Default of Credit Card Clients Payment Intentions using Linear Discriminant Analysis, K-Nearest Neighbor and Random Forest Algorithms

Swapna Korepu
13516699
Coventry University
Coventry, England

***Abstract*—This paper investigates the prediction of clients' behaviour in paying off their credit card balance using machine learning algorithms such as Support Vector Machines, K-Nearest Neighbor, Random Forest, Adaptive Boosting, and Extremely Randomized Trees. The analysis is based on the Default Credit Card Clients dataset. Various data preparation techniques were employed to handle class imbalance, encode categorical data, and extract relevant features from the dataset. The study utilizes the Python programming language and leverages machine learning libraries to implement the techniques and algorithms. Performance measures and prediction results were obtained and visually compared and discussed.**

Keywords— Default credit card clients, behaviour prediction, machine learning algorithms, Support Vector Machines, K-Nearest Neighbour, Random Forest, Adaptive Boosting, Extremely Randomized Trees, data preparation techniques, class imbalance, categorical data encoding, feature extraction, Python programming language, performance measures, prediction results, dataset analysis.

I. Introduction

The dataset Default of Credit Card Clients from UCI Machine Learning Repository provides valuable insights into credit card default behaviour of customers. In today's financial landscape, credit card default is a critical issue for credit card companies and financial institutions. Default occurs when a customer fails to make timely payments on their credit card debt, resulting in significant financial implications for both the customer and the lender.

The dataset contains information about various factors that contribute to credit card default, including demographic attributes of customers, their repayment history, and credit card usage patterns. By analysing this dataset, to achieve a deeper understanding of the factors that influence default behaviour and develop effective strategies to mitigate credit risk.

Furthermore, to address the issue of class imbalance in the dataset, techniques like SMOTE (Synthetic Minority Over-sampling Technique), over-sampling, and under-sampling can be employed. These techniques aim to balance the distribution of the classes and improve the overall performance of the models.

Analysing the confusion matrix, classification report, and accuracy scores, to evaluate the performance of the models under different sampling techniques. Additionally, the Receiver Operating Characteristic (ROC) curves and the corresponding Area Under the Curve (AUC) values provide a comprehensive assessment of the model's predictive power and discrimination ability.

II. The Data Set

The Default of Credit Card Clients dataset contains information about credit card clients and their payment behaviour. It consists of 30,000 instances or observations, where each instance represents one credit card account. Each account is associated with various attributes that provide information about the client and their credit card usage. The dataset was obtained from the UCI Machine Learning repository and does not have any missing values.

Each instance in the dataset corresponds to a unique visit or interaction with the credit card company's services. It is important to note that for returning customers who visited the site multiple times, only the first visit from the analysed time is included in the dataset. Subsequent visits are not considered.

The dataset contains 24 attributes or features, which can be categorized into the following groups:

1. Demographic and Credit-related Attributes:
    - LIMIT_BAL: Credit card limit (in New Taiwan dollars)
    - SEX: Gender (1 = male, 2 = female)
    - EDUCATION: Education level (1 = graduate school, 2 = university, 3 = high school, 4 = others)
    - MARRIAGE: Marital status (1 = married, 2 = single, 3 = others)
    - AGE: Age in years
2. Repayment Status Attributes (in months):
    - PAY_0 to PAY_6: Payment status (PAY_0 represents the repayment status in September, PAY_2 represents the repayment status in August, and so on)
3. Bill Statement Amount Attributes (in New Taiwan dollars) for the past six months:
    - BILL_AMT1 to BILL_AMT6: Bill statement amount (BILL_AMT1 represents the bill amount in September, BILL_AMT2 represents the bill amount in August, and so on)
4. Payment Amount Attributes (in New Taiwan dollars) for the past six months:
    - PAY_AMT1 to PAY_AMT6: Amount paid (PAY_AMT1 represents the payment amount in September, PAY_AMT2 represents the payment amount in August, and so on)
5. Target Variable:
    - Default payment next month(Payments): Default payment (1 = yes, 0 = no). This indicates whether the client defaulted on the credit card payment in the following month.

TABLE 1. DATASET FEATURES

No	Description	Type	Categorical Value Range
1	Limit Balance	Numeric	N/A
2	Sex	Categorical	1 = Male, 2 = Female
3	Education	Categorical	1 = Graduate School, 2 = University, 3 = High School, 4 = Others

4	Marital Status	Categorical	1 = Married, 2 = Single, 3 = Others
5	Age	Numeric	N/A
6	Payment Delay for September	Categorical	-1 = Paid on time, 1 = 1 month delay, 2 = 2 months delay, and so on
7	Payment Delay for August	Categorical	-1 = Paid on time, 1 = 1 month delay, 2 = 2 months delay, and so on
8	Payment Delay for July	Categorical	-1 = Paid on time, 1 = 1 month delay, 2 = 2 months delay, and so on
9	Payment Delay for June	Categorical	-1 = Paid on time, 1 = 1 month delay, 2 = 2 months delay, and so on
10	Payment Delay for May	Categorical	-1 = Paid on time, 1 = 1 month delay, 2 = 2 months delay, and so on
11	Payment Delay for April	Categorical	-1 = Paid on time, 1 = 1 month delay, 2 = 2 months delay, and so on
12	Amount of Bill for September	Numeric	N/A
13	Amount of Bill for August	Numeric	N/A
14	Amount of Bill for July	Numeric	N/A
15	Amount of Bill for June	Numeric	N/A
16	Amount of Bill for May	Numeric	N/A
17	Amount of Bill for April	Numeric	N/A
18	Payments	Categorical	0 = No, 1 = Yes

III. Data Preparation

This section covers the data analysis on Class Imbalance refers to an unequal distribution of the target variable (class) within the dataset. It means that the number of instances belonging to one class is significantly higher or lower than the number of instances belonging to the other class.

The dataset Default of Credit Card Clients is a classic example of class imbalance. It consists of information about credit card clients, where the target variable indicates whether a client will default on their credit card payment (default class) or not (non-default class). Typically, the non-default class is more prevalent in such datasets, making it imbalanced.

A. Class Imbalance

Murphy He in the book A Probabilistic Perspective for Machine Learning (2012), the author highlights the issue of class imbalance in machine learning and the importance of addressing it to improve algorithm performance. The book discusses various techniques that can be used to handle class imbalance, including random under-sampling, random over-sampling, and SMOTE over-sampling.

Applying these techniques to the Default of Credit Card Clients dataset from the UCI Machine Learning Repository, it was observed that the dataset contains 30,000 instances, each representing a unique credit card client. The dataset consists of 23 attributes, capturing information such as demographic details, credit history, and payment behaviour. The final attribute indicates whether the client defaulted on their credit card payment (class 1) or not (class 0).

Analysing the dataset, it was found that there is a significant class imbalance. Specifically, there are 22,165 instances classified as non-default (class 0) and only 7,835 instances classified as default (class 1), indicating an imbalanced distribution.

To visually depict the class imbalance, refer to Figure 1, which illustrates the distribution of instances across the two classes. The figure highlights the disproportionate number of instances between the default and non-default classes.

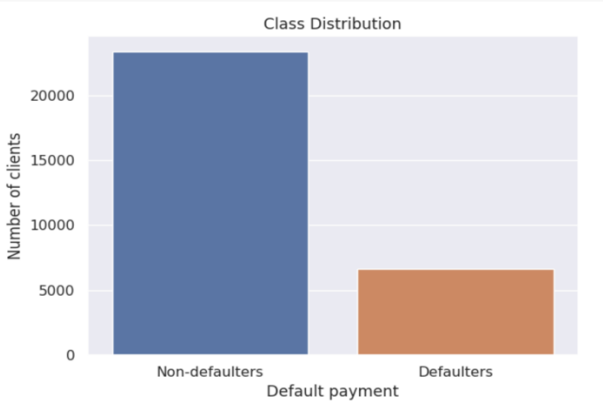

Figure 1. Class Imbalance

To address the class imbalance in the dataset, several techniques can be employed:

1)*Random Under-sampling:*

This technique involves randomly selecting a subset of instances from the majority class (non-default) to match the number of instances in the minority class (default). This is beneficial to balance the class distribution by reducing the dominance of the majority class. However, this approach may result in the loss of potentially useful information present in the majority class instances that are not selected.

