STOCK PRICE PREDICTION

For Beginners

by Vaibhav Rai

PROBLEM

The stock market is very popular today. Many people invest their money with the expectation of receiving a good return. If you have expertise and knowledge of the stock market, it is best for you but, if you are an a beginner, the stock market is too risky for you, and you may lose your money.

AIM

So my aim is to create a system that helps beginners to invest in the stock market so that their losses are minimized. To create this system I used Time Series technique.

Objectives

01

Import Packages

First of all, I will import all the necessary libraries that we will use throughout the project. This generally includes libraries for data manipulation, data visualization, and others based on the specific needs of the project

02

Loading the Dataset

Next, I will load the dataset from yfinance into a pandas Data Frame which will facilitate easy manipulation and analysis:

03

Data Cleaning & Transformation

Clean the dataset by handling missing values, duplicates, and outliers, preparing it for effective Forecasting

04

Feature Engineering

To explore and implement effective feature engineering techniques that significantly improve the accuracy of our forecasting models, So we analyze the characteristics of the data (e.g., seasonality, trends, outliers).

05

Splitting the data into train and test

Next, i will splitting data into train and test sets is crucial for reliable and unbiased evaluation of model performance on unseen data

06

AutoTs Forecasting

In the final phase, I will forecast what the price of Tata Motors' shares will be in 30 days using Autots.

Loading the Dataset

In the first phase, I downloaded Tata Motors' stock price data from Yahoo Finance and created columns for forecasting purposes. The most crucial column in this data set is Close, which tells us what the stock price was at the end of every trading day.

```
[68] d1 = today.strftime("%Y-%m-%d") #"%Y-%m-%d" this is a format of year-month-day
                                     # to format this datetime object into a string

[69] end_date = d1

[70] d2 = date.today() - timedelta(days=730) #A timedelta object displays the amount of time—in days, seconds, and microseconds

[71] d2 = d2.strftime("%Y-%m-%d")

[72] start_date = d2

[73] data=yf.download('TATAMOTORS.NS', start=start_date, end=end_date, progress=False) #

[74] data['Date']= data.index

[75] data=data[["Date", "Open", "High", "Low", "Close", "Adj Close", "Volume"]]

[76] data.reset_index(drop=True, inplace=True)
```

```
data.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2022-02-23	481.750000	485.399994	475.299988	477.000000	475.517944	17703998
1	2022-02-24	455.950012	461.549988	405.450012	427.950012	426.620361	57265685
2	2022-02-25	444.850006	465.700012	441.600006	459.750000	458.321564	48876182
3	2022-02-28	445.000000	457.350006	440.450012	454.049988	452.639252	34475468
4	2022-03-02	452.950012	454.250000	444.549988	447.600006	446.209320	25791134

Primary Analysis

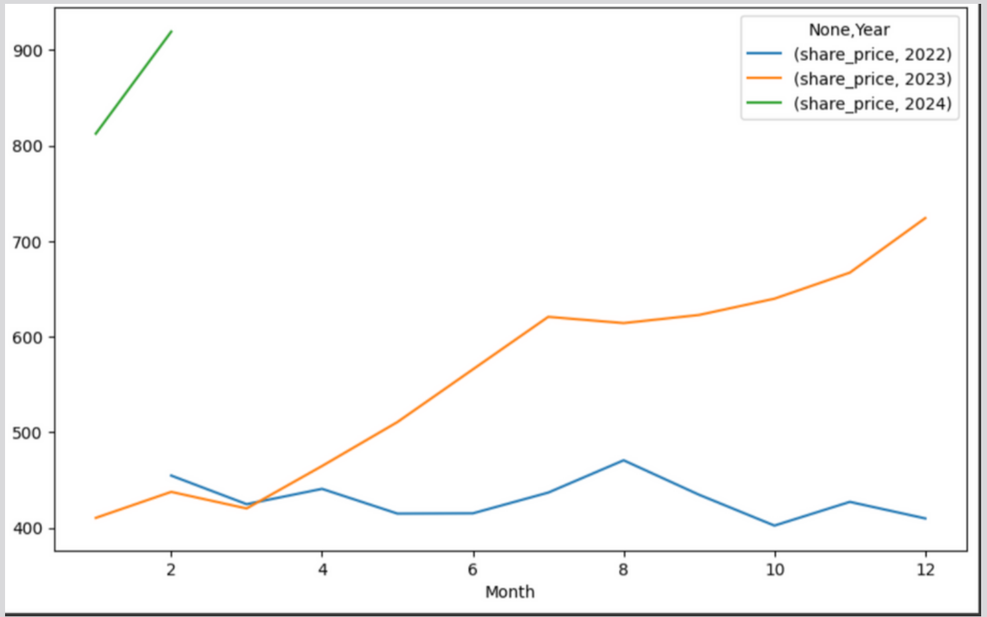
I separated the date column into three parts -
1.Year
2.Quarter and
3.Month
so that we could better understand the data for forecasting.

```
# Primary Analysis

data2['Year'] = data2['Date'].dt.year
data2['Qtr'] = data2['Date'].dt.quarter
data2['Month'] = data2['Date'].dt.month
data2.head()
```

	Date	share_price	Year	Qtr	Month
0	2022-02-23	477.000000	2022	1	2
1	2022-02-24	427.950012	2022	1	2
2	2022-02-25	459.750000	2022	1	2
3	2022-02-28	454.049988	2022	1	2
4	2022-03-02	447.600006	2022	1	3

In this graph we can clearly see the year by year growth.



Whenever we work on time series, our data should be stationary and it should be in a range and this is one of the properties of time series.

To check if data is stationary or not

In this step we check whether our data is stationary or not. So as we can see our p value is greater than alpha so the null hypothesis will be accepted. so our data is not stationary

```
# Augmented Dickey - Fuller Test

def adf_test(data):
    res = adfuller(data)
    print('ADF-Test_Stat',res[0])
    print('p-val',res[1])
    if res[1]> 0.05:
        print('Ho accepted - Data is non-stationary')
    else:
        print('H1 accepted - Data is stationary')
```

```
[95] adf_test(data)

ADF-Test_Stat 2.152766535532003
p-val 0.9988413373180293
Ho accepted - Data is non-stationary
```

To make the data stationary we use differencing method

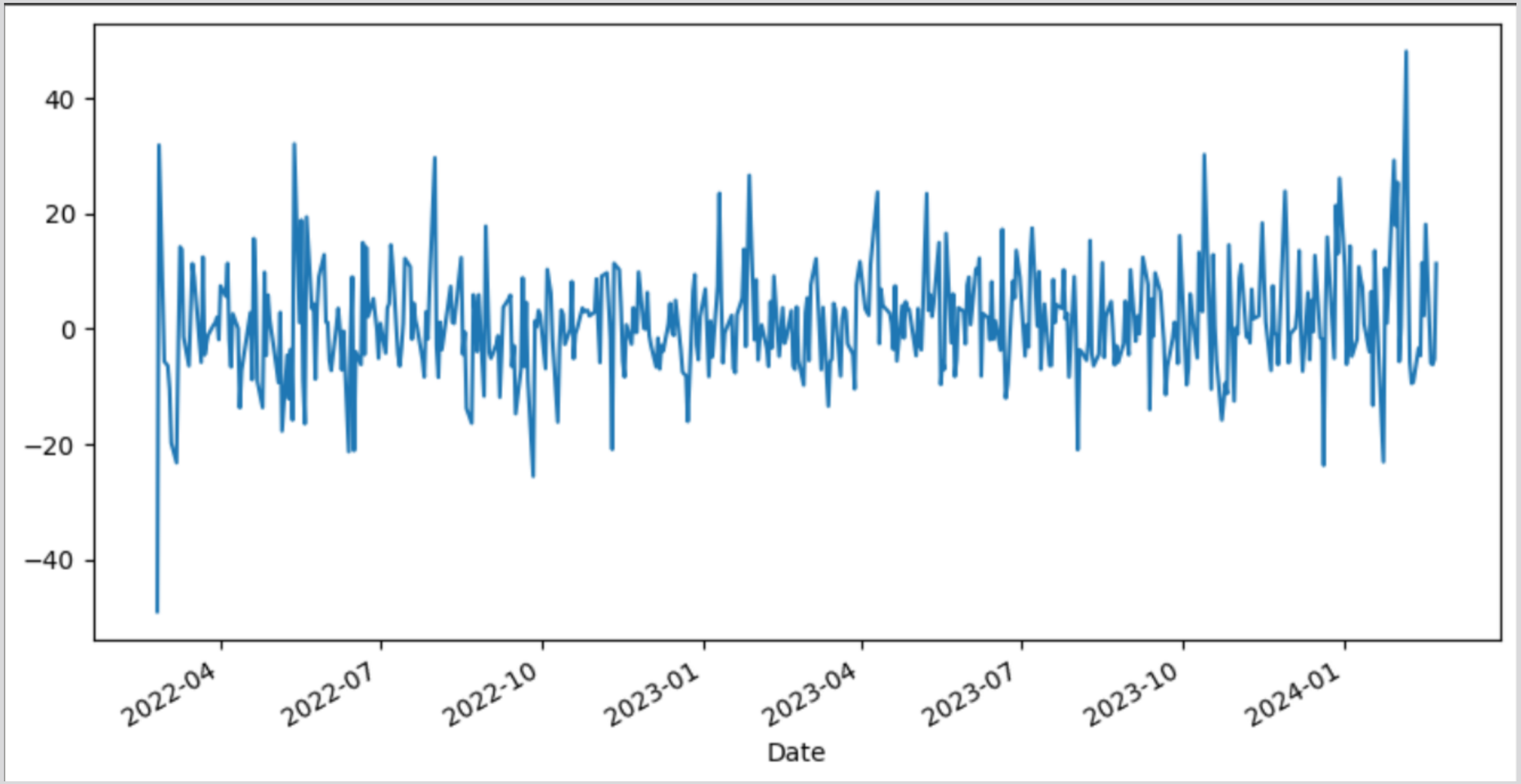
by the using of differing method our data has become stationary. So as we can see our p value is less than alpha then the null hypothesis will be rejected. so our data is stationary

```
[96] data['share_price_diff'] = data['share_price'].diff()
     data= data.dropna()
```

```
adf_test(data['share_price_diff'].dropna())

ADF-Test_Stat -22.259809559762978
p-val 0.0
H1 accepted - Data is stationary
```

Stationary data

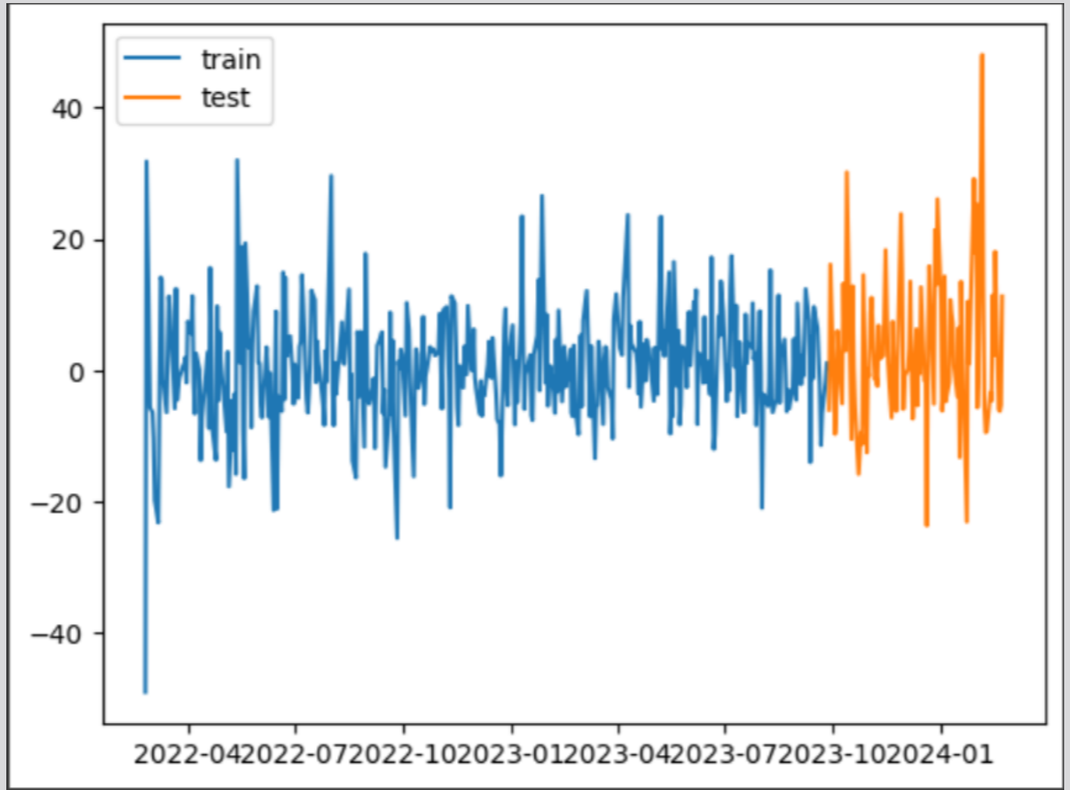


Splitting the data into train and test

By splitting data, we ensure our models are generalizable, unbiased, and equipped to handle new data, leading to more accurate and reliable predictions.

