WiDS ‘22 - ‘23 Final Documentation

WIDS 28 - Stock Market Prediction using Time Series Forecasting

Mentors - Shivesh Gupta, Saumya Sheth

Team Member Name	Roll Number	Email-Id
Prashant Ranjan	210040117	210040117@iitb.ac.in

Introduction to Problem Statement

To predict stock prices of TATAMOTORS by applying Time series forecasting using the Auto Regressive Integrated Moving Average (ARIMA) Model.

Existing Resources

https://github.com/shiveshcodes/Time-Series-Forecasting-Resources

This resource contains all the concepts related to time series forecasting and also some tutorials on how to use time series analysis

Proposed Solution

Reading our Data

The first step in any time series is to read our data and see how it looks like. The following code snippet demonstrates how to do that.

```
In [3]:
import pandas as pd
import numpy as np
```

```
In [4]:
df=pd.read_csv('TATAMOTORS.csv',index_col='DATE'    ,parse_dates=True)
df=df.dropna()
print('Shape of data',df.shape)
df.head()
```

The code is rather simple to understand. In order to ensure that pandas recognises that it is working with date values rather than string values, we read the data using pd.read csv and parse date=True.

We then remove any missing numbers and print the data's form. The dataset's first five rows are displayed by df.head(). The result for this should look like this:

In [7]:
```
import pandas as pd
df=pd.read_csv('TATAMOTORS.csv',index_col='Date'    ,parse_dates=True)
df=df.dropna()
print('Shape of data',df.shape)
df.head()
```

```
Shape of data (2456, 14)
```

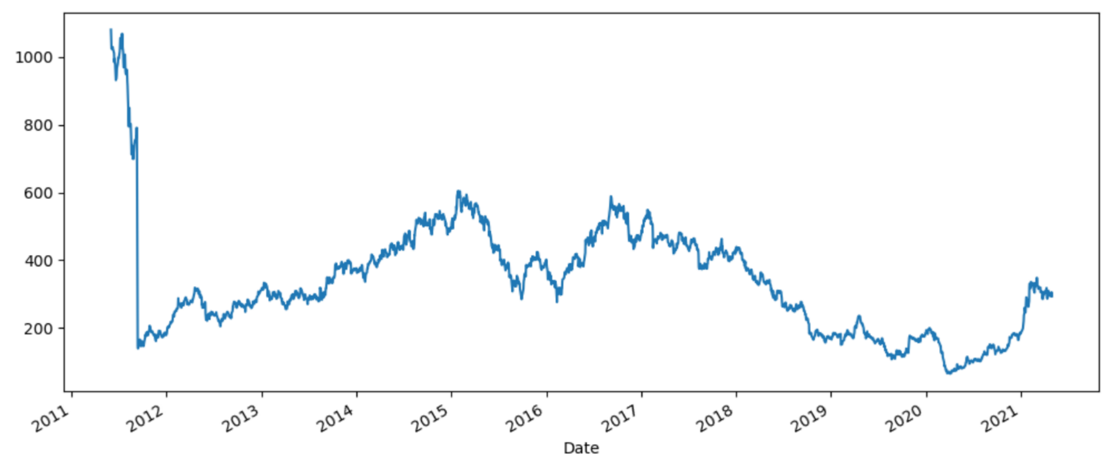
Out[7]:

Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverble
2011-06-01	TATAMOTORS	EQ	1095.3	1096.0	1097.0	1075.60	1077.25	1079.9	1083.36	2412309	2.613407e+14	76698.0	1152083.0	0.4776
2011-06-02	TATAMOTORS	EQ	1079.9	1070.0	1070.0	1042.20	1046.15	1048.2	1050.62	2604237	2.736056e+14	74650.0	1149341.0	0.4413
2011-06-03	TATAMOTORS	EQ	1048.2	1048.0	1058.0	1014.25	1024.60	1023.8	1031.52	3600090	3.713581e+14	94611.0	1472425.0	0.4090
2011-06-06	TATAMOTORS	EQ	1023.8	1011.5	1031.8	1010.25	1022.20	1023.7	1020.38	2076683	2.119014e+14	64914.0	451682.0	0.2175
2011-06-07	TATAMOTORS	EQ	1023.7	1017.0	1033.8	1015.65	1028.00	1027.2	1027.38	2117760	2.175745e+14	56772.0	805423.0	0.3803

Next we will plot our data

In [8]:
```
df['Close'].plot(figsize=(12,5))
```

Out[8]: <AxesSubplot:xlabel='Date'>



Next we will check whether our data is stationary or not

In [13]:

```python
def ad_test(dataset):
     dftest = adfuller(dataset, autolag = 'AIC')
     print("1. ADF : ",dftest[0])
     print("2. P-Value : ", dftest[1])
     print("3. Num Of Lags : ", dftest[2])
     print("4. Num Of Observations Used For ADF Regression:",      dftest[3])
     print("5. Critical Values :")
     for key, val in dftest[4].items():
         print("\t",key, ": ", val)
ad_test(df['Close'])
```

```
1. ADF :  -4.855592403236513
2. P-Value :  4.2522405224656046e-05
3. Num Of Lags :  25
4. Num Of Observations Used For ADF Regression: 2430
5. Critical Values :
	 1% :  -3.4330439182185093
	 5% :  -2.862730143690387
	 10% :  -2.5674035621263696
```

If $p < 0.05$; Data is stationary

if $p > 0.05$; Data is not stationary

In [15]:

```python
from pmdarima import auto_arima
import warnings
warnings.filterwarnings("ignore")
stepwise_fit = auto_arima(df['Close'], trace=True,
suppress_warnings=True)
stepwise_fit.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=20290.397, Time=1.40 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=20295.083, Time=0.14 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=20290.194, Time=0.58 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=20289.574, Time=0.89 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=20294.189, Time=0.10 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=20286.875, Time=2.09 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=20288.668, Time=3.69 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=20288.530, Time=3.75 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=20287.542, Time=1.16 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=20287.572, Time=1.01 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=20285.927, Time=0.75 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=20288.571, Time=0.37 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=20289.194, Time=0.23 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=20287.823, Time=0.77 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=20287.689, Time=1.59 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=20286.623, Time=0.25 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=20286.653, Time=0.52 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=20289.539, Time=1.40 sec

Best model:  ARIMA(1,1,1)(0,0,0)[0]
Total fit time: 20.717 seconds
```

Out[15]: SARIMAX Results

Dep. Variable:	y	**No. Observations:**	2456
Model:	SARIMAX(1, 1, 1)	**Log Likelihood**	-10139.963
Date:	Wed, 18 Jan 2023	**AIC**	20285.927
Time:	19:28:50	**BIC**	20303.345
Sample:	0	**HQIC**	20292.256
	- 2456		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.7157	0.172	-4.158	0.000	-1.053	-0.378
ma.L1	0.7651	0.165	4.642	0.000	0.442	1.088
sigma2	226.5090	0.655	346.048	0.000	225.226	227.792

Ljung-Box (L1) (Q):	0.11	**Jarque-Bera (JB):**	132921009.31
Prob(Q):	0.73	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.07	**Skew:**	-27.94
Prob(H) (two-sided):	0.00	**Kurtosis:**	1141.56

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

There is a lot of info about your model here that we can see. You will also be able to see each AR and MA term's coefficients. These are nothing more than the values of the variables labelled as "Some Constant" in the previous AR/MA model equation.

Generally speaking, a variable's magnitude indicates how much of an impact it has on the result.

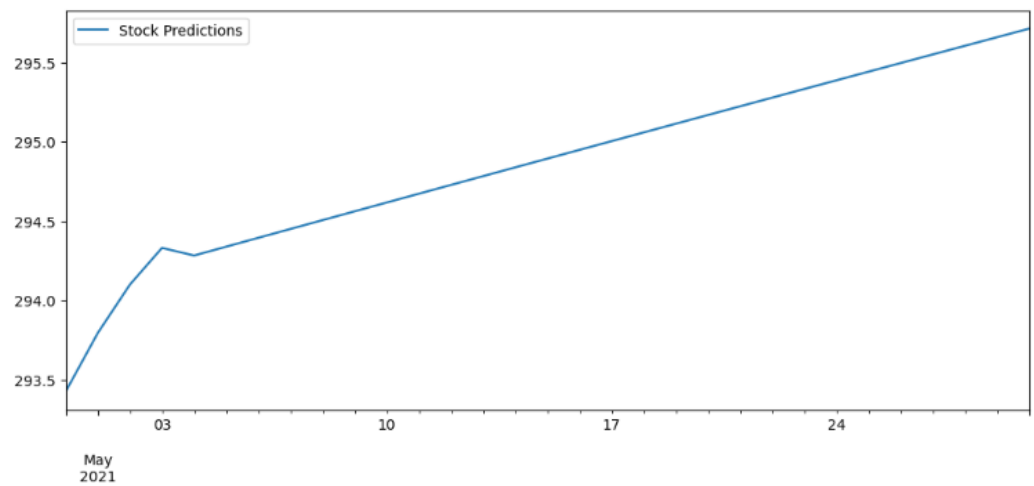
Finally results of our prediction

In [49]:
```
index_future_dates=pd.date_range(start="2021-04-30", end="2021-05-30")
pred=model2.predict(start=len(df),end=len(df)+30,typ="levels").rename("Stock Predictions")
pred.index=index_future_dates
print(pred)
```

```
2021-04-30    293.425074
2021-05-01    293.794394
2021-05-02    294.099516
2021-05-03    294.331086
2021-05-04    294.282883
2021-05-05    294.338774
2021-05-06    294.394593
2021-05-07    294.450342
2021-05-08    294.506019
2021-05-09    294.561624
2021-05-10    294.617159
2021-05-11    294.672623
2021-05-12    294.728016
2021-05-13    294.783338
2021-05-14    294.838590
2021-05-15    294.893771
2021-05-16    294.948882
2021-05-17    295.003922
2021-05-18    295.058892
2021-05-19    295.113792
2021-05-20    295.168621
2021-05-21    295.223381
2021-05-22    295.278070
2021-05-23    295.332690
2021-05-24    295.387240
2021-05-25    295.441721
2021-05-26    295.496131
2021-05-27    295.550473
2021-05-28    295.604745
2021-05-29    295.658947
2021-05-30    295.713081
Freq: D, Name: Stock Predictions, dtype: float64
```

