DEPARTMENT OF MATHEMATICS AND STATISTICS , IIT KANPUR

Time Series Analysis And forecasting of Gold prices in Mumbai

Submitted by :
Arvind Singh Yadav (191026)
Sahil Saini (191118)
Deepak Kumar Bind (191037)
Shivraj Meghwal (191136)
Rupvant Dayanand Waghmare (191116)

Submitted To : Prof. Amit Mitra

November 29, 2020

Contents

1 INTRODUCTION

A time series is a series of data points indexed in time with some order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data. Time series analysis is used in different mathematical and economical fields. It can also be helpful in building early waring predictions system for future adverse events.

Our variable of interest lies in time series analysis of Gold prices in Rupees. The price is of 10 grams of gold in Mumbai.

The data can be viewed by the below given link from official RBI datasets.

https://dbie.rbi.org.in/DBIE/dbie.rbi?site=statistics

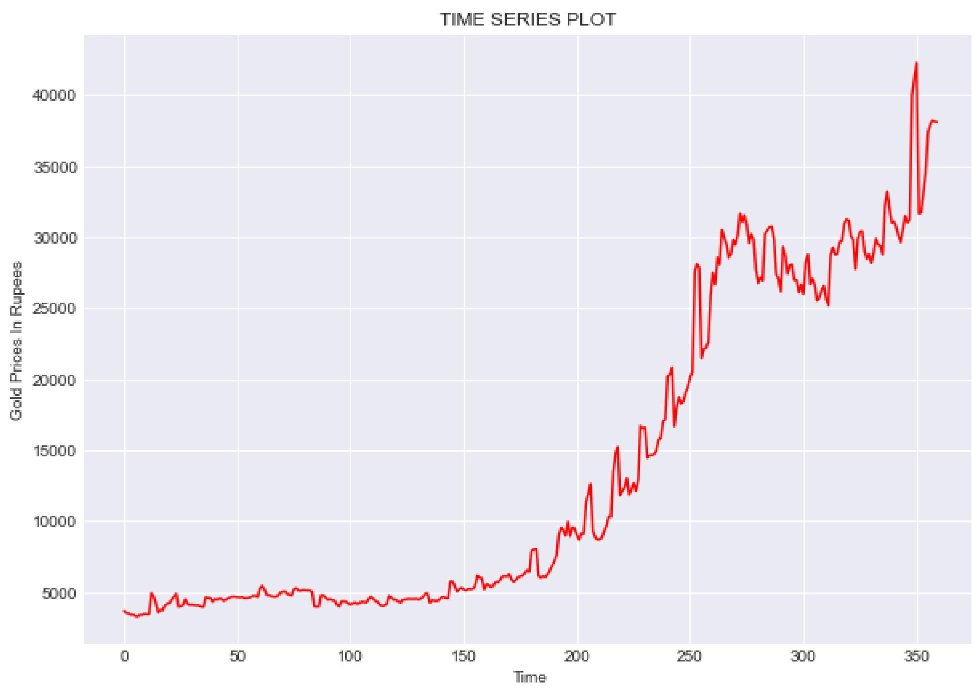
In this project we have done the following things :
1.)Detection of presence of deterministic components and removing them.
2.)Find out which model fits the data and estimate it's parameter.
3.)Forecast monthly Price of gold.

2 INFORMATION REGARDING THE DATA-SET

Our data-set consists of monthly price of gold collected from real life data .Data is from 1990(January) to 2019(October) total of 358 data points.
For forecasting purposes the data is divided in two parts i.e in testing and training . We have taken 352 points in training and remaining 6 for testing .
Analysis will be done on training set.

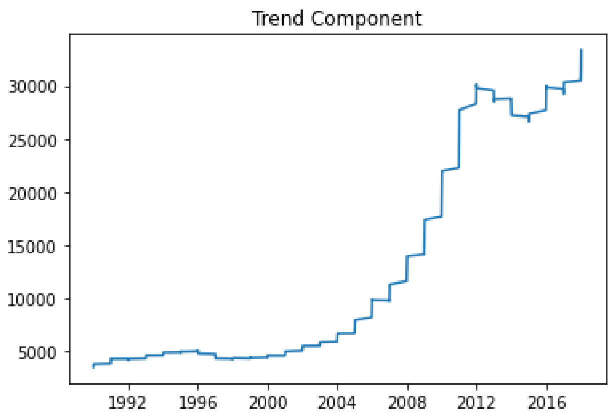
Mathematically Y_t is price of Gold for training set in which t ranges from 1 to 352.

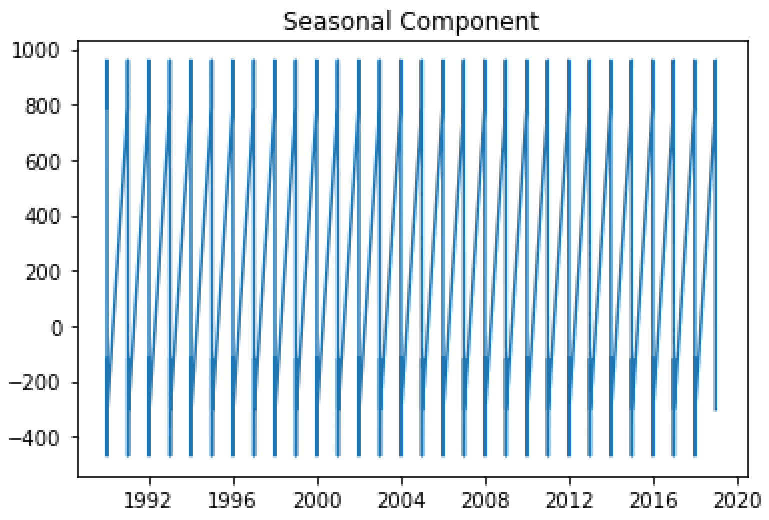
Data can be visualised as given plot below:

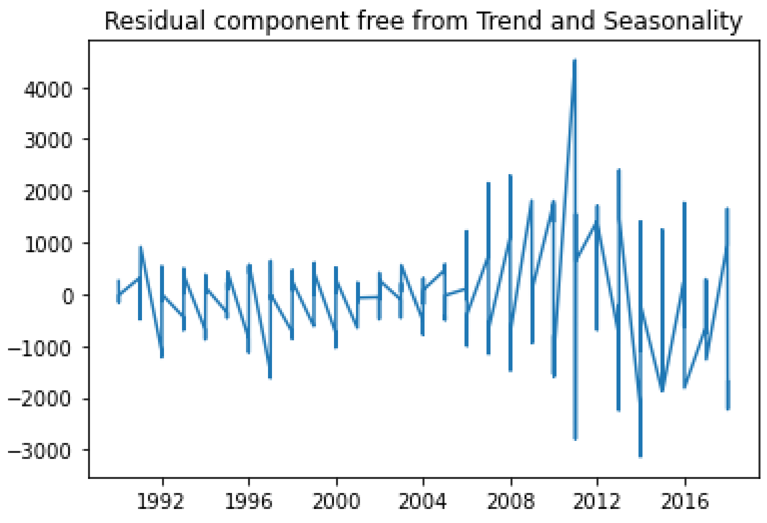


From the visual inspection it is clear that the mean and variance of the data had not been the constant throughout the time points.

Also there is evidence of trend here, because the graph has been rising upward with respect to time.
We can also visualize our data using a method called time- series decomposition that allows us to decompose our time series into three distinct component :trend,seasonality,and noise. each components given below:




Seasonal Component
1000
800
600
400
200
0
−200
−400
1992
1996
2000
2004
2008
2012
2016
2020


Residual component free from Trend and Seasonality
4000
3000
2000
1000
0
−1000
−2000
−3000
1992
1996
2000
2004
2008
2012
2016

3 Checking Randomness of Data

Firstly, we will check whether our data is random or not. For this we have used **'Turning Point Test'**.

- Hypothesis :

  H_0: Series is purely random.

  against

  H_1:Series is not random.

  Y_i is a turning point if $Y_i > Y_{i-1}$ and $Y_i > Y_{i+1}$ or $Y_i < Y_{i-1}$ and $Y_i < Y_{i+1}$. In this case, turning point is answer.

$$T = \sum_{i=2}^{n-1} T_i$$

- Test Statistic : Under H_0 ,

$$Z = \frac{T - E(T)}{V(T)} \sim N(0,1)$$

; where

$$E(T) = \frac{2(n-2)}{3}$$

and

$$V(T) = \frac{16n-29}{90}$$

- Decision criterion:

  Reject H_0 at α level of significance(l.o.s) if obs $|Z| > Z_{\alpha/2}$

- Conclusion :

$$|Z| = 9.041 > 1.989$$

and hence, we reject our null hypothesis. Thus, our data is not purely random and there is some trend in the model

4 Checking for the presence of deterministic components

1. Testing of presence of Trend in the model:

   We have to check if there is a trend in the model or not. **Relative Ordering test** is conducted.

   - Hypothesis :

     H_0: Trend is absent in the model

     against

     H_1: Trend is present in the model.

