Springboard Data Science Career Track | Nikki Seegars
Capstone Project I: Mortgage Foreclosure Prediction

Introduction

The American dream is the promise of the opportunity to achieve prosperity is available to all regardless of origin or social class. For many people this dream invariably includes home ownership. Home ownership has been between 62%-69% since 1965 with the 3rd quarter of 2019 at 64.8%[1]. Approximately 86% of all home buyers finance their homes[2]. Overall home foreclosure rates are pretty low, partially due to the current precautions that mortgage lenders take when approving loans. However, there is an opportunity to reduce the number of foreclosures and increase profits for the lender.

Problem Statement

The need persists to continually improve upon the mortgage foreclosure prediction models that lenders use when approving mortgage loan applications. This capstone project will explore machine learning algorithms that will make an incremental improvement in determining whether or not a potential loan is likely to foreclose or not.

Data Source

Fannie Mae, formally known as the Federal National Mortgage Association, is a US government-sponsored enterprise which was founded in 1938 with the intent to help stimulate the economy during the Great Depression. Fannie Mae purchases mortgages in the secondary market and guarantees the loans. Fannie Mae owns a significant amount of all US mortgages and has made available a subset of the loan records going back to the year 2000 on their website. There are two categories of mortgages captured in the data set: (1) single-family, 30 year and less, fixed rate, full documentation loans and (2) Home Affordable Refinance Program (HARP) 30 year fixed loans. For this analysis, only data from the first category was used.

The mortgage data is captured in .txt files, two files exist for each quarter of the year. The acquisition files contain loan identifying data and have 25 columns: Loan Identifier, Origination Channel, Seller Name, Original Interest Rate, Original Unpaid Balance, Original Loan Term, Origination Date, First Payment Date, Original Loan-to-Value (LTV), Original Combined Loan-to-Value (CLTV), Number of Borrowers, Original Debt to Income Ratio, Borrower Credit Score at Origination, First Time Home Buyer Indicator, Loan Purpose, Property Type, Number of Units, Occupancy Type, Property State, Zip Code Short, Primary Mortgage Insurance Percent, Product Type, Co-Borrower Credit Score at Origination, Mortgage Insurance Type, Relocation Mortgage Indicator. The performance files contain monthly data regarding each loan and have 31 columns: Loan Identifier, Monthly Reporting Period, Servicer Name, Current

[1] Federal Reserve Bank of Saint Louis, “Homeownership Rates for United States”, October 29, 2019, https://fred.stlouisfed.org/series/RHORUSQ156N
[2] National Association of Realtors, “2019 Profile of Home Buyers and Sellers”, November 2019, https://www.nar.realtor/sites/default/files/documents/2019-profile-of-home-buyers-and-sellers-highlights-11-07-2019.pdf

Interest Rate, Current Actual Unpaid Balance, Loan Age, Remaining Months to Legal Maturity, Adjusted Months to Maturity, Maturity Date, Metropolitan Statistical Area (MSA), Current Loan Delinquency Status, Modification Flag, Zero Balance Code, Zero Balance Effective Date, Last Paid Installation Date, Foreclosure Date, Disposition Date, Foreclosure Costs, Property Preservation and Repair Costs, Asset Recovery Costs, Miscellaneous Holding Expenses and Credits, Associated Taxes for Holding Property, Net Sale Proceeds, Credit Enhancement Proceeds, Repurchase Make Whole Proceeds, Other Foreclosure Proceeds, Non Interest Bearing Unpaid Balance, Principal Forgiveness Amount, Repurchase Make Whole Proceeds Flag, Foreclosure Principal Write-Off Amount, Servicing Activity Indicator. One quarter's data, loans originating in the 4th quarter of 2008, was used to build and test the predictive model.

Data Wrangling

Data wrangling is the process of taking raw data that may be messy and diverse and converting it to a format that supports usage for data analysis. Neither of the .txt files contain a header column so the headers were assigned using Python and the pandas.read_csv() function. The acquisition file has 342,129 records and the performance file has 13,788,511 records. All of the rows from the performance .txt file with Zero Balance Code with NaN (not a number) values were dropped since these represent either months prior to the month of the loan payoff and/or a loan that is currently active. The Zero Balance Code indicates why the loan's Current Actual Unpaid Balance has a $0 value. These are all loans which are no longer active and have had their balances paid off. Ninety-three percent of all Fannie Mae mortgages originating in the 4th quarter of 2008 have had their balances paid off.

Zero Balance Code	Description
1	Prepaid or Matured
2	Third Party Sale, foreclosure to a party not the lender or former owner
3	Short Sale, net proceeds which fall short of the debts secured by loan
6	Repurchased, reversal of loan to original lender
9	Deed-in-Lieu, lender real estate owned (REO) foreclosure
15	Note Sale, sale of mortgage to another party
16	Reperforming Loan Sale, sale of loans previously behind 90 days or more

Table 1. Zero Balance Codes

The two .txt files were merged into one pandas DataFrame on the Loan Identifier columns.

Each of the files has missing data in some of the columns however only a few of those columns with missing data are critical to the development of the predictive model. To account for the 25 Number of Borrowers returning NaN values the Borrower Credit Score at Origination and Co-Borrower Credit Score at Origination columns were referenced to change the value to 2 if a value existed for the Co-Borrower Credit Score at Origination or 1 if the Borrower Credit Score at Origination was not null. The rows were dropped from the DataFrame for the 9 records with Number of Borrowers equal to 1 but had a value in Co-Borrower Credit Score and the 26,738 records with Number of Borrowers greater than 1 and Co-Borrower Credit Score as NaN. For the 2 records with missing Original Combined Loan-to-Value (CLTV) the Original Loan-to-Value (LTV) was copied into the Original Combined Loan-to-Value (CLTV) column. The 6,549 records that had NaN values for the Original Debt to Income Ratio were filled in with the mean value from all of the records containing data. Four columns with categorical data were converted into numeric values. The First Time Home Buyer Indicator column had all “Y” values converted to 1 and all other values converted to 0. The categories for Property Type were changed to numeric values as follows: 1 = “SF” (single family), 2 = “PU” (planned unit development), 3 = “CO” (condo), 4 = “CP” (co-op), 5 = “MH” (multi house). The Loan Purchase column had three categorical values converted to the following numeric values: 1 = “P” (purchase), 2 = “C” (cash-out refinance), 3 = “R” (cash-out no refinance). The Occupancy Type column had three categorical values converted to the following numeric values: 1 = “P” (principal), 2 = “I” (investment), 3 = “S” (second).

Exploratory Data Analysis

The total number of closed loans in the data set is 292,705 and the total number of open loans is 22,291. For this analysis only the closed loans needed to be evaluated since their foreclosure status is known.

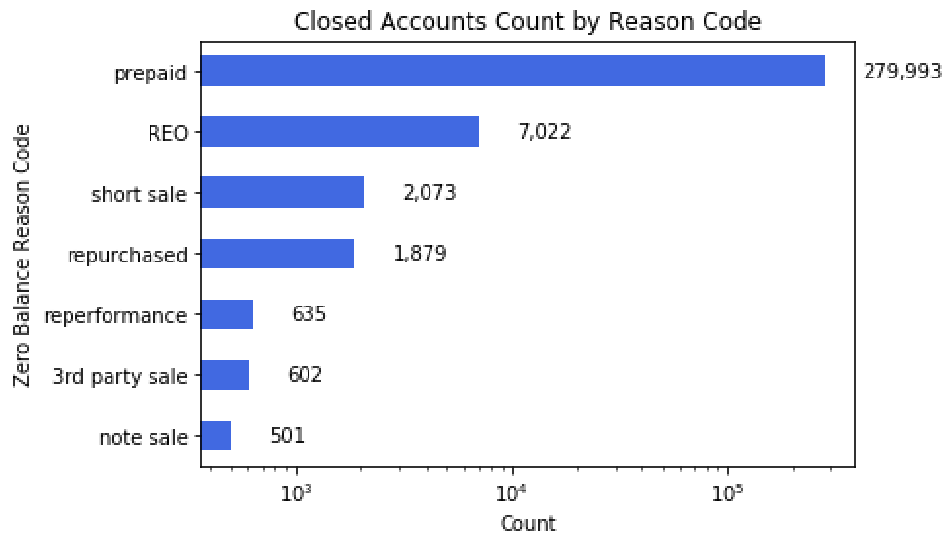


Chart 1. Closed Mortgage Accounts Count by Zero Balance Code.

Whenever a lender decides whether to extend credit to a customer one of the key metrics evaluated is the applicant’s credit score. In the US, the FICO (Fair, Isacc, and Company) scoring model with 300 as lowest and 850 as the highest score, is often used to measure individuals creditworthiness. The score is based on several factors including payment history, debt burden, length of credit history, types of credit used, and recent credit inquiries on consumer’s credit report.

The vast majority of all loans were prepaid by borrowers with very few loans having closure due to REO foreclosure. With a higher number of records, the prepaid accounts have a higher median, 763, and wider spread than any of the other Zero Balance Code categories. REO foreclosures accounts have a median a bit lower, 710, than those of the prepaid accounts. Examining the data closely in

Chart 2 there is a range overlap that exists between all of the Zero Balance Codes which indicates there are other factors to consider when predicting which applicants would likely result in a foreclosure.

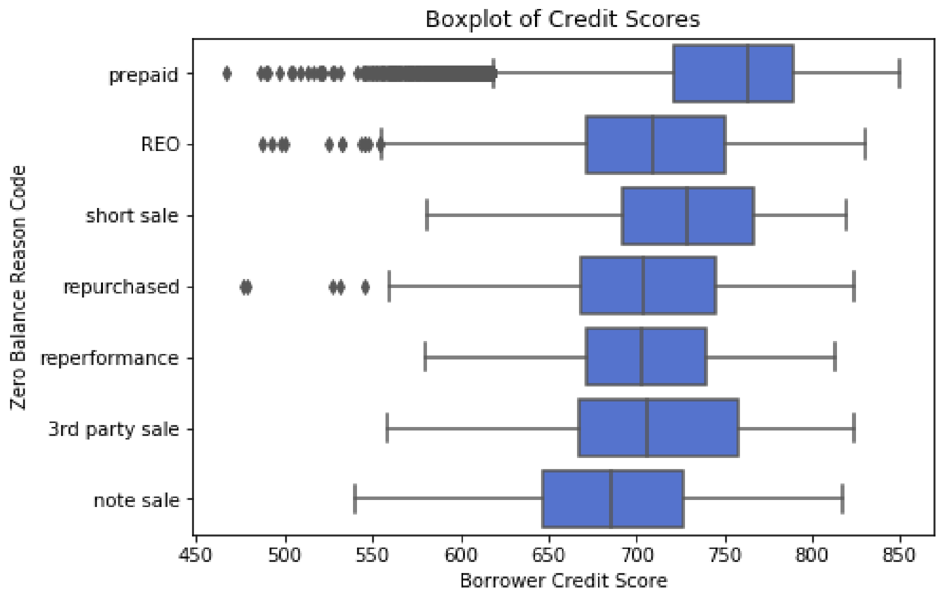


Chart 2. Credit Scores Boxplot by Zero Balance Code.

Although most of the loans have a 30 year fixed rate, most of the accounts were closed within 2,000 days (~ 7.5 years) from the loan origination date. There may be many reasons why a loan's lifespan is less than its term including refinancing by the customer, home sales, foreclosure, or short sale. The chosen quarter's data set makes a good choice since the origination date is ~ 10 years and it will give insight into how mortgages perform over the long term.

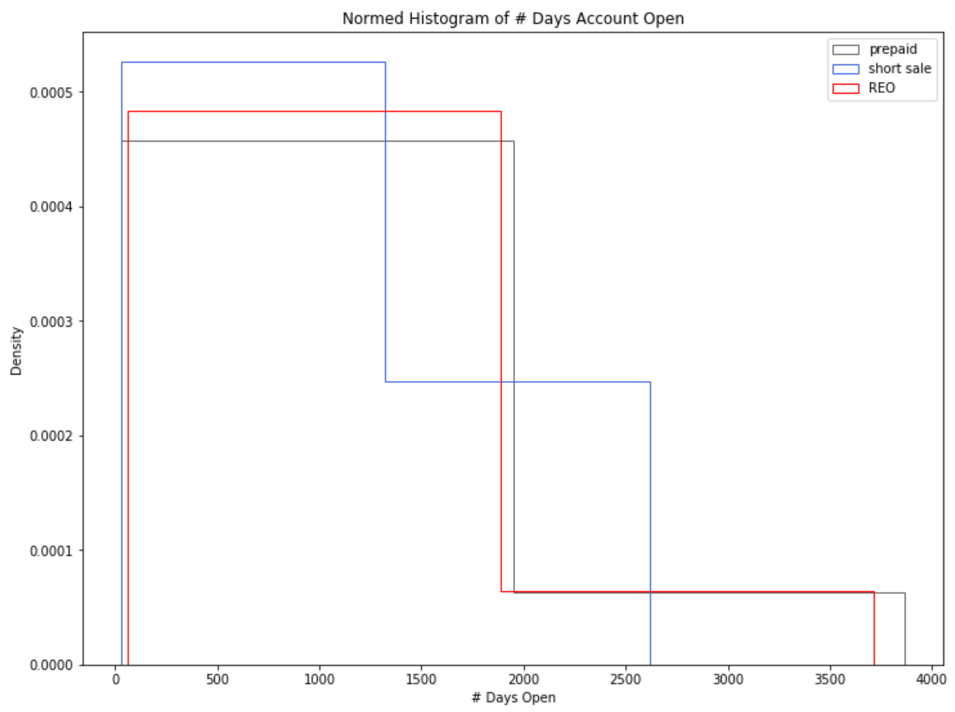


Chart 3. Number of Days Mortgage Account Open by Zero Balance Code.

Inferential Statistics

Inferential statistics use random sampling from a population to make deductions about a data set. Inferences about the larger population are made by examining the relationships between variables within a smaller sample. In statistics, hypothesis testing makes a statement or assumption about a data set parameter. The null hypothesis is a comparison of the parameters that is assumed to be true. The test confirms if the null hypothesis should be either accepted or rejected. The alternate hypothesis is the opposite of the null hypothesis and is what is assumed to be untrue. These two hypotheses are mutually exclusive, only one can be true and the other false. The null hypothesis explored for this data set is that there is not a difference between credit scores of REO accounts and prepaid accounts. This is a measure of the difference between the two means and since the measurement is something other than the mean that makes it a good candidate for bootstrap sampling.

Bootstrapping is a resampling statistical technique that consists of taking many samples with replacement. Other inferential statistics methods such as the z and t statistic is used to estimate the population mean where bootstrap is more flexible and can measure a multitude of summary statistics.

$\overline{m_P}$ = sample mean prepaid loans

$\overline{m_R}$ = sample mean of REO loans

$\overline{x_P}$ = mean of first half of permuted sample

$\overline{x_R}$ = mean of second half of permuted sample

n = number of samples

Null Hypothesis: $\overline{m_P} - \overline{m_R} = \overline{x_P} - \overline{x_R}$

Alternate Hypothesis: $\overline{m_P}$ - $\overline{m_R}$ $\neq$ $\overline{x_P}$ - $\overline{x_R}$

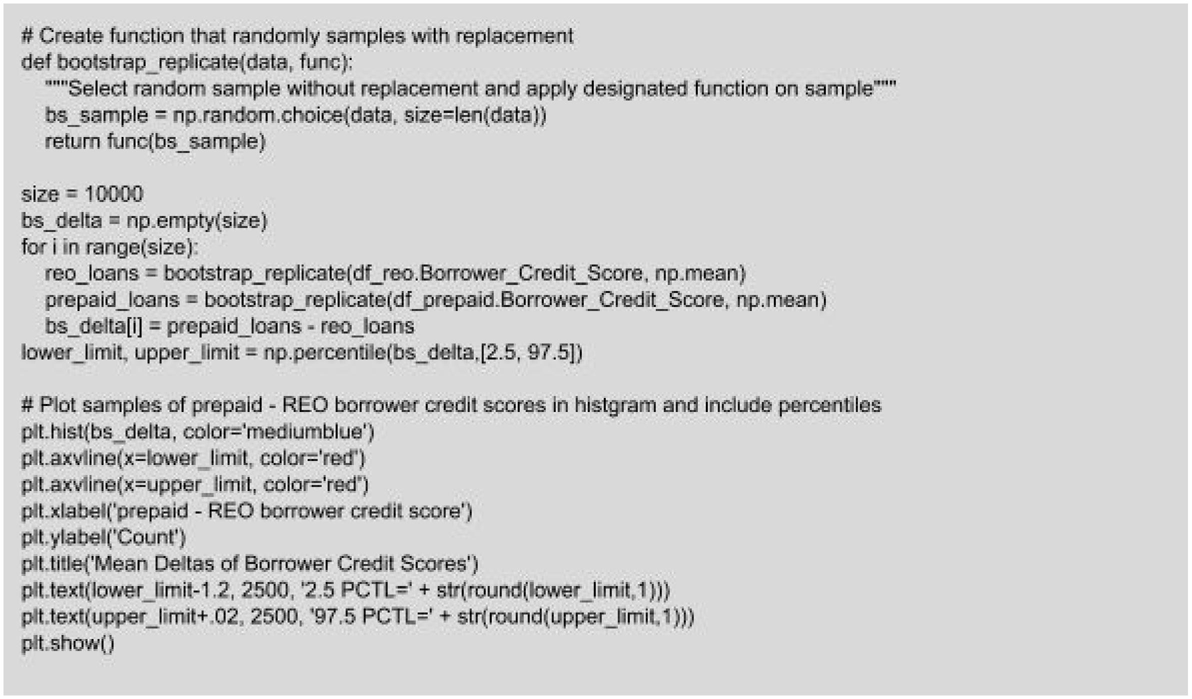
The p-value is used in hypothesis testing to determine whether to accept or reject the null hypothesis. The p-value represents the probability of obtaining test results at least as extreme as the results observed in the test.

$$p - value \; = \; \sum \big[\, (\overline{m_P} - \overline{m_R}) >= (\overline{x_P} - \overline{x_R}) \, \big] \; / \; n$$

The smaller the p-value, the greater the significance the comparison is and indicates that the null hypothesis does not adequately explain the observation. A level of significance of 0.05 was used to evaluate the null hypothesis.

Several steps were involved in creating calculating the p-value:

1. Took 10,000 samples of the difference between the means of prepaid & REO loan through random sampling with replacement.
2. Determined the 95% confidence interval.



```
# Create function that randomly samples with replacement
def bootstrap_replicate(data, func):
    """Select random sample without replacement and apply designated function on sample"""
    bs_sample = np.random.choice(data, size=len(data))
    return func(bs_sample)

size = 10000
bs_delta = np.empty(size)
for i in range(size):
    reo_loans = bootstrap_replicate(df_reo.Borrower_Credit_Score, np.mean)
    prepaid_loans = bootstrap_replicate(df_prepaid.Borrower_Credit_Score, np.mean)
    bs_delta[i] = prepaid_loans - reo_loans
lower_limit, upper_limit = np.percentile(bs_delta,[2.5, 97.5])

# Plot samples of prepaid - REO borrower credit scores in histgram and include percentiles
plt.hist(bs_delta, color='mediumblue')
plt.axvline(x=lower_limit, color='red')
plt.axvline(x=upper_limit, color='red')
plt.xlabel('prepaid - REO borrower credit score')
plt.ylabel('Count')
plt.title('Mean Deltas of Borrower Credit Scores')
plt.text(lower_limit-1.2, 2500, '2.5 PCTL=' + str(round(lower_limit,1)))
plt.text(upper_limit+.02, 2500, '97.5 PCTL=' + str(round(upper_limit,1)))
plt.show()
```

