

**ANL252  
Python for Data Analytics**

**End-of-Course Assessment  
July 2022 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI Number** |
| Wong Hang Sin | Q2071045 |

**Tutorial Group :** T09

**Submission Date :** 05/09/22

**Question 1**

Categorical variables:   
- Gender, Marital, Education, Rating, S(n)  
  
Numeric variables:   
- Limit, Balance, Income, Age, B(n), R(n)  
  
**Question 2**

**Data pre-processing Task 1**

|  |
| --- |
| #Import relevant libraries  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns |
| #Import the dataset  df = pd.read\_csv("ECA\_data.csv") |
| #Check that the data is imported correctly  df |

We notice that there are 18769 rows but the unique customer ID ends at 18766 so there might be 3 duplicated customer ID and information. Hence, we proceed to identify the duplicates and remove them.

|  |
| --- |
| #Check for duplicates  print('Number of rows of duplicated dataset: ', df.duplicated().sum()) |
| #Visualize the duplicated observations  df[df.duplicated()] |
| #Removing 3 rows of duplicated customer information  df.drop\_duplicates((['ID']), inplace = True)  #Check shape of dataset to confirm removal of 3 rows  df.shape |

**Data pre-processing Task 2**

|  |
| --- |
| #Check for missing/null values in the dataset  df.isnull().sum() |
| #Visualize the null observations in EDUCATION  df[df['EDUCATION'].isnull()] |
| #Visualize the null observations in MARITAL  df[df['MARITAL'].isnull()] |

We notice that there are 13 missing values in the EDUCATION column and 38 in the MARITAL column, total 51 missing values. There are also 2 rows with missing values in both the EDUCATION and MARITAL column. The dataset contains a huge amount of data and removing some of them will have little effect as there is already ample data to work with. Replacing these missing values with the mode or mean would not make any sense as well as it will not be an accurate representation. Thus, we proceed to remove the rows with missing values.

|  |
| --- |
| #Dropping the rows with missing values  df.dropna(inplace = True)  #Check shape of dataset to confirm removal of 49 rows  df.shape |

**Data pre-processing Task 3**

|  |
| --- |
| #Check the domain of the columns to identify invalid values  for col in df:      print(df[col].value\_counts())        print('\n') |

We discover that there are 5 customers who are -1 years old and 5 who are 199 years old, which is impossible. Upon further checking, no more errors in ages are found. Thus, we proceed to remove these 10 rows.

|  |
| --- |
| #Check if there are any more error values in age  print( 'Number of customers with age lower than 1: ' , len(df[df['AGE'] < 1]) )  print( 'Number of customers with age higher than 100: ' , len(df[df['AGE'] > 100]) ) |
| #Remove the data of these 10 customers  df = df[(df['AGE']> 0 ) & (df['AGE']<100)]  #Check dataset to confirm removal of 10 rows  df.shape |

**Data pre-processing Task 4**

|  |
| --- |
| #Identify if any data types need to be modified  df.dtypes |

As the values of R3 are identified as objects, we need to convert them into integers.

|  |
| --- |
| #By running this code, we discover that some R3 values have '$' in them through the error received  #[int(element) for element in df.loc[:, 'R3']] |
| #Finding the values of R3 that have '$' and ',' in them  filt = df['R3'].str.startswith('$')  df.loc[filt, 'R3'] |
| #Removing the '$' from R3  df['R3'] = df['R3'].str.replace('$', '',)  #Removing the ',' from R3  df['R3'] = df['R3'].str.replace(',', '',)  #Rhanging R3 from object to integer  df['R3'] = df['R3'].astype(int) |
| #Check that R3 is converted to integer  df.dtypes |

**Question 3**

**Insight and Plot 1:**Investigate the distribution of age and finding the mean age.

|  |
| --- |
| #Calculating the mean age  mean\_age = df['AGE'].mean()  #Plotting the histogram  #Age by count with distribution line, set bin at 15  sns.histplot(df['AGE'], kde=True, bins=15, color='green')  #Plotting the mean  plt.axvline(mean\_age,0,1, color='blacK') |

Chart, histogram

Description automatically generated

Customer ages are normally distributed and skewed to the right with a mean age of 35.5 years old.

