Springboard--DSC
Capstone Project 2
Lending Club Loan Prediction
By Yitian Cauthen
June, 2021

## 1 Introduction

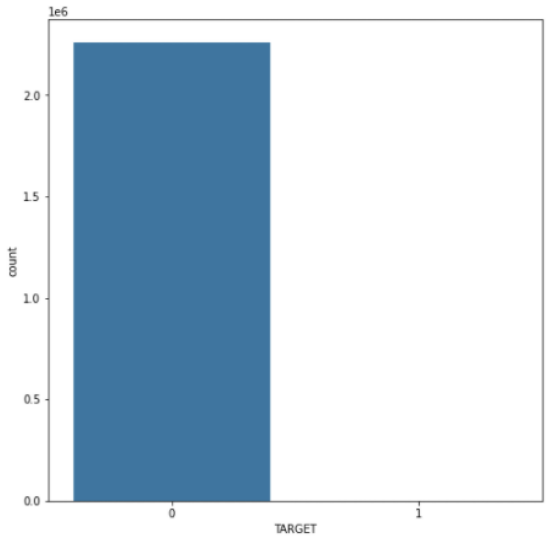
Using a collection of accepted Lending Club Loans between 2007 and 2018, the purpose of this project is to build a model that predicts the likelihood or probability a person will default on a loan given their current status and qualities. The dataset includes over 2 million observations with 151 features composed of each individual's information. I use 4 different classification algorithms with 2 resampling methods in order to identify crucial features in the prediction that may determine the probability of default.

## 2 Approach

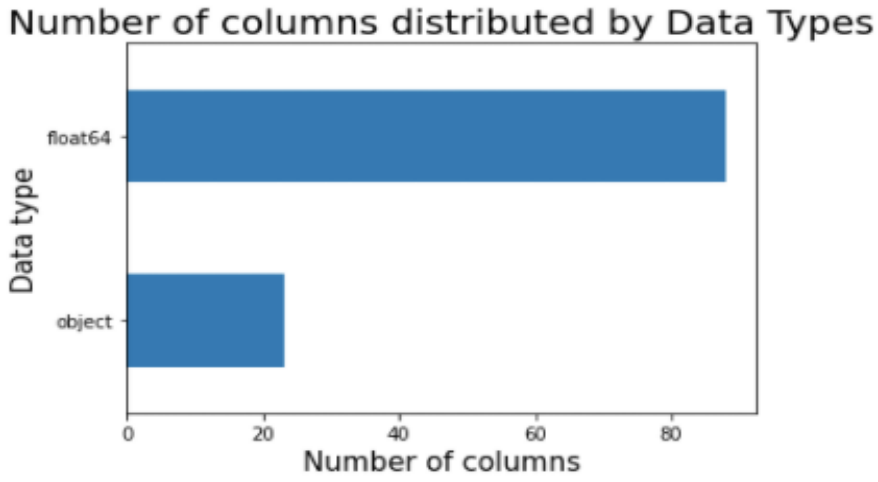
2.1 Data Acquisition and Wrangling:
The dataset was taken from Kaggle and was roughly 2.5gb in size, I used SQLite to read the file and removed all columns that were giving me dtype warnings i.e. columns with URLs. Columns with 90% or more missing data were removed. Data types were examined but not changed. A few minor changes were made i.e. removing the symbols from employment length. The file was then pickled for further analysis.

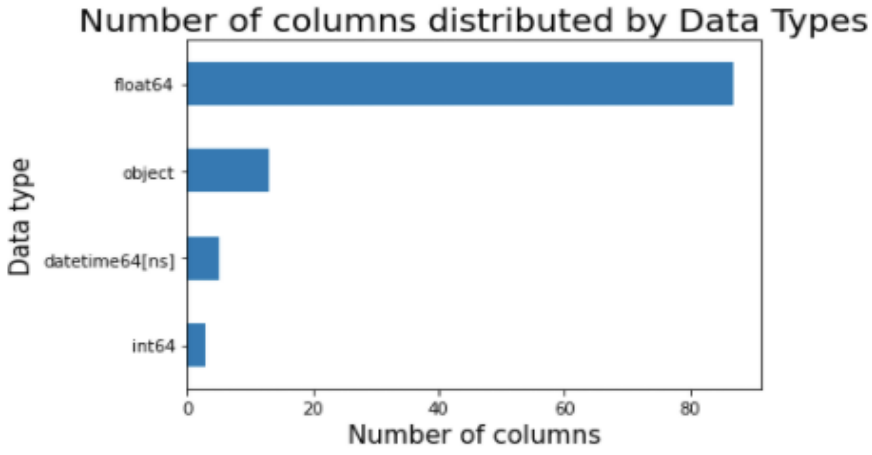
2.2 Storytelling and Inferential Statistics:
This was a case of a class imbalance problem. Upon further examination of the Target variable, I noticed that it only made up 40/2260628 or roughly .001% of the dataset.

```
In [7]: print(str(df['TARGET'].value_counts()[0]), df['TARGET'].value_counts()[0]/len(df) * 100)
        print(str(df['TARGET'].value_counts()[1]), df['TARGET'].value_counts()[1]/len(df) * 100)

        2260628 99.99823061148298
        40 0.0017693885170224023

In [8]: plt.figure(figsize=(8, 8))
        sns.countplot('TARGET', data=df)

        C:\Users\user\anaconda3\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From
        version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in a
        n error or misinterpretation.
          FutureWarning
Out[8]: <AxesSubplot:xlabel='TARGET', ylabel='count'>
```



This is normal for a healthy business because you obviously do not want the number of anomalous transactions (in this case loans that defaulted) to exceed those that were paid off. I then went over each feature and imputed the missing values using mean, median, and mode paying careful attention to what each of the features were and what they meant. I also made sure that the data types for each of the features were appropriate.




The resulting changes led to 2 additional data types and a reduction from 151 to 108 features while maintaining the same number of rows.

2.3 Baseline Modeling:
For the baseline models I used Logistic Regression and Random Forest Classifier on an 80/20 split without any resampling methods. The results are shown below.

Logistic Regression no resampling:

```
[Training Classification Report]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1805744
           1       0.00      0.00      0.00        32
```

```
    accuracy                           1.00   1805776
   macro avg       0.50      0.50      0.50   1805776
weighted avg       1.00      1.00      1.00   1805776

[Test Classification Report]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    451437
           1       0.00      0.00      0.00         8

    accuracy                           1.00    451445
   macro avg       0.50      0.50      0.50    451445
weighted avg       1.00      1.00      1.00    451445
```

Random Forest No Resampling:

```
[Training Classification Report]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1805744
           1       1.00      0.97      0.98        32

    accuracy                           1.00   1805776
   macro avg       1.00      0.98      0.99   1805776
weighted avg       1.00      1.00      1.00   1805776

[Test Classification Report]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    451437
           1       0.00      0.00      0.00         8

    accuracy                           1.00    451445
   macro avg       0.50      0.50      0.50    451445
weighted avg       1.00      1.00      1.00    451445
```

With the lack of data points, I decided to generate synthetic samples using SMOTE and ADASYN to solve for the class imbalance problem. Since the 2 baseline models proved to have performed very poorly and seemed to have overfit, it is evident there is a class imbalance problem since the model may only be including examples from the majority class.

2.4 Extended Modeling:

I decided to include 2 additional classification algorithms (XGB and LGBM) along with 2 resampling methods (SMOTE and ADASYN) to hopefully solve the class imbalance problem for a total of 4 classification algorithms with 2 resampling methods. The models seemed to have performed better than the baseline, although the performance still wasn't great. It was predicting a few points in the minority class which is still better than none. A few of the biggest changes I made were to append an evenly divided minority class to a 80/20 train test split and use a combination of undersampling and oversampling. The undersampling occurred before

resampling where I took a random sample of 500,000 from the original 2,000,000 without including the minority class. I then appended an evenly divided minority class to each set.

## 3 Findings

Although the baseline model for logistic regression performed the best out of the 4 models in terms of accuracy with 79% accuracy, LGBM with hyperparameter tuning performed the best from a business standpoint.

Performance Metrics for 4 Classification Algorithms with 80/20 Split

SMOTE

| | Model | Accuracy | Precision | Recall | F1 | Support |
|---|---|---|---|---|---|---|
| 0 | lr_train_0 | 0.98790 | 1.00000 | 0.98000 | 0.99000 | 400000 |
| 1 | lr_train_1 | 0.98790 | 0.98000 | 1.00000 | 0.99000 | 400000 |
| 2 | lr_test_0 | 0.97520 | 1.00000 | 0.98000 | 0.99000 | 100000 |
| 3 | lr_test_1 | 0.97520 | 0.00000 | 0.62000 | 0.00000 | 8 |
| 4 | rf_test_0 | 0.99990 | 1.00000 | 1.00000 | 1.00000 | 100000 |
| 5 | rf_test_1 | 0.99990 | 0.00000 | 0.00000 | 0.00000 | 8 |
| 6 | xgb_train_0 | 1.00000 | 1.00000 | 1.00000 | 1.00000 | 400000 |
| 7 | xgb_train_1 | 1.00000 | 1.00000 | 1.00000 | 1.00000 | 400000 |
| 8 | xgb_test_0 | 1.00000 | 1.00000 | 1.00000 | 1.00000 | 100000 |
| 9 | xgb_test_1 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 8 |
| 10 | lgbm_train_0 | 1.00000 | 1.00000 | 1.00000 | 1.00000 | 400000 |
| 11 | lgbm_train_1 | 1.00000 | 1.00000 | 1.00000 | 1.00000 | 400000 |
| 12 | lgbm_test_0 | 0.99990 | 1.00000 | 1.00000 | 1.00000 | 100000 |
| 13 | lgbm_test_1 | 0.99990 | 0.00000 | 0.00000 | 0.00000 | 8 |

ADASYN

| | Model | Accuracy | Precision | Recall | F1 | Support |
|---|---|---|---|---|---|---|
| 0 | lr_train_0 | 0.98790 | 1.00000 | 0.98000 | 0.99000 | 400000 |
| 1 | lr_train_1 | 0.98790 | 0.98000 | 1.00000 | 0.99000 | 400000 |
| 2 | lr_test_0 | 0.97520 | 1.00000 | 0.98000 | 0.99000 | 100000 |
| 3 | lr_test_1 | 0.97520 | 0.00000 | 0.62000 | 0.00000 | 8 |
| 4 | rf_test_0 | 0.99990 | 1.00000 | 1.00000 | 1.00000 | 100000 |
| 5 | rf_test_1 | 0.99990 | 0.00000 | 0.00000 | 0.00000 | 8 |
| 6 | xgb_train_0 | 1.00000 | 1.00000 | 1.00000 | 1.00000 | 400000 |
| 7 | xgb_train_1 | 1.00000 | 1.00000 | 1.00000 | 1.00000 | 400000 |
| 8 | xgb_test_0 | 0.99990 | 1.00000 | 1.00000 | 1.00000 | 100000 |
| 9 | xgb_test_1 | 0.99990 | 0.00000 | 0.00000 | 0.00000 | 8 |
| 10 | lgbm_train_0 | 1.00000 | 1.00000 | 1.00000 | 1.00000 | 400000 |
| 11 | lgbm_train_1 | 1.00000 | 1.00000 | 1.00000 | 1.00000 | 400000 |
| 12 | lgbm_test_0 | 0.99990 | 1.00000 | 1.00000 | 1.00000 | 100000 |
| 13 | lgbm_test_1 | 0.99990 | 0.00000 | 0.00000 | 0.00000 | 8 |

LGBM w/ SMOTE & Hyperparameter Optimization

| Model | Accuracy | Precision | Recall | F1 | Support |
|---|---|---|---|---|---|
| lgbm_tuned_tr_0 | 0.99 | 1.00 | 0.99 | 1.00 | 400000 |
| lgbm_tuned_tr_1 | 0.99 | 1.00 | 1.00 | 1.00 | 400000 |
| lgbm_tuned_tst_0 | 0.99 | 1.00 | 0.99 | 1.00 | 100000 |
| lgbm_tuned_tst_1 | 0.99 | 0.00 | 0.10 | 0.00 | 20 |

Confusion Matrix Test LGBM w/ SMOTE &  Hyperparameter Optimization:
[[99113   887]
[   18     2]]

Confusion Matrix Test LR w/ SMOTE:
[[79410 20590]
[    8    12]]

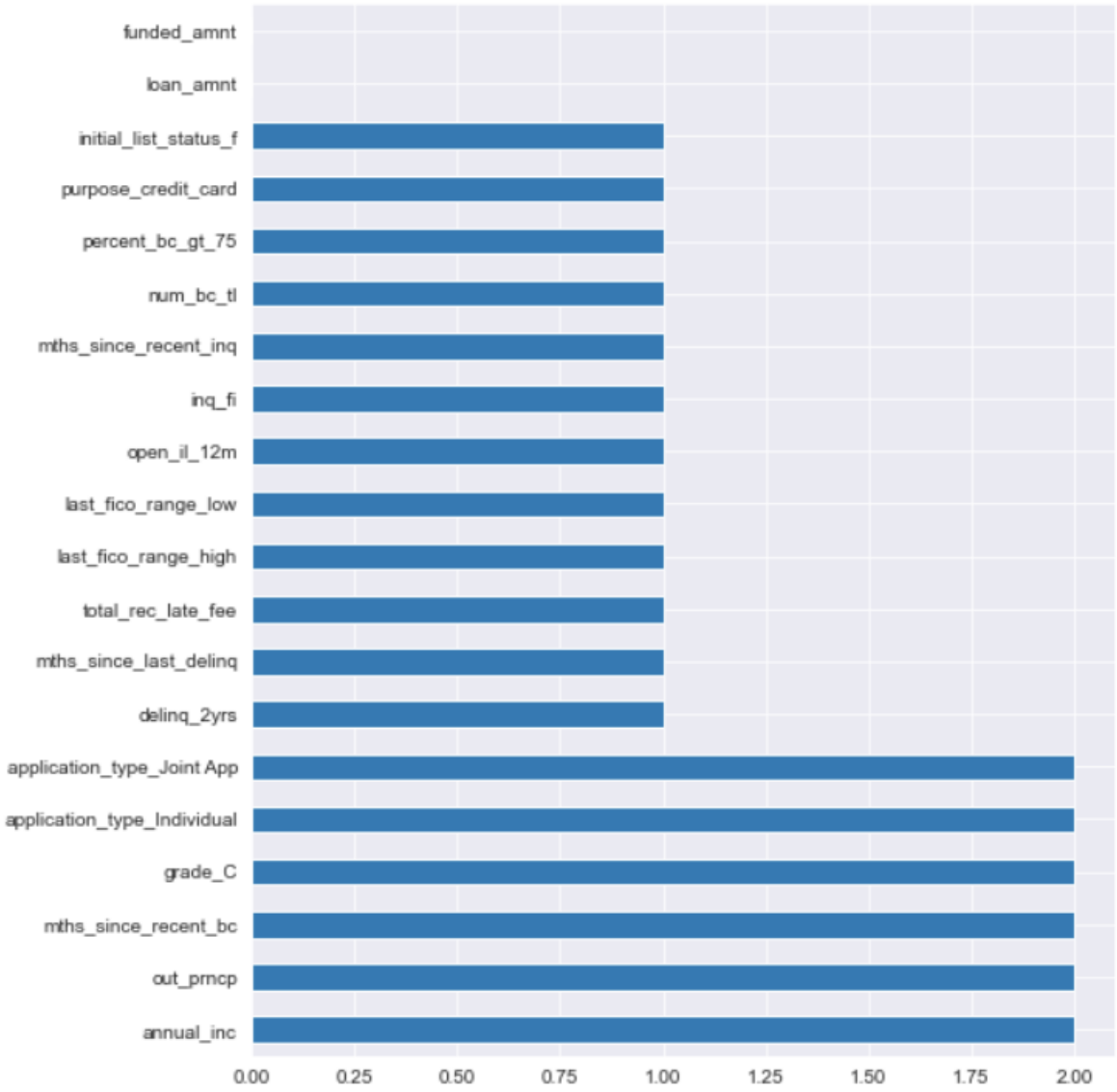
Upon further analysis of the confusion matrices of both, although LR correctly predicts 12 true positives there are still 20590 instances of false negatives.

Consider a simplified example of an average loan of $10,000 over 3 years at 10% interest. The loan company would have $3,000 in gross profit. If the company invested say 5% instead they could potentially earn $1,500.

If net profit was $1500 for a good loan, a bad loan would lose $10,000 plus the $1,500 totalling $11,500.

In the test set for LR w/ SMOTE, although the company would avoid losing 12 * $11,500  or $138,000, it would give up 20590 * $1,500 or about $30.7 million while the LGBM w/ SMOTE & HO test set would only give up 887 * $1,500 or roughly $1.3 million.

## 4 Conclusions and Future Work




This was an example of a binary classification problem where the goal was to predict the probability of an individual defaulting on a loan given their attributes. Upon closer analysis of the feature importances, more work can be done to create a better predictive model. Although it seemed like annual income, grade, remaining outstanding principle, months since recent balance, and application type had the highest feature importance, the relevant features could be applied to a new model to increase the performance.

## 5 Recommendations for the Clients

1. Collect more data on individuals with defaulted loans
2. Consider examining the aforementioned features with higher importance more closely with regards to new applicants
3. Try to build new models with new resampling methods or without imputing missing data i.e. build a model only with available data

## 6 Consulted Resources


1. https://www.kaggle.com/pragyanbo/a-hitchhiker-s-guide-to-lending-club-loan-data
2. https://towardsdatascience.com/predicting-loan-repayment-5df4e0023e92
3. https://www.kaggle.com/mlisovyi/lightgbm-hyperparameter-optimisation-lb-0-761
4. https://towardsdatascience.com/data-cleaning-with-python-and-pandas-detecting-missing-values-3e9c6ebcf78b
5. https://pbpython.com/pandas_dtypes.html
6. https://machinelearningmastery.com/calculate-feature-importance-with-python/
7. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
8. https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8