Shailesh Anil Mishra
22M0223

WiDS '22 - '23 Final Documentation

UID: 28 Stock Market Prediction using Time Series Forecasting

Mentors: Shivesh Gupta and Saumya Seth

Team Member Name	**Roll Number**	**Email-Id**
Shailesh Anil Mishra	22M0223	22M0223@iitb.ac.in

❖ Introduction to Problem Statement

Using traditional statistical learning approaches for stock price forecasting by analyzing HCL stock Data using the ARIMA model. We aim to assist HCL with decisions relating to stock price forecasting using time series analysis.

❖ Existing Resources:

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

It has a Step-by-Step Process of Time Series Forecasting Using Machine Learning. With adequate theoretical knowledge for beginners.

❖ Proposed Solution:

The CRISP-DM methodology (Devi, B.U., Sundar, D. and Alli, P., 2013) is a 6 steps (Business understanding, Data understanding, and Data Preparation, Modeling, Evaluation, and Deployment) process for identifying, selecting, and assessing conditional mean models for discrete, Univariate time series data. For this dissertation proposed steps are:
Step 1: Business Data Understanding

- Business objectives
- Business Importance
- Sample data collection

- Describe and explore the data

Step 2: Data Preparation

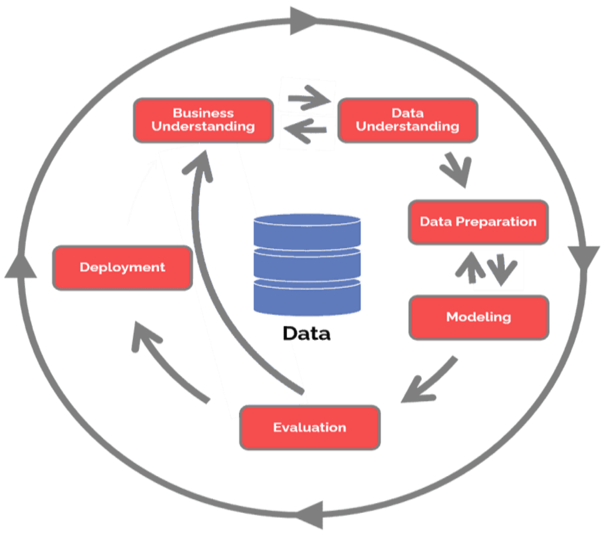
- Transform the data to stabilize the attributes
- Integrate and format data

Step 3: Data Modeling

- Examine the data to identify potential models
- Estimation and testing
- Predict and forecast

Step 4: Data Evaluation

- Evaluate results



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

The method used in this study to develop the ARIMA model for stock price forecasting is explained in detail in the subsections below. Stock data used in this research work are historical daily stock prices obtained from - Kaggle. In this research, the closing price is chosen to represent the price of the index to be predicted. The closing price is chosen because it reflects all the activities of the index on a trading day. To determine the best ARIMA model among several experiments performed, the following criteria are used in this study for the stock index.

Week 1&2:- Setting up Jupyter Notebook and learning the basics of Python Language also getting acquainted with python libraries like-

1. **NumPy** is a popular Python library for multi-dimensional array and matrix processing because it can be used to perform a great variety of **mathematical operations**.
2. **Pandas** is another Python library that is built on top of NumPy, responsible for preparing high-level **data sets** for machine learning and training.
3. **Matplotlib** is a Python library focused on **data visualization** and primarily used for creating beautiful graphs, plots, histograms, and bar charts.
4. **Statsmodels** is a Python library built specifically for statistics. Statsmodels is built on top of NumPy, SciPy, and matplotlib, but it contains more advanced functions for **statistical testing** and modeling that you won't find in numerical libraries like NumPy or SciPy.
5. **PMDArima** is an open-source Python library that is used for **time series forecasting and also helps in creating time series plots**. It is easy to use and generates a time-series forecast on the ARIMA model.

Week 3:- Learning the theoretical basis of Machine Learning in general and Time Series Data in particular.

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn.
Types of machine learning problems:

- **Supervised learning:** The model or algorithm is presented with example inputs and their desired outputs and then finds patterns and connections between the input and the output.
- **Unsupervised learning:** Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses.

Different Machine Learning Models:

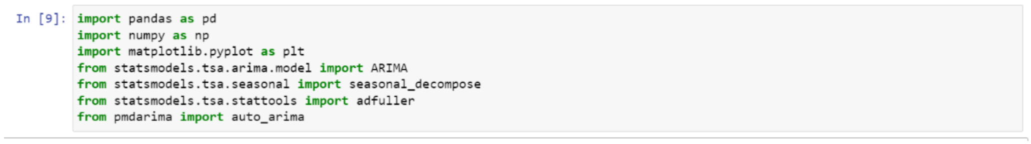
- **Classification**: As the name suggests, Classification is the task of “classifying things” into sub-categories.
- **Regression**: A regression problem is when the output variable is a real or continuous value, such as “salary” or “weight”.

Time Series Forecasting
In Machine Learning, Time Series Forecasting refers to the use of statistical models to predict future values using previously recorded observations. The underlying intention of time series forecasting is determining how to target variables will change in the future by observing historical data from the time perspective, defining the patterns, and yielding short or long-term predictions on how change occurs – considering the captured patterns.

- **ARIMA:** Autoregression employs observations collected during previous time steps as input data for making regression equations that help predict the value to be generated at the next time step. ARIMA or an AutoRegressive Integrated Moving Average model combines autoregression and moving average principles, making forecasts correspond to linear combinations of past variable values and forecast errors, being one of the most popular approaches due to that.

Week 4&5:- Complete the model implementation and learn about good coding practices.