```
# Split the data into training and testing sets
train_data, test_data = data[[ "share_price_diff"]].iloc[:int(0.8 * len(data))], data[[ "share_price_diff"]].iloc[int(0.8 * len(data)):]
```

As we can see after splitting the data into train and test. The result is the same which means our data is a generalized data. it has low bias and low variance which is absolutely perfect for our forecasting.



Now I will run the model by the help of AutoTS

first of all what is AutoTS

AutoTS is a time series package for Python designed for rapidly deploying high-accuracy forecasts at scale which use Ensemble learning Method. In 2023, AutoTS has won in the M6 forecasting competition, delivering the highest performance investment decisions across 12 months of stock market forecasting.

AutoTS includes a wide range of statistical, machine learning, and deep learning models. suitable for time series forecasting. These include:

- Statistical models: ARIMA, SARIMA, Prophet, Exponential Smoothing, etc.
- Machine learning models: Random Forest, XGBoost, LightGBM, etc.
- Deep learning models: LSTMs, Transformers, etc.

AutoTS doesn't require to manually choose a model. Instead, it automatically evaluates a
large number of different models on your stock data, considering factors like:

- Data characteristics: Seasonality, trends, stationarity, etc.
- Model complexity: Balance between accuracy and interpretability.
- Computational cost: Training time and resource requirements.

Model Selection:
Based on its evaluation, AutoTS selects the model that achieves the best performance on a chosen metric, such as mean squared error (MSE) or mean absolute error (MAE).

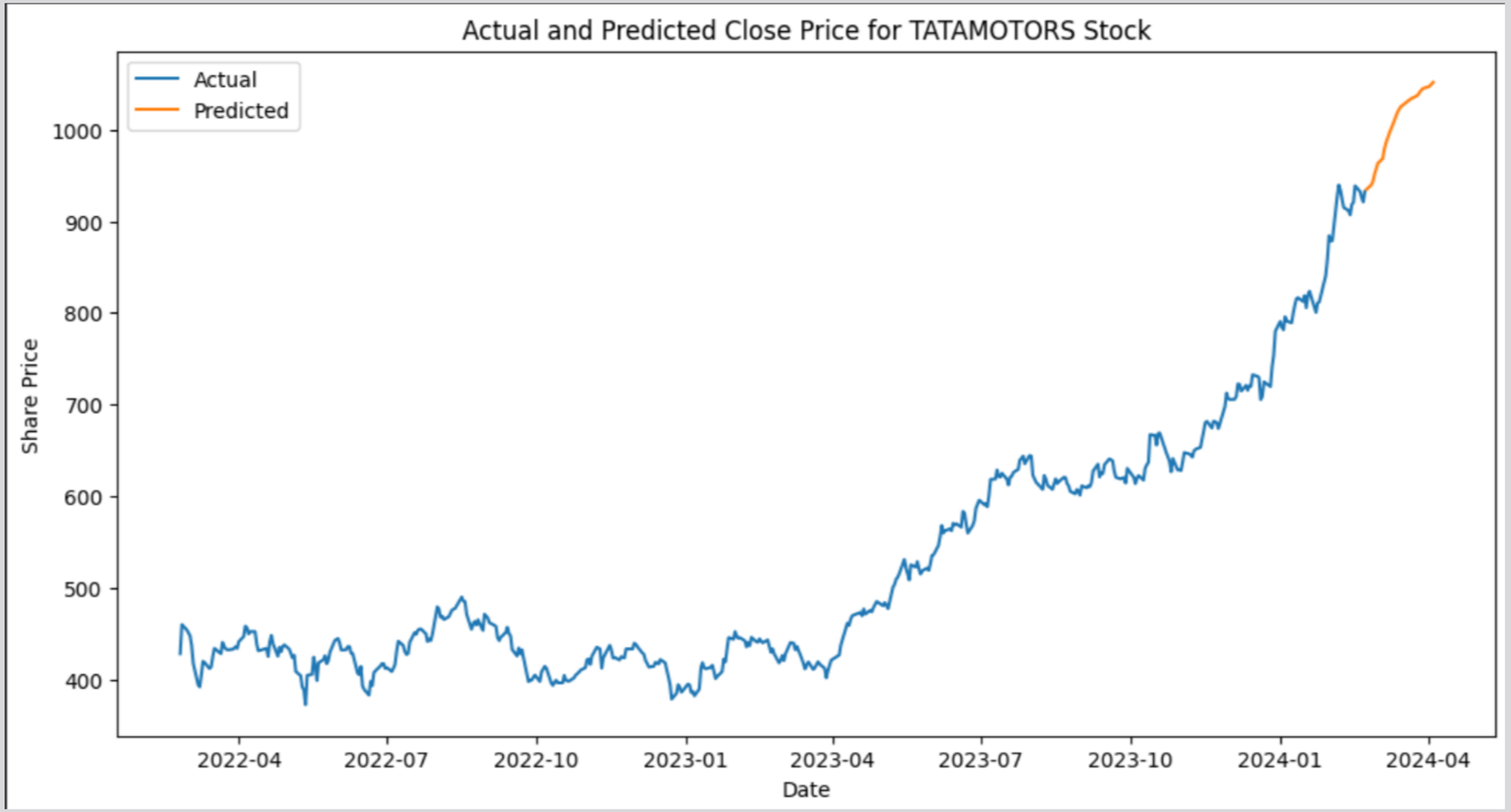
While AutoTS automates the process, we can still fine-tune the chosen model by adjusting hyperparameters or incorporating additional features into our data

