

UNIVERSITI TUNKU ABDUL RAHMAN

LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE

UECS3213/ UECS3453/ UECS3483 DATA MINING

GROUP ASSIGNMENT

Jan 2024 Trimester

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NO.** | **STUDENT NAME** | **PROGRAMME** | **PRACTICAL GROUP** | **STUD**  **ID** |
| **1.** | ALWIN CHONG WEI SHEN | SE | P3 | 2300327 |
| **2.** | ERIC LIM JUN KIT | AM | P3 | 2105156 |
| **3.** | NG WENG LAM | SE | P2 | 2207666 |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **No.** | **Contents** | **Page No.** |
| 1.0 | Introduction | 1 |
| 2.0 | Methodology | 1 – 3 |
| 3.0 | Models Training, Analysis & Selection | 3 – 5 |
| 4.0 | Prediction on New Loan Applicant | 6 |
| 5.0 | Conclusion | 6 |
|  | Reference | 7 |

1. **Introduction**

Objective – To predict 20 new loan applicants whether they will at risk of getting loan default.

**2.0 Methodology**

Software Requirements: **Python/ Jupyter Notebook**

* 1. Read Dataset File

Before we start, we first need to **read the CSV files** to ensure the connection between Python and the CSV files.

* 1. Data Splitting

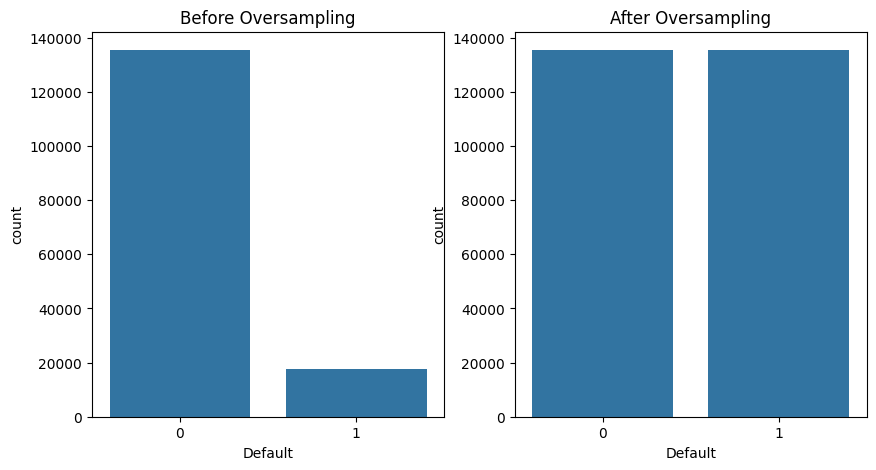
The **purpose of data splitting is to check the performance and accuracy of our models** using different datasets, such as training, testing, and validation data. However, before doing that, we **set a random seed at 42 to ensure randomness** of the data splitting. We then split our dataset into **60% training data, 20% test data, and 20% validation data**. The reason for using the **same size for the test and validation** datasets is to **prevent** two common model issues which are **overfitting and underfitting**.

* 1. Data Exploration & Data Preprocessing

To explore the dataset’s structure, we **check the first five rows of the dataset**. Then, we **check the dimensions** of the train dataset to **determine their split size**. This allows us to **verify that the data is split as intended**, with 60% for training, 20% for validation, and 20% for testing~~.~~ Moving on, we **checked for the data types for each column** to know well about the data types of the values. Next, we **examined the column names to understand** **features** in the dataset. This provides insights into the type of data that we are working with and helps us plan our data preprocessing steps. Moving on, we **dropped ‘LoanID’ column which is not useful**. After that, we **checked if there are any missing values and duplicate rows** from the dataset. If found, then we would need to remove it. The reason of removing is the **missing values and duplicate rows** **cause some problems, such as inflating of the size of dataset, results when interacting with model might be distorted, or assumptions of some of the tests might be violated**. The results that we observed are, the datasets are perfect and do not consist of any missing values and duplicate rows. Moving on, we **converted the categorical data to numerical form**. This is essential in data science for several reasons, such as facilitate the use of machine learning algorithms, enable mathematical operations, improve features representation, enhance dimensionality reduction, enable distance calculation, enhanced model performance and provide flexibility in data analysis (Ratna, 2023). With this method, we will have a **higher accuracy of predictive models and better forecasting of the target variable**. On the column of **Loan Term**, we had **reduced scale** by converting **months to years**. Scaling facilitates meaningful comparisons between features, improves model convergence, and prevents certain features from overshadowing others based solely on their magnitude (Bhandari, 2020).

