1 Introduction [15 points]

Group Details

Our group is composed of Jason Yang, Pierre Walker, James Wang, and Cyrus Fiori. Our Kaggle team name is *The Navier Folks*.

Rankings

AUC score on public leaderboard: 0.68608 (rank 5)
AUC score on private leaderboard: 0.69109 (rank 39) (ouch)

Links

Colab

- Data Preprocessing: link
- Logistic Ridge Regression: link
- Random forest: link
- Neural network (selected model): link

Piazza: link

Division of labor

PW coded and tested the random forest model, tested the model, made the Colab demo, and wrote the relevant report sections. **JY** pre-processed data in preparation for the machine learning models, contributed to the neural network, and wrote the relevant report sections. **JW** coded and tested the linear regression model, tested L_2 regularization, and wrote the relevant report sections. **CF** contributed to the neural network, wrote the relevant report sections, and completed other sections in the report not in the scope of the other group members.

2 Overview [15 points]

Models and techniques attempted

Three different techniques were considered: Random forests, regression, and neural networks, as described below.

Regression

We first attempted a logistic regression on the data to determine if the data could be separated while avoiding overfitting. To further prevent overfitting, L_2 regularization parameters were tested over a range of values. Logistic regression was able to achieve classification accuracies of up to 0.69 using L_2 regularization.

Random Forest

After attempting a linear model, we decided to consider a non-linear approach. As a result, a Random Forest Classifier was developed. Treating the minimum leaf size and maximum depth as hyperparameters, a range of values was scanned to obtain the optimal parameters based on the validation set. However, the errors obtained did not show significant improvement over logistic regression (typically predicting all cases to be "Fully Paid").

Neural Network

In parallel with random forest implementation, we developed a neural network as an alternative non-linear method. PyTorch packages were leveraged to implement this network.

Work timeline

Work on the project began on Monday Feb 7, 2022. At this stage, the team examined the data set and discussed the best course of action. It was decided that three of us attempt different approaches (listed above) and the final member perform data processing. By the end of the evening, we had developed codes to fit different models and processed the data in such a way that it was ready to be regressed. However, the errors obtained using these models were found to be very poor. The next day, the issues which resulted in the poor performance were identified and addressed; it was found that our models were reaching the top 10 in the kaggle competition. We spent a few more hours refining the data and models until we reached top 5 and decided that we were satisfied with the model developed.

3 Approach [20 points]

Data exploration, processing and manipulation

The steps we took for data processing were as follows:
1) Load in the raw data; **2)** Convert the percentage values to decimals; **3)** Concatenate the train and test data into a single dataframe; **4)** Drop the columns with too many categories (not useful due to high dimensionality) or too few categories (not useful as no information). **5)** Convert the datetimes into continuous values, namely "earliest_cr_line," "issue_d," and "emp_length." This was helpful because time has a numerical meaning. **6)** Convert loan grades to continuous values, as these have numerical meaning. **7)** Drop the "grades" and only keep "sub-grades" because these capture the same but more information. **8)** Impute the missing data by taking the mean of each numeric column; **9)** Impute the missing data for the categorical data by taking the most common value in each column; **10)** Convert the loan status to 1 if it is "Charged Off" or 0 if it is "Fully Paid." **11)** Turn the remaining categorical columns into one-hot encodings ("home_ownership","verification_status","purpose","intial_list_status"); and finally, **12)** save the data for use in the training and test predictions.

Details of models and techniques

We found that 85% of the data was classified as "Fully Paid," which caused many of the models to classify all of the samples as "Fully Paid." Thus, for our models, we decided to subsample the training set such that only 70% of the data was classified as "Fully Paid."

Logistic regression with either L_1 or L_2 regularization was selected as an exploratory model for fitting the data. Logistic regression is a low-parameter model, which is advantageous for preventing overfitting. Furthermore, the use of regularization with L_1 or L_2 logistic regression is advantageous for preventing undue sensitivity of the model to outliers in the training data. Notably, since L_1 regularization is sparse, using L_1 regularization can inform users of the features most important for classification. Finally, logistic regression is able to predict a classification probability instead of simple binary classification values, which drives down the AUC. Drawbacks of using logistic regression include that it is a linear classifier and risks underfitting or misclassifying points if the data is not linearly separable. Furthermore, logistic regression uses stochastic gradient descent to converge to its solution. If learning rate parameters are too small or large, the model will not converge. Also, if the data in the training set is heavily biased towards one class, as we have seen here, stochastic gradient descent will calculate gradients on batches that may not include the other class.

Initially, a random forest model seemed like an ideal choice for the given data as it is a strategy that can handle both continuous and categorical data types, not requiring any re-scaling or standardisation. Furthermore, as we observed a large variance by comparing the training and validation error; random forests, by design, are intended to reduce this variance. However, unlike logistic regression, random forests themselves are not easily interpretable, as is their black-box nature. As such, when the model performed poorly, aside from examining the performance, it was difficult to analyse the model's behaviour. Unfortunately, despite sklearn providing a very easy-to-use framework and optimising the hyperparameters, the results from the random forest model were found to be comparable to those obtained using logistic regression.

Neural networks (NNs) can theoretically approximate any function to arbitrary accuracy, which makes them a good non-linear model choice. Furthermore, they are good for ensembling, as NNs with different random initializations can learn different models. Thus, we also decided to train a NN.

4 Model Selection [20 points]

Scoring

We primarily used the area under the receiver-operator curve (AUC ROC). This is a good measure of performance, as it accounts for both precision (fraction of samples classified a certain way are correct) and recall (fraction of samples from a given classification that are identified). To further discern where our models were performing well and where they were performing poorly, we printed a classification report table with various classification metrics such as precision, recall, F1, etc.

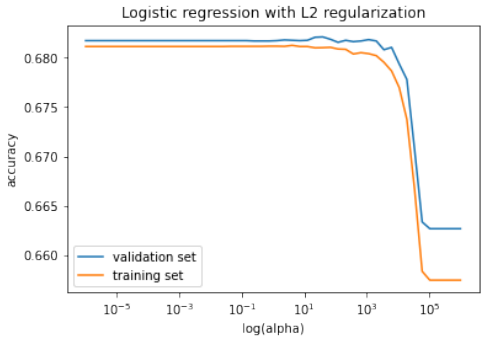
Validation and test

For all of our models, we generally performed validation by performing a single 80/20 split on the training data. The validation score was then used to optimize the relevant hyperparameters given below:

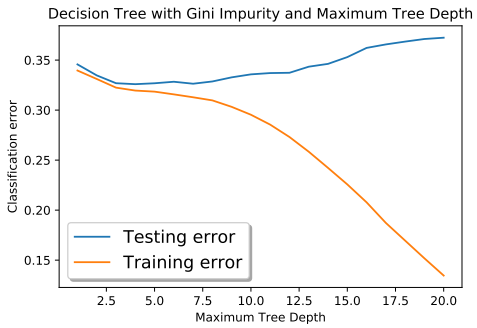
- Logistic Ridge Regression: λ
- Random Forest: minimum leaf size, maximum depth
- Neural Network: number of hidden layers, number of nodes per layer, number of training epochs

In particular, for the NN, we stopped training when the validation error did not seem to drop for several epochs. After optimizing the hyperparameters, the final model was trained on the entire training set. In the case of the NN, 16 NNs with different random initializations where ensembled to produce the final average prediction.

The logistic regression accuracy curves and the random forest classification error curves are shown below:



(a) Accuracy with α in logistic regression



(b) Accuracy with maximum depth in random forest

Figure 1

The decaying accuracy of the curve at high α indicates that at high enough α, the model starts to underfit the data. However, there are values of α that maximize accuracy and thus are good regularization parameters for our model. Similarly, in the case of the maximum depth of our random forest, increasing the depth reduces our training error; however, the variance increases. We observe a minima in our maximum depth for the testing error allowing us to identify the optimised hyperparameter. A similar process was repeated for the minimum leaf size. Overall, our NN produced the highest Kaggle AUC ROC score.

5 Conclusion [20 points]

Selected model

We selected the NN as our model of choice as it produced the best AUC ROC score on the Kaggle public leaderboard.

Insights

As a simple way to identify the top descriptors, we decided to look at the absolute value of the weights from ridge logistic regression. Top 10 descriptors with their respective weights from the ridge regression model, and whether they are positively or negatively correlated with charging off a loan.

Descriptor	Absolute Weight	Correlation
int_rate	0.602	+
annual_inc	0.240	-
term_(months)	0.198	+
sub_grade (higher means closer to A1)	0.111	-
dti	0.110	+
issue_d (higher means more recent)	0.108	+
open_acc	0.083	+
renewable_energy (categorical)	0.082	+
total_acc	0.065	-
car	0.043	-

Overall, we learned that data processing is incredibly time consuming. Furthermore, having the training and test data being sampled from the true underlying distribution of x and y is important. When we artificially sub-sampled x based on y, this may have caused our performance to drop on the private leaderboard.

Challenges

One key challenge in achieving models which scored similarly to the TA benchmark on Kaggle was the non-intuitive way in which the loan labels were encoded during model implementation. We originally encoded "Charged-off" as 0, when it should have been encoded as 1. Encoding "Charged-off" as zero resulted in our model's AUC calculation being the opposite of what it should have been (i.e., $1-AUC_{actual}$), resulting in artificially poor model performance. After reverting this error on advice of the TAs, we obtained accurate model performance.

Additionally, we could have tried other models, such as adaboost. Furthermore, we found that the precision on the "Fully Paid" category was very high, but the precision on the "Charged Off" category is very low, which suggests that the model is still very bad at learning when a loan will not be paid off. Future improvements could be made in this regard, potentially inspired by methods used in anomaly detection.

6 Extra Credit [5 points]

The AUC score acts as a reasonably effective metric when we care about both precision and recall for classification. However, in this case, we care more about identifying loans that will not be paid off. Thus, it might make more sense to put more weight on the recall of loans that are "Charged Off" or use the F1 score. Noticeably, many models on the leaderboard score well on AUC, but we suspect that their recall on "Charged Off" loans are very low, if not zero.

Of the models that we trained, the random forest and the neural networks are parallelizable. Namely, each of the trees in the random forest can be trained independently. Furthermore, among all the samples of the batch of backpropagation for the NN, the gradients can be calculated independently.

Other useful data could include credit score, age, and marital status for individuals.