HDMA Washington Home Loans Prediction

Overview

This project is divided into 4 parts

Data Wrangling – Methods used to transform data into statistical usable format

Data Story – Visual insights into data

EDA – A hypothesis test to analyze bias and correlation

Prediction model – Machine learning algorithms used and methods applied to predict the model

Conclusion – Findings of the Machine learning models

Introduction

The Home Mortgage Disclosure Act (HMDA) requires many financial institutions to maintain, report, and publicly disclose information about mortgages. The project uses this data to make loan approval predictions.

Data is at https://www.kaggle.com/miker400/washington-state-home-mortgage-hdma2016/home

This project builds prediction model to help bank make data driven decisions and features affecting loan approval.

This project is divided into three parts **Data Wrangling**, **EDA** and **Prediction Model.**

Data Wrangling

Following data cleaning methods are used

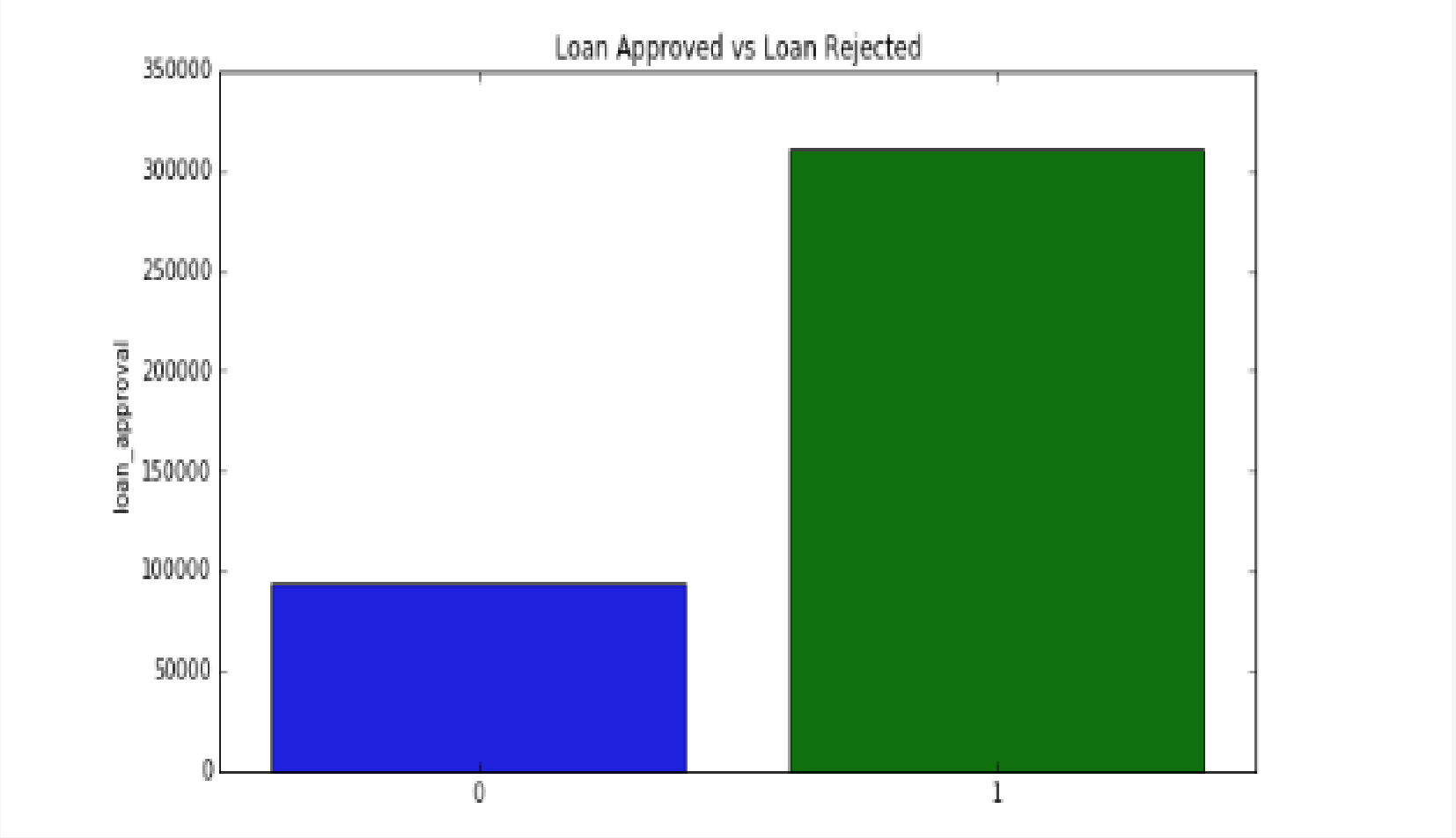
- **Drop Columns** - Columns with few data elements are dropped
- **Column Data Types** – Data type of Columns with categorical data is changed to “Category”
- **Missing Data –** Columns having around 90% of “NaN” values are dropped, since there were 11 columns with less than 90% data and also imputed values cause bias in data. Columns having fewer missing values(2%) of “NaN” data are **imputed** to median values and values based on other column values.

