PRUDENT TECHOLOGIES AND CONSULTING

HACKATHON 2024

DATA SCIENCE

USE CASE 1.1 Retail Sales Forecasting

By:

R.K.R. Sai Tarun

Innomatics Research Labs

Hyderabad

Problem Statement:

Developing a Regression model for retail sales forecasting.

Note:

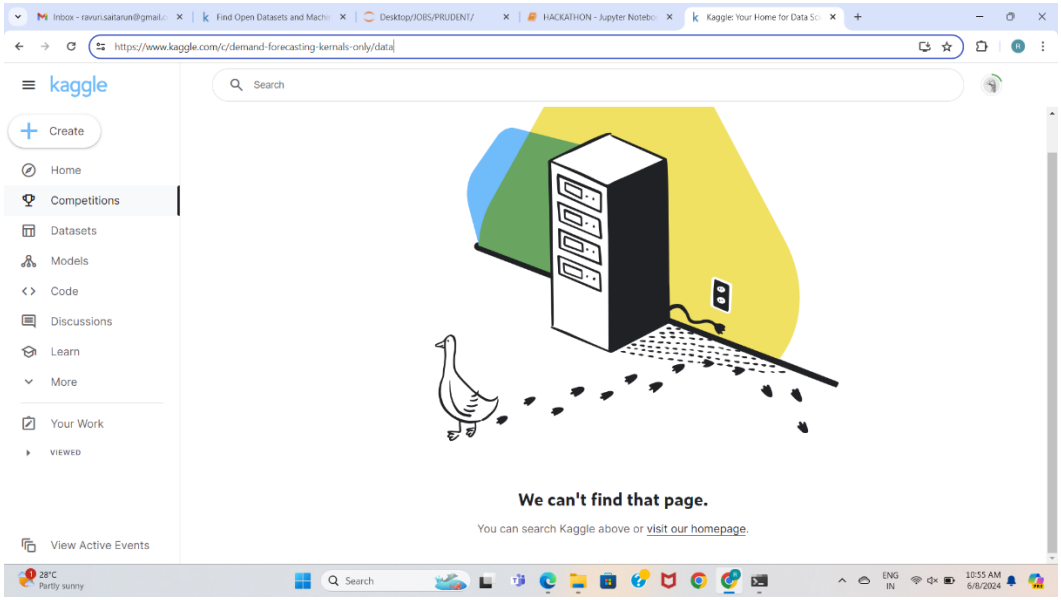
Platform: Jupyter Notebook

File name: Prudent_Hackathon.ipynb

1.Data Collection:

Import all the required packages for the project.

As the dataset from the link provided by the company is not available on Kaggle I choose another dataset from Kaggle and performed the analysis and model building which consists of 5000rows and 14columns.



I also attached the dataset(.csv) file in the drive folder for your reference.

Reading the dataset(.csv) file by using pandas module and performing some basic operations like info, describe and column names. Where i found out that the data set is not processed and it has some null values.

2.Data Cleaning (Preprocessing):

i. Finding Null Values

Seperating the rows which has null values and droping the rows from the original dataset and preparing the final dataframe named df_final.

```
# These are the rows which has order ID has NaN value insted of original value

df[df["Order_ID"].isna()]
```

]:

	Region	Country	Item_Type	Sales_Channel	Order_Priority	Order_Date	Order_ID	Ship_Date	Units_Sold	Unit_SellingPrice	Unit_MakingCost	Total
4997	Asia	Myanmar	Baby Food	Offline	H	11/23/2016	NaN	12-10-2016	5204	255.28	159.42	1
4998	Europe	Finland	Clothes	Online	L	4/22/2014	NaN	05-11-2014	9410	109.28	35.84	1
4999	Sub-Saharan Africa	Tanzania	Vegetables	Online	L	07-08-2011	NaN	08-07-2011	2450	154.06	90.93	

In [6]:

```
#These are the rows which has Item type as NaN value instead of original value

df[df["Item_Type"].isna()]
```

Out[6]:

