

# Yes Bank's Stock Price Prediction using Machine Learning Model

**Shubhan Deshmukh Email –Id: - [Shubhamdeshmukh278@gmail.com](mailto:Shubhamdeshmukh278@gmail.com)**

**Abstract:** *To determine the YES bank's stock's future value on the national stock exchange. The advantage of a successful prediction of a stock's future price could result in insignificant profit. The efficient-market hypothesis recommends that stock costs mirror all right now accessible data and any value changes that are not founded on recently uncovered data subsequently are unpredictable. We have to build models which help us to predict the future stock prices. Models like ARIMA, FbProphet and some regression models are the machine learning models which are some of the best time series machine learning models. Each has its own perspective to choose for a time series machine learning model and use them for the prediction. We will be using all models for prediction of stock price and compute the results and compare them for best accuracy and performance.*

**Keywords:** Stock price prediction, FbProphet model, Arima Model, Regression models

## I. Problem Statement

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.

## II. Data Description

- **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 refers to the maximum prices in a given time period.
- **Low:** Low refers 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.

## III. Introduction

Time series forecasting has become a trend in recent years by the researchers. There are many interesting methods and algorithms, which have been proposed for prediction. Whether one wishes to predict some stock prices or some the consumption of electricity, time plays an important factor that is widely considered for the time Series Models. For example, it would

be very interesting to know when the price of the stock is going to rise up but also when it is going to rise up. A time series simply can be defined as a series of data points which have been ordered in a time orderly manner. In time series, time is usually the independent variable and the goal is usually to make a forecast for the future. Our goal here is to predict the stock price of Yes Bank more accurately.

#### 4. Exploratory Data Analysis

##### A) Data Cleaning :-

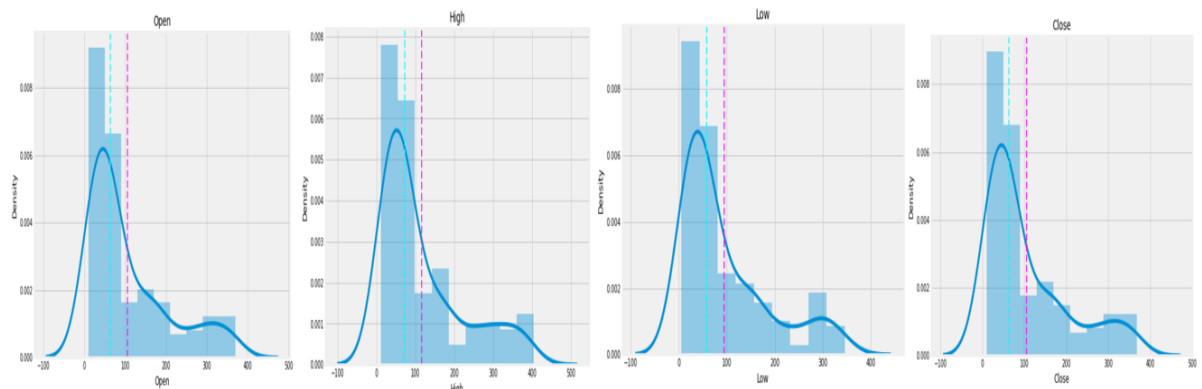
The Given Date in data is of format MMM-YY is converted to proper date of YYYY-MM-DD and given date column has dtype as object converting it into date time format.

##### B) Null values Treatment :-

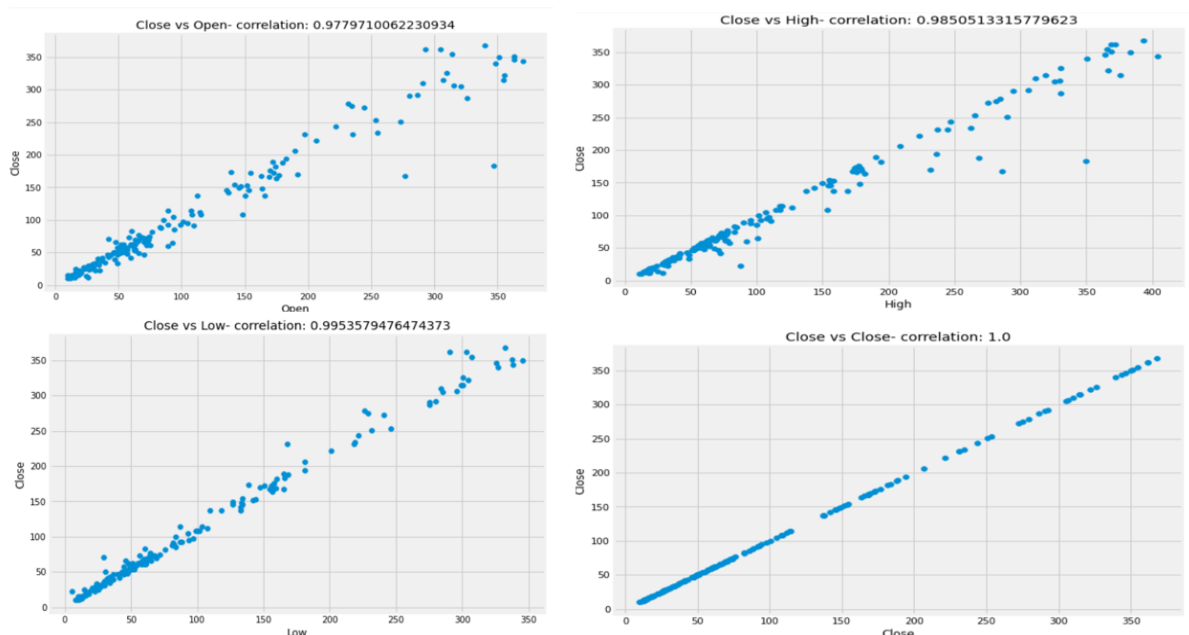
Our dataset does not contain null values which might tend to disturb our accuracy hence we dropped them at the beginning of our project in order to get a better result.

##### C) Data Visualization :-

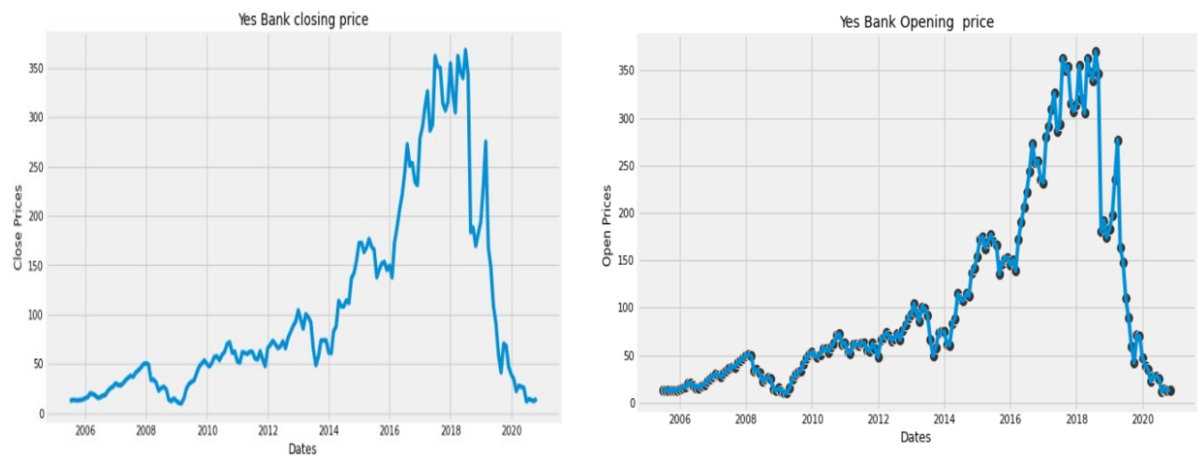
a) Univariate Analysis – In the dataset all features histogram plot are right skewed.



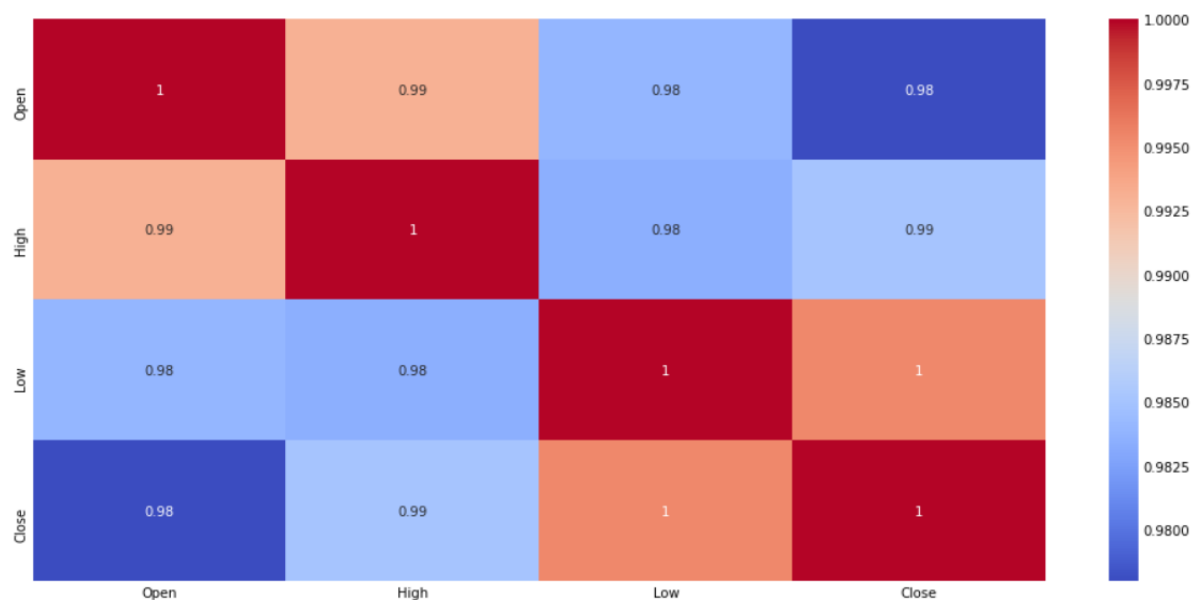
b) Bivariate Analysis – Bivariate analysis showed that close price has high correlation with other features.



C) Close Price and Open prices – Bar plot shows that the stock close price and Open prices decreased after year 2018 the probably because of Rana Kapoor case and hit the stock price heavily.



D) Correlation –All variables show the highest correlation among them.



D) Setting Date column as Index –

	Open	High	Low	Close
Date				
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

## V. Modelling

### A) Linear Regression –

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression algorithm shows a linear relationship between a dependent (y) (in our case is Close Price) and one or more independent (in our case Open, Low, high) variables, hence called as linear regression.

a) **Lasso Regression** - lasso (least absolute shrinkage and selection operator) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model. When we deployed the lasso regression model's result are given below table.

```
MSE : 0.16979450921198339
RMSE : 0.41206129302809236
MAE : 0.35059764622064893
R2 : 0.8200828459676572
```

Here we have an R-Square value 82.00%.

b) Ridge Regression - Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values to be far away from the actual values.

```
MSE : 0.16979450921198339
RMSE : 0.41206129302809236
MAE : 0.35059764622064893
R2 : 0.8200828459676572
```

Here we have an R-Square value of 82.00%.

**Cross Validation** :- Basically Cross Validation is a technique using which Model is evaluated on the dataset on which it is not trained i.e. it can be a test data or can be another set as per availability or feasibility. Here our best parameter is 'alpha': 0.01 and cv = 3. When we deployed cross validation, the lasso **regression** model's results are given below table. Cross Validation for **ridge regression** we found the best parameter 'alpha': 20 and cv = 5.

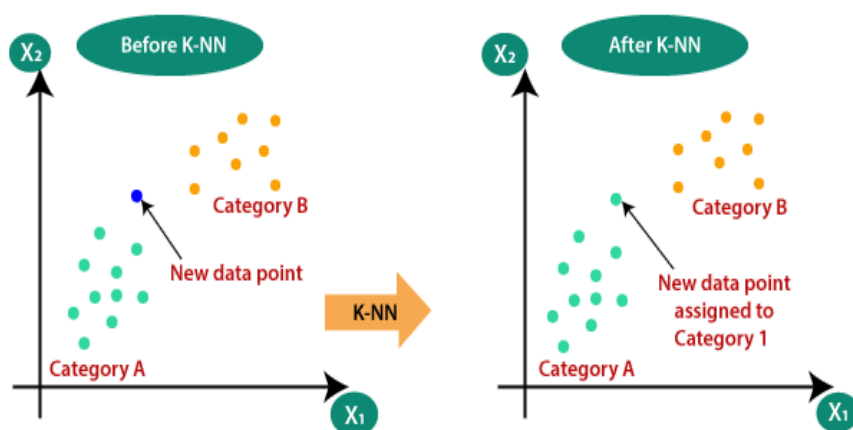
MSE : 0.16979450921198339	MSE : 0.17460972631724542
RMSE : 0.41206129302809236	RMSE : 0.41786328663481004
MAE : 0.35059764622064893	MAE : 0.3541738283648222
R2 : 0.8200828459676572	R2 : 0.8149805599064218

Here we have R-Square value 82.00% for lasso regression and 81.49% for ridge regression.

## B) KNN Regression -

KNN (Nearest neighbors) Regressor: - KNN regression is a non-parametric method that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same neighborhood. A simple implementation of KNN regression is to calculate the average of the numerical target of the  $K$  nearest neighbors. Another approach uses an inverse distance weighted average of the  $K$  nearest neighbors. In KNN regression, the KNN algorithm is used for estimating continuous variables. One such algorithm uses a weighted average of the  $k$  nearest neighbors, weighted by the inverse of their distance. This algorithm works as follows:

1. Compute the Euclidean from the query example to the labeled examples.
2. Order the labeled examples by increasing distance.
3. Find a heuristically optimal number  $k$  of nearest neighbors, based on RMSE. This is done using cross validation.
4. Calculate an inverse distance weighted average with the  $k$ -nearest multivariate neighbors.



## K-Fold Cross-Validation

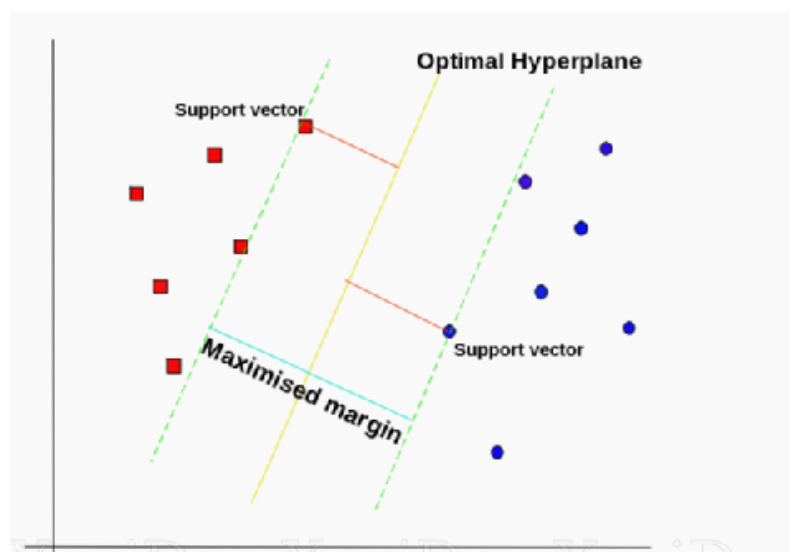
Cross-validation is when the dataset is randomly split up into 'k' groups. One of the groups is used as the test set and the rest are used as the training set. The model is trained on the training set and scored on the test set. Then the process is repeated until each unique group has been used as the test set. In our case we take  $n\_splits$  value = 7 and value of  $n$  neighbors is 2. Our model gives accuracy about 92.01%

```
#K-Fold cross validation
knn_kfold = model_selection.KFold(n_splits=7, random_state=42)
results_kfold = model_selection.cross_val_score(knn_model, X_test, y_test.astype('int'), cv=knn_kfold)
print("Accuracy: ", results_kfold.mean()*100)
```

Accuracy: 92.01159951159951

### C) SVM Model –

SVM:-Support Vector Regression is a supervised learning algorithm that is used to predict discrete values. Support Vector Regression uses the same principle as the SVMs. The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyperplane that has the maximum number of points. Support Vector Regression uses the same principle of Support Vector Machines. In other words, the approach of using SVMs to solve regression problems is called Support Vector Regression or SVR.



A kernel is a function which places a low dimensional plane to a higher dimensional space where it can be segmented using a plane. In other words, it transforms linearly inseparable data to separable data by adding more dimensions to it.

- Linear kernel: Dot product between two given observations
- Polynomial kernel: This allows curved lines in the input space
- Radial Basis Function (RBF): It creates complex regions within the feature space

In our model we used the linear kernel.

```
#importing SVM model
from sklearn.svm import SVR
svm_regressor = SVR(kernel='linear')#Linear
#Fitting the model
svm_model=svm_regressor.fit(X_train,y_train)
#prediction
y_svm_pred=svm_model.predict(X_test)
```

## VI. Evaluation of Regression models

Evaluating our models we will consider the following metrics-

- 1) **MAE** – Mean Absolute Error is the absolute difference between the target variable and value predicted by the model .The MAE is more robust to outliers.
- 2) **MSE** – Mean Squared Error is one of the most preferred metrics for a regression model .It is simply an averaged squared difference between the target value and value predicted by the regression model.
- 3) **RMSE** – Root mean squared value is the square root averaged squared difference between the target value and value predicted by the regression model.
- 4) **MAPE** – Mean Absolute Percentage Error is also known as mean absolute percentage deviation is a measure of prediction accuracy of forecasting methods in statistics.
- 5) **R-Square** – The metric that helps us to compare the current model with a constant model baseline and tell us how much our model is better.

	Model_Name	MAE	MSE	RMSE	MAPE	Rsquare
1	KNeighborsRegressor	0.06	0.01	0.09	1.82	0.99
0	LinearRegression	0.35	0.17	0.41	9.54	0.82
2	SVR	0.33	0.20	0.45	9.87	0.79

Observation from above table:

- **KNN** Regressor gives lowest MAE, MSE, RMSE, MAPE and best  $R^2$  value.
- Overall we can say that **KNN** is the **best model** among all regression models which gives around **92.00%** accuracy of predicting stock price of our dataset.

## **VII. Time Series Forecasting**

A time series simply can be defined as a series of data points which have been ordered in a time orderly manner. In time series, time is usually the independent variable and the goal is usually to make a forecast for the future.

However, there are some aspects which need to be considered when dealing with time series.

Is it stationary?

Is there seasonality?

Is the target variable autocorrelated?

### **a) Stationarity**

Stationarity is an important characteristic of the time series. If the statistical properties do not change over the time, the time series can be referred to as stationary. It can be also depicted as that it has constant mean and variance, and the covariance is independent of time. Since we see a growing trend in the stock prices, or the volatility might increase over the time period (meaning the variance is changing).

### **b) How to test if a process is stationary?**

The Dickey-fuller test is used, which is a statistical test that we run to determine if a time series process is stationary or not. The dickey-fuller test is a test which tests the null hypothesis that a union root is present.

If it is, then  $p > 0$ , and the process is stationary.

Otherwise,  $p = 0$ , the null hypothesis is rejected, and the process is considered to be stationary.

### **c) Seasonality**

The periodic fluctuations in the time series are referred to as the seasonality. For example, the online sales of a company or website increased during some vacation or festival like Diwali before slowing down again.

The seasonality can also be derived from the autocorrelation plot if it has a sinusoidal shape. The look at the period can give the length of the season.

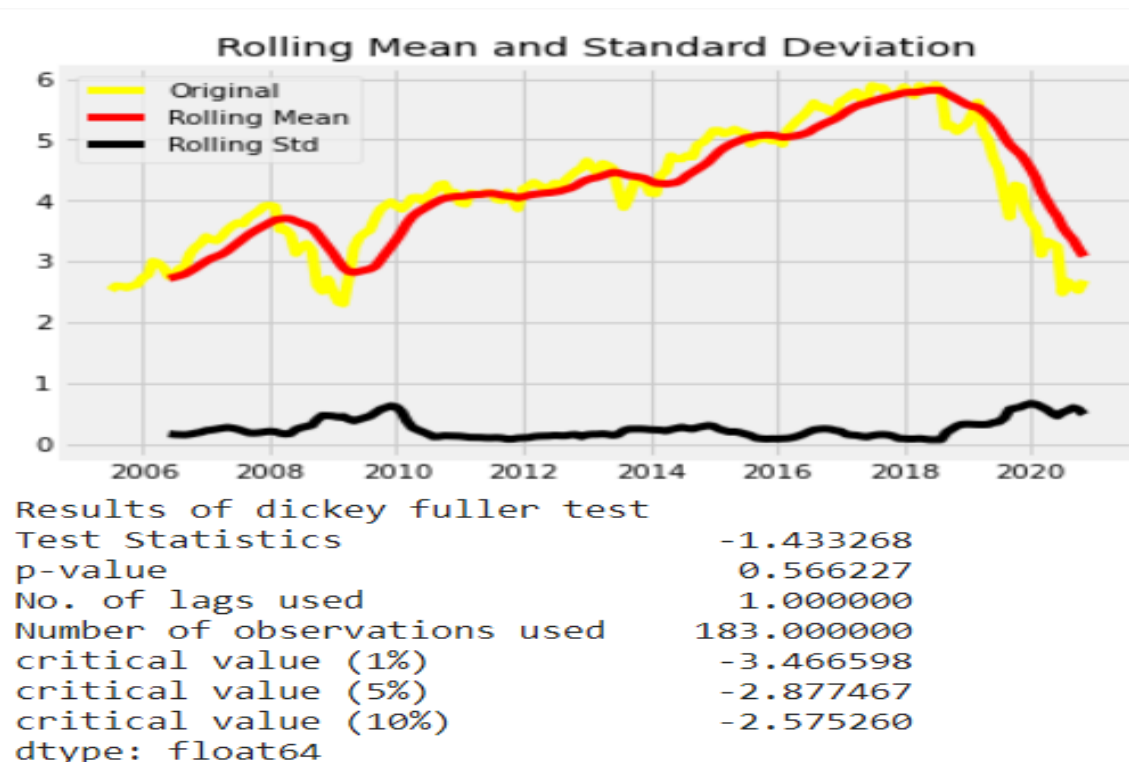


#### d) Autocorrelation

The autocorrelation informally can be described as the similarity between observations as a function of the time lag between them. If the autocorrelation plot looks like a sinusoidal function, it hints for the seasonality and you can find its value by finding the period in the plot.

#### ADF test

We'll use the Augmented Dickey Fuller (ADF) test to check if the price series is stationary.



Here we can 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.

So let us separate Trend and Seasonality from the time series.

#### How to remove Stationarity?

Subtract the previous value from the current value. Now if we just differ once, we might not get a stationary series so we might need to do that multiple times.

And the minimum number of differencing operations needed to make the series stationary needs to be imputed into our **ARIMA** model.

## VIII. ARIMA MODEL

### ARIMA

#### So what exactly is an ARIMA model?

ARIMA, short for 'Auto Regressive Integrated Moving Average' is actually a class of models that 'explains' a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values.

The **AR** stands for the autoregressive and refers to using the lagged values of our target variable to make our prediction. For example, we can use the stock price of today, yesterday's and day before yesterday's prices of stock to forecast tomorrow's stock price. That would be an AR (3) model as it uses 3 lagged values to make its prediction.

The **I** stands for integrated. It means that instead of taking the raw values from the target, we are differencing them. For example, our stock price prediction model would try to forecast tomorrow changes in the stock prices. The reason we need this is that many time series exhibit a trend, making the raw values non-stationary and when we take the difference, it makes our y variable more stationary.

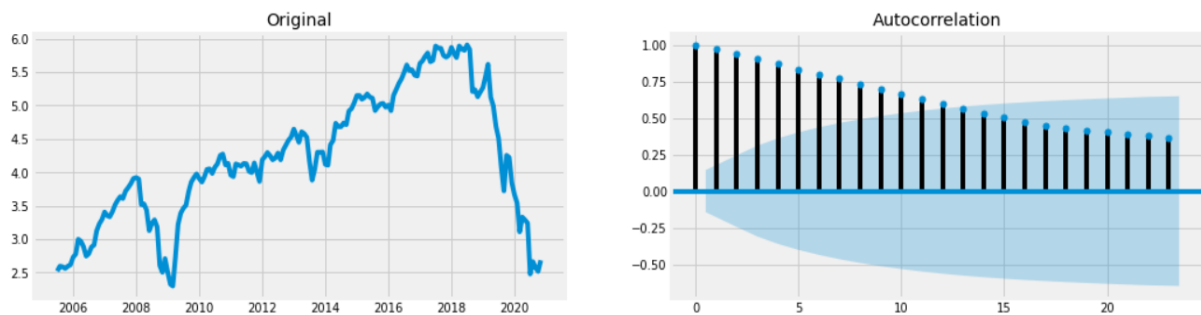
The **MA** stands for the Moving average. A moving average model takes the lagged prediction errors as inputs. This kind of parameter is not a directly observable parameter unlike the other parameters and this parameter. At a high level, feeding this model errors back to itself serves to push it somewhat toward the correct values (the actual Y values).

An ARIMA model is characterized by 3 terms (p, d, q) :

- p is the order of the AR term
- d is the number of differencing required to make the time series stationary
- q is the order of the MA term

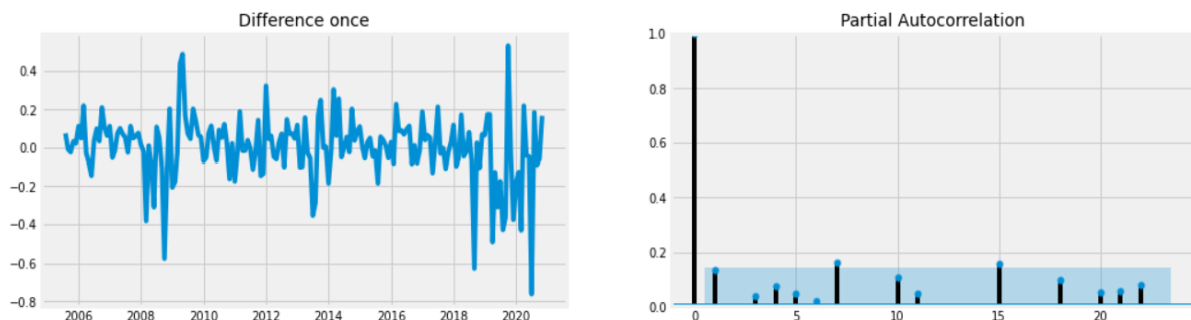
As we see in the parameters required by the model, any stationary time series can be modelled with ARIMA models.

### a) Determining the value of $d$ –



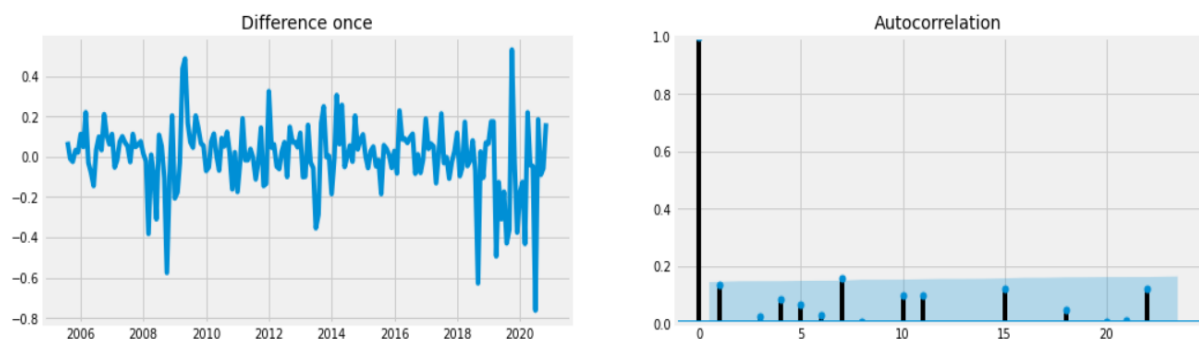
The above fig shows the original series graph here to determine the value of  $d$ . We import the 'ndiffs' from pmdarima that will give us the perfect value of  $d$  and this gives us the value for time series is  $d = 1$ .

### b) Determining the value of $p$ –



$P$  is the order of the Auto Regressive (AR) term. It refers to the number of lags to be used as predictors. We can find out the required number of AR terms by inspecting the Partial Autocorrelation (PACF) plot. The partial autocorrelation represents the correlation between the series and its lags. We can observe that the PACF lag 6 is significant as it's above the significance line.

### C) Determining the value of $q$ –



$q$  is the order of the Moving Average (MA) term. It refers to the number of lagged forecast errors that should go into the ARIMA Model. We can look at the ACF plot for the number of MA terms. From the above graph we can see that the  $q = 6$ .

Predicting the algorithm –

```
from statsmodels.tsa.arima_model import ARIMA

# ARIMA Model
model = ARIMA(new_df.Close, order=(6,1, 6))
result = model.fit(dis=-1)
print(result.summary())
```

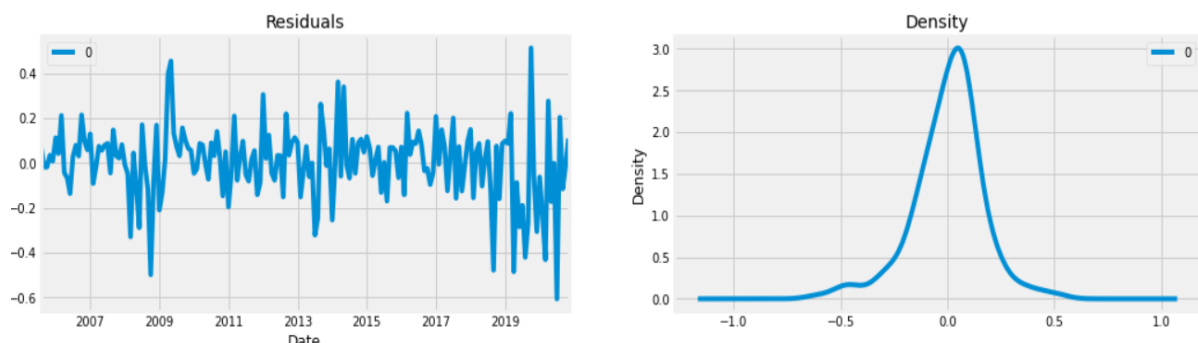
```
=====
                        ARIMA Model Results
=====
Dep. Variable:          D.Close      No. Observations:          184
Model:                 ARIMA(6, 1, 6)  Log Likelihood             71.391
Method:                css-mle        S.D. of innovations         0.162
Date:                  Thu, 05 Aug 2021  AIC                        -114.782
Time:                  05:36:45         BIC                        -69.772
Sample:                08-01-2005      HQIC                       -96.539
                             - 11-01-2020

=====
```

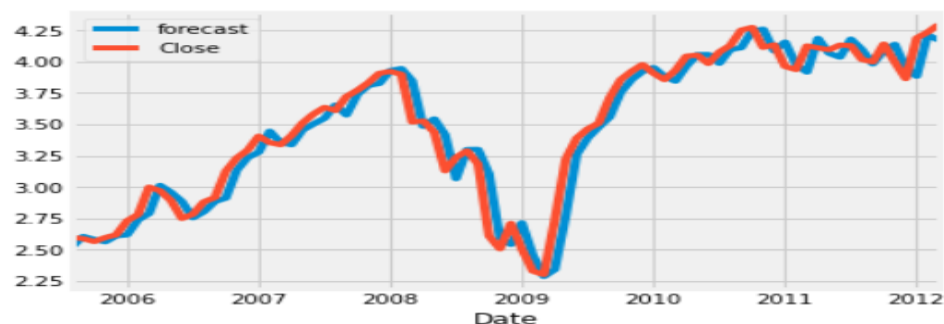
	coef	std err	z	P> z	[0.025	0.975]
const	0.0013	0.011	0.112	0.910	-0.021	0.023
ar.L1.D.Close	0.5286	0.081	6.521	0.000	0.370	0.687
ar.L2.D.Close	0.6097	0.124	4.923	0.000	0.367	0.852
ar.L3.D.Close	-0.2181	0.120	-1.814	0.070	-0.454	0.018
ar.L4.D.Close	-0.4472	0.113	-3.952	0.000	-0.669	-0.225
ar.L5.D.Close	-0.2170	0.130	-1.668	0.095	-0.472	0.038
ar.L6.D.Close	0.7322	0.086	8.509	0.000	0.564	0.901
ma.L1.D.Close	-0.3982	0.061	-6.556	0.000	-0.517	-0.279
ma.L2.D.Close	-0.7454	0.088	-8.469	0.000	-0.918	-0.573
ma.L3.D.Close	0.1077	0.071	1.516	0.130	-0.032	0.247
ma.L4.D.Close	0.6740	0.082	8.221	0.000	0.513	0.835
ma.L5.D.Close	0.2905	0.081	3.588	0.000	0.132	0.449
ma.L6.D.Close	-0.9286	0.052	-17.886	0.000	-1.030	-0.827

The model summary reveals a lot of information. The table in the middle is the coefficient table where the values under ‘coef’ are the weights of the respective terms.

Let’s plot the residuals to ensure there are no patterns (that is, look for constant mean and variance). Here below fig we can see that the residual errors seem fine with near zero mean and uniform variance.



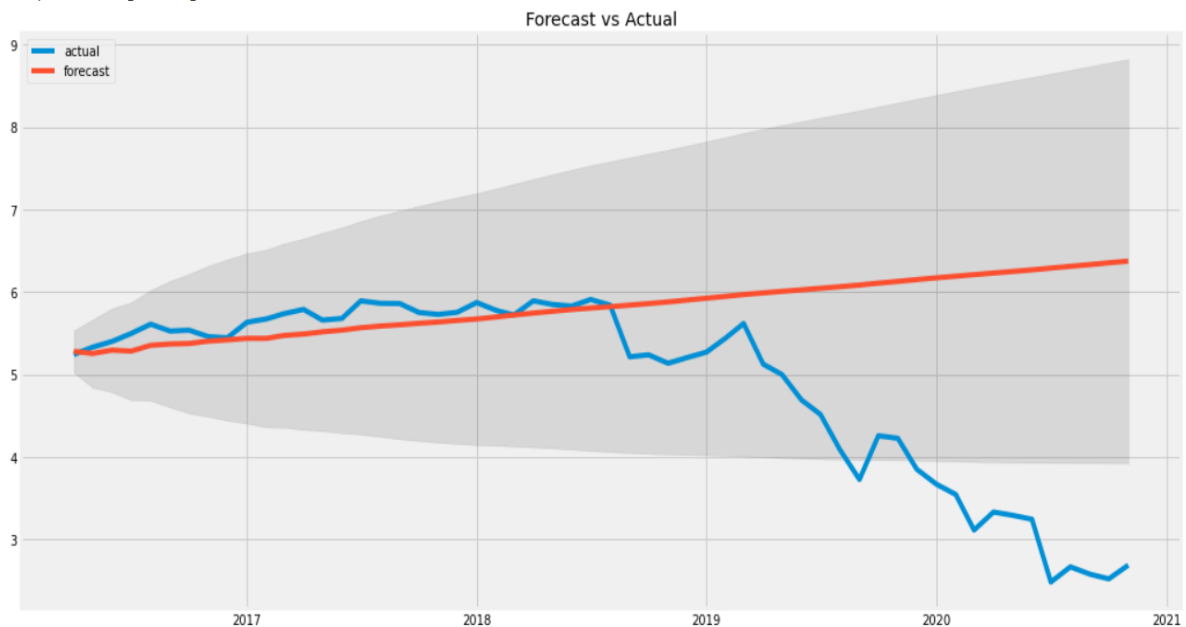
. Let’s plot the actuals against the fitted values using plot\_predict ().



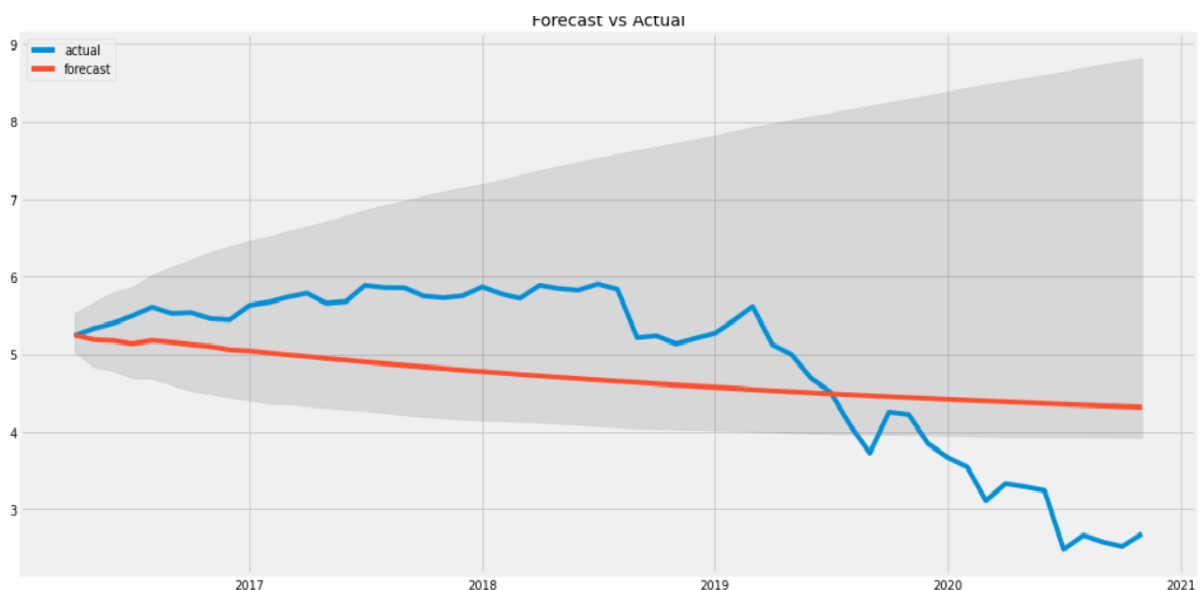
## ARIMA Model Forecast –

Here split the data into train-test and considering the next 56 months we get this below forecast graph. From the chart, the ARIMA(6,1,6) model seems to give a directionally correct forecast and the actual observed values lie within the confidence band. That seems fine.

```
129
56
<matplotlib.legend.Legend at 0x7f75e1a58550>
```



From the chart, the ARIMA (6, 1, 6) model seems to give a directionally correct forecast. That seems fine. But each of the predicted forecasts is consistently below the actuals. That means, by adding a small constant to our forecast, the accuracy will certainly improve. So, there is definitely scope for improvement. So, what I am going to do is to increase the order of differencing to zero, that is set  $d=0$  and iteratively increase  $p$  to up to 6 and then  $q$  up to 5.



## Evaluating the ARIMA model –

The commonly used accuracy metrics to judge forecasts are:

1. Mean Absolute Percentage Error (MAPE)
2. Mean Error (ME)
3. Mean Absolute Error (MAE)
4. Mean Percentage Error (MPE)
5. Root Mean Squared Error (RMSE)
6. Correlation between the Actual and the Forecast (correlation)
7. Min-Max Error (min max)

```
{'corr': 0.7370254134928292,  
 'mae': 0.8038407954019161,  
 'mape': 0.1741228551226052,  
 'me': 0.1955643978890437,  
 'minmax': 0.15823646827377502,  
 'mpe': 0.03489344569145076,  
 'rmse': 0.9170913846121032}
```

```
| # Accuracy  
errors = abs(fc2 - test)  
mape = 100 * (errors / test)  
accuracy = 100 - np.mean(mape)  
print('Accuracy:', round(accuracy, 2), '%.')
```

Accuracy: 80.8 %.

Here we can see that around **17.41% MAPE** (Mean Absolute Percentage Error) implies the model is about 81 % accurate in predicting the test set observations.

## IX. FbProphet Model

It is an additive based model, which is a procedure for forecasting time series data in which the non-linear trends are fit with yearly, weekly, and daily seasonality and as well as the holiday effects. The best results of this model are to be seen when the time series have strong seasonal effects and several seasons of historical data.

The prophet model is vigorous to missing data and the trend shifts and handles the outliers as well. Prophet is an open source software released by Facebook's Core Data Science team. The complex statistical modelling is handled by the Stan library and is a prerequisite for prophet.

Prophet follows the sklearn model API. The instance of the Prophet class is created and then calls its fit and predict methods. The data frame in the prophet model always has two columns: ds and y. The ds (date stamp) column should be a format expected by pandas, it can be of any format like YYYY-MM-DD HH:MM: SS for a timestamp and YYYY-MM-DD for a date. The y column must be numeric, and it should represent the measurement or attribute which you need to forecast.

### Fitting the model

Prophet has default fit weekly and yearly seasonality in our case we have added the seasonality as monthly. Seasonality has low uncertainty at the start of each month where there are data points, but has very high posterior variance in between. When fitting Prophet to monthly data, only make monthly forecasts, which can be done by passing the frequency into make\_future\_dataframe:

```
### initialize the Model
model = Prophet()
model.fit(stock_df) # 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.
```

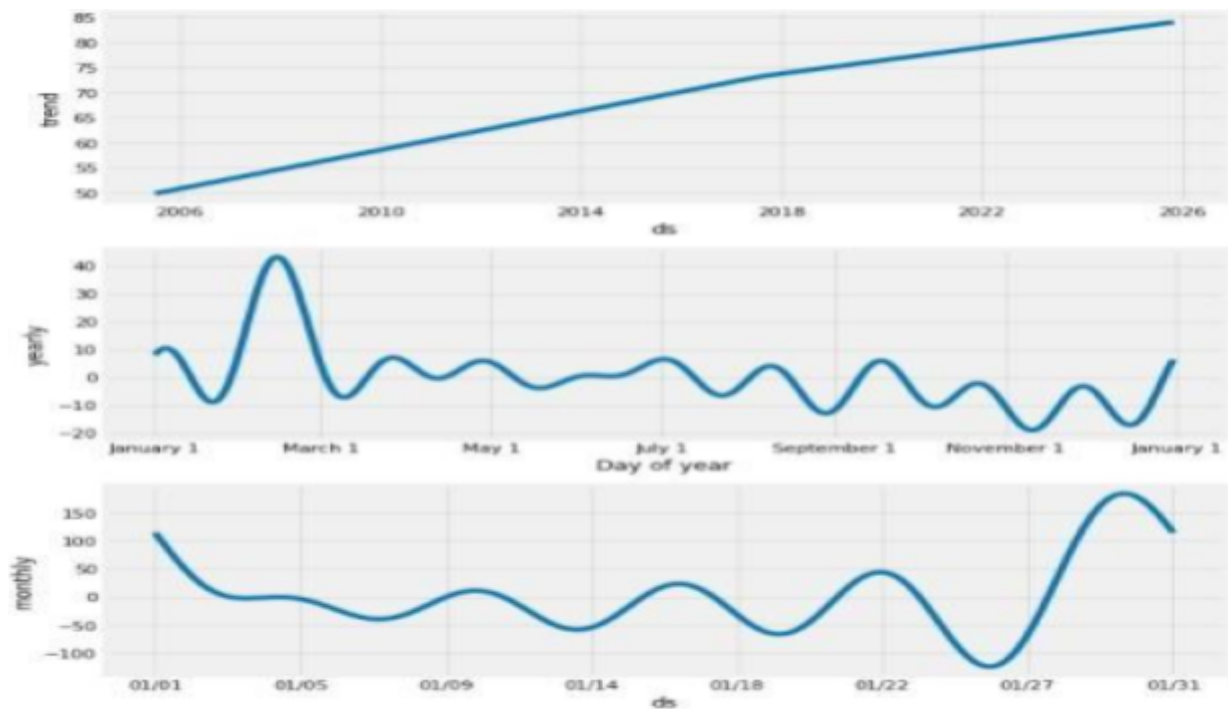
INFO:fbprophet:Disabling weekly seasonality. Run prophet with weekly\_seasonality=True to override this.

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. We have successfully fit the data to the Facebook Prophet model. Now let's have a look at the stock price prediction made by the model:

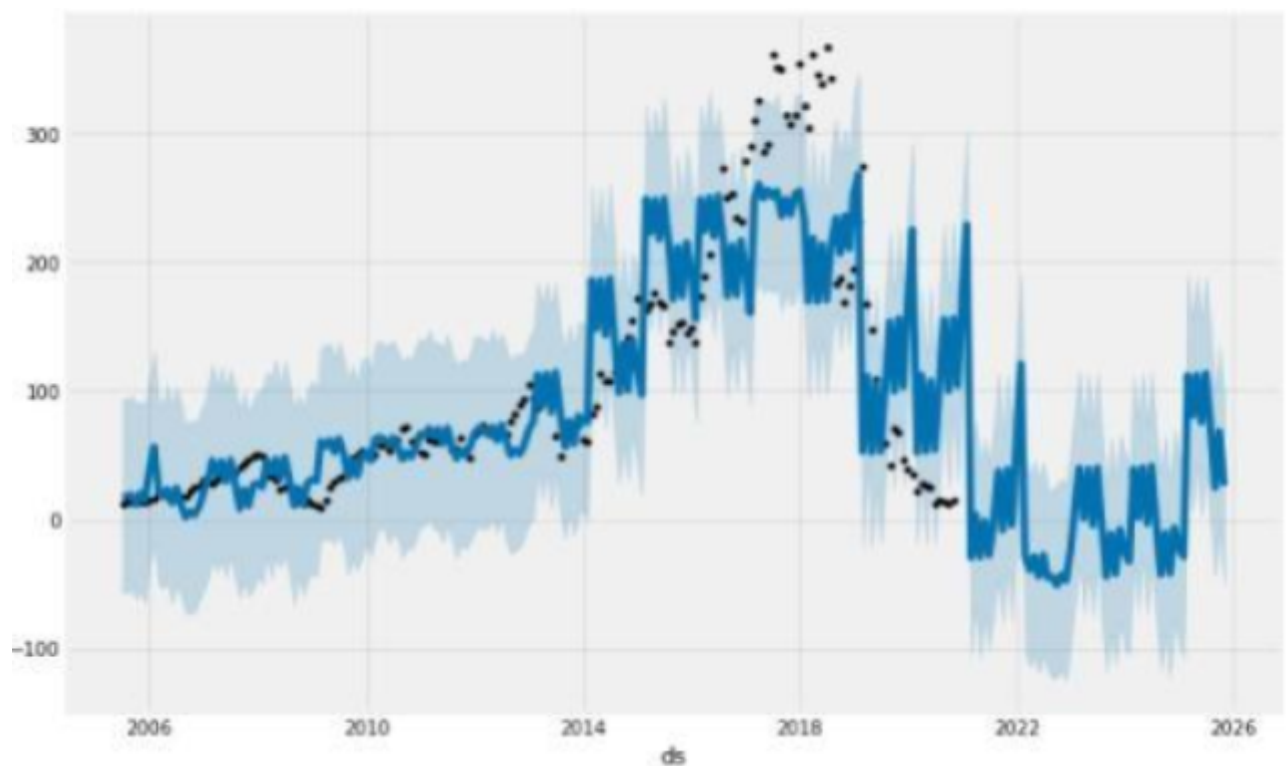
Components of our model:

- {'additive': ['monthly', 'yearly', 'additive\_terms', 'extra\_regressors\_additive', 'holidays'],
- 'multiplicative': ['multiplicative\_terms', 'extra\_regressors\_multiplicative']}



From above we can see that the trend is increasing and when we look at the yearly plot it reaches a peak value before March, probably the reason for that March ending. (Financial year end of India.)

Plot of predicted projection.



Here we can see that our model predicted overall well as most predicted values lie in the confidence interval.



## Cross Validation –


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

This cross validation procedure can be done automatically for a range of historical cut-offs using the cross validation function. We specify the forecast horizon (horizon), and then optionally the size of the initial training period (initial) and the spacing between cut-offs dates (period). By default, the initial training period is set to three times the horizon, and cut-offs are made every half a horizon.

### Parameters used:

- 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 cut-offs dates): string with pd.Timedelta compatible style. Simulated forecasts will be done at every period. If not provided,  $0.5 * \text{horizon}$  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,  $3 * \text{horizon}$  is used.

```
crv_df=cross_validation(model,horizon="365 days",period='180 days',initial='1096 days')
```

```
INFO:fbprophet:Making 24 forecasts with cutoffs between 2008-07-02 00:00:00 and 2019-11-02 00:00:00  
100%  24/24 [00:58<00:00, 2.42s/it]
```

We will consider the horizon for 1 year and initial 3 years data.

The output of cross validation is a dataframe with the true values  $y$  and the out-of-sample forecast values that, at each simulated forecast date and for each cut-offs date. In particular, a forecast is made for every observed point between cut-offs and cut-offs + horizon. This dataframe can then be used to compute error measures of that vs.  $y$ .

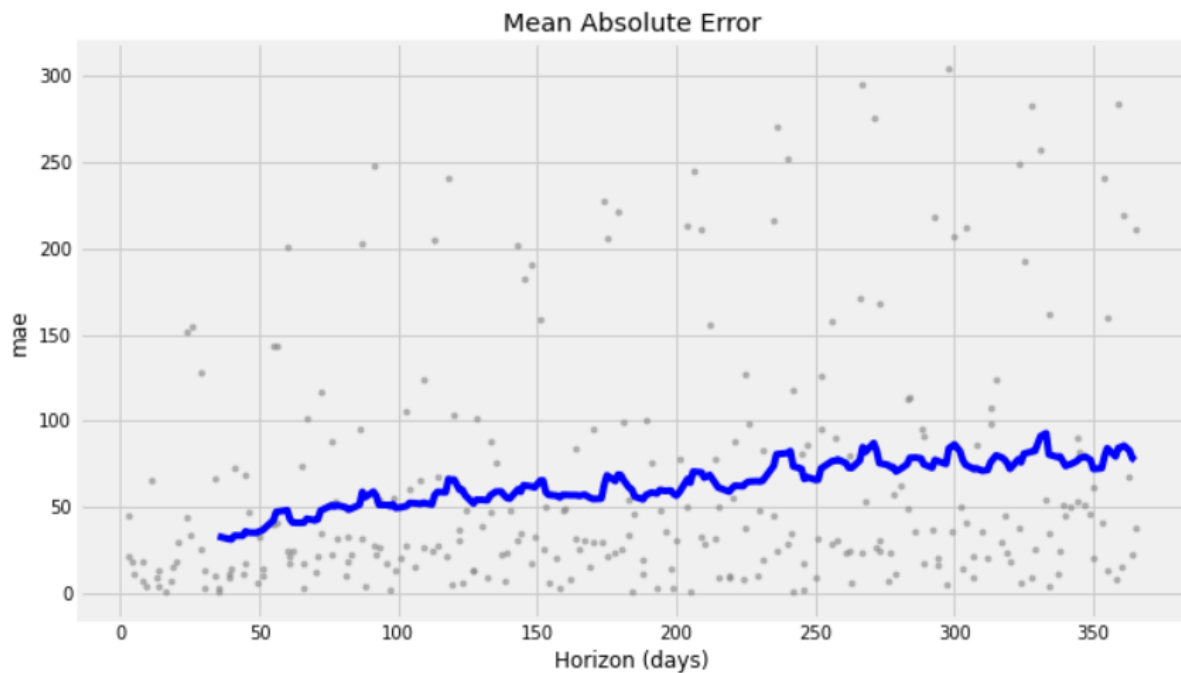
	ds	yhat	yhat_lower	yhat_upper	y	cutoff
0	2008-08-01	40.245191	38.961651	41.462829	26.83	2008-07-02
1	2008-09-01	41.067000	39.885731	42.314863	24.13	2008-07-02
2	2008-10-01	40.789432	39.487402	42.025925	13.58	2008-07-02
3	2008-11-01	42.973994	41.641828	44.234486	12.26	2008-07-02
4	2008-12-01	40.762470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

## Evaluating the FbProphet model –

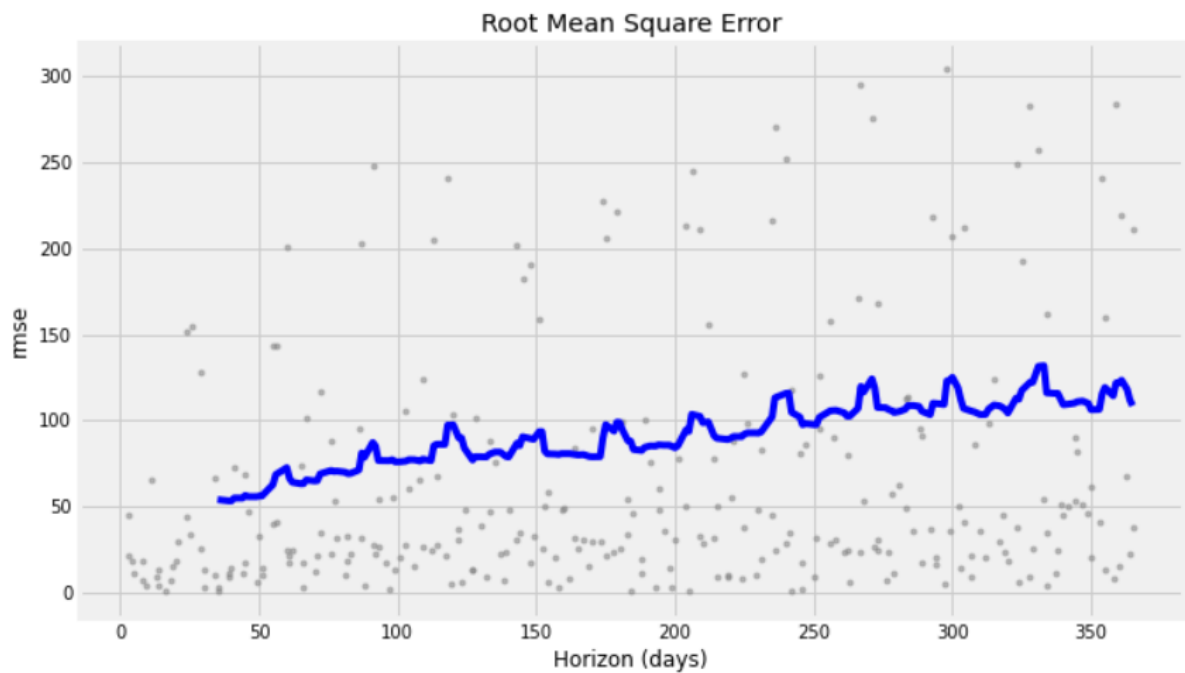
- Here, we use the **performance metrics** utility to compute the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and the coverage of the  $\hat{y}_{lower}$  and  $\hat{y}_{upper}$  estimates.

	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

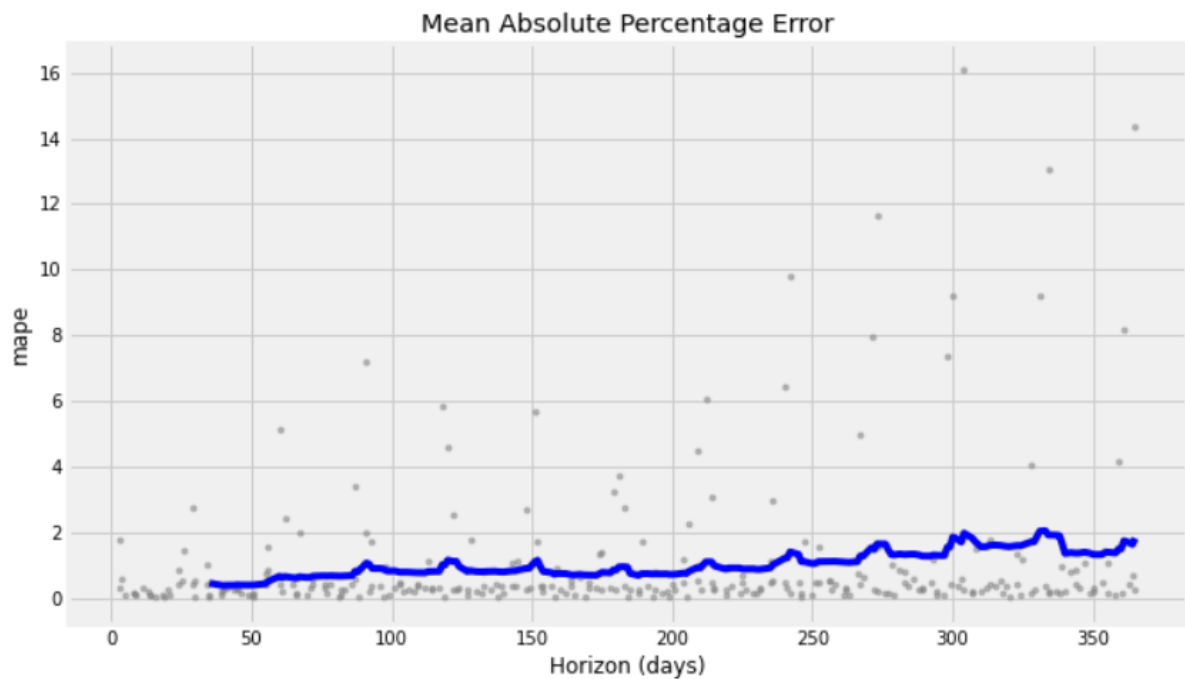
### a) MAE – Mean Absolute Error



## b) Root Mean Square Error



## c) Mean Absolute percentage Error



Here as shown for MAPE (Mean Absolute Percentage Error). Dots show the absolute percent error for each prediction in `crv_df`. 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 2.5% 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.

## **X. Challenges**

- Small dataset so it's difficult to get good model accuracy.
- Making the data stationary for time series models.
- Selecting the values of p, d, q for ARIMA.
- Adding a monthly trend to FbProphet.

## **XI. Conclusion**

- In EDA part we observed that
  - 1) There is an increase in the trend of Yes Bank's stock's Close price till 2018 then a sudden decrease.
  - 2) There is an increase in the trend of Yes Bank's stock's Open price till 2018 and then sudden decrease.
  - 3) We observed that open vs close price graph concluded that after 2018 yes bank's stock hit drastically.
- We implemented three Regressor
  - 1) KNN Regressor with highest accuracy - 92.00%
  - 2) linear regression accuracy - 82.00%
  - 3) SVM Regressor accuracy - 79.00%
- We implemented the ARIMA model and we got accuracy about 81.00%.
- We used a time series approach for the FbProphet forecast: the error of 2.5% 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.