EE 559 Mathematical Pattern Recognition Final Project

Credit Card

Shivansh Amattya, amattya@usc.edu

Akash Capirala, capirala@usc.edu

May 5, 2023

1. Abstract

Credit cards have become an essential element of daily life due to their convenience and plethora of benefits. Having a safe and convenient option to pay for offline and online activities reduces the need to carry huge quantities of cash or find ATMs. We have concentrated on identifying clients of what sex and with what educational background are paying their bills on time or may default. In this paper, we have implemented seven different models, namely Trivial, Nearest Means, Support vector Machine, Perceptron, Naive Bayes Classifier, Logistic Regression, K nearest neighbors, Random Forest, and Artificial Neural Networks, to be able to find the best model that is correctly able to predict based on inputs. We found that the best-performing model based on accuracy and F1 score is the Random Forest model with a max depth parameter of 5. We obtain testing accuracy and F1 score of 81.96% and 80.25% respectively.

2. Introduction

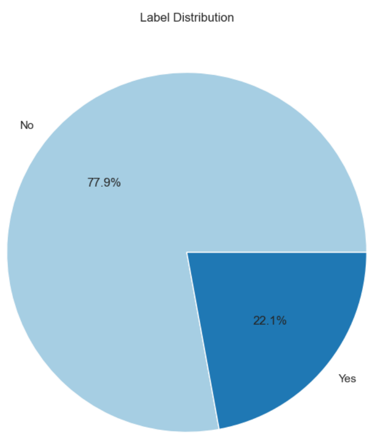
2.1. Problem Statement and Goals

The credit card dataset contains payment information for 30,000 Taiwanese clients for six months, from April 2005 to September 2005. The dataset contains information about credit card users, such as their credit limit, gender, education, marital status, age, repayment status, bill statements, and total amount paid. The repayment status indicates whether a user made timely or late payments during the specified month. The amount paid on the bill statements is the amount paid by the user for the corresponding month. The goal is to predict whether the customer defaults on payment or not. This is a binary classifier task.

3. Approach and Implementation

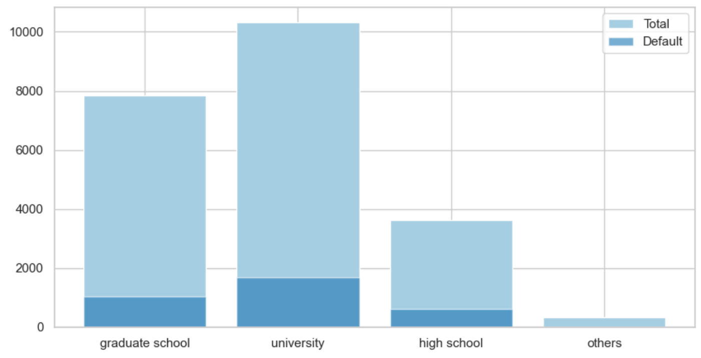
3.1. Dataset Usage

The training dataset has 27,000 data points. The testing dataset has 3000 data points. The datasets have no null values, and the figure below depicts the class distribution of the training dataset. The model is hyperparameter tuned using k-fold validation with the number of folds being 5. We have also used train_test_split for splitting the data into training and validation sets. The validation is done using the training set to test the accuracy of the model on the testing set, which remains unseen during the training process



3.2. Preprocessing

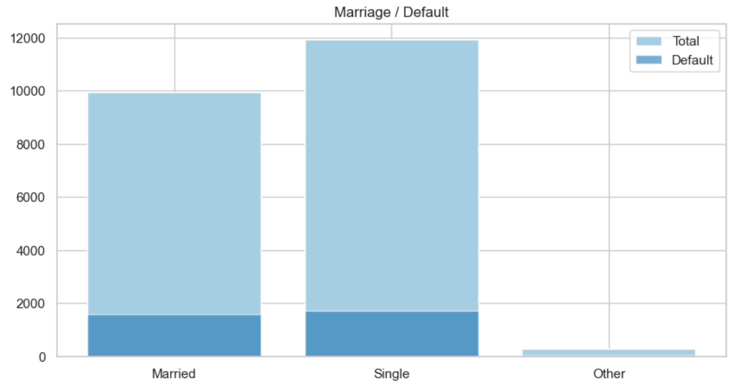
Data preparation is cleaning, transforming, and organizing raw data in preparation for analysis or modeling, making it a crucial part of any machine learning task. We clean the data by reassigning feature labels that are unidentified to known labels. (eg. We reassign the numbers 0, 5, and 6 in EDUCATION to class 4).
Another step in our data-cleaning process is removing outliers using a Quantile-based outlier removal method. We eliminate any data points that are not between the 1st and 99th percentiles. After this, we scale the data using StandardScaler, a part of the scikit_learn preprocessing module, so that no single feature dominates the analysis due to variances in scale or unit. Another dimension we have considered is upsampling the data to compensate for the class imbalance. This is being done using SMOTE which is a part of the imblearn module.



- Percentage of graduate school is 13.15%

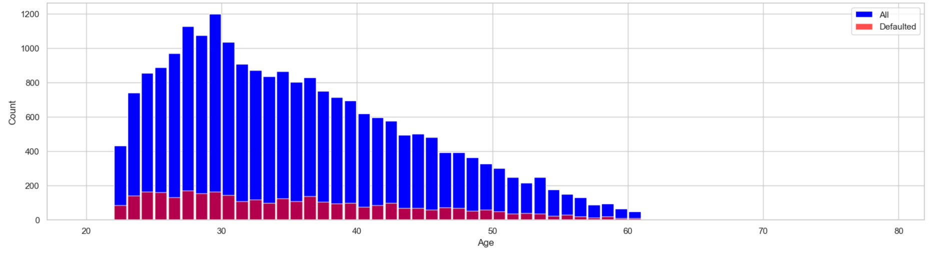
- Percentage of university is 16.20%
- Percentage of high school is 17.25%
- Percentage of others is 1.73%

As we are able to see from the above bar plot,
The highest default % are in high school



- Percentage of Married is 15.99%
- Percentage of Single is 14.29%
- Percentage of Other is 15.57%

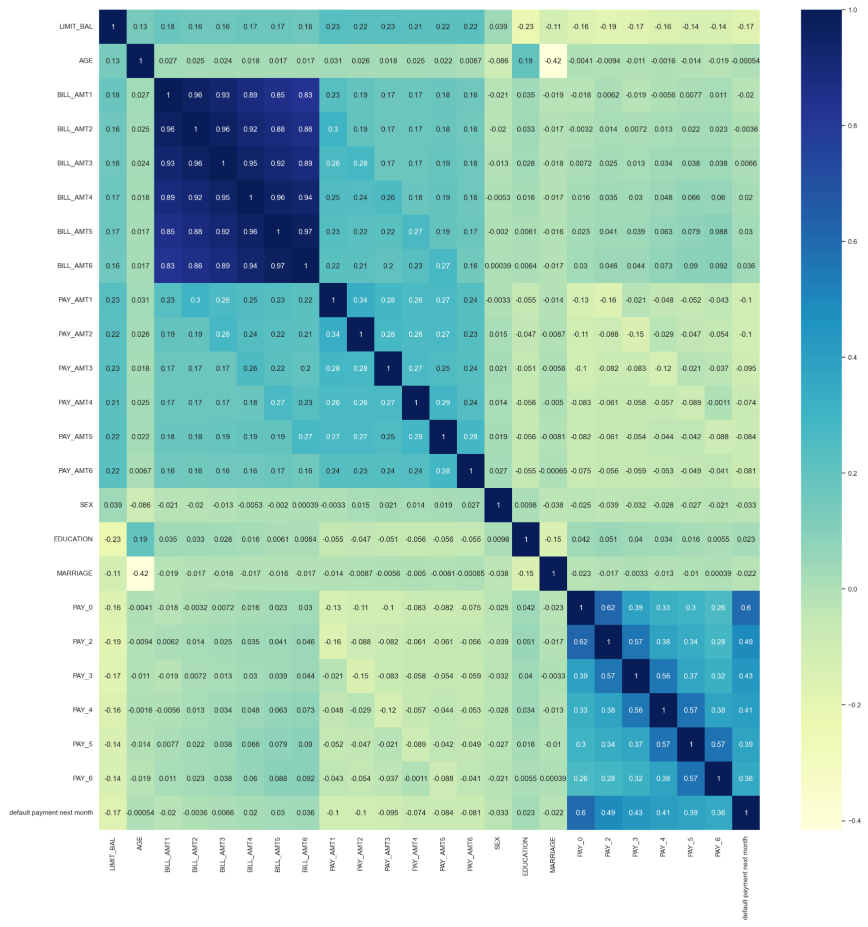
The highest default % are Married



In this we have the bar plot indicating the percentage of defaulter across all ages

3.4. Feature dimensionality adjustment

We employed LDA for feature dimensionality reduction, giving us a linear combination of features that maximizes class separation while maintaining class-specific information. We chose LDA over PCA as LDA maximizes class separation, which is not the case in PCA. The projection maximizes the between-class scatter matrix to within-class scatter matrix ratio. The eigenvectors and eigenvalues of the extended eigenvalue problem created by the scatter matrices are calculated to achieve this projection. The eigenvectors corresponding to the largest eigenvalues represent the optimal linear discriminant directions. The feature space is reduced in dimensionality to a smaller D=1 feature space. Removing the unnecessary feature from our dataset may lower the likelihood of overfitting and the computation speed/requirement. To better understand the relationship between each feature, we generated a heat map using Pearson's correlation coefficient that tells how each feature is related.



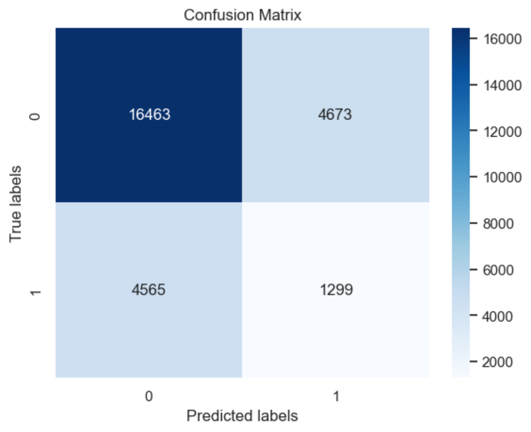
3.5. Training, classification and/or regression, and model selection

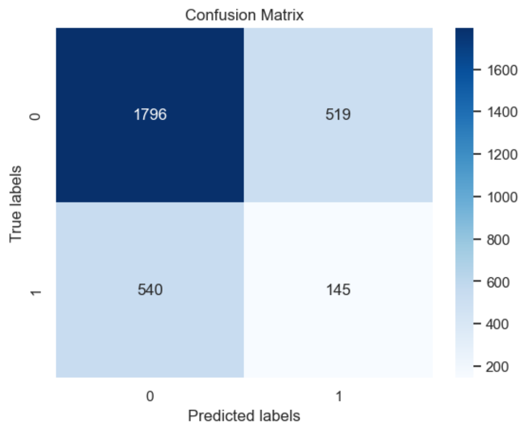
Trivial system

A trivial system is a straightforward classification system that gives class labels to data points based on the probability distribution of the class labels in the training dataset. In a two-class binary classification problem, the basic system randomly assigns class labels 0 and 1 to new data points based on the proportion of training data points belonging to each class. Ni denotes the number of training data points with class i, and the total number of training data points is denoted by N.

A system that generates random class assignments (0, 1) with probabilities N0/N and N1/N, respectively.

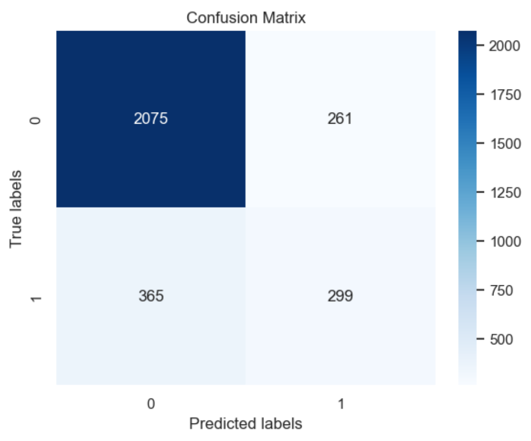
The confusion matrix for training and testing is as follows:

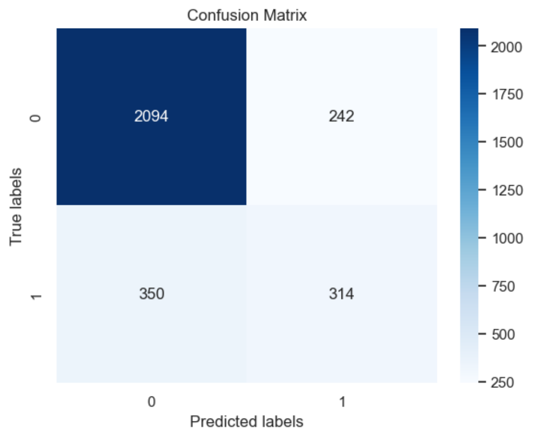




The Baseline Model - Nearest Means Classifier:
We utilized the Nearest means classifier as the baseline model. It assigns the class name with the closest mean to the data point. The mean for each target class is calculated throughout training. Among all calculated distances, the shortest distance is chosen and assigned to the corresponding class. We perform k-fold cross-validation (k = 5) to obtain the cross-validation score and, in the end, train the model on the whole dataset. We can observe from the results that the baseline works better on the imbalanced dataset and performing LDA.

Testing Confusion matrix for the imbalanced dataset and the balanced dataset is as follows:

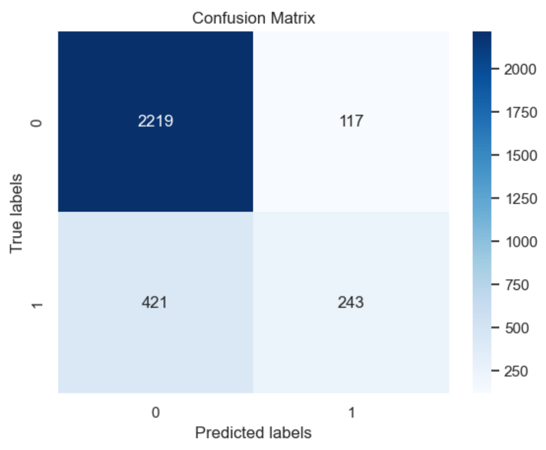


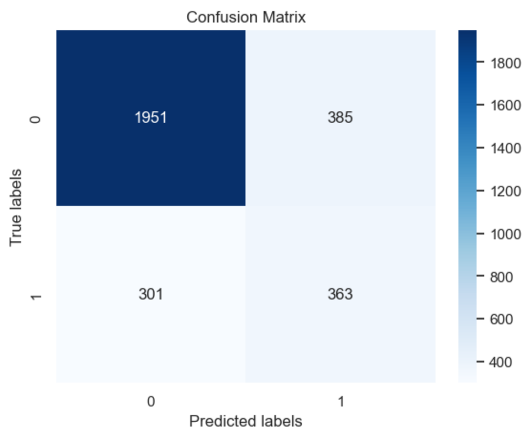


K-nearest-neighbors (KNN):

KNN uses a distance measure to identify the K nearest neighbors in the training data. The new data point's class designation is selected by majority voting among its K nearest neighbors. We perform hyperparameter tuning using k - fold cross validation using the gridsearchCV module. The optimal parameters obtained are the number of nearest neighbors being 49 and the weights being uniform i.e., all the nearest data points are given equal priority when classifying the new data point. We perform k-fold cross-validation (k = 5) on the best model to obtain the cross-validation score and train the model on the whole dataset. We can observe that we get the best results on the imbalance dataset and performing LDA.

Testing Confusion matrix for the imbalanced dataset and the balanced dataset is as follows:





The perceptron algorithm:

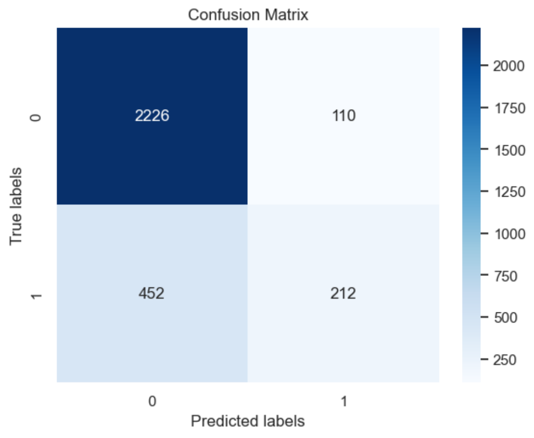
It corresponds to a two-class model in which the input vector x is transformed using a fixed nonlinear transformation to yield a feature vector (x), which is then used to build a generalized linear model of the form y(x) = f wT(x), where the nonlinear activation function f() is given by a step function of form f(a) = +1, a 0 1, a 0. The vector (x) typically contains a bias component 0(x)=1. The algorithm used to find the perceptron's parameters w is most easily motivated by error function minimization. All patterns must meet wT(xn)tn > 0.

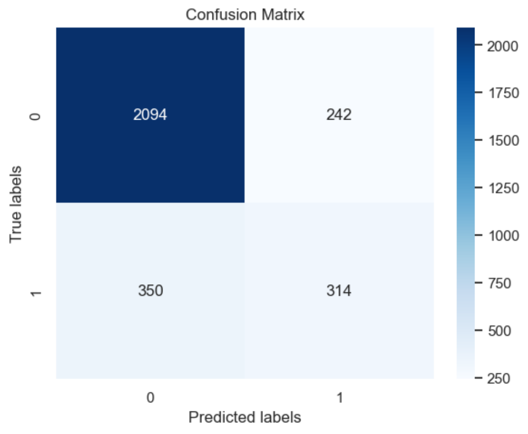
The perceptron criteria correlate zero error with each correctly classified pattern, whereas it attempts to minimize the number wT(xn)tn for any misclassified pattern xn. The perceptron criterion is therefore given by

EP(w) = − n∈M wTφntn. We perform hyperparameter tuning using k - fold cross validation using the gridsearchCV module. The optimal parameters obtained are the learning rate of 0.0001, max iteration being

100, and using the L1 penalty. We perform k-fold cross-validation (k = 5) on the best model to obtain the cross-validation score and train the model on the whole dataset. We can observe that we get the best results on the imbalanced dataset and performing LDA.

Testing Confusion matrix for the imbalanced dataset and the balanced dataset is as follows:

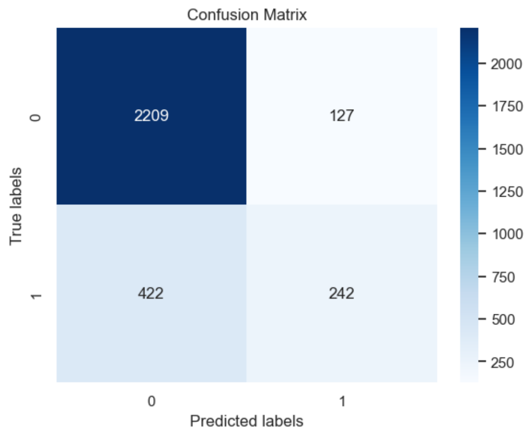


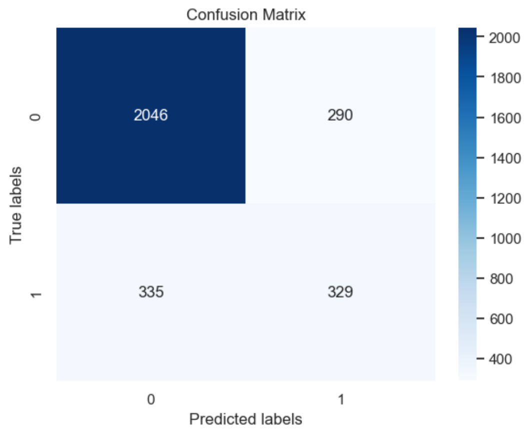


Logistic Regression:

It is a type of regression analysis where the target is categorical and has only two possible outcomes, often represented as 0 and 1. Logistic regression aims to estimate the probability that an instance belongs to a particular class based on its features. The logistic regression model uses the logistic function (also known as the sigmoid function) $\sigma(a) = 1 / 1 + \exp(-a)$, Here $\sigma(\cdot)$ is the logistic sigmoid function, to map the linear combination of the input features to a probability value between 0 and 1. Our model's best hyperparameters are C = 0.01 using the L2 penalty on the preprocessed data. We obtain the cross-validation accuracy (k = 5) using 5-fold cross-validation and then train the model on the whole dataset. We can observe from the results that the model works better on the imbalanced dataset and performing LDA.

Testing Confusion matrix for the imbalanced dataset and the balanced dataset is as follows:

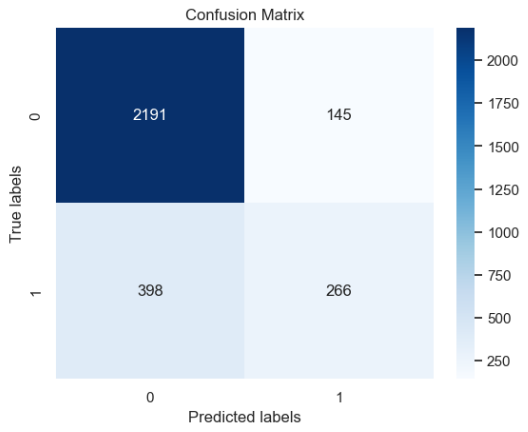


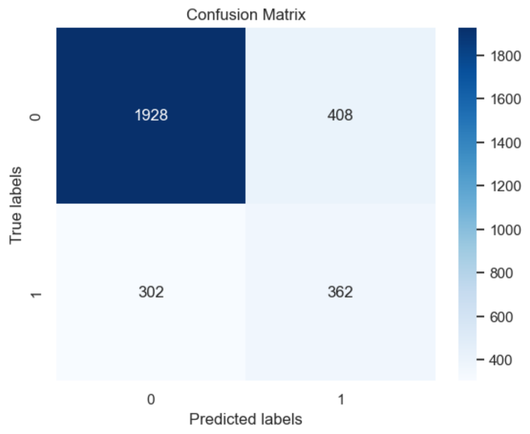


ANN/Feed-forward Network Functions:
An artificial neural network is an attempt to imitate the network of neurons that comprise the human brain for the computer to learn and make judgments in a human-like manner. The number of input nodes is the number of hidden nodes The loss function used is binary cross entropy. It computes the difference between the expected and actual class probabilities of a binary classification model. The optimizer we use is Adam Optimizer. Adam combines the benefits of two optimization approaches, Adagrad and RMSprop. We used two activation functions s ReLu and Sigmoid. ReLu is an activation function with the formula f(x) = max(0, x).ReLU aids in the resolution of the vanishing gradient problem. For positive inputs, ReLU gives a consistent positive gradient, allowing

for smoother gradient flow during training. The Sigmoid activation function:
Any real-valued number can be mapped to a value between 0 and 1 using the S-shaped curve. The following is the definition of the sigmoid function:1 / (1 + exp(-x)) f(x) x to a number between 0 and 1. We perform hyperparameter tuning using k - fold cross validation using the gridsearchCV module. The optimal parameters obtained are a learning rate of 0.01 and a weight decay of 0.01. We perform k-fold cross-validation on the best model to obtain the cross-validation score and train the model on the whole dataset. We can observe that we get the best results on the imbalanced dataset and performing LDA.

Testing Confusion matrix for the imbalanced dataset and the balanced dataset is as follows:



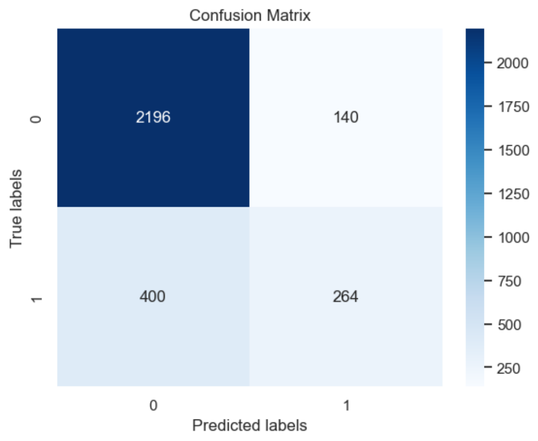


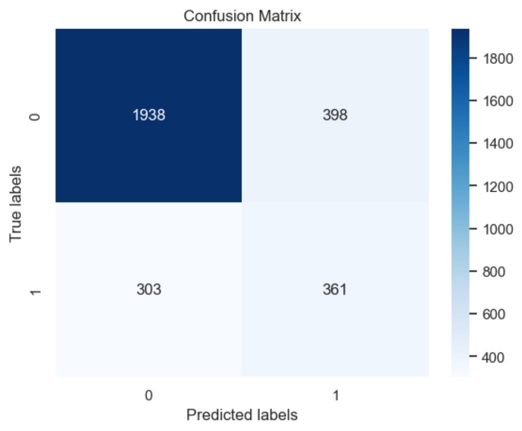
Random Forest:

Random uses decision trees for training on the data. Each tree is built with a random subset of features and a random subset of training data. The trees create predictions collectively by averaging or voting and assessing suitable criteria like accuracy, precision, recall, or F1 score.

We can eliminate overfitting in Random forests as we average the predictions of numerous trees. Random Forest measures feature importance and notes which features are more influential in the prediction process than individual decision trees. We perform hyperparameter tuning using k - fold cross validation using the gridsearchCV module. The optimal parameters obtained are a max depth of 5. We perform k-fold cross-validation on the best model to obtain the cross-validation score and train the model on the whole dataset. We can observe that we get the best results on the imbalanced dataset and performing LDA.

Testing Confusion matrix for the imbalanced dataset and the balanced dataset is as follows:

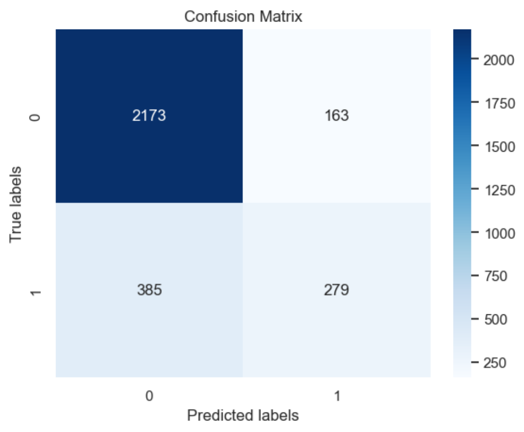


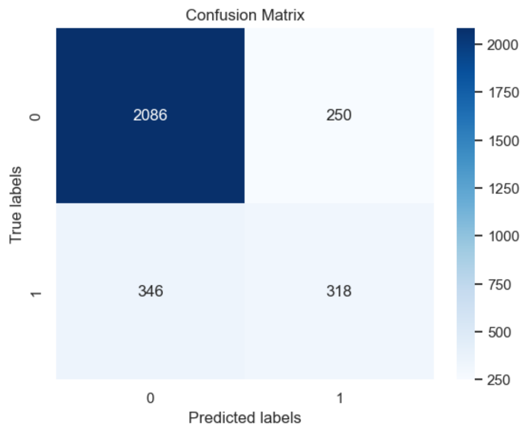


Naive Bayes:

It is widely used in classification tasks. By integrating the prior probability of the class with the conditional probabilities of the features given that class, Naive Bayes calculates the chance of a data point belonging to a certain class. Despite its simplicity and "naive" assumption, Naive Bayes is frequently effective and computationally efficient. We perform hyperparameter tuning using k - fold cross validation using the gridsearchCV module. The optimal parameter obtained is the var_smoothing value of 1e-30 with Gaussian Naive Bayes. We perform k-fold cross-validation on the best model to obtain the cross-validation score and train the model on the whole dataset. We can observe that we get the best results on the imbalanced dataset and performing LDA.

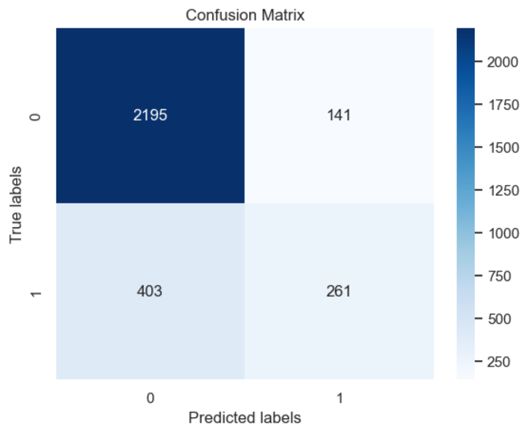
Testing Confusion matrix for the imbalanced dataset and the balanced dataset is as follows:

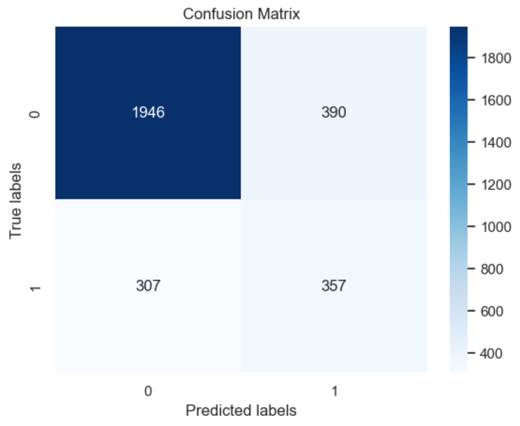




Support Vector Machine (SVM):
SVM aims to find the best hyperplane that separates the two classes in the input feature space. The hyperplane is determined by support vectors, a subset of training instances closest to the decision boundary. The optimal parameters we obtain by performing k-fold cross-validation are C = 100, gamma = 10, and the kernel is an Rbf kernel. We obtain the cross-validation accuracy using k = 5 k-fold cross-validation and then train the model on the whole dataset. We can observe from the results that the model works better on the imbalanced dataset and performing LDA.

Testing Confusion matrix for the imbalanced dataset and the balanced dataset is as follows:





4. Results and Analysis: Comparison and Interpretation

Model/Accuracy	Training Accuracy	Training F1 score	Testing Accuracy	Testing F1 score
Trival Model	65.78	65.89	64.70	64.50

Balanced Dataset

Model/Accuracy	Baseline - Nearest Means	SVM	Logistic Regression	K - Nearest Neighbours	Random Forest	Artificial Neural Network	Naive Bayes	Perceptron
Training Accuracy	87.41	89.45	88.59	89.51	89.49	89.42	88.04	87.58
Training F1 score	87.40	89.44	88.59	89.499	89.47	89.86	88.03	87.56
Validation Accuracy	88.21	89.44	80.60	89.40	89.45	89.42	88.03	84.80
Validation F1 Score	88.19	89.45	80.60	89.41	89.46	89.75	88.03	85.24
Testing Accuracy	80.26	76.73	79.163	77.13	76.63	76.36	80.13	80.26
Testing F1 score	79.61	77.23	78.80	77.67	77.17	50.52	79.55	79.61

Imbalanced Dataset

Model/Accuracy	Baseline - Nearest Means	SVM	Logistic Regression	K - Nearest Neighbors	Random Forest	Artificial Neural Network	Naive Bayes	Perceptron
Training Accuracy	88.51	90.06	89.97	90.04	90.09	89.91	89.79	89.50
Training F1 score	88.86	89.72	89.43	89.37	89.73	64.62	89.68	88.52
Validation Accuracy	88.23	90.03	89.98	89.85	90.00	.8973	89.79	89.43
Validation F1 Score	88.22	90.37	90.52	90.45	90.39	6241	89.90	89.86
Testing Accuracy	80.56	81.86	81.69	78.2	81.96	81.86	81.73	81.26
Testing F1 score	79.82	80.11	79.63	79.95	80.25	49.81	80.31	78.65

We tested all classifiers with and without upsampling the dataset model, including baseline model. We can see from the above that using the imbalanced dataset works best and provides high testing accuracy for all classifiers.

As a part of feature reduction, we utilized linear discriminant analysis and reduced the feature space to 1. Therefore, the DOF for all models is one. The total number of constraints for all models is 27000, as the final model is trained on the whole dataset. We got the best testing accuracy for Random forest, which is 81.96%, because the learning method combines multiple decision trees to make predictions, capturing various aspects of the data, and taking the mean of the forecasts of multiple trees, it models complex patterns and achieves high accuracy. ANNs also produce comparable testing accuracy, which is comparable to Random Forest. This is because Random Forest is an ensemble of decision trees, whereas ANNs are interconnected neurons or units. Both methods produce predictions by combining numerous models, leading to enhanced performance and resilience. But ANN's testing F1 score is less than 50%, making it unreliable.

We got 81.69% for Logistic Regression, which was quite similar to Naive Bayes because both are probabilistic models that assess the likelihood of an instance belonging to a specific class. Regression models the log chances of the target variable using a linear mixture of features and a sigmoid function. In contrast, Gaussian Naive Bayes assumes that features are typically distributed and generates class probabilities based on this assumption.

The nearest means model also provides a decent accuracy of 80.56% on the test data, indicating that the data from both classes are clustered together and not too far away, causing the centroids to be positioned in the middle. As a result, this approach accurately classifies more points. SVM using RBF kernel performs very well at separating non-linearly separable data and has obtained accuracies of 81.86, which was very close to perceptron's accuracy of 81.26 on testing. For KNN, we got the lowest accuracy, 78.20 as, which was lower than the baseline. This could be attributed to the training dataset distribution can influence KNN performance. A similar testing dataset distribution as compared to the training dataset in real data may not be the case.

The 5. Libraries used and what you coded yourself

NumPy is a Python module for array manipulation and numerical computations that is freely available. NumPy, which stands for Numerical Python, allows users to work with arrays efficiently, conduct linear algebra and Fourier transform operations, and manipulate matrices.

Pandas is a Python library that provides data structures that are efficient, adaptable, and easy to use when working with labeled or relational data. This was utilized in our project to read CSV files. It is intended to simplify and streamline Python data analysis processes, acting as a fundamental and powerful tool for practical, real-world data analysis.

sklearn.metrics: This module contains a variety of metrics for assessing model performance. Mean_squared_error, for example, computes the mean squared error between expected and actual values.

sklearn.neighbors: This module offers an implementation of the K-Nearest Neighbors method for classification tasks (KNeighborsClassifier). It also includes the NearestCentroid classifier, which classifies using the nearest centroid approach.

sklearn.model_selection: This module contains methods and classes for selecting and evaluating models. KFold is a cross-validation approach that divides training and testing data into K folds. GridSearchCV extensively searches for the optimum combination of parameters for a given estimator.

sklearn.SVM: Support Vector Machines (SVM) are implemented in this module for classification (SVC) and regression tasks. SVMs are binary classification supervised learning models that can be expanded to perform multi-class sorting.

sklearn. Metrics. Pairwise: This module includes routines for calculating pairwise distances and sample similarities. The Radial Basis Function (RBF) kernel is computed between two matrices using rbf_kernel.

sklearn.linear_model: This module contains several linear models for regression and classification problems. Logistic regression is used for logistic regression (classification), whereas linear regression is used for linear regression.

sklearn.utils: This module contains utility functions for rearranging data (shuffle) and doing other operations.

sklearn.cluster: Clustering methods are implemented in this module. KMeans is a well-known algorithm for dividing data into K clusters based on feature similarity.

sklearn.decomposition: This module includes dimensionality reduction methods. PCA (Principal Component Analysis) converts high-dimensional data into a lower-dimensional representation while retaining critical information.

sklearn.preparation: This package contains data preprocessing functions like scaling, encoding, and imputation. StandardScaler is a feature standardizer that removes the mean and scaled to unit variance.

seaborn: While not a scikit-learn component, seaborn is a popular data visualization package that works well with scikit-learn. It offers high-level APIs for creating informative and visually appealing statistics visuals.

Tensorflow.Keras: The TensorFlow library creates and trains deep learning models in this module. It contains classes and functions for defining neural networks (Sequential), layering (Dense), and optimizing models (Adam, SGD, RMSprop).

learn.over_sampling: This module contains methods for dealing with skewed datasets. SMOTE (Synthetic Minority Over-sampling Technique) is a well-known technique for balancing class distribution by creating synthetic samples of the minority class.

Matplotlib.pyplot is a module of the Matplotlib library, a popular Python data visualization package. It has a set of functions for creating and customizing plots and charts. To make it easier to use, the module is usually imported as plt. You can use matplotlib.pyplot to create line plots, scatter plots, bar plots, histograms, and other visual attributes such as labels, titles, colors, and other visual properties.

6. Contributions of each team member

Shivansh Amattya is in charge of data analysis and designing the trivial and baseline models, SVM, Perceptron, and Random Forest models.

Akash Capirala is in charge of data preprocessing and building the ANN, KNN, Naive Bayes, and Logistic Regression models.

Each report section was a collaborative effort with equal input from both sides.

7. Summary and conclusions

This research aimed to use a binary classification job to forecast whether or not clients would default on credit card payments. For six months, the dataset comprised payment information for 30,000 Taiwanese clients, including credit limit, gender, education, marital status, age, repayment status, bill statements, and total amount paid.

Several models were used to solve the classification challenge, including Trivial, Nearest Means, Support Vector Machine, Multilayer Perceptron, Naive Bayes Classifier, Logistic Regression, K nearest neighbors, Random Forest, and Artificial Neural Networks. The training dataset had 27,000 data points, while the testing dataset had 3,000. The training dataset's class distribution was examined better to understand the balance of default and non-default occurrences.

The raw data was cleaned and transformed using preprocessing techniques. Unknown feature labels were substituted with known labels, and data point inconsistencies were investigated. Negative bill amounts were set to zero, and outliers in categorical features were deleted. Data points that met certain criteria were removed. Outliers were identified and eliminated using a quantile-based outlier removal approach.

Linear Discriminant Analysis (LDA) was used for feature selection, extraction, and dimensionality modification. LDA seeks to maximize class separation while conserving class-specific information, and minimizing feature space dimensionality. Standardization was used to guarantee that features of varying scales did not dominate the analysis.

The entire process of the project included the implementation of several models, the use of preprocessing techniques, feature selection and extraction, and the use of cross-validation to choose the optimal model for forecasting credit card payment defaults. The model with the highest accuracy and capacity to generalize well to unseen data, as determined by performance evaluation, would be picked.

The following will be our future projects:

Feature Engineering: Additional feature engineering approaches could be used to derive more meaningful and informative features from the existing dataset. This may entail developing new variables or merging existing ones to capture various credit card usage and payment behavior elements.

Advanced Modeling approaches: While the project has previously deployed multiple machine learning models, modeling approaches continually evolve. More advanced techniques, such as gradient boosting machines, deep learning architectures, or ensemble methods, such as stacking, could increase the predictive performance of the models.

References

1. https://pandas.pydata.org/docs/
2. https://scikit-learn.org/0.21/documentation.html
3. https://scikit-learn.org/stable/modules/clustering.html
4. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
5. https://scikit-learn.org/stable/modules/preprocessing.html
6. https://seaborn.pydata.org/
7. https://www.tensorflow.org/api_docs/python/tf/keras
8. https://imbalanced-learn.org/stable/over_sampling.html
9. https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html
10. https://numpy.org/doc/
11. Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf