## Comparative Analysis of Machine Learning Models for Credit Score Prediction

Ayushi Srivastava, Harsh Mehta, Hua Jiang, Yumu Chen

Apr 26, 2024

**Abstract**

In the rapidly evolving financial landscape, credit score prediction remains a pivotal factor in determining the creditworthiness of individuals and enterprises. Traditional credit scoring systems often rely on historical financial data and linear statistical methods, which may not capture complex patterns and non-linear interactions in the data. This paper explores the application of advanced machine learning (ML) models to enhance the accuracy and efficiency of credit score predictions. We compare several ML techniques, including logistic regression, decision trees, support vector machines, Naive Bayes, and KNN, using a dataset that encompasses a broad spectrum of demographic and financial attributes. Our results demonstrate that certain ML models, notably decision tree and KNN, significantly outperform traditional methods in predicting credit scores. Our findings suggest that ML models hold substantial promise for transforming credit scoring processes by providing more accurate, reliable, and equitable evaluations of creditworthiness.

# **Introduction**

The advent of machine learning in financial applications has significantly enhanced the prediction accuracy of credit scores, a vital metric in assessing creditworthiness. Our project leverages a multi-class classification approach to provide a nuanced analysis of credit scores using five distinct algorithms. The overarching goal is to uncover which factors most significantly influence credit scores and determine the most effective prediction model. This report encapsulates our methodology, from data preprocessing to model evaluation, offering insights into the predictive power of each algorithm.

This paper embarks on a comparative analysis of various machine learning algorithms for the prediction of credit scores. Leveraging a diverse dataset encompassing a spectrum of financial attributes, including payment history, debt-to-income ratio, and credit utilization, we delve into the efficacy of five distinct models: logistic regression, decision tree, Naive Bayes, Support Vector Machine (SVM), and k-Nearest Neighbors (KNN).

Our baseline model, logistic regression, serves as a benchmark against which we measure the performance of the alternative algorithms. Through rigorous experimentation and evaluation, we seek to elucidate the strengths and limitations of each model, shedding light on their applicability and robustness in credit score prediction tasks.

As we navigate through the intricacies of each algorithm and scrutinize their predictive capabilities, our endeavor is not only to ascertain the most adept model for credit score prediction but also to contribute to the broader discourse on the integration of machine learning in financial decision-making processes. By elucidating the nuances of predictive modeling in the realm of credit assessment, we aim to empower stakeholders with insights that facilitate more informed and effective lending practices.

# **Related Work**

Prior research in this domain has primarily focused on binary classification problems, distinguishing between good and bad credit scores. However, our approach extends this by employing a multiclass classification framework, offering a more nuanced understanding of credit score categories. Previous studies have utilized similar datasets to predict creditworthiness but often with a limited scope of algorithms. Our work aims to build on these foundations, applying a broader range of models to glean deeper insights.

While researching the previous works on credit scoring using machine learning, one paper published in May 2020 (Kuppili)[1] has discussed the current development of using Spiking Extreme Learning Machine (SELM) in the field. Although not directly related to the primal models being used for this project, it is still intriguing to compare the model effectiveness of our project versus advanced models such as SELM.

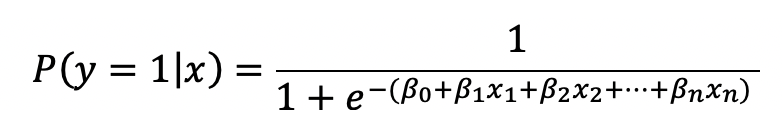
1. **Model Design**

In this section, we provide a comprehensive description of the mathematical formulations and algorithmic implementations of the machine learning models utilized in our study for credit score prediction. Each model is meticulously detailed, encompassing the underlying mathematical principles, algorithmic procedures, and key parameters involved in their execution. By elucidating the intricacies of logistic regression, support vector machines (SVM), decision trees, k-nearest neighbors (KNN), and Naive Bayes classifiers, we aim to provide a clear understanding of their methodologies and functionalities in the context of credit score prediction

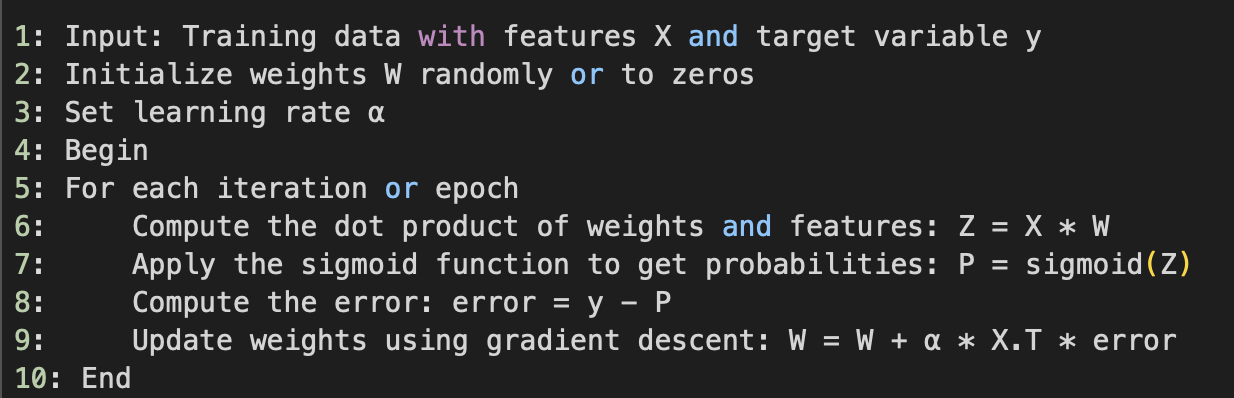
## **3.1 Logistic Regression**

Logistic Regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although more complex extensions exist for multi-class problems. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

For a binary classification problem with a binary dependent variable *y* and independent variables *x1 , x2 , ….xn*, the logistic regression model can be represented mathematically as:



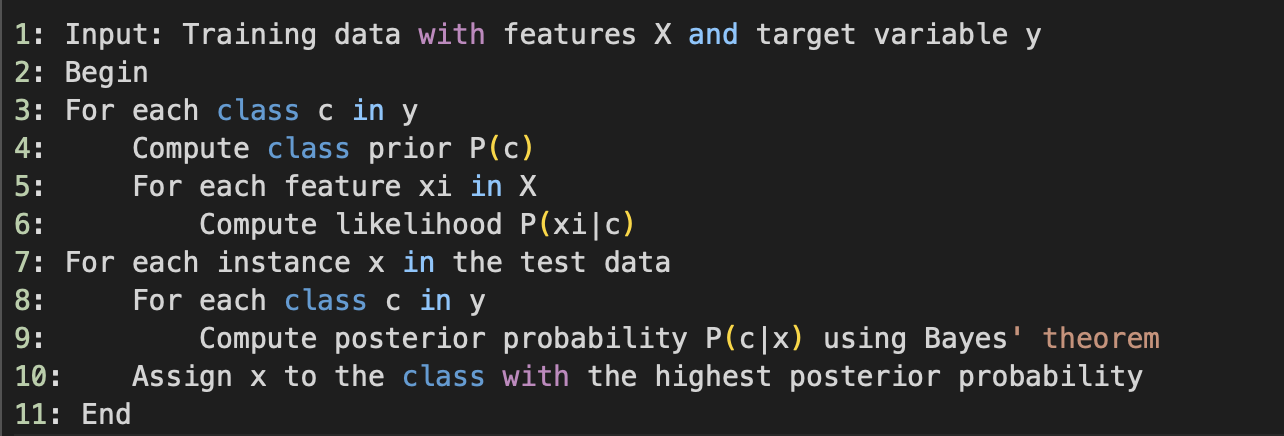
The algorithm for logistic regression is as follows:



**3.2 Naive Bayes**

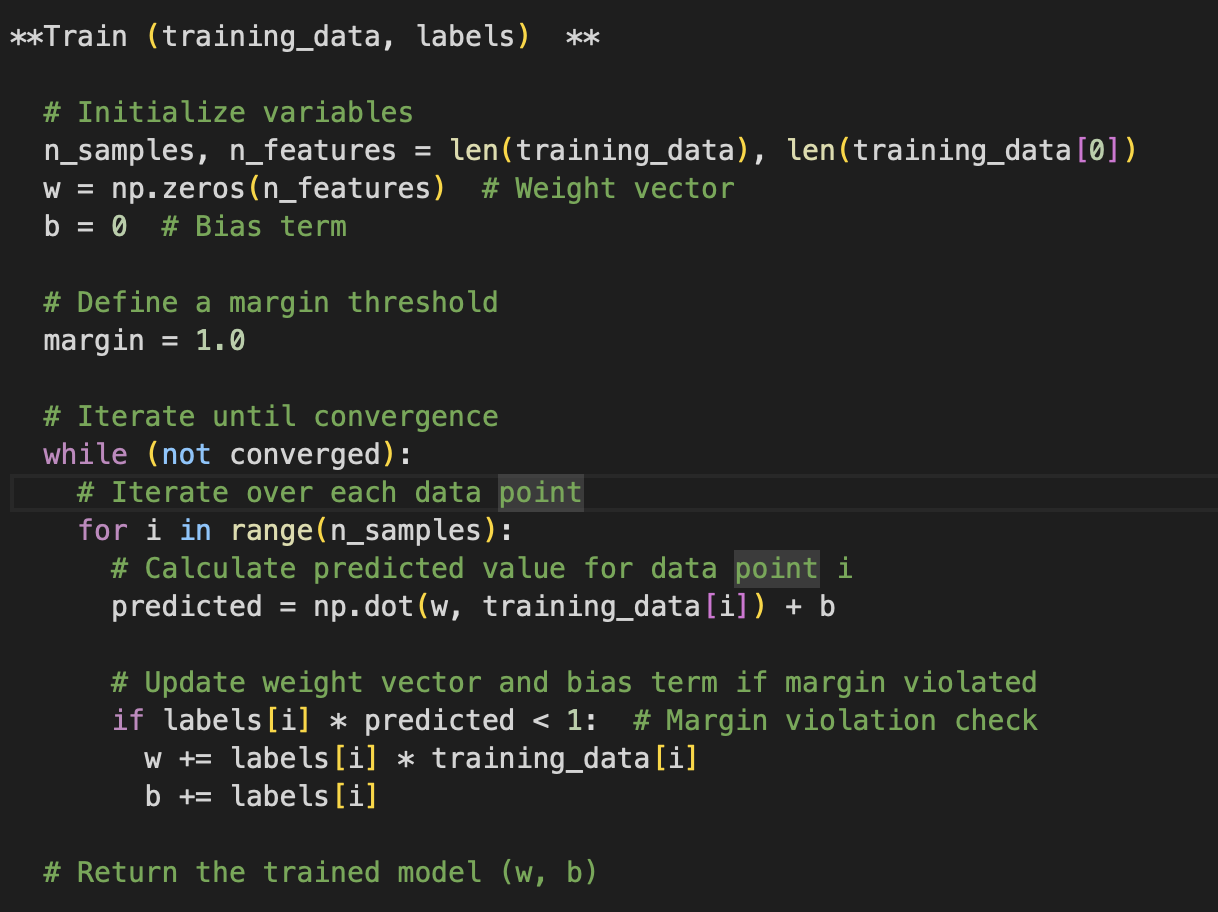
Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayes operates on the principle of conditional probability, calculating the probability of a label given the features using Bayes' theorem. It assumes that the features are conditionally independent given the class label, which simplifies the calculation.It is highly scalable and is known for its simplicity and effectiveness in handling large datasets. Given a feature vector *X=( x1 , x2 , ….xn )* ,and a class label *y*, the probability of *y* given *X* is computed using Bayes' theorem.

The algorithm for Naive Bayes is as follows:



**3.3 Support Vector Machine (SVM)**

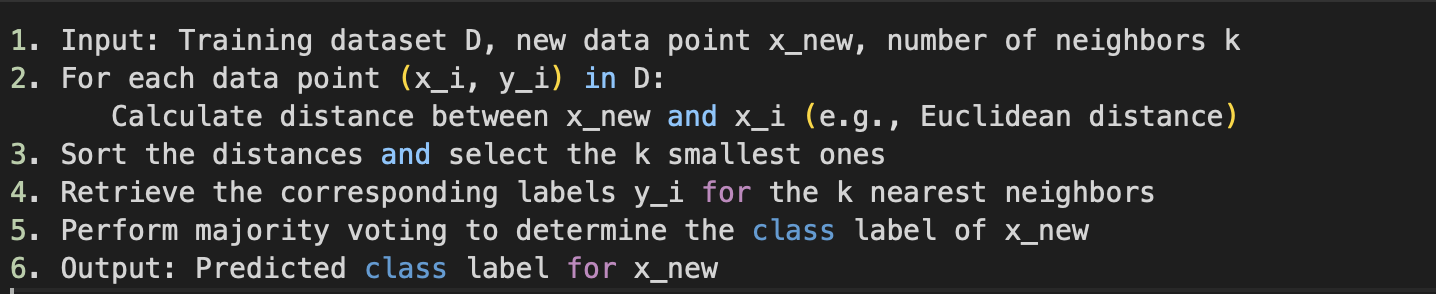
SVM is a powerful classification method that works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. A separator between the categories is found, then the data is transformed in such a way that the separator could be drawn as a hyperplane. Given a set of training data points {(xi,yi)}i=1n where *xi* represents the feature vector and *yi* represents the class label (either -1 or 1 for binary classification), SVM aims to find the optimal hyperplane *wTx+b=0* that maximally separates the classes while minimizing the margin. The algorithm for SVM is as follows:



**3.4 K-Nearest Neighbors (KNN)**

KNN is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. It classifies objects based on their proximate neighbors’ classes. The main concept behind k-nearest neighbors is as follows. Given a point whose class we do not know, we can try to understand which points in our feature space are closest to it. These points are the k-nearest neighbors. Since similar things occupy similar places in feature space, it’s very likely that the point belongs to the same class as its neighbors. Based on that, it’s possible to classify a new point as belonging to one class or another.

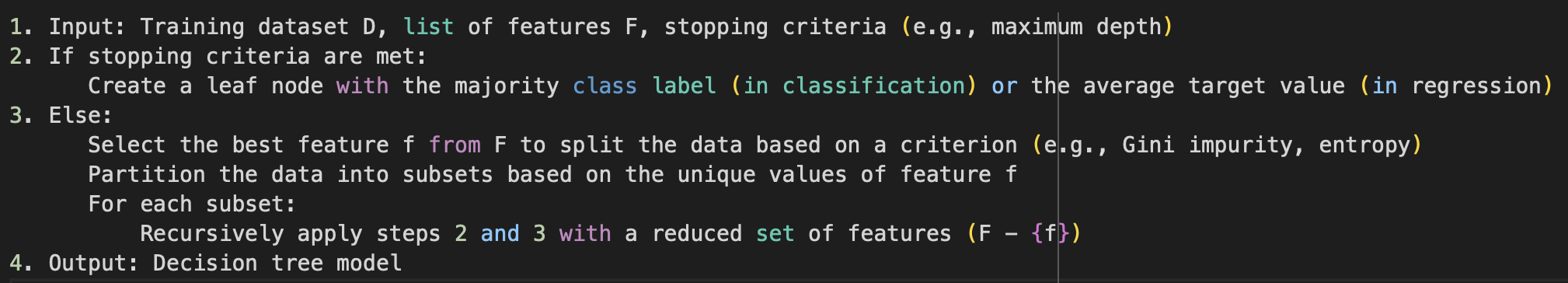
Given a set of labeled training data *D={(x1,y1),(x2,y2),...,(xn,yn)}*where *xi* represents the feature vector of the i-th data point and *yi* is its corresponding class label, there is a new data point *xnew* for which we want to predict the class label. We calculate the distance between  *xnew* and each data point xi in the training set. Common distance metrics include Euclidean distance, Manhattan distance, or Minkowski distance. Next step is to select the *k* data points from the training set that are closest to xnew​ based on the computed distances. The class label for xnew is then determined by taking a majority vote among the class labels of its *k* nearest neighbors.



**3.5 Decision Tree**

Decision trees are a non-linear predictive modeling tool that can be used for both classification and regression. The model learns to predict values of the target variable by learning simple decision rules inferred from the data features. A decision tree builds model predictions in the form of a tree structure. It divides the dataset into branches, which represent an ensemble of decision points about the main characteristics affecting the outcome.

Initially, the algorithm selects the best feature to split the data based on a certain criterion, such as Gini impurity, entropy, or information gain. This feature selection process aims to find the most informative attribute that separates the data into distinct subsets.Once a feature is selected, the algorithm partitions the data into subsets based on the unique values of the chosen feature. This splitting process is repeated recursively for each subset until a stopping condition, like maximum depth or no more features to split, is met. At each leaf node of the tree, the algorithm assigns a class label (in classification) or a predicted value (in regression) based on the majority class or the average value of data points in that node. This ensures that the Decision Tree provides meaningful predictions for new instances based on the learned patterns in the training data.



# **Experimental Results**

**4.1 Dataset**

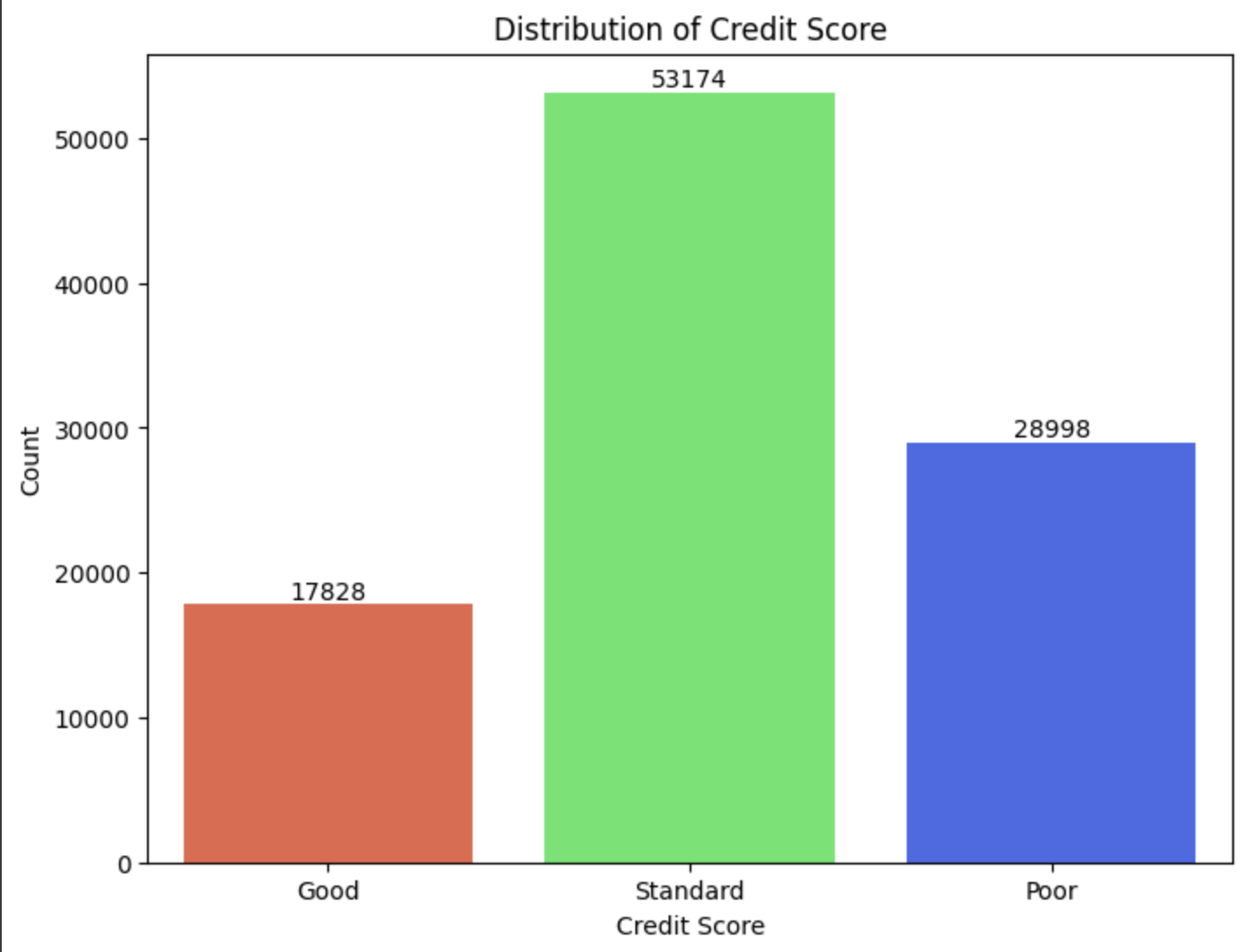
The dataset contains 150,000 entries and 28 attributes, encompassing a range of financial and personal information. The preprocessing approach taken for the dataset in this machine learning project was thorough and multifaceted, focusing on data cleaning, managing missing values, conducting exploratory data analysis (EDA), and encoding features for model readiness.