  ❖For ex - Substitute missing values in msamd_name with value based on value in census_tract_number

- **Outliers -** Outliers are visually inspected with box plots and filtered out

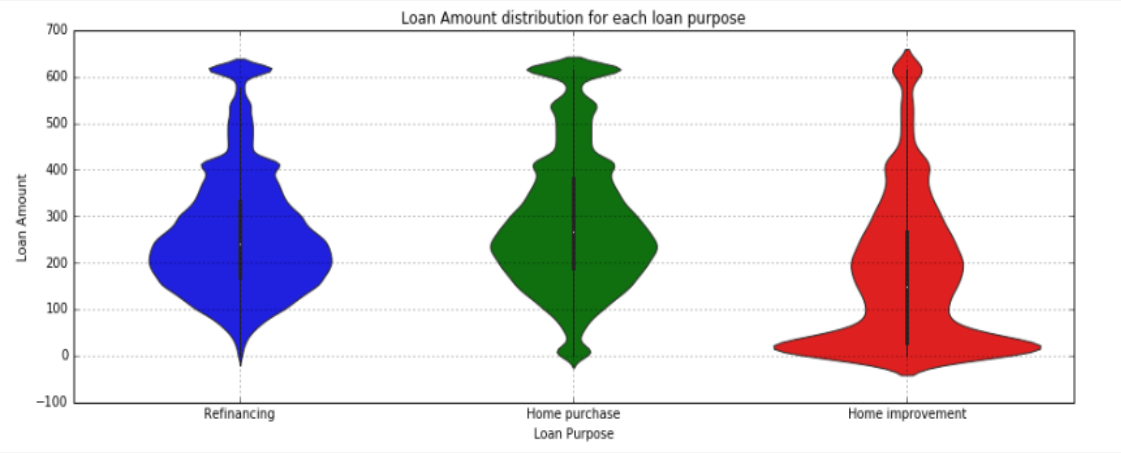
Data Story

There are 310593 loan approved applications and 93595 loans rejected. Data is not a balanced data.

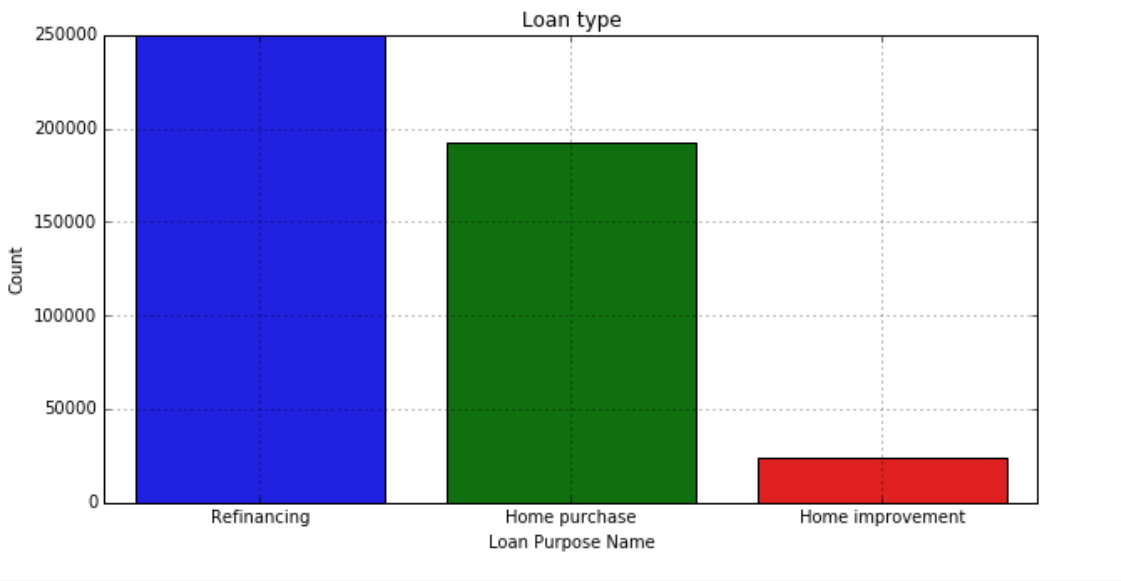


Data Story

- Distribution of loan amounts are between 150K to 300K across all the Metropolitan areas, loan purpose and action taken.