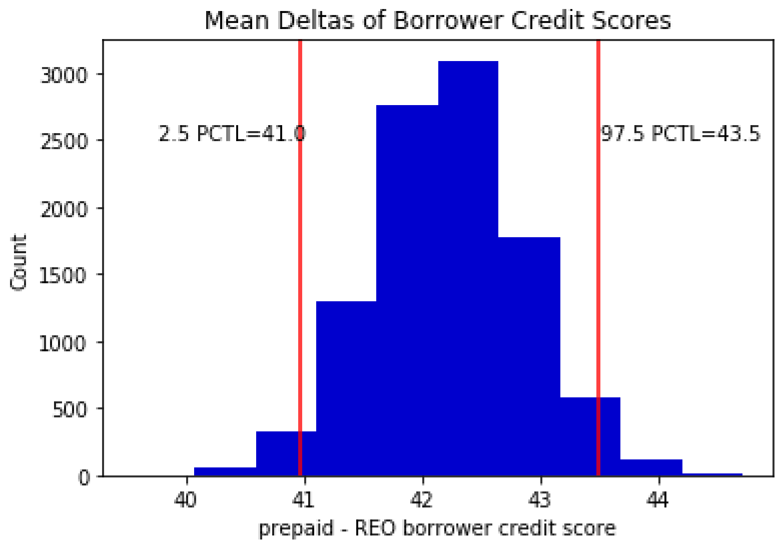


Figure 3. Distribution of the Difference Between Prepaid and REO Borrower Credit Scores

3. Randomly permuted (changed order) and split data into two groups.

```
# Combine and permute two datasets
def permutation_sample(data1, data2, func):
    """Generate a permutation sample from the two datasets."""

    # Concatenate the data sets: data
    data = np.concatenate((data1, data2))

    # Permute the concatenated array: permuted_data
    permuted_data = np.random.permutation(data)

    # Split the permuted array into two: perm_sample_1, perm_sample_2
    perm_sample_1 = permuted_data[:len(data1)]
    perm_sample_2 = permuted_data[len(data1):]

    diff = func(perm_sample_1, perm_sample_2)

    return diff

# Take difference of means between two datasets
def diff_of_mean(perm_sample_1, perm_sample_2):
    """Difference of mean of two arrays."""

    # The difference of means of data_1, data_2: diff
      diff = np.mean(perm_sample_1) - np.mean(perm_sample_2)

     return diff

# Loop through and find the delta between two simulated data sets of the same sample size
perm_replicates = np.empty(size)
reo_data = np.array(df_reo.Borrower_Credit_Score)

for i in range(size):
    perm_replicates[i] = permutation_sample(
        reo_data, np.random.choice(np.array(df_prepaid.Borrower_Credit_Score),
                            len(reo_data)), diff_of_mean)
print(perm_replicates)
```

4. Calculated the difference of the means of known samples and the permuted samples.

```
# Determine p-value for null hypothesis
p = np.sum(abs(perm_replicates) >= np.mean(bs_delta)) / len(perm_replicates)
```

The p-value was calculated as 0 indicating that there is a difference between the borrowers' credit scores of accounts from prepaid versus REO accounts. The null hypothesis is rejected since 0 < 0.05. It's fair to say that there is significance in the credit scores of the two groups and thus credit scores likely factor into the likelihood of a foreclosure.

Machine Learning

Machine learning is a branch of artificial intelligence that uses algorithms to learn without explicitly being programmed. There are three types of machine learning categories: supervised, unsupervised, and reinforcement learning. Supervised learning is when labels exist and the output is already known. In order to create a model, the data is split into two groups, training and test. The model is trained on the majority of the records (i.e. 75%) and the remaining data is used to test the model (i.e. 25%). Unsupervised learning happens without labeled data, resulting in an independent learning process. Without any known output values, techniques such as discovering patterns are used to label the data. Reinforcement learning involves feedback and past experiences. Each time a new step is taken, feedback from the previous step is used to learn and predict the next step making reinforcement learning an iterative process.

This data set has labeled data so a supervised learning model was appropriate. The two types of supervised learning categories are regression and classification. Regression methods are used when the data is numeric and is used to predict a continuous value. Classification methods deal with categorical data used to predict distinct classes such as likely to foreclose and unlikely to foreclose. There are many classification models, however three of the more common ones were evaluated for this problem: k nearest neighbors, logistic regression, and random forest. K nearest neighbors predict the class of new data by using its proximity to the labeled data. Logistic regression uses the logistic function, an S-shaped curve that can take any value between 0 and 1, for binary classification. Random forest is an ensemble method that aggregates multiple decision trees (makes classification based on a series of rules from the features of the data set) and uses the majority vote to determine the class.

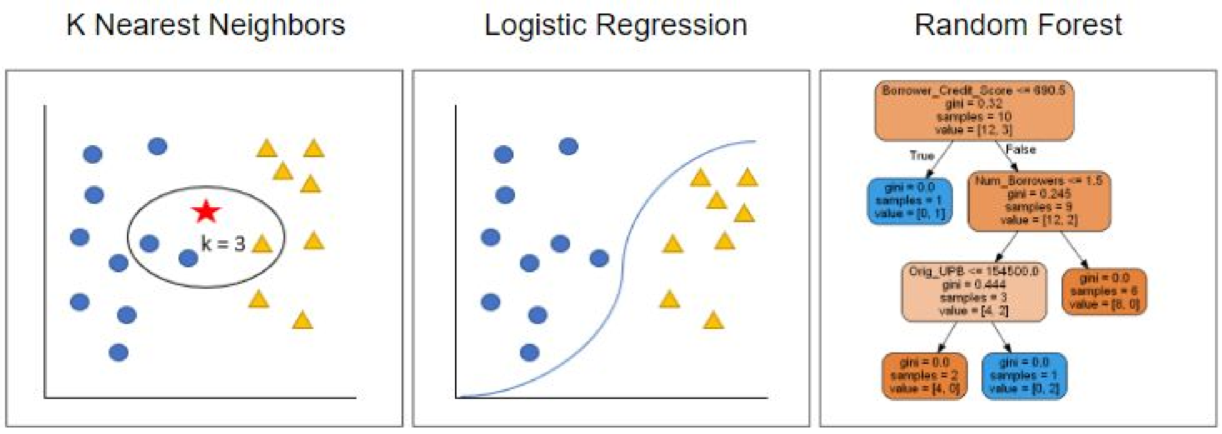
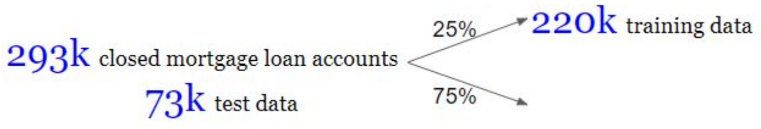


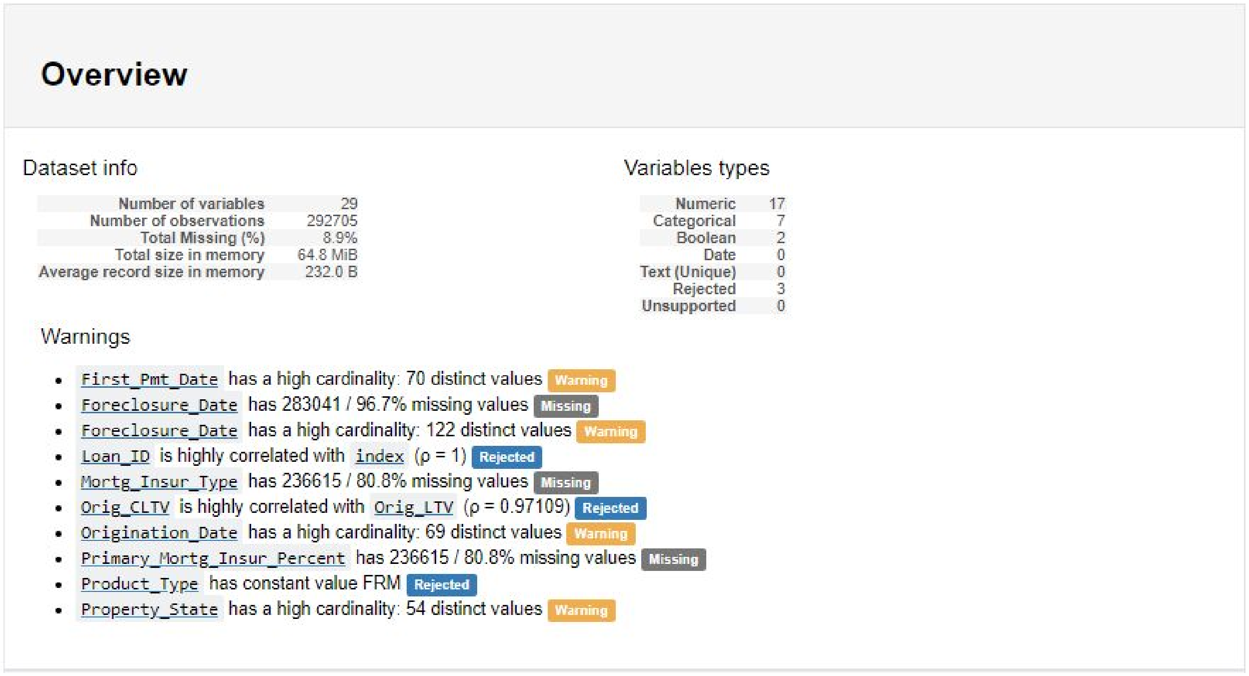
Figure 4. Classification algorithms explored,

The records were split into two subsets, training (75%) and test (25%). The training subset was used to learn the data and make predictions on the test data set.



The acquisition DataFrames was combined with the columns Loan Identifier, Zero Balance Code and Foreclosure Date from the performance DataFrame. These three columns are the only ones that have pertinent information that would be useful in building a prediction model. A classification column was added onto the new merged data frame with a 1 indicating a foreclosure and a 0 indicating not a foreclosure based on if a value was in the Foreclosure Date column.

After reviewing the pandas_profiling.ProfileReport(), several other columns were removed from the relevant feature set based on a high number of distinct, high correlation to another feature or having a constant value.



Overview

Dataset info

Number of variables	29
Number of observations	292705
Total Missing (%)	8.9%
Total size in memory	64.8 MiB
Average record size in memory	232.0 B

Variables types

Numeric	17
Categorical	7
Boolean	2
Date	0
Text (Unique)	0
Rejected	3
Unsupported	0

Warnings

- First_Pmt_Date has a high cardinality: 70 distinct values Warning
- Foreclosure_Date has 283041 / 96.7% missing values Missing
- Foreclosure_Date has a high cardinality: 122 distinct values Warning
- Loan_ID is highly correlated with index (ρ = 1) Rejected
- Mortg_Insur_Type has 236615 / 80.8% missing values Missing
- Orig_CLTV is highly correlated with Orig_LTV (ρ = 0.97109) Rejected
- Origination_Date has a high cardinality: 69 distinct values Warning
- Primary_Mortg_Insur_Percent has 236615 / 80.8% missing values Missing
- Product_Type has constant value FRM Rejected
- Property_State has a high cardinality: 54 distinct values Warning

Figure 5. pandas_profiling.ProfileReport() Overview..

The eleven features used in the model are: Borrower Credit Score, Co-Borrower Credit Score, Original Debt to Income Ratio, Original Loan-to-Value (LTV), Original Unpaid Balance, First Time Home Buyer Indicator, Loan Purpose, Property Type, Number of Borrowers, Occupancy Type, and Zip Code. The target or label is the Classification (1 or 0) column to represent foreclosure or not.

Each of the three models were evaluated using the F1 score to compare their test data results. F1 is a composite score of the precision and recall scores. Precision measures the proportion of positive identifications that are correct: True Positive / (True Positive + False Positive). Recall measures the proportion of actual positives correctly identified: True Positive / (True Positive + False Negative). The F1 score is the weighted average of the precision and recall score: 2 * (Precision * Recall) / (Precision + Recall).

The random forest classifier was the first to be executed and measured. The hyperparameters were set with n_estimators (number of trees in the forest) = 100, max_features (the number of features to consider when looking for the best split) = the square root of the count of features, and min_samples_split (the minimum number of samples required to split an internal node) = 2.

```
# Random forest classifier
def random_forest(n_estimator, max_feature, min_samples_split):
    global y_test, y_predict, model
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state=0)
    model = RandomForestClassifier(n_estimators=n_estimator, random_state=0, max_features=max_feature,
                              min_samples_split=min_samples_split)
    model.fit(X_train, y_train)
    y_predict = model.predict(X_test)
    return y_test, y_predict, model

# Initial random forest run of model using default hyperparameters
random_forest(100, int(np.sqrt(len(features))), 2)
print(classification_report(y_test, y_predict))
```

Output:

```
             precision    recall  f1-score   support

          0       0.97      1.00      0.98     70885
          1       0.68      0.02      0.04      2292

avg / total       0.96      0.97      0.95     73177
```

The total F1 score for the random forest classifier is fairly high at 95%. However, upon closer inspection there is a major imbalance between the two classifications since there are so many more records that did not result in foreclosure than those that did. The F1 score for the predicted foreclosures is 4% which is unacceptably low. To mitigate the imbalance, resampling in the form of random undersampling was performed to reduce the count of non-foreclosures and make them equal to the number of foreclosures. The function NearMiss() is used for the undersampling, reducing the non-foreclosure data set size to 9.7k to match that of the foreclosure data set.

```
# Resample data for undersampling
From imblearn.under_sampling import NearMiss
nr = NearMiss()
X, y = nr.fit_sample(X, y)
```

Repeating the random forest model, increases the F1 score to 79% for foreclosures.

Output:

```
             precision    recall  f1-score   support

          0       0.79      0.77      0.78      2402
          1       0.78      0.79      0.79      2430

avg / total       0.78      0.78      0.78      4832
```

Next for evaluation was k nearest neighbors with hyperparameter n_neighbors (number of neighbors) = 5. The F1 score for foreclosures was 75%.

```
# k nearest neighbors
X_train, X_test, y_train, y_test = train_test_split(
          X, y, test_size = 0.25, random_state=0)

knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train, y_train)

y_predict = knn.predict(X_test)
print(classification_report(y_test, y_predict))
```

Output:

```
          precision    recall  f1-score   support

       0       0.72      0.89      0.80      2402
       1       0.86      0.66      0.75      2430

avg / total       0.79      0.78      0.77      4832
```

Finally, the logistic regression classifier was evaluated resulting in an F1 score of 66%.

```
# Logistic regression
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
                                                    random_state=0)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_predict = logreg.predict(X_test)
print(classification_report(y_test, y_predict))
```

Output:

```
             precision    recall  f1-score   support

       0       0.65      0.59      0.62      2402
       1       0.63      0.69      0.66      2430

avg / total       0.64      0.64      0.64      4832
```

The random forest classifier with the highest F1 score of 79% for foreclosed loans was chosen as the best choice for this classification problem. The next steps involved tuning the three hyperparameters to achieve a slightly better F1 score. The n_estimators evaluated were 10, 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000. The max_features evaluated were 2, 3, 4, 5, 6, 7, 8, 9, 10, and 11. The min_sample_split values estimated were 2, 4, 8, 16, 32, 64, 128, and 256. The best random forest F1 score of 80% for foreclosures was achieved by using n_samples = 400, max_features = 2, and min_sample_split = 64.

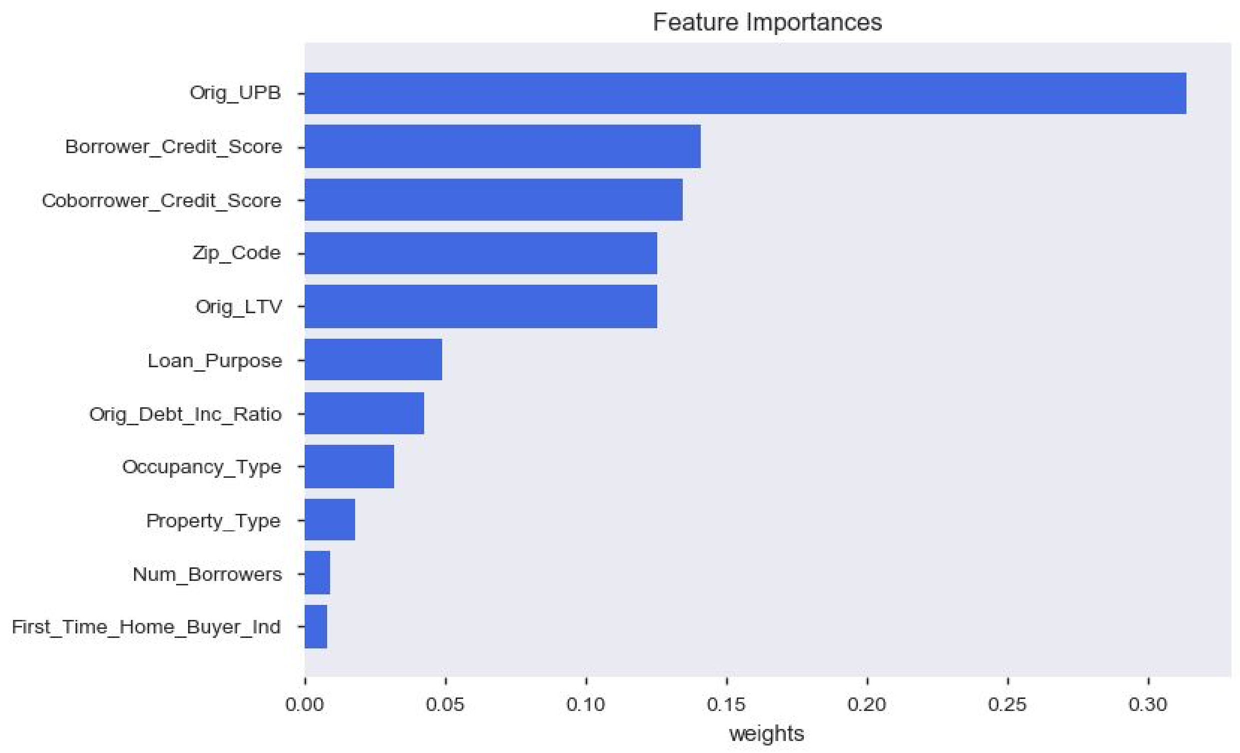
Output:

```
            precision    recall  f1-score   support

       0       0.81      0.75      0.78      2402
       1       0.77      0.82      0.80      2430

avg / total       0.79      0.79      0.79      4832
```

To understand how each of the features contributed to the random forest model, the feature_importance method was applied to the model.

```
# Combine features and feature importances
zipped = zip(features, model.feature_importances_)
importances, features = zip(*sorted(zip(list(model.feature_importances_), features)))

# Graph feature importances in descending order
_ = plt.barh(list(range(0,11)), importances, color='royalblue')
_ = plt.yticks(np.arange(11), list(features))
_ = plt.grid(False)
_ = plt.xlabel('weights')
_ = plt.title('Feature Importances')
```

Output:



The Original Unpaid Balance overwhelmingly held the most weight at over 30%. Borrower Credit Score, Co-Borrower Credit Score, Zip Code and Original Loan to Value (LTV) all had weights between 12% and 14%. The other features had smaller impacts to the model.

<u>Conclusion</u>

- Using the random forest classifier on the Fannie Mae data set provided a fairly good model, with an F1 score of 80%, to predict the loans that may foreclose with just under 20,000 records.
- This model may be improved by reducing the number of features and further tuning of the hyperparameters.

- Evaluation of data from additional year’s quarters may improve upon the model’s accuracy scores.