2)*Random Over-sampling:*

In this technique, instances from the minority class (default) are randomly duplicated or replicated to increase their representation in the dataset. This helps to balance the class distribution by augmenting the instances of the minority class. However, it may lead to overfitting if the minority class instances are excessively replicated.

3)*SMOTE Over-sampling:*

SMOTE (Synthetic Minority Over-sampling Technique) is a popular technique for handling class imbalance. It generates synthetic samples for the minority class by interpolating the feature space between existing instances. It creates new synthetic instances by considering the feature values of the minority class instances and their nearest neighbours. SMOTE helps to balance the class distribution by increasing the number of instances in the minority class, while also introducing diversity through the generation of synthetic samples. A visual example of SMOTE displayed in Figure 2.

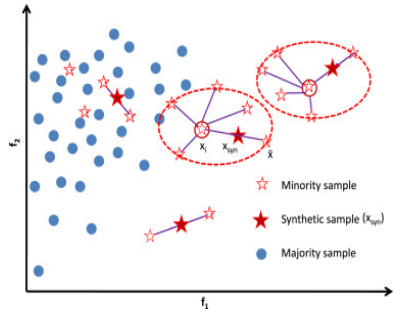

Figure 2. SMOTE Example

B. *Categorical Feature Encoding*

Machine learning algorithms typically require numerical inputs, and categorical data needs to be encoded into a suitable format. In the Default of Credit Card Clients dataset, there are several categorical features that need to be encoded for machine learning analysis.

ENCODING CATEGORICAL FEATURES TO BINARY:

To encode categorical features in the dataset, preferrable to use techniques such as Label Encoding and One-Hot Encoding. Label Encoding assigns a unique numeric label to each category within a feature. For example, the categorical feature "Education" may have categories like "graduate school," "university," "high school," etc. These categories can be encoded as 0, 1, 2, respectively.

One-Hot Encoding, on the other hand, creates binary columns for each category within a feature. Each column represents a category, and a value of 1 indicates the presence of that category, while a value of 0 indicates its absence. This approach avoids any ordinal relationship among categories. For instance, the categorical feature "Marital Status" may have categories like "married," "single," and "others." One-Hot Encoding would create three binary columns, where "married" would be represented as (1, 0, 0), "single" as (0, 1, 0), and "others" as (0, 0, 1).

By applying categorical feature encoding techniques like Label Encoding and One-Hot Encoding, transform the categorical data into a suitable numerical format for machine learning algorithms to process. This ensures that the algorithms can effectively learn from and make predictions based on the encoded categorical features in the dataset.

DUMMY VARIABLE REMOVAL:

Once the categorical feature encoding is completed, the dummy variables are removed to avoid multicollinearity issues and reduce the dimensionality of the dataset. Removing one category does not result in information loss, as it can be inferred from the remaining category columns with zero values. This technique was also applied in the Default of Credit Card Clients dataset.

For example, if we consider the categorical feature "Education" with categories "graduate school," "university," "high school," and "others," after applying One-Hot Encoding, four binary columns representing these categories are created. To remove one category (e.g., "others"), simply drop the corresponding column. The information about "others" can still be captured by the absence of the "graduate school," "university," and "high school" columns, which would all have zero values.

This approach helps to reduce the dimensionality of the dataset by eliminating redundant information and avoids multicollinearity, where one category can be predicted from the others. By removing one category, we maintain the necessary information for machine learning algorithms while keeping the dataset more manageable.

C. *Data Standardization*

Data standardization, also known as data normalization, is a pre-processing step that is often applied to numeric features in a dataset. It involves transforming the values of numeric features to have a standard scale, typically with a mean of 0 and a standard deviation of 1. This process ensures that all the features are on a similar scale, which can be beneficial for certain machine learning algorithms.

FORMULA:

Standardized Value = (Value - Mean) / Standard Deviation

Where:

- "Value" is the original value of the data point.
- "Mean" is the mean (average) of the feature in the dataset.
- "Standard Deviation" is the standard deviation of the feature in the dataset.

In this paper, standardization was completed utilizing python Standard Scaler class from pre-processing module in scikit-learn library.

D. *Correlation Among Features*

This technique involves randomly selecting a subset of instances from the majority class (non-default) to match the number of instances in the minority class (default). This helps balance the class distribution by reducing the dominance of the majority class.

