Yes Bank Stock Price Prediction

Saurabh Yadav
Debaprasad Mohapatra

Abstract: Accurate prediction of stock market returns is a very challenging task due to the volatile and non-linear nature of the financial stock markets. With the introduction of machine learning, time series forecasting and increased computational capabilities, programmed methods of prediction have proved to be more efficient in predicting stock prices. In this work, regression and time series forecasting techniques have been utilized for predicting the next day closing price for one company belonging to the Finance sector of operation. The financial data: Open, High, Low and Close prices of stock are used for creating new variables which are used as inputs to the model. The models are evaluated using standard strategic indicators: RMSE and MAPE. The low values of these two indicators show that the models are efficient in predicting stock closing prices.

CHAPTER 1- OBJECTIVES AND INTRODUCTION

1.1 INTRODUCTION:

Stock market is characterized as dynamic, unpredictable and non-linear in nature. Predicting stock prices is a challenging task as it depends on various factors including but not limited to political conditions, global economy, company's financial reports and performance etc. Thus, to maximize the profit and minimize the losses, techniques to predict values of the stock in advance by analyzing the trend over the last few years, could prove to be highly useful for making stock market movements. Traditionally, two main approaches have been proposed for predicting the stock price of an organization. Technical analysis method uses historical prices of stocks like closing and opening price, volume traded, adjacent close values etc. of the stock for predicting the future price of the stock. The second type of analysis is qualitative, which is performed on the basis of external factors like company profile, market situation, political and economic factors, textual information in the form of financial news articles, social media and even blogs by economic analysts [3]. Nowadays, advanced intelligent techniques based on either technical or fundamental analysis are used for predicting stock prices. Particularly, for stock market analysis, the data size is huge and also non-linear. To deal with this variety of data an efficient model is needed that can identify the hidden patterns and complex relations in this large data set. Machine learning techniques in this area have proved to improve efficiencies by 60-80 percent as compared to the past methods.

1.2 OBJECTIVE:

Yes Bank is a well-known bank in the Indian financial domain. Since 2018, it has been in the news because of the fraud case involving Rana Kapoor. Owing to this fact, it was interesting to see how that impacted the stock prices of the company and whether Time series models or any other predictive models can do justice to such situations. This dataset has monthly stock prices of the bank since its inception and includes closing, starting, highest, and lowest stock prices of every month. The main objective is to predict the stock's closing price of the month.

CHAPTER 2- DATA COLLECTION AND DATA PREPARATION

2.1 Data Summary:

The dataset of YES BANK has monthly stock prices of the bank since its inception and includes closing, starting, highest, and lowest stock prices of every month of around 180 observations.

It contains the following features:

- **Date:** It denotes date of investment done (in our case we have month and year).
- **Open:** Open means the price at which a stock started trading when the opening bell rang.
- **High:** High refer to the maximum prices in a given time period.
- **Low:** Low refer to the minimum prices in a given time period.
- **Close:** Close refers to the price of an individual stock when the stock exchange closed for the day.

2.2 Data Collection

Before building any machine learning model, it is vital to understand what the data is, and what are we trying to achieve. Data exploration reveals the hidden trends and insights and data preprocessing makes the data ready for use by ML algorithms.
So, let's begin. . .
To proceed with the problem dealing first we will load our dataset that is given to us in a csv file into a data frame.
Mount the drive and load the csv file into a data frame.

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

```
[ ] # loading the csv data to a Pandas DataFrame
    df = pd.read_csv("/content/drive/MyDrive/Almabetter/Module 4: Machine Learning/Capstone Project 2/dat/data_YesBank_StockPrices.csv")
```

Importing required python libraries:

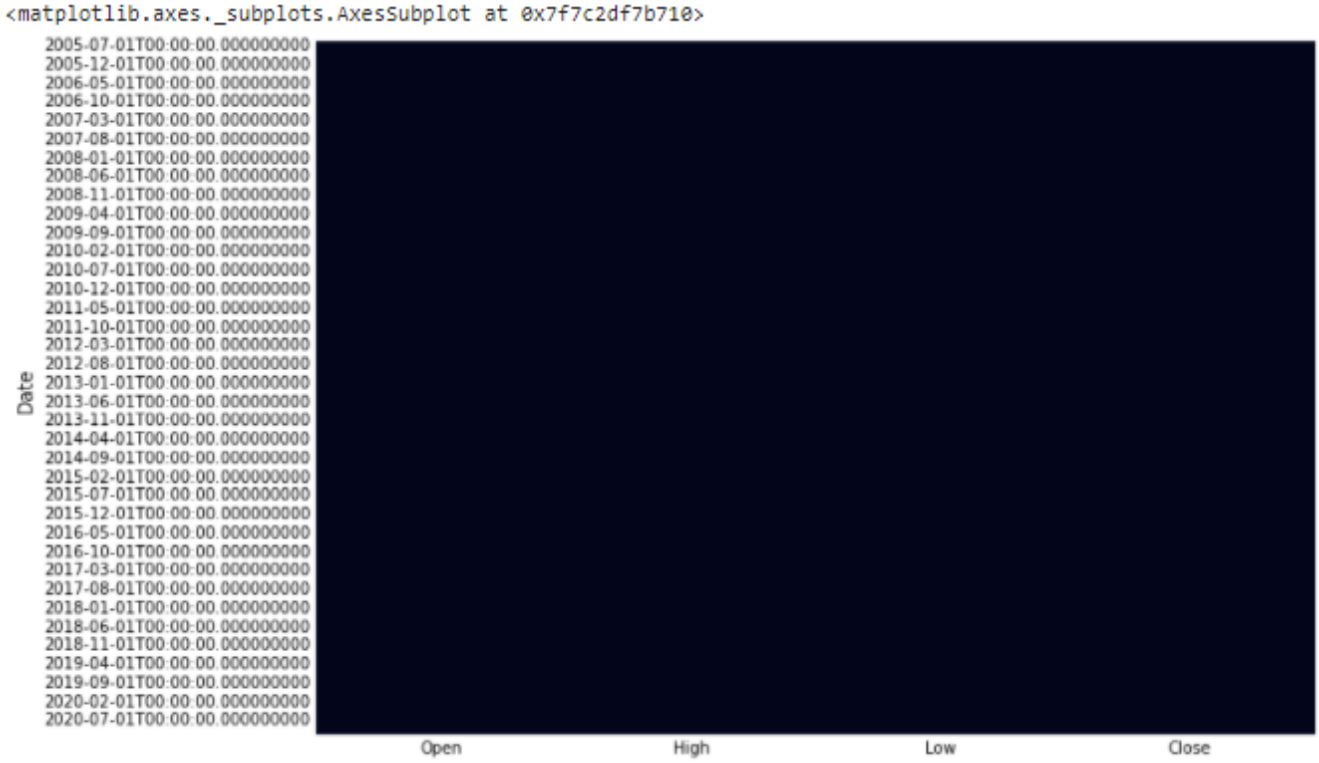
```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
from pylab import rcParams
rcParams['figure.figsize'] = (10, 6)
%matplotlib inline
import seaborn as sns

from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima_model import ARIMA
# from pmdarima.arima import auto_arima

from sklearn.metrics import mean_squared_error, mean_absolute_error
import math
import numpy as np
```

CHAPTER 3- EXPLORATORY DATA ANALYSIS/ Data Preprocessing

The primary goal of EDA is to support the analysis of data prior to making any conclusions. Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods.