```
[113] from autots import AutoTS
```

```
[ ] model = AutoTS(forecast_length=30, frequency='infer', ensemble='simple')
```

```
Using 1 cpus for n_jobs.
```

```
[115] model = model.fit(data1, date_col='Date', value_col='share_price', id_col=None)
      prediction = model.predict()
      forecast = prediction.forecast
      forecast
```



2024-02-23	934.668254
2024-02-26	939.629242
2024-02-27	943.557815
2024-02-28	951.418905
2024-02-29	957.302395
2024-03-01	963.205598
2024-03-04	968.434765
2024-03-05	978.710381
2024-03-06	985.635572
2024-03-07	990.513469
2024-03-08	995.838887
2024-03-11	1009.104846
2024-03-12	1014.041968
2024-03-13	1018.659031
2024-03-14	1021.990367
2024-03-15	1024.863810
2024-03-18	1029.080124
2024-03-19	1030.434700
2024-03-20	1031.836960
2024-03-21	1033.208076
2024-03-22	1034.313437
2024-03-25	1036.830710

as we can see we predict the next 30 days stock price of tatamotors by the help of Autots forecasting. But there is no guarantee that it will be 100% correct because the stock market depends on the situation and if there is any war or political instability or any disaster then the stock price will change.

Thank You


Euro Stoxx 50
Dax
6601,37
6600
6400
Global Titans Asia
Umsatz 29.10.2010