**Analytics Vidhya Job-a-thon September 2021 Project Approach**

**By Niket khuthia**

**Supplement Sales Prediction**

Your Client WOMart is a leading nutrition and supplement retail chain that offers a comprehensive range of products for all your wellness and fitness needs.   
  
WOMart follows a multi-channel distribution strategy with 350+ retail stores spread across 100+ cities.   
  
Effective forecasting for store sales gives essential insight into upcoming cash flow, meaning WOMart can more accurately plan the cashflow at the store level.  
  
Sales data for 18 months from 365 stores of WOMart is available along with information on Store Type, Location Type for each store, Region Code for every store, Discount provided by the store on every day, Number of Orders everyday etc.

**Your task is to predict the store sales for each store in the test set for the next two months.**

Data Dictionary

Train Data

|  |  |
| --- | --- |
| Variable | Definition |
| ID | Unique Identifier for a row |
| Store\_id | Unique id for each Store |
| Store\_Type | Type of the Store |
| Location\_Type | Type of the location where Store is located |
| Region\_Code | Code of the Region where Store is located |
| Date | Information about the Date |
| Holiday | If there is holiday on the given Date, 1 : Yes, 0 : No |
| Discount | If discount is offered by store on the given Date, Yes/ No |
| #Orders | Number of Orders received by the Store on the given Day |
| Sales | Total Sale for the Store on the given Day |

Test Data

|  |  |
| --- | --- |
| Variable | Definition |
| ID | Unique Identifier for a row |
| Store\_id | Unique id for each Store |
| Store\_Type | Type of the Store |
| Location\_Type | Type of the location where Store is located |
| Region\_Code | Code of the Region where Store is located |
| Date | Information about the Date |
| Holiday | If there is holiday on the given Date, 1 : Yes, 0 : No |
| Discount | If discount is offered by store on the given Date, Yes/ No |

Sample\_Submission

|  |  |
| --- | --- |
| Variable | Definition |
| ID | Unique Identifier for a row |
| Sales | Total Sale for the Store on the given Day |

Evaluation

The evaluation metric for this competition is [MSLE](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_log_error.html" \t "_blank) \* 1000 across all entries in the test set.

Public and Private Split

Test data is further divided into Public (First 20 Days) and Private (Last 41 Days). You will make the prediction for two months (61 days).

The complete project is done with the help of Python codes using Jupyter Notebook in Anaconda

After reading the problem statement, I began with looking at the datasets provided and tried to understand the data and requirements. It seemed that this is a Multivariate Time series Regression Problem with Forecasting and analysis but then realized that the data is not fit for running a time series approach because time is considered as categorical variable because it is repeated due to unique store Id.

After some efforts, I thought of building a multivariate regression model and XGBoost Regressor Model and random forest model to be the best fit.

I chose Linear regression model to move forward because of high Accuracy (All Below steps performed for Other Models too)

The steps included:

-Importing the necessary libraries

-Reading the files into the notebook

-Preparing the data by doing EDA- for cleaning and transforming the data, I took the help of klib library. It was found that there is no missing data or duplicate values and changed the data type of column ‘Date’ from object to data time.

-Doing feature engineering, Date column was split into Year, Month and Day, and dummies were created for the categorical data for the purpose of consideration in multiple linear regression. Also, original date column was dropped

Store ID and ID were also dropped because these features were redundant to the model.

-After that the actual process of model building started, where I assigned X to dependent variables and Y to independent variable. The TRAIN dataset was divided into train and test dataset and the model was trained on 70% of the data. Using this, values were predicted for the rest 30% of the data (test/validating data).

-Here, the #Order column was dropped from the X variable because that was the only factor strongly influencing the Sales and increasing the R2 score, removing it made the prediction of sales because of other categorical variables in the data.

-Intercept and coefficients were predicted and the model was run on the TEST\_FINAL dataset. Performance metrics were calculated (R2 score, RSME and MSLE ) .

**Code**

****

#!/usr/bin/env python  
# coding: utf-8  
  
# In[1]:  
  
  
#pip install Pyforest   
#code for installing required libraries in environment  
  
  
# In[2]:  
  
  
#pip install -U klib  
import klib #Using Klib fot EDA Purpose  
  
  
# In[3]:  
  
  
df=pd.read\_csv("TRAIN.csv") #importing the data  
df\_test\_final=pd.read\_csv("TEST\_FINAL.csv") #importing the data  
  
  
# In[4]:  
  
  
df.head() #Viewing the data  
  
  
# In[5]:  
  
  
df\_test\_final.head() #Viewing the data  
  
  
# In[6]:  
  
  
klib.missingval\_plot(df) # returns a figure containing information about missing values  
  
  
# In[7]:  
  
  
klib.missingval\_plot(df\_test\_final) # returns a figure containing information about missing values  
  
  
# In[8]:  
  
  
df.info() #info before cleaning the data  
  
  
# In[9]:  
  
  
df=klib.data\_cleaning(df) # performs datacleaning (drop duplicates & empty rows/cols, adjust dtypes,...)  
df\_test\_final=klib.data\_cleaning(df\_test\_final) # performs datacleaning (drop duplicates & empty rows/cols, adjust dtypes,...)  
  
