to\_datetime(df.Date,format='%Y-%m-%d')

df.index = df['Date']

#plot

plt.figure(figsize=(16,8))

plt.plot(df['Appl\_Close'], label='Close Price history')

plt.title('APPLE Close Price History')

plt.xlabel('Date', fontsize=18)

plt.ylabel('Close Price USD ($)', fontsize=18)

#plot for NASDAQ

plt.figure(figsize=(16,8))

plt.plot(df['Nasd\_Close'], label='Close Price history')

plt.title('NASDAQ Close Price History')

plt.xlabel('Date', fontsize=18)

plt.ylabel('Close Price USD ($)', fontsize=18)

**Moving average**

#creating dataframe with date and the target variable

data = df.sort\_index(ascending=True, axis=0)

new\_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Appl\_Close', 'Nasd\_Close'])

for i in range(0,len(data)):

     new\_data['Date'][i] = data['Date'][i]

     new\_data['Appl\_Close'][i] = data['Appl\_Close'][i]

# NOTE: While splitting the data into train and validation set, we cannot use random splitting since that will destroy the time component. So here we have set the last year’s data into validation and the 4 years’ data before that into train set.

# splitting into train and validation

train = new\_data[:987]

valid = new\_data[987:]

# shapes of training set

print('\n Shape of training set:')

print(train.shape)

# In the next step, we will create predictions for the validation set and check the RMSE using the actual values.

# making predictions

preds = []

for i in range(0,valid.shape[0]):

    a = train['Appl\_Close'][len(train)-248+i:].sum() + sum(preds)

    b = a/248

    preds.append(b)

# checking the results (RMSE value)

rms=np.sqrt(np.mean(np.power((np.array(valid['Appl\_Close'])-preds),2)))

print('\n RMSE value on validation set:')

print(rms)

#plot

valid['Predictions'] = 0

valid['Predictions'] = preds

plt.plot(train['Appl\_Close'])

plt.plot(valid[['Appl\_Close', 'Predictions']])

plt.title('Moving Average for Apple Close Price History')

plt.xlabel('Date in Days', fontsize=18)

plt.ylabel('Close Price USD ($)', fontsize=18)

#creating dataframe with date and the target variable

data = df.sort\_index(ascending=True, axis=0)

new\_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Appl\_Close', 'Nasd\_Close'])

for i in range(0,len(data)):

     new\_data['Date'][i] = data['Date'][i]

     new\_data['Nasd\_Close'][i] = data['Nasd\_Close'][i]

# NOTE: While splitting the data into train and validation set, we cannot use random splitting since that will destroy the time component. So here we have set the last year’s data into validation and the 4 years’ data before that into train set.

# splitting into train and validation

train = new\_data[:987]

valid = new\_data[987:]

# shapes of training set

print('\n Shape of training set:')

print(train.shape)

# In the next step, we will create predictions for the validation set and check the RMSE using the actual values.

# making predictions

preds = []

for i in range(0,valid.shape[0]):

    a = train['Nasd\_Close'][len(train)-248+i:].sum() + sum(preds)

    b = a/248

    preds.append(b)

# checking the results (RMSE value)

rms=np.sqrt(np.mean(np.power((np.array(valid['Nasd\_Close'])-preds),2)))

print('\n RMSE value on validation set:')

print(rms)

#plot

valid['Predictions'] = 0

valid['Predictions'] = preds

plt.plot(train['Nasd\_Close'])

plt.plot(valid[['Nasd\_Close', 'Predictions']])

plt.title('Moving Average for Nasdaq Close Price History')

plt.xlabel('Date in Days', fontsize=18)

plt.ylabel('Close Price USD ($)', fontsize=18)

**Linear Regression**

#setting index as date values

df['Date'] = pd.to\_datetime(df.Date,format='%Y-%m-%d')

df.index = df['Date']

#sorting

data = df.sort\_index(ascending=True, axis=0)

#creating a separate dataset

new\_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Appl\_Close'])

for i in range(0,len(data)):

    new\_data['Date'][i] = data['Date'][i]

    new\_data['Appl\_Close'][i] = data['Appl\_Close'][i]

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import statsmodels.api as sm

AAPL\_price = pd.read\_csv('Apple\_Nasd.csv',usecols=['Date', 'Appl\_Close'])

