DSC-478 PROJECT SUBMISSION

Black Friday Sales Prediction

by

Sidhant Thakur (2020181)

Dhruv Chandulal Dobariya (2069889)

Saransh Thakur (2020070)

Under the guidance of: Dr. Bamshad Mobasher

DePaul University

16th November 2022

Dataset Summary:

We got this dataset from the Dataset Link. The dataset contains 12 variables in total.

We have a mixture of Categorical, Numerical, and binary variables.

1. User_ID: ID of user
2. Product_ID: ID of product
3. Gender : Sex of the user
4. Age: Age of user
5. Occupation: Occupation
6. City_Category: City (A, B or C)
7. Stay_In_Current_City_Years: Number of years' user has been staying in current city
8. Marital_Status: Marital status of user
9. Product_Category_1: Category of product
10. Product_Category_2: Category in which product can also be fit in
11. Product_Category_3: Category in which product can also be fit in
12. Purchase: Total purchase amount in 1 shopping

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	NaN	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	NaN	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	NaN	7969

This is how the data looks like, then I get the description of all the data by using describe ().

```
[ ] data.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

The size and info about dataset are as below:

```
data.shape

(550068, 12)
```

```
Run cell (⌘/Ctrl+Enter)
cell executed since last change

executed by SIDHANT THAKUR
4:33 PM (1 hour ago)
executed in 0.282s
```

```
ore.frame.DataFrame'>
58 entries, 0 to 550067
tal 12 columns):
                                  Non-Null Count   Dtype
---  ------                       --------------   -----
 0   User_ID                      550068 non-null  int64
 1   Product_ID                   550068 non-null  object
 2   Gender                       550068 non-null  object
 3   Age                          550068 non-null  object
 4   Occupation                   550068 non-null  int64
 5   City_Category                550068 non-null  object
 6   Stay_In_Current_City_Years   550068 non-null  object
 7   Marital_Status               550068 non-null  int64
 8   Product_Category_1           550068 non-null  int64
 9   Product_Category_2           376430 non-null  float64
 10  Product_Category_3           166821 non-null  float64
 11  Purchase                     550068 non-null  int64
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

Project Goal:

The aim of this project is to build a prediction model which will be implemented to identify user purchase behavior and throw offers accordingly.

Methods used:

There are several approaches to this project. We decided to use Linear regression, Decision Tree Regressor, Random Forest Regressor, and XGBoost Regressor for this project. Using KDD process, we cleaned our data first. We did exploratory analysis by plotting visualizations to get some insights about data. After cleaning and exploratory analysis, we split our data into 2 which are test and train. Test and train splits were used further in the application of algorithm.

Data Cleaning:

Data have 31% null value in product_catrogory_2 and 69% in the product_category_3

Checking percentage of Null Value

```
[ ] data.isnull().sum()/data.shape[0]*100

    User_ID                       0.000000
    Product_ID                    0.000000
    Gender                        0.000000
    Age                           0.000000
    Occupation                    0.000000
    City_Category                 0.000000
    Stay_In_Current_City_Years    0.000000
    Marital_Status                0.000000
    Product_Category_1            0.000000
    Product_Category_2           31.566643
    Product_Category_3           69.672659
    Purchase                      0.000000
    dtype: float64
```

There are 31% null values in the Product_Category_2 and 69% null values in the Product_Category_3

To handle the missing value, we replace it with the mean and transformed values of categorical variable to dummy where required.

Handling missing value and replacing it with mean

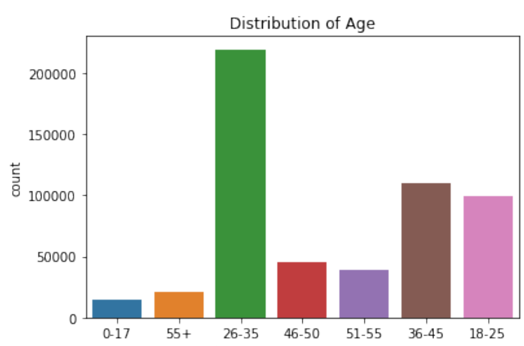
```
[ ] mean1 = df.Product_Category_2.mean()
    mean2 = df.Product_Category_3.mean()
    df['Product_Category_2'] =df['Product_Category_2'].fillna(mean1).astype('int64')
    df['Product_Category_3'] =df['Product_Category_3'].fillna(mean2).astype('int64')
```

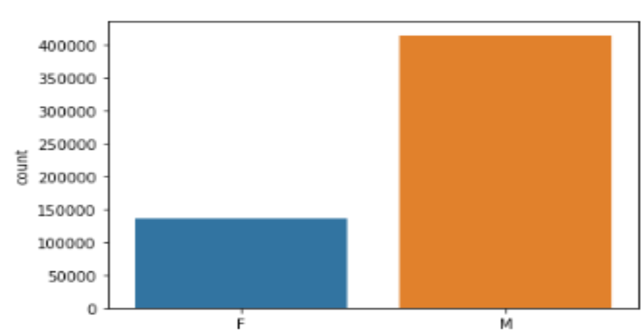
```
[ ] df.isnull().sum()
```

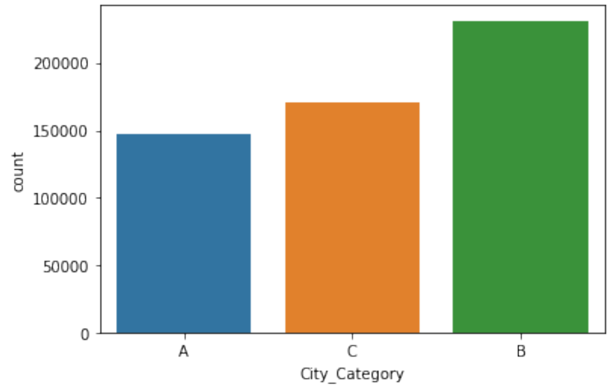
```
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Marital_Status                0
Product_Category_1            0
Product_Category_2            0
Product_Category_3            0
Purchase                      0
Stay_In_Current_City_Years_0  0
Stay_In_Current_City_Years_1  0
Stay_In_Current_City_Years_2  0
Stay_In_Current_City_Years_3  0
Stay_In_Current_City_Years_4+ 0
dtype: int64
```

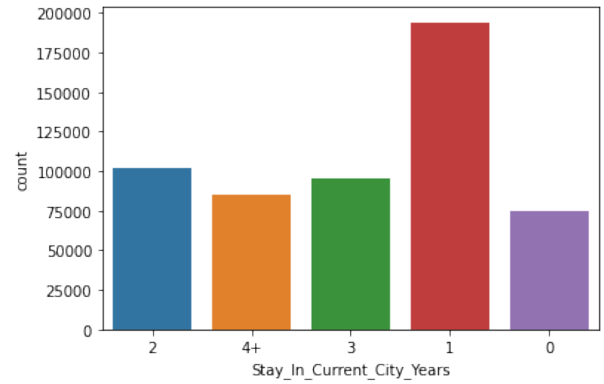
Exploratory Analysis:

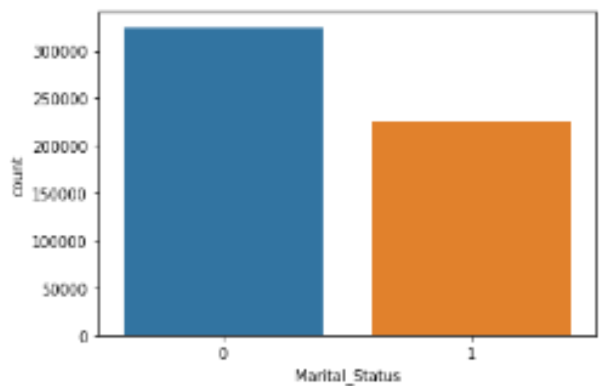
Using Python and its libraries like matplotlib we did exploratory analysis on various variables of our dataset. The visualizations below give a better idea about variables.

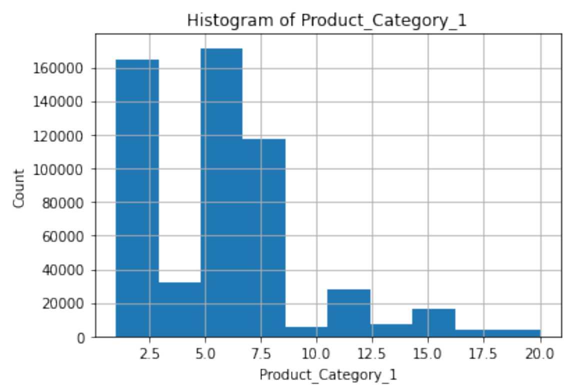


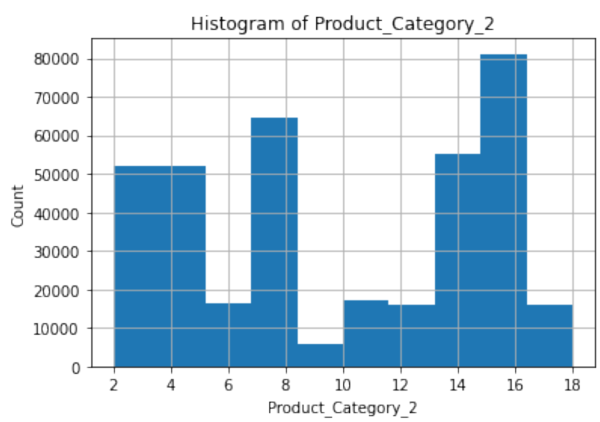


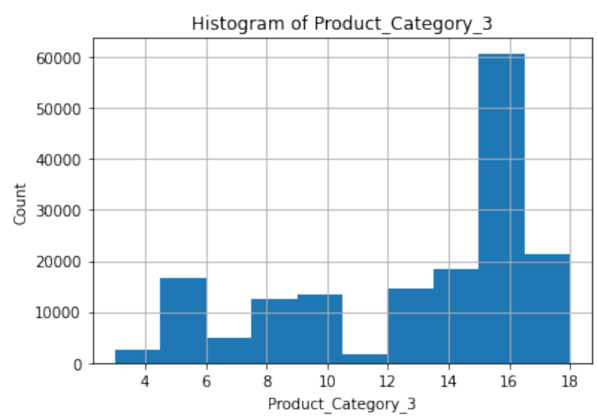












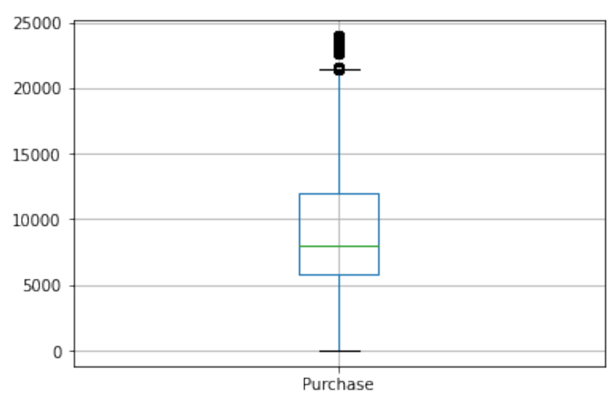
Insights from EDA

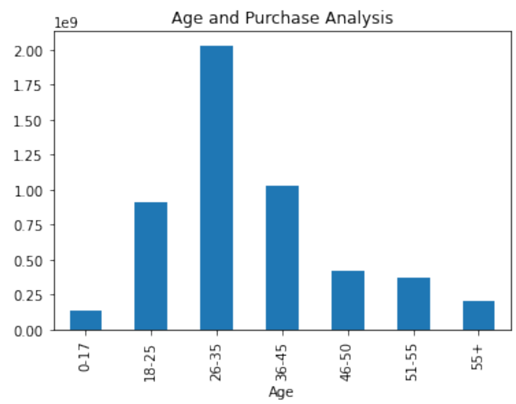
- The age group 26-35 makes the greatest number of purchases.
- On average, males spend more money on purchases than females.
- It is observed that city category B has made the greatest number of purchases.
- It appears that the longer someone lives in that city, the less likely they are to purchase new items. As a result, if someone is new in town and needs a

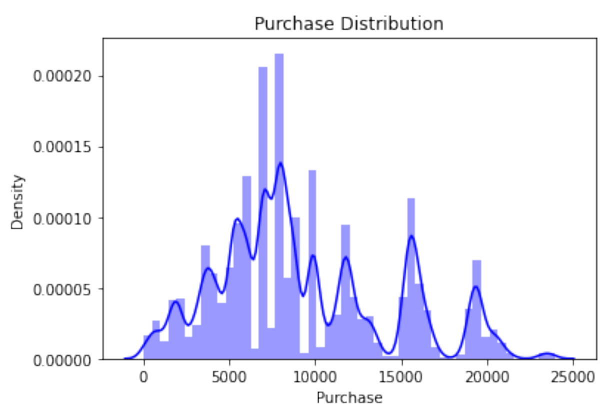
lot of new things for their house, they will take advantage of the low prices on Black Friday to get everything they need.

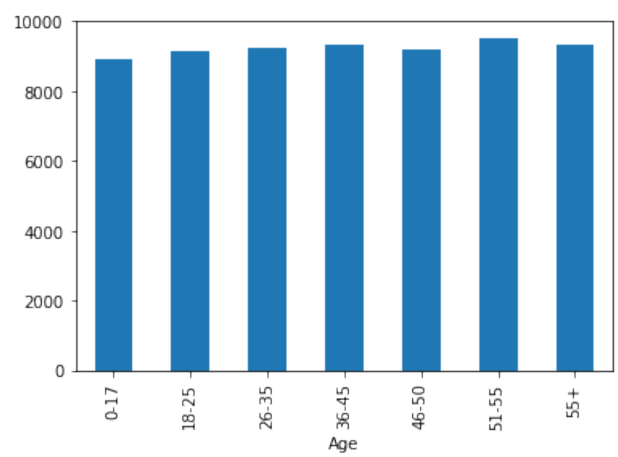
Target Variable:

Our target variable is Purchase as we are predicting sales based on multi-variate analysis. The below visualizations throw some light on Purchase variable which includes but is not limited to Standard Distribution and Skewness. The target variable is positively skewed and also has some outliers.

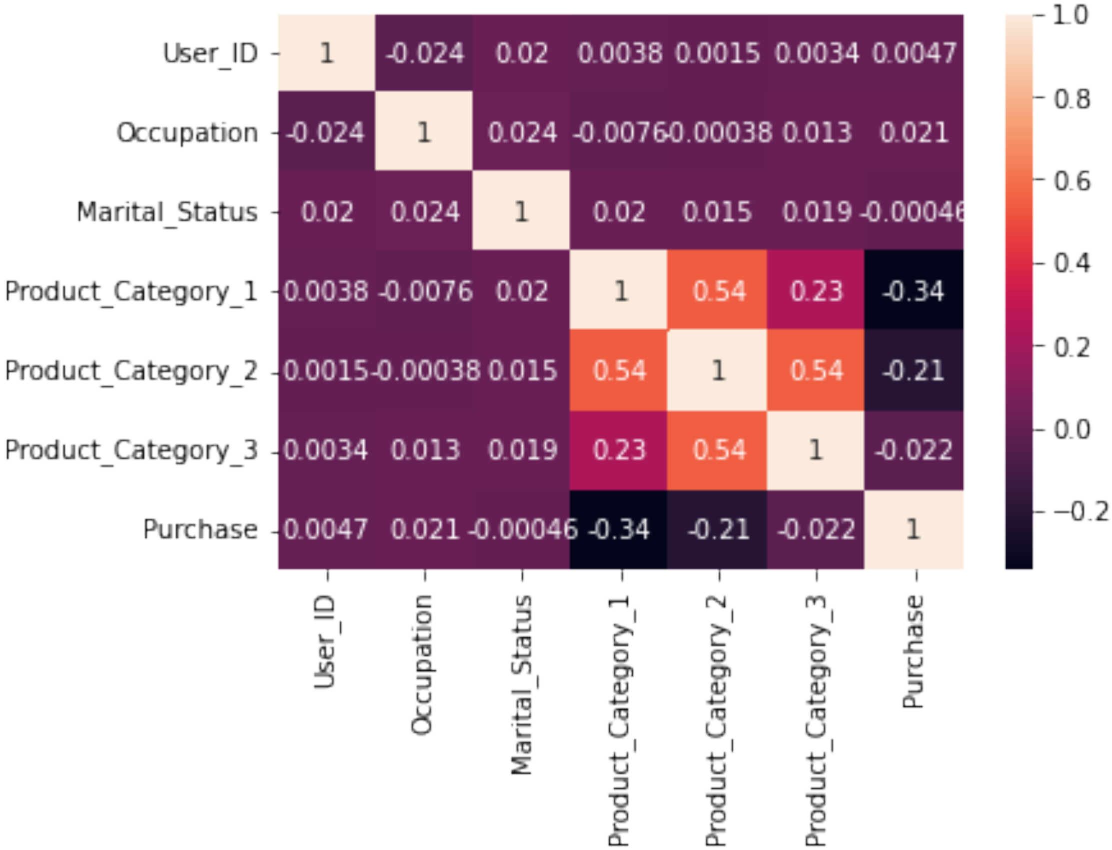








Checking Collinearity using collinearity matrix or heatmap. We can see that there is some collinearity between the group of product category variables.



Application of Algorithms:

Using Scikit-learn library of Python we applied Linear Regression, Ridge regression, Lasso Regression, Decision Tree Regressor, Random Forest Regressor, XGBoost Regressor.

To perform our analysis, we split data using train and test split.

Splitting data into training and testing sets

```
[42] X = df.drop("Purchase",axis=1)
     y=df['Purchase']
```

```
[43] from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
```

We used RMSE and accuracy for our criteria of evaluation for models.

⇨ Linear Regression:

```
[44] from sklearn.linear_model import LinearRegression
```

```
[45] # Create linear regression object
     lr = LinearRegression()

     # Train the model using the training set
     lr.fit(X_train,y_train)
```

```
LinearRegression()
```

```
[46]
     lr.intercept_
```

```
9771.710228149399
```

```
[47]
     lr.coef_
```

```
array([ 5.01001319e+02,  1.23174920e+02,  5.91253477e+00,  3.42425874e+02,
       -6.37363865e+01, -4.11666400e+02, -7.50016112e+01,  1.12594084e+02,
       -2.12600529e+01, -4.43380422e-01,  2.82172911e+01, -5.47551593e+00,
       -1.03834182e+00])
```

```
[48] y_pred = lr.predict(X_test)
```

```
[49] from sklearn.metrics import mean_absolute_error,mean_squared_error, r2_score
```

```
[50] mean_absolute_error(y_test, y_pred)
```

```
3602.9252605187157
```

```
[51] mean_squared_error(y_test, y_pred)
```

```
22013414.220225006
```

```
print("Accuracy of the LinearRegression model comes to be: \n ")
r2_score(y_test, y_pred)
```

```
Accuracy of the LinearRegression model comes to be:

0.12972381106967878
```

```
[53] rmse_train = np.sqrt(mean_squared_error(y_test, y_pred))
     print("RMSE on Test Data: ", rmse_train)
```

```
RMSE on Test Data:  4691.8455025954345
```

We applied Linear Regression and we got RMSE on test data is 4691 and accuracy of the linear regression model is 0.1297

⇨ Lasso Regression:

```
[54] from sklearn.linear_model import Lasso
     reg2 = Lasso()

[55] reg2.fit(X_train,y_train)

     Lasso()

[56] pred2 = reg2.predict(X_test)

[57] pred2

     array([ 7089.28409483,  2704.04577618, 12327.95754969, ...,
            11701.04040913,  8737.91334058, 11611.52846247])

[58] print("Accuracy of the LassoRegression model comes to be: \n ")
     print(reg2.score(X_train,y_train))

     Accuracy of the LassoRegression model comes to be:

     0.12881817953022945

[63] rmse_train = np.sqrt(mean_squared_error(y_test, pred2))
     print("RMSE on Test Data: ", rmse_train)

     RMSE on Test Data:  4691.8601575414505
```

We applied Lasso Regression and we got RMSE on test data is 4691 and accuracy of the Lasso regression model is 0.1297

⇨ Ridge Regression:

```
[59] # Importing model
     from sklearn.linear_model import Ridge
     reg3 = Ridge()

[60] reg3.fit(X_train, y_train)

     Ridge()

[61] pred3= reg3.predict(X_test)

[62] print("Accuracy of the RidgeRegression model comes to be: \n ")
     print(reg3.score(X_train,y_train))

     Accuracy of the RidgeRegression model comes to be:

     0.12881933091851294

[64] rmse_train = np.sqrt(mean_squared_error(y_test, pred3))
     print("RMSE on Test Data: ", rmse_train)

     RMSE on Test Data:  4691.845506376657
```

We applied Ridge Regression and we got RMSE on test data is 4691 and accuracy of the Lasso regression model is 0.1297

⇨ **Decision Tree Regressor:**

```
[ ] from sklearn.tree import DecisionTreeRegressor

    # create a regressor object
    regressor = DecisionTreeRegressor(random_state = 0)
```

```
[ ] regressor.fit(X_train, y_train)
```

```
DecisionTreeRegressor(random_state=0)
```

```
[ ] dt_y_pred = regressor.predict(X_test)
```

```
[ ] mean_absolute_error(y_test, dt_y_pred)
```

```
2343.4309253102556
```

```
[ ] mean_squared_error(y_test, dt_y_pred)
```

```
10993262.43644692
```

```
[ ] r2_score(y_test, dt_y_pred)
```

```
0.56539342596334
```

```
[ ] from math import sqrt
    print("RMSE of Decision tree regressor Model is ",sqrt(mean_squared_error(y_test, dt_y_pred)))
```

```
RMSE of Decision tree regressor Model is  3315.608908850216
```

The RMSE of Decision tree regressor model is 3315.60 and accuracy of the Decision tree regression model is 56.53

⇨ **Random Forest Regressor:**

```
[ ] from sklearn.ensemble import RandomForestRegressor

    # create a regressor object
    RFregressor = RandomForestRegressor(random_state = 0)
```

```
[ ] RFregressor.fit(X_train, y_train)
```

```
RandomForestRegressor(random_state=0)
```

```
[ ] rf_y_pred = RFregressor.predict(X_test)
```

```
[ ] mean_absolute_error(y_test, rf_y_pred)
```

```
2210.292709670269
```

```
[ ] mean_squared_error(y_test, rf_y_pred)
```

```
9205334.953478044
```

```
[ ] r2_score(y_test, rf_y_pred)
```

```
0.6360771781698631
```

```
[ ] from math import sqrt
    print("RMSE of Random forest regresion Model is ",sqrt(mean_squared_error(y_test, rf_y_pred)))
```

```
RMSE of Random forest regresion Model is  3034.0294912011063
```

Observed RMSE for Random Forest Regressor is 3034029 and accuracy of the random forest regression model is 63.60

⇨ XGBoost Regressor:

```
[ ] from xgboost.sklearn import XGBRegressor
    xgb_reg = XGBRegressor(learning_rate=1.0, max_depth=6, min_child_weight=40, seed=0)

    xgb_reg.fit(X_train, y_train)
```

```
[04:10:30] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
XGBRegressor(learning_rate=1.0, max_depth=6, min_child_weight=40, seed=0)
```

```
[ ] xgb_y_pred = xgb_reg.predict(X_test)
```

```
[ ] mean_absolute_error(y_test, xgb_y_pred)
```

```
2136.779779876488
```

```
[ ] mean_squared_error(y_test, xgb_y_pred)
```

```
8209058.072280257
```

```
[ ] r2_score(y_test, xgb_y_pred)
```

```
0.6754638920441516
```

```
[ ] from math import sqrt
    print("RMSE of XGBoost Regression Model is ",sqrt(mean_squared_error(y_test, xgb_y_pred)))
```

```
RMSE of XGBoost Regression Model is  2865.1453841437537
```

XGBoost Regression Model gave RMSE 2695.14, which is the lowest of all.

Random forest regression model has a accuracy is 67.54, which is highest in all model

Conclusion:

- Males showed more interest in the black Friday sales than as compared to that of females.
- Age group 26-35 were more active than other age groups.
- People from 20 different occupations showed their interest in the black Friday sales.
- People from city B were more as compared to city A & C.
- Maximum people are staying in their respective city from 1 year.

- 59.09% people were unmarried and 40.91 % were married among total participation.
- There are 18 subcategories of products in category 1.
- There are 17 subcategories of products in category 2.
- There are 15 subcategories of products in category 3.

Accuracy and RMSE value of different models used

- By using Linear Regression model

Accuracy achieved: 12.97 and RMSE on test data is 4691

- By using Lasso Regression model

Accuracy achieved: 12.97 and RMSE on test data is 4691

- By using Ridge Regression model

Accuracy achieved: 12.97 and RMSE on test data is 4691

- By using Decision Tree Regressor model

Accuracy achieved: 56.53 and RMSE on test data is 3315.60

- By using Random Forest Regressor model

Accuracy achieved: 63.60 and RMSE on test data is 3034029

- By using XGBoost Regressor model

Accuracy achieved: 67.54 and RMSE on test data is 2695.14, which is the lowest of all.

Performance of XGBoost Regressor is better in terms of accuracy as compared to linear regression, lasso regression, ridge regression, random forest regressor and Decision tree regressor.

Moreover, XGBoost has a lowest RMSE value in all the model.