     R: Number of discordant pairs

     $$E(R) = \frac{n(n-1)}{4}$$

     If R>E(R): indication of falling trend
     R<E(R): indication of rising trend.
     R is related with Kendell's T, the rank correlation coefficient.

     $$T = \frac{1-4R}{n(n-1)}$$

     under H_0 ,

     $$E(T) = 0$$

     and

     $$V(T) = \frac{2(2n+5)}{9n(n-1)}$$

   - Test Statistic : Under H_0 ,

     $$Z = \frac{T - E(T)}{V(T)} \sim N(0,1)$$

   - Decision criterion:

     Reject H_0 at α level of significance(l.o.s) if obs $|Z| > Z_{\alpha/2}$

- Conclusion :

  There is rising trend in our model.

  $$|Z| = 22.3106 > 1.96(Z_{0.025}),$$

  and hence we reject our null hypothesis. We conclude that there is presence of trend in the model.

2. Elimination of trend and seasonality :

   - **Differencing** :

     From above Relative Ordering Test and Friedman's non parametric test we can see that there is a presence of trend and sesonality components in time series model.

     Now we will remove trend and seasonality components using **Differencing**.

     This is a method of elimination of trend and seasonality without estimating the trend and seasonality components.

     We have additive model as:

     $$Y_t = m_t + s_t + e_t$$

     where,
     Y_t: time series
     m_t: trend component
     s_t: seasonality component
     e_t: irregular component
     ∇=first difference operator

     $$\nabla Y_t = Y_t - Y_{t-1}$$

     Here we did first order differencing because as we go on increasing order of differencing it reduces precision .

3. Checking for Randomness of Data(Differenced Data) :

   Firstly, we will check whether our data is random or not. For this we have used **'Turning Point Test'**.
   For this we are using differenced data.

   - Hypothesis :

     $$H_0\text{: Series is purely random.}$$

     against

     $$H_1\text{:Series is not random.}$$

     Y_i is a turning point if $Y_i > Y_{i-1}$ and $Y_i > Y_{i+1}$ or $Y_i < Y_{i-1}$ and $Y_i < Y_{i+1}$. In this case, turning point is answer.

     $$T = \sum_{i=2}^{n-1} T_i$$

   - Test Statistic : Under H_0 ,

     $$Z = \frac{T - E(T)}{V(T)} \sim N(0,1)$$

     ; where

     $$E(T) = \frac{2(n-2)}{3}$$

     and

     $$V(T) = \frac{16n-29}{90}$$

   - Decision criterion:

     Reject H_0 at α level of significance(l.o.s) if obs $|Z| > Z_{\alpha/2}$

   - Conclusion :

     $$|Z| = 1.988 < 1.989$$

     and hence, we accept our null hypothesis. Thus, our data is purely random and there is no trend in the model

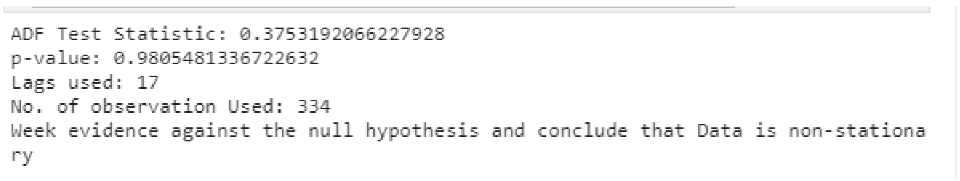
4. Checking for Stationarity :

- **Augmented Dickey-Fuller (ADF) Test :**

  The Augmented Dickey-Fuller test allows for higher-order autoregressive processes by including ΔY_{t-p} in the model. But our test is still if $\Delta\gamma = 0$

$$\Delta Y_t = \alpha + \beta t + \gamma Y_{t-1} + \delta_1 \Delta Y_{t-1} + \delta_2 \Delta Y_{t-1} + ...$$

  The null hypothesis for both tests is that the data are non-stationary. We want to REJECT the null hypothesis for this test, so we want a p-value of less that 0.05 (or smaller).

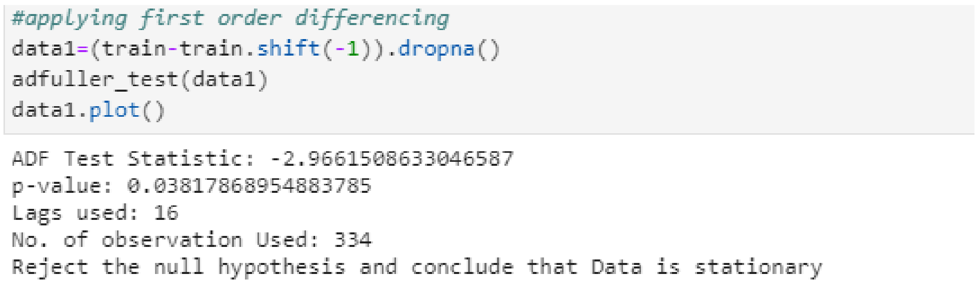


```
ADF Test Statistic: 0.3753192066227928
p-value: 0.9805481336722632
Lags used: 17
No. of observation Used: 334
Week evidence against the null hypothesis and conclude that Data is non-stationa
ry
```

  From above test it is clear that data are not stationary. Now we will go for first order difference .

$$\nabla Y_t = Y_t - Y_{t-1}$$

  now we will apply the adf test for the differenced data.



```
#applying first order differencing
data1=(train-train.shift(-1)).dropna()
adfuller_test(data1)
data1.plot()
```

```
ADF Test Statistic: -2.9661508633046587
p-value: 0.03817868954883785
Lags used: 16
No. of observation Used: 334
Reject the null hypothesis and conclude that Data is stationary
```

  from above table observe that p value has found to be less than the 0.05 hence data is now stationary.

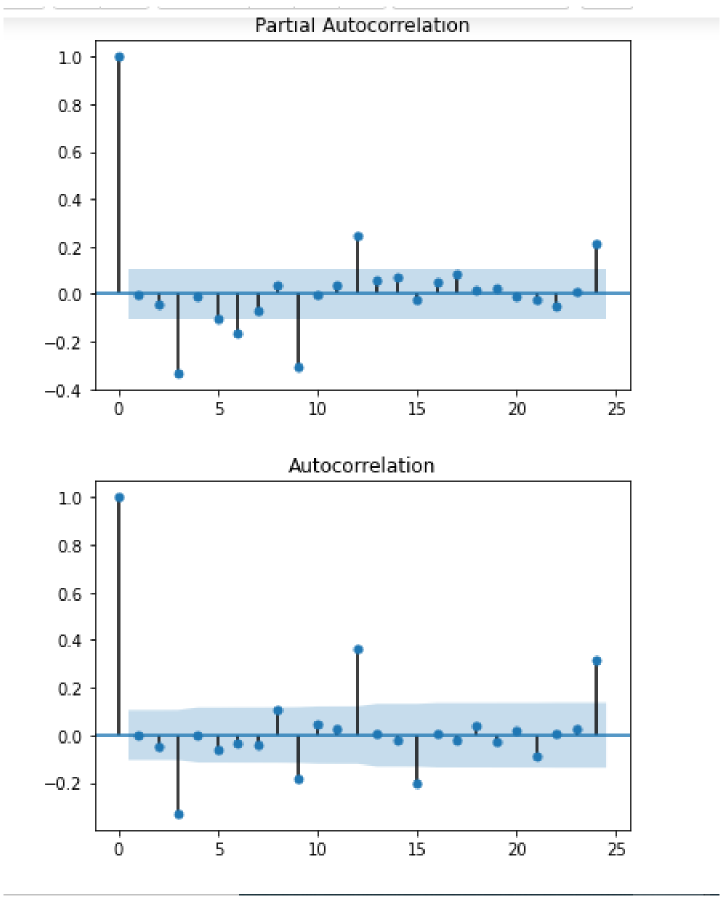
  Now we will look at the acf and pacf plots of the first order difference of data Y_t

5 Model Identification

Now, we proceed to check the model of the data to which it belong.

Firstly, we shall try to implement Auto Regressive Moving Average (ARMA) model and observe ACF and PACF plots for the model identification purpose. Then we shall estimate the model parameters through the criterion of minimum Akaike Information Criterion (AIC).

- ACF and PACF plots :

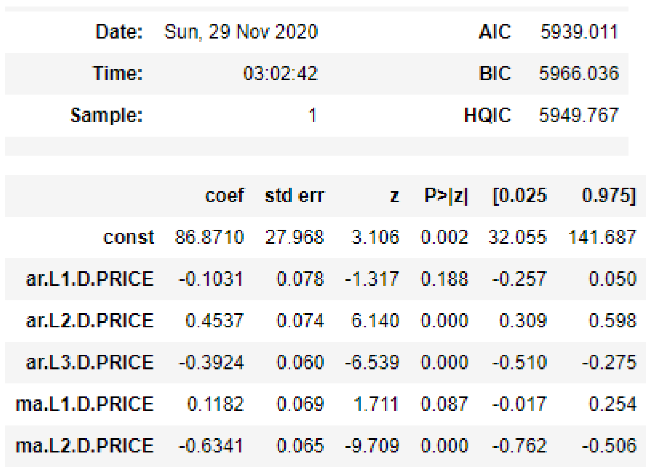


We can see that both the ACF and PACF of the first differencing doesn't cuts off this implies that both Moving Average(MA) and Auto-Regressive(AR) parameters are present ,so our model is ARIMA(p,d,q).

Now we will try to minimize AIC for different values of p q starting from 1 because we have d=1 from ADF.

- **ESTIMATION OF ARIMA MODEL PARAMETERS:**

Since it is evident from acf and pacf plots of first differenced data that model is ARIMA so we will check the AIC of each possible ARIMA(p,d,q) with p,d,q starting from 0, and the model with minimum AIC will be chosen.
It is found that the ARIMA(3,1,2) has minimum AIC among all the possible ARIMA models.
Summary of the above model is given below:



Date:	Sun, 29 Nov 2020	AIC	5939.011
Time:	03:02:42	BIC	5966.036
Sample:	1	HQIC	5949.767

	coef	std err	z	P>|z|	[0.025	0.975]
const	86.8710	27.968	3.106	0.002	32.055	141.687
ar.L1.D.PRICE	-0.1031	0.078	-1.317	0.188	-0.257	0.050
ar.L2.D.PRICE	0.4537	0.074	6.140	0.000	0.309	0.598
ar.L3.D.PRICE	-0.3924	0.060	-6.539	0.000	-0.510	-0.275
ma.L1.D.PRICE	0.1182	0.069	1.711	0.087	-0.017	0.254
ma.L2.D.PRICE	-0.6341	0.065	-9.709	0.000	-0.762	-0.506

Value of minimum AIC :5939.011
Values of estimated p and estimated q : 3,2
Thus, we obtain ARIMA(3,1,2)

The Maximum Likelihood Estimation technique to calculate the value of the 6 (3 AR +2 MA +1 constant) parameters from the random data.The estimation of parameters obtained as:

Coefficients	ar1	ar2	ar3	ma1	ma2	intercept
Est. Values	-0.1031	0.4537	-0.3924	0.1182	-.6341	86.8710

Obtained ARIMA(3,1,2) model is given as:

$$Y_t = 86.8710 + (-0.1031)Y_{t-1} + 0.4537Y_{t-2} + (-0.3924)Y_{t-3} + \varepsilon_t + 0.1182\varepsilon_{t-1} + (-.6341)\varepsilon_{t-2}$$

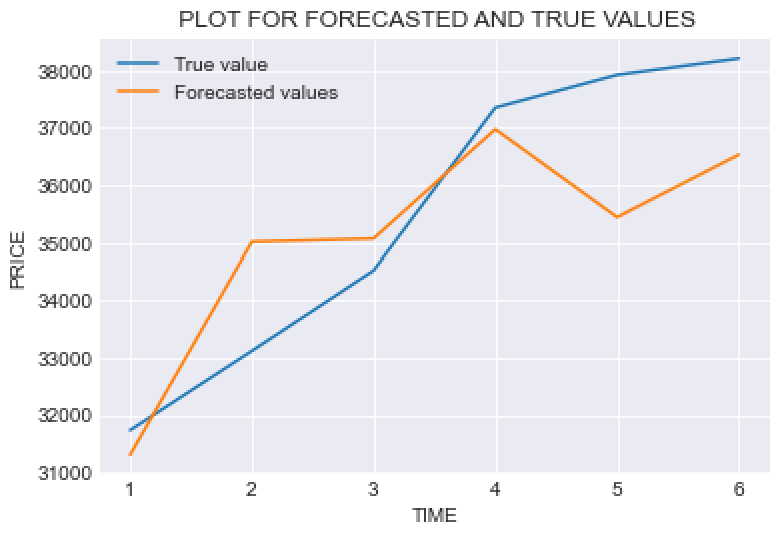
6 Forecasting

Time series forecasting is the use of a model to predict future values based on previously observed values. For testing we have left the last 6 data points . Now we will predict this 6 points and compare them .

Following table shows both true and forecasted values.

True Values	Predicted values
31737.55	31306.994267
33121.26	35016.110421
34522.26	35072.438446
37356.15	36978.011838
37926.74	35442.053130
38213.55	36533.338133

Below is the graphical representation of the above:



To forecast Key Performance Indicator(KPI) :

- **RMSE**:
  The Root Mean Squared Error (RMSE) is a strange KPI but a very helpful one. It is defined as the square root of the average squared error.

  $$RMSE = \sqrt{\tfrac{1}{n}\sum \varepsilon_t^2}$$

  - Disadvantages:
    * RMSE is not scaled to the demand.
    * RMSE emphasizes greater errors than the small ones

- **MAPE :**
  The Mean Absolute Percentage Error(MAPE) is one of the most commonly used KPIs to measure forecast accuracy.
  MAPE is the sum of the individual absolute errors divided by the demand (each period separately). It is the average of the percentage errors.

  $$MAPE = \tfrac{100\%}{n}\sum \frac{\varepsilon_t}{y_t}$$

  MAPE is a really strange forecast Key Performance Indicator(KPI).

  Interpretation:
  $MAPE < 10\%$: highly accurate forecasting.
  $10\% < MAPE < 20\%$: Good forecasting.
  $20\% < MAPE < 50\%$: reasonable forecasting.
  $MAPE > 50\%$: inaccurate forecasting.

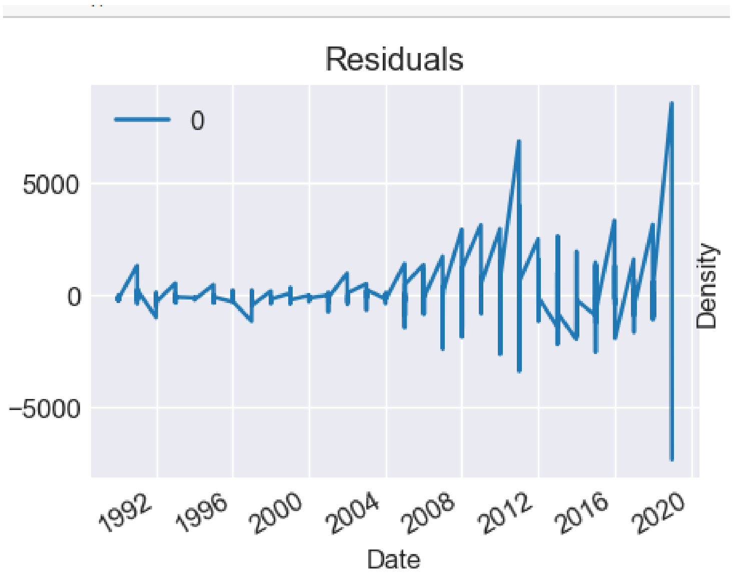
Key Performance Indicator(KPI)	$forecast$
Mean	35479.585
RMSE(Root means squared error)	1484.26760
MAPE (Mean Absolute Percentage Error)	0.034386

From above table we can see that our MAPE=3.4386% < 10% ,thus our forecasting is highly accurate.