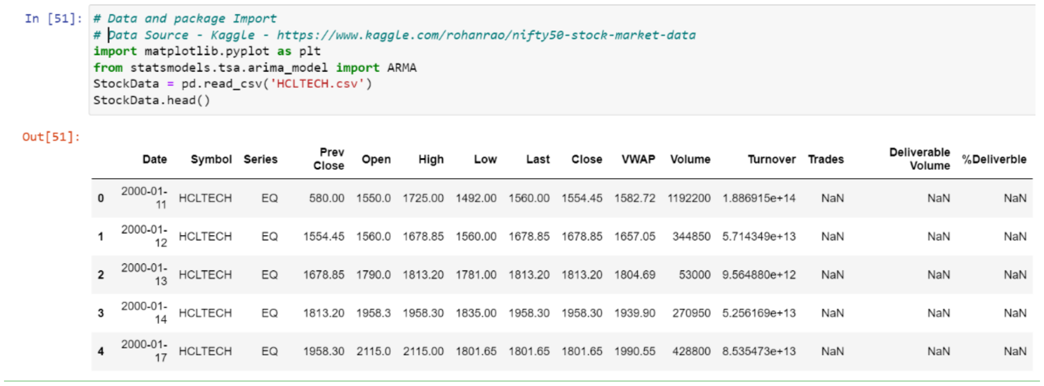
1. Importing Libraries



```
In [9]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from statsmodels.tsa.arima.model import ARIMA
        from statsmodels.tsa.seasonal import seasonal_decompose
        from statsmodels.tsa.stattools import adfuller
        from pmdarima import auto_arima
```

2. Read the Data

Firstly, collecting data on Kaggle to access HCL stock data.

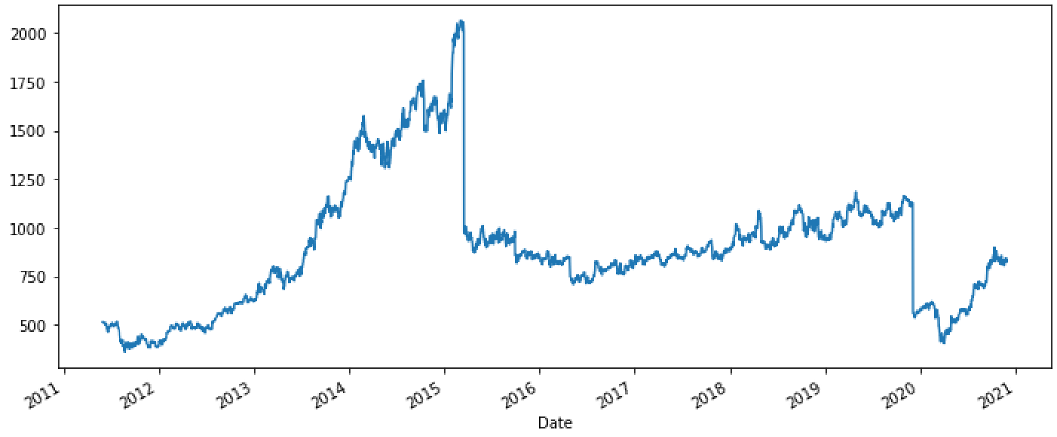


```
In [51]: # Data and package Import
         # Data Source - Kaggle - https://www.kaggle.com/rohanrao/nifty50-stock-market-data
         import matplotlib.pyplot as plt
         from statsmodels.tsa.arima_model import ARMA
         StockData = pd.read_csv('HCLTECH.csv')
         StockData.head()
```

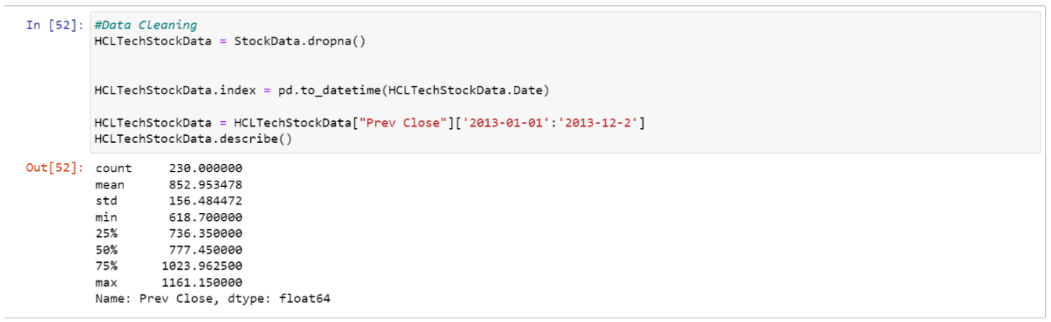
Out[51]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverble
0	2000-01-11	HCLTECH	EQ	580.00	1550.0	1725.00	1492.00	1560.00	1554.45	1582.72	1192200	1.886915e+14	NaN	NaN	NaN
1	2000-01-12	HCLTECH	EQ	1554.45	1560.0	1678.85	1560.00	1678.85	1678.85	1657.05	344850	5.714349e+13	NaN	NaN	NaN
2	2000-01-13	HCLTECH	EQ	1678.85	1790.0	1813.20	1781.00	1813.20	1813.20	1804.69	53000	9.564880e+12	NaN	NaN	NaN
3	2000-01-14	HCLTECH	EQ	1813.20	1958.3	1958.30	1835.00	1958.30	1958.30	1939.90	270950	5.256169e+13	NaN	NaN	NaN
4	2000-01-17	HCLTECH	EQ	1958.30	2115.0	2115.00	1801.65	1801.65	1801.65	1990.55	428800	8.535473e+13	NaN	NaN	NaN

