Support Vector Machine

Tan Kar Min

Chapter 1 Prerequisite

1.1 Importing libraries and dataset

In [2]:
```
import pandas as pd
import numpy as np
import sklearn
import imblearn
```

In [32]:
```
dataset = pd.read_csv("Training dataset.csv")
```

1.2 Overview of Dataset

In [33]:
```
dataset.shape
```

Out[33]:
```
(2100, 24)
```

The given dataset contains 2100 observations and 24 variables, out of which 23 are features and 1 is the label, representing the credit score. The features include demographic information about bank users such as age, occupation, annual income, monthly inhand salary, as well as credit-related data such as the number of credit cards, credit mix, and outstanding debts. The label, credit score, is divided into three categories, which are 1-poor, 2-standard and 3-good. The first 10 records of the dataset provide a brief overview of these variables:

In [30]:
```
dataset.head(10)
```

Out[30]:

	ID	Month	Age	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate	Num_of_Loan	...	Credit_Mix
0	106981	8	41	2	14619.585	1005.298750	7	7	19	1	...	2
1	108774	1	28	12	70883.440	5663.953333	4	4	10	3	...	2
2	111896	3	29	12	14395.830	1027.652500	8	8	28	7	...	1
3	32731	2	25	1	11189.065	1159.422083	6	3	15	3	...	2
4	128760	7	37	3	78956.730	6523.727500	7	3	14	2	...	2
5	151390	5	50	11	21167.555	1829.962917	2	7	9	3	...	3
6	69108	7	43	14	44964.220	3898.018333	6	7	15	3	...	2
7	26275	7	50	13	140390.320	11888.193330	5	2	4	3	...	3
8	139554	1	29	6	54284.940	4673.745000	7	8	30	7	...	1
9	99702	1	18	3	19375.760	1633.646667	4	3	6	1	...	2

10 rows × 24 columns

In [31]:
```
dataset.describe()
```

Out[31]:

	ID	Month	Age	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate	Num
count	2100.000000	2100.000000	2100.000000	2100.000000	2100.000000	2100.000000	2100.000000	2100.000000	2100.000000	21
mean	83333.452381	4.411429	33.197143	7.927619	51244.805155	4255.554221	5.407143	5.576667	14.532857	
std	41013.880408	2.279068	10.754180	4.325104	38801.235929	3226.388893	2.553747	2.099327	8.736433	
min	10033.000000	1.000000	14.000000	1.000000	7011.685000	319.556250	0.000000	0.000000	1.000000	
25%	48108.000000	2.000000	24.000000	4.000000	19696.500000	1641.972917	4.000000	4.000000	7.000000	
50%	81490.000000	4.000000	33.000000	8.000000	37087.920000	3101.737500	6.000000	5.000000	13.000000	
75%	118895.250000	6.000000	42.000000	12.000000	73327.380000	6118.762500	7.000000	7.000000	20.000000	
max	155610.000000	8.000000	56.000000	15.000000	179987.280000	15101.940000	11.000000	11.000000	34.000000	

8 rows × 24 columns

The summary statistics of the dataset indicate that the average value of each feature varies significantly from one another. Some features, including outstanding debt, annual income, and monthly in-hand salary, have a large standard deviation. The count of all features is 2100, which suggests that there are no incomplete entries in the dataset.

Chapter 2 Supervised Learning

2.1 Introduction to supervised machine learning

Supervised machine learning is a type of machine learning where the algorithm is trained on a labelled dataset [1]. Labelled data refers to the dataset that contains both the input variables and the corresponding output variables (label). In supervised learning, the data is divided into input variables (features) and output variables (labels). The algorithm predicts the output variable of new unseen data based on the feature variables by learning on patterns in the labelled training dataset [1].

Supervised learning entails two distinct types of learning: classification and regression. Regression is used when the output is continuous. By contrast, classification is used when the output is categorical. For this particular task, we will be using a classification supervised machine learning model to predict and categorize individuals into specific credit score brackets.

To train a supervised learning algorithm, it is important to split the labelled dataset into a training set and a test set. The training set is used to train the model, and the test set is used to evaluate the performance of the trained model on new, unseen data [1]. The training set is typically larger than the test set, and the data is randomly sampled to avoid bias.

2.2 Separating the features and the label

Prior to building the model, the dataset is separated into an input dataset (basic bank details and credit-related information) and a corresponding output or label dataset (credit score).

Certain features that were considered insignificant in determining an individual's credit score were eliminated. The decision to exclude these features were based on their correlation with the credit score, with those having a correlation coefficient below 0.1 being excluded. Additionally, the monthly inhand salary was removed as having both annual income and monthly inhand income seems redundant.

In [ ]:
```
#examining the correlation of each feratures with credit score
corr=dataset.corr()
corr.iloc[:, -1]
```

Out[31]:
```
ID                         -0.026606
Month                      -0.005878
Age                         0.126065
Occupation                 -0.028465
Annual_Income               0.221317
Monthly_Inhand_Salary       0.216986
Num_Bank_Accounts          -0.368232
Num_Credit_Card            -0.413036
Interest_Rate              -0.474586
Num_of_Loan                -0.373984
Delay_from_due_date        -0.409188
Num_of_Delayed_Payment     -0.372057
Changed_Credit_Limit       -0.168301
Num_Credit_Inquiries       -0.426879
Credit_Mix                  0.484907
Outstanding_Debt           -0.379331
Credit_Utilization_Ratio    0.077528
Credit_History_Age          0.395466
Payment_of_Min_Amount      -0.376792
Total_EMI_per_month         0.015477
Amount_invested_monthly     0.179008
Payment_Behaviour          -0.106978
Monthly_Balance             0.225686
Credit_Score                1.000000
Name: Credit_Score, dtype: float64
```

In [11]:
```
#splitting the dataset and excluding features
X = dataset.drop(['Credit_Score','ID', 'Month', 'Occupation', 'Monthly_Inhand_Salary', 'Credit_Utilization_Ratio', 'Total_EMI_per_m
y = dataset.iloc[:, -1].values # Label Data
```

2.3 Spliting dataset for training and testing

The train_test_split() function from sklearn.model_selection is used to split the dataset into 80% training dataset and 20% testing dataset randomly.

In [12]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 42)
```

It was observed that the number of observations in each credit score categories are not balance in the training dataset. To avoid bias, the RandomOverSampler function is used to over-sample the minority classes by picking samples at random with replacement [2].

In [13]:
```
# Count the frequency of each credit score categories in training dataset
y_train_series = pd.Series(y_train)
value_counts = y_train_series.value_counts()
print(value_counts)
```

```
2    900
1    506
3    274
dtype: int64
```

In [14]:
```
from imblearn.over_sampling import RandomOverSampler

# Define the oversampler
oversampler = RandomOverSampler(random_state=42)

# Fit and apply the oversampling to the data
X_train, y_train= oversampler.fit_resample(X_train, y_train)
```

Chapter 3 Classification

3.1 Introduction to binary and multi-class classification

The classification machine learning model is further divided into two categories, binary and multi-class classification. The main difference between binary and multi-class classification is the number of possible classes or categories that the classifier can predict. In binary classification, there are only two possible outcomes, while in multi-class classification, there are three or more possible outcomes.

In binary classification, the input data is analyzed and classified into one of two possible classes, such as "Yes" or "No" or "True" or "False".

In multi-class classification, the input data is analyzed and classified into one of three or more possible classes, such as "small," "medium," or "large," or "low," "medium," or "high."

In this assignment, the target outcome of the intellegent system is divided into 3 different credit score categories (poor, standard or good). Hence, a multi-class classification model will be employed.

3.2 Feature Scaling or Normalization

Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without feature scaling or normalization. This is particularly important in models that consider the distance between observations [3]. Most classifiers use the Euclidean distance to measure the distance between two points. If one feature has a wide range of values, it can dominate the distance calculation [3]. To ensure that each feature has an equal impact on the final distance, it is important to normalize the range of all features, so that they are comparable and have the same scale [3].

In the context of this assignment, it is crucial to perform feature scaling as we are using Support Vector Machine (SVM) for classification. The SVM algorithm needs to determine a decision boundary between different classes in the dataset [4]. This boundary is chosen to maximize the distance from the nearest points from different classes[4]. Therefore, the distance between data points plays a significant role in determining the decision boundary of the SVM classifier. Furthermore, based on the summary statistics of the dataset, it is evident that the features are on different scales and range. This indicates the need for scaling or normalization of the data to ensure proportionate contribution of each feature to the final distance.

A built in function StandardScaler() from sklearn.preprocessing is used to normalize the dataset.

In [15]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train) #scale it to have a mean of 0 and a certain standard deviation
X_test = scaler.transform(X_test) # apply the mean and standard deviation to the testing dataset
```

3.3 Support Vector Machine (SVM)

Support Vector Machines (SVMs) are a type of machine learning algorithm used for supervised learning. It is generally used as a classifier, but it can also be employed in regression problems [4]. SVM is commonly used in various applications such as face detection, classification of genes, and handwriting recognition[4].

In linear regression, the algorithm aims to find the best-fit line that explains the relationship between the input and output variables. The relationship between the input and output variables is assumed to be linear, and the algorithm tries to minimize the sum of squared errors between the predicted and actual values. Unlike SVM, linear regression cannot handle classfication problems well. In contrast, SVM algorithm aims to create a hyperplane that separates the data into different classes, and the algorithm achieves this by finding the points closest to the line from all the classes, which are known as support vectors [4]. The distance between the hyperplane and the support vectors is known as the margin, and the algorithm aims to find the hyperplane that maximizes this margin [4].

Compared to linear regression, SVM are more suitable for classification tasks and can handle non-linearly separable data by transforming the low-dimensional input space to a higher-dimensional feature space using a kernel function [4]. The kernel function maps the input data to a higher dimensional space, making it more efficient and accurate in separating complex problems [4].

Here are some commonly used kernel:

1. Linear Kernel: A linear kernel is a simple kernel that can be used as a dot product of two given observations. It is useful for linearly separable data, where the decision boundary is a straight line [4].
2. Polynomial Kernel: The polynomial kernel is a more generalized form of the linear kernel. It can handle curved or nonlinear input spaces by transforming them into a higher dimensional space. A higher degree polynomial can capture more complex relationships between the input variables [4].
3. Radial Basis Function (rbf) Kernel: The rbf kernel is a popular kernel function used in SVM classification. It maps the input space into an infinite dimensional space, which makes it useful for handling non-linearly separable data. The rbf kernel function is of the form K(x, xi) = exp(-gamma * sum((x – xi^2)), where gamma is a parameter that controls the shape of the decision boundary. The rbf kernel is useful for capturing complex relationships between the input variables [4].

The SVM model for this assignment were built with a rbf kernel:

In [16]:
```
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='rbf', C=0.25, gamma='auto', random_state=0)

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

Out[16]:
```
SVC(C=0.25, gamma='auto', random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Chapter 4 Prediction

The SVM model built was tested agaisnt the testing dataset:

In [17]:
```
#Predict the output for test dataset
y_pred = clf.predict(X_test)
```

4.1 Confusion Matrix

Confusion matrix for visualising the performance of the model:

In [18]:
```
from sklearn.metrics import confusion_matrix

#generate confusion matrix
cm = confusion_matrix(y_test, y_pred)

#add labels to the matrix for better visualisation
labels = [1,2,3]
cm = pd.DataFrame(cm, index=labels, columns=labels)
cm
```

Out[18]:

	1	2	3
1	94	14	24
2	43	114	49
3	3	13	66

The rows of the confusion matrix represent the actual value while the columns represent predicted values [5].

Hence, accordinng to the confusion matrix:

Credit score 1:

Among the 140 credit score 1 predicted by the model, 94 were actually score 1, while 43 were score 2 incorrectly predicted to be score 1 and 3 were score 3 incorrectly predicted to be score 1. 14 individuals of score 1 were falsely predicted as score 2 and 24 were falsely predicted as score 3.

Precision: 94 / (94 + 43 + 3) = 0.671, Recall: 94 / (94 + 14 + 24) = 0.712

Credit score 2:

Among the 141 credit score 2 predicted by the model, 114 were actually score 2, while 14 were score 1 incorrectly predicted to be score 2 and 13 were score 3 incorrectly predicted to be score 2. 43 individuals of score 2 were falsely predicted as score 1 and 49 were falsely predicted as score 3.

Precision: 114 / (14 + 114 + 13) = 0.809, Recall: 114 / (43 + 114 + 49) = 0.553

Credit score 3:

Among the 139 credit score 3 predicted by the model, 66 were actually score 3, while 24 were score 1 incorrectly predicted to be score 3 and 49 were score 2 incorrectly predicted to be score 3. 3 individuals of score 3 were falsely predicted as score 1 and 13 were falsely predicted as score 2.

Precision: 66 / (24 + 49 + 66) = 0.475, Recall: 66 / (3 + 13 + 66) = 0.805

The accuracy counted based on the confusion matrix are as below: (94 + 114 + 66) / (94 + 14 + 24 + 43 + 114 + 49 + 3 + 13 + 66) = 0.652

4.2 Quadratic Weighted Kappa (QWK)

Quadratic Weighted Kappa (QWK) is a statistical measure used to assess the agreement between two outcomes who assign scores or ratings to a set of items [6]. The QWK coefficient is calculated using the observed agreement between the algorithm's prediction and true labels, taking into account the possibility of agreement by chance [6]. It ranges from -1 to 1, with values closer to 1 indicating a higher level of agreement. -1 indicates a complete disagreement, 0 indicates a random agreement and 1 indicates a complete agreement [6].

QWK takes into account not only the number of ratings that agree but also the degree of disagreement between the ratings [6]. It assign different weights based on the differences between actual and predicted values, which is particularly useful when the scale being used has many categories [6].

The QWK calculation involves four steps. First, a histogram matrix is created that shows a rating of i (actual) that received a predicted value j. [6]. Second, a matrix of weights is constructued by calculating the difference between actual and predicted values. Third, a histogram matrix of expected rating is calculated, assuming no correlation between rating scores [6]. Finally, the QWK is calculated using the three histograms, with a higher QWK value indicating better agreement between the two sets of ratings [6].

In [19]:
```
from sklearn.metrics import cohen_kappa_score

qwk_score = cohen_kappa_score(y_test, y_pred, weights='quadratic')

print("QWK score:", qwk_score)
```

```
QWK score: 0.5393312717433321
```

It was stated that a QWK score of 0.4-0.6 is considered as having a good agreement [6]. Our model gives a QWK score of 0.539, suggesting that the model has a relative good performance but not yet optimal.

Chapter 5 Kaggle Submission Preparation

The “FIT1043-Credit-Scores-Submission.csv” file consists of 900 rows of features data with no labels (credit score). The SVM model built was used to predict the credit score for individuals in this dataset. The predicted output will be submitted to Kaggle as a part of the in class competition. The output is evaluated using QWK in the Kaggle competition.

In [34]:
```
#import files
#feature dataset
subdataset = pd.read_csv("Prediction dataset.csv")
#submission file template
submissionfile = pd.read_csv("Submission template.csv")
```

In [35]:
```
#exclude unused features
Xsub = subdataset.drop(['ID', 'Month', 'Occupation', 'Monthly_Inhand_Salary', 'Credit_Utilization_Ratio', 'Total_EMI_per_month'], a
#scale the dataset
Xsub = scaler.transform(Xsub) # apply the mean and standard deviation to the dataset
```

In [36]:
```
#predict credit score using the SVM model built
ysub_pred = clf.predict(Xsub)
```

In [37]:
```
#export the predicted credit score to the submission file
submissionfile['Credit_Score']=ysub_pred
submissionfile.to_csv("Prediction result.csv", index=False)
```

Chapter 6 Conclusion

To summarize, this assignment involved developing a multi-class supervised machine learning model that categorizes bank users into three credit score categories - poor (1), standard (2), and good (3) - based on their bank details and credit-related information. The model was built using the support vector machine algorithm with a rbf kernel. The evaluation of the final model using the testing dataset achieved an accuracy of 0.652 and a QWK score of 0.539. The Kaggle submission prediction achieved a QWK score of 0.556. Hence, the SVM model built in this assignment was suggest to have a moderate performance.

Chapter 7 Reference

[1] javatpoint (2022). Supervised Machine learning - Javatpoint. [online] www.javatpoint.com (http://www.javatpoint.com). Available at: https://www.javatpoint.com/supervised-machine-learning (https://www.javatpoint.com/supervised-machine-learning).

[2] Brownlee, J. (2020). Random Oversampling and Undersampling for Imbalanced Classification. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/ (https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/).

[3] Roy, B. (2020). All about Feature Scaling. [online] Medium. Available at: https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35 (https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35).

[4] Navlani, A. (2019). Support Vector Machines with Scikit-learn Tutorial. [online] datacamp. Available at: https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python (https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python).

[5] Bharathi (2021). Confusion Matrix for Multi-Class Classification. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/ (https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/).

[6] BANERJEE, P. (2020). Simple Explanation of Quadratic Weighted Kappa. [online] kaggle.com. Available at: https://www.kaggle.com/code/prashant111/simple-explanation-of-quadratic-weighted-kappa/notebook (https://www.kaggle.com/code/prashant111/simple-explanation-of-quadratic-weighted-kappa/notebook).