The considered dataset has many attributes, and the presence of strongly correlated features may lead to a drop in the performances of some classification algorithms. Indeed, there are some methods that assume the independence of the predictors and benefit from dimensionality reduction.

Pearson's Correlation Coefficient ρ helps us to find out the relationship between two variables, measuring the strength of association between them. It is calculated as the covariance between two features X and Y (numerator) divided by the product of their standard deviations (denominator):

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y}$$

The value of Pearson's Correlation Coefficient can be between −1 and +1.

- +1 means that they are highly correlated,
- 0 means no correlation,
- −1 means that there is a negative correlation (inverse proportion).

Note that the Pearson's correlation Coefficient is only able to capture linear trends, hence it might lead to a value of 0 for strongly non-linearly correlated variables (e.g. quadratic trend).

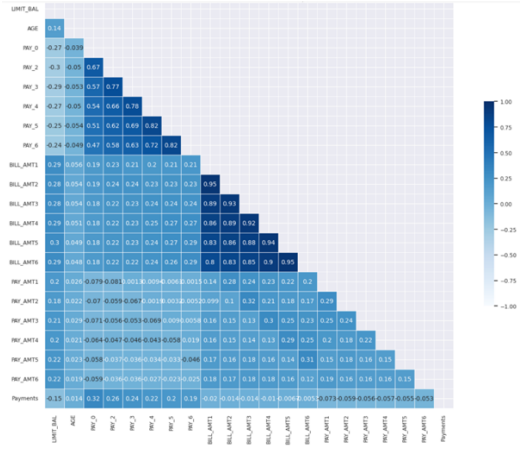

Figure 3. Correlation among features

E. Dimensionality Reduction and Feature Extraction:

Dimensionality reduction is a technique used to reduce the number of features or variables in a dataset while retaining the most relevant information. It serves multiple purposes, including improving the speed of data processing, enabling visualization in lower dimensions, and addressing the curse of dimensionality.

In the context of the Default Credit Card Clients dataset, dimensionality reduction techniques such as Linear Discriminant Analysis (LDA), Principal Component Analysis (PCA), and Kernel PCA can be applied.

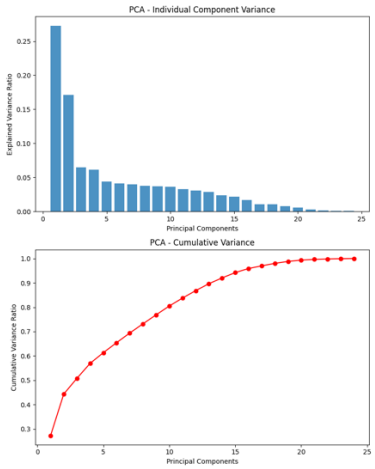

Figure 4. PCA Individual Component Variance & Cumulative Variance

Kernel PCA is a dimensionality reduction technique that extends the traditional PCA by incorporating a kernel function. It allows for the separation of features that are non-linearly separable in the original feature space. By mapping the data into a higher-dimensional feature space, Kernel PCA can capture complex relationships and reveal hidden patterns that may not be evident in the original data.

In the case of the "Default Credit Card Clients" dataset, Kernel PCA can be implemented using the scikit-learn library in Python. This implementation provides a flexible and efficient way to perform Kernel PCA and obtain the results in github.

IV. Machine Learning Classification Techniques

A. K-NearestNeighbor(K-NN)

K-Nearest Neighbors is one of the most straightforward classification algorithms to use and understand. KNN is recommended to be used on models where the class is defined by a limited number of attributes, so it seems to fit this model. The Euclidean distance function is commonly used to measure the distance between two points.

In the case of the Default Credit Card Clients dataset, KNN is suitable because it can handle a complex target function. The dataset contains multiple attributes that can influence the default payment status of credit card clients. By considering the attributes of neighbouring instances, KNN can make predictions based on the similarities between instances.

The choice of the number of neighbours (k) in the KNN algorithm is important. There is no general algorithm to determine the optimal value of k. However, a commonly used approach is to take the square root of the number of testing examples (n) as a starting point. In the case of the "Default Credit Card Clients" dataset, with 545 testing examples, a value of k=23 has been found to provide the highest accuracy. Choosing an odd number for k helps to avoid tie situations when determining the class.

KNN has several advantages that make it suitable for this project. It can handle complex target functions, which is important considering the multiple attributes in the dataset. Additionally, KNN utilizes the entire information of the dataset, and the training process does not require much time. However, it is essential to pre-process the data before applying KNN to remove any irrelevant information. One drawback of KNN is the computational cost, as it requires calculating the distance between each record in the dataset.

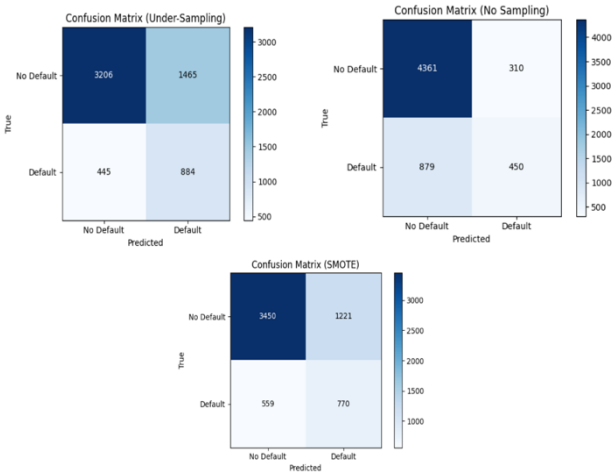

Figure 5. Confusion Matrices KNN Classifier

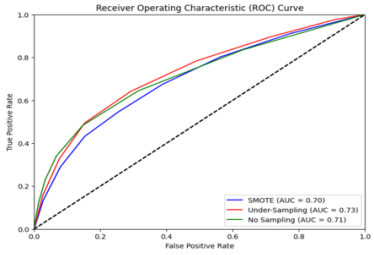

Figure 6. ROC Curve K-Nearest

TABLE 2 K-NEAREST RESULTS SUMMARY

Classification Report (SMOTE):				
Class	Precision	Recall	F1-Score	Support
0	0.86	0.74	0.79	4671
1	0.39	0.58	0.46	1329
Accuracy	0.7			6000
Macro Avg	0.62	0.66	0.63	6000
Weighted Avg	0.76	0.7	0.72	6000
Classification Report (Under-Sampling):				
Class	Precision	Recall	F1-Score	Support
0	0.88	0.69	0.77	4671
1	0.38	0.67	0.48	1329
Accuracy	0.68			6000
Macro Avg	0.63	0.68	0.63	6000
Weighted Avg	0.77	0.68	0.71	6000
Classification Report (No Sampling):				
Class	Precision	Recall	F1-Score	Support
0	0.83	0.93	0.88	4671
1	0.59	0.34	0.43	1329
Accuracy	0.8			6000
Macro Avg	0.71	0.64	0.66	6000
Weighted Avg	0.78	0.8	0.78	6000
GridSearch CV				
Best	{'n_neighbors': 7, 'weights': 'uniform'}			

Parameters:	
Best Score:	0.799866708

B. Random Forest:

Random Forest is a popular machine learning algorithm that belongs to the ensemble learning family. It is a combination of multiple decision trees, where each tree is trained on a random subset of the training data and features. The final prediction of the Random Forest is determined by aggregating the predictions of individual trees.

The purpose of using Random Forest for the given dataset, "default credit card clients," can be to predict whether a credit card user will default on their payment or not based on various features available in the dataset. Random Forest has several advantages that make it suitable for this task:

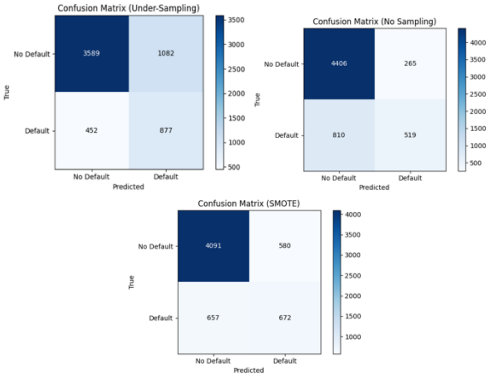

Figure 7. Random Forest

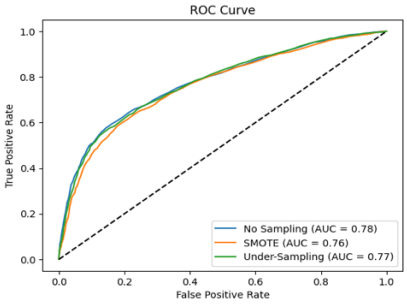

Figure 8. ROC Curve Random Forest

TABLE 3 Random Forest

Classification Report (SMOTE):				
	precision	recall	f1-score	support
0	0.86	0.88	0.87	4671
1	0.54	0.51	0.52	1329
Accuracy			0.79	6000
Macro avg	0.7	0.69	0.69	6000
weighted avg	0.79	0.79	0.79	6000
Classification Report (Under-Sampling):				
	precision	recall	f1-score	support
0	0.89	0.77	0.82	4671
1	0.45	0.66	0.53	1329
Accuracy			0.74	6000
Macro avg	0.67	0.71	0.68	6000
weighted avg	0.79	0.74	0.76	6000
Classification Report (No Sampling):				
	precision	recall	f1-score	support
0	0.84	0.94	0.89	4671
1	0.66	0.39	0.49	1329
Accuracy			0.82	6000
Macro avg	0.75	0.67	0.69	6000
weighted avg	0.8	0.82	0.8	6000

C. Linear Discriminant Analysis:

LDA stands for Linear Discriminant Analysis, which is a dimensionality reduction technique and a supervised learning algorithm used for classification. It is commonly used for solving classification problems where the goal is to predict the class or category of a given sample based on its features.

The dataset contains many features (e.g., demographic information, payment history, credit limit, etc.). Applying LDA can help reduce the dimensionality of the dataset by extracting a smaller set of features that are most relevant for predicting the default status of credit card clients.

LDA provides a way to visualize and understand the discriminative information present in the dataset. The transformed features obtained from LDA can be interpreted as linear combinations of the original features, allowing for a better understanding of the factors that contribute to the default status.

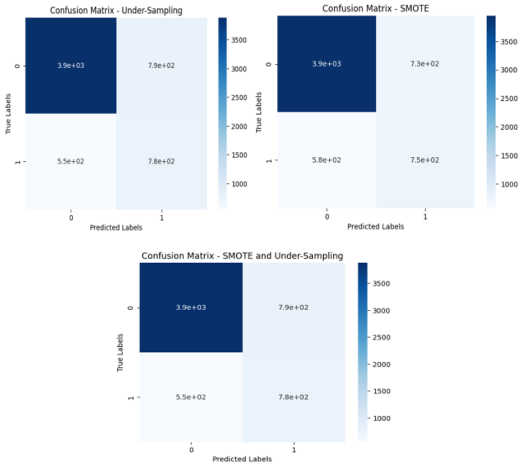

Figure 9. Linear Discriminant Analysis

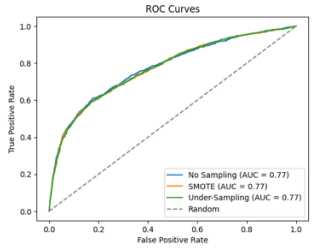

Figure 10. ROC Curve Linear Discriminant Analysis

TABLE 4 Linear Discriminant Analysis

Classification Report (SMOKE and Under-Sampling):				
Class	Precision	Recall	F1-Score	Support
0	0.88	0.83	0.85	4671
1	0.5	0.59	0.54	1329
Accuracy	0.78			6000
Macro Avg	0.69	0.71	0.7	6000
Weighted Avg	0.79	0.78	0.78	6000
Classification Report (SMOTE):				
Class	Precision	Recall	F1-Score	Support
0	0.87	0.84	0.86	4671
1	0.51	0.57	0.54	1329
Accuracy	0.78			6000
Macro Avg	0.69	0.71	0.7	6000

Weighted Avg	0.79	0.78	0.79	6000
Classification Report (Under-Sampling):				
Class	Precision	Recall	F1-Score	Support
0	0.88	0.83	0.85	4671
1	0.5	0.59	0.54	1329
Accuracy	0.78			6000
Macro Avg	0.69	0.71	0.7	6000
Weighted Avg	0.79	0.78	0.78	6000
GridSearch CV				
Best Parameters:	{'shrinkage': None, 'solver': 'lsqr'}			
Best Score:	0.809658905			

V. Application of techniques and Results

Based on the provided classification reports for KNN, LDA, and Random Forest models, we can analyze the performance of each model.

KNN:

- KNN was evaluated using three different sampling techniques: SMOTE, under-sampling, and no sampling.
- The performance metrics vary across the sampling techniques.
- The SMOTE sampling technique achieved an accuracy of 0.7, with a weighted average F1-score of 0.72.
- The under-sampling technique achieved an accuracy of 0.68, with a weighted average F1-score of 0.71.
- The no sampling technique achieved the highest accuracy of 0.8, with a weighted average F1-score of 0.78.
- The best parameters for KNN were found to be {'n_neighbors': 7, 'weights': 'uniform'}.

LDA:

- LDA was evaluated using three different sampling techniques: SMOTE, under-sampling, and no sampling.
- The performance metrics are consistent across the sampling techniques.
- All three techniques achieved an accuracy of 0.78, with a weighted average F1-score of 0.78.
- The best parameters for LDA were found to be {'shrinkage': None, 'solver': 'lsqr'}.

Random Forest:

- Random Forest was evaluated using three different sampling techniques: no sampling, SMOTE, and under-sampling.
- The performance metrics vary across the sampling techniques.
- The no sampling technique achieved an accuracy of 0.82, with a weighted average F1-score of 0.80.
- The SMOTE technique achieved an accuracy of 0.79, with a weighted average F1-score of 0.79.
- The under-sampling technique achieved an accuracy of 0.74, with a weighted average F1-score of 0.76.

Overall, the performance of the models varied depending on the sampling technique used. The no sampling technique generally achieved higher accuracy, while the SMOTE and under-sampling techniques had slightly lower accuracy but balanced performance across the classes. It is important to note that these results are specific to the given dataset, and the performance of the models may vary for different datasets or with different parameter settings.

VI. Conclusion

The Random Forest model demonstrated the highest accuracy, ranging from 74% to 82% depending on the sampling technique used. It exhibited relatively high precision and recall for predicting non-default cases but had some difficulty in accurately predicting defaults.

The LDA model achieved an accuracy of 78% across all sampling techniques and demonstrated balanced precision and recall for both default and non-default cases. It showed stable performance in predicting credit card client defaults.

The KNN model achieved moderate accuracy of 70%, and its performance varied depending on the sampling technique used. It performed better with the SMOTE sampling technique, but overall had relatively lower precision and recall for predicting defaults.

Further improvements can be made by exploring other algorithms, such as boosting models and ensembles, which could potentially enhance the predictive performance. Additionally, incorporating more economic indicators into the dataset may lead to a more generalized model that can adapt to the rapidly changing financial market.

Overall, while the models presented some limitations, the Random Forest model showed the highest performance, followed by the LDA model. These findings can provide valuable insights for financial institutions in assessing the creditworthiness of clients and making informed decisions.

VII. References

1. Gnanadesikan, R., Kettenring, J. R., & Dillon, W. R. (1988). Kernel principal component analysis: A new technique for linear and nonlinear feature extraction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 10(6), 873-886.
2. Hackeling, G. (2014). Mastering Machine Learning with scikit-learn. Packt Publishing Ltd.
3. Hill, N. J., & Lewicki, M. S. (2006). Statistics: Methods and Applications: A Comprehensive Reference for Science, Industry, and Data Mining. StatSoft.
4. N. V. Chawla, et al., "SMOTE: synthetic minority over-sampling technique", Journal of artificial intelligence research, 321-357, 2002. https://arxiv.org/abs/1106.1813.
5. M. Mohri, et al., "Foundations of machine learning", MIT press, 2018. https://cs.nyu.edu/~mohri/mlbook/.
6. A. Smola, S.V.N. Vishwanathan, "Introduction to Machine Learning", 2008. https://alex.smola.org/drafts/thebook.pdf
7. F. Pedregosa, et al., "Scikit-learn: Machine Learning in Python", Journal of Machine Learning Research, Volume 12, 2825-2830, 2011. https://scikit-learn.org/stable/.