3.1) Missing values:

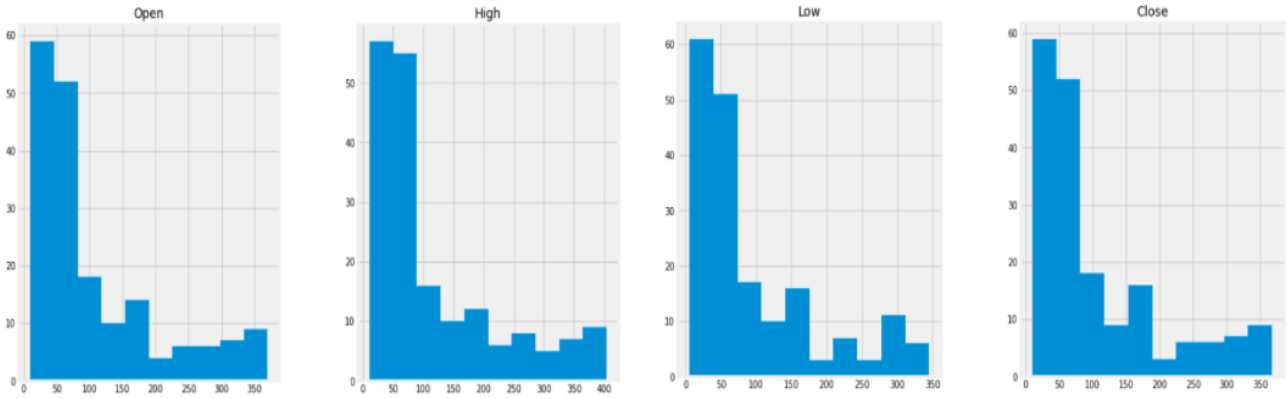
The next single-line code will visualize the location of missing values.



As seen above we don't have any missing values in the data.

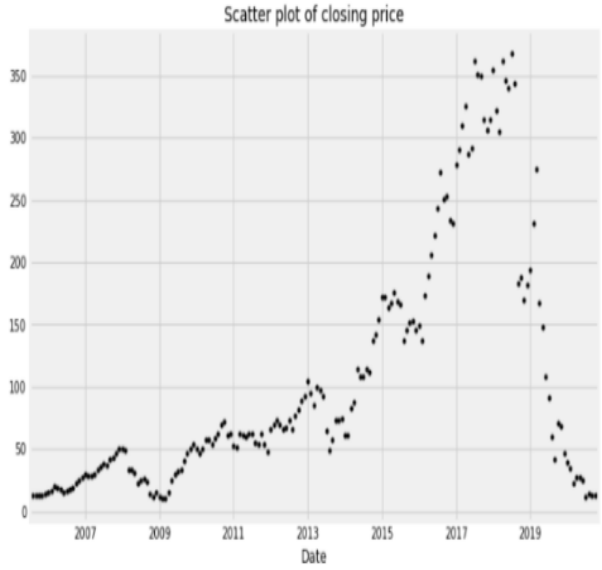
3.2) Data Visualization

Distribution Of Features: Plotted histogram to see the distribution of the data.

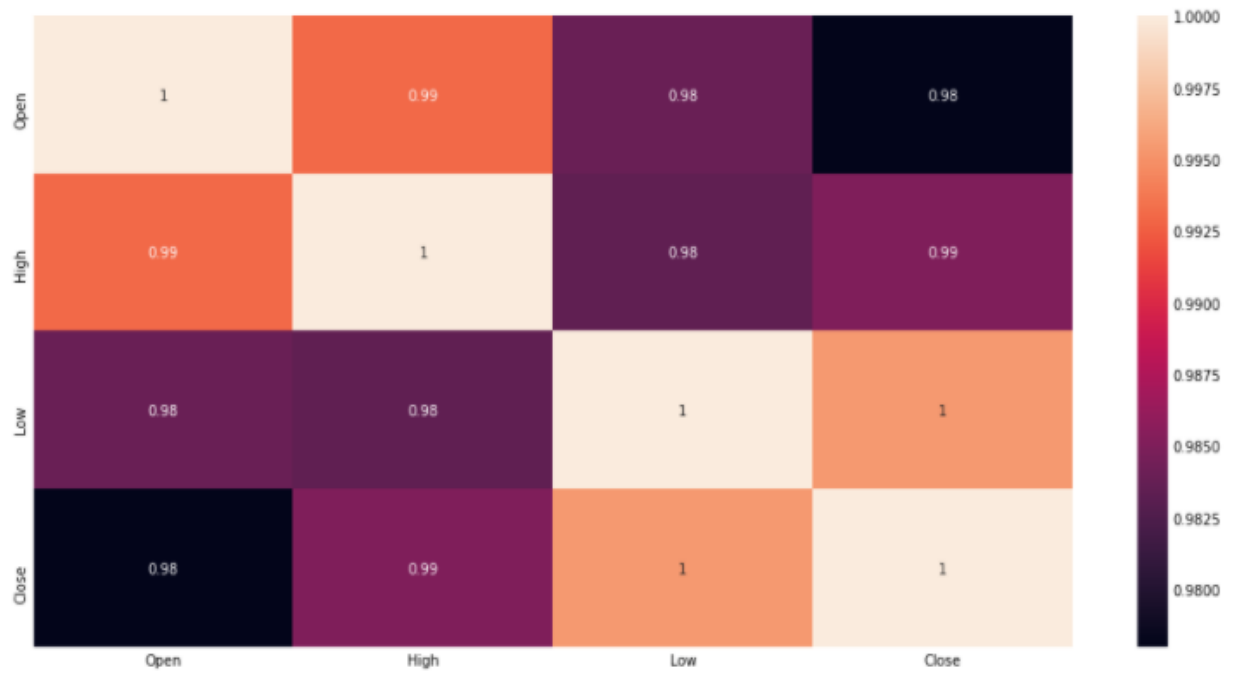


It can be observed that in the starting of the month the stock prices are high but after the fraud incident happened the stock prices gradually decreased.

Analysis of target variable:



Correlation between the variables:



Here, all variables show the highest correlation among them.

3.3) Data Preprocessing and Feature Selection

The Given Date in data is of format MMM-YY is converted to the proper date of YYYY-MM-DD. Since, 'Date' column has dtype as object we've converted it into date-time format. Then setting the date column as an index.

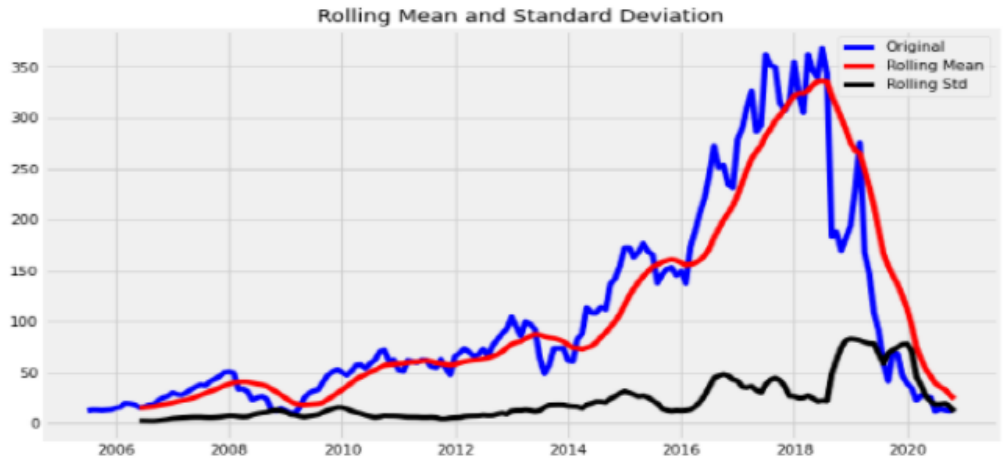
	Date	Open	High	Low	Close
0	Jul-05	13.00	14.00	11.25	12.46
1	Aug-05	12.58	14.88	12.55	13.42
2	Sep-05	13.48	14.87	12.27	13.30
3	Oct-05	13.20	14.47	12.40	12.99
4	Nov-05	13.35	13.88	12.88	13.41

Converted

Date	Open	High	Low	Close
2005-07-01	13.00	14.00	11.25	12.46
2005-08-01	12.58	14.88	12.55	13.42
2005-09-01	13.48	14.87	12.27	13.30
2005-10-01	13.20	14.47	12.40	12.99
2005-11-01	13.35	13.88	12.88	13.41

- For time series forecasting we select only the closing price feature of stock.
- Then we've checked if the series is stationary or not because time series analysis only works with stationary data.