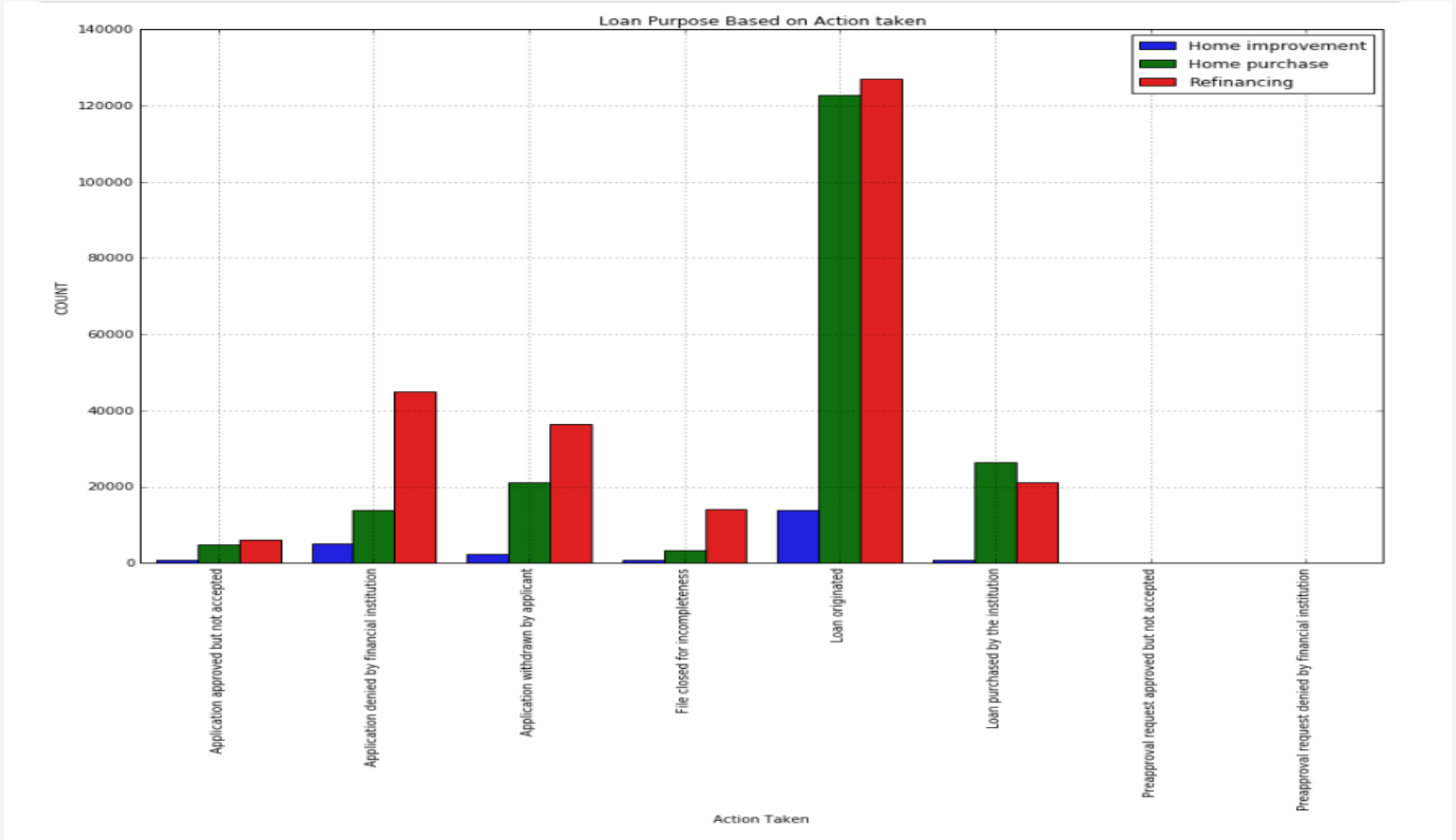


- 53% of the loans applications are Refinancing, 40% are for Home Purchase and 7% are for Home Improvement.

