**Black Friday Sales Analysis**

**using various classifiers and regressors**

*submitted by*

**Pallav Gupta**

**(16BCE0941)**

**B. TECH**

# COMPUTER SCIENCE ENGINEERING



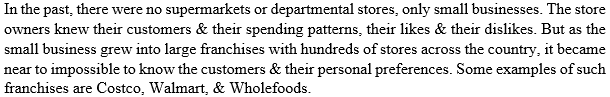
**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**Abstract**

Amid the deals on Black Friday, all the retailers are packed. Many items are discounted with limits & clients surge in purchase of the items. It’s hard for clients to purchase the items even with a strong arrangement. In any case, the shop proprietors face significantly more trouble on controlling the group with constrained staff & in focusing on imminent clients. A few methods have been utilized to handle this issue, yet they are not unreasonably effective. A model for predicting is a system that demonstrated promise in taking care of this issue. I focus on predicting models to build an exact & proficient algorithm to analyze the clients spending before & yield the future spending of the client with similar features. Distinctive data analysing techniques such as regressors, NN & classifiers to develop a model for prediction are actualized & an examination is implemented dependent on their exhibition & exactness of predicting. These machine learning techniques are executed using various algorithms to find the optimal prediction. I applied six distinct ML algorithms. Furthermore, I apply the visuals & data pre-process to obtain the optimum results.

**Keywords:** Black Friday, Model for predicating, Regressors, NN, ML

1. **Introduction**



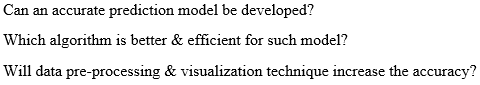
These stores with no legitimate information of their client base are attempting to fulfil the client needs. Along these lines, predicating models are expected to all the more likely comprehend client inclinations.

Building a predicating model relies upon different features, for example, the location & the time. The day, Black Friday, after Thanksgiving is the biggest day for shopping cheaply in United States of America. This day in the wake of Thanks giving which denotes the start of the shopping for Christmas season.

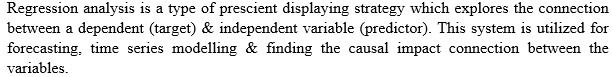




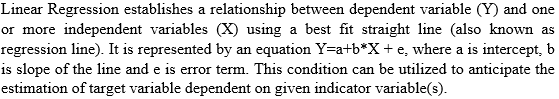
**Problem statement**



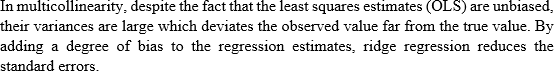
1. **Literature survey**

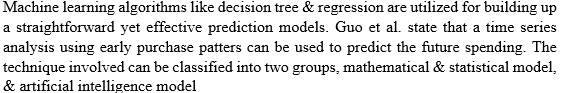
The shape of regression line, the type of dependent variable & number of independent variable. [11][12]

1. **Linear Regression**



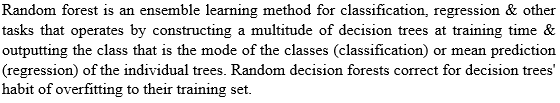
1. **Ridge Regression**

Ridge Regressor is a method utilized when the information experiences multicollinearity (autonomous factors are exceptionally related).

1. **Decision Tree**
2. **XGBoost**

The XGB model executes the step-by-step, ridge regression internally which progressively chooses the features & remoevs the multicollinearity with the features.

1. **Random forest**



The serious issue with the current predicting model is that the information utilized for advancement contains several anomalies such as unaccounted for values or wrong data. Additionally, choice of optimal algorithm plays a major role in building a precise model.

1. **Proposed system**

Our system involves the application of machine learning techniques to predict the testing values in 4 steps-

1. **Data analysing**

In this stage I will just analysis various components of our data like mean, median, standard deviation, frequency etc. Also, I’ll find skew, kurtosis followed by correlation matrix with respect to Purchase values.

1. **Data pre-processing**

This stage will involve calculating all the NA values & replacing them with some integer so that processing can be carried out. Later I can also display the unique value frequencies of all the columns & finally send this data into a modified train & test csv files.

1. **Machine learning techniques**

After pre-processing the data, I will use the modified train & test files to apply the above given algorithms to find which is the best algorithm to calculate the purchase values by calculating the minimum rmse values.

1. **Rule Based Learning**

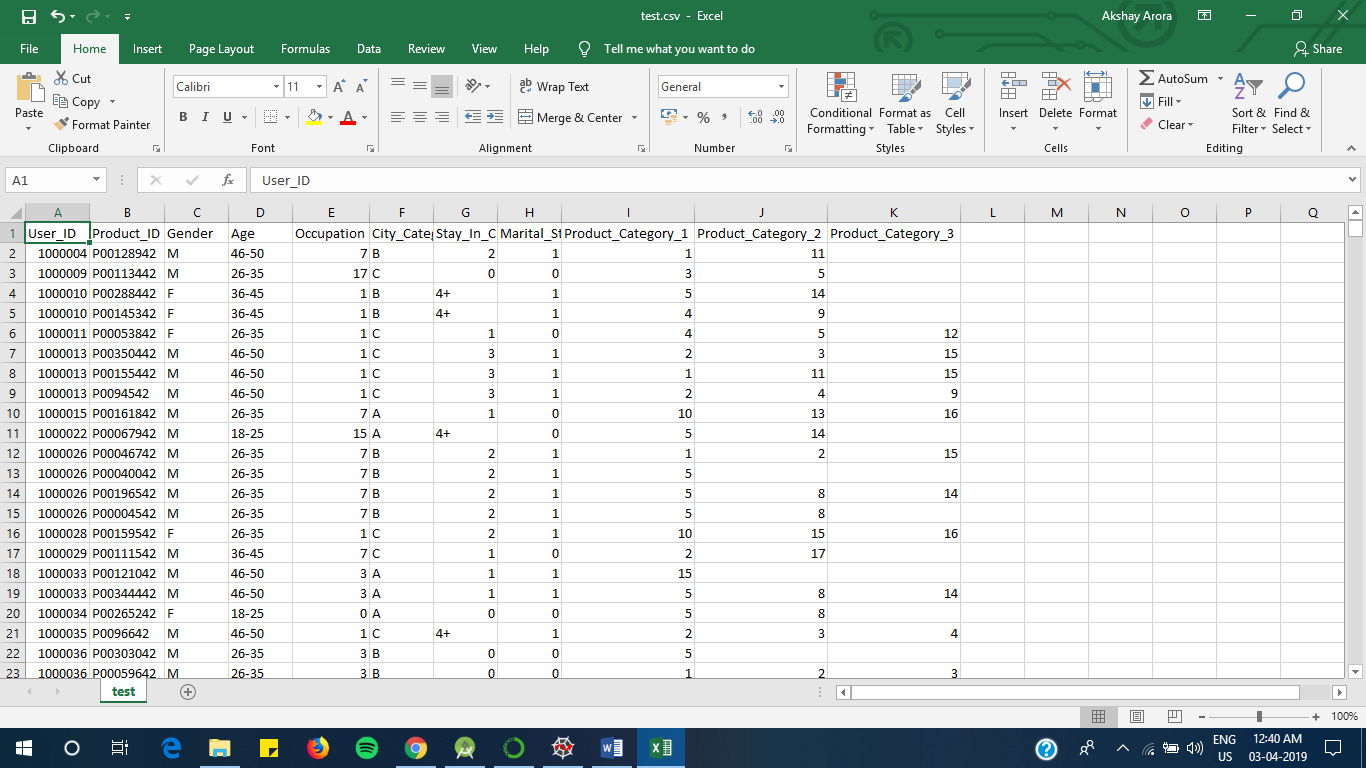
At the end I will apply rule-based learning to get the best possible rules & apply them to the dataset & finally apply it to the best obtained algorithm to get the perfect rmse scores.

**Dataset Used-**

Our dataset is the Black Friday Sales Dataset in Kaggle. In this dataset I have the information about the Age, Occupation, City, Duration stayed, Marital status, the quantity of products bought of various types & the total amount spent. I are using these inputs to find the most necessary attributes, potentially excluding some attributes. Finally, I arrive at the conclusion from applying these models to find which model is best suited to predict Purchase trend of customers.



**Fig-1: Train.csv**



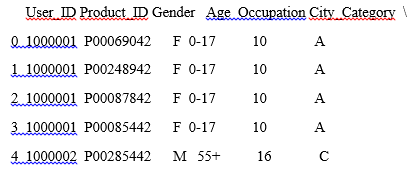
**Fig-2:Test.csv (No Purchase Values)**

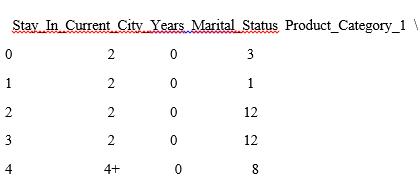
**4. Proposed System Analysis**

**Data Analysing**

In this step I just use common descriptive statistics techniques & apply them on our existing data like mean, median, standard deviation, frequency etc. Also, I’ll find skew, kurtosis followed by correlation matrix with respect to Purchase values.

The output obtained from the given codes in the appendix is-





Product\_Category\_2 Product\_Category\_3 Purchase

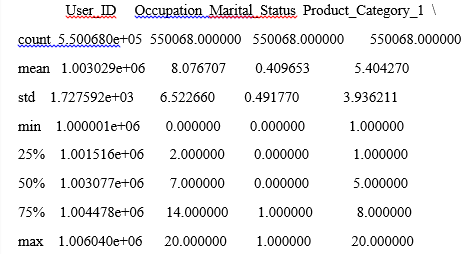
0 NaN NaN 8370

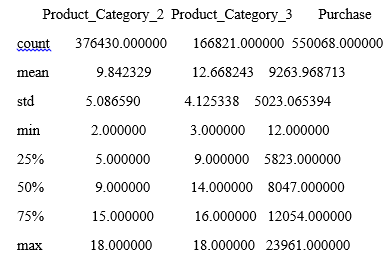
1 6.0 14.0 15200

2 NaN NaN 1422

3 14.0 NaN 1057

4 NaN NaN 7969





There are 544177 duplicate IDs for 550068 total entries

Skew is: 0.6001400037087128

Kurtosis: -0.338378

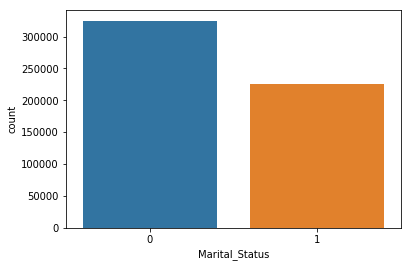
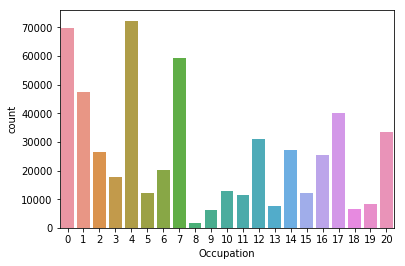


Fig-3: Occupation vs count Fig-4: Marital status vs count

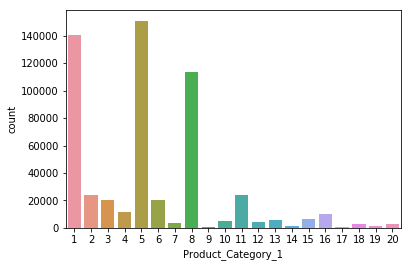
 

Fig-5: PC1 vs count Fig-6: PC2 vs count

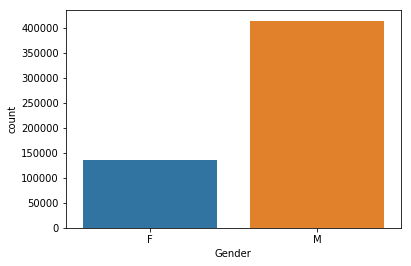
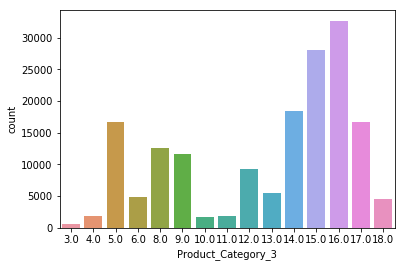
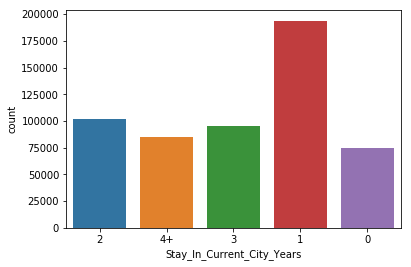
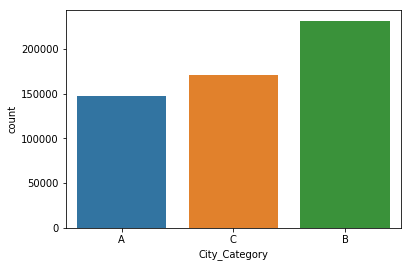
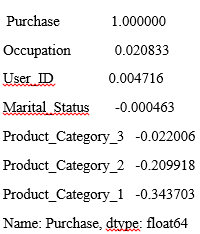


Fig-7: PC3 vs count Fig-8: gender vs count

Correlation from Purchase



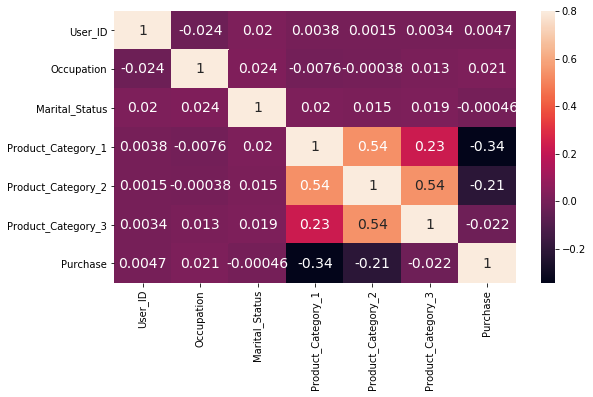


Fig-9: Correlation Matrix

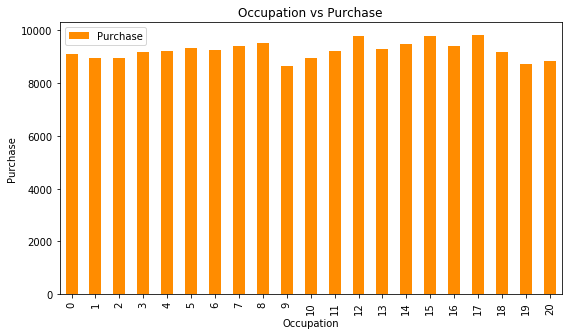
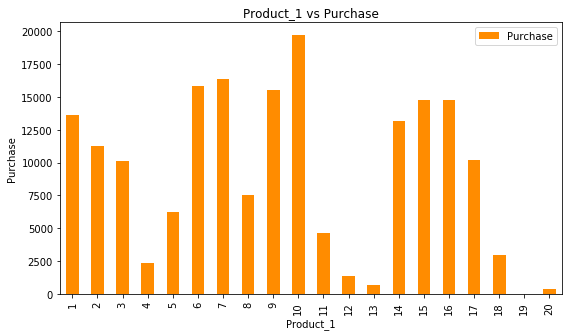
 

Fig-10: occupation vs purchase Fig-11: PC1 vs purchase

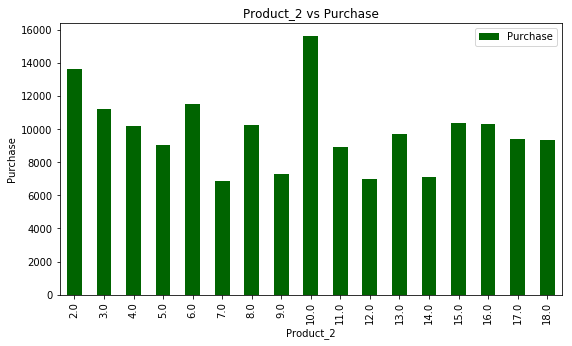
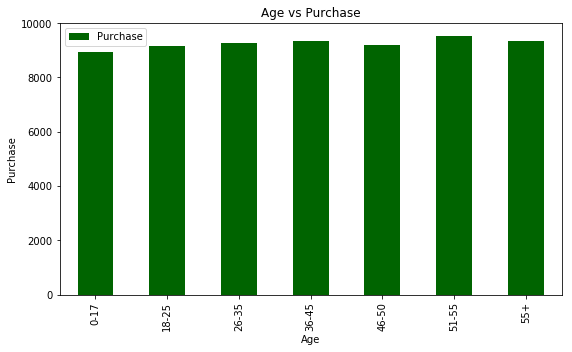
 

Fig-12: PC2 vs purchase Fig-13: Age vs purchase

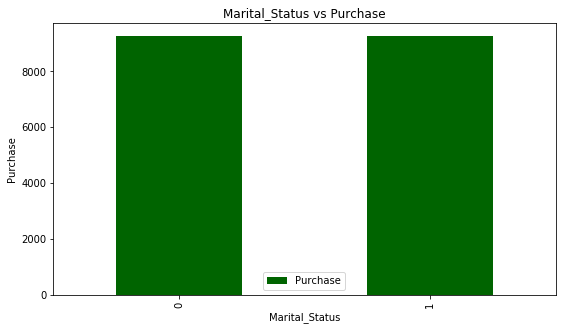
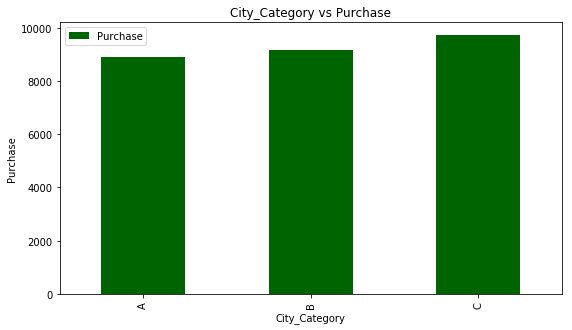
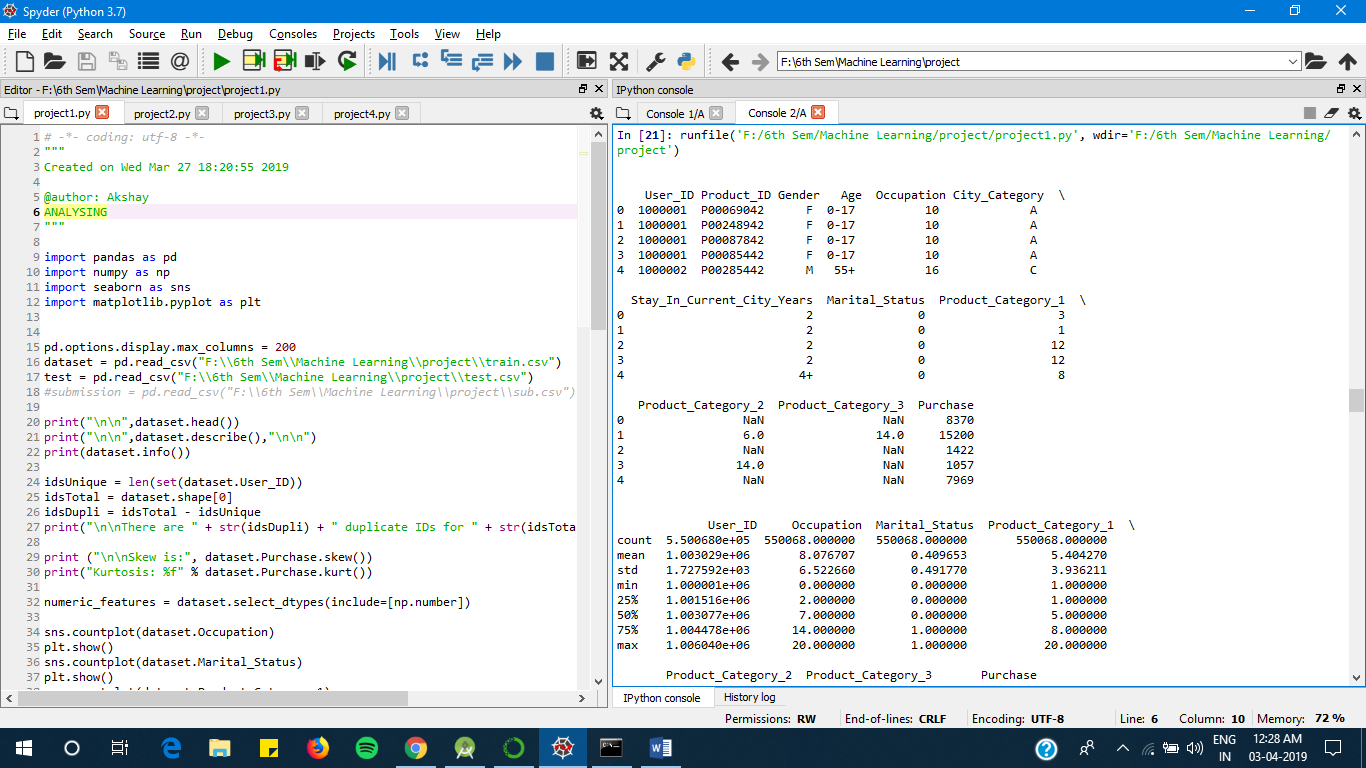
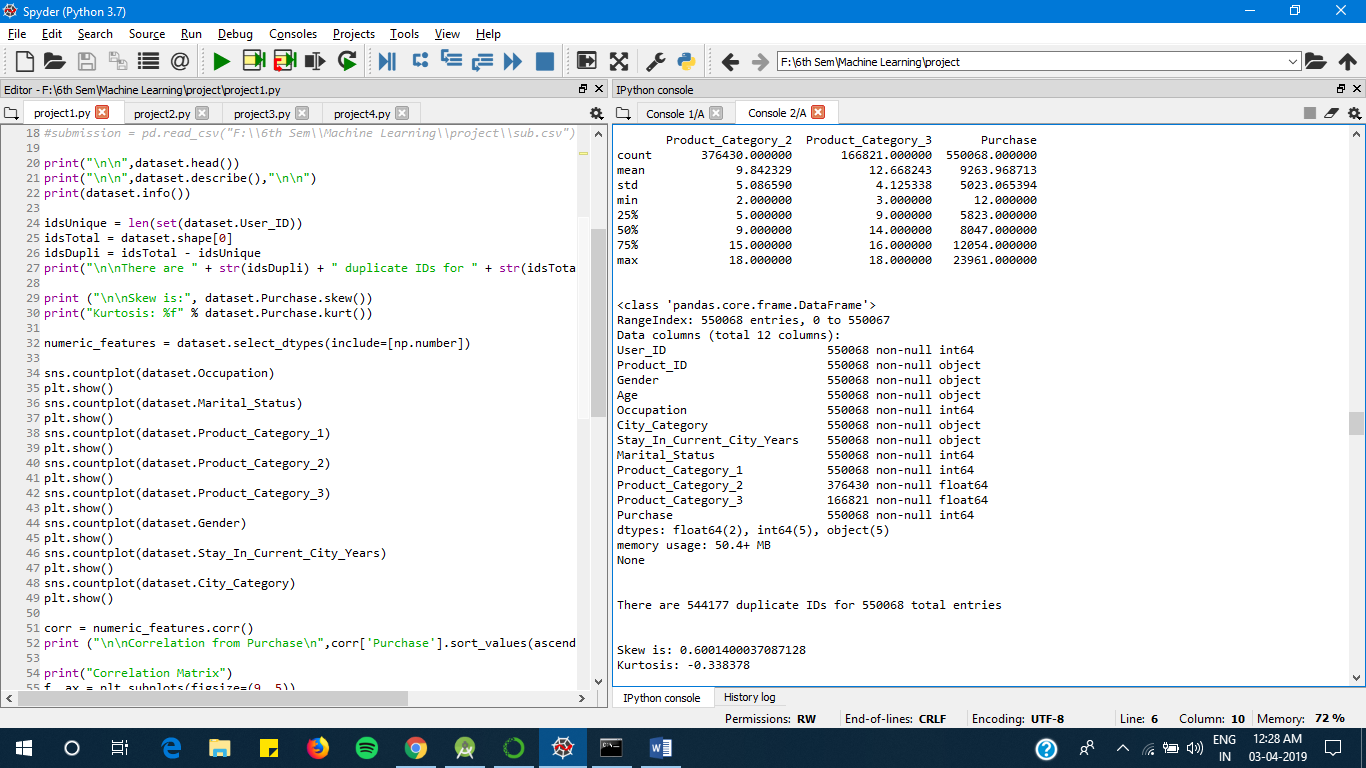
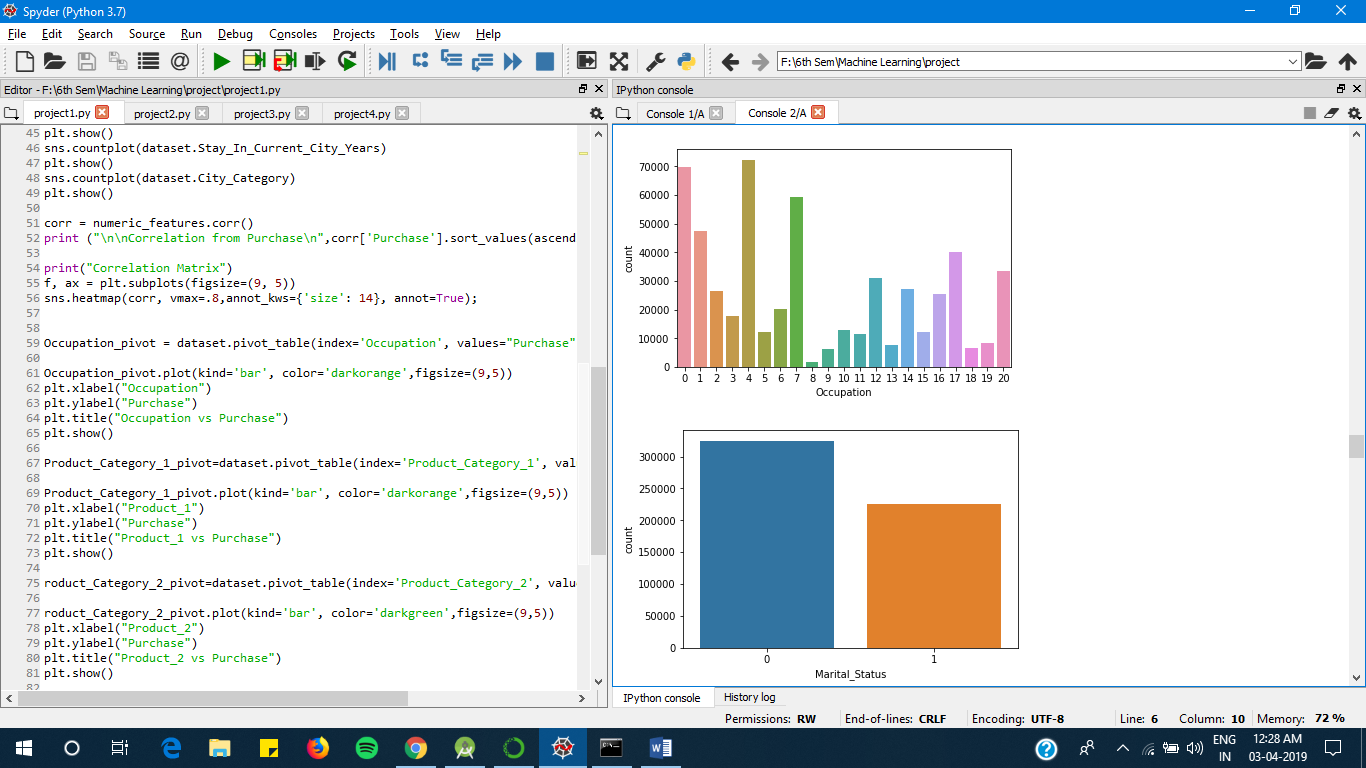
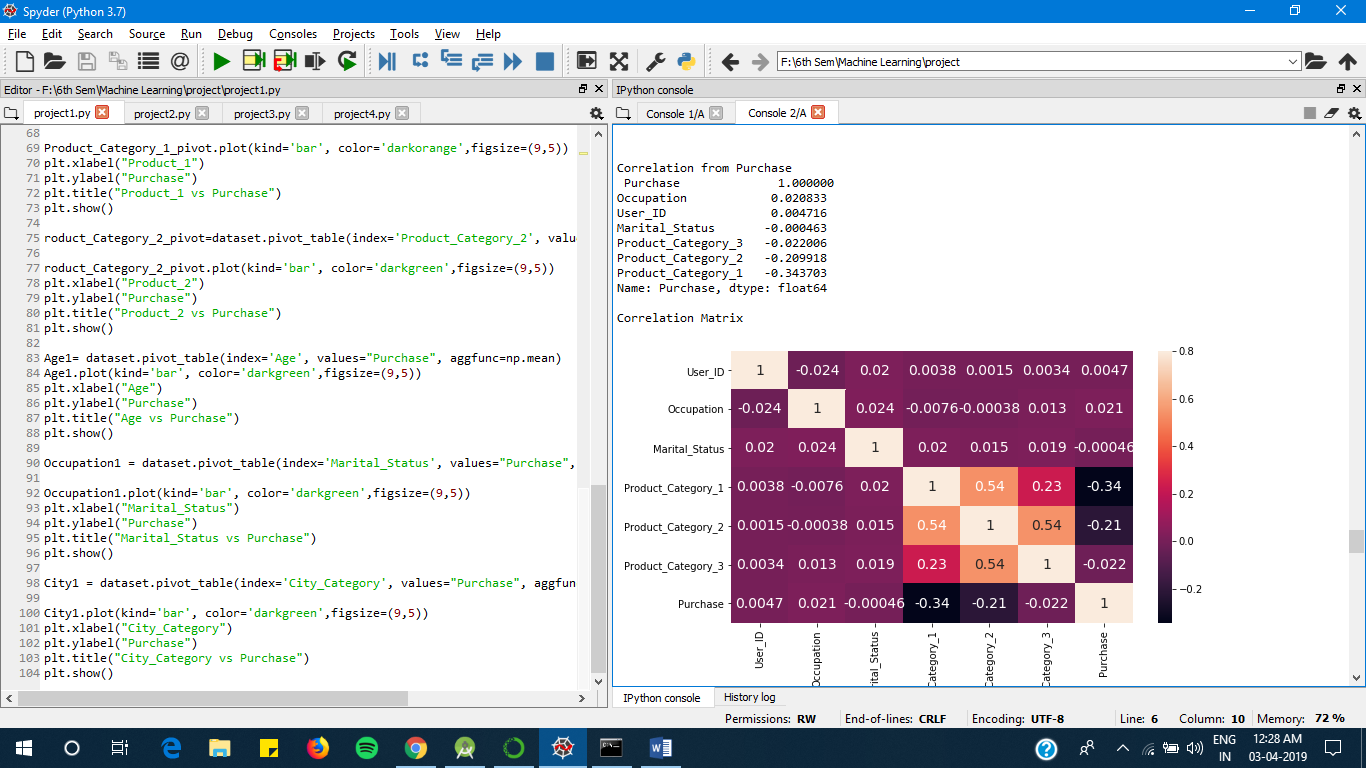
 

Fig-14: Marital status vs purchase Fig-15: city vs purchase









**PRE- PROCESSING**

Data pre-processing is an essential step in the process of machine learning. It includes data cleaning & data partitioning

This stage will involve removing all the NA (null) values & replacing them with some integer so that processing can be carried out. Later I can also display the unique value frequencies of all the columns & finally send this data into a modified train & test .csv files.

Because of our dataset being majority numerical in nature, I use the partitioning technique to remove the presence of unique non-numerical values & convert them to numerical.

The output obtained on pre-processing is-

This is the frequency distribution for Gender:

M 590031

F 193636

Name: Gender, dtype: int64

This is the frequency distribution for Age:

26-35 313015

36-45 156724

18-25 141953

46-50 65278

51-55 54784

55+ 30579

0-17 21334

Name: Age, dtype: int64

This is the frequency distribution for City\_Category:

B 329739

C 243684

A 210244

Name: City\_Category, dtype: int64

This is the frequency distribution for Stay\_In\_Current\_City\_Years:

1 276425

2 145427

3 135428

4+ 120671

0 105716

Name: Stay\_In\_Current\_City\_Years, dtype: int64

This is the frequency distribution for source:

train 550068

test 233599

Name: source, dtype: int64

Index(['F', 'M'], dtype='object')

[0 1]

1 590031

0 193636

Name: Gender, dtype: int64

Index(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'], dtype='object')

[0 1 2 3 4 5 6]

2 313015

5 156724

6 141953

3 65278

4 54784

1 30579

0 21334

Name: Age, dtype: int64

Index(['2', '4+', '3', '1', '0'], dtype='object')

[0 1 2 3 4]

3 276425

0 145427

2 135428

1 120671

4 105716

Name: Stay\_In\_Current\_City\_Years, dtype: int64

Index(['A', 'C', 'B'], dtype='object')

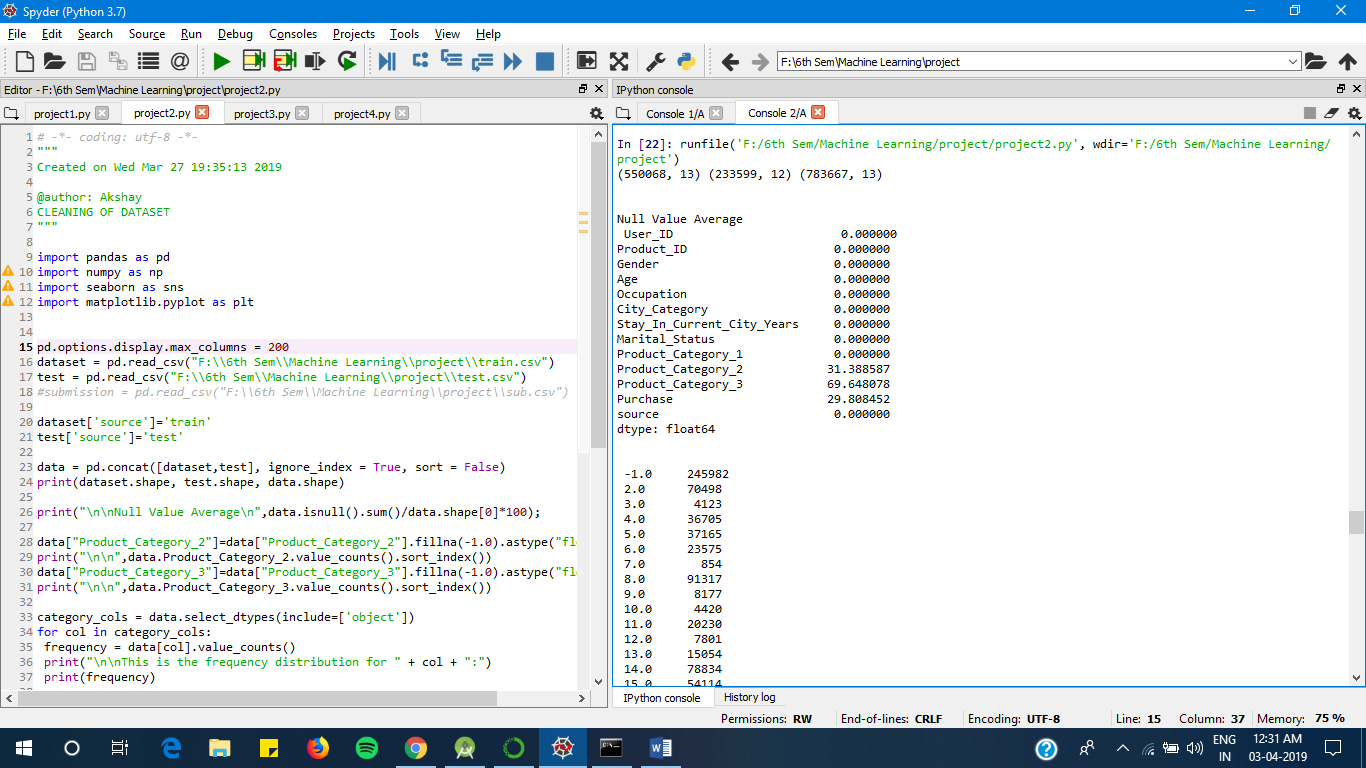
[0 1 2]

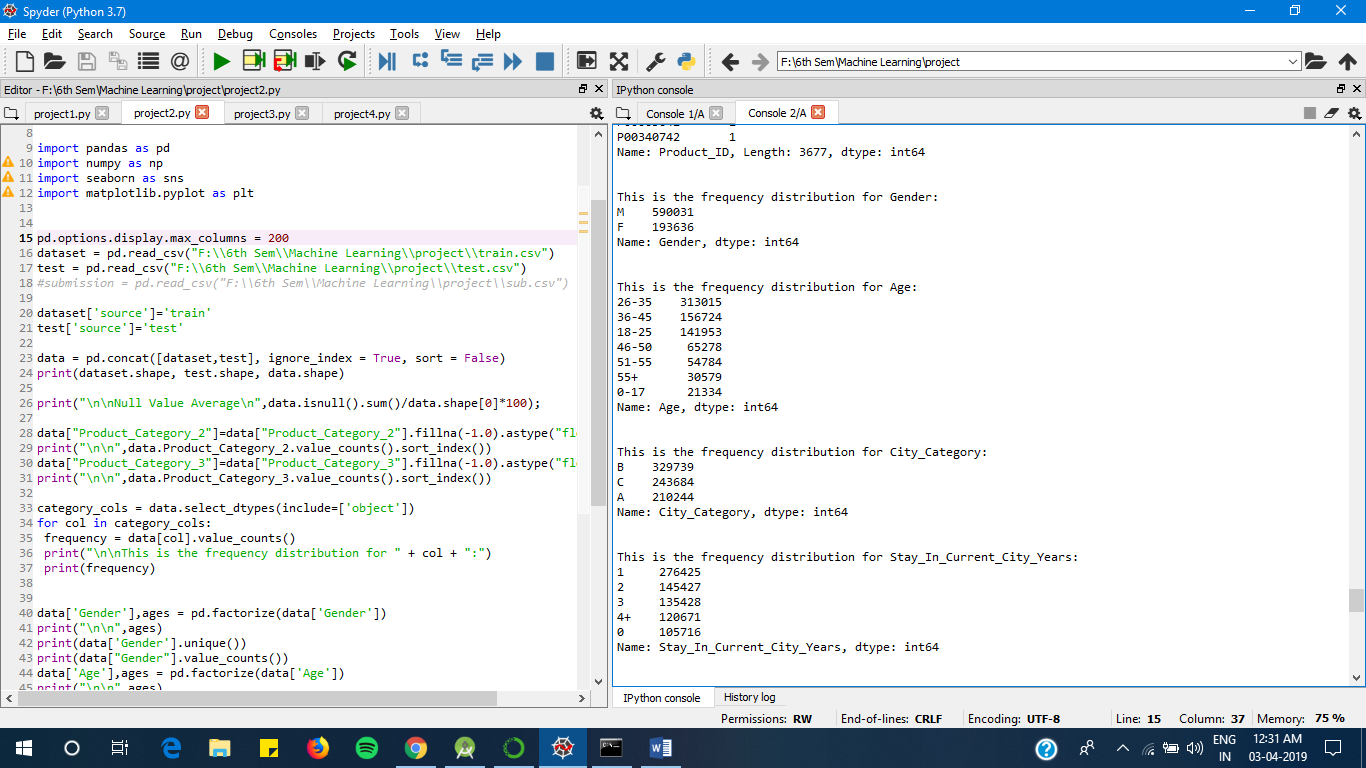
2 329739

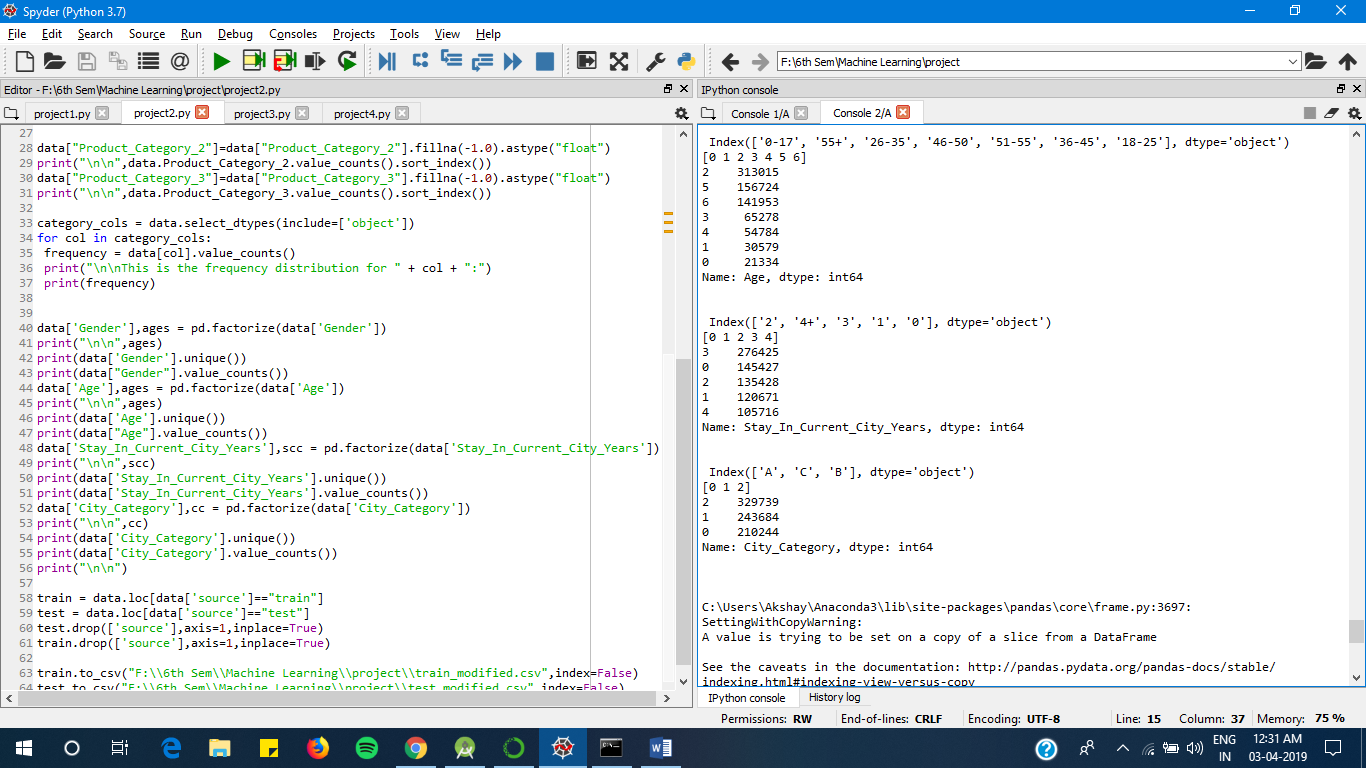
1 243684

0 210244

Name: City\_Category, dtype: int64







**Modified Datasets-**

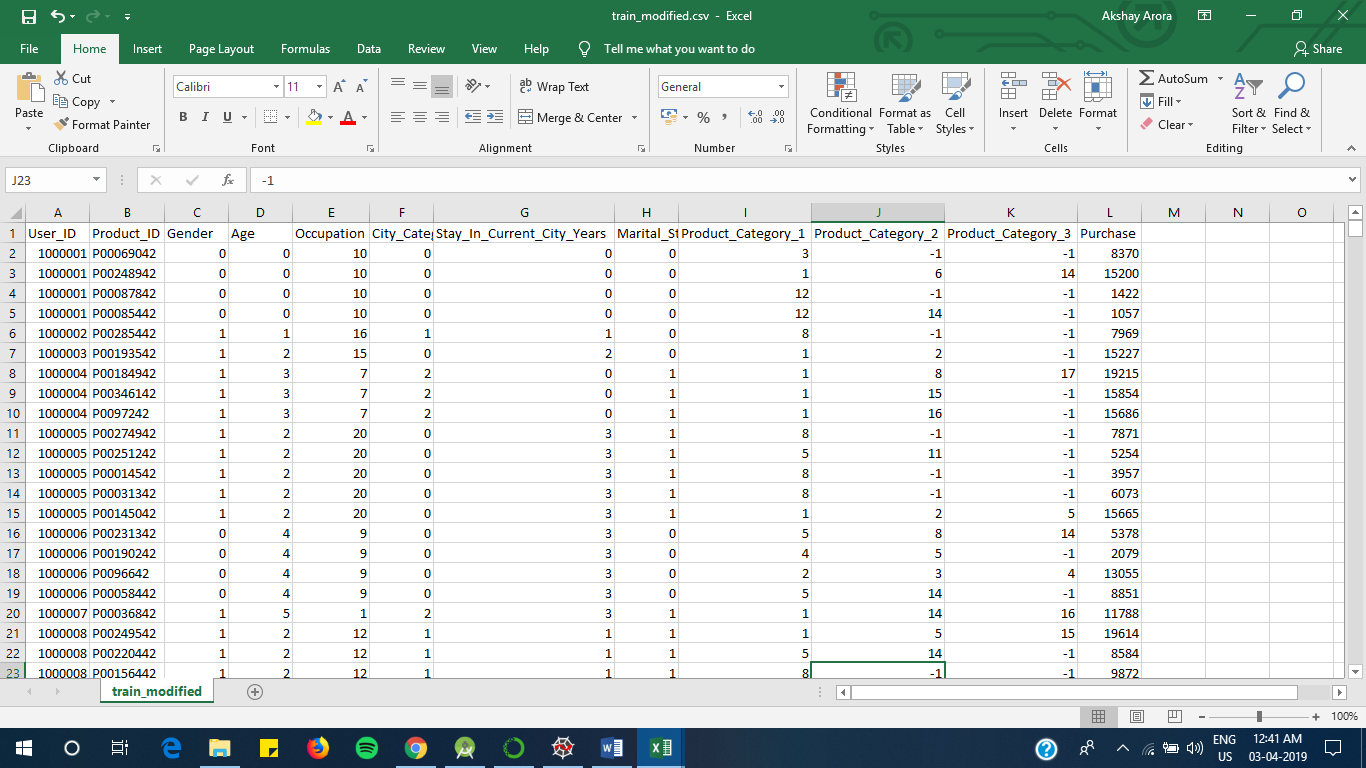


Fig-16: Train\_modified.csv

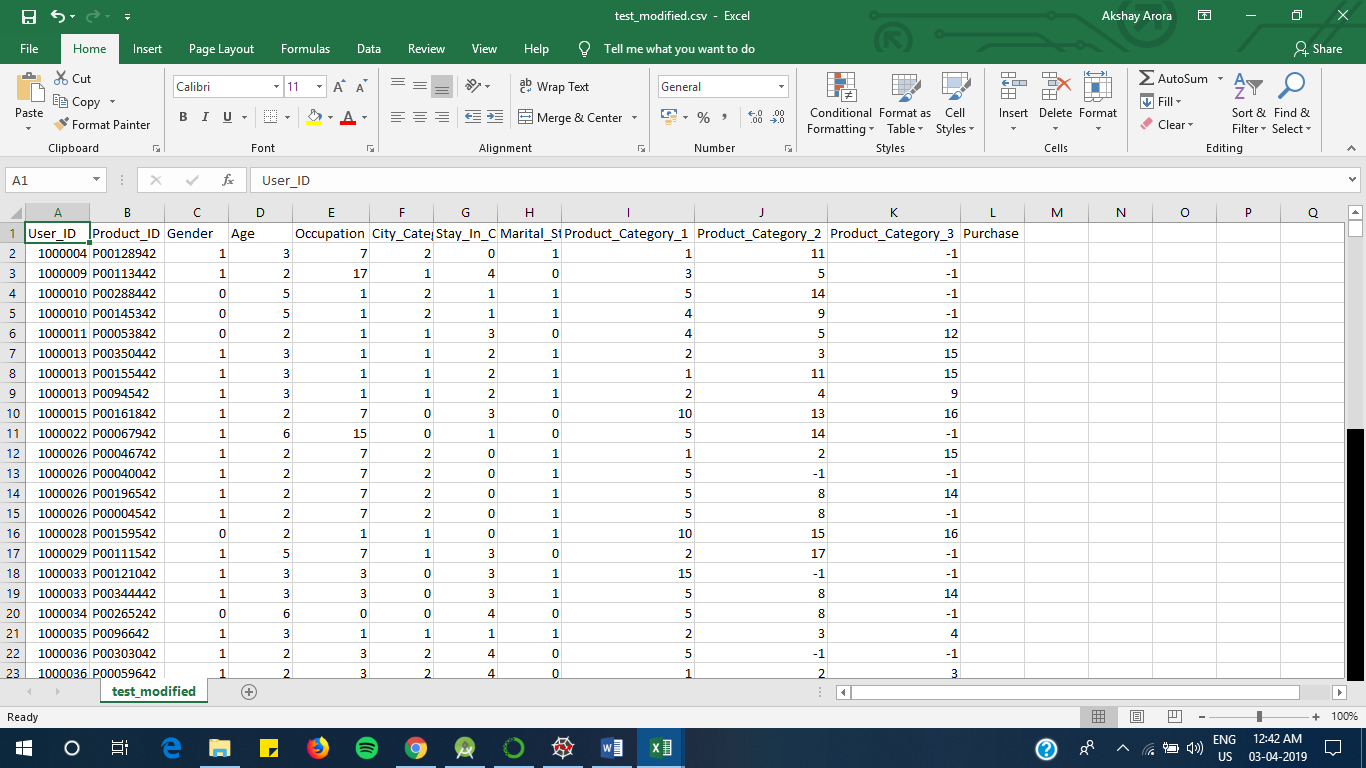


Fig-17:Test\_Modified.csv

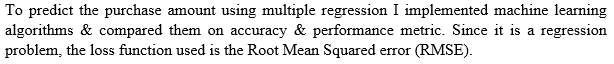
**5. Implementation**

Before selecting the models to use for the training process, I need to decide the columns/features that can be used as predictors & drop the others.

This decision needs to be made on the basis on the data analysis done at the first stage of this process.

For eg: when I made the correlation matrix of the dataset, I found that the column “purchase” was highly correlating with the column “Occupation”. This implies the column occupation should be included in the predictors.

The next step, is to use the modified train & test data & apply machine learning algorithms of various types as read about in the survey.

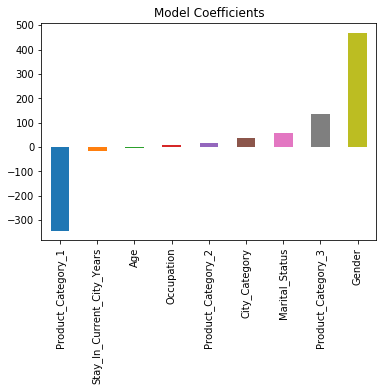
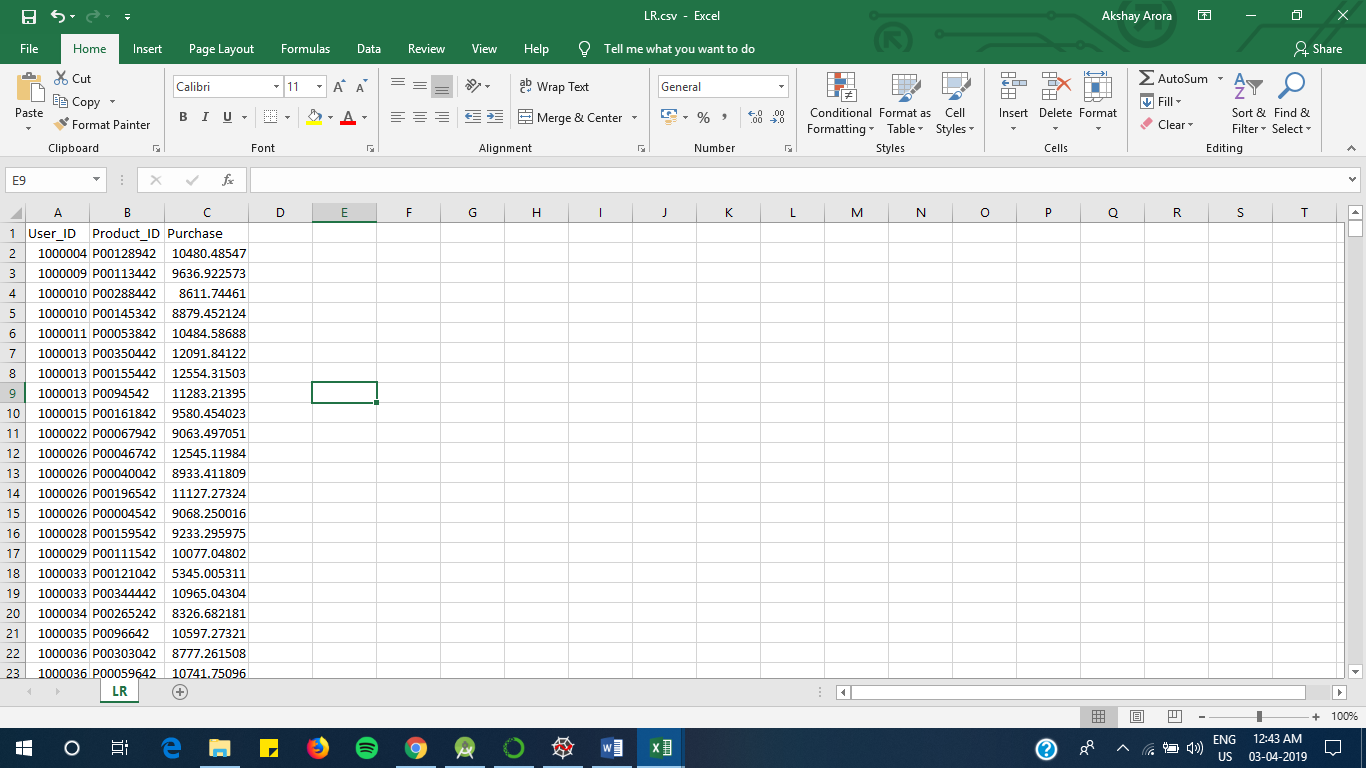
1. **Linear Regression**

The linear regression using python's skLearn library was implemented on the transformed dataset. This was the simplest of the implementations in terms of complexity of the model.

Model Report

RMSE : 4632

CV Score : Mean - 4635 | Std - 35.02 | Min - 4545 | Max – 4688

  Fig-18: Model coefficients Fig-19: Predicted values in LR.csv

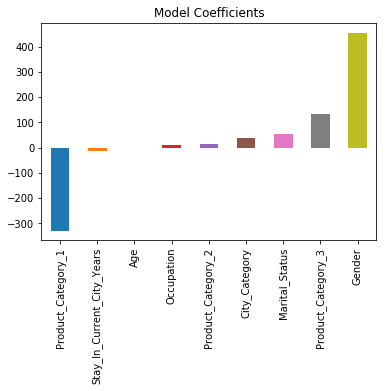
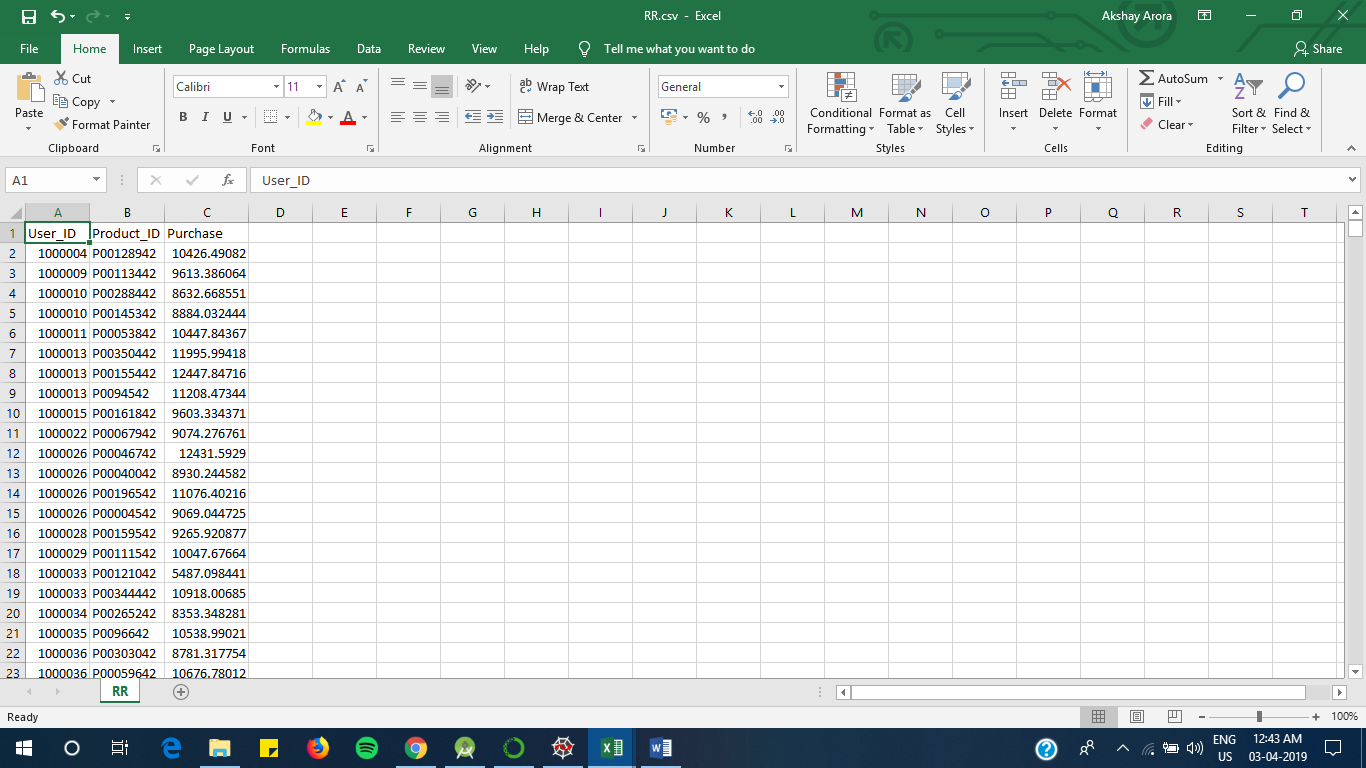
1. **Ridge regression**

The ridge regression using python's skLearn library was implemented on the transformed dataset.

Model Report

RMSE : 4633

CV Score : Mean - 4636 | Std - 31.86 | Min - 4570 | Max - 4687

  Fig-20: Model coefficients Fig-21: Predicted values in RR.csv

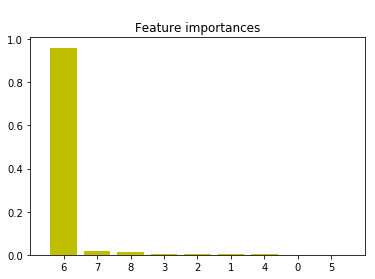
1. **Decision Tree Regression**

Machine learning algorithms like decision tree & regression are used for developing a simple yet efficient prediction models. Guo et al. state that a time series analysis using early purchase patters can be used to predict the future spending. The technique involved can be classified into two groups, mathematical & statistical model, & artificial intelligence model [4]. The Decision Tree technique comes under the artificial intelligence model, which develops a tree with root node containing the most important feature & subsequent nodes in the tree with less ranking features.

Model Report

RMSE : 2916

CV Score : Mean - 2947 | Std - 19.9 | Min - 2907 | Max - 2977

Feature ranking:

x1. feature 6 (0.960362)

x2. feature 7 (0.015366)

x3. feature 8 (0.010731)

x4. feature 3 (0.004482)

x5. feature 2 (0.003268)

x6. feature 1 (0.002386)

x7. feature 4 (0.001721)

x8. feature 0 (0.000868)

x9. feature 5 (0.000815)

Fig-22: Feature importance

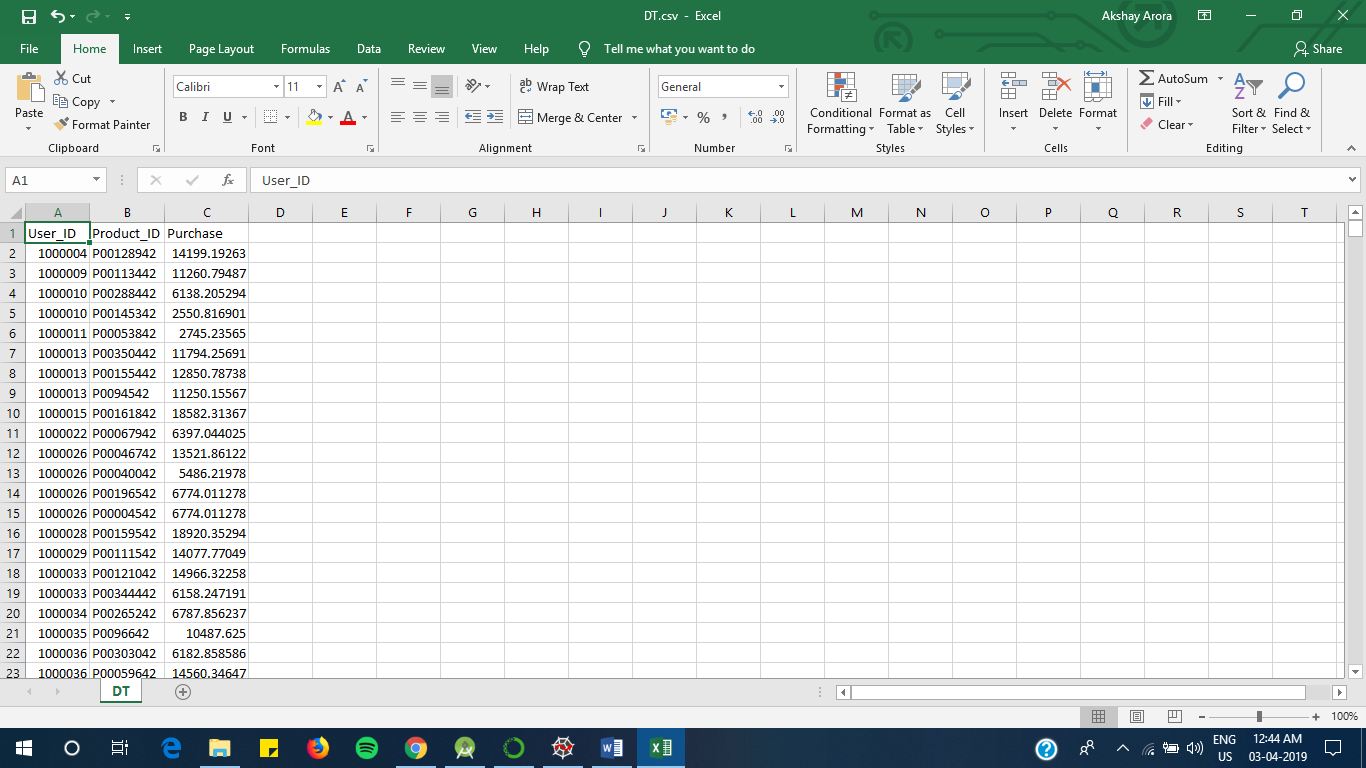
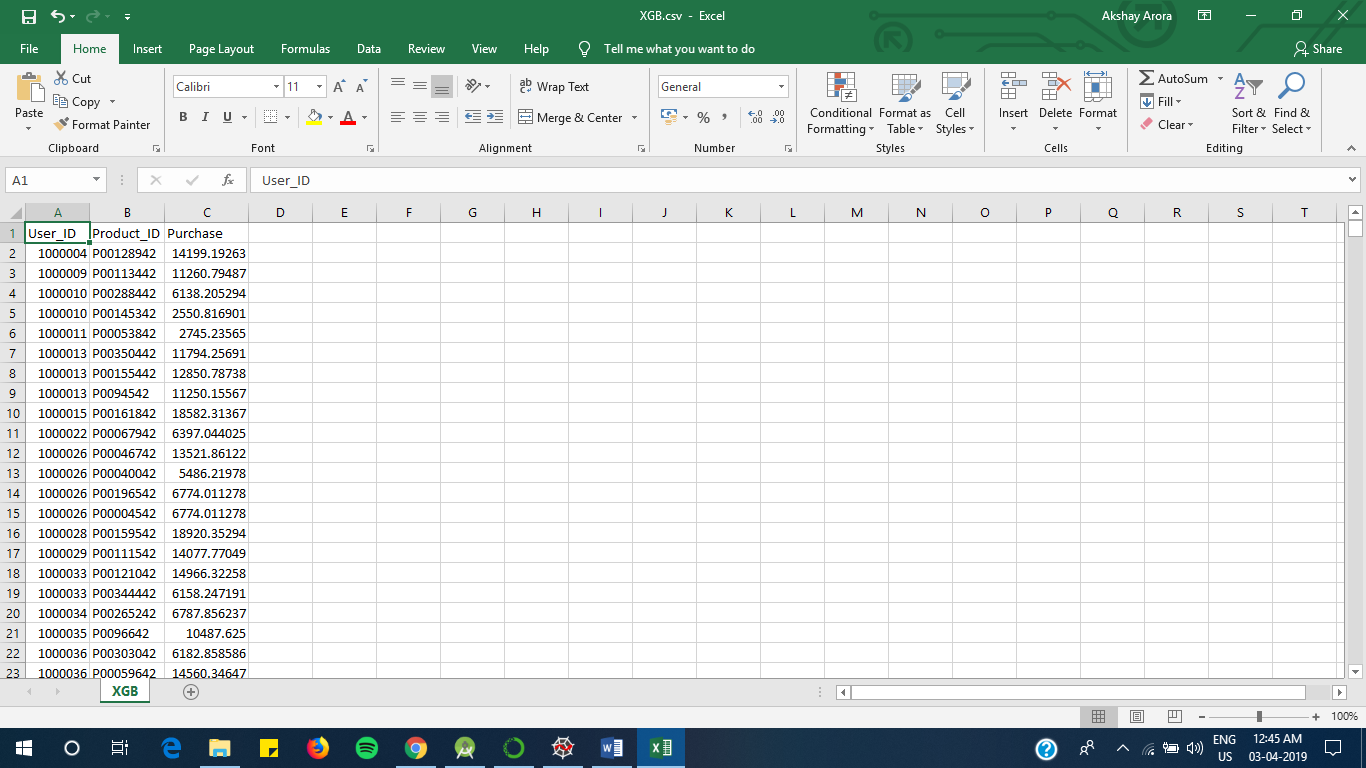


Fig-23: Predicted values in DT.csv

1. **XGB Regression**

The XGBoost model internally implements the stepwise, ridge regression which dynamically selects the features & removes the multi-collinearity with the features. This implementation gave the bet results of this dataset. It uses ensemble model to learn from the weak predictors & eliminate the less important features to develop a strong model.

Mean Absolute Error : 392.22502938349544

RMSE : 2950

Feature order:

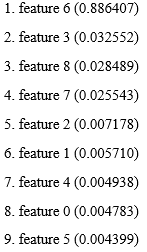


Fig-24: Predicted values in XGB.csv

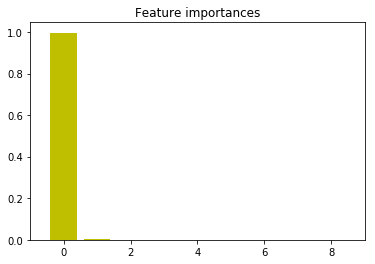
1. **Random forest regression**

Model Report

RMSE : 3754

CV Score : Mean - 3714 | Std - 22.85 | Min - 3672 | Max - 3750

Mean Absolute Error : 3.7333049827565437

RMSE : 3754

Feature order:

1. feature 6 (0.996204)

2. feature 8 (0.001988)

3. feature 7 (0.001205)

4. feature 3 (0.000603)

5. feature 5 (0.000000)

6. feature 4 (0.000000)

7. feature 2 (0.000000)

8. feature 1 (0.000000)

9. feature 0 (0.000000)

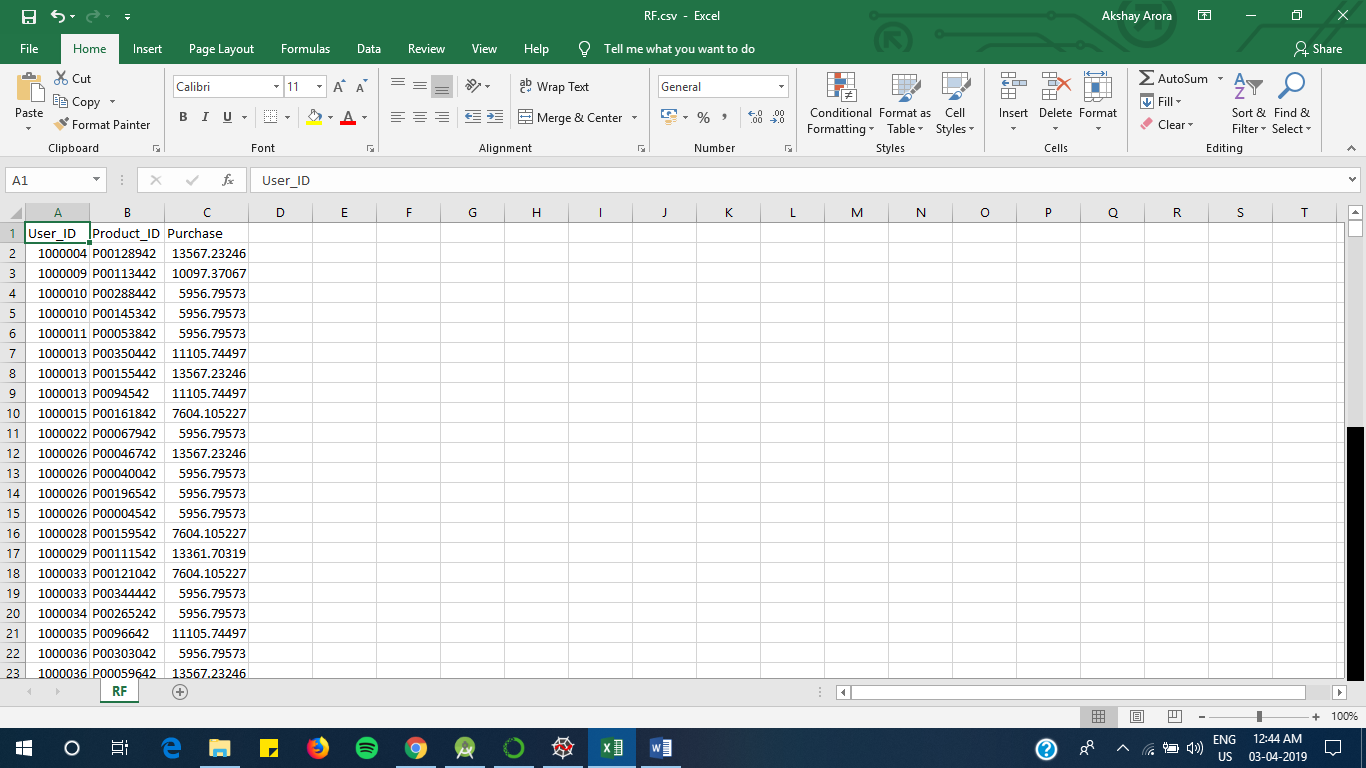
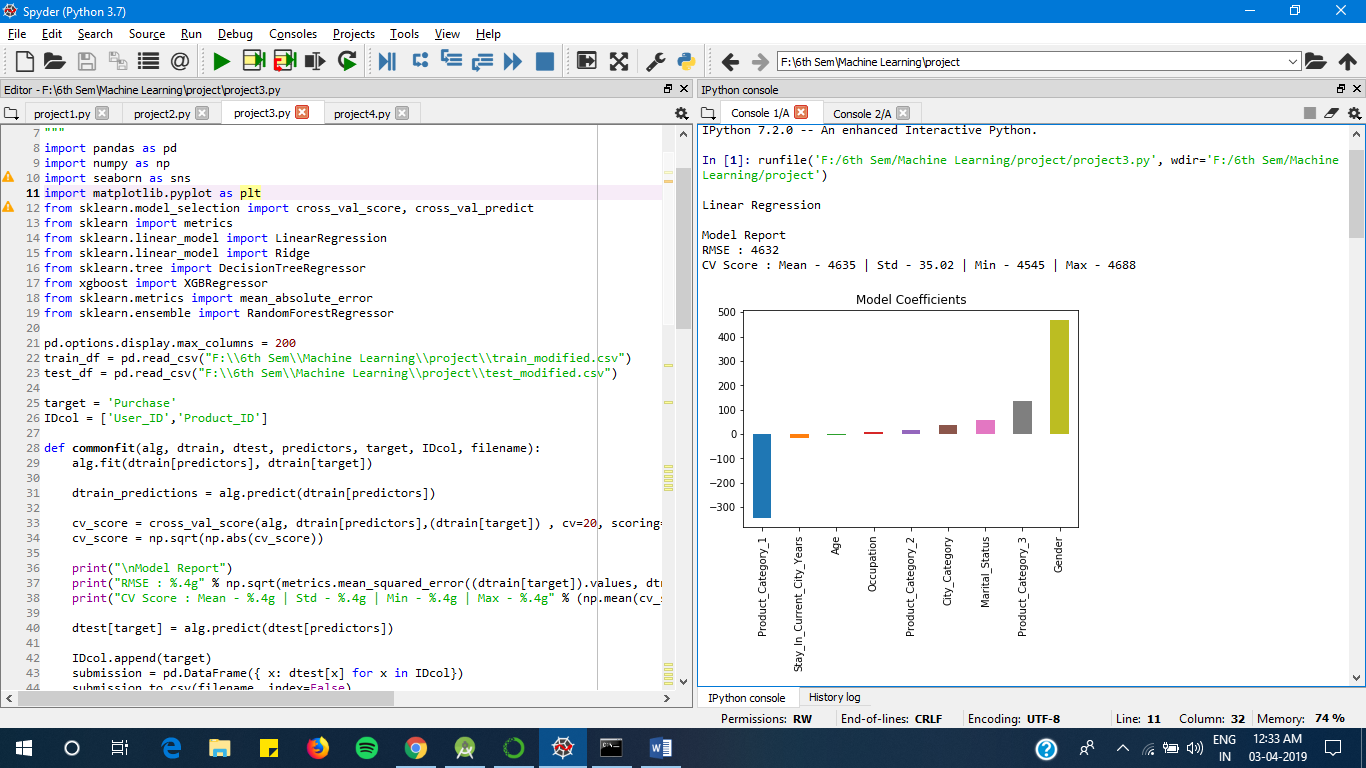
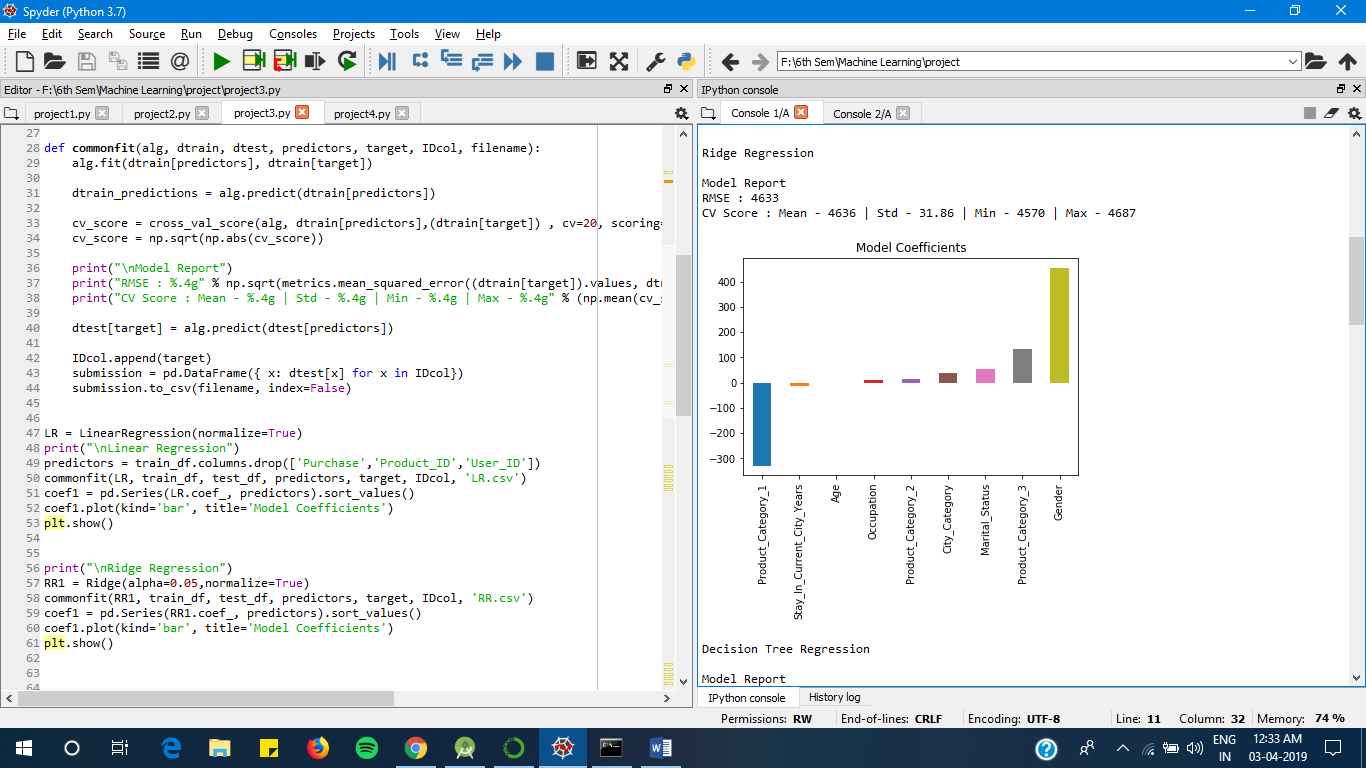
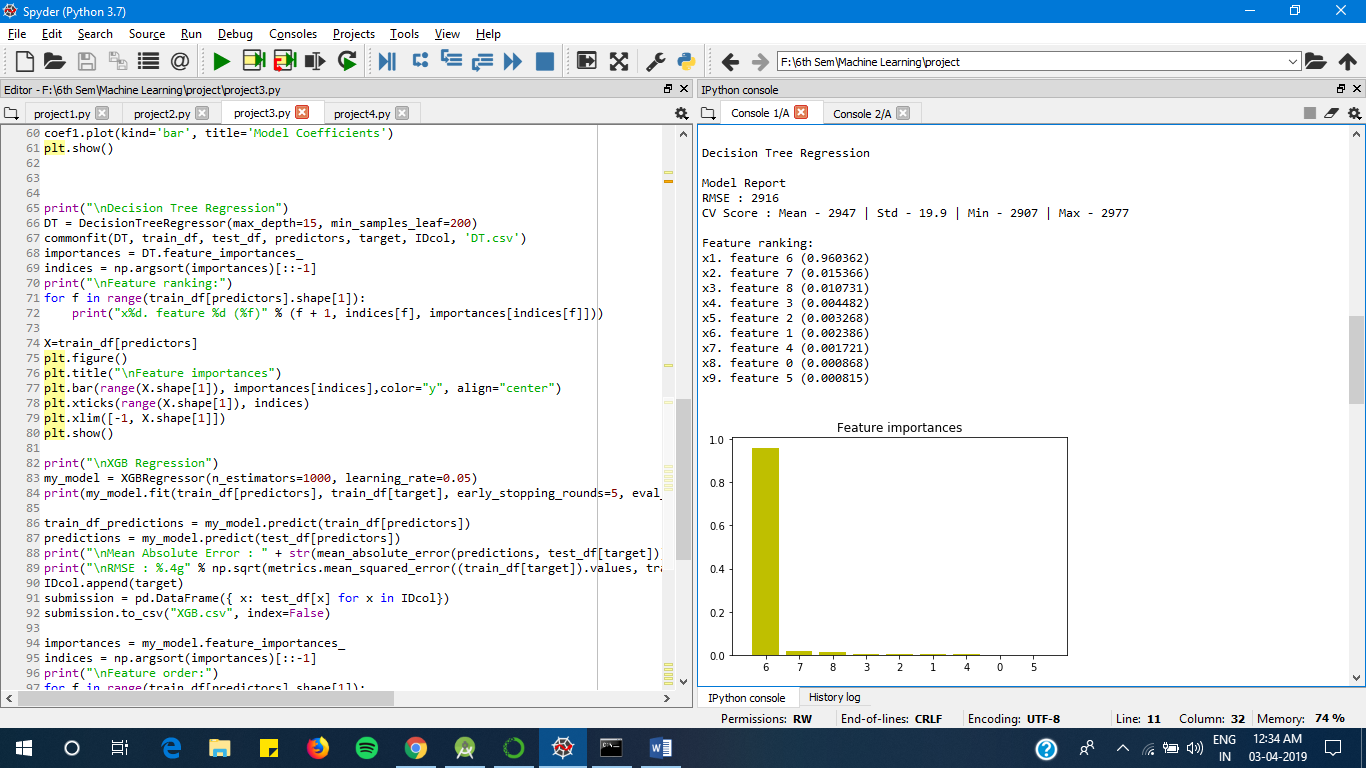


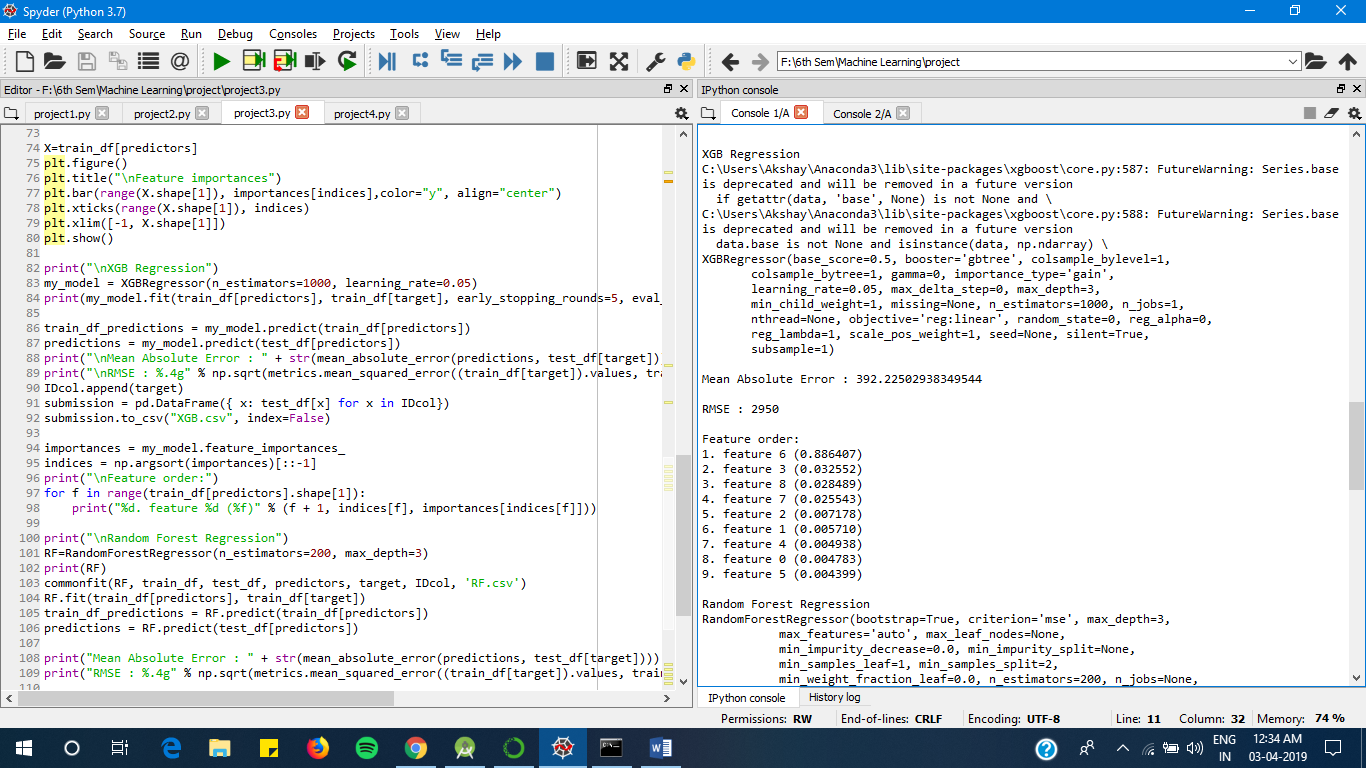
Fig-25: Predicted values in RF.csv

**OUTPUTS-**



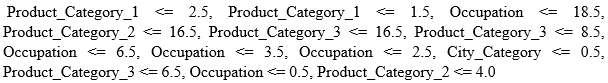


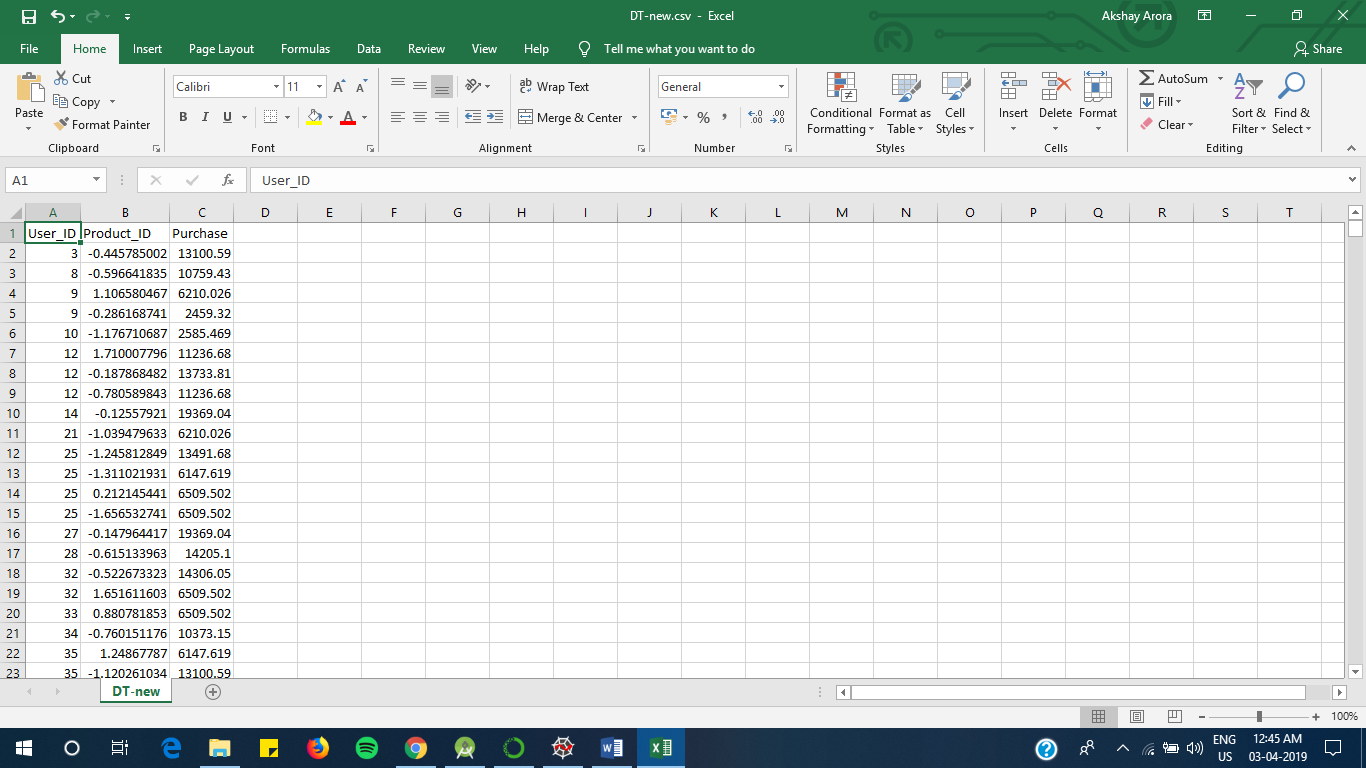






1. **Rule Based Learning**



Before Rule Based:

Model Report

RMSE : 2996

CV Score : Mean - 3242 | Std - 54.63 | Min - 3031 | Max - 3289

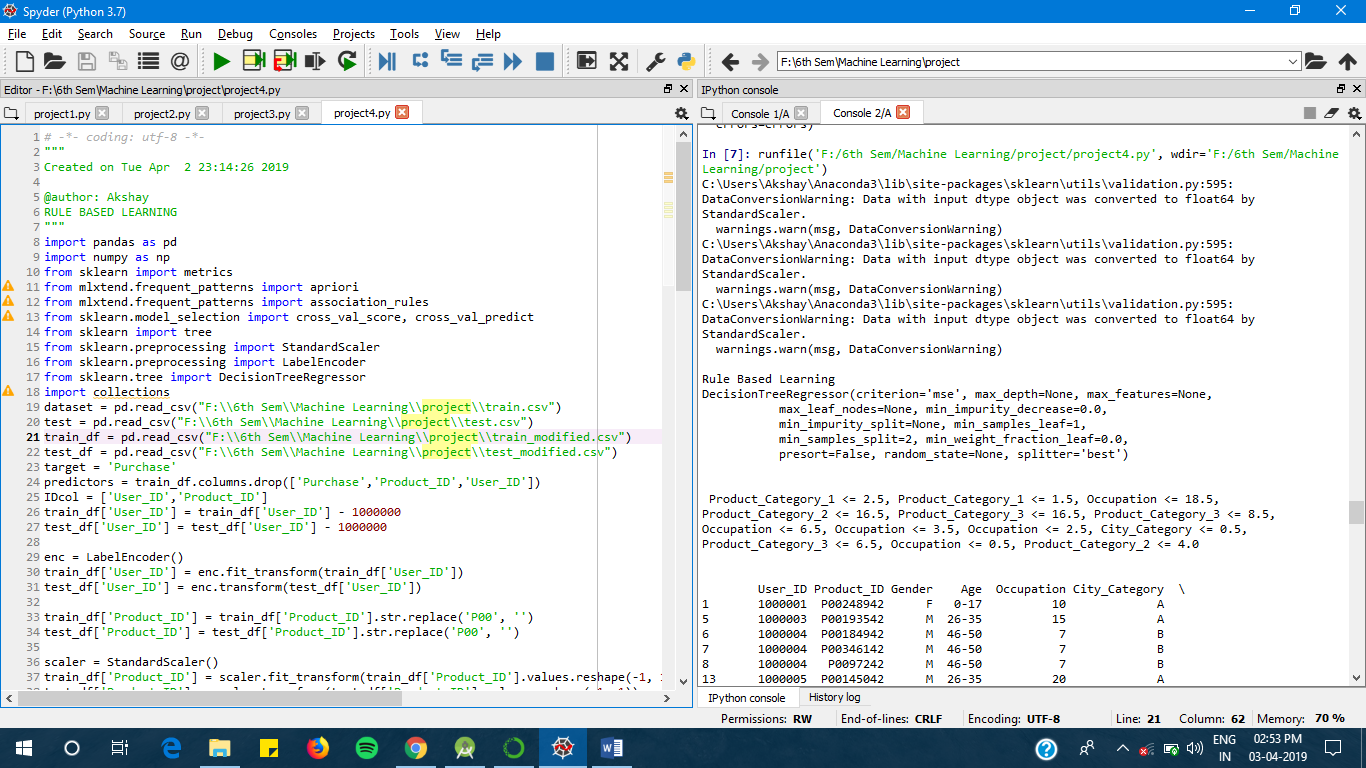
After Rule Based:

Model Report

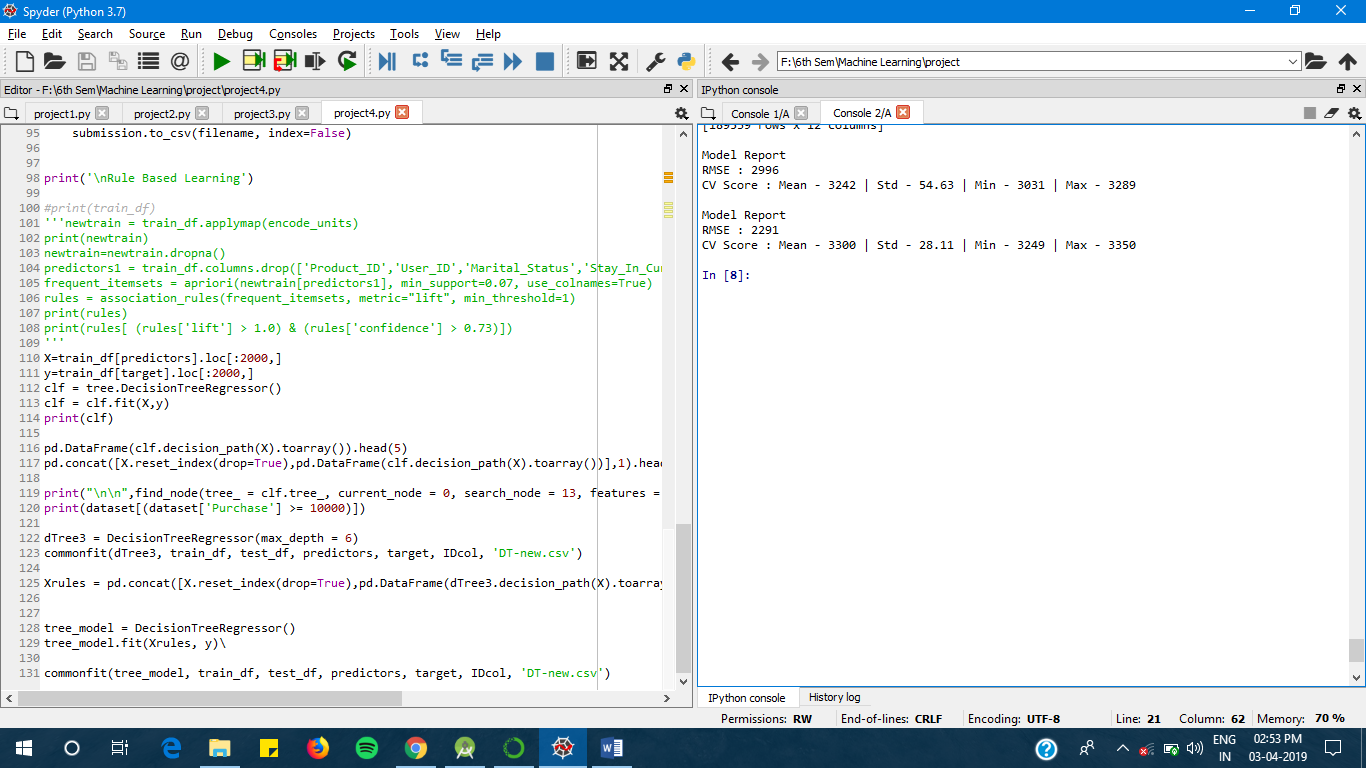
RMSE : 2291

CV Score : Mean - 3300 | Std - 28.11 | Min - 3249 | Max – 3350

Fig-26: Predicted values in DT-new.csv







**RESULT**

 Lower the value of RMSE better the prediction by the algorithm.

According to the obtained RMSE value Rule based decision tree is the most optimized algorithm to analyse the black Friday sales

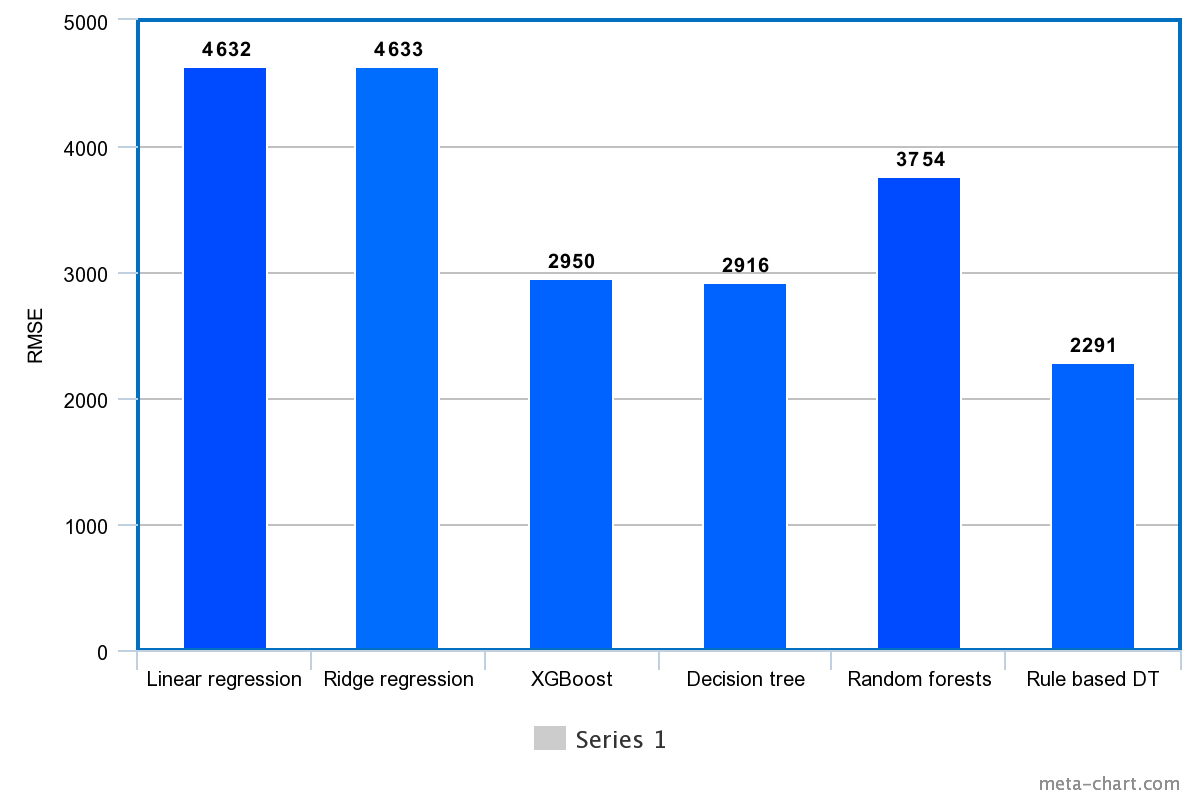
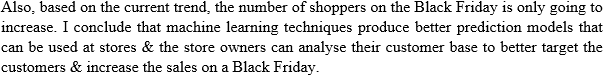
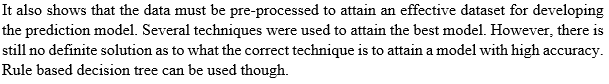


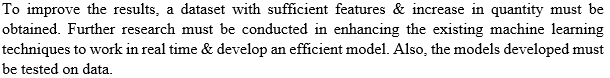
Fig-27: RMSE values of different classifiers & regressors

**CONCLUSION**









**REFERENCES**

[1] M. Petrescu & M. Murphy, "Black Friday & Cyber Monday: a case study" in International Journal of Electronic Marketing & Retailing (IJEMR), voL 5, no.3, 2013.

[2] L. P. Barroso, W. O. Bussab, & M. Knott, "Best linear unbiased predictor in the mixed model with incomplete data," Communications in Statistics Theory & Methods, vol. 27, no. 1, pp . 121-129, 1998. doi: 10.108010361092980883265 4J.

[3] I. Guyon & A. Elisseeff, "An introduction to variable & feature selection," J. Mach. Learn. Res., vol. 3, pp. 1157-1182, Mar. 2003.

[4] Z. X. Guo, W. K. Wong , & M. Li, "A multivariate intelligent decision-making model for retail sales forecasting," Decision Support Syst., voL 55, pp -247-255, Apr. 2013 .

[5] A. Soroush , A. Bahreininejad, & 1. van den Berg , "A hybrid customer prediction system based on multiple forward stepwise logistic regression mode ," Intell . Data Anal. , vol. 16, pp. 265-278, Mar. 2012.

[6] L. Bing & S. Yuliang, "Prediction ofuser 's purchase intention based on machine learning," 3rd International Conference on Soft Computing Machine Intelligence (ISCMI)., pp.99·103, Nov. 2016 .

[7] Y. Qin & H. Li, "Sales forecast based on BP neural network", 2011 IEEE 3rd International Conference on Communication Software & Network., pp. 186-189, May 2011

[8] K. Singh & R. Wajgi, "Data analysis & visualization of sales data," 2016 World Conference on Futuristic Trends in Research & Innovation for Social Welfare (Startup Conclave), Coimbatore, pp. 1-6, Mar. 2016 .

[9] https j/datahack.analyticsvidhya.com/contest!black·friday/#data\_dictionary

[10] https .zIwww .analyticsvidhya.comlblog/2015108/comprehensive-guide-re gression!

[11] https j Imachinelearningmastery.comlregression-tutorial-keras-deep-Ieam i ng-library-python!

[12] http ://scikit-Ieam.org/stable/auto\_examplesilinear\_modellplot\_ols.html

**APPENDIX (SAMPLE CODE)**

**ANALYSING**

# -\*- coding: utf-8 -\*-

"""

Created on Wed Mar 27 18:20:55 2019

ANALYSING

"""

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

pd.options.display.max\_columns = 200

dataset = pd.read\_csv("F:\\6th Sem\\Machine Learning\\project\\train.csv")

test = pd.read\_csv("F:\\6th Sem\\Machine Learning\\project\\test.csv")

#submission = pd.read\_csv("F:\\6th Sem\\Machine Learning\\project\\sub.csv")

print("\n\n",dataset.head())

print("\n\n",dataset.describe(),"\n\n")

print(dataset.info())

idsUnique = len(set(dataset.User\_ID))

idsTotal = dataset.shape[0]

idsDupli = idsTotal - idsUnique

print("\n\nThere are " + str(idsDupli) + " duplicate IDs for " + str(idsTotal) + " total entries")

print ("\n\nSkew is:", dataset.Purchase.skew())

print("Kurtosis: %f" % dataset.Purchase.kurt())

numeric\_features = dataset.select\_dtypes(include=[np.number])

sns.countplot(dataset.Occupation)

plt.show()

sns.countplot(dataset.Marital\_Status)

plt.show()

sns.countplot(dataset.Product\_Category\_1)

plt.show()

sns.countplot(dataset.Product\_Category\_2)

plt.show()

sns.countplot(dataset.Product\_Category\_3)

plt.show()

sns.countplot(dataset.Gender)

plt.show()

sns.countplot(dataset.Stay\_In\_Current\_City\_Years)

plt.show()

sns.countplot(dataset.City\_Category)

plt.show()

corr = numeric\_features.corr()

print ("\n\nCorrelation from Purchase\n",corr['Purchase'].sort\_values(ascending=False),"\n")

print("Correlation Matrix")

f, ax = plt.subplots(figsize=(9, 5))

sns.heatmap(corr, vmax=.8,annot\_kws={'size': 14}, annot=True);

Occupation\_pivot = dataset.pivot\_table(index='Occupation', values="Purchase", aggfunc=np.mean)

Occupation\_pivot.plot(kind='bar', color='darkorange',figsize=(9,5))

plt.xlabel("Occupation")

plt.ylabel("Purchase")

plt.title("Occupation vs Purchase")

plt.show()

Product\_Category\_1\_pivot=dataset.pivot\_table(index='Product\_Category\_1', values="Purchase", aggfunc=np.mean)

Product\_Category\_1\_pivot.plot(kind='bar', color='darkorange',figsize=(9,5))

plt.xlabel("Product\_1")

plt.ylabel("Purchase")

plt.title("Product\_1 vs Purchase")

plt.show()

roduct\_Category\_2\_pivot=dataset.pivot\_table(index='Product\_Category\_2', values="Purchase")

roduct\_Category\_2\_pivot.plot(kind='bar', color='darkgreen',figsize=(9,5))

plt.xlabel("Product\_2")

plt.ylabel("Purchase")

plt.title("Product\_2 vs Purchase")

plt.show()

Age1= dataset.pivot\_table(index='Age', values="Purchase", aggfunc=np.mean)

Age1.plot(kind='bar', color='darkgreen',figsize=(9,5))

plt.xlabel("Age")

plt.ylabel("Purchase")

plt.title("Age vs Purchase")

plt.show()

Occupation1 = dataset.pivot\_table(index='Marital\_Status', values="Purchase", aggfunc=np.mean)

Occupation1.plot(kind='bar', color='darkgreen',figsize=(9,5))

plt.xlabel("Marital\_Status")

plt.ylabel("Purchase")

plt.title("Marital\_Status vs Purchase")

plt.show()

City1 = dataset.pivot\_table(index='City\_Category', values="Purchase", aggfunc=np.mean)

City1.plot(kind='bar', color='darkgreen',figsize=(9,5))

plt.xlabel("City\_Category")

plt.ylabel("Purchase")

plt.title("City\_Category vs Purchase")

plt.show()

**PRE-PROCESSING**

# -\*- coding: utf-8 -\*-

"""

Created on Wed Mar 27 19:35:13 2019

CLEANING OF DATASET

"""

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

pd.options.display.max\_columns = 200

dataset = pd.read\_csv("F:\\6th Sem\\Machine Learning\\project\\train.csv")

test = pd.read\_csv("F:\\6th Sem\\Machine Learning\\project\\test.csv")

#submission = pd.read\_csv("F:\\6th Sem\\Machine Learning\\project\\sub.csv")

dataset['source']='train'

test['source']='test'

data = pd.concat([dataset,test], ignore\_index = True, sort = False)

print(dataset.shape, test.shape, data.shape)

print("\n\nNull Value Average\n",data.isnull().sum()/data.shape[0]\*100);

data["Product\_Category\_2"]=data["Product\_Category\_2"].fillna(-1.0).astype("float")

print("\n\n",data.Product\_Category\_2.value\_counts().sort\_index())

data["Product\_Category\_3"]=data["Product\_Category\_3"].fillna(-1.0).astype("float")

print("\n\n",data.Product\_Category\_3.value\_counts().sort\_index())

category\_cols = data.select\_dtypes(include=['object'])

for col in category\_cols:

frequency = data[col].value\_counts()

print("\n\nThis is the frequency distribution for " + col + ":")

print(frequency)

data['Gender'],ages = pd.factorize(data['Gender'])

print("\n\n",ages)

print(data['Gender'].unique())

print(data["Gender"].value\_counts())

data['Age'],ages = pd.factorize(data['Age'])

print("\n\n",ages)

print(data['Age'].unique())

print(data["Age"].value\_counts())

data['Stay\_In\_Current\_City\_Years'],scc = pd.factorize(data['Stay\_In\_Current\_City\_Years'])

print("\n\n",scc)

print(data['Stay\_In\_Current\_City\_Years'].unique())

print(data['Stay\_In\_Current\_City\_Years'].value\_counts())

data['City\_Category'],cc = pd.factorize(data['City\_Category'])

print("\n\n",cc)

print(data['City\_Category'].unique())

print(data['City\_Category'].value\_counts())

print("\n\n")

train = data.loc[data['source']=="train"]

test = data.loc[data['source']=="test"]

test.drop(['source'],axis=1,inplace=True)

train.drop(['source'],axis=1,inplace=True)

train.to\_csv("F:\\6th Sem\\Machine Learning\\project\\train\_modified.csv",index=False)

test.to\_csv("F:\\6th Sem\\Machine Learning\\project\\test\_modified.csv",index=False)

**PREDICTING**

# -\*- coding: utf-8 -\*-

"""

Created on Wed Mar 27 21:20:26 2019

PREDICTING

"""

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import cross\_val\_score, cross\_val\_predict

from sklearn import metrics

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Ridge

from sklearn.tree import DecisionTreeRegressor

from xgboost import XGBRegressor

from sklearn.metrics import mean\_absolute\_error

from sklearn.ensemble import RandomForestRegressor

pd.options.display.max\_columns = 200

train\_df = pd.read\_csv("F:\\6th Sem\\Machine Learning\\project\\train\_modified.csv")

test\_df = pd.read\_csv("F:\\6th Sem\\Machine Learning\\project\\test\_modified.csv")

target = 'Purchase'

IDcol = ['User\_ID','Product\_ID']

def commonfit(alg, dtrain, dtest, predictors, target, IDcol, filename):

alg.fit(dtrain[predictors], dtrain[target])

dtrain\_predictions = alg.predict(dtrain[predictors])

cv\_score = cross\_val\_score(alg, dtrain[predictors],(dtrain[target]) , cv=20, scoring='neg\_mean\_squared\_error')

cv\_score = np.sqrt(np.abs(cv\_score))

print("\nModel Report")

print("RMSE : %.4g" % np.sqrt(metrics.mean\_squared\_error((dtrain[target]).values, dtrain\_predictions)))

print("CV Score : Mean - %.4g | Std - %.4g | Min - %.4g | Max - %.4g" % (np.mean(cv\_score),np.std(cv\_score),np.min(cv\_score),np.max(cv\_score)))

dtest[target] = alg.predict(dtest[predictors])

IDcol.append(target)

submission = pd.DataFrame({ x: dtest[x] for x in IDcol})

submission.to\_csv(filename, index=False)

LR = LinearRegression(normalize=True)

print("\nLinear Regression")

predictors = train\_df.columns.drop(['Purchase','Product\_ID','User\_ID'])

commonfit(LR, train\_df, test\_df, predictors, target, IDcol, 'LR.csv')

coef1 = pd.Series(LR.coef\_, predictors).sort\_values()

coef1.plot(kind='bar', title='Model Coefficients')

plt.show()

print("\nRidge Regression")

RR1 = Ridge(alpha=0.05,normalize=True)

commonfit(RR1, train\_df, test\_df, predictors, target, IDcol, 'RR.csv')

coef1 = pd.Series(RR1.coef\_, predictors).sort\_values()

coef1.plot(kind='bar', title='Model Coefficients')

plt.show()

print("\nDecision Tree Regression")

DT = DecisionTreeRegressor(max\_depth=15, min\_samples\_leaf=200)

commonfit(DT, train\_df, test\_df, predictors, target, IDcol, 'DT.csv')

importances = DT.feature\_importances\_

indices = np.argsort(importances)[::-1]

print("\nFeature ranking:")

for f in range(train\_df[predictors].shape[1]):

print("x%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

X=train\_df[predictors]

plt.figure()

plt.title("\nFeature importances")

plt.bar(range(X.shape[1]), importances[indices],color="y", align="center")

plt.xticks(range(X.shape[1]), indices)

plt.xlim([-1, X.shape[1]])

plt.show()

print("\nXGB Regression")

my\_model = XGBRegressor(n\_estimators=1000, learning\_rate=0.05)

print(my\_model.fit(train\_df[predictors], train\_df[target], early\_stopping\_rounds=5, eval\_set=[(test\_df[predictors], test\_df[target])], verbose=False))

train\_df\_predictions = my\_model.predict(train\_df[predictors])

predictions = my\_model.predict(test\_df[predictors])

print("\nMean Absolute Error : " + str(mean\_absolute\_error(predictions, test\_df[target])))

print("\nRMSE : %.4g" % np.sqrt(metrics.mean\_squared\_error((train\_df[target]).values, train\_df\_predictions)))

IDcol.append(target)

submission = pd.DataFrame({ x: test\_df[x] for x in IDcol})

submission.to\_csv("XGB.csv", index=False)

importances = my\_model.feature\_importances\_

indices = np.argsort(importances)[::-1]

print("\nFeature order:")

for f in range(train\_df[predictors].shape[1]):

print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

print("\nRandom Forest Regression")

RF=RandomForestRegressor(n\_estimators=200, max\_depth=3)

print(RF)

commonfit(RF, train\_df, test\_df, predictors, target, IDcol, 'RF.csv')

RF.fit(train\_df[predictors], train\_df[target])

train\_df\_predictions = RF.predict(train\_df[predictors])

predictions = RF.predict(test\_df[predictors])

print("Mean Absolute Error : " + str(mean\_absolute\_error(predictions, test\_df[target])))

print("RMSE : %.4g" % np.sqrt(metrics.mean\_squared\_error((train\_df[target]).values, train\_df\_predictions)))

importances = RF.feature\_importances\_

indices = np.argsort(importances)[::-1]

print("Feature order:")

for f in range(train\_df[predictors].shape[1]):

print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

X=train\_df[predictors]

plt.figure()

plt.title("Feature importances")

plt.bar(range(X.shape[1]), importances[indices],color="y", align="center")

plt.xlim([-1, X.shape[1]])

plt.show()

**RULE BASED LEARNING**  
# -\*- coding: utf-8 -\*-

"""

Created on Tue Apr 2 23:14:26 2019

RULE BASED LEARNING

"""

import pandas as pd

import numpy as np

from sklearn import metrics

from mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules

from sklearn.model\_selection import cross\_val\_score, cross\_val\_predict

from sklearn import tree

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeRegressor

import collections

dataset = pd.read\_csv("F:\\6th Sem\\Machine Learning\\project\\train.csv")

test = pd.read\_csv("F:\\6th Sem\\Machine Learning\\project\\test.csv")

train\_df = pd.read\_csv("F:\\6th Sem\\Machine Learning\\project\\train\_modified.csv")

test\_df = pd.read\_csv("F:\\6th Sem\\Machine Learning\\project\\test\_modified.csv")

target = 'Purchase'

predictors = train\_df.columns.drop(['Purchase','Product\_ID','User\_ID'])

IDcol = ['User\_ID','Product\_ID']

train\_df['User\_ID'] = train\_df['User\_ID'] - 1000000

test\_df['User\_ID'] = test\_df['User\_ID'] - 1000000

enc = LabelEncoder()

train\_df['User\_ID'] = enc.fit\_transform(train\_df['User\_ID'])

test\_df['User\_ID'] = enc.transform(test\_df['User\_ID'])

train\_df['Product\_ID'] = train\_df['Product\_ID'].str.replace('P00', '')

test\_df['Product\_ID'] = test\_df['Product\_ID'].str.replace('P00', '')

scaler = StandardScaler()

train\_df['Product\_ID'] = scaler.fit\_transform(train\_df['Product\_ID'].values.reshape(-1, 1))

test\_df['Product\_ID'] = scaler.transform(test\_df['Product\_ID'].values.reshape(-1, 1))

def encode\_units(x):

if x <= 0:

return 0

if x >= 1:

return 1

def find\_node(tree\_, current\_node, search\_node, features):

child\_left = tree\_.children\_left[current\_node]

child\_right = tree\_.children\_right[current\_node]

split\_feature = str(features[tree\_.feature[current\_node]])

split\_value = str(tree\_.threshold[current\_node])

if child\_left != -1:

if child\_left != search\_node:

left\_one = find\_node(tree\_, child\_left, search\_node, features)

else:

return(str(split\_feature)+" <= "+str(split\_value))

else:

return ""

if child\_right != -1:

if child\_right != search\_node:

right\_one = find\_node(tree\_, child\_right, search\_node, features)

else:

return(str(split\_feature)+" > "+str(split\_value))

else:

return ""

if len(left\_one)>0:

return(str(split\_feature)+" <= "+str(split\_value)+", "+left\_one)

elif len(right\_one)>0:

return(str(split\_feature)+" > "+str(split\_value)+","+right\_one)

else:

return ""

def commonfit(alg, dtrain, dtest, predictors, target, IDcol, filename):

alg.fit(dtrain[predictors], dtrain[target])

dtrain\_predictions = alg.predict(dtrain[predictors])

cv\_score = cross\_val\_score(alg, dtrain[predictors],(dtrain[target]) , cv=20, scoring='neg\_mean\_squared\_error')

cv\_score = np.sqrt(np.abs(cv\_score))

print("\nModel Report")

print("RMSE : %.4g" % np.sqrt(metrics.mean\_squared\_error((dtrain[target]).values, dtrain\_predictions)))

print("CV Score : Mean - %.4g | Std - %.4g | Min - %.4g | Max - %.4g" % (np.mean(cv\_score),np.std(cv\_score),np.min(cv\_score),np.max(cv\_score)))

dtest[target] = alg.predict(dtest[predictors])

IDcol.append(target)

submission = pd.DataFrame({ x: dtest[x] for x in IDcol})

submission.to\_csv(filename, index=False)

print('\nRule Based Learning')

#print(train\_df)

'''newtrain = train\_df.applymap(encode\_units)

print(newtrain)

newtrain=newtrain.dropna()

predictors1 = train\_df.columns.drop(['Product\_ID','User\_ID','Marital\_Status','Stay\_In\_Current\_City\_Years'])

frequent\_itemsets = apriori(newtrain[predictors1], min\_support=0.07, use\_colnames=True)

rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1)

print(rules)

print(rules[ (rules['lift'] > 1.0) & (rules['confidence'] > 0.73)])

'''

X=train\_df[predictors].loc[:2000,]

y=train\_df[target].loc[:2000,]

clf = tree.DecisionTreeRegressor()

clf = clf.fit(X,y)

print(clf)

pd.DataFrame(clf.decision\_path(X).toarray()).head(5)

pd.concat([X.reset\_index(drop=True),pd.DataFrame(clf.decision\_path(X).toarray())],1).head(5)

print("\n\n",find\_node(tree\_ = clf.tree\_, current\_node = 0, search\_node = 13, features = X.columns.tolist()),"\n\n")

print(dataset[(dataset['Purchase'] >= 10000)])

dTree3 = DecisionTreeRegressor(max\_depth = 6)

commonfit(dTree3, train\_df, test\_df, predictors, target, IDcol, 'DT-new.csv')

Xrules = pd.concat([X.reset\_index(drop=True),pd.DataFrame(dTree3.decision\_path(X).toarray()).iloc[:,1:]],1)

tree\_model = DecisionTreeRegressor()

tree\_model.fit(Xrules, y)\

commonfit(tree\_model, train\_df, test\_df, predictors, target, IDcol, 'DT-new.csv')