Credit Score Machine Learning Project Report

ECS 171 - Winter 2024

Maia Burton, Harjot Gill, Yuan En Jolyn Loh, Nhan Nguyen

January 8, 2024 - March 14, 2024

Contents

1 Introduction **3**

2 Literature Review **3**

3 Dataset Description **4**

3.1 Inflation Consideration 4

3.2 Data Size and Practicality 4

4 Proposed Solution and Experimental Results **5**

4.1 Data Pre-processing Steps 5

4.2 Correlation Matrix 6

4.3 Model Choice: Logistic Regression vs. Neural Network 7

4.3.1 Logistic Model and Neural Network Model Assessment 7

4.3.2 Low Recall Rate 8

4.4 Working with the Neural Network 9

4.4.1 Hyperparameter Tuning 10

4.4.2 Early Stopping 10

4.4.3 Under and Over Sampling 11

4.5 Final Results 11

5 Conclusion and Discussion **12**

6 Road-map and Action Plan (Estimated Dates) **13**

7 Works Cited **14**

1 Introduction

Among the various challenges faced by financial institutions, assessing creditworthiness remains predominant. Credit scoring, a method used to evaluate the likelihood of a borrower repaying their debts, had become an essential tool for financial institutions. Traditionally, credit scoring has been done through manual means based on historical data and personal judgement. However, our approach with supporting or completely replacing machine learning (ML) techniques may uncover patterns that are not immediately apparent to human analysis.

The application of machine learning in credit score analysis offers a promising future for enhancing decision-making processes, improving prediction accuracy, and reducing potential losses from bad debts. These ML models can consider a wide range of variables, including borrowers' annual income, the number of bank accounts, and number of credit cards, and even behavioral patterns such as monthly balance.

This project aims to explore the effectiveness of different machine learning techniques in the context of credit scoring applications. Additionally, the project will address the challenges associated with implementing ML models, including data preprocessing, feature selection, model training, and evaluation. Through this analysis, we intend to provide insights into how machine learning can revolutionize credit scoring processes, thereby enabling financial institutions to make more informed lending decisions and ultimately contribute to a more stable and efficient financial system.

2 Literature Review

We looked into 2 papers that looked into credit scoring with machine learning. The first, by Shrawan Kumar Trivedi, looked at different methods of machine learning to determine credit scoring. Looking into the paper revealed they used several methods such as the "SVM", "Random Forest" and more; however they did not use the Multi-Level Perceptron and used the dataset German Credit Data from University of California, Irvine [2]. The paper influenced our decision to move towards the Multi-Level Perceptron and to try another dataset to avoid following the paper and working more exploratory.

The second paper we looked at was an article regarding neural network performance versus other methods of machine learning by Xolani Dastil, Turgay Celik, Moshe Potsane. They found that neural networks performed better than methods such as the logistic regression and that deep neural networks were not investigated as much as other methods, such as the mentioned logistic regression and support vector machines when researching papers [1].

The research into the papers impacted our decisions in regards to our dataset decision and our model choices. We looked into the German Credit Data dataset from UC Irvine, however we felt that the format of the dataset was difficult to read. Each row and column was described by some value, for example "A24", and would be corresponded to a table which contained what it meant. We looked into different versions of the dataset made by community members but we decided

to pivot away from it as we wanted to work with our dataset as originally as possible. We then found a dataset from Kaggle that provided similar features and was easier to read and work with.

Our model choice was between the Logistic Regression and the Multi-Layer Perceptron. We ultimately decided on these two as the Logistic Regression felt like it could work as a baseline and not require much time to implement. We wanted to work with the Multi-Layer Perceptron as we wanted to investigate how well it could perform considering the lack of investigations into its application for credit scoring from the second paper. We also just wanted to work with neural networks as it seemed to be one of the buzzwords terms associated with machine learning and we wanted to work with it to see what is actually involved with a neural network.

3 Dataset Description

The dataset describes people's information including but not limited to annual salary, age, credit history, number of credit cards, debt, and their credit score classification. For both the model's conceptualization and execution phases, we discussed several considerations about the dataset we would use. The following discussion outlines these decisions made by our research group.

The **dataset** we chose to use is from Paris Rohan on Kaggle.
Link Here: https://www.kaggle.com/datasets/parisrohan/credit-score-classification

3.1 Inflation Consideration

Our group deliberated on how to account for inflation given if the dataset was relatively old or had a vast range of time. The age of our dataset would determine whether or not we should incorporate inflation, adding an additional step to our data processing, converting all numerical values to be equivalent. Understanding the time span in which inflation became relevant was crucial. Within five years, inflation may not be a primary concern.

The timestamp for the dataset we decided to work with stated that it was uploaded 2 years ago. Because of this, we did not account for inflation.

3.2 Data Size and Practicality

The amount of data required for a learning model, especially for credit scoring, varies depending on complexity, diversity, and the number of features. As a general goal, we wanted tens of thousands of examples for our goal—we deemed it a moderately complex problem, encompassing income, employment history, past loan performance, etc. Simpler models may require fewer data points, while complex models might need significantly more.

While having a large dataset is generally beneficial, extremely large datasets could lead to higher computational costs and longer training times without corresponding improvements in

the model's performance. The quality of the data is also important, and if additional data was redundant or irrelevant, it would require more work to weed it out.

The dataset we worked with had over 47,000 records and accounts for 29 independent variables. We felt this was a substantial volume for the model to learn from, reducing the likelihood of overfitting and improving its predictability accuracy. Additionally, the variety of variables contribute to a comprehensive learning environment ensuring varied scenarios and data patterns.

4 Proposed Solution and Experimental Results

4.1 Data Pre-processing Steps

Formatting
When we looked into the actual data, we noticed there were lots of errors, missing values, and garbled text. We first handled the erroneous characters by removing them, then removed all rows that contained NaN values. In addition to the irregular data values, some independent variables such as credit history length were listed as strings rather than numerical values. Those had to be converted into the proper variable type.

We also dropped a handful of columns we felt were unnecessary for giving a credit score such as ID and SSN.

Normalization and Handling Outliers
We applied Min-Max Scaling to normalize the feature values to a range of [0,1]. This step ensured that all input features had the same scale, allowing for the neural network to train more efficiently and improves convergence. We chose to use this type of scaling because of its ability to preserve the original distribution of values while bringing them onto a common scale.

While Min-Max Scaling does not handle outliers well and can skew the scale, we handled this before applying the method—using the 25th quartile and the 75th quartile to remove outliers with a standard multiplier of 1.5.

Encoding
Our model incorporates categorical variables, such as loan types and payment behavior. We opted for one-hot encoding to transform these categorical variables into a format that can be provided to the machine learning algorithms learned in class. This method converted each category value into a new binary column, which is necessary for the neural network to process the categorical data effectively. The choice of One-Hot Encoding was made to eliminate accidental ranking or order among categories which could happen with numerical encoding. Each class is treated as unique and independent.

We did recognize that we were working with a large number of independent variables which could make working with one-hot encoding difficult. Despite increasing the number of columns and potentially leading to higher-dimensional space, we determined the trade-off to be worthwhile. This decision was based on the enhanced model performance and interpretability provided

by one-hot encoding, ensuring distinct categories of loan types and payment behaviors are accurately represented.

4.2 Correlation Matrix

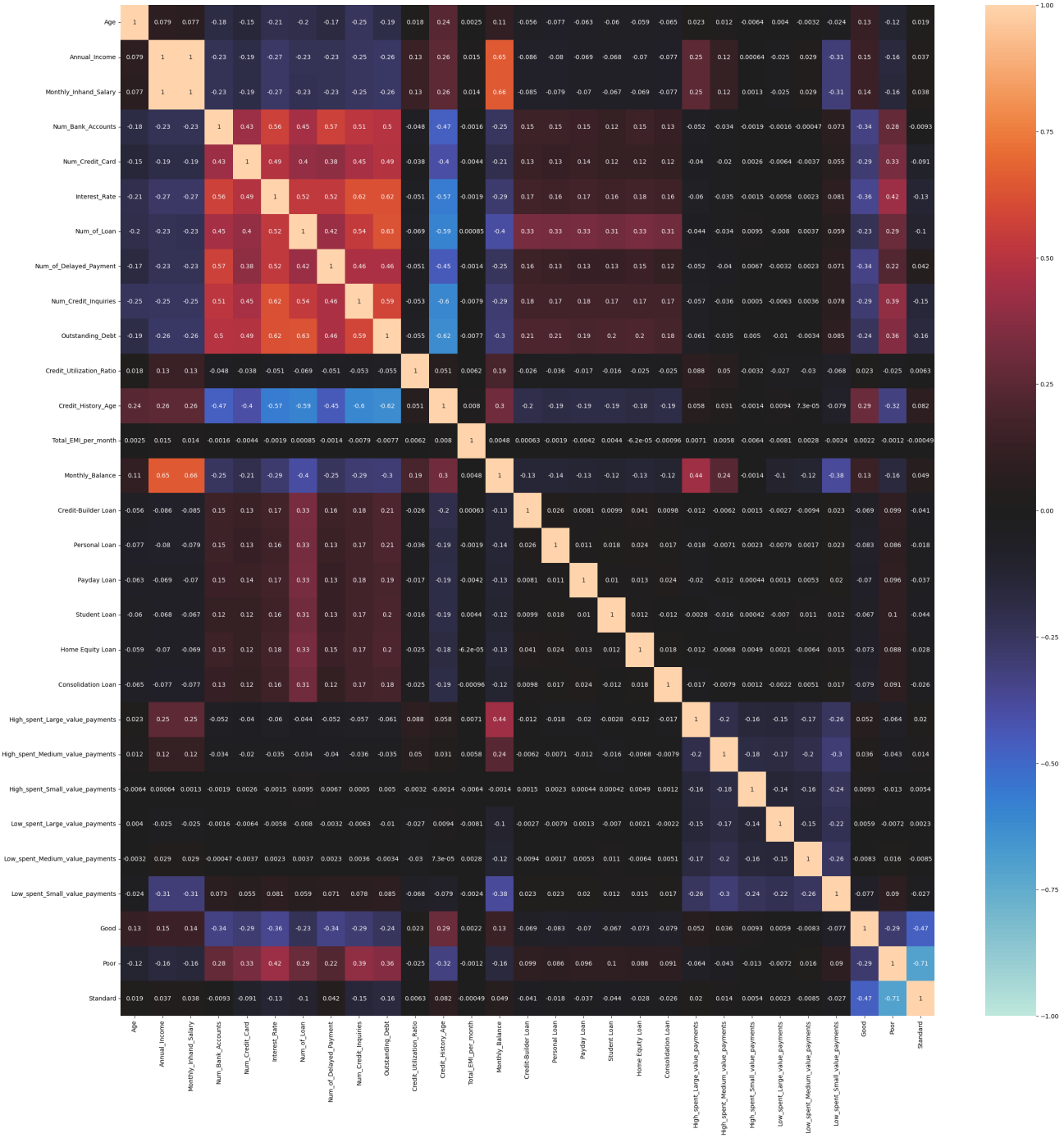


Fig. 1. Correlation Matrix

We investigated the correlation matrix of the dataset after preprocessing and could not make out strong correlations in regards to scoring between Good, Standard and Poor. The only areas we noticed that had a direct effect were the Age, Salaries, Number of Credit Cards/Bank Accounts/Loans/Credit Inquires, Interest Rate, Debt, and Credit History Age. We could have solely focused on these attributes but we chose not to in order to take deeper consideration of factors such as how a person pays their balance and types of loans, which are used in real life in consideration to one's credit score.

4.3 Model Choice: Logistic Regression vs. Neural Network

Initially, our group explored both multi-variate logistic regression and neural network models to asses their efficacy in predicting credit worthiness. Logistic regression, being a simpler and more interpretable model, offered a straightforward approach. In contrast, neural networks provided a more complex and potentially more accurate alternative. This means we used most of the columns present after preprocessing.

FOR ALL MODELS AND TABLES ASSUME: Train/Test Split 20%

Logistic Regression Model				
	Precision	**Recall**	**F1-Score**	**Support**
Good	0.52	0.16	0.24	1424
Poor	0.62	0.50	0.56	3052
Standard	0.59	0.78	0.68	4913
Accuracy	0.60		0.60	9389
Macro Avg.	0.58	0.48	0.49	9389
Weighted Avg.	0.59	0.60	0.57	9389

After rigorous testing and evaluation, the MLP Classifier was selected for its flexibility and capacity to model complex non-linear relationships inherent in the data in addition to its superior performance in terms of accuracy, interpretability, and computational efficiency.
In addition, we were interested in seeing how far we could push the neural network for this application as from the literature there were only a handful of papers that used a neural network.

4.3.1 Logistic Model and Neural Network Model Assessment

Although the logistic regression model and the baseline neural network had similar accuracies, we opted to work with the neural network for several reasons:

- Complexity and Flexibility
- Data Handling

Neural Network Baseline Performance Split Train/Test 20% 1 Hidden Layer - 100 nodes				
	Precision	**Recall**	**F1-Score**	**Support**
Good	0.60	0.29	0.39	1387
Poor	0.72	0.70	0.71	4998
Standard	0.70	0.64	0.67	3004
Accuracy	0.61			
Macro Avg.	0.68	0.54	0.59	9389
Weighted Avg.	0.70	0.62	0.65	9389
solver="sgd", activation="logistic", learning_rate_init="0.1", batch_size=64, hidden_layer_sizes=(100), max_iter=1000, random_state=21				

- Model Adjustments

Logistic models are linear and improvements are mainly based on changing the data—better feature selection, data quality, and regularization. Neural networks on the other hand involve learning rates, layers, and offer advanced techniques such as hyperparameter tuning. They can also be improved to work with the data and handle more complex data without extensive engineering rather than making an effort to have the data to work for a logistic model. Additionally, logistic regression is more sensitive to outliers and non-linear relationships. When working with 29 different independent variables, we want a neural network to be able to identify patterns the human eye may not be able to notice.

4.3.2 Low Recall Rate

When establishing our performance baselines for the neural network and logistic regression, we noticed a rather low recall rate. This remained a persistent issue throughout our various NN models, leading us to suspect that the cause is due to having an unbalanced dataset, where there is a low number of "good" credit risk records.

Upon further inspection, the number of good occurrences makes up about 25%. The low recall rate imply the model's inability to learn sufficient information about the positive class.

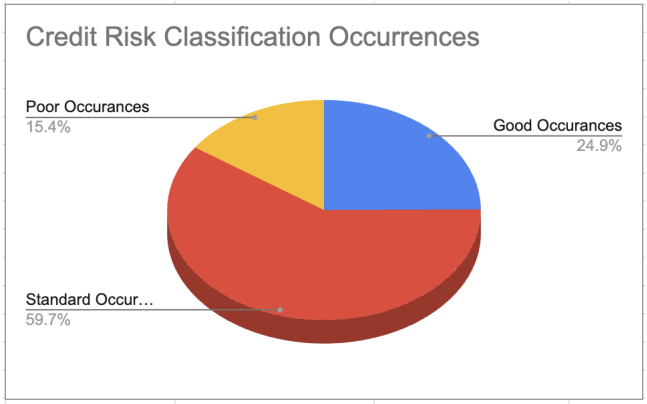
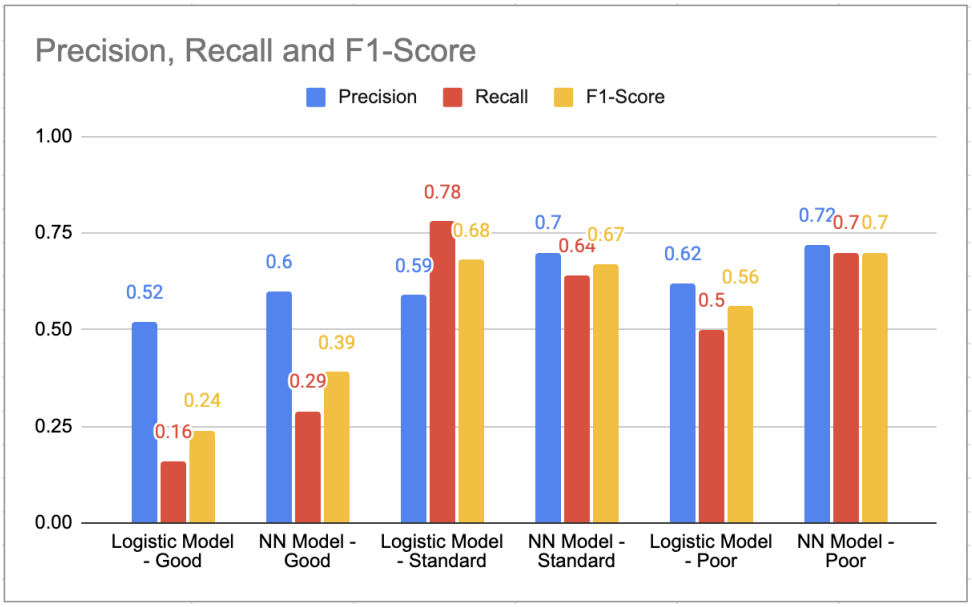


Fig. 2. Classification Imbalance

The F1-score was a useful evaluation measurement that accounted for the mean of precision and recall. The F1-score indicated a balance between precision and recall with values between 0 and 1 where a number closer to 1 indicated a better balance between the two. When classifying both the good and poor credit risk records, the neural network performed better.

Thinking about the class unbalance, we considered altering class weights, assigning higher weights to the positive class, and optimizing thresholds, the probability necessary for an instance to be classified as positive (though the latter may cause false positives). Ultimately, we decided on tuning the parameters of the model as much as possible as we felt it was more pressing to improve its performance as much as possible.



4.4 Working with the Neural Network

Configuration Choices

- **Solver:** Stochastic Gradient Descent (SGD) was chosen for its efficiency and effectiveness in handling large data sets and its ability to converge to the global minimum for convex error surfaces and a local minimum for non-convex error surfaces.
- **Activation Function:** The Rectified Linear Unit (ReLU) activation function was selected for its simplicity and capability to accelerate the convergence of SGD, thanks to its lack of upper bound—non-saturating form.
- **Regularization:** A modest regularization term prevents overfitting by penalizing larger weights. We set this parameter to be 0.00035, giving it a weak effect due to the complex dataset we were working with and allow the model to learn the underlying patterns without being overly penalized.

We allocated a 90:10 training and testing ratio, ensuring a substantial dataset for training while still retaining a significant portion for testing and validation purposes. When comparing a neural network to a logistic regression, we used 100 notes and one hidden layer. As we progressed, an iterative approach was adopted to refine the model's architecture. This involved increasing both the number of nodes per layer and the number of layers themselves to improve model accuracy. Ultimately, this process led us to settle on a structure comprising ten hidden layers, which provided a balance between complexity and computational feasibility—[200, 180, 160, 140, 120, 100, 80, 60, 40, 20].

Further adjustments were made to the model's hyper-parameters, including L2 regularization to prevent overfitting, learning rate to optimize the speed and quality of convergence, and batch sizes to manage the memory usage and efficiency of the training process. Initially, these changes resulted in our model achieving an average accuracy range between 73% to 76% without any hyperparameter tuning.

4.4.1 Hyperparameter Tuning

During our exploration of hyperparameter tuning with our neural network model, we encountered several challenges and key findings. The process was extensive, taking approximately 13 hours to complete. Despite this substantial time investment, we navigated through various configurations, experimenting with different layers and max iterations set at 300, 500, 700, and 900. Our approach included both grid search and random search techniques to systematically explore the hyperparameter space. However, these efforts did not result in the desired outcome.

We observed no significant improvement in model accuracy as a result of these adjustments. Moving forward, it's clear that additional tuning would take up more time and resources, suggesting we should attempt a different approach or alternative modeling techniques.

4.4.2 Early Stopping

To further enhance our model's performance and ensure it did not memorize the training data, we incorporated early stopping. Despite these efforts, the ceiling for accuracy appeared to be

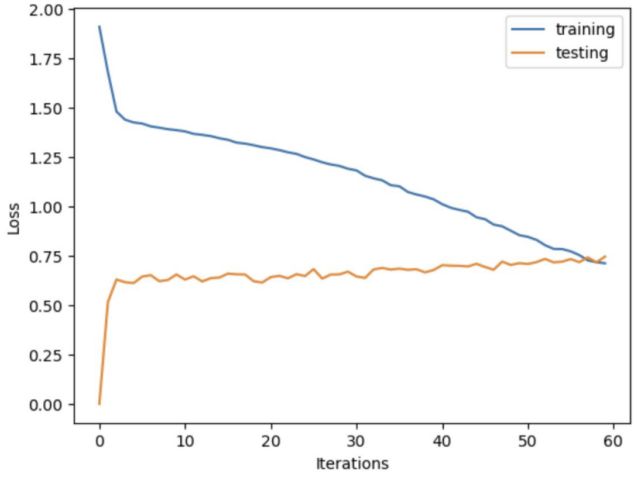
set around the initial figures, suggesting that future improvements might require exploring alternative model architectures, feature engineering techniques, or data quality enhancements.

4.4.3 Under and Over Sampling

Due to the discrepancy of recall and precision of certain classifications to others, we looked into under and oversampling. We started with undersampling the majority labels, Standard and Poor, to the numbers of Good. We ultimately suffered a worse accuracy, about 50%, so we abandoned this approach and tried oversampling.

Oversampling lead to a good increase of accuracy from the mid 60s to high 60s through doubling the amount of Good samples and taking 60% of Standard samples, matching to the number of Poor samples. We weren't aware of specific oversampling methods at the time we made this decision so we stuck with this.

4.5 Final Results




```
Accuracy :  0.734963325183374
Mean Square Error :  0.16528117359413205
Confusion Matrix for each label :
[[[2397  321]
  [ 196 1176]]

 [[2316  350]
  [ 569  855]]

 [[2522  274]
  [ 318  976]]]
Classification Report :
              precision    recall  f1-score   support

           0       0.79      0.86      0.82      1372
           1       0.71      0.60      0.65      1424
           2       0.78      0.75      0.77      1294

   micro avg       0.76      0.74      0.75      4090
   macro avg       0.76      0.74      0.75      4090
weighted avg       0.76      0.74      0.74      4090
 samples avg       0.74      0.74      0.74      4090
```

After we finalized training the model, from an accuracy standpoint we reached 73.5%, with a MSE of 0.165. Our precision for each of the classifications averages around the mid 70s whilst our recall is also around the mid 70s. However, we must note that the recall of the Standard classification is weaker at .60 and the recall of the Good classification is strong at .86.

Our loss curves show that our training loss decreases over time, but our testing loss appears to stay the same. We're not entirely sure as to why this is the case but we believe it could be due to the dataset. To do our best to mitigate this, we used early stopping and reduced the number of iterations to where our curves we be as close to each other as possible. Another factor could have been with our EDA of the dataset; we had noticed there were duplicate rows but they all contained unique IDs so we chose to keep the entries and we've could have applied more techniques to prepare the dataset better.

5 Conclusion and Discussion

We concluded that our results show reasonable accuracy for credit scoring and is sufficient for a person to reasonably learn what their credit score could be. Versus models that financial institutions use, they have much more data than we as students would be able to find. What is interesting to note that modern credit scoring systems are based off of FICO's rating system and use similar data but through their own way. We do believe that machine learning has its place with credit scoring as it may reveal trends or certain factors that FICO's rating system might miss even as the standard.

Despite this, we felt that it could be improved. Had we shifted our focus towards other models, it is likely our metrics would improve. Another factor was our dataset requiring a good amount of preprocessing work and effort to ensure smooth running. This could have been a sign for us to look for more datasets to work around, however we put in time to preprocess the data so we felt like we didn't want to waste our effort and preprocess another dataset. We also could have used more EDA methods to improve the dataset before training our neural network. However, this doesn't discount our effort and will provide a stepping stone for us to learn from.

Overall, the project was a decent success and displays our collaborative effort for credit scoring. We learned a lot together about the strengths and struggles with training a neural network and we're glad to say we should be able to apply what we've learned to different projects and ideas we may have in the future.

6 Road-map and Action Plan (Estimated Dates)

1. **Describing the Problem Scientifically:** January 25, 2024 - January 29, 2024
    - Define credit risk analysis, identifying key variables influencing loan defaults
    - Form a hypothesis and define success criteria for trained ML model
2. **Background Study (Literature Review or Related Work):** January 29, 2024 - February 9, 2024
    - Review current approaches in credit score analysis and analyze strengths and weaknesses of existing methods (data sets used, techniques studied, etc.)
3. **Dataset:** Understanding and Exploratory Data Analysis: January 29, 2024 - February 9, 2024
    - Find dataset(s) to be used for the model training from credible sources
    - Perform exploratory data analysis to understand data features, distributions and outliers
4. **Developing Accurate Prediction Model(s):** February 10, 2024 - February 27, 2024
    - Select a few machine learning models, train them using a dataset, and compare performance
    - Perform hyperparameter tuning: regularization, learning rate, number of epochs, termination criteria
    - Assessing and quantifying the fit of a candidate for a loan based on key factors.
5. E**valuation of the model(s) and testing the performance:** February 27, 2024 - March 1, 2024
    - Select evaluation criteria (type of error to use)
    - Finalize on the most appropriate model based on training and test results
        - Check for underfitting or overfitting
6. **Developing a basic web-based front-end to invoke and run the model(s) on input data and display the prediction output:** February 30, 2024 - March 4, 2024
    - Tech stack: Streamlit

7 Works Cited

References

[1] Xolani Dastile, Turgay Celik, and Moshe Potsane. Statistical and machine learning models in credit scoring: A systematic literature survey. *Applied Soft Computing*, 91:106263, 2020.

[2] Shrawan Kumar Trivedi. A study on credit scoring modeling with different feature selection and machine learning approaches. *Technology in Society*, 63:101413, 2020.