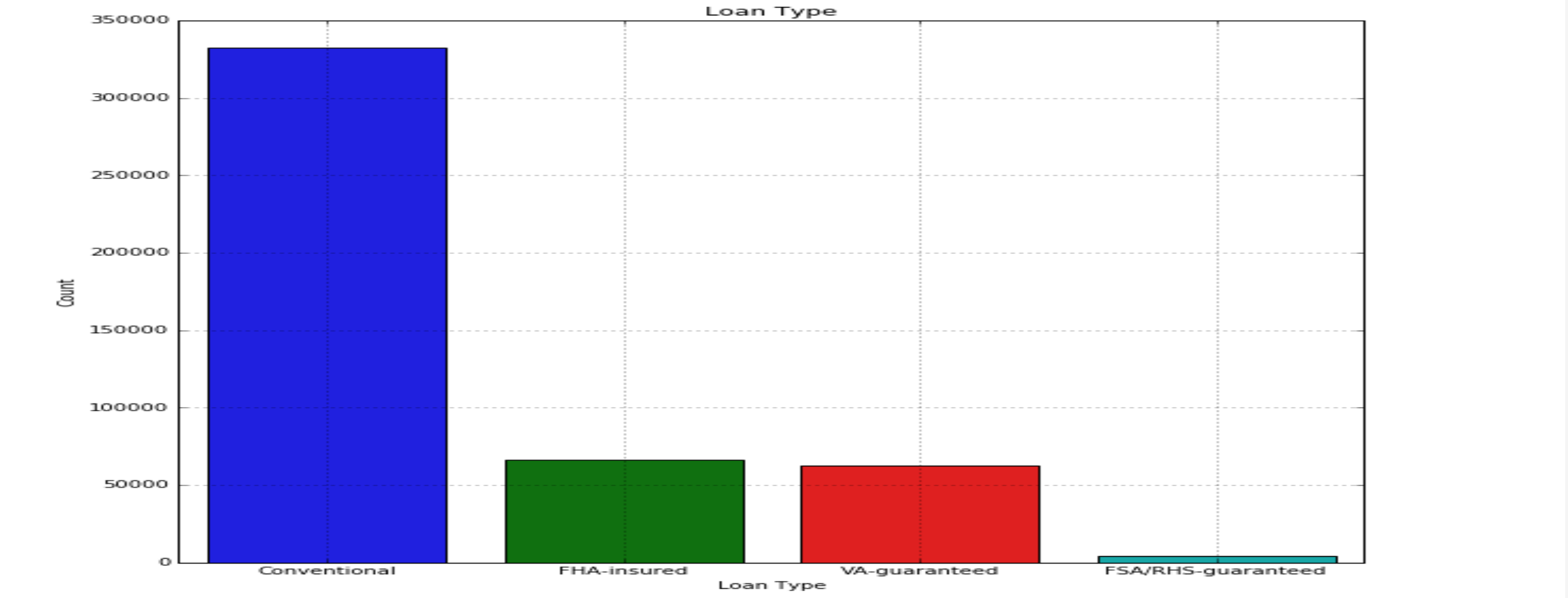


Data Story

Approximately 40% of the loans are approved in each loan purpose Home purchase and Refinancing, and 20% for Home Improvement.



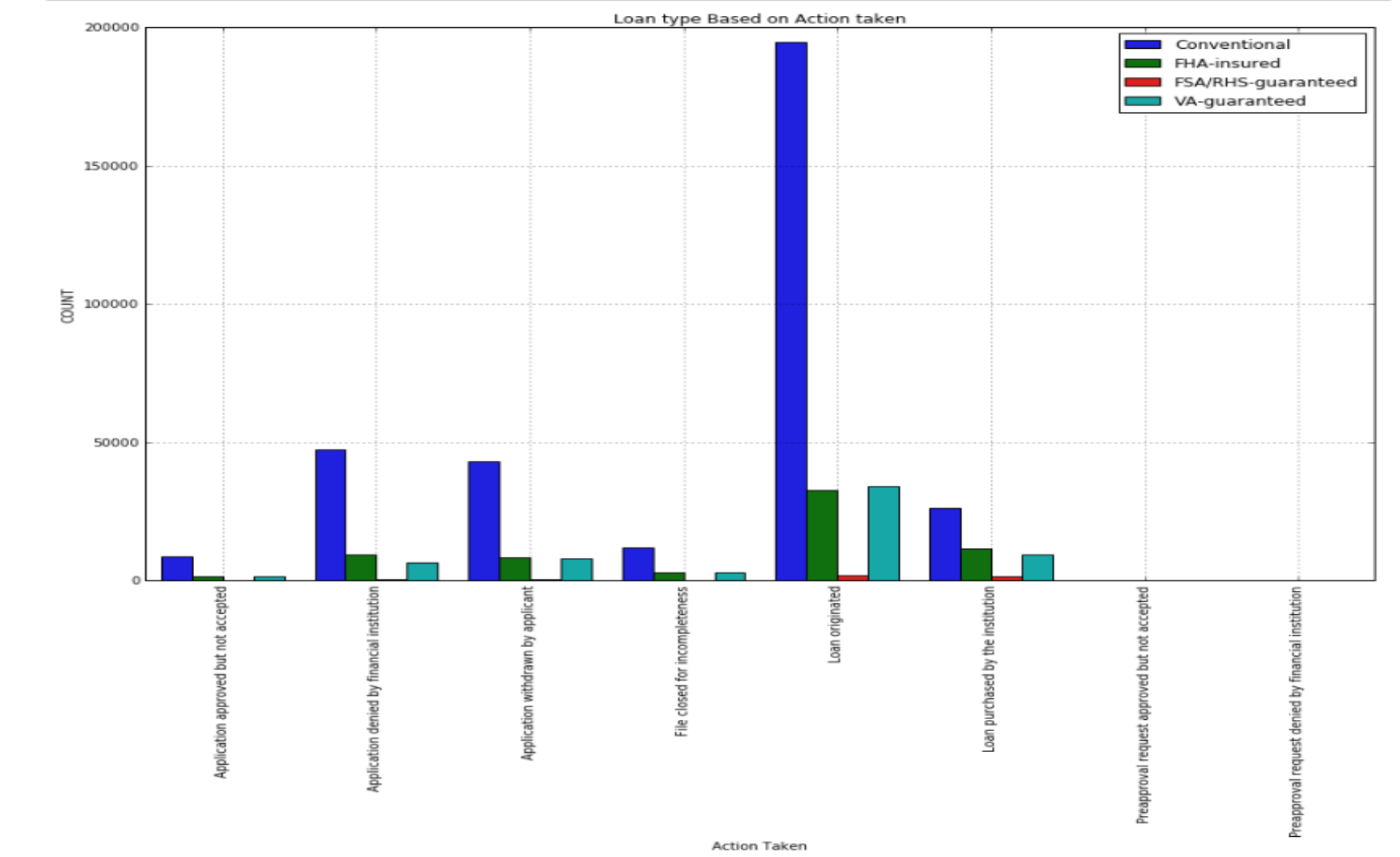
Data Story



Approximately 70% of the loans applications are Conventional loan type.

Data Story

Approximately 60% of the loans approved are Conventional loans.



EDA

Gender Bias

A hypothesis test was conducted to determine Gender bias.

With a t-statistic value 7.83 and P-value 5.10, failed to reject null hypothesis and determined that there is a gender bias in loan approval.