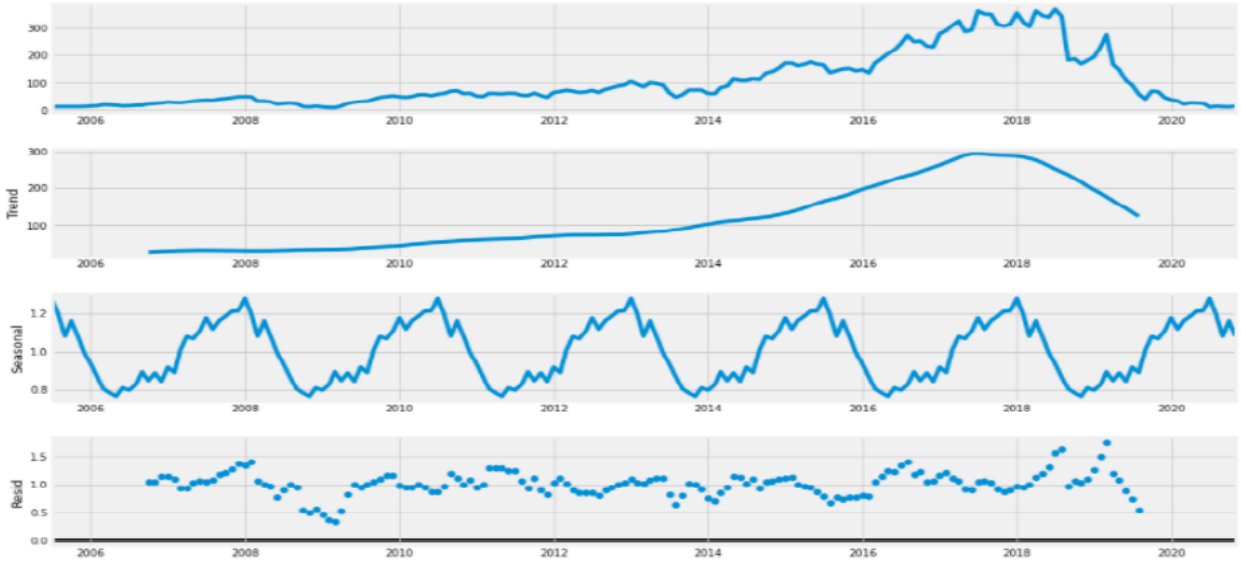
To check if the series is stationary or not we've used the Rolling window and ADF (Augmented Dickey-Fuller) Test.




```
Results of dickey fuller test
Test Statistics                  -1.906409
p-value                           0.329052
No. of lags used                 14.000000
Number of observations used     170.000000
critical value (1%)              -3.469413
critical value (5%)              -2.878696
critical value (10%)             -2.575917
dtype: float64
weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary.
```

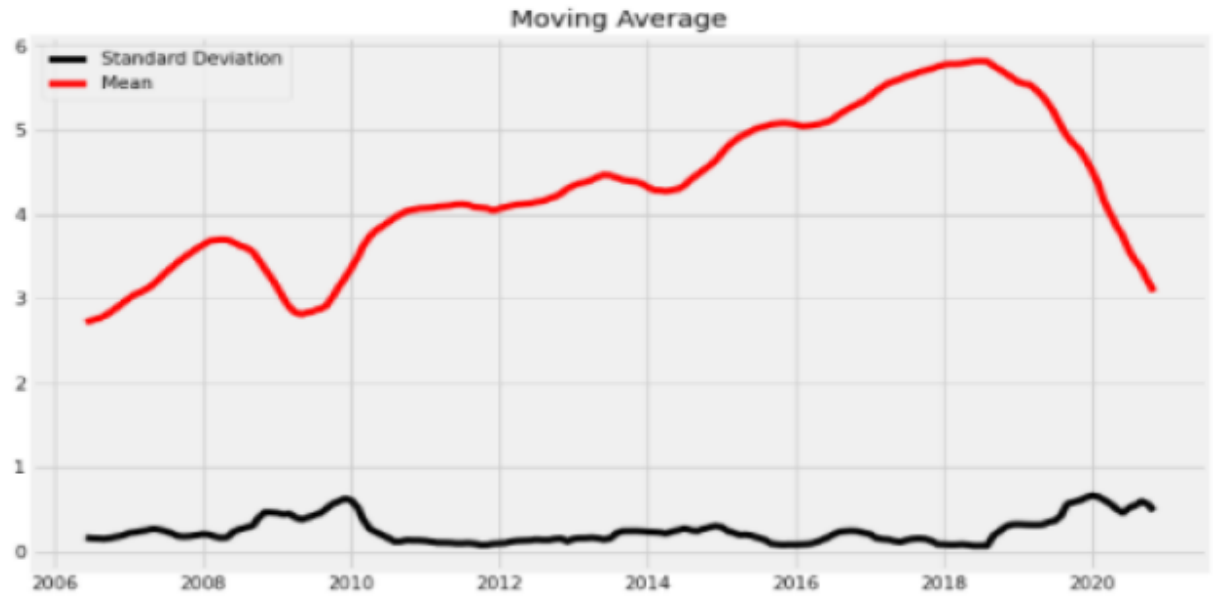
- We see that the p-value is greater than 0.05 so we cannot reject the Null hypothesis. Also, the test statistics are greater than the critical values. so the data is non-stationary.
- In order to perform a time series analysis, we may need to separate seasonality and trend from our series. The resultant series will become stationary through this process.

Now separating Trend and Seasonality from the time series.



Seasonal Decompose

Now, we start by taking a log of the series to reduce the magnitude of the values and reduce the rising trend in the series. Then after getting the log of the series, we find the rolling average of the series.
A rolling average is calculated by taking input for the past 12 months and giving a mean consumption value at every point further ahead in series.



Standard deviation follows the straight line

For the fbprophet (Facebook's library for time series forecasting) the input to prophet is always a data-frame with two columns: 'ds' and 'y‘.
The ds (datestamp) column should be of a format expected by Pandas, ideally YYYY-MM-DD for a date or YYYY-MM-DD HH:MM:SS for a timestamp. The y column must be numeric, and represents the measurement we wish to forecast.

Date	Close
2005-07-01	12.46
2005-08-01	13.42
2005-09-01	13.30
2005-10-01	12.99
2005-11-01	13.41

Converted

	ds	y
0	2005-07-01	12.46
1	2005-08-01	13.42
2	2005-09-01	13.30
3	2005-10-01	12.99
4	2005-11-01	13.41

CHAPTER 4-Model building, Predictions and Forecasting

4.1) Regression Approach

The 4 regression models build for the data are:

- **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.
- **Random Forest:** It is a supervised learning algorithm that uses ensemble learning methods for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.
- **XGboost:** XGBoost stands for “Extreme Gradient Boosting”. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements Machine Learning algorithms under the Gradient Boosting framework. It provides a parallel tree boosting to solve many data science problems in a fast and accurate way.
- **k-Nearest Neighbours:** Another interesting ML algorithm that one can use here is kNN (k nearest neighbours). Based on the independent variables, kNN finds the similarity between new data points and old data points.

Defining parameters:

```
# Difining parameters for grid search
params = {'n_neighbors':[2,3,4,5,6,7,8,9]} #giving range of k value
model = GridSearchCV(knn, params, cv=5) #Applying gridsearchCV for selecting optimal no. of k values
```

Combining all regression models:

	Model_Name	MAE	MSE	RMSE	MAPE
0	LinearRegression	3.05	19.99	4.47	5.40
1	RandomForestRegressor	3.87	29.89	5.47	6.17
3	KNeighborsRegressor	4.70	50.70	7.12	6.45
2	XGBRegressor	4.30	33.32	5.77	6.86

Observation from above table:

- Linear Regression gives the lowest MAE, MSE, RMSE, MAPE.
- Here, we can say that Linear Regression is the best model that can be used for the stock price prediction.

4.2) Time Series Approach:

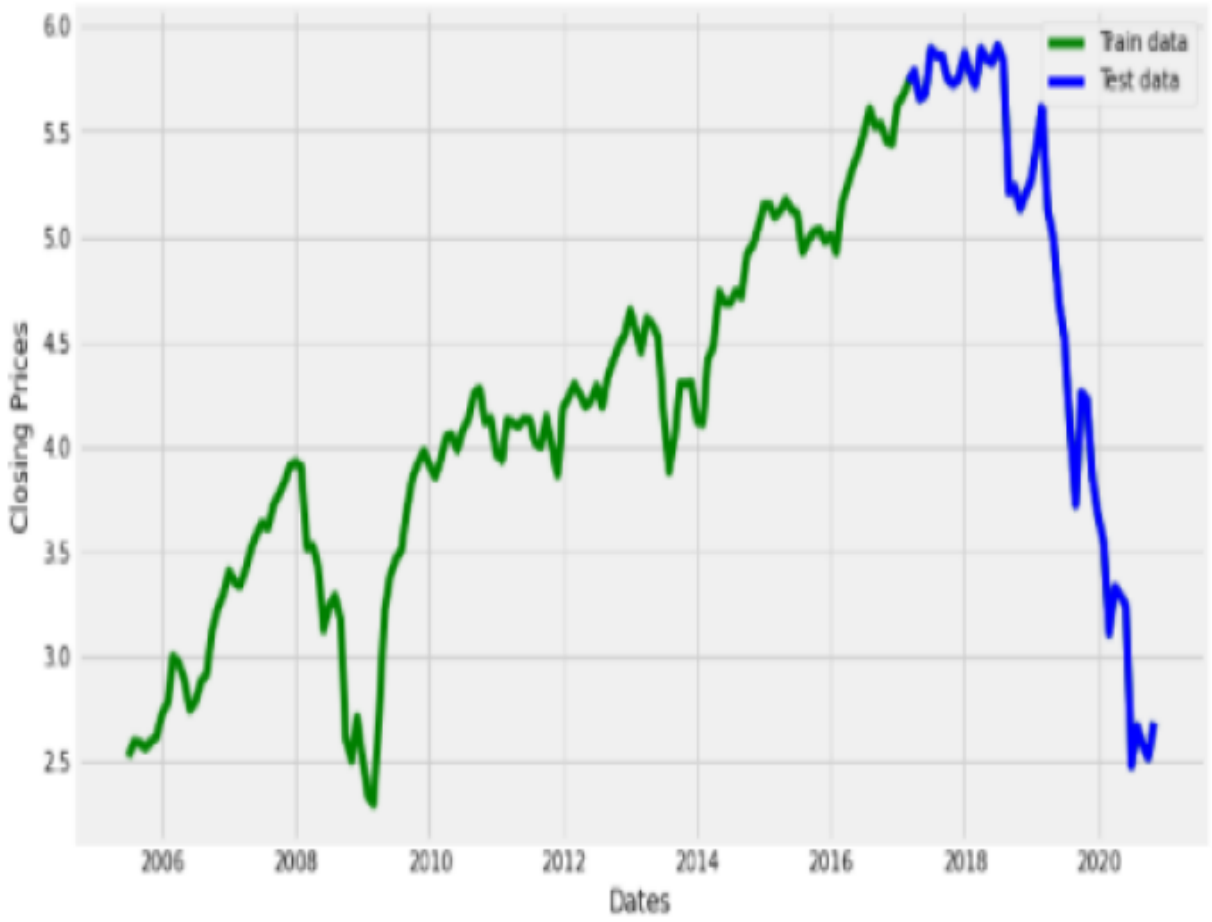
4.2.1) Auto-ARIMA: Auto-ARIMA uses brute force and tries different combinations of p, q, and d and then returns the best model after evaluation. It uses mean squared error to evaluate the best model. It also uses Akaike Information Criteria (AIC) and Bayesian information criterion (BIC) which are statistical measures of goodness of fit and the simplicity of the model.

Automatically discover the optimal order for an ARIMA model.

- The auto_arima function seeks to identify the most optimal parameters for an ARIMA model, and returns a fitted ARIMA model.
- The auro_arima function works by conducting differencing tests (i.e., Kwiatkowski–Phillips–Schmidt–Shin, Augmented Dickey-Fuller or Phillips–Perron) to determine the order of differencing, d, and then fitting models within ranges of defined start_p, max_p, start_q, max_q ranges.
- If the seasonal optional is enabled, auto_arima also seeks to identify the optimal P and Q hyper- parameters after conducting the Canova-Hansen to determine the optimal order of seasonal differencing D.

Split data into train and test :

<matplotlib.legend.Legend at 0x7f7c11c47390>

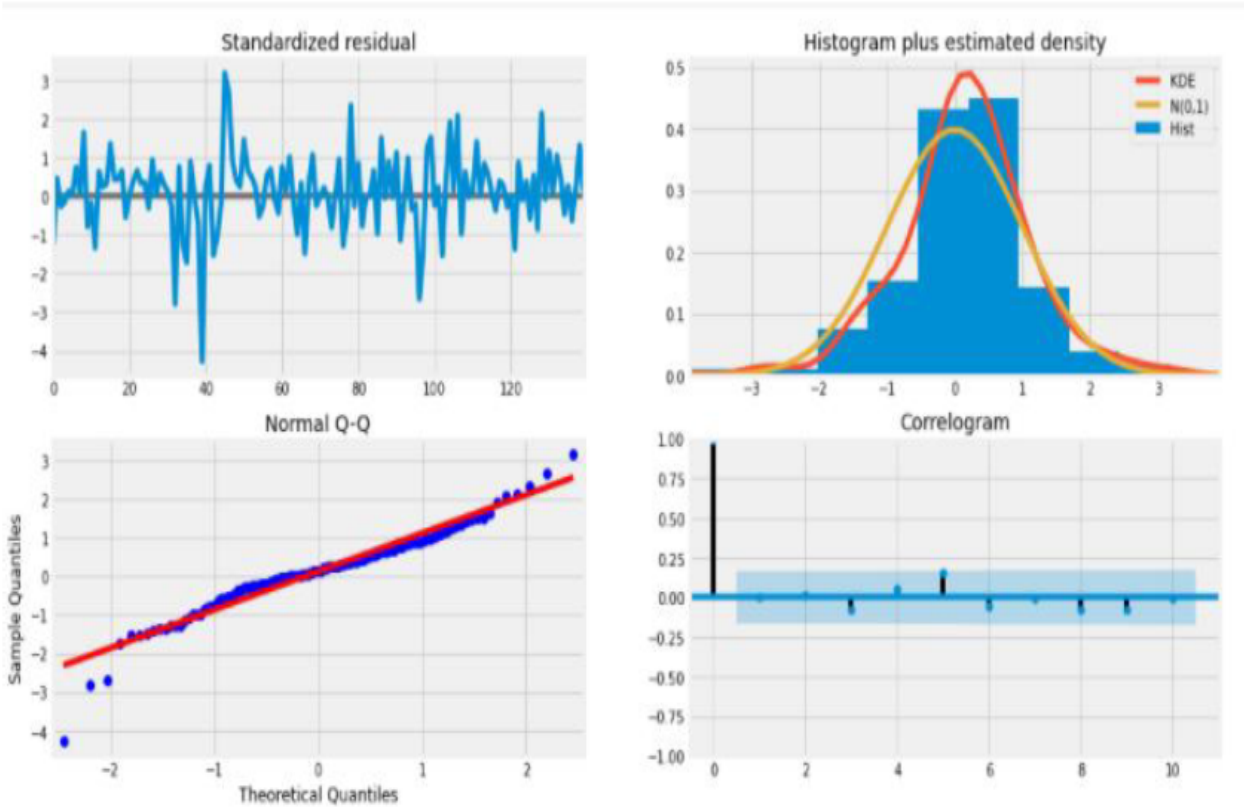


Here, we use Auto ARIMA to get the best parameters without even plotting ACF and PACF graphs.

```
Best model:  ARIMA(2,0,1)(0,0,0)[0] intercept
Total fit time: 4.949 seconds
                               SARIMAX Results
==========================================================================================
Dep. Variable:                           y   No. Observations:                  140
Model:               SARIMAX(2, 0, 1)   Log Likelihood                81.589
Date:                Thu, 08 Jul 2021   AIC                           -153.179
Time:                        19:37:40   BIC                           -138.471
Sample:                             0   HQIC                          -147.202
                                  - 140
Covariance Type:                  opg
==========================================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
intercept      0.0499      0.095      0.527      0.598      -0.136       0.236
ar.L1          0.4876      0.160      3.043      0.002       0.174       0.802
ar.L2          0.5001      0.161      3.110      0.002       0.185       0.815
ma.L1          0.7828      0.129      6.090      0.000       0.531       1.035
sigma2         0.0177      0.001     12.022      0.000       0.015       0.021
------------------------------------------------------------------------------------------
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                60.14
Prob(Q):                              0.93   Prob(JB):                         0.00
Heteroskedasticity (H):               0.65   Skew:                            -0.57
Prob(H) (two-sided):                  0.14   Kurtosis:                         6.00
==========================================================================================
```

The Auto-ARIMA model provided the value of p, d, and q as 2,0 and 1 respectively.

Residual plots from Auto-ARIMA:



Here,

- Top left: The residual errors seem to fluctuate around a mean of zero and have a uniform variance.
- Top Right: The density plot suggest normal distribution with mean zero.
- Bottom left: All the dots should fall perfectly in line with the red line. Any significant deviations would imply the distribution is skewed.
- Bottom Right: The Correlogram, aka, ACF plot shows the residual errors are not auto-correlated.

Overall, it seems to be a good fit. Let's start forecasting the stock prices.

Now building ARIMA model with provided optimal parameters p(2), d(0) and q(1):

ARMA Model Results

Dep. Variable:	Close	No. Observations:	140
Model:	ARMA(2, 1)	Log Likelihood	81.589
Method:	css-mle	S.D. of innovations	0.133
Date:	Thu, 08 Jul 2021	AIC	-153.179
Time:	19:39:11	BIC	-138.471
Sample:	07-01-2005 - 02-01-2017	HQIC	-147.202

	coef	std err	z	P>|z|	[0.025	0.975]
const	4.0716	0.985	4.132	0.000	2.140	6.003
ar.L1.Close	0.4875	0.173	2.821	0.005	0.149	0.826
ar.L2.Close	0.5002	0.173	2.894	0.004	0.161	0.839
ma.L1.Close	0.7828	0.128	6.117	0.000	0.532	1.034

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.0082	+0.0000j	1.0082	0.0000
AR.2	-1.9827	+0.0000j	1.9827	0.5000
MA.1	-1.2774	+0.0000j	1.2774	0.5000

Forecasting the auto ARIMA model:

An auto-ARIMA model uses past data to understand the pattern in the time series. Using these values, the model captured a decreasing trend in the series.

Performance metrics for auto-Arima:

Metrics	**Errors**
MSE(Mean Square Error)	1.476
MAE(Mean Absolute Error)	0.8569
RMSE(Root Mean Square Error)	1.216
MAPE(Mean Absolute Percentage Error)	0.255

- Around 25.5% MAPE implies the model is about 74.5% accurate in predicting the test set observations.
- Also evident from the plot, the model has captured a trend in the series, but does not focus on the seasonal part. In the next section, we will implement a time series model that takes both trend and seasonality of a series into account i.e. fbprophet.

4.2.2) Fbprophet

Prophet, or “Facebook Prophet,” is an open-source library for univariate (one variable) time series forecasting developed by Facebook. FBProphet uses time as a regressor and tries to fit several linear and nonlinear functions of time as components. By default, FBProphet will fit the data using a linear model but it can be changed to the nonlinear model (logistics growth) from its arguments.

- Prophet is Facebook's library for time series forecasting.
- Prophet follows the sklearn model API.
- Here, we create an instance of the Prophet class and then call its fit and predict methods.
- The input to Prophet is always a dataframe with two columns: **'ds' and 'y'**
- The **ds (datestamp)** column should be of a format expected by Pandas, ideally YYYY-MM-DD for a date or YYYY-MM-DD HH:MM:SS for a timestamp. The **y** column must be numeric, and represents the measurement we wish to forecast.

Now,

- Fitting the model by instantiating a new Prophet object.
- Then we call its fit method and pass in the historical dataframe.
- Then predictions are made on a dataframe with a column **ds** containing the dates for which a prediction is to be made. We can get a suitable data frame that extends into the future a specified number of days using the helper method **Prophet.make_future_dataframe.**

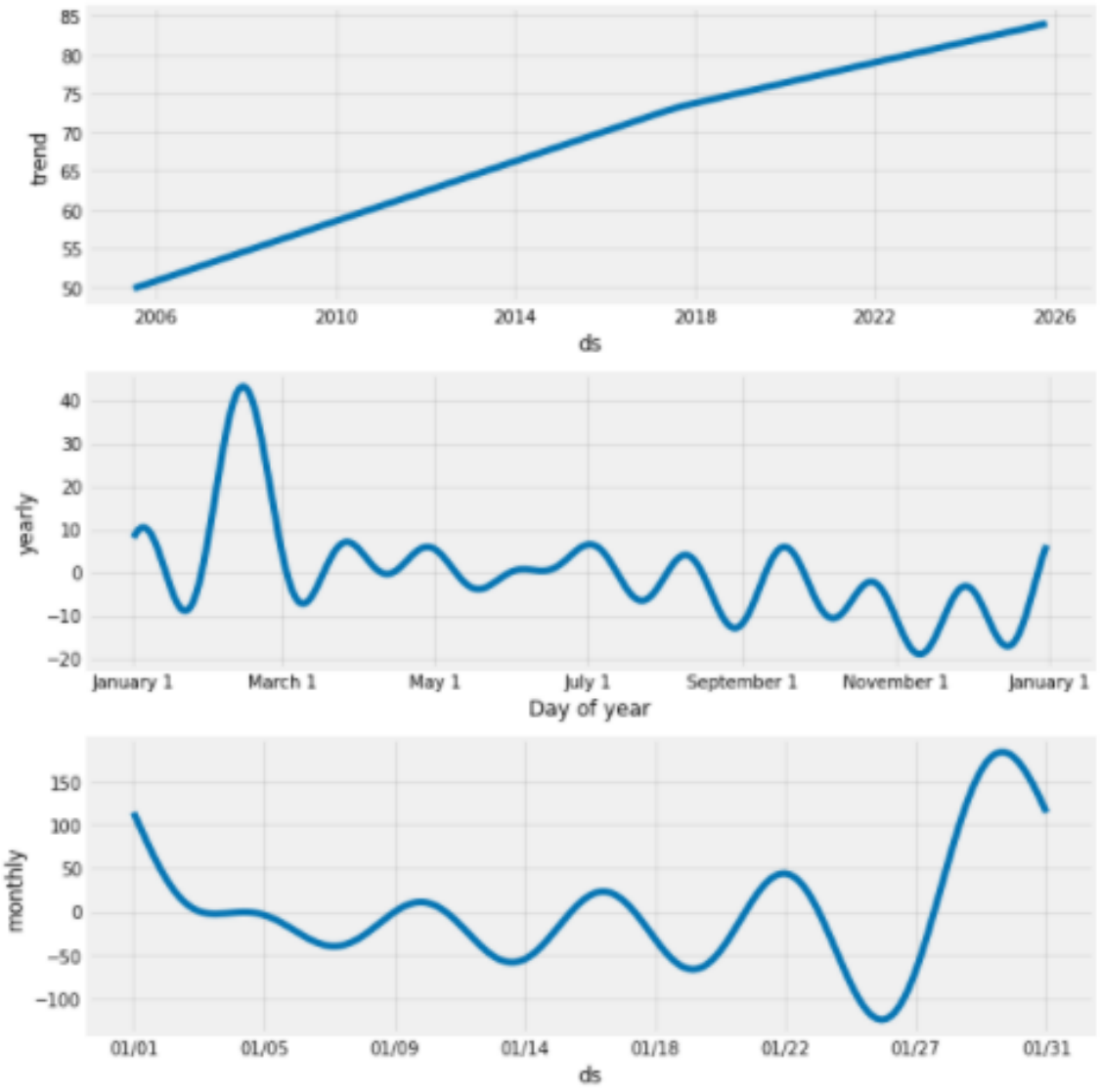
```
[ ] ### intiialize the Model
    model = Prophet()
    model.fit(df_close)# fit the model using all data
    future = model.make_future_dataframe(periods=60, freq='MS') # 'MS' used here is month-start, means the data point is placed on the start of each month.

    INFO:fbprophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
    INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
```

Here, we make the future dataframe of 5 years, that's why we've provided the period as 60 months.

Prophet will by default fit weekly and yearly seasonalities, if the time series is more than two cycles long. It will also fit daily seasonality for a sub-daily time series. You can add other seasonalities (monthly, quarterly, hourly) using the add_seasonality method (Python). Below we've added the seasonality as monthly.

Let’s see the components of Fbprophet:

```
# Let's plot the components of our model
model.plot_components(forecast);
```

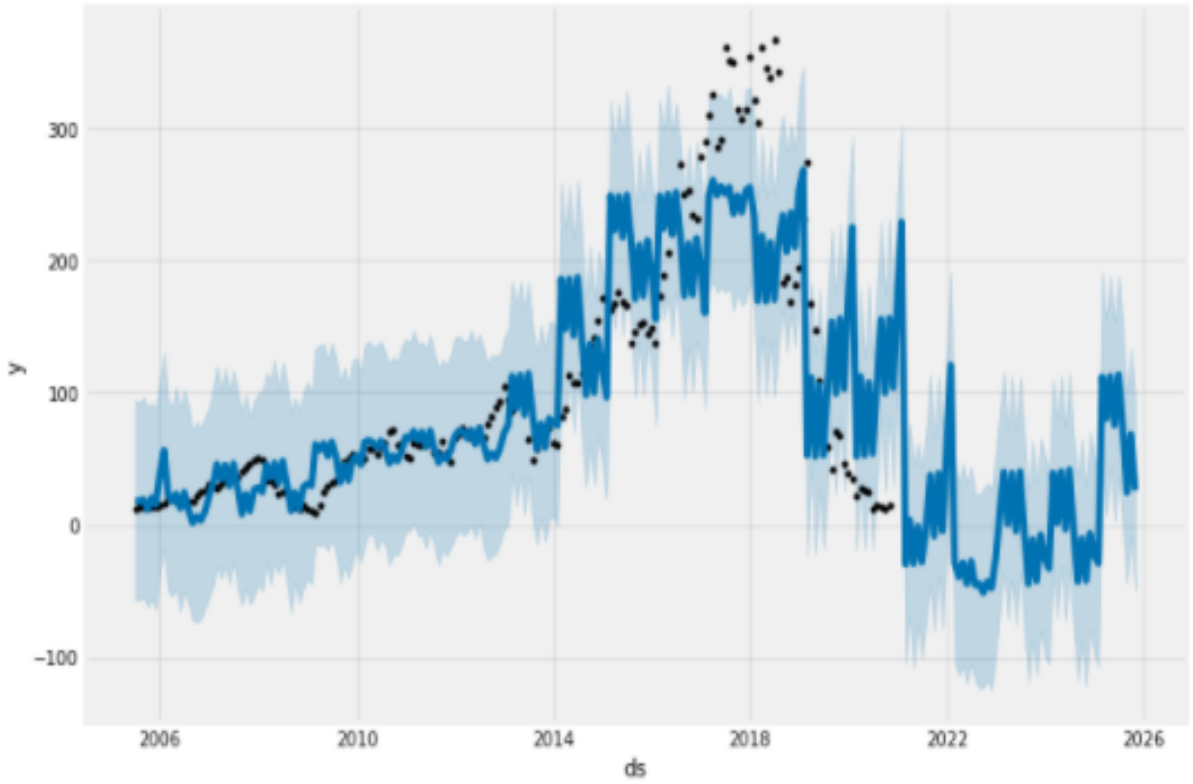



```
# Let's see the components of our model
model.component_modes
```

```
{'additive': ['monthly',
  'yearly',
  'additive_terms',
  'extra_regressors_additive',
  'holidays'],
 'multiplicative': ['multiplicative_terms', 'extra_regressors_multiplicative']}
```

Forecasting model predictions:

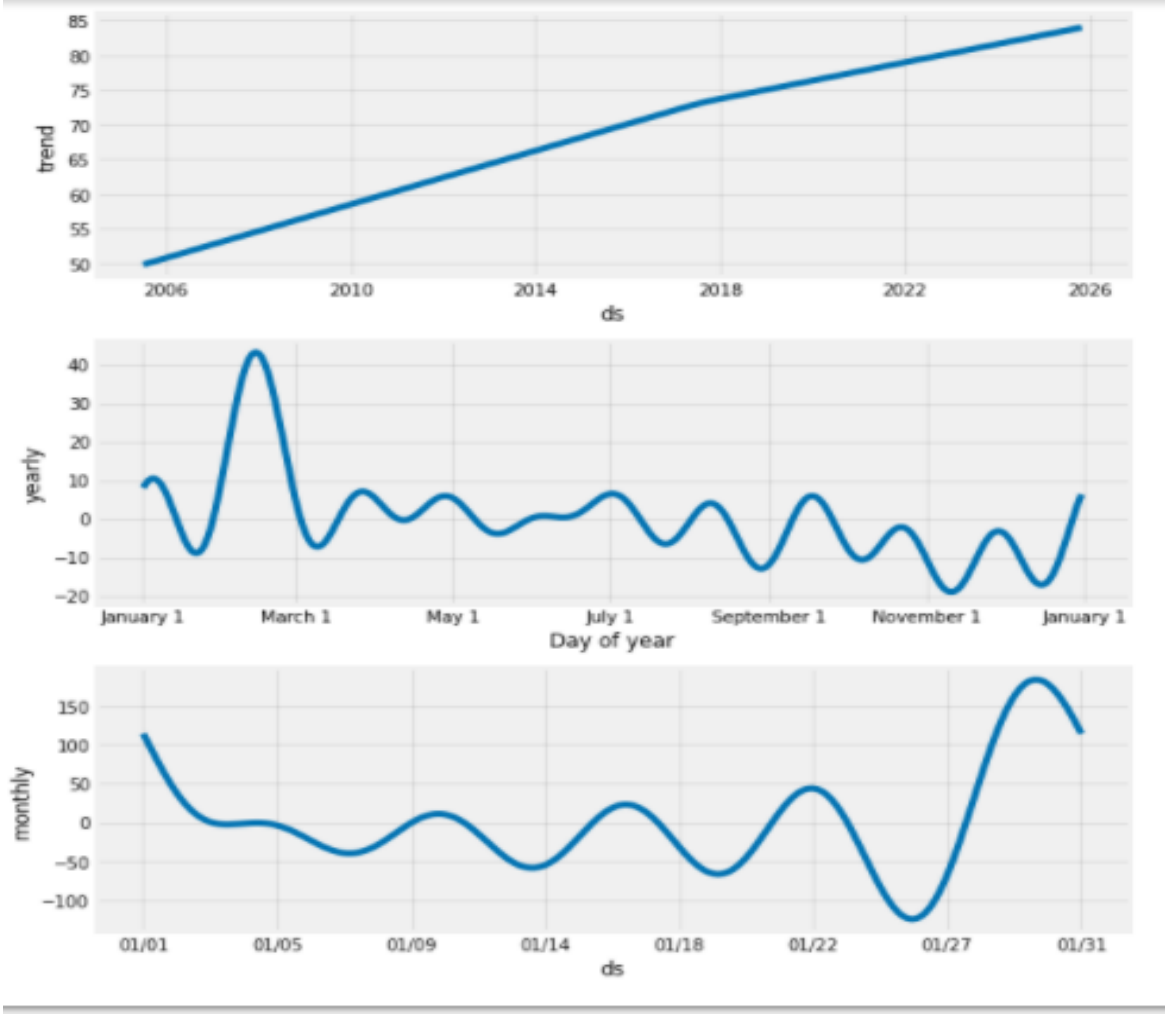
Now, we plot the forecast by calling the **Prophet.plot method** and passing in our forecast dataframe.



From the above plot we can see that there is an increase in trend in the year 2024 and 2025 onwards.

Now,

Plotting prediction components of our model to see the trend, yearly seasonality, and weekly seasonality of the time series.



Observed Trends:

- YES BANK's stock price is showing signs of upper trend yearly.
- YES BANK's stock price shows upper trend signs during February and in December month it tends to give low stock prices.
- YES BANK's stock price is showing signs of upper trend during the end of the month.

Cross Validation for time series:

Prophet includes functionality for time series cross validation to measure forecast error using historical data. This is done by selecting cutoff points in the history, and for each of them fitting the model using data only up to that cutoff point. We can then compare the forecasted values to the actual values.

Parameters used:(model, horizon="365 days", period='180 days', initial='1095 days')

Where,

- model: Prophet class object. Fitted Prophet model.
- horizon: string with pd.Timedelta compatible style, e.g., '5 days', '3 hours', '10 seconds'.
- period(spacing between cutoff dates) : string with pd.Timedelta compatible style. Simulated forecasts will be done at every period. If not provided, 0.5 * horizon is used.
- initial(the size of the initial training period) : string with pd.Timedelta compatible style. The first training period will include at least this much data. If not provided, 3 * horizon is used.
- Here, the output of cross_validation is a dataframe with the true values y and the out-of-sample forecast values yhat, at each simulated forecast date and for each cutoff date.

	ds	yhat	yhat_lower	yhat_upper	y	cutoff
0	2008-08-01	40.245191	38.961651	41.462829	26.83	2008-07-02
1	2008-09-01	41.067000	39.885731	42.314863	24.13	2008-07-02
2	2008-10-01	40.789432	39.487402	42.025925	13.58	2008-07-02
3	2008-11-01	42.973994	41.641828	44.234486	12.26	2008-07-02
4	2008-12-01	40.762470	39.520050	42.014857	15.03	2008-07-02

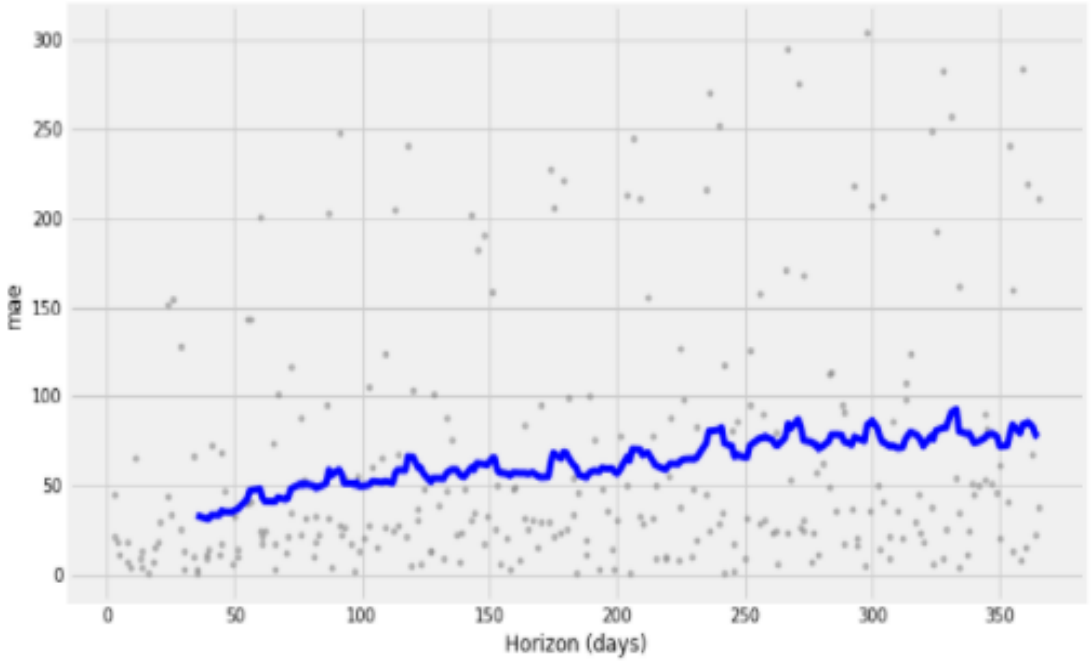
Evaluation Metrics:

Used the performance_metrics utility to compute the Mean Squared Error(MSE), Root Mean Squared Error(RMSE),Mean Absolute Error(MAE), Mean Absolute Percentage Error(MAPE) and the coverage of the the yhat_lower and yhat_upper estimates.

