BLACK FRIDAY SALES PREDICTION

PROJECT USING MongoDB AND PySpark

VAIBHAV ARORA

IMPORTANT LINKS

COLAB NOTEBOOK :
https://colab.research.google.com/drive/1eGfaM5o8QZoRWGpR1omgDFOswgYlbSrm?usp=sharing

DATA :
https://drive.google.com/file/d/132RNyVqCpz6U-uvIZv_AUkwkrkYL35Pw/view?usp=share_link

PROBLEM STATEMENT

Retail is the sale of goods and services from individuals or businesses to the end- user. The retail industry provides consumers with goods and services for their everyday needs. In retail one of crucial part is to understand the consumer behavior and make various arrangements for the sales of the company. A retail company “ABC Private Limited” wants to understand the customer purchase behavior (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.

ABOUT THE DATASET

This dataset comprises of sales transactions captured at a retail store. This is a regression problem. The dataset has 550,069 rows and 12 columns.

Problem: Predict purchase amount.

Data Overview

Dataset has 550068 rows (transactions) and 12 columns (features) as described below:

- User_ID: Unique ID of the user.
- Product_ID: Unique ID of the product.
- Gender: indicates the gender of the person making the transaction.
- Age: indicates the age group of the person making the transaction.
- Occupation: shows the occupation of the user, already labeled with numbers 0 to 20.
- City_Category: User's living city category. Cities are categorized into 3 different categories 'A', 'B' and 'C'.
- Stay_In_Current_City_Years: Indicates how long the users has lived in this city.
- Marital_Status: is 0 if the user is not married and 1 otherwise.
- Product_Category_1 to _3: Category of the product. All 3 are already labaled with numbers.
- Purchase: Purchase amount.

EDA on MongoDB

```
[23] table = db.Project
     table.count_documents({})
     # Total number of rows in dataframe is 550068.

     550068
```

```
# null value
print("Null Values:")
for i in list_columns[1:]:
    print(i,":", table.count_documents({"{}".format(i): "none"}))
```

```
Null Values:
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 : 0
Purchase : 0
```