In [51]:
```
pred.plot(figsize=(12,5),legend=True)
```

Out[51]:
```
<AxesSubplot:>
```



Methodology & Progress (Mention the work done week-wise)

Setup of Jupyter Notebook and introduction to Python essentials in Week 1.

Week 2: Understanding the theoretical underpinnings of Time Series Data and Machine Learning in general.Learned about supervised and unsupervised learning

Week 3: Starting to develop models and become familiar with common AI/ML libraries. Learnt about time series forecasting and its different models.

Week 4: Finish implementing the model and become familiar with acceptable coding procedures. Applied all the things learned to predict the stocks.

Week 5: Project debugging and completion, as well as discussions on further learnings.

Results

https://github.com/prashantr00082/WIDS-Project--Stock-market-prediction-using-time-series-forecasting

Learning Value

Learned machine learning concepts. Learned about the application of the Python programming language and the principles of machine learning. Learned about different models of Time series forecasting.

Learned about how to read stock market data. Opening and closing prices etc.

Learned various models of time seriesforecasting, which are moving average and exponential smoothing. Methods for measuring timed data are referred to as times series. Autoregression (AR), Moving Average (MA), Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving Average (ARIMA), and Seasonal Autoregressive Integrated Moving-Average are examples of common kinds (SARIMA).

The ARIMA model is a well-liked and commonly applied statistical technique for time series forecasting. It is one of the most widely used models for predicting data from linear time series. Since this model is well-known to be reliable, effective, and has a significant potential for short-term share market prediction, it has been employed extensively in the fields of finance and economics. The two most popular methods for predicting time series are exponential smoothing and ARIMA models, both of which offer complementary approaches to the issue. While ARIMA models seek to characterise the auto-correlation (Autocorrelation is the degree of resemblance between a given time series and a lagged version of itself over subsequent time periods), exponential smoothing models are based on a description of the trend and seasonality in the data.

Tech-stack Used

Programming Language used - Python

Jupyter Notebook

Anaconda

Numpy

Pandas

Matplotlib

Sckitlearn

statsmodel.api

Suggestions for others

Before jumping into the coding part of this project. Firstly try to understand the concepts behind all the models.

Understand the following method for time series forecasting carefully:

Smoothing-based models

Data smoothing is a statistical approach used in time series forecasting that entails reducing outliers from a time series data collection to enhance the visibility of a trend. Some kind of random variation is present in every collection of data gathered over time. Data smoothing reveals underlying trends and cyclical components while removing or reducing random variance.

Moving-average model

In time series analysis, the moving-average model (MA model), also known as moving-average process, is a common approach for modelling univariate time series. The moving-average model specifies that the output variable depends linearly on the current and various past values of a stochastic (imperfectly predictable) term.

Together with the autoregressive (AR) model (covered below), the moving-average model is a special case and key component of the more general ARMA and ARIMA models of time series, which have a more complicated stochastic structure.

Contrary to the AR model, the finite MA model is always stationary.

Exponential Smoothing model

Exponential smoothing is a rule of thumb technique for smoothing time series data using the exponential window function. Exponential smoothing is an easily learned and easily applied procedure for making some determination based on prior assumptions by the user, such as seasonality. Different types of exponential smoothing include single exponential smoothing, double exponential smoothing, and triple exponential smoothing (also known as the Holt-Winters method).

Contribution by each Team Member

Its was a individual project. Repositories of other people can be found in the following sheet.

https://docs.google.com/spreadsheets/d/1jtrquiuvpFIchgc6fiTetB1z2CwwHIgNHxo7xNCx6QU/edit#gid=0

References and Citations

https://youtube.com/playlist?list=PLqYFiz7NM_SMC4ZgXplbreXlRY4Jf4zBP

https://youtu.be/Y2khrpVo6qI

https://towardsdatascience.com/time-series-forecasting-predicting-stock-prices-using-an-arima-model-2e3b3080bd70

https://sites.google.com/site/econometricsacademy/econometrics-models/time-series-arima-models?pli=1

https://www.analyticsvidhya.com/blog/2020/11/stock-market-price-trend-prediction-using-time-series-forecasting/

https://www.influxdata.com/time-series-forecasting-methods/