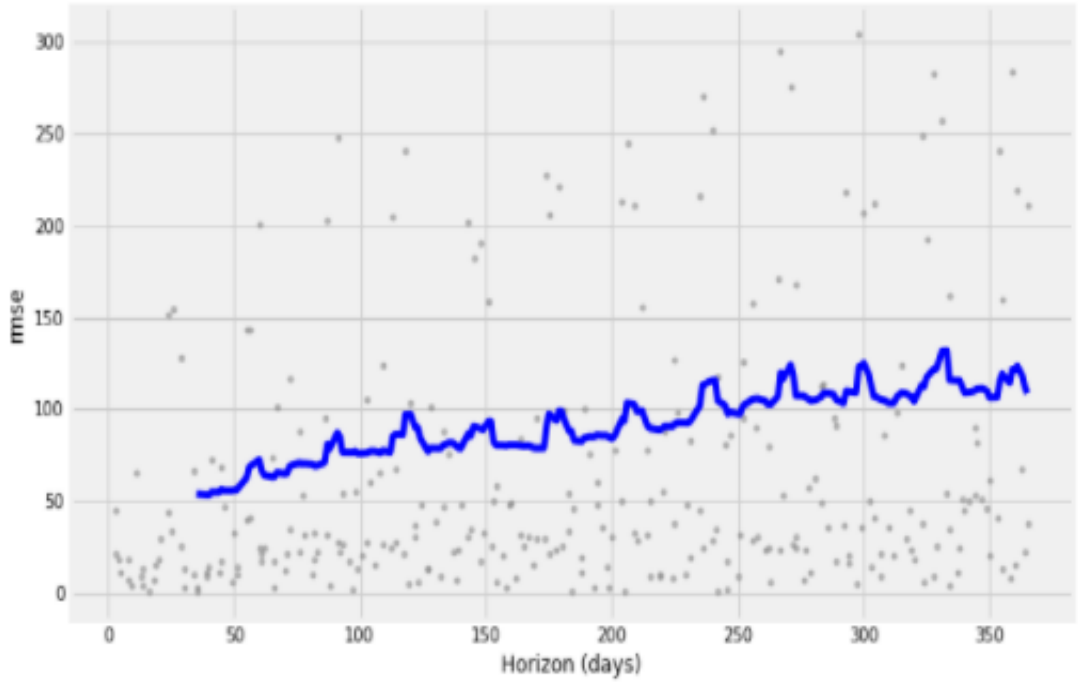
	horizon	mse	rmse	mae	mape	mdape	coverage
0	35 days	2909.646693	53.941141	32.995170	0.445647	0.209830	0.464286
1	39 days	2827.846854	53.177503	31.325250	0.381288	0.154778	0.500000
2	40 days	2823.260689	53.134364	31.181460	0.369197	0.154778	0.500000
3	41 days	3005.183478	54.819554	33.352355	0.377365	0.183549	0.464286
4	44 days	3002.675408	54.796673	33.301062	0.380515	0.207394	0.428571

Plotting Metrics:

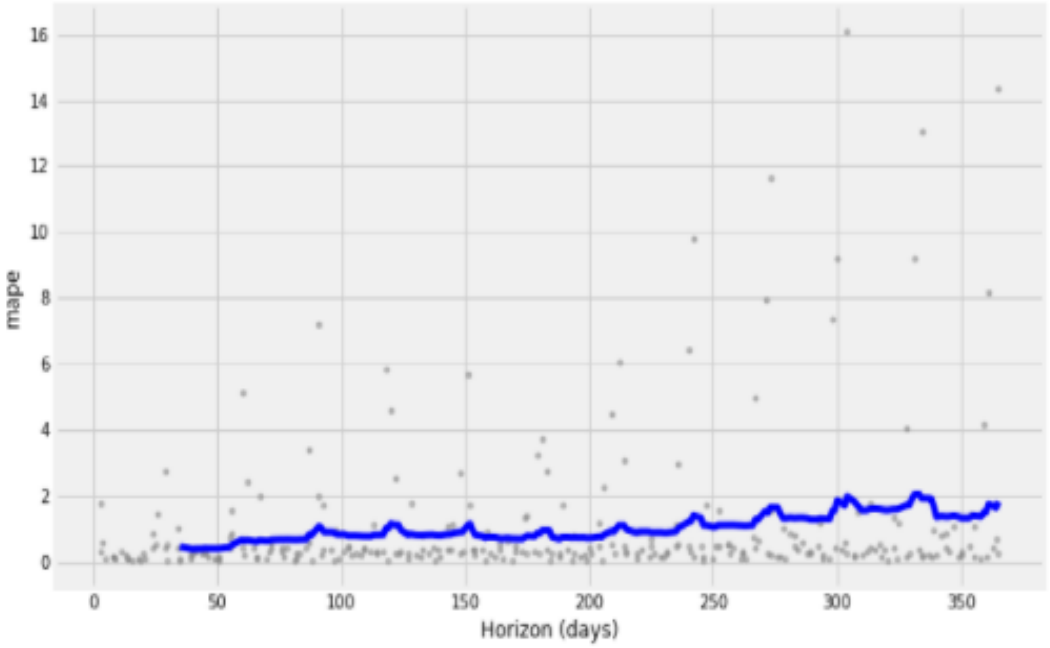
- **Mean Absolute Error (MAE)**



- **Root Mean Square Error (RMSE)**



- **Mean Absolute Percentage Error (MAPE)**



Here as shown for MAPE(Mean Absolute Percentage Error). Dots show the absolute percent error for each prediction in df_cv. The blue line shows the MAPE, where the mean is taken over a rolling window of the dots.

We see for this forecast that errors around 3% are typical for predictions one month into the future, and that errors increase up to around 18-19% for predictions that are a year out.

CHALLENGES

- Because of the small dataset it's difficult to get good model accuracy.
- Making the data stationary for time series models.
- Adding a monthly trend to fbprophet was a difficult task.

CONCLUSION AND FUTURE SCOPE

- There is an increase in trend of Yes Bank stock price till 2018 and then a sudden decrease.
- Using a regression approach we get the best performing model as Linear Regression model with accuracy of 94.60% which is excellent.
- On the other hand, using the time series approach for the FBprophet forecast the error of 3% is typical for predictions one month into the future, and that errors increase up to around 18-19% for predictions that are a year out.
- Fbprophet is the best performing time series model in terms of accuracy.
- February is the month where there are more upward trends for the stock.
- July also shows the upward trend but not as much as February.
- November and December are months with the most downward pressure.
- On a yearly basis there is a decrease in trend from 2021 to 2024 then an increase in trend from 2024 onwards.

Contributors Role

Debaprasad Mohapatra:

- Performed EDA (Exploratory Data Analysis)
- Build Regression Models such as, Linear Regression, Random Forest, XG-boost, knn(K-Nearest neighbors).

Saurabh Yadav:

- Performed Data preprocessing
- Build Time Series Models such as Auto-ARIMA and Fbprophet.