Racial Bias

A Hypothesis test was defined to evaluate if there is racial bias between ‘Not Hispanic or Latino’ and ‘Hispanic or Latnio’.

After calculating t-statistic value 16.62 and P-value 0, it was determined that there is no racial bias in loan approval.

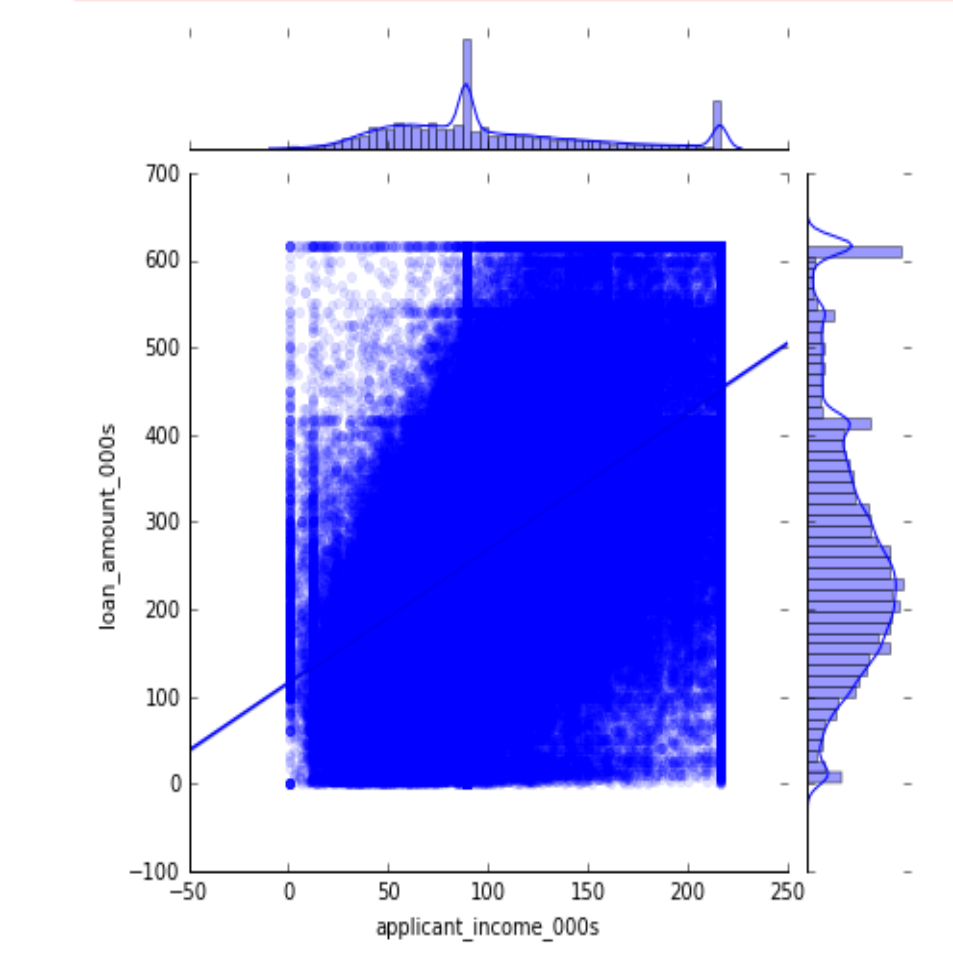
EDA

Correlation Between Loan Amount and Applicant Income

Pearson correlation coefficient 0.55 was calculated using np.corrcoef() and a hypothesis test was calculated to test the correlation coefficient.

A t-statistic 460.16, confidence interval (-3.48, 3.48) and P-value 0, was calculated and determined that there was a low chance of getting the observed correlation coefficient.

However scatter plot does show some positive correlation between loan amount and applicant income.



Prediction Model

Four models are used for loan approval prediction

- Logistic Regression from SKLearn
- GLM and Logistic Regression from StatsModels
- Random Forest
- XGBOOST

Data will be trained and tested on all the above four algorithms. Model will be evaluated by calculating

- Accuracy score
- Precision and Recall
- Confusion matrix
- ROC Area

Prediction Model - Logistic Regression using SKLearn

- Split data for training and testing with all the features.
- **Recursive Feature elimination(RFE)** method is used for feature selection.
- After find most important features from RFE, again split data for train and test based on features selected.
- Second split data for tuning hyperparameter **C** using **GridSearchCV and C value of 10** was obtained.
- **Confusion Matrix** and **Precision** and **Recall** is computed for determining accuracy.
- Above steps are repeated for all 25 features by eliminating one at a time

Prediction Model - Logistic Regression using SKLearn

Test data

Precision and Recall

class	precision	recall	f1-score	support
0	0.77	0.77	0.77	18723
1	0.93	0.93	0.93	62115

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	14377	4346
Actual 1	4203	57912

Accuracy Score : 89.42
Area Under curve score : 0.89424

Training data

Precision and Recall

class	precision	recall	f1-score	support
0	0.78	0.77	0.77	18723
1	0.93	0.93	0.93	62115

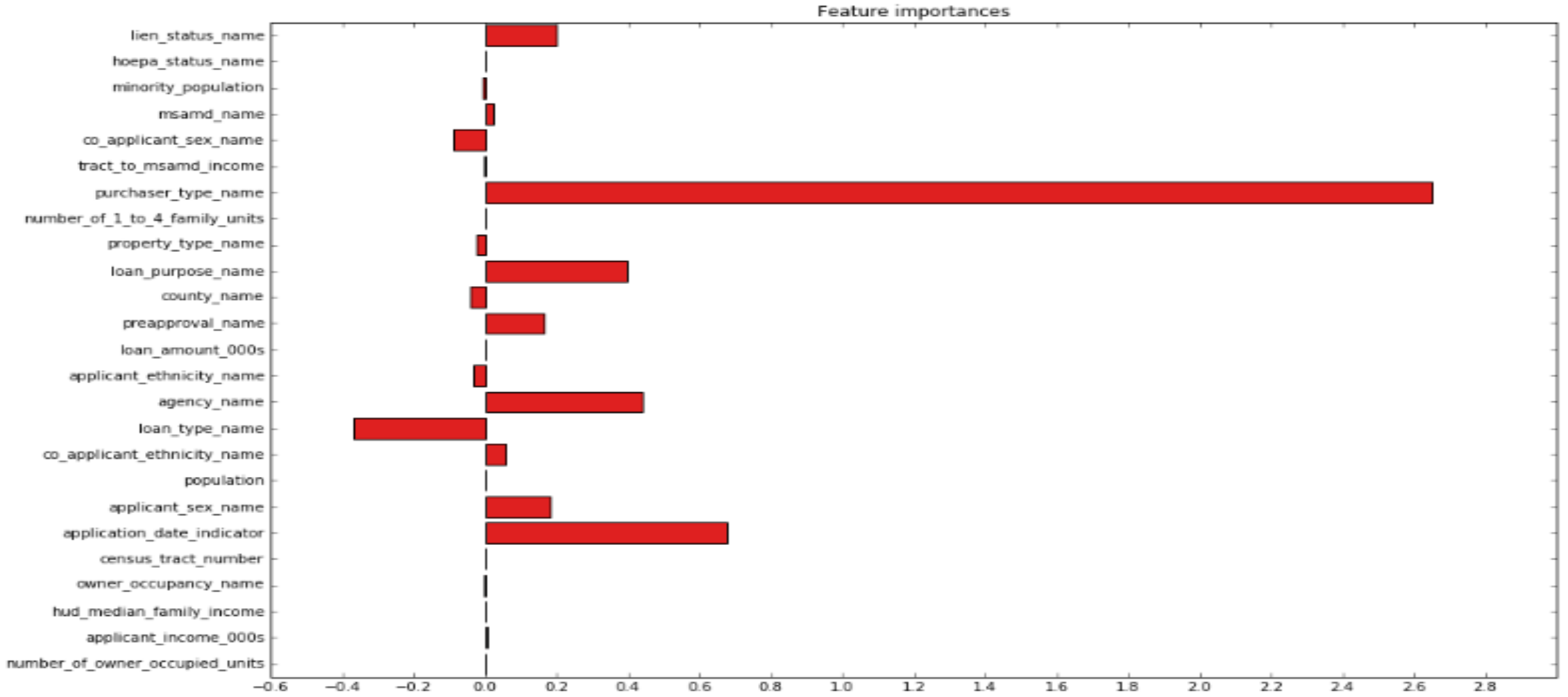
Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	57378	17494
Actual 1	16410	232068

Accuracy Score : 89.51
Area Under curve score : 0.95682

Prediction Model - Logistic Regression using SKLearn

Feature Importance



Prediction Model – Logit and GLM using stats model

Both logit and GLM yield same result.

Test data

Precision and recall

class	precision	recall	f1-score	support
0	0.74	0.83	0.79	18723
1	0.95	0.91	0.93	62115

Confusion matrix

	Predicted 0	Predicted 1
Actual 0	15624	3099
Actual 1	5350	56765

Accuracy Score : 89.55
Area Under curve score : 0.95870

Training data

Precision and Recall

class	precision	recall	f1-score	support
0	0.75	0.84	0.79	18723
1	0.95	0.92	0.93	62115

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	62523	12349
Actual 1	21113	227365

Accuracy Score : 89.65
Area Under curve score : 0.95933

Prediction Model – Random Forest

In this model hyperparameters are tuned using **RandomizedSearchCV**. Hyperparameters found in RandomizedSearchCV are used to for learning **RandomForestClassifier**. This model is tested on test data and evaluated using accuracy, precision and recall, and confusion matrix.

Hyperparameters tuning

n_estimators - the number of trees in the forest. Usually higher number of trees the better to learn the data. n_estimators = [4, 8, 16, 32, 64, 100, 200]

max_features - number of features to consider when looking for the best split. max_features = [2, 4, 5, 10, 15, 20, 25]

max_depth - depth of the tree, the more splits it has and it captures more information about the data. But as the tree gets very deep, it might lead to overfitting. max_depth - [4, 8, 10, 12, 16, 32, 64]

Prediction Model – Random Forest

Hyperparameters tuning continued ….

min_samples_split - number of samples required to split an internal node. smaller samples might lead to overfitting. min_samples_split = [2, 4, 6, 8, 10, 12, 16, 32, 64]

min_samples_leaf - The minimum number of samples required to be at a leaf node. smaller samples might lead to overfitting. min_samples_leaf = [2, 4, 6, 8, 10, 12, 16, 32, 64]

criterion - ['gini','entropy']

bootstrap - [True, False]

