Consumer Financial Protection Bureau

Consumer Complaints Analysis Final Report

Capstone Data Science Project

Prepared By

Krishna Charan B C

Date: August 6th , 2018

Table of Contents

Executive Summary

Introduction

Consumer Finance Protection Bureau has been instituted to provide a single point of regulatory board to enforce federal consumer financial laws and to protect consumers in the financial market. CFPB focuses on imposing laws that outlaw discrimination in consumer finance and researching the consumer experience with financial products. CFPB is facing challenges in maintaining the integrity in consumer market, financial institutions are unnecessarily spending resources to resolve the consumer complaints, even if it is not their fault, and consumers are skeptical to select financial products and services from a particular provider because of lack of information and knowledge. The undertaken project will facilitate CFPB to enforce the federal consumer financial laws with a proof point, optimize the usage of resources of financial institutions, and assist the consumers in decision making while opting for products or services offered in financial market.

Methodology

Extensive research has been conducted by reviewing several business articles and relevant industry journals to understand the disputes in financial market and latest techniques adopted to tackle the disputes. Two key performance indicators (KPIs) have been identified as relevant to measure the outcome of those techniques. Predicting the financial institutions' responses to consumer complaints and forecasting the potential issues to be raised related to financial products with given conditions, would be the answer for many disputes. Machine learning techniques, like Random Forest, Decision Tree and Neural Net classification, used in undertaken project, are performing up to industry standard for preventive complaint analysis. High information gain capability of these machine learning techniques, enables to outperform in the industry.

Current state analysis

At present, in financial industry consumer complaint resolution rate averaged approximately 70 percent. Mortgage related consumer complaints are most frequent. Though proportion of complaints from females are high, responses from financial institutions are not biased. When the house value is less, so

the mortgage amount, it raises more number of complaints, but turn out to be misunderstanding of consumers. When the house value is more, so the mortgage amount, it raises less number of complaints, but turn out to be fault of financial institutions. Consumers in the age range of 30~40 raise more number of complaints related to mortgage and high proportion of those complaints turn out to be fault of financial institutions. Consumers in the age range of 50~60 raise less number of complaints related to mortgage and high proportion of those complaints turn out to be misunderstanding of consumers.

Opportunities and Recommendations

In advance knowledge of company responses and issues to be raised, will assist CFPB to categorize the complaints to maintain integrity. Financial institutions can address to complaints, which have potential for financial payoff and mitigate the factors leading to financial payoff. Consumers can opt for financial products by understanding the potential upcoming issues.

Understanding the demographics of consumers, CFPB and financial institutions can educate the consumers to reduce the number of complaints because of the misunderstanding. CFPB and financial institutions can use the techniques to optimize the scarce resources.

Deployment of models in real business operations, while considering the business constraints, will allow the stakeholders to function efficiently, however, continuous monitoring, improvement, and viability check of outcomes are necessary to make it a success.

Business Understanding

Consumer Financial Protection (CFPB)

The Consumer Financial Protection Bureau (CFPB) is the federal agency that focuses on consumer financial protection and assist in resolving consumer complaints to maintain the integrity. CFPB's officials receive consumer complaints directly from consumers about the challenges they face from financial institutions, bring their concerns to the attention of related financial institutions, and assist in addressing the issues. Figure 1 below outlines the conceptual steps CFPB follows to solve the customer complaints.

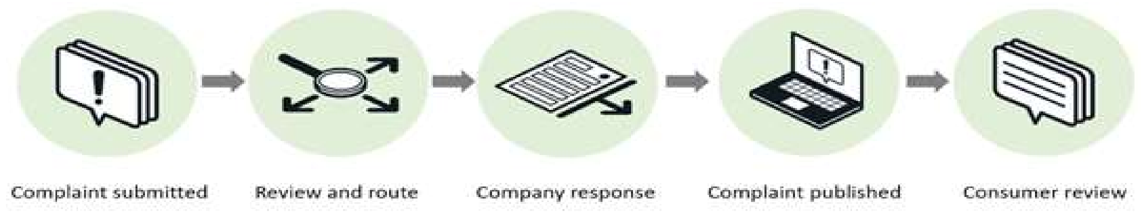


Figure 1: CFPB consumer complaints' processing outline

Business Objectives and Success Criteria

Analysis of the consumer complaints that have been logged in with Consumer Financial Protection Bureau (CFPB) will explore the factors that drive the consumers to register complaints with CFPB. Accordingly, fine tuning the business processes can improve the services provided by the financial institutions, increase complaint resolution rate, identify the serious complaints for timely resolution and optimize the usage of various resources.

Understanding the flaws in the process of offering financial services and products will help CFPB in advising financial institutions for reasonable complaint resolution, prioritizing the important issues and refining CFPB's own process to tackle the customer complaints. Ultimately analysis will aid all the stakeholders, which includes CFPB, financial institutions, and consumers, to take economic and feasible decisions. Key performance indicators or the success factors of the undertaken project are complaint resolution rate and number of the serious issues registered. Deployment of the strategies evolved from the analysis, when considering the operational constraints, must increase complaint resolution rate and decrease the number of serious issues registered. With the available customer complaints dataset from CFPB following analyses will be conducted to address above written issues:

1. Predicting the response given by the financial institutions to a consumer complaint

Each observation in the CFPB dataset describes the complaint registered by an unique consumer against a financial institution and the responses that the company gave for the same complaint. Predicting the responses provided by the financial institutions will assist in prioritize the complaints which will be resolved by official explanation and which will lead to financial payoff. CFPB can categorize the complaints to take required action and to maintain integrity. Financial institutions can focus more on the complaints, which includes financial payoff, to improve customer service and work on the factors leading to financial payoff.

2. Classify the seriousness of a complaint based on demographic data

Adding demographic features to original CFPB dataset the seriousness of a complaint will be predicted and the complaint will be classified either as consumer fault or as financial institution fault. This analysis will facilitate to identify demographics that lead to seriousness of complaints. CFPB and financial institutions can track the demographics of consumers to identify the seriousness of the complaints. Financial institutions can revise and add extra terms and conditions to offer services at particular demographic region to reduce the probability of raising serious complaints.

3. Predict mortgage issues based on Personal Income Tax data

Adding income tax return data features to original CFPB dataset enables to predict the issues generated with mortgage products. Assuming that all zip codes have different types of people with varying levels of financial well beings and demographics, data analysis will be conducted to explore and predict if gross income or financial condition in any zip code affects the issues raised in mortgage related products. Such information would be valuable to the financial institutions to predict and prepare for potential issues they are likely to face in the future based on zip code. Also, products and services terms and conditions can be revised and added to improve the product or service quality. Also, financial institutions can take precautionary steps to avoid the issues raised for mortgage related products or services.

Data Mining Objectives and Success Criteria

1. Predicting the response given by the financial institutions to a consumer complaint

Using the different attributes of the CFPB complaints dataset, the response from the financial institutions would be predicted using the models built based on training dataset after data partition. Models are built on the concepts of multinomial classification using machine learning techniques. Comparing the performance statistics of different models, one would be recommended to satisfy the stated business objective. Getting higher predicting accuracy would be the success criteria.

2. Classify the seriousness of a complaint based on demographic data

Various classification models are developed using machine learning concepts, to identify complaints' seriousness level based on the demographics and other significant attributes, even before the regulatory authority (CFPB) forwards the complaint to the respective financial institution. Zip codes will be clustered looking into demographics similarities. For this CFPB dataset and demographics dataset has been merged. Classifying the seriousness of the complaints with higher accuracy would be the success criteria.

3. Predict mortgage issues based on Personal Income Tax data

To approach this problem CFPB consumers' complaint dataset and income tax return dataset has been merged. Based on the new dataset attributes, different issues raised under the products related to mortgages has been predicted using different machine learning techniques. Classifying the issues related to products with higher accuracy would be the success criteria.

Data Understanding

Dataset # 1: Consumer Financial Complaints Dataset

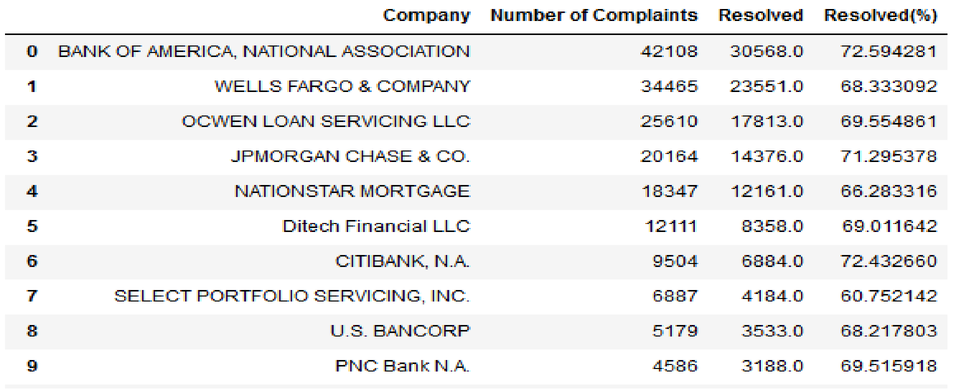
1. Data Definition

Consumer Financial Protection Bureau provides thousands of consumers' complaints about financial products and services offered by various financial institutions. Those complaints are published at their website after the company responds or after 15 days, whichever comes first. The dataset has complaints received by the CFPB from December, 2011 to July, 2018. Data dictionary is presented in table A1 in appendix. For each complaint the consumer provides the Issue, Sub Issue, Product, Sub Product along with the narrative description of the complaint, which all are included in the dataset. In addition, variables such as Company, Company response, State, ZIP code, Timely response provided or not and whether consumer has disputed back for the company response has been provided. Each complaint is uniquely identified by "Complaint ID".

2. Descriptive Analysis

Most of the variables in complaints dataset are categorical variables. The top 10 companies to receive the most complaints in the dataset were:

Table 1: Companies to receive the most complaints and Complaint Resolution Rate

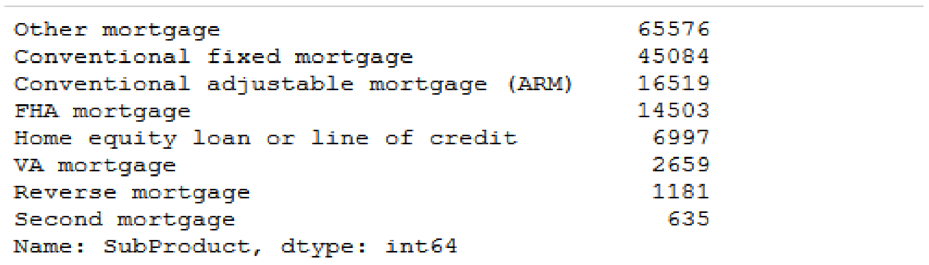


	Company	Number of Complaints	Resolved	Resolved(%)
0	BANK OF AMERICA, NATIONAL ASSOCIATION	42108	30568.0	72.594281
1	WELLS FARGO & COMPANY	34465	23551.0	68.333092
2	OCWEN LOAN SERVICING LLC	25610	17813.0	69.554861
3	JPMORGAN CHASE & CO.	20164	14376.0	71.295378
4	NATIONSTAR MORTGAGE	18347	12161.0	66.283316
5	Ditech Financial LLC	12111	8358.0	69.011642
6	CITIBANK, N.A.	9504	6884.0	72.432660
7	SELECT PORTFOLIO SERVICING, INC.	6887	4184.0	60.752142
8	U.S. BANCORP	5179	3533.0	68.217803
9	PNC Bank N.A.	4586	3188.0	69.515918

From table1, it can be observed that, the companies receiving the most complaints have the most complaint resolution rate and Bank of America, Wells Fargo, Ocwen and JPMorgan were the top 4 companies with the most number of complaints. The average wait time before a complaint was received by CFPB and then sent to the Company for Resolution was 4.90 days. The minimum and maximum number of wait time (in days) was 0 day and 1553 days.

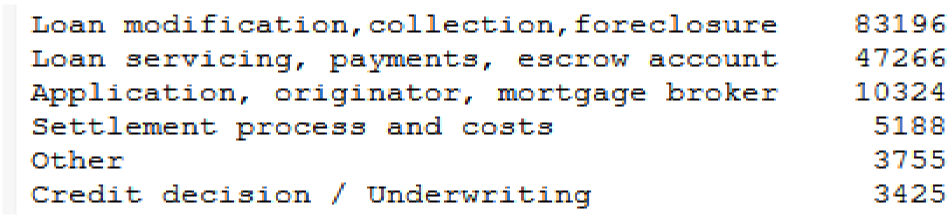
3. Exploratory Analysis

Dataset has been explored to understand the features related complaints. From the analysis, mortgage related issues are the most frequent complaint received by the regulatory authority. So further analysis has been conducted on mortgage related complaints. Following are the sub products related to mortgage across all the financial institutions and “Other Mortgage” sub products is the most frequent one.



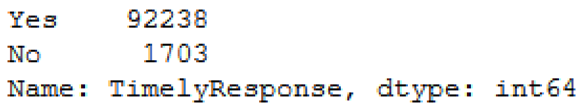
```
Other mortgage                          65576
Conventional fixed mortgage             45084
Conventional adjustable mortgage (ARM)  16519
FHA mortgage                            14503
Home equity loan or line of credit       6997
VA mortgage                              2659
Reverse mortgage                         1181
Second mortgage                           635
Name: SubProduct, dtype: int64
```

Further drilling down the mortgage issues, it has been found that Loan modification, collection, foreclosure and Loan servicing, payments, escrow account related issues are the most frequent among the complaints.



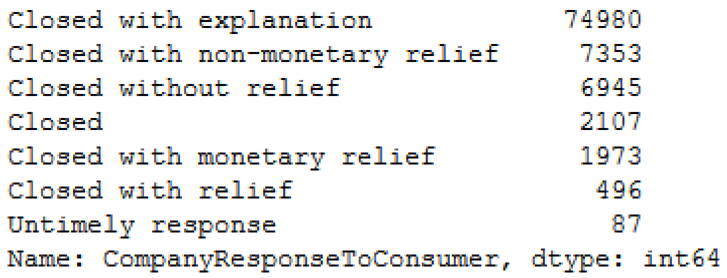
```
Loan modification,collection,foreclosure    83196
Loan servicing, payments, escrow account    47266
Application, originator, mortgage broker    10324
Settlement process and costs                 5188
Other                                        3755
Credit decision / Underwriting               3425
```

Also, it can be observed that 97 percent of the complaints receive Timely Response by the institutions according to stipulated resolution time frame set by the CFPB.



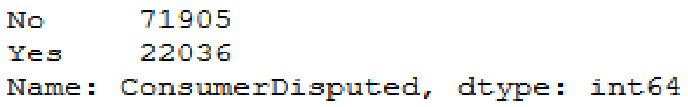
```
Yes    92238
No      1703
Name: TimelyResponse, dtype: int64
```

The company response to the consumer is a factorial data, with 9 different response levels. It can be observed that a high proportion of complaints are responded with an explanation and closed, indicating no fault of the company. The responses “Closed with monetary relief”, “Closed with non-monetary relief” and “Closed with relief” indicate error or fault from the company related to the issue or complaint raised by the consumer. Also, the responses “Response pending” and “Untimely response” indicate that the institute has not responded to the complaint within the stipulated time frame.



```
Closed with explanation            74980
Closed with non-monetary relief     7353
Closed without relief               6945
Closed                              2107
Closed with monetary relief         1973
Closed with relief                   496
Untimely response                     87
Name: CompanyResponseToConsumer, dtype: int64
```

Also, it has been found that 24 percent of company responds has been disputed.



```
No     71905
Yes    22036
Name: ConsumerDisputed, dtype: int64
```

Exploratory analysis graphs are generated using Tableau and presented in appendix. Figure A1 represents the distribution of Consumer Complaints throughout the Country, California having the highest number of complaints. Figure A2 represents the distribution of mortgage consumer complaints throughout the Country, California having the highest number of mortgage related complaints. Figure A3 displays how company response to the customer varies with respect to companies. Figure A4 displays how company response to the customer varies with respect to different years. Figure A5 displays the distribution of issues of mortgage complaints. Figure A6 displays the distribution of mortgage complaints with respect to different institutions. Figure A7 displays the distribution of complaints with respect to different products. Figure A8 displays the variation in company response with respect to different zip codes.

4. Data Quality assessment

Considering the massive task of accumulating all the parameters for a consumer complaint against financial institutes, CFPB has done a remarkable performance in terms of data quality. However, to remove the identity in the data, CFPB has scrubbed off the personal details and few other attributes that would help us to provide a precise and accurate business solutions. Vital data observations like ZIP codes of small communities, Company’s narrative response to consumer have been scrubbed off, leaving us with poor data quality. Also, personal identification attributes like age, gender, income levels, education level, occupation of the consumer if available, through the financial institution data, would have greatly added value to the analysis. It is important to notice that the consumer complaints from the CFPB does not include lawsuits, as they are forwarded to the respective regulatory authorities by the CFPB.

Dataset # 2: Demographics dataset

1. Data Definition

This dataset provides demographic data like Number of houses, Cost of Living, Gender Population, etc. for each ZIP code, which is obtained to add useful information to complaints dataset from the CFPB. The demographics dataset has attributes like cost of living, houses with and without mortgage, house value, household income, which potentially add useful information for the mortgage related complaints analysis. Also, ZIP code is the lowest level of granularity to obtain the demographics data. Demographics information is available on the www.city-data.com, which does not provide the option to download the dataset. Hence, data mining techniques like web-scraping has been utilized to extract the information to local file system as a dataset. The data dictionary of the demographic dataset has been tabulated in the Table A2, in the appendix. The dataset, thus includes demographics for 1324 unique datasets of Texas state and accounts 16 different demographic features for each ZIP code.

2. Descriptive Analysis

The average number of houses in Texas state is 8300 houses per ZIP code, with approximately equal gender distribution of average 10000 male/female. The average houses with mortgage is 2700 and average 1900 houses without mortgage. The percent of renters is 34 with a median gross rent of 850 and the mean cost of living is 86.

3. Exploratory Analysis

Figure A9, in appendix, displays how estimated house income varies in area with high complaints and higher complaints rate is obtained in houses with higher estimated median household income. Similarly, higher median gross rent values are associated with more number of complaints as observed in Figure A10, which displays how gross rent varies in area with high complaints. Figure A11 displays population density varies in areas with high complaints, where a direct relation of population density is observed with complaints rate. Figure A12 displays the variation in sex ratio in areas with high complaints, here although it can be observed that a higher complaint rate by the female over males, the company did not discriminate its response based on the gender. Figure A13 displays the variation in unemployment percentage in areas with high complaints, where direct relation of unemployment rate with complaints rate is observed.

4. Data Quality assessment

The dataset includes demographics dataset for unique ZIP codes of Texas state, however, the website www.city-data.com does not provide data for all the unique data. Without other alternative source available to extract data for the missing ZIP codes, we consider only the ZIP codes available. The data has been manually web scraped and hence is has decent richness.

Dataset # 3: Income Tax Return dataset

1. Data Definition

The SOI Tax Stats - Individual Income Tax Statistics dataset from the IRS has selected income and tax items classified by State, Zip Code, and the size of adjusted gross income. The data are based on individual income tax returns filed with the IRS during tax years from 2012 till 2015. The data dictionary has been presented in Table: A3 in appendix.

2. Descriptive Analysis

Table 2: Tax return value based on different groups

	Columns	Mean	Median
0	single_rtns	4842.255642	790.0
1	joint_rtns	3828.020672	780.0
2	dependents	6931.148131	1170.0
3	farm_rtns	120.196064	30.0
4	unemployment	621.079888	120.0
5	agi_grp1	49615.034907	8655.0
6	agi_grp2	89507.797906	16212.0
7	agi_grp3	86376.693266	16234.5
8	agi_grp4	78238.373091	14109.0
9	agi_grp5	165855.856499	22532.0
10	agi_grp6	220606.168144	10053.5

From the table:2, the average number of single people per zip code is 4842.26. The median number of single people per zip code is 790. The average number of married people, number of dependents, and number of unemployed people are 3828, 6931, 120, and 621 per zip code respectively. The median number of married people, number of dependents, and number of unemployed people are 780, 1170, 30, and 120 per zip code respectively.

The average aggregate adjusted annual gross income for people earning less than $25k, earning between $25k-50k, earning between $50k-$75k, earning between $75k-$100k, earning between $100k-$200k, and earning more than $200k are $49,615, $89,507, $86,376, $78,238,

$165,856, and $220,606 respectively. The median aggregate adjusted annual gross income for people earning less than $25k, earning between $25k-50k, earning between $50k-$75k, earning between $75k-$100k, earning between $100k-$200k, and earning more than $200k are $8,655, $16,212, $16,234, $14,109, $22,532, and $10,053 respectively.

3. **Exploratory Analysis**

The figure: A14 in appendix shows that not every zip code has equally distributed adjusted annual gross income. It can also be observed that the mean adjusted gross income for each agi_groups is relatively larger than the median adjusted gross income for each agi_groups. From the figure: A15 in appendix it can be observed that the interquartile range for number of dependents is larger. The median for the number of people filing single tax returns, join tax returns, and tax returns with dependents has roughly same range of median values.

From the Figure: A16 in appendix it can be observed that the interquartile range for people having aggregate adjusted annual gross income for each zip code of $100k to $200k is the largest. The median aggregate adjusted annual gross income for each zip code is about the same across all US zip codes.

4. **Data Quality assessment:**

The IRS dataset has different scales being used, thus some standardization of the data needs to be done for analysis. The IRS dataset has no null values.

Data Preparation

Objective 1: Predict Company Response to a Consumer Complaint

Part - I (Company response considering CFPB complaints dataset only)

For this objective, the complaint dataset from the CFPB is utilized to predict Company response to a Complaint. The target variable of interest “Company Response to Consumer” has 7 categories. The analysis has been narrowed down to mortgage product complaints with issues related to “Loan modification, collection, foreclosure” and “Loan servicing, payments, escrow account”. To simplify the model building, the categorical variables except for the target variable are converted to numerical variables. Only complete cases were considered for the analysis, as the some of the Company Responses for the Consumer Complaint were missing in the dataset. Since, many of the variables were measured on different scales, the values on the merged dataset were scaled by normalizing each non-categorical variable. For the analysis, the 80% of the data was divided into training set and 20% of the data was divided into the test set. For model validation, K-fold cross validation technique was used, where the original sample is randomly partitioned into k equal sized subsamples. Of the subsamples, one sample is chosen as validation data and the rest k-1 samples are used as training data, until all folds are fully utilized for training and testing the analysis.

Part - II (Company response to complaint after integrating IRS dataset)

For this objective, the complaint dataset was merged with IRS dataset to determine if adding the tax return data will improve the predicting accuracy of the Company Response. A pivot table was created to make 6 different Annual Gross Income categories a separate variable. All categories inside the complaint dataset were number coded for analysis. Only complete cases were considered for the analysis, as the some of the Company Responses for the Consumer Complaint were missing in the dataset. The two datasets were merged by zip codes. For the merge, only those rows of records, where the zip codes were present at both the complaint dataset and the IRS dataset were used. Since, many of the variables were measured on different scales, the values on the merged dataset were scaled by normalizing each non-categorical variable. For the analysis, the 80% of the data was divided into training set and 20% of the data was divided into the test set. For model validation, K-fold cross validation technique was used, where the original sample is randomly partitioned into k equal sized subsamples. Of the subsamples, one sample is chosen as validation data and the rest k-1 samples are used as training data, until all folds are fully utilized for training and testing the analysis.

Objective 2: Predicting the Seriousness of Complaints

Seriousness of a complaint is measured based on whether the complaint costs monetary or non-monetary loss to the institution, i.e. fault of the company. Such responses include “Closed with monetary relief”, “Closed with non-monetary relief” and “Closed with relief”; and are categorized as “Company_fault” response. Similarly, the responses “Closed”, “Closed with explanation”, “Closed without relief” are identified as complaint raised due to the misunderstanding of the consumer and are not of any serious concern to the institution and hence categorized as “Consumer_fault”.

The missing values identified in the demographics dataset have been ignored or excluded from the dataset, with no scope to obtain dataset and avoid the risk of erroneously simulating or imputing demographic values for ZIP codes in the dataset. The demographics data has been standardized on a common scale. In order to understand the ZIP codes demographics as a group and with intention to reduce the observations, cluster analysis is performed on the demographics dataset and 3 different clusters are identified to classify the observations or ZIP codes as per the scree plot in figure: A17

In appendix. Different clusters with different cluster sizes has been presented in figure: A18 in appendix.