  
# In[10]:  
  
  
df.info() #info after cleaning the data  
  
  
# In[11]:  
  
  
df.drop('hash\_order', axis=1, inplace=True)# dropping unwanted column order  
  
  
# In[12]:  
  
  
#df=df.set\_index('date') # setting date column as index  
  
  
# In[13]:  
  
  
df['store\_type'].describe() #viewing Statistics of categorical data and looking for unique values  
  
  
# In[14]:  
  
  
df['location\_type'].describe() #viewing Statistics of categorical data and looking for unique values  
  
  
# In[15]:  
  
  
df['region\_code'].describe() #viewing Statistics of categorical data and looking for unique values  
  
  
# In[16]:  
  
  
df['discount'].describe() #viewing Statistics of categorical data and looking for unique values  
  
  
# In[17]:  
  
  
le = LabelEncoder() #applying label encoder   
df['discount'] = le.fit\_transform(df.discount) #Applying Label encoding to discount column  
  
le = LabelEncoder() #applying label encoder   
df\_test\_final['discount'] = le.fit\_transform(df\_test\_final.discount) #Applying Label encoding to discount column  
  
  
# In[18]:  
  
  
# Splitting Date column  
df['date'] = pd.to\_datetime(df['date'])  
df['year'] = df['date'].dt.year  
df['month'] = df['date'].dt.month  
df['day'] = df['date'].dt.day  
  
df\_test\_final['date'] = pd.to\_datetime(df\_test\_final['date'])  
df\_test\_final['year'] = df\_test\_final['date'].dt.year  
df\_test\_final['month'] = df\_test\_final['date'].dt.month  
df\_test\_final['day'] = df\_test\_final['date'].dt.day  
  
  
# In[19]:  
  
  
df.describe() #viewing Statistics of numerical data and looking for negative values, outlaiers  
  
  
# In[20]:  
  
  
df.corr()  
  
  
# In[21]:  
  
  
# here store type and st  
df.head() #here we see that it is a multi variate time series forecasting problem.   
  
  
# In[22]:  
  
  
# here store type and st  
df\_test\_final.head() #here we see that it is a multi variate time series forecasting problem.   
  
  
# In[23]:  
  
  
df.plot( y='sales', figsize=(20,10))  
  
  
# In[24]:  
  
  
#Setting the value for X and Y  
x = df[['holiday', 'discount', 'year','month','day','store\_type','location\_type','region\_code']]  
y = df['sales']  
  
  
# In[25]:  
  
  
#Applying One hot Encoding on both Test and train Dataset  
  
df = pd.get\_dummies(df,columns=["store\_type","location\_type","region\_code"],drop\_first=True)  
x = pd.get\_dummies(x,columns=["store\_type","location\_type","region\_code"],drop\_first=True)  
df\_test\_final = pd.get\_dummies(df\_test\_final,columns=["store\_type","location\_type","region\_code"],drop\_first=True)  
  
  
# In[26]:  
  
  
df.head()  
  
  
# In[27]:  
  
  
df\_test\_final.head()  
  
  
# In[28]:  
  
  
x.head()  
  
  
# In[29]:  
  
  
y.head()  
  
  
# In[30]:  
  
  
#create train and test split  
from sklearn import model\_selection  
x\_train,x\_test,y\_train,y\_test = model\_selection.train\_test\_split(x,y,test\_size=0.2,random\_state=42)  
  
  
# In[31]:  
  
  
#using train, test datasets  
from sklearn import linear\_model as lm  
reg = lm.LinearRegression(normalize=True)  
reg.fit(x\_train, y\_train)  
  
# print intercept and coefficients  
print(reg.intercept\_)  
print(reg.coef\_)  
  
  
# In[32]:  
  
  
  
#predictions on test dataset  
predictions = reg.predict(x\_test)  
validate = pd.DataFrame({'Actual': y\_test, 'Predicted': predictions})  
  
  
# In[33]:  
  
  
# Evaluating  
from sklearn import metrics  
from sklearn.metrics import mean\_squared\_log\_error  
print('Root Mean Squared Error:',np.sqrt(metrics.mean\_squared\_error(y\_test, predictions)))  
print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, predictions))  
print('Mean Squared Log Error:', mean\_squared\_log\_error(y\_test, predictions)\*1000)  
  
  
# In[34]:  
  
  
import statsmodels.formula.api as smf  
import statsmodels.api as sm  
lm = smf.ols(formula='sales ~ holiday + discount + year + month + day + store\_type\_S2 + store\_type\_S3 + store\_type\_S4 + location\_type\_L2 + location\_type\_L3 + location\_type\_L4 + location\_type\_L5 + region\_code\_R2 + region\_code\_R3 + region\_code\_R4', data=df).fit()  
lm.conf\_int()  
lm.summary()#viewing the metrics of linear regressions  
  
  
# In[35]:  
  
  
predictions = reg.predict(x\_test) #predicting on the test dataset  
  
  
# In[36]:  
  
  
df.head()  
  
  
# In[37]:  
  
  
#Training on the whole model  
reg.fit(x, y)  
  
  
# In[38]:  
  
  
df\_id=df\_test\_final['id'] #copping Id on the final Dataset  
df\_test\_final=df\_test\_final.drop(['store\_id','date','id'], axis=1) #Dropping the columns  
  
  
# In[39]:  
  
  
df\_test\_final.info()  
  
  
# In[40]:  
  
  
predictions = reg.predict(df\_test\_final) #Applying the predictions on the unknown dataset  
  
  
# In[41]:  
  
  
predictions = pd.DataFrame(predictions)#converting into dataframe  
  
predicted\_sales= pd.concat([df\_id, predictions],axis=1) #combining two datasets  
  
predicted\_sales.columns = ['ID', 'Sales'] #renaming Columns  
  
  
# In[42]:  
  
  
predicted\_sales.describe() #checking The predictions  
  
  
# In[43]:  
  
  
y.describe()  
  
  
# In[44]:  
  
  
predicted\_sales.to\_csv('Leanear\_Regression\_prediction.csv',index=False) #getting Output  
  
  
# In[ ]: