FIT1043 Introduction to Data Science

Assignment 2

Juliet Chow, 32112602

14^{th} September 2021

INTRODUCTION

This assignment is to conduct predictive analytics, through machine learning using Python in the Jupyter Notebook environment. This assignment will test my ability to: read and describe the data using basic statistics, split the dataset into training and testing, conduct binary classification using Random Forest and Decision Tree, implement linear regression for prediction, communicate the output of my analysis and experience independent model evaluation through reporting the metrics.

2 data will be given, that are: loan_data and Customers-shop. We are ask to predict whether or not the borrower paid back their loan in full and decide whether to focus company's efforts on mobile application or website for both of the data respectively. We will learn how to predict the desirable results by using Decision Tree Classifier, Random Forest Classifiers, and Linear Regression.

Check The Location of Jupyter Notebook File

In [1]:
```
pwd
```

Out[1]:
```
'/Users/Juliet/Documents/Monash Uni/FIT1043/Assignment/Assignment 2/Assignment 2Data'
```

Checking the location of the intended files, they should be in a folder called "Assignment 2Data"

Question 1

INTRODUCTION

Loan_data.csv will be used as our data. The dataset contains various information for borrowers who have requested loan from an investor (lender). Hopefully, as an investor we would want to invest in people who showed a profile of having a high probability of paying us back.

The objective for the first question is to create a model to classify and predict whether or not the borrower paid back their loan in full.

The flow of this assignment will include

- Read and describe the data using basic statistics
- Split the dataset into training and testing
- Conduct binary classification using Random Forest and Decision Tree
- Communicate the output of the analysis

Importing Libraries

The first step is to import library **pandas**, which is an open source data analysis tool for python programming language. The purpose of importing this library is we would like to use the data structure such as DataFrame and it's associated functions such as reading from CSV files and so on.

Matplotlib and seaborn library should also be imported. In these libraries, they have useful functions to make a better visualization of plotting the necessary graphs.

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Reading loan_data.csv

In [3]:
```
loan_data = pd.read_csv("loan_data.csv")
```

In [4]:
```
loan_data
```

Out[4]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal
0	1	0.1189	829.10	11.350407	19.48	737	5639.958333	28854
1	1	0.1071	228.22	11.082143	14.29	707	2760.000000	33623
2	1	0.1357	366.86	10.373491	11.63	682	4710.000000	3511
3	1	0.1008	162.34	11.350407	8.10	712	2699.958333	33667
4	1	0.1426	102.92	11.299732	14.97	667	4066.000000	4740
...	...	...	...	...	...	...	...	...
9573	0	0.1461	344.76	12.180755	10.39	672	10474.000000	215372
9574	0	0.1253	257.70	11.141862	0.21	722	4380.000000	184
9575	0	0.1071	97.81	10.596635	13.09	687	3450.041667	10036
9576	0	0.1600	351.58	10.819778	19.18	692	1800.000000	0
9577	0	0.1392	853.43	11.264464	16.28	732	4740.000000	37879

9578 rows × 13 columns

In [5]:
```
loan_data.head()
```

Out[5]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	re
0	1	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	
1	1	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	
2	1	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	
3	1	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	
4	1	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	

In [6]:
```
loan_data.tail()
```

Out[6]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal
9573	0	0.1461	344.76	12.180755	10.39	672	10474.000000	215372
9574	0	0.1253	257.70	11.141862	0.21	722	4380.000000	184
9575	0	0.1071	97.81	10.596635	13.09	687	3450.041667	10036
9576	0	0.1600	351.58	10.819778	19.18	692	1800.000000	0
9577	0	0.1392	853.43	11.264464	16.28	732	4740.000000	37879

Read the loan_data.csv files and store it into loan_data variable.

Then, check the first and last five rows of the data using .head() and .tail() functions consecutively. The file has been correctly ouputted

Do We Need to Wrangle the Data?

In [7]:
```
loan_data.isna().sum()
```

Out[7]:
```
credit.policy        0
int.rate             0
installment          0
log.annual.inc       0
dti                  0
fico                 0
days.with.cr.line    0
revol.bal            0
revol.util           0
inq.last.6mths       0
delinq.2yrs          0
pub.rec              0
not.fully.paid       0
dtype: int64
```

There is no missing values in any of the columns, thus we do not need to wrangle the data

Basic Descriptive Statistics

In [8]:
```
loan_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   credit.policy      9578 non-null   int64  
 1   int.rate           9578 non-null   float64
 2   installment        9578 non-null   float64
 3   log.annual.inc     9578 non-null   float64
 4   dti                9578 non-null   float64
 5   fico               9578 non-null   int64  
 6   days.with.cr.line  9578 non-null   float64
 7   revol.bal          9578 non-null   int64  
 8   revol.util         9578 non-null   float64
 9   inq.last.6mths     9578 non-null   int64  
 10  delinq.2yrs        9578 non-null   int64  
 11  pub.rec            9578 non-null   int64  
 12  not.fully.paid     9578 non-null   int64  
dtypes: float64(6), int64(7)
memory usage: 972.9 KB
```

All data types of loan_data are all numbers in integer or float.

In [9]:
```
loan_data.describe()
```

Out[9]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days
count	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	
mean	0.804970	0.122640	319.089413	10.932117	12.606679	710.846314	
std	0.396245	0.026847	207.071301	0.614813	6.883970	37.970537	
min	0.000000	0.060000	15.670000	7.547502	0.000000	612.000000	
25%	1.000000	0.103900	163.770000	10.558414	7.212500	682.000000	
50%	1.000000	0.122100	268.950000	10.928884	12.665000	707.000000	
75%	1.000000	0.140700	432.762500	11.291293	17.950000	737.000000	
max	1.000000	0.216400	940.140000	14.528354	29.960000	827.000000	1

Firstly, there are 9578 borrowers with their information given. More than 75% of the borrowers meets the credit underwriting criteria of investor. It shows from the average that is 0.805 out of 9578 borrowers. Moreover, half of the borrowers are considered to be less risky than the others. It shows from the value of 50 percentile, 0.122100, which is lower than the average interest rate.