**Insight and Plot 2:**Investigate relationship between customer education level and income for different genders.

|  |
| --- |
| #Plotting the bar graph  #x-axis is education, y-axis is income, gender as categories, remove confidence interval  sns.barplot(data=df, x=Education\_Full, y='INCOME', hue=Gender\_Full, ci=False, estimator=np.mean) |

**Chart, bar chart

Description automatically generated**

On average, based on highest level of education attained, postgraduate customers have the highest income followed by others, tertiary and lastly high school. Also, except for postgraduates, female customers earn a higher income than male customers for all other education levels.

**Insight and Plot 3:**  
Investigate relationship between customer marital status and credit limit for different customer ratings.

|  |
| --- |
| #Plotting the bar graph  #x-axis is limit, y-axis is marital, rating as categories, remove confidence interval, horizontal orientation, change color  sns.barplot(data=df, x='LIMIT', y=Marital\_Full, hue=Rating\_Full, ci=False, orient = 'h', color='#A6AC22', estimator=np.mean) |

Chart, bar chart

Description automatically generated  
On average, customers who are single have the highest total credit limit followed by married and lastly other customers. Also, for all different marital status, customers with good rating have higher total credit limits.

**Insight and Plot 4:**Investigate relationship between customer marital status and current credit balance for different genders.

|  |
| --- |
| #Plotting the bar graph  #x-axis is marital, y-axis is balance, gender as categories, remove confidence interval, change color  sns.barplot(data=df, x=Marital\_Full, y='BALANCE', hue=Gender\_Full, ci=False, color='#E49D9F', estimator=np.mean) |

**Chart, bar chart

Description automatically generated**On average, customers who are single have the highest current credit balance followed by those married and others with the lowest. Additionally, regardless of marital status, male customers have higher current credit balances on average.

**Insight and Plot 5:**  
Investigate the relationship between customer income and credit limit.

|  |
| --- |
| #Plotting the scatter graph  #x-axis is income, y-axis is limit  sns.scatterplot(data=df, x='INCOME', y='LIMIT', estimator=np.mean) |

Chart, scatter chart

Description automatically generated  
Customer credit limit and income are highly positively correlated where limit increases as income increases.

**Question 4**

|  |
| --- |
| #Import more relevant libraries  import sklearn  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler  from sklearn.linear\_model import LinearRegression  from sklearn.metrics import r2\_score, explained\_variance\_score, mean\_absolute\_error, mean\_squared\_error  from sklearn import preprocessing  from math import sqrt |

**Further data pre-processing**1. Feature selection

As there are many variables, we select the ones we want to fit into the model to reduce machine computation required.

To select the more important features, we run a correlation to identify which variables are more correlated to our target variable B1.

|  |
| --- |
| #Change heatmap size  fig , ax = plt.subplots(figsize = (20,8))  #Visualising correlation with heatmap  sns.heatmap(df.corr(), annot=True) |

Chart

Description automatically generated with medium confidence

Variables with correlation coefficient to B1 that are closer to 1 or -1 signifies a stronger correlation. This means that B1 is more likely to increase or decrease when values of these variables change.

Thus, the independent variables with highest correlation that we want are: Balance, B1, B2, B3, B4 & B5.

|  |
| --- |
| #Dropping the other columns that are not useful  df\_new = df.drop(['ID', 'LIMIT', 'INCOME', 'RATING', 'GENDER', 'EDUCATION', 'MARITAL', 'AGE', 'S1', 'S2', 'S3', 'S4', 'S5', 'R1', 'R2', 'R3', 'R4', 'R5'], axis = 1)  #Visualising the new dataframe  df\_new.head() |

2. Standardisation

Variables with wider range of values will likely have greater impact in the regression model than variables with smaller range values, therefore needing to be scaled down.

