Credit Card Approvals Report - Support Vector Machines

Dinh Minh NGUYEN , Luong Phuong Truc HUYNH, Quang Huy PHUNG

Université Jean Monnet

Professors: Sri Kalidindi & Thibaud Leteno & Richard Serrano

Summary—*This analysis showcases various analytical techniques applied to evaluate a company's credit card application approval process. The final support vector machines model, trained on our processed data using cross-validation, achieved an accuracy rate of 89% in predicting application outcomes. This performance surpassed the baseline model by a margin of 5%. Fairness metrics are conducted on the trained model, revealing that imbalances in distributions of qualified and unqualified instances of a certain ethnic group in the dataset give biased but justifiable predictions.*

***Keywords*—**Support Vector Machines, Model Fairness, Data Visualization, Linear Regression, Decision Tree, Cross-validation

	Part 1	**Part 2**
Dinh Minh Nguyen	33%	33%
Luong Phuong Truc Huynh	33%	33%
Quang Huy Phung	33%	33%
ChatGPT	1%	1%
Total	**100%**	**100%**

Table 1. Contributions

1. Dataset Summary

The Credit Card Approval dataset provides information on credit card applicants, including demographic and financial attributes, alongside the approval status of their credit card applications. The first step in any analysis is to obtain the dataset. We use the original data that is unclean.

Preproccessing Data

First review of the original data, we noticed that it has many meaningless symbols that are masked for the confidentiality of the data and missing values denoted by '?'. To make it easier to work with, we gave them variable working names based on the type of their contained data.

For the missing values, we first identify which ones are missing and then determine how to address them. Some possible methods are removing them, zeroing them out, or estimating a plug value. We scan through the dataset that missing values are labeled with '?'. For each of them, we convert the missing values to nan which Python will interpret differently than a character value.

Continuous Variables

Continuous Variables include `Age`, `Debt`, `YearsEmployed`, `CreditScore`, and `Income`. As we can see from Table 2, `Age` has 12 missing values that we have to fill in. We could simply use the mean of all existing values to do so. A more accurate method would be checking the relationship among the numeric values and using a linear regression model to fill them in.

Feature	NumberMissing
Gender	12
Age	12
Married	6
BankCustomer	6
Industry	9
Ethnicity	9
ZipCode	13

Table 2. Missing Values in Dataset

Table 3 shows the correlation between `Age` and continuous variables. The largest value is 0.396 meaning `Age` is most closely correlated with `YearsEmployed`.

	Debt	YearsEmployed	CreditScore	Income
Age	0.202	**0.396**	0.186	0.019

Table 3. Correlation between `Age` and other Continuous Variables

Linear Regression. We can build a linear regression model based on this information. The model has two coefficients: `Intercept` and `YearsEmployed`. `Age` missing values are calculated by multiplying the `YearsEmployed` coefficient with values of `YearsEmployed` and adding the intercept.

$$\begin{pmatrix} \text{Intercept} \\ \text{YearsEmployed} \end{pmatrix} = \begin{pmatrix} 28.446953 \\ 1.412399 \end{pmatrix}$$

For example, the `YearsEmployed` value is 3. The formula is then $3 \times 1.412399 + 28.446953 = 32.6841489$. All 12 missing values in `Age` will be filled in this way.

Descriptive Statistics. The next step of working with continuous variables is to standardize or calculate the z-score. We will calculate the z-score by subtracting the mean from each value and dividing it by the standard deviation. When we plot the histograms, the distribution looks the same but the z-scores are easier to work with because the values are measured in standard deviations instead of raw values. Another thing to note is that the data is skewed to the right (Figure 1).

Next, we compare the credit status by the value of AgeNorm. We use a boxplot to show the mean value for each group and the quartiles. In "Boxplot of AgeNorm by Credit Status" (Figure 2), the median of the two groups is slightly different with the age of approved applications being slightly closer to the mean than the denied applications. We also see that the interquartile range is greater on the 'Approved' than the others. We interpret these facts as the younger credit applicants are less likely to be granted credit. However, several outlying applicants with high values were still not granted credit.

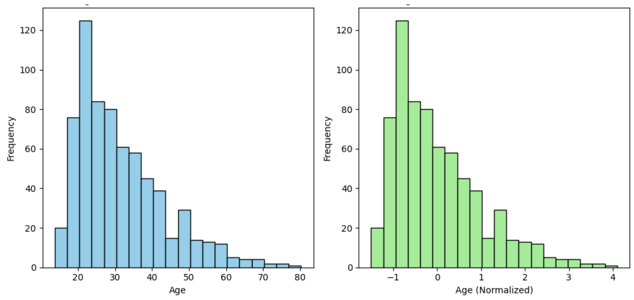

Figure 1. Age Distribution before and after normalization

We do similar with the other continuous variables. From boxplots of `DebtNorm`, `YearsEmployedNorm`, `CreditScoreNorm`, and `IncomeNorm` in Figure 2, we can see the distribution is different between the variables. Income has the least amount of variance because the boxes are tightly grouped about the mean. As we mentioned above, the data is skewed to the right so the median is less than the mean. Therefore logarithmic transformation could be a good candidate for this dataset.

Figure 3 shows the continuous variables after first taking the log of each value and then normalizing it as similar to the above. The boxplots seem to get more informational value now because for each dataset the mean of the approved applications is further distributed from the mean of those denied. This difference will help the classifier algorithm to distinguish between the values later. We should specifically notice for the `IncomeLog` and `CreditScoreLog` variables that the applicants who did not receive credit were still heavily skewed to the right when compared to those who were granted credit. This means that a low `IncomeLog` or `CreditScoreLog` score is likely a good predictor for making the application decision.

Categorical Variables

For 13 missing values of `ZipCode`, we just fill in the most common value. While for `Gender`, `Married`, `BankCustomer`, `Industry`, and `Ethnicity`, we use a more complex and accurate method which is the decision tree to fill in them.

Decision Tree. The decision tree assigns the most frequent value of the target categorical variable among the instances that reached that leaf node. This value is used as the predicted value for instances with missing values.

For example, in row 248, the decision tree examines the values of other columns like `Married` = u, `BankCustomer` = g, and `EducationLevel` = c. It traverses the decision tree, evaluating conditions at each node to predict the most likely value for `Gender`. It learns rules such as $\{u, g, c\} \rightarrow \{1\}$, indicating that this combination is associated with the value 1 for `Gender`.

2. Support Vector Machines Theory

Support vector machines (SVM) seek decision rules by exploiting linearity in a given dataset. For a training set $T = \{z_i = (\bar{x}_i, y_i)\}_{i=1}^{n}$, where $\bar{x}_i$ is a vector in d-dimension and $y_i \in \{-1,\ 1\}$, an SVM classifier attempts to derive a hypothesis $h \in \mathcal{H} : \mathbb{R}^d \rightarrow \{-1,\ 1\}$ of the form:

$$h(\bar{x}) = \operatorname{sign}(\langle \bar{w},\ \bar{x} \rangle + b)$$

Thus, $h(\bar{x})$ is a hyperplane in $\mathbb{R}^d$, predicting label $\hat{y} = h(\bar{x})$ for any input vector $\bar{x}$ in that space. This hyperplane is characterized by a weight vector $\bar{w}$ and a bias term b.

If the direction of $\bar{w}$ points to the positive half plane where instances $\bar{x}_i$ with $y_i = 1$ reside, then the optimal value of $\bar{w}$ and b are ones that maximize the margin γ:

$$\gamma = \min_i \ y_i(\langle \bar{w},\ \bar{x}_i \rangle + b)$$

This is an optimization problem and solving it equivalence to solving the following linear program

$$\begin{aligned} &\min \ f(\bar{w}, b, \xi) = \frac{1}{2}||w||^2 + C\sum_i \xi_i \\ &\text{subject to } \begin{cases} y_i(\langle \bar{w},\ \bar{x}_i \rangle + b) \geq 1 - \xi_i \\ \xi_1,\ \xi_2,\ \xi_3,\ \dots,\ \xi_n \geq 0 \end{cases} \end{aligned} \tag{1}$$

Ideally, the value $y_i(\langle \bar{w},\ \bar{x}_i \rangle + b)$ should be ≥ 1 for all x_i in T. However, instances of z_i in T usually clump together, so the program allows some samples to be at a distance ξ_i from their correct margin boundary.

The penalty term C controls the strength of this penalty, and as a result, acts as an inverse regularization parameter.

For a training of h following the Perceptron model [1], when converged, $\bar{w}$ will look like a linear combination of all instances $\bar{x}_i$ in T:

$$\bar{w} = \sum_i y_i \alpha_i \bar{x}_i$$

Here, α_i counts the number of times $\bar{x}_i$ triggers the update of $\bar{w}$. With this representation of $\bar{w}$, h can now be rewritten into its dual form:

$$h(\bar{x}) = \sum_i y_i \alpha_i \langle \bar{x}_i,\ \bar{x} \rangle + b$$

Likewise, the linear program 1 can also be transformed into its dual form where the problem reduces to optimizing coefficients α_i that weights the importance of each vector x_i.

$$\begin{aligned} &\max \ g(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j y_i y_j \alpha_i \alpha_j \langle \bar{x}_i,\ \bar{x}_j \rangle \\ &\text{subject to } \begin{cases} \sum_{i=1}^{n} y_i \alpha_i = 0 \\ 0 \leq \alpha_i \leq C,\ 1 \leq i \leq n \end{cases} \end{aligned} \tag{2}$$

When converged, instances with α_i greater than 0 are qualified as the support vectors of the hyperplane.

The dual forms of h allow the training process to depend only on the dot product between any pair $\bar{x}_i,\ \bar{x}_j$ in T. This opens up a possibility for h to efficiently classify non-linear distributions of z_i in T.

If a kernel function $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is defined so that for a non-linear mapping $\theta : \mathbb{R}^d \rightarrow \mathcal{F}$,

$$\mathcal{K}(\bar{x},\ \bar{x}') = \langle \theta(\bar{x}),\ \theta(\bar{x}') \rangle$$

Then, by replacing all occurrences of $\langle \bar{x}_i,\ \bar{x}_j \rangle$ with $\mathcal{K}(\bar{x}_i,\ \bar{x}_j)$ in the linear settings of h, the classifier will look at all instances $\bar{x}$ in T as if they were mapped to some non-linear space $\mathcal{F}$.

$\mathcal{K}$ could be defined to be a linear or polynomial function of $\langle \bar{x},\ \bar{x}' \rangle$, but the well-known RBF kernel function is favorable in practice:

$$\mathcal{K}_{\text{RBF}}(\bar{x},\ \bar{x}') = e^{-\gamma||\bar{x}-\bar{x}'||^2}$$

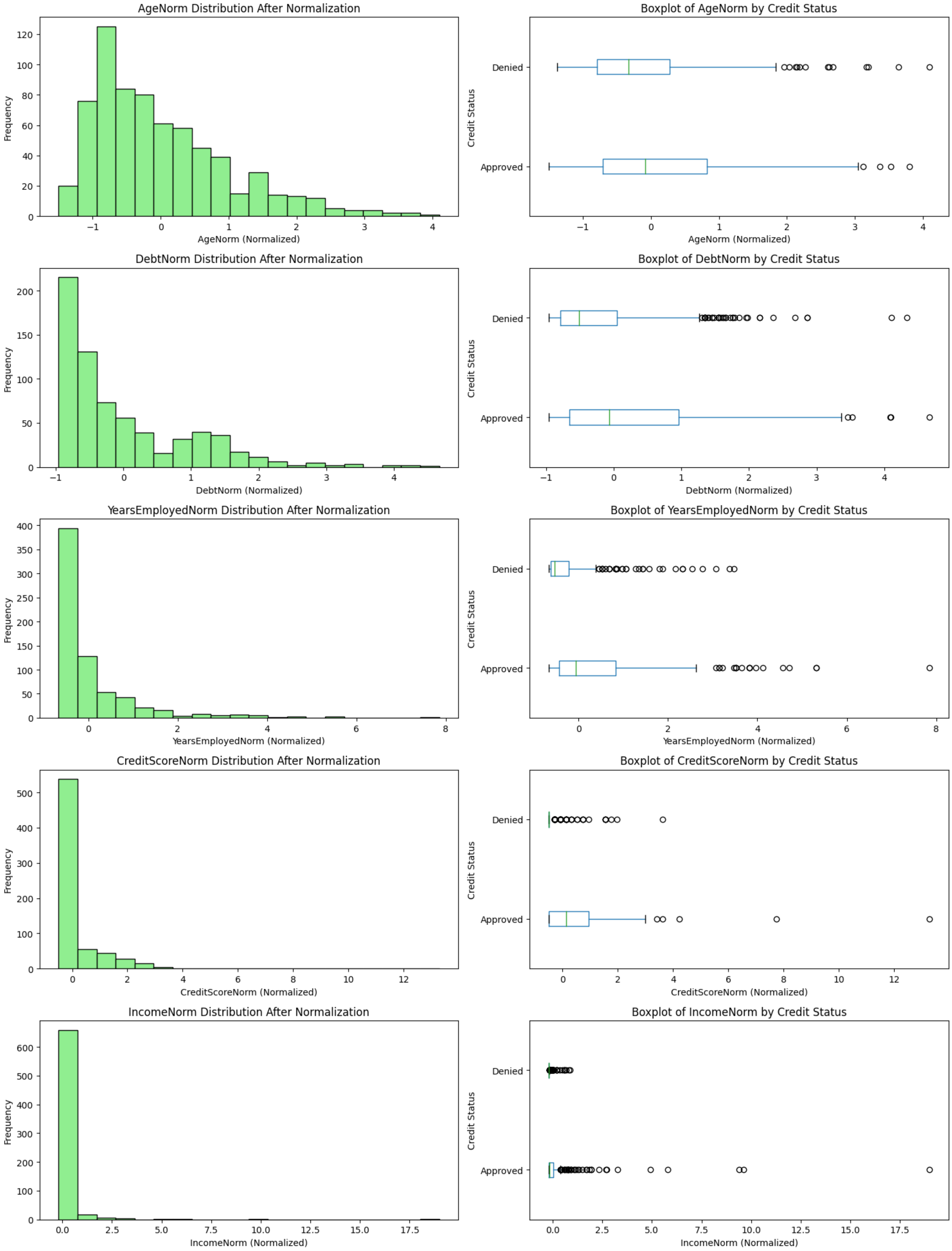

Figure 2. Distribution and boxplot of Normalized Continuous Variables by Credit Status

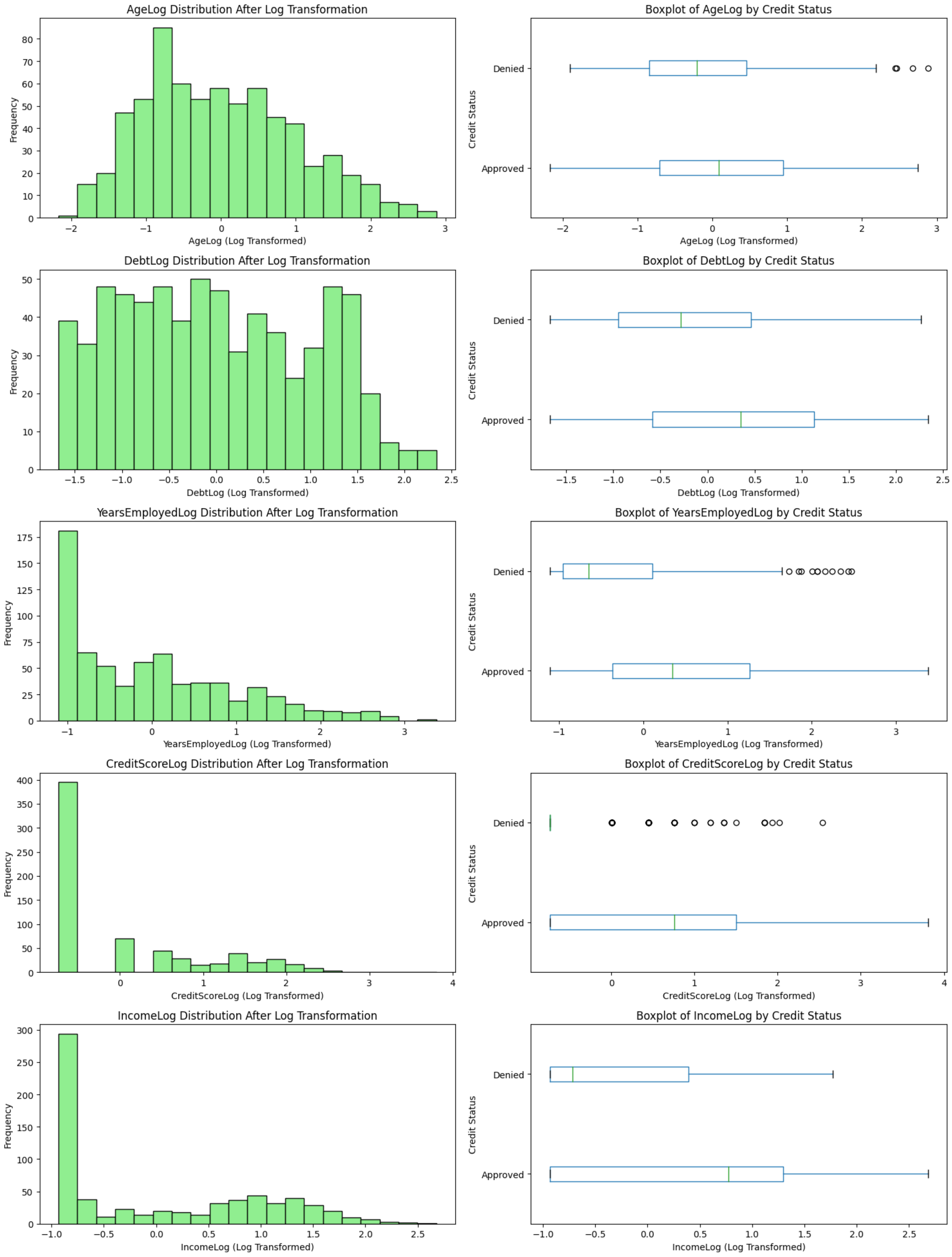

Figure 3. Distribution and boxplot of Logarithmic Continuous Variables by Credit Status

The γ term in $\mathcal{K}_{\text{RBF}}$ controls how fast the exponential term decays. As a result, a smaller value of γ relaxes the margin constraint, while a bigger γ introduces intricate boundaries.

3. Hyperparameter Tuning

In this work, we favor the non-linear SVM classifier since the number of training examples is dominant compared to that of features [2]. We explicitly use the RBF kernel to prevent overfitting. There are two parameters for an RBF kernel: C and γ, as discussed in the previous section. It is not known beforehand which C and γ are best for a given problem; consequently, some kind of model selection, or hyperparameter tuning, must be done.

To perform hyperparameter tuning, we leverage the `GridSearchCV` class which conducts an extensive search based on all possible combinations of parameters. What's nice about this class is that internally, the library performs k-fold validation to select which set of hyperparameters gives the best validation accuracy. Therefore, cross-validation is already handled upfront.

However, `GridSearchCV` incurs a heavy tradeoff. Too many combinations of C and γ would result in expensive computations, while too few of them might not cover the possible optimal range. Following the practice suggested in [2], we start with a coarse grid search of relatively few combinations of C and γ covering a large range. The suggested starting ranges are $[2^{-5}, 2^{15}]$ and $[2^{-15}, 2^{3}]$ for C and γ, respectively. Once the optimal set of values in this coarse search is found, we define a new search range based on where those values are, and with finer steps. By the time we get to the third search, the optimal value for C and γ are 66.0948 and 3.05×10^{-5}, respectively.

4. Model Evaluation

After conducting experiments on the dataset, we discovered that a train-to-test ratio of 75:25 yielded the highest accuracy. Therefore, we adopted this ratio as the default for our final results. For the baseline, we trained an SVM with the available cleaned version, which achieved 84% accuracy. However, after preprocessing the data, including applying logarithmic transformation to address skewed data and using decision tree to fill missing data in categorical variables, our processed data trained with SVM achieved an accuracy of 89%, marking a notable improvement of 5%. Additionally, the best hyperparameters found for the SVM model were C = 330.35, gamma = 0.00406, kernel = 'rbf'. The model achieved a precision of 0.875 for class 0 and 0.909 for class 1, with recall values of 0.923 and 0.854, respectively. The corresponding F1-scores were 0.898 for class 0 and 0.881 for class 1.

5. Fairness

The practice of fairness in machine learning ensures that decisions made by statistical hypotheses are unbiased and equitable across different demographic groups. In this credit card approval problem, there's a high chance the trained classifier may favor one ethnic group over another. Therefore, this section demonstrates the analysis process to detect such unfairness before and after training the model.

5.1. Statistical Analysis

As we suspect groups in the `Ethnicity` feature may suffer from biased decisions, Figure 4 counts the number of occurrences of each ethnic group in the training dataset. With such a skewed distribution, there is a high chance this dataset would generate biased estimators for the approval of credit cards.

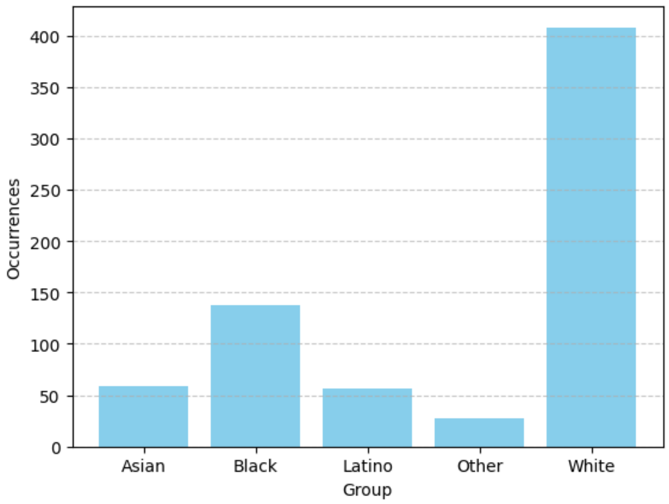

Figure 4. Unbalanced groups in `Ethnicity` feature

A more quantitative approach to test this assumption is to ask if there is a statistically significant difference in how credit is granted between ethnicities that could indicate discrimination. Such a question can be answered by carrying out a χ-squared test between random variables E and C, where E is the ethnic groups and C denotes the approval decisions.

We define the null and alternative hypotheses for this test as follows

- H_0: Ethnicity and approval are independent.
- H_1: Approval status is dependent on the ethnicity and the bank has a compliance risk.

Table 4 summarizes this statistic test. It is obvious that we have enough evidence to reject H_0, meaning approval status is dependent on the ethnicity.

χ^2 statistics	p-value
36.9945	1.8×10^{-7}

Table 4. χ^2 test of independence between `Ethnicity` and approval status

From these statistical results, we proceed to measure fairness metrics on the trained model to verify if the unbalanced dataset affects its approval decision.

5.2. Fairness Metrics

Several fairness metrics can quantify if a classifier is biased in its decision. Most metrics rely on the *confusion matrix* obtained from the model's decision on each group, as shown in Table 5.

Layout	
TP	FP
FN	TN

Asian	
6	2
1	6

Black	
19	3
1	11

Latino	
2	1
1	10

Other	
4	0
1	1

White	
43	13
4	44

Table 5. Confusion matrices for each ethnic group

Approval rate The approval rate is the probability for which the model thinks a certain ethnic group should get approved based on their profile. This metric examines if the model favors any group over another.

$$\text{Approval rate} = \frac{TP + FP}{TP + FP + TN + FN}$$

.

Equal Opportunity This metric concerns the true positive rate (TPR) of the model in predicting if a certain ethnic group should get their profile approved. TPR measures how sensitive the model behaves when exposed to a qualified ethnic group.

$$\text{TPR} = \frac{TP}{TP + FN}$$

.

Demographic Parity The demographic parity quantifies the model's accuracy in approving and rejecting a certain ethnic group based on their profile.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

.

Metric	Asian	Black	Latino	Other	White
Appr. rate	0.53	0.65	0.21	0.67	0.54
TPR	0.86	0.95	0.67	0.80	0.91
Accuracy	0.80	0.88	0.86	0.83	0.84

Table 6. Fairness metrics for different ethnic groups

Table 6 computes all the discussed metrics based on the model's prediction. The metrics at first suggest that the `Latino` group gets unfair decisions, with only 21% of the time this group gets approved for their credit card application. The TPR also supports this judgment since the model is less sensitive to qualified Latino applicants. It seems that the model treats Latino profiles more strictly than others.

However, the accuracy metric claims that the model's accuracy for the Latino group is comparable to that for the others. The only explanation for this phenomenon is that the dataset contains less qualified Latino applicants. In other words, the model learned how to reject an unqualified Latino applicant more than it learned how to accept a qualified one.

This reveals an important insight. It is the distribution of qualified and unqualified instances in each ethnic group in the dataset that matters, not the number of them in each group. While we could drop unqualified Latino profiles in the dataset to invoke balance, it is not harmful to let the model increase its specificity when exposed to unqualified Latino applicants.

6. Conclusion

We have trained an SVM classifier to predict if a person applying for a credit card is trustworthy. The process went through several stages, arriving at the final 85% test accuracy. The training revealed which features contribute more to the prediction of credit card approval, and which features are of no interest. The study of fairness left one important insight: the distributions of qualified and unqualified profiles of a certain ethnic group in the dataset matter more than the number of occurrences of instances in each group. While provoking this balance is possible, keeping the trained specificity is preferable.

References

[1] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms* (Cornell Aeronautical Laboratory. Report no. VG-1196-G-8). Spartan Books, 1962. [Online]. Available: https://books.google.ca/books?id=7FhRAAAAMAAJ.

[2] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines", *ACM Transactions on Intelligent Systems and Technology*, vol. 2, 27:1–27:27, 3 2011, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.