Black Friday Sales Analysis using various classifiers and regressors

Abstract

During the Black Friday sale, all the retail shops are crowded. Most products are marked down with discounts and customers rush in to buy the products. It is difficult for customers to buy the products even with a solid plan.

But, the shop owners face even more difficulty on controlling the crowd with limited staff and in targeting prospective customers. Several techniques have been employed to tackle this problem, but they are not that successful. A prediction model is a technique that has proved promising in solving the problem.

We focus on the field of prediction models to develop an accurate and efficient algorithm to analyse the customer spending in the past and output the future spending of the customers with same features.

Different machine learning techniques such as regression and neural network to develop a prediction model are implemented and a comparison is done based on their performance and accuracy of prediction. These techniques are implemented using different algorithms to find the best predication.

We implemented six different machine learning algorithms. Further, we apply the data pre-processing and visualization techniques to attain the optimal results.

Keywords: Black Friday Sales, Prediction model, Regression, Neural network, Machine Learning

1. Introduction

In the past, there were no supermarkets or departmental stores, only small businesses. The store owners knew their customers and their spending patterns, their likes and their dislikes. But as the small business grew into large franchises with hundreds of stores across the country, it became near to impossible to know the customers and their personal preferences. Some examples of such franchises are Costco, Walmart, and Wholefoods.

These stores without any proper knowledge of their customer base are struggling to satisfy the customer needs. Thus, prediction models are needed to better understand customer preferences

Building a prediction model depends on various features such as the location and the time. Black Friday is the largest shopping day of the year in United States of America. Black Friday is the day after Thanksgiving Day which marks the beginning of the shopping season for Christmas.

A prediction model developed for Black Friday can only be used during that day because customer spending differs drastically between a normal day and a Black Friday; this is because discounts and price reductions attract more customers.

Finally, better visualization techniques are required to portray the findings and help the store owners understand their customers.

Problem statement

Can an accurate prediction model be developed?

Which algorithm is better and efficient for such model?

Will data pre-processing and visualization technique increase the accuracy?

2. Literature survey

Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (predictor) This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables. The shape of regression line, the type of dependent variable and number of independent variable.

1. **Linear Regression**
   Linear Regression establishes a relationship between dependent variable (Y) and one or more independent variables (X) using a best fit straight line (also known as regression line). It is represented by an equation Y=a+b*X + e, where a is intercept, b is slope of the line and e is error term. This equation can be used to predict the value of target variable based on given predictor variable(s).
2. **Ridge Regression**
   Ridge Regression is a technique used when the data suffers from multicollinearity (independent variables are highly correlated). In multicollinearity, even though the least squares estimates (OLS) are unbiased, their variances are large which deviates the observed value far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors.
3. **Decision Tree**
   Machine learning algorithms like decision tree and 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 and statistical model, and artificial intelligence model
4. **XGBoost**
   The XGBoost model internally implements the stepwise, ridge regression which dynamically selects the features and 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 and eliminate the less important features to develop a strong model.
5. **Random forest**
   Random forest is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

The major problem with the existing prediction model is that the data used for development contains several irregularities such as missing values or wrong information. Also, selection of right algorithm plays a major role in developing an accurate model.

3. 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 we will just analysis various components of our data like mean, median, standard deviation, frequency etc. Also, we'll find skew, kurtosis followed by correlation matrix with respect to Purchase values.

2. Data pre-processing

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

3. Machine learning techniques

After pre-processing the data, we will use the modified train and 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.

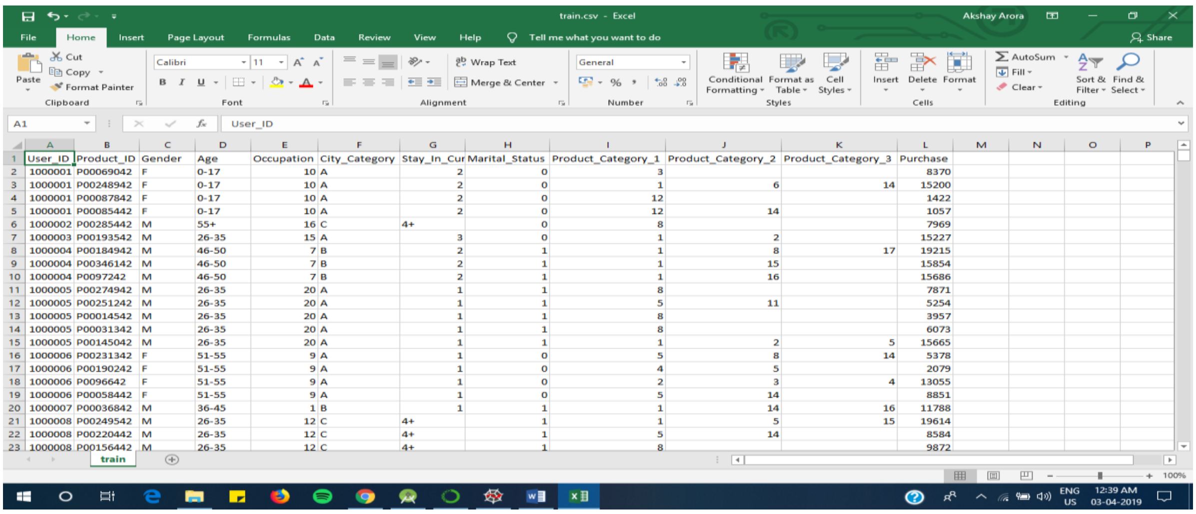
4. Rule Based Learning

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

Dataset Used-

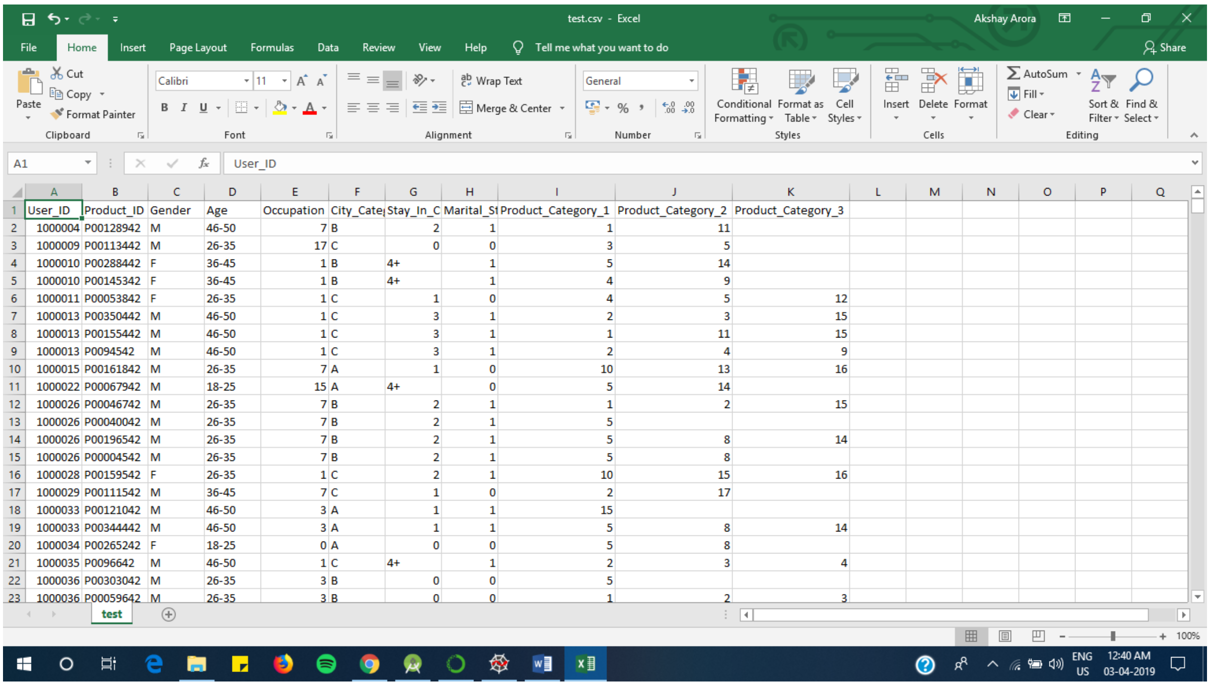
Our dataset is the Black Friday Sales Dataset in Kaggle. In this dataset we have the information about the Age, Occupation, City, Duration stayed, Marital status, the quantity of products bought of various types and the total amount spent. We are using these inputs to find the most necessary attributes,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

Train.csv



	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Cur	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
2	1000001	P00069042	F	0-17	10	A	2	0	3			8370
3	1000001	P00248942	F	0-17	10	A	2	0	1	6	14	15200
4	1000001	P00087842	F	0-17	10	A	2	0	12			1422
5	1000001	P00085442	F	0-17	10	A	2	0	12	14		1057
6	1000002	P00285442	M	55+	16	C	4+	0	8			7969
7	1000003	P00193542	M	26-35	15	A	3	0	1	2		15227
8	1000004	P00184942	M	46-50	7	B	2	1	1	8	17	19215
9	1000004	P00346142	M	46-50	7	B	2	1	1	15		15854
10	1000004	P0097242	M	46-50	7	B	2	1	1	16		15686
11	1000005	P00274942	M	26-35	20	A	1	1	8			7871
12	1000005	P00251242	M	26-35	20	A	1	1	5	11		5254
13	1000005	P00014542	M	26-35	20	A	1	1	8			3957
14	1000005	P00031342	M	26-35	20	A	1	1	8			6073
15	1000005	P00145042	M	26-35	20	A	1	1	1	2	5	15665
16	1000006	P00231342	F	51-55	9	A	1	0	5	8	14	5378
17	1000006	P00190242	F	51-55	9	A	1	0	4	5		2079
18	1000006	P0096642	F	51-55	9	A	1	0	2	3	4	13055
19	1000006	P00058442	F	51-55	9	A	1	0	5	14		8851
20	1000007	P00036842	M	36-45	1	B	1	1	1	14	16	11788
21	1000008	P00249542	M	26-35	12	C	4+	1	1	5	15	19614
22	1000008	P00220442	M	26-35	12	C	4+	1	5	14		8584
23	1000008	P00156442	M	26-35	12	C	4+	1	8			9872

Test.csv (No Purchase Values)



User_ID	Product_ID	Gender	Age	Occupation	City_Cate	Stay_In_C	Marital_St	Product_Category_1	Product_Category_2	Product_Category_3
1000004	P00128942	M	46-50	7	B	2	1	1	11	
1000009	P00113442	M	26-35	17	C	0	0	3	5	
1000010	P00288442	F	36-45	1	B	4+	1	5	14	
1000010	P00145342	F	36-45	1	B	4+	1	4	9	
1000011	P00053842	F	26-35	1	C	1	0	4	5	12
1000013	P00350442	M	46-50	1	C	3	1	2	3	15
1000013	P00155442	M	46-50	1	C	3	1	1	11	15
1000013	P0094542	M	46-50	1	C	3	1	2	4	9
1000015	P00161842	M	26-35	7	A	1	0	10	13	16
1000022	P00067942	M	18-25	15	A	4+	0	5	14	
1000026	P00046742	M	26-35	7	B	2	1	1	2	15
1000026	P00040042	M	26-35	7	B	2	1	5		
1000026	P00196542	M	26-35	7	B	2	1	5	8	14
1000026	P00004542	M	26-35	7	B	2	1	5	8	
1000028	P00159542	F	26-35	1	C	2	1	10	15	16
1000029	P00111542	M	36-45	7	C	1	0	2	17	
1000033	P00121042	M	46-50	3	A	1	1	15		
1000033	P00344442	M	46-50	3	A	1	1	5	8	14
1000034	P00265242	F	18-25	0	A	0	0	5	8	
1000035	P0096642	M	46-50	1	C	4+	1	2	3	4
1000036	P00303042	M	26-35	3	B	0	0	5		
1000036	P00059642	M	26-35	3	B	0	0	1	2	3

4. Proposed System Analysis

Data Analysing

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

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

	User_ID	Product_ID	Gender	Age	Occupation	City_Category \
0	1000001	P00069042	F	0-17	10	A
1	1000001	P00248942	F	0-17	10	A
2	1000001	P00087842	F	0-17	10	A
3	1000001	P00085442	F	0-17	10	A
4	1000002	P00285442	M	55+	16	C

	Stay_In_Current_City_Years	Marital_Status	Product_Category_1 \
0	2	0	3
1	2	0	1
2	2	0	12
3	2	0	12

4	4+	0	8

	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

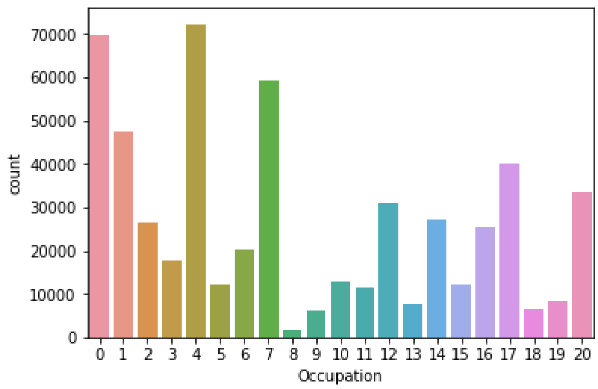
	User_ID	Occupation	Marital_Status	Product_Category_1 \
count	5.500680e+05	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270
std	1.727592e+03	6.522660	0.491770	3.936211
min	1.000001e+06	0.000000	0.000000	1.000000
25%	1.001516e+06	2.000000	0.000000	1.000000
50%	1.003077e+06	7.000000	0.000000	5.000000
75%	1.004478e+06	14.000000	1.000000	8.000000
max	1.006040e+06	20.000000	1.000000	20.000000

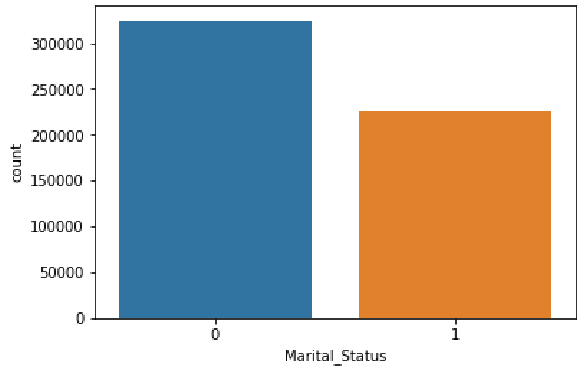
	Product_Category_2	Product_Category_3	Purchase
count	376430.000000	166821.000000	550068.000000
mean	9.842329	12.668243	9263.968713
std	5.086590	4.125338	5023.065394
min	2.000000	3.000000	12.000000
25%	5.000000	9.000000	5823.000000
50%	9.000000	14.000000	8047.000000
75%	15.000000	16.000000	12054.000000
max	18.000000	18.000000	23961.000000

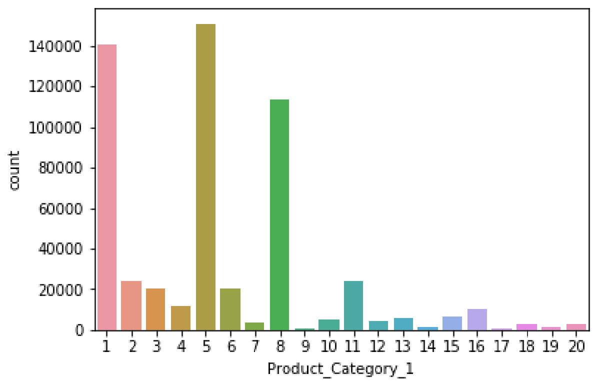
There are 544177 duplicate IDs for 550068 total entries

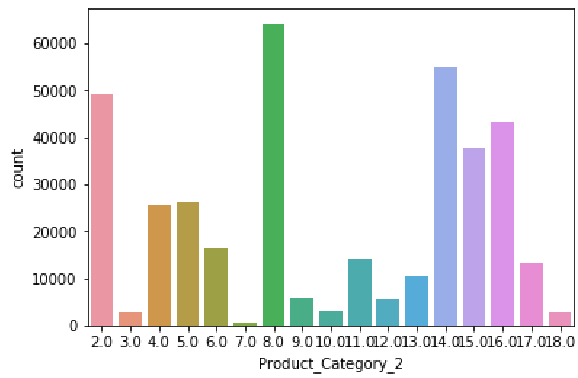
Skew is: 0.6001400037087128

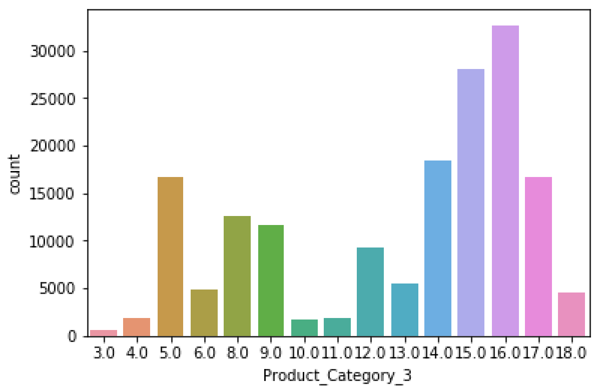
Kurtosis: -0.338378

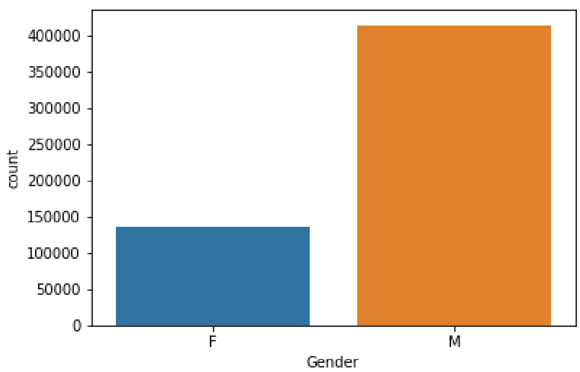

count
0 10000 20000 30000 40000 50000 60000 70000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Occupation

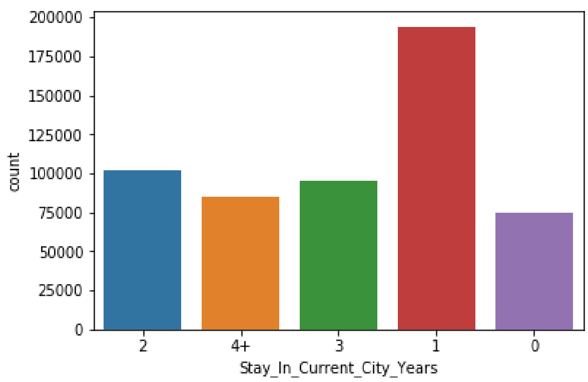

count
0 50000 100000 150000 200000 250000 300000
0 1
Marital_Status

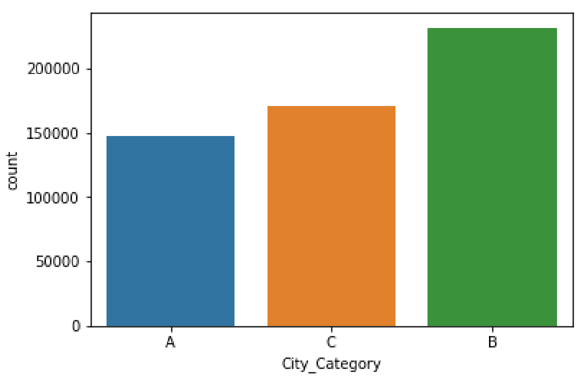

count
0 20000 40000 60000 80000 100000 120000 140000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Product_Category_1


count
0 10000 20000 30000 40000 50000 60000
2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0 17.0 18.0
Product_Category_2


count
0 5000 10000 15000 20000 25000 30000
3.0 4.0 5.0 6.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0 17.0 18.0
Product_Category_3


count
0 50000 100000 150000 200000 250000 300000 350000 400000
F M
Gender


count
0 25000 50000 75000 100000 125000 150000 175000 200000
2 4+ 3 1 0
Stay_In_Current_City_Years

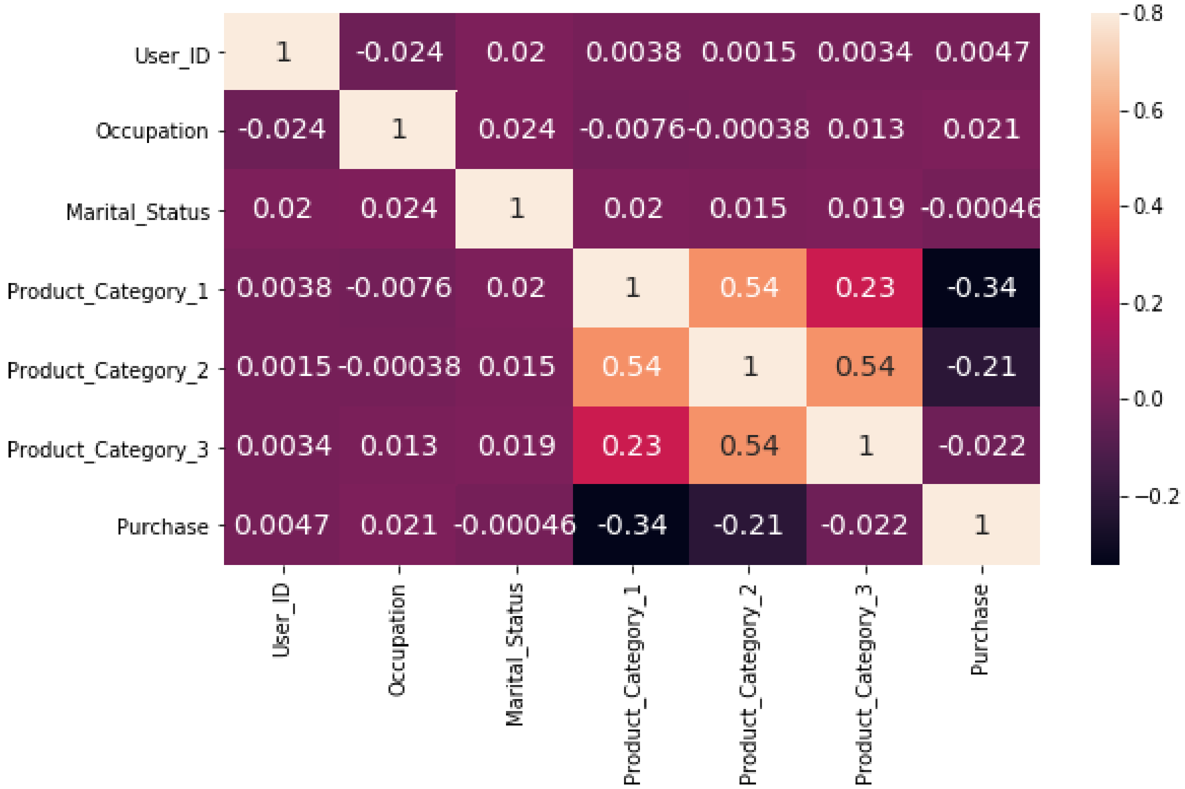

count
0 50000 100000 150000 200000
A C B
City_Category

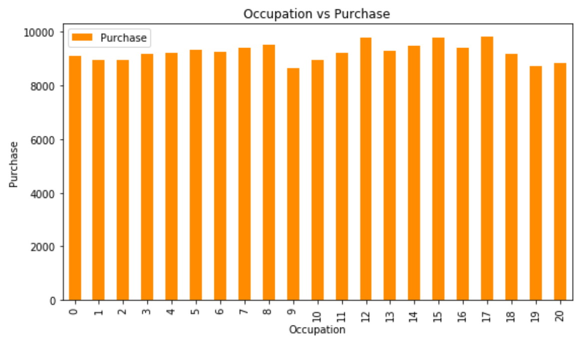
Correlation from Purchase

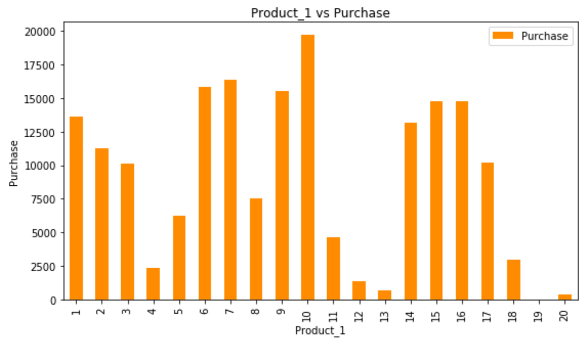
Purchase	1.000000
Occupation	0.020833
User_ID	0.004716
Marital_Status	-0.000463
Product_Category_3	-0.022006
Product_Category_2	-0.209918
Product_Category_1	-0.343703

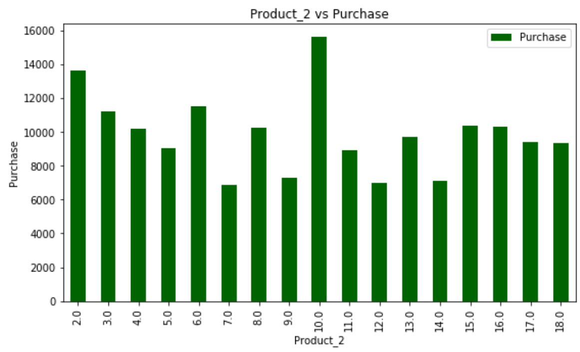
Name: Purchase, dtype: float64

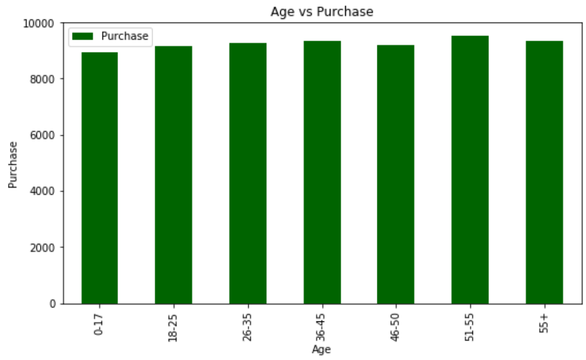
Correlation Matrix

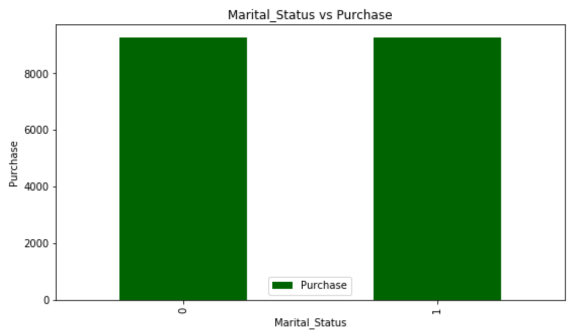


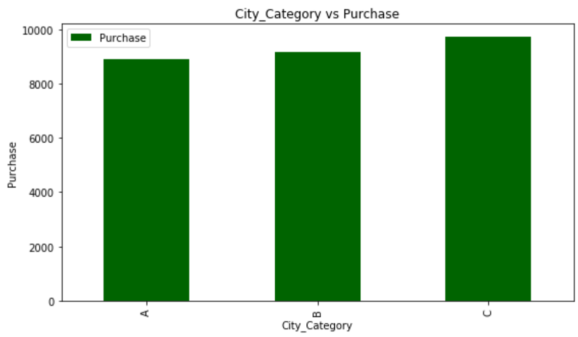

Occupation vs Purchase
Purchase
Purchase
0 2000 4000 6000 8000 10000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Occupation

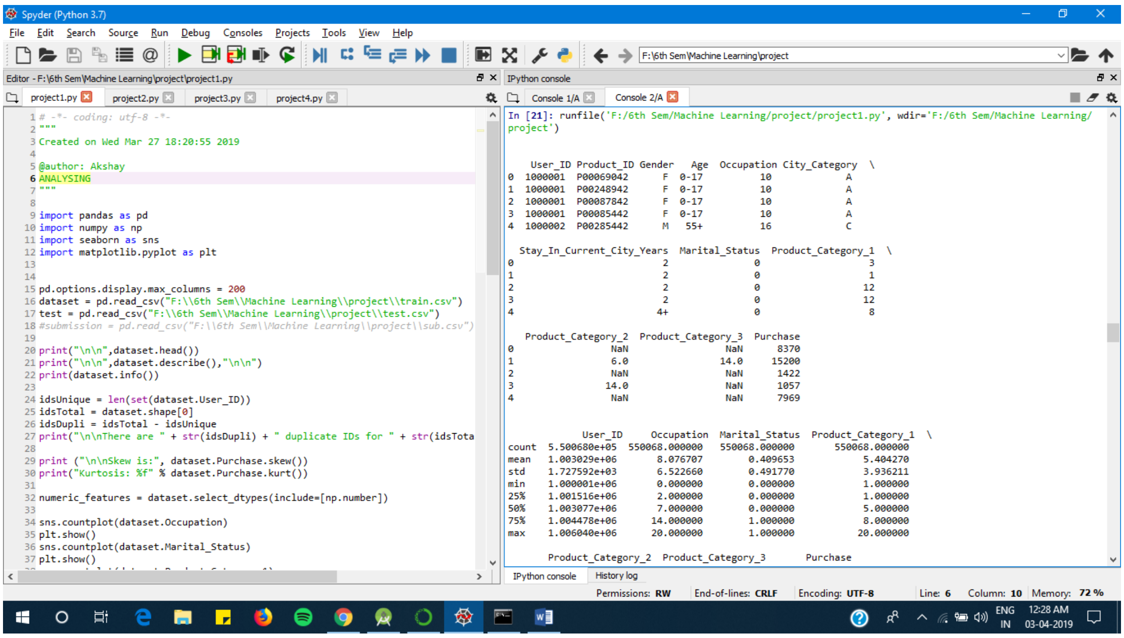

Product_1 vs Purchase
Purchase
Purchase
0 2500 5000 7500 10000 12500 15000 17500 20000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Product_1

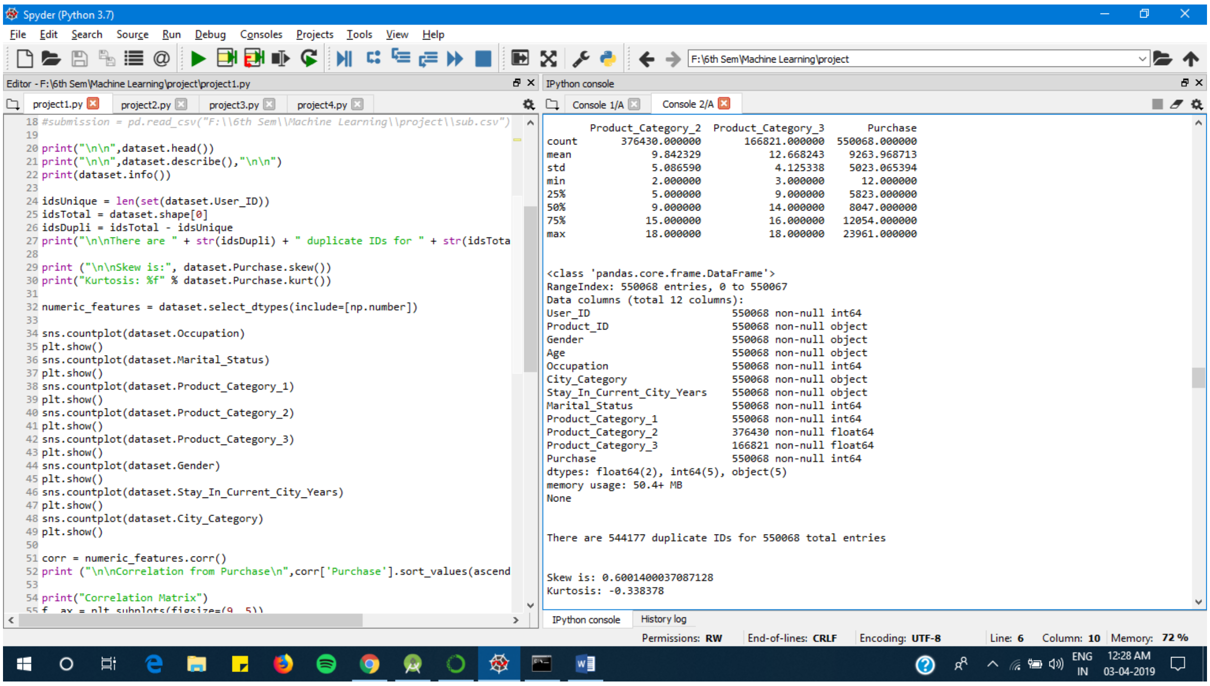

Product_2 vs Purchase
Purchase
Purchase
0 2000 4000 6000 8000 10000 12000 14000 16000
2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0 17.0 18.0
Product_2

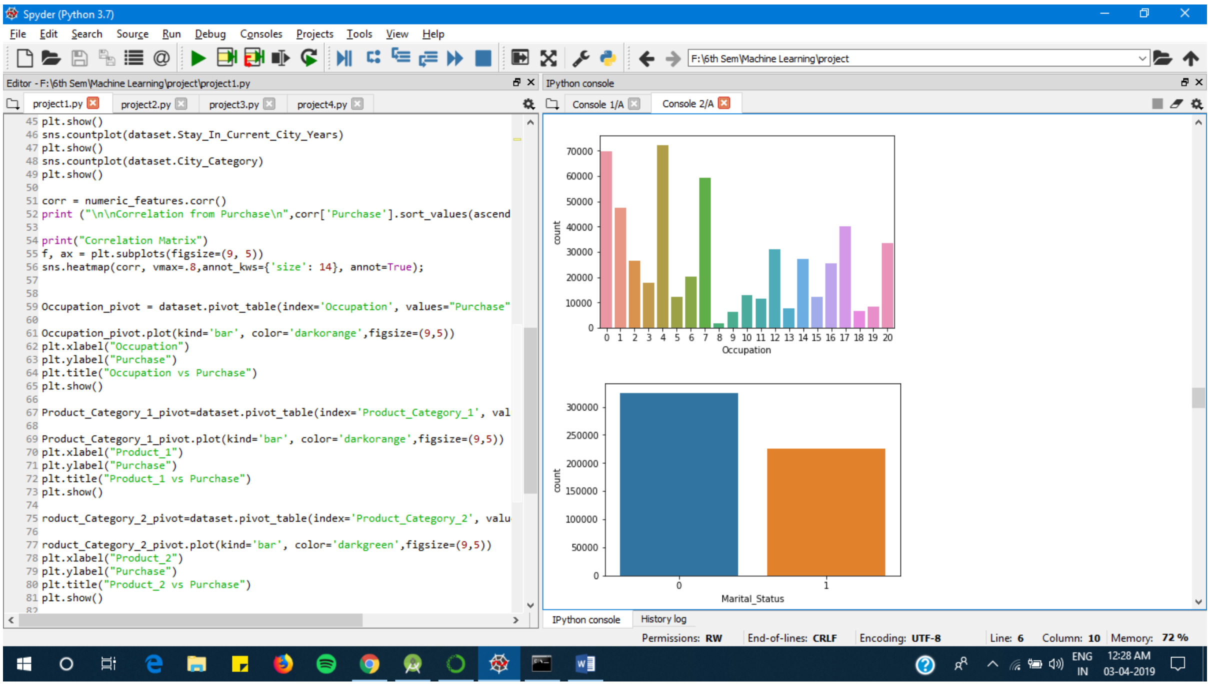

Age vs Purchase
Purchase
Purchase
0 2000 4000 6000 8000 10000
0-17 18-25 26-35 36-45 46-50 51-55 55+
Age

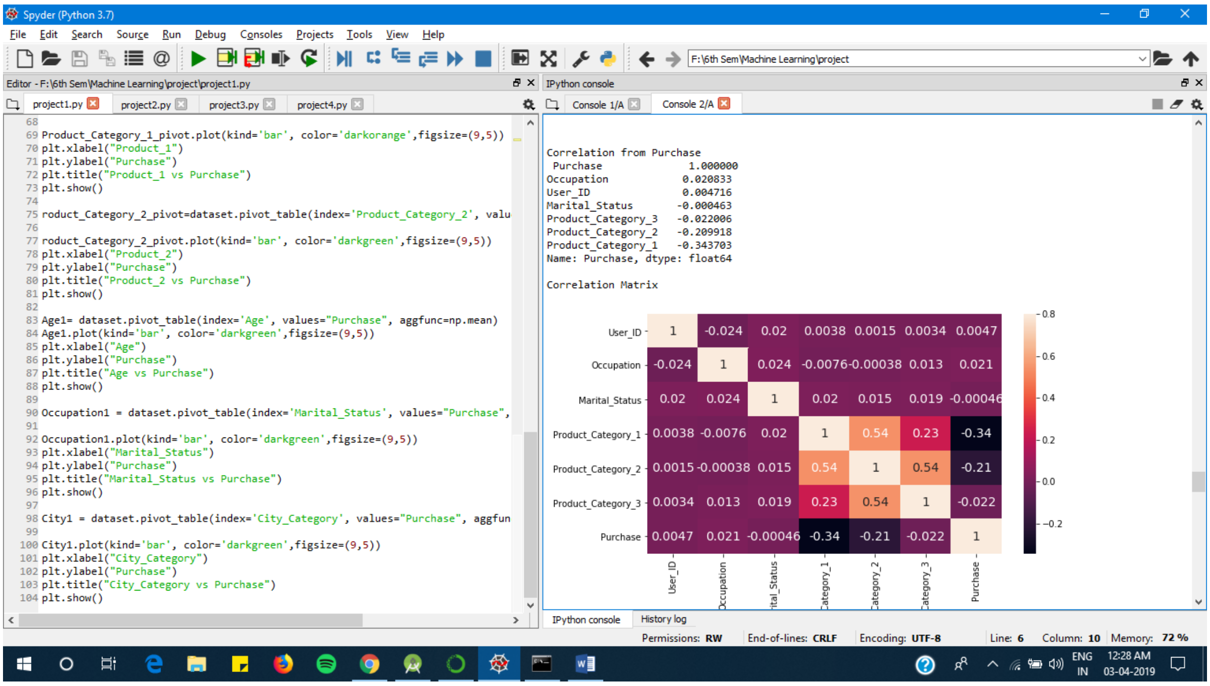

Marital_Status vs Purchase
Purchase
Purchase
0 2000 4000 6000 8000
0 1
Marital_Status


City_Category vs Purchase
Purchase
Purchase
0 2000 4000 6000 8000 10000
A B C
City_Category


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\6th Sem\Machine Learning\project
Editor - F:\6th Sem\Machine Learning\project\project1.py
project1.py project2.py project3.py project4.py
-*- coding: utf-8 -*-
"""
Created on Wed Mar 27 18:20:55 2019
@author: Akshay
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(idsTota
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()
IPython console
Console 1/A Console 2/A
In [21]: runfile('F:/6th Sem/Machine Learning/project/project1.py', wdir='F:/6th Sem/Machine Learning/project')
User_ID Product_ID Gender Age Occupation City_Category \
0 1000001 P00069042 F 0-17 10 A
1 1000001 P00248942 F 0-17 10 A
2 1000001 P00087842 F 0-17 10 A
3 1000001 P00085442 F 0-17 10 A
4 1000002 P00285442 M 55+ 16 C
Stay_In_Current_City_Years Marital_Status Product_Category_1 \
0 2 0 3
1 2 0 1
2 2 0 12
3 2 0 12
4 4+ 0 8
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
User_ID Occupation Marital_Status Product_Category_1 \
count 5.500680e+05 550068.000000 550068.000000 550068.000000
mean 1.003029e+06 8.076707 0.409653 5.404270
std 1.727592e+03 6.522660 0.491770 3.936211
min 1.000001e+06 0.000000 0.000000 1.000000
25% 1.001516e+06 2.000000 0.000000 1.000000
50% 1.003077e+06 7.000000 0.000000 5.000000
75% 1.004478e+06 14.000000 1.000000 8.000000
max 1.006040e+06 20.000000 1.000000 20.000000
Product_Category_2 Product_Category_3 Purchase
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 6 Column: 10 Memory: 72 %
ENG IN 12:28 AM 03-04-2019


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\6th Sem\Machine Learning\project
Editor - F:\6th Sem\Machine Learning\project\project1.py
project1.py project2.py project3.py project4.py
#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(idsTota
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(ascend
print("Correlation Matrix")
IPython console
Console 1/A Console 2/A
Product_Category_2 Product_Category_3 Purchase
count 376430.000000 166821.000000 550068.000000
mean 9.842329 12.668243 9263.968713
std 5.086590 4.125338 5023.065394
min 2.000000 3.000000 12.000000
25% 5.000000 9.000000 5823.000000
50% 9.000000 14.000000 8047.000000
75% 15.000000 16.000000 12054.000000
max 18.000000 18.000000 23961.000000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
User_ID 550068 non-null int64
Product_ID 550068 non-null object
Gender 550068 non-null object
Age 550068 non-null object
Occupation 550068 non-null int64
City_Category 550068 non-null object
Stay_In_Current_City_Years 550068 non-null object
Marital_Status 550068 non-null int64
Product_Category_1 550068 non-null int64
Product_Category_2 376430 non-null float64
Product_Category_3 166821 non-null float64
Purchase 550068 non-null int64
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
None
There are 544177 duplicate IDs for 550068 total entries
Skew is: 0.6001400037087128
Kurtosis: -0.338378
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 6 Column: 10 Memory: 72 %
ENG IN 12:28 AM 03-04-2019


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\6th Sem\Machine Learning\project
Editor - F:\6th Sem\Machine Learning\project\project1.py
project1.py project2.py project3.py project4.py
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(ascend
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"
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', val
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', valu
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()
IPython console
Console 1/A Console 2/A
70000
60000
50000
40000
30000
20000
10000
0
count
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Occupation
300000
250000
200000
150000
100000
50000
0
count
0 1
Marital_Status
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 6 Column: 10 Memory: 72 %
ENG IN 12:28 AM 03-04-2019



PRE- PROCESSING

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

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

Because of our dataset being majority numerical in nature, we use the partitioning technique to remove the presence of unique non-numerical values and 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')

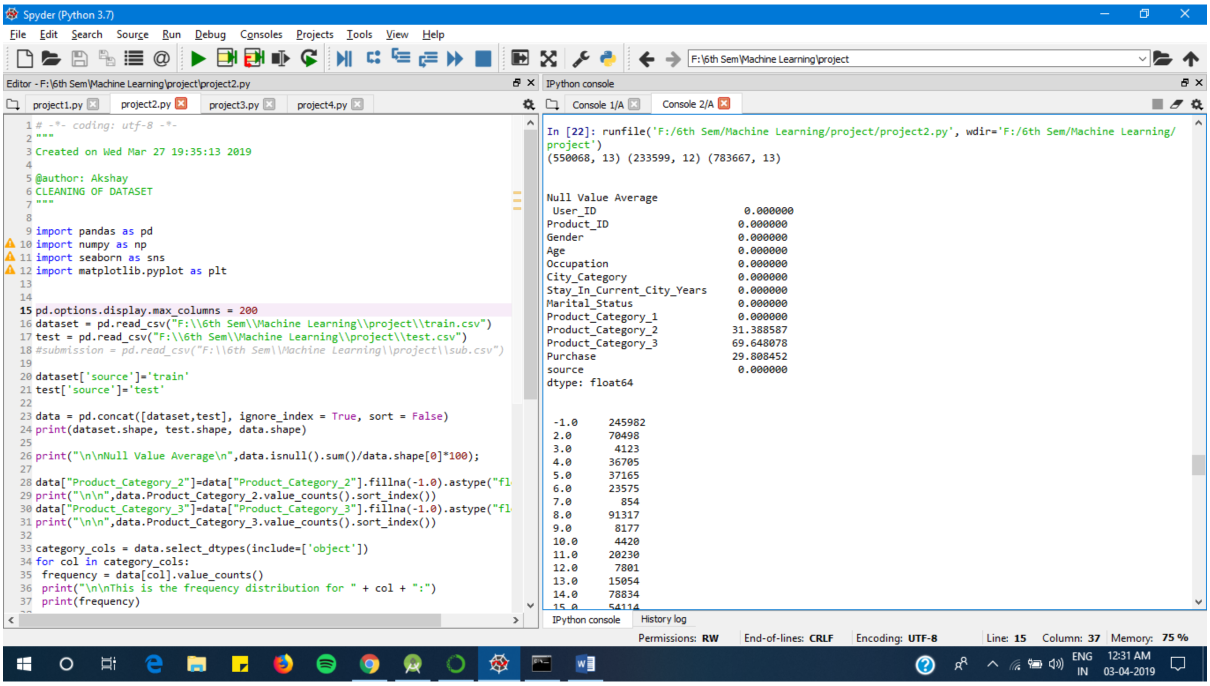
[0 1 2]

2   329739

1   243684

0   210244

Name: City_Category, dtype: int64


Spyder (Python 3.7)
Editor - F:\6th Sem\Machine Learning\project\project2.py
project1.py
project2.py
project3.py
project4.py
-*- coding: utf-8 -*-
"""
Created on Wed Mar 27 19:35:13 2019

@author: Akshay
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)

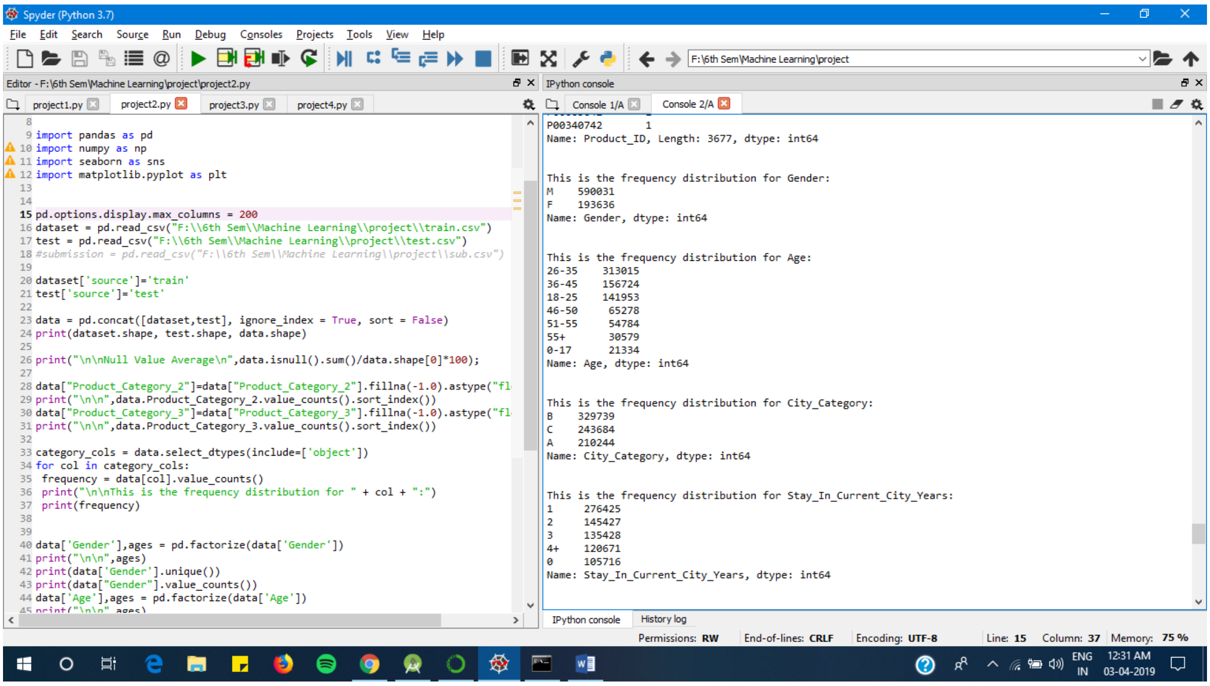
print("\n\nNull Value Average\n",data.isnull().sum()/data.shape[0]*100);

data["Product_Category_2"]=data["Product_Category_2"].fillna(-1.0).astype("fl
print("\n\n",data.Product_Category_2.value_counts().sort_index())
data["Product_Category_3"]=data["Product_Category_3"].fillna(-1.0).astype("fl
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)
IPython console
Console 1/A
Console 2/A
In [22]: runfile('F:/6th Sem/Machine Learning/project/project2.py', wdir='F:/6th Sem/Machine Learning/project')
(550068, 13) (233599, 12) (783667, 13)

Null Value Average
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 31.388587
Product_Category_3 69.648078
Purchase 29.808452
source 0.000000
dtype: float64

-1.0 245982
2.0 70498
3.0 4123
4.0 36705
5.0 37165
6.0 23575
7.0 854
8.0 91317
9.0 8177
10.0 4420
11.0 20230
12.0 7801
13.0 15054
14.0 78834
IPython console
History log
Permissions: RW
End-of-lines: CRLF
Encoding: UTF-8
Line: 15
Column: 37
Memory: 75 %
ENG
IN
12:31 AM
03-04-2019


Spyder (Python 3.7)
Editor - F:\6th Sem\Machine Learning\project\project2.py
project1.py
project2.py
project3.py
project4.py
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("fl
print("\n\n",data.Product_Category_2.value_counts().sort_index())
data["Product_Category_3"]=data["Product_Category_3"].fillna(-1.0).astype("fl
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)

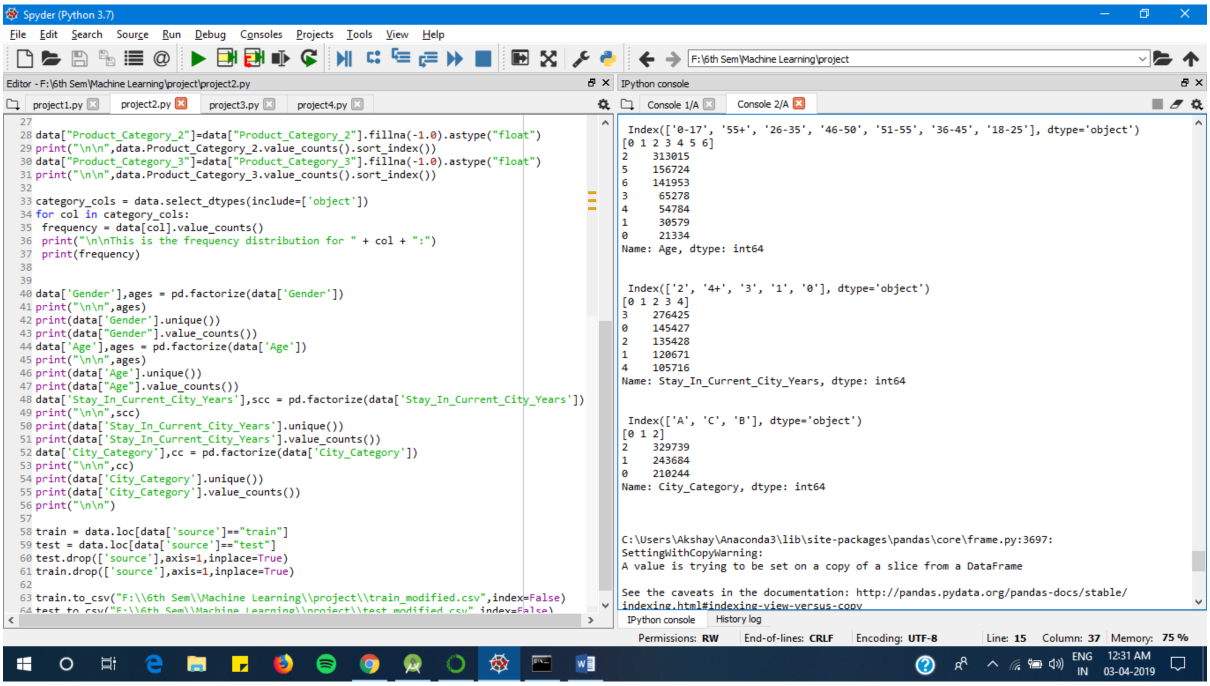
data['Gender'],ages = pd.factorize(data['Gender'])
print("\n\n",ages)
print(data['Gender'].unique())
print(data["Gender"].value_counts())
data['Age'],ages = pd.factorize(data['Age'])
IPython console
Console 1/A
Console 2/A
P00340742 1
Name: Product_ID, Length: 3677, dtype: int64

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

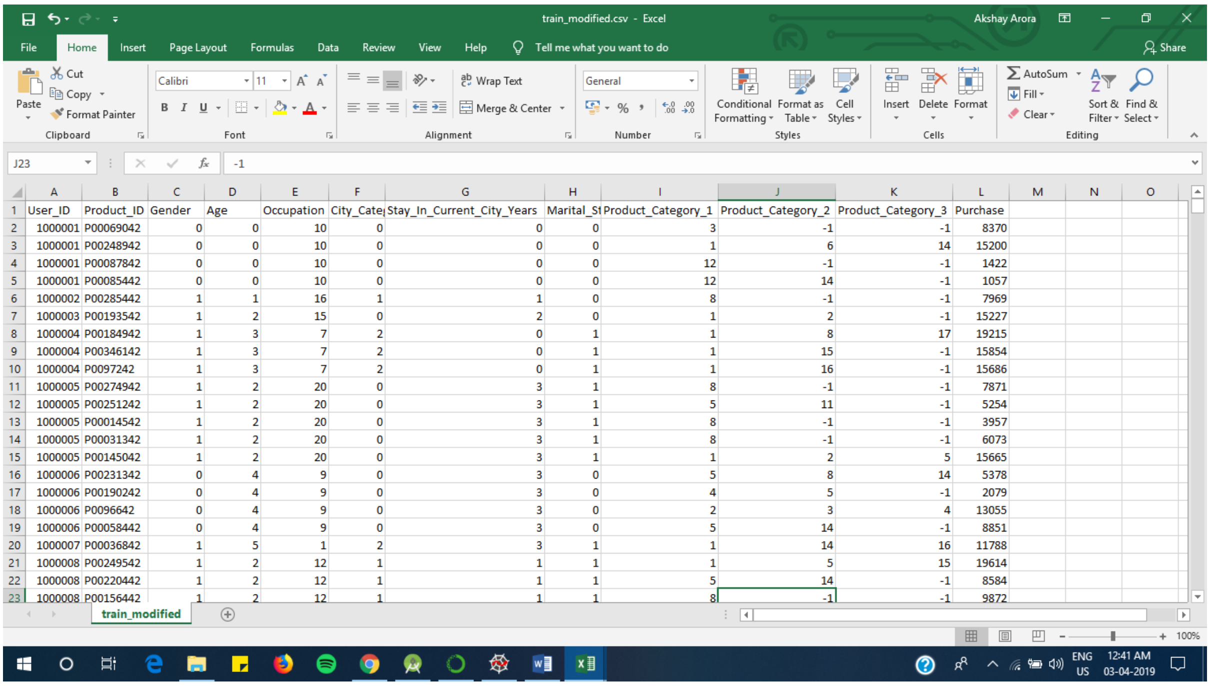
This is the frequency distribution for City_Category:
B 329739
C 243684
A 210244
Name: City_Category, dtype: int64

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
IPython console
History log
Permissions: RW
End-of-lines: CRLF
Encoding: UTF-8
Line: 15
Column: 37
Memory: 75 %
ENG
IN
12:31 AM
03-04-2019



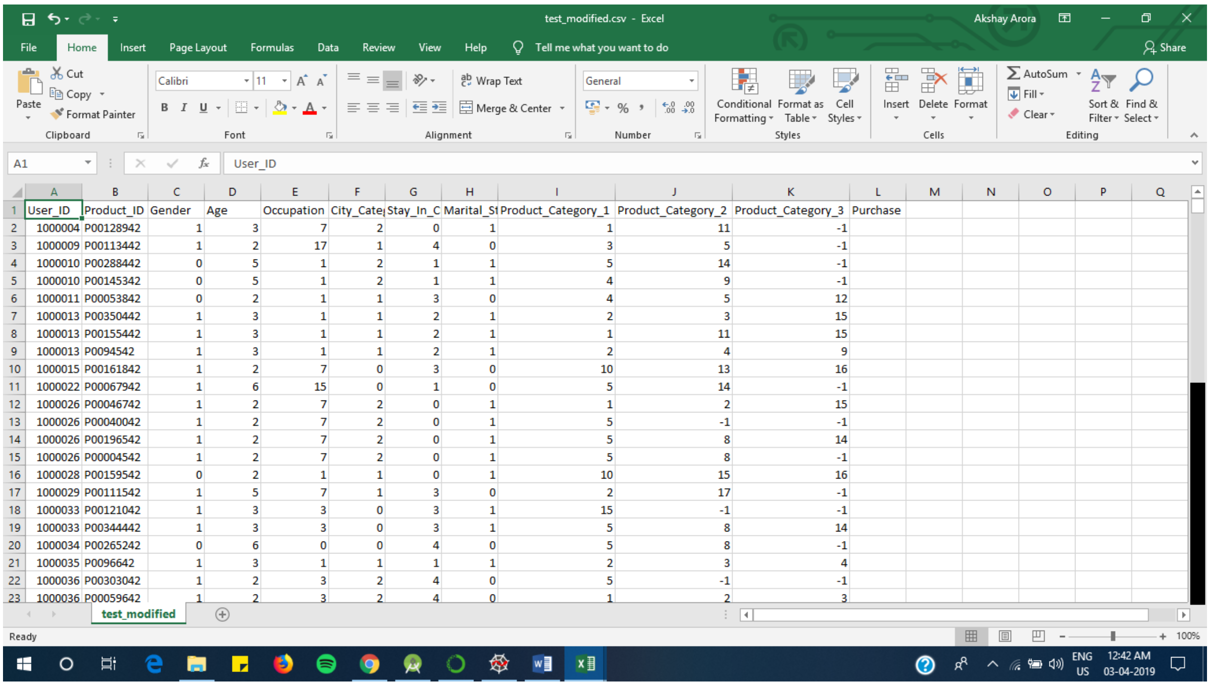
Modified Datasets-

Train_modified.csv



User_ID	Product_ID	Gender	Age	Occupation	City_Cate	Stay_In_Current_City_Years	Marital_St	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
1000001	P00069042	0	0	10	0	0	0	3	-1	-1	8370
1000001	P00248942	0	0	10	0	0	0	1	6	14	15200
1000001	P00087842	0	0	10	0	0	0	12	-1	-1	1422
1000001	P00085442	0	0	10	0	0	0	12	14	-1	1057
1000002	P00285442	1	1	16	1	1	0	8	-1	-1	7969
1000003	P00193542	1	2	15	0	2	0	1	2	-1	15227
1000004	P00184942	1	3	7	2	0	1	1	8	17	19215
1000004	P00346142	1	3	7	2	0	1	1	15	-1	15854
1000004	P0097242	1	3	7	2	0	1	1	16	-1	15686
1000005	P00274942	1	2	20	0	3	1	8	-1	-1	7871
1000005	P00251242	1	2	20	0	3	1	5	11	-1	5254
1000005	P00014542	1	2	20	0	3	1	8	-1	-1	3957
1000005	P00031342	1	2	20	0	3	1	8	-1	-1	6073
1000005	P00145042	1	2	20	0	3	1	1	2	5	15665
1000006	P00231342	0	4	9	0	3	0	5	8	14	5378
1000006	P00190242	0	4	9	0	3	0	4	5	-1	2079
1000006	P0096642	0	4	9	0	3	0	2	3	4	13055
1000006	P00058442	0	4	9	0	3	0	5	14	-1	8851
1000007	P00036842	1	5	1	2	3	1	1	14	16	11788
1000008	P00249542	1	2	12	1	1	1	1	5	15	19614
1000008	P00220442	1	2	12	1	1	1	5	14	-1	8584
1000008	P00156442	1	2	12	1	1	1	8	-1	-1	9872

Test_Modified.csv



User_ID	Product_ID	Gender	Age	Occupation	City_Cate	Stay_In_C	Marital_St	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
1000004	P00128942	1	3	7	2	0	1	1	11	-1	
1000009	P00113442	1	2	17	1	4	0	3	5	-1	
1000010	P00288442	0	5	1	2	1	1	5	14	-1	
1000010	P00145342	0	5	1	2	1	1	4	9	-1	
1000011	P00053842	0	2	1	1	3	0	4	5	12	
1000013	P00350442	1	3	1	1	2	1	2	3	15	
1000013	P00155442	1	3	1	1	2	1	1	11	15	
1000013	P0094542	1	3	1	1	2	1	2	4	9	
1000015	P00161842	1	2	7	0	3	0	10	13	16	
1000022	P00067942	1	6	15	0	1	0	5	14	-1	
1000026	P00046742	1	2	7	2	0	1	1	2	15	
1000026	P00040042	1	2	7	2	0	1	5	-1	-1	
1000026	P00196542	1	2	7	2	0	1	5	8	14	
1000026	P00004542	1	2	7	2	0	1	5	8	-1	
1000028	P00159542	0	2	1	1	0	1	10	15	16	
1000029	P00111542	1	5	7	1	3	0	2	17	-1	
1000033	P00121042	1	3	3	0	3	1	15	-1	-1	
1000033	P00344442	1	3	3	0	3	1	5	8	14	
1000034	P00265242	0	6	0	0	4	0	5	8	-1	
1000035	P0096642	1	3	1	1	1	1	2	3	4	
1000036	P00303042	1	2	3	2	4	0	5	-1	-1	
1000036	P00059642	1	2	3	2	4	0	1	2	3	

5. Implementation

Before selecting the models to use for the training process, we need to decide the columns/features that can be used as predictors and 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 we made the correlation matrix of the dataset, we 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 and test data and apply machine learning algorithms of various types as read about in the survey.

To predict the purchase amount using multiple regression we implemented machine learning algorithms and compared them on accuracy and performance metric. Since it is a regression problem, the loss function used is the Root Mean Squared error (RMSE).

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

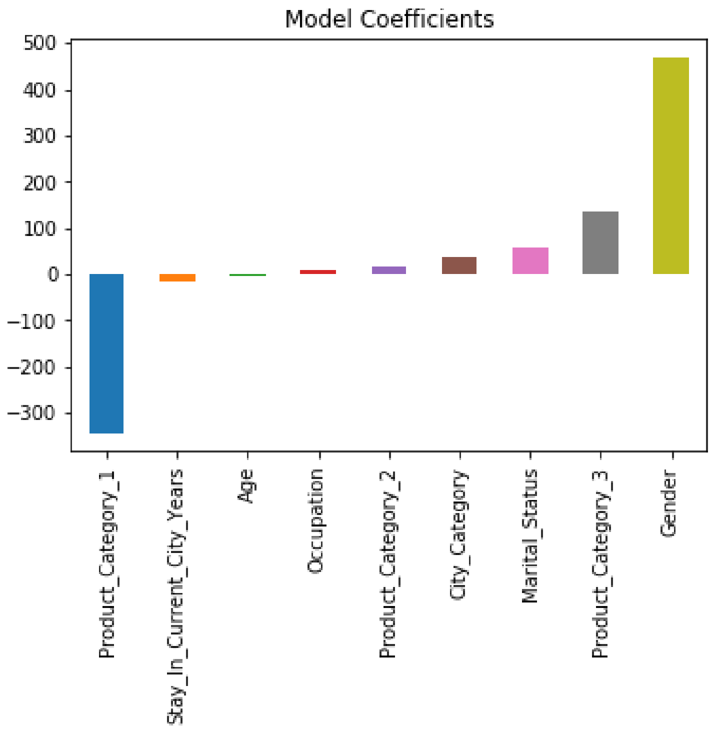
1. Linear Regression

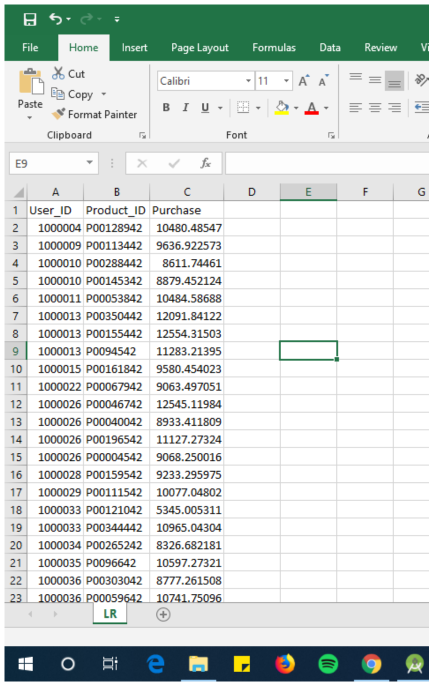
The linear regression using python's skLearn library was implemented on the transformed dataset. This was the simplest of the implementations in terms of complexity of the model.

Model Report

RMSE : 4632

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





	User_ID	Product_ID	Purchase
2	1000004	P00128942	10480.48547
3	1000009	P00113442	9636.922573
4	1000010	P00288442	8611.74461
5	1000010	P00145342	8879.452124
6	1000011	P00053842	10484.58688
7	1000013	P00350442	12091.84122
8	1000013	P00155442	12554.31503
9	1000013	P0094542	11283.21395
10	1000015	P00161842	9580.454023
11	1000022	P00067942	9063.497051
12	1000026	P00046742	12545.11984
13	1000026	P00040042	8933.411809
14	1000026	P00196542	11127.27324
15	1000026	P00004542	9068.250016
16	1000028	P00159542	9233.295975
17	1000029	P00111542	10077.04802
18	1000033	P00121042	5345.005311
19	1000033	P00344442	10965.04304
20	1000034	P00265242	8326.682181
21	1000035	P0096642	10597.27321
22	1000036	P00303042	8777.261508
23	1000036	P00059642	10741.75096

Predicted values in LR.csv

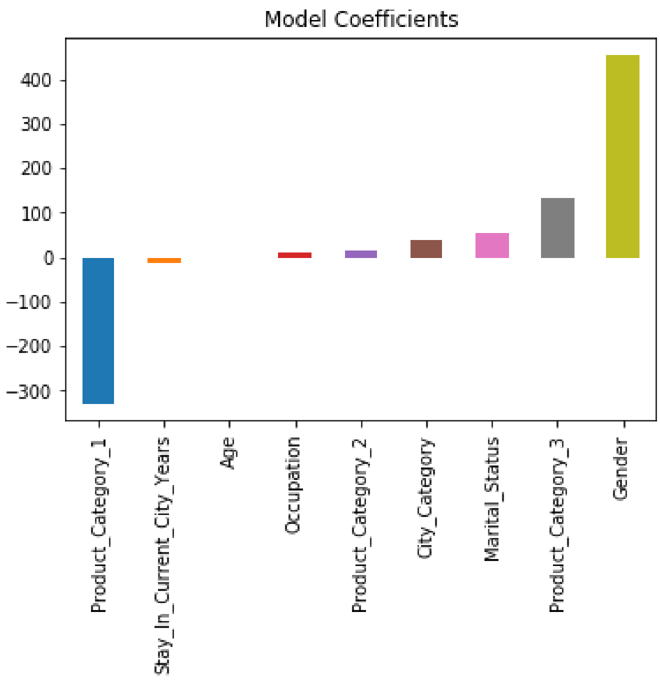
2. Ridge regression

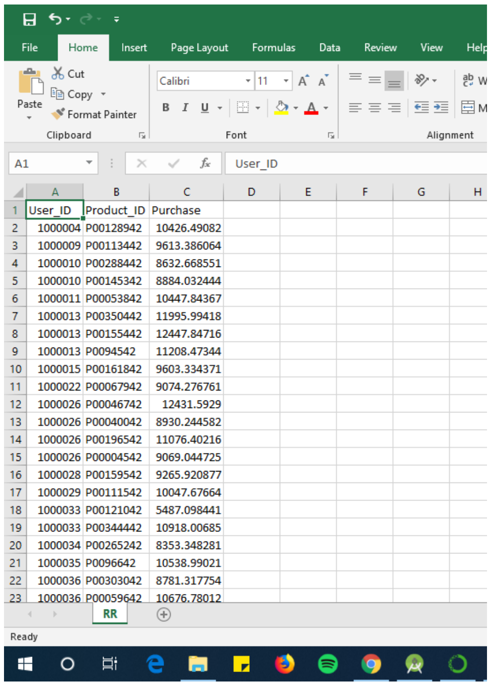
The ridge regression using python's skLearn library was implemented on the transformed dataset.

Model Report

RMSE : 4633

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





User_ID	Product_ID	Purchase
1000004	P00128942	10426.49082
1000009	P00113442	9613.386064
1000010	P00288442	8632.668551
1000010	P00145342	8884.032444
1000011	P00053842	10447.84367
1000013	P00350442	11995.99418
1000013	P00155442	12447.84716
1000013	P0094542	11208.47344
1000015	P00161842	9603.334371
1000022	P00067942	9074.276761
1000026	P00046742	12431.5929
1000026	P00040042	8930.244582
1000026	P00196542	11076.40216
1000026	P00004542	9069.044725
1000028	P00159542	9265.920877
1000029	P00111542	10047.67664
1000033	P00121042	5487.098441
1000033	P00344442	10918.00685
1000034	P00265242	8353.348281
1000035	P0096642	10538.99021
1000036	P00303042	8781.317754
1000036	P00059642	10676.78012

Predicted values in RR.csv

3. Decision Tree Regression

Machine learning algorithms like decision tree and 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 and statistical model, and 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 and 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)

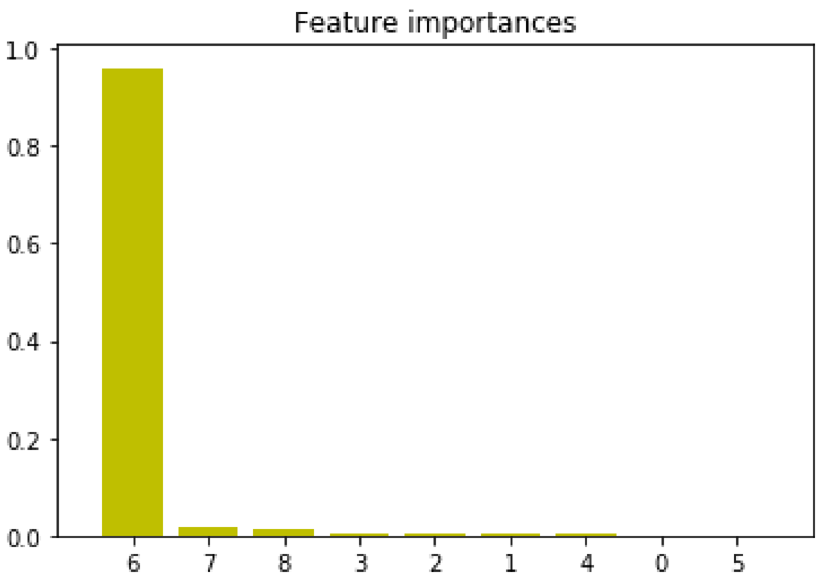
x5. feature 2 (0.003268)

x6. feature 1 (0.002386)

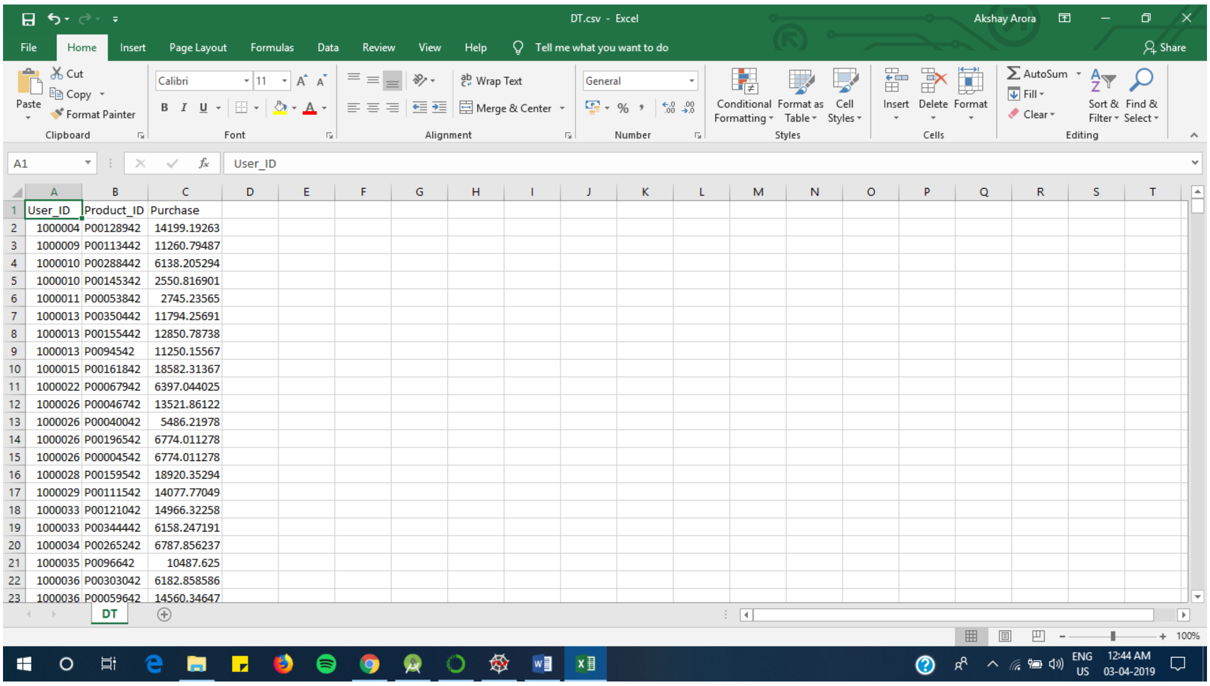
x7. feature 4 (0.001721)

x8. feature 0 (0.000868)

x9. feature 5 (0.000815)



Predicted values in DT.csv



User_ID	Product_ID	Purchase
1000004	P00128942	14199.19263
1000009	P00113442	11260.79487
1000010	P00288442	6138.205294
1000010	P00145342	2550.816901
1000011	P00053842	2745.23565
1000013	P00350442	11794.25691
1000013	P00155442	12850.78738
1000013	P0094542	11250.15567
1000015	P00161842	18582.31367
1000022	P00067942	6397.044025
1000026	P00046742	13521.86122
1000026	P00040042	5486.21978
1000026	P00196542	6774.011278
1000026	P00004542	6774.011278
1000028	P00159542	18920.35294
1000029	P00111542	14077.77049
1000033	P00121042	14966.32258
1000033	P00344442	6158.247191
1000034	P00265242	6787.856237
1000035	P0096642	10487.625
1000036	P00303042	6182.858586
1000036	P00059642	14560.34647

4. XGB Regression

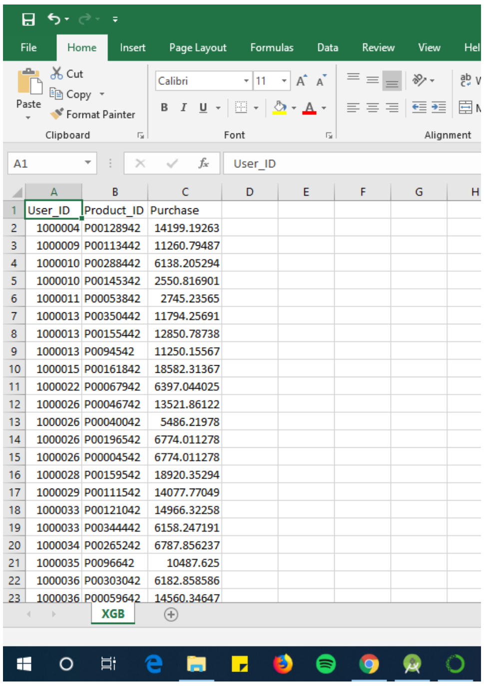
The XGBoost model internally implements the stepwise, ridge regression which dynamically selects the features and removes the multi-collinearity with the features. This implementation gave the bet results of this dataset. It uses ensemble model to learn from the weak predictors and eliminate the less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

Mean Absolute Error : 392.22502938349544

RMSE : 2950

Feature order:

1. feature 6 (0.886407)
2. feature 3 (0.032552)
3. feature 8 (0.028489)
4. feature 7 (0.025543)
5. feature 2 (0.007178)
6. feature 1 (0.005710)
7. feature 4 (0.004938)
8. feature 0 (0.004783)
9. feature 5 (0.004399)



User_ID	Product_ID	Purchase
1000004	P00128942	14199.19263
1000009	P00113442	11260.79487
1000010	P00288442	6138.205294
1000010	P00145342	2550.816901
1000011	P00053842	2745.23565
1000013	P00350442	11794.25691
1000013	P00155442	12850.78738
1000013	P0094542	11250.15567
1000015	P00161842	18582.31367
1000022	P00067942	6397.044025
1000026	P00046742	13521.86122
1000026	P00040042	5486.21978
1000026	P00196542	6774.011278
1000026	P00004542	6774.011278
1000028	P00159542	18920.35294
1000029	P00111542	14077.77049
1000033	P00121042	14966.32258
1000033	P00344442	6158.247191
1000034	P00265242	6787.856237
1000035	P0096642	10487.625
1000036	P00303042	6182.858586
1000036	P00059642	14560.34647

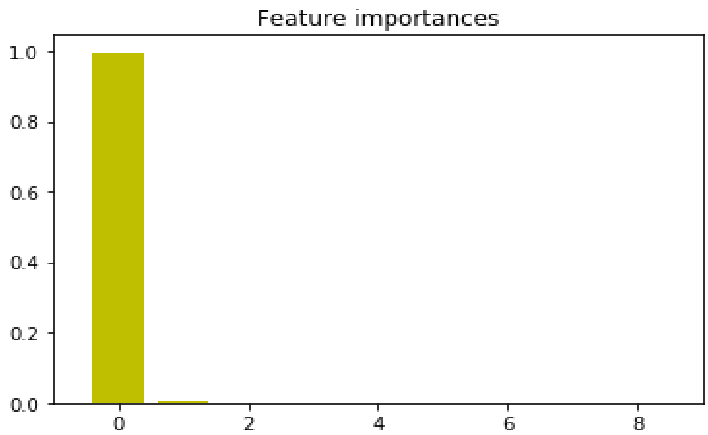
Predicted values in XGB.csv

5. Random forest regression

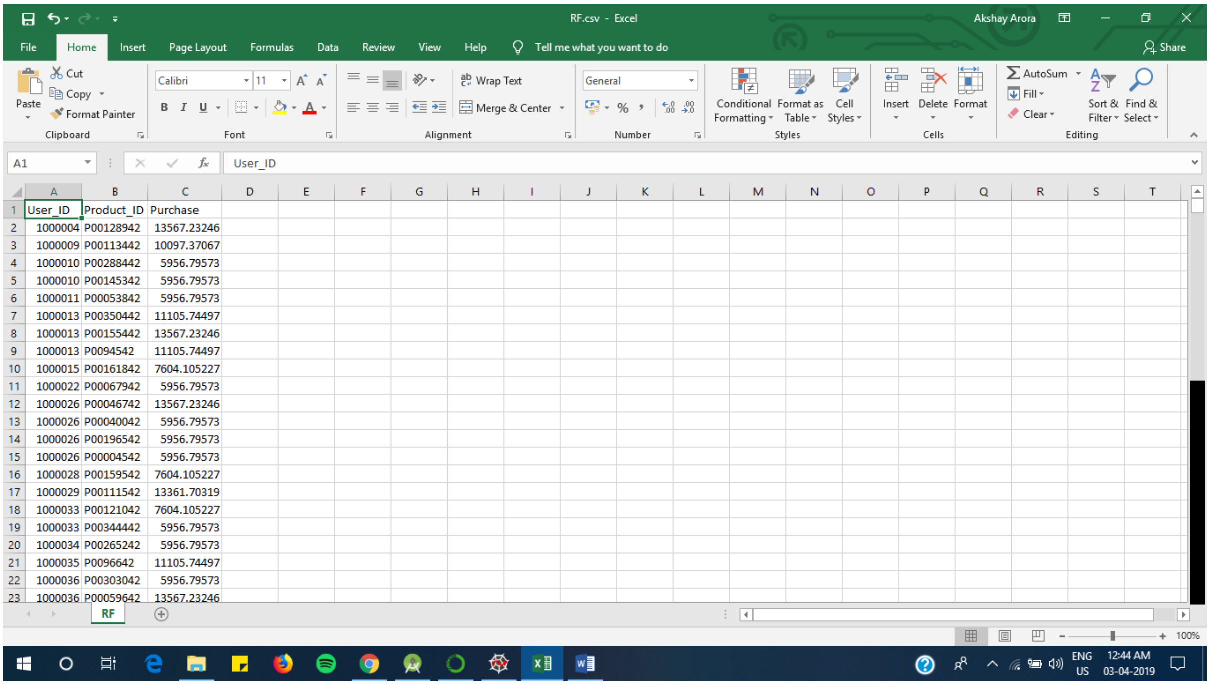
Model Report
RMSE : 3754
CV Score : Mean - 3714 | Std - 22.85 | Min - 3672 | Max - 3750
Mean Absolute Error : 3.7333049827565437
RMSE : 3754

Feature order:

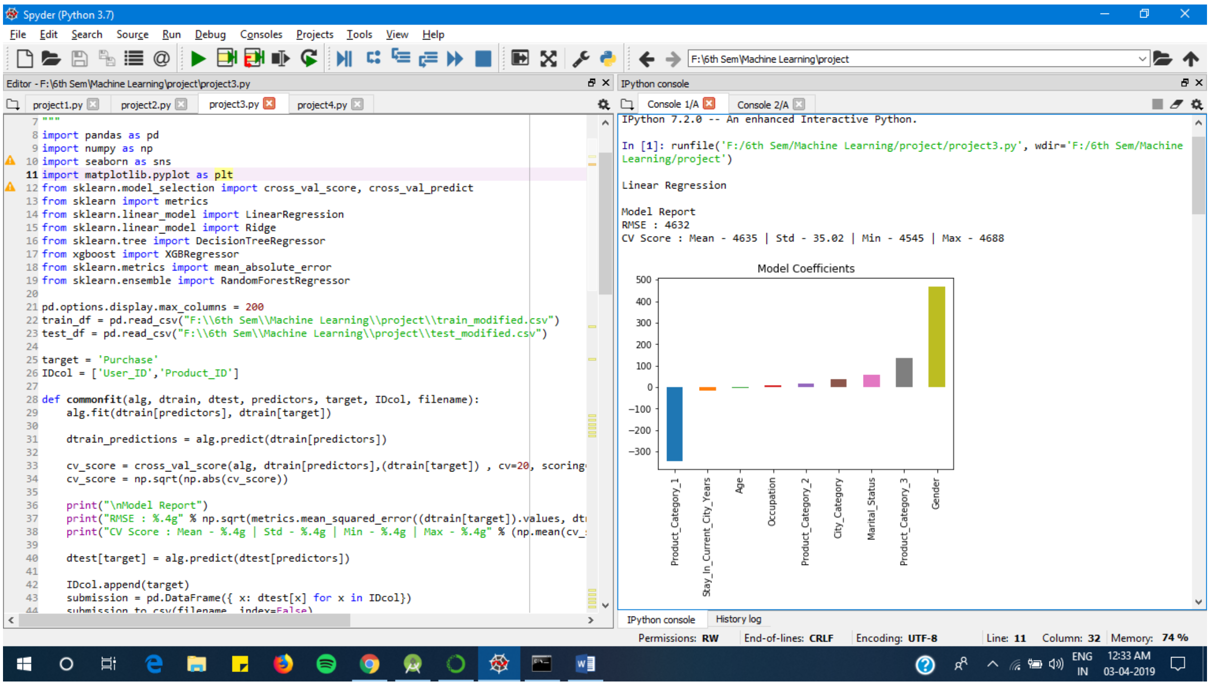
1. feature 6 (0.996204)
2. feature 8 (0.001988)
3. feature 7 (0.001205)
4. feature 3 (0.000603)
5. feature 5 (0.000000)
6. feature 4 (0.000000)
7. feature 2 (0.000000)
8. feature 1 (0.000000)
9. feature 0 (0.000000)

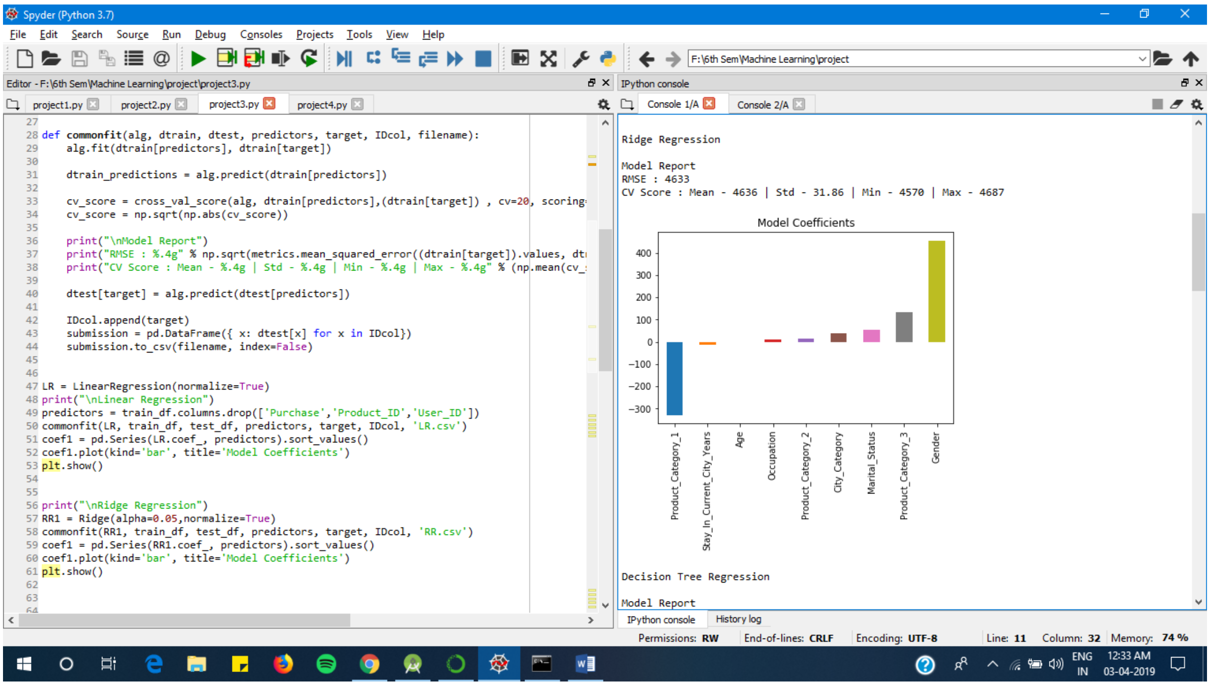


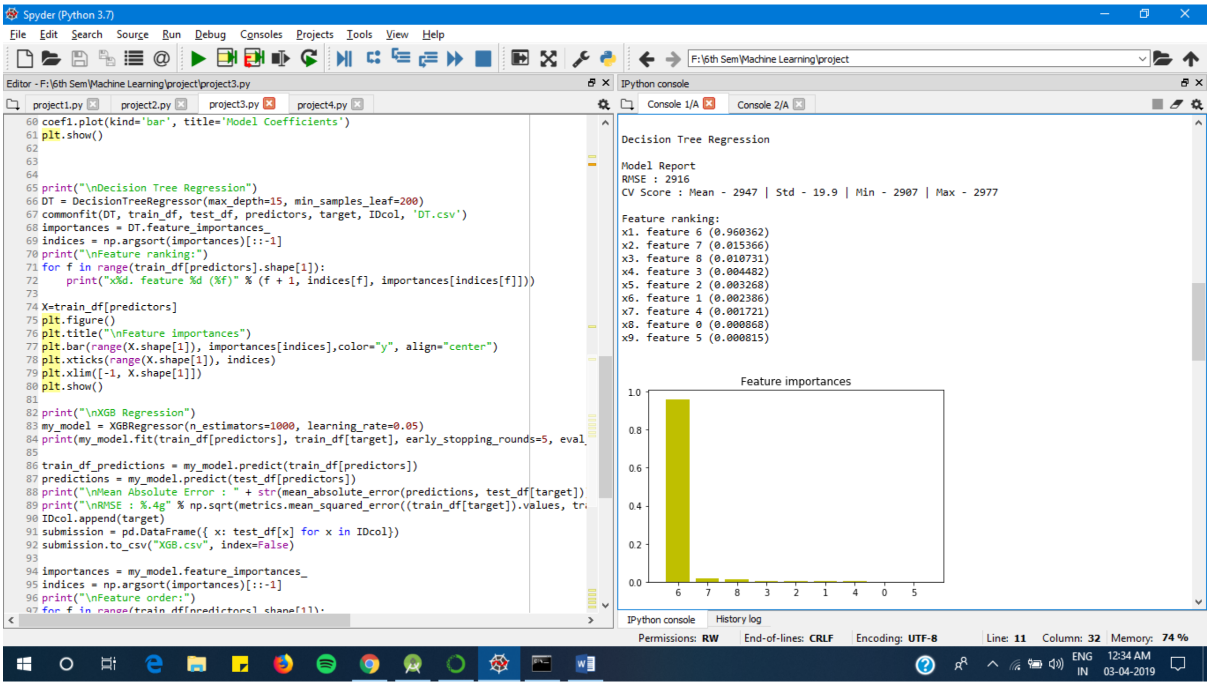
Predicted values in RF.csv

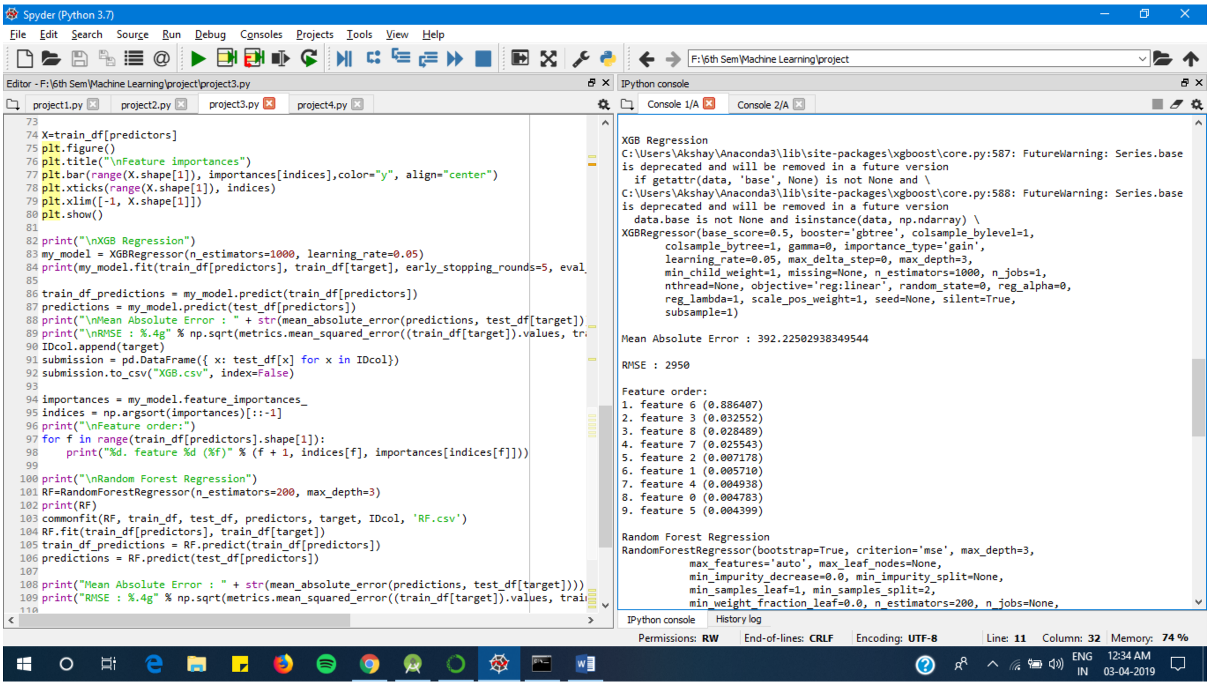


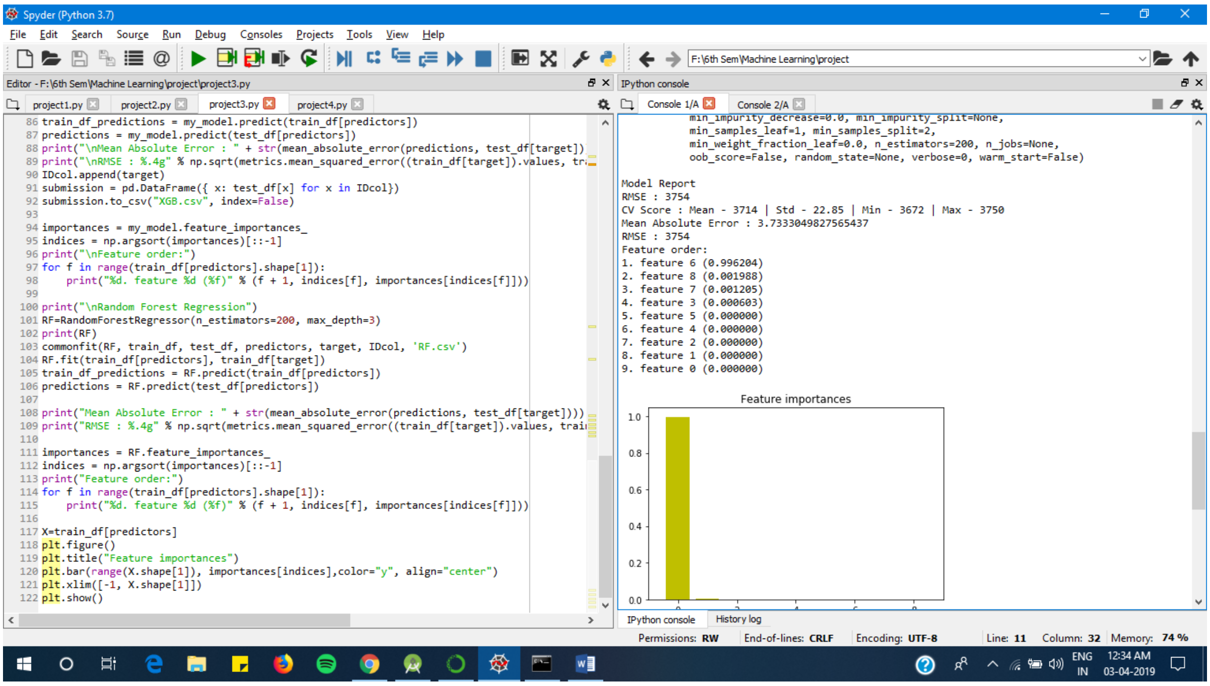
OUTPUTS-




Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\6th Sem\Machine Learning\project
Editor - F:\6th Sem\Machine Learning\project\project3.py
project1.py project2.py project3.py project4.py
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
    cv_score = np.sqrt(np.abs(cv_score))
    print("\nModel Report")
    print("RMSE : %.4g" % np.sqrt(metrics.mean_squared_error((dtrain[target]).values, dt
    print("CV Score : Mean - %.4g | Std - %.4g | Min - %.4g | Max - %.4g" % (np.mean(cv_
    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()
IPython console
Console 1/A Console 2/A
Ridge Regression
Model Report
RMSE : 4633
CV Score : Mean - 4636 | Std - 31.86 | Min - 4570 | Max - 4687
Model Coefficients
400
300
200
100
0
−100
−200
−300
Product_Category_1
Stay_In_Current_City_Years
Age
Occupation
Product_Category_2
City_Category
Marital_Status
Product_Category_3
Gender
Decision Tree Regression
Model Report
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 11 Column: 32 Memory: 74 %
ENG IN 12:33 AM 03-04-2019


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\6th Sem\Machine Learning\project
Editor - F:\6th Sem\Machine Learning\project\project3.py
project1.py project2.py project3.py project4.py
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_
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, tr
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:")
IPython console
Console 1/A Console 2/A
Decision Tree Regression
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)
Feature importances
1.0
0.8
0.6
0.4
0.2
0.0
6 7 8 3 2 1 4 0 5
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 11 Column: 32 Memory: 74 %
ENG IN 12:34 AM 03-04-2019


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\6th Sem\Machine Learning\project
Editor - F:\6th Sem\Machine Learning\project\project3.py
project1.py project2.py project3.py project4.py
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_
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, tr
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, trai
IPython console
Console 1/A Console 2/A
XGB Regression
C:\Users\Akshay\Anaconda3\lib\site-packages\xgboost\core.py:587: FutureWarning: Series.base is deprecated and will be removed in a future version
if getattr(data, 'base', None) is not None and \
C:\Users\Akshay\Anaconda3\lib\site-packages\xgboost\core.py:588: FutureWarning: Series.base is deprecated and will be removed in a future version
data.base is not None and isinstance(data, np.ndarray) \
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bytree=1, gamma=0, importance_type='gain',
learning_rate=0.05, max_delta_step=0, max_depth=3,
min_child_weight=1, missing=None, n_estimators=1000, n_jobs=1,
nthread=None, objective='reg:linear', random_state=0, reg_alpha=0,
reg_lambda=1, scale_pos_weight=1, seed=None, silent=True,
subsample=1)
Mean Absolute Error : 392.22502938349544
RMSE : 2950
Feature order:
1. feature 6 (0.886407)
2. feature 3 (0.032552)
3. feature 8 (0.028489)
4. feature 7 (0.025543)
5. feature 2 (0.007178)
6. feature 1 (0.005710)
7. feature 4 (0.004938)
8. feature 0 (0.004783)
9. feature 5 (0.004399)
Random Forest Regression
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=3,
max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=None,
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 11 Column: 32 Memory: 74 %
ENG IN 12:34 AM 03-04-2019


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\6th Sem\Machine Learning\project
Editor - F:\6th Sem\Machine Learning\project\project3.py
project1.py project2.py project3.py project4.py
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, tr
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, trai
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()
IPython console
Console 1/A Console 2/A
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=None,
oob_score=False, random_state=None, verbose=0, warm_start=False)
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)
Feature importances
1.0
0.8
0.6
0.4
0.2
0.0
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 11 Column: 32 Memory: 74 %
ENG IN 12:34 AM 03-04-2019

6. Rule Based Learning

Product_Category_1 <= 2.5, Product_Category_1 <= 1.5, Occupation <= 18.5, Product_Category_2 <= 16.5, Product_Category_3 <= 16.5, Product_Category_3 <= 8.5, Occupation <= 6.5, Occupation <= 3.5, Occupation <= 2.5, City_Category <= 0.5, Product_Category_3 <= 6.5, Occupation <= 0.5, Product_Category_2 <= 4.0

Before Rule Based:

Model Report

RMSE : 2996

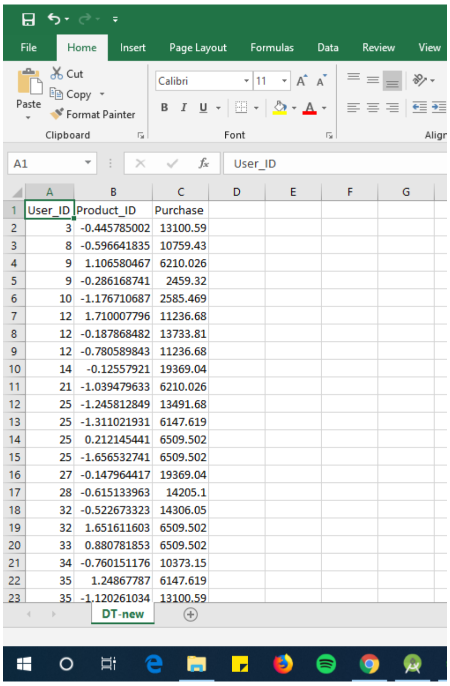
CV Score : Mean - 3242 | Std - 54.63 | Min - 3031 | Max - 3289

After Rule Based:

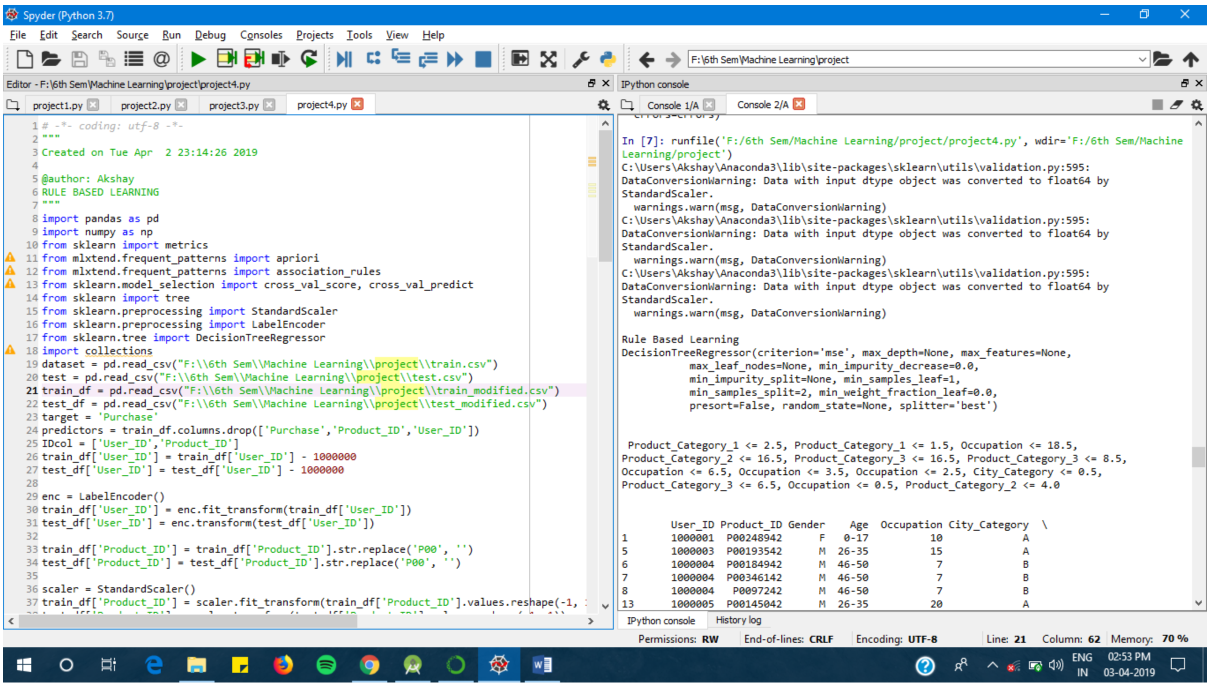
Model Report

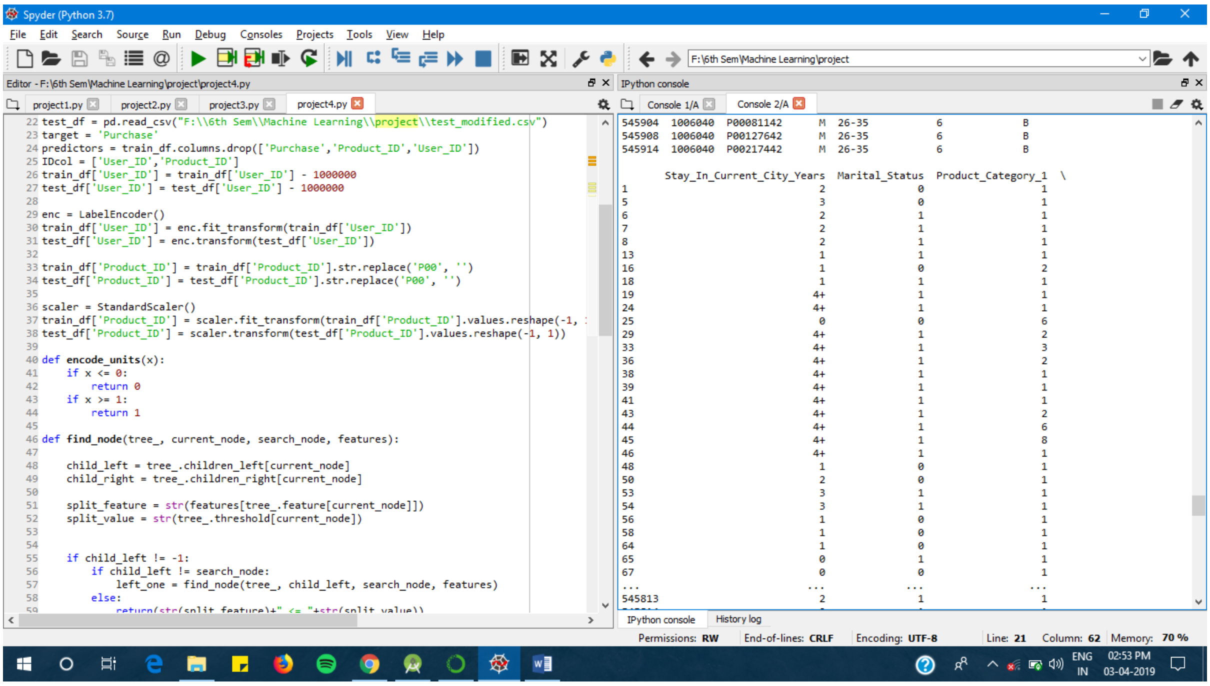
RMSE : 2291

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

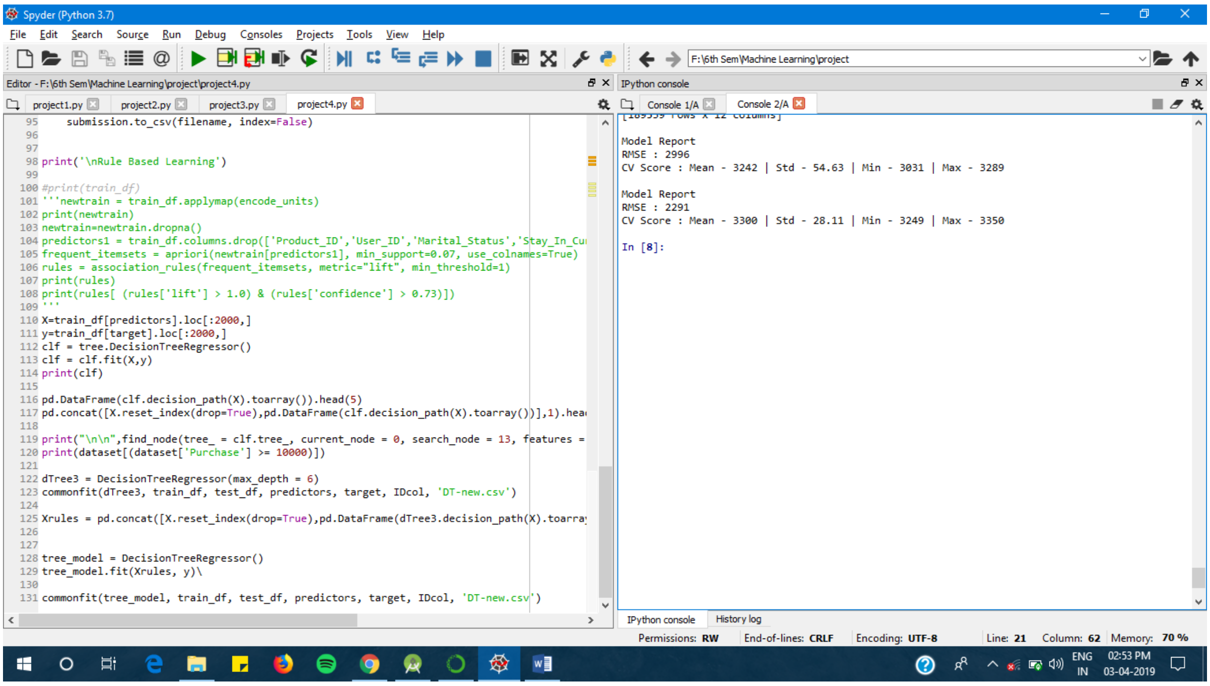


Predicted values in DT-new.csv





Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\6th Sem\Machine Learning\project
Editor - F:\6th Sem\Machine Learning\project\project4.py
project1.py project2.py project3.py project4.py
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,
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:
IPython console
Console 1/A Console 2/A
545904 1006040 P00081142 M 26-35 6 B
545908 1006040 P00127642 M 26-35 6 B
545914 1006040 P00217442 M 26-35 6 B
Stay_In_Current_City_Years Marital_Status Product_Category_1 \
1 2 0 1
5 3 0 1
6 2 1 1
7 2 1 1
8 2 1 1
13 1 1 1
16 1 0 2
18 1 1 1
19 4+ 1 1
24 4+ 1 1
25 0 0 6
29 4+ 1 2
33 4+ 1 3
36 4+ 1 2
38 4+ 1 1
39 4+ 1 1
41 4+ 1 1
43 4+ 1 2
44 4+ 1 6
45 4+ 1 8
46 4+ 1 1
48 1 0 1
50 2 0 1
53 3 1 1
54 3 1 1
56 1 0 1
58 1 0 1
64 1 0 1
65 0 1 1
67 0 0 1
... ...
545813 2 1 1
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 21 Column: 62 Memory: 70 %
ENG IN 02:53 PM 03-04-2019



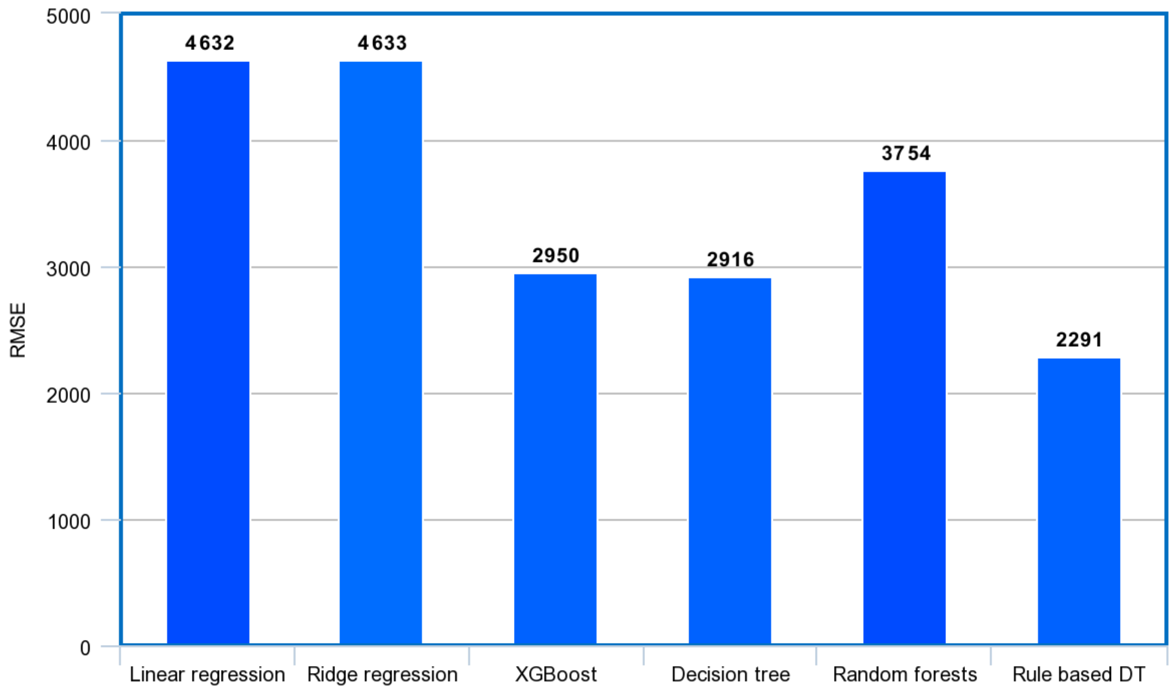
Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\6th Sem\Machine Learning\project
Editor - F:\6th Sem\Machine Learning\project\project4.py
project1.py project2.py project3.py project4.py
submission.to_csv(filename, index=False)
print('\nRule Based Learning')
#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_Cu
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).hea
print("\n\n",find_node(tree_ = clf.tree_, current_node = 0, search_node = 13, features =
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).toarra
tree_model = DecisionTreeRegressor()
tree_model.fit(Xrules, y)\
commonfit(tree_model, train_df, test_df, predictors, target, IDcol, 'DT-new.csv')
IPython console
Console 1/A Console 2/A
Model Report
RMSE : 2996
CV Score : Mean - 3242 | Std - 54.63 | Min - 3031 | Max - 3289
Model Report
RMSE : 2291
CV Score : Mean - 3300 | Std - 28.11 | Min - 3249 | Max - 3350
In [8]:
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 21 Column: 62 Memory: 70 %
ENG IN 02:53 PM 03-04-2019

RESULT

The below figure depicts the plot of RMSE for all the above implementation for visual comparison.

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



CONCLUSION

We conclude that the complex models like neural network are an overkill for simple problems like regression. And simpler models along with proper data cleaning perform well for the regression.

Also, based on the current trend, the number of shoppers on the Black Friday is only going to increase. We conclude that machine learning techniques produce better prediction models that can be used at stores and the store owners can analyse their customer base to better target the customers and increase the sales on a Black Friday.

It also shows that the data must be pre-processed to attain an effective dataset for developing the prediction model. Several techniques were used to attain the best model. However, there is still no definite solution as to what the correct technique is to attain a model with high accuracy. Rule based decision tree can be used though.

To improve the results, a dataset with sufficient features and increase in quantity must be obtained. Further research must be conducted in enhancing the existing machine learning techniques to work in real time and develop an efficient model. Also, the models developed must be tested on data.

REFERENCES

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

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

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

[4] Z. X. Guo, W. K. Wong , and 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, and 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 and 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 and H. Li, "Sales forecast based on BP neural network", 2011 IEEE 3rd International Conference on Communication Software and Network., pp. 186-189, May 2011

[8] K. Singh and R. Wajgi, "Data analysis and visualization of sales data," 2016 World Conference on Futuristic Trends in Research and 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

@author: Akshay
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

@author: Akshay
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

@author: Akshay
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

@author: Akshay
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'].valu
es.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')
```