	Region	Country	Item_Type	Sales_Channel	Order_Priority	Order_Date	Order_ID	Ship_Date	Units_Sold	Unit_SellingPrice	Unit_MakingCost
4980	Asia	Indonesia	NaN	Offline	L	12-05-2010	328383767.0	12/18/2010	4909	81.73	56.67
4981	Sub-Saharan Africa	Eritrea	NaN	Online	H	03-07-2015	354593455.0	04-03-2015	938	651.21	524.96
4982	Middle East and North Africa	Iran	NaN	Offline	H	8/21/2016	265185773.0	10-05-2016	3218	47.45	31.79
4983	Asia	South Korea	NaN	Online	M	07-03-2010	422709548.0	08-04-2010	8004	109.28	35.84
4984	Asia	Cambodia	NaN	Offline	L	10-03-2015	810026109.0	10/18/2015	3581	154.06	90.93
4985	Asia	Laos	NaN	Offline	L	10/13/2010	580309283.0	10/25/2010	5268	437.20	263.33
4986	Asia	Sri Lanka	NaN	Online	C	12-01-2012	307045400.0	12-02-2012	412	651.21	524.96
4987	Europe	Norway	NaN	Offline	M	1/14/2014	634033286.0	1/15/2014	3394	81.73	56.67
4988	Middle East and North Africa	United Arab Emirates	NaN	Online	H	8/31/2014	925384271.0	10-12-2014	625	668.27	502.54
4989	Asia	Thailand	NaN	Online	C	08-08-2012	615522181.0	9/20/2012	5547	152.58	97.44
4990	Europe	Ukraine	NaN	Offline	L	4/14/2014	559183347.0	5/21/2014	3633	205.70	117.11

After Cleaning the dataset the size of data is 4986rows and 14 columns.

ii. Finding Duplicate Values

There are no duplicate values in the dataset

iii. Handling Outliers

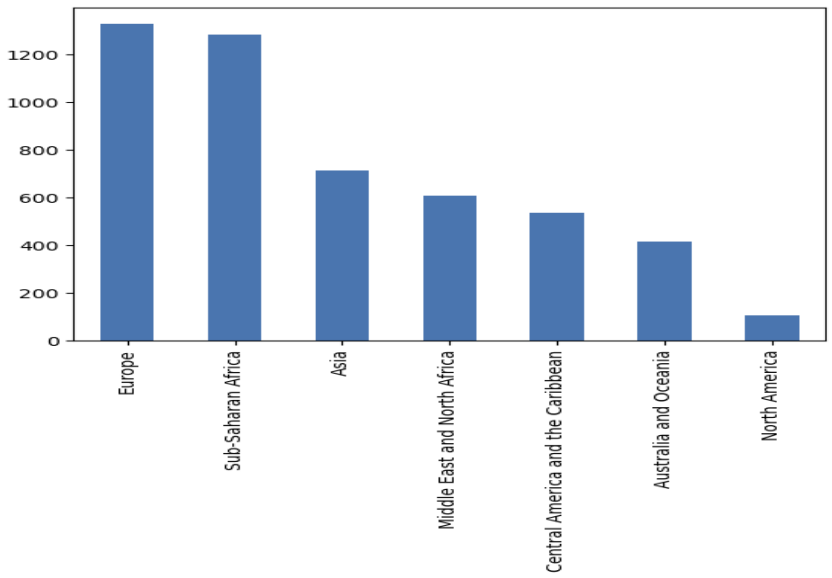
After the data cleaning step here comes the analysis part

3.Exploratory Data Analysis (EDA):

In this section the analysis comes in three parts

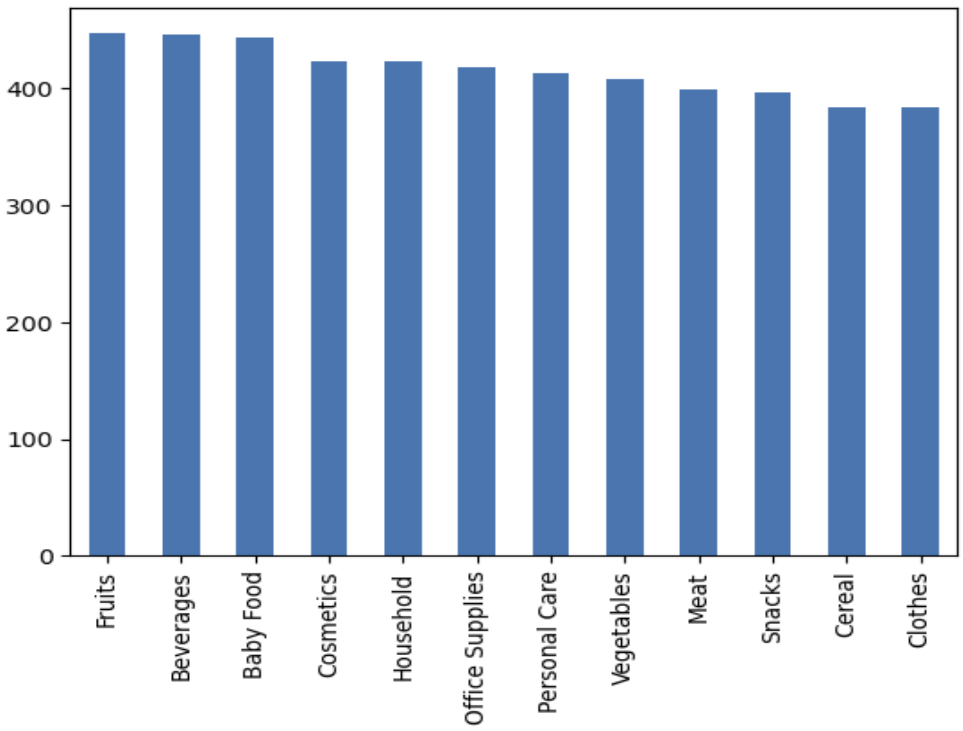
- Univariate Analysis : Analysing a single column
- Bivariate Analysis : Analysing two columns at a time
- MultiVariate Analysis : Analysis multiple columns at a time.

```
In [16]: df_final['Region'].value_counts().plot(kind="bar")
Out[16]: <Axes: >
```



#We are having more number of orders from Europe region and least number of orders from North American region

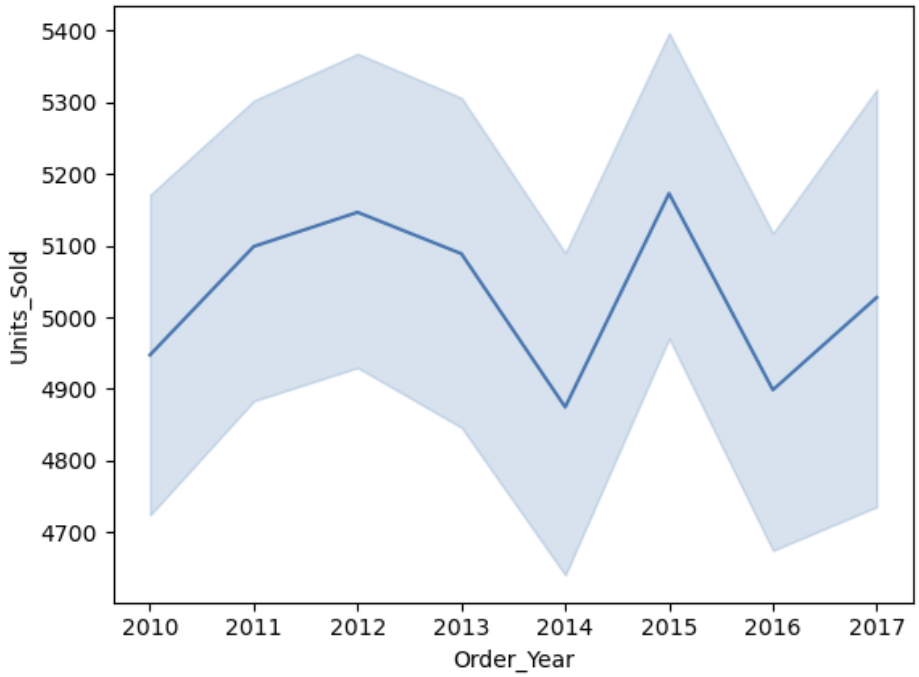
```
In [19]: df_final['Item_Type'].value_counts().plot(kind="bar")
Out[19]: <Axes: >
```



#most ordered item is fruits and less ordered items are clothes and cereal

In [25]:
```
sns.lineplot(x = "Order_Year", y = "Units_Sold", data = df_final)
```

Out[25]: <Axes: xlabel='Order_Year', ylabel='Units_Sold'>

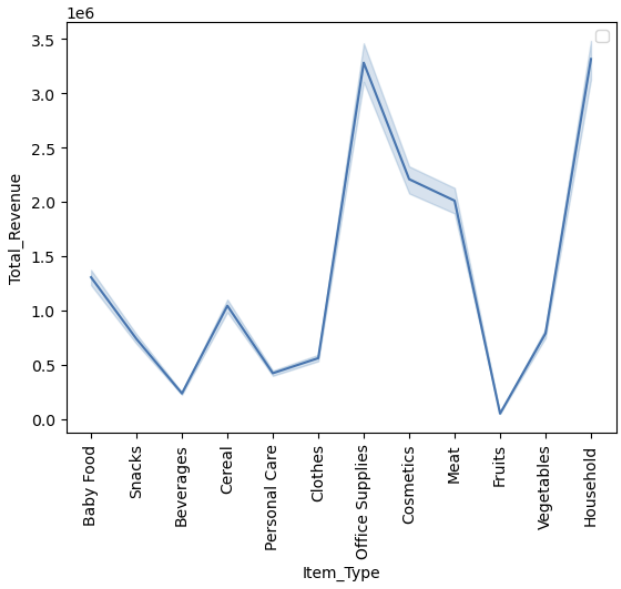


Observing the graph between units sold with respect to year

In [26]:
```
my_plot=sns.lineplot(x = "Item_Type", y = "Total_Revenue", data = df_final)
my_plot.set_xticklabels(my_plot.get_xticklabels(), rotation=90)
plt.legend()
```

```
C:\Users\Tarun\AppData\Local\Temp\ipykernel_20840\3531305193.py:2: UserWarning: FixedFormatter should only be used together
with FixedLocator
  my_plot.set_xticklabels(my_plot.get_xticklabels(), rotation=90)
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when lege
nd() is called with no argument.
```

Out[26]: <matplotlib.legend.Legend at 0x17b2826ac50>

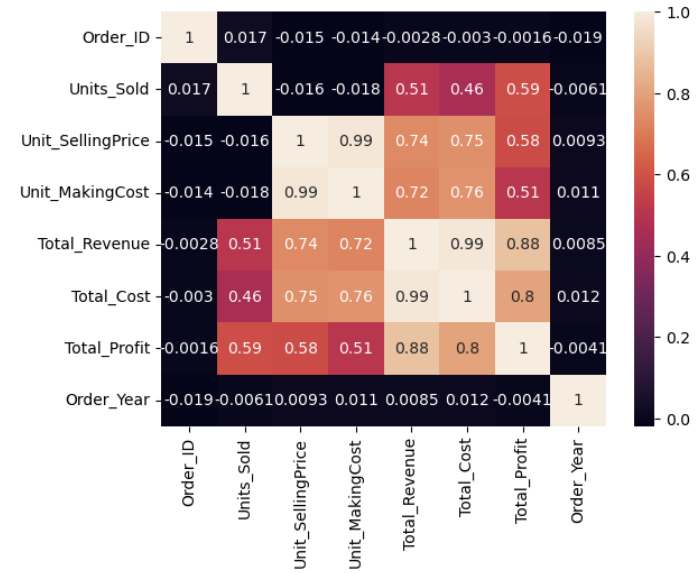


#From the above Line plot we can observe that office supplies and houshold items made the highest revenue

```
In [30]: #Heat map can be used only on numerical data

sns.heatmap(df_final.corr(), annot=True)
plt.show()
```

```
C:\Users\Tarun\AppData\Local\Temp\ipykernel_20840\1194232717.py:3: FutureWarning: The default value of numeric_only in DataF
rame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of nu
meric_only to silence this warning.
  sns.heatmap(df_final.corr(), annot=True)
```



#Heat map shows how columns are related to each other

From the above heatmap we can understand that unit_selling_price is more related to unit_makingcost(0.99)

From the above scatter from we observed that revenue is linear to profit as we can see in heat map both are corelated to each other (0.88)

4. Regression Model Building:

Firstly we are dividing the final dataframe into two different dataframes which consists all the numerical data in on dataframe and categorical data into another dataframe

For training a model we need that dat only in numerical format that is why we divided in two seperate dataframes and then we convert the categorical dataframe into numerical with standard scalar ,onehotencoder and simple imputation methods and creating pipelines for easy use.

Then we combine both the dataframes with a pipeline and then train the model and complete the evaluation metrics

For building the model we split the data into 3 parts x_train,x_cv,x_test

x_train=training model

x_cv=for testing performance of model

x_test = for finding final model R2 Score

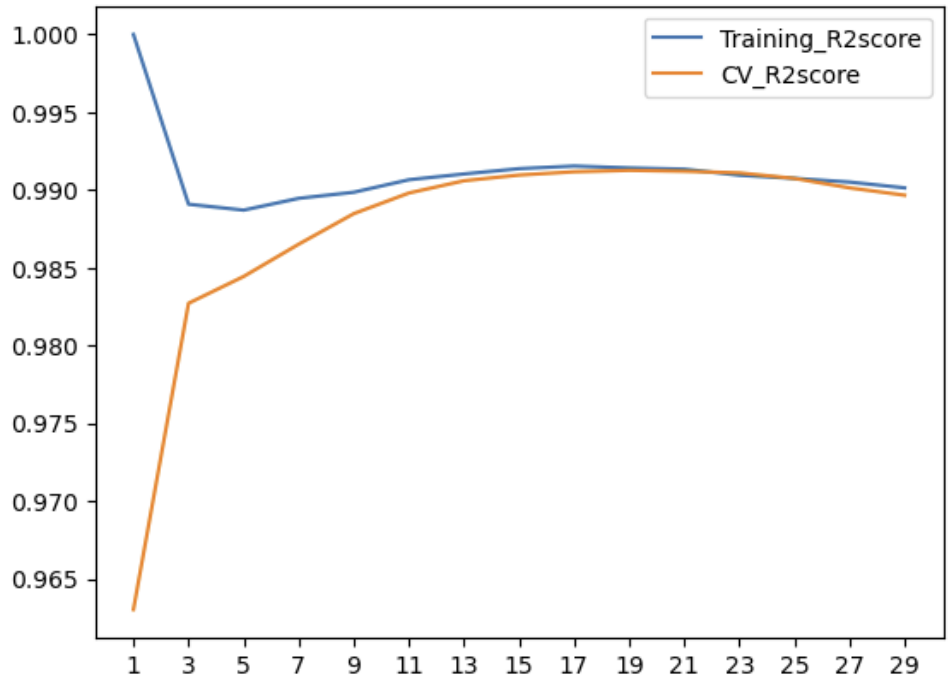
For finding the Genaralized R2 score the model shouldnt seen the data before for that we are splitting the data into two parts first x_train and x_test

Then we never use x_test data till the final model r2 score calculation or best model

For finding best model we again split the x_train data into x_train and x_cv

Ploting the graph between training error and cross validation error for finding the best hyperparameter value

In [59]:
```
plt.plot(range(1,30,2),tr_r2score,label="Training_R2score")
plt.plot(range(1,30,2),cv_r2score,label="CV_R2score")
plt.xticks(range(1,30,2))
plt.legend()
plt.show()
```



We plotted the graph between training r2score and cv r2score

we should choose the model where both the errors are less and the hyperparameter value is best

Because Hyperparameter value less underfitting model

Hyperparameter value high overfitting model

Hyperparameter value middle bestfitting model

So from the above graph we are choosing hyperparameter value as 7

By selecting the hyperparameter value we are building the final model and finding its R2score

```
In [60]: final_r2score=0

         knn=KNeighborsRegressor(n_neighbors=7)
         final_model=knn.fit(x_trainp,y_train)
         predicted_yi=final_model.predict(x_testp)
         final_r2score=r2_score(y_test,predicted_yi)
```

```
In [61]: final_r2score

Out[61]: 0.9851295722862534
```

After the evalution metrics making the model ready for deploying by creating pickle files.

```
In [64]: pickle.dump(ctp,open(r"C:\Users\Tarun\Desktop\JOBS\PRUDENT\Hackathon\CTP.PKL","wb"))
         pickle.dump(model,open(r"C:\Users\Tarun\Desktop\JOBS\PRUDENT\Hackathon\MODEL.PKL","wb"))
```

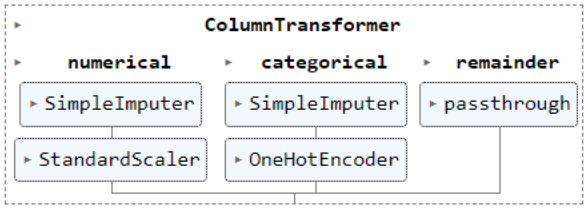
```
In [ ]:
```

```
In [65]: ctp=pickle.load(open(r"C:\Users\Tarun\Desktop\JOBS\PRUDENT\Hackathon\CTP.PKL","rb"))
         model=pickle.load(open(r"C:\Users\Tarun\Desktop\JOBS\PRUDENT\Hackathon\MODEL.PKL","rb"))
```

```
In [ ]:
```

```
In [66]: ctp
```

Out[66]:




```
In [67]: model
```

Out[67]: