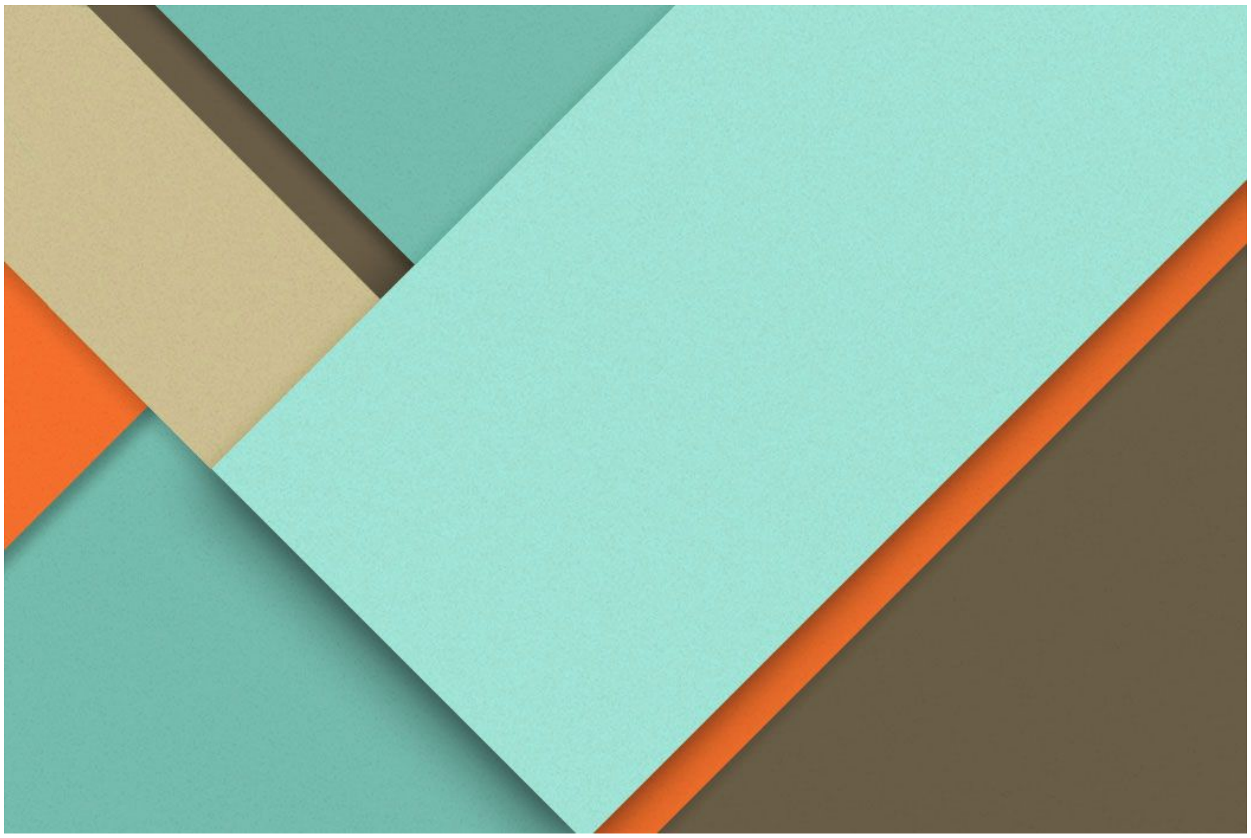


Probability of Defaulting

Springboard Capstone 1 Final Project

Brock Nosbisch

09.09.2019

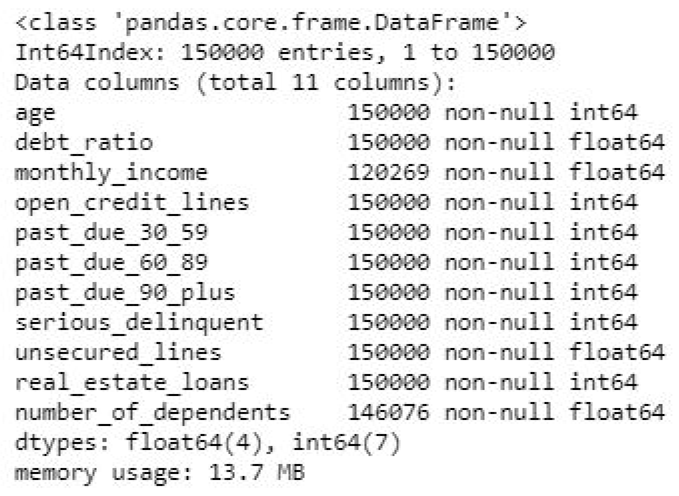
Problem Statement

My client is any bank, credit union, or other financial institution that gives loans to individuals. Predicting whether someone will default on their loan is key to the company deciding on whether to give that person a loan or not. The original intent was to predict the probability of each person defaulting but it transformed into finding the best model to produce the highest amount of revenue for the company.

The Dataset

The dataset used in this project came from a Kaggle competition and the dataset was manually downloaded and imported into the Data Wrangling notebook. Although I am using an existing Kaggle competition's dataset, the end result (or goal) of this project is a little different than the competition's. I am emphasizing on the fact to use several algorithms (yet to be defined) and review the performance, strengths, and weaknesses of each.

The dataset includes 11 features for 150,000 individuals. The data contains history for the past 2 years.



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150000 entries, 1 to 150000
Data columns (total 11 columns):
age                       150000 non-null int64
debt_ratio                150000 non-null float64
monthly_income            120269 non-null float64
open_credit_lines         150000 non-null int64
past_due_30_59            150000 non-null int64
past_due_60_89            150000 non-null int64
past_due_90_plus          150000 non-null int64
serious_delinquent        150000 non-null int64
unsecured_lines           150000 non-null float64
real_estate_loans         150000 non-null int64
number_of_dependents      146076 non-null float64
dtypes: float64(4), int64(7)
memory usage: 13.7 MB
```

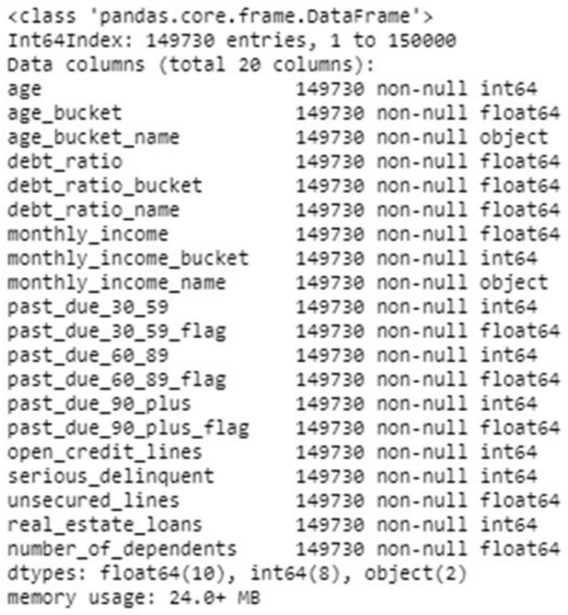
Data Wrangling

The dataset was not clean so some Data Wrangling had to be performed. I used several packages to figure out the null values, "bad" data, outliers, etc. The data was indexed, data was reordered, some data removed, some null values set to -1, several buckets/bins created for fields, etc.

I started data wrangling by importing the dataset provided and then set the index and re-ordered the columns so that it is easier to see the correlations in a table format. Two functions (one for min/max values and one for most/least common values) were created so that they can be used when reviewing the data values. Then I reviewed the DataFrame and each field/column individually and made notes on if there appeared to be any NULL values, "bad" data, outliers, etc.

After reviewing the data, I started cleaning the data by first removing the record where the Age was 0. This was only a single record in the dataset. Then I had originally removed all records where Monthly Income was NULL but I am now including this set of records in the dataset. I also removed the records that had 96 or 98 in the past due columns. NULL Monthly Incomes are set to -1. Next I updated all "Number of Dependents" values to -1 where the value was NULL. I want to review the correlation of the Nulls and 96/98 values to the other data to see if I see trends before deciding what to do with the data. I also added a new column to store Monthly Incomes by bucket/bin and converted any data types that were wrong.

Updated Dataset After Data Wrangling



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 149730 entries, 1 to 150000
Data columns (total 20 columns):
age                        149730 non-null int64
age_bucket                 149730 non-null float64
age_bucket_name            149730 non-null object
debt_ratio                 149730 non-null float64
debt_ratio_bucket          149730 non-null float64
debt_ratio_name            149730 non-null float64
monthly_income             149730 non-null float64
monthly_income_bucket      149730 non-null int64
monthly_income_name        149730 non-null object
past_due_30_59             149730 non-null int64
past_due_30_59_flag        149730 non-null float64
past_due_60_89             149730 non-null int64
past_due_60_89_flag        149730 non-null float64
past_due_90_plus           149730 non-null int64
past_due_90_plus_flag      149730 non-null float64
open_credit_lines          149730 non-null int64
serious_delinquent         149730 non-null int64
unsecured_lines            149730 non-null float64
real_estate_loans          149730 non-null int64
number_of_dependents       149730 non-null float64
dtypes: float64(10), int64(8), object(2)
memory usage: 24.0+ MB
```

Field Definitions

30-59 Days Past Due: Number of times borrower has been 30-59 days past due but no worse in the last 2 years.

60-89 Days Past Due: Number of times borrower has been 60-89 days past due but no worse in the last 2 years.

90+ Days Past Due: Number of times borrower has been 90 days or more past due.

Serious Delinquent: Flag to show if a person experiences 90 days past due delinquency or worse.

Age: Age of borrower in years.

Age Bucket: Age of borrower bucketed in 10 year increments (20-29, 30-39, 40-49, ... 90-99, 100+)

Monthly Income: Monthly income.

Monthly Income Range: Monthly income bucketed (0-499, 500-999, 1000-2499, 2500-4999, 5000-7499, 7500-9999, 10000-24999, 25000-49999, 50000+); NULL means that no monthly income value was provided.

Debt Ratio: Monthly debt payments, alimony, living costs divided by monthly gross income.

Debt Ratio Quantile: 10% increments of the Debt Ratio. Each 10% bucket contains approximately the same amount of people.

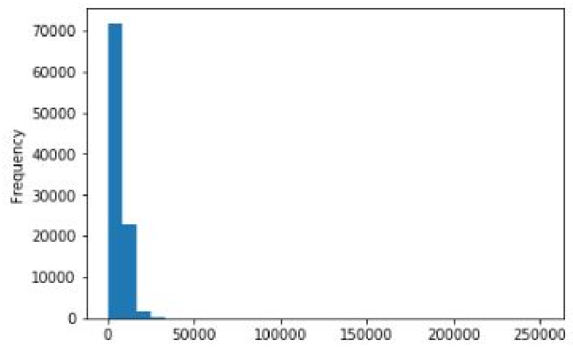
Number of Open Credit Lines (and Loans): Number of open loans (installment like car loan or mortgage) and lines of credit (e.g. credit cards)

Unsecured Lines: Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits.

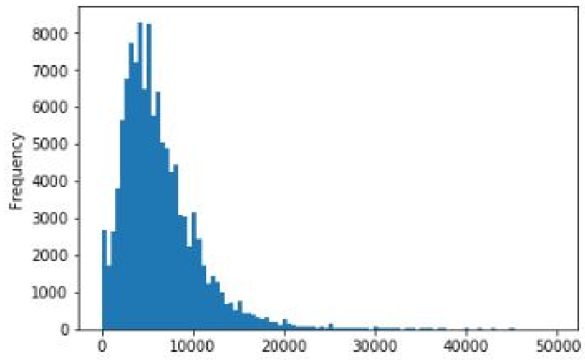
Real Estate Loans: Number of mortgage and real estate loans including home equity lines of credit.

Number of Dependents: Number of dependents in family excluding themselves (spouse, children, etc.)

Monthly Income Counts (entire dataset, 30 bins)



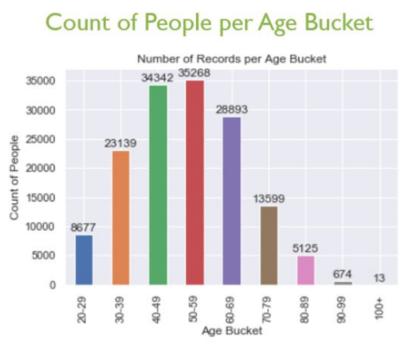
Monthly Income Counts (monthly income < $50k, 100 bins)

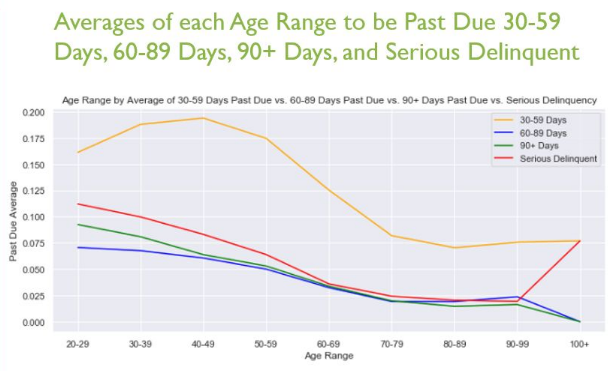


Exploratory Data Analysis

During EDA, several observations of interest were found.

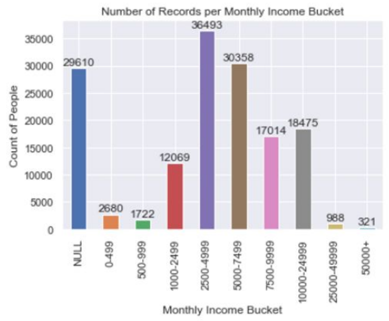
1) The younger in age you are, the higher the probability that you will be 60-89 Days Past Due, 90+ Days Past Due, and/or Serious Delinquent.



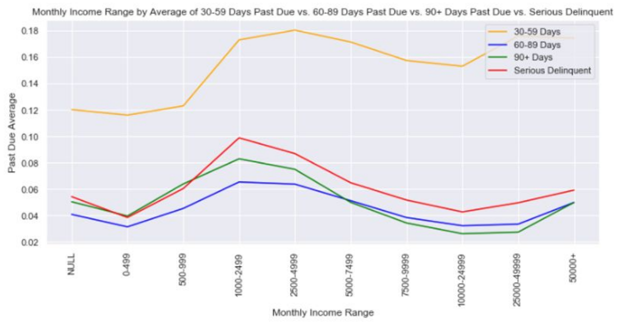


2) The 1000-2499 and 2500-4999 Monthly Income Ranges are most likely to be Past Due when compared to other income ranges.

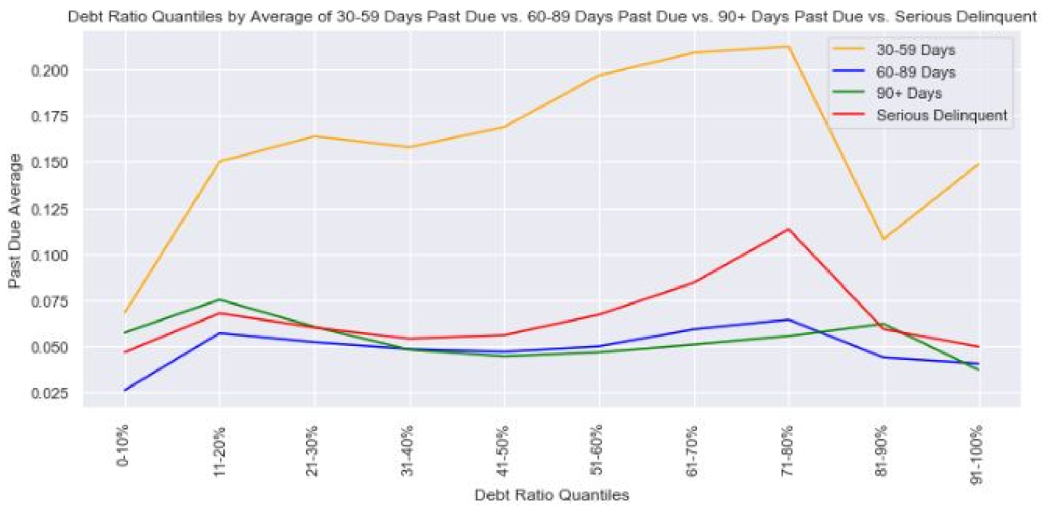
Count of People per Monthly Income Range



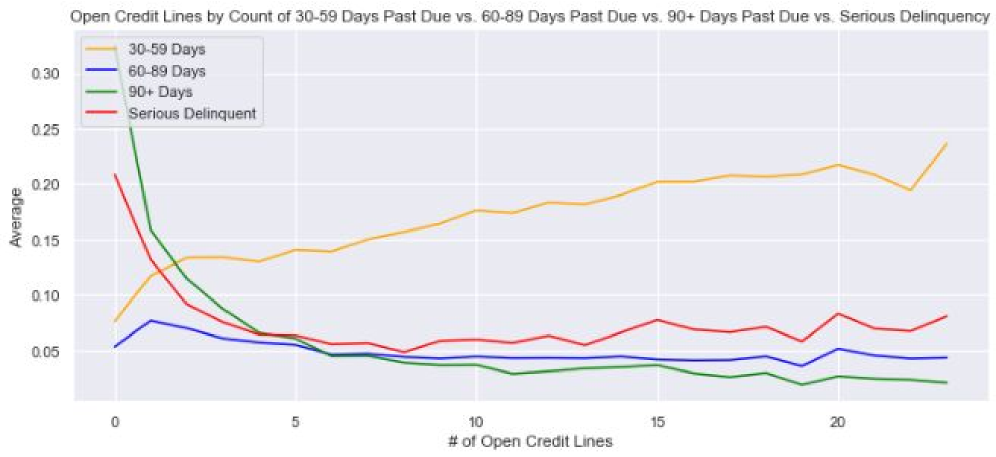
Averages of each Income Range to be Past Due 30-59 Days, 60-89 Days, 90+ Days, and Serious Delinquent



3) The Debt Ratio quantiles show that the Past Due Bucket averages gradually increase from the 0-10% quantile, dip a little bit after the 11-20% quantile, stay flat until the 41-50% quantile, and then keep rising up to the 71-80% quantile where the average is highest.



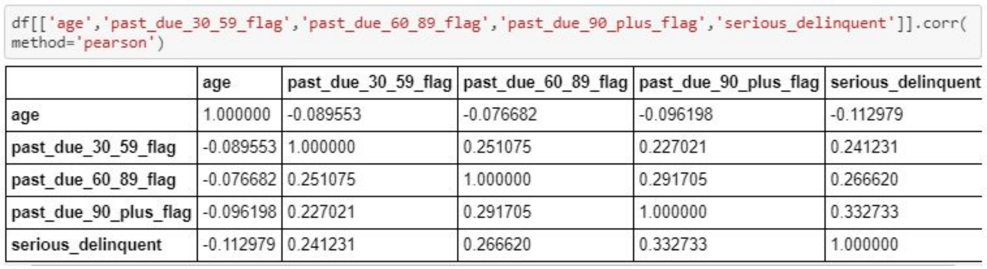
4) The 60-89, 90+, and Serious Delinquent averages are flat after the # of Open Credit Lines reaches ~4 and above. (Do note that any Open Credit Limits with < 500 people are excluded (# > 23))



Statistical Inference

To confirm some of the above observations that may prove to be useful, I applied inferential statistics and found the correlations that are of statistical significance and a few that have practical significance. The details of the below high-level results can be found here.

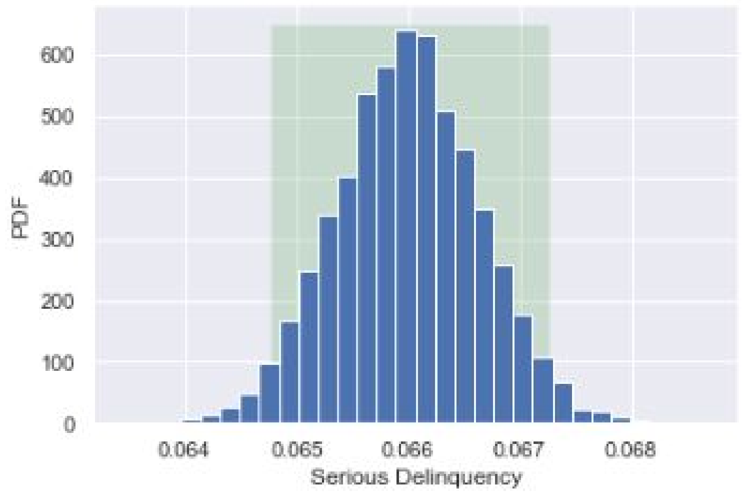
- Past Due Buckets correlated to Serious Delinquency (Statistical and Practical Significance)
- Monthly Income Buckets, Debt Ratio, and Open Credit Lines correlated to Serious Delinquency (Statistical Significance)
- Past Due Buckets correlated to Age (Statistical Significance)



```
df[['age','past_due_30_59_flag','past_due_60_89_flag','past_due_90_plus_flag','serious_delinquent']].corr(
method='pearson')
```

	age	past_due_30_59_flag	past_due_60_89_flag	past_due_90_plus_flag	serious_delinquent
age	1.000000	-0.089553	-0.076682	-0.096198	-0.112979
past_due_30_59_flag	-0.089553	1.000000	0.251075	0.227021	0.241231
past_due_60_89_flag	-0.076682	0.251075	1.000000	0.291705	0.266620
past_due_90_plus_flag	-0.096198	0.227021	0.291705	1.000000	0.332733
serious_delinquent	-0.112979	0.241231	0.266620	0.332733	1.000000

- The Bootstrap Estimate of Serious Delinquent Mean shows the 95% confidence interval for Seriously Delinquency is from 6.47% to 6.72%.



Machine Learning

Custom Scorer

I created a Custom Scorer that calculates the profit / loss and can be reused for each algorithm.

Cost Factors:

- Average Loan Amount (defaulted to $50,000)
- Average Interest Rate (defaulted to 4%)
- Average Loan Length (defaulted to 2 years)

Calculations:

- If the scorer correctly predicts a non-defaulter then the profit is 4,000 (loan amount x rate x length).
- If the scorer incorrectly labels a defaulter as a non-defaulter then the loss is 50,000 (loan amount).
  - For simplicity, I assume the defaulter has never made a payment.
- If the scorer incorrectly labels a non-defaulter as a defaulter then the loss is 4,000 (profit we would have taken in).

Machine Learning Algorithms

I ran the entire dataset using Logistic Regression, k-NN, Random Forest, and SVM and also ran each for 3 different buckets of Monthly Incomes. The best overall results for Predicting Defaults is using the Logistic Regression using Balanced Weight. 65% of the Defaults were predicted but a large amount of non-defaults were mislabeled as defaults (24%). Similar results were returned when running the Monthly Income buckets using Logistic Regression with Balanced Weight.

I also performed the same above tests but on a custom scorer that can be modified to show the profit / loss using different algorithms. The custom scoring has 3 cost factors, Average Loan Amount, Average Interest Rate (revenue taken in), and Average Loan Length. These factors have been set to 50,000, 4 percent, and 2 years respectively.

Using these factors, if the scorer correctly predicts a non-defaulter then the profit is 4,000 (loan amount x rate x length). If the scorer incorrectly labels a defaulter as a non-defaulter then the loss is 50,000 (loan amount). If the scorer incorrectly labels a non-defaulter as a defaulter then the loss is 4,000 (profit we would have taken in).

The below results show the Profits taken in for each algorithm. The Logistic Regression (3 Monthly Income Buckets with Balanced Weight Class) had the best Profit predicted with over 110 million followed by Logistic Regression (Balanced Class Weight) and Random Forest Resampling (3 Monthly Income Buckets) all had Profits of 79+ million dollars. All others resulted in 70 million dollars or less. Do note that SVM was run initially but had to be removed due to computational power.

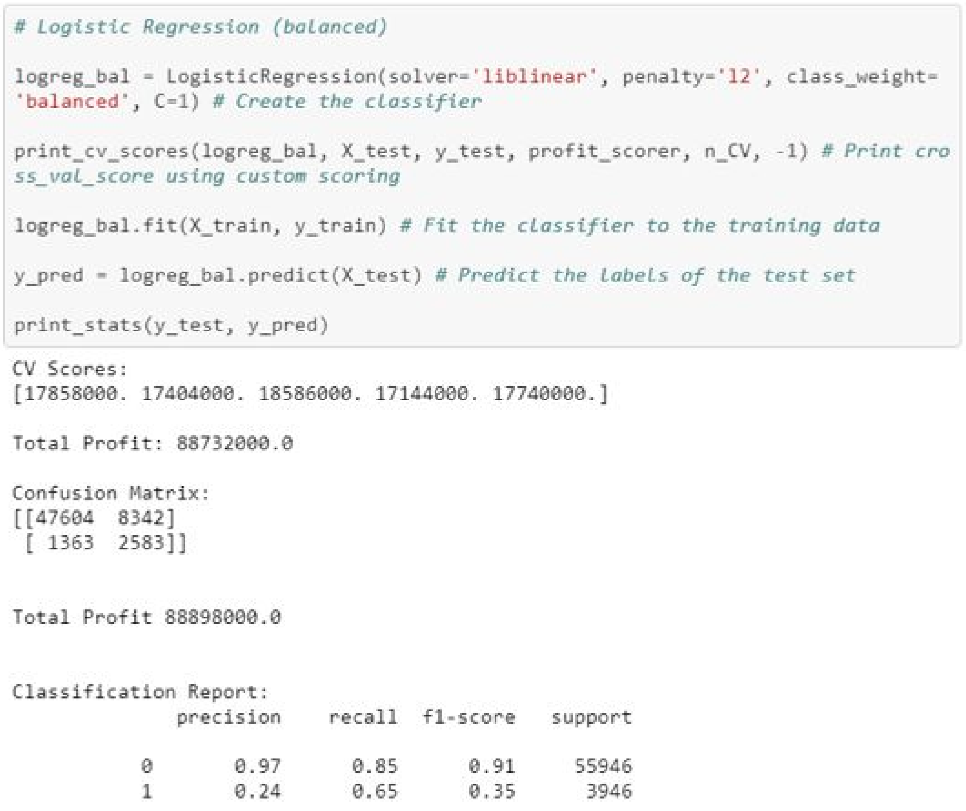
Results for each model tested:

Algorithm	**Profit**
Logistic Regression	50,930,000.00
Logistic Regression (tuned parameters)	56,720,000.00
Logistic Regression (Balanced Class Weight)	88,898,000.00
k-NN (Best Neighbors, Uniform Weight)	26,746,000.00
k-NN Resampling (Default Counts = Non-Default Counts)	20,728,000.00
Random Forest (Max Features = 10)	59,024,000.00
Random Forest (Balanced Weight Class)	52,662,000.00
Random Forest Resampling (Default Counts = Non-Default Counts)	69,020,000.00
Logistic Regression (3 Monthly Income Buckets)	113,208,000.00*

k-NN (3 Monthly Income Buckets)	42,288,000.00*
Random Forest (3 Monthly Income Buckets)	79,794,000.00*

* It is important to note that the 3 monthly income datasets had different train/test/split data. This is because I took the original dataset, split it up into the 3 mentioned buckets, and then did a train/test/split for each.

For metric purposes, I will state that the Logistic Regression (Balanced Class Weight) is the best model since splitting the dataset into 3 separate datasets by monthly income may have caused higher numbers.



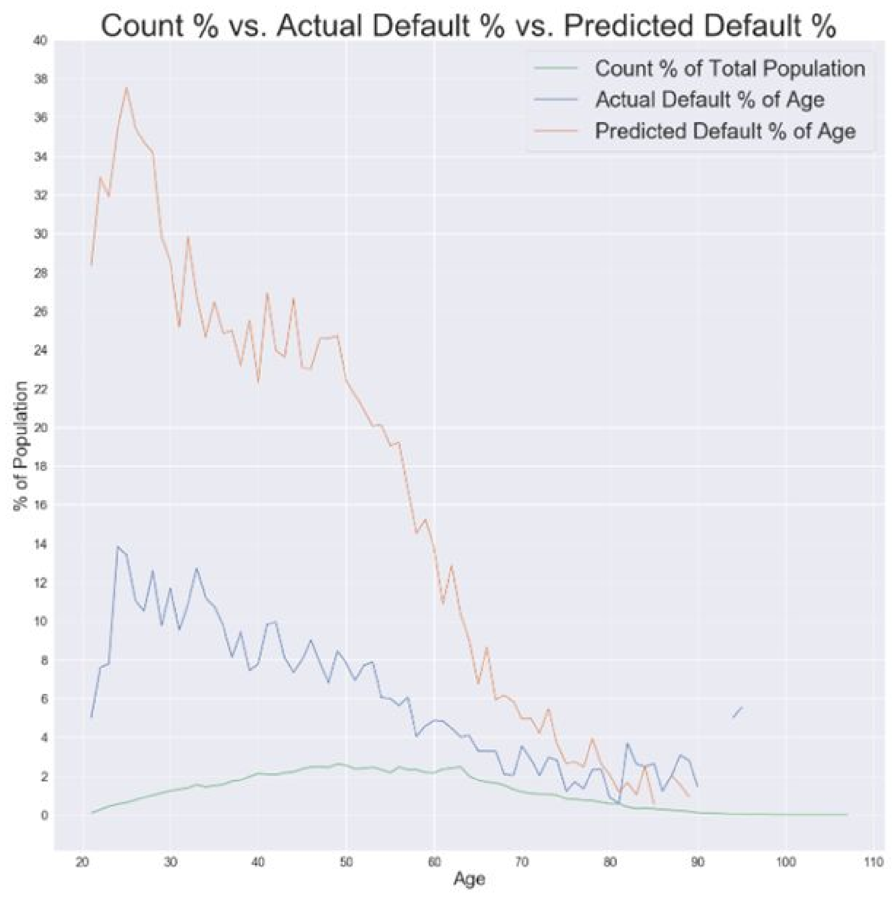
```
# Logistic Regression (balanced)

logreg_bal = LogisticRegression(solver='liblinear', penalty='l2', class_weight=
'balanced', C=1) # Create the classifier

print_cv_scores(logreg_bal, X_test, y_test, profit_scorer, n_CV, -1) # Print cro
ss_val_score using custom scoring

logreg_bal.fit(X_train, y_train) # Fit the classifier to the training data

y_pred = logreg_bal.predict(X_test) # Predict the labels of the test set

print_stats(y_test, y_pred)
```

```
CV Scores:
[17858000. 17404000. 18586000. 17144000. 17740000.]

Total Profit: 88732000.0

Confusion Matrix:
[[47604  8342]
 [ 1363  2583]]


Total Profit 88898000.0


Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.85      0.91     55946
           1       0.24      0.65      0.35      3946
```

Metrics by Subgroups (using Logistic Regression w/ Balanced Class Weight)

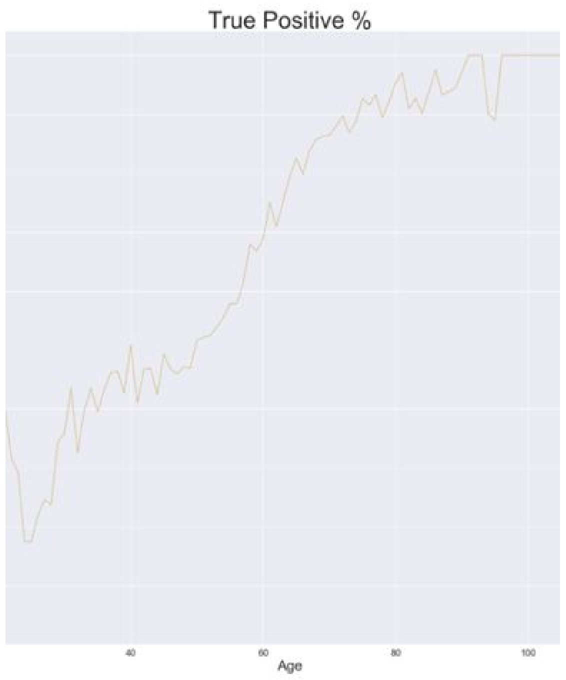
For an analysis of metrics by subgroups, I am using the Logistic Regression (Balanced Class Weight) model since I defined it as the best model. We can see from the below graph that the model is predicting a good portion of the younger ages to default so it appears the best single model is biased against younger people. For instance, if you are in your mid-20's, you have over a 35% chance of being predicted as a defaulter.

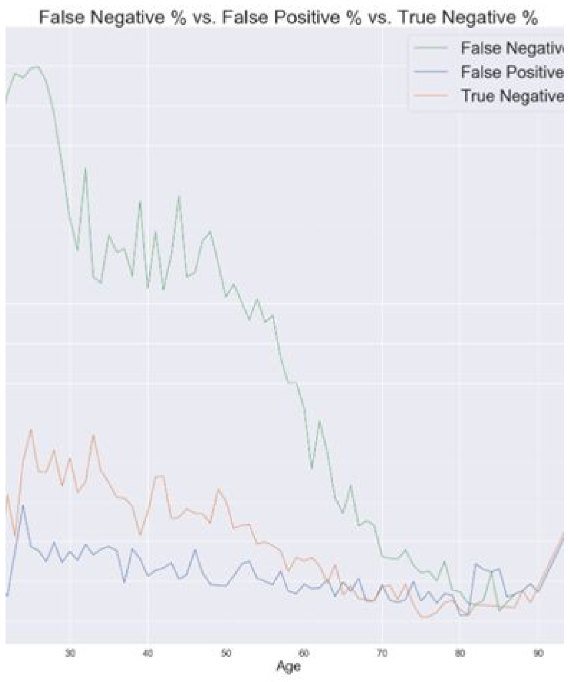
To maximize profits, predicting a portion of people to Default when they would have actually paid leads to more profit. Using our default numbers, it takes 12.5 non-defaulters to make up the losses from 1 defaulter.



The model had False Negatives for 15% of the total non-Defaulters with the highest % of False Negatives in the mid-20's (up to 29% of total population). This means that by being in your mid-20's, you have over a 25% of being declined a loan even though you would have paid back the loan in full.

So in summary, younger people are higher risk and have more uncertainty among outcomes.





Another analysis of the breakdown of Subgroups of those who were predicted to Default but would have paid, those who we did predict to default and did default, and those who we did not predict to default but did default. One interesting data point I see is that 22% of the people predicted to Default but had not previously gone 90 days past due. For those who predicted to Default and actually did Default, 53% of the people had previously gone 90 days past due. For those who were not predicted to Default but did Default, only 1.5% had previously gone Past Due 90 Days.

Basically the model is good at predicting for those who have been late in the past but not so good for those that have never been past due previously.

Recommendations

Here is a list of my recommendations:

- The Logistic Regression (Balanced Class Weight) model is the best overall model to use to get the most profit.
- It is best to target older people to give loans to. The younger someone is, the more likely they are going to be 60+ days past due.
- People in the 1000-2499 and 2500-4999 Monthly Income ranges are more likely to default so target individuals who make more or less than them.
- Predicting more people to default is better since it takes 12.5 non-defaulters to make up the losses from 1 defaulter.

-

Conclusion

The Logistic Regression with Balanced Class Weight is the best model Profit-wise. The model does seem to be biased against younger people though. In the end, the final decision on whether to give someone a loan or not should not be solely based on the model.

Possible Next Steps:

- If time permitted, I would add additional algorithms (ex. xgboost).
- Find additional data source(s) to combine with the dataset used.
- Perform more analysis on biases.