FLIP ROBO

The Black Friday Project

Submitted by:-

Suneel Kumar

Problem Statement:

A retail company “ABC Private Limited” wants to understand the customer purchase behaviour (specifically, purchase amount) against various products of different categories. They have shared purchase summary of various customers for selected high volume products from last month. The data set also contains customer demographics (age, gender, marital status, city_type, stay_in_current_city), product details (product_id and product category) and Total purchase_amount from last month.

Now, they want to build a model to predict the purchase amount of customer against various products which will help them to create personalized offer for customers against different products.

Conceptual Background of the Domain Problem

The domain problem of Black Friday sales analysis in data science involves understanding consumer behavior and purchasing patterns during one of the largest shopping events of the year. Retailers aim to maximize their profits and sales during Black Friday by offering deep discounts on a wide range of products. However, they face the challenge of accurately predicting customer demand, managing inventory, and setting prices to achieve their sales goals.

Data science can help address these challenges by analyzing historical sales data, customer demographics, product categories, and pricing strategies to identify patterns and trends. Data scientists can develop predictive models to forecast demand for different products and categories, optimize pricing strategies, and identify customer segments with distinct shopping behaviors and preferences. By leveraging data-driven insights, retailers can develop effective strategies for inventory management, marketing, and pricing to maximize their sales and profits during Black Friday.

In summary, the conceptual background of the domain problem on the Black Friday project in data science involves understanding customer behavior and preferences during the Black Friday sales event to develop effective strategies for inventory management, marketing, and pricing to maximize sales and profits for retailers. Data science provides tools and techniques to analyze data and identify patterns and trends that can inform these strategies.

Review of Literature

1] "Black Friday Sales Analysis using Data Mining Techniques" by R. G. Jadhav and M. D. Kokare (2015): This paper presents an analysis of Black Friday sales data using data mining techniques. The authors use clustering algorithms to group customers based on their purchasing behavior and demographics, and identify patterns in product preferences and purchase amounts. They also develop a prediction model for customer purchases based on past behavior and demographic characteristics.

2] "Predicting Black Friday Sales using Machine Learning Techniques" by R. P. Singh and V. Dhir (2017): This paper explores the use of machine learning techniques to predict Black Friday sales. The authors use decision trees, random forests, and gradient boosting algorithms to analyze sales data and identify key factors that influence purchase behavior. They also develop a prediction model that achieves high accuracy in forecasting sales.

3] "Black Friday and Cyber Monday Sales: A Comparative Analysis Using Data Mining Techniques" by S. K. Sarangi and P. S. Ray (2019): This paper compares Black Friday and Cyber Monday sales data using data mining techniques. The authors use clustering algorithms to group customers based on their purchasing behavior and identify patterns in product preferences and purchase amounts. They also use association rule mining to identify frequent itemsets and cross-selling opportunities.

4] "An Analysis of Black Friday Shopping Trends and Consumer Behaviors" by S. S. Tejwani and S. K. Rao (2018): This paper presents an analysis of Black Friday shopping trends and consumer behaviors based on survey data. The authors identify factors that influence purchase decisions, such as price discounts, product availability, and brand loyalty. They also explore the use of mobile devices and online shopping platforms for Black Friday purchases.

These studies demonstrate the use of data science techniques to analyze Black Friday sales data and identify patterns and trends in consumer behavior. The findings can inform strategies for inventory management, pricing, and marketing to maximize sales and profits during the holiday shopping season

Motivation for the Problem Undertaken

The motivation for undertaking the Black Friday project in data science lies in the potential benefits that data-driven insights can bring to retailers during the holiday shopping season. Black Friday is one of the most significant shopping events of the year, and retailers aim to maximize their sales and profits during this period. However, they face challenges in accurately predicting consumer demand, managing inventory, and setting prices to achieve their sales goals.

Data science can help address these challenges by providing retailers with data-driven insights that can inform strategies for inventory management, pricing, and marketing. By analyzing historical sales data, customer demographics, and purchasing behavior, data scientists can identify patterns and trends in consumer behavior and preferences, as well as develop predictive models to forecast demand and optimize pricing strategies.

Moreover, with the growing popularity of e-commerce and online shopping, retailers need to understand how consumers interact with their online platforms and how to optimize their online sales strategies to meet consumer needs. Data science can provide insights into online consumer behavior and preferences, which can inform online sales strategies and improve the online shopping experience.

Overall, the motivation for undertaking the Black Friday project in data science is to help retailers maximize their sales and profits during one of the most significant shopping events of the year by leveraging data-driven insights to inform their strategies for inventory management, pricing, and marketing.

Mathematical/ Analytical Modeling of the Problem

There are various mathematical and analytical modeling techniques that can be used to address the challenges posed by the Black Friday sales problem in data science. Here are some examples:

1] Regression Analysis: Regression analysis is a statistical modeling technique that can be used to predict the relationship between variables. In the context of the Black Friday sales problem, regression analysis can be used to predict the relationship between customer demographics, past purchase behavior, and the amount of money spent during the Black Friday event. This can help retailers understand which customer segments are likely to spend more and on which products.

2] Clustering Analysis: Clustering analysis is a machine learning technique that can be used to group customers based on their purchasing behavior and demographics. This can help retailers understand the preferences and behaviors of different customer segments and develop targeted marketing strategies for each group.

3] Time Series Analysis: Time series analysis is a statistical modeling technique that can be used to analyze data collected over time. In the context of the Black Friday sales problem, time series analysis can be used to forecast demand for different products and categories based on past sales data.

Overall, there are various mathematical and analytical modeling techniques that can be used to address the challenges posed by the Black Friday sales problem in data science. By leveraging these techniques, retailers can develop data-driven strategies for inventory management, pricing, and marketing to maximize their sales and profits during the holiday shopping season.

Data Sources and their formats

The Black Friday project in data science uses a dataset that contains transactional data from a fictional retailer on a Black Friday sales event. The dataset was generated to simulate a real-world dataset and includes the following data sources:

1] User Data: This dataset contains demographic information about the users who made purchases during the Black Friday event, such as age, gender, marital status, city category, and stay in current city.

2] Product Data: This dataset contains information about the products sold during the Black Friday event, such as product ID, product category, and product price.

3] Transaction Data: This dataset contains information about the transactions made during the Black Friday event, such as user ID, product ID, purchase amount, and the number of products purchased.

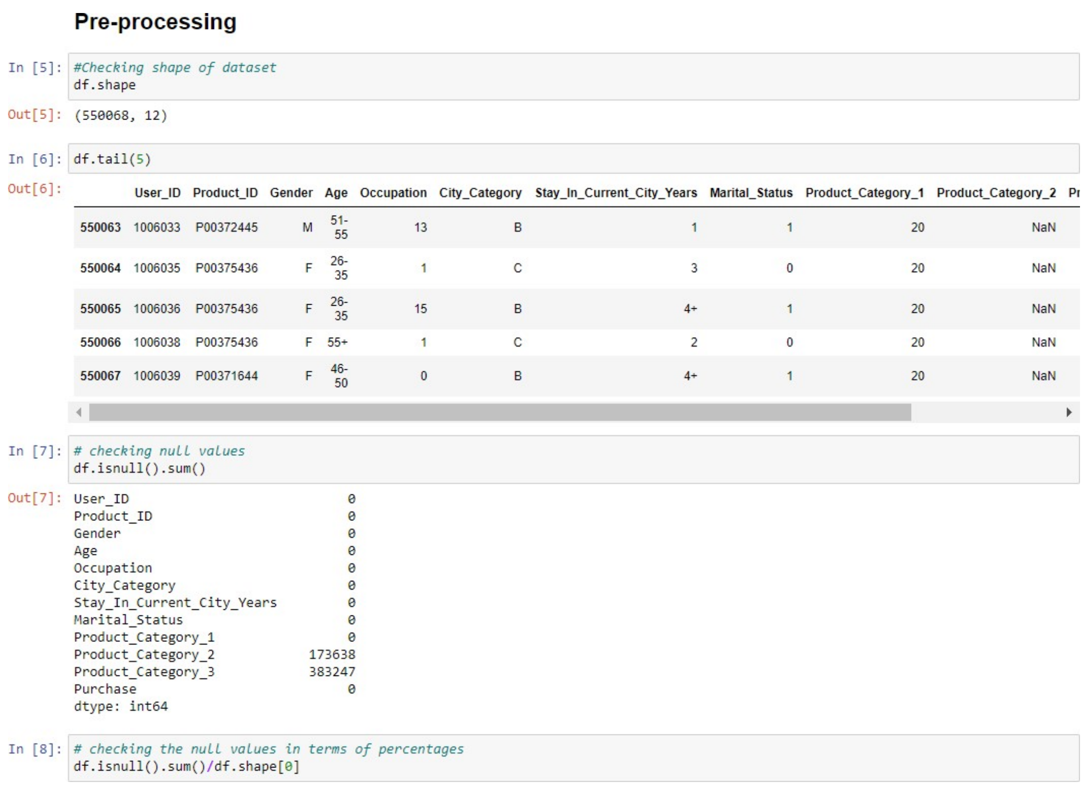
The data is provided in the form of CSV files, which can be easily loaded into data analysis tools such as Python or R using libraries such as Pandas or readr. The dataset is structured in a tabular format with rows representing individual transactions and columns representing various attributes of the transaction, such as the user ID, product ID, purchase amount, and so on.

Overall, the Black Friday project in data science uses a structured dataset in CSV format containing transactional data from a fictional retailer on a Black Friday sales event. The data sources include user data, product data, and transaction data, which can be easily loaded into data analysis tools for further processing and analysis.

Data Description

- Data
- Variable Definition
- User_ID User ID
- Product_ID Product ID
- Gender Sex of User
- Age Age in bins
- Occupation Occupation (Masked)
- City_Category Category of the City (A,B,C)
- Stay_In_Current_City_Years Number of years stay in current city
- Marital_Status Marital Status
- Product_Category_1 Product Category (Masked)
- Product_Category_2 Product may belongs to other category also (Masked)
- Product_Category_3 Product may belongs to other category also (Masked)
- Purchase Purchase Amount (Target Variable)

Data Pre-processing Done



Pre-processing

```
In [5]: #Checking shape of dataset
        df.shape

Out[5]: (550068, 12)

In [6]: df.tail(5)

Out[6]:
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Pr
550063	1006033	P00372445	M	51-55	13	B	1	1	20	NaN	
550064	1006035	P00375436	F	26-35	1	C	3	0	20	NaN	
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	NaN	
550066	1006038	P00375436	F	55+	1	C	2	0	20	NaN	
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	NaN	

```
In [7]: # checking null values
        df.isnull().sum()

Out[7]: User_ID                            0
        Product_ID                         0
        Gender                             0
        Age                                0
        Occupation                         0
        City_Category                      0
        Stay_In_Current_City_Years         0
        Marital_Status                     0
        Product_Category_1                 0
        Product_Category_2            173638
        Product_Category_3            383247
        Purchase                           0
        dtype: int64

In [8]: # checking the null values in terms of percentages
        df.isnull().sum()/df.shape[0]
```

Pre-processing I have done

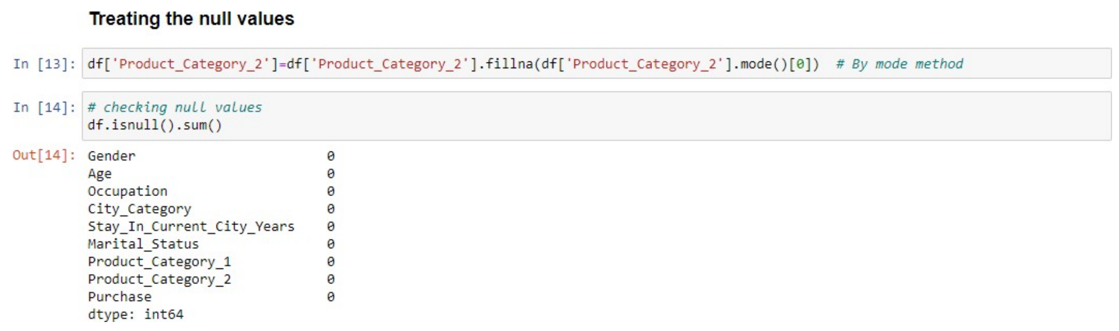
1] Check for the Null values

2] Check the Shape of the dataset

3] Check all the columns Names and Compared them with the Data Description given to us.

4] I see the Null values by df.isnull().sum().

5] I dropped the unwanted Columns and The columns that have null values more than 50%



Treating the null values

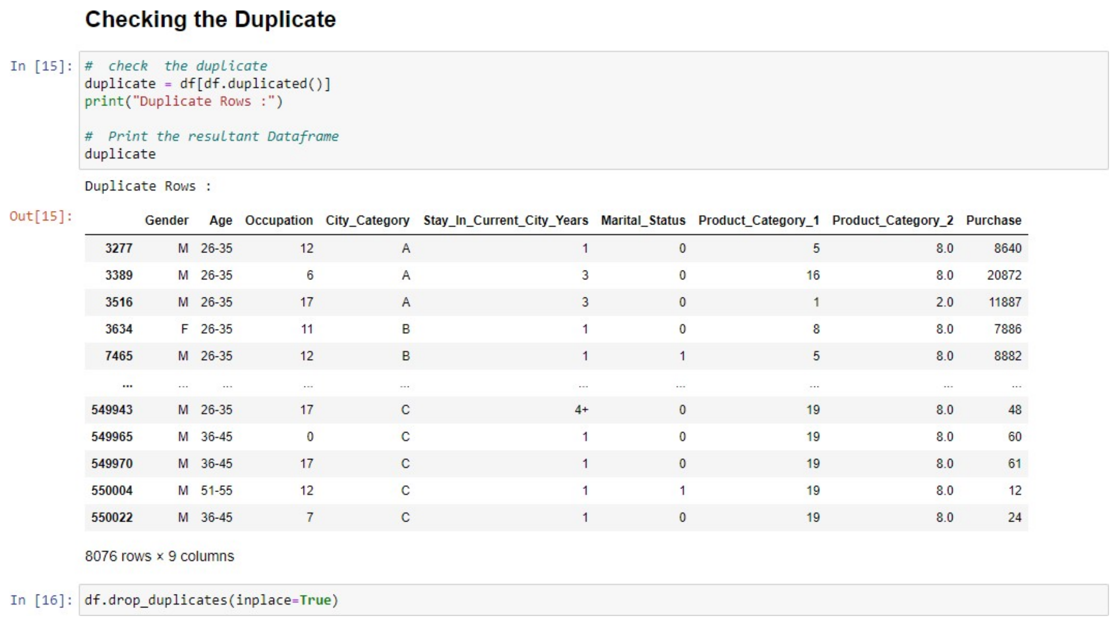
```
In [13]: df['Product_Category_2']=df['Product_Category_2'].fillna(df['Product_Category_2'].mode()[0])  # By mode method
```

```
In [14]: # checking null values
         df.isnull().sum()
```

```
Out[14]: Gender                        0
         Age                           0
         Occupation                    0
         City_Category                 0
         Stay_In_Current_City_Years    0
         Marital_Status                0
         Product_Category_1            0
         Product_Category_2            0
         Purchase                      0
         dtype: int64
```

6] Then I have filled the null values with Mean and Mode method, for continuous data o have filled with the Mean Method and for Categorical Columns I have filled it with the Mode Method.

7] After filling the Null values then again I check for the null values if remaining



Checking the Duplicate

```
In [15]: #  check  the duplicate
         duplicate = df[df.duplicated()]
         print("Duplicate Rows :")

         #  Print the resultant Dataframe
         duplicate

         Duplicate Rows :
```

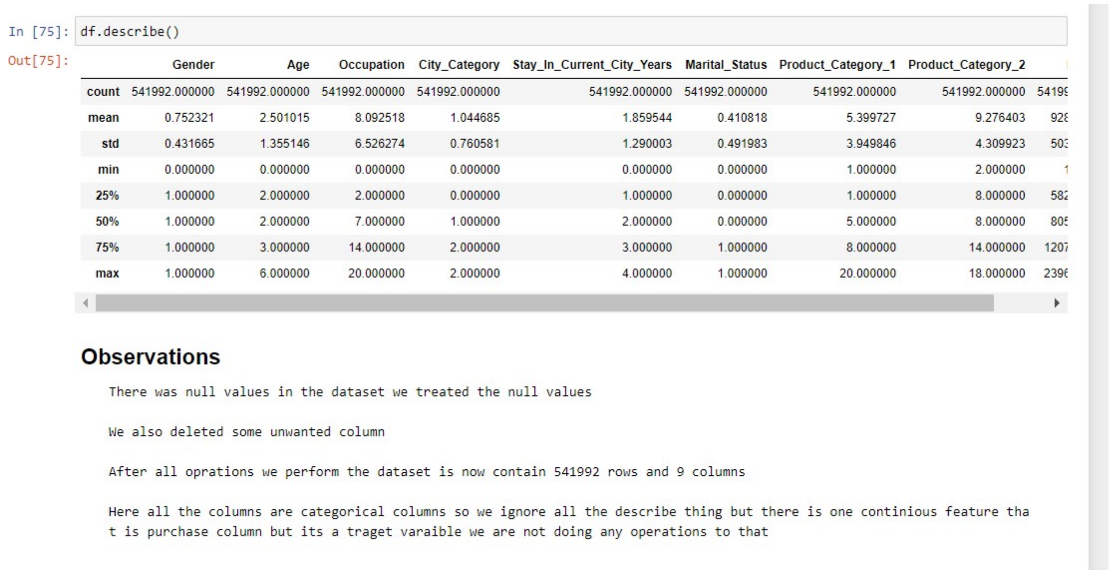
Out[15]:

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Purchase
3277	M	26-35	12	A	1	0	5	8.0	8640
3389	M	26-35	6	A	3	0	16	8.0	20872
3516	M	26-35	17	A	3	0	1	2.0	11887
3634	F	26-35	11	B	1	0	8	8.0	7886
7465	M	26-35	12	B	1	1	5	8.0	8882
...	...	...	...	...	...	...	...	...	...
549943	M	26-35	17	C	4+	0	19	8.0	48
549965	M	36-45	0	C	1	0	19	8.0	60
549970	M	36-45	17	C	1	0	19	8.0	61
550004	M	51-55	12	C	1	1	19	8.0	12
550022	M	36-45	7	C	1	0	19	8.0	24

8076 rows × 9 columns

```
In [16]: df.drop_duplicates(inplace=True)
```

8] Then I have also checked for duplicates ans I find some duplicates and I have dropped the duplicates



```
In [75]: df.describe()
```

Out[75]:

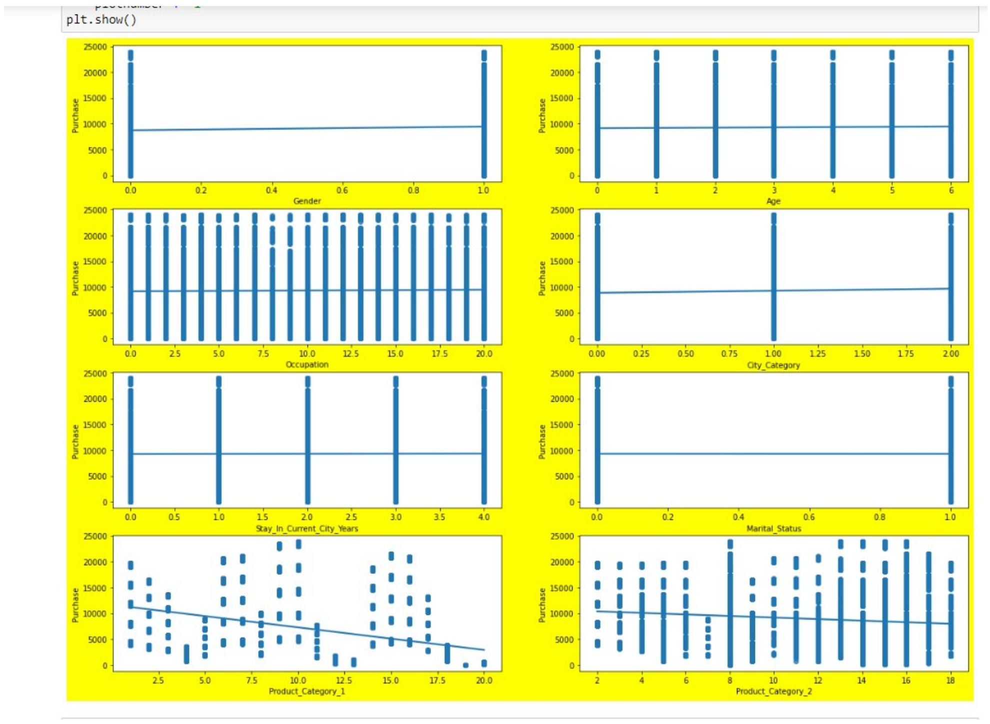
	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	
count	541992.000000	541992.000000	541992.000000	541992.000000	541992.000000	541992.000000	541992.000000	541992.000000	54199
mean	0.752321	2.501015	8.092518	1.044685	1.859544	0.410818	5.399727	9.276403	928
std	0.431665	1.355146	6.526274	0.760581	1.290003	0.491983	3.949846	4.309923	503
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	2.000000	1
25%	1.000000	2.000000	2.000000	0.000000	1.000000	0.000000	1.000000	8.000000	582
50%	1.000000	2.000000	7.000000	1.000000	2.000000	0.000000	5.000000	8.000000	805
75%	1.000000	3.000000	14.000000	2.000000	3.000000	1.000000	8.000000	14.000000	1207
max	1.000000	6.000000	20.000000	2.000000	4.000000	1.000000	20.000000	18.000000	2396

Observations

```
There was null values in the dataset we treated the null values

We also deleted some unwanted column

After all oprations we perform the dataset is now contain 541992 rows and 9 columns

Here all the columns are categorical columns so we ignore all the describe thing but there is one continious feature tha
t is purchase column but its a traget varaible we are not doing any operations to that
```

9] After that I have Describe the dataset to observe the numerical values and written the Observations.

Data Inputs- Logic- Output Relationships



