

**ANL252 (Online)**

**Python for Data Analytics**

# **End-of-Course Assessment**

**July 2022 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Tan Wenying Audrey** | **K2170805** |

**Tutorial Group: ­­­­­­­­­­­­ T 09**

**Instructor’s Name: Dr Munish Kumar**

**Submission Date: 05/09/2022**

**Section A**

**Question 1**

The categorical variables are ID, RATING, GENDER, EDUCATION, MARITAL, and S(n). RATING, and GENDER are binary attributes which is a nominal attribute. Although ID, EDUCATION, MARITAL, and S(n) are numerals, they are not supposed to be used quantitatively but for identification purposes.

The numerical variables are LIMIT, BALANCE, INCOME, AGE, B(n), and R(n).

LIMIT, BALANCE, INCOME, AGE and R(n) are ratio-scaled attributes while B(n) is an interval-scaled attribute.

**Question 2**

**Data pre-processing tasks**

**Task 1**

The first data pre-processing task was to find the null value and either replace or drop them. It was observed that Education and Marital columns has a total of 13 and 38 nulls respectively. The percentage missing out of each column was at 7% for Education and rather significantly at 20% for Marital. Therefore, the decision to replace the null was made. Next, the skewness of the two columns were done. Marital was negatively skewed while Education was close to normal distribution. Next, df.unique() was used to check what were the unique values in these columns. It was observed that 0,1,2,3 and nan were present in Education while 0,1,2, and nan were present in Marital. Descriptive names in Appendix 1 were used to replace these values. The data for the columns were nominal data and the mode can be found in these data. Also, the data in the columns were skewed. Thus, the mode was used to replace the null data in both columns. The mode of Marital was ‘Married’ , and this was used to replace the null values in the column of Marital. The mode of Education was ‘Tertiary’, and this was used to replace the null values in the column of Education.

**Figure 1**

*Education and Marital columns count of null values*

Text

Description automatically generated

**Task 2**

The second task was to conduct a reasonability test on each column. df.describe() was used to check the data. It was observed the Age had a minimum of -1 and max of 199 which was out of the norm. A check was done on the skewness of Age and it was observed that it was positively skewed. The values of -1 and 199 were replaced by the median as the median is not affected by extremely large or small values.

**Figure 2**

*Age shows extreme minimum and maximum values of -1 and 199*

Table

Description automatically generated

**Task 3**

The third task was to check if the columns contained special characters. Only column R3 had special characters detected after running a regex search iteration through the dataframe. A replacement for the special character was done with ‘ ’. A check was then done to ensure that it was changed.

**Figure 3**

*Results of search of special character in each column. R3 has special characters.*

Text

Description automatically generated

**Task 4**

The fourth task was to perform data reduction. Before this, R3 was earlier detected to be an object type, it was changed to integer type. Also, since BALANCE was earlier detected to be a float type, it was changed to integer type. Then two dictionaries with the description in Appendix 1 were created to replace the values in S1 to S5. Delays of more than 1 month were all renamed under ‘Delays\_more\_than\_1\_month’. This was to reduce the different categories used in customer repayment status.

**Figure 4**

*Value in S1 to S5 were changed from numerical to text.*

**A picture containing table

Description automatically generated**

**Question 3**

**Insight 1 on the gender and age of customers**

Both the minimum age for both genders start at the same age of around 21. This would be reasonable as the age of majority is set at 21 in Singapore. The maximum age for males is slightly above 60 while it stays at 60 for females. The median age for male and female customers are around the same at early to mid 30s. This is represented by the horizontal line in the middle of the box. The mean age is around 35 for both male and females. This is represented by the “X” in the boxplot. 50% of female customers and 50% of male customers are in the ages of late 20s to early 40s. From the outlier in the male boxplot, the oldest male customer is around the age of 80. From the outlier in the female boxplot, the oldest female customer is around the age of 74. The chart shows that their customers have age ranges scattered from early 20s to 80s.

**Figure 5**

*Boxplot of age for male and female*

Chart, box and whisker chart

Description automatically generated

**Insight 2 on the relationship between income and limit with repayment status information**

There is a strong positive correlation between income and limit. As the income increase the limit for the customer increase as well. The credit facility would want to be confident that the customers have the ability to repay their debts and used income as a determinant of credit limit granted. It is observed from the scatterplot that there are mostly minimum repayments followed by prompt repayments and finally delays of more than a month. The customers with minimum repayments, prompt repayments and delays of more than a month were scattered across all income levels. However, the top income earners of more than $800,000 do not have repayment delays of more than 1 month. These top earners were mostly prompt or had minimum sum payments.

**Figure 6**

*Scatterplot of income against limit with repayment status as hue*

Chart, scatter chart

Description automatically generated

**Insight 3 on the relationship between billable amount in the most recent month and the credit balance of customers with repayment status information**

There is a strong positive correlation between billable amount in the most recent month and the credit balance of customers. Generally, as the billable amount in the most recent month increases, the credit balance of the customer will increase. Most prompt repayments had billable amounts of less than $300,000 and credit balances of less than $60,000. Most minimum repayments were clustered at billable amounts of less than $300,000 and credit balances of less than $60,000. Most delays of more than a month were along the 45 degrees line of the chart but stop at billable amounts of less than $600,000 and credit balance of less than $110,000.

**Figure 7**

*Scatterplot of billable amount against credit balance with repayment status as hue*

Chart, scatter chart

Description automatically generated

**Insight 4 on the relationship between billable amount in the most recent month and the credit balance of customers with rating information**

It is observed that there were significantly more customers with bad ratings than good ratings. Most bad ratings were of prompt and minimum repayments and had billable amount in recent month of less than $300,000 and credit balances of less than $60,000. The bad ratings in orange were mostly on the line of best fit. Even though the good ratings in blue were also mostly on the line of best fit, the rest of the points were more dispersed.

**Figure 8**

*Scatterplot and lineplot of billable amount against credit balance with rating as hue*

Chart, scatter chart

Description automatically generated

**Insight 5 on education level and gender of customers**

Most customers had tertiary level education, followed by postgraduate, then high school and finally others. There were overall more female than male customers in each education level as seen by the larger area of pink (Female) than blue (Male) in each stacked bar chart.

**Figure 9**

*Stacked bar chart of count of male and female customer’s education level.*

Chart, box and whisker chart

Description automatically generated

**Question 4**

**Getting dummy variables**

Columns of Education, Marital, and S1 to S5, were not numerical data. They were object type and contained text. Further data pre-processing was first done on these columns. This was through getting dummy variables on these columns through pandas get\_dummies function. These new columns with dummy variables were then joined to the original dataframe while the original columns were then dropped.

**Figure 10**

*Dummy data.*

Table

Description automatically generated

**Feature selection**

Next, two methods of feature selection were done so that comparison can be made and one that yielded the better results would be used. The first was by using the filter method. A heatmap was created to show the correlation coefficient of the relationship between the different features. The correlation coefficient shows the strength of the linear relationship between the two features. Next, the relevant features with correlation coefficient of more than 0.5 were selected. BALANCE, B1, B2, B3, B4, and B5 were features of interest.

**Figure 11**

*Heatmap was created and only features with correlations coefficients of more than 0.50 were selected.*

**A picture containing text

Description automatically generated**

Feature selection with the embedded method was then used. Regularisation to penalise the extra features in the model to keep the feature of interest was done. The LassoCV() function was used to fit the lasso regression model. Lasso decreased the coefficient of the features that are not important. Lasso then selected 8 features and eliminated the other 22 features. The selected features were R1, R4, LIMIT, B5, R3, R2, B2, and BALANCE. BALANCE, B2 and B5 appeared in both types of features selection.

**Figure 12**

*Selected important features using Lasso Model*

Graphical user interface, application, Teams

Description automatically generated

**Linear regression**

Linear regression was done on features from the filter method and the embedded method. The data was first split into training and testing data. Test size was set at 30%. Then an object of linear regression class was created before fitting the training data of the dependent and independent variables. A prediction of the test set on the independent variables was done. A dataframe and a plotted bar chart were created to show both the actual and predicted values.

|  |  |
| --- | --- |
| **Figure 13**  *Actual vs predicted value using the features from filter method*  Table  Description automatically generated | **Figure 14**  *Actual vs predicted value using the features from embedded method* |

Model evaluation was done and a higher R square and lower mean absolute error (MAE), means square error (MSE) and root mean square error (RMSE) would be preferable. In this case, the embedded method yielded better results for R square and MAE. The ideal state would be to have the predicted values as close to the actual values as possible. It was observed in the plotted chart using the features in the embedded model, that there was a high R square of 94.62% and a very strong positive linear relationship between the actual and predicted dependent variable. On average, the model prediction will be off by approximately $5,980.50 as derived from the MAE. Being an absolute value, the MSE and RMSE are not used to compare across different models.

|  |  |
| --- | --- |
| **Figure 15**  *Model evaluation using the features from filter method* | **Figure 16**  *Model evaluation using the features from embedded method* |

**Figure 17**

*Actual vs predicted dependent variable*

**Graphical user interface, chart, scatter chart

Description automatically generated**

**Question 5**

**Multiple linear regression equation and explanation**

The multiple linear regression equation is as such:

= 667.33+ 2.5530- 0.4283- 0.0214+ 0.0023+ 0.0070+ 0.0713+ 0.1678+ 0.5572

Where:

= Predicted customer billable amount in the most recent month

= BALANCE/ Customer current credit balance (snapshot in time)

= R1/ Customer previous repayment amount, paid in the most recent month

= R4/ Customer previous repayment amount, paid in the previous 3rd month

= LIMIT/ Customer total limit

= B5/ Customer billable amount in the previous 4th month

= R3/ Customer previous repayment amount, paid in the previous 2nd month

= R2/ Customer previous repayment amount, paid in the previous 1st month

= B2/ Customer billable amount in the previous 1st month

: Each $1 increase in the customer’s current credit balance results in an average increase in customer billable amount in the most recent month by $2.55, assuming no changes in the other variables.

Each $1 increase in the customer previous repayment amount, paid in the most recent month results in an average decrease in customer billable amount in the most recent month by $0.43, assuming no changes in the other variables.

Each $1 increase in the customer previous repayment amount, paid in the previous 3rd month results in an average decrease in customer billable amount in the most recent month by $0.02, assuming no changes in the other variables.

Each $10,000 increase in the customer total limit results in an average increase in customer billable amount in the most recent month by $23, assuming no changes in the other variables.

Each $1 increase in the customer billable amount in the previous 4th month results in an average increase in customer billable amount in the most recent month by $0.0070, assuming no changes in the other variables.

Each $1 increase in the customer previous repayment amount, paid in the previous 2nd month results in an average increase in customer billable amount in the most recent month by $0.07 assuming no changes in the other variables.

Each $1 increase in the customer previous repayment amount, paid in the previous 1st month results in an average increase in customer billable amount in the most recent month by $0.17 assuming no changes in the other variables.

Each $1 increase in the customer billable amount in the previous 1st month results in an average increase in customer billable amount in the most recent month by $0.56 assuming no changes in the other variables.

Figure 18 and Figure 19 below were checked against each other to ensure that the intercept and the coefficients were similar.

**Figure 18**

*Intercept and coefficients*

Table

Description automatically generated

**Figure 19**

*OLS Regression Results*

A screenshot of a computer

Description automatically generated with medium confidence

**Key insights from results obtained in Question 4**

From the results obtained in Question 4, the embedded method had better results as it had higher R square and lower MAE than the filter method. For the chosen embedded method, the values of the actual and predicted dependent variables were close to each other as seen in Figure 14 and Figure 20. There are a few exceptions where the predicted value may have been much higher than the actual value as seen in Figure 20. However, the Pearson’s r is still high at 94.62%. This implies that the actual and predicted values of the dependent variables are strongly correlated. This is seen by the very strong positive linear relationship between the actual and predicted dependent variable in Figure 17. The predicted dependent variable will increase as the actual dependent variable increase. On average, the model prediction will be off by approximately $5,980.50 as derived from the MAE. The MSE is the average sum of squares for all points in the linear regression chart. The RMSE is the square root of the MSE. The MSE and RMSE at 298,698,871.75 and 17,282.91 were very high as the error values were not using the same scale as the target.

**Figure 20**

Chart, bar chart

Description automatically generated

**Section B**

**Question 6**

**Organisation of code**

**Python codes (in text) for Question 2:**

**Libraries**

#import pandas

import pandas as pd

#import matplotlib

import matplotlib

import matplotlib.pyplot as plt

import matplotlib.patches as mpatches

#import seaborn

import seaborn as sns

#Import numpy

import numpy as np

#Import sklearn and modules

import sklearn

from sklearn import metrics

from sklearn import linear\_model

from sklearn import preprocessing

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import RidgeCV, LassoCV, Ridge, Lasso

from sklearn.model\_selection import train\_test\_split

#Import statsmodel

import statsmodels.api as sm

**Import Excel**

# reading data frame from the csv file and replacing symbols as null values

df = pd.read\_csv('ECA\_Data.csv')

**Check basic information**

df.info()

df.describe()

**Data Pre-Processing**

**Task 1: Find Null values and replace or drop them**

#get sum of null for columns with at least one null value

df[df.columns[df.isnull().any()]].isnull().sum()

#Find out the percentage of missing values in each column in df

percent\_missing = df[df.columns[df.isnull().any()]].isnull().sum() \* 100 / len(df)

df1 = pd.DataFrame({'percent\_missing': percent\_missing})

display(df1)

#check the skewness of the columns. Use median instead of mean to replace null if it is skewed.

df.skew()[["EDUCATION", "MARITAL"]]

#check the unique values in each column

print(df.EDUCATION.unique())

print(df.MARITAL.unique())

#replace the values in EDUCATION and MARITAL with description data

df=df.replace({'EDUCATION':{0:'Others',1:'Postgraduate',2:'Tertiary',3:'High School'},

'MARITAL':{0:'Others',1:'Single',2:'Married'}})

display(df)

#find the mode for Marital

df['MARITAL'].mode()

#Replace null in Marital by the mode of Marital as the data is skewed

df['MARITAL'] = np.where((df['MARITAL'].isnull()), df['MARITAL'].mode(), df['MARITAL'])

#find the mode for education

df['EDUCATION'].mode()

#Replace null in Education by the mode of Education as the data is skewed

df['EDUCATION'] = np.where((df['EDUCATION'].isnull()), df['EDUCATION'].mode(), df['EDUCATION'])

#Check that there is no more null

df[df.columns[df.isnull().any()]].isnull().sum()

**Task 2: Conduct reasonability test on each column-Replace the outliers for ages**

df.describe()

#check the skewness of the columns. Use median instead of mean to replace null if it is skewed.

df.skew()[["AGE"]]

#Age should not show -1 or 199

#Replace these values with the median age

df['AGE'].replace(to\_replace=[-1, 199], value=df['AGE'].median(), inplace=True)

#Change data type for age to integer

df['AGE']=df['AGE'].astype(int)

#Check that it is changed

df.describe()

**Task 3: Find and replace special characters**

#check the columns with special characters

import re

my\_dict={}

#regex for all special characters

regex = re.compile('[^\w\s-]|\_')

#Iterate through df and find special character

for col in df.columns:

countx=len(df)

county=0

for i in df[col]:

#if no special characters are found

if(regex.search(str(i)) == None):

countx=countx-1

else:

county=county+1

if countx != 0 and county>0:

my\_dict.update({col:'FOUND SPECIAL CHAR'})

else:

my\_dict.update({col:'NO SPECIAL CHAR'})

my\_dict

#replace special characters except "-" with " ".

df.replace(to\_replace=r'[\W\-]', value=r'', inplace=True, regex=True)

#Check that all special characters were removed

import re

my\_dict={}

#regex for all special characters

regex = re.compile('[^\w\s-]|\_')

#Iterate through df and find special character

for col in df.columns:

countx=len(df)

county=0

for i in df[col]:

#if no special characters are found

if(regex.search(str(i)) == None):

countx=countx-1

else:

county=county+1

if countx != 0 and county>0:

my\_dict.update({col:'FOUND SPECIAL CHAR'})

else:

my\_dict.update({col:'NO SPECIAL CHAR'})

my\_dict

**Task 4: Data Reduction**

df['R3']=df['R3'].astype(int)

df['BALANCE']=df['BALANCE'].astype(int)

#create a dictionary to replace the values in S1,S3,S4 later

keys = [-1,0,1,2,3,4,5,6,7]

values = ['Prompt','Minimum','Delays\_more\_than\_1\_month','Delays\_more\_than\_1\_month',

'Delays\_more\_than\_1\_month','Delays\_more\_than\_1\_month','Delays\_more\_than\_1\_month',

'Delays\_more\_than\_1\_month','Delays\_more\_than\_1\_month']

dictionary = {}

for i in range(len(keys)):

dictionary[keys[i]] = values[i]

dictionary

#create a dictionary to replace the values in S2 later

keys1 = [-1,0,1,2,3,4,5,6,7,8]

values1 = ['Prompt','Minimum','Delays\_more\_than\_1\_month','Delays\_more\_than\_1\_month',

'Delays\_more\_than\_1\_month','Delays\_more\_than\_1\_month','Delays\_more\_than\_1\_month',

'Delays\_more\_than\_1\_month','Delays\_more\_than\_1\_month','Delays\_more\_than\_1\_month']

dictionary1 = {}

for i in range(len(keys1)):

dictionary1[keys1[i]] = values1[i]

#replace the values in S1 to S4 with description data

df=df.replace({'S1':dictionary,

'S2':dictionary1,

'S3':dictionary,

'S4':dictionary,

'S5':dictionary})

df.tail()

**Python codes (in text) for Question 3:**

**Data Visualisation**

#Extract all the rows with Gender = 1 or Female

contain\_values = df[df['GENDER']==1]

#For these rows with Gender =F, filter out the Salary

contain\_values = contain\_values.filter(['AGE'])

#Extract all the rows with Gender = 0 or Male

contain\_values2 = df[df['GENDER']==0]

#For these rows with Gender =M, filter out the Salary

contain\_values2 = contain\_values2.filter(['AGE'])

contain\_values['AGE']=contain\_values['AGE'].astype(int)

contain\_values2['AGE']=contain\_values2['AGE'].astype(int)

#Combining data of salaries from M and F into numpy array

data = np.array([contain\_values, contain\_values2], dtype=object)

#Set the size of the plot

fig = plt.figure(figsize =(6, 6))

#Creating axes instance

ax = fig.add\_axes([0, 0, 1, 1])

#Creating plot

bp = ax.boxplot(data)

#X and Y Labels

plt.xlabel('Gender')

plt.ylabel('Age')

#Adding title

plt.title("Age for Male and Female")

#Adding ticks

plt.xticks([1, 2],['Female', 'Male'] )

#Plot the boxplot

box = plt.boxplot(data, #array to be plotted

patch\_artist=True, #fill with color

flierprops={'markeredgecolor': 'None'}, #no marker edger for outliers

showmeans=True, #show the mean

meanprops={"marker":"x","markerfacecolor":"black", "markeredgecolor":"black"}) #set the marker type and colour of the mean

#Fill the outliers with two colours

cols = ['pink', 'skyblue']

for f, fc in zip(box['fliers'], cols):

f.set\_markerfacecolor(fc)

#Fill the boxplot with two colours

colors = ['pink', 'skyblue']

for patch, color in zip(box['boxes'], colors):

patch.set\_facecolor(color)

#set legend

pink\_patch=mpatches.Patch(color='pink',label="Female") #Set colour for Females in legend

blue\_patch=mpatches.Patch(color='skyblue',label="Male") #Set colour for Males in legend

plt.legend(loc='center right', bbox\_to\_anchor=(1.2,0.5), ncol=1, handles=[pink\_patch, blue\_patch],title='Gender')

#show plot

plt.show()

#Plot scatterplot with Seaborn

#To set the colours per Unit

colour\_dict1 = dict({'Prompt':'seagreen',

'Minimum':'pink',

'Delays\_more\_than\_1\_month':'cyan'})

#Set the order in the legend

#orders=['0','1']

#To set the side of the scatterplot

sns.set(rc={"figure.figsize":(8, 4)})

#To display the scatterplot

g =sns.scatterplot(x="INCOME", y="LIMIT",

hue="S1",

data=df,

palette=colour\_dict1)

#Rename axis

plt.xlabel('Income')

plt.ylabel('Limit')

#To start the x and y axis at point 0

plt.xlim(0)

plt.ylim(0)

#To set the title

plt.title('Scatterplot of Income against limit with repayment status as Hue')

#To set the legend location of scatterplot

g.legend(loc='center left', bbox\_to\_anchor=(1, 0.5), ncol=1,title='Repayment status')

#Plot scatterplot with Seaborn

#To set the colours per Unit

colour\_dict = dict({'Prompt':'seagreen',

'Minimum':'pink',

'Delays\_more\_than\_1\_month':'cyan'})

#Set the order in the legend

#orders=['0','1']

#To set the side of the scatterplot

sns.set(rc={"figure.figsize":(8, 4)})

#To display the scatterplot

g =sns.scatterplot(x="B1", y="BALANCE",

hue="S1",

data=df,

palette=colour\_dict)

#Rename axis

plt.xlabel('Billable amount in recent month')

plt.ylabel('Credit Balance')

#To start the x and y axis at point 0

plt.xlim(0)

plt.ylim(0)

#To set the title

plt.title('Scatterplot of billable amount against credit balance with repayment status as hue')

#To set the legend location of scatterplot

g.legend(loc='center left', bbox\_to\_anchor=(1, 0.5), ncol=1, title='Repayment status')

# Use seaborn to plot lineplot

h=sns.lmplot(x='B1',

y='BALANCE',

hue='RATING',

data=df,

legend=True,

palette="muted",

scatter\_kws={"s":10},

height=10)

labels=['Good', 'Bad']

for t, l in zip(h.\_legend.texts, labels):

t.set\_text(l)

#Rename axis

plt.xlabel('Billable amount in recent month')

plt.ylabel('Credit Balance')

plt.title("Scatterplot and lineplot of billable amount against credit balance with rating as hue")

plt.show(h)

#create new dataframe of only Education and Gender

df\_ed\_gen= df[['EDUCATION','GENDER']]

# Get the #'s data points in the groups

gdf = df\_ed\_gen.groupby(['EDUCATION', 'GENDER'])['EDUCATION'].count()

# Since we want to stack by gender, lets make them columns

gdf = gdf.unstack('GENDER')

# If you want to choose a subset of columns to plot

col\_to\_plot = gdf.columns.tolist()

#sort education by largest count

edu\_order=['Tertiary', 'Postgraduate', 'HighSchool', 'Others']

# Plot command

gdf[col\_to\_plot].loc[edu\_order].plot(kind='bar', stacked=True, color=['skyblue','pink'])

plt.legend(title='Gender',labels=['Male', 'Female'])

plt.title("Count of male and female customer's education level", y=1.02);

plt.ylabel("Count of customers", labelpad=14)

plt.xlabel("Education level", labelpad=14)

**Python codes (in text) for Question 4:**

**Linear Regression to predict B1**

**Getting dummy data**

df.dtypes

#convert education to dummy data

dummy\_ed= pd.get\_dummies(df['EDUCATION'],drop\_first=True,prefix="Ed")

df.drop(['EDUCATION'], axis=1, inplace=True)

df = df.join(dummy\_ed)

df.head()

#convert marital to dummy data

dummy\_ed1= pd.get\_dummies(df['MARITAL'],drop\_first=True,prefix="Mar")

df.drop(['MARITAL'], axis=1, inplace=True)

df = df.join(dummy\_ed1)

df.head()

#convert S1 to dummy data

dummy\_ed2= pd.get\_dummies(df['S1'],drop\_first=True,prefix="Repmt\_statusS1")

df.drop(['S1'], axis=1, inplace=True)

df = df.join(dummy\_ed2)

#convert S2 to dummy data

dummy\_ed3= pd.get\_dummies(df['S2'],drop\_first=True,prefix="Repmt\_statusS2")

df.drop(['S2'], axis=1, inplace=True)

df = df.join(dummy\_ed3)

#convert S3 to dummy data

dummy\_ed4= pd.get\_dummies(df['S3'],drop\_first=True,prefix="Repmt\_statusS3")

df.drop(['S3'], axis=1, inplace=True)

df = df.join(dummy\_ed4)

#convert S4 to dummy data

dummy\_ed5= pd.get\_dummies(df['S4'],drop\_first=True,prefix="Repmt\_statusS4")

df.drop(['S4'], axis=1, inplace=True)

df = df.join(dummy\_ed5)

df.head()

#convert S5 to dummy data

dummy\_ed6= pd.get\_dummies(df['S5'],drop\_first=True,prefix="Repmt\_statusS5")

df.drop(['S5'], axis=1, inplace=True)

df = df.join(dummy\_ed6)

df.head()

**Feature selection with filter method**

matrix=df.corr().round(2)

fig, ax =plt.subplots(figsize=(20,20))

sns.heatmap(matrix, annot=True, linewidths=0.3)

plt.show()

#Correlation with output variable

cor\_target = abs(matrix["B1"])

#Selecting highly correlated features

relevant\_features = cor\_target[cor\_target>0.5]

relevant\_features

**Feature selection with embeded method**

x= df.drop(['B1','ID'], axis =1)

y=df['B1']

reg = LassoCV()

reg.fit(x, y)

print("Best alpha using built-in LassoCV: %f" % reg.alpha\_)

print("Best score using built-in LassoCV: %f" %reg.score(x,y))

coef = pd.Series(reg.coef\_, index = x.columns)

print("Lasso picked " + str(sum(coef != 0)) + " variables and eliminated the other " + str(sum(coef == 0)) + " variables")

print("Feature importance using Lasso Model:")

imp\_coef = coef.sort\_values()

print(imp\_coef)

#sort the series and also remove zero values

imp\_coef = coef.sort\_values()

out=imp\_coef[imp\_coef!=0]

print(out)

matplotlib.rcParams['figure.figsize'] = (25.0, 10.0)

out.plot(kind = "barh")

plt.title("Feature importance using Lasso Model")

**Linear Regression from features using filter method**

# setting the x and y values

x= df[['BALANCE','B2','B3','B4','B5']]

y=df['B1']

# splitting the data

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.3, random\_state = 42)

# creating an object of LinearRegression class

LR = LinearRegression()

# fitting the training data

LR.fit(x\_train,y\_train)

#Prediction of test set

y\_pred\_LR= LR.predict(x\_test)

#Predicted values

print("Prediction for test set: {}".format(y\_pred\_LR))

#Actual value and the predicted vallue

LR\_diff=pd.DataFrame({'Actual value':y\_test,'Predicted value': y\_pred\_LR})

display(LR\_diff)

LR\_diff = LR\_diff.head(25)

LR\_diff.plot(kind='bar',figsize=(16,10))

plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')

plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

plt.title("Comparison of actual vs predicted values")

plt.show()

#Model Evaluation

meanAbErr = metrics.mean\_absolute\_error(y\_test, y\_pred\_LR)

meanSqErr = metrics.mean\_squared\_error(y\_test, y\_pred\_LR)

rootMeanSqErr = np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred\_LR))

print('R squared: {:.2f}'.format(LR.score(x,y)\*100))

print('Mean Absolute Error:', meanAbErr)

print('Mean Square Error:', meanSqErr)

print('Root Mean Square Error:', rootMeanSqErr)

**Linear Regression from features using embedded method**

# setting the x and y values

x1= df[['BALANCE','R1','R4','LIMIT','B5','R3','R2','B2']]

y1= df['B1']

# splitting the data

x1\_train, x1\_test, y1\_train, y1\_test = train\_test\_split(x1, y1, test\_size = 0.3, random\_state = 42)

# creating an object of LinearRegression class

LR1 = LinearRegression()

# fitting the training data

LR1.fit(x1\_train,y1\_train)

#Prediction of test set

y\_pred\_LR1= LR1.predict(x1\_test)

#Predicted values

print("Prediction for test set: {}".format(y\_pred\_LR1))

#Actual value and the predicted value

LR\_diff1=pd.DataFrame({'Actual value':y1\_test,'Predicted value': y\_pred\_LR1})

display(LR\_diff1)

LR\_diff1 = LR\_diff1.head(25)

LR\_diff1.plot(kind='bar',figsize=(16,10))

plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')

plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

plt.title("Comparison of actual vs predicted values")

plt.show()

#Model Evaluation

meanAbErr = metrics.mean\_absolute\_error(y1\_test, y\_pred\_LR1)

meanSqErr = metrics.mean\_squared\_error(y1\_test, y\_pred\_LR1)

rootMeanSqErr = np.sqrt(metrics.mean\_squared\_error(y1\_test, y\_pred\_LR1))

print('R squared: {:.2f}'.format(LR1.score(x1,y1)\*100))

print('Mean Absolute Error:', meanAbErr)

print('Mean Square Error:', meanSqErr)

print('Root Mean Square Error:', rootMeanSqErr)

#plt.scatter(x=y1\_test,y=y\_pred\_LR1,color ='blue');

sns.regplot(x=y1\_test,y=y\_pred\_LR1,scatter\_kws={"color": "blue"}, line\_kws={"color": "red"});

plt.xlabel('Actual');

plt.ylabel('Predicted');

plt.title("Actual vs Predicted dependent variable")

**Python codes (in text) for Question 5:**

**Model Equation (Features selected from embedded method)**

# creating an object of LinearRegression class

LR1 = LinearRegression()

# fitting the training data

model= LR1.fit(x1\_train,y1\_train)

# Intercept and Coefficient

print("Intercept: ",model.intercept\_)

print("Coefficients:")

pd.DataFrame(model.coef\_,x1\_train.columns,columns=['Coeff'])

#double check with statsmodel results

x1\_train\_f = sm.add\_constant(x1\_train) # adding a constant

olsmod = sm.OLS(y1\_train,x1\_train\_f).fit()

print(olsmod.summary())

print("The model equation is: ")

print("Predicted Y=667.33+2.5530Balance-0.4283R1-0.0214R4+0.0023LIMIT+0.0070B5+0.0713R3+0.1678R2+0.5572B2")

**The End**