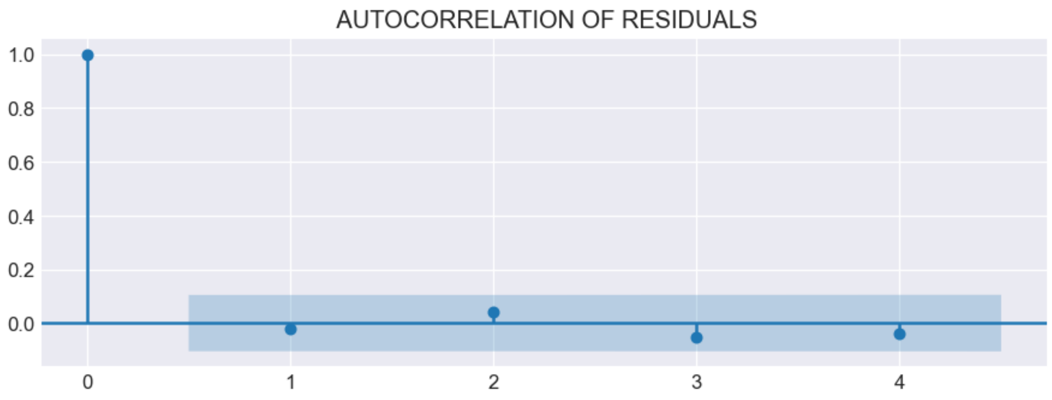
7 Residual analysis

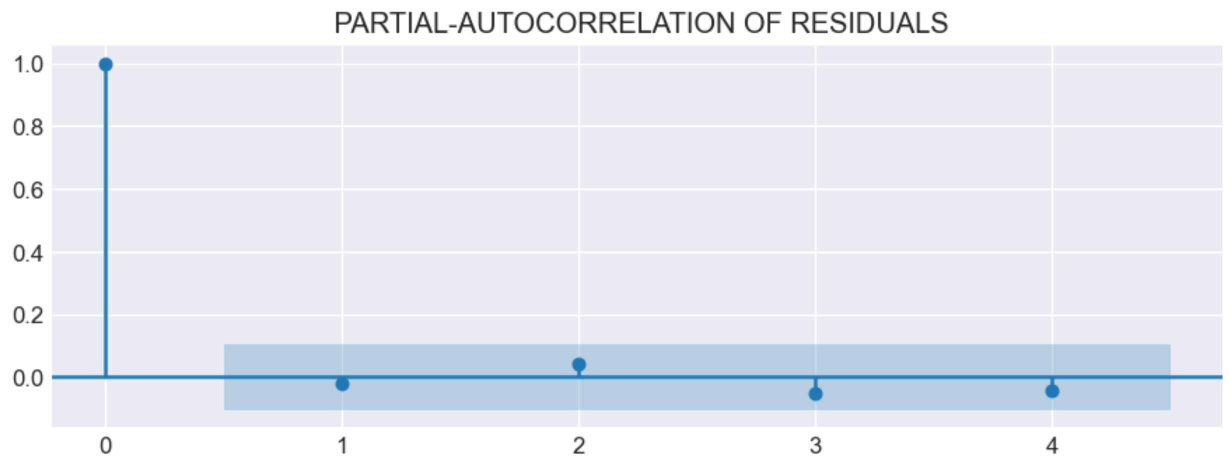
After fitting the model we should check for the randomness of residuals.

Following is the graph of residuals between fitted and predicted values.



ACF and PACF of the residuals:





From the above graph we can say that there is no significant spike hence the residuals are random.

8 Conclusion:

Analysis of Time series of gold price from 1990-2019 gives us the ARIMA (3,1,2) by minimizing the Akaike information criterion after examining the data for stationarity .The model we have developed cannot work for long run.There was an inflation in price which was captured in residual plot .We have used python language and most of the analysis is done in jupyter notebook.This model can be used to predict the future rates for short period of time.

9 Codes

Python Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from pylab import rcParams

data=pd.read_csv("tsa.csv")
data['Date']=pd.to_datetime(data['Date'])
data=pd.DataFrame(data)
df=data.set_index('Date',True)
df.plot()

# Plot the data
data['PRICE'].plot(figsize=(10, 7),color='r')
plt.ylabel("Gold Prices In Rupees")
plt.xlabel("Time")
plt.title("TIME SERIES PLOT ")
plt.show()

#data splitting into test and train
train=df.iloc[:352,:]
train=pd.DataFrame(train)

test=df.iloc[352:358,:]
test=pd.DataFrame(test)
train.plot()
test.plot()
train.describe()

from statsmodels.tsa.seasonal import seasonal_decompose
ans=seasonal_decompose(train,model='additive',period=12)

#plotting trend component
plt.plot(ans.trend)
plt.title("Trend Component")

#plotting seasonal component
plt.plot(ans.seasonal)
plt.title("Seasonal Component")

#plotting random component
plt.plot(ans.resid)
plt.title('Residual component')


#test for Randomness (Turning point test)
# Ho=Series is purely random
#H1=series is not purely random
def randtst(x):
  p=0
  n=len(x)
  for i in range(1,(n-1)):
    if ((x[i]>x[i+1])and(x[i]>x[i-1])) or ((x[i]<x[i+1])and(x[i]<x[i-1])):
      p=p+1

  print(p)
  exp_p=(2/3)*(n-2)
  Var_p=(16*n-29)/90
  teststat=(p-exp_p)/sqrt(Var_p)
  print("Test statistic=")
  print(teststat)

  if abs(teststat)>1.989:
    print("On the basis of data,series is not purely random")

  else:
    print("On the basis of data, series is purely random")
randtst(np.array(train))


#Test for presence of trend(Relative ordering test)
#Ho=No trend present
#H1=trend present
def trendtest(x):
    Q=0
    n=len(x)
    for i in range(n-1):
        for j in range(i+1,n):
            if x[i]>x[j]:
                Q=Q+1


    exp_Q=n*(n-1)/4
    tau=1-4*Q/(n*(n-1))
    var_tau=2*(2*n+5)/(9*n*(n-1))
    teststat=tau/sqrt(var_tau)
    print("test statisitic=")
    print(teststat)
    zalpha=1.959964
    if abs(teststat)>zalpha:
        print("Trend present")
    else:
        print("No trend present")

trendtest(np.array(train))


#cheking whether the series is stationary or not
from statsmodels.tsa.stattools import adfuller
# Ho :Data is non-stationary
# H1 :Data is stationary
def adfuller_test(series):
    model=adfuller(series)
    labels=['ADF Test Statistic','p-value','Lags used','No. of observation Used']
    for value,label in zip(model,labels):
        print(label+": "+str(value))
    if model[1]<=0.05:
        print("Reject the null hypothesis and conclude that Data is stationary")
    else:
        print("Week evidence against the null hypothesis and conclude that Data is non-stationary")


adfuller_test(train)

#applying first order differencing
data1=(train-train.shift(-1)).dropna()
adfuller_test(data1)
data1.plot()
plt.title('Stationary data free from Trend and Seasonality')

#test for Randomness (Turning point test) after Differencing  1st order
# Ho=Series is purely random
#H1=series is not purely random
def randtst(x):
  p=0
  n=len(x)
  for i in range(1,(n-1)):
    if ((x[i]>x[i+1])and(x[i]>x[i-1])) or ((x[i]<x[i+1])and(x[i]<x[i-1])):
      p=p+1

  print(p)
  exp_p=(2/3)*(n-2)
  Var_p=(16*n-29)/90
  teststat=(p-exp_p)/sqrt(Var_p)
  print("Test statistic=")
  print(teststat)
  if abs(teststat)>1.989:
    print("On the basis of data,series is not purely random")

  else:
    print("On the basis of data, series is purely random")
randtst(np.array(data1))


#plotting acf and pacf plot
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf

fig=sm.graphics.tsa.plot_pacf(data1,lags=24)
fig=sm.graphics.tsa.plot_acf(data1,lags=24)

# Code for auto ARIMA
import pmdarima
from pmdarima.arima import auto_arima
opt=auto_arima(train,trace=True,suppress_warning=True)
opt.summary()

 #model fitting
```

```
from statsmodels.tsa.arima_model import ARIMA

model=ARIMA(train,order=(3,1,2))
model_fit=model.fit()
model_fit.summary()

#residual plot after model fitting
residuals=pd.DataFrame(model_fit.resid)
fig,ax=plt.subplots(1,2)
residuals.plot(title='Residuals',ax=ax[0])
residuals.plot(kind='kde',title='Density',ax=ax[1])
plt.show()
model_fit.plot_predict(dynamic=False)
plt.show()

#plottng acf of residuals after model fitting
plot_acf(residuals,lags=4)
plt.title(' AUTOCORRELATION OF RESIDUALS')

#plottng pacf of residuals after model fitting
plot_pacf(residuals,lags=4)
plt.title(' PARTIAL-AUTOCORRELATION OF RESIDUALS')

#Forecasting
new=train.values
new=new.tolist()
new= [item for sublist in new for item in sublist]
len(new)

forcast=model_fit.predict(start=352,end=358,dynamic=True)
forcast=forcast.drop(351,axis=0)
forcast=forcast.values
result=[]
for i in range(len(forcast)):
   a=forcast[i]+new[351+i]
   result.append(a)
   new.append(a)
result=pd.DataFrame(result)
result

test=np.array(test)
test=pd.DataFrame(test)
result=np.array(result)
result=pd.DataFrame(result)
ax=test.plot()
result.plot(ax=ax)

#Root mean square error
from math import sqrt
def rmse(x,y):
   sum1=0
   for i in range(len(x)):
      a=x[i]-y[i]
      a*=a
```

```
        sum1+=abs(a)
    return(sqrt(sum1/len(x)))
rmse(np.array(test),np.array(result))

#Mean Absolute Percentage Error
mape=np.mean(np.abs(result-test)/np.abs(test))
mape

#forcasting glod prices of first five month of 2020
forcast1=model_fit.predict(start=352,end=362,dynamic=True)
forcast1=forcast1.drop(351,axis=0)
forcast1=forcast1.values
result1=[]
for i in range(len(forcast1)):
    a=forcast1[i]+new[351+i]
    result1.append(a)
    new.append(a)
result1=pd.DataFrame(result1)
result1

# forecasting plot
result1.plot()
plt.showimport statsmodels.api as sm
from pylab import rcParams

data=pd.read_csv("tsa.csv")
data['Date']=pd.to_datetime(data['Date'])
data=pd.DataFrame(data)
df=data.set_index('Date',True)
df.plot()
```

Reference

1. Class Notes of prof. Amit Mitra
2. David A. Dickey. "Stationarity Issues in Time Series Models"
3. Akaike, H. (21 December 1981), "This Week's Citation Classic" , Current Contents Engineering, Technology, and Applied Sciences, 12 (51): 42 [Hirotogu Akaike comments on how he arrived at AIC]
4. Akaike, H. (21 December 1981), "This Week's Citation Classic" , Current Contents Engineering, Technology, and Applied Sciences, 12 (51): 42 [Hirotogu Akaike comments on how he arrived at AIC]