Thus, we standardise the data using Standard Scaler to make values in range so no columns will over predict.

|  |
| --- |
| #Saving independent variables as X  X = df\_new.drop('B1', axis = 1)  #Saving dependent target variable as y  y = df\_new[['B1']]  #Saving Standard Scaler used to standardise our variables  x\_scaler = StandardScaler()  y\_scaler = StandardScaler() |
| #Standardising the independent variables  x\_scaler.fit(X)  X\_stn = pd.DataFrame(x\_scaler.transform(X), columns = X.columns)  #Standardising the dependent variable  y\_scaler.fit(y)  y\_stn = pd.DataFrame(y\_scaler.transform(y), columns = ['B1']) |

**Splitting the Raw Data**

Data has to be split into training and testing dataset. Training dataset is used to construct the model while testing dataset is for evaluating the model's accuracy of predicting unseen data.

|  |
| --- |
| #Divide into training and testing data with proportion of 70:30  X\_train , X\_test , y\_train , y\_test = train\_test\_split(X\_stn, y\_stn, train\_size=0.70, test\_size=0.30, random\_state=20)  #Visualise shape of training and testing datas  print(X\_train.shape)  print(X\_test.shape)  print(y\_train.shape)  print(y\_test.shape) |

**Running the Linear Regression**

|  |
| --- |
| #lm:linear model, insert intercept used for prediction  lm = LinearRegression(fit\_intercept = True)  #Fit training data into lm  lm.fit(X\_train, y\_train)  #Calculate predicted y values  y\_pred = lm.predict(X\_train) |

**Evaluating the Linear Regression Model**

1. Determine model accuracy

For training dataset: calculate R Square using 2 methods where both will return the same result.

For testing dataset: calculate R Square using the score function only.

|  |
| --- |
| #Model Accuracy on training dataset  #Method 1 using the function: score  print('The Accuracy on the training dataset is: ', lm.score(X\_train, y\_train))  #Method 2 using the function: r2\_score  print('The Accuracy using method 2 on the training dataset is: ', r2\_score(y\_train,y\_pred))  print("")  #Model Accuracy on testing dataset  print('The Accuracy on the testing dataset is: ', lm.score(X\_test, y\_test)) |

The Accuracy on the training dataset is: 0.9414104617520799

The Accuracy using method 2 on the training dataset is: 0.9414104617520799

The Accuracy on the testing dataset is: 0.9380816877709176  
  
**Explanation:**

R Square is a measure of how close the data are to the fitted regression line.

The accuracy score represents the coefficient of determination (R^2) which is 1 at maximum and can be negative.

In this case, our model explains 94.1% of the training data and 93.8% of the testing data.

2. Root Mean Squared Error (RMSE)

|  |
| --- |
| #Calculating RMSE for both training and testing dataset using the fucntion: mean\_squared\_error  print('The RMSE on the training dataset is: ',sqrt(mean\_squared\_error(y\_train,y\_pred)))  print('The RMSE on the testing dataset is: ',sqrt(mean\_squared\_error(y\_test,lm.predict(X\_test)))) |

The RMSE on the training dataset is: 0.24083404019301868

The RMSE on the testing dataset is: 0.251727745884312  
 **Explanation:**

RMSE is the standard deviation of the residuals where residuals refer to the difference between the predicted values and the regression line.

Thus, RMSE is a measure of how spread the residuals are where the lower the value the better.  
  
3. Mean Absolute Error (MAE)

|  |
| --- |
| #Calculating MAE using the function: mean\_absolute\_error  print('The MAE on the training dataset is: ',mean\_absolute\_error(y\_train,y\_pred))  print('The MAE on the testing dataset is: ',mean\_absolute\_error(y\_test,lm.predict(X\_test))) |

The MAE on the training dataset is: 0.09005864020525482

The MAE on the testing dataset is: 0.09348893149506347  
  
**Explanation:**

MAE is the average of all the absolute errors where absolute errors are the differences between the true value (y\_train) and the predicted value (y\_pred). The lower the MAE the better.

4. Calculating the Coefficients

The weights of the independent variables.

|  |
| --- |
| #Calculating Coefficients  print('Coefficients: ', lm.coef\_ ) |

Coefficients: [[5.49558671e-01 3.66116583e-01 4.85434827e-02 9.32289694e-05

3.59337445e-02]]  
  
5. Calculating the Intercept

The expected mean value of Y when all X=0

|  |
| --- |
| #Calculating Intercept  print('Intercept: ', lm.intercept\_) |

Intercept: [-0.00083561]

6. Plotting the Model of Actual values vs Predicted values  
Visualise how well our predictions are

|  |
| --- |
| #Increasing figure size  plt.figure(figsize=(15,10))  #Plotting Actuals vs Predicted  plt.scatter(y\_train, y\_pred, c='green')  #Perfect line used to inteprete the results, not the regression line  plt.plot([y\_train.min(), y\_train.max()], [y\_train.min(), y\_train.max()], 'k--', c='red', lw=3)  #X-axis label  plt.xlabel('Actuals')  #Y-axis label  plt.ylabel('Predicted Values')  #Title  plt.title('Actuals Vs Predicted Values') |

Chart, scatter chart

Description automatically generated

**Explanation:**

The red line represents an assumption that our predicted values are 100% correct.

The closer the data points are to the red line, the closer the predictions are.

The further away the data points are to the red line, the further the predictions are.

7. Plotting Residuals

|  |
| --- |
| #Increasing figure size  plt.figure(figsize=(15,10))  #Plotting residuals  sns.residplot(y\_train, y\_pred, color='green')  #X-axis label  plt.xlabel('Actual B1')  #Y-axis label  plt.ylabel('Residuals')  #Title  plt.title('Actuals Vs Residuals') |

Chart, scatter chart

Description automatically generated  
**Explanation:**

The closer the data points are to the dotted line, the closer the predictions are.

The further away the data points are to the dotted line, the further the predictions are.

**Using the Linear Regression Model to Make Predictions**

To test the accuracy of the predictions, we use 2 methods to compare the predicted value with the actual value.

|  |
| --- |
| #Testing prediction accuracy method 1  pred1 = y\_scaler.inverse\_transform(lm.predict(x\_scaler.transform(X.loc[[2], :])))  print('The predicted B1 is: ', pred1)  #Testing prediction accuracy method 2  pred1 = y\_scaler.inverse\_transform(lm.predict(x\_scaler.transform([[65397.85, 352484, 338823, 283288, 185288]])))  print('The predicted B1 is: ', pred1) |

In this case, referring to actual values from the dataset, the actual B1 is 343591 while the predicted value is 356227, which is quite close meaning the model is fairly accurate.  
  
**Making an Actual Prediction**

|  |
| --- |
| #Random values for the independent variables  #BALANCE = 70584  #B2 = 41658  #B3 = 39276  #B4 = 27454  #B5 = 19781  #Prediction Calculator  pred1 = y\_scaler.inverse\_transform(lm.predict(x\_scaler.transform([[70584, 41658, 39276, 27454, 19781]])))  print('The predicted B1 is: ', pred1) |

The predicted B1 is 232548.

**Question 5**

From question 4,

|  |
| --- |
| #Calculating Coefficients  print('Coefficients: ', lm.coef\_ ) |

Coefficients: [[5.49558671e-01 3.66116583e-01 4.85434827e-02 9.32289694e-05

3.59337445e-02]]

|  |
| --- |
| #Calculating Intercept  print('Intercept: ', lm.intercept\_) |

Intercept: [-0.00083561]  
  
Building on from insights in question 4, the linear equation is:

**B1 = 0.550\*BALANCE + 0.366\*B2 + 0.0485\*B3 + 0.0000932\*B4 + 0.0359\*B5 – 0.00083561**All the independent variables are positively correlated to B1 where an increase in any independent variable will result in an increase in B1 and vice versa. For instance, for every 1 unit increase in BALANCE, B1 will increase by 0.550.  
  
Next, a customer’s current credit balance has the greatest impact on his/her billable amount in the most recent month B1 as it has the largest coefficient.   
  
B2 which represents the billable amount in the previous 1st month has the 2nd greatest impact. BALANCE and B2 are still highly correlated to B1 and have a significant impact on it.   
  
However, B3, B4 and B5 all have lesser impact on B1 with B4, the billable amount in the previous 3rd month having the least impact. This means that for these variables to affect B1, it must increase or decrease on a larger scale than that for BALANCE and B2.

**References**

sklearn.model\_selection.train\_test\_split. (2022). <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>

sklearn.preprocessing.StandardScaler. (2022). <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>

sklearn.linear\_model.LinearRegression. (2022). <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html>

sklearn.metrics.r2\_score. (2022). <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html>

sklearn.metrics.mean\_squared\_error. (2022). <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html>

sklearn.metrics.mean\_absolute\_error. (2022). <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html>