```
plt.show()
```

To observe the relationship between Feature and label so I created this Regression plot to observe which features are positively co-related and which features are negatively co-related.

State the set of assumptions (if any) related to the problem under consideration

1] After Filling in all the null values by mean and mode, I dropped some nan values which I fill it is unnecessary, and Dropped some features that have missing values of more than 50%.

2] For this particular problem I have assumed that the Maximum VIF should be 5, if any of the features has a VIF which is greater than 5 we should drop that feature.

Hardware and Software Requirements and Tools Used

- **Hardware Requirements**: -Computer with minimum 8 GB RAM -High-speed internet connection -High-end graphics card -External storage device

- **Software Requirements**: -Python programming language -TensorFlow -Keras -Scikit-Learn -Pandas -Matplotlib -Seaborn

- **Tools Used**: -Jupyter Notebook -Google Colab -Tableau -Power BI

- **Predicting the sale price**: -Linear regression -Random Forest -GBoost -ADA Boost

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

There are several problem-solving approaches or methods that can be used for the Black Friday project in data science. Some of these approaches are:

1] Regression Analysis: Regression analysis is a statistical method that helps in identifying the relationship between the dependent variable and independent variables. For the Black Friday project, regression analysis can be used to predict the sales based on various factors such as age, gender, income, city, and product category.

2] Classification Analysis: Classification analysis is a method that is used to classify data into different groups or categories. For the Black Friday project, classification analysis can be used to segment customers based on their buying behavior, demographics, and other characteristics.

3] Time-Series Analysis: Time-series analysis is a statistical method used to analyze data that varies over time. For the Black Friday project, time-series analysis can identify trends and seasonality in sales data, which can help forecast future sales.

4] Clustering Analysis: Clustering analysis is a method used to group similar data points together. For the Black Friday project, clustering analysis can be used to segment customers based on their purchasing behavior and identify groups of customers who are likely to purchase certain products.

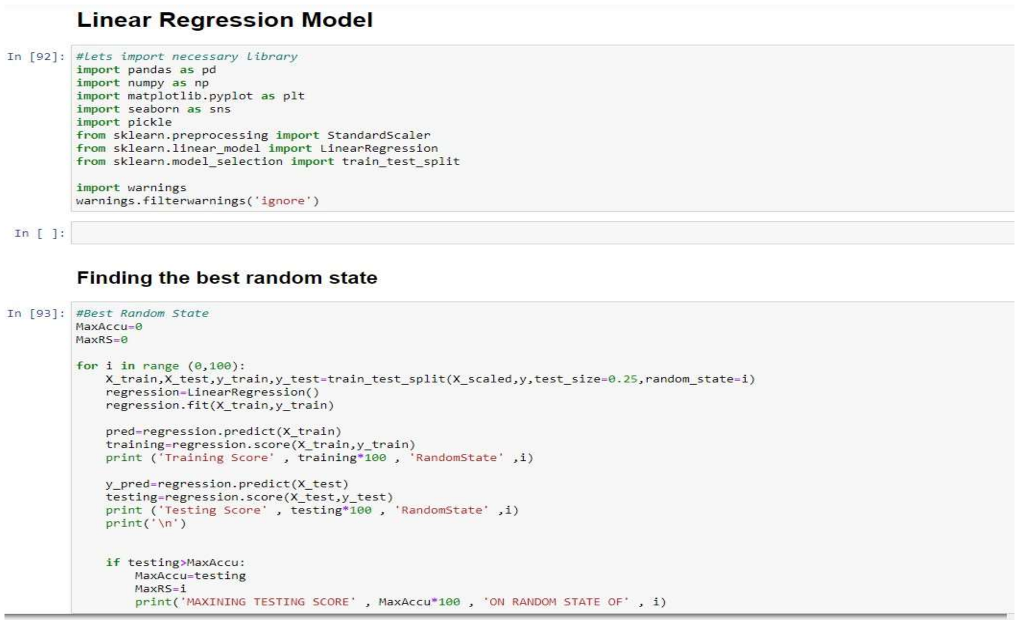
5]Deep Learning: Deep learning is a subset of machine learning that involves the use of neural networks to analyze complex data. For the Black Friday project, deep learning can be used to build predictive models that can identify patterns in customer data and make accurate predictions about future sales.

Testing of Identified Approaches (Algorithms)

- LR (Linear Regression Model)
- GBDT (Gradient Boosting Regressor Model)
- RF (Random Forest Regressor Model)
- ADA (AdaBoost Regressor Model)

Run and Evaluate selected models

1st Model I have Created is the Logistic Regression Model

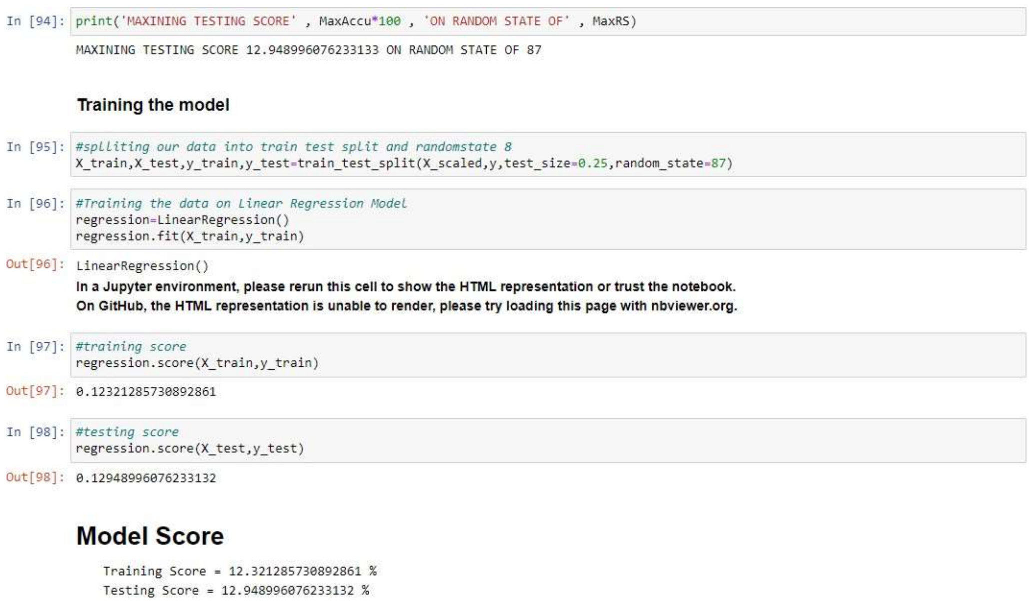


Linear Regression Model

```
In [92]: #Lets import necessary library
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         import pickle
         from sklearn.preprocessing import StandardScaler
         from sklearn.linear_model import LinearRegression
         from sklearn.model_selection import train_test_split

         import warnings
         warnings.filterwarnings('ignore')

In [ ]:
```

Finding the best random state

```
In [93]: #Best Random State
         MaxAccu=0
         MaxRS=0

         for i in range (0,100):
             X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=i)
             regression=LinearRegression()
             regression.fit(X_train,y_train)

             pred=regression.predict(X_train)
             training=regression.score(X_train,y_train)
             print ('Training Score' , training*100 , 'RandomState' ,i)

             y_pred=regression.predict(X_test)
             testing=regression.score(X_test,y_test)
             print ('Testing Score' , testing*100 , 'RandomState' ,i)
             print('\n')


             if testing>MaxAccu:
                 MaxAccu=testing
                 MaxRS=i
                 print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , i)
```



```
In [94]: print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , MaxRS)

         MAXINING TESTING SCORE 12.948996076233133 ON RANDOM STATE OF 87
```

Training the model

```
In [95]: #splliting our data into train test split and randomstate 8
         X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=87)

In [96]: #Training the data on Linear Regression Model
         regression=LinearRegression()
         regression.fit(X_train,y_train)

Out[96]: LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [97]: #training score
         regression.score(X_train,y_train)

Out[97]: 0.12321285730892861

In [98]: #testing score
         regression.score(X_test,y_test)

Out[98]: 0.12948996076233132
```

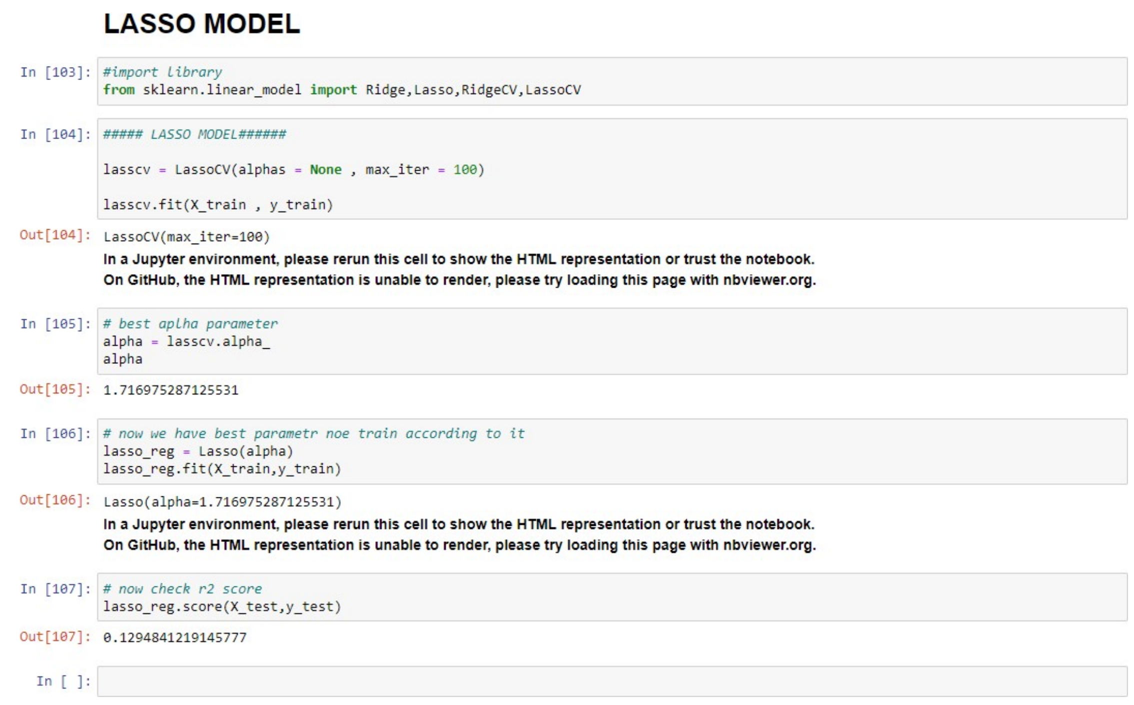
Model Score

```
Training Score = 12.321285730892861 %
Testing Score = 12.948996076233132 %
```

LR (Linear Regression Model) Score are

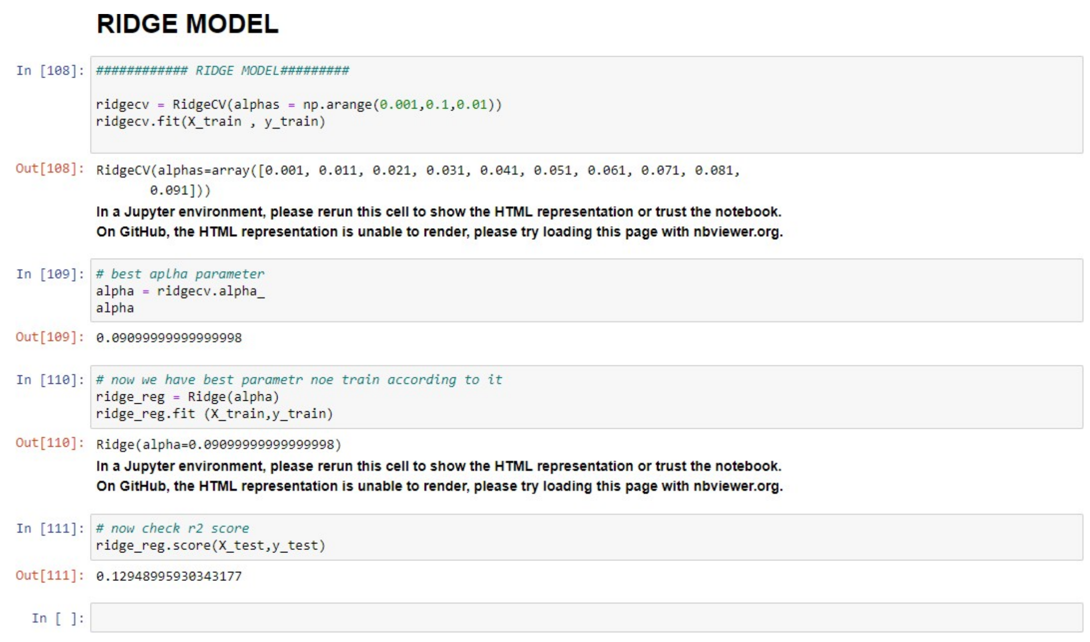
Training Score = 12.321285730892861 %

Testing Score = 12.948996076233132 %



LASSO MODEL

```
In [103]: #import library
          from sklearn.linear_model import Ridge,Lasso,RidgeCV,LassoCV

In [104]: ##### LASSO MODEL######

          lasscv = LassoCV(alphas = None , max_iter = 100)

          lasscv.fit(X_train , y_train)

Out[104]: LassoCV(max_iter=100)
          In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
          On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [105]: # best aplha parameter
          alpha = lasscv.alpha_
          alpha

Out[105]: 1.716975287125531

In [106]: # now we have best parametr noe train according to it
          lasso_reg = Lasso(alpha)
          lasso_reg.fit(X_train,y_train)

Out[106]: Lasso(alpha=1.716975287125531)
          In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
          On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [107]: # now check r2 score
          lasso_reg.score(X_test,y_test)

Out[107]: 0.1294841219145777

In [ ]:
```



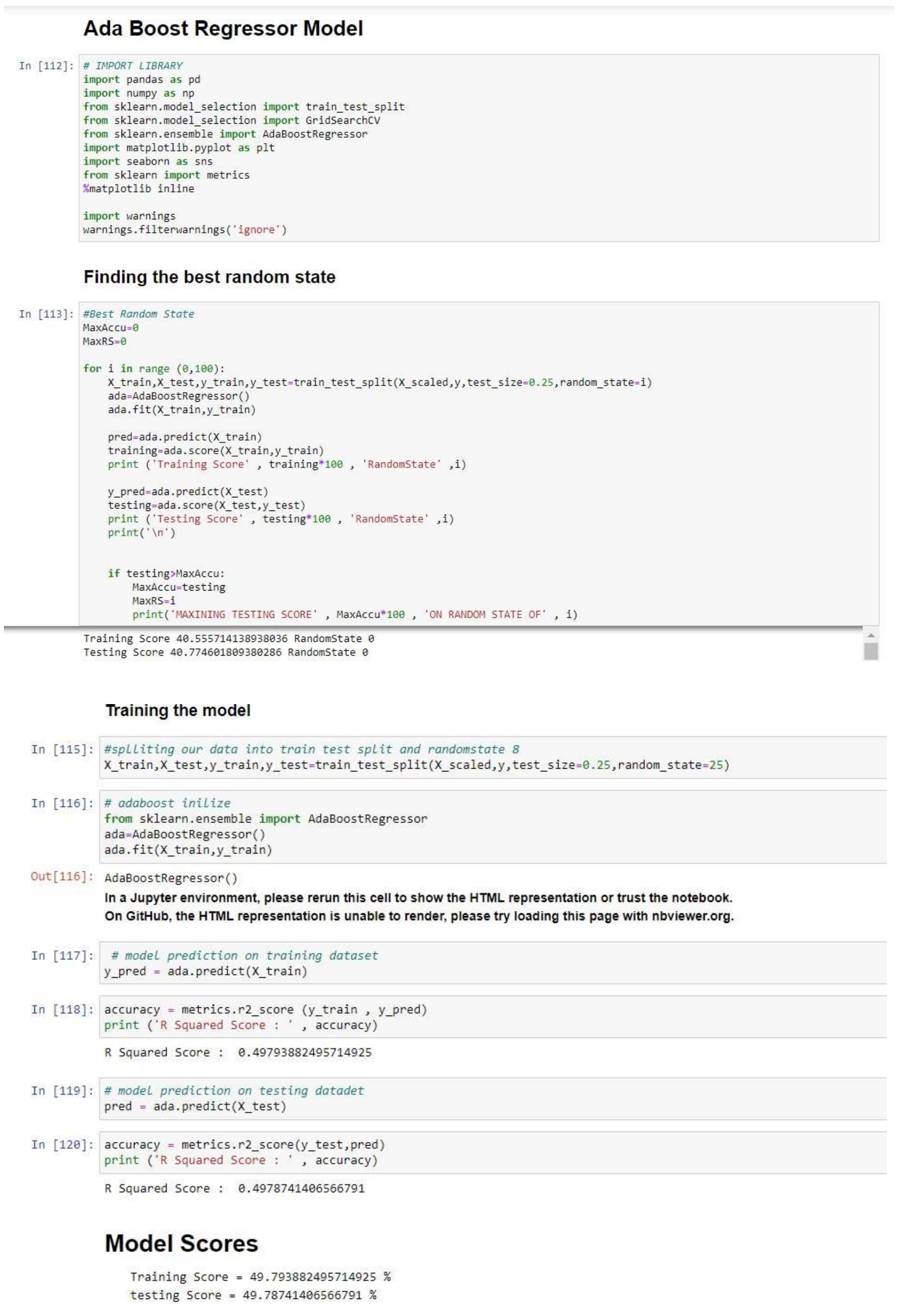
RIDGE MODEL

```
In [108]: ############ RIDGE MODEL#########

          ridgecv = RidgeCV(alphas = np.arange(0.001,0.1,0.01))
          ridgecv.fit(X_train , y_train)

Out[108]: RidgeCV(alphas=array([0.001, 0.011, 0.021, 0.031, 0.041, 0.051, 0.061, 0.071, 0.081,
                 0.091]))
          In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
          On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [109]: # best aplha parameter
          alpha = ridgecv.alpha_
          alpha

Out[109]: 0.09099999999999998

In [110]: # now we have best parametr noe train according to it
          ridge_reg = Ridge(alpha)
          ridge_reg.fit (X_train,y_train)

Out[110]: Ridge(alpha=0.09099999999999998)
          In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
          On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [111]: # now check r2 score
          ridge_reg.score(X_test,y_test)

Out[111]: 0.12948995930343177

In [ ]:
```

LASSO SCORES = 12.94841219145777 %

RIDGE SCORES = 12.948995930343177 %

2nd Model I have Created is Ada Boost Regressor Model



Ada Boost Regressor Model

```
In [112]: # IMPORT LIBRARY
          import pandas as pd
          import numpy as np
          from sklearn.model_selection import train_test_split
          from sklearn.model_selection import GridSearchCV
          from sklearn.ensemble import AdaBoostRegressor
          import matplotlib.pyplot as plt
          import seaborn as sns
          from sklearn import metrics
          %matplotlib inline

          import warnings
          warnings.filterwarnings('ignore')
```

Finding the best random state

```
In [113]: #Best Random State
          MaxAccu=0
          MaxRS=0

          for i in range (0,100):
              X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=i)
              ada=AdaBoostRegressor()
              ada.fit(X_train,y_train)

              pred=ada.predict(X_train)
              training=ada.score(X_train,y_train)
              print ('Training Score' , training*100 , 'RandomState' ,i)

              y_pred=ada.predict(X_test)
              testing=ada.score(X_test,y_test)
              print ('Testing Score' , testing*100 , 'RandomState' ,i)
              print('\n')


              if testing>MaxAccu:
                  MaxAccu=testing
                  MaxRS=i
                  print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , i)

          Training Score 40.555714138938036 RandomState 0
          Testing Score 40.774601809380286 RandomState 0
```

Training the model

```
In [115]: #splliting our data into train test split and randomstate 8
          X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=25)

In [116]: # adaboost inilize
          from sklearn.ensemble import AdaBoostRegressor
          ada=AdaBoostRegressor()
          ada.fit(X_train,y_train)

Out[116]: AdaBoostRegressor()
          In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
          On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [117]:  # model prediction on training dataset
          y_pred = ada.predict(X_train)

In [118]: accuracy = metrics.r2_score (y_train , y_pred)
          print ('R Squared Score : ' , accuracy)

          R Squared Score :  0.49793882495714925

In [119]: # model prediction on testing datadet
          pred = ada.predict(X_test)

In [120]: accuracy = metrics.r2_score(y_test,pred)
          print ('R Squared Score : ' , accuracy)

          R Squared Score :  0.4978741406566791
```

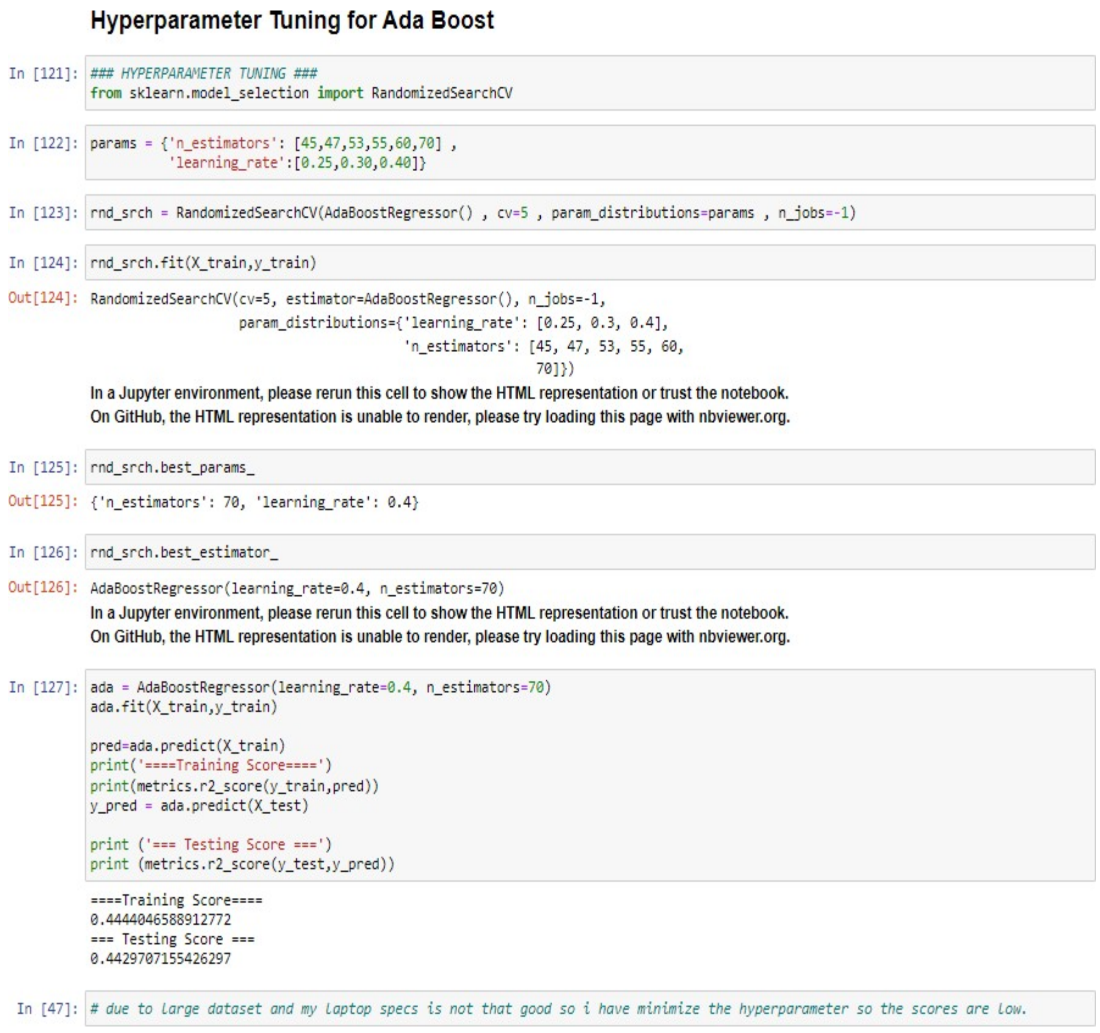
Model Scores

```
Training Score = 49.793882495714925 %
testing Score = 49.78741406566791 %
```

Ada Boost Regressor Model Scores

Training Score = 49.793882495714925 %

Testing Score = 49.78741406566791 %


Hyperparameter Tuning for Ada Boost

```
In [121]: ### HYPERPARAMETER TUNING ###
          from sklearn.model_selection import RandomizedSearchCV

In [122]: params = {'n_estimators': [45,47,53,55,60,70] ,
                    'learning_rate':[0.25,0.30,0.40]}

In [123]: rnd_srch = RandomizedSearchCV(AdaBoostRegressor() , cv=5 , param_distributions=params , n_jobs=-1)

In [124]: rnd_srch.fit(X_train,y_train)

Out[124]: RandomizedSearchCV(cv=5, estimator=AdaBoostRegressor(), n_jobs=-1,
                             param_distributions={'learning_rate': [0.25, 0.3, 0.4],
                                                  'n_estimators': [45, 47, 53, 55, 60,
                                                                   70]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [125]: rnd_srch.best_params_

Out[125]: {'n_estimators': 70, 'learning_rate': 0.4}

In [126]: rnd_srch.best_estimator_

Out[126]: AdaBoostRegressor(learning_rate=0.4, n_estimators=70)
```

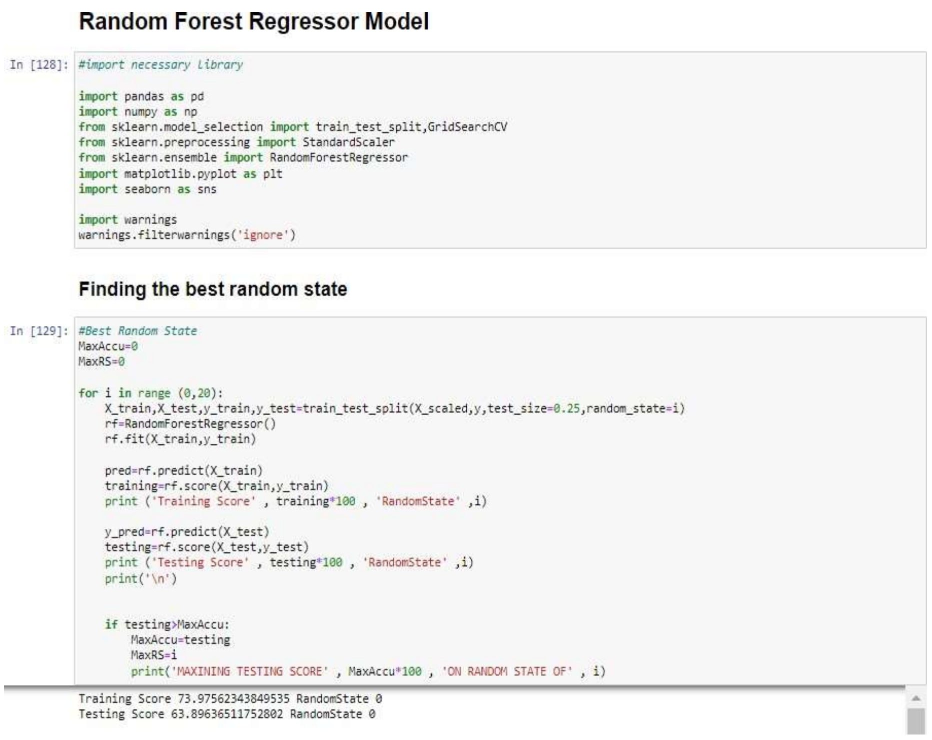
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [127]: ada = AdaBoostRegressor(learning_rate=0.4, n_estimators=70)
          ada.fit(X_train,y_train)

          pred=ada.predict(X_train)
          print('====Training Score====')
          print(metrics.r2_score(y_train,pred))
          y_pred = ada.predict(X_test)

          print ('=== Testing Score ===')
          print (metrics.r2_score(y_test,y_pred))

          ====Training Score====
          0.4444046588912772
          === Testing Score ===
          0.4429707155426297

In [47]: # due to large dataset and my laptop specs is not that good so i have minimize the hyperparameter so the scores are low.
```

<u>Model Score after Hyperparameter Tuning</u>

Training Score = 44.44046588912772 %
Testing Score = 44.29707155426297 %

3rd Model I have Created is Random Forest Regressor



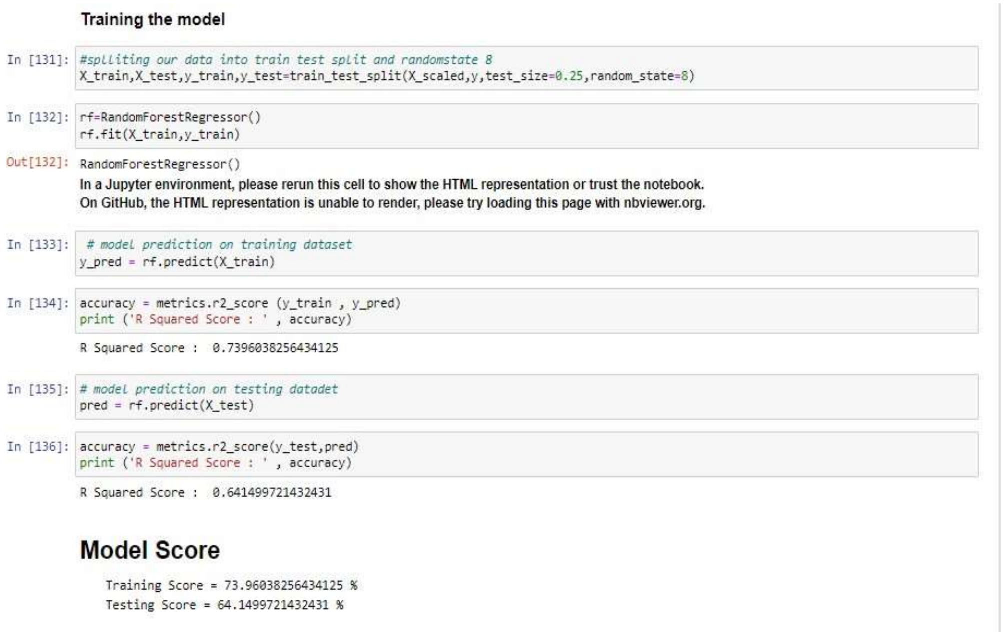
Random Forest Regressor Model

```
In [128]: #import necessary library

          import pandas as pd
          import numpy as np
          from sklearn.model_selection import train_test_split,GridSearchCV
          from sklearn.preprocessing import StandardScaler
          from sklearn.ensemble import RandomForestRegressor
          import matplotlib.pyplot as plt
          import seaborn as sns

          import warnings
          warnings.filterwarnings('ignore')
```

Finding the best random state

```
In [129]: #Best Random State
          MaxAccu=0
          MaxRS=0

          for i in range (0,20):
              X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=i)
              rf=RandomForestRegressor()
              rf.fit(X_train,y_train)

              pred=rf.predict(X_train)
              training=rf.score(X_train,y_train)
              print ('Training Score' , training*100 , 'RandomState' ,i)

              y_pred=rf.predict(X_test)
              testing=rf.score(X_test,y_test)
              print ('Testing Score' , testing*100 , 'RandomState' ,i)
              print('\n')


              if testing>MaxAccu:
                  MaxAccu=testing
                  MaxRS=i
                  print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , i)
```

```
Training Score 73.97562343849535 RandomState 0
Testing Score 63.89636511752802 RandomState 0
```



Training the model

```
In [131]: #spliting our data into train test split and randomstate 8
          X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=8)

In [132]: rf=RandomForestRegressor()
          rf.fit(X_train,y_train)

Out[132]: RandomForestRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

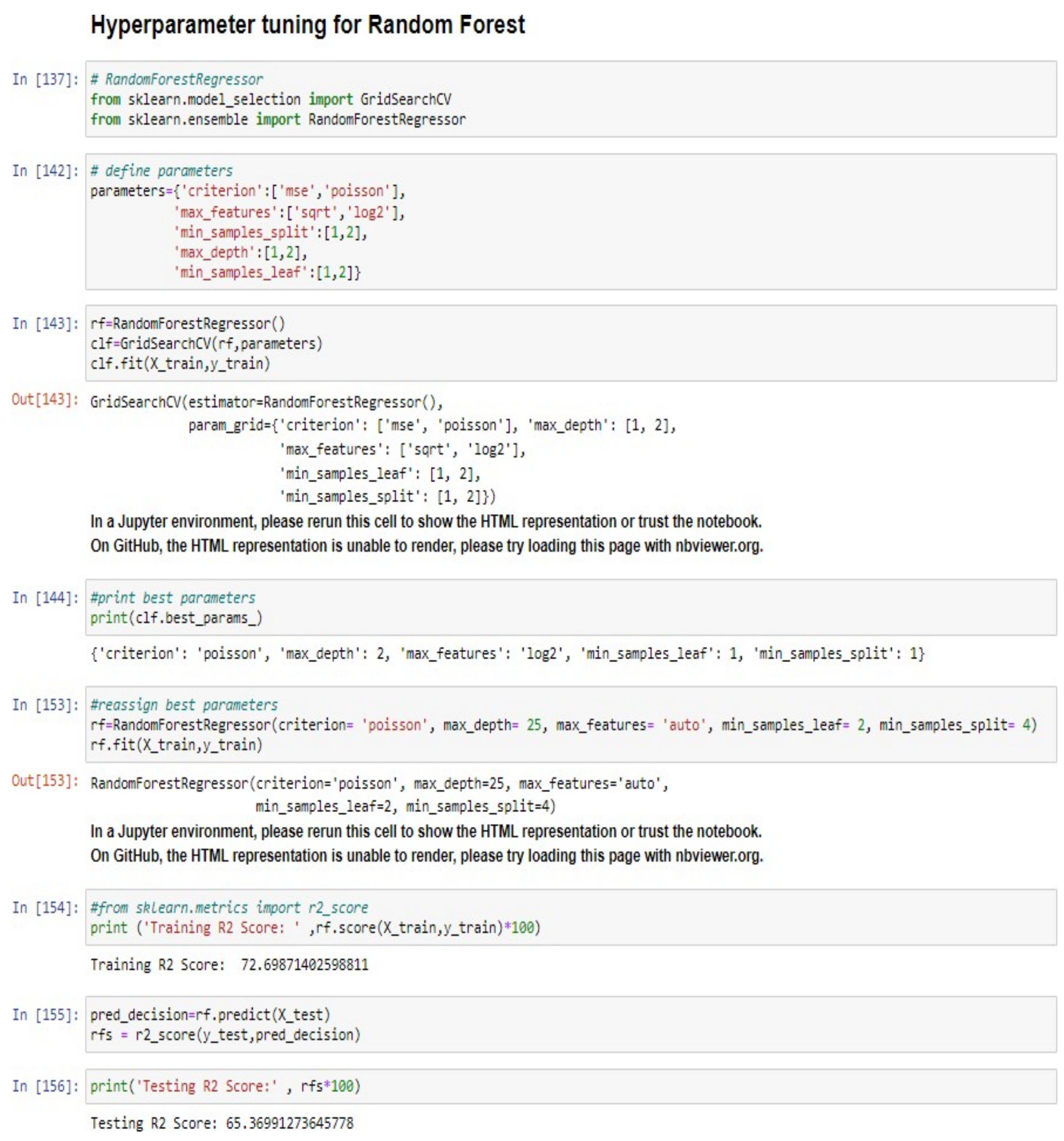
```
In [133]:  # model prediction on training dataset
          y_pred = rf.predict(X_train)

In [134]: accuracy = metrics.r2_score (y_train , y_pred)
          print ('R Squared Score : ' , accuracy)

          R Squared Score :  0.7396038256434125

In [135]: # model prediction on testing datadet
          pred = rf.predict(X_test)

In [136]: accuracy = metrics.r2_score(y_test,pred)
          print ('R Squared Score : ' , accuracy)

          R Squared Score :  0.641499721432431
```

Model Score

```
Training Score = 73.96038256434125 %
Testing Score = 64.1499721432431 %
```

Random Forest Regressor Model Score

Training Score = 73.96038256434125 %
Testing Score = 64.1499721432431 %



Hyperparameter tuning for Random Forest

```
In [137]: # RandomForestRegressor
          from sklearn.model_selection import GridSearchCV
          from sklearn.ensemble import RandomForestRegressor
```

```
In [142]: # define parameters
          parameters={'criterion':['mse','poisson'],
                      'max_features':['sqrt','log2'],
                      'min_samples_split':[1,2],
                      'max_depth':[1,2],
                      'min_samples_leaf':[1,2]}
```

```
In [143]: rf=RandomForestRegressor()
          clf=GridSearchCV(rf,parameters)
          clf.fit(X_train,y_train)

Out[143]: GridSearchCV(estimator=RandomForestRegressor(),
                       param_grid={'criterion': ['mse', 'poisson'], 'max_depth': [1, 2],
                                   'max_features': ['sqrt', 'log2'],
                                   'min_samples_leaf': [1, 2],
                                   'min_samples_split': [1, 2]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [144]: #print best parameters
          print(clf.best_params_)

          {'criterion': 'poisson', 'max_depth': 2, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 1}
```

```
In [153]: #reassign best parameters
          rf=RandomForestRegressor(criterion= 'poisson', max_depth= 25, max_features= 'auto', min_samples_leaf= 2, min_samples_split= 4)
          rf.fit(X_train,y_train)

Out[153]: RandomForestRegressor(criterion='poisson', max_depth=25, max_features='auto',
                                min_samples_leaf=2, min_samples_split=4)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [154]: #from sklearn.metrics import r2_score
          print ('Training R2 Score: ' ,rf.score(X_train,y_train)*100)

          Training R2 Score:  72.69871402598811
```

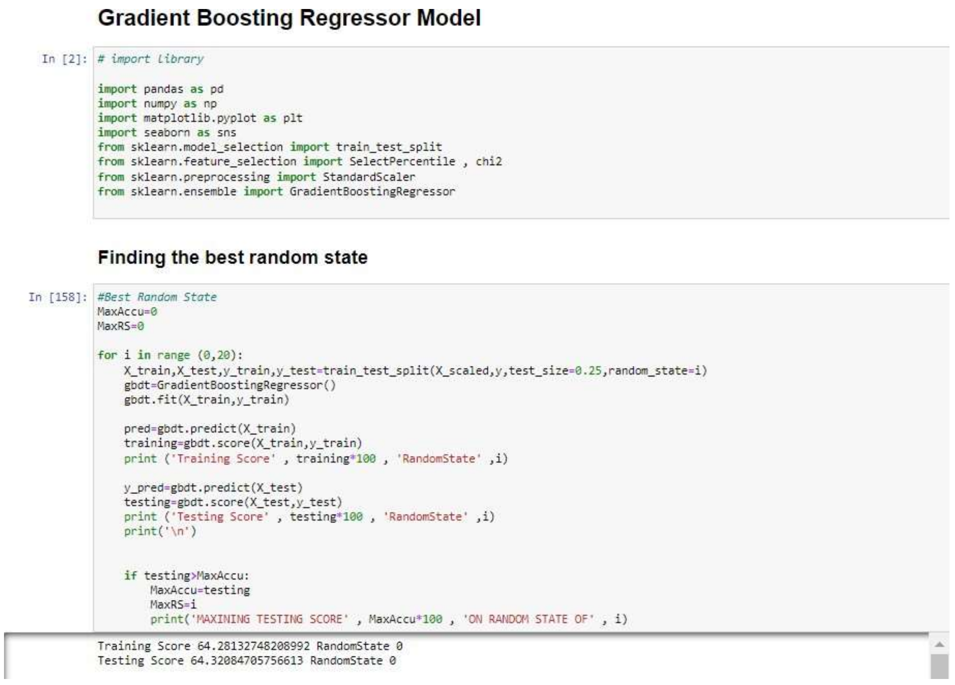
```
In [155]: pred_decision=rf.predict(X_test)
          rfs = r2_score(y_test,pred_decision)
```

```
In [156]: print('Testing R2 Score:' , rfs*100)

          Testing R2 Score: 65.36991273645778
```

Model Score after Hyperparameter Tuning

Training Score = 72.69871402598811 %
Testing Score = 65.36991273645778 %

4th Model I have Created is Gradient Boosting Regressor Model

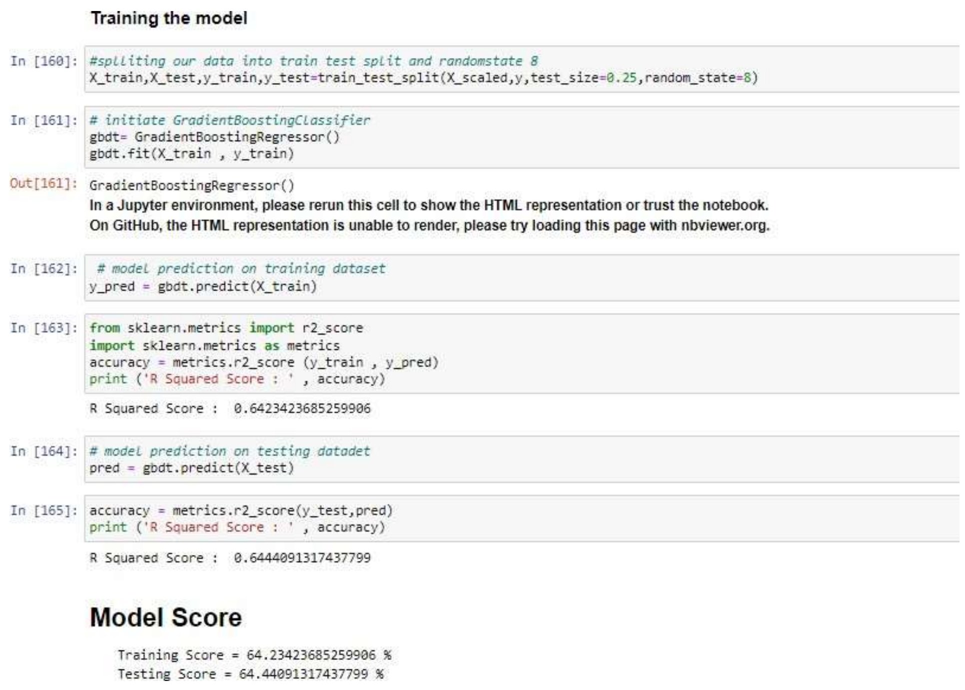


Gradient Boosting Regressor Model

```
In [2]: # import library

        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.model_selection import train_test_split
        from sklearn.feature_selection import SelectPercentile , chi2
        from sklearn.preprocessing import StandardScaler
        from sklearn.ensemble import GradientBoostingRegressor
```

Finding the best random state

```
In [158]: #Best Random State
          MaxAccu=0
          MaxRS=0

          for i in range (0,20):
              X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=i)
              gbdt=GradientBoostingRegressor()
              gbdt.fit(X_train,y_train)

              pred=gbdt.predict(X_train)
              training=gbdt.score(X_train,y_train)
              print ('Training Score' , training*100 , 'RandomState' ,i)

              y_pred=gbdt.predict(X_test)
              testing=gbdt.score(X_test,y_test)
              print ('Testing Score' , testing*100 , 'RandomState' ,i)
              print('\n')


              if testing>MaxAccu:
                  MaxAccu=testing
                  MaxRS=i
                  print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , i)

          Training Score 64.28132748208992 RandomState 0
          Testing Score 64.32084705756613 RandomState 0
```



Training the model

```
In [160]: #spliting our data into train test split and randomstate 8
          X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=8)

In [161]: # initiate GradientBoostingClassifier
          gbdt= GradientBoostingRegressor()
          gbdt.fit(X_train , y_train)

Out[161]: GradientBoostingRegressor()
          In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
          On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [162]:  # model prediction on training dataset
          y_pred = gbdt.predict(X_train)

In [163]: from sklearn.metrics import r2_score
          import sklearn.metrics as metrics
          accuracy = metrics.r2_score (y_train , y_pred)
          print ('R Squared Score : ' , accuracy)

          R Squared Score :  0.6423423685259906

In [164]: # model prediction on testing datadet
          pred = gbdt.predict(X_test)

In [165]: accuracy = metrics.r2_score(y_test,pred)
          print ('R Squared Score : ' , accuracy)

          R Squared Score :  0.6444091317437799
```

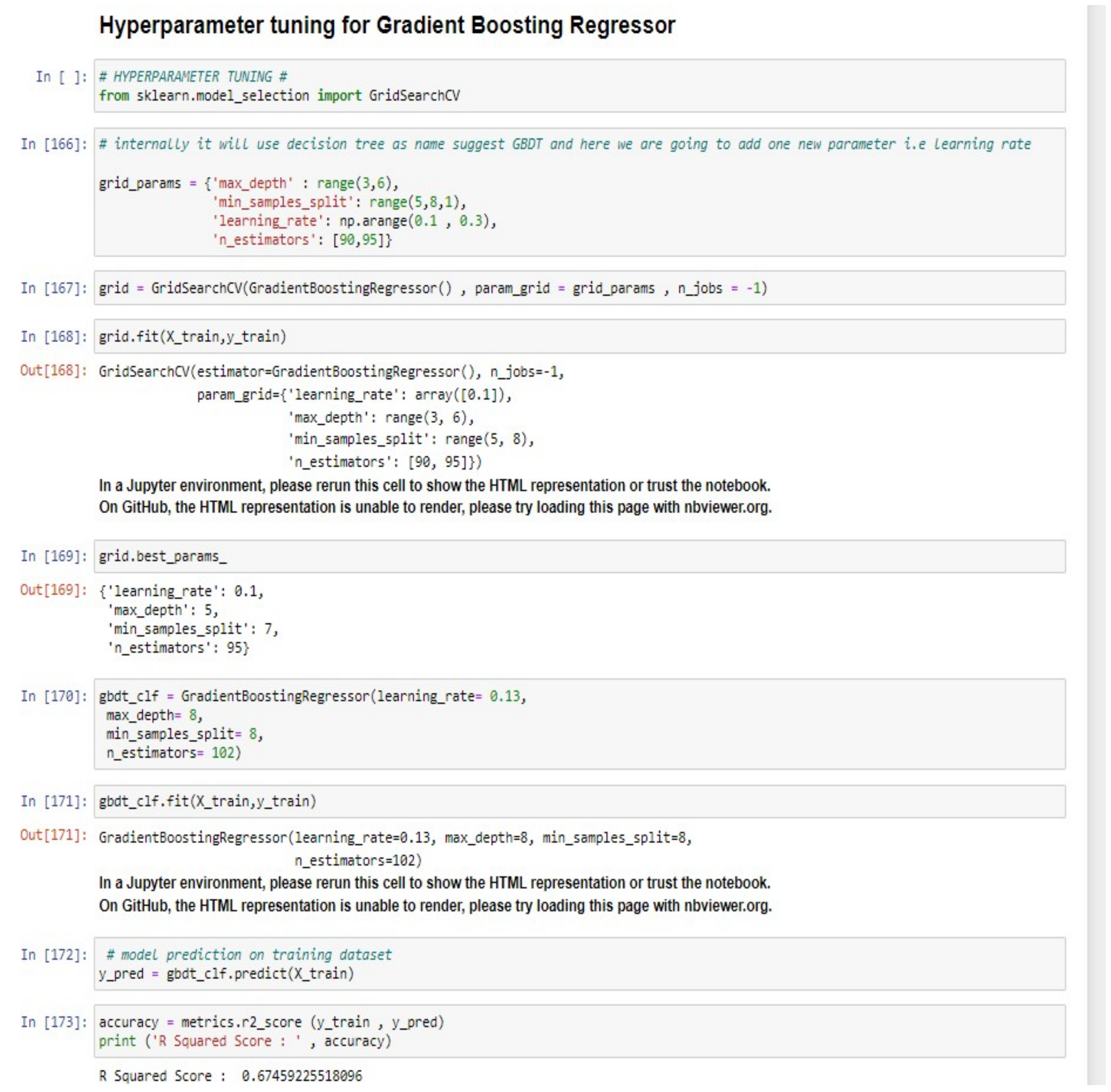
Model Score

```
  Training Score = 64.23423685259906 %
  Testing Score = 64.44091317437799 %
```

Gradient Boosting Regressor Model Model Score

Training Score = 64.23423685259906 %

Testing Score = 64.44091317437799 %



Hyperparameter tuning for Gradient Boosting Regressor

```
In [ ]: # HYPERPARAMETER TUNING #
        from sklearn.model_selection import GridSearchCV
```

```
In [166]: # internally it will use decision tree as name suggest GBDT and here we are going to add one new parameter i.e learning rate

          grid_params = {'max_depth' : range(3,6),
                         'min_samples_split': range(5,8,1),
                         'learning_rate': np.arange(0.1 , 0.3),
                         'n_estimators': [90,95]}
```

```
In [167]: grid = GridSearchCV(GradientBoostingRegressor() , param_grid = grid_params , n_jobs = -1)
```

```
In [168]: grid.fit(X_train,y_train)

Out[168]: GridSearchCV(estimator=GradientBoostingRegressor(), n_jobs=-1,
                       param_grid={'learning_rate': array([0.1]),
                                   'max_depth': range(3, 6),
                                   'min_samples_split': range(5, 8),
                                   'n_estimators': [90, 95]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [169]: grid.best_params_

Out[169]: {'learning_rate': 0.1,
           'max_depth': 5,
           'min_samples_split': 7,
           'n_estimators': 95}
```

```
In [170]: gbdt_clf = GradientBoostingRegressor(learning_rate= 0.13,
           max_depth= 8,
           min_samples_split= 8,
           n_estimators= 102)
```

```
In [171]: gbdt_clf.fit(X_train,y_train)

Out[171]: GradientBoostingRegressor(learning_rate=0.13, max_depth=8, min_samples_split=8,
                                    n_estimators=102)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

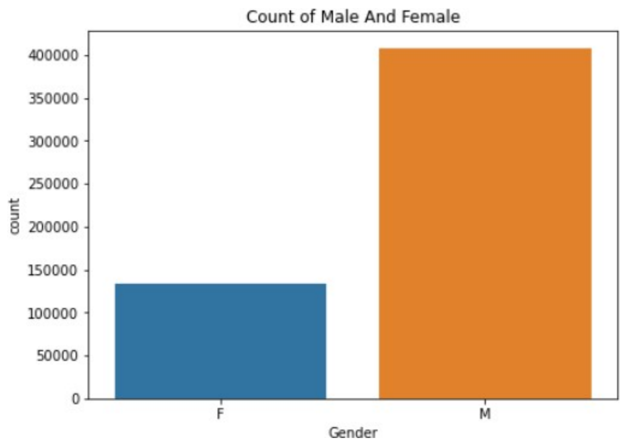
```
In [172]:  # model prediction on training dataset
          y_pred = gbdt_clf.predict(X_train)
```

```
In [173]: accuracy = metrics.r2_score (y_train , y_pred)
          print ('R Squared Score : ' , accuracy)

          R Squared Score :  0.67459225518096
```

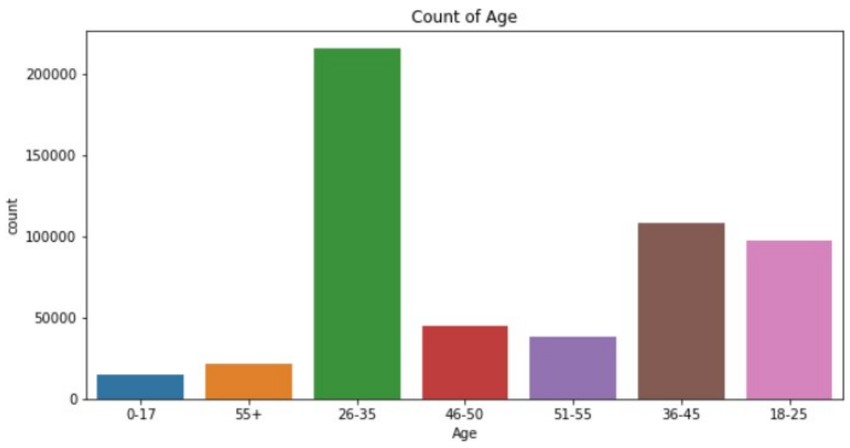
<u>Model Score after Hyperparameter Tuning</u>

Training Score = 67.459225518096 %
Testing Score = 66.51716962640886 %

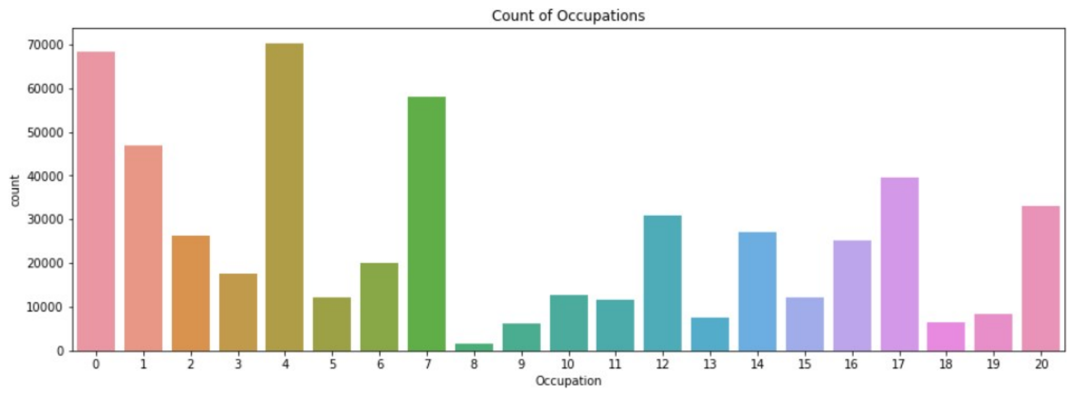
Visualizations and EDA



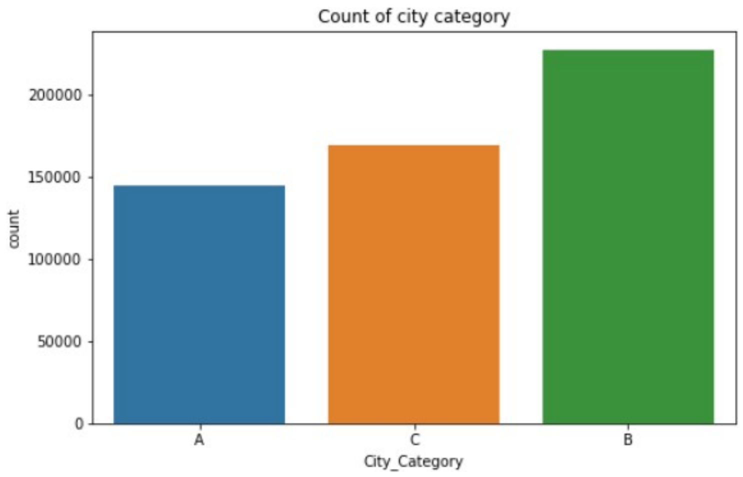
From the above bar graph, we observe the count of the male is more than female where the count of the male is 407752 and females is 134240.



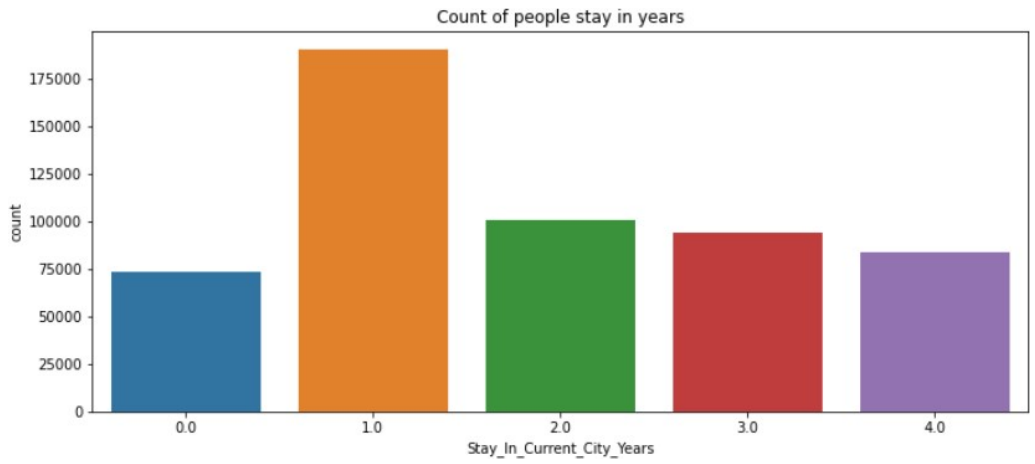
From the above bar graph, we observe there is a range of age ranging from 0 to 55+ and the most count or the people present most is of age ranging from 26-35 age followed by 36-45 age and 18-25 age and the count for that age group is 216095, 108693 and 97587.



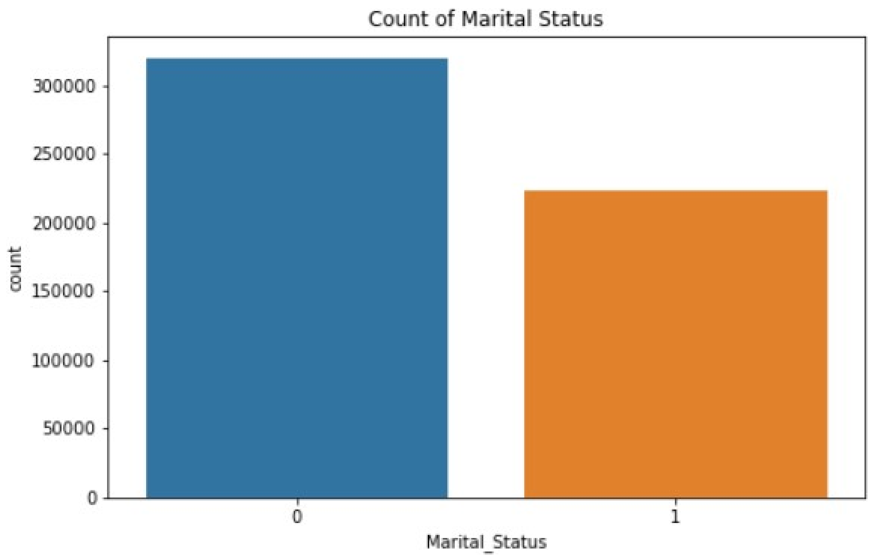
From the above bar graph, we observe the occupation of the people and the people are mostly from occupation 4 with 70274 people and the least amount of people are from occupation 8 with only 1536 people.



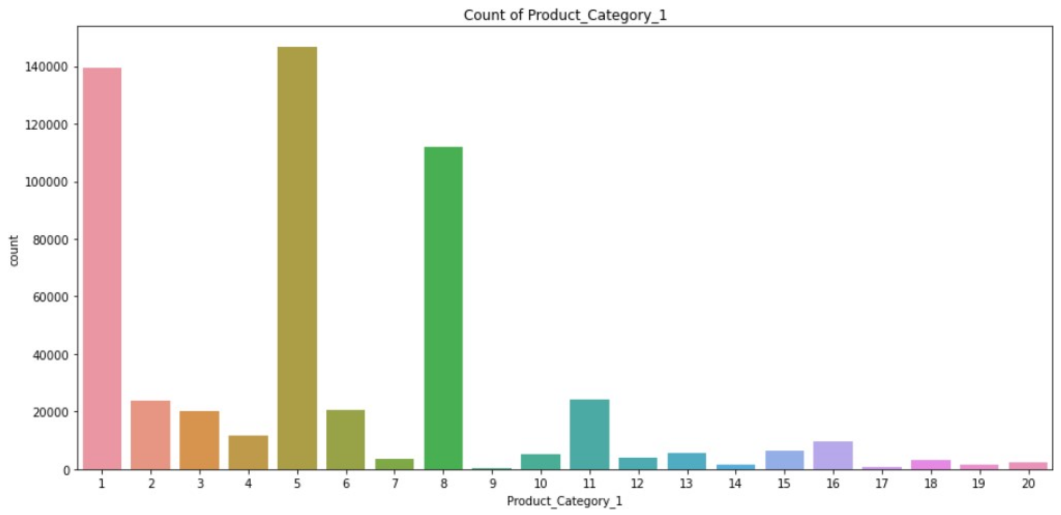
From the above Bar graph, we observe there are 3 city category and most of the people is from the B city category with 227377 people followed by the C city category with 169417 people and the least is from the A city category with 145198 people.



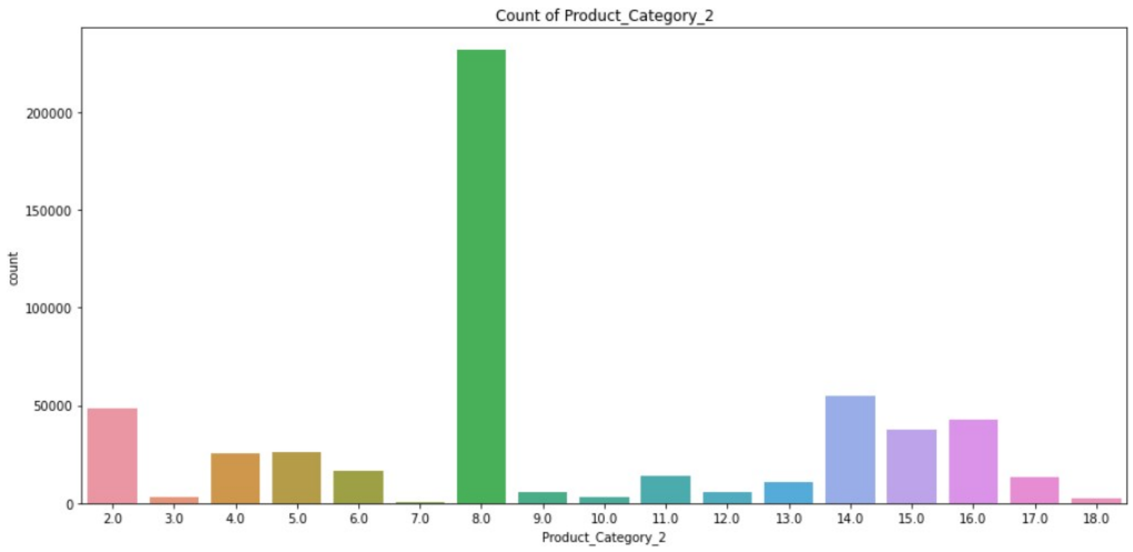
From the above bar graph we observe most of the people stay in the current city for 1 year is most which means most of the people are new cummer and we observe next is from 2 to 4+ years the count of the people stay in the current city is decreasing.



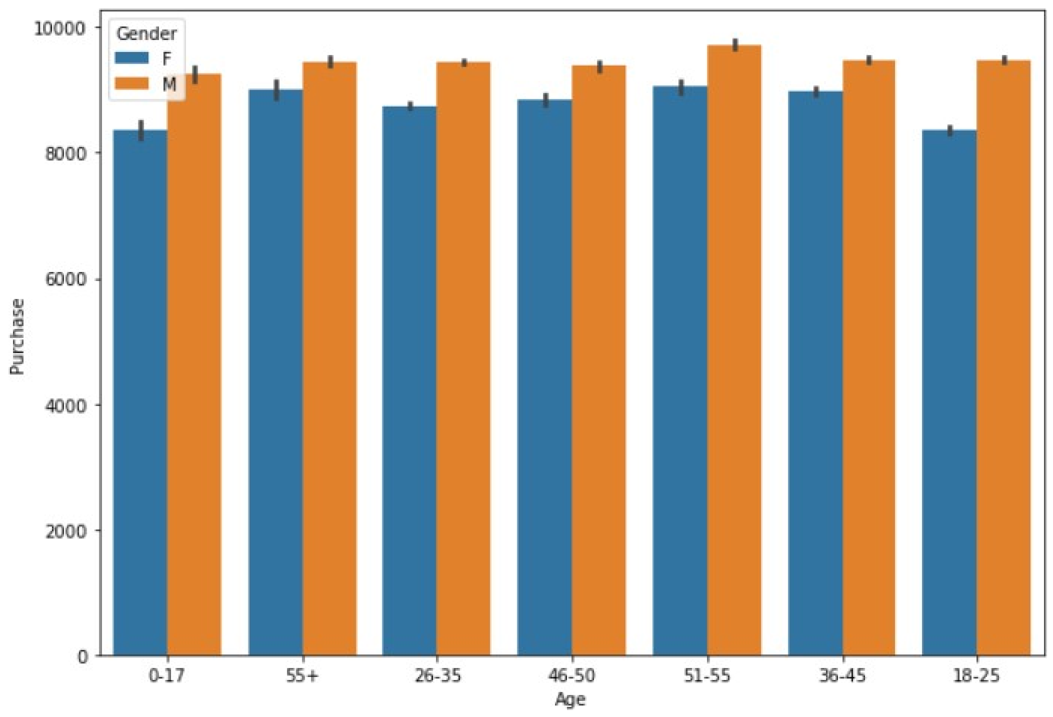
From the above bar graph, we observe most of the people are single around 319332 people, and 222660 people are married.



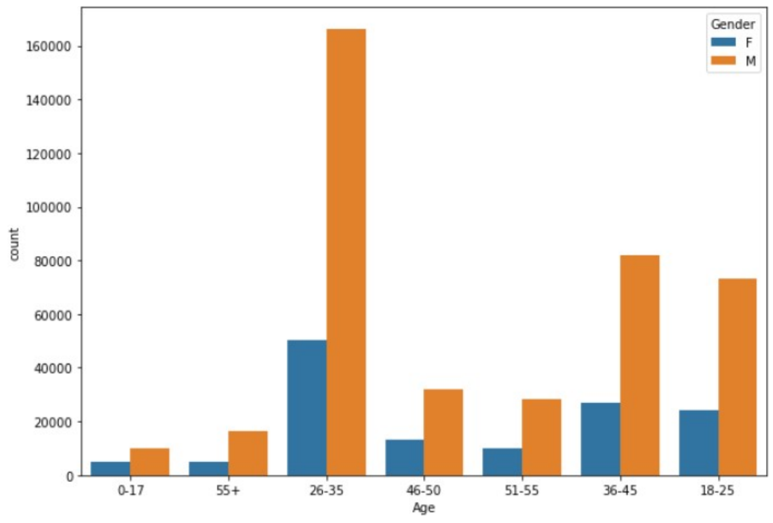
From the above bar graph, we observe there are around 20 categories in Product_Category_1, and the count which most is for 5,1,8 categories with the count of 146496, 139489, and 111933, etc.



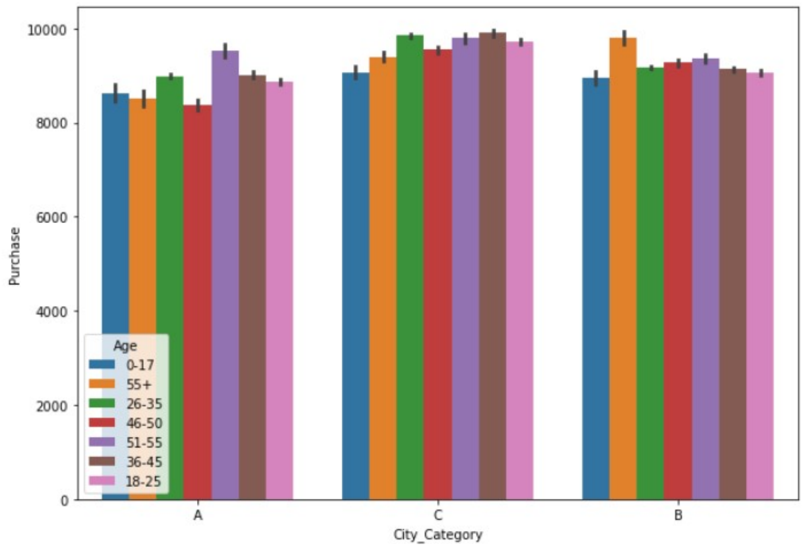
From the above bar graph, we observe there are around 17 categories in Product_Category_2, and the count which most is for 8,14 categories with the count of 231507 and 54599, etc.



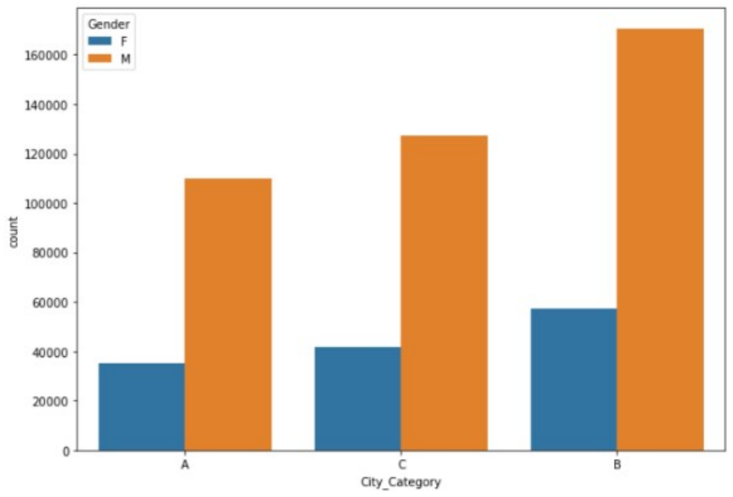
From the above bar graph, we observe the age group and purchase with additional information on gender and we observe that in each category male purchase is greater than female.



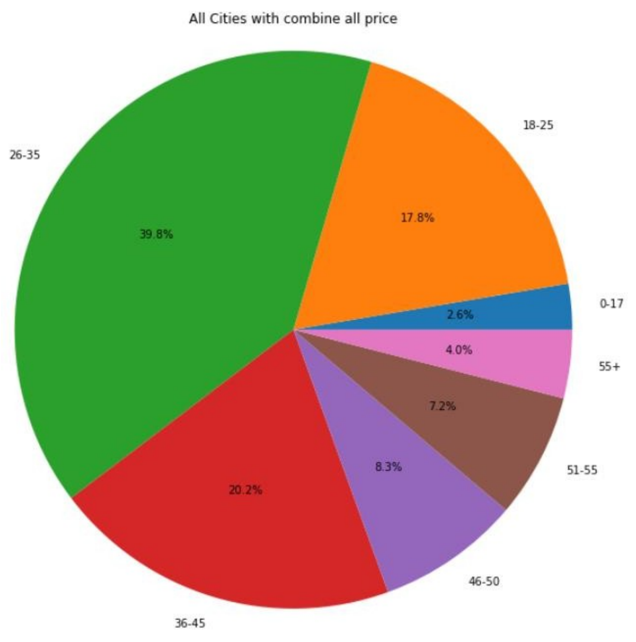
From the above bar graph, we observe the age group and information of gender and we observe that in each category male count is more than the female, especially in the age group 26-35 age group.



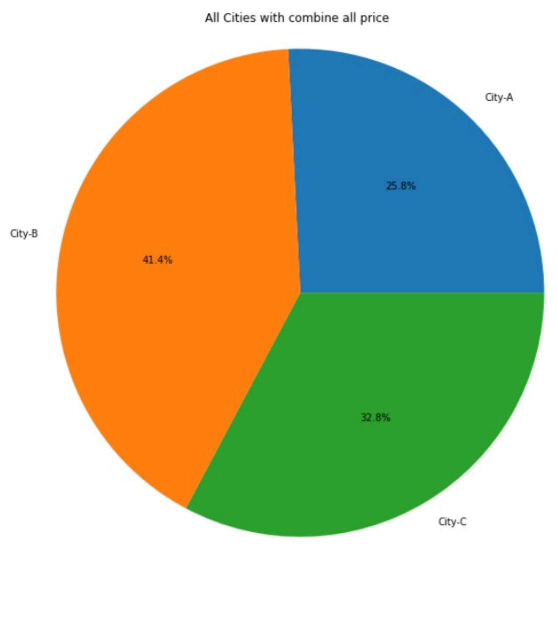
From the above bar graph, we observe the city category and purchase with additional information on age group and we observe that in each city category, the purchase history of all different age groups is different there is no trend.



From the above bar graph, we observe that city category and Gender information we already know that the male count is greater than the female count but the count of the male from B city category is followed by C city category and last A city category and this is same for female.



From the above bar graph, we observe the total sum of money spent by each age group around 39.8 % of purchases came from the age group 26-35 followed by 20.2% of purchases came from the age group 36-45 and around 17.8% purchase came from the age group of 18-25.



From the above bar graph we observe the total sum of money spent by each age city category around 41.4 % of purchases came from the B city category followed by 32.8% of purchases came from the c city category and around 25.8% of purchases came from the A city category.

CONCLUSION

Key Findings and Conclusions of the Study

So from above all 4 model scores, we observe Gradient Boosting Regressor Model is best Suited model for this particular model as the training score is 64.23423685259906 % and the testing score is 64.44091317437799 % thus saving this model and we will use this model to prediction on the test dataset.

Learning Outcomes of the Study in respect of Data Science

1) first we identify null values and fill the null values by using mode mehod

2) then I identified duplicates and I have dropped duplicates

3) performed EDA and wrote all observations for each graph

4) then i dropped unnecessary columns

5) then applied a label encoder to the categorical columns

6) then also plotted the Distribution plot and regression plot

7) then plotted boxplot to remove outliers

8) then treated outliers with the Z-score method

9) then scaled data and Also check for VIF

10) then find the co-relation between feature and label by the CORR method

11) then created 4 models that is Gradient Boosting Regressor, Random Forest Regressor ,linear Regressor model, Ada Boosting regressor model with hyperparameter tuning for all 4 models and also Cross-validations

12) At last I selected the best model according to their Hyper-parameter score and Training(R2) and testing score(R2)

Limitations of this work and Scope for Future Work

Limitations:

1] Data quality: One of the biggest limitations in any data science project is the quality of the data. If the data is incomplete or inaccurate, it can lead to incorrect insights and predictions.

2] Scope of analysis: Another limitation is the scope of the analysis. Depending on the data available, the analysis may be limited to a specific geographic region, product category, or time period.

3] Model accuracy: The accuracy of the predictive models used in the analysis can also be a limitation. The models may not be able to capture all the factors that affect consumer behavior during Black Friday.

Scope for Future Work:

1] Incorporating more data sources: Future work could include incorporating more data sources to improve the accuracy of the analysis. For example, data from social media platforms could be analyzed to understand consumer sentiment towards specific products or brands.

2] Deeper analysis: Future work could also involve a deeper analysis of the data to uncover hidden patterns or insights. This could include using advanced machine learning techniques such as deep learning.

3] Real-time monitoring: Another area of future work could be to develop real-time monitoring systems that can analyze Black Friday sales as they happen. This could help retailers make informed decisions about pricing and inventory management during the event.

4] Personalization: Finally, future work could explore ways to personalize the Black Friday shopping experience for individual consumers. This could include using data to recommend products based on a consumer's browsing and purchase history.