Black Friday Sales Prediction

Bhavesh Khanchandani

Problem Statement

- 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 summaries of various customers for selected high volume products from last month.
- The data set also contains customer demographics (age, gender, marital status, citytype, stayincurrentcity), product details (productid and product category) and Total purchase amount from last month.
- Now, they want to build a model to predict the purchase amount of customers against various products which will help them to create personalized offers for customers against different products.

Proposed Solution

Data Preprocessing : This step performs all pre-processing steps such as data manipulation, data filling, converting categorical into numeric, and all processes.

The EDA process involves performing

1. Univariate Analysis
2. Bivariate analysis
3. Removing Missing values if any / Outlier treatment
4. Machine Learning : Probability of purchase. and also check if the model is to be underfitting or overfitting if it has then solves this by using cross-validation technique, or perform hyperparameter tuning to improve model performance.

Descriptive Analysis

```
# Datatype info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 11 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  int64
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  int32
 3   Age                         550068 non-null  int32
 4   Occupation                  550068 non-null  int64
 5   City_Category               550068 non-null  int32
 6   Stay_In_Current_City_Years  550068 non-null  int32
 7   Marital_Status              550068 non-null  int64
 8   Product_Category_1          550068 non-null  int64
 9   Product_Category_2          550068 non-null  int32
 10  Purchase                    550068 non-null  int64
dtypes: int32(5), int64(5), object(1)
memory usage: 35.7+ MB
```

```
# Null values
df.isnull().sum()
```

```
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
dtype: int64
```

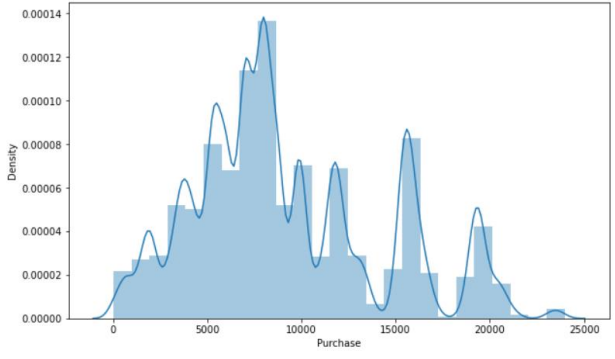
```
# Unique values
df.apply(lambda x: len(x.unique()))
```

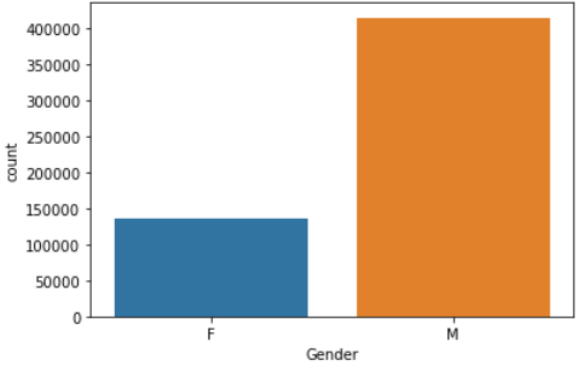
```
User_ID                       5891
Product_ID                    3631
Gender                           2
Age                              7
Occupation                      21
City_Category                    3
Stay_In_Current_City_Years       5
Marital_Status                   2
Product_Category_1              20
Product_Category_2              18
Product_Category_3              16
Purchase                     18105
dtype: int64
```

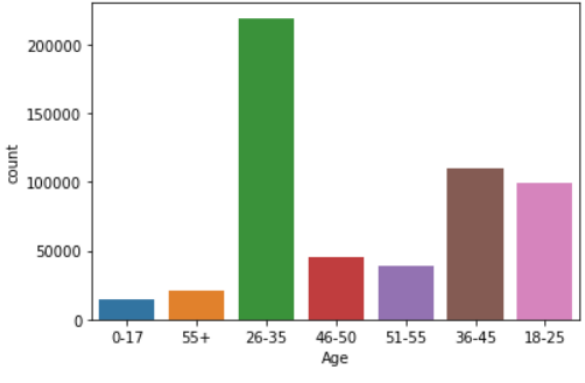
```
= # statistical info
df.describe()
```

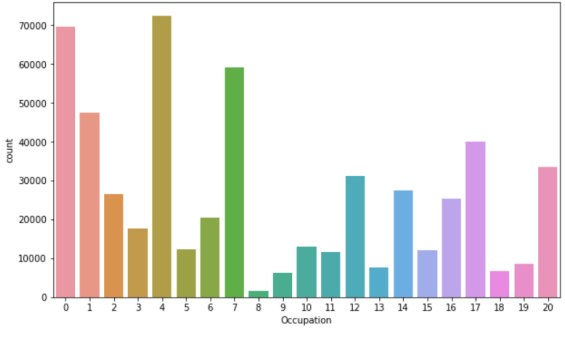
	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	376430.000000	166821.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9.842329	12.668243	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5.086590	4.125338	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	2.000000	3.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5.000000	9.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	9.000000	14.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	15.000000	16.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	18.000000	18.000000	23961.000000

Exploratory Data Analysis





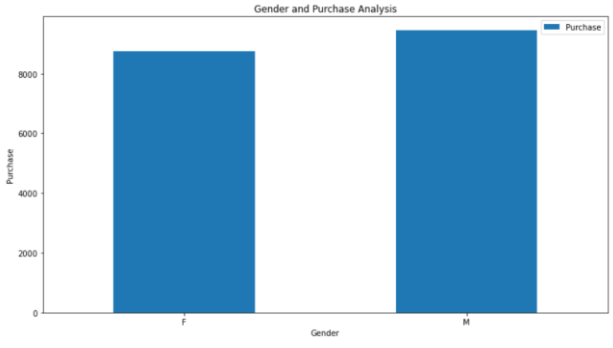


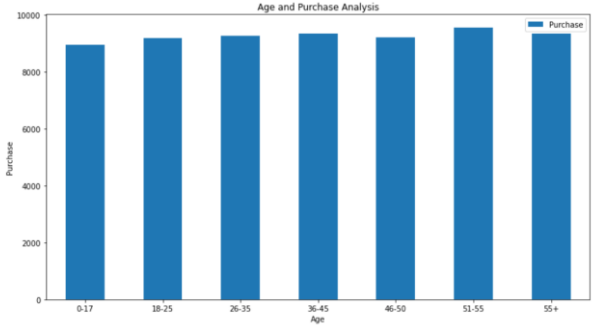


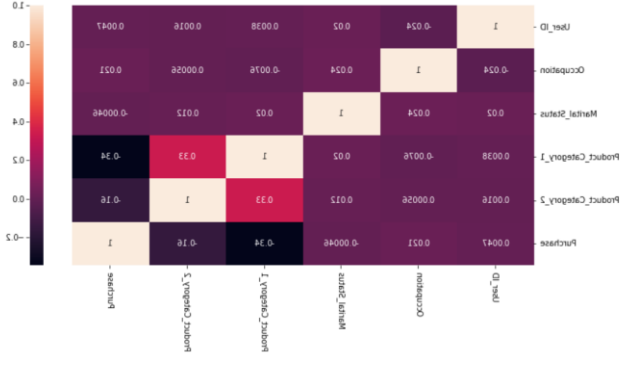
Observations

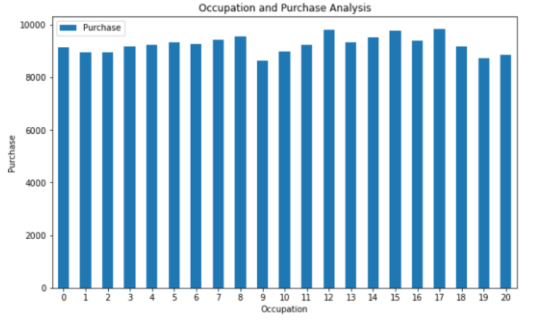
- Graph 1 We can see the Highest purchase range is between 5000 to 10000
- In graph 2 we can see that M ie Males have Placed more orders, Maybe they are more interested in black Friday sales than females.
- In graph 3 we can see that the most orders were placed by the age group 26-35
- In graph 4 we can see that occupations 0, 4, 7, have placed more orders

Exploratory Data Analysis









Observations

- Graph 5 We can see That males have higher purchase amount than females
- In graph 6 we can see age group 51-55 have higher purchase or spending capacity
- In graph 7 we can see the correlation between columns
- In graph 9 we can see purchases made by every occupation type

Preprocessing the Dataset

- Looking for null values we can see that product category 2 and 3 have null values we can treat them by simply dropping the product category 3 column because it has many missing values more than 60%, we will fill the product category 2 column with the mean value of that column.
- Outlier Treatment: We can remove the outliers by using techniques like IQR or z-score here I have used IQR technique to remove outliers in purchase column

```
df.isnull().sum()
```

```
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
```

Machine Learning Modelling

Encoding

- Method used One hot encoding on Gender, Age, CityCategory, StayInCurrentCityYears, Occupation, ProductCategory1, ProductCategory2 columns as this gave a good result.

Input Split

- Splitting the dataset between target variable ie Purchase and input variables ie rest of the columns as X and y and running them through train_test_split

```
print(X.shape)
print(y.shape)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(547391, 76)
(547391,)
(383173, 76)
(383173,)
(164218, 76)
(164218,)
```

Machine Learning Modelling

We apply machine learning agorithms on the dataset lets check scores

- Linear Regression

```
training score =  0.6347153750403496
Testing score =  0.6350016170765882
Mean Absolute error = 2260.2782310954826
Mean Squared error = 8934811.918436665
Root Mean Squared error= 2989.1155746201357
R2score =  0.6350016170765882
```

- Ridge Regression

```
training score = 0.6347213760941048
Testing score = 0.6350018572367329
Mean Absolute error = 2260.399594864551
Mean Squared error = 8934806.039545625
Root Mean Squared error= 2989.114591236948
R2score =  0.6350016170765882
```

- Lasso Regression

```
training score = 0.6347213760940762
Testing score = 0.6350018580884018
Mean Absolute error = 2260.399604181418
Mean Squared error = 8934806.018697584
Root Mean Squared error= 2989.1145877496206
R2score =  0.6350016170765882
```

- Random Forest Regression

```
training score = 0.7386877920874092
Testing score = 0.6315343220780456
Mean Absolute error = 2260.399594864551
Mean Squared error = 8934806.039545625
Root Mean Squared error= 2989.114591236948
R2score =  0.6350016170765882
```

Conclusion

The Model performs almost similarly in all the algorithms