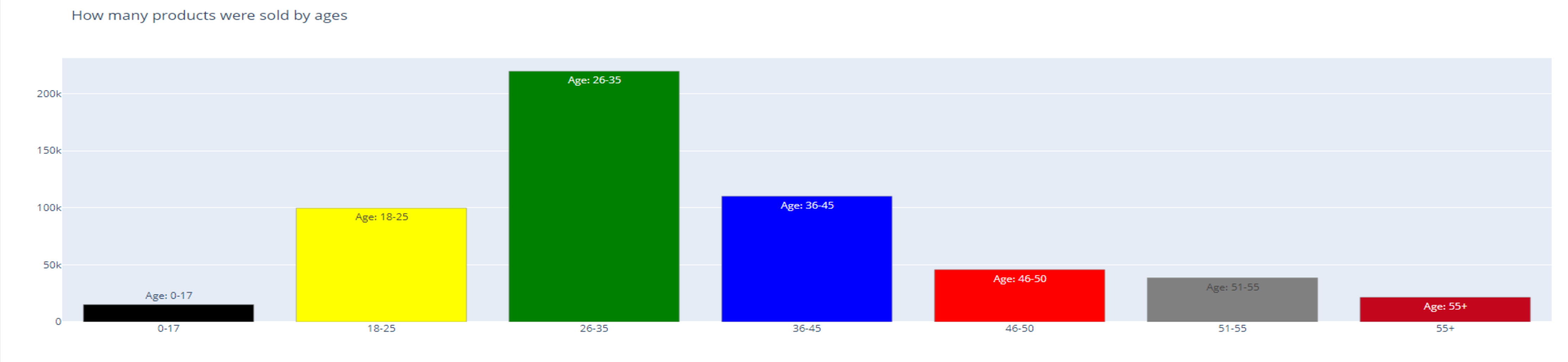
```
df.info()
```

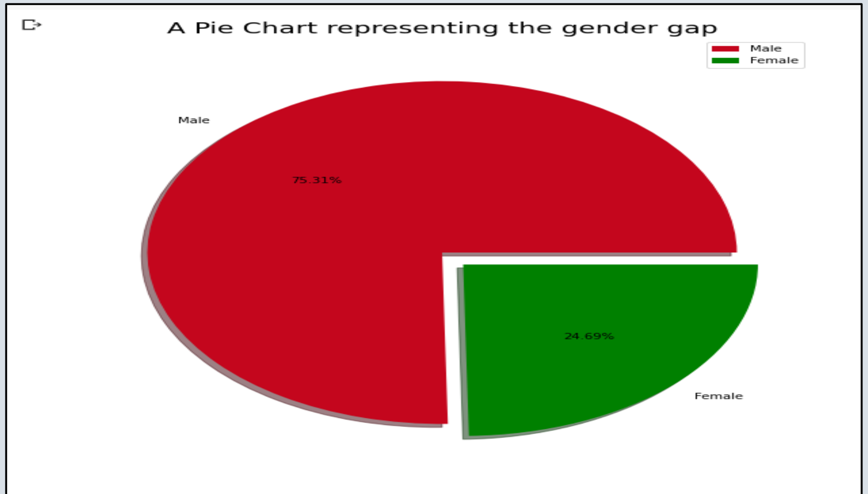
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0                               550068 non-null  object
 1   User_ID                     550068 non-null  int64
 2   Product_ID                  550068 non-null  object
 3   Gender                      550068 non-null  object
 4   Age                         550068 non-null  object
 5   Occupation                  550068 non-null  int64
 6   City_Category               550068 non-null  object
 7   Stay_In_Current_City_Years  550068 non-null  object
 8   Marital_Status              550068 non-null  int64
 9   Product_Category_1          550068 non-null  int64
 10  Product_Category_2          550068 non-null  float64
 11  Purchase                    550068 non-null  int64
dtypes: float64(1), int64(5), object(6)
memory usage: 50.4+ MB
```

```
[340] # Creating Dataframe
      df = pd.DataFrame(list(table.find({},{"_id":0})))
      df.head()
```

		User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Purchase
0	1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	15200
1	2	1000001	P00087842	F	0-17	10	A	2	0	12	9.0	1422
2	0	1000001	P00069042	F	0-17	10	A	2	0	3	9.0	8370
3	3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	1057
4	4	1000002	P00285442	M	55+	16	C	4+	0	8	9.0	7969

Categorical Variables





On average the male gender spends more money on purchase contrary to female.

Insights from Variable

Cheapest Product

Product ID : P00370293 has the Expensive Product with **Amount 12**.

A basic observation is that:

1. Product P00265242 is the most popular product.
2. Most of the transactions were made by men.
3. Age group with most transactions was 26-35.
4. City Category with most transactions was B

Minimum orders and Purchase

Age : 0-17 ~ Purchase : 134913183 is the category of Age where purchase frequency is minimum and purchase amount is also minimum

Maximum orders

Occupation Category : 0 ~ Purchase : 635406958 is the category of Occupation where purchase frequency is maximum

The Whale Customer

User_ID : 1004277 ~ Purchase_Amount : 10536909 has the maximum Purchase.

High End Product

Product ID : P00052842 has the Expensive Product with **Amount 23961**.

Maximum Purchase

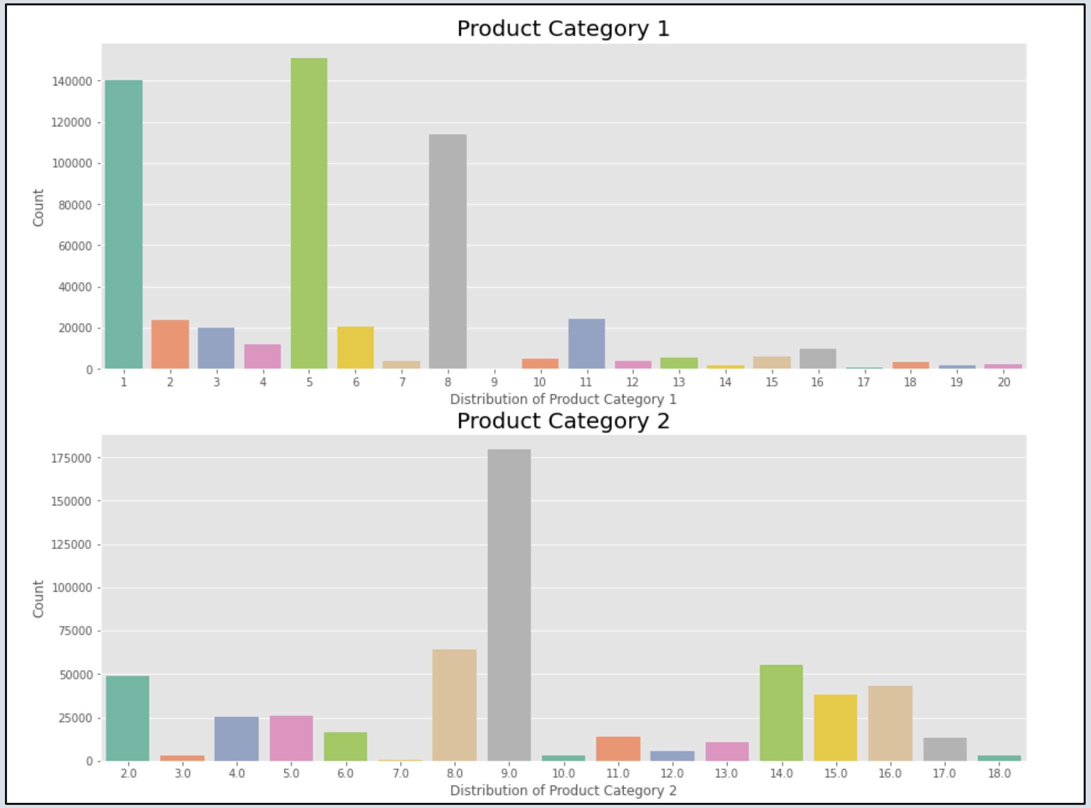
Occupation Category : 4 ~ Purchase : 2031770578 is the category of Occupation where purchase is maximum

Maximum Purchase

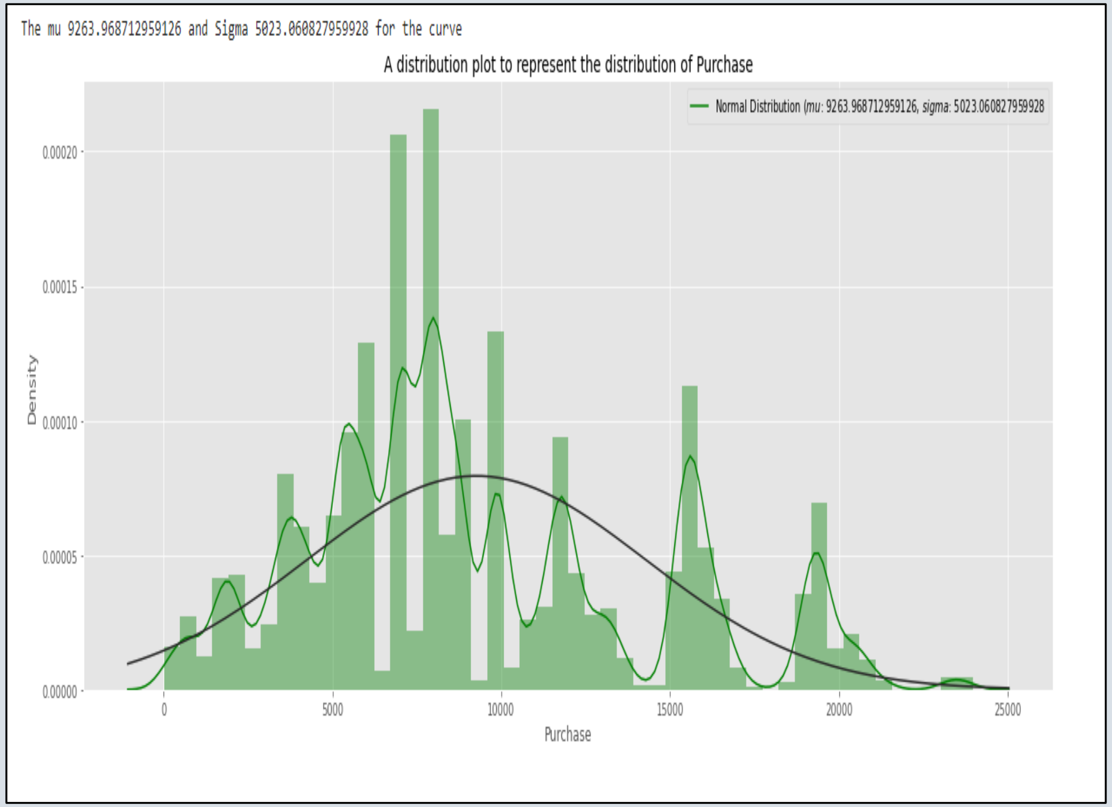
Age : 26-35 ~ Purchase : 2031770578 is the category of Age where purchase is maximum

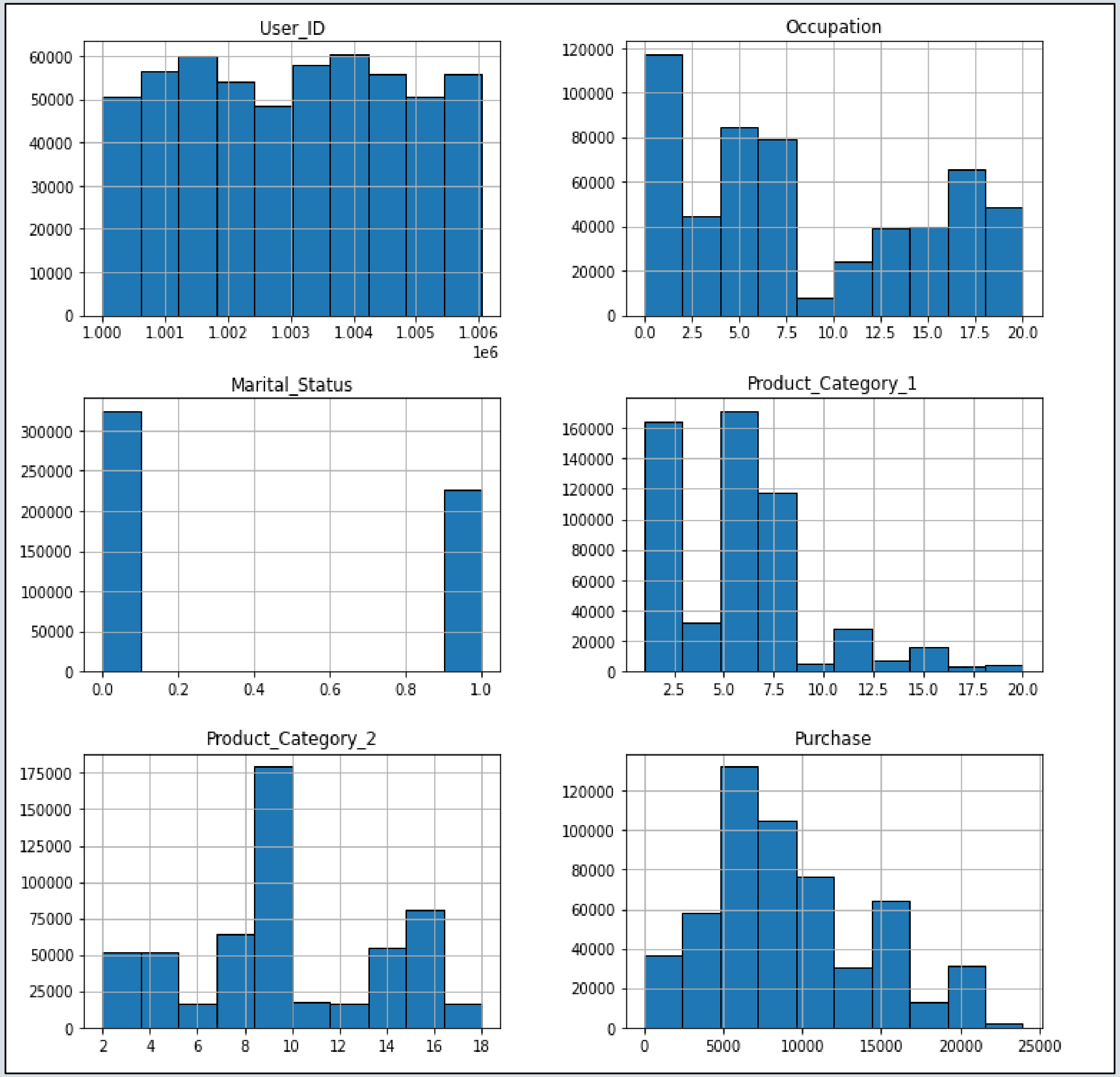
The Loyal Customer

User ID : 1001680 ~ Purchase Amount : 8699596 has max frequency.



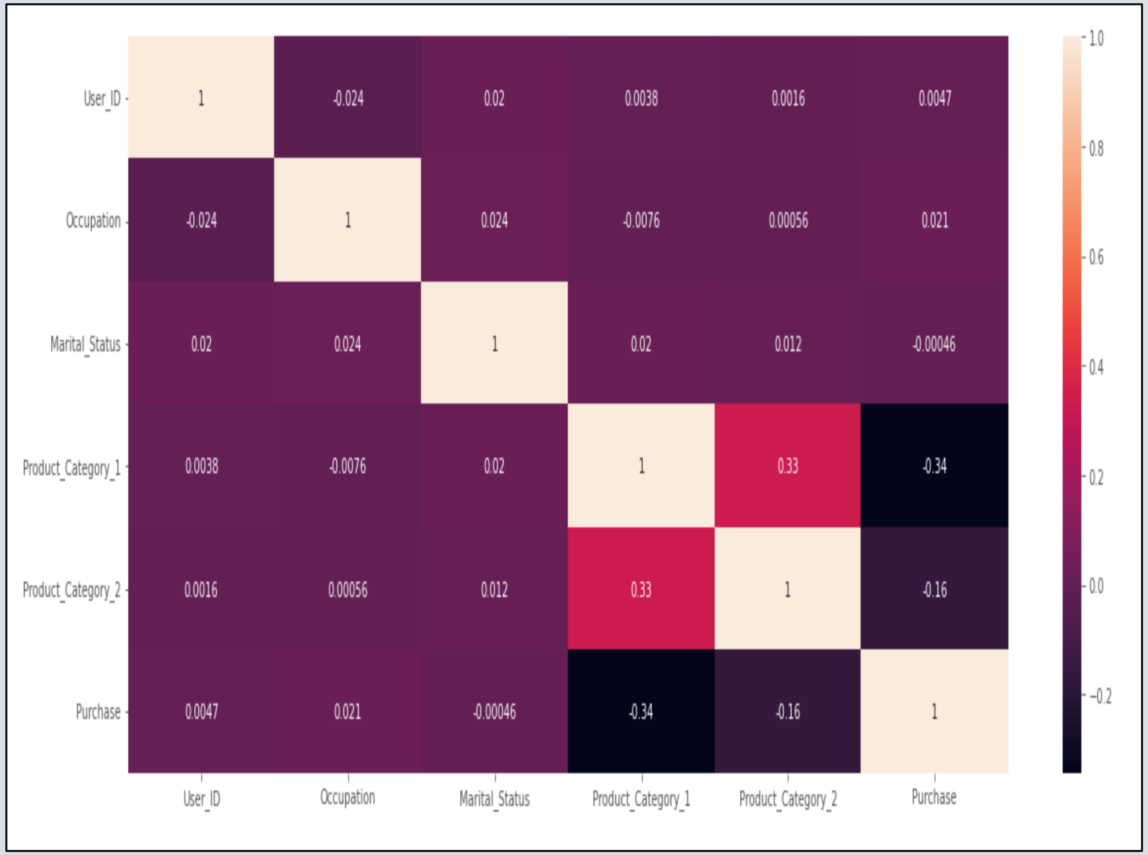
Numerical Variable





Checking for Multicollinearity

From the correlation heatmap we can see that the linear association/ correlation between our variables is not more than 0.4 in all the cases which can be considered as weak correlation. So we can conclude that there are minimal chances of multicollinearity in our dataset.



Data Pre-Processing

- Dropping Irrelevant Variables
- For the purpose of data pre-processing we have used the following tools :
  - String Indexer
  - Assembler
  - Standard Scaler

```
Data_StringIndexer2.show(5)
```

```
+------+----+----------+-------------+--------------------------+--------------+------------------+------------------+--------+-----------+--------+------------------+-------------------------------+
|Gender| Age|Occupation|City_Category|Stay_In_Current_City_Years|Marital_Status|Product_Category_1|Product_Category_2|Purchase|GenderIndex|AgeIndex|City_CategoryIndex|Stay_In_Current_City_YearsIndex|
+------+----+----------+-------------+--------------------------+--------------+------------------+------------------+--------+-----------+--------+------------------+-------------------------------+
|     F|0-17|        10|            A|                         2|             0|                 1|               6.0|   15200|        1.0|     6.0|               2.0|                            1.0|
|     F|0-17|        10|            A|                         2|             0|                12|               9.0|    1422|        1.0|     6.0|               2.0|                            1.0|
|     F|0-17|        10|            A|                         2|             0|                 3|               9.0|    8370|        1.0|     6.0|               2.0|                            1.0|
|     F|0-17|        10|            A|                         2|             0|                12|              14.0|    1057|        1.0|     6.0|               2.0|                            1.0|
|     M| 55+|        16|            C|                        4+|             0|                 8|               9.0|    7969|        0.0|     5.0|               1.0|                            3.0|
+------+----+----------+-------------+--------------------------+--------------+------------------+------------------+--------+-----------+--------+------------------+-------------------------------+
only showing top 5 rows
```

```
Data_assembler.show(5)
```

```
+------+----+----------+-------------+--------------------------+--------------+------------------+------------------+--------+-----------+--------+------------------+-------------------------------+--------------------+
|Gender| Age|Occupation|City_Category|Stay_In_Current_City_Years|Marital_Status|Product_Category_1|Product_Category_2|Purchase|GenderIndex|AgeIndex|City_CategoryIndex|Stay_In_Current_City_YearsIndex|            features|
+------+----+----------+-------------+--------------------------+--------------+------------------+------------------+--------+-----------+--------+------------------+-------------------------------+--------------------+
|     F|0-17|        10|            A|                         2|             0|                 1|               6.0|   15200|        1.0|     6.0|               2.0|                            1.0|[1.0,6.0,10.0,2.0...|
|     F|0-17|        10|            A|                         2|             0|                12|               9.0|    1422|        1.0|     6.0|               2.0|                            1.0|[1.0,6.0,10.0,2.0...|
|     F|0-17|        10|            A|                         2|             0|                 3|               9.0|    8370|        1.0|     6.0|               2.0|                            1.0|[1.0,6.0,10.0,2.0...|
|     F|0-17|        10|            A|                         2|             0|                12|              14.0|    1057|        1.0|     6.0|               2.0|                            1.0|[1.0,6.0,10.0,2.0...|
|     M| 55+|        16|            C|                        4+|             0|                 8|               9.0|    7969|        0.0|     5.0|               1.0|                            3.0|[0.0,5.0,16.0,1.0...|
+------+----+----------+-------------+--------------------------+--------------+------------------+------------------+--------+-----------+--------+------------------+-------------------------------+--------------------+
only showing top 5 rows
```

```
[ ] scaled_df.show(5)
```

```
+------+----+----------+-------------+--------------------------+--------------+------------------+------------------+--------+-----------+--------+------------------+-------------------------------+--------------------+--------------------+
|Gender| Age|Occupation|City_Category|Stay_In_Current_City_Years|Marital_Status|Product_Category_1|Product_Category_2|Purchase|GenderIndex|AgeIndex|City_CategoryIndex|Stay_In_Current_City_YearsIndex|            features|     features_scaled|
+------+----+----------+-------------+--------------------------+--------------+------------------+------------------+--------+-----------+--------+------------------+-------------------------------+--------------------+--------------------+
|     F|0-17|        10|            A|                         2|             0|                 1|               6.0|   15200|        1.0|     6.0|               2.0|                            1.0|[1.0,6.0,10.0,2.0...|[2.31908001899471...|
|     F|0-17|        10|            A|                         2|             0|                12|               9.0|    1422|        1.0|     6.0|               2.0|                            1.0|[1.0,6.0,10.0,2.0...|[2.31908001899471...|
|     F|0-17|        10|            A|                         2|             0|                 3|               9.0|    8370|        1.0|     6.0|               2.0|                            1.0|[1.0,6.0,10.0,2.0...|[2.31908001899471...|
|     F|0-17|        10|            A|                         2|             0|                12|              14.0|    1057|        1.0|     6.0|               2.0|                            1.0|[1.0,6.0,10.0,2.0...|[2.31908001899471...|
|     M| 55+|        16|            C|                        4+|             0|                 8|               9.0|    7969|        0.0|     5.0|               1.0|                            3.0|[0.0,5.0,16.0,1.0...|[0.0,3.0712334040...|
+------+----+----------+-------------+--------------------------+--------------+------------------+------------------+--------+-----------+--------+------------------+-------------------------------+--------------------+--------------------+
only showing top 5 rows
```

Dropping irrelevant features

```
[ ] Data = Data.drop('User_ID')

[ ] Data = Data.drop('Product_ID')
```

Train Test Split

Our dataset is split into training and testing in the ratio of 80 percent, 20 percent respectively.

```
[ ] # Split the data into train and test sets
    train_data, test_data = scaled_df.randomSplit([.8,.2],seed=1234)
```

```
train_data.show(5)
```

Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Purchase	GenderIndex	AgeIndex	City_CategoryIndex	Stay_In_Current_City_YearsIndex	features	features_scaled
F	0-17	0	A	2	0	1	2.0	12113	1.0	6.0	2.0	1.0	[1.0,6.0,0.0,2.0,...	[2.31908001899471...
F	0-17	0	A	2	0	3	4.0	10962	1.0	6.0	2.0	1.0	[1.0,6.0,0.0,2.0,...	[2.31908001899471...
F	0-17	0	A	2	0	5	9.0	7029	1.0	6.0	2.0	1.0	[1.0,6.0,0.0,2.0,...	[2.31908001899471...
F	0-17	0	A	2	0	8	9.0	5960	1.0	6.0	2.0	1.0	[1.0,6.0,0.0,2.0,...	[2.31908001899471...
F	0-17	0	A	2	0	8	9.0	9835	1.0	6.0	2.0	1.0	[1.0,6.0,0.0,2.0,...	[2.31908001899471...

only showing top 5 rows

```
[ ] test_data.show(5)
```

Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Purchase	GenderIndex	AgeIndex	City_CategoryIndex	Stay_In_Current_City_YearsIndex	features	features_scaled
F	0-17	0	A	2	0	3	4.0	10807	1.0	6.0	2.0	1.0	[1.0,6.0,0.0,2.0,...	[2.31908001899471...
F	0-17	0	A	2	0	5	14.0	7180	1.0	6.0	2.0	1.0	[1.0,6.0,0.0,2.0,...	[2.31908001899471...
F	0-17	0	B	1	0	1	6.0	15282	1.0	6.0	0.0	0.0	(8,[0,1,6,7],[1.0...	(8,[0,1,6,7],[2.3...
F	0-17	0	B	1	0	1	8.0	19052	1.0	6.0	0.0	0.0	(8,[0,1,6,7],[1.0...	(8,[0,1,6,7],[2.3...
F	0-17	0	B	1	0	1	8.0	19253	1.0	6.0	0.0	0.0	(8,[0,1,6,7],[1.0...	(8,[0,1,6,7],[2.3...

only showing top 5 rows

MODEL TRAINING USING PYSPARK

1. Linear Regression
2. Random Forest
3. Gradient Boost Regressor

3 different models used for training the data and the outputs were evaluated

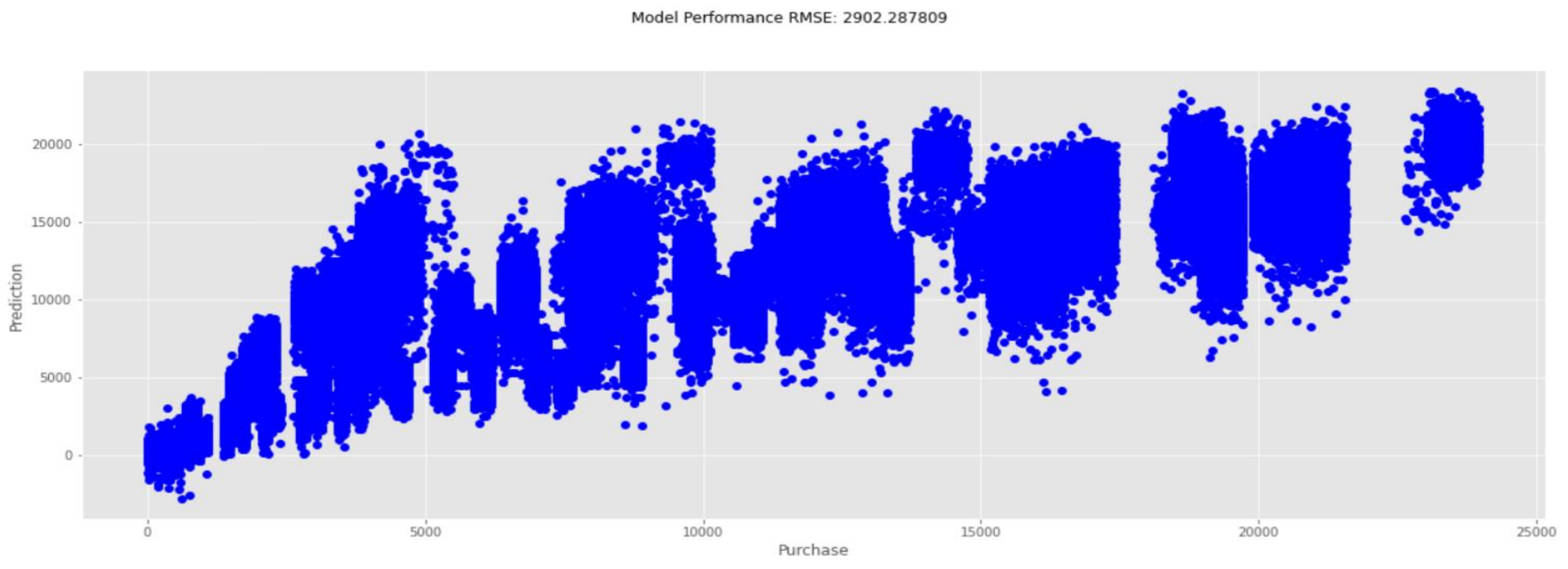
```
[341] # Get the RMSE
      print("Linear Regression RMSE:", linearModel.summary.rootMeanSquaredError)

      Linear Regression RMSE: 4705.371320576718
```

```
print("Random Forest RMSE: ", rmse)

Random Forest RMSE:  3820.1179413368436
```

```
[343] print("Gradient Boost Regressor RMSE: ", rmse1)

      Gradient Boost Regressor RMSE:  2902.2878086091832
```

MODEL EVALUATION

```
import matplotlib.pyplot as plt
evaluator = RegressionEvaluator(labelCol="Purchase", predictionCol="prediction", metricName="rmse")
rmse1 = evaluator.evaluate(predicted_GBR)
rfPred = gbrmodel.transform(scaled_df)
rfResult = rfPred.toPandas()
plt.plot(rfResult.Purchase, rfResult.prediction, 'bo')
plt.xlabel('Purchase')
plt.ylabel('Prediction')
plt.suptitle("Model Performance RMSE: %f" % rmse1)
plt.show()
```



CONCLUSION

Gradient Boost Regressor is giving the best result in our case when compared with other models because GBR starts with building a primary model from available training data sets then it identifies the errors present in the base model. After identifying the error, a secondary model is built, and further, a third model is introduced in this process. In this way, this process of introducing more models is continued until we get a complete training data set by which model predicts correctly.