I. <u>Code Sheets</u>

The sheets can be divided into mainly three types:

1. Main Sheet

   This sheet is to get the forecast output of all the methods together. Two different models/outputs can be generated. One gives the minimum mean absolute percentage error (mape) of each method along with best configuration. Other model gives the forecast values for the given sales by each method.

2. MovingAvg/Simple Expo/Double Expo/A_Triple Expo/M_tripleExpo/ARIMA_

   These sheets are to get the forecast output of a particular method. The name of the sheet suggests the method used in it to produce the output. All of these sheets give the same two kinds of output as the main sheet one.

3. Graphs

   This sheet is to look at the graphs of each method individually. For each method, graph represents the actual sales value and forecasted values using that particular method.

II. <u>Format of Input File</u>

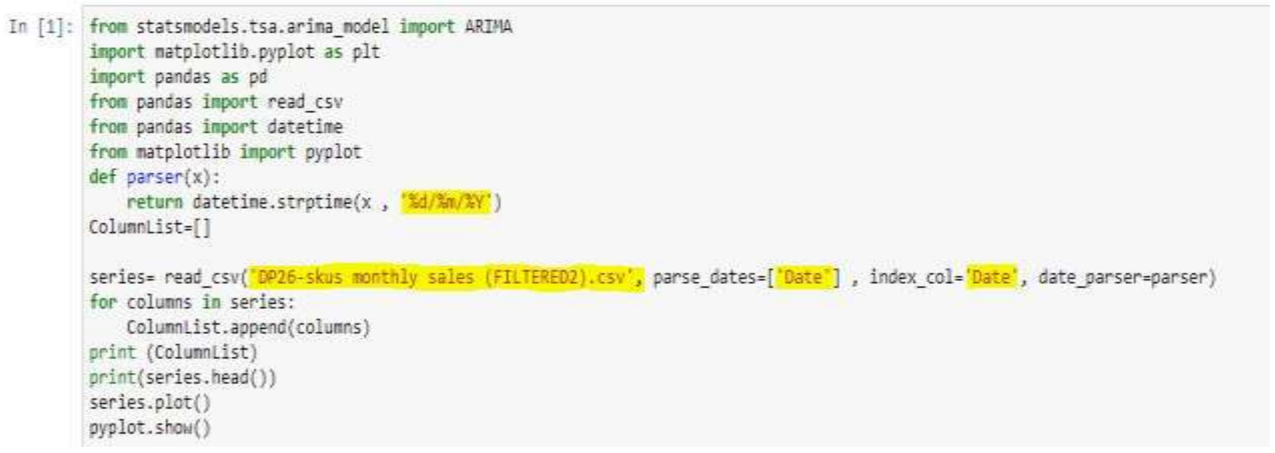
Few formatting styles of the input file need attention before importing it in jupyter notebook to run.

1. The file should have only two types of column. One is the date column and other is the sales data column/columns.
2. Heading of the date column should be written as ‘Date’.
3. The date should be in the format ‘dd/mm/yy’.

III. <u>Saving the Input File</u>

The input file should be saved at the same location where the code sheets are saved. The path of input file and jupyter notebook should be the same. Also, input file has to be saved in csv fomat (.csv extension).

IV. Importing the Input File



```
In [1]: from statsmodels.tsa.arima_model import ARIMA
        import matplotlib.pyplot as plt
        import pandas as pd
        from pandas import read_csv
        from pandas import datetime
        from matplotlib import pyplot
        def parser(x):
            return datetime.strptime(x , '%d/%m/%Y')
        ColumnList=[]

        series= read_csv('DP26-skus monthly sales (FILTERED2).csv', parse_dates=['Date'] , index_col='Date', date_parser=parser)
        for columns in series:
            ColumnList.append(columns)
        print (ColumnList)
        print(series.head())
        series.plot()
        pyplot.show()
```

Every sheet has the first input cell as the one shown above. This cell mainly imports all the necessary modules and the input file. The highlighted text in the picture above is of our concern here.

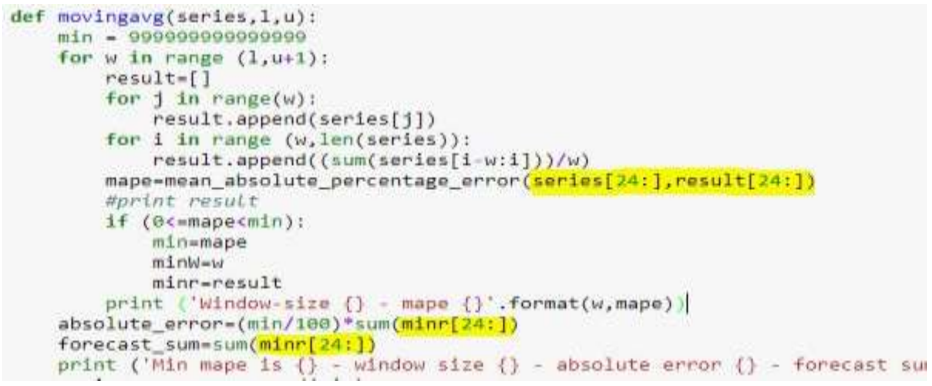
Series=read_csv('Name of the file to be imported',…….) – this is where you import the input file by writing the name under which it is saved along with the extension. Here, 'DP26-skus…..csv' is the name of the file.

Another highlighted text, '%d/%m/%Y', represents the required format of the dates in the input file. Changes with respect to this in the file should to be reflected here too. For example, if the format of date is 30-10-1998 instead of the required 30/10/1998 then accordingly in the code '%d/%m/%Y' should be changed to '%d-%m-%Y'.

'Date' is the next highlighted text in the picture. This has to be in accordance with the name of the date column in the input file. The required format of the name is 'Date'. Any changes in the file lead to changes in here too. For example, if it is named as 'DATE' instead of 'Date' (required) then accordingly 'Date' has to be replaced with 'DATE' everywhere in the code.

V. Changes in the calculation of MAPE

MAPE can either be calculated for a specific period of time or for the entire time period of the dataset. Changes have to be done in the code according to the requirement.



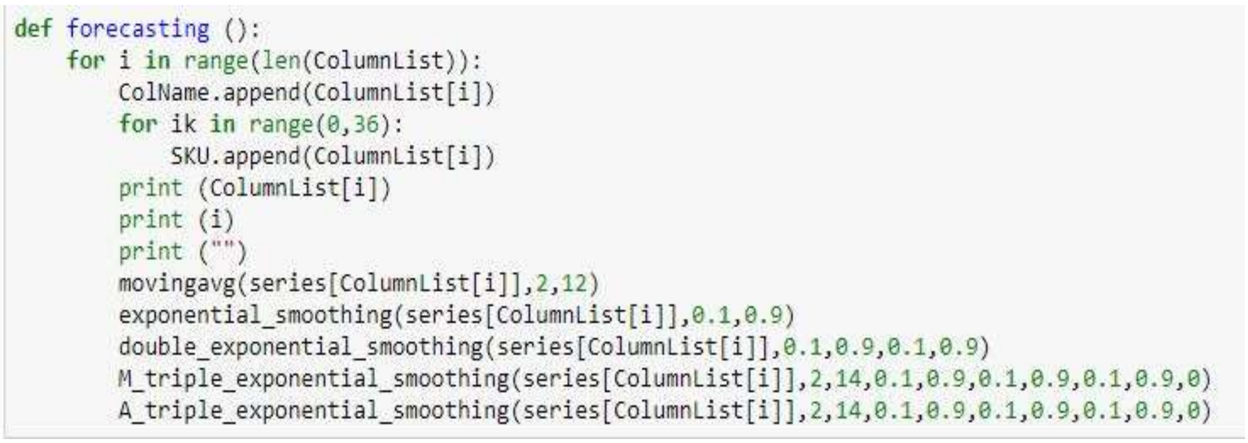
```
def movingavg(series,l,u):
    min = 999999999999999
    for w in range (l,u+1):
        result=[]
        for j in range(w):
            result.append(series[j])
        for i in range (w,len(series)):
            result.append((sum(series[i-w:i]))/w)
        mape=mean_absolute_percentage_error(series[24:],result[24:])
        #print result
        if (0<=mape<min):
            min=mape
            minW=w
            minr=result
        print ('Window-size {} - mape {}'.format(w,mape))
    absolute_error=(min/100)*sum(minr[24:])
    forecast_sum=sum(minr[24:])
    print ('Min mape is {} - window size {} - absolute error {} - forecast su
```

The highlighted text shows the part which has to be changed in the code.