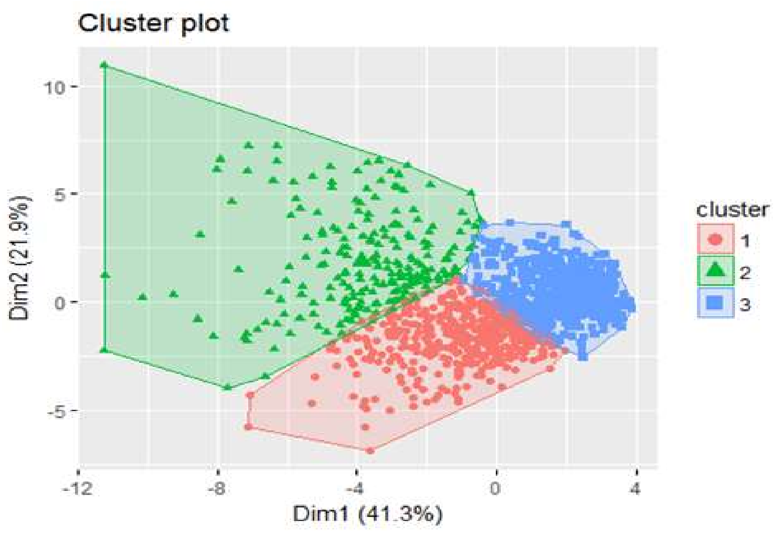
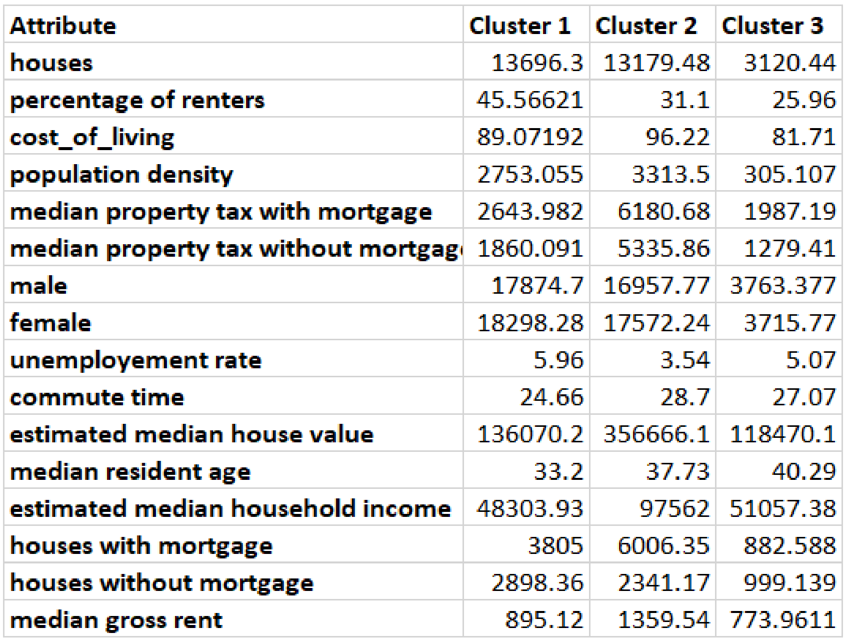


Figure 2: Cluster representation for size or number of clusters 3

Table 3: Cluster attributes



Attribute	Cluster 1	Cluster 2	Cluster 3
houses	13696.3	13179.48	3120.44
percentage of renters	45.56621	31.1	25.96
cost_of_living	89.07192	96.22	81.71
population density	2753.055	3313.5	305.107
median property tax with mortgage	2643.982	6180.68	1987.19
median property tax without mortgag	1860.091	5335.86	1279.41
male	17874.7	16957.77	3763.377
female	18298.28	17572.24	3715.77
unemployement rate	5.96	3.54	5.07
commute time	24.66	28.7	27.07
estimated median house value	136070.2	356666.1	118470.1
median resident age	33.2	37.73	40.29
estimated median household income	48303.93	97562	51057.38
houses with mortgage	3805	6006.35	882.588
houses without mortgage	2898.36	2341.17	999.139
median gross rent	895.12	1359.54	773.9611

From table 3 it can be observed that,

Cluster 1, represents an ***unsettled community*** with more renters, more unemployment rate, less taxes, mediocre population density and cost of living.

Cluster 2, represents a ***community with more working community***, as this cluster has more houses, less renters, more cost of living, high population density, less unemployment rate with more commute time, high house values and high gross rent. Also, this cluster has high number of houses with mortgage.

Cluster 3, represents a ***settled community with higher resident age*** group and less houses with mortgage, also has lower number of houses, less percent of renters, less cost of living, lower taxes and lower population density.

However, after embedding the cluster features with the complaints dataset, it did not significantly increase the accuracy to predict the target variables, i.e. company response to a complaint raised by the consumer and hence clusters features have been excluded for further analysis, same can be visualized in figure A19 in appendix.

The demographics dataset is integrated with the complaints dataset to understand the factors affecting the type of complaint being raised. The motive of using the demographic dataset is that, few of the economic attributes and regional attributes raise consumer complaints which are partly due to the misunderstandings of the consumer. Also, in some instances some of the company faults arise more in complaints with particular economic or demographic attributes. For example, in a rural community a complaint may be raised due to sheer misunderstanding of the issue by the consumer, while in a metropolitan area an error often occurs due to heavily overburdened employee. In order to understand the importance of such factors, consumer complaints dataset has been integrated with the demographics dataset using ZIP code as key variable.

As mentioned, the target variable is categorized into 2 categories, namely "Consumer_fault" and "Company_fault". Also, observing a seasonality in the company response in exploratory analysis, Month part is extracted from the Date of receiving the complaint and included as a new variable in the merged dataset. The dataset is narrowed to Texas state and mortgage product related "Loan modification, collection, foreclosure" and "Loan servicing, payments, escrow account" issues. The integrated dataset is partitioned into 80:20 proportion for training and test datasets, to develop a model using the training dataset and validate or test the model built using the test dataset. ZIP code has been excluded for model building, as this variable will nullify or dominate the other demographic attributes integrated into the complaints dataset. After observing the correlation between the variables of interest in table A4 in appendix, variables like female, houses, unemployment, etc. have been excluded from further model building to mitigate the issue of multi-collinearity.

Objective 3: Predict mortgage issues based on Personal Income Tax data

The objective is to determine if the issues could be accurately predicted by adding tax return data features. The target variable of interest is the Issues variable, which had 6 categories. As for Objective 1, Part 2 of the analysis, the complaint and IRS datasets were used for the analysis where the IRS dataset was pivoted to include 6 AGI groups as variables in the analysis. The complaint dataset was merged with the IRS dataset, as has been done for Objective 1, Part 2 of the analysis. Since, many of the variables are measured on different scales, the values on the merged dataset has been scaled by normalizing each non-categorical variable. For the analysis, the 80% of the data has been divided into training set and 20% of the data was divided into the test set. For model validation, K-fold cross validation technique has been used, where the original sample is randomly partitioned into k equal sized subsamples.

Modeling

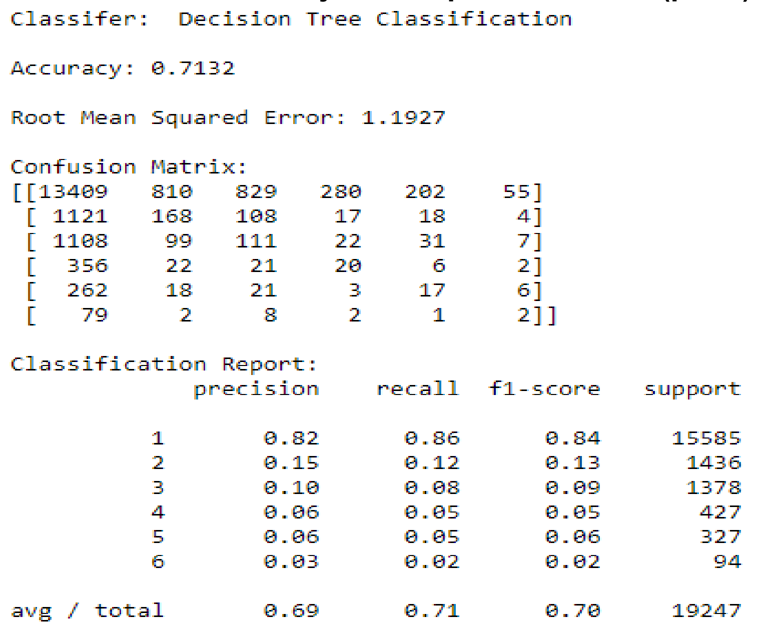
Objective 1: Predict Company Response to Consumer Complaints

For this objective Decision Tree Classifier and Random Forest Classifiers has been used. Decision tree classifier is technique in which a root is split into various sub-nodes or leaf. The population or sample is split into two or more homogeneous sets (or sub-populations) based on most significant splitter. The fundamental idea of Random Forest is to combine many decision trees into a single model. In this algorithm, many trees are grown as opposed to a single decision tree while modeling. To classify a new object based on attributes, each tree gives a classification and there is “votes” for that class. The forest chooses the classification having the most votes (over all the trees in the forest). Both Decision Tree algorithm and Random Forest algorithm works satisfactory for classification of categorical variables.

GridSearchCV from the Scikit Learn package from python has been used to hyper-tune the parameters. A set of parameters specific to each decision tree classifier and random forest classifier has been used. For Decision Tree classification, the parameter criterion is used to classify the data to split the root into different nodes based on the Gini index (it measures impurity) or the entropy (it measures information gain). For Random forest classifier, the parameters n_estimators(it is the number of trees in the forest) and criterion which is how to split the root into its respective nodes has been used. Then, GridSearchCV function is used which methodically builds and evaluates a model for each combination of algorithm parameters specified in a grid and give out the best estimator for the parameters supplied. Fivefold cross validation technique has been used to hyper-tune the parameters.

Confusion matrix or model summary statistics

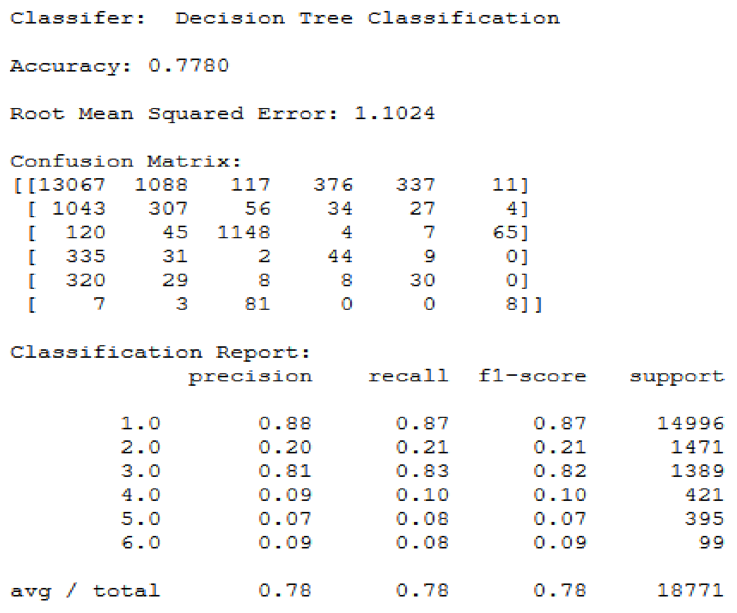
- **Confusion matrix for models with just Complaints dataset (part I)**



```
Classifer:  Decision Tree Classification

Accuracy: 0.7132

Root Mean Squared Error: 1.1927

Confusion Matrix:
[[13409   810   829   280   202    55]
 [ 1121   168   108    17    18     4]
 [ 1108    99   111    22    31     7]
 [  356    22    21    20     6     2]
 [  262    18    21     3    17     6]
 [   79     2     8     2     1     2]]

Classification Report:
             precision    recall  f1-score   support

          1       0.82      0.86      0.84     15585
          2       0.15      0.12      0.13      1436
          3       0.10      0.08      0.09      1378
          4       0.06      0.05      0.05       427
          5       0.06      0.05      0.06       327
          6       0.03      0.02      0.02        94

avg / total       0.69      0.71      0.70     19247
```

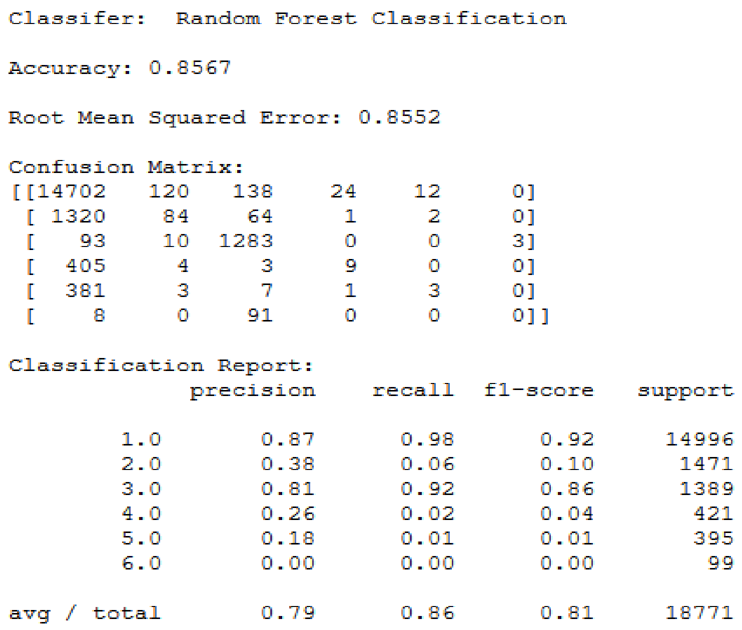
From the above confusion matrix, it can be observed that, accuracy is 71.32 %, average recall is 71 %, and average precision is 69 %.

Confusion matrix for models with merged Complaints and IRS datasets (part II)



```
Classifer:  Decision Tree Classification

Accuracy: 0.7780

Root Mean Squared Error: 1.1024

Confusion Matrix:
[[13067  1088   117   376   337    11]
 [ 1043   307    56    34    27     4]
 [  120    45  1148     4     7    65]
 [  335    31     2    44     9     0]
 [  320    29     8     8    30     0]
 [    7     3    81     0     0     8]]

Classification Report:
             precision    recall  f1-score   support

        1.0       0.88      0.87      0.87     14996
        2.0       0.20      0.21      0.21      1471
        3.0       0.81      0.83      0.82      1389
        4.0       0.09      0.10      0.10       421
        5.0       0.07      0.08      0.07       395
        6.0       0.09      0.08      0.09        99

avg / total       0.78      0.78      0.78     18771
```

From the above confusion matrix, it can be observed that, accuracy is 77.80 %, average recall is 78 %, and average precision is 78 %.



```
Classifer:  Random Forest Classification

Accuracy: 0.8567

Root Mean Squared Error: 0.8552

Confusion Matrix:
[[14702   120   138    24    12     0]
 [ 1320    84    64     1     2     0]
 [   93    10  1283     0     0     3]
 [  405     4     3     9     0     0]
 [  381     3     7     1     3     0]
 [    8     0    91     0     0     0]]

Classification Report:
             precision    recall  f1-score   support

        1.0       0.87      0.98      0.92     14996
        2.0       0.38      0.06      0.10      1471
        3.0       0.81      0.92      0.86      1389
        4.0       0.26      0.02      0.04       421
        5.0       0.18      0.01      0.01       395
        6.0       0.00      0.00      0.00        99

avg / total       0.79      0.86      0.81     18771
```

From the above confusion matrix, it can be observed that, accuracy is 85.67 %, average recall is 86 %, and average precision is 79 %.

For Objective 1, Random Forest Classifier from part II performs better than the other models. Confusion matrix for the Random Forest Classifier gives a much-detailed representation of what is going on with the labels. There are 14966 labels for ‘Closed with explanation’. Out of these, the model was successful in identifying 14702 labels correctly as ‘Closed with explanation. Similarly, of the 1471 labels for ‘Closed with non-monetary relief’, the classifier was able to correctly predict only 84 of the ‘Closed with non-monetary relief successfully. Of the 1389 labels for ‘Closed without relief’, the classifier was able to predict 1283 labels for ‘Closed with relief’ accurately. Of the 421 labels for ‘Closed’ the classifier was able to correctly predict 9 of the labels correctly. Of the 395 labels for ‘Closed with monetary relief’ the classifier was able to correctly predict only 3 of the labels correctly. Finally of the 99 labels for ‘Closed with relief’ the classifier was able to predict none of the labels correctly.

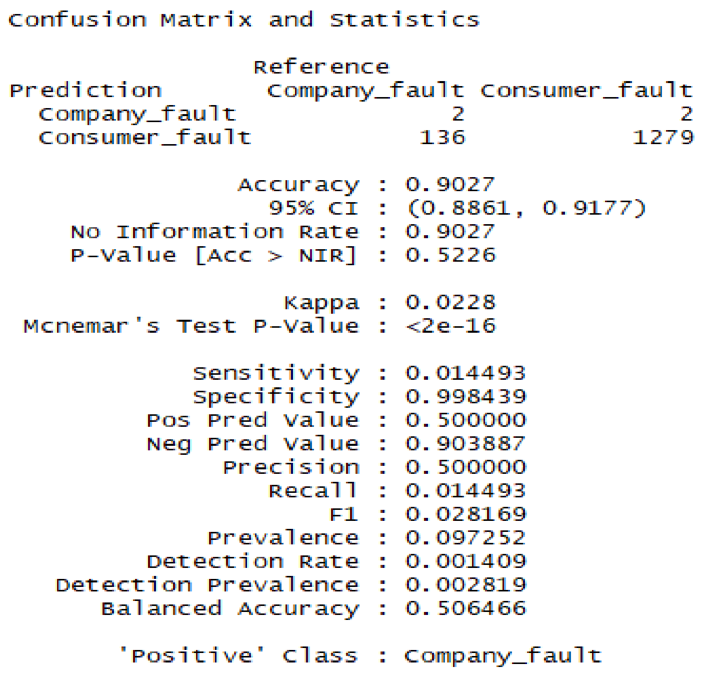
By comparing the Part I and Part II models, integrating the IRS dataset has significantly improved the model performance to predict the company’s response to a complaint. From Random Forest classification, feature importance scores were taken out. From the table A5 in appendix the top 5 scores revealed that during classification- year, month, zip code, aggregated adjusted annual gross income of people making $200k and above, aggregated adjusted annual gross income of people making below $25k, and number of tax returns of unemployed people inside a zip code were most important for creating trees with homogeneous group of nodes for the Random Forest classifier. However, using only the complaints dataset, variable importance has presented in table A6 in appendix. Using only the complaint dataset to predict the company response, ZIP code, Company and Sub-Product represent significant effect on the target variable. After integrating the IRS dataset with complaints dataset, we observed that the variables Company, ZIP continue to represent significant importance in predicting company response. Additionally, most of the variables from the integrated IRS dataset like unemployment, dependents, single returns have equally and significantly contributed to the model in predicting of target variable.

Objective 2: Classify the seriousness of a complaint based on demographic data

The target variable of this objective has two categorical values. Classification algorithms like Decision Tree classification, SVM and nnet have been used in R. Initially, Logistic Regression was applied, however, decision tree has been utilized for most of the model building, which provides better classification rules. The target variable is unbalanced, having a huge difference in distribution ratio. Hence, sampling techniques like Over sampling, under sampling, both sampling have been adopted.

- **Model performance comparison (confusion matrix)**

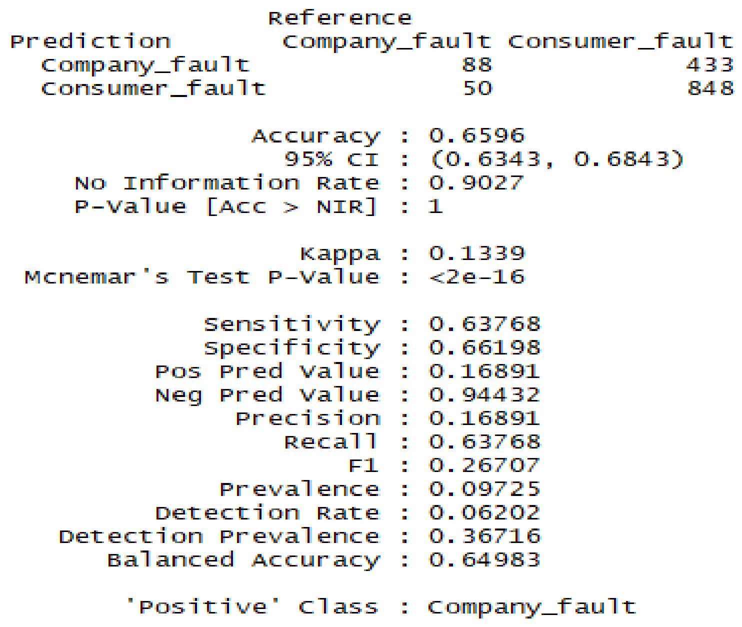
Confusion Matrix and Statistics for Decision tree (with unbalanced data):



```
Confusion Matrix and Statistics

                 Reference
Prediction        Company_fault Consumer_fault
  Company_fault               2              2
  Consumer_fault            136           1279

               Accuracy : 0.9027
                 95% CI : (0.8861, 0.9177)
    No Information Rate : 0.9027
    P-Value [Acc > NIR] : 0.5226

                  Kappa : 0.0228
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.014493
            Specificity : 0.998439
         Pos Pred Value : 0.500000
         Neg Pred Value : 0.903887
              Precision : 0.500000
                 Recall : 0.014493
                     F1 : 0.028169
             Prevalence : 0.097252
         Detection Rate : 0.001409
   Detection Prevalence : 0.002819
      Balanced Accuracy : 0.506466

       'Positive' Class : Company_fault
```

From the above confusion matrix, it can be observed an accuracy of 90%, which is actually an accuracy paradox as observed in the confusion matrix. Most of the predictions for target variable is Consumer_fault, which actually represents majority proportion of response in the dataset. Hence to mitigate this issue, models are developed after balancing the target variable in training dataset.

Confusion matrix for Over sampled train data



```
                 Reference
Prediction        Company_fault Consumer_fault
  Company_fault              88            433
  Consumer_fault             50            848

               Accuracy : 0.6596
                 95% CI : (0.6343, 0.6843)
    No Information Rate : 0.9027
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1339
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.63768
            Specificity : 0.66198
         Pos Pred Value : 0.16891
         Neg Pred Value : 0.94432
              Precision : 0.16891
                 Recall : 0.63768
                     F1 : 0.26707
             Prevalence : 0.09725
         Detection Rate : 0.06202
   Detection Prevalence : 0.36716
      Balanced Accuracy : 0.64983

       'Positive' Class : Company_fault
```

Performance metrics comparison for models built:

Table 4: Performance Statistics

Model	Balancing	Accuracy	Sensitivity	Specificity	AUC
Decision Tree	Unbalanced	90.2	1.44	99.84	50.6
	Over Sampling	66	64	66	65
	Under Sampling	46	77.53	42.7	60
	ROSE	61	67	60	64
SVM	Unbalanced	90	0	100	50
	Over Sampling	77	35	82	59
Nnet	Over Sampling	62	47	64	55.6

Tradeoffs have been made on accuracy of the model, as the key business requirement is to predict the Company_fault with high accuracy, i.e the complaints lead to financial payoff. From the models built, specifically from decision tree models using over sampled train data, following are the relatively more important features responsible for a particular type of the response to a complaint.

From the table A7 in appendix, "Company" variable has a significant effect on the target variable. Company response has some seasonal effect, it seems in certain months, the institutions may or may not respond with their own fault with a fear of losing value on the financial statements. From the demographics, perspective, percentage of renters in a ZIP code, median resident age, and commute time to work are somehow important to predict company response. For e.g., a person who has long commute time, will hardly find time to complaint, even if he or she complaints, it may not be a right representation of the issue and hence fails to get the complaint responded with some relief. Similarly, as observed in the exploratory analysis, gender doesn't really affect the company's response to a complaint.

Graphical depictions of models and results

The decision tree algorithm classified the responses, i.e target variable according to the following rules as represented visually as well. The target variable is initially classified using the Company variable, followed by the median resident age, commute time, percent of renters and month of receiving the complaint. These variables hold high importance in classifying the target variable as discussed in the importance of variables table.

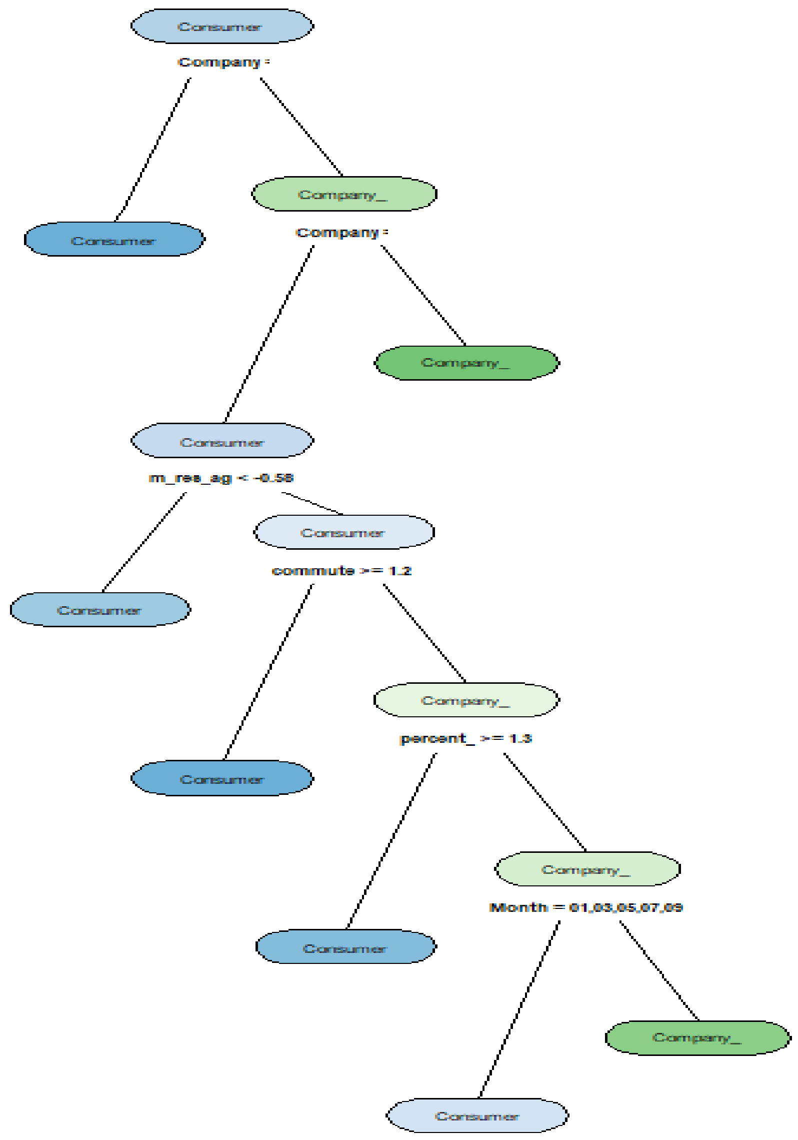


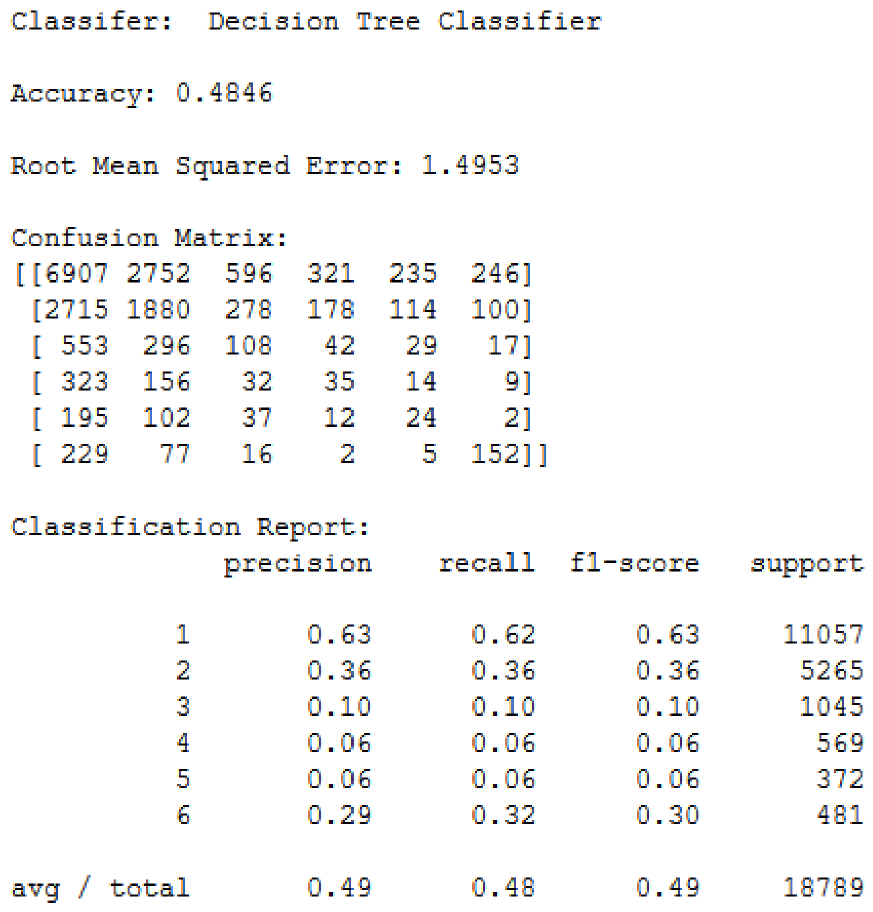
Figure 3: Decision Tree classification Plot

Objective 3: Predict mortgage issues based on Personal Income Tax data

For this objective Decision Tree Classifier and Random Forest Classifiers have been used. The Issues variable has 6 categorical responses. Both Decision Tree algorithm and Random Forest Algorithm works best when a classification of categorical variables. Similar to Objective

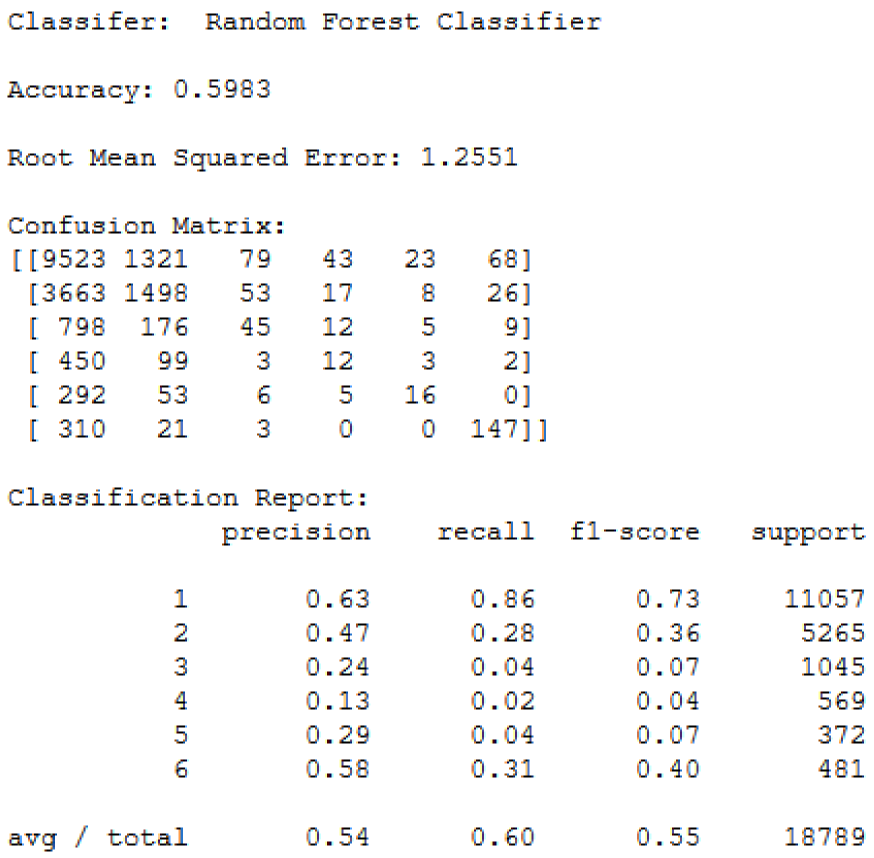
1, Part 2, GridSearchCV from the Scikit Learn package has been used to hyper-tune the parameters. For Decision Tree classification, the parameter criterion has been used to classify the data to split the root into different nodes based on the Gini index (it measures impurity) or the entropy (it measures information gain). For Random forest classifier, the parameters n_estimators(it is the number of trees in the forest) and criterion which is how to split the root into its respective nodes has been used. Fivefold cross validation technique has been used to hyper-tune the parameters.

- **Confusion matrix for each model used**



```
Classifer:  Decision Tree Classifier

Accuracy: 0.4846

Root Mean Squared Error: 1.4953

Confusion Matrix:
[[6907 2752  596  321  235  246]
 [2715 1880  278  178  114  100]
 [ 553  296  108   42   29   17]
 [ 323  156   32   35   14    9]
 [ 195  102   37   12   24    2]
 [ 229   77   16    2    5  152]]

Classification Report:
             precision    recall  f1-score   support

          1       0.63      0.62      0.63     11057
          2       0.36      0.36      0.36      5265
          3       0.10      0.10      0.10      1045
          4       0.06      0.06      0.06       569
          5       0.06      0.06      0.06       372
          6       0.29      0.32      0.30       481

avg / total       0.49      0.48      0.49     18789
```

From the above confusion matrix, the accuracy for Decision Tree model is 48.46%, precision is 48%, and F1 score is 49%.



```
Classifer:  Random Forest Classifier

Accuracy: 0.5983

Root Mean Squared Error: 1.2551

Confusion Matrix:
[[9523 1321   79   43   23   68]
 [3663 1498   53   17    8   26]
 [ 798  176   45   12    5    9]
 [ 450   99    3   12    3    2]
 [ 292   53    6    5   16    0]
 [ 310   21    3    0    0  147]]

Classification Report:
             precision    recall  f1-score   support

          1       0.63      0.86      0.73     11057
          2       0.47      0.28      0.36      5265
          3       0.24      0.04      0.07      1045
          4       0.13      0.02      0.04       569
          5       0.29      0.04      0.07       372
          6       0.58      0.31      0.40       481

avg / total       0.54      0.60      0.55     18789
```

From the above confusion matrix accuracy to predict issues has been increased to 59.83% using random forest model. The precision is 54%, recall is 60% and F1 score is 55% for the same.

Comparing the models for Objective 3, Random Forest Classifier performs better than the other models. Looking into the confusion matrix, it can be observed that of the 11057 labels for ‘Loan modification,collection,foreclosure’ the classifier correctly predicted 9523 of this label, of the 5265 labels for 'Loan servicing, payments, escrow account'- the classifier correctly classified 1498 of this label correctly, of the 1045 labels for 'Application, originator, mortgage broker'- the classifier correctly classified 45 of this label, of the 569 labels for 'Settlement process and costs'- the classifier correctly classified 12 of this label, of the 372 labels for 'Credit decision / Underwriting' the classifier correctly classified 16 labels correctly, and of the 481 labels for 'Other' the classifier correctly classified 147 of this label.

From the table: A8 in appendix, for Random Forest classification top 5 important features are - month, zip code, aggregated adjusted annual gross income of people making $200k and above, number of tax returns with dependents, and aggregated adjusted annual gross income of people making below $25k for creating trees with homogeneous group of nodes.

Evaluation

Objective 1: Predict Company Response to Consumer Complaints

From a regulatory point of view, it is important to predict Company Responses to Consumer Complaints correctly, i.e. correctly predict positives given the sample is actually positive. From the Random Forest classification for this Objective, the recall rate is about 86%, which tells that the model will accurately predict the Company Responses to Consumers 86% of the time based on their IRS Income Tax return data. Financial institutions can focus more on the complaints, which includes financial payoff, to improve customer service and work on the factors leading to financial payoff. CFPB can develop new policies to regulate and monitor the financial complaints. Also, it will help CFPB to categorize the complaints to take required action and to maintain market integrity.

Objective 2: Classify the seriousness of a complaint based on demographic data

As indicated in the confusion matrix for the decision tree model based on over sampled train data, the company fault complaints have been 66 percent correctly identified and hence could save resources and efforts in analyzing complaints. The consumer faults complaints predicted can be ignored or not scrutinized and be responded with a consumer fault. However, the company fault complaints predicted can be further analyzed or scrutinized to resolve the issue. Business value has been created by correctly predicting a company fault, which has to be identified by the earliest and scrutinized to respond back to the consumer to improve customer service. Also, predicting the company faults with higher precision, would avoid utilizing the precious resources to resolve the consumer faulty complaints. This not only improves the response time, but also efficiently resolves complaints raised in a smart manner.

As discussed month, commute time, percent of renters, and resident age being weighed significantly, so an institution can include such parameters in their mortgage loan offering process to avoid complaints. Also, may be for certain ZIP codes the complaints are raising due to customer misunderstanding, where the institution may make efforts in educating the consumers before actually doing business with them. Better predictions can further reduce the efforts or resources utilized in analyzing responses.

Objective 3: Predict mortgage issues based on Personal Income Tax data

From a consumer point of view, it is important to predict Mortgage Issues correctly, i.e. correctly predict positives given the sample is actually positive. From the Random Forest classification for this Objective, the recall rate is about 60%, which tells that the model will accurately predict 60% of the time the issues the customers will face given their tax return data and the company the customer is trying to go for various mortgage products. From a Consumer point of view, knowing what type of issues they are most likely to encounter with a given company given their tax return information will be a good knowledge to have. They can make wiser decision in choosing the best companies to do business with. Also, knowing what consumers from different zip codes with varying groups of people with different AGIs will face what type of mortgage related issues will make the consumers aware if any biases are being made by financial companies on their communities and the place where they live in.

From the importance scores about the zip codes, one can tell if some Issues like “Loan modification,collection,foreclosure” were a result of a recession or changes to the economy of a geographical area. Likewise, people making a lot of money and people making less money tells us that different groups of people may have different type of issues with their mortgages.

By making correct predictions about the Issues, consumers can make future predictions about what type of Issues will the consumer likely encounter when they use mortgage services. This will aware consumers of the Issues they may encounter when they use a mortgage service from a financial institution and if the companies are being biased towards any particular group of people based on how much they make or where they are from.

Deployment

Deployable Results Summary

Three datasets, i.e. complaints dataset from CFPB, IRS dataset and web-scraped Demographics dataset have been utilized. The datasets are integrated using the ZIP code as key variable and necessary data cleaning and pre-processing steps are performed. Models have been developed separately for 3 distinct business objectives, i.e. prediction of company response, seriousness or genuineness of the complaint along with issue prediction. The 3 best performing models as evaluated in the above sections will be deployed in the production environment considering business operation constraints.

Future scope

Predicting the probability of a complaint being raised based on the consumer's live transactional data and personal data, even before a complaint is lodged. For example, a mortgage product consumer's personal income data, transactional data can be integrated with the performance of the mortgage product account and the probability of a complaint about to be lodged can be predicted. Additional data of the complaints from the financial institution can be integrated with the complaints dataset, to improve the data richness.

The analysis can be expanded to different lines of business, i.e. complaints related to different kinds of products served by the financial institutions. Also, case analysis for each financial institution can be performed and included as a factor of variable in the analysis.

Potential challenges in gaining user acceptance of recommendations

The potential challenge of the current analysis can be gaining acceptance from the regulatory board, as the approach suggested is considering to resolve or prioritize only complaints which are at the fault of company and reduce the resource effort on the complaints with consumer misunderstandings. This issue can be mitigated by convincing the authority the value created by improving the response time to critical issues and agreeing to resolve the complaints predicted as consumer misunderstandings by sampling the complaints dataset and put efforts to improve accuracy and consistency of the prediction algorithms.

Potential decision-making heuristics and biases

Cognitive illusions arise due to the limited capacity of human processing of solving complex problems, as a result people use decision making rules of thumb, for example, a complaint from a rich community is treated as a fault of the company and a complaint of a less educated person is considered heuristically as a pure misunderstanding of the consumer. Such heuristics create biases and often lead to misjudgment.

Framing bias: Framing of a prediction summary, often leads to misguided decision making. For example, predicting 30 percent of issue being loan modification is likely to be responded differently than predicting 70 percent of issue not being loan modification.

Availability bias: The prevalence of media stories of a particular financial institution can create a bias that all the complaints raised against that institution are critical. However, this may not to be true case after resolving all the issues.

Anchor Bias: A certain level of prediction can be viewed and responded differently if the decision maker is unaware of the anchor levels. For example, a prediction of 10 percent genuine complaints in a particular ZIP code can be viewed serious, if the average percent of genuinely in that particular region is around 9-10 percent.

Information overload bias: It is assumed that by adding more data sources should help the prediction accuracy, however this may increase the complexity of building models and eventually increase the resources and computation power without gaining much improved knowledge.

Countermeasures such as ensuring forecasting analysts undertake debiasing to reduce overconfidence and availability biases, ensuring false or misleading media reports do not factor the model building, ensuring the prediction results are prepared in such a way to avoid missing out the base rates, and providing feedback to the decision makers and the forecasters to enable learning to take place to achieve accurate predictions will assist to reduce the biases.

Plan for monitoring, maintaining, updating models

The new complaints datasets will continuously be sampled to test the accuracy of the predictions and monitor the consistency of variable importance in the models. Also, possible additional data sources may be integrated to consider the wide range of aspects involved in analyzing the complaints. The models developed with repeated iterations of CRISP-DM phases, after receiving continuous feedback on the prediction models and with an improved accuracy, should be made available to the business users or decision makers with minimum understanding of data mining tools. The business users at regulatory board or financial institution may be provided with a Graphical user interface (sample provided in figure A20), that gives them the required attributes and statistical values to assist their decision making. Any feedback from the users or concerns being faced, need to be brought up with the