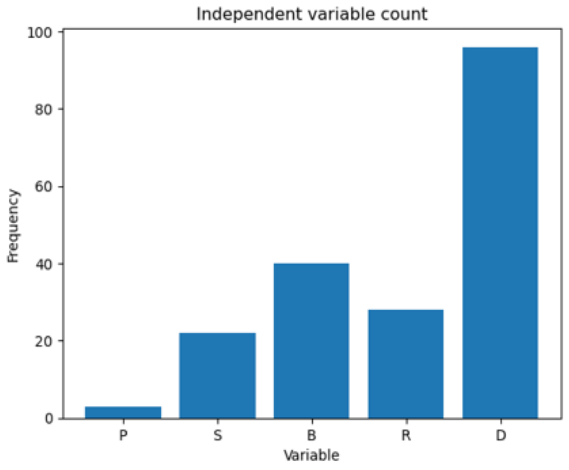
Credit Default Prediction

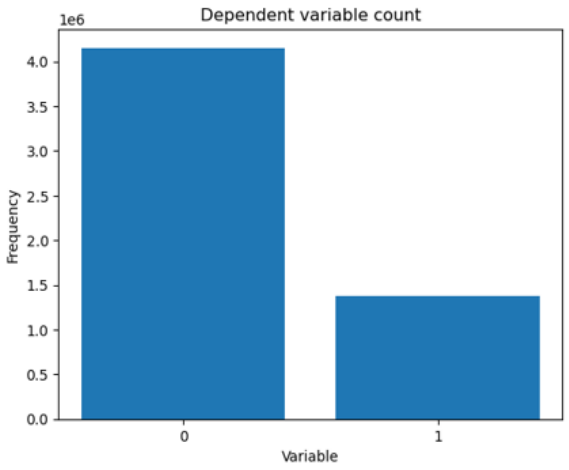
Ignacio Fernandez Alcaraz, Hong Hao (Brandon) Chen, Xinyu (Amber) Chen, Donald Donnelly

1. Overview

Credit cards play an essential role in the modern day of consumer spending habits. However, not all of them will be able to utilize this tool wisely. How do credit card issuers know if card holders will pay back what they owe? This is a complex problem, and thus credit default prediction is crucial to managing risk in the consumer lending market. Our objective is to predict the probability that a customer does not pay back credit card balance based on customer profile. No payment after 90 days of the latest statement due date is considered to be a default event. We are going to observe 18 months of data and apply machine learning techniques (classification model).

2. The Dataset



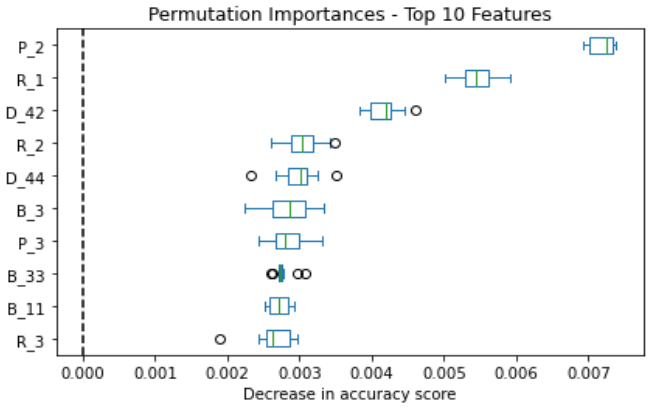



```
df.head()
```

customer_ID	S_2	P_2	D_39	B_1	B_2	R_1	S_3	D_41	B_3	...	D_137	D_138	D_139	D_140	D_141	D_142
73fe9d7c79be5f...	2017-03-09	0.938477	0.001734	0.008728	1.006836	0.009224	0.124023	0.008774	0.004707	...	NaN	NaN	0.002426	0.003706	0.003819	NaN
73fe9d7c79be5f...	2017-04-07	0.936523	0.005775	0.004925	1.000977	0.006153	0.126709	0.000798	0.002714	...	NaN	NaN	0.003956	0.003166	0.005032	NaN
73fe9d7c79be5f...	2017-05-28	0.954102	0.091492	0.021652	1.009766	0.006817	0.123962	0.007599	0.009422	...	NaN	NaN	0.003269	0.007328	0.000427	NaN
73fe9d7c79be5f...	2017-06-13	0.960449	0.002455	0.013687	1.002930	0.001372	0.117188	0.000685	0.005531	...	NaN	NaN	0.006119	0.004517	0.003201	NaN
73fe9d7c79be5f...	2017-07-16	0.947266	0.002483	0.015190	1.000977	0.007607	0.117310	0.004654	0.009308	...	NaN	NaN	0.003672	0.004944	0.008888	NaN

A total of 189 features are anonymized and normalized, and fall into the five general categories: P = Payment (3 features), S = Spend (22 features), B = Balance (40 features), R = Risk (28 features), and D = Delinquency (96 features). The dataset includes a customer ID column and the 189 features, which are broken down into a date column, 177 continuous variables and 11 categorical variables. For the outcome, or dependent variable, 0 = no default and 1 = default.

Before doing any data cleaning, we fit a random forest classifier model in order to analyze the MDI and permutation feature importance. The top five features of importance goes to P_2, R_1, D_62, R_2, and B_3. As seen on the graph to the right, P_2, R_1, R_2, and B_3 were among the top features in the permutation importance analysis as well, however D_62 did not overlap between the two tests.



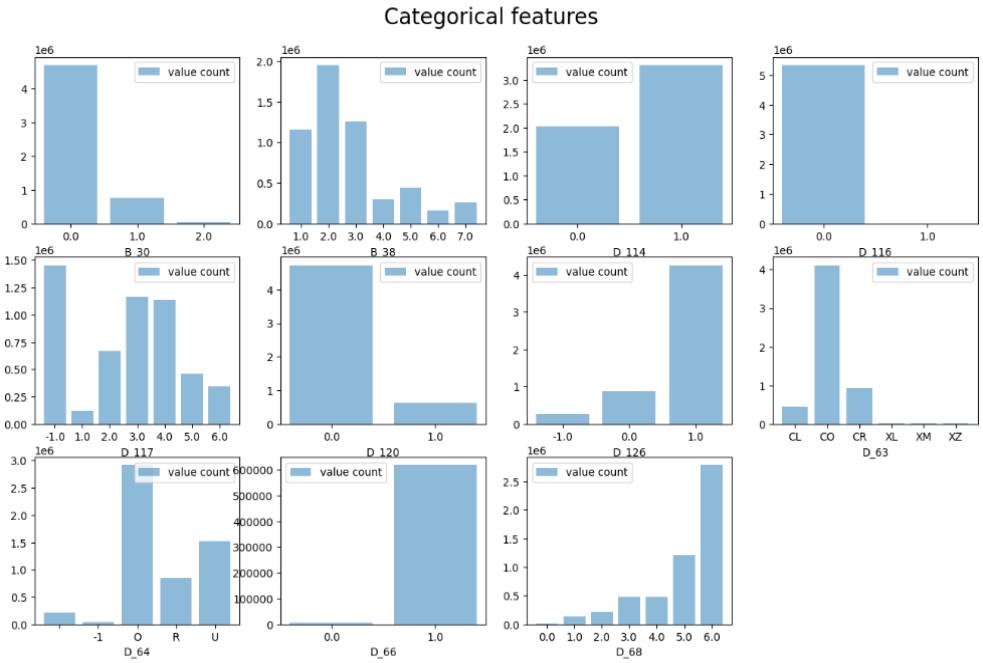
3. Data Cleaning

3.1 NaN handling

One of the first things we noticed about the data was that there were many missing values. In order to begin data exploration and modeling, we initially imputed the mean value of each column to fill the missing data. It was infeasible to drop the rows with missing values because virtually every row in the dataset contained a NaN. After our first few iterations, we decided to dig into these missing data points a little deeper.

We found there were 3 groups of variables with the same number of NaN values; two groups with 13 variables and one group with 11 variables. Unsurprisingly, all three groups had their missing values in the same rows. We therefore believed that these NaN could be telling us something about a customer (i.e. a new account, skipped reporting, etc.). We then decided to fill all NaN values with 0, and saw our assumption was ultimately correct as all our previous models improved in accuracy.

3.2 Categorical variables



Since there was no obvious indication the categorical variables were ordinal, we created dummy variables for all the categorical variables, resulting in a new total feature count of 224.

3.2 Permutation Importance - Negative Values

As a result of our permutation importance analysis, we actually noticed there were many variables that, on average, increased the accuracy of our random forest when removed. We therefore removed them as features, which resulted in a small increase in the accuracy of the forest.

3.3 Removing the Date Column

The dependent variable (1 for default, 0 for no default) for every row of a customer that defaulted at any point in time was 1. This led us to conclude that the date column was not beneficial because we could not use it to tell us when a default occurred. The data column was then dropped, which converted the dataset from panel data to cross-sectional data, rendering it much easier to work with.

3.4 Grouping by Customer ID

Lastly, due to the fact that there were multiple rows for one customer, we tried grouping by customer ID and using the mean of each feature. Not only did this cut the data points from 5,531,451 to 458,913, it also improved the accuracy on all our models. At this point, we were comfortable with concluding our data cleaning and moving onto modeling.

Final dataset: total data points: 458,913, total features: 177

4. Modeling

Since the dataset was made up of 150+ features and now 458,913 data points, it was decided that we should explore models that can handle large volumes such as random forest, SVM, and neural networks.

4.1 Random Forest Classifier

The original test accuracy is around 87% with train accuracy of 99%. By setting min_samples_leaf to 10, 20, 30, 40, 50, the test accuracy increases to 88% when min_samples_leaf equals to 20 and decreases after 20. Tuning other parameters did not increase the accuracy further.

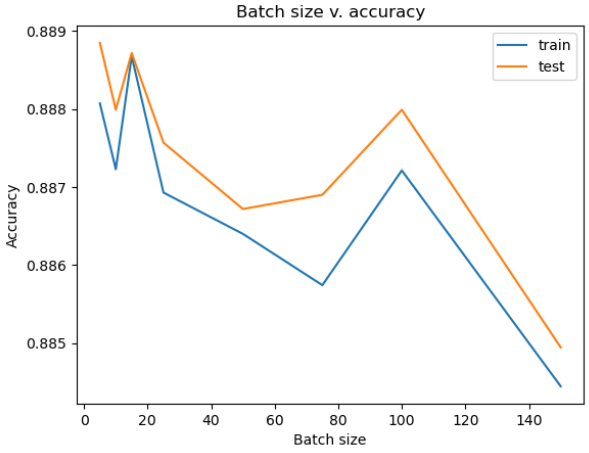
4.2 SVM

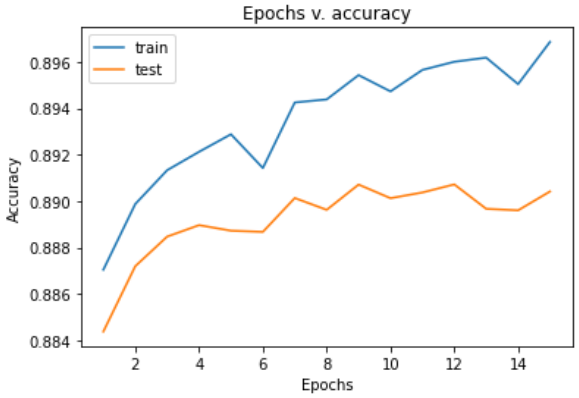
Due to the large amount of data, we had to run the features through a RBF kernel approximator for performance. The approximator used random Fourier features to approximate a RBF kernel feature map. However, it only had a train and test accuracy of around 74%. It didn't do as well as the decision tree or the neural network, so we didn't pursue further.

4.3 Neural network

We ended up using keras instead of sklearn due to the keras package's flexibility and robustness when it comes to neural nets, even though the parameters were not as straightforward. The final neural network used had 3 hidden layers with 20, 10, and 10 nodes using ReLU activation functions, and an output node using the sigmoid activation function (since our desired output was 0 or 1).

For the batch size, a keras parameter that defines the number of samples that will be propagated through the network, we tried multiple values and graphed them against the accuracy on our test data. We ultimately found that a batch size of 20 prevented overfitting while producing the best accuracy on the test set. We performed a similar test on epochs, which is generally defined as "one pass over the entire dataset". The accuracy generally increased with the number of epochs until around 10, so we chose 10.



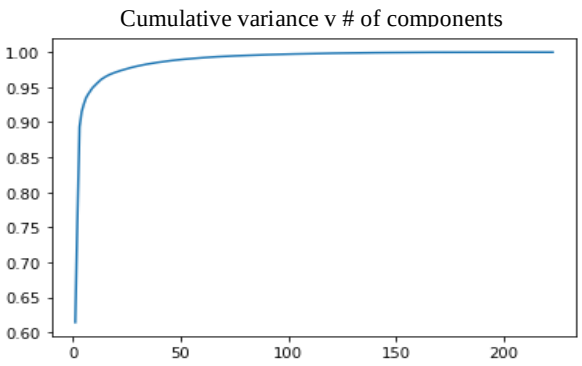


Our final neural net produced a train accuracy of 89.3%, and a test accuracy of 88.9%.

4.4 K-means clustering

Our first attempt at K-means clustering over all the features proved to be unsuccessful. Although we did observe an elbow in the scree plot at 3 clusters, two of the clusters only contained one data point. Alternatively, we tried K-means clustering on the top 11 features from our MDI feature importance analysis. This produced two clusters, both of which had a significant number of data points.

Interestingly, when running neural networks on the clusters independently, the model on the first cluster resulted in a test accuracy of 94.1%, while the second cluster's accuracy fell to 79.1%. We thought about running a separate neural net for data points identified to be in the first cluster, and then returning to our original neural net for the second cluster, but for simplicity and computation time, we moved away from the idea and stuck with our original model.



4.5 Neural Networks using PCA

We attempted to make the Neural Network more accurate by performing PCA on the data. To figure out how many components to use in the PCA, we looped from 1 to 177 (the number of features) and graphed the cumulative variances for each value. We saw that at 50 components, around 98% of the variance was accounted for. We then transformed the features dataset into 50-component PCA and retrained the neural network. The result was a train and test accuracy of 88.9%, and a train and test loss of around 8.2%. We then tried using PCA with a component size equal to the feature size, essentially replacing all 177 features. Retraining the neural network resulted in an increased accuracy on the test set to 89.2%, which was ultimately our final model.

5. Conclusion

The final model we went with was the neural network defined in 4.5, where all 177 features were converted into PCA components. Although the neural net's output varied slightly after each run due to the nature of neural networks, the accuracy on the test data was consistently superior to the rest of the models we tried. Our final test accuracy was 89.2%, where the model incorrectly predicted default 5.2% of the time, incorrectly predicted no default 5.7% of the time, and correctly predicted either default or no default 89.2% of the time.