The number 24 in the square brackets tells the index of the row from which mape needs to be calculated. Note indexing in python starts from 0. Index can be calculated from row number in excel (generally add 2 to the row number). Also, same number should be written at all four highlighted places. If mape has to be calculated on the entire dataset with no time period restriction then remove '[number:]' (eg-[24:]) from all four places.

The above picture's first line of code is 'def…..'. In the same manner each method's function starts with the 'def..' code line. Now, these above mentioned mape changes have to be made in each method's function.

VI. <u>Main function</u>



```
def forecasting ():
    for i in range(len(ColumnList)):
        ColName.append(ColumnList[i])
        for ik in range(0,36):
            SKU.append(ColumnList[i])
        print (ColumnList[i])
        print (i)
        print ("")
        movingavg(series[ColumnList[i]],2,12)
        exponential_smoothing(series[ColumnList[i]],0.1,0.9)
        double_exponential_smoothing(series[ColumnList[i]],0.1,0.9,0.1,0.9)
        M_triple_exponential_smoothing(series[ColumnList[i]],2,14,0.1,0.9,0.1,0.9,0.1,0.9,0)
        A_triple_exponential_smoothing(series[ColumnList[i]],2,14,0.1,0.9,0.1,0.9,0.1,0.9,0)
```

This is the main function which gives the desired output. This will give the output for all the skus/ sales columns in the input file. If output for one particular sku is desired then this can be done in two ways:

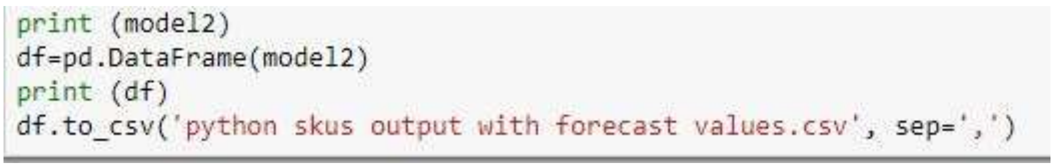
1. Firstly delete the 'for i in range…:' line and 'print (i)' line. Then replace 'ColumnList[i]' with 'series['name of the column/sku']'. For example, if the name of the column which has the sales data of the sku desired is '101. Then 'ColumnList[i]' from everywhere will be replaced by series[''101].
2. Calculate the index of the sku for which output is required. This can again be done through excel. Suppose the index is n. Then replace 'len(ColumnList)' with 'n,n+1'. Rest remains the same.

In line 'for ik in……):' number 36 is relative to input file. The number 36 represents the number of data points in all sales data columns. Thus this number has to be changed according to the data in the input file.

NOTE: All the columns in the input file must be of same length. Make sure every sales column has at least non-zero figures (preferably during the period for which mape is calculated) and as many non-zero figures as possible for the time period for which mape is calculated, at least 2.

VII. Exporting the output file

Two kinds of output will be generated and both will be exported to two different csv files. These files should be saved in the same location where input file and code sheets are saved. These files should pre-exist. Make sure these files are blank otherwise the data in them will be lost as it will be replaced by the output that we are exporting.



```
print (model2)
df=pd.DataFrame(model2)
print (df)
df.to_csv('python skus output with forecast values.csv', sep=',')
```

The name of the file into which the output has to be exported needs to be written in the last line, ‘df.to_csv(‘name of the file.csv’….)’.

Change ‘model2’ to ‘model’ and the name of the file to export other output. Or else copy paste the entire input cell into a new cell, make the changes and then run the cell.

Model has the output of best mape, best configuration etc. for every method. Model 2 has the output of forecast values with min mape for every method.