Moving on, we **normalized all the numeric data and set into four decimal places**. The goal of **normalization is to transform features to be on a similar scale**, thus it **enhances the performance and training stability of the model** (Google Developers, n.d.). Then, we used **One Hot Encoding** on categorical data that we changed to numerical form previously to improve the performance of our algorithm. By using one hot encoding, we **created dummy variables** for each of the unique **variables in the targeted column**. Each unique variable became a new feature which is also called binary features.

After that, we realized the class imbalance issue on Default by checking the count of 0’s and 1’s. We obtained the count of 0’s is 135408 and 1’s only consists of 17788. Sincethe **data was** **imbalanced**, we used **Synthetic Minority Oversampling Technique (SMOTE) to oversample the dataset**, so that we obtained a balance dataset. After oversampling, we observed that the count of 0’s and 1’s is both 135408. **Therefore, it became more balanced.**

Figure 1: The Distribution of Default

Lastly, we used Variance Inflation Factor (VIF) for feature reduction with an assumption, which any features with VIF values more than 3 will be considered strong multicollinearity. In this case, we removed the features that have VIF values more than 3. The reason of removal of those features is to avoid **multicollinearity as it might lead several problems** after when model is built to test, such as **unstable parameter estimates** and **difficulties in interpreting the effects of individual predictors**.

1. **Models Training, Analysis & Selection**

Table below shows which models are used for training and prediction.

|  |  |  |
| --- | --- | --- |
| Models | Is it used for training and prediction? | |
| Yes/No | If no, why? |
| Decision Tree (DT) | Yes | - |
| Gaussian Naïve Bayes (GausNB) | Yes | - |
| K Nearest-Neighbors (KNN) | Yes | - |
| Logistic Regression (LR) | Yes | - |
| Neural Network (MLP) | Yes | - |
| Random Forest (RF) | Yes | - |
| Support Vector Machine (SVM) | No | * Take a very long training time * Hardware limitation, prolonged training will cause CPU overheating |

Table below shows the result obtained from each model trained and tested on **Validation** data:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Default = 0 | | | Default = 1 | | | Accuracy | AUC  Score |
| Precision | Recall | F1 | Precision | Recall | F1 |
| DT | 0.91 | 0.67 | 0.77 | 0.16 | 0.48 | 0.24 | 0.65 | 0.58 |
| GausNB | 0.93 | 0.77 | 0.84 | 0.23 | 0.52 | 0.32 | 0.74 | 0.72 |
| KNN | **0.94** | 0.61 | 0.74 | 0.19 | **0.71** | 0.30 | 0.62 | 0.72 |
| LR | **0.91** | **0.90** | **0.91** | **0.31** | **0.34** | **0.32** | **0.84** | **0.73** |
| MLP | **0.91** | **0.88** | **0.89** | **0.27** | **0.35** | **0.30** | **0.82** | **0.72** |
| RF | 0.91 | 0.85 | 0.88 | 0.25 | 0.39 | 0.31 | 0.79 | 0.71 |
| SVM | - | - | - | - | - | - | - | - |

Table below shows the result obtained from each model trained and tested on **Test** data:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Default = 0 | | | Default = 1 | | | Accuracy | AUC  Score |
| Precision | Recall | F1 | Precision | Recall | F1 |
| DT | 0.91 | 0.62 | 0.73 | 0.15 | 0.53 | 0.24 | 0.60 | 0.57 |
| GausNB | 0.93 | 0.78 | 0.85 | 0.24 | 0.54 | 0.33 | 0.75 | 0.72 |
| KNN | **0.94** | 0.61 | 0.74 | 0.19 | **0.71** | 0.30 | 0.62 | 0.72 |
| LR | **0.91** | **0.91** | **0.91** | **0.33** | **0.35** | **0.34** | **0.84** | **0.74** |
| MLP | **0.91** | **0.88** | **0.90** | **0.28** | **0.36** | **0.32** | **0.82** | **0.72** |
| RF | 0.92 | 0.83 | 0.87 | 0.25 | 0.43 | 0.32 | 0.79 | 0.71 |
| SVM | - | - | - | - | - | - | - | - |

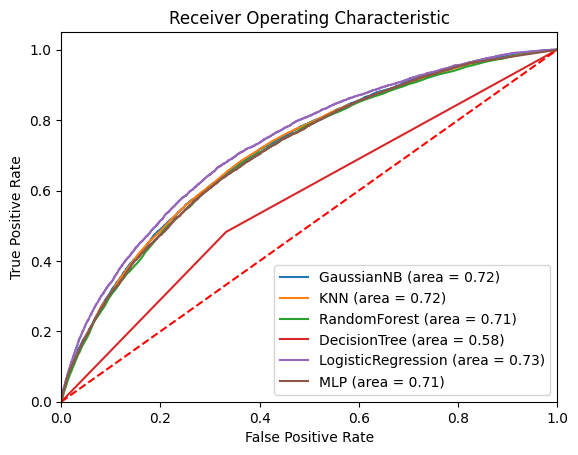
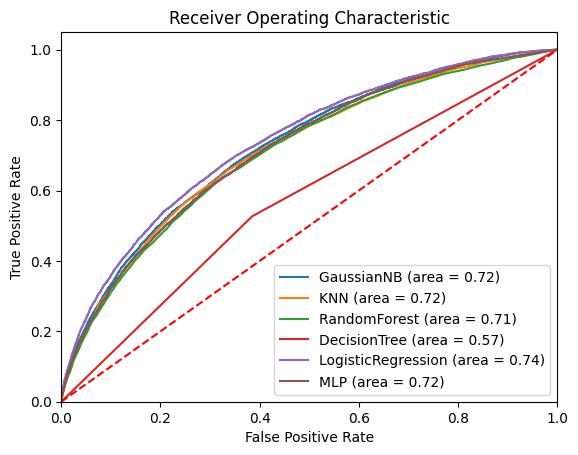


Figure 2: ROC Curve on **Validation** data(left) vs ROC Curve on **Test** data(right)

**Validation data:**

In the non-default **class**, the model with the **highest precision is KNN (0.94)**, while GausNB has the second highest Precision (0.93). **LR** and **MLP** got **0.91** in Precision, lower than KNN. However, **LR (0.90, 0.91)** and **MLP (0.88, 0.89)** perform better with **higher Recall and F1-scores** compared to **KNN (0.61, 0.74)**.

While **in the default class**, **LR (0.31, 0.32) obtained highest Precision and F1-score.** For Recall, KNN (0.71) performs better than LR (0.34). and the models with the greatest F1-Score are **GausNB** **(0.32)** and **LR (0.32)**.

Furthermore, **LR (0.84, 0.73)** obtained the highest accuracy and AUC score comparedto other models.

**Test data:**

In the non-default **class**, the model with the highest precision is **KNN (0.94)**, while GausNB has the second highest Precision (0.93). LR and MLP still get a high precision score of 0.91, and their Recall and F1- scores are higher than KNN, which are **LR (0.91, 0.91)** and **MLP (0.88, 0.90)**.

While in the default **class**, **LR (0.33, 0.34) obtained the highest Precision and F1-score.** For Recall, KNN **(0.71) obtained the highest compared to LR (0.35)**.

Furthermore, **LR (0.84, 0.74)** obtained the highest accuracy and AUC score comparedto other models.

**Conclusion of Analysis:**

All in all, **LR** had the highest **accuracy**, **AUC score**, **decent recall**, and **F1 scores** for **Default 0** even though the **accuracy of KNN** is comparatively **higher than LR for Default 0**. In Default 1, **MLP** and **LR's** **accuracy**, **recall**, and **F1 scores** for **both Default 0** and **1** are **closer**.

Thus, the best model chosen is **Logistic Regression (LR).**

1. **Prediction on New Loan Applicant**

Below is the predicted default based on the model output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LoanID** |  |  |  | **Default** |
| A01 |  |  |  | **0** |
| A02 |  |  |  | **0** |
| A03 |  |  |  | **0** |
| A04 |  |  |  | **1** |
| A05 |  |  |  | **0** |
| A06 |  |  |  | **1** |
| A07 |  |  |  | **0** |
| A08 |  |  |  | **0** |
| A09 |  |  |  | **0** |
| A10 |  |  |  | **0** |
| B01 |  |  |  | **0** |
| B02 |  |  |  | **0** |
| B03 |  |  |  | **0** |
| B04 |  |  |  | **0** |
| B05 |  |  |  | **0** |
| B06 |  |  |  | **1** |
| B07 |  |  |  | **1** |
| B08 |  |  |  | **0** |
| B09 |  |  |  | **0** |
| B10 |  |  |  | **0** |

After we had chosen our best model, we applied it to predict the default value for 20 loan IDs. Among these, **only 4 Loan IDs** were **categorised as having a default**, while **the other Loan IDs** were categorized as not having a default.

Therefore**, all bank loans except A04, A06, B06, and B07** should be approved.

1. **Conclusion**

After training all the models, the best model selected is **Logistic Regression** as it has astonishingly high **precision**, **decent recall**, and **F1 scores** for **Default 0, acceptable scores on precision, recall and F1-score for Default 1 with good accuracy and AUC scores.**

**REFERENCES**

Ratna, A. (2023). Importance of converting categorical data to numerical data (Data science). Medium. Available at: https://medium.com/@ratnaamulya681/importance-of-converting-categorical-data-to-numerical-data-data-science-8a5982f55ed4.

Bhandari, A. (2020). Feature Scaling: Engineering, Normalization, and Standardization (Updated 2024). Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization

Google Developers. (n.d.). Normalization | Machine Learning. Available at: https://developers.google.com/machine-learning/data-prep/transform/normalization.