**4.2 Exploratory data analysis**

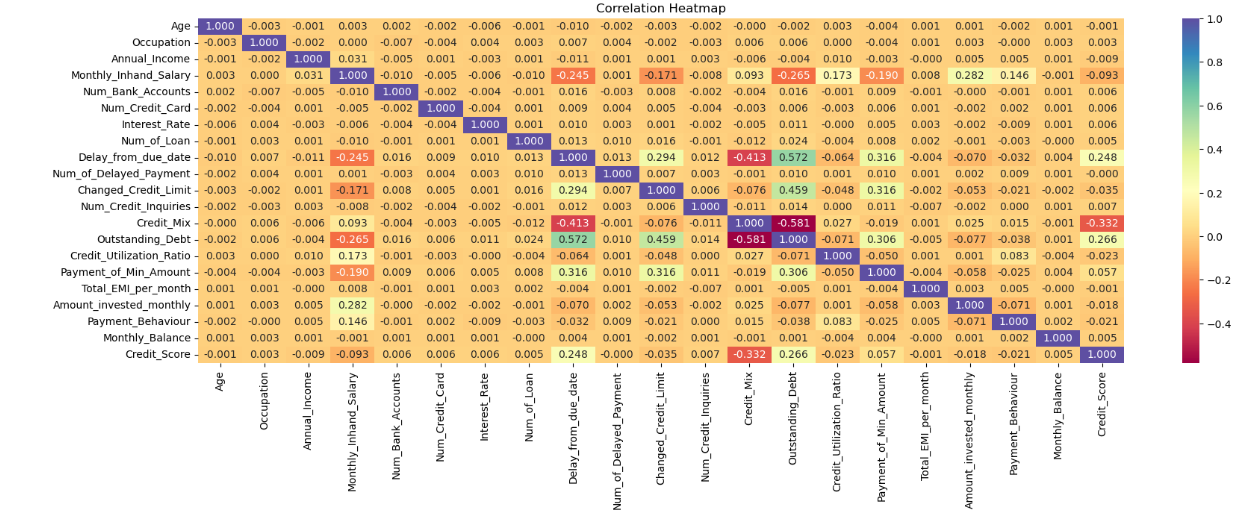
The process began with identifying and modifying columns that contained underscores in their names. These underscores were removed, and the columns were converted to numeric types to ensure uniformity and prevent processing errors, a vital step for data consistency. In the data cleaning phase, the decision to omit specific columns like 'ID', 'Month', 'SSN', and 'Name' was made to streamline the dataset by focusing only on variables with significant predictive potential. The strategy for handling missing values was meticulously crafted, involving custom functions to calculate mean and mode for data segments, thus intelligently imputing missing values based on their context. This approach significantly improved the quality of the dataset by replacing missing data with relevant statistical figures.

Further cleaning efforts were made to address special characters and transform textual data into a numerical format through label encoding. This step is essential since many machine learning algorithms require numerical input to function correctly. We also handled the outliers to ensure that statistical analyses and machine learning models are not unduly influenced by extreme values.

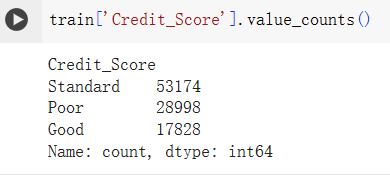
We also embraced visual EDA techniques, notably the creation of a custom-colored distribution plot for the 'Credit\_Score' variable, to gain deeper insights into the data's distribution, trends, and potential outliers. Such visual analyses are crucial for identifying factors that may influence the performance of the predictive models. The below figure shows the label distribution.



The generation of a correlation heatmap provided a visual overview of the relationships between numeric variables. This visualization helped us in uncovering potential multicollinearity or significant associations, guiding feature selection and modeling strategy.



Before proceeding to do the data cleaning it is important to address the distribution of the response variables inside the dataset which can be crucial in explaining some of the anomalies later encountered in the project. Inside the training data frame, most of the responses are for ‘standard’, although this is inline with the reality that most people will have a normal credit score, this label imbalance has nevertheless became a detriment for the model accuracies.

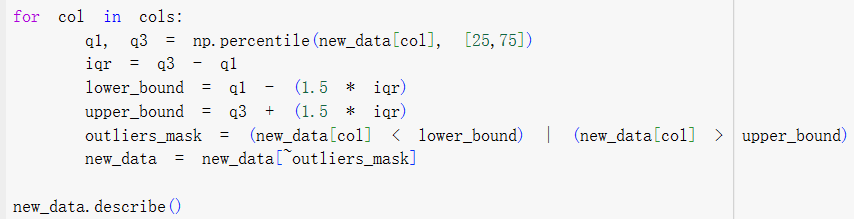


**4.3 Data Processing**

Within the dataframe, the two most major operations have been done to ensure the data integrity before the model training which are the null value handling and the outlier handling. For the null value, the team has used a primitive way of filling it with the mean value for its simplicity and hopes to not change the average of the data. For the exact implementation, a simple imputer with a mean strategy is used for the data. Looking back at the method, there are other methods that could also be attempted such as using KNN to fill the mean value in order to prevent the error from happening.

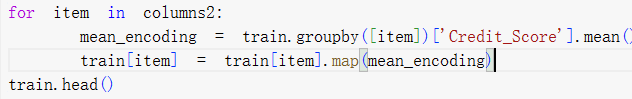


For handling the outlier of the data, a method is written to identify the quantile of the feature and if the sample falls out of the bound it will be removed.



Continuing from the initial run of data cleaning, the model training didn’t go so smooth as the cleaning. Out of the 5 models most of them performed poorly having accuracies average around 50-60%. For the remaining SVM model, run simply failed. It became clear that a re-clean of the data was necessary.

On top of the first cleaning run, the target encoding is now implemented in the second run. From calculating the mean of groups of the categorical variable and finally swapping the value, target encoding has at last increased the modeling accuracy albeit bringing new problems. The specific will be discussed in the result section.

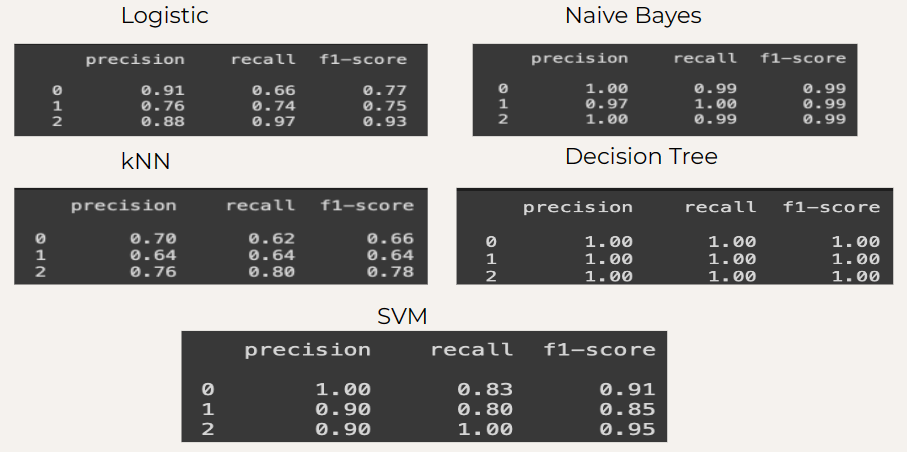


**4.4. Model Outcomes**

After implementing the target encoding in the final cleaning run, the accuracy of the models has been improved significantly; even the previously not working SVM model is now working and yields a prediction accuracy of 92 percent. However, despite the boost in accuracy for the most case, a new problem of overfitting has appeared as shown in the graphs below. Logistic regression seemed to have high overall accuracy. While an AUC of 0.97 for logistic regression does suggest that the model has excellent overall performance in terms of its ability to discriminate between positive and negative instances, an F1-score of 0.77 for class 0 indicates that while the model is highly discriminative overall, there may be some room for improvement in terms of its precision and recall for this specific class. The same goes for other models with high AUC values.

| **Model** | **Accuracy** |
| --- | --- |
| Logistic Regression | 0.851 |
| K-Nearest Neighbors | 0.7183 |
| Decision Tree | 0.9993 |
| Naive Bayes | 0.9914 |
| Support Vector Machine | 0.9286 |





After some discussion, it's clear that addressing the data cleaning process and refining the feature engineering can significantly impact the performance of the model and help mitigate overfitting issues. Since target encoding may lead to overfitting, we might consider alternative encoding methods such as one-hot encoding or label encoding. Due to its nature to directly assign the average of groups of data to encode categorical variables, features with small sample size (many nan values) might end up having monotonous value and lead to overfitting at last.

Instead of filling NaN values with the mean, using KNN imputation might be better. Due to KNN’s nature of choosing value out of samples that are close (more similar) to the subject, KNN can be better at maintaining the feature distribution and potentially giving higher accuracy without overfitting. It might also be helpful to refine our feature engineering. Although we have done some basic implementations such as dropping features with the majority of NaN values, it can be helpful to further bin features together to reduce noise and capture non-linear relationships. This can help improve model performance and interpretability.

**5. Conclusion**

For the key takeaway of the project, the first point is that training on a real life sample size scenario can be unstable at times. Despite efforts to address issues such as errors, low accuracy, and overfitting, the process can still be prone to instability and prolonged execution times. The grid search decision tree by itself can take up hours to run not to mention this is just the vanilla version without any boosting. Besides the instability, another key takeaway from the project is that knowing only the top down theory is not enough. Moreover, in the context of credit score prediction, addressing label imbalance is crucial to ensure the model's effectiveness and reliability in assessing creditworthiness. Techniques such as oversampling of minority class instances, cost-sensitive learning, and ensemble methods can help mitigate the impact of label imbalance and prevent the model from being biased towards the majority class. While the ROC curve and AUC are valuable metrics for evaluating classification models, they may not adequately capture the performance when dealing with imbalanced datasets. In such cases, precision and recall become more important metrics to consider.

The future scope for our implementation is to re-clean the dataset, without the target encoding method and encompassing other improvements like one-hot encoding or label encoding and utilizing KNN imputation for NaN values. We will also address the label imbalance to avoid bias towards the majority class. Using these techniques should yield results that are of adequate accuracy without overfitting. After training completion, a horizontal comparison will be done again between the models to decide which is the most optimum and coefficients of different features would be computed to decide which are the most important for credit scores prediction.

**References**

[1] Kuppili, V., Tripathi, D., & Reddy Edla, D. (2020). Credit score classification using spiking extreme learning machine. Computational Intelligence, 36(2), 402–426. https://doi-org.proxy.cc.uic.edu/10.1111/coin.12242