NASD\_price = pd.read\_csv('Apple\_Nasd.csv',usecols=['Date', 'Nasd\_Close'])

X = sm.add\_constant(NASD\_price['Nasd\_Close'])

model = sm.OLS(AAPL\_price['Appl\_Close'],X)

results = model.fit()

plt.scatter(NASD\_price['Nasd\_Close'],AAPL\_price['Appl\_Close'],alpha=0.3)

y\_predict = results.params[0] + results.params[1]\*NASD\_price['Nasd\_Close']

plt.plot(NASD\_price['Nasd\_Close'],y\_predict, linewidth=3)

plt.xlim(2500,16000)

plt.ylim(0,160)

plt.xlabel('Nasdaq Price')

plt.ylabel('Apple Price')

plt.title('Linear Regression')

print(results.summary())

**Multivariate analysis**

import seaborn as sns

from pylab import rcParams

import matplotlib.pyplot as plt

from matplotlib import rc

import pandas as pd

import numpy as np

from tqdm.notebook import tqdm

import pytorch\_lightning as pl

from sklearn.preprocessing import MinMaxScaler

import torch

import torch.autograd as autograd

import torch.nn as nn

import torch.nn.functional as F

import torch.optim as optim

from torch.utils.data import Dataset, DataLoader

!pip install --quiet pytorch-lightning==1.2.5

!pip install --quiet tqdm==4.59.0

%matplotlib inline

%config InlineBackend.figure\_format='retina'

sns.set(style='whitegrid', palette='muted', font\_scale=1.2)

HAPPY\_COLORS\_PALETTE = ["#01BEFE", "#FFDD00", "#FF7D00", "#FF006D", "#ADFF02", "#8F00FF"]

sns.set\_palette(sns.color\_palette(HAPPY\_COLORS\_PALETTE))

rcParams['figure.figsize']=12,8

tqdm.pandas()

pl.seed\_everything(42)

df=pd.read\_csv('Apple\_Nasd.csv')

df

df["Prev\_Apl\_Close"] = df.shift(1)['Appl\_Close']

df["Prev\_Nas\_Close"] = df.shift(1)['Nasd\_Close']

df["Apl\_Change"]= df.progress\_apply(

    lambda row: 0 if np.isnan(row.Prev\_Apl\_Close) else row.Appl\_Close-row.Prev\_Apl\_Close,

    axis=1

)

df1=df.drop(['Date'], axis=1)

df1=df.drop(['Appl\_EPS'], axis=1)

df1=df.drop(['Nasd\_EPS'], axis=1)

scaler = MinMaxScaler(feature\_range=(-1, 1))

scaler = scaler.fit(df1)

sc\_df=df1

sc\_df=pd.DataFrame(

    scaler.transform(sc\_df),

    index=sc\_df.index,

    columns=sc\_df.columns

    )

df["Date"]=pd.to\_datetime(df.Date,format="%Y-%m-%d")

df.index=df['Date']

plt.figure(figsize=(16,8))

plt.title('Closing Price History of Apple and NASDAQ ')

plt.xlabel('Date in Days', fontsize=18)

plt.ylabel('Scaled Close Price', fontsize=18)

plt.plot(sc\_df["Nasd\_Close"])

plt.plot(sc\_df["Appl\_Close"])

plt.legend(['Apple', 'NASDAQ'])

row=[]

for \_, row in tqdm(df.iterrows(), total=df.shape[0]):

  row\_data = dict(

      Appl\_Close= row.Appl\_Close,

      Nasd\_Close=row.Nasd\_Close,

  )

features\_df=pd.DataFrame(row)

train\_size=int(len(df1)\*.8)

train\_size

train\_df, test\_df = df1[:train\_size], df1[train\_size +1:]

train\_df.shape, test\_df.shape

scaler = MinMaxScaler(feature\_range=(-1, 1))

scaler = scaler.fit(train\_df)

train\_df=pd.DataFrame(

    scaler.transform(train\_df),

    index=train\_df.index,

    columns=train\_df.columns

)

test\_df=pd.DataFrame(

    scaler.transform(test\_df),

    index=test\_df.index,

    columns=test\_df.columns

)

train\_df.head()

test\_df.head()

def create\_sequences(input\_data: pd.DataFrame, target\_column, Sequence\_length):

  sequence=[]

  data\_size = len(input\_data)

  for i in tqdm(range(data\_size - sequence\_length)):

    sequence = input\_data[i:i+sequence\_length]

    label\_position = i + sequence\_length

    label = input\_data.iloc[label\_position][target\_column]

    sequences.append((sequence, label))

  return sequences

df["Date"]=pd.to\_datetime(df.Date,format="%Y-%m-%d")

df.index=df['Date']

plt.figure(figsize=(16,8))

plt.title('Closing Price History of Apple and NASDAQ ')

plt.xlabel('Date in Days', fontsize=18)

plt.ylabel('Scaled Close Price', fontsize=18)

plt.plot(test\_df["Nasd\_Close"],label='Test Close Price')

plt.plot(train\_df["Nasd\_Close"],label='Train Close Price')

plt.plot(test\_df["Appl\_Close"],label='Test Close Price')

plt.plot(train\_df["Appl\_Close"],label='Train Close Price')

plt.legend([ 'Apple Test','Apple Train','NASDAQ Test', 'NASDAQ Train'])

plt.plot(train\_df["Nasd\_Diff"],label='Close Price')

plt.plot(train\_df["Appl\_Diff"],label='Close Price')

plt.title('Price Difference in two consecutive days')

plt.xlabel('Date in Days', fontsize=18)

plt.ylabel('Scaled', fontsize=18)

plt.legend([ 'NASDAQ Scales Train', 'Apple Scaled Train'])

plt.plot(train\_df["Nasd\_Diff\_Percent"],label='Price Percent')

plt.plot(train\_df["Appl\_Diff\_Percent"],label='Price\_Percent')

plt.title('Price Difference in percentage of two consecutive days')

plt.xlabel('Date in Days', fontsize=18)

plt.ylabel('Scaled', fontsize=18)

plt.legend([ 'NASDAQ Scales Train', 'Apple Scaled Train'])

**LSTM**

import math

import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, LSTM

import matplotlib.pyplot as plt

plt.style.use('fivethirtyeight')

import tensorflow as tf

#Get the data

df =pd.read\_csv('Apple\_Nasd.csv')

df

# visualize the closing price history

df["Date"]=pd.to\_datetime(df.Date,format="%Y-%m-%d")

df.index=df['Date']

plt.figure(figsize=(16,8))

plt.title('APPLE Close Price History')

plt.plot(df['Appl\_Close'])

plt.xlabel('Date', fontsize=18)

plt.ylabel('Close Price USD ($)', fontsize=18)

plt.show()

df.shape

# visualize the closing price history

df["Date"]=pd.to\_datetime(df.Date,format="%Y-%m-%d")

df.index=df['Date']

plt.figure(figsize=(16,8))

plt.title('NASDAQ Close Price History')

plt.plot(df['Nasd\_Close'])

plt.xlabel('Date', fontsize=18)

plt.ylabel('Close Price USD ($)', fontsize=18)

plt.show()

data = df.filter(['Appl\_Close'])

# Convert the dataframe to a numpy array

dataset = data.values

# Get the number or 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

#Create training dataset

#Create the scaled training data set

train\_data = scaled\_data[0:training\_data\_len, :]

#Split the data into x\_train and y\_train data sets

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])

if i <= 61:

print(x\_train)

print(y\_train)

print()

# Convert the x\_train and y\_train to numpy arrays

x\_train, y\_train = np.array(x\_train), np.array(y\_train)

x\_train.shape

#Reshape the data

x\_train = np.reshape(x\_train,(x\_train.shape[0], x\_train.shape[1], 1))

x\_train.shape

# Build the LSTM

model=Sequential()

model.add(LSTM(50,return\_sequences=True,input\_shape=(x\_train.shape[1],1)))

model.add(LSTM(50, return\_sequences=False))

model.add(Dense(25))

model.add(Dense(1))

# Compile the model

model.compile(optimizer='adam', loss='mean\_squared\_error')

#Train the model

model.fit(x\_train,y\_train,epochs=1,batch\_size=1)

#Create the testing data set

#Create a new array containing scaled values from index 1803 to 2329

test\_data = scaled\_data[training\_data\_len - 60: , :]

#Create the data sets x\_test and y\_test

x\_test = []

y\_test = dataset[training\_data\_len:, :]

for i in range(60, len(test\_data)):

x\_test.append(test\_data[i-60:i, 0])

# Convert the data into a numpy array

x\_test = np.array(x\_test)

x\_test.shape

#Reshape the data

x\_test = np.reshape(x\_test, (x\_test.shape[0], x\_test.shape[1], 1))

# Get the models predicted price values

predictions = model.predict(x\_test)

predictions = scaler.inverse\_transform(predictions)

# Get the root mean square error (RMSE)

rmse = np.sqrt( np.mean(predictions - y\_test)\*\*2)

rmse

# Plot the data

train = data[:training\_data\_len]

valid = data[training\_data\_len:]

valid['Predictions'] = predictions

#Visualize the data

plt.figure(figsize=(16,8))

plt.title('Prediction Model')

plt.xlabel('Date', fontsize=18)

plt.ylabel('Close price USD $')

plt.plot(train['Appl\_Close'])

plt.plot(valid[['Appl\_Close', 'Predictions']])

plt.legend(['Train', 'Val', 'Predictions'])

plt.show()

**ARIMA MODEL**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

df = pd.read\_csv('Apple\_Nasd.csv')

df.tail()

df["Date"]=pd.to\_datetime(df.Date,format="%Y-%m-%d")

df.index=df['Date']

plt.figure(figsize=(16,8))

plt.plot(df["Appl\_Close"],label='Closing Price',c='blue')

df.head()

df.describe()

df.set\_index('Date',inplace=True)

from pylab import rcParams

rcParams['figure.figsize'] = 15, 7

df.plot()

from statsmodels.tsa.stattools import adfuller

test\_result=adfuller(df['Appl\_Close'])

def adfuller\_test(Appl\_Close):

result=adfuller(Appl\_Close)

labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations']

for value,label in zip(result,labels):

print(label+' : '+str(value) )

df['Sales First Difference'] = df['Appl\_Close'] - df['Appl\_Close'].shift(1)

df['Seasonal First Difference']=df['Appl\_Close']-df['Appl\_Close'].shift(12)

df.head()

# Again testing if data is stationary

adfuller\_test(df['Seasonal First Difference'].dropna())

df['Seasonal First Difference'].plot()

#Autocorrelation

from pandas.plotting import autocorrelation\_plot

autocorrelation\_plot(df['Appl\_Close'])

plt.show()

#Partialcorrelation

from statsmodels.graphics.tsaplots import plot\_acf,plot\_pacf

import statsmodels.api as sm

fig = plt.figure(figsize=(12,8))

ax1 = fig.add\_subplot(211)

fig = sm.graphics.tsa.plot\_acf(df['Seasonal First Difference'].dropna(),lags=40,ax=ax1)

ax2 = fig.add\_subplot(212)

fig = sm.graphics.tsa.plot\_pacf(df['Seasonal First Difference'].dropna(),lags=40,ax=ax2)

# For non-seasonal data

#p=1, d=1, q=0 or 1

from statsmodels.tsa.arima\_model import ARIMA

model=ARIMA(df['Appl\_Close'],order=(1,1,1))

model\_fit=model.fit()

model\_fit.summary()

df['forecast']=model\_fit.predict(start=90,end=2300,dynamic=True)

df[['Appl\_Close','forecast']].plot(figsize=(12,8))

import statsmodels.api as sm

model=sm.tsa.statespace.SARIMAX(df['Appl\_Close'],order=(1, 1, 1),seasonal\_order=(1,1,1,12))

results=model.fit()

df['forecast']=results.predict(start=900,end=2228,dynamic=True)

df[['Appl\_Close','forecast']].plot(figsize=(12,8))

from pandas.tseries.offsets import DateOffset

future\_dates=[df.index[-1]+ DateOffset(months=x)for x in range(0,24)]

future\_datest\_df=pd.DataFrame(index=future\_dates[1:],columns=df.columns)

future\_datest\_df.tail()

future\_df=pd.concat([df,future\_datest\_df])

future\_df['forecast'] = results.predict(start = 1104, end = 2320, dynamic= True)

future\_df[['Appl\_Close', 'forecast']].plot(figsize=(12, 8))

predictions\_ARIMA\_diff = pd.Series(results.fittedvalues, copy=True)

predictions\_ARIMA\_diff\_cumsum = predictions\_ARIMA\_diff.cumsum()

predictions\_ARIMA\_log = pd.df(df\_log['Appl\_Close'].iloc[0], index=df\_log.index)

predictions\_ARIMA\_log = predictions\_ARIMA\_log.add(predictions\_ARIMA\_diff\_cumsum, fill\_value=0)

predictions\_ARIMA = np.exp(predictions\_ARIMA\_log)

plt.plot(df\_log)

plt.plot(predictions\_ARIMA)

MISCELENEOUS

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

!matplotlib inline

from matplotlib.pylab import rcParams

rcParams['figure.figsize']=20,10

from keras.models import Sequential

from keras.layers import LSTM,Dropout,Dense

from sklearn.preprocessing import MinMaxScaler

df = pd.read\_csv('Apple\_Nasd.csv')

df.head()

#Plot of price difference

df["Date"]=pd.to\_datetime(df.Date,format="%Y-%m-%d")

df.index=df['Date']

plt.figure(figsize=(16,8))

plt.plot(df["Nasd\_Diff"],label='Different Price',c='blue')

df["Date"]=pd.to\_datetime(df.Date,format="%Y-%m-%d")

df.index=df['Date']

plt.figure(figsize=(16,8))

plt.plot(df["Appl\_Diff"],label='Different Price', c='red')

df["Date"]=pd.to\_datetime(df.Date,format="%Y-%m-%d")

df.index=df['Date']

plt.figure(figsize=(16,8))

plt.plot(df["Appl\_EPS"],label='EPS Growth', c='blue')

plt.plot(df["Nasd\_EPS"],label='EPS Growth', c='red')

df["Date"]=pd.to\_datetime(df.Date,format="%Y-%m-%d")

df.index=df['Date']

plt.figure(figsize=(16,8))

plt.plot(df["Appl\_Diff\_Percent"],label='Different Price', c='red')

df["Date"]=pd.to\_datetime(df.Date,format="%Y-%m-%d")

df.index=df['Date']

plt.figure(figsize=(16,8))

plt.plot(df["Nasd\_Diff\_Percent"],label='Different Price',c='blue')

df["Date"]=pd.to\_datetime(df.Date,format="%Y-%m-%d")

df.index=df['Date']

plt.figure(figsize=(16,8))

plt.plot(df["Nasd\_EPS"],label='Different Price',c='blue')

plt.plot(df["Nasd\_Close"],label='Different Price', c='red')

data=df.sort\_index(ascending=True,axis=0)

new\_dataset=pd.DataFrame(index=range(0,len(df)),columns=['Date','Appl\_Close'])

for i in range(0,len(data)):

new\_dataset["Date"][i]=data['Date'][i]

new\_dataset["Appl\_Close"][i]=data["Appl\_Close"][i]

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()

**AutoTS**

import pandas as pd

!pip install auto-ts==0.0.38

from auto\_ts import auto\_timeseries

df = pd.read\_csv("Apple\_Nasd.csv")

df.head()

df1=df.drop(columns='Date')

df1.head()

import datetime

import pandas as pd

dti = pd.date\_range("2012-07-02 00:00:00", periods=2328, freq="B", name="day")

dti=pd.DataFrame(dti)

dti.head(10)

df1=pd.concat([dti,df1],axis=1)

df1.head()

model = auto\_timeseries( score\_type='rmse', seasonality=True, model\_type='best', verbose=2)

train\_size=int(len(df1)\*.8)

train\_size

tr\_df, te\_df = df1[:train\_size], df1[train\_size +1:]

tr\_df.shape, te\_df.shape

tr\_df.info()

model.fit(traindata= tr\_df, ts\_column="day", target="Appl\_Close", cv=3)

model.get\_leaderboard()

model.get\_model("auto\_SARIMAX").summary()

1. **Resources:**

<https://www.kaggle.com/kashnitsky/topic-9-part-1-time-series-analysis-in-python>

<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/>

<http://article.sapub.org/10.5923.j.ajms.20160605.02.html#Sec3.2>

<https://pypi.org/project/AutoTS/>

<https://www.kaggle.com/sajikim/time-series-forecasting-methods-example-python/notebook>

<https://towardsdatascience.com/how-to-model-time-series-data-with-linear-regression-cd94d1d901c0>