```
!pip install pmdarima
```

```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore',category=DeprecationWarning)
import warnings
from statsmodels.tools.sm_exceptions import ConvergenceWarning
warnings.simplefilter('ignore', ConvergenceWarning)
import statsmodels.api as sm
from pylab import rcParams
rcParams['figure.figsize'] = 20,5
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

## DATA UNDERSTANDING

```
#READING THE DATASET
df = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/TSF/TimeSeries_Sales_Data_MiniProject.xls",parse_dates=True,index_col='Order Date',usecols=('Category','Sales','Order Date'))
df.head()
```

| | Category | Sales |
|---|---|---|
| **Order Date** | | |
| **2016-11-08** | Furniture | 261.9600 |
| **2016-11-08** | Furniture | 731.9400 |
| **2016-06-12** | Office Supplies | 14.6200 |
| **2015-10-11** | Furniture | 957.5775 |
| **2015-10-11** | Office Supplies | 22.3680 |

```
df['Category'].value_counts().plot(kind='bar',color='pink')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f925f8fe910>
```


```
df.isnull().sum()
```

```
Category    0
Sales       0
dtype: int64
```

## DATA CATEGORISING FOR ANALYSIS

```
#category wise data frames
one = df.loc[df['Category'] == 'Office Supplies']
two = df.loc[df['Category'] == 'Furniture']
three = df.loc[df['Category'] == 'Technology']
```

```
#selecting only columns required for analysis
offsup = one['Sales']
furn = two['Sales']
tech = three['Sales']
```

## FUNCTIONS TO USE IN FURTHER MODEL BUILDING AND FORECASTING

```
from statsmodels.tsa.stattools import adfuller

def checkstationarity(data):
  pvalue = adfuller (data)[1]
  if pvalue < 0.05:
    msg =  "pvalue={}. Data is Stationary. Proceed to model building".format(pvalue)
  else:
```

```
    msg = "pvalue={}. Data is not Stationary. Make the data stationary before model building".format(pvalue)
  return msg
```

```
from statsmodels.tsa.seasonal import seasonal_decompose

def checkcharacterstics(data):
  dc = seasonal_decompose(data,period=12)
  plt.rcParams.update({'figure.figsize': (16,8)})
  dc.plot().suptitle('Data Decomposition', fontsize=16)
```

```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf

def plotlag(data):
  fig,ax = plt.subplots(1,2,figsize=(20,4))
  plot_acf(data,lags=20,ax=ax[0])
  plot_pacf(data,lags=20,ax=ax[1])
  plt.show()
```

```
import statsmodels.api as sm

def ljungoxtest(model):
  pval = sm.stats.acorr_ljungbox(model.resid,lags=[1],return_df=True)['lb_pvalue'].values
  if pval < 0.05:
    print('Reject H0, bad model')
  else:
    print('Accept H0, good model')
```

```
def plotpred(data,pred):
  data.plot(label='Actual Dataset')
  pred.plot(label='Predicted Test Set')
  plt.legend()
  plt.show()
```

```
def forecasting(model,data):
  global fdata
  forecast = model.predict(start=len(data),end=len(data)+23,dynamic=True)
  fdata = data.append(forecast)
  fdata.plot(label="Current Sales")
  forecast.plot(label="Future Sales")
  plt.legend()
  plt.title("Forecasted Sales for the Upcoming Years")
  plt.show()
```

## TIME SERIES FORECASTING FOR OFFICE SUPPLIES CATEGORY

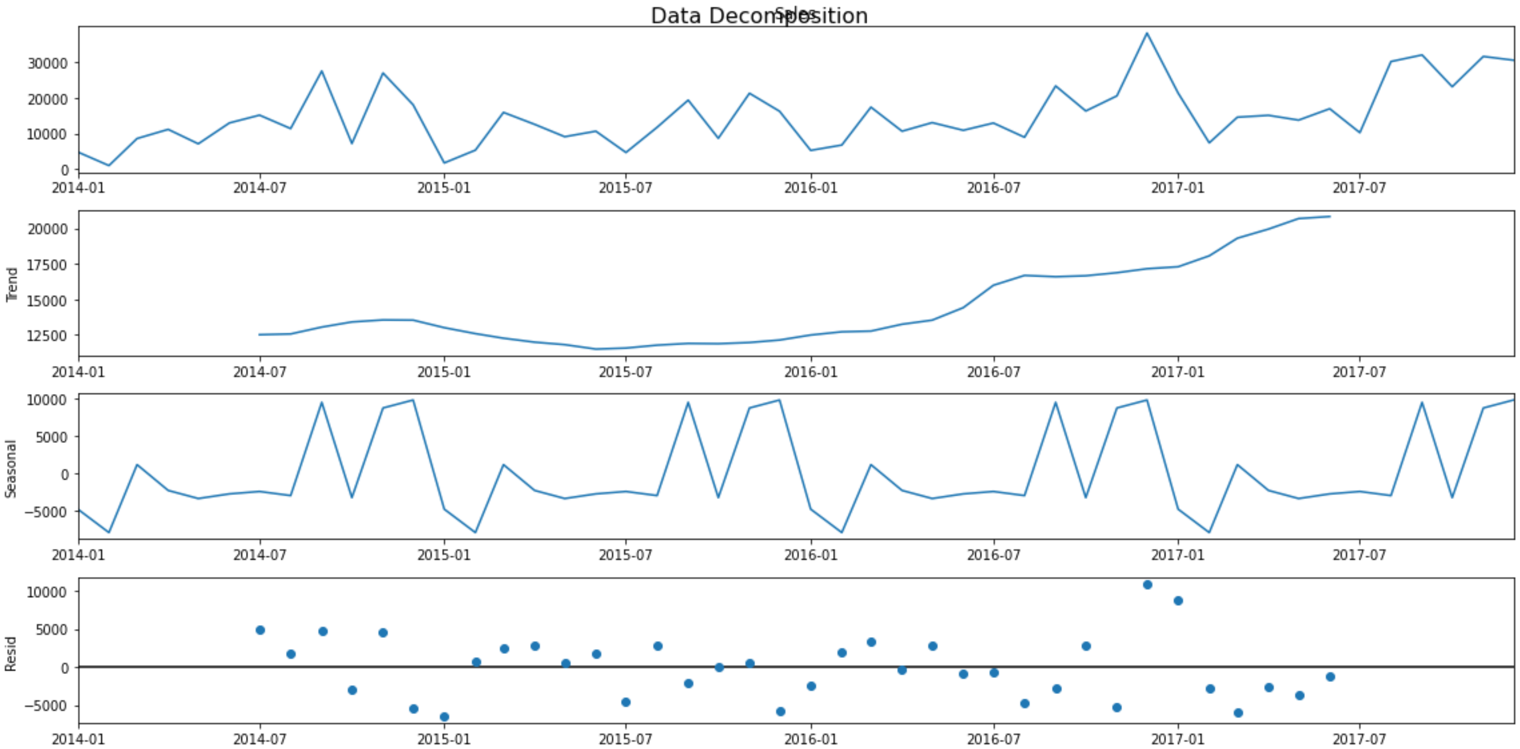
```
#CONVERTING INDEX TO DATE TIME FORMAT
offsup.index = pd.to_datetime(offsup.index)
```

```
#SORTING THE INDEX
offsup.sort_index(ascending=True,inplace=True)
```

```
#SAMPLING THE DATA MONTHLY FOR DATA REDUCTION
osdata = offsup.resample('MS').sum()
osdata.head()
```

```
Order Date
2014-01-01     4851.080
2014-02-01     1071.724
2014-03-01     8605.879
2014-04-01    11155.074
2014-05-01     7135.624
Freq: MS, Name: Sales, dtype: float64
```

```
#PLOTS TO CHECK TREND,SEASONALITY,RESIDS
checkcharacterstics(osdata)
```




*DATA HAS ALL THREE TIME SERIES CHARACTERSTICS TREND, SEASONALITY AND NOISE*

```
#CHECKING FOR DATA STATIONARITY
checkstationarity(osdata)
```

```
'pvalue=0.32948727549472817. Data is not Stationary. Make the data stationary before model building'
```

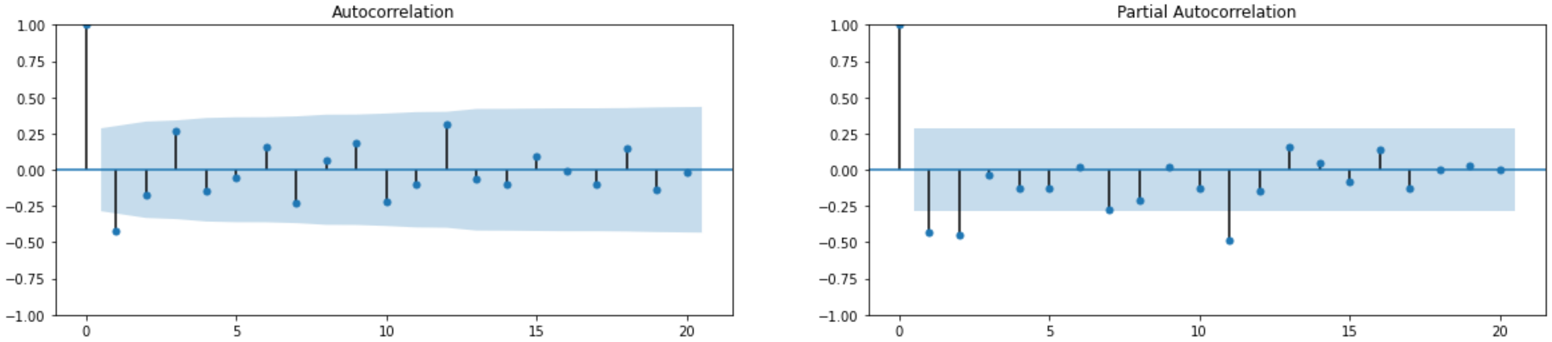
```
osdata_diff = osdata - osdata.shift(1)
osdata_diff.dropna(inplace=True)
checkstationarity(osdata_diff)
```

```
'pvalue=0.00042872624590977896. Data is Stationary. Proceed to model building'
```

1. *SINCE DATA IS NON STATIONARY WE CAN MOVE WITH EITHER ARIMA OR SARIMA.*
2. *SINCE THERE IS SEASONALITY PRESENT I WILL CHOOSE SARIMA MODEL.*

```
#CHECKING LAG VALUES
plotlag(osdata_diff)
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/graphics/tsaplots.py:348: FutureWarning: The default method 'yw' can produce PACF values outside of the [-1,1] interval. After 0.13, the default
  warnings.warn(
```

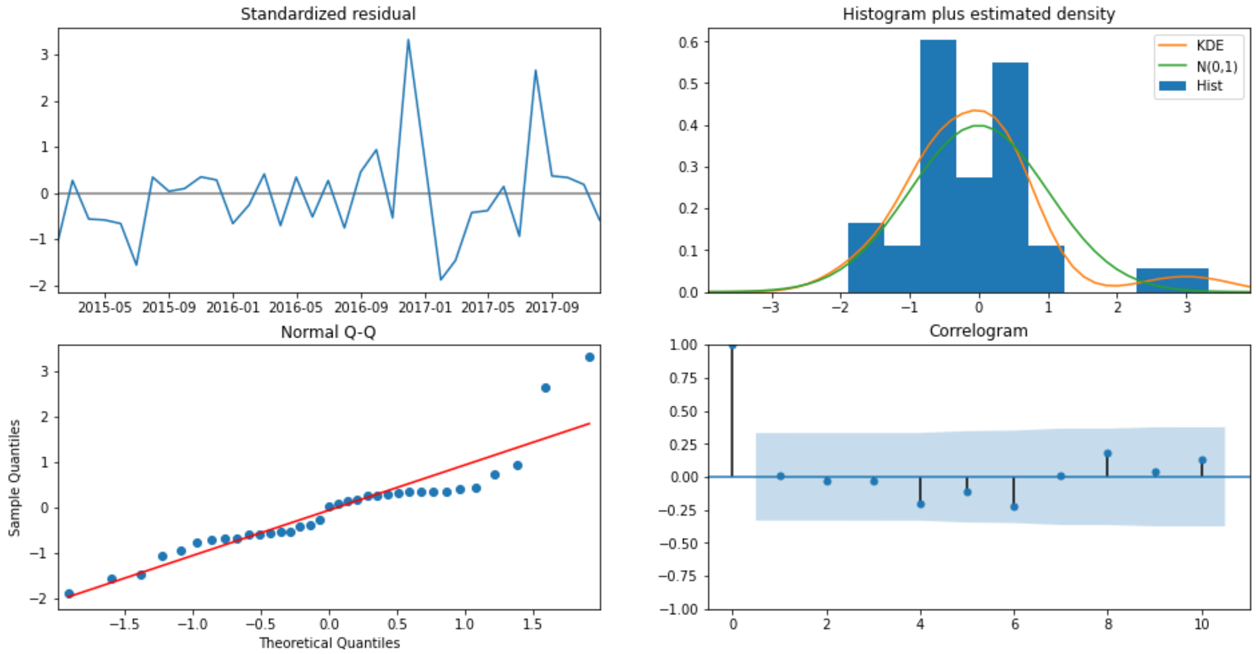



```
#FINDING BEST P,Q,D,p,q,d values for building the model
from pmdarima import auto_arima
m1 = auto_arima(y=osdata,start_p=1,start_q=1,max_p=3,max_q=3,m=12,d=1,D=1,start_P=0,start_Q=0,seasonal=True,error_action='ignore',suppress_warnings=True,trace=Tr
print(m1.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,1,0)[12]             : AIC=730.407, Time=0.07 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=740.437, Time=0.03 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=726.023, Time=0.12 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=717.172, Time=0.28 sec
 ARIMA(0,1,1)(0,1,0)[12]             : AIC=728.668, Time=0.05 sec
 ARIMA(0,1,1)(1,1,1)[12]             : AIC=718.898, Time=0.45 sec
 ARIMA(0,1,1)(0,1,2)[12]             : AIC=718.892, Time=0.87 sec
 ARIMA(0,1,1)(1,1,0)[12]             : AIC=718.780, Time=0.26 sec
 ARIMA(0,1,1)(1,1,2)[12]             : AIC=725.260, Time=0.54 sec
 ARIMA(0,1,0)(0,1,1)[12]             : AIC=730.324, Time=0.08 sec
 ARIMA(1,1,1)(0,1,1)[12]             : AIC=722.788, Time=0.17 sec
 ARIMA(0,1,2)(0,1,1)[12]             : AIC=718.277, Time=0.62 sec
 ARIMA(1,1,0)(0,1,1)[12]             : AIC=722.473, Time=0.37 sec
 ARIMA(1,1,2)(0,1,1)[12]             : AIC=723.414, Time=0.98 sec
 ARIMA(0,1,1)(0,1,1)[12] intercept   : AIC=723.739, Time=0.15 sec

Best model:  ARIMA(0,1,1)(0,1,1)[12]
Total fit time: 5.070 seconds
                                SARIMAX Results
```

```
==============================================================================================
Dep. Variable:                                   y   No. Observations:                   48
Model:             SARIMAX(0, 1, 1)x(0, 1, 1, 12)   Log Likelihood                -355.586
Date:                             Tue, 31 Jan 2023   AIC                            717.172
Time:                                     14:16:16   BIC                            721.838
Sample:                                 01-01-2014   HQIC                           718.783
                                      - 12-01-2017
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.6782      0.117     -5.811      0.000      -0.907      -0.449
ma.S.L12      -0.6428      0.285     -2.256      0.024      -1.201      -0.084
sigma2      3.679e+07   4.94e-09   7.45e+15      0.000    3.68e+07    3.68e+07
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                23.75
Prob(Q):                              0.93   Prob(JB):                         0.00
Heteroskedasticity (H):               2.76   Skew:                             1.30
Prob(H) (two-sided):                  0.09   Kurtosis:                         6.08
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 2.64e+31. Standard errors may be unstable.
```

```
#CHECKING FOR MODEL DIAGNOSTICS BEFORE BULIDING
m1.plot_diagnostics()
plt.show()
```



1. Standardised Residuals : Thereare obvious patterns in the residuals with no uniform mean and variance.
2. KDE Curve : Dosent show normal distributions, data is skewed.
3. Normal Q-Q : Most of the data points are not on the straight line/normal reference line.

```
split1 = int(0.8 * len(osdata))
train1 = osdata[:split1]
test1 = osdata[split1:]
```

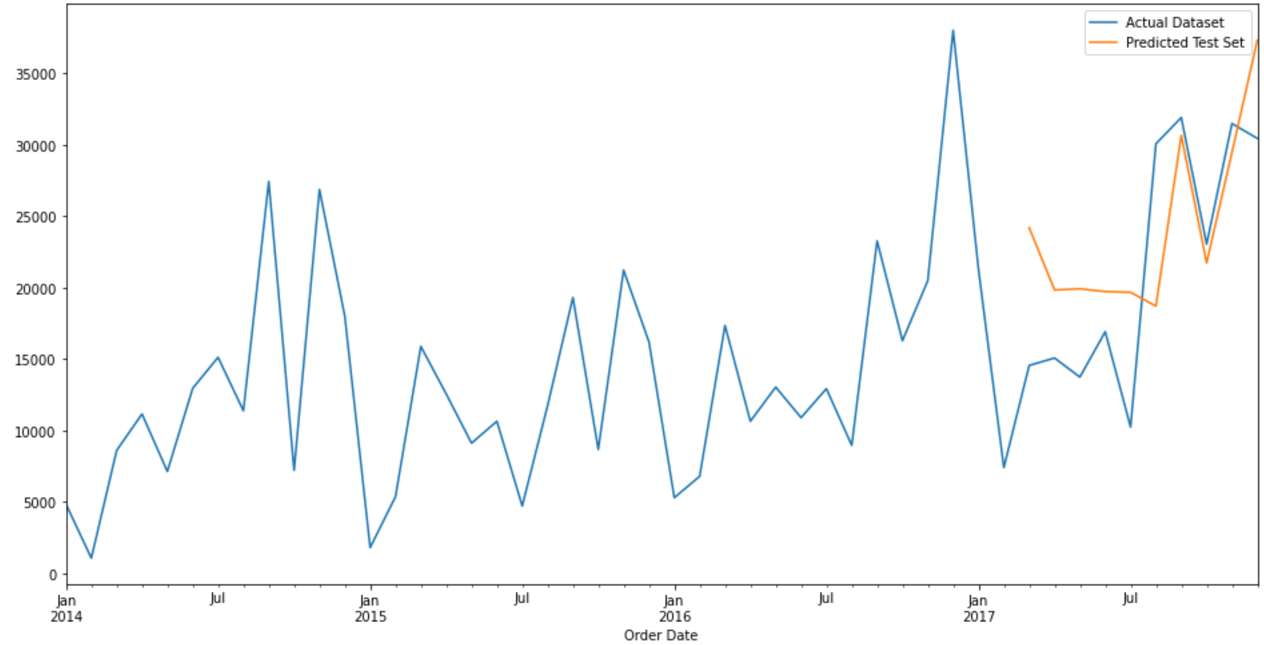
```
#BUILDING THE SARIMA MODEL FOR OFFICE SUPPLIES
os_model = sm.tsa.statespace.SARIMAX(train1,order=(1,1,1),seasonal_order=(0,1,1,12)).fit()
print(os_model.summary())
```

```
                                     SARIMAX Results                                      
==========================================================================================
Dep. Variable:                              Sales   No. Observations:                   38
Model:             SARIMAX(1, 1, 1)x(0, 1, 1, 12)   Log Likelihood                -253.947
Date:                            Tue, 31 Jan 2023   AIC                            515.894
Time:                                    14:57:14   BIC                            520.770
Sample:                                01-01-2014   HQIC                           517.247
                                     - 02-01-2017                                         
Covariance Type:                              opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.0075      0.415      0.018      0.986      -0.805       0.820
ma.L1         -0.6175      0.320     -1.930      0.054      -1.245       0.010
ma.S.L12      -0.4241      0.741     -0.573      0.567      -1.876       1.027
sigma2      5.149e+07   8.66e-09   5.94e+15      0.000    5.15e+07    5.15e+07
===================================================================================
Ljung-Box (L1) (Q):                   0.45   Jarque-Bera (JB):                15.68
Prob(Q):                              0.50   Prob(JB):                         0.00
Heteroskedasticity (H):               3.60   Skew:                             1.10
Prob(H) (two-sided):                  0.09   Kurtosis:                         6.19
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 4.75e+31. Standard errors may be unstable.
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/statespace/sarimax.py:866: UserWarning: Too few observations to estimate starting parameters for seasonal ARMA. All parameters except for va
  warn('Too few observations to estimate starting parameters%s.'
```

```
#LJUNG BOX TEST
ljungoxtest(os_model)
```

```
Accept H0, good model
```

```
#PERFORMACE METRICS and PREDICTED DATA
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error
pred1 = os_model.predict(start=len(train1),end=len(osdata)-1,dynamic=True)
mse1 = mean_squared_error(test1,pred1)
mape1 = mean_absolute_percentage_error(test1,pred1)
plotpred(osdata,pred1)
```




```
#FORECASTING
forecasting(os_model,osdata)
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/statespace/kalman_filter.py:2290: ValueWarning: Dynamic prediction specified to begin during out-of-sample forecasting period, and so has no
  warn('Dynamic prediction specified to begin during'
```

INTERPRETATIONS:

1. SARIMA MODEL WAS USED HERE FOR FORECASTING DUE TO THE PRESENCE OF SEASONALITY.
2. DATA WAS NOT STATIONARY IN THIS CATEGORY.
3. PATTERNS ARE QUIET SIMILAR TO THE PREVIOUS YEAR SALES RECORD.
4. UPWARD TREND CAN BE SEEN IN THE OVERALL DATA AND THE DATA FORECASTED.
5. YEAR END SALES IN OFFICE SUPPLY CATEGORY ARE COMPARITIVELY MORE.

## TIME SERIES FORECASTING FOR FURNITURE CATEGORY

```
#CONVERTING INDEX TO DATE TIME FORMAT
furn.index = pd.to_datetime(furn.index)
```

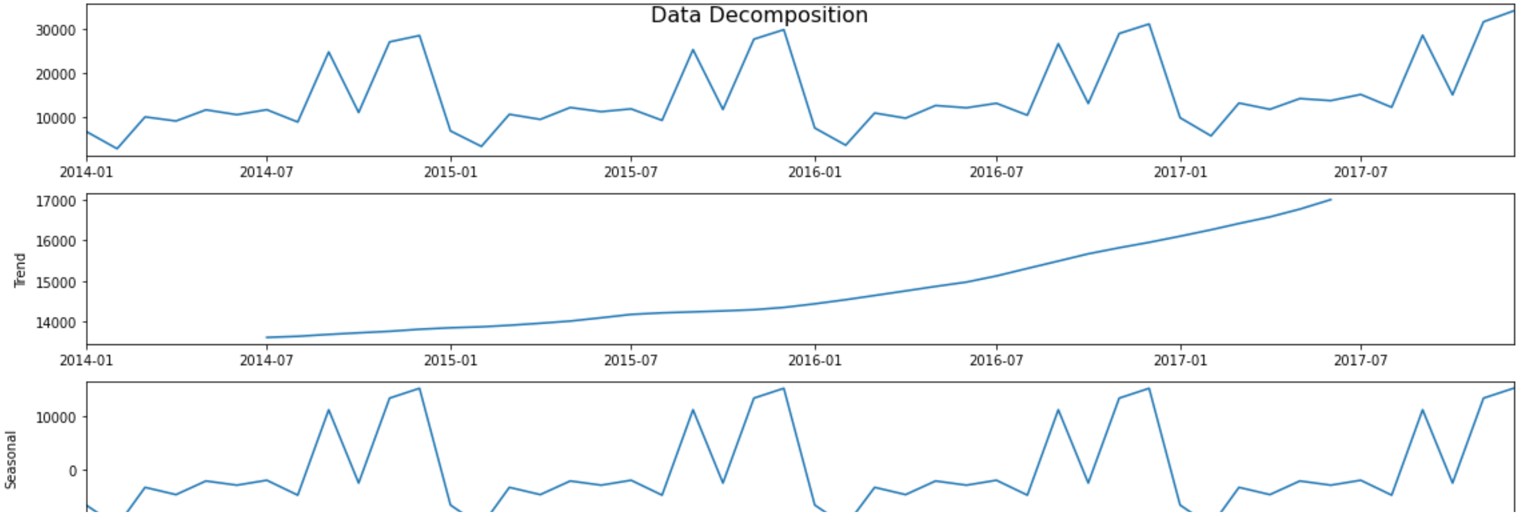
```
#SORTING THE INDEX
furn.sort_index(ascending=True,inplace=True)
```

```
#SAMPLING THE DATA MONTHLY FOR DATA REDUCTION
furndata = furn.resample('MS').sum()
furndata.head()
```

```
Order Date
2014-01-01     6242.525
2014-02-01     1839.658
2014-03-01    14573.956
2014-04-01     7944.837
2014-05-01     6912.787
Freq: MS, Name: Sales, dtype: float64
```

```
#SMOOTHING
from statsmodels.tsa.api import ExponentialSmoothing
df2_smo = ExponentialSmoothing(furndata,trend='add',seasonal='add',seasonal_periods=12).fit()
furndata = df2_smo.fittedvalues
```

```
#PLOTS TO CHECK TREND,SEASONALITY,RESIDS
checkcharacterstics(furndata)
```




*DATA HAS ALL THREE TIME SERIES CHARACTERSTICS TREND, SEASONALITY AND NOISE*

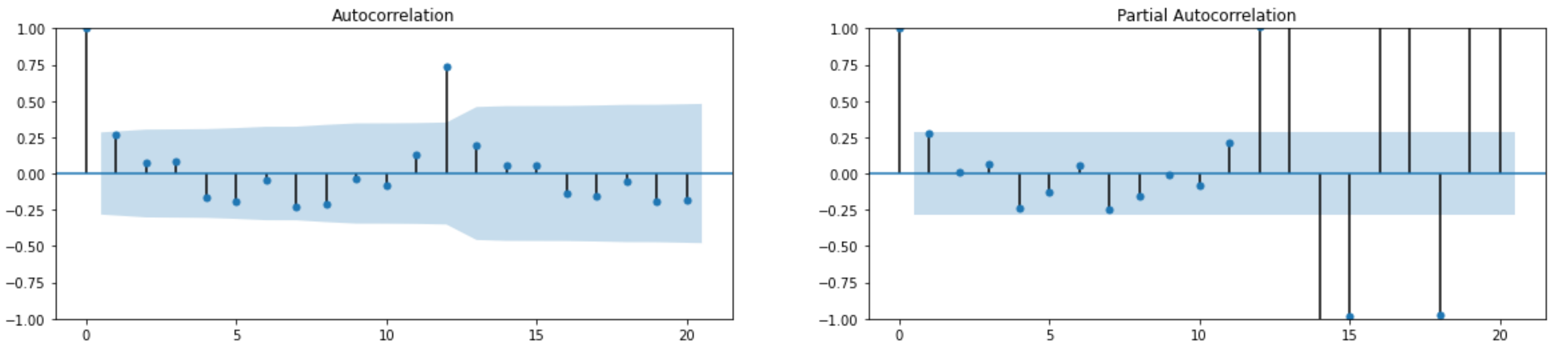
```
#CHECKING FOR DATA STATIONARITY
checkstationarity(furndata)
```

```
'pvalue=9.513079933718842e-05. Data is Stationary. Proceed to model building'
```

1. *SINCE DATA IS STATIONARY WE CAN MOVE WITH ANY OF THE EITHER ARMA/ARIMA/SARIMA.*
2. *SINCE THERE IS SEASONALITY PRESENT I WILL CHOOSE SARIMA MODEL.*

```
#CHECKING LAG VALUES
plotlag(furndata)
```

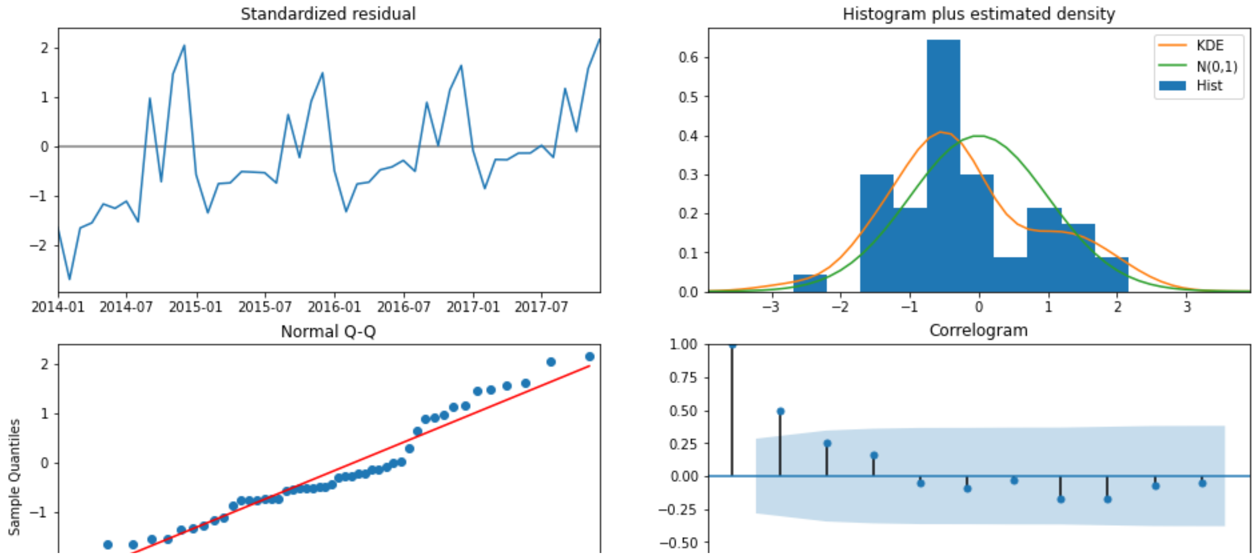
```
/usr/local/lib/python3.8/dist-packages/statsmodels/graphics/tsaplots.py:348: FutureWarning: The default method 'yw' can produce PACF values outside of the [-1,1] interval. After 0.13, the default
  warnings.warn(
```



```
#FINDING BEST P,Q,D,p,q,d values for building the model
m2 = auto_arima(y=furndata,start_p=1,start_q=1,max_p=9,max_q=2,m=12,d=0,D=0,start_P=0,start_Q=0,seasonal=True,error_action='ignore',suppress_warnings=True,trace=True)
print(m2.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,0,0)[12] intercept   : AIC=1008.144, Time=0.04 sec
 ARIMA(0,0,0)(0,0,0)[12] intercept   : AIC=1008.133, Time=0.02 sec
 ARIMA(1,0,0)(1,0,0)[12] intercept   : AIC=975.883, Time=0.31 sec
 ARIMA(0,0,1)(0,0,1)[12] intercept   : AIC=989.458, Time=0.08 sec
 ARIMA(0,0,0)(0,0,0)[12]             : AIC=1074.445, Time=0.01 sec
 ARIMA(1,0,0)(0,0,0)[12] intercept   : AIC=1006.161, Time=0.03 sec
 ARIMA(1,0,0)(2,0,0)[12] intercept   : AIC=990.474, Time=0.22 sec
 ARIMA(1,0,0)(1,0,1)[12] intercept   : AIC=985.751, Time=0.14 sec
 ARIMA(1,0,0)(0,0,1)[12] intercept   : AIC=987.766, Time=0.14 sec
 ARIMA(1,0,0)(2,0,1)[12] intercept   : AIC=inf, Time=2.67 sec
 ARIMA(0,0,0)(1,0,0)[12] intercept   : AIC=1003.891, Time=0.13 sec
 ARIMA(2,0,0)(1,0,0)[12] intercept   : AIC=989.294, Time=0.27 sec
 ARIMA(1,0,1)(1,0,0)[12] intercept   : AIC=inf, Time=1.38 sec
 ARIMA(0,0,1)(1,0,0)[12] intercept   : AIC=1001.164, Time=0.14 sec
 ARIMA(2,0,1)(1,0,0)[12] intercept   : AIC=inf, Time=0.91 sec
 ARIMA(1,0,0)(1,0,0)[12]             : AIC=inf, Time=0.45 sec

Best model:  ARIMA(1,0,0)(1,0,0)[12] intercept
Total fit time: 6.980 seconds
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                  y   No. Observations:                   48
Model:             SARIMAX(1, 0, 0)x(1, 0, 0, 12)   Log Likelihood                -483.941
Date:                            Tue, 31 Jan 2023   AIC                            975.883
Time:                                    15:03:21   BIC                            983.367
Sample:                                01-01-2014   HQIC                           978.711
                                     - 12-01-2017
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept   9373.4714   1560.511      6.007      0.000    6314.927    1.24e+04
ar.L1         -0.2183      0.192     -1.135      0.256      -0.595       0.159
ar.S.L12       0.5494      0.105      5.244      0.000       0.344       0.755
sigma2      2.604e+07      0.227   1.15e+08      0.000     2.6e+07     2.6e+07
===================================================================================
Ljung-Box (L1) (Q):                  12.36   Jarque-Bera (JB):                 1.38
Prob(Q):                              0.00   Prob(JB):                         0.50
Heteroskedasticity (H):               0.44   Skew:                             0.40
Prob(H) (two-sided):                  0.11   Kurtosis:                         2.78
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 4.8e+23. Standard errors may be unstable.
```

```
#CHECKING FOR MODEL DIAGNOSTICS BEFORE BULIDING
m2.plot_diagnostics()
plt.show()
```




1. Standardised Residuals : Thereare no obvious patterns in the residuals.
2. KDE Curve : Shows normal distributions, data is skewed.
3. Normal Q-Q : Most of the data points are on the straight line/normal reference line.

```
split2 = int(0.8 * len(furndata))
train2 = furndata[:split2]
test2 = furndata[split2:]
```

```
#BUILDING THE SARIMA MODEL FOR OFFICE SUPPLIES
furn_model = sm.tsa.statespace.SARIMAX(train2,order=(1,0,1),seasonal_order=(1,0,0,12)).fit()
print(furn_model.summary())
```

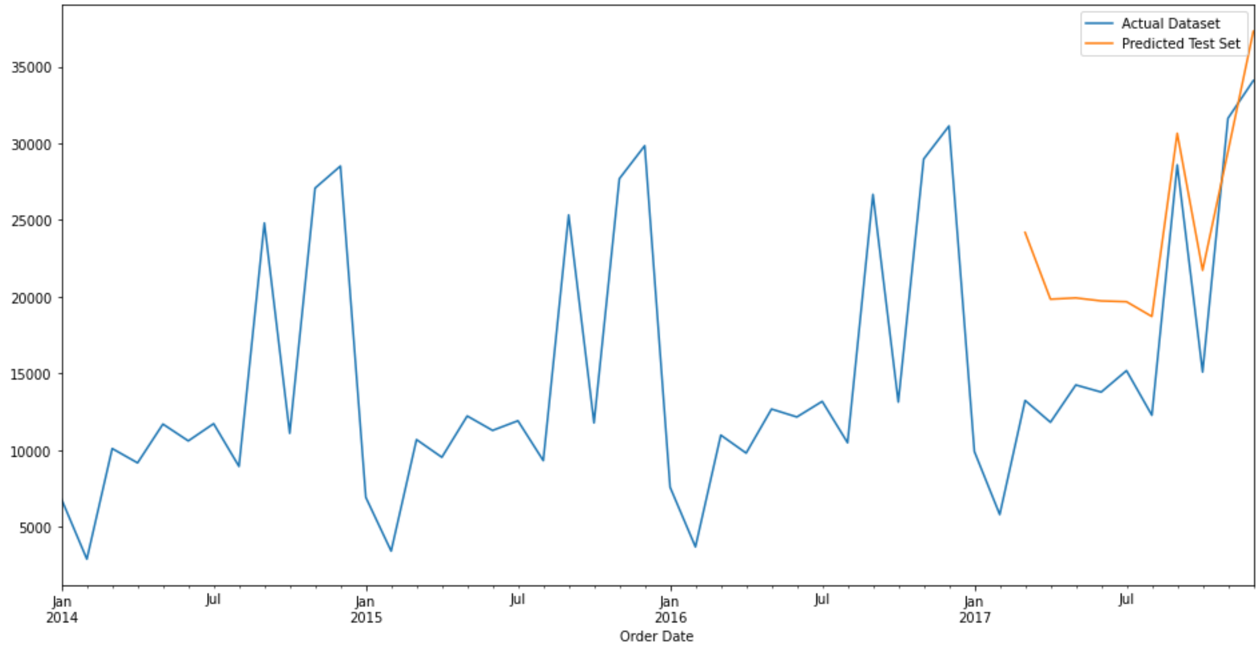
```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/statespace/sarimax.py:997: UserWarning: Non-stationary starting seasonal autoregressive Using zeros as starting parameters.
  warn('Non-stationary starting seasonal autoregressive'
                                      SARIMAX Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:                   38
Model:             SARIMAX(1, 0, 1)x(1, 0, [], 12)   Log Likelihood                -358.764
Date:                             Tue, 31 Jan 2023   AIC                            725.529
Time:                                     15:03:26   BIC                            732.079
Sample:                                 01-01-2014   HQIC                           727.859
                                      - 02-01-2017
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9998      0.133      7.500      0.000       0.739       1.261
ma.L1         -0.9994      0.259     -3.854      0.000      -1.508      -0.491
ar.S.L12       0.9878      0.010    102.698      0.000       0.969       1.007
sigma2      5.302e+06   1.87e-08   2.83e+14      0.000      5.3e+06     5.3e+06
===================================================================================
Ljung-Box (L1) (Q):                   4.88   Jarque-Bera (JB):                14.15
```

```
Prob(Q):                              0.03   Prob(JB):                         0.00
Heteroskedasticity (H):               0.53   Skew:                             1.33
Prob(H) (two-sided):                  0.27   Kurtosis:                         4.38
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 6.51e+30. Standard errors may be unstable.
```

```
#LJUNG BOX TEST
ljungoxtest(furn_model)
```

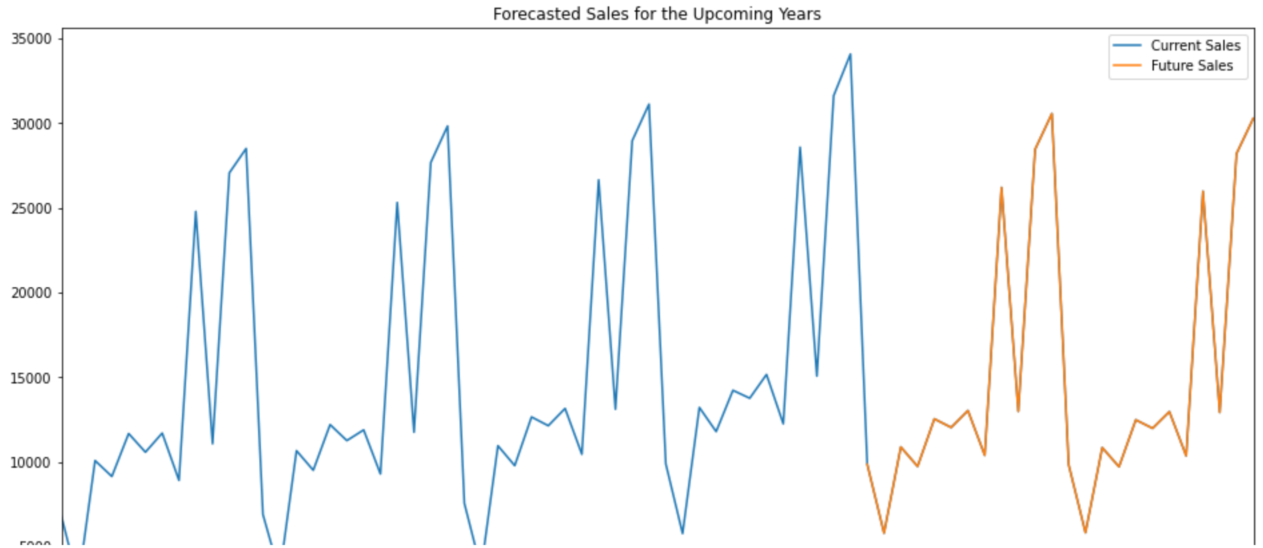
```
Reject H0, bad model
```

```
#PERFORMACE METRICS and PREDICTED DATA
pred2 = os_model.predict(start=len(train2),end=len(furndata)-1,dynamic=True)
mse2 = mean_squared_error(test2,pred2)
mape2 = mean_absolute_percentage_error(test2,pred2)
plotpred(furndata,pred2)
```




```
#FORECASTING
forecasting(furn_model,furndata)
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/statespace/kalman_filter.py:2290: ValueWarning: Dynamic prediction specified to begin during out-of-sample forecasting period, and so has no
  warn('Dynamic prediction specified to begin during'
```




INTERPRETATIONS:

1. SARIMA MODEL WAS USED HERE FOR FORECASTING DUE TO THE PRESENCE OF SEASONALITY.
2. DATA WAS STATIONARY IN THIS CATEGORY.
3. PATTERNS ARE VERY SIMILAR TO THE PREVIOUS YEAR SALES RECORD.
4. VERY SLIGHT UPWARD TREND CAN BE SEEN IN THE OVERALL DATA AND THE DATA FORECASTED.
5. YEAR END SALES IN FURNITURE SALES ARE COMPARITIVELY MORE.
6. DIPS IN SALES CAN BE SEEN 1ST QUARTER.

## TIME SERIES FORECASTING FOR TECHNOLOGY CATEGORY

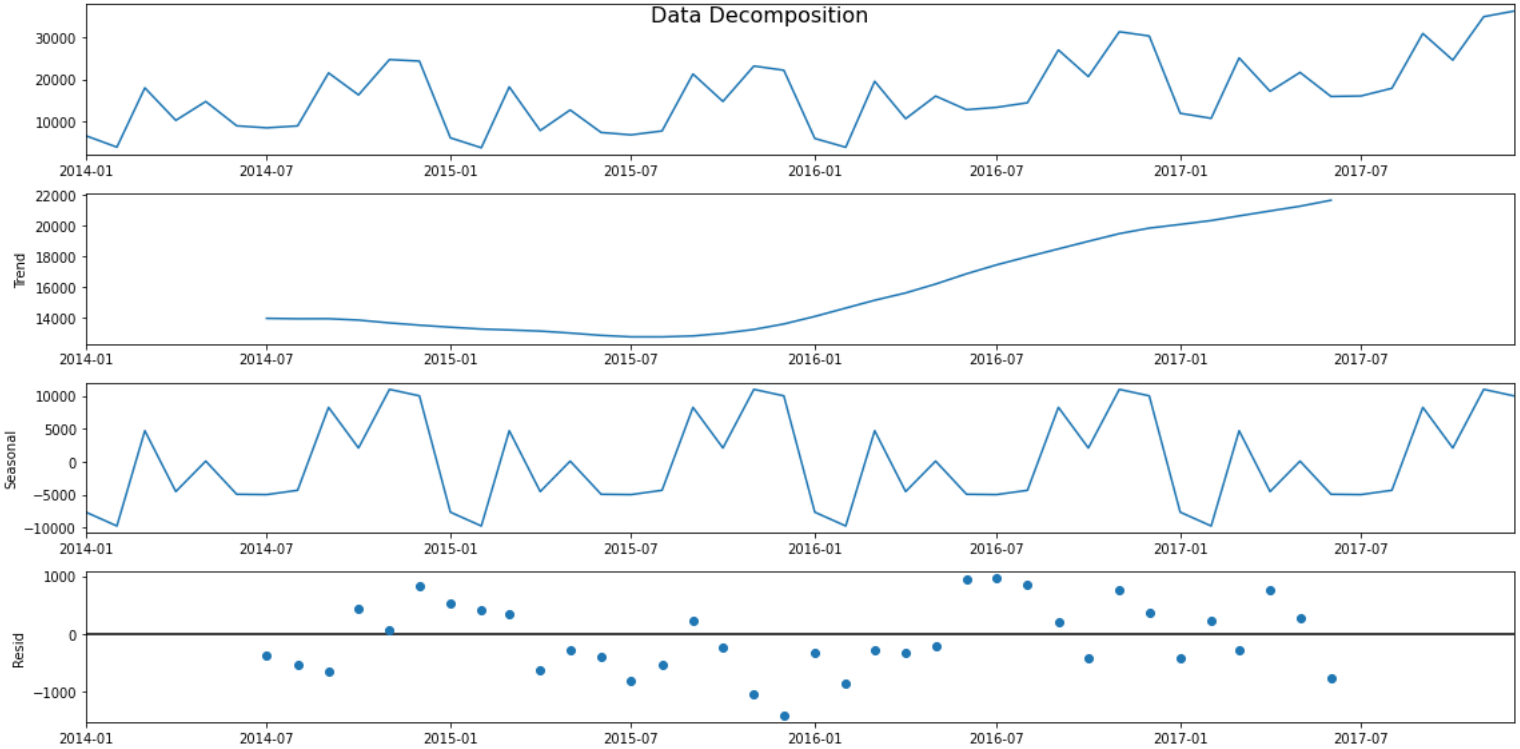
```
#CONVERTING INDEX TO DATE TIME FORMAT
tech.index = pd.to_datetime(tech.index)


#SORTING THE INDEX
tech.sort_index(ascending=True,inplace=True)


#SAMPLING THE DATA MONTHLY FOR DATA REDUCTION
techdata = tech.resample('MS').sum()
techdata.head()
```

```
Order Date
2014-01-01     3143.290
2014-02-01     1608.510
2014-03-01    32511.174
2014-04-01     9195.434
```

```
2014-05-01     9599.876
Freq: MS, Name: Sales, dtype: float64
```

```
#SMOOTHING
from statsmodels.tsa.api import ExponentialSmoothing
df_smo = ExponentialSmoothing(techdata,trend='add',seasonal='add',seasonal_periods=12).fit()
techdata = df_smo.fittedvalues
```

```
#PLOTS TO CHECK TREND,SEASONALITY,RESIDS
checkcharacterstics(techdata)
```




*DATA HAS ALL THREE TIME SERIES CHARACTERSTICS TREND, SEASONALITY AND NOISE*

```
#CHECKING FOR DATA STATIONARITY
checkstationarity(techdata)
```

```
'pvalue=0.9852373582089293. Data is not Stationary. Make the data stationary before model building'
```

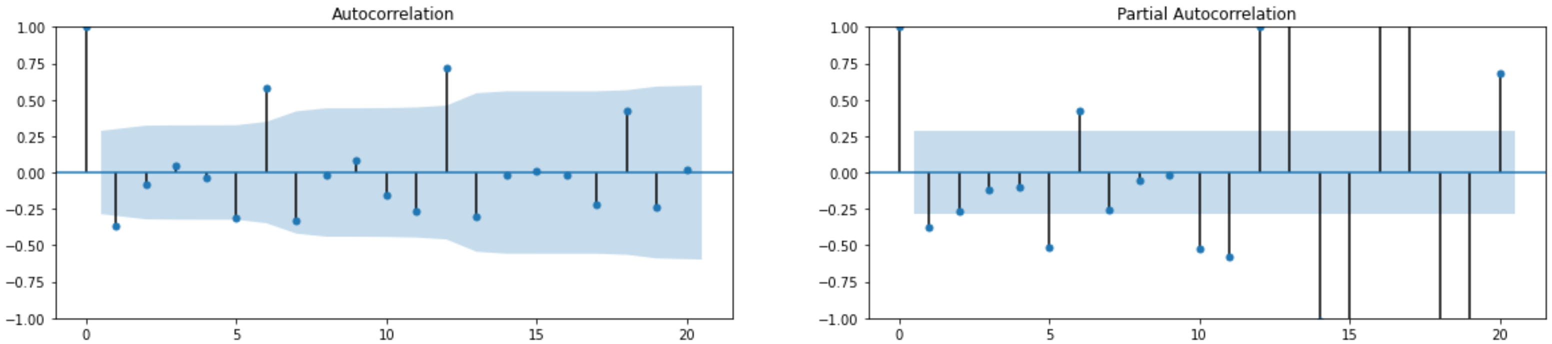
```
techdata_diff = techdata - techdata.shift(1)
techdata_diff.dropna(inplace=True)
checkstationarity(techdata_diff)
```

```
'pvalue=1.5818270748199294e-16. Data is Stationary. Proceed to model building'
```

1. *SINCE DATA IS NOT STATIONARY WE CAN MOVE WITH ANY OF THE EITHER ARMA/ARIMA/SARIMA.*
2. *SINCE THERE IS SEASONALITY PRESENT I WILL CHOOSE SARIMA MODEL.*

```
#CHECKING LAG VALUES
plotlag(techdata_diff)
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/graphics/tsaplots.py:348: FutureWarning: The default method 'yw' can produce PACF values outside of the [-1,1] interval. After 0.13, the default
  warnings.warn(
```

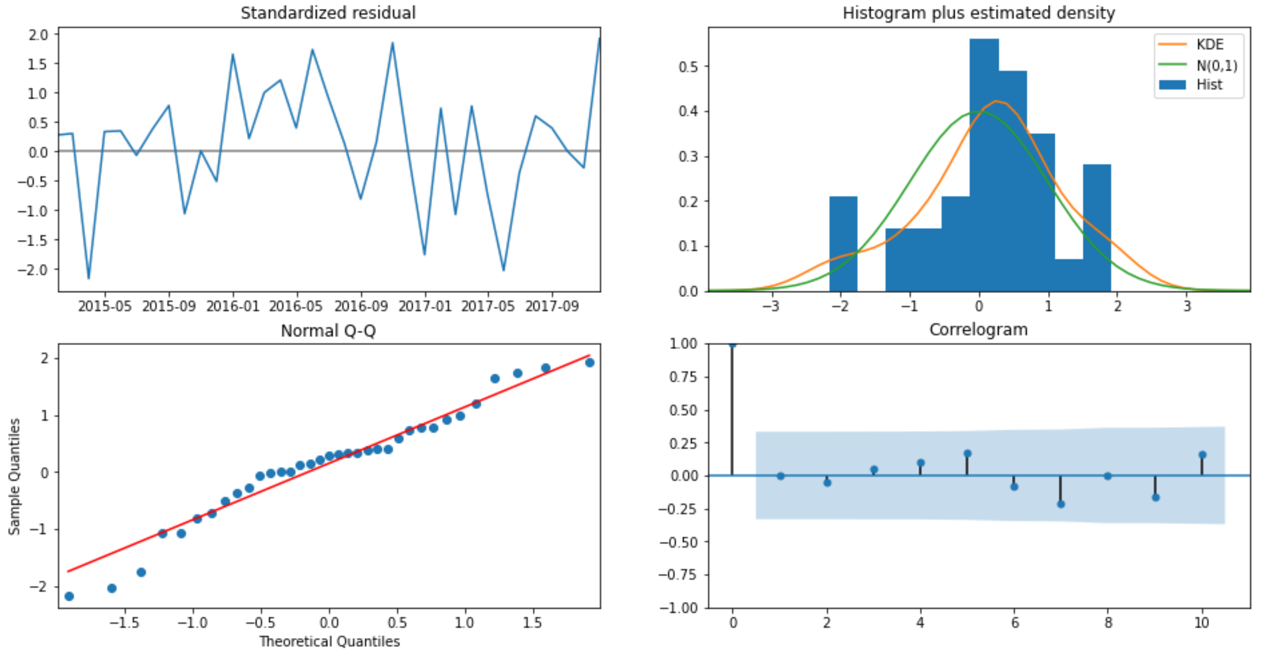



```
#FINDING BEST P,Q,D,p,q,d values for building the model
m3 = auto_arima(y=techdata,start_p=1,start_q=1,max_p=14,max_q=3,m=12,d=1,D=1,start_P=0,start_Q=0,seasonal=True,error_action='ignore',suppress_warnings=True,trace=True)
print(m3.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,1,0)[12]             : AIC=602.198, Time=0.07 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=598.230, Time=0.03 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=601.606, Time=0.09 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=601.604, Time=0.08 sec
 ARIMA(0,1,0)(1,1,0)[12]             : AIC=599.802, Time=0.05 sec
 ARIMA(0,1,0)(0,1,1)[12]             : AIC=599.680, Time=0.05 sec
 ARIMA(0,1,0)(1,1,1)[12]             : AIC=inf, Time=0.38 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=600.176, Time=0.07 sec
 ARIMA(0,1,1)(0,1,0)[12]             : AIC=600.193, Time=0.04 sec
 ARIMA(0,1,0)(0,1,0)[12] intercept   : AIC=599.435, Time=0.02 sec

Best model:  ARIMA(0,1,0)(0,1,0)[12]
Total fit time: 0.899 seconds
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                              y   No. Observations:                     48
Model:             SARIMAX(0, 1, 0)x(0, 1, 0, 12)   Log Likelihood                  -298.115
Date:                        Tue, 31 Jan 2023   AIC                              598.230
Time:                                14:16:30   BIC                              599.785
Sample:                            01-01-2014   HQIC                             598.767
                                 - 12-01-2017
Covariance Type:                          opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
```

```
sigma2      1.462e+06   3.66e+05      3.993      0.000    7.44e+05    2.18e+06
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 1.09
Prob(Q):                              0.98   Prob(JB):                         0.58
Heteroskedasticity (H):               1.45   Skew:                            -0.43
Prob(H) (two-sided):                  0.53   Kurtosis:                         3.07
===================================================================================

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

```
#CHECKING FOR MODEL DIAGNOSTICS BEFORE BULIDING
m3.plot_diagnostics()
plt.show()
```




1. Standardised Residuals : Thereare obvious patterns in the residuals with no uniform mean and variance.
2. KDE Curve : Dosent show normal distributions, data is skewed.
3. Normal Q-Q : Most of the data points are not on the straight line/normal reference line.

```
split3 = int(0.8 * len(techdata))
train3 = techdata[:split3]
test3 = techdata[split3:]
```

```
#BUILDING THE SARIMA MODEL FOR OFFICE SUPPLIES
tech_model = sm.tsa.statespace.SARIMAX(train3,order=(1,1,1),seasonal_order=(0,1,0,12)).fit()
print(tech_model.summary())
```

```
                                    SARIMAX Results                                     
==========================================================================================
Dep. Variable:                                   y   No. Observations:                   38
Model:             SARIMAX(1, 1, 1)x(0, 1, [], 12)   Log Likelihood                -212.577
Date:                             Tue, 31 Jan 2023   AIC                            431.154
Time:                                     15:06:07   BIC                            434.811
Sample:                                 01-01-2014   HQIC                           432.169
                                      - 02-01-2017                                         
Covariance Type:                               opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.0709      7.925      0.009      0.993     -15.463      15.604
ma.L1         -0.0871      7.867     -0.011      0.991     -15.507      15.333
sigma2      1.283e+06   3.56e+05      3.603      0.000    5.85e+05    1.98e+06
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 1.66
Prob(Q):                              0.98   Prob(JB):                         0.44
Heteroskedasticity (H):               1.50   Skew:                            -0.60
Prob(H) (two-sided):                  0.58   Kurtosis:                         3.40
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
/usr/local/lib/python3.8/dist-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'
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/statespace/sarimax.py:978: UserWarning: Non-invertible starting MA parameters found. Using zeros as starting parameters.
  warn('Non-invertible starting MA parameters found.'
```

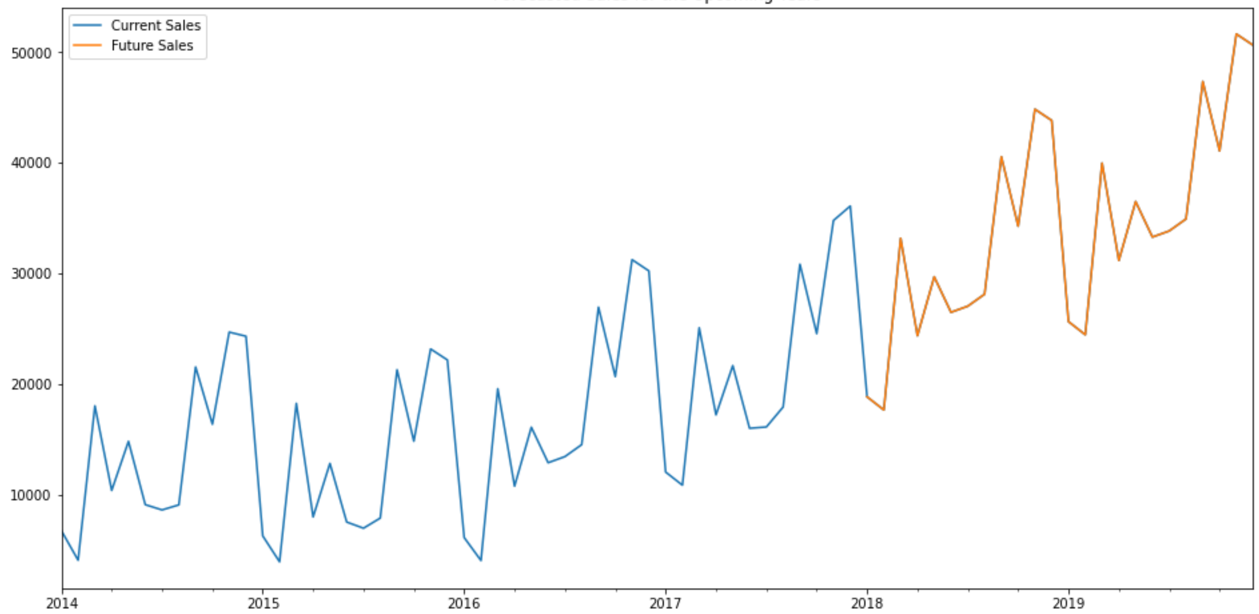
```
#LJUNG BOX TEST
ljungoxtest(tech_model)
```

```
Accept H0, good model
```

```
#PERFORMACE METRICS and PREDICTED DATA
pred3 = tech_model.predict(start=len(train3),end=len(techdata)-1,dynamic=True)
mse3 = mean_squared_error(test3,pred3)
mape3 = mean_absolute_percentage_error(test3,pred3)
plotpred(techdata,pred3)
```


```
#FORECASTING
forecasting(tech_model,techdata)
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/statespace/kalman_filter.py:2290: ValueWarning: Dynamic prediction specified to begin during out-of-sample forecasting period, and so has no
  warn('Dynamic prediction specified to begin during'
```




INTERPRETATIONS:

1. SARIMA MODEL WAS USED HERE FOR FORECASTING DUE TO THE PRESENCE OF SEASONALITY.
2. DATA WAS NOT STATIONARY IN THIS CATEGORY.
3. PATTERNS ARE QUIET SIMILAR TO THE PREVIOUS YEAR SALES RECORD.
4. STRONG UPWARD TREND CAN BE SEEN IN THE OVERALL DATA AND THE DATA FORECASTED.
5. YEAR END SALES IN TECH SUPPLY CATEGORY ARE COMPARITIVELY MORE.

```
frame = {'CATEGORY':['OFFICE SUPPLIES','FURNITURE','TECHNOLOGY'],'RMSE':[np.sqrt(mse1),np.sqrt(mse2),np.sqrt(mse3)],'MAPE':[mape1,mape2,mape3]}
table = pd.DataFrame(frame)
table
```

| | CATEGORY | RMSE | MAPE |
|---|---|---|---|
| **0** | OFFICE SUPPLIES | 6586.551225 | 0.327788 |
| **1** | FURNITURE | 6140.841973 | 0.383515 |
| **2** | TECHNOLOGY | 2721.212192 | 0.113917 |

*MAPE is defined as the percentage of the average of absolute difference between forecasted values and true values, divided by true value.*

*The lower the MAPE, the better the model is.*

# INNOVATION

```
techdata = pd.DataFrame(fdata)
techdata
```

| | 0 |
|---|---|
| **2014-01-01** | 6729.032286 |
| **2014-02-01** | 4051.774224 |
| **2014-03-01** | 18015.409966 |
| **2014-04-01** | 10361.914810 |
| **2014-05-01** | 14801.616282 |
| **...** | ... |
| **2019-08-01** | 34917.944206 |
| **2019-09-01** | 47352.872265 |
| **2019-10-01** | 41075.208297 |
| **2019-11-01** | 51650.875246 |
| **2019-12-01** | 50634.243836 |

72 rows × 1 columns

```
techdata[1] = list(techdata.index)
```

```
techdata
```

| | Technology_Sales | Date |
|---|---|---|
| **2014-01-01** | 6729.032286 | 2014-01-01 |
| **2014-02-01** | 4051.774224 | 2014-02-01 |
| **2014-03-01** | 18015.409966 | 2014-03-01 |
| **2014-04-01** | 10361.914810 | 2014-04-01 |
| **2014-05-01** | 14801.616282 | 2014-05-01 |
| **...** | ... | ... |
| **2019-08-01** | 34917.944206 | 2019-08-01 |
| **2019-09-01** | 47352.872265 | 2019-09-01 |
| **2019-10-01** | 41075.208297 | 2019-10-01 |
| **2019-11-01** | 51650.875246 | 2019-11-01 |

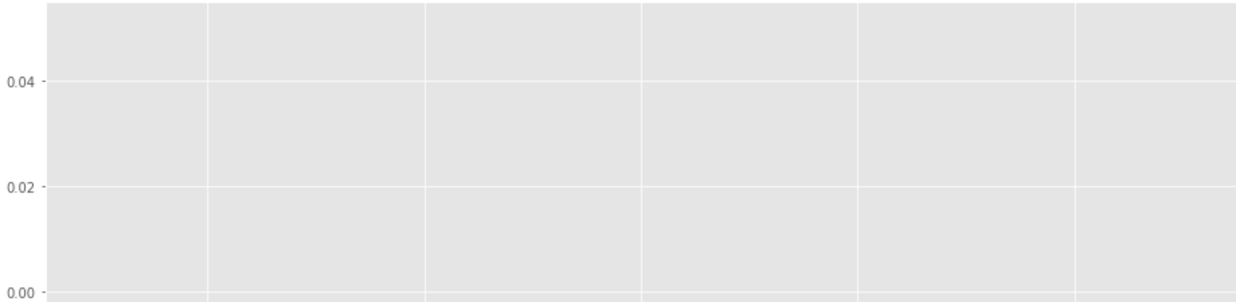
```
techdata.rename(columns={0:'Technology_Sales',1:'Date'},inplace=True)
```

72 rows × 2 columns

```
techdata.to_csv('Technology Sales.csv')
```

```
from matplotlib.animation import FuncAnimation
from itertools import count
x=[]
y=[]
fig,ax = plt.subplots()
ax.plot(x,y)
plt.style.use("ggplot")
counter=count(0,1)

def update(i):
  idx=next(counter)
  x.append(techdata.loc[idx,techdata['Date']])
  y.append(techdata.loc[idx,techdata['Technology_Sales']])
  plt.cla()
  ax.plot(x,y)

ani=FuncAnimation(fig=fig,func=update,interval=200,frames = len(x) + 1)
plt.show()
```

```
Traceback (most recent call last):
  File "/usr/local/lib/python3.8/dist-packages/matplotlib/cbook/__init__.py", line 196, in process
    func(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/matplotlib/animation.py", line 951, in _start
    self._init_draw()
  File "/usr/local/lib/python3.8/dist-packages/matplotlib/animation.py", line 1743, in _init_draw
    self._draw_frame(next(self.new_frame_seq()))
  File "/usr/local/lib/python3.8/dist-packages/matplotlib/animation.py", line 1766, in _draw_frame
    self._drawn_artists = self._func(framedata, *self._args)
  File "<ipython-input-410-b93ab201a488>", line 12, in update
    x.append(techdata.loc[idx,techdata['Date']])
  File "/usr/local/lib/python3.8/dist-packages/pandas/core/indexing.py", line 925, in __getitem__
    return self._getitem_tuple(key)
  File "/usr/local/lib/python3.8/dist-packages/pandas/core/indexing.py", line 1100, in _getitem_tuple
    return self._getitem_lowerdim(tup)
  File "/usr/local/lib/python3.8/dist-packages/pandas/core/indexing.py", line 838, in _getitem_lowerdim
    section = self._getitem_axis(key, axis=i)
  File "/usr/local/lib/python3.8/dist-packages/pandas/core/indexing.py", line 1164, in _getitem_axis
    return self._get_label(key, axis=axis)
  File "/usr/local/lib/python3.8/dist-packages/pandas/core/indexing.py", line 1113, in _get_label
    return self.obj.xs(label, axis=axis)
  File "/usr/local/lib/python3.8/dist-packages/pandas/core/generic.py", line 3776, in xs
    loc = index.get_loc(key)
  File "/usr/local/lib/python3.8/dist-packages/pandas/core/indexes/datetimes.py", line 700, in get_loc
    raise KeyError(key)
KeyError: 0
```




```
f1 = pd.read_csv("/content/Office Suuply Sales.csv",parse_dates=True,usecols=('Office_Supplies_Sales','Date'),index_col='Date')
f2 = pd.read_csv("/content/Furniture Sales.csv",parse_dates=True,usecols=('Furniture_Sales','Date'),index_col='Date')
f3 = pd.read_csv("/content/Technology Sales.csv",parse_dates=True,usecols=('Technology_Sales','Date'),index_col='Date')
```



```
f3
```

| | Technology_Sales |
|---|---|
| **Date** | |
| **2014-01-01** | 6729.032286 |
| **2014-02-01** | 4051.774224 |
| **2014-03-01** | 18015.409966 |
| **2014-04-01** | 10361.914810 |
| **2014-05-01** | 14801.616282 |
| **2019-09-01** | 47352.872265 |

```
f1['Furniture_Sales'] = f2.Furniture_Sales
```

```
f1['Technology_Sales'] = f3.Technology_Sales
```

```
category_forecasts = f1.copy()
```

```
category_forecasts
```

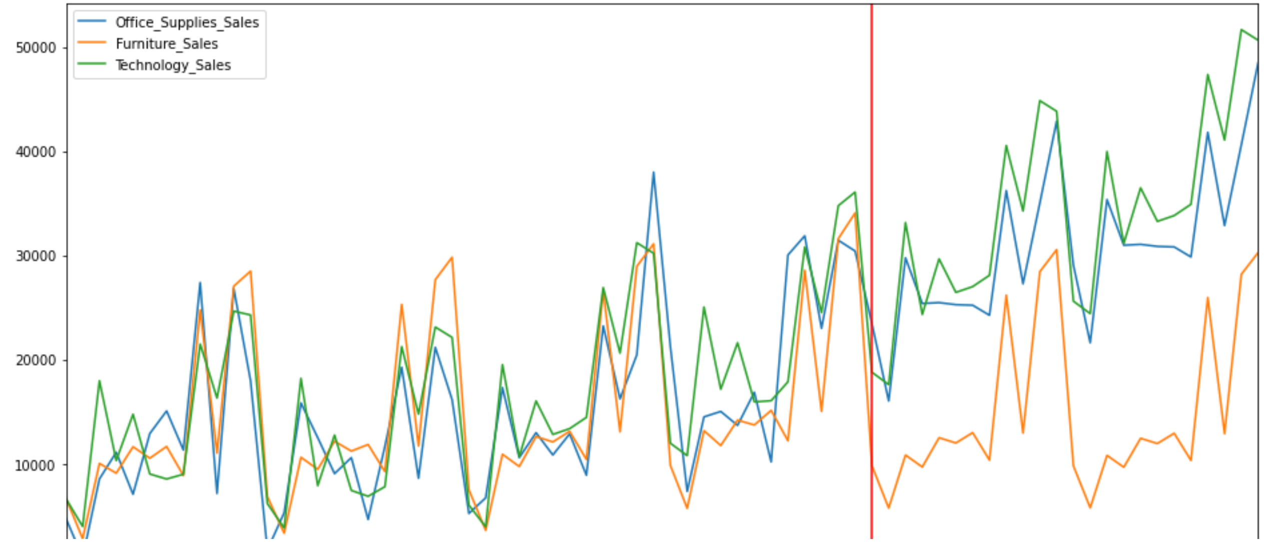
| | Office_Supplies_Sales | Furniture_Sales | Technology_Sales |
|---|---|---|---|
| **Date** | | | |
| **2014-01-01** | 4851.080000 | 6790.384086 | 6729.032286 |
| **2014-02-01** | 1071.724000 | 2880.555398 | 4051.774224 |
| **2014-03-01** | 8605.879000 | 10091.659067 | 18015.409966 |
| **2014-04-01** | 11155.074000 | 9156.845289 | 10361.914810 |
| **2014-05-01** | 7135.624000 | 11687.553932 | 14801.616282 |
| **...** | ... | ... | ... |
| **2019-08-01** | 29869.816649 | 10376.759128 | 34917.944206 |
| **2019-09-01** | 41815.848384 | 25988.638943 | 47352.872265 |
| **2019-10-01** | 32883.562624 | 12937.631855 | 41075.208297 |
| **2019-11-01** | 40620.064180 | 28212.634448 | 51650.875246 |
| **2019-12-01** | 48467.427362 | 30292.353643 | 50634.243836 |

72 rows × 3 columns

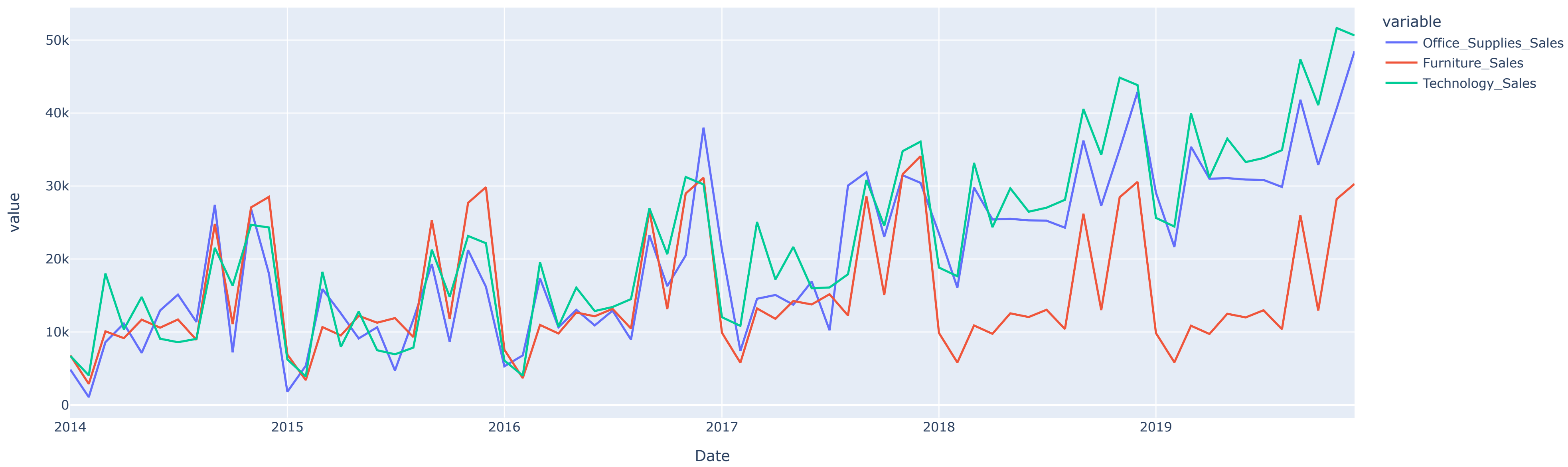
```
category_forecasts.to_csv('category_forecasts.csv')
```

```
category_forecasts.plot()
plt.axvline(x='2018-01-01',c='red')
```

```
<matplotlib.lines.Line2D at 0x7f925f8fbd60>
```




```
import plotly.express as px
fig = px.line(category_forecasts, x=category_forecasts.index, y=category_forecasts.columns)
fig.show()
```



✓ 0s completed at 11:14 PM