Time Series Company Stock And Investment Project

Submitted by:
UTKARSH VARDHAN

ACKNOWLEDGMENT

I would like to express my deepest appreciation to the Team of Flip Robo for giving a full length description of the project and also handling queries and reviewing the project same time. My SME or mentor Astha Mishra has helped me in the formation of this project where I was stuck with some problems and it was cured by my SME , therefore I would like to extend my sincere thanks to Astha Mishra.

I very much appreciate Data-Trained Education for its valuable advice, suggestion and experience they gave me during the training period, because of that only I was able to complete this Project.

There were some errors and problems occurred in between the project solution where I was able to rectify with the help of the internet or webs like kaggle and github etc.

INTRODUCTION

- Business Problem Framing

Company having problems in predicting company's Stock and investment details for the upcoming years. By applying time series techniques to do the predictions. Also, we have to follow the proper procedure to divide your data between training, validation and testing dataset.

Stock detail years: 2009 - 2017

Prediction years: 2018 to 2021

We have been provided with the company's Stock and investment details for the last 8 years.

The file contains the company's last 8 years of stock details and investment details. As a big share of profit is invested in gold and oil, It's important to keep a watch over it.

We need to perform time series analysis over their dataset and give predictions for the next 4 years.This has to be done by observing model output and comparing it with 2018 - 2020.

- Review of Literature

According to my analysis and understanding of the project it has been elevated that the organization has the consistent investment on gold and oil throughout the year. This research is done in python language which is able to predict the nature of the trend of investment on gold and oil and also the company stock for every month. Most of our models are predicting a good accuracy score which is going to help the clients in predicting their future decisions.

- Motivation for the Problem Undertaken

The motivation and the objective behind this project is to work on the data with all the techniques and enhance the result of the models so that our model predicts a higher accuracy score. This is because our clients will be able to predict the future decisions on the basis of investment and company stock. How much the company is having: stock accordingly they can invest in the gold and oil or which is more profitable.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem
  1. Stock detail years: 2009 - 2017
  2. Prediction years: 2018 to 2021
  3. The file contains the company's last 8 years of stock details and investment details. As a big share of profit is invested in gold and oil, It's important to keep a watch over it.
  4. You need to perform time series analysis over their dataset and give predictions for the next 4 years.
  5. This has to be done by observing model output and comparing it with 2018 - 2020.
  6. Features for analysing the model is two main investment i.e. Gold and Oil other than this we have Company stock.
  7. There are some values which are "0" , I have treated the same by replacing it with mode and mean values.
  8. Shape of the data is 5 columns and 1984 rows, within that we have removed 1 column of others investment which is of no use to us.
  9. With this dataset we have one common column of date which will be used in predicting for all other columns.
  10. We have divided our one project into 3 different separate project for Oil , Gold and Company stock.
- Data Sources and their formats

1) Data types :

```
Date            object
Oil             float64
Gold            float64
Comp_stock      float64
Other_share     float64
dtype: object
```

2) Data Description:

	Oil	**Gold**	**Comp_stock**	**Other_share**
count	1984.000000	1984.000000	1984.000000	1984.000000
mean	-0.000435	-0.020152	0.001007	0.001269
std	0.030869	0.140965	0.016017	0.019733
min	-1.000000	-1.000000	-0.123558	-0.126568

25%	-0.011021	-0.005881	-0.006926	-0.008492
50%	0.000277	0.000000	0.000876	0.000840
75%	0.010734	0.005454	0.009708	0.011632
max	0.119511	0.049577	0.088741	0.157457

The above description states the total count of the entry made that is rows, it also shows the mean , min , max, std 25% and 75% such as , taking in consideration of the Oil that is the investment made on the oil by the company which has 1984 count where mean or average days is -0.000435. Minimum investment is 1.000000 where maximum is 0.119511. It also states the 25% which is -0.011021 and 75% is 0.010734. As we could see that the 3rd quartile is greater than the mean of the data that shows the dataset is containing outliers which might be removed later , likewise we could detect other variables too, that is Gold , Comp_stock.

- Data Preprocessing Done

1) Synthesizing the date column which was done while loading the data file in python:

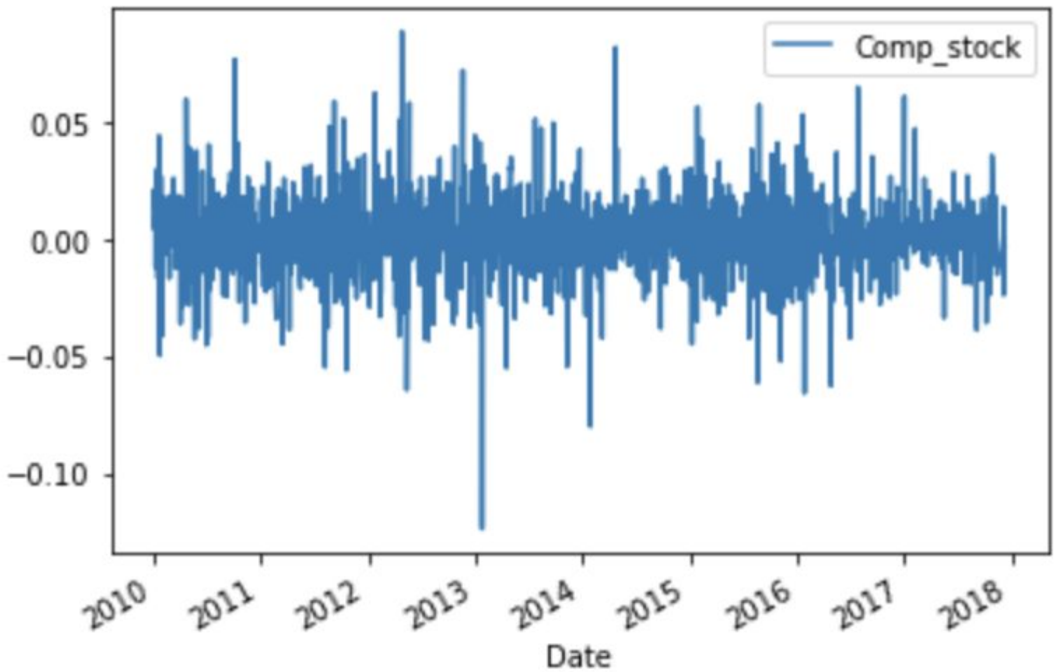
```
#loading the given datasets:
df= pd.DataFrame(df1)

# convert the datetime column to a datetime type
df.Date = pd.to_datetime(df.Date)

# set the column as the index
df.set_index('Date', inplace=True)
```

2) I have Dropped Other Investment as such data would not be used for predicting.
3) Checking for outliers:

```
import seaborn as sns
columns={'Oil', 'Gold', 'Comp_stock', 'Other_share'}
for i in columns:
    plt.figure()
    plt.clf()
    sns.boxplot(df[i],palette="deep")
```

```
plt.title(i)
plt.show()
```

4) Plotting Company stock:



According to the above graph we see that the trend is almost uneven and great fall in the company stock was in the starting of 2014.

- Hardware and Software Requirements and Tools Used

import numpy as np: numpy is used in the dataset for working with the arrays, working in domain of linear algebra, fourier transform, and matrices
import pandas as pd : Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.
import matplotlib.pyplot as plt: It is used to check the missing values in our dataset also used for the histogram which is to detect the count of various features lying in different groups .
import seaborn as sns : Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

from sklearn.model_selection import GridSearchCV: GridSearchCV is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters.
from scipy.stats import z score: The standard score (more commonly referred to as a z-score) is a very useful statistic because it (a) allows us to calculate the probability of a score occurring within our normal distribution and (b) enables us to compare two scores that are from different normal distributions.
import warnings
warnings.filterwarnings('ignore') : The warn() function defined in the ' warning ' module is used to show warning messages. The warning module is actually a subclass of Exception which is a built-in class in Python. filter_none. # program to display a warning message. import warnings.
from sklearn.externals import joblib: To save the file in object format.
from statsmodels.tsa.stattools import adfuller : The null hypothesis of the Augmented Dickey-Fuller is that there is a unit root, with the alternative that there is no unit root. If the p value is above a critical size, then we cannot reject that there is a unit root. The p-values are obtained through regression surface approximation from MacKinnon 1994, but using the updated 2010 tables. If the p-value is close to significant, then the critical values should be used to judge whether to reject the null.
from statsmodels.tsa.arima_model import ARIMA : AR-Auto Regressive + MA -- Moving Average AR (p=auto regresive lags)+I(Integration d= order of differentiation)+ MA (q=Moving AVG)

- Very PowerFull Model
    - AR and MA are seperate model binded by intregation. #### AR is basically corelation between previous time period to current.

* We plot partial autocorelation graph to predict the value of p.= PACF

- #### MA we basically take the avg of event happened in diff t1 t2 t3 e.i. time periods.
- * For q value we plot Auto Corelation Plot = ACF

import math: These functions cannot be used with complex numbers; use the functions of the same name from the cmath module if you require support for complex numbers.
from sklearn.metrics import mean_squared_error, mean_absolute_error:

MAE: It is not very sensitive to outliers in comparison to MSE since it doesn't punish huge errors. It is usually used when the performance is measured on continuous variable data. It gives a linear value, which averages the weighted individual differences equally. The lower the value, better is the model's performance. **MSE:** It is one of the most commonly used metrics, but least useful when a single bad prediction would ruin the entire model's predicting abilities, i.e when the dataset contains a lot of noise. It is most useful when the dataset contains outliers, or unexpected values (too high or too low values). **RMSE:** In RMSE, the errors are squared before they are averaged. This basically implies that RMSE assigns a higher weight to larger errors. This indicates that RMSE is much more useful when large errors are present and they drastically affect the model's performance. It avoids taking the absolute value of the error and this trait is useful in many mathematical calculations. In this metric also, lower the value, better is the performance of the model.
from pandas.plotting import autocorrelation_plot : Autocorrelation plots are a commonly-used tool for checking randomness in a data set. This randomness is ascertained by computing autocorrelations for data values at varying time lags. If random, such autocorrelations should be near zero for any and all time-lag separations. If non-random, then one or more of the autocorrelations will be significantly non-zero.
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf: Autocorrelation and Partial Autocorrelation

- Identification of an AR model is often best done with the PACF.
  - For an AR model, the theoretical PACF "shuts off" past the order of the model. The phrase "shuts off" means that in theory the partial autocorrelations are equal to 0 beyond that point. Put another way, the number of non-zero partial autocorrelations gives the order of the AR model. By the "order of the model" we mean the most extreme lag of x that is used as a predictor.
- Identification of an MA model is often best done with the ACF rather than the PACF.
  - For an MA model, the theoretical PACF does not shut off, but instead tapers toward 0 in some manner. A clearer pattern for an MA model is in the ACF. The ACF will have non-zero autocorrelations only at lags involved in the model.

import statsmodels.api as sm: statsmodels supports specifying models using R-style formulas and pandas DataFrames. Here is a simple example using ordinary least squares: In [1]: import numpy as np In [2]: import statsmodels. api as sm In [3]: import statsmodels.

from pmdarima.arima import auto_arima : The auto_arima function fits the best ARIMA model to a univariate time series according to a provided information criterion (either AIC, AICc, BIC or HQIC). The function performs a search (either stepwise or parallelized) over possible model & seasonal orders within the constraints provided, and selects the parameters that minimize the given metric. The auto_arima function can be daunting. There are a lot of parameters to tune, and the outcome is heavily dependent on a number of them. In this section, we lay out several considerations you'll want to make when you fit your ARIMA models.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods) & Testing of Identified Approaches (Algorithms)

Model Testing in finding the Accuracy score:

Company_Stock:

```
# adfuller give us 5 values
test_result=adfuller(df_Comp_Stock['Comp_stock'])

# just showing the ouput of test_result
print(" The Values given as output by adfuller is : \n 'ADF Test Statistic','p-value','#Lags Used','Number of Observations Used'\n\n",test_result)

#Ho: It is non stationary
#H1: It is stationary

def adfuller_test_Comp_stock(Comp_stock):
    result=adfuller(Comp_stock)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations Used']
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )
    if result[1] <= 0.05:
```

```
    print("strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data is stationary")
  else:
    print("weak evidence against null hypothesis, time series is non-stationary ")

adfuller_test_Comp_stock(df_Comp_Stock['Comp_stock'])
```

Auto Regressive Model

```
from pandas.plotting import autocorrelation_plot
autocorrelation_plot(df_Comp_Stock['Comp_stock'])
plt.show()

from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import statsmodels.api as sm
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
# No  values are null so we will start from 1
fig = sm.graphics.tsa.plot_acf(df_Comp_Stock['Comp_stock'].iloc[1:],lags=40,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df_Comp_Stock['Comp_stock'].iloc[1:],lags=40,ax=ax2)


#split data into train and training set
df_log=df_Comp_Stock
train_data, test_data = df_log[3:int(len(df_log)*0.9)], df_log[int(len(df_log)*0.9):]
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel('Dates')
plt.ylabel('Comp_Stock')
plt.plot(df_log, 'green', label='Train data')
plt.plot(test_data, 'blue', label='Test data')
plt.legend()


from pmdarima.arima import auto_arima
```

```
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())

from statsmodels.tsa.arima_model import ARIMA
model=ARIMA(df_Comp_Stock['Comp_stock'],order=(0,0,0)) # this order is p d q(0 or 1)
model_fit=model.fit()
model_fit.summary()

# Forecast
fc, se, conf = model_fit.forecast(199, alpha=0.05)  # 95% confidence
fc_series = pd.Series(fc, index=test_data.index)
lower_series = pd.Series(conf[:, 0], index=test_data.index)
upper_series = pd.Series(conf[:, 1], index=test_data.index)
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train_data, label='training')
plt.plot(test_data, color = 'blue', label='Comp_stock')
plt.plot(fc_series, color = 'orange',label='forecast')
plt.fill_between(lower_series.index, lower_series, upper_series,
                 color='k', alpha=.10)
plt.title('Company Stock  Prediction')
plt.xlabel('Time')
plt.ylabel('Comp_stock')
```

```
plt.legend(loc='upper left', fontsize=8)
plt.show()

from sklearn.metrics import mean_squared_error, mean_absolute_error
import math
# report performance
mse = mean_squared_error(test_data, fc)
print('MSE: '+str(mse))
mae = mean_absolute_error(test_data, fc)
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(test_data, fc))
print('RMSE: '+str(rmse))

import statsmodels.api as sm  # SARIMAX - seasonal arimax
model=sm.tsa.statespace.SARIMAX(df_Comp_Stock['Comp_stock'],order=(0, 0, 0)) #(p,d,q)
results_stock=model.fit()
df_Comp_Stock['forecast_stock']=results_stock.predict(start=1781,end=1981,dynamic=False)
df_Comp_Stock[['Comp_stock','forecast_stock']].plot(figsize=(12,8))
```

Gold - Testing For Stationarity

```
#plot Gold
df_gold.plot()


df_gold = df_gold.replace(0, np.nan)
df_gold
df_gold.isnull().sum()
df_gold["Gold"] = df_gold["Gold"].fillna(df_gold["Gold"].dropna().mode().values[0] )
test_result_gold=adfuller(df_gold['Gold']) # adfuller give us 5 values
# just showing the ouput of test_result
print(" The Values given as output by adfuller is : \n 'ADF Test Statistic','p-value','#Lags Used','Number of Observations Used'\n\n",test_result_gold)
#Ho: It is non stationary
#H1: It is stationary
```

```
def adfuller_test(Gold):
    result=adfuller(Gold)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations Used']
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )
    if result[1] <= 0.05:
        print("strong evidence against the null hypothesis(Ho), reject the null hypothesis.
Data  is stationary")
    else:
        print("weak evidence against null hypothesis, time series is non-stationary ")
adfuller_test(df_gold['Gold'])
df_gold=df_gold.dropna(how='any')
autocorrelation_plot(df_gold['Gold'])
plt.show()

fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
# No values are null so we will start from 1
fig = sm.graphics.tsa.plot_acf(df_gold['Gold'].iloc[1:],lags=40,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df_gold['Gold'].iloc[1:],lags=40,ax=ax2)

#split data into train and training set
df_log=df_gold
train_data_gold, test_data_gold = df_log[3:int(len(df_log)*0.9)],
df_log[int(len(df_log)*0.9):]
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel('Dates')
plt.ylabel('Gold')
plt.plot(df_log, 'green', label='Train data')
plt.plot(test_data_gold, 'blue', label='Test data')
plt.legend()

from pmdarima.arima import auto_arima
model_autoARIMA = auto_arima(train_data_gold, 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())

from statsmodels.tsa.arima_model import ARIMA
model_gold=ARIMA(df_gold['Gold'],order=(3,0,2)) # this order is p d q
model_fit_gold=model_gold.fit(disp=-1)
model_fit_gold.summary()

# Forecast
fc, se, conf = model_fit_gold.forecast(199, alpha=0.05)  # 95% confidence
fc_series = pd.Series(fc, index=test_data_gold.index)
lower_series = pd.Series(conf[:, 0], index=test_data_gold.index)
upper_series = pd.Series(conf[:, 1], index=test_data_gold.index)
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train_data_gold, label='training')
plt.plot(test_data_gold, color = 'blue', label='Gold')
plt.plot(fc_series, color = 'orange',label='forecast_gold')
plt.fill_between(lower_series.index, lower_series, upper_series,
            color='k', alpha=.10)
plt.title('Gold Investment Prediction')
plt.xlabel('Time')
plt.ylabel('Gold')
plt.legend(loc='upper left', fontsize=8)
plt.show()

import statsmodels.api as sm  # SARIMAX - seasonal arimax
model=sm.tsa.statespace.SARIMAX(df_gold['Gold'],order=(3, 0, 2)) #(p,d,q)
results=model.fit()
df_gold['forecast_gold']=results.predict(start=1781,end=1982,dynamic=False)
df_gold[['Gold','forecast_gold']].plot(figsize=(12,8))

from sklearn.metrics import mean_squared_error, mean_absolute_error
import math
# report performance
mse = mean_squared_error(test_data_gold, fc)
print('MSE: '+str(mse))
mae = mean_absolute_error(test_data_gold, fc)
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(test_data_gold, fc))
print('RMSE: '+str(rmse))
```

Oil - Testing For Stationarity

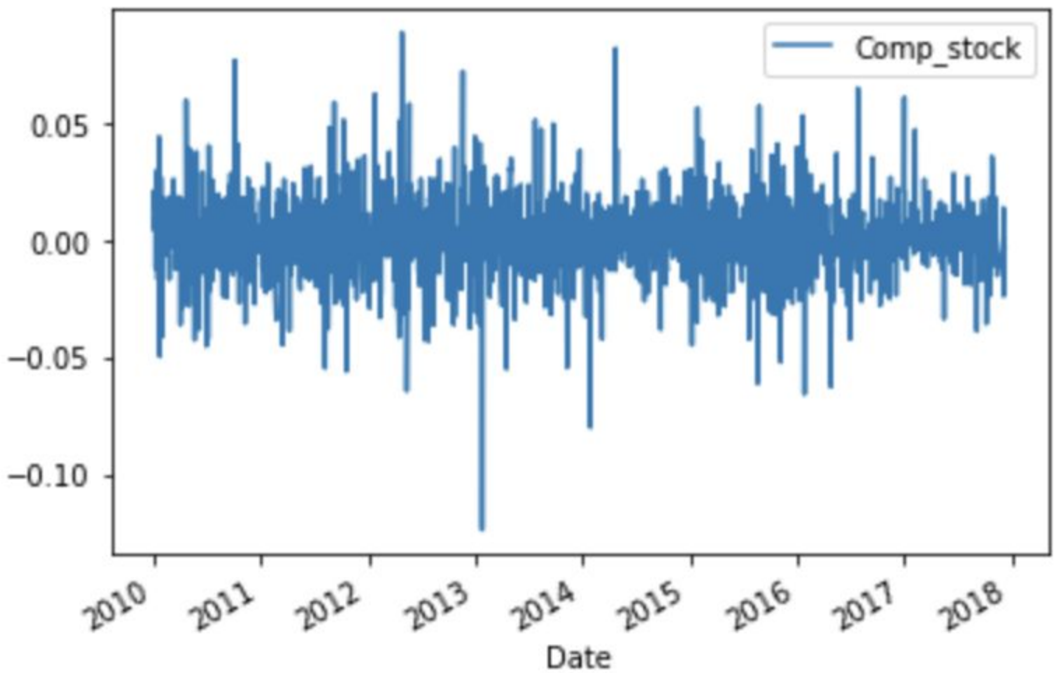
```
#plot Comp_stock
df_oil.plot()

df_oil = df_oil.replace(0, np.nan)
df_oil

df_oil.isnull().sum()

df_oil.fillna(df_oil.mean(), inplace=True)

df_oil.Oil.round(3)

df_oil.shape

from scipy.stats import zscore
z_score=abs(zscore(df_oil))
print(df_oil.shape)

df_Oil=df_oil.loc[(z_score<2).all(axis=1)]
print(df_Oil.shape)

df_Oil.isnull().sum()

test_result_oil=adfuller(df_Oil['Oil']) # adfuller give us 5 values

# just showing the ouput of test_result
print(" The Values given as output by adfuller is : \n 'ADF Test Statistic','p-value','#Lags Used','Number of Observations Used'\n\n",test_result_oil)

#Ho: It is non stationary
#H1: It is stationary

def adfuller_test_oil(Oil):
   result=adfuller(Oil)
   labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations Used']
   for value,label in zip(result,labels):
      print(label+' : '+str(value) )
   if result[1] <= 0.05:
      print("strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data  is stationary")
   else:
      print("weak evidence against null hypothesis, time series is non-stationary ")
adfuller_test_oil(df_Oil['Oil'])
df_Oil.plot()

autocorrelation_plot(df_Oil['Oil'])
```

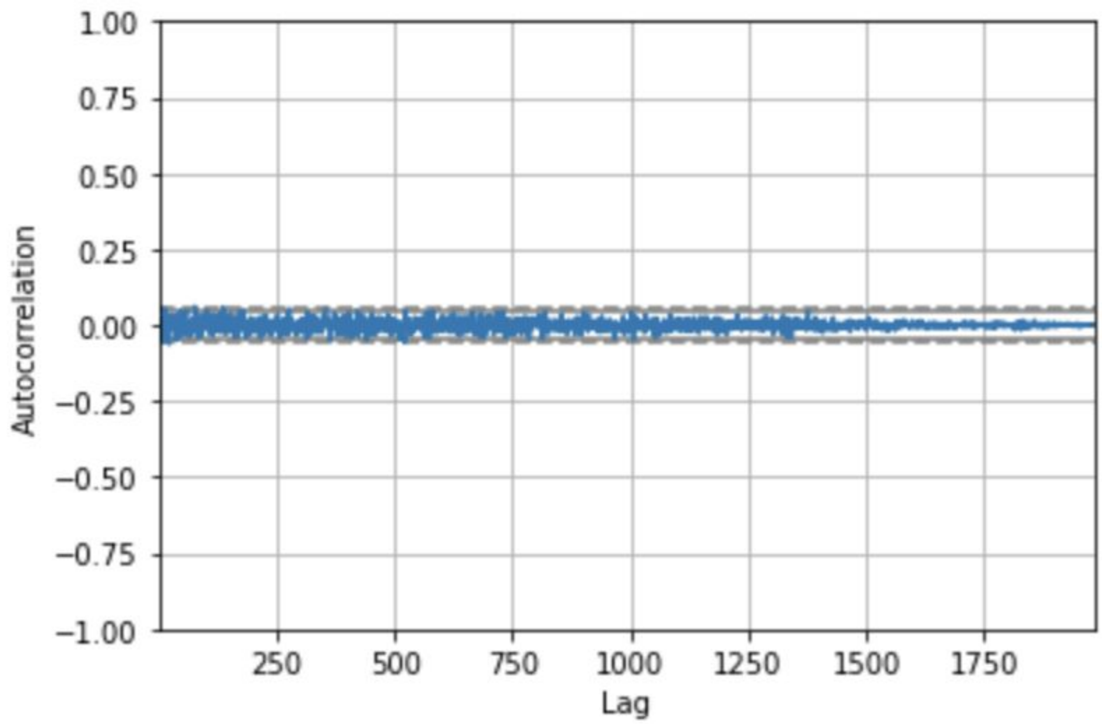
```
plt.show()

fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
# No values are null so we will start from 1
fig = sm.graphics.tsa.plot_acf(df_Oil['Oil'].iloc[1:],lags=40,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df_Oil['Oil'].iloc[1:],lags=40,ax=ax2)

#split data into train and training set
df_log=df_Oil
train_data_oil, test_data_oil = df_log[3:int(len(df_log)*0.9)], df_log[int(len(df_log)*0.9):]
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel('Dates')
plt.ylabel('Oil')
plt.plot(df_log, 'green', label='Train data')
plt.plot(test_data_oil, 'blue', label='Test data')
plt.legend()

from pmdarima.arima import auto_arima
model_autoARIMA = auto_arima(train_data_oil, 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())

from statsmodels.tsa.arima_model import ARIMA
model_oil=ARIMA(df_Oil['Oil'],order=(1,0,1)) # this order is p d q
model_fit_oil=model_oil.fit(disp=-1)
model_fit_oil.summary()

# Forecast
fc, se, conf = model_fit_oil.forecast(196, alpha=0.05)  # 95% confidence
fc_series = pd.Series(fc, index=test_data_oil.index)
lower_series = pd.Series(conf[:, 0], index=test_data_oil.index)
upper_series = pd.Series(conf[:, 1], index=test_data_oil.index)
plt.figure(figsize=(12,5), dpi=100)
```

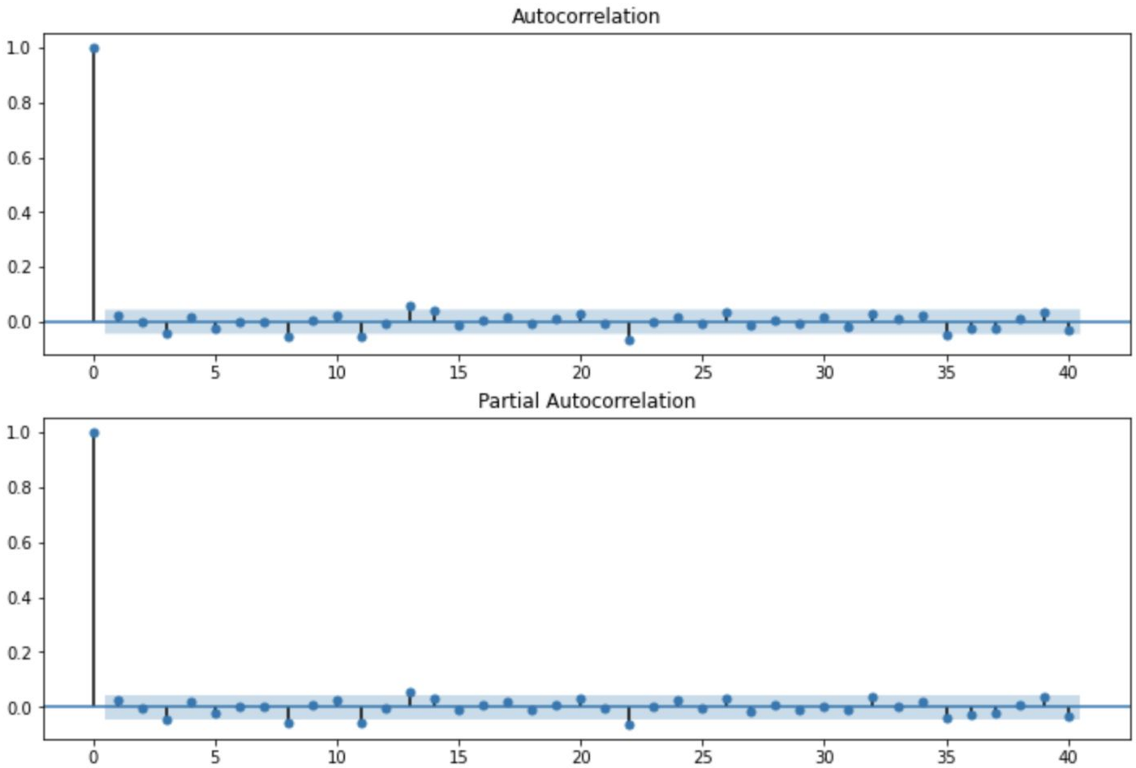
```
plt.plot(train_data_oil, label='training')
plt.plot(test_data_oil, color = 'blue', label='Oil')
plt.plot(fc_series, color = 'orange',label='forecast_Oil')
plt.fill_between(lower_series.index, lower_series, upper_series,
           color='k', alpha=.10)
plt.title('Oil Investment Prediction')
plt.xlabel('Time')
plt.ylabel('Oil')
plt.legend(loc='upper left', fontsize=8)
plt.show()

import statsmodels.api as sm  # SARIMAX - seasonal arimax
model=sm.tsa.statespace.SARIMAX(df_Oil['Oil'],order=(0, 0, 0)) #(p,d,q)
results=model.fit()
df_Oil['forecast_Oil']=results.predict(start=1756,end=1954,dynamic=False)
df_Oil[['Oil','forecast_Oil']].plot(figsize=(12,8))

from sklearn.metrics import mean_squared_error, mean_absolute_error
import math
# report performance
mse = mean_squared_error(test_data_oil, fc)
print('MSE: '+str(mse))
mae = mean_absolute_error(test_data_oil, fc)
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(test_data_oil, fc))
print('RMSE: '+str(rmse))
```

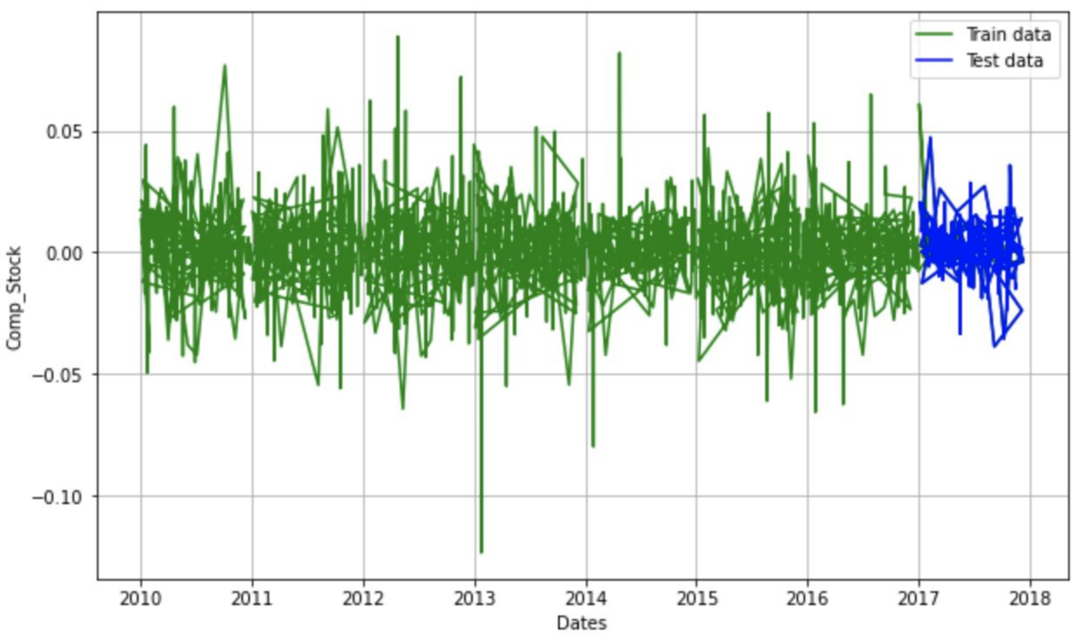
- Run and Evaluate selected models & Visualizations :

Company Stock





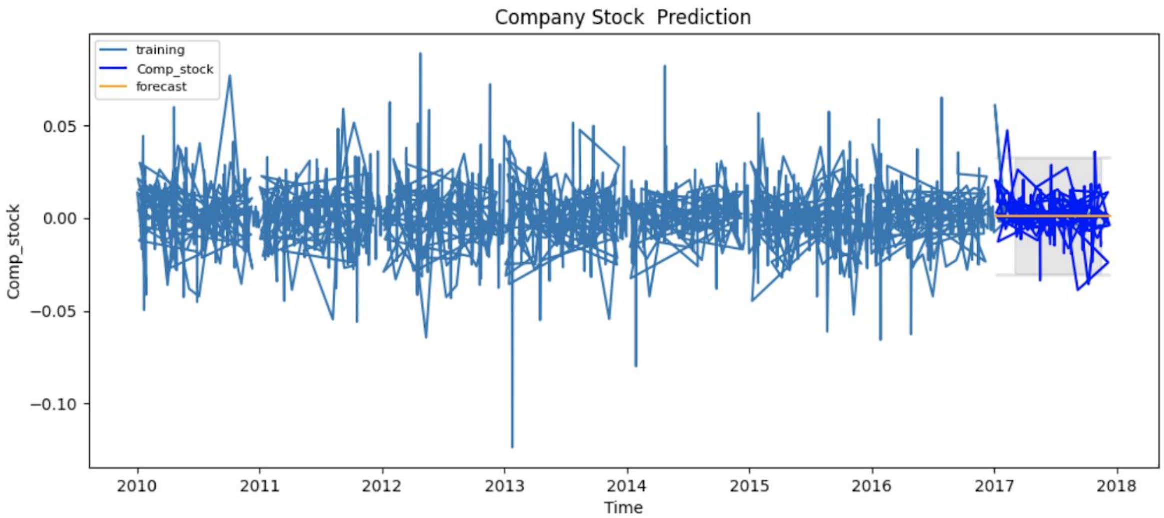

Autocorrelation
Partial Autocorrelation

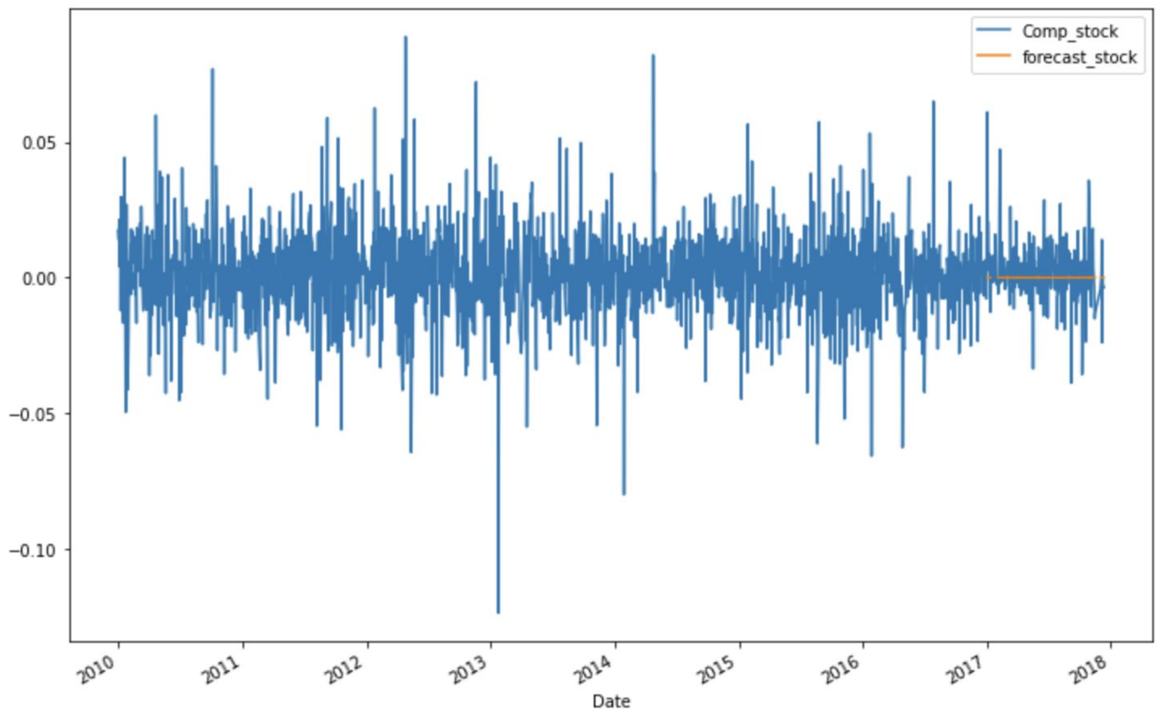

Train data
Test data
Comp_Stock
Dates

```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=-9553.320, Time=0.12 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=-9552.460, Time=0.05 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=-9552.456, Time=0.25 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=-9550.460, Time=0.28 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=-9557.120, Time=0.30 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=-9555.987, Time=0.17 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=-9555.990, Time=0.40 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=-9553.950, Time=0.39 sec

Best model:  ARIMA(0,0,0)(0,0,0)[0] intercept
Total fit time: 1.978 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1780
Model:                        SARIMAX   Log Likelihood                4780.560
Date:                Sun, 01 Nov 2020   AIC                          -9557.120
Time:                        09:26:04   BIC                          -9546.151
Sample:                             0   HQIC                         -9553.068
                               - 1780
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0009      0.000      2.408      0.016       0.000       0.002
sigma2         0.0003   5.15e-06     52.885      0.000       0.000       0.000
===================================================================================
Ljung-Box (Q):                       54.70   Jarque-Bera (JB):              1370.46
Prob(Q):                              0.06   Prob(JB):                         0.00
Heteroskedasticity (H):               0.82   Skew:                            -0.10
Prob(H) (two-sided):                  0.02   Kurtosis:                         7.29
===================================================================================

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

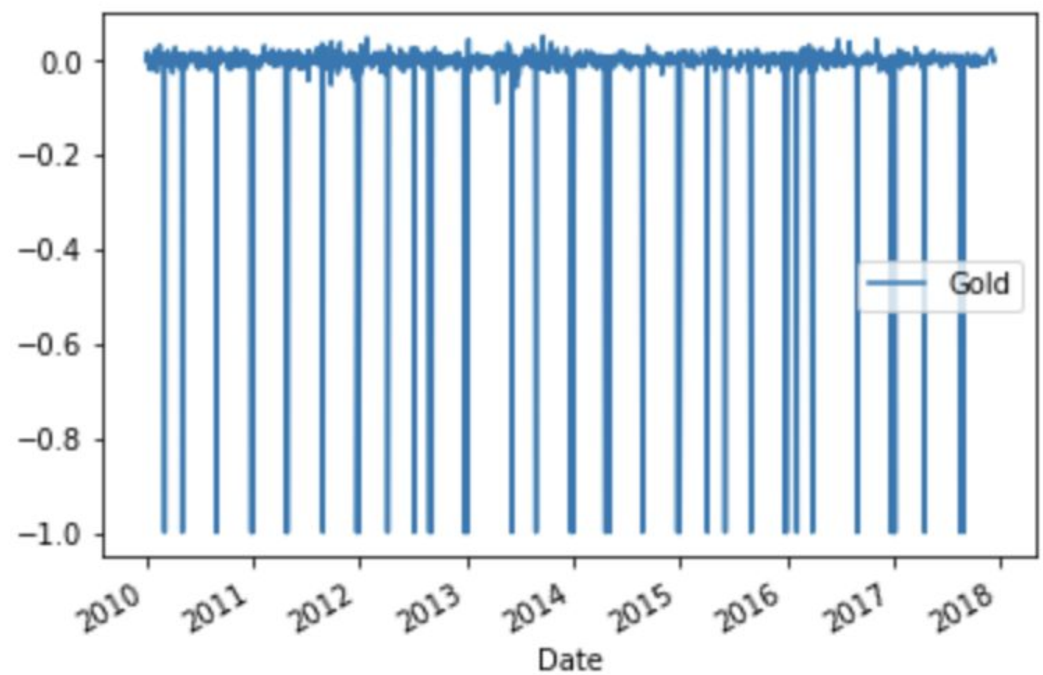
Dep. Variable:	Comp_stock	**No. Observations:**	1982
Model:	ARMA(0, 0)	**Log Likelihood**	5381.003
Method:	css	**S.D. of innovations**	0.016
Date:	Sun, 01 Nov 2020	**AIC**	-10758.006
Time:	09:26:04	**BIC**	-10746.823
Sample:	0	**HQIC**	-10753.898

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	0.0010	0.000	2.802	0.005	0.000	0.002

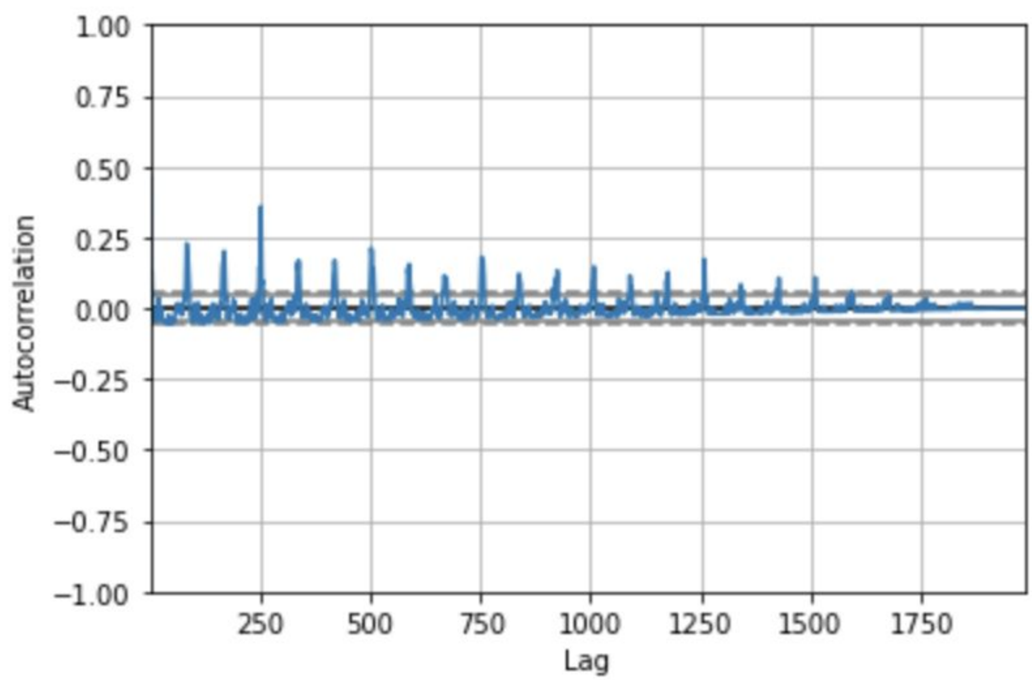

Company Stock Prediction
training
Comp_stock
forecast
0.05
0.00
−0.05
−0.10
Comp_stock
2010
2011
2012
2013
2014
2015
2016
2017
2018
Time

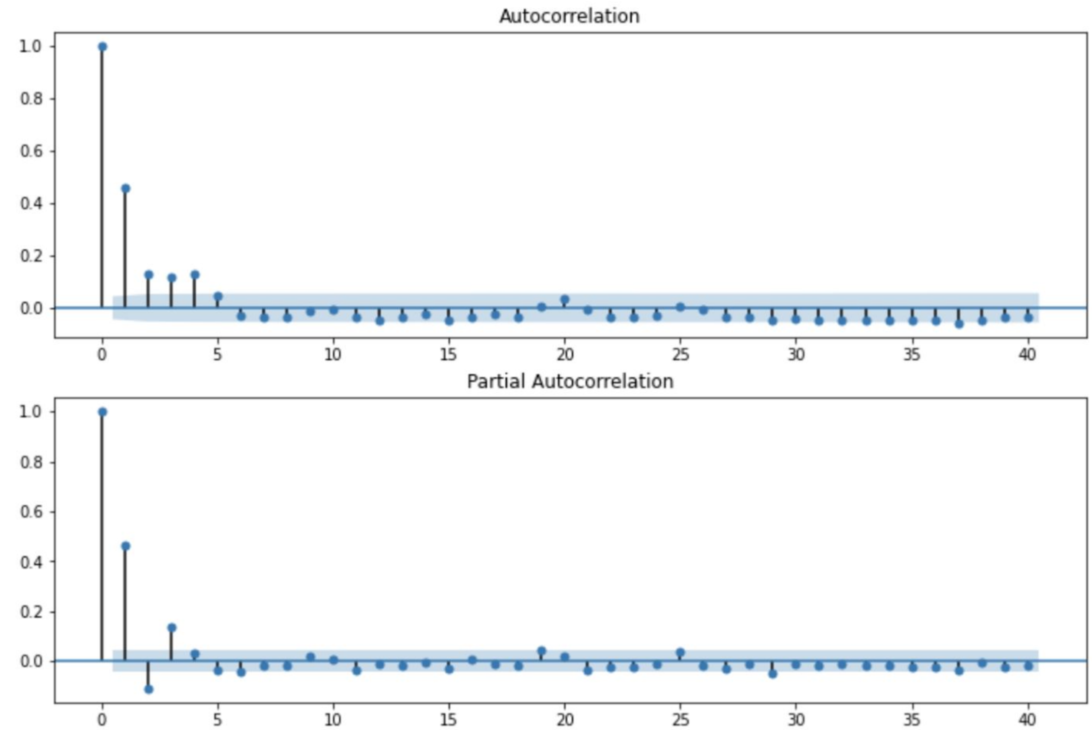

Comp_stock
forecast_stock
0.05
0.00
−0.05
−0.10
2010
2011
2012
2013
2014
2015
2016
2017
2018
Date

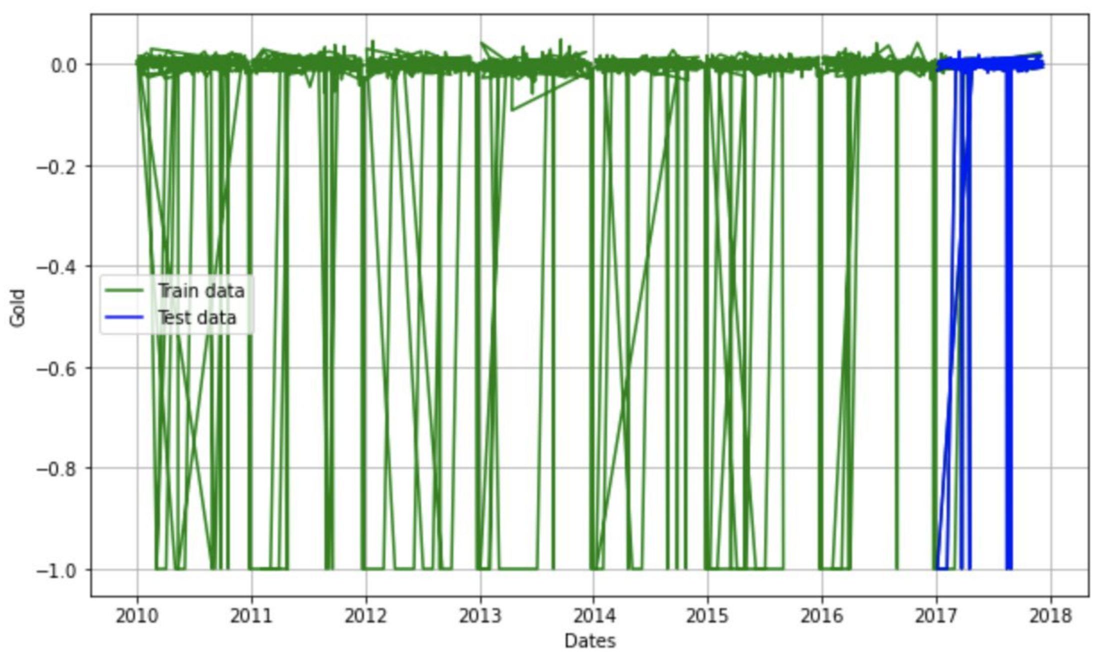
Gold :




```
ADF Test Statistic : -16.314261015489084
p-value : 3.1736637996189835e-29
#Lags Used : 5
Number of Observations Used : 1978
strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data  is stat
ionary
```




Autocorrelation
Partial Autocorrelation

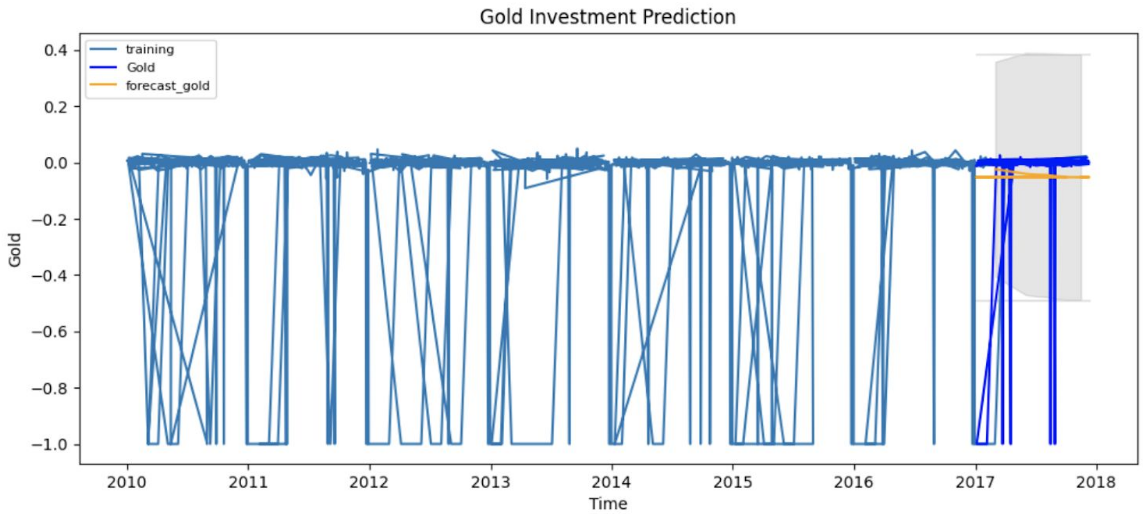

Train data
Test data
Gold
Dates

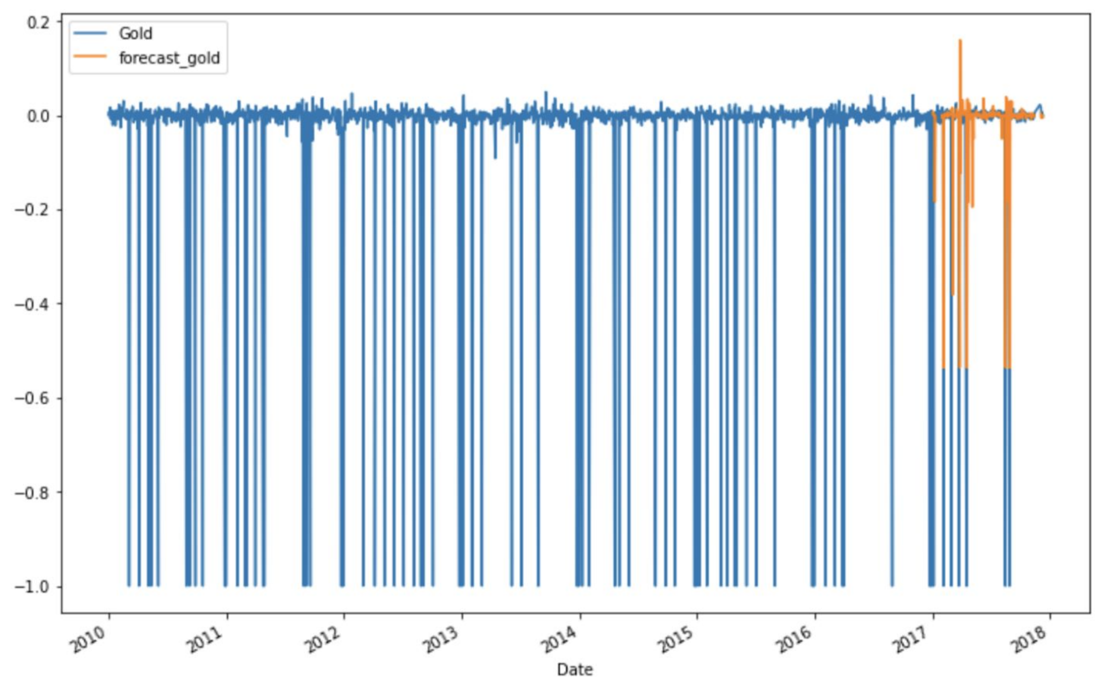
```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=-161.252, Time=0.12 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=-656.840, Time=0.04 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=-637.818, Time=0.14 sec
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=-664.217, Time=0.10 sec
 ARIMA(3,0,0)(0,0,0)[0]             : AIC=-705.590, Time=0.16 sec
 ARIMA(3,0,1)(0,0,0)[0]             : AIC=-707.856, Time=0.55 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=-676.681, Time=0.48 sec
 ARIMA(3,0,2)(0,0,0)[0]             : AIC=-710.829, Time=0.57 sec
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=-695.624, Time=0.62 sec
 ARIMA(3,0,3)(0,0,0)[0]             : AIC=-708.908, Time=1.11 sec
 ARIMA(2,0,3)(0,0,0)[0]             : AIC=-698.849, Time=1.12 sec
 ARIMA(3,0,2)(0,0,0)[0] intercept   : AIC=-737.960, Time=1.83 sec
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=-722.468, Time=2.32 sec
 ARIMA(3,0,1)(0,0,0)[0] intercept   : AIC=-733.715, Time=1.73 sec
 ARIMA(3,0,3)(0,0,0)[0] intercept   : AIC=-736.705, Time=2.71 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=-714.296, Time=1.41 sec
 ARIMA(2,0,3)(0,0,0)[0] intercept   : AIC=-729.430, Time=5.79 sec

Best model:  ARIMA(3,0,2)(0,0,0)[0] intercept
Total fit time: 20.840 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1782
Model:               SARIMAX(3, 0, 2)   Log Likelihood                 375.980
Date:                Sun, 01 Nov 2020   AIC                           -737.960
Time:                        09:26:27   BIC                           -699.561
Sample:                             0   HQIC                          -723.778
                               - 1782
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -0.0293      0.020     -1.469      0.142      -0.068       0.010
ar.L1          0.7030      0.123      5.723      0.000       0.462       0.944
ar.L2         -0.5414      0.118     -4.591      0.000      -0.772      -0.310
ar.L3          0.2876      0.053      5.451      0.000       0.184       0.391
ma.L1         -0.1888      0.124     -1.526      0.127      -0.431       0.054
ma.L2          0.2961      0.130      2.279      0.023       0.041       0.551
sigma2         0.0384      0.001     29.286      0.000       0.036       0.041
===================================================================================
Ljung-Box (Q):                       33.19   Jarque-Bera (JB):             19385.38
Prob(Q):                              0.77   Prob(JB):                         0.00
Heteroskedasticity (H):               0.88   Skew:                            -3.24
Prob(H) (two-sided):                  0.12   Kurtosis:                        17.81
===================================================================================

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

Dep. Variable:	Gold	**No. Observations:**	1984
Model:	ARMA(3, 2)	**Log Likelihood**	431.004
Method:	css-mle	**S.D. of innovations**	0.195
Date:	Sun, 01 Nov 2020	**AIC**	-848.008
Time:	09:26:28	**BIC**	-808.857
Sample:	0	**HQIC**	-833.627

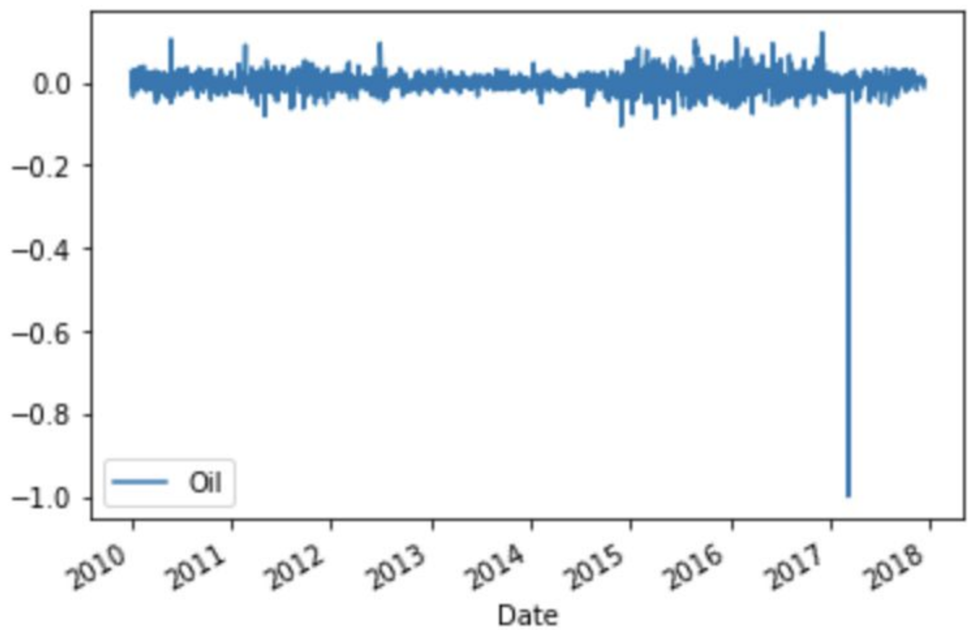
	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	-0.0523	0.009	-6.109	0.000	-0.069	-0.036
ar.L1.Gold	0.6659	0.139	4.800	0.000	0.394	0.938
ar.L2.Gold	-0.4955	0.126	-3.942	0.000	-0.742	-0.249
ar.L3.Gold	0.2640	0.051	5.228	0.000	0.165	0.363
ma.L1.Gold	-0.1464	0.143	-1.024	0.306	-0.427	0.134
ma.L2.Gold	0.2557	0.107	2.401	0.016	0.047	0.464

	Real	**Imaginary**	**Modulus**	**Frequency**
AR.1	0.0873	-1.4891j	1.4916	-0.2407
AR.2	0.0873	+1.4891j	1.4916	0.2407
AR.3	1.7023	-0.0000j	1.7023	-0.0000
MA.1	0.2862	-1.9567j	1.9775	-0.2269
MA.2	0.2862	+1.9567j	1.9775	0.2269

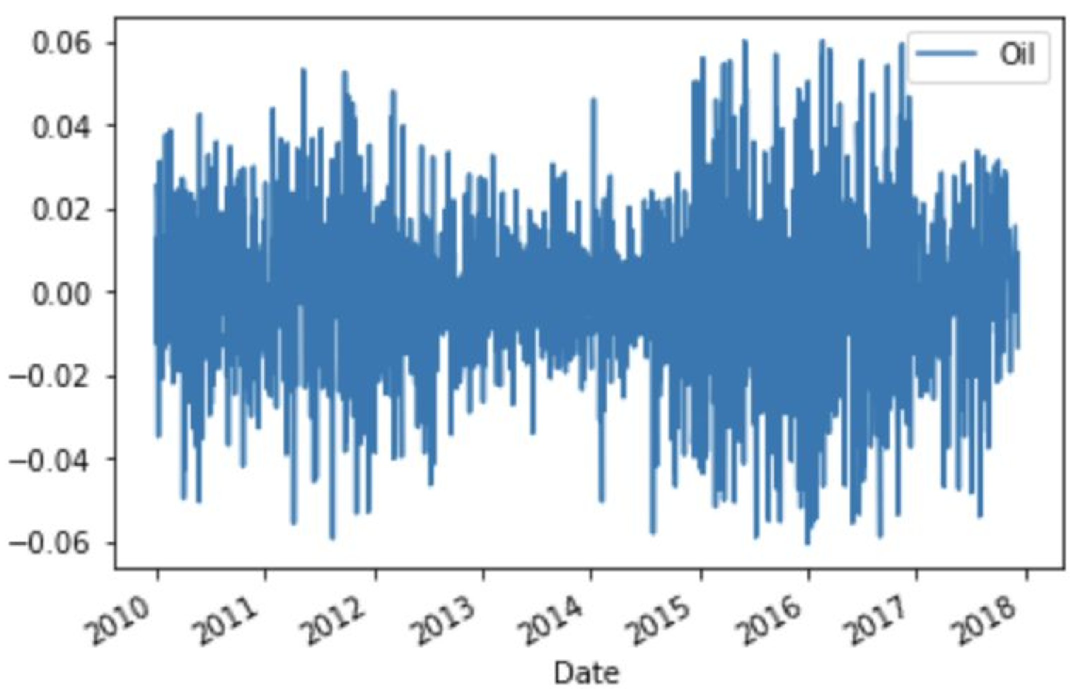

Gold Investment Prediction
training
Gold
forecast_gold
Gold
Time
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
2010
2011
2012
2013
2014
2015
2016
2017
2018

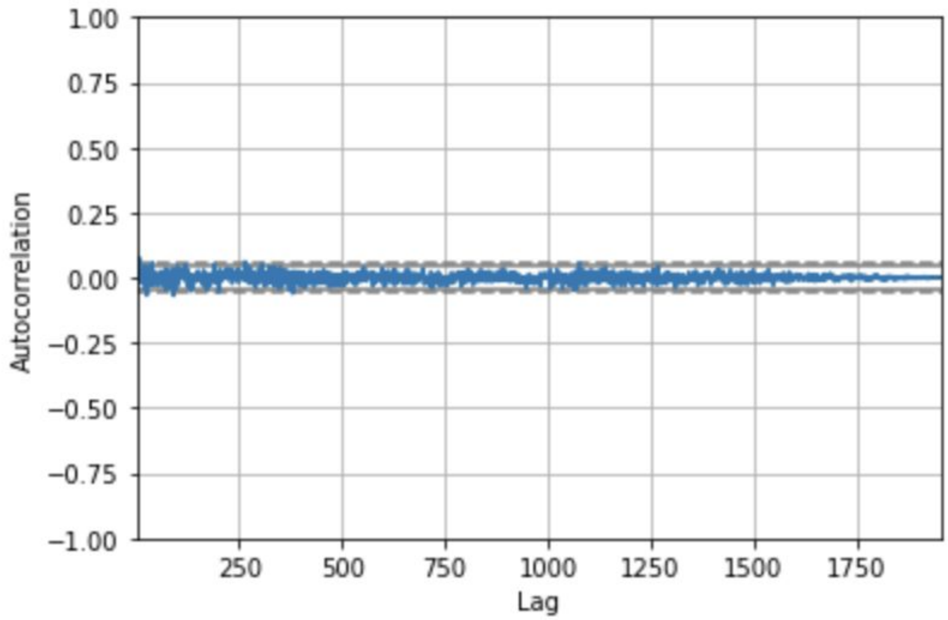

Gold
forecast_gold
Date
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
2010
2011
2012
2013
2014
2015
2016
2017
2018

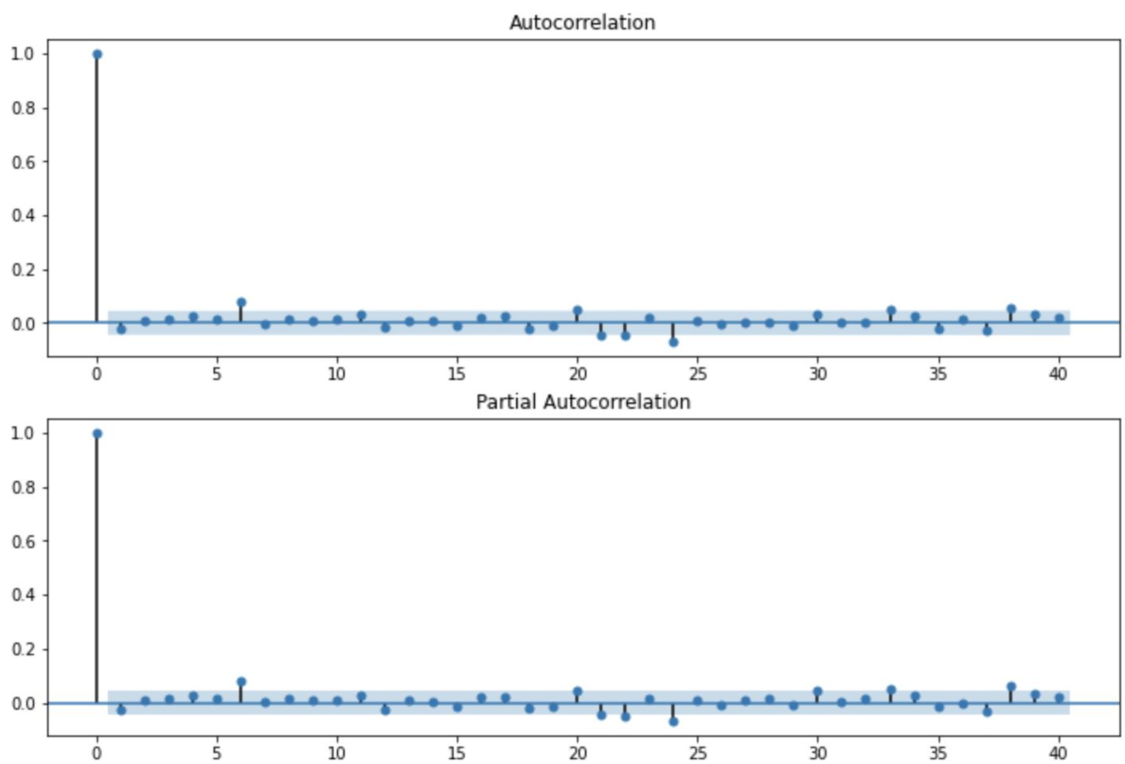
Oil:

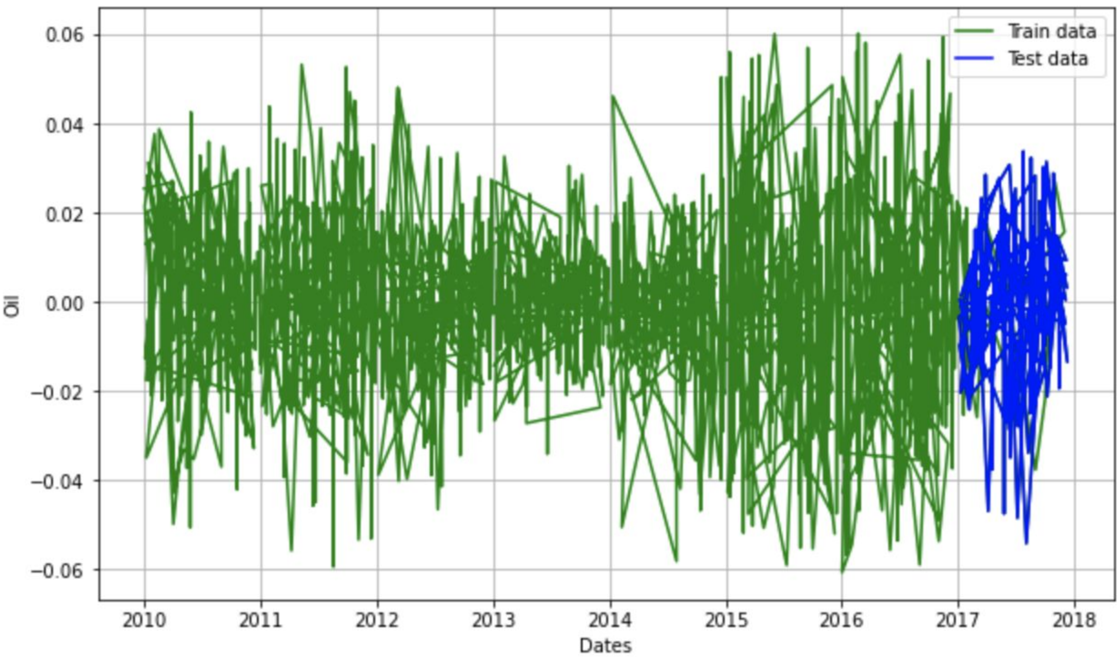



```
ADF Test Statistic : -16.079299916851483
p-value : 5.40018309042038e-29
#Lags Used : 5
Number of Observations Used : 1949
strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data  is stat
ionary
```




1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Autocorrelation
250
500
750
1000
1250
1500
1750
Lag


Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
40
Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
40

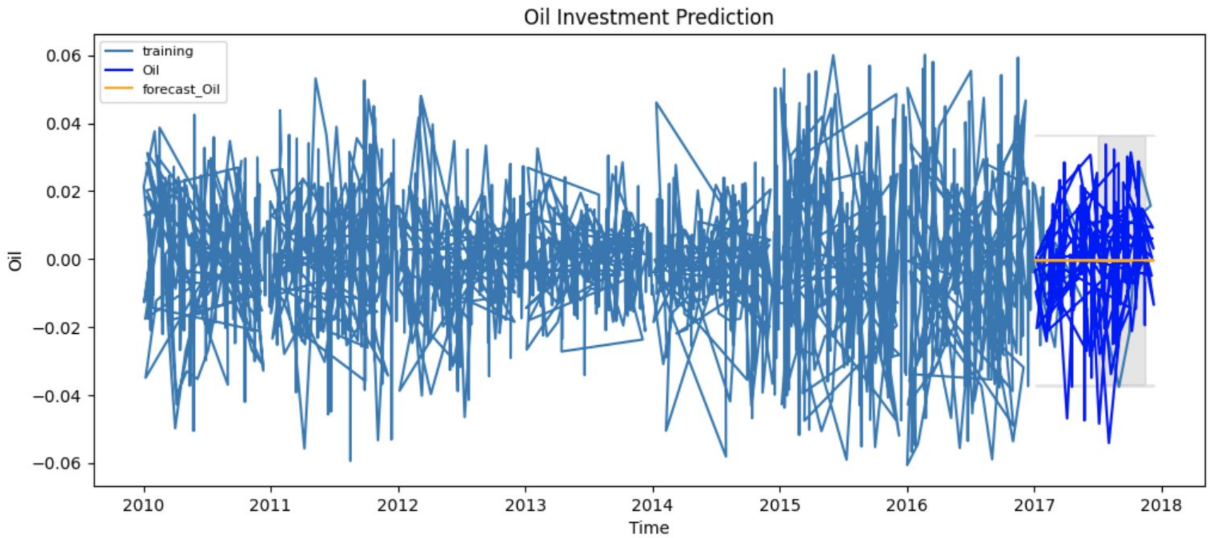



```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=-8918.414, Time=0.13 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=-8917.617, Time=0.11 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=-8917.618, Time=0.22 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=-8915.618, Time=0.24 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=-8917.449, Time=0.29 sec

Best model:  ARIMA(0,0,0)(0,0,0)[0]
Total fit time: 1.010 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1756
Model:                        SARIMAX   Log Likelihood                4460.207
Date:                Sun, 01 Nov 2020   AIC                          -8918.414
Time:                        09:26:32   BIC                          -8912.943
Sample:                             0   HQIC                         -8916.392
                               - 1756
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2         0.0004   1.06e-05     34.472      0.000       0.000       0.000
===================================================================================
Ljung-Box (Q):                       62.88   Jarque-Bera (JB):                36.87
Prob(Q):                              0.01   Prob(JB):                         0.00
Heteroskedasticity (H):               1.87   Skew:                            -0.05
Prob(H) (two-sided):                  0.00   Kurtosis:                         3.70
===================================================================================

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

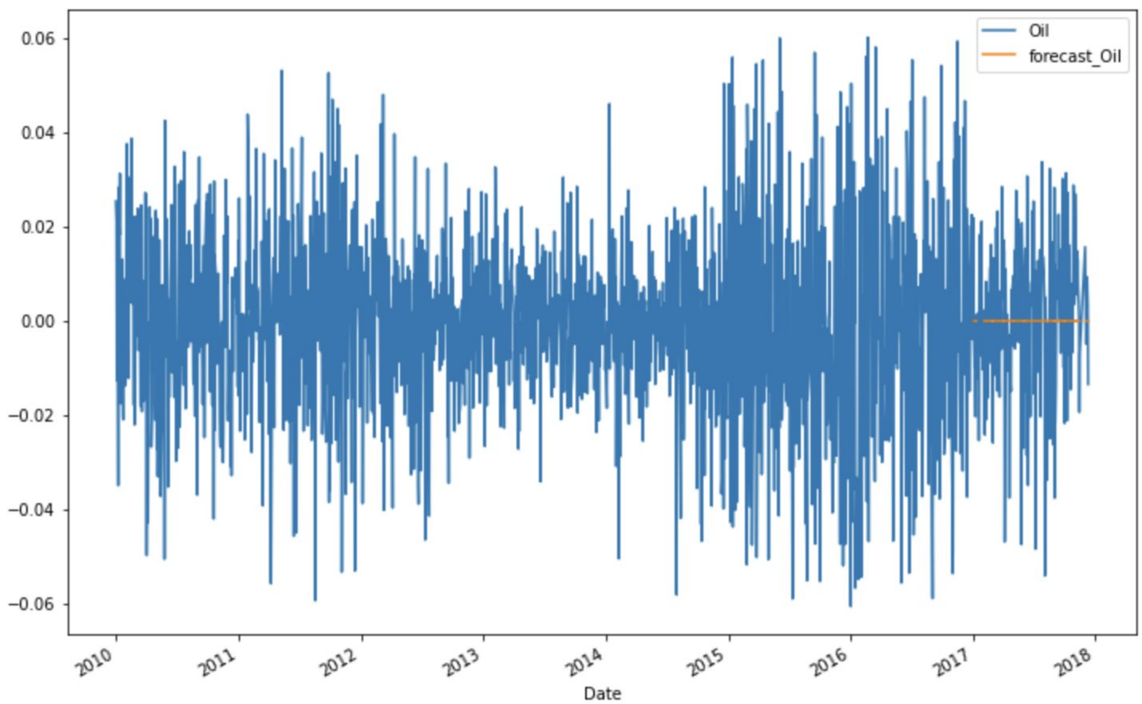
Dep. Variable:	Oil	**No. Observations:**	1955
Model:	ARMA(1, 1)	**Log Likelihood**	4998.066
Method:	css-mle	**S.D. of innovations**	0.019
Date:	Sun, 01 Nov 2020	**AIC**	-9988.131
Time:	09:26:33	**BIC**	-9965.819
Sample:	0	**HQIC**	-9979.929

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	-0.0003	0.000	-0.828	0.408	-0.001	0.000
ar.L1.Oil	-0.2007	0.875	-0.229	0.819	-1.916	1.515
ma.L1.Oil	0.1773	0.879	0.202	0.840	-1.546	1.900

	Real	**Imaginary**	**Modulus**	**Frequency**
AR.1	-4.9837	+0.0000j	4.9837	0.5000
MA.1	-5.6398	+0.0000j	5.6398	0.5000



Oil Investment Prediction
training
Oil
forecast_Oil
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
Oil
2010
2011
2012
2013
2014
2015
2016
2017
2018
Time



Oil
forecast_Oil
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
2010
2011
2012
2013
2014
2015
2016
2017
2018
Date

CONCLUSION

- Key Findings and Conclusions of the Study

Key findings:

a) Almost 50% of the data had negative value
b) Mean_squared_error, mean_absolute_error is low hence the accuracy is high for the project.
c) Auto correlation and Partial autocorrelation is mostly downwards in all the separation of the projects which is under company stock, gold and oil.
d) Gold Investment shows 80% accuracy where company stock and oil shows above 95% 0f accuracy by using ARIMA and SARIMAX model

- Learning Outcomes of the Study in respect of Data Science

1) Got a practice of Time Series data analysis.
2) Learning of the data or the trend which is mostly uneven but stationary.
3) The Biggest challenge was to predict or forecast the values of the company stock and investments as most of the future prediction was constant in between 0 and 1 .
4) Best algorithm for me in this project was :

```
import statsmodels.api as sm  # SARIMAX - seasonal arimax

model=sm.tsa.statespace.SARIMAX(df_gold['Gold'],order=(3, 0, 2))
#(p,d,q)

results=model.fit()

df_gold['forecast_gold']=results.predict(start=1781,end=1982,dynamic=
False)
df_gold[['Gold','forecast_gold']].plot(figsize=(12,8))
```

- Limitations of this work and Scope for Future Work
    1) Too real a data set to work on, Need more polished data which could be analysed. Most of the values are negative.
    2) The data which have been provided are mostly uneven hence it was very difficult to make accurate decisions.

thank you