In addition, the distribution of monthly installment are to the right, or positive side. The reason is because the mean is greater than the median. It can be also known that monthly installment have fewer large values as the data is skewed to the right. Furthermore, almost half of the borrowers managed have a credit score above the average of 710.85.

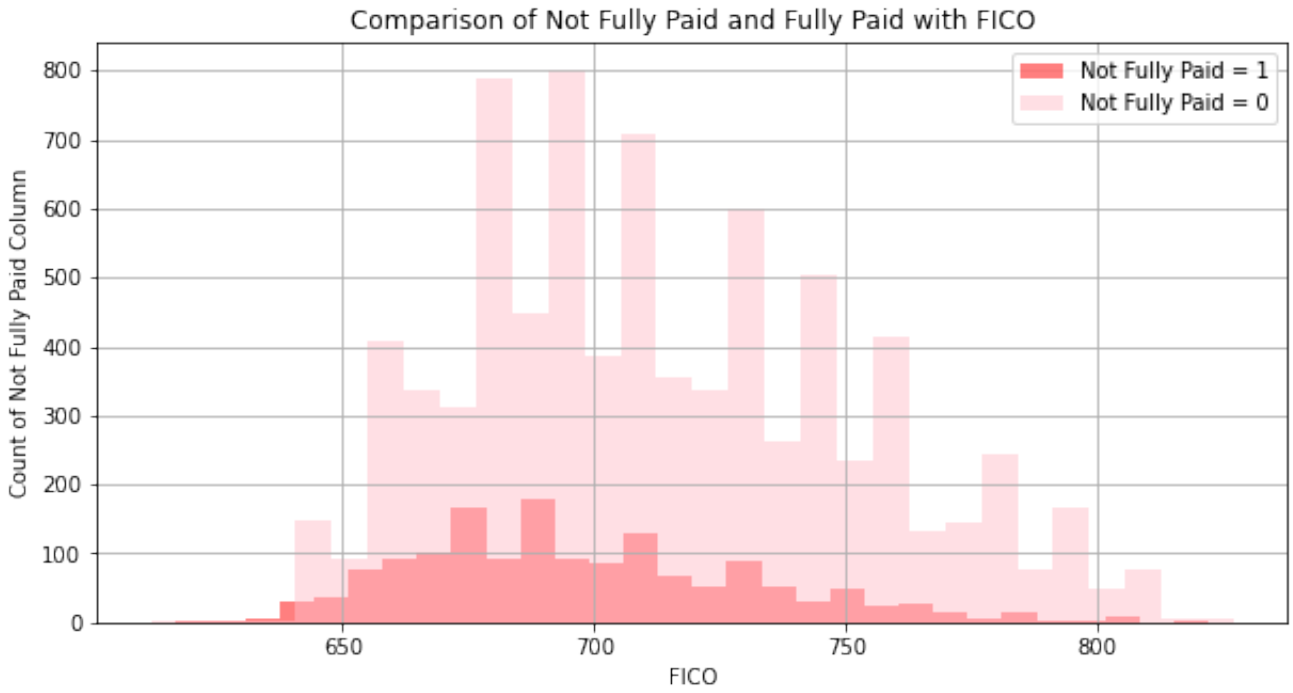
Moreover, the data for log.annual.inc is clustered around the mean as it has a very low value of standard deviation of 0.614813. Lastly, from the value of 25% that is 0 and the average of not fully paid column that is considered low (0.16 out of 9578 borrowers with the maximum value of 1), it indicates that less than 25% of the borrowers have fully paid their loan.

Create a histogram of not.fully.paid column on top of each other, one for each not.fully.paid outcome (0 and 1).

In [10]:

```
# Set the size of out plot
plt.figure(figsize=(10,5))

# Create histogram for each not fully paid values in respective to fico
loan_data[loan_data['not.fully.paid']==1]['fico'].hist(bins=30,
                                                       color='red',
                                                       alpha=0.5,
                                                       label='Not Fully Paid
loan_data[loan_data['not.fully.paid']==0]['fico'].hist(bins=30,
                                                       color='pink',
                                                       alpha = 0.5,
                                                       label='Not Fully Paid

# Show the legend
plt.legend()

# Set title
plt.title('Comparison of Not Fully Paid and Fully Paid with FICO')

# Set x-label
plt.xlabel("FICO")

# Set y-label
plt.ylabel("Count of Not Fully Paid Column")
```

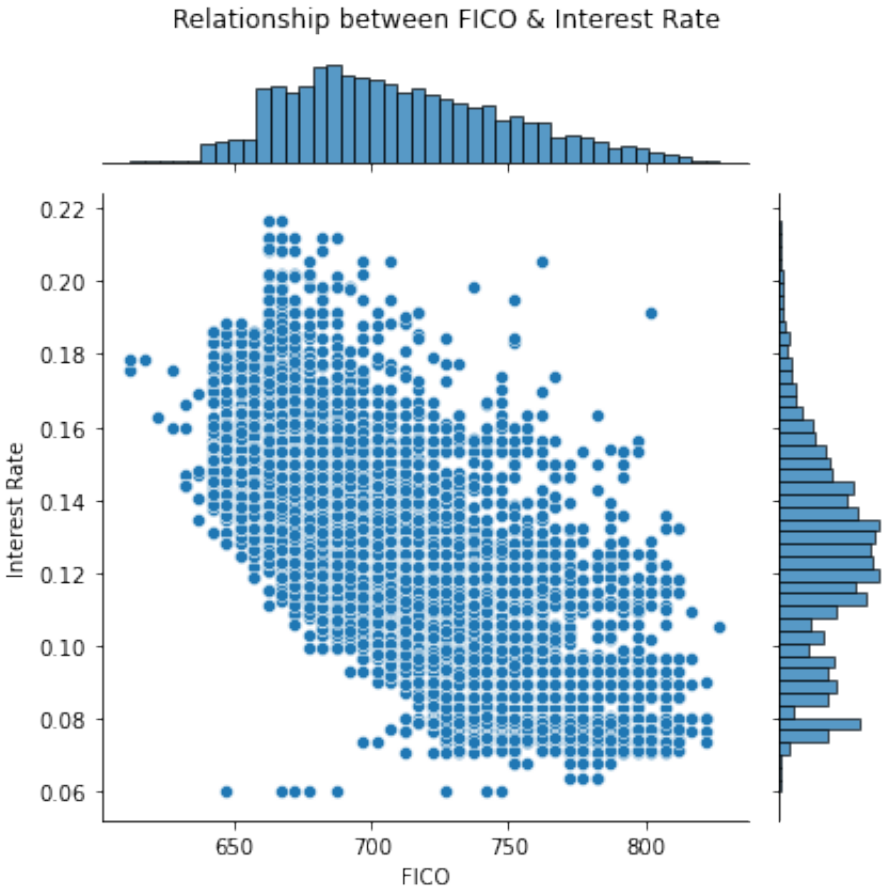
Out[10]: Text(0, 0.5, 'Count of Not Fully Paid Column')



Create a plot to show the relationship between fico and interest rate.

In [11]:

```
# Join fico and int.rate column and plot them from loan_data using seaborn jo
fico_int = sns.jointplot(x='fico', y='int.rate', data=loan_data)

# Set title
fico_int.fig.suptitle('Relationship between FICO & Interest Rate')

# Set x-label
fico_int.ax_joint.set_xlabel('FICO')

# Set y-label
fico_int.ax_joint.set_ylabel('Interest Rate')

# Adjusting labels
plt.tight_layout()
```


Relationship between FICO & Interest Rate
Interest Rate
0.22
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
650
700
750
800
FICO

SUPERVISED LEARNING

Explain supervised machine learning, the notion of labelled data, and the training and test datasets.

Supervised learning is a subcategory of machine learning. Labelled datasets are used to train alogrithms for predicting or classifying data accurately. Its accuracy can be measured and learn over time by using labelled inputs and outputs by the model (IBM Cloud Education, 2020). Supervised learning is suitable for regression and classification problems (David Petersson, 2021).

According to Techopedia (2020), the notion of labelled data is a designation for pieces of data to identify certain properties or classifications that have been tagged by labels.

Training datasets are datasets that are used to fit the parameters of a machine learning model by assigning relevant labels with the goal of training it. Their purpose is to teach algorithm in making accurate predictions (Label Your Data, 2021).

Test datasets are a set of datas that are used to test a machine learning model after it has been trained on a training data set (Techopedia, 2018). The purpose of test dataset is to prove that the machine learning model was trained effectively (Applause App Quality, 2021).

Separate the features and the label and mention what is your label.

In [12]:
```
X = loan_data.drop('not.fully.paid', axis = 1)
y = loan_data['not.fully.paid']
```

The features would be all of the columns in loan_data except not.fully.paid column represent by **X**, and the label would be not.fully.paid column represents by **y**. Features are the independent variables and label is the dependent variable. This means the the result of label will depend on the features.

Use the sklearn.model_selection.train_test_split function to split your data for training (80 %) and testing (20%).

In [13]:
```
from sklearn.model_selection import train_test_split
```

Import train_test_split from sklearn.model_selection modules to split the train and test datasets

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, ra
```

I use random_state as 0 to have a constant randomization with the test size of 20% and train sets of 80%.

CLASSIFICATION

Explain the difference between binary and multi-class classification and point what type of classification is your current problem.

Binary classification is a classification of a given data that is being classified into two classes. It predicts which of the two groups the thing belongs to. For example, whether an email will be a spam or not. In this case, assign 1 if it is a spam and 0 otherwise. On the other hand, multi-class classification is the task of classifying elements into different classes. It does not restrict itself to any number of classes unlike binary. For instance, classifying books according to the subject.

For this current problem, it would be binary classification because we are classifying our problems into two classes. We are considering whether or not the borrower will fully pay back their loan. If they fully paid their loan, assign 1 and 0 otherwise. Therefore, it is a binary classification

Train a Decision Tree Model

In [15]:
```
from sklearn.tree import DecisionTreeClassifier
```

Import DecisionTreeClassifier from sklearn.tree module

In [16]:
```
# Fitting Decision Tree Classification to the Training set
dtclassifier = DecisionTreeClassifier(criterion = 'entropy',
                                      random_state = 0)
dtclassifier.fit(X_train, y_train)
```

Out[16]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

Predict

In [17]:
```
# Predicting the Test set results
y_pred = dtclassifier.predict(X_test)
```

In [18]:
```
y_pred
```

Out[18]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [19]:
```
y_test
```

Out[19]:
```
3343    0
9       0
1741    0
5160    0
7760    0
       ..
2113    0
9282    0
2434    0
7287    1
318     0
Name: not.fully.paid, Length: 1916, dtype: int64
```

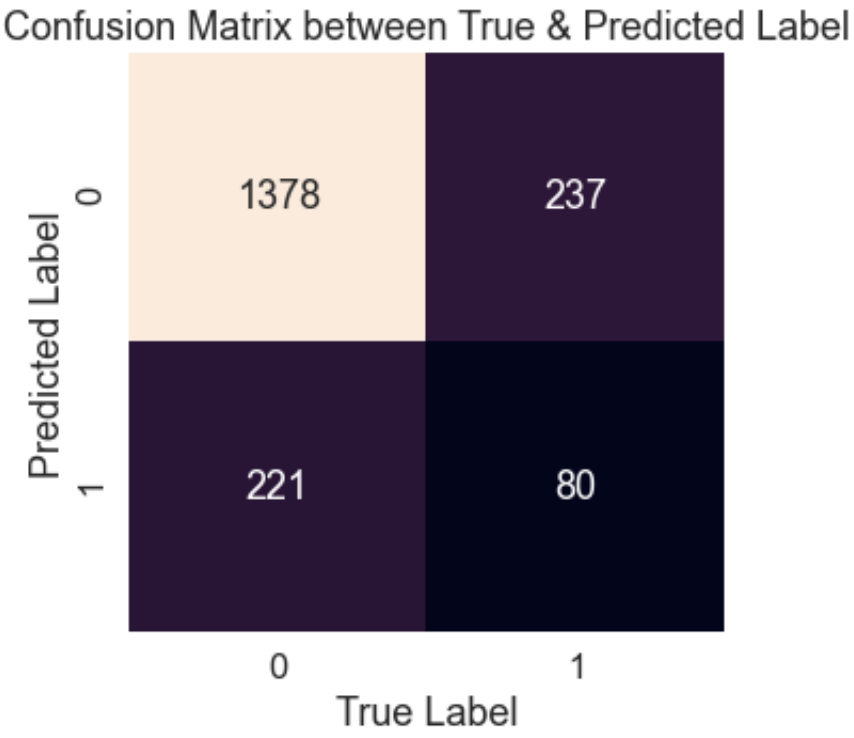
Display Confusion Matrix

In [20]:
```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
conf_mat = confusion_matrix(y_test, y_pred)
conf_mat
```

Out[20]:
```
array([[1378,  237],
       [ 221,   80]])
```

In [21]:
```
# Set the font scale
sns.set(font_scale=1.5)
# Plots a confusion matrix using Seaborn's heatmap().
fig, ax = plt.subplots(figsize=(5,5))
ax = sns.heatmap(conf_mat,
                 fmt ='d',
                 annot= True, # Annotate the boxes with conf_mat info
                 cbar = False)
# Set title
plt.title("Confusion Matrix between True & Predicted Label")

# Set x-label
plt.xlabel("True Label")

# Set y-label
plt.ylabel("Predicted Label")
```

Out[21]:
```
Text(16.5, 0.5, 'Predicted Label')
```



Explain Confusion Matrix

From the confusion matrix where both True Label and Predicted Label are Positive, 1378 are predicted to be able to pay back their loan fully and it is true (**True Positive**). Furthermore, there are 80 borrowers who are predicted unable to fully paid back their loan and it is true (**True Negative**). In addition, 237 borrowers are predicted able to pay back their loan, where in fact they are unable to do that (**False Positive**). Lastly, we predicted that 221 borrowers are unable to pay back their loan fully. In fact, they are able to do that(**False Negative**).

Measure the Performance for Decision Tree Classifier

In [22]:
```
dtclassifier.score(X_test,y_test)
```

Out[22]:
```
0.7609603340292276
```

0.76096 represents our decision tree model are 76.096% accurate

Train a Random Forest Model

In [23]:
```
from sklearn.ensemble import RandomForestClassifier
```

Import RandomForestClassifier from sklearn.ensemble module

In [24]:
```
# Fitting Random Forest Classification to the Training set
rfclassifier = RandomForestClassifier(n_estimators = 2000,
                                      criterion = 'entropy',
                                      random_state = 0
                                     )
rfclassifier.fit(X_train, y_train)
```

Out[24]:
```
RandomForestClassifier(criterion='entropy', n_estimators=2000, random_state=0)
```

Predict

In [25]:
```
# Predicting the Test set results
y_pred = rfclassifier.predict(X_test)
```

In [26]:
```
y_pred
```

Out[26]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [27]:
```
y_test
```

Out[27]:
```
3343    0
9       0
1741    0
5160    0
7760    0
       ..
2113    0
9282    0
2434    0
7287    1
318     0
Name: not.fully.paid, Length: 1916, dtype: int64
```

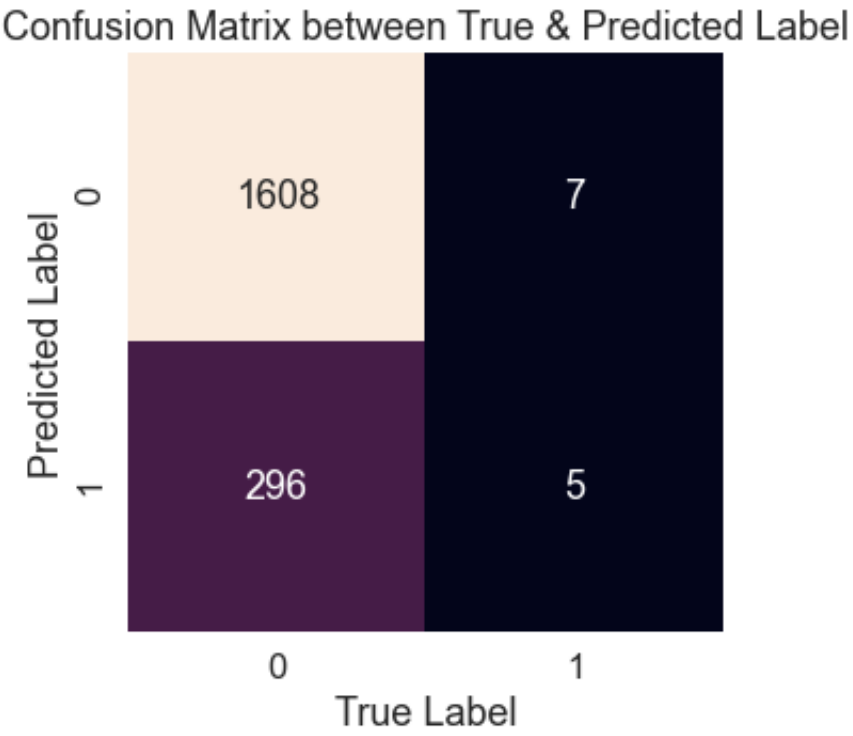
Display Confusion Matrix

In [28]:
```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
conf_mat = confusion_matrix(y_test, y_pred)
conf_mat
```

Out[28]:
```
array([[1608,    7],
       [ 296,    5]])
```

In [29]:
```
# Plots a confusion matrix using Seaborn's heatmap()
fig, ax = plt.subplots(figsize=(5,5))
ax = sns.heatmap(conf_mat,
                 fmt ='d',
                 annot= True, # Annotate the boxes with conf_mat info
                 cbar = False)
# Set title
plt.title("Confusion Matrix between True & Predicted Label")

# Set x-label
plt.xlabel("True Label")

# Set y-label
plt.ylabel("Predicted Label")
```

Out[29]: Text(16.5, 0.5, 'Predicted Label')



Explain Confusion Matrix

1608 are predicted to be able to pay back their loan fully and it is true (**True Positive**). Furthermore, there are 5 borrowers which are predicted unable to fully paid back their loan and it is true (**True Negative**). In addition, 7 borrowers are predicted able to pay back their loan, where in fact they are unable to do that (**False Positive**). Lastly, we predicted that 296 borrowers are unable to pay back their loan fully. In fact, they are able to do that(**False Negative**).

Measure the Performance for Random Forest Classifier

In [30]:
```
rfclassifier.score(X_test,y_test)
```

Out[30]: 0.8418580375782881

0.84186 represents our random forest model are 84.186% accurate

CONCLUSION

I measure the performance of both classfiers by using their accuracy. I use .score(X_test, y_test) function to do measurement. The actual formula of .score function is (True Positive + True Negative)/(True Positive + True Negative + False Negative + False Positive).

Both models can be compared since their random state are the same. I use 0 as the random state for both classifiers. From the previous calculation, we got the accuracy of 76.096% and 84.186% for Decision Tree and Random Forest model respectively. Random Forest model has a higher accuracy than Decision Tree's.

This may happen because decision tree model gives high importance to a particular set of features. But the random forest chooses features randomly during the training process. Thus, it does not depend highly on any specific set of features. This randomized feature selection makes random forest much more accurate than a decision tree. With Random Forest model that has a higher performance rate, it can be concluded that Random Forest model performs better than Decision Tree model.

Question 2

INTRODUCTION

Customers-shop.csv will be used as our data. The dataset contains various information such as: average session of in-store style advice sessions, average time spent on application & websites by customers and the length of membership. The company is trying to decide whether to focus their efforts on their mobile app experience or their website.

The objective for the second question is to create a model to decide whether to focus company's efforts on mobile application or website.

The flow of this assignment will include

- Read and describe the data using basic statistics
- Split the dataset into training and testing
- Implement linear regression for prediction
- Communicate the output of the analysis
- Experience independent model evaluation through reporting the metrics

Importing Libraries

The first step is to import library **pandas**, which is an open source data analysis tool for python programming language. The purpose of the importing this library is that we would like to use the data structure such as DataFrame and it's associated functions such as reading from CSV files and so on.

Matplotlib and seaborn library should also be imported. In these libraries, they have useful functions to make a better visualization of plotting the necessary graphs.

The purpose of numpy library is to calculate RSME for accuracy metrics.

In [31]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Read Customers-shop.csv

In [32]:
```
customers_shop = pd.read_csv("Customers-shop.csv")
```

In [33]:
```
customers_shop
```

Out[33]:

	Customer info-color Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	Violet	34.497268	12.655651	39.577668	4.082621	587.951054
1	DarkGreen	31.926272	11.109461	37.268959	2.664034	392.204933
2	Bisque	33.000915	11.330278	37.110597	4.104543	487.547505
3	SaddleBrown	34.305557	13.717514	36.721283	3.120179	581.852344
4	MediumAquaMarine	33.330673	12.795189	37.536653	4.446308	599.406092
...	...	...	...	...	...	...
495	Tan	33.237660	13.566160	36.417985	3.746573	573.847438
496	PaleVioletRed	34.702529	11.695736	37.190268	3.576526	529.049004
497	Cornsilk	32.646777	11.499409	38.332576	4.958264	551.620146
498	Teal	33.322501	12.391423	36.840086	2.336485	456.469510
499	DarkMagenta	33.715981	12.418808	35.771016	2.735160	497.778642

500 rows × 6 columns

In [34]:
```
customers_shop.head()
```

Out[34]:

	Customer info-color Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	Violet	34.497268	12.655651	39.577668	4.082621	587.951054
1	DarkGreen	31.926272	11.109461	37.268959	2.664034	392.204933
2	Bisque	33.000915	11.330278	37.110597	4.104543	487.547505
3	SaddleBrown	34.305557	13.717514	36.721283	3.120179	581.852344
4	MediumAquaMarine	33.330673	12.795189	37.536653	4.446308	599.406092

In [35]:
```
customers_shop.tail()
```

Out[35]:

	Customer info-color Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
495	Tan	33.237660	13.566160	36.417985	3.746573	573.847438
496	PaleVioletRed	34.702529	11.695736	37.190268	3.576526	529.049004
497	Cornsilk	32.646777	11.499409	38.332576	4.958264	551.620146
498	Teal	33.322501	12.391423	36.840086	2.336485	456.469510
499	DarkMagenta	33.715981	12.418808	35.771016	2.735160	497.778642

Read the customers_shop.csv files and store it into customers_shop variable.

Then, check the first and last five rows of the data using .head() and .tail() functions consecutively. The file has been correctly ouputted

Do we need to wrangle the data?

In [36]:
```
customers_shop.isna().sum()
```

Out[36]:
```
Customer info-color Avatar    0
Avg. Session Length           0
Time on App                   0
Time on Website               0
Length of Membership          0
Yearly Amount Spent           0
dtype: int64
```

There is no missing values in any of the columns, thus we do not need to wrangle the data

Basic Descriptive Statistics

In [37]:
```
customers_shop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 6 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Customer info-color Avatar  500 non-null    object 
 1   Avg. Session Length         500 non-null    float64
 2   Time on App                 500 non-null    float64
 3   Time on Website             500 non-null    float64
 4   Length of Membership        500 non-null    float64
 5   Yearly Amount Spent         500 non-null    float64
dtypes: float64(5), object(1)
memory usage: 23.6+ KB
```

All datatypes for each column of customers_shop data are float except the first column which is an object that represents Customer info-color Avatar

In [38]:
```
customers_shop.describe()
```

Out[38]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
count	500.000000	500.000000	500.000000	500.000000	500.000000
mean	33.053194	12.052488	37.060445	3.533462	499.314038
std	0.992563	0.994216	1.010489	0.999278	79.314782
min	29.532429	8.508152	33.913847	0.269901	256.670582
25%	32.341822	11.388153	36.349257	2.930450	445.038277
50%	33.082008	11.983231	37.069367	3.533975	498.887875
75%	33.711985	12.753850	37.716432	4.126502	549.313828
max	36.139662	15.126994	40.005182	6.922689	765.518462

Firstly, there are 500 customers with their information given. On average, customers spend 33 minutes in the store for in-store style advice sessions. The data for average session length are quite left skewed since the median is bigger than the mean. In addition, half of the recorded customers spend their time around 11.98 minutes and 37.07 minutes on app and websites respectively.

Moreover, the data for time on app and time on website are both clustered around the mean as they have very low value of standard deviation 0.994216 and 1.010489 respectively. Furthermore, on average the length of membership of customers is around 3.5 years. Lastly, the yearly amount spent by the customers are around $499.31/year. Additionally, the data for yearly amount spent column shows a bit of variety since the value of standard deviation is quite high.

Create a scatter plot to compare the Time on Website and Yearly Amount Spent columns

In [39]:

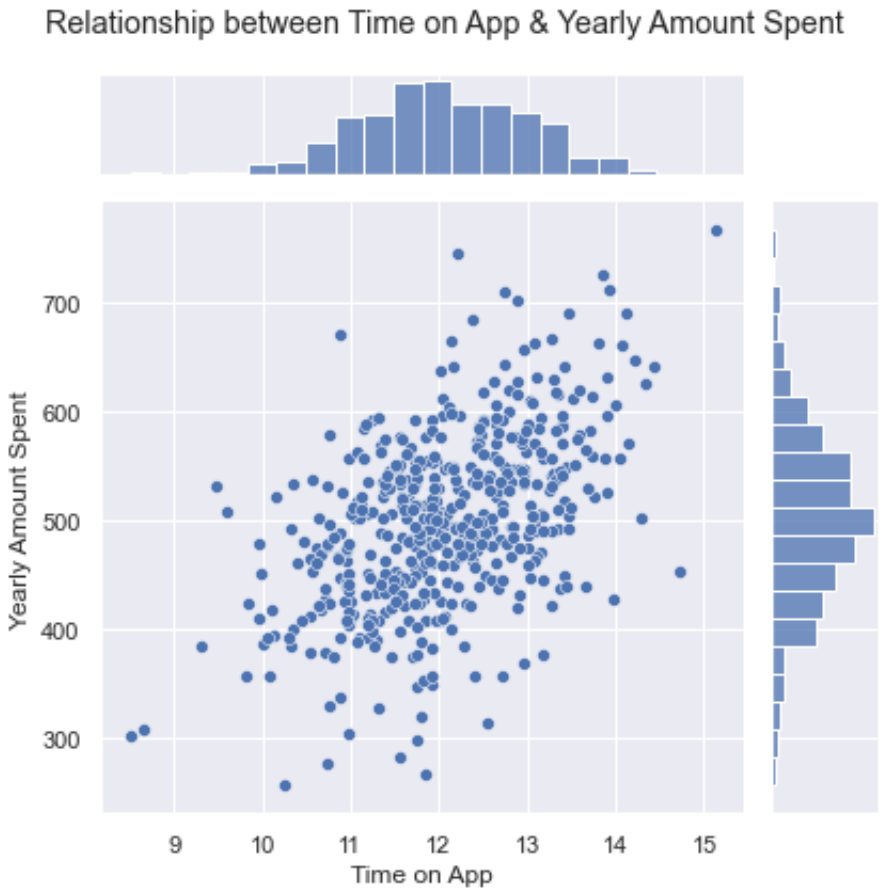
```
# Set the font scale
sns.set(font_scale=1)

# Join the plot of Time on Website and Yearly Amount Spent by using Seaborn's
web_spent = sns.jointplot(x = 'Time on Website', y= 'Yearly Amount Spent', da

# Set title
web_spent.fig.suptitle('Relationship between Time on Website & Yearly Amount

# Set x-label
web_spent.ax_joint.set_xlabel('Time on Website')

# Set y-label
web_spent.ax_joint.set_ylabel('Yearly Amount Spent')

# Adjusting labels
plt.tight_layout()
```

Does the correlation make sense?

No, there is no correlation between the time customers stay on the website and the yearly amount spent because the points on a scatterplot do not show any pattern.

Create a scatter plot to compare the Time on App and Yearly Amount Spent columns

In [40]:
```
# Join the plot of Time on App and Yearly Amount Spent by using Seaborn's joi
app_spent = sns.jointplot(x = 'Time on App', y= 'Yearly Amount Spent', data =

# Set title
app_spent.fig.suptitle('Relationship between Time on App & Yearly Amount Spen

# Set x-label
app_spent.ax_joint.set_xlabel('Time on App')

# Set y-label
app_spent.ax_joint.set_ylabel('Yearly Amount Spent')

# Adjusting labels
plt.tight_layout()
```



Compare this correlation with corelation at part c

The correlation between Time on App and Yearly Amount Spent makes more sense than the correlation between Time on Websie and Yearly Amount Spent. It is because the points of the scatter plot of Time on App and Yearly Amount Spent creates a visible pattern which is increasing although it is not a apparent. On the other hand, the points of the scatter plot for Time on Website and Yearly Amount Spend does not correlate with each other. Since its scatter plot does not show any pattern at all.

SUPERVISED LEARNING

Separate the features and the label and mention what is your label.

In [41]:
```
X = customers_shop.select_dtypes(exclude = 'object').drop('Yearly Amount Spen
y = customers_shop['Yearly Amount Spent']
```

The features would be all of the columns in customers_shop except Yearly Amount Spent column and the column that has object as its data type that is Customer info-color Avatar. It is represents by **X**. The label would be Yearly Amount Spent column represents by **y**. Features are the independent variables and label is the dependent variable. This means the the result of label will depend on the features.

Use the sklearn.model_selection.train_test_split function to split your data for training (70%) and testing (30%).

In [42]:
```
from sklearn.model_selection import train_test_split
```

Import train_test_split from sklearn.model_selection modules to split the train and test datasets

In [43]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, ra
```

From the question given, it required us to have testing sets 30% and training data 70%. That is why the test_size = 0.3.

We will have 70% of training datasets of X data and the other 30% of testing datasets of X. Furthermore, we will also have 70% of the training datasets for the label and the other 30% for the testing datasets.

I use random_state as 42 to have a constant randomization

Regression

Train a Linear regression model (to import the package use: sklearn.linear_model import LinearRegression)

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

Import LinearRegression from sklearn.linear_model module

In [45]:
```
# Fitting Linear Regression Model to the Training set
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
```

Out[45]:
```
LinearRegression()
```

Report the coefficients of the regression model

In [46]:
```
coef_customers_shop = pd.DataFrame(linear_model.coef_, X.columns, columns=['C
coef_customers_shop
```

Out[46]:

	Coefficient
Avg. Session Length	25.724256
Time on App	38.597135
Time on Website	0.459148
Length of Membership	61.674732

A positive coefficient : the value of the independent variable increases, the mean of the dependent variable increases A negative coefficient : the value of the independent variable increases, the dependent variable decreases.

In our case, there is a positive correlation between average session length, time on app, time on website and length of membership to yearly amount spent. When there is one-unit increase in :

- Average Session Length increases the Yearly Amount Spent by **25.724256** units
- Time on App increases the Yearly Amount Spent by **38.597135** units
- Time on Website increases the Yearly Amount Spent by **0.459148** units
- Length of Membership increases the Yearly Amount Spent by **61.674732** units

Predict

Using the testing dataset you created in 2(b) above, conduct the prediction for the 'yearly-amount-spend'.

In [47]:
```
# Predicting the Test set results
y_pred = linear_model.predict(X_test)
```

In [48]:
```
y_pred
```

Out[48]:
```
array([403.66993074, 542.57756279, 427.06591624, 502.02460418,
       410.12143558, 569.93442498, 531.93431357, 506.29651001,
       408.71870637, 473.97737102, 441.46912713, 425.33703067,
       425.12972304, 527.61676736, 431.45684042, 424.07691828,
       575.76543275, 484.8985653 , 458.35936887, 481.96502194,
       502.32441485, 513.63783571, 507.58877004, 646.57464281,
       450.24372153, 496.27043414, 556.40457802, 554.95630854,
       399.64237181, 325.84623149, 532.89783273, 478.12238679,
       501.05701852, 305.9733584 , 505.77244465, 483.79591959,
       518.83315284, 438.18241848, 456.71094212, 471.04609471,
       494.44008982, 445.31155776, 508.78802753, 501.04594195,
       488.83499651, 535.38079536, 595.20129815, 514.04714863,
       280.76758287, 433.10112339, 421.70823429, 481.23640159,
       584.71372254, 608.77480967, 563.98513435, 494.72804856,
       394.52133409, 456.41975284, 573.08767517, 499.69842395,
       512.83276999, 392.12434057, 480.05057696, 481.54520305,
       475.11173582, 546.27175344, 430.85039071, 602.16082001,
       422.36951266, 493.5728019 , 528.74970327, 581.49002655,
       620.191393  , 512.56880316, 411.76623851, 498.47637507,
       461.51337567, 446.41371033, 448.07229951, 535.44710408,
       599.45225289, 619.33717643, 494.1591908 , 671.99976413,
       532.46469823, 438.90606299, 515.04975264, 546.78219536,
       331.94282078, 510.51987441, 536.57891032, 500.19533626,
       376.92345779, 573.73961372, 479.68031618, 588.61435488,
       485.69922213, 456.40200842, 399.2519786 , 451.50989293,
       519.40693806, 434.71194239, 596.13049592, 487.91791955,
       407.46691815, 524.16812778, 504.12982768, 452.1154063 ,
       524.21791272, 457.59311634, 444.19371581, 457.80432915,
       448.76590762, 438.3178902 , 677.04967978, 566.09639249,
       651.93616673, 381.08127931, 577.55772544, 578.35797049,
       518.61431299, 538.94532336, 377.43012217, 663.30814875,
       523.8315883 , 456.86065608, 446.07594386, 388.55038274,
       521.03242156, 431.94999246, 460.08016322, 426.31959494,
       433.30417066, 634.89577553, 462.41086075, 460.7167386 ,
       512.49535287, 703.8303388 , 411.84238599, 551.54681381,
       553.33669555, 409.68202105, 423.3449132 , 509.66438614,
       509.88865186, 543.67591791, 504.31300455, 519.188022  ,
       520.03155195, 535.13855027])
```

In [49]:
```
y_test
```

Out[49]:
```
361    401.033135
73     534.777188
374    418.602742
155    503.978379
104    410.069611
          ...
266    554.003093
23     519.340989
222    502.409785
261    514.009818
426    530.766719
Name: Yearly Amount Spent, Length: 150, dtype: float64
```

In [50]:
```
from sklearn import metrics
from math import sqrt
```

- Import metrics from sklearn module
- From math module import square root

In [51]:
```
print('Mean Absolute Error (MAE)     :',
      metrics.mean_absolute_error(y_test, y_pred), ' ',
      (1./len(y_test))*(sum(abs(y_test-y_pred))))
print('Mean Squared Error (MSE)      :',
      metrics.mean_squared_error(y_test, y_pred), ' ',
      (1./len(y_test))*(sum((y_test-y_pred)**2)))
print('Root Mean Squared Error (RMSE):',
      np.sqrt(metrics.mean_squared_error(y_test, y_pred)), ' ',
      sqrt((1./len(y_test))*(sum((y_test-y_pred)**2))))
```

```
Mean Absolute Error (MAE)     : 8.426091635451497   8.426091635451494
Mean Squared Error (MSE)      : 103.915541207752   103.91554120775203
Root Mean Squared Error (RMSE): 10.19389725314867   10.193897253148672
```

We cannot use classification accuracy to evaluate regression model predictions. Instead, error/accuracy metrics should be used for evaluating predictions made on regression problems

I used three error metrics that are commonly used for evaluating and reporting the performance of a regression model, they are: Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). According to Akshita Chugh (2021), Mean Absolute Error represents the average of the absolute difference between the actual and predicted values in the dataset. It measures the average of the residuals in the dataset. Mean Squared Error represents the average of the squared difference between the original and predicted values in the data set. It measures the variance of the residuals. Root Mean Squared Error is the square root of Mean Squared error. It measures the standard deviation of residuals.

MAE is telling us that on average our predictions for yearly amount spent are off by roughly 8.426 dollars. We can compare from the mean yearly amount spent which is 499.314038 dollars. The error is about 1.69% of mean house yearly amount spent. Our predictions can be considered good from MAE as the percentage error is very low. However, remember MAE is not sensitive to outliers.

In addition, MSE is telling us the average squared predictions for yearly amount spent is 103.916. We get a squared unit outpur after calculating MSE. If outliers present in the dataset, most of the outliers will be penalized and the calculated MSE is bigger. In short, MSE is not robust to outliers which were an advantage in MAE.

Furthermore, RMSE is roughly $ 10.194. This means the error is about 2.04% of mean house yearly amount spent. As it can be seen, this value is higher than MAE. RMSE can still be acceptable, it all depends on us. When large errors are present and affect the model's performace, it is much more useful to use RMSE. The absolute value of the error will be avoided by RMSE and this is beneficial in many mathematical calculations

For all of these three metrics, the lower the value, better is the performance of the model.

In [52]:
```
linear_model.score(X_test, y_test)
```

Out[52]:
```
0.9808757641412774
```

0.98088 represents our linear regression model are 98.088% accurate

CONCLUSION

Conclude your assignment by stating which variable is more important for increasing the yearly-amount-spent. Is it better for the company to focus their efforts on their mobile app experience or their website?

It is shown from the points of the scatter plot and coefficient that Time on App correlates more with Yearly Amount Spent than Time on Website. From the scatter plot of Time on App and Yearly Amount Spent we can see a visible pattern which is increasing, whereas Time on Website and Yearly Amount Spent does not show any pattern at all.

Moreover, the coefficient of Time on App are almost 84 times larger than Time on Website. Revenue obtained through App seems to be more important than Website. Having said that, we should not omit the website directly. The company can improve it to increase its revenue.

In fact, the most important variable for increasing the yearly amount spent is the Length of Membership. Its coefficient is almost 2 times larger than Time on App. But, because the question ask which one is better to be focused on whether on their mobile app or their website. Therefore, the company is better to focus their efforts on their mobile app experience than their website.

CONCLUSION

To conclude, I can already conduct predictive analytics, through machine learning using Python in the Jupyter Notebook environment. I am also able to finish the objectives, such as: read and describe the data using basic statistics, split the dataset into training and testing, conduct binary classification using Random Forest and Decision Tree, implement linear regression for prediction, communicate the output of my analysis and experience independent model evaluation through reporting the metrics.

Now, I have already had experience with the Collect, Wrangle, Analyse and Present process that is core to the role of a Data Scientist.

Reference List

- IBM Cloud Education. (2020, August 19). *Supervised Learning*. IBM. https://www.ibm.com/cloud/learn/supervised-learning
- Petersson, D. (2021, March 26). *supervised learning*. SearchEnterpriseAI. https://searchenterpriseai.techtarget.com/definition/supervised-learning
- Techopedia. (2020, July 1). *Labeled Data*. Techopedia.Com. https://www.techopedia.com/definition/33695/labeled-data
- Techopedia. (2018, May 3). *Test Set*. Techopedia.Com. https://www.techopedia.com/definition/33279/test-set
- Label Your Data. (2021, February 18). *Machine Learning and Training Data: What You Need to Know*. Label Your Data Team. https://labelyourdata.com/articles/machine-learning-and-training-data
- Applause App Quality, Inc. (2021, August 26). *Training, Validation and Testing Data Explained*. Applause. https://www.applause.com/blog/training-data-validation-data-vs-test-data
- Sharma, A. (2020, May 12). *Decision Tree vs. Random Forest - Which Algorithm Should you Use?* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/
- Frost, J. (2021, August 25). *How to Interpret P-values and Coefficients in Regression Analysis*. Statistics By Jim. https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/
- Brownlee, J. (2021, February 15). *Regression Metrics for Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/regression-metrics-for-machine-learning/
- Chugh, A. (2020, December 11). *MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better?* Medium. https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e
- S. (2019, June 3). *MAE vs MSE vs RMSE*. ZeroSpectrum. http://zerospectrum.com/2019/06/02/mae-vs-mse-vs-rmse/
- S. (2021, August 10). *What is Mean Squared Error, Mean Absolute Error, Root Mean Squared Error and R Squared?* Studytonight. https://www.studytonight.com/post/what-is-mean-squared-error-mean-absolute-error-root-mean-squared-error-and-r-squared