In [1]:
```
!pip install pmdarima

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import pyplot
from pandas import DataFrame
import os

from statsmodels.tsa.stattools import adfuller
from pmdarima import auto_arima
from pandas.plotting import autocorrelation_plot

from sklearn.metrics import mean_squared_error
from math import sqrt

from statsmodels.tsa.arima.model import ARIMA
```

```
Collecting pmdarima
  Obtaining dependency information for pmdarima from https://files.pythonhosted.org/packages/ac/f8/6c9364602b13f0dba129b53acd1344859690911a4d5021560d9fd6aa087f/pmdarima-2.0.4-cp311-cp311-win_amd64.whl.metadata
  Downloading pmdarima-2.0.4-cp311-cp311-win_amd64.whl.metadata (8.0 kB)
Requirement already satisfied: joblib>=0.11 in c:\users\hfwal\anaconda3\lib\site-packages (from pmdarima) (1.2.0)
Collecting Cython!=0.29.18,!=0.29.31,>=0.29 (from pmdarima)
  Obtaining dependency information for Cython!=0.29.18,!=0.29.31,>=0.29 from https://files.pythonhosted.org/packages/18/ec/f47a721071d084d6c2b6783eb8d058b964b1450cb708d920d0d792f42001/Cython-3.0.10-cp311-cp311-win_amd64.whl.metadata
  Using cached Cython-3.0.10-cp311-cp311-win_amd64.whl.metadata (3.2 kB)
Requirement already satisfied: numpy>=1.21.2 in c:\users\hfwal\anaconda3\lib\site-packages (from pmdarima) (1.24.4)
Requirement already satisfied: pandas>=0.19 in c:\users\hfwal\anaconda3\lib\site-packages (from pmdarima) (1.5.2)
Requirement already satisfied: scikit-learn>=0.22 in c:\users\hfwal\anaconda3\lib\site-packages (from pmdarima) (1.4.1.post1)
Requirement already satisfied: scipy>=1.3.2 in c:\users\hfwal\anaconda3\lib\site-packages (from pmdarima) (1.10.0)
Requirement already satisfied: statsmodels>=0.13.2 in c:\users\hfwal\anaconda3\lib\site-packages (from pmdarima) (0.14.0)
Requirement already satisfied: urllib3 in c:\users\hfwal\anaconda3\lib\site-packages (from pmdarima) (1.26.16)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\users\hfwal\anaconda3\lib\site-packages (from pmdarima) (68.0.0)
Requirement already satisfied: packaging>=17.1 in c:\users\hfwal\anaconda3\lib\site-packages (from pmdarima) (23.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\hfwal\anaconda3\lib\site-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\hfwal\anaconda3\lib\site-packages (from pandas>=0.19->pmdarima) (2023.3.post1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\hfwal\anaconda3\lib\site-packages (from scikit-learn>=0.22->pmdarima) (2.2.0)
Requirement already satisfied: patsy>=0.5.2 in c:\users\hfwal\anaconda3\lib\site-packages (from statsmodels>=0.13.2->pmdarima) (0.5.3)
Requirement already satisfied: six in c:\users\hfwal\anaconda3\lib\site-packages (from patsy>=0.5.2->statsmodels>=0.13.2->pmdarima) (1.16.0)
Downloading pmdarima-2.0.4-cp311-cp311-win_amd64.whl (614 kB)
   ---------------------------------------- 0.0/614.7 kB ? eta -:--:--
   ---------------------------------------- 0.0/614.7 kB ? eta -:--:--
    --------------------------------------- 10.2/614.7 kB ? eta -:--:--
   - -------------------------------------- 30.7/614.7 kB 325.1 kB/s eta 0:00:02
   ------ --------------------------------- 112.6/614.7 kB 819.2 kB/s eta 0:00:01
   ---------------------- ----------------- 317.4/614.7 kB 1.8 MB/s eta 0:00:01
   ---------------------------------- ----- 532.5/614.7 kB 2.4 MB/s eta 0:00:01
   ---------------------------------------- 614.7/614.7 kB 2.4 MB/s eta 0:00:00
Using cached Cython-3.0.10-cp311-cp311-win_amd64.whl (2.8 MB)
Installing collected packages: Cython, pmdarima
Successfully installed Cython-3.0.10 pmdarima-2.0.4
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
tables 3.8.0 requires blosc2~=2.0.0, which is not installed.
```

In [57]:

```
dataa = pd.read_csv(r"C:\Users\hfwal\Downloads\archive (10)\shampoo_sales.csv")
dataa.head(n=5)
```

Out[57]:

	Month	Sales
0	1-01	266.0
1	1-02	145.9
2	1-03	183.1
3	1-04	119.3
4	1-05	180.3

In [58]:
```
dataa['Month'].iloc[2:]
```

Out[58]:
```
2     1-03
3     1-04
4     1-05
5     1-06
6     1-07
7     1-08
8     1-09
9     1-10
10    1-11
11    1-12
12    2-01
13    2-02
14    2-03
15    2-04
16    2-05
17    2-06
18    2-07
19    2-08
20    2-09
21    2-10
22    2-11
23    2-12
24    3-01
25    3-02
26    3-03
27    3-04
28    3-05
29    3-06
30    3-07
31    3-08
32    3-09
33    3-10
34    3-11
35    3-12
Name: Month, dtype: object
```

In [59]:
```
#Editing the data in the month column
dataa['Temp_date'] = 0
#sequencing in a way such that appropriate dates are incoporated
initial_year = 2019
counter = 0
for indx,row in dataa.iterrows():
    if counter < 12 :
        year = initial_year
    elif counter >=12 and counter <24:
        year = initial_year + 1
    elif counter >=24:
        year = initial_year + 2
    dataa['Temp_date'].iloc[indx] = str(year) + "-" + dataa['Month'].iloc[indx][2:]
    counter += 1
```

```
dataa['Month'] = dataa['Temp_date']
dataa.drop(columns = 'Temp_date', inplace = True)
dataa['Month'] = dataa['Month'] + "-" + "01"
dataa = dataa[0:36]
```

```
C:\Users\hfwal\AppData\Local\Temp\ipykernel_7956\1973257320.py:13: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  dataa['Temp_date'].iloc[indx] = str(year) + "-" + dataa['Month'].iloc[indx][2:]
```

In [60]:
```
dataa.head()
```

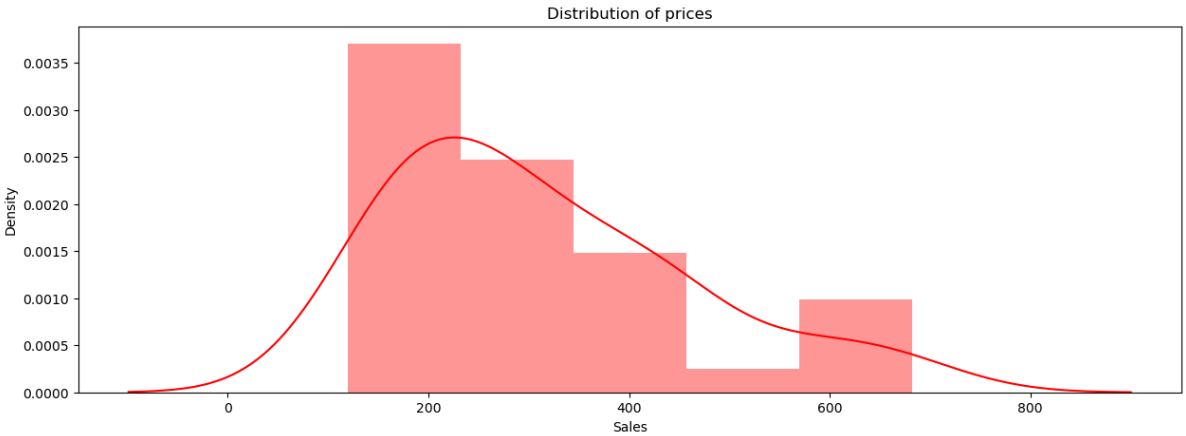
Out[60]:

	Month	Sales
0	2019-01-01	266.0
1	2019-02-01	145.9
2	2019-03-01	183.1
3	2019-04-01	119.3
4	2019-05-01	180.3

In [61]:
```
dataa = dataa.set_index(['Month'])
dataa = dataa.rename({'Sale of shampoo over 3 years':'Sales'}, axis = 1)
```

In [62]:
```
plt.figure(figsize =(15,5))
plt.title("Distribution of prices")
ax = sns.distplot(dataa['Sales'],color = 'red')
```

```
C:\Users\hfwal\AppData\Local\Temp\ipykernel_7956\1687281287.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax = sns.distplot(dataa['Sales'],color = 'red')
```



Statistical Test

In [63]:
```
#Regression Analysis (Augmented Dickey-Fuller test to check if data points are stat
def reg_test(df) :
    dataa_test = adfuller(df, autolag = 'AIC')

    print("1) ADF :", dataa_test[0])
    print("2) P_Value : ",dataa_test[1])
    print("3) Number of Lags : ",dataa_test[2])
    print("4) Number of Observations used for Regression : ",dataa_test[3])
    print("5) Critical Values : ")
    for key , val in dataa_test[4].items():
          print("\t", key,": ",val)



reg_test(dataa['Sales'])
#Output : The result is that data is not stationery.
```

```
1) ADF : 3.0601420836411797
2) P_Value :  1.0
3) Number of Lags :  10
4) Number of Observations used for Regression :  25
5) Critical Values :
         1% :  -3.7238633119999998
         5% :  -2.98648896
         10% :  -2.6328004
```

Fitting the audto arima function

In [64]:
```
fitt = auto_arima(dataa['Sales'],trace=True,suppress_warnings = True)
#ARIMA parameters given to extract the best score out of the model : (1,1,2)
```

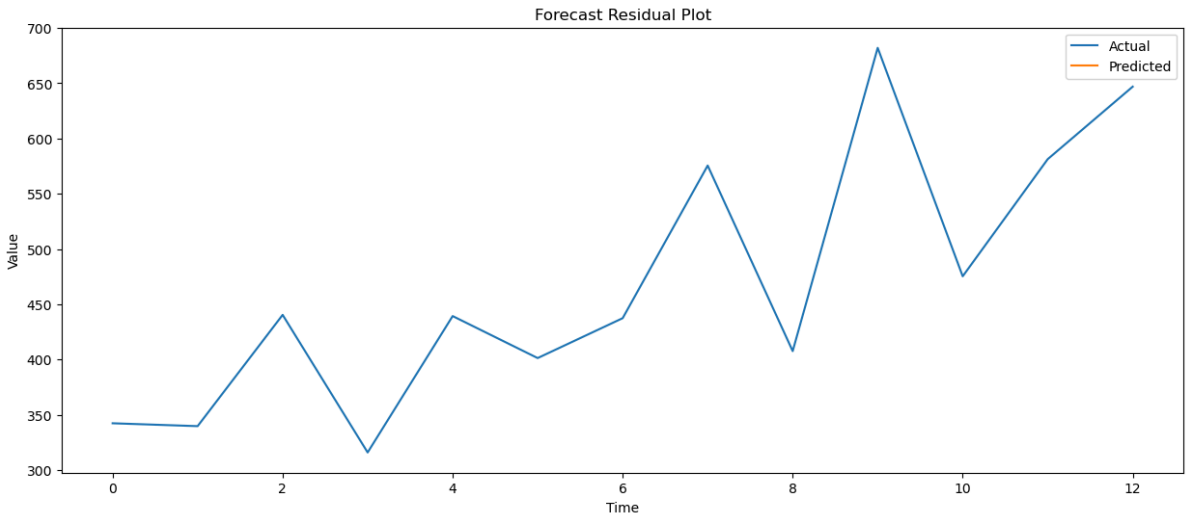
```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.30 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=430.873, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=405.977, Time=0.09 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=408.967, Time=0.12 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=429.229, Time=0.04 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=403.573, Time=0.10 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=404.633, Time=0.16 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=404.226, Time=0.18 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=402.407, Time=0.15 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=401.463, Time=0.31 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.13 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=inf, Time=0.44 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AIC=inf, Time=0.15 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=inf, Time=0.42 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=inf, Time=0.13 sec

Best model:  ARIMA(1,1,2)(0,0,0)[0] intercept
Total fit time: 2.743 seconds
```

In [65]:
```
#Seperating the test and train data
X = dataa.values
size = int(len(X)*0.66)
train, test = X[0:size],X[size:len(X)]
history = [x for x in train]
predictions = list()
```

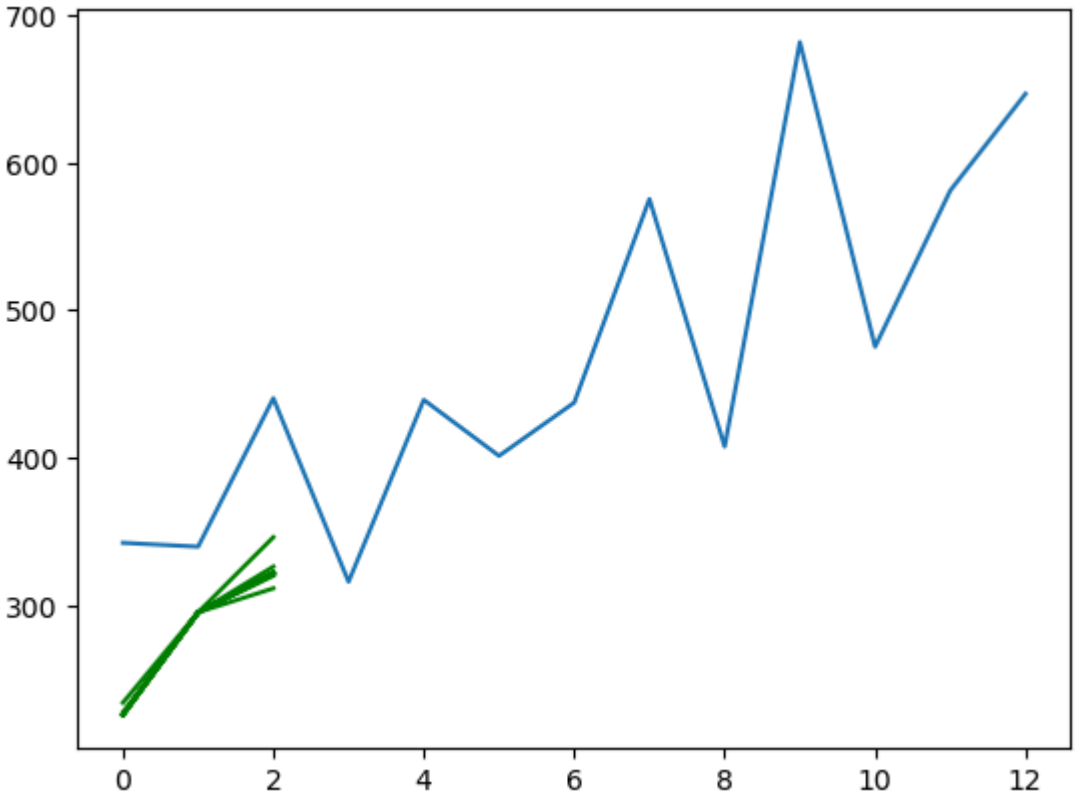
In [98]:
```
#Fitting the ARIMA Model :
for s in range (len(test)):
    model = ARIMA(history,order = (1,1,2))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[s]
    history.append(obs)
    difference = yhat - obs

    print("predicted=%f,expected =%f,difference=%f" %(yhat,obs,difference))
```

```
predicted=578.365680,expected =342.300000,difference=236.065680
predicted=567.494268,expected =339.700000,difference=227.794268
predicted=379.530246,expected =440.400000,difference=-60.869754
predicted=317.097673,expected =315.900000,difference=1.197673
predicted=350.672826,expected =439.300000,difference=-88.627174
predicted=372.114670,expected =401.300000,difference=-29.185330
predicted=418.335346,expected =437.400000,difference=-19.064654
predicted=430.674462,expected =575.500000,difference=-144.825538
predicted=480.185808,expected =407.600000,difference=72.585808
predicted=523.737410,expected =682.000000,difference=-158.262590
predicted=522.780225,expected =475.300000,difference=47.480225
predicted=593.484322,expected =581.300000,difference=12.184322
predicted=552.635002,expected =646.900000,difference=-94.264998
```

In [99]:
```
# Ploting the residuals
plt.figure(figsize=(15, 6))
plt.plot(test, label="Actual")
plt.plot(output, label="Predicted")
plt.title("Forecast Residual Plot")
plt.xlabel("Time")
plt.ylabel("Value")
plt.legend()
plt.show()
```



In [144…
```
#Finding difference  between actual and forecasted values :
pyplot.plot(test)
pyplot.plot(predictions,color = 'green')
pyplot.show()       #Green is the estimated line(future)
```



Experimenting after changing ARIMA paramters

In [102…

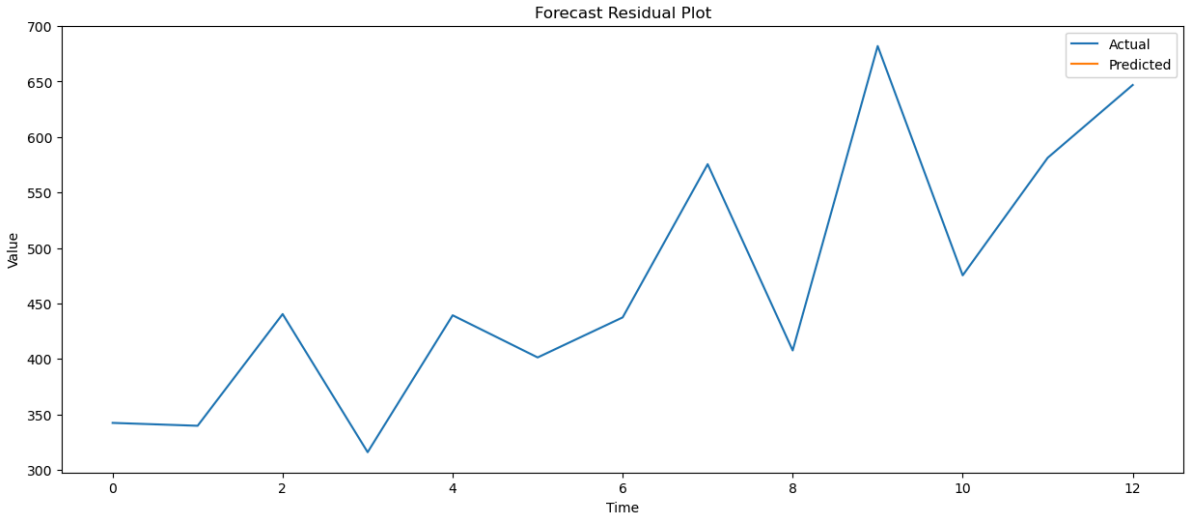
```
#Fitting the ARIMA Model :
for s in range (len(test)):
    model = ARIMA(history,order = (0,1,2))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[s]
    history.append(obs)
    difference = yhat - obs

    print("predicted=%f,expected =%f,difference=%f" %(yhat,obs,difference))
```

```
predicted=582.327627,expected =342.300000,difference=240.027627
predicted=553.936035,expected =339.700000,difference=214.236035
predicted=380.265994,expected =440.400000,difference=-60.134006
predicted=312.015308,expected =315.900000,difference=-3.884692
predicted=337.030950,expected =439.300000,difference=-102.269050
predicted=369.712696,expected =401.300000,difference=-31.587304
predicted=421.351661,expected =437.400000,difference=-16.048339
predicted=439.356541,expected =575.500000,difference=-136.143459
predicted=487.852273,expected =407.600000,difference=80.252273
predicted=521.086710,expected =682.000000,difference=-160.913290
predicted=534.409210,expected =475.300000,difference=59.109210
predicted=587.307618,expected =581.300000,difference=6.007618
predicted=559.648638,expected =646.900000,difference=-87.251362
```

In [103…

```
# Ploting the residuals
plt.figure(figsize=(15, 6))
plt.plot(test, label="Actual")
plt.plot(output, label="Predicted")
plt.title("Forecast Residual Plot")
```

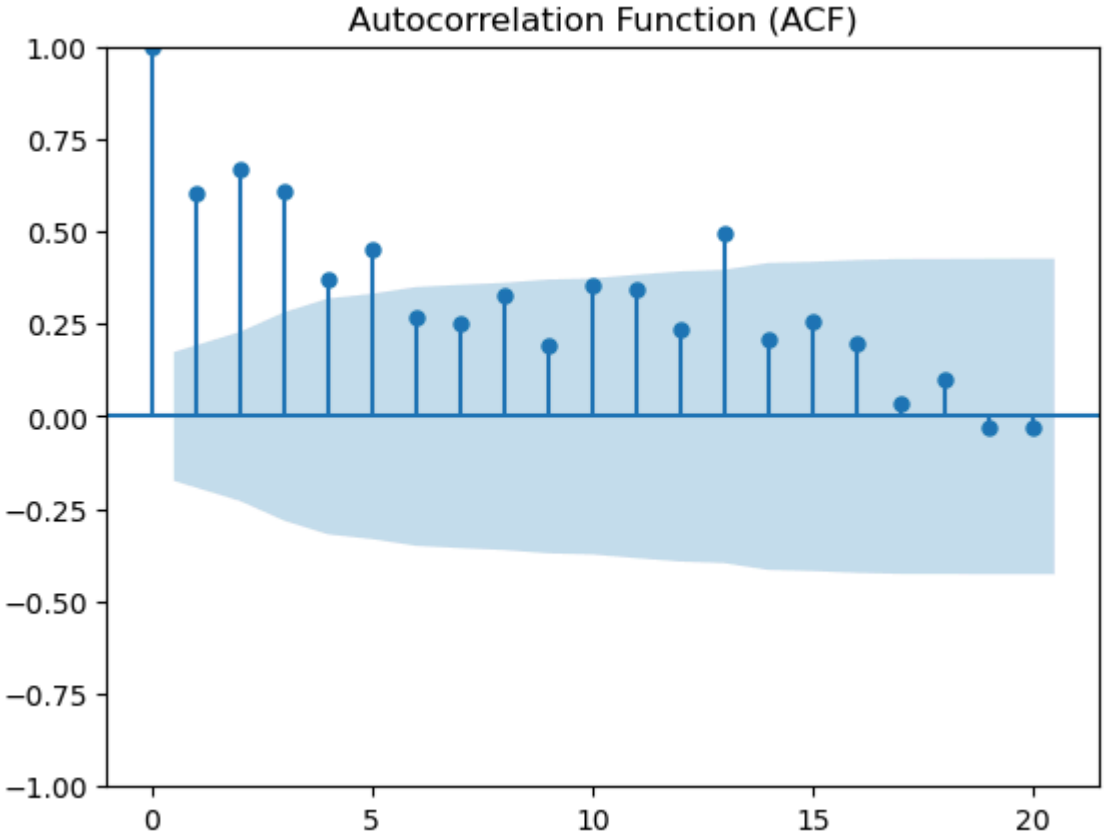
```
plt.xlabel("Time")
plt.ylabel("Value")
plt.legend()
plt.show()
```



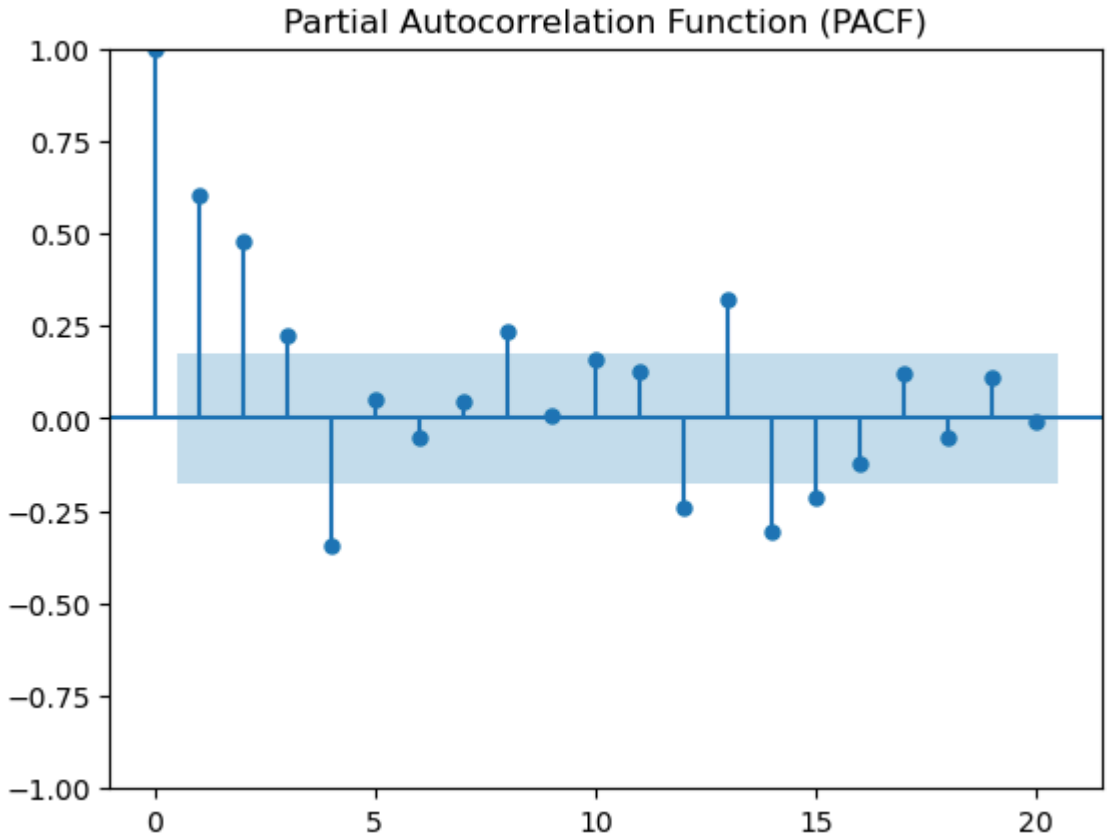
In [104…

```
# Plot the autocorrelation function (ACF) and partial autocorrelation function (PAC
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plt.figure(figsize=(12, 6))
plot_acf(history, lags=20)
plt.title("Autocorrelation Function (ACF)")
plt.show()

plt.figure(figsize=(12, 6))
plot_pacf(history, lags=20)
plt.title("Partial Autocorrelation Function (PACF)")
plt.show()
#The point where the bars/lines are exceeding the confidence interval is where the
```

```
<Figure size 1200x600 with 0 Axes>
```




```
<Figure size 1200x600 with 0 Axes>
```



SEASONAL ARIMA MODEL

In [147…
```
#Seperating the test and train data
X = dataa.values
size = int(len(X)*0.66)
train, test = X[0:size],X[size:len(X)]
history = [x for x in train]
predictions = list()
```

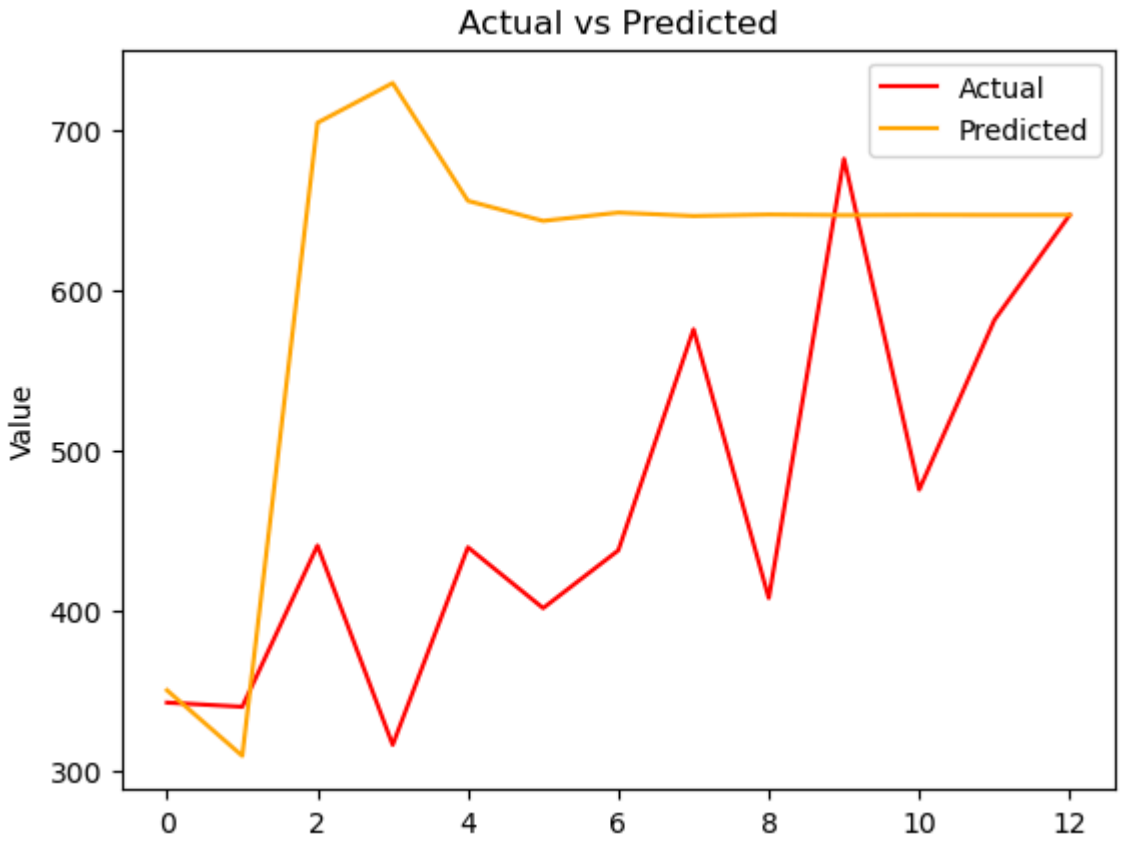
In [148…
```
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Fitting the Seasonal ARIMA model
for h in range (len(test)):
    model = SARIMAX(history,order = (1,1,2))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[s]
    history.append(obs)
    difference = yhat - obs


    print("predicted=%f,expected =%f,difference=%f" %(yhat,obs,difference))
```

```
predicted=349.970807,expected =646.900000,difference=-296.929193
predicted=309.041646,expected =646.900000,difference=-337.858354
C:\Users\hfwal\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:966: UserWarning: Non-stationary starting autoregressive parameters found. Using zeros as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
```

```
predicted=704.489308,expected =646.900000,difference=57.589308
predicted=729.296351,expected =646.900000,difference=82.396351
predicted=655.725780,expected =646.900000,difference=8.825780
predicted=643.192526,expected =646.900000,difference=-3.707474
predicted=648.375981,expected =646.900000,difference=1.475981
predicted=646.259620,expected =646.900000,difference=-0.640380
predicted=647.175301,expected =646.900000,difference=0.275301
predicted=646.780223,expected =646.900000,difference=-0.119777
predicted=646.952040,expected =646.900000,difference=0.052040
predicted=646.877308,expected =646.900000,difference=-0.022692
predicted=646.909893,expected =646.900000,difference=0.009893
```

In [149…

```
# Plotting the actual and predicted values
plt.plot(test, label="Actual", color='red')
plt.plot(predictions, label="Predicted", color='orange')
plt.title("Actual vs Predicted")
plt.ylabel("Value")
plt.legend()


plt.show()
```



In [150…

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(test, predictions)
print("Mean Squared Error (MSE):", mse)
```

```
Mean Squared Error (MSE): 37608.45739496699
```

In [151…

```
from sklearn.ensemble import RandomForestRegressor
import numpy as np

# Train multiple ARIMA models with different configurations
models = []
predictions = []

for order in [(1, 0, 0), (0, 1, 1), (1, 1, 1)]:
```

```
    model = ARIMA(train, order=order)
    model_fit = model.fit()
    models.append(model_fit)
    predictions.append(model_fit.forecast(steps=len(test)))

# Stack predictions and train meta-model
stacked__X = np.vstack(predictions).T
meta_model = RandomForestRegressor(n_estimators=80)  # Random Forest with 80 trees
meta_model.fit(stacked__X, test)

# Make predictions using the meta-model
ensemble__prediction = meta_model.predict(stacked__X)
```

```
C:\Users\hfwal\anaconda3\Lib\site-packages\sklearn\base.py:1474: DataConversionWar
ning: A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

#Plot actual values and ensemble prediction plt.figure(figsize=(10, 6)) plt.plot(test, label='Actual', color='blue') plt.plot(ensemble__prediction, label='Ensemble Prediction', color='orange') plt.title('Actual vs Ensemble Prediction') plt.xlabel('Time') plt.ylabel('Value') plt.legend() plt.show()

In [153…

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(test, ensemble_prediction)
print("Mean Squared Error (MSE):", mse)
```

```
Mean Squared Error (MSE): 1194.9305389423057
```

INTERPRETING THE MODEL

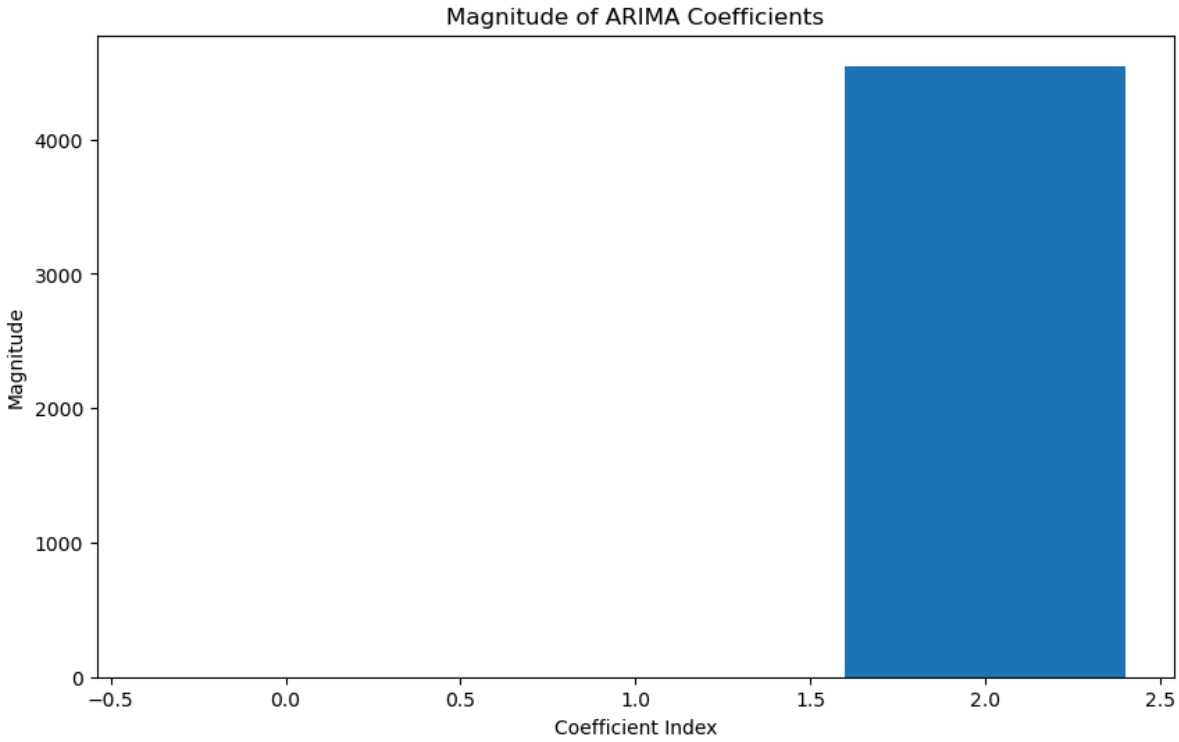
In [155…

```
# Fit ARIMA model
model = ARIMA(train, order=(2, 1, 0))
model_fit = model.fit()

# Analyze coefficients
print("ARIMA Coefficients:")
print(model_fit.params)
```

```
ARIMA Coefficients:
[-7.79829814e-01 -1.23247583e-01  4.54725494e+03]
```

In [156…

```
# Plotting the magnitudes of the estimated ARIMA coefficients to find out more abou
plt.figure(figsize=(10, 6))
plt.bar(range(len(model_fit.params)), np.abs(model_fit.params))
plt.title('Magnitude of ARIMA Coefficients')
plt.xlabel('Coefficient Index')
plt.ylabel('Magnitude')
plt.show()
```


Magnitude of ARIMA Coefficients
Magnitude
Coefficient Index
0
1000
2000
3000
4000
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

In [ ]: