PROBABILITY OF DEFAULTING

SPRINGBOARD CAPSTONE 1 FINAL

BY: BROCK NOSBISCH

OVERVIEW

- The Problem
- Dataset
- Exploratory Data Analysis
- Machine Learning
- Metrics by Subgroups
- Conclusions

THE PROBLEM

- 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.

DATASET

- The original dataset comes from a Kaggle competition found here.
- 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
```

DATASET

- Additional features like Age Buckets, Monthly Income Buckets, Past Due Flags, etc. were added to the dataset.
- Some data also needed cleaned. Monthly Income and Number of Dependents had NULL values and the Past Due fields values had records with 96 and 98 values.

```
<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
```

DATASET (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.)

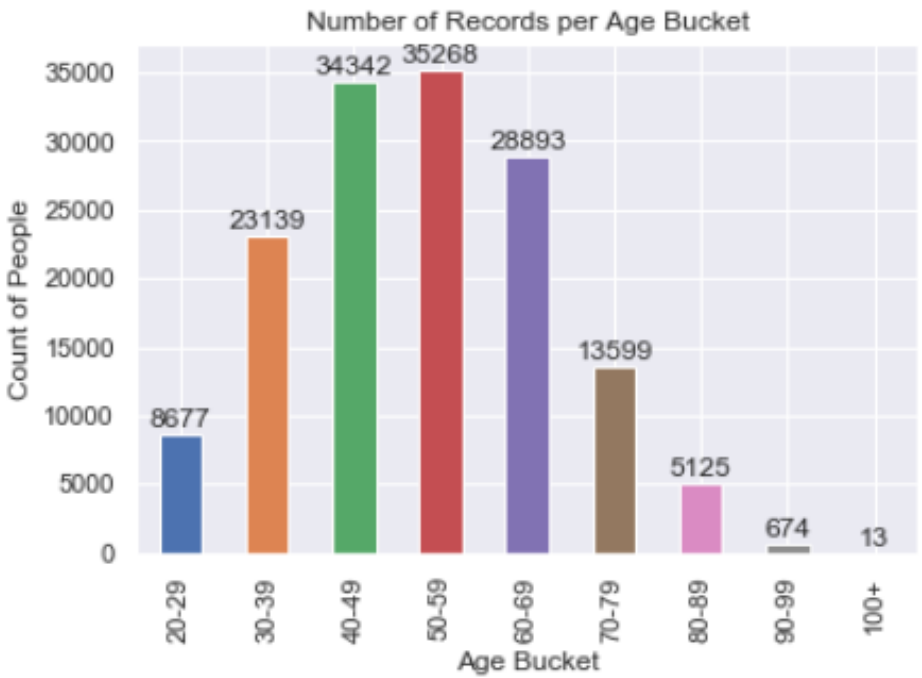
EXPLORATORY DATA ANALYSIS

Several observations of interest were found during EDA. Here are the high-level results.

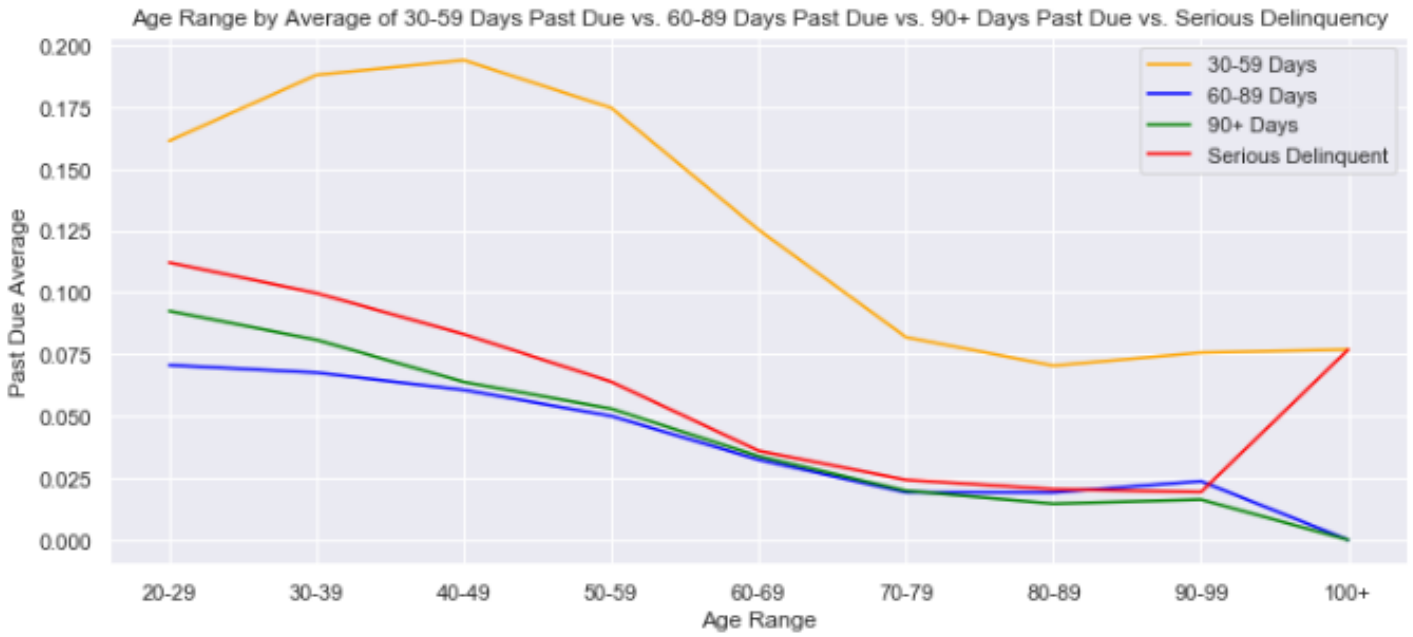
- The younger in age you are, the higher the probability that you will be 60-89 Days Past Due, 90+ Days Past Due, and/or Serious Delinquent.
- The $1000-$2499 and $2500-$4999 Monthly Income Ranges are most likely to be Past Due when compared to other income ranges.
- The Debt Ratio quantiles show that 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.
- 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))

EXPLORATORY DATA ANALYSIS (AGE)

Count of People per Age Bucket

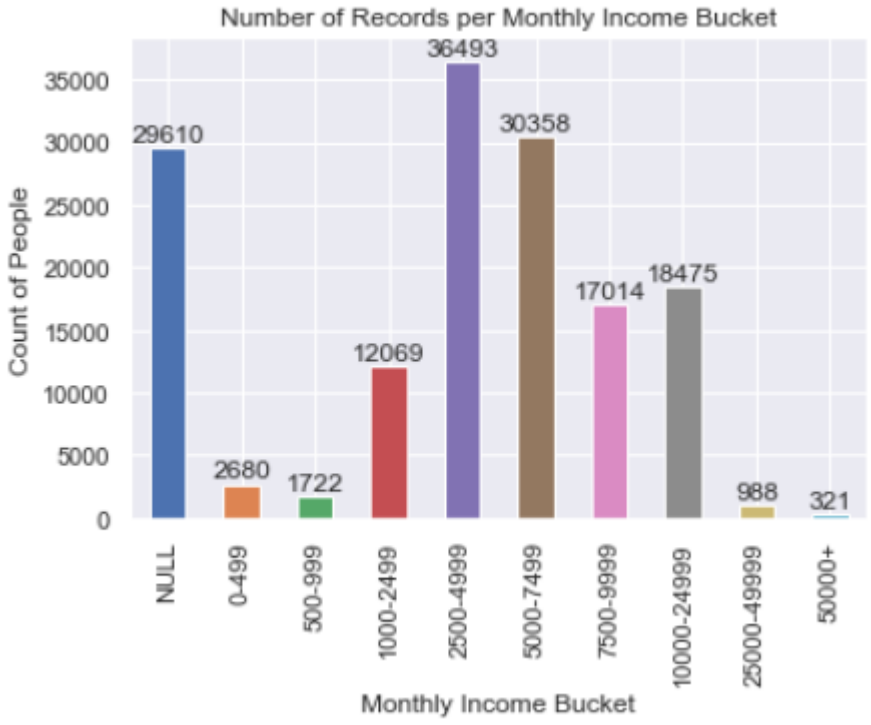


Averages of each Age Range to be Past Due 30-59 Days, 60-89 Days, 90+ Days, and Serious Delinquent

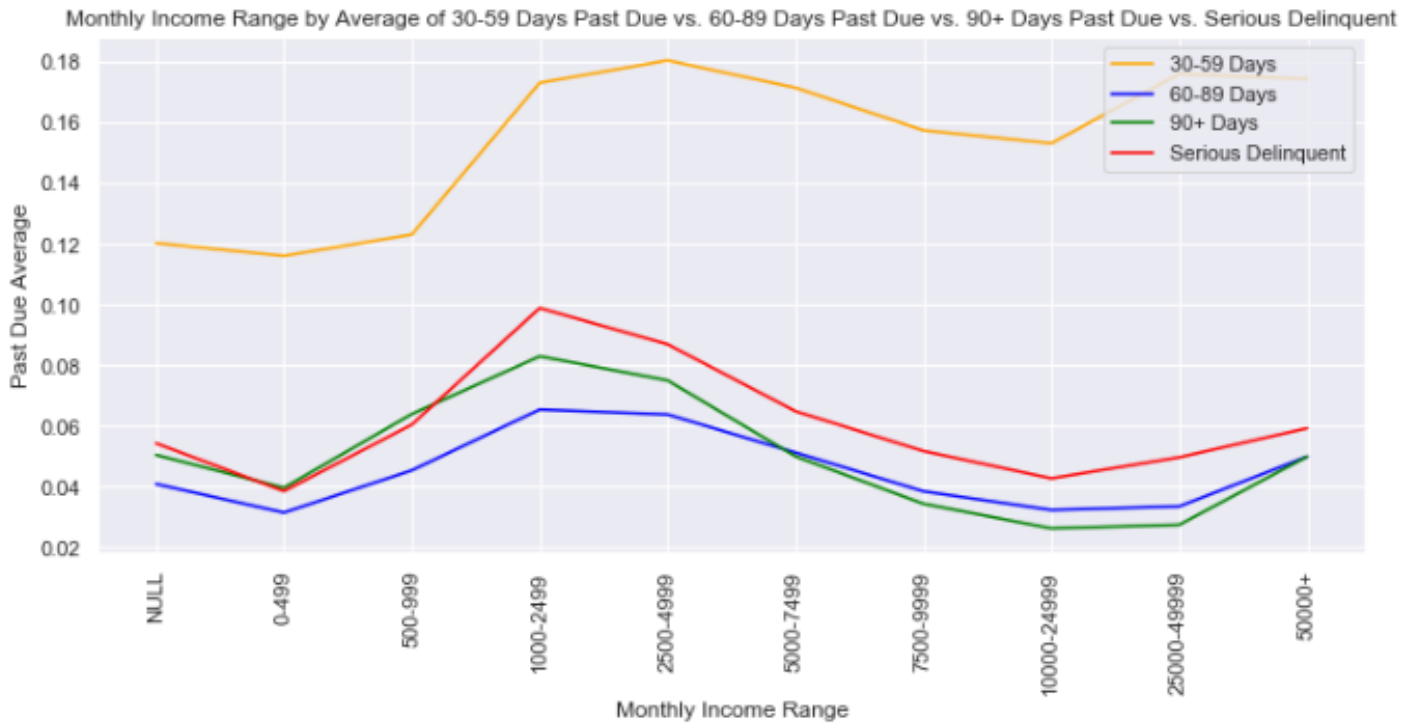


EXPLORATORY DATA ANALYSIS (MONTHLY INCOME)

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



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 (SUMMARY)

- The 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) both having profits of $79+ millions.
- All others resulted in $70 million or less. Do note that SVM was run initially but had to be removed due to computational power.

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.

MACHINE LEARNING (LOGISTIC REGRESSION)

- For Logistic Regression, I ran 3 tests.
  - Default parameters, CV=10
  - Default parameters, tuned parameters
  - Balanced Class Weight, Ridge Regression Penalty (L2), CV=10

```
Confusion Matrix:
[[55608   338]
 [ 3403   543]]


Total Profit 50930000.0


Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.99      0.97     55946
           1       0.62      0.14      0.22      3946
```

```
Confusion Matrix:
[[55511   435]
 [ 3273   673]]


Total Profit 56654000.0


Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.99      0.97     55946
           1       0.61      0.17      0.27      3946
```

```
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
```

MACHINE LEARNING (LOGISTIC REGRESSION - BALANCED)

```
# 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
```

- As you can see from the previous slide, the best performing model was the Balanced Class Weight, Ridge Regression Penalty (L2), CV=10.
- I consider this model to be the best out of all the models (excluding the 3 monthly income bucket models since they had different datasets than the rest).
- Total Profit: $88,898,000

MACHINE LEARNING (K-NEAREST NEIGHBORS)

- For k-NN, I ran 2 tests. The results for each were not subpar.
  - Find the best k-NN parameters using Grid Search and use it.
  - Upsample the data so that out of the training set, 50% have been serious delinquent and 50% have not.

```
Confusion Matrix:
[[51139  4807]
 [ 3292   654]]


Total Profit 20728000.0


Classification Report:
              precision    recall  f1-score   s

           0       0.94      0.91      0.93
           1       0.12      0.17      0.14
```

```
Confusion Matrix:
[[55885    61]
 [ 3931    15]]


Total Profit 26746000.0


Classification Report:
              precision    recall  f1-score   su

           0       0.93      1.00      0.97
           1       0.20      0.00      0.01
```

MACHINE LEARNING (K-NEAREST NEIGHBORS – BEST PARAMS)

- For the k-NN (Best Neighbors, Uniform Weight) run, the Precision was 93% and Recall was ~100% but that is because the model only predicted 61 total defaults. Only 15 of those 61 actually defaulted.
- Profit was only ~$26 million because of this.

```
knn = KNeighborsClassifier(n_neighbors=knn_neighbors['n_neighbors'], weights='uniform')

print_cv_scores(knn, X_test, y_test, profit_scorer, n_CV, -1) # Print cross_val_score using custom scoring

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

print('Prediction: {}'.format(y_pred))
print('k-NN Score: {}'.format(knn.score(X_test, y_test)))

print_stats(y_test, y_pred)
```

```
k-NN Score: 0.9333466907099446
Confusion Matrix:
[[55885    61]
 [ 3931    15]]


Total Profit 26746000.0


Classification Report:
              precision    recall  f1-score   suppo

           0       0.93      1.00      0.97     559
           1       0.20      0.00      0.01      39
```

MACHINE LEARNING (RANDOM FOREST)

- For Random Forest, I ran 3 tests. The results were pretty decent with profit in the range of 52 – 69 million dollars.
  - Max Features = 10
  - Balanced Weight Class
  - Upsample

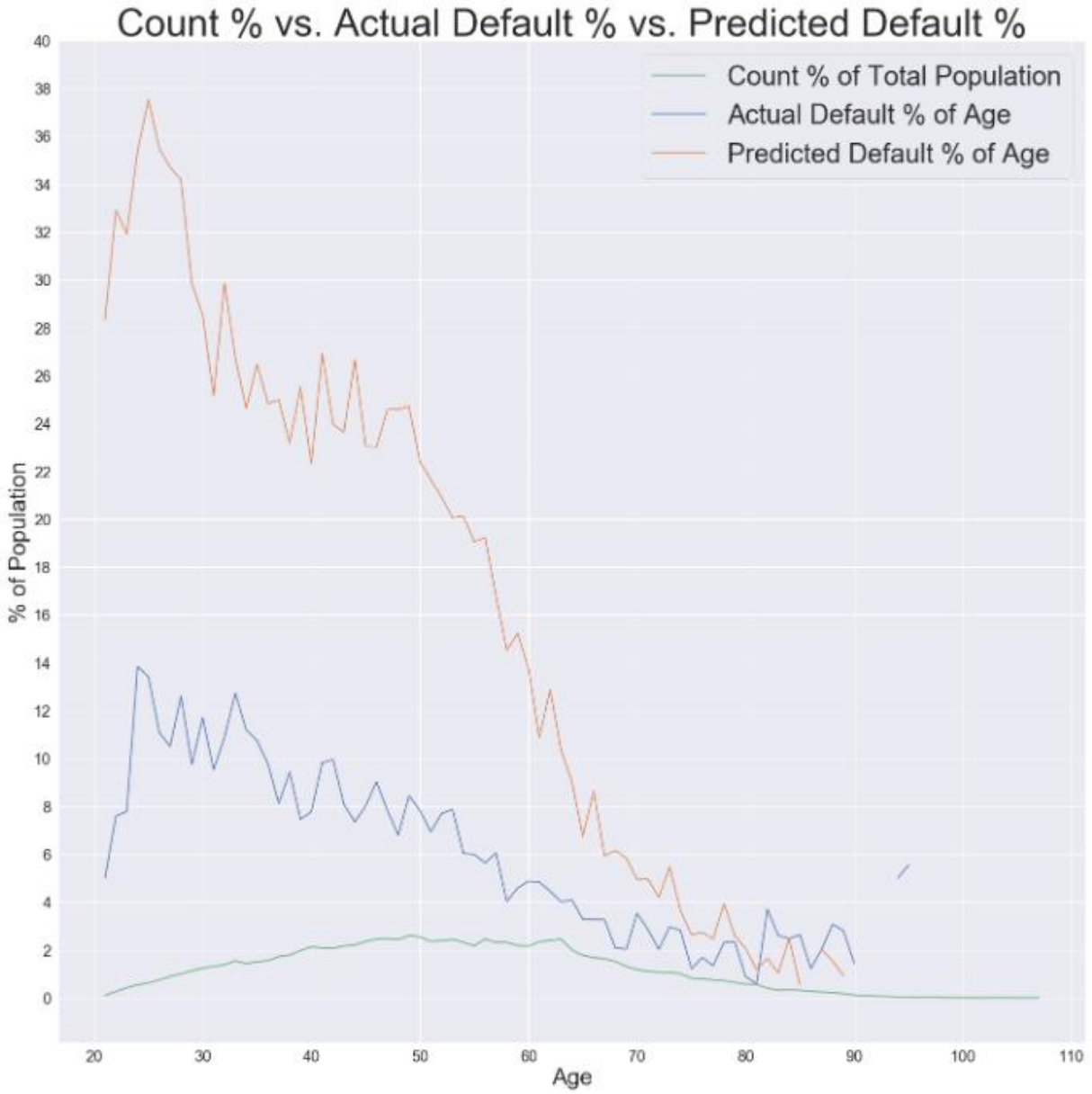
```
Confusion Matrix:
[[55276   670]
 [ 3188   758]]


Total Profit 59024000.0


Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.99      0.97     55946
           1       0.53      0.19      0.28      3946
```

```
Confusion Matrix:
[[54763  1183]
 [ 2906  1040]]


Total Profit 69020000.0


Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.98      0.96     55946
           1       0.47      0.26      0.34      3946
```

```
Confusion Matrix:
[[55462   484]
 [ 3345   601]]


Total Profit 52662000.0


Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.99      0.97     55946
           1       0.55      0.15      0.24      3946
```

METRICS BY SUBGROUPS (LOGISTIC REGRESSION – BALANCED)

- For analysis of metrics by subgroups, I am using the Logistic Regression (Balanced Class Weight) model since it is being defined as the best model.
- This graph is showing the Count % of Total Population by Age, Actual Default % by Age, and Predicted Default % by Age.



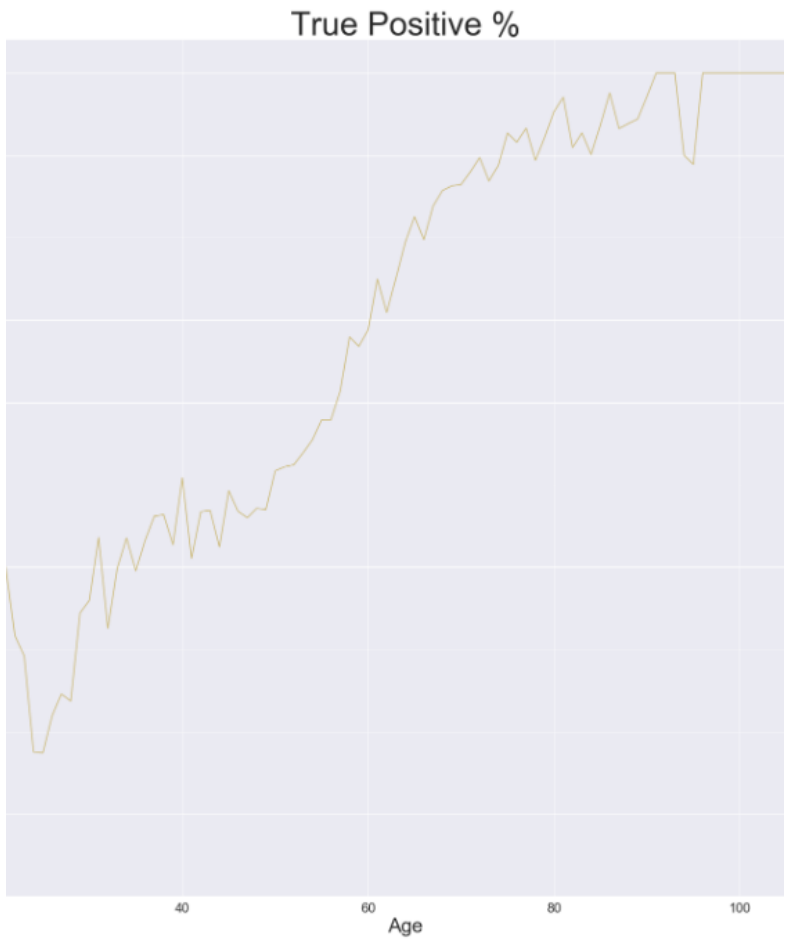
METRICS BY SUBGROUPS (LOGISTIC REGRESSION – BALANCED)

- 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.

METRICS BY SUBGROUPS (LOGISTIC REGRESSION – BALANCED)

- 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.



METRICS BY SUBGROUPS (LOGISTIC REGRESSION – BALANCED)

- 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.

CONCLUSIONS

- Logistic Regression with Balanced Class Weight is the best model Profit-wise.
- The model does seem to be biased against younger people.
- 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.