3. Data Visualization

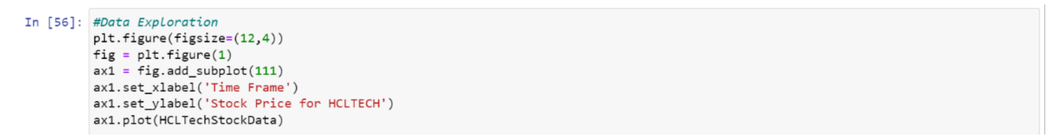


4. Data Cleaning

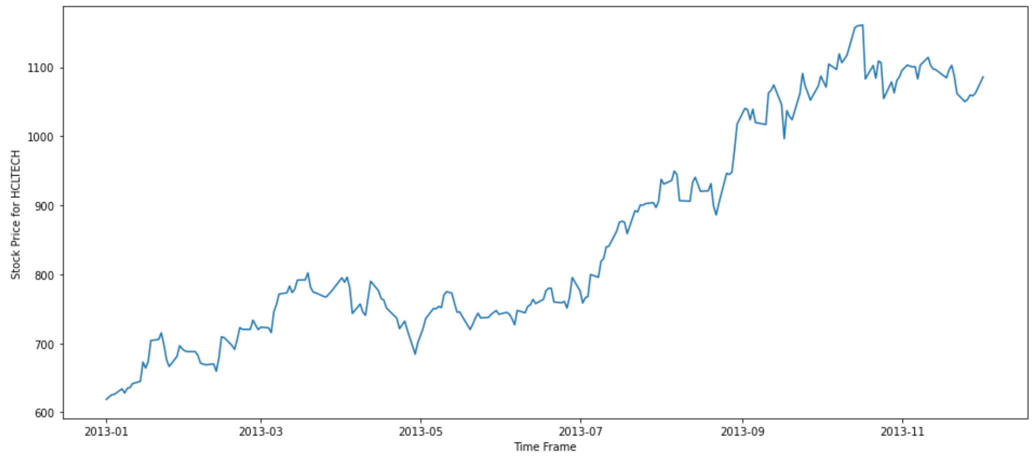


```
In [52]: #Data Cleaning
         HCLTechStockData = StockData.dropna()


         HCLTechStockData.index = pd.to_datetime(HCLTechStockData.Date)

         HCLTechStockData = HCLTechStockData["Prev Close"]['2013-01-01':'2013-12-2']
         HCLTechStockData.describe()

Out[52]: count     230.000000
         mean      852.953478
         std       156.484472
         min       618.700000
         25%       736.350000
         50%       777.450000
         75%      1023.962500
         max      1161.150000
         Name: Prev Close, dtype: float64
```

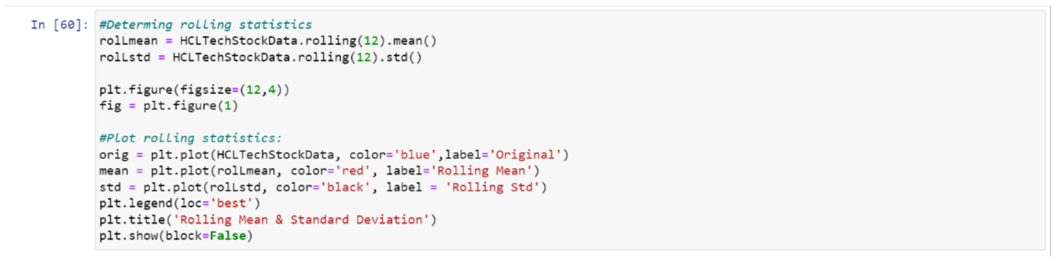
5. Data Exploration



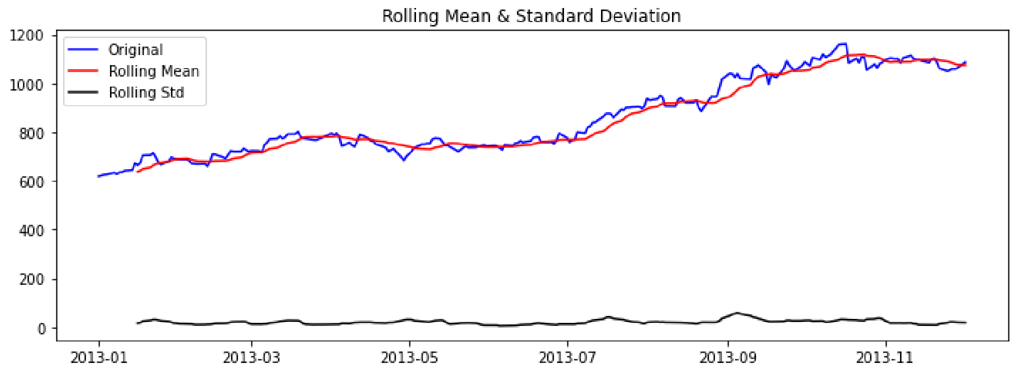
```
In [56]: #Data Exploration
         plt.figure(figsize=(12,4))
         fig = plt.figure(1)
         ax1 = fig.add_subplot(111)
         ax1.set_xlabel('Time Frame')
         ax1.set_ylabel('Stock Price for HCLTECH')
         ax1.plot(HCLTechStockData)
```



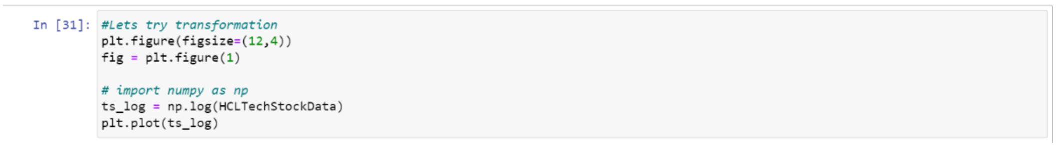
6. Determining the Rolling Statistics



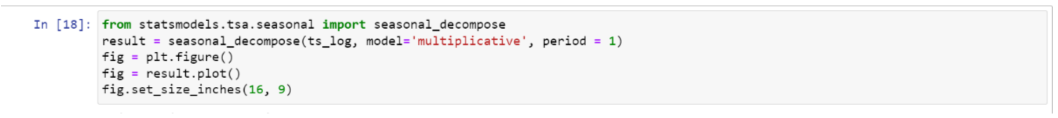
```
In [60]: #Determing rolling statistics
         rolLmean = HCLTechStockData.rolling(12).mean()
         rolLstd = HCLTechStockData.rolling(12).std()

         plt.figure(figsize=(12,4))
         fig = plt.figure(1)

         #Plot rolling statistics:
         orig = plt.plot(HCLTechStockData, color='blue',label='Original')
         mean = plt.plot(rolLmean, color='red', label='Rolling Mean')
         std = plt.plot(rolLstd, color='black', label = 'Rolling Std')
         plt.legend(loc='best')
         plt.title('Rolling Mean & Standard Deviation')
         plt.show(block=False)
```



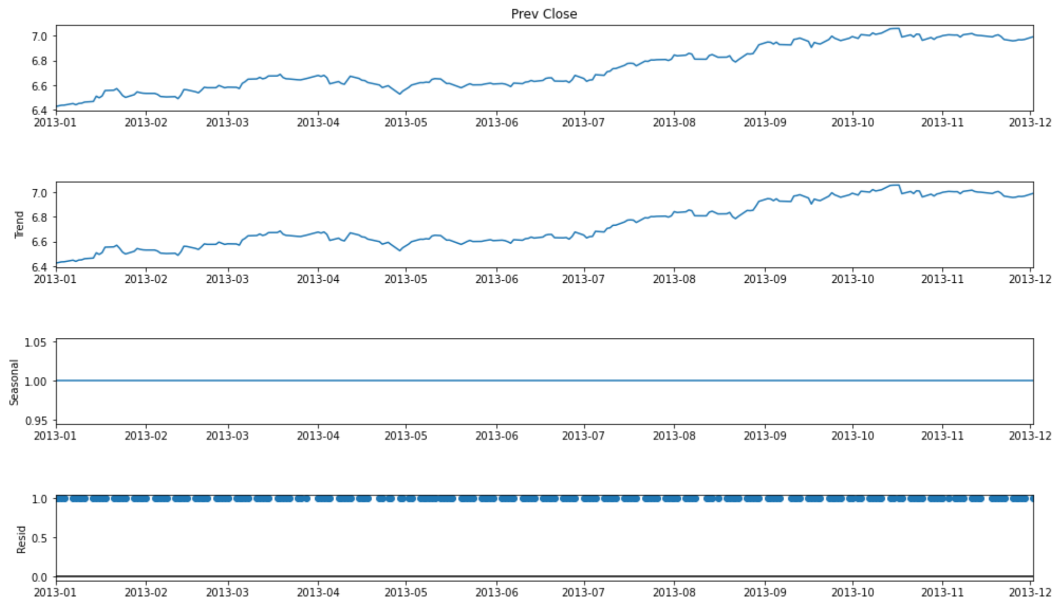
7. Data Transformation



```
In [31]: #Lets try transformation
         plt.figure(figsize=(12,4))
         fig = plt.figure(1)

         # import numpy as np
         ts_log = np.log(HCLTechStockData)
         plt.plot(ts_log)
```



```
In [18]: from statsmodels.tsa.seasonal import seasonal_decompose
         result = seasonal_decompose(ts_log, model='multiplicative', period = 1)
         fig = plt.figure()
         fig = result.plot()
         fig.set_size_inches(16, 9)
```



8. Running Dickey-Fuller Test

Before building the model, it’s essential to know if the series is stationary or not because time series analysis only works with stationary data. When applying ARIMA mode, it’s critical to find the order of differencing (d) since the purpose of difference is to make the time series stationary.

We will only need differencing if the series is non-stationary, otherwise, the d is 0 (zero) in ARIMA. In this article, we will use ***ADF (Augmented Dickey-Fuller)*** Test to check if a series is stationary or not.

The ADF test is one of the most popular statistical tests to determine the presence of unit roots in the series and help to understand if the series is stationary or not. The null and alternate hypothesis of this test is:

Null Hypothesis: The series has a unit root (value of a =1)

The null hypothesis of the test is that the time series can be represented by a unit root.

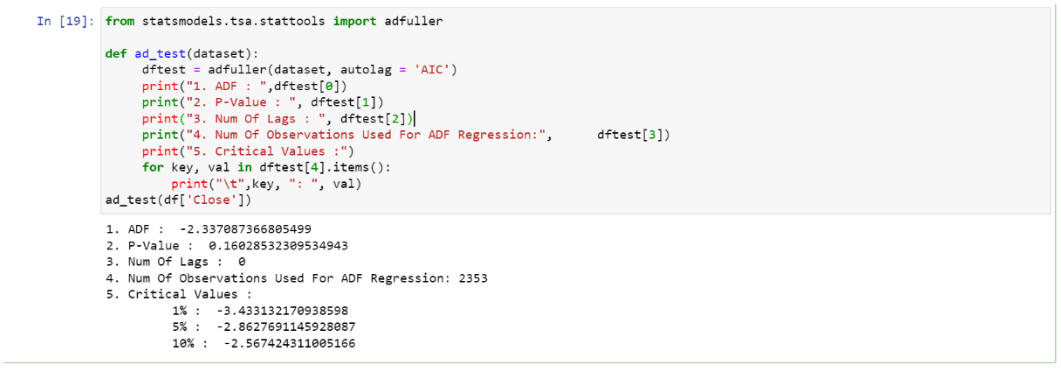
If failed to be rejected, it suggests the time series has a unit root, meaning it is non-stationary. It has some time-dependent structure.

Alternate Hypothesis: The series has no unit root.

The alternate hypothesis (rejecting the null hypothesis) is that the time series is stationary.

The null hypothesis is rejected; it suggests the time series does not have a unit root, meaning it is stationary. It does not have a time-dependent structure.

If the p-value < 0.05 (significance level), we reject the null hypothesis and infer the series is stationary.



```
In [19]: from statsmodels.tsa.stattools import adfuller

         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 :  -2.337087366805499
2. P-Value :  0.16028532309534943
3. Num Of Lags :  0
4. Num Of Observations Used For ADF Regression: 2353
5. Critical Values :
	 1% :  -3.433132170938598
	 5% :  -2.8627691145928087
	 10% :  -2.567424311005166
```

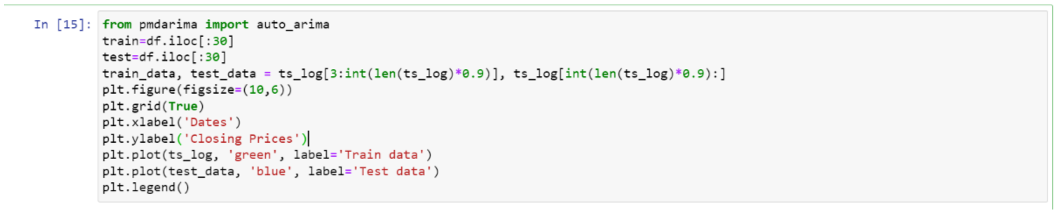
In this case, the p-value > 0.05, so we fail to reject the null hypothesis and we can say that the series is non-stationary. This means that the series can be linear or difference stationary.

We then go ahead with finding the order of differencing. Instead of choosing the value of p, d, and q by observing the plots of ACF and PACF, we use Auto ARIMA to get the best parameters.

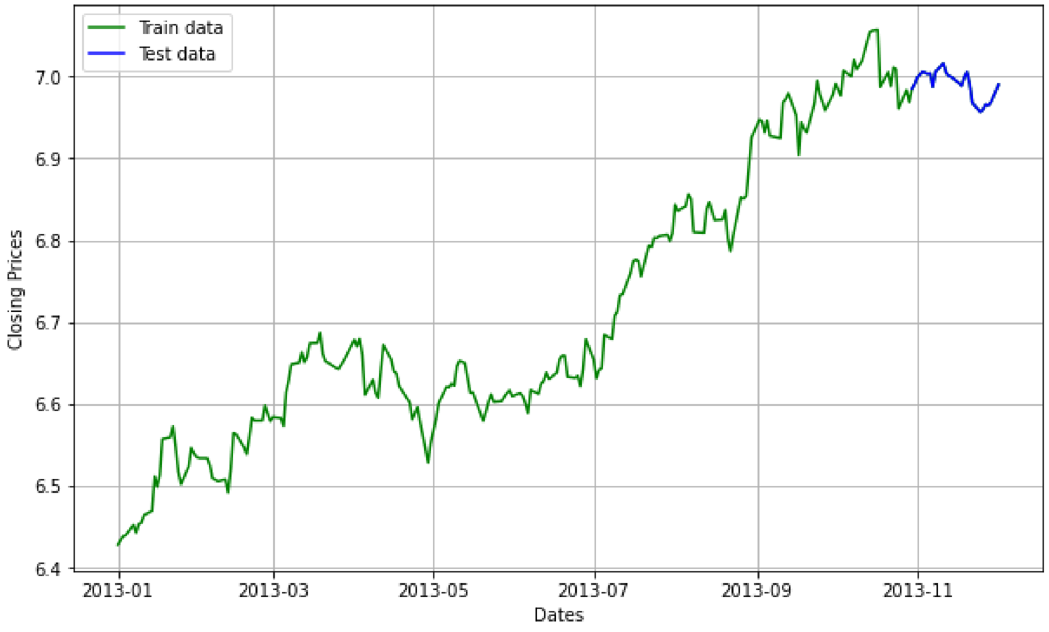
Auto ARIMA, which Automatically discovers the optimal order for an ARIMA model, can identify the optimal parameters for an ARIMA model based on the R function.

9. Training and Testing

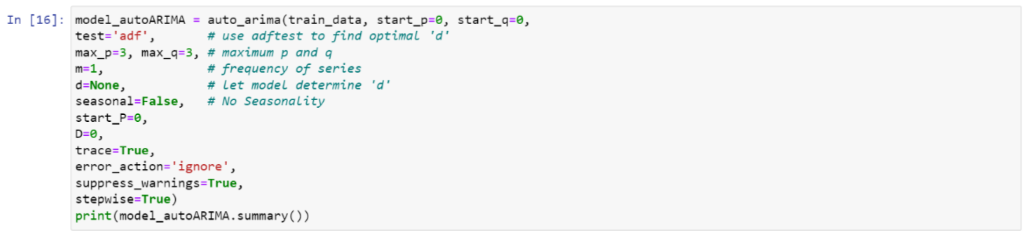
Now we are going to create an ARIMA model and will train it with the closing price of the stock on the train data. So let us split the data into training and test set and visualize it.



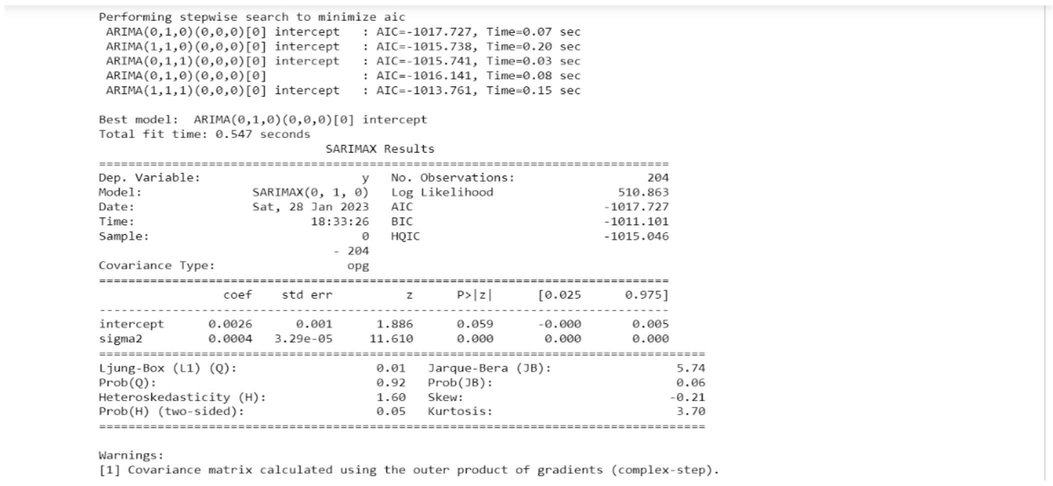
```
In [15]: from pmdarima import auto_arima
         train=df.iloc[:30]
         test=df.iloc[:30]
         train_data, test_data = ts_log[3:int(len(ts_log)*0.9)], ts_log[int(len(ts_log)*0.9):]
         plt.figure(figsize=(10,6))
         plt.grid(True)
         plt.xlabel('Dates')
         plt.ylabel('Closing Prices')
         plt.plot(ts_log, 'green', label='Train data')
         plt.plot(test_data, 'blue', label='Test data')
         plt.legend()
```



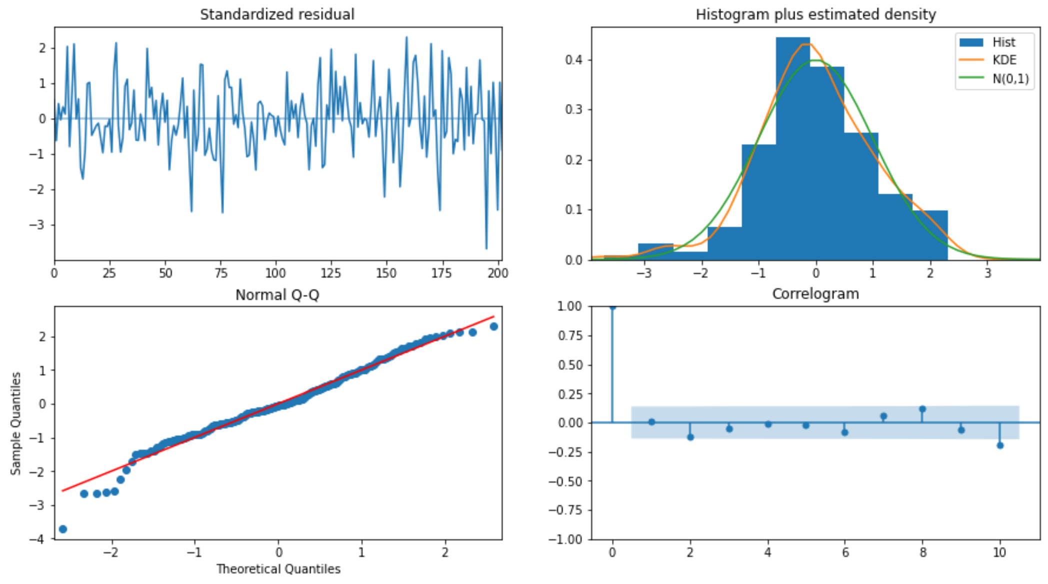
10. SARIMAX Result



```
In [16]: model_autoARIMA = auto_arima(train_data, start_p=0, start_q=0,
         test='adf',       # use adftest to find optimal 'd'
         max_p=3, max_q=3, # maximum p and q
         m=1,              # frequency of series
         d=None,           # let model determine 'd'
         seasonal=False,   # No Seasonality
         start_P=0,
         D=0,
         trace=True,
         error_action='ignore',
         suppress_warnings=True,
         stepwise=True)
         print(model_autoARIMA.summary())
```



```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-1017.727, Time=0.07 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-1015.738, Time=0.20 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-1015.741, Time=0.03 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-1016.141, Time=0.08 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-1013.761, Time=0.15 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0] intercept
Total fit time: 0.547 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  204
Model:               SARIMAX(0, 1, 0)   Log Likelihood                 510.863
Date:                Sat, 28 Jan 2023   AIC                          -1017.727
Time:                        18:33:26   BIC                          -1011.101
Sample:                             0   HQIC                         -1015.046
                                - 204
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0026      0.001      1.886      0.059      -0.000       0.005
sigma2         0.0004   3.29e-05     11.610      0.000       0.000       0.000
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                 5.74
Prob(Q):                              0.92   Prob(JB):                         0.06
Heteroskedasticity (H):               1.60   Skew:                            -0.21
Prob(H) (two-sided):                  0.05   Kurtosis:                         3.70
===================================================================================

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



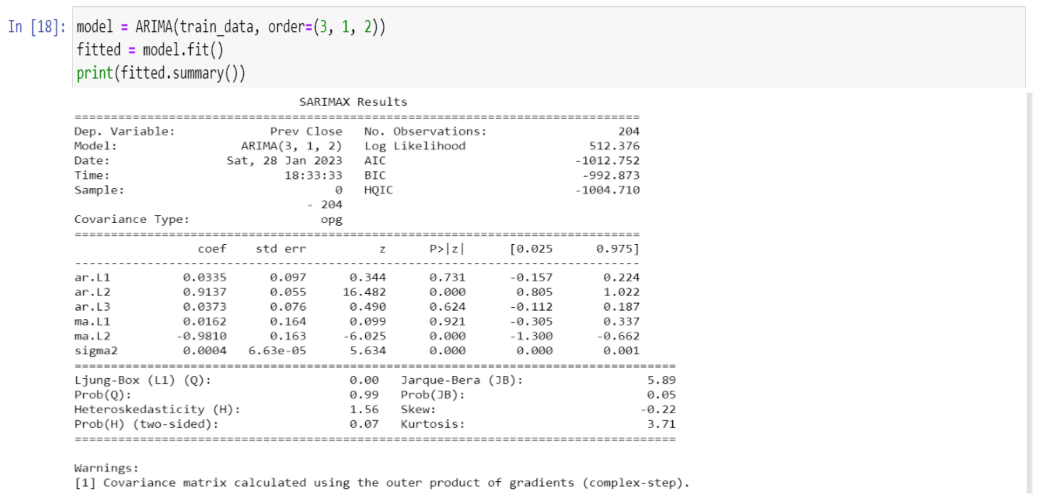
Standardized residual: fluctuates around a mean of zero and has a uniform variance.

Histogram plus estimated density: suggests normal distribution with mean zero.

Normal Q-Q: falls in line with the red line. If there are any deviations, it implies the distribution is skewed.

Correlogram: the ACF plot displays residual errors that are not autocorrelated. If there is any autocorrelation, it implies there are some patterns in the residual error that are not explained in the model and we need to find more predictors for the model.

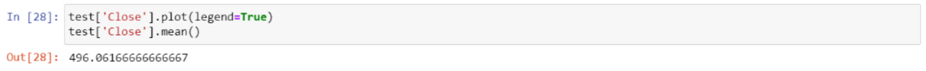
Overall, it seems to be a good fit. Let’s start creating an ARIMA model with provided optimal parameters p, d, and q. forecasting the stock prices.



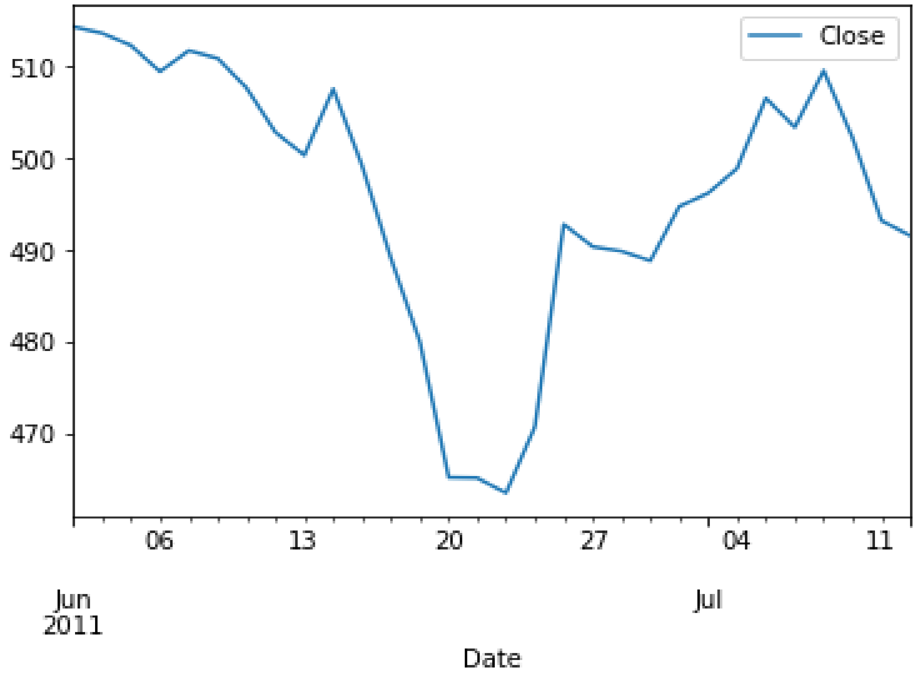
```
In [18]: model = ARIMA(train_data, order=(3, 1, 2))
         fitted = model.fit()
         print(fitted.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:             Prev Close   No. Observations:                  204
Model:                 ARIMA(3, 1, 2)   Log Likelihood                 512.376
Date:                Sat, 28 Jan 2023   AIC                          -1012.752
Time:                        18:33:33   BIC                           -992.873
Sample:                             0   HQIC                         -1004.710
                                - 204
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.0335      0.097      0.344      0.731      -0.157       0.224
ar.L2          0.9137      0.055     16.482      0.000       0.805       1.022
ar.L3          0.0373      0.076      0.490      0.624      -0.112       0.187
ma.L1          0.0162      0.164      0.099      0.921      -0.305       0.337
ma.L2         -0.9810      0.163     -6.025      0.000      -1.300      -0.662
sigma2         0.0004   6.63e-05      5.634      0.000       0.000       0.001
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 5.89
Prob(Q):                              0.99   Prob(JB):                         0.05
Heteroskedasticity (H):               1.56   Skew:                            -0.22
Prob(H) (two-sided):                  0.07   Kurtosis:                         3.71
===================================================================================

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

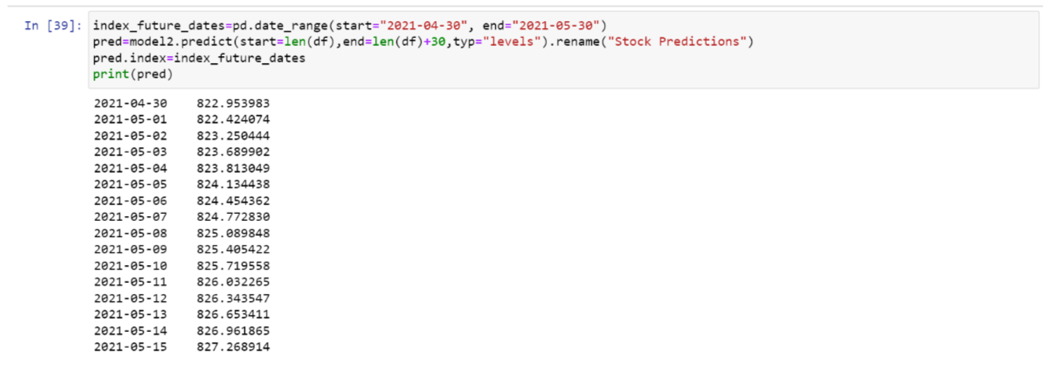
11. Testing



```
In [28]: test['Close'].plot(legend=True)
         test['Close'].mean()

Out[28]: 496.06166666666667
```

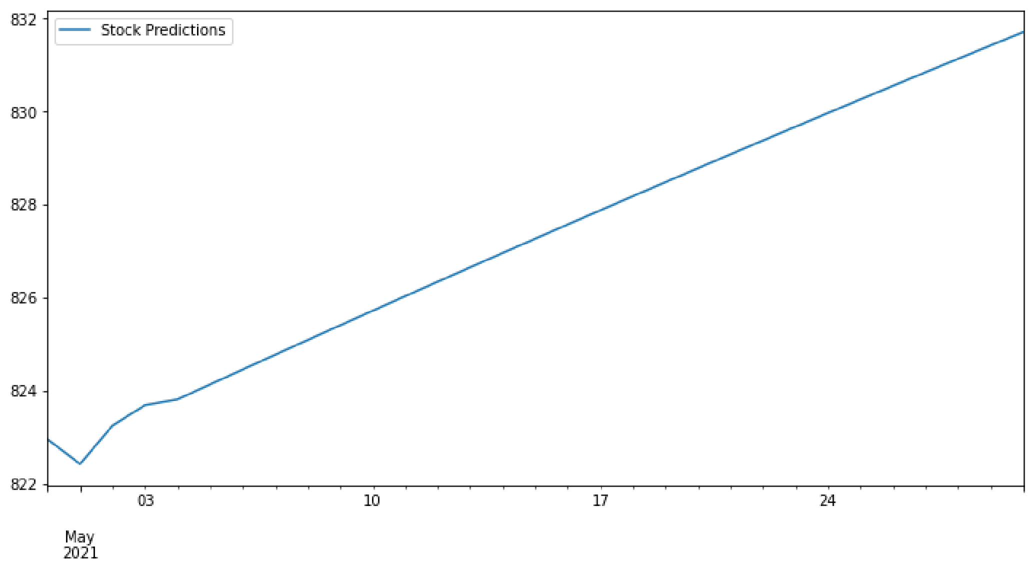


12. Making Prediction



```
In [39]: 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    822.953983
2021-05-01    822.424074
2021-05-02    823.250444
2021-05-03    823.689902
2021-05-04    823.813049
2021-05-05    824.134438
2021-05-06    824.454362
2021-05-07    824.772830
2021-05-08    825.089848
2021-05-09    825.405422
2021-05-10    825.719558
2021-05-11    826.032265
2021-05-12    826.343547
2021-05-13    826.653411
2021-05-14    826.961865
2021-05-15    827.268914
```



❖ **Results:**

Link to the repository:

https://github.com/shaicodes/WIDS-Project-Stock-Market-Prediction-Using-Time-Series-Forecasting-

❖ Learning Value:

Coding tests a variety of abilities. It hones problem-solving and analysis skills, such as finding errors and thinking logically. Further, coding often helps people develop teamwork and interpersonal skills since software and application projects are often cross-disciplinary and collaborative.
Let alone by solving all the errors we learn more than just studying that particular syntax.

- Identify and understand the problem.
- Researching and analyzing.
- Brainstorming.
- Coming up with the best solutions.

These are some of the perks of solving an error in the code.

❖ Tech-stack Used:

Data Source - Kaggle - https://www.kaggle.com/rohanrao/nifty50-stock-market-data

Libraries Used-

- NumPy
- Pandas
- Matplotlib
- Statsmodels
- PMDArima

IDE Used- Jupyter Notebook

Language Used- Python

❖ Suggestions for others:

I encountered a few errors while working on this project which I would like to pinpoint.

- **TypeError**: _fit() got an unexpected keyword argument 'disp'

If you are using the stats model you will encounter this error as the new version does not support 'disp'. Can refer to the link below for the solution.
https://stackoverflow.com/questions/70223523/statsmodel-typeerror-fit-got-an-unexpected-keyword-argument-disp

- **TypeError**: seasonal_decompose() got an unexpected keyword argument 'freq'

If you see this refer to the link below for solution. You just need to replace 'freq' with 'period'
https://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.seasonal_decompose.html

❖ References and Citations:

https://esource.dbs.ie/bitstream/handle/10788/3830/msc_chouksey_s_2019.pdf?sequence=1&isAllowed=y#:~:text=Time%20series%20forecasting%20can%20be,available%20on%20time%20series%20forecasting
https://www.analyticsvidhya.com/blog/2020/11/stock-market-price-trend-prediction-using-time-series-forecasting/
https://www.econstor.eu/bitstream/10419/144781/1/861766660.pdf
https://eprints.lmu.edu.ng/2357/1/UKSim2014_IEEE.pdf
https://medium.com/@lumeilin301/predict-stock-prices-using-arima-in-python-432bce506ddc