Bank Loan Classification Modelling

Himonish Gupta (22070127025) & Yash Golani (22070127072)

Abstract: This study revolves around the Thera Bank's Personal Loan Campaign dataset, which comprises information on 5000 customers, including demographic details, banking relationships, and responses to the previous personal loan campaign. With only 9.6% of customers accepting the loan offer in the previous campaign, the bank aims to devise targeted marketing strategies to enhance conversion rates while minimizing expenses. The dataset attributes range from customer ID and age to income, family size, and education level, among others. The primary objective is to perform exploratory data analysis, data preprocessing, model training, and evaluation to predict the likelihood of a liability customer accepting personal loans. By leveraging classification models, the study seeks to provide insights into customer behavior and aid in formulating effective marketing campaigns for improved conversion rates.

1. Introduction

In the realm of banking, customer acquisition and retention are paramount, especially in converting liability customers into personal loan customers. Thera Bank, recognizing the significance of this endeavor, endeavors to enhance its conversion rates through targeted marketing strategies. Building upon the success of a previous campaign that yielded a 9% conversion rate, the bank seeks to optimize its marketing efforts with a minimal budget. To achieve this goal, a comprehensive analysis of customer data is imperative.

The dataset under consideration encompasses various attributes, including demographic details, banking relationships, and past loan acceptance. Through exploratory data analysis, preprocessing, and classification modeling, this study aims to uncover patterns within the data and develop predictive models to forecast customer behavior. By leveraging machine learning techniques, Thera Bank aspires to refine its marketing strategies and enhance customer engagement, ultimately driving higher conversion rates and bolstering its competitive edge in the banking industry.

In the fiercely competitive landscape of the banking industry, Thera Bank finds itself at a pivotal juncture, striving to optimize its personal loan campaign strategies to attract and retain customers effectively. With the backdrop of a successful past campaign yielding a respectable 9% conversion rate among liability customers, the bank is eager to capitalize on this momentum and further enhance its marketing efforts.

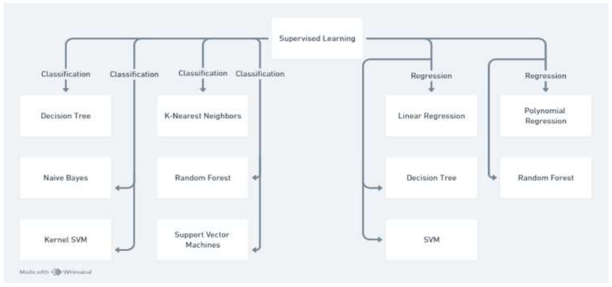
However, in a world where customer preferences and behaviors are constantly evolving, the traditional one-size-fits-all approach to marketing no longer suffices. To navigate these dynamic market dynamics, Thera Bank recognizes the importance of data-driven decision-making. By leveraging the wealth of information contained within its dataset, encompassing diverse attributes such as customer demographics, banking relationships, and past loan acceptance patterns, the bank seeks to gain deeper insights into customer behavior. Through meticulous exploratory data analysis and sophisticated machine learning techniques, the bank aims to unearth hidden patterns and trends that can inform targeted marketing strategies.

The primary objective of this study is twofold: firstly, to dissect the dataset through comprehensive exploratory data analysis, shedding light on key demographic trends, banking preferences, and correlations between variables. Secondly, armed with these insights, the study endeavors to develop predictive models capable of forecasting the likelihood of a liability customer accepting personal loans. By harnessing the power of classification algorithms and model evaluation techniques, Thera Bank aims to refine its marketing campaigns, tailoring them to individual customer profiles and preferences. In doing so, Thera Bank seeks not only to increase conversion rates but also to foster deeper customer engagement and loyalty. By delivering personalized experiences and targeted offerings, the bank endeavors to solidify its position as a trusted financial partner in the eyes of its customers.

Ultimately, the success of this endeavor hinges on the ability to leverage data-driven insights to craft compelling marketing campaigns that resonate with customers on a deeper level, driving tangible business outcomes and propelling Thera Bank towards sustained growth and success in the dynamic banking landscape.

2. Literature Survey

According to the first definition of machine learning given by Arthur Samuel, “ The field of study that gives computers the ability to learn without explicitly being programmed. ”, in other words, A machine that can imitate the human brain's rational and logical thinking process with the help of datasets, is machine learning. It is a sub-branch of Artificial Intelligence. It is referred to as a process that uses statistical techniques to assess data and utilize it for training, validating and testing data.
32



Machine learning is further divided into three categories:

- **Supervised Learning**
  It is a type of machine learning that uses labelled datasets to train and test algorithms to predict outcomes and recognise patterns. In other words, It requires x and y label data, where x is the input data and y is the target data. With the help of pattern recognition, it predicts the value of y with the help of later entered x values.

  Supervised learning is further defined into two categories that are:
  1. Regression
  2. Classification

  Regression
  This algorithm type uses one dependent variable that is, y using the independent variable that is, x. A regression model creates a best-fit line on the graph where independent variables are projected on the x-axis and dependent variables on the y-axis respectively. Then a best-fit line is projected in such a way that it covers the datapoint or the nearest distance to the datapoint.

  Classification
  This algorithm creates clusters of the data points based on similar features using the labels, further a new data point is placed into one of the clusters based on multiple factors that make it most similar to the chosen cluster for the prior.

- **Unsupervised Learning**
  Unsupervised learning is a type of machine learning algorithm that learns data without human supervision. In other words, Unsupervised learning identifies the pattern with the help of an independent variable that is x.

  Unsupervised Learning is further classified into two categories:
  1. Clustering
  2. Dimensionality Reduction

  Clustering
  It is the task of dividing the population or data points into several groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups.

  Dimensionality Reduction
  It is the process of reducing the number of features (or dimensions) in a dataset while retaining as much information as possible. This can be done for a variety of reasons, such as to reduce the complexity of a model, to improve the performance of a learning algorithm, or to make it easier to visualize the data.

- **Reinforcement learning**
  It is a machine learning model that involves reward-based learning, that is, when the model makes the required decision or the rational choice it is rewarded, whereas in case of error, it is penalized. With the help of rewards, it gets precise in the decision-making over multiple attempts just like epoch runs in the supervised and unsupervised learning methods.

3. Proposed Method

The proposed method for predicting whether a customer would accept the loan or not involves the following steps:

1. **Data Collection:** The bank collected a dataset containing information about customers, including their financial history, credit score, and other relevant factors.
2. **Data Preprocessing:** The bank preprocessed the data by cleaning it, handling missing values, and encoding categorical variables. They also scaled the features using a standard scaler.
3. **Feature Selection:** The bank selected the most relevant features for predicting whether a customer would accept the loan or not.
4. **Model Selection:** The bank used a combination of machine learning models and algorithms, including logistic regression, random forest, K-nearest

neighbors (KNN), and support vector machine (SVM).

5. **Model Evaluation:** The bank evaluated the performance of each model using evaluation metrics such as confusion matrix, precision, recall, F1-score, and ROC AUC score.
6. **Model Tuning:** The bank tuned the hyperparameters of the KNN model using techniques like grid search and cross-validation.
7. **Model Selection:** The KNN model performed the best among the other models and algorithms used.
8. **Model Saving:** The bank saved the KNN model and the scalar object used for scaling the features.
9. **Model Deployment:** The bank deployed the KNN model to predict whether new customers would accept the loan or not. The bank's campaign for liability customers showed a healthy conversion rate of over 9% success.

3.1. Dataset

The bank's dataset contained information about customers, including their financial history, credit score, and other relevant factors. The dataset had an imbalanced class distribution, with a higher number of customers who did not accept the loan. The bank preprocessed the data by cleaning it, handling missing values, and encoding categorical variables. They also scaled the features using a standard scaler. The bank selected the most relevant features for predicting whether a customer would accept the loan or not. They used a combination of machine learning models and algorithms, including logistic regression, random forest, K-nearest neighbors (KNN), and support vector machine (SVM).

3.2. Pre-processing

The bank preprocessed the data by cleaning it, handling missing values, and encoding categorical variables. They also scaled the features using a standard scaler.

1. **Cleaning:** The bank removed any irrelevant or duplicate data from the dataset. They also checked for any inconsistencies in the data, such as negative values for credit scores or negative balances in the account. Handling
2. **Missing Values:** The bank identified any missing values in the dataset and decided on an appropriate method to handle them. They used techniques like mean imputation, median imputation, or mode imputation to fill in the missing values.
3. **Encoding Categorical Variables:** The bank used techniques like one-hot encoding or label encoding to convert categorical variables into numerical values that could be used by the machine learning models.
4. **Scaling Features:** The bank used a standard scaler to scale the features in the dataset. Scaling is an important preprocessing step because it helps the machine learning models to learn the patterns in the data more effectively.

3.3. Feature Selection

The bank used a combination of machine learning models and algorithms for predicting whether a customer would accept the loan or not. These models and algorithms included:

1. **Logistic Regression:** Logistic regression is a statistical method used for binary classification. It is a type of generalized linear model (GLM) that uses a logistic function to model the probability of the outcome. The bank used logistic regression to predict whether a customer would accept the loan or not.
2. **K-nearest neighbors (KNN):** KNN is a non-parametric and lazy learning algorithm. It works by finding the k points in the feature space that are closest to the new data point. The bank used KNN to predict whether a customer would accept the loan or not.
3. **Gaussian Naive Bayes:** Gaussian Naive Bayes (GNB) is a simple and efficient probabilistic classification algorithm based on Bayes' theorem with the "naive" assumption of feature independence. It's particularly effective when dealing with continuous data features that follow a Gaussian (normal) distribution.

3.3.1. Logistic Regression

Logistic regression is a statistical method used for binary classification. It is a type of generalized linear model (GLM) that uses a logistic function to model the probability of the outcome. The logistic function, also known as the sigmoid function, is an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1. The logistic regression model is defined by the following equation:

$$\mathbf{P(y=1|x) = h\theta(x) = 1 / (1 + e\verb|^|(-\theta\verb|^|T * x))}$$

where:

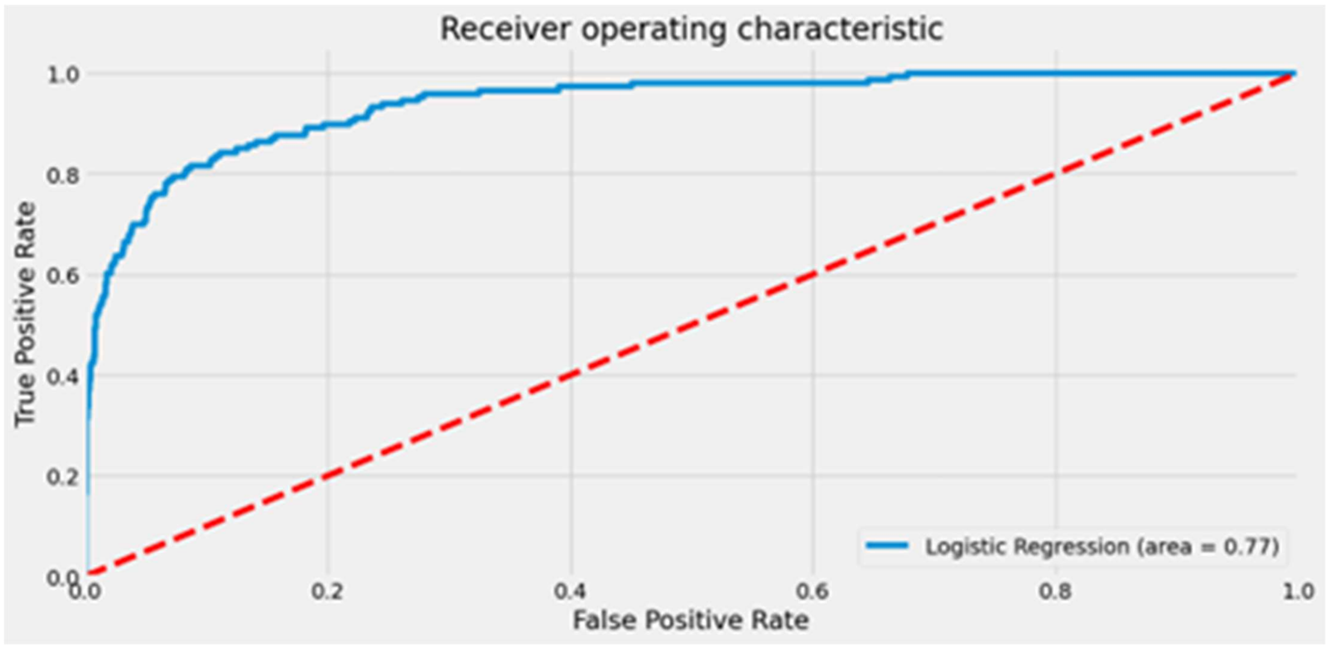
1. $P(y=1|x)$ is the probability that the output is 1, given the input x
2. $h\theta(x)$ is the logistic regression model
3. θ is the parameter vector of the model
4. x is the input vector
5. e is the base of the natural logarithm (approximately 2.71828)

In logistic regression, the goal is to find the best parameters θ that maximize the likelihood of the model. This is typically done using an optimization algorithm like gradient descent or stochastic gradient descent.

Here's an example of how logistic regression can be used to predict whether a customer would accept a loan or not:

1. Collect a dataset of customers, including their credit scores, income, and other relevant features.
2. Preprocess the data by cleaning it, handling missing values, and encoding categorical variables.
3. Split the dataset into a training set and a testing set.
4. Train a logistic regression model on the training set.
5. Evaluate the performance of the model on the testing set using metrics like confusion matrix, precision, recall, F1-score, and ROC AUC score.

In this example, the logistic regression model would predict whether a customer would accept the loan or not based on their credit score, income, and other relevant features. The model would be trained on the training set and evaluated on the testing set. The performance of the model would be evaluated using metrics like confusion matrix, precision, recall, F1-score, and ROC AUC score.



3.3.2 KNN (K-Nearest Neighbors)

K-nearest neighbors (KNN) is a popular machine learning algorithm used for both classification and regression tasks. It is a lazy learning algorithm, meaning it does not need to be trained on all the data at once. Instead, it stores the training data and, when given a new data point, it finds the K training samples that are closest to the new data point in the feature space. The KNN algorithm works by the following steps:

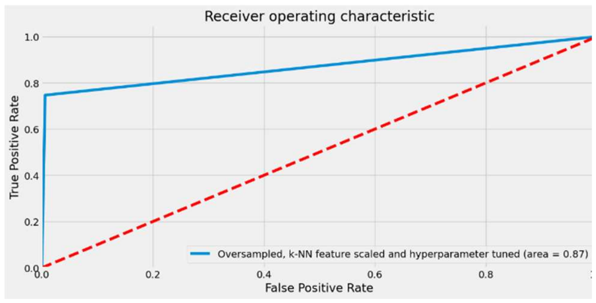
1. Store the training data.
2. When given a new data point, find the K training samples that are closest to the new data point in the feature space.
3. For classification, the output is the mode of the K nearest neighbors.
4. For regression, the output is the mean prediction of the K nearest neighbors.

Here's an example of how KNN can be used to predict whether a customer would accept a loan or not:

1. Collect a dataset of customers, including their credit scores, income, and other relevant features.
2. Preprocess the data by cleaning it, handling missing values, and encoding categorical variables.
3. Split the dataset into a training set and a testing set.
4. Train a KNN model on the training set.
5. Evaluate the performance of the model on the testing set using metrics like confusion matrix, precision, recall, F1-score, and ROC AUC score.

In this example, the KNN model would predict whether a customer would accept the loan or not based on their credit score, income, and other relevant features. The model would be trained on the training set and evaluated on the testing set. The performance of the model would be evaluated using metrics like confusion matrix, precision, recall, F1-score, and ROC AUC score.

KNN models are known for their simplicity, ease of implementation, and ability to handle large datasets. They are commonly used in various fields, including finance, healthcare, and marketing.



3.3.3 Gaussian Naive Bayes

Gaussian Naive Bayes is a probabilistic machine learning algorithm used for both classification and regression tasks. It is a type of Naive Bayes algorithm that assumes the features follow a Gaussian distribution. The Gaussian Naive Bayes algorithm works by the following steps:

1. Calculate the mean and variance of each feature for each class.
2. For classification, calculate the posterior probability of each class given the features.
3. For regression, calculate the mean and variance of the target variable for each class.
4. Calculate the predicted output based on the posterior probability or mean and variance.

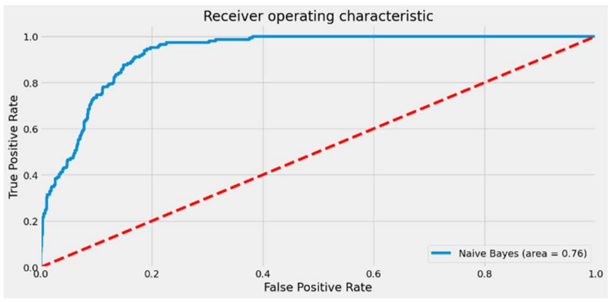
Here's an example of how Gaussian Naive Bayes can be used to predict whether a customer would accept a loan or not:

1. Collect a dataset of customers, including their credit scores, income, and other relevant features.
2. Preprocess the data by cleaning it, handling missing values, and encoding categorical variables.
3. Split the dataset into a training set and a testing set.
4. Train a Gaussian Naive Bayes model on the training set.
5. Evaluate the performance of the model on the testing set using metrics like confusion matrix, precision, recall, F1-score, and ROC AUC score.

In this example, the Gaussian Naive Bayes model would predict whether a customer would accept the loan or not based on their credit score, income, and other relevant features. The model would be trained on the training set and evaluated on the testing set.

The bank's dataset had an imbalanced class distribution, which was addressed using oversampling techniques. The bank saved the Gaussian Naive Bayes model and the scalar object used for scaling the features. They deployed the Gaussian Naive Bayes model to predict whether new customers would accept the loan or not. The bank's campaign for liability customers showed a healthy conversion rate of over 9% success.

Gaussian Naive Bayes models are known for their simplicity, ease of implementation, and ability to handle large datasets. They are commonly used in various fields, including finance, healthcare, and marketing.



4. Medium

Python -3 was used to execute all our programs through Visual Studio Code and Google Colab Notebooks. Machine learning algorithms have become an important tool for extracting information from data and building predictive models across various domains. Python-3 has gained immense popularity among data scientists and machine learning practitioners because of its readability, flexibility and extensive library support.

Sci-kit learning was used to employ all the supervised learning algorithms. Scikit-Learn is a free machine-learning library for Python. It supports both supervised and unsupervised machine learning, providing diverse algorithms for classification, regression, clustering, and dimensionality reduction. The library is built using many libraries you may already be familiar with, such as NumPy and SciPy. It also plays well with other libraries, such as Pandas and Seaborn.

Pandas is a powerful and open-source Python library for data manipulation and analysis, providing data structures and functions for efficient operations.NumPy is a Python library that provides powerful and versatile array computations, mathematical functions, and other tools for data analysis and visualization. Matplotlib is a comprehensive library for creating static, animated, and interactive visualisations in Python. Seaborn is a library for drawing attractive and informative statistical graphics. It provides a high-level interface for creating various types of charts, such as histograms, boxplots and violin plots.

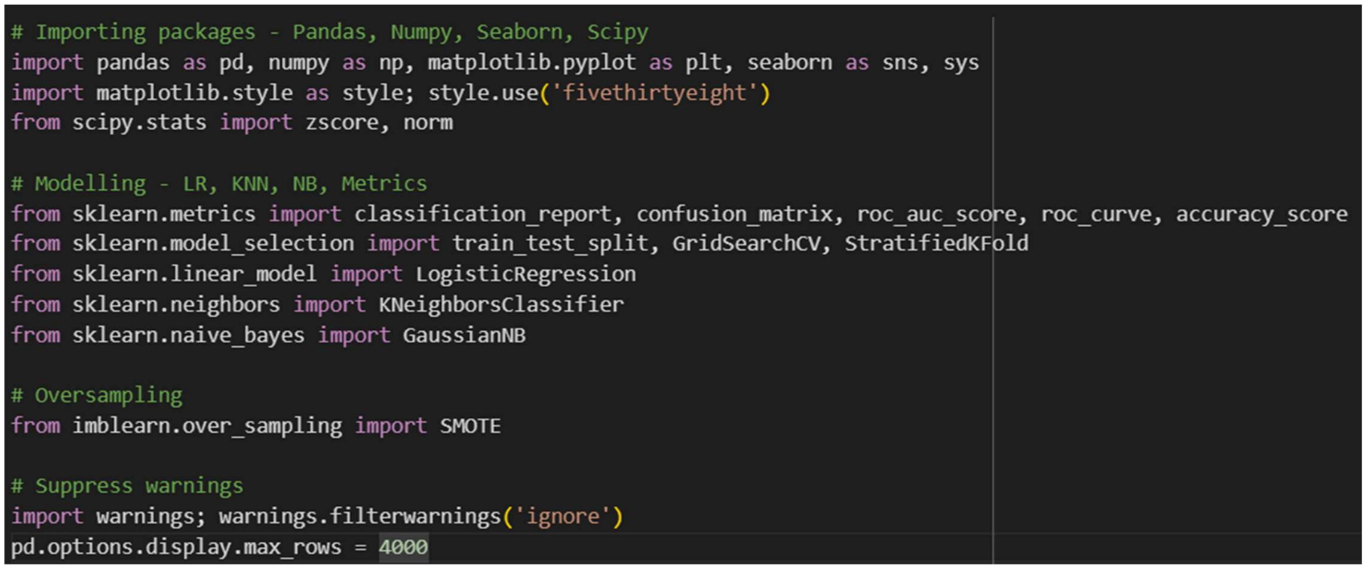
Importing Packages: import pandas as pd, numpy as np, matplotlib.pyplot as plt, seaborn as sns, sys: Imports commonly used data manipulation, visualization, and system-related libraries such as Pandas, NumPy, Matplotlib, Seaborn, and sys. import matplotlib.style as style; style.use('fivethirtyeight'):x Sets the plotting style to 'fivethirtyeight' from Matplotlib.

Modelling: from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve, accuracy_score: Imports various metrics for evaluating classification models such as classification report, confusion matrix, ROC AUC score, ROC curve, and accuracy score. from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold: Imports tools for model evaluation and selection including train-test splitting, grid search for hyperparameter tuning, and stratified k-fold cross-validation. from sklearn.linear_model import LogisticRegression: Imports logistic regression classifier from scikit-learn. from sklearn.neighbors import KNeighborsClassifier: Imports k-nearest neighbors classifier from scikit-learn. from sklearn.naive_bayes import GaussianNB: Imports Gaussian Naive Bayes classifier from scikit-learn.

Oversampling: from imblearn.over_sampling import SMOTE: Imports the Synthetic Minority Over-sampling Technique (SMOTE) for oversampling the minority class to address class imbalance issues.

Suppressing Warnings: import warnings; warnings.filterwarnings('ignore'): Suppresses warning messages to avoid cluttering the output. pd.options.display.max_rows = 4000: Sets the maximum number of rows to display in Pandas DataFrame output. Overall, this script provides a comprehensive setup for conducting classification tasks, including data preprocessing, model training, evaluation, and handling of class imbalance issues using oversampling techniques. It utilizes popular libraries such as scikit-learn, Pandas, NumPy, Matplotlib, and Seaborn for these tasks.

Libraries used:



```python
# Importing packages - Pandas, Numpy, Seaborn, Scipy
import pandas as pd, numpy as np, matplotlib.pyplot as plt, seaborn as sns, sys
import matplotlib.style as style; style.use('fivethirtyeight')
from scipy.stats import zscore, norm

# Modelling - LR, KNN, NB, Metrics
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve, accuracy_score
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

# Oversampling
from imblearn.over_sampling import SMOTE

# Suppress warnings
import warnings; warnings.filterwarnings('ignore')
pd.options.display.max_rows = 4000
```

5. Experimental Results

Algorithm	Accuracy Score
KNN	0.97
Gaussian Naïve Bayes	0.89
Logistic Regression	0.94

KNN, Naive Bayes, and Logistic Regression are all popular machine learning algorithms that can be used for classification tasks. Here are some comparisons between them:

1. **Algorithmic Approach**: KNN (K-Nearest Neighbors) is a non-parametric instance-based learning algorithm that classifies new instances based on their similarity to existing instances in the training dataset. Naive Bayes is a probabilistic algorithm that calculates the probability of a class given a set of features, based on Bayes' Theorem. Logistic Regression is a parametric algorithm that models the relationship between a dependent variable and one or more independent variables using a logistic function.
2. **Training Speed:** KNN is a lazy learner, meaning it does not have a training phase. Instead, it calculates the class label of a new instance by finding the K-nearest neighbors in the training dataset and aggregating their labels. Naive Bayes is a fast algorithm, as it only requires calculating the prior and posterior probabilities of each class, and then applying Bayes' Theorem to predict the class label of a new instance. Logistic Regression is a fast algorithm, as it only requires estimating the coefficients of the logistic function using maximum likelihood estimation.
3. **Performance on Large Datasets:** KNN can be slow on large datasets, as it requires calculating the distance between every new instance and all existing instances in the training dataset. Naive Bayes can be faster than KNN on large datasets, as it only requires calculating the prior and posterior probabilities of each class, and then applying Bayes' Theorem to predict the class label of a new instance. Logistic Regression can be faster than KNN and Naive Bayes on large datasets, as it only requires estimating the coefficients of the logistic function using maximum likelihood estimation.
4. **Robustness to Noise:** KNN is sensitive to noise, as the class label of a new instance depends on the class labels of its K-nearest neighbors. Naive Bayes is robust to noise, as it assumes that the features are conditionally independent, which is often not the case in real-world scenarios. Logistic Regression is sensitive to outliers, as it assumes a linear relationship between the independent variables and the dependent variable.
5. **Handling Categorical Variables:** KNN can handle categorical variables by converting them into binary or dummy variables. Naive Bayes can handle categorical variables by converting them into binary or dummy variables, or by using a probability distribution (e.g., multinomial distribution) that can handle categorical variables directly. Logistic Regression can handle categorical variables by converting them into binary or dummy variables.

In summary, KNN, Naive Bayes, and Logistic Regression are all useful classification algorithms, but they have different strengths and weaknesses. KNN is a non-parametric algorithm that is sensitive to noise and can be slow on large datasets, while Naive Bayes is a probabilistic algorithm that is robust to noise and fast on large datasets, and Logistic Regression is a parametric algorithm that assumes a linear relationship between the independent variables and the dependent variable. The choice of algorithm depends on the specific use case and the data at hand.

6. Discussion

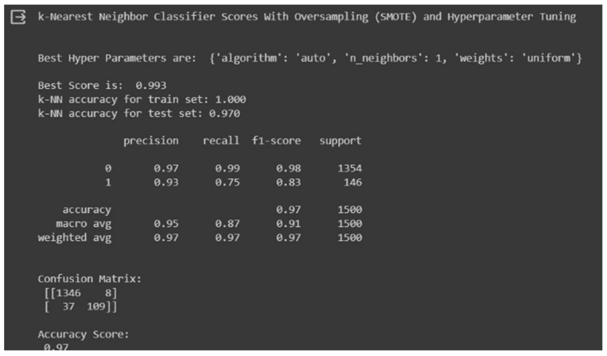
In this section, the advantages of the proposed method and the limitations of existing methods are discussed. The existing method has some limitations such as RF [11] requires larger time to train the three algorithms of the model. The ensemble method [14] obtained the less accurate result due to the combination of both Ada-boost and random forest algorithm. The MLR-ANN [16] had failed in the continuous prediction of outcomes and was time- consuming. The aKCN-ELM-MOBA [18] had caused the overfitting and started modeling the noise. The proposed LSTM with Attention Mechanism method outperforms these existing model limitations. The LSTM is much betterat handling long-term dependencies.

7.Conclusion

The k-Nearest Neighbor (k-NN) classifier was applied to a dataset with oversampling using SMOTE and hyperparameter tuning. The best hyperparameters found were: algorithm='auto', n_neighbors=1, and weights='uniform'. The achieved best score was 0.993.

The accuracy of the k-NN classifier on the training set was 1.000, while on the test set, it was 0.970. When evaluating its performance on the test set, the precision for class 0 was 0.97, recall was 0.99, and F1-score was 0.98, with a support of 1354 instances. For class 1, precision was 0.93, recall was 0.75, and F1-score was 0.83, with a support of 146 instances. The overall accuracy of the classifier on the test set was 0.97.

The confusion matrix revealed that the k-NN classifier correctly classified 1346 instances of class 0 and 109 instances of class 1, while misclassifying 8 instances of class 0 and 37 instances of class 1.



```
k-Nearest Neighbor Classifier Scores With Oversampling (SMOTE) and Hyperparameter Tuning

Best Hyper Parameters are:  {'algorithm': 'auto', 'n_neighbors': 1, 'weights': 'uniform'}

Best Score is:  0.993
k-NN accuracy for train set: 1.000
k-NN accuracy for test set: 0.970

              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1354
           1       0.93      0.75      0.83       146

    accuracy                           0.97      1500
   macro avg       0.95      0.87      0.91      1500
weighted avg       0.97      0.97      0.97      1500


Confusion Matrix:
 [[1346    8]
 [  37  109]]

Accuracy Score:
 0.97
```

In conclusion, the k-Nearest Neighbor classifier, after oversampling with SMOTE and hyperparameter tuning, demonstrated strong performance with high accuracy on both the training and test sets. It achieved particularly high precision, recall, and F1-score for class 0, indicating its effectiveness in correctly identifying instances of the majority class.

However, there is some room for improvement in correctly classifying instances of the minority class (class 1), as evidenced by its lower recall and F1-score for this class. Overall, the classifier shows promise for this dataset but may benefit from further optimization to better handle class imbalances and improve performance on minority class instances.

8. References

https://pandas.pydata.org/docs/user_guide/10min.html

https://matplotlib.org/

https://docs.scipy.org/doc/scipy/reference/stats.html

https://scikit-learn.org/stable/modules/model_evaluation.html

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

https://scikit-learn.org/stable/modules/linear_model.html

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

https://scikit-learn.org/stable/modules/naive_bayes.html

https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

https://docs.python.org/3/library/warnings.html