Default of Credit Card Prediction

IE7300 : Statistical Learning for Engineering

Spring 2024

Group 12

Date : 19th April 2024

Soumitra Bhagdikar

Rajendra Goparaju

Shubhang Yadav Sandaveni

Vishal Reddy Vookanti

Khushboo Galrani

Abstract

Our goal in this project is to assess how customers in Taiwan typically behave when it comes to making their payments. We aim to identify whether they can be considered reliable or unreliable based on their past payment habits. To achieve this, we plan to create machine learning models using logistic regression techniques. Logistic regression is a valuable tool for classifying customers into two categories like this, as it helps us analyze how different factors influence the likelihood of default. We will concentrate on thoroughly understanding the dataset, preparing it appropriately, and choosing the most relevant features to train our classification models. Furthermore, we will fine-tune the models to ensure they perform at their best. We will also evaluate the models using metrics such as precision and recall.

Introduction

Understanding credit card defaults is essential for banks to minimize risks and maintain stability in today's financial world. Our project centers on analyzing the 'Default of Credit Card Clients Dataset' from the UCI Machine Learning Repository, which provides valuable information on Taiwanese credit card users' behavior between April and September 2005. As technology evolves and market competition intensifies, assessing the accuracy of predictive models in identifying potential defaulters is paramount. Our goal is to assess different classification algorithms to predict credit card defaults effectively. The project aims to analyze data thoroughly to uncover patterns that can help assess credit risk. It will evaluate different models to find the best method for predicting defaults, which will improve credit risk management strategies. The goal is to provide actionable insights through rigorous analysis that can optimize lending practices and reduce risks related to credit card defaults.

Dataset Description

The data consists of 30000 entries with a total of 25 columns, each providing essential information regarding credit transactions, demographic characteristics, repayment history, and payment amounts.

Source Dataset Link: https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients

Below is a detailed description of each column:

ID: Unique identifier for each client.

LIMIT_BAL: The amount of credit extended to the client in New Taiwan (NT) dollars, inclusive of individual and family/supplementary credit.

SEX: Gender of the client, here 1 represents male and 2 represents female.

EDUCATION: Level of education of the client, categorized into :

1: Graduate school

2: University

3: High school

4: Others

MARRIAGE: Marital status of the client, categorized into:

1: Married

2: Single

3: Others

AGE: Age of the client in years.

PAY_0 to PAY_6: Repayment status of the client for the respective months from September 2005 to April 2005. The scale denotes the delay in payment, ranging from -1 (pay duly) to 9 (payment delay for nine months and above).

BILL_AMT1 to BILL_AMT6: Amount of bill statement for the respective months from September 2005 to April 2005 to September 2005 in NT dollars.

PAY_AMT1 to PAY_AMT6: Amount of previous payment made by the client for the respective months from April 2005 to September 2005 in NT dollars.

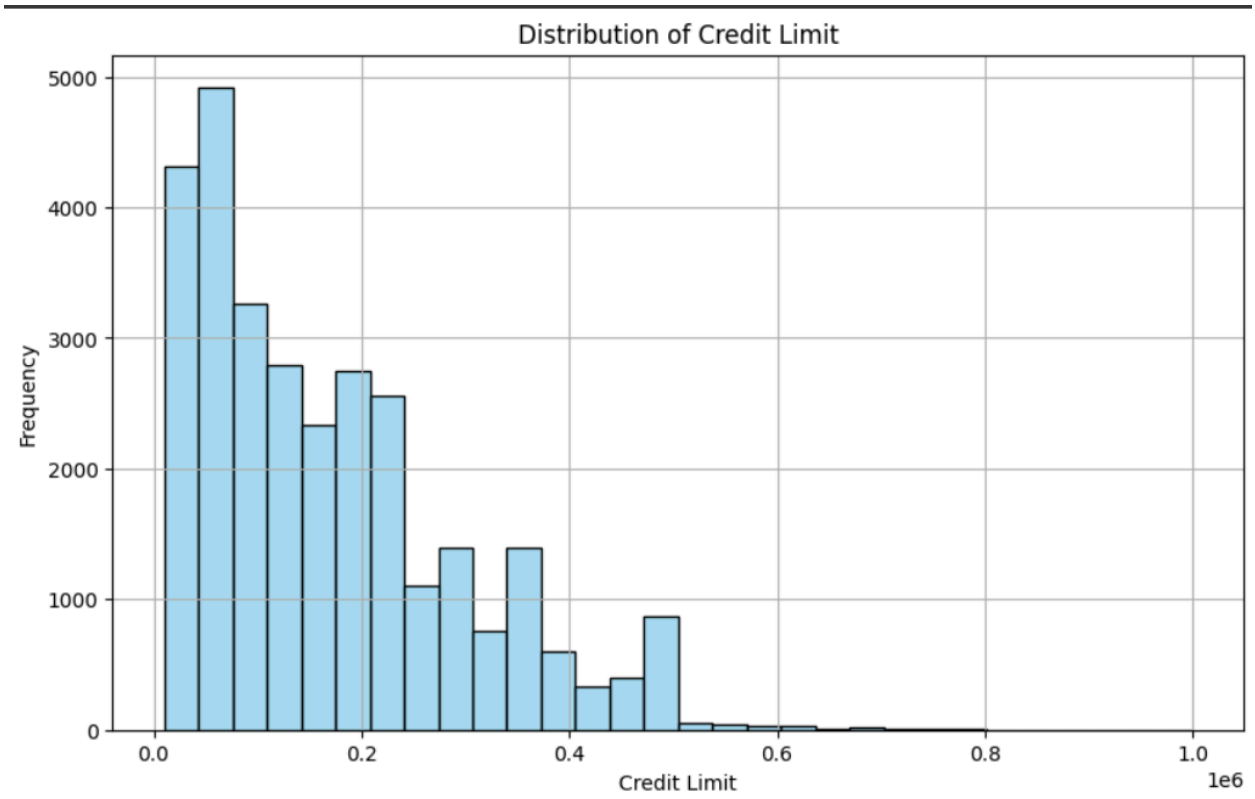
default payment next month: Binary indicator of default payment, where:

1: Default payment (yes)

0: No default payment (no)

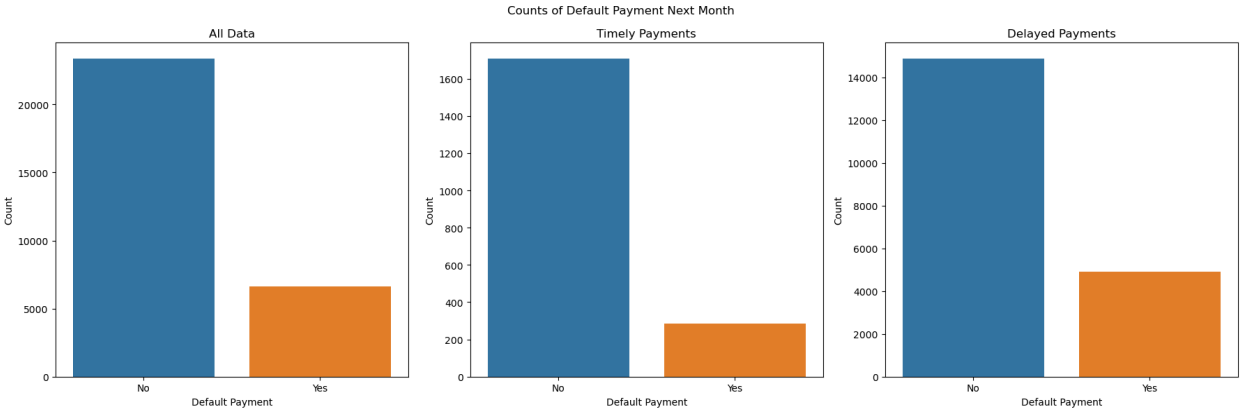
Exploratory Data Analysis

1) Credit Limit



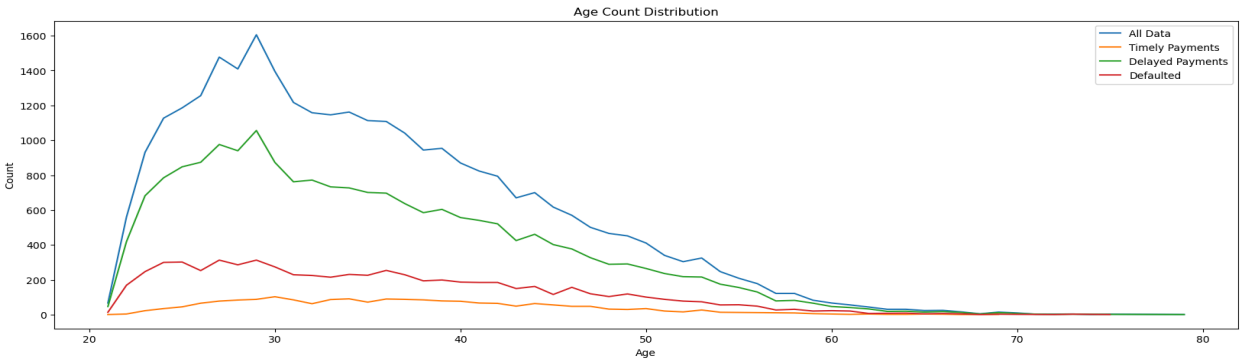
The histogram of credit limit distribution is right-skewed, with the majority of individuals having lower credit limits and a few possessing significantly higher limits. The peak at the lower end suggests common lower credit limit allocations, while the long tail indicates the presence of outliers with substantial credit limits. This pattern illustrates a concentration of lower credit limits within the customer base.

2) Default Payments



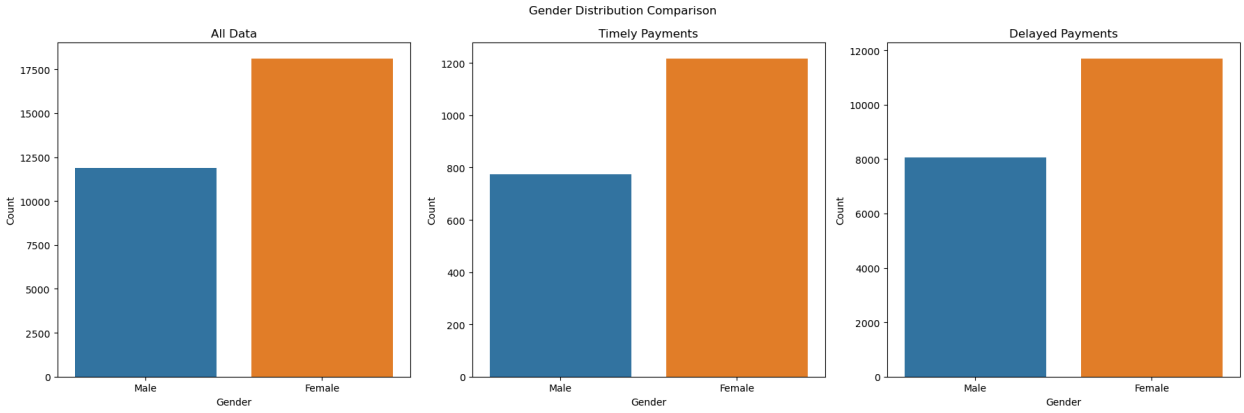
- The graph illustrates the distribution of population across three distinct groups: the entire population, individuals who consistently make timely payments, and those who frequently experience delays in payment.

3) Age Distribution



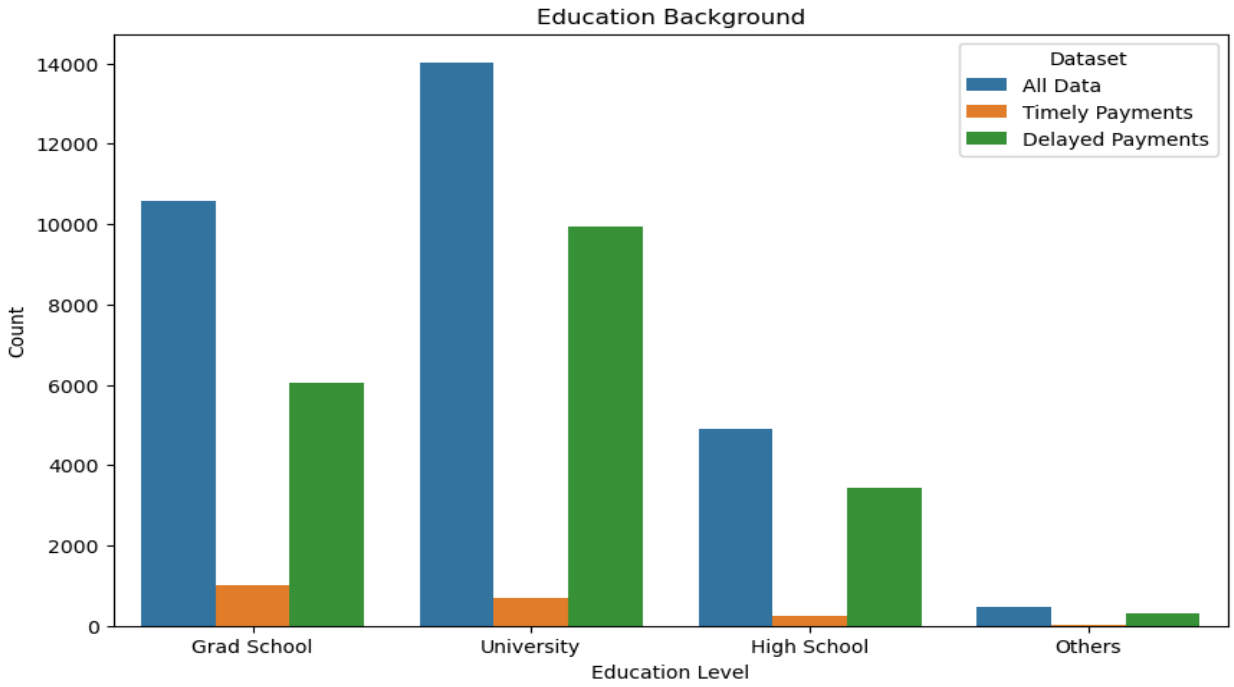
- The dataset comprises individuals aged between 20 and 40 years. Notably, within this age bracket, there is a notable prevalence of delayed payments compared to older age groups.

4) Gender Count



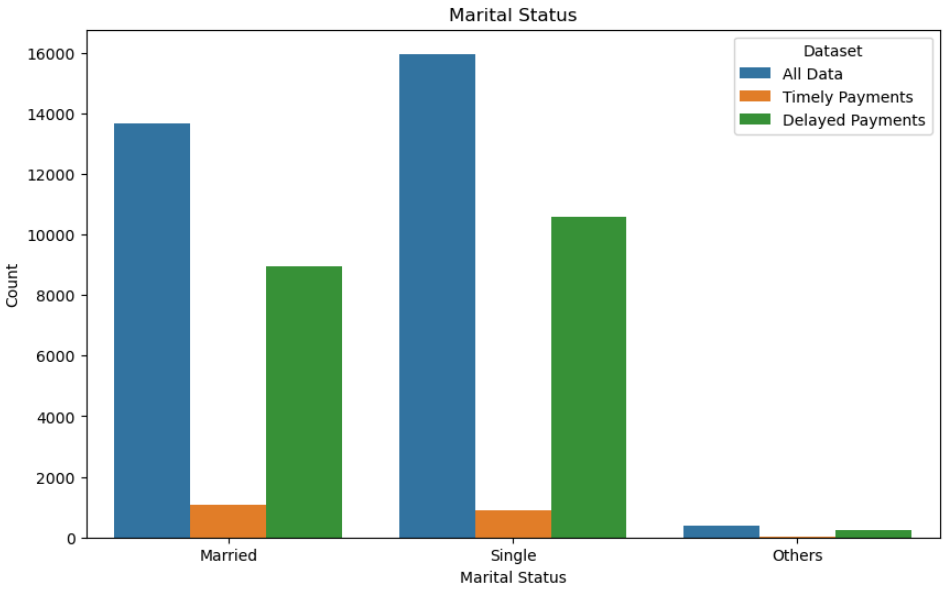
- The visualization depicts the gender distribution among three distinct cohorts: the general population, individuals consistently meeting payment deadlines, and those frequently encountering delays in payment. It is evident that the dataset comprises a higher proportion of females compared to males. Notably, approximately 68% of females and 66% of males experience delays in payment, while only 6.8% of both genders consistently pay their dues on time each month.

5) Education Level



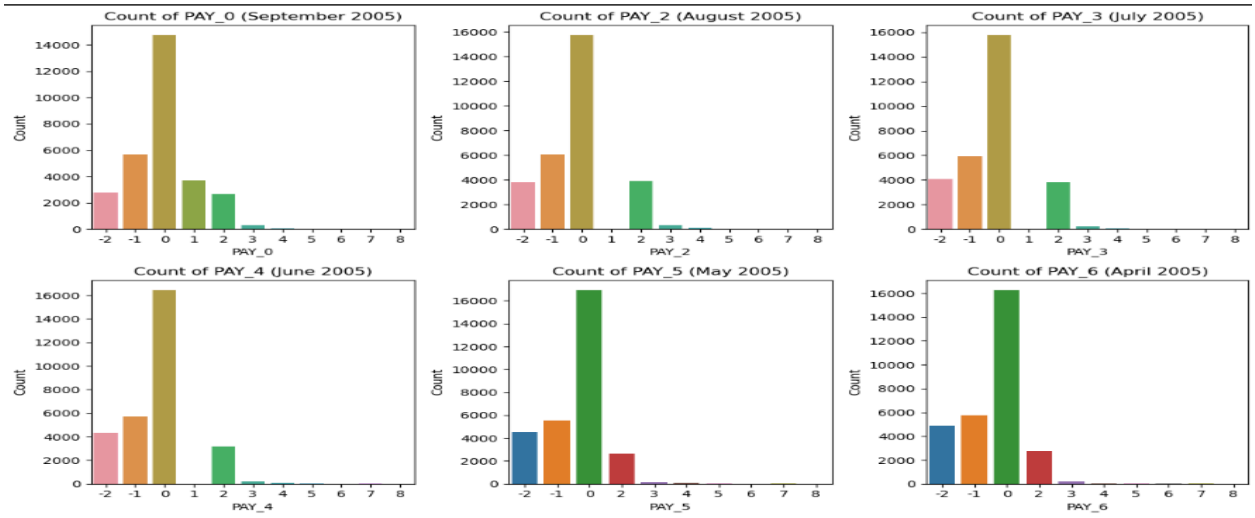
- The visual representation depicts the educational background distribution within three distinct segments: the general population, individuals maintaining consistent punctuality in payments, and those prone to frequent delays in payment. It is notable that a predominant portion of individuals ensuring timely payments hold graduate degrees. Conversely, a considerable proportion of individuals experiencing delays in payment are University graduate

6) Marital Status



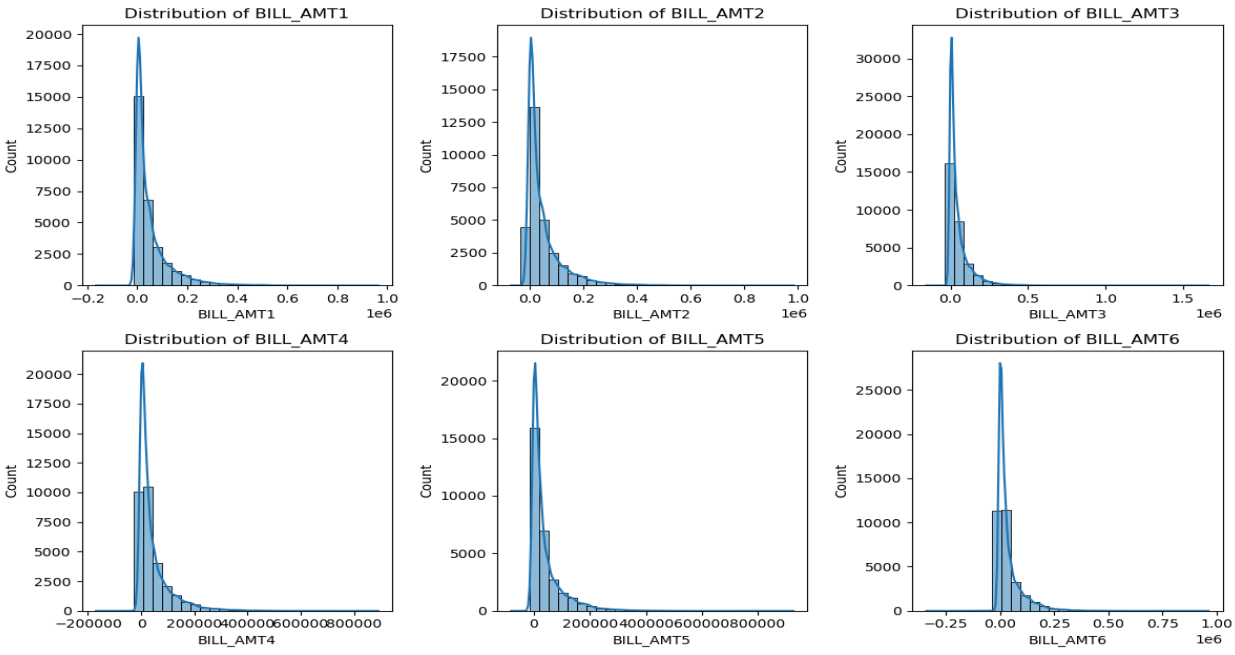
- The visualization portrays the marital status distribution across three distinct segments: the overall population, individuals consistently meeting payment deadlines, and those frequently encountering delays in payment. It is discernible that married individuals constitute a minority within the dataset, with the majority categorized as single or falling into other marital statuses. Notably, single individuals are prominent among those ensuring timely payments, while the "others" category predominantly comprises individuals experiencing delays in payment.

7) Delays in payment



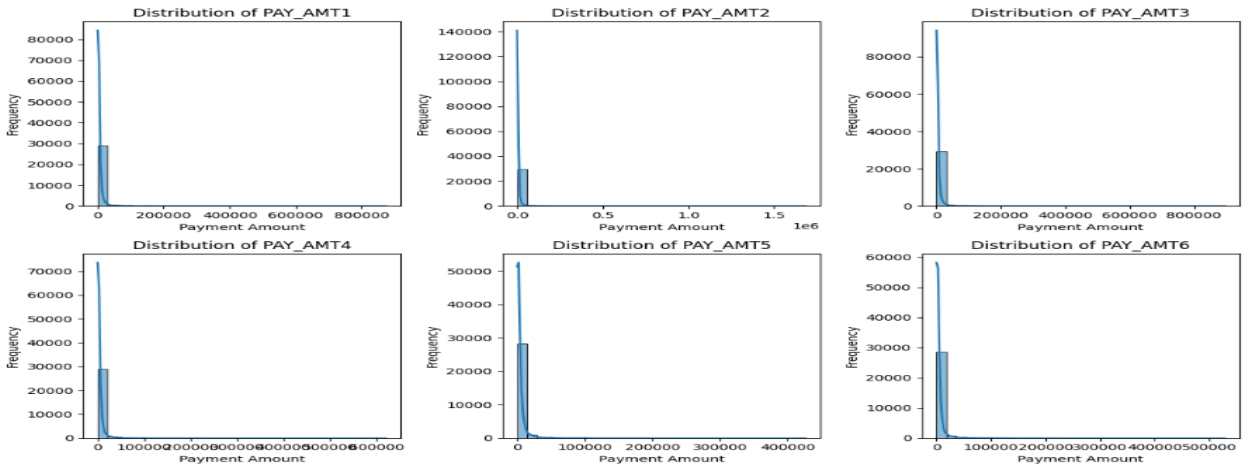
- The visualization above illustrates the count of individuals with each status of delay across the span of six months

8) Bill amount over months (Sept 2005 to April 2005)



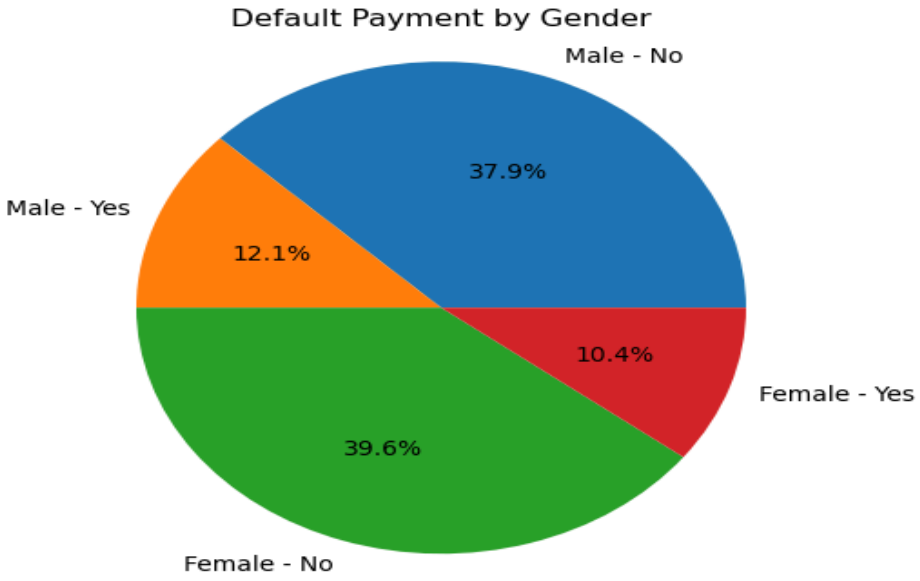
- The visualization above illustrates the credit card bills across the span of six months

9) Paid amount over months (Sept 2005 to April 2005)



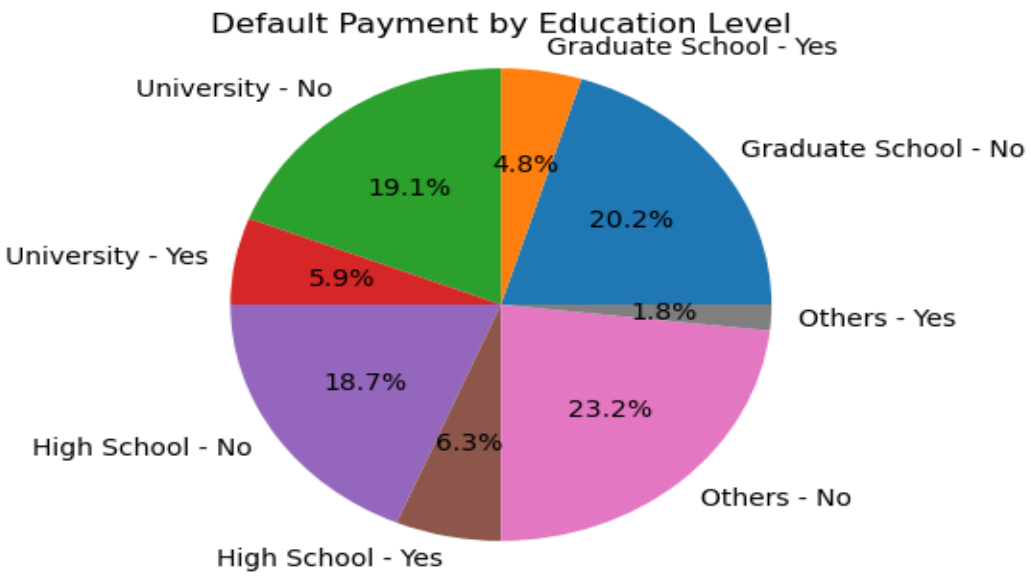
- The visualization above illustrates the credit card bills paid across the span of six months.

10) Gender & Default Payment



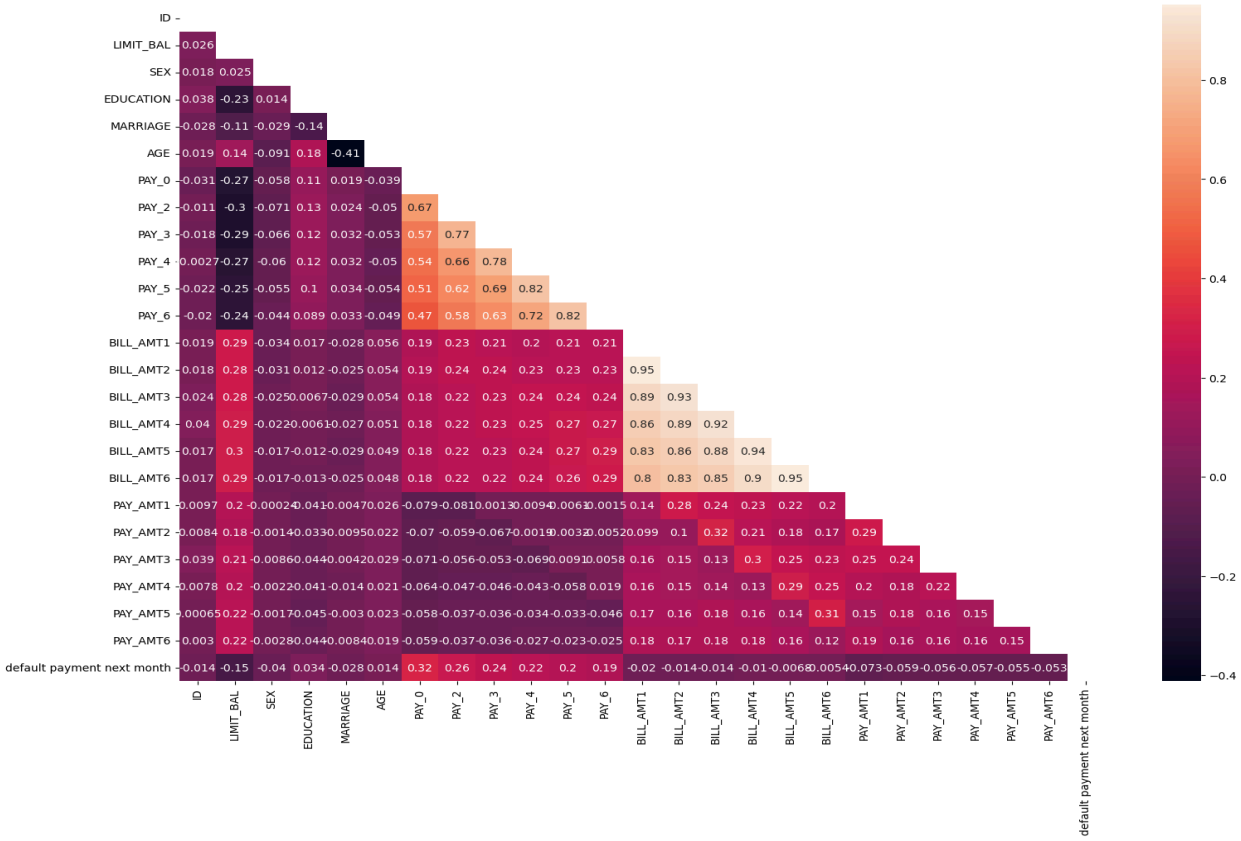
- The pie chart presented above showcases the gender distribution in relation to default payment status. It reveals that a majority of males default on their payments, whereas a majority of females consistently fulfill their payment obligations.

11) Education vs Default Payment



- The pie chart displayed above depicts the distribution of education levels concerning default payment status. It highlights that the majority of individuals with a high school education tend to default on their payments, while the majority of individuals from non-specified categories exhibit a pattern of non-defaulting behavior.

12) Correlation Matrix



A lower triangular correlation matrix, wherein each cell represents the correlation coefficient between two variables. This coefficient measures the strength and direction of the relationship between the variables:

- A correlation coefficient of 1.0 indicates a perfect positive correlation, signifying that both variables increase together.
- A coefficient of -1.0 denotes a perfect negative correlation, indicating that one variable increases as the other decreases.
- A coefficient of 0 suggests no correlation, implying no linear relationship between the variables.

Correlation serves as a statistical tool to quantify how closely two variables change together, with the correlation coefficient varying between -1 and 1.

Method

Logistic Regression Model

Our analysis centers on predicting whether individuals will default on their credit card payments, a problem inherently suited to logistic regression for several compelling reasons:

Appropriateness for Binary Outcomes: The target variable, 'default' or 'no default', is binary. Logistic regression is tailored for such dichotomous variables, modeling the log odds of the dependent variable, thereby providing a clear probabilistic framework.

Model Transparency: The financial sector values transparency, and logistic regression offers interpretable results where each predictor's impact on the outcome is quantified. This transparency aids stakeholders in understanding how predictors influence risk, an essential factor in strategic decision-making.

Efficiency and Scalability: Logistic regression is not computationally demanding, making it an excellent choice for both prototyping and large-scale deployment, allowing for quick adjustments and updates to the model in operational environments.

Benchmarking Capabilities: It serves as an excellent baseline model, providing a benchmark for evaluating the performance of more sophisticated models, which can be crucial in justifying the deployment of more complex algorithms.

Data Pre-Processing

Outliers

In the preprocessing phase, we identified outliers within the dataset. However, given the financial nature of the data specifically credit card transactions and bill amounts each outlier was carefully evaluated

Handling the null values and Feature selection

- The dataset exhibited no missing values, eliminating the need for imputation and ensuring a complete set of records for the analysis.
- We enhanced model efficiency by eliminating non-predictive attributes, including the 'ID' column, to focus solely on variables relevant to predicting credit card defaults.
- All data was converted to integer format, standardizing the data type across the dataset for consistency in subsequent analytical processes.

Standardization of Continuous Variables: For the continuous data fields, we implemented standardization using StandardScaler from scikit-learn. This approach adjusts the features to have a mean of zero and a standard deviation of one, ensuring that each feature contributes equally to the model without being dominated by features with larger scales.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_standardized = scaler.fit_transform(continuous_data)
```

Balancing the dataset:

Class Imbalance Correction: To explore the effect of class distribution on model performance, we implemented two different sampling strategies:

Case 1 - Oversampling : Enhancing Model Robustness: Training and Evaluation with Balanced Datasets

Using the Synthetic Minority Over-sampling Technique (SMOTE), we increased the number of instances in the minority class by creating synthetic samples. This approach aims to provide a balanced dataset that helps the model learn to identify the minority class more effectively.

```
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.model_selection import train_test_split
import numpy as np

# Assuming x and y are your features and target variable respectively
X_train, X_test, Y_train, Y_test = train_test_split(x, y, stratify=y, test_size=0.3)

# Dynamically adjust k_neighbors based on the class distribution
smote = SMOTE(k_neighbors=20)

# fit target and predictor variable
x_smote , y_smote = smote.fit_resample(X_train, Y_train)
print()
print('Original dataset shape:', Counter(Y_train))
print('Resampled dataset shape:', Counter(y_smote))
```

```
Original dataset shape: Counter({0: 16355, 1: 4645})
Resampled dataset shape: Counter({0: 16355, 1: 16355})
```

Case 2- Undersampling: We reduced the size of the majority class to match the minority class, aiming to eliminate the model's bias toward the majority class by providing an equal representation of both classes

```
# import library
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(random_state=42, replacement=True)

# fit predictor and target varialbe
x_rus, y_rus = rus.fit_resample(X_undersample_train,Y_undersample_train)

print('original dataset shape:', Counter(Y_undersample_train))
print('Resample dataset shape', Counter(y_rus))
```

```
original dataset shape: Counter({0: 16380, 1: 4620})
Resample dataset shape Counter({0: 4620, 1: 4620})
```

Principal Component Analysis (PCA):

The PCA process was applied to the oversampled and undersampled datasets. The transformation to a lower-dimensional space was performed with the aim to retain as much variance as possible while reducing the number of input variables for the logistic regression models.PCA Implementation:

We developed a custom PCA class to perform dimensionality reduction, encapsulating the steps in an object-oriented manner. The PCA algorithm proceeded as follows:

1.Standardization: Prior to PCA, continuous data were standardized using StandardScaler to ensure that the PCA's assessment of variance was not unduly influenced by the scale of different features.

2.Covariance Matrix Computation: Calculated the covariance matrix to understand how features vary in relation to one another.

3.Eigen-decomposition: Performed eigen-decomposition on the covariance matrix to identify principal components.

4.Component Selection: Selected a number of principal components such that 95% of the variance in the data could be explained. This decision was made to maintain a balance between data compression and information retention.

5.Projection: Projected the original data onto the new feature space created by the selected principal components.

Model Implementation

In our project, we have used logistic regression to train and evaluate using two distinct balanced datasets:

These datasets were generated using two common techniques: Synthetic Minority Over-sampling Technique (SMOTE) and Random Under-Sampling.

Train and Evaluate Balanced Dataset using SMOTE

The balanced dataset was prepared by ensuring an equal number of instances for each class label. To address the binary classification challenge of predicting credit card defaults on top of the balanced dataset, we developed the LogisticRegressionWithRegularization class. This custom implementation was necessary due to the unique nature of credit card default data, which often exhibits nonlinear relationships and the parameters were carefully selected and calibrated to develop a model that is sensitive to the complexities of credit card default data.

The rationale for each parameter is as follows:

Learning Rate (learning_rate): The learning rate parameter regulates the step size in weight updates, crucial for avoiding overshooting loss function, which is essential for a data set like credit card defaults, where each step needs to be carefully calculated to avoid overstepping into less optimal solutions.

Number of Iterations (num_iterations): This parameter specifies the number of passes over the entire data set that the algorithm will take to converge to the best weights. The credit default data, which can exhibit complex patterns, may require multiple passes for the model to learn effectively.

Regularization Strength (lambda_param): Regularization, with the lambda parameter controlling its strength, penalizes larger coefficients, preventing overfitting by discouraging overly complex models. In credit default data, where predictive accuracy is crucial, regularization helps generalization to unseen data, maintaining model reliability.

Regularization Type (regularization): The model allows for the option of 'ridge' (L2) or 'lasso' (L1) regularization. Ridge regularization is useful for handling multicollinearity and keeping model interpretability by shrinking coefficients evenly. Lasso regularization can select more relevant features by reducing the coefficients of less important features to zero.

Weights (weights): Initially set to None, the weights are learned from the data during the training process. They represent the coefficients for each feature in the model, determining how each predictor variable will influence the prediction of credit default.

Bias (bias): Similar to weights, the bias term is learned from the data and represents the intercept of the model, accounting for the effect on the output not captured by the feature variables. In credit default prediction, this term adjusts the threshold at which the decision boundary is set.

Implementation Details

The class implements a sigmoid function to map predicted values to probabilities, a fit method to train the model, a predict method to classify new data, and a predict_proba method to assess the probability of default. The fit method also incorporates a custom loss calculation that accounts for the chosen regularization to ensure the model appropriately penalizes complexity in favor of a generalizable solution.

Here, we have used the dataset that has been balanced using SMOTE. In machine learning, selecting appropriate hyperparameters is crucial for model performance. This report details a systematic approach to find the optimal hyperparameters for a logistic regression model with

regularization.Our project investigates the impact of learning rate, lambda parameter, and regularization technique on model accuracy.

Model Training and Evaluation using SMOTE:

1. A grid search is performed, iterating through all combinations of learning rates, lambda parameters, and regularization techniques.

In our case, grid search refers to the systematic exploration of a predefined set of hyperparameters to identify the combination that yields the best model performance. Here's how grid search is implemented in our project.

a. **Defined Hyperparameters**: The hyperparameters under consideration are learning rate, lambda parameter, and regularization technique. Learning rates are chosen from the list [0.1, 0.01, 0.001].

b. **Lambda parameters**:Three lambda parameters are examined and they are selected from [0.1, 0.01, 0.001].

c. **Regularizations**: Regularization techniques include 'ridge', 'lasso', and no regularization. Iterating through Combinations: Nested loops are used to iterate through all combinations of these hyperparameters. For each combination of learning rate, lambda parameter, and regularization technique, a logistic regression model is trained and evaluated.

2. Inside the nested loops, a logistic regression model is instantiated with the current hyperparameters. The model is trained using the transformed training data (X_train_transformed, y_smote).

3. Predictions are made on the test data (X_test_transformed) using the trained model. Accuracy is calculated as the mean of correct predictions. Train and validation losses are computed using the model's compute_loss method.
4. Identifying the Best Parameters: During each iteration, if the accuracy obtained with the current hyperparameters surpasses the previous best accuracy, the current hyperparameters are recorded as the new best parameters.

5. Output: For each combination of hyperparameters, the code prints the accuracy achieved. At the end of the grid search, the code prints the best parameters found based on the highest accuracy observed during the search.

Train and Evaluate Balanced Dataset using Under Sampling:

In alignment with our approach to enhancing model robustness through dataset balancing, undersampling techniques were employed to address class imbalance. Following the undersampling procedure, the resulting dataset underwent the same rigorous training and evaluation process as previously discussed for the balanced dataset. This comprehensive methodology ensured a consistent evaluation framework, allowing for a thorough comparison of model performance between balanced and undersampled datasets.

Results and Comparison

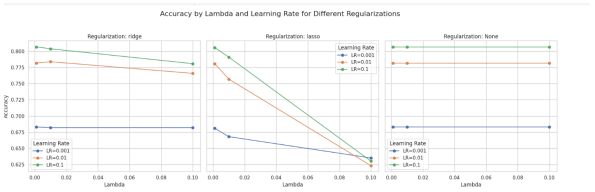
We have evaluated the performance of a logistic regression model with regularization using various metrics and visualizations.

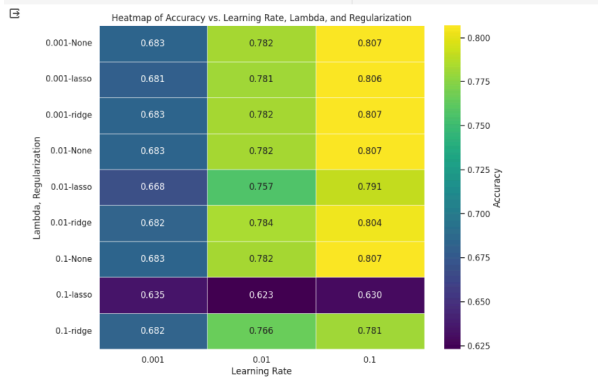
1. Confusion Matrix and Metrics: Confusion Matrix Computation: A custom function, compute_confusion_matrix, calculates the confusion matrix for binary classification. The matrix includes True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).
2. Metrics Calculation: Metrics such as Precision, Recall, and F1 Score are computed using the confusion matrix.

a. Precision is the proportion of true positive predictions among all positive predictions.
b. Recall (Sensitivity) measures the proportion of actual positives that were correctly identified.
c. F1 Score represents the harmonic mean of precision and recall, providing a balance between the two metrics. ROC-AUC Score: The ROC-AUC score, a measure of model performance across various threshold settings, is calculated using the sk-learn library.
d. Visualization: Model Accuracy vs. Regularization Parameter: A plot illustrates the relationship between model accuracy and the regularization parameter (lambda) for different regularization techniques (Ridge, Lasso, None). Accuracy is computed across varying lambda values to understand the impact of regularization on model performance.

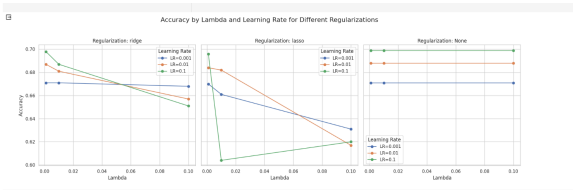
The below provided graphs compare the accuracy of a machine learning model across different lambda values and learning rates under three types of regularization: ridge, lasso, and none.

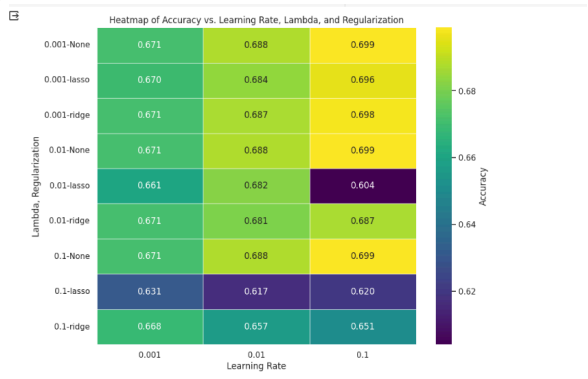
SMOTE





UNDERSAMPLING





- Overall Accuracy: The accuracy levels in the first graph for the dataset that has been balanced using SMOTE are generally higher than those in the second graph for all types of regularization and learning rates. This could indicate a difference in the underlying data, model complexity, or a feature set that affects the overall performance of the model.
- Learning Rate Impact: The impact of the learning rate varies with the type of regularization. For ridge regularization, a higher learning rate (0.1) consistently results in higher accuracy in both graphs. Under lasso regularization, the trends are less consistent, with different learning rates performing better at different lambda values.
- Model Generalization: In the first graph for the dataset that has been balanced using SMOTE, the model with ridge regularization and a learning rate of 0.1 seems to generalize better across different lambda values compared to other combinations. In the second graph for the dataset that has been balanced using under sampling, the ridge regularization still appears to perform better, but the difference in accuracy between learning rates is narrower, especially at lower lambda values.

Bias Variance Tradeoff

The bias-variance tradeoff describes the problem of simultaneously minimizing two sources of error that prevent supervised learning algorithms from generalizing beyond their training set.

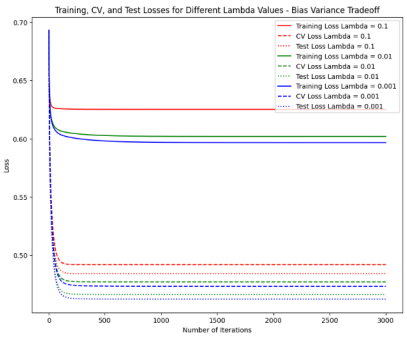
The bias is an error from erroneous assumptions in the learning algorithm. High bias can cause the model to miss the relevant relations between features and target outputs (underfitting). The variance is an error from sensitivity to small fluctuations in the training set. High variance can cause overfitting: modeling the random noise in the training data rather than the intended outputs.

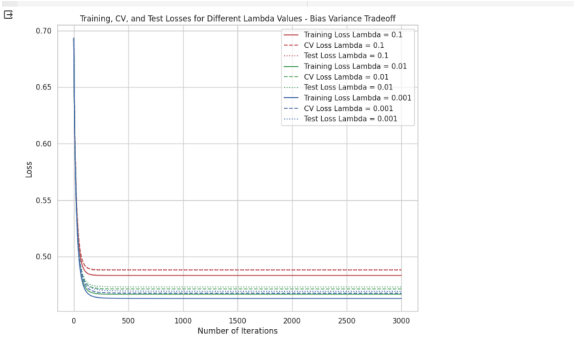
We have divided both the balanced datasets that have been generated using SMOTE and undersampling into three different types.

1. Training Dataset: Used to fit the machine learning model. Ideally, the model learns the underlying patterns in the data. However, if the model is too simple (high bias), it may not learn these patterns well, leading to underfitting. If the model is too complex (high variance), it might learn the noise in the data as well, leading to overfitting.
2. Cross-Validation (CV) Dataset: Utilized to tune hyperparameters, such as the regularization strength (lambda). The model is not trained on this dataset. Instead, it's used to validate the performance of the model with different hyperparameters to find the right balance between bias and variance. The goal is to select hyperparameters that allow the model to generalize well to new, unseen data, not just to perform well on the training data.
3. Test Dataset: Once the best hyperparameters are chosen using the training and CV datasets, the model's performance is assessed on the test dataset. This is unseen data that plays no role in model training or hyperparameter tuning. It provides an unbiased evaluation of the final model fit on the training data.

The graphs shown below demonstrates the concept of bias-variance tradeoff in machine learning through the lens of loss minimization for different lambda values with respect to the number of iterations for two different types of balanced datasets.

SMOTE **UNDERSAMPLING**





- The first graph for the dataset that has been balanced using SMOTE, demonstrates the classic bias-variance tradeoff: as lambda decreases, the model fits the training data better (lower training loss) but runs the risk of fitting too closely to the training data and not generalizing well (higher CV/test loss).

- In the second graph for the dataset that has been balanced using under sampling, the effect of lambda on the bias-variance tradeoff is also evident, but the tradeoff is less pronounced since the test and CV losses remain closer to the training losses across different lambdas.

Summary

The dataset that has been balanced using SMOTE and trained using logistic regression excels in accuracy, precision, recall, and F1 score, making it generally more reliable for scenarios where accurate identification of positive cases is crucial, and the cost of false positives is high.