Best Parameters are **n_estimators – 200, max_features - 10, max_depth - 12, min_samples_split - 4, min_samples_leaf - 8, criterion - gini, bootstrap - False**

Prediction Model – Random Forest

Test data

Precision and recall

class	precision	recall	f1-score	support
0	0.80	0.83	0.81	18723
1	0.95	0.94	0.94	62115

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	15609	3114
Actual 1	3994	58121

Accuracy Score : 91.21
Area Under curve score : 0.97015

Training data

Precision and recall

class	precision	recall	f1-score	support
0	0.80	0.84	0.82	18723
1	0.95	0.94	0.95	62115

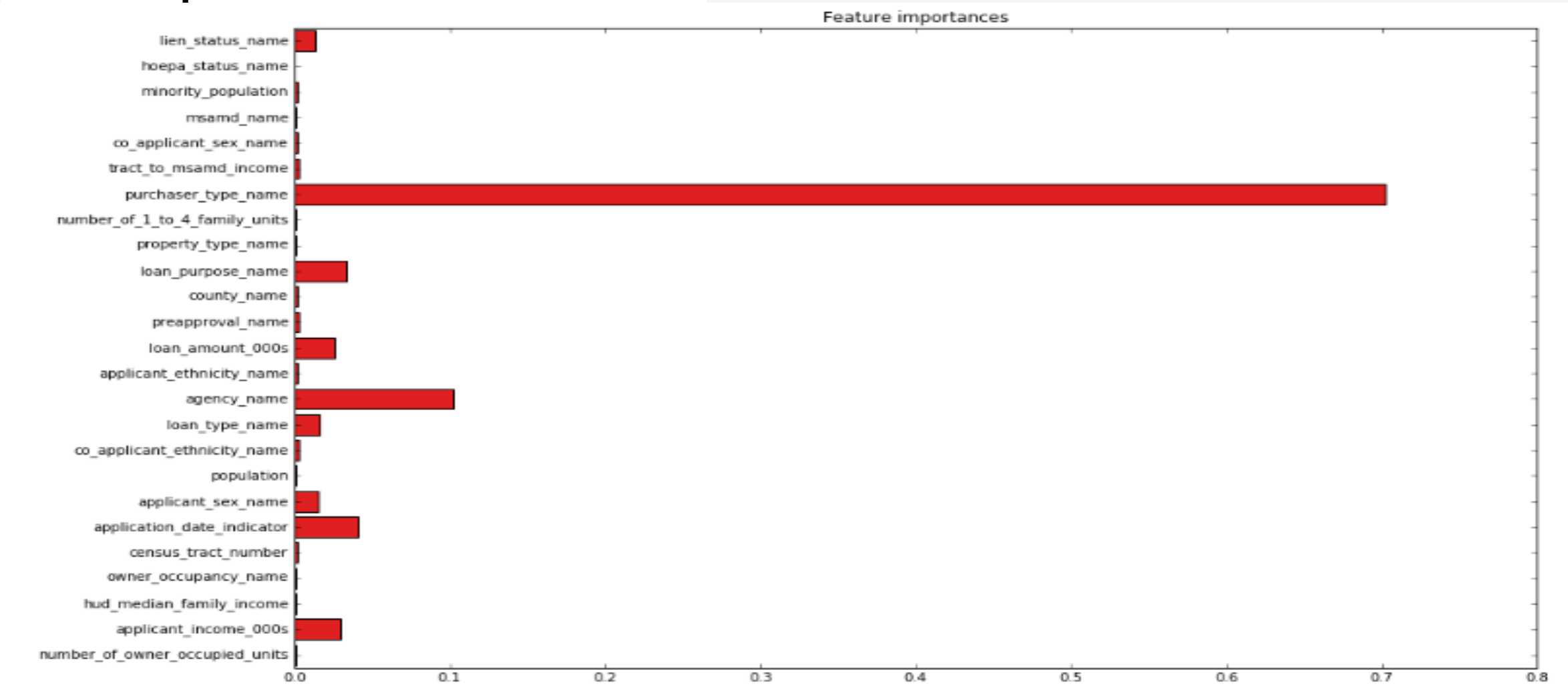
Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	63157	11715
Actual 1	15318	233160

Accuracy Score : 91.64
Area Under curve score : 0.97363

Prediction Model – Random Forest

Feature Importance



Prediction Model – XGBOOST

In this model hyperparameters are tuned using **RandomizedSearchCV**. Hyperparameters found in **RandomizedSearchCV** are used to for learning **XGBClassifier**. This model is tested on test data and evaluate using precision and recall, and confusion matrix.

Hyperparameters Tuning

n_estimators - number of trees to grow. Larger the tree size better the model, but more numbers of trees can be computationally expensive and affects the performance of the model n_estimators = [4, 8, 16, 32, 64, 100, 200]

max_depth - depth of the tree, the more splits it has and it captures more information about the data. But as the tree gets very deep, it might lead to overfitting max_depth = [4, 8, 10, 12, 16, 32, 64]

Prediction Model – XGBOOST

Hyperparameters Tuning continued….

min_child_weight - Minimum sum of instance weight needed in a child. min_child_weight = [2, 4, 6, 8, 10, 12, 16, 32, 64]

gamma - [0.1, 0.2, 0.3, 0.4, 0.5]

colsample_bytree - Subsample ratio of columns when constructing each tree. colsample_bytree = [0.2, 0.4, 0.6, 0.8]

colsample_bylevel - Subsample ratio of columns for each split, in each level colsample_bylevel = [0.2, 0.4, 0.6, 0.8]

Best Parameters are **n_estimators - 64, max_depth - 32, min_child_weight - 64, gamma – 0.3, colsample_bytree- 0.8, colsample_bylevel – 0.2**

Prediction Model – XGBOOST

Test data

Precision and recall

class	precision	recall	f1-score	support
0	0.80	0.83	0.82	18723
1	0.95	0.94	0.94	62115

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	15621	3102
Actual 1	3889	58226

Accuracy Score : 91.35
Area Under curve score : 0.97056

Training data

Precision and recall

class	precision	recall	f1-score	support
0	0.81	0.85	0.83	18723
1	0.95	0.94	0.95	62115

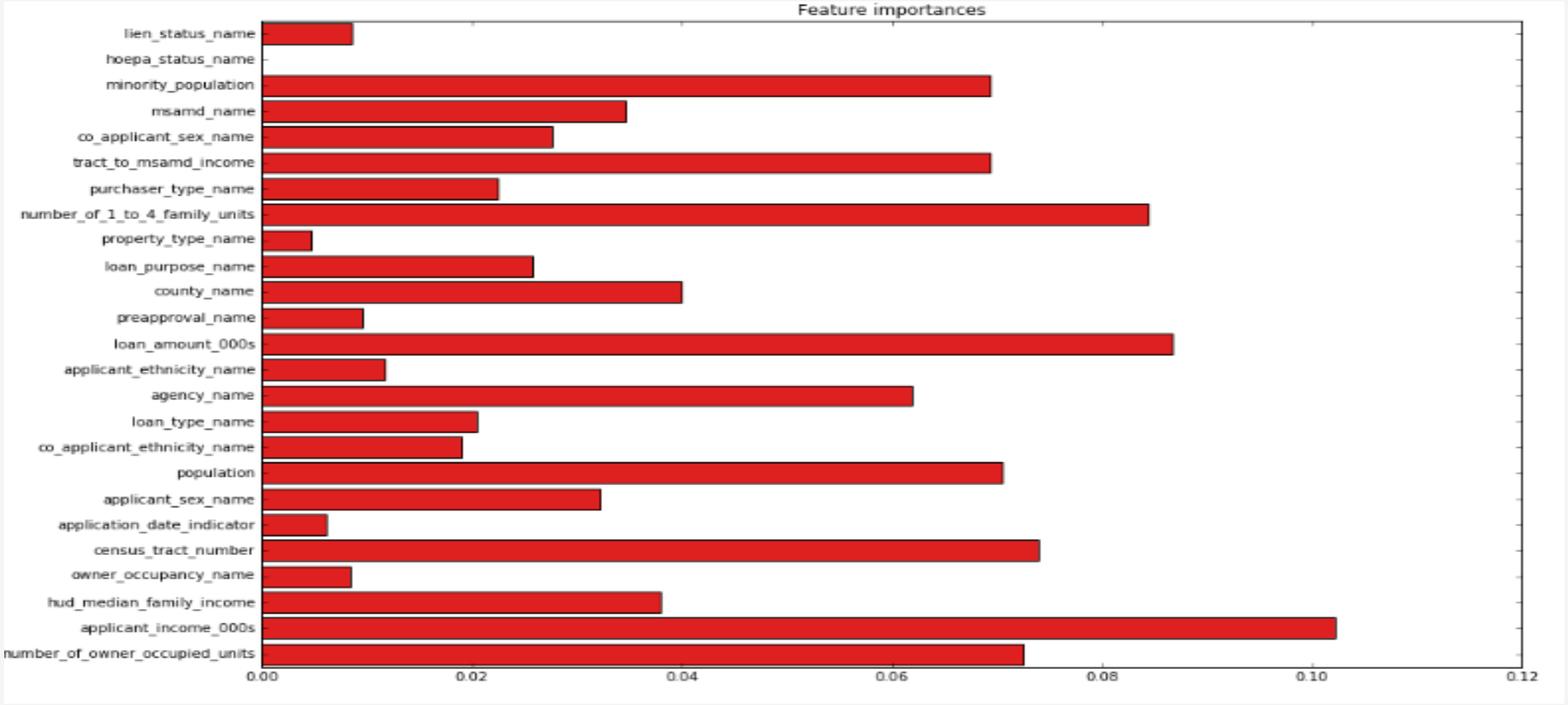
Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	63644	11228
Actual 1	14538	233940

Accuracy Score : 92.03
Area Under curve score : 0.97434

Prediction Model – XGBOOST

Feature Importance



Conclusion

Among all the four models, **XGBOOST** and **Random Forest** do better than other models. **XGBOOST** and **Random Forest** both have almost same precision and recall. **XGBOOST** is 1% better(F1-Score) at predicting loan rejection rate than **Random Forest**.