Black Friday Prediction

-by Vishnu Derkar

Content

Introduction

1.1 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.

1.2 Data:

The details of data attributes in the dataset are as follows –

* 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 belong to other category also (Masked)
* Product_Category_3: Product may belong to other category also (Masked)
* Purchase: Purchase Amount (Target Variable)

First few rows of the data are shown below:

Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
P00069042	F	0-17	10	A	2	0	3	NaN	NaN
P00248942	F	0-17	10	A	2	0	1	6.0	14.0
P00087842	F	0-17	10	A	2	0	12	NaN	NaN
P00085442	F	0-17	10	A	2	0	12	14.0	NaN
P00285442	M	55+	16	C	4+	0	8	NaN	NaN

Checking general information and description of data using `df.describe()` and `df.info()`.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 783667 entries, 0 to 233598
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     783667 non-null  int64
 1   Product_ID                  783667 non-null  object
 2   Gender                      783667 non-null  object
 3   Age                         783667 non-null  object
 4   Occupation                  783667 non-null  int64
 5   City_Category               783667 non-null  object
 6   Stay_In_Current_City_Years  783667 non-null  object
 7   Marital_Status              783667 non-null  int64
 8   Product_Category_1          783667 non-null  int64
 9   Product_Category_2          537685 non-null  float64
 10  Product_Category_3          237858 non-null  float64
 11  Purchase                    550068 non-null  float64
dtypes: float64(3), int64(4), object(5)
memory usage: 77.7+ MB
```

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
count	7.836670e+05	783667.000000	783667.000000	783667.000000	537685.000000	237858.000000	550068.000000
mean	1.003029e+06	8.079300	0.409777	5.366196	9.844506	12.668605	9263.968713
std	1.727267e+03	6.522206	0.491793	3.878160	5.089093	4.125510	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	2.000000	3.000000	12.000000
25%	1.001519e+06	2.000000	0.000000	1.000000	5.000000	9.000000	5823.000000
50%	1.003075e+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

From the above description of data columns Product_Category _2, Product_Category _3 and Purchase can be seen to have missing data.

Methodology

2.1 Pre Processing:

Data pre-processing is an important step in analysing the data for building a good model. Data pre-processing is used to transform raw data to understandable data. To do this we have certain operations such as dropping unnecessary columns, renaming of columns, mapping of data range into integers ones.

In pre-processing we have dropped the columns which won't be adding any value to the model i.e. "UserID", "ProductID". After having done that we have one hot encoded the Age variable using mapping technique. One hot encoding is used here for converting the categorical variable such as the range of age into integer variable which our model can understand. Similarly, I have converted the values of Stay_In_The_City into an integer type from categorical type variable. The Gender column is simply mapped from as male: 1 and female: 0.
At the end I have only renamed the columns "Product_Category_1" as "cat1", "Product_Category_2" as "cat2", "Product_Category_3" as "cat3".

2.1.1 Dealing with missing values:

To findout the missing values in the dataset we use `df.isnull().sum()` which provides us with following data.

```
Gender                          0
Age                             0
Occupation                      0
City_Category                   0
Stay_In_Current_City_Years      0
Marital_Status                  0
cat1                            0
cat2                       245982
cat3                       545809
Purchase                   233599
dtype: int64
```

To deal with these missing values we have to fill those nan values but before let's see what type unique values are present in the missing data columns.

```
df.Purchase.unique()
```

```
array([ 8370., 15200.,  1422., ...,   123.,   613.,    nan])
```

```
df.cat2.unique()
```

```
array([nan,  6., 14.,  2.,  8., 15., 16., 11.,  5.,  3.,  4., 12.,  9.,
       10., 17., 13.,  7., 18.])
```

```
df.cat3.unique()
```

```
array([nan, 14., 17.,  5.,  4., 16., 15.,  8.,  9., 13.,  6., 12.,  3.,
       18., 11., 10.])
```

Here the cat2 and cat3 are having few values and to fill the missing values we choose the mode method to fill them. This is mostly because these are categorical values and mode method will give generalise the pattern better than the mean and median method.
For the Purchase column we see that it ranges from 12 to 23961. So we will use the mean value of the purchase columns to fill the missing values.

Rechecking the missing values again.

```
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
cat1                          0
cat2                          0
cat3                          0
Purchase                      0
dtype: int64
```

2.1.1 Dealing with categorical data:

For dealing with the categorical variable we will use the dummy method. In this dataset we have City_Category as a categorical variable. So we use pd.get_dummies function to create a dataframe by the name city.

```
city = pd.get_dummies(df['City_Category'],drop_first=True)
city
```

	B	C
0	0	0
1	0	0
2	0	0
3	0	0
4	0	1
...	...	...
233594	1	0
233595	1	0
233596	1	0
233597	0	1
233598	1	0

783667 rows × 2 columns

Now that we have this dataframe we will concatenate it with the df dataframe using pd.concat function. After concatenation operation we will drop the City_Category column and convert the columns “B” and “C” from categorical to integer and recheck using df.dtypes function.

2.1.3 Creating a check point:

At the end of data pre-processing we will create a check point by the name df_i. this will ensure that even if make a mistake in further steps but our progress till this check point will be safe.

```
df_i.head(10)
```

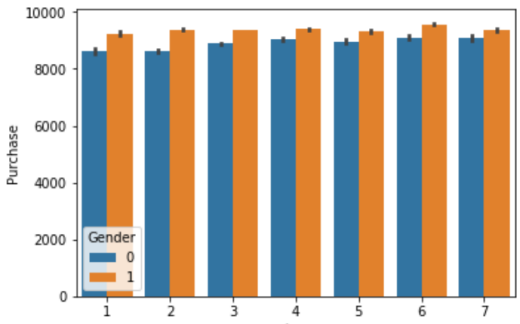
	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	cat1	cat2	cat3	Purchase	B	C
0	0	1	10	2	0	3	8.0	16.0	8370.0	0	0
1	0	1	10	2	0	1	6.0	14.0	15200.0	0	0
2	0	1	10	2	0	12	8.0	16.0	1422.0	0	0
3	0	1	10	2	0	12	14.0	16.0	1057.0	0	0
4	1	7	16	4	0	8	8.0	16.0	7969.0	0	1
5	1	3	15	3	0	1	2.0	16.0	15227.0	0	0
6	1	5	7	2	1	1	8.0	17.0	19215.0	1	0
7	1	5	7	2	1	1	15.0	16.0	15854.0	1	0
8	1	5	7	2	1	1	16.0	16.0	15686.0	1	0
9	1	3	20	1	1	8	8.0	16.0	7871.0	0	0

2.2 Data Visualization:

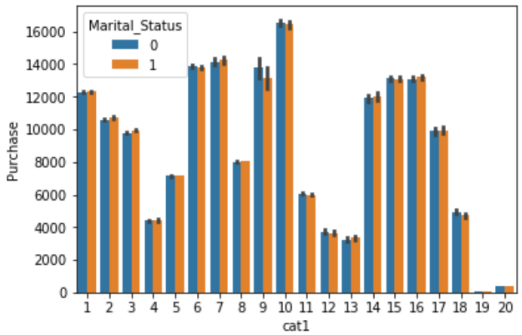
2.2.1 Plotting target variable against other variables:

By plotting the target variable against other variables, we are trying to understand the data and its distribution with respect to the target variable.

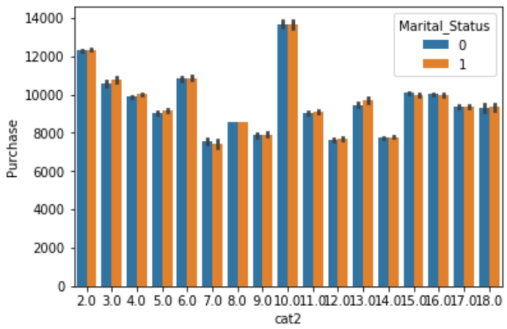
Age Vs Purchase



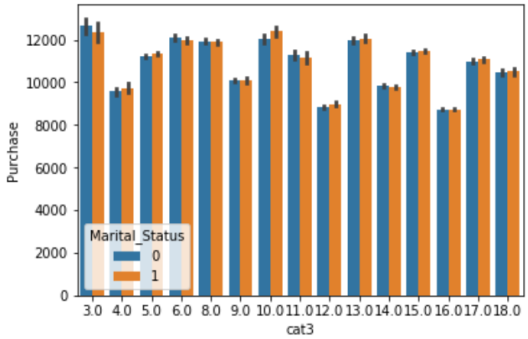
Cat1 Vs Purchase



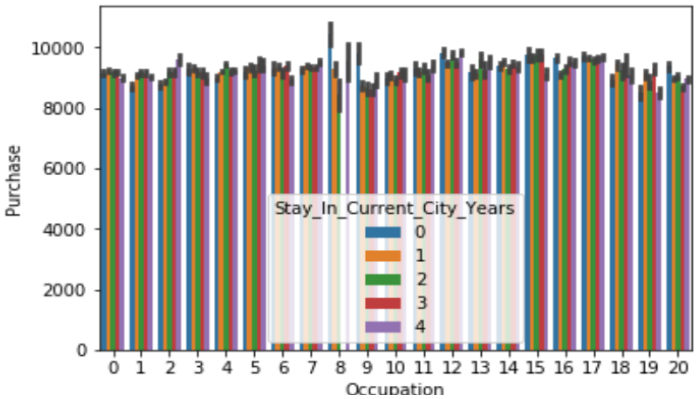
Cat2 Vs Purchase



Cat3 Vs Purchase



Occupation Vs Purchase



The above plots shows that irrespective of the age group people have been making purchase of more or less the same amount but the made by men are more than women. Like Age the occjupation of people is also not likely to affect the purchasing power of the people.
For Cat1, Cat2, Cat3 vs Purchase we see the marital status makes no difference.
Cat1 contains some of the least purchased products and also shows a lot of variability of being purchased by customers. Cat2 shows a little less variability of purchase than Cat1. In Cat2 least purchase goes around 8000. In Cat3 a better stability is seen as compared to Cat2 and Cat1. The least purchase is just above 8000.

2.2.1 Scaling Data:

For this dataset we choose standard scaling of data using Standard Scaler. StandardScaler removes the mean and scales each feature/variable to unit variance. StandardScaler can be influenced by outliers (if they exist in the dataset) since it involves the estimation of the empirical mean and standard deviation of each feature.

Modelling

3.1 Model Preparation:

After completing pre-processing of data now we go forward with model preparation. For this data we have selected four models Decision Tree, Random Forest, XGBoost.

Decision Trees (DTs) are non-parametric supervised learning method used for classification and regression. Decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules.

Random forest is a supervised learning algorithm. It creates an ensemble of decision trees using bagging method. Bagging method is generally improving the result of decision tree over time.

XGBoost Algorithm is an ensemble of decision tree which uses gradient boosting framework. This algorithm is seen to generally performs better than other algorithm.

For model preparation we have defined a function `simple_modeling`, which will give us the direct output of the mean absolute error. From which we can decide which model to select.

3.2 Mean Absolute Error:

Mean absolute error (**MAE**) is a measure of errors between paired observations expressing the same phenomenon. Examples of *Y* versus *X* include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. MAE is calculated as:

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}.$$ [1]

We got the following result for mean absolute error for the models:

For Decision Tree:

```
Mean Squared Error for Decision Tree 12065483.71055182
Mean Absolute Error for Decision Tree 2590.882988266301
```

For Random Forest:

```
Mean Squared Error for Random Forest: 10498849.0118499
Mean Absolute Error for Random Forest: 2490.418418350609
```

For XgBoost:

```
Mean Squared Error for XGBoost: 9276665.625133606
Mean Absolute Error for XGBoost: 2410.5585039343177
```

3.3 Model Selection:

Based on the above results we can see that XGBoost has the least Mean Squared Error. So XGBoost is the selected model for this dataset.