FIT1043 Introduction to Data Science

Assignment 2

Renee Yeo Shu Ting 33518904

1. Introduction

The purpose of this report is to investigate, visualise data and build machine learning models from the data provided by the finance company using Python. We are going to use dataset named "FIT1043-Credit-Scores-Dataset.csv" to work on this assignment. It consists of data of basic bank details and credit information in it. We are to create an automated system that categorises people based on their credit scores in order to eliminate human labour.

The report's rough outline is as follows:

1. Introduction
2. Supervised Learning
3. Classification
4. Kagggle Submission
5. Conclusion

In [1]:
```
import os
import pandas as pd
```

In [2]:
```
os.getcwd()
```

Out[2]:
```
'/Users/reneeyeo/Desktop'
```

In [3]:
```
os.chdir('/Users/reneeyeo/Desktop/Assignment2s datasets-20230418')
```

In [4]:
```
# Load csv file into Pandas DataFrame
data = pd.read_csv('FIT1043-Credit-Scores-Dataset.csv')
```

In [5]:
```
# The first 5 rows in the dataset
data.head()
```

Out[5]:

	ID	Month	Age	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_
0	106981	8	41	2	14619.585	1005.298750	
1	108774	1	28	12	70883.440	5663.953333	
2	111896	3	29	12	14395.830	1027.652500	
3	32731	2	25	1	11189.065	1159.422083	
4	128760	7	37	3	78956.730	6523.727500	

5 rows × 24 columns

In [6]:
```
# The last 5 rows in the dataset
data.tail()
```

Out[6]:

	ID	Month	Age	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_B
2095	148811	6	42	8	82154.92	6753.243333	
2096	136926	1	27	15	152104.68	12603.390000	
2097	49566	1	40	4	129569.52	10831.460000	
2098	27815	6	21	2	69506.16	5868.180000	
2099	56408	3	41	10	27392.76	2556.730000	

5 rows × 24 columns

In [7]:
```
# Find for number of data instances and variables in the dataset
data.shape
```

Out[7]:
```
(2100, 24)
```

In [8]:
```
# Prints out random 6 rows of data
data.sample(6)
```

Out[8]:

	ID	Month	Age	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_B
1784	23928	5	23	1	48780.56	3835.046667	
897	143130	1	24	2	21206.48	1631.206667	
1961	45358	5	23	11	18138.00	1464.500000	
2009	123504	7	17	9	40944.82	3588.068333	
845	58842	1	51	2	171292.56	12574.940070	
1209	75059	6	35	3	17263.94	1621.661667	

6 rows × 24 columns

In [9]:
```
# Prints out the different data types in the dataset
data.dtypes
```

Out[9]:
```
ID                          int64
Month                       int64
Age                         int64
Occupation                  int64
Annual_Income             float64
Monthly_Inhand_Salary     float64
Num_Bank_Accounts           int64
Num_Credit_Card             int64
Interest_Rate               int64
Num_of_Loan                 int64
Delay_from_due_date         int64
Num_of_Delayed_Payment      int64
Changed_Credit_Limit      float64
Num_Credit_Inquiries        int64
Credit_Mix                  int64
Outstanding_Debt          float64
Credit_Utilization_Ratio  float64
Credit_History_Age          int64
Payment_of_Min_Amount       int64
Total_EMI_per_month       float64
Amount_invested_monthly   float64
Payment_Behaviour           int64
Monthly_Balance           float64
Credit_Score                int64
dtype: object
```

2. Supervised Learning

Supervised machine learning is a type of machine learning where the model is trained on a dataset that includes both input and the desired output. The datas are labelled and the algorithms learn to predict the output from the input data. The goal is to approximate the mapping function so well that when you have new input data (e.g x), you can predict the output variable (e.g y) for that data.

The notion of labelled data, and the training and test datasets:

The labelled data are datas that has been assigned to a label, (e.g. images, text, files, videos etc). They provide informative context so that a machine learning model and learn from it and produce the desired outputs. For example, if you are truing to train a model to classify images of cats and dogs, the labels would be "cat" and "dog".

Training datasets are datasets that is used to train the model. They are a subset of the original data that is used to train the machine learning model, whereas testing datasets are datasets that is used to evaluate the model's performance. They could check the accuracy of the model which is also an indication of how well the model will perform on new data.

In addition, training datasets is generally larger in size compared to testing datasets.

In [10]:
```
# Separate the features and the label
features = data.drop(['Credit_Score', 'Month', 'Age', 'Occupation', 'Annu
label = data['Credit_Score']
```

Dropping out the unwanted features improves the QWK score on kaggle.

In [11]:
```
from sklearn.model_selection import train_test_split

# Split the data for training and testing
X_train, X_test, y_train, y_test = train_test_split(features, label, test
```

3. Classification

There are two types of classification: binary classification and multi-class classification.

Binary Classification:

Binary classification is a process of classification in which a given data is being classified into classes. It's goal is to predict one of the two possible outcomes (e.g. Yes/No)

Multi-class Classification:

Multi-class classification is the task of categorising elements into various classes. It's goal is to predict one of more than two outcomes.

Difference between binary and multi-class classfication:

Binary classification only classifies at most of two classes' objects while multi-class classification can have any number of classes in it whereas it can classifies more than two classes' objects.

Describe what you understand from this need to normalise data:

Normalizing is a process of organizing data in the dataset. It can help the database to be more flexible by elimating redundancy and inconsistent data. Every feature in the dataset should be normalised since the distance will be greatly impacted if one of them has a wide range of values. One advantages of normalizing data is to be able to create a clear visual to visualise datasets.

The purpose of normalizing data:

1. reduce data redundancy
    - duplicate data can lead to errors and make it difficult to keep track of changes

1. improve data accuracy
    - can help to ensure data is entered consistently and errors are caught early

1. easier to use
    - it helps to create a consistent data structure that is easy to understand and query

In [12]:
```
from sklearn.preprocessing import StandardScaler

# Scaling of the data
scaler = StandardScaler()

# Fit and transform the data
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Describe SVM

Support vector machines, often known as SVMs, are a form of machine learning technique that may be used for classification, regression, and outlier identification. They are based on the concept of locating a hyperplane that divides the data points into two groups. Support vectors are data points that are close to the hyperplane.

Support vector machines are different with linear regression in a few ways:

1. SVMs can be used for non-linear problems while linear regression can't.
    - SVMs can employ kernel functions to map data into a higher-dimensional space, allowing the problem to become linear.

1. SVMs can handle outliers better than linear regression
    - SVMs don't fit a line or plane through the data points. Instead, they find a hyperplane that separates the data points into two classes

1. SVMs needs longer training time while linear regression needs shorter training time

1. Prediction time for SVMs is longer while prediction time for linear regression is shorter

Explain kernel in SVM

A kernel is a function that helps us to map data points using a hyperplane from a lower dimensional space to a higher dimensional space. It works without increasing the computational cost, and make the problem easier to solve.

There are different types of kernels that can be used for example like:

1. Linear kernel ('linear')
2. Polynomial kernel ('polynomial')
3. Radial basis function kernel ('rbf')

In [13]:
```
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
import joblib

# Initialise the models
clf1 = RandomForestClassifier(n_estimators = 150, random_state = 42)
clf2 = SVC(kernel = 'rbf', C = 1, gamma = 0.1)

# Ensemble the voting classifier
ensemble = VotingClassifier(estimators=[('rf', clf1), ('svc', clf2)], vot

# Train the model using the training sets
ensemble.fit(X_train, y_train)
```

Out[13]:
```
VotingClassifier(estimators=[('rf',
                              RandomForestClassifier(n_estimators=150,
                                                     random_state=42)),
                             ('svc', SVC(C=1, gamma=0.1))])
```

I am using ensemble method to ensemble the models in order to achieve higher score of agreement.

In [14]:
```
# Predict the response for test dataset
y_pred = ensemble.predict(X_test)
```

In [15]:
```
from sklearn.metrics import confusion_matrix

# Compute the confusion matrix for the test data
conf_mat = confusion_matrix(y_test, y_pred)

# Display the confusion matrix
print(conf_mat)
```

```
[[116  43   6]
 [ 46 185  31]
 [  3  50  45]]
```

Confusion Matrix:

A confusion matrix is a table that is used to assess the effectiveness of a classification model. The dimensions are equal to the number of classes we have, in here we have 3 classes, hence it is a 3x3 matrix.

A confusion matrix has four different categories:

- True Positive (TP)
- True Negative (TN)
- False Positive (FP)
- False Negative (FN)

It could be used to compute measures like as accuracy, precision, and recall. Overall, the confusion matrix is a tool for evaluating the model's performance, since it helps to discover the parts that are functioning well and those that require development.

Quadratic Weighted Kappa (QWK):

The quadratic weighted kappa (QWK) statistic measures the degree of agreement between two raters. It could take into account of the possibility of chance agreement. QWK is often used to evaluate the performance of predicted scores in datasets. It ranges -1 to 1 where -1 is no agreement and 1 is complete agreement.

In [16]:
```
from sklearn.metrics import cohen_kappa_score

# Calculate the QWK score
score = cohen_kappa_score(y_test, y_pred, weights = 'quadratic')
print(f"QWK: {score:.3f}")
```

```
QWK: 0.579
```

Explanation of QWK Score:

From the QWK score of my predicted datasets, I can know that the agreement could be at moderate mode, where more that half of the predicted data agrees.

Kaggle Submission

In [17]:
```
# Read the competition data file
submission_dataset = pd.read_csv("FIT1043-Credit-Scores-Submission.csv")
```

In [18]:
```
# Sets features
submission_dataset = submission_dataset.drop(['Month', 'Age', 'Occupation

# Scale the data
X = scaler.transform(submission_dataset)

# Predict the data
new_y_pred = ensemble.predict(X)
```

In [19]:
```
# Create a new dataset to predict
submission_results = pd.DataFrame({'ID': submission_dataset['ID'], 'Credi

# Save data to a CSV file
submission_results.to_csv('33518904-ReneeYeo-v43.csv', index = False)
```

Conclusion

In conclusion, this assignment has given me a great opportunity to learn how to build a machine learning model and how they can be use to solve real-world problems. I have learn different terms like kernel and QWK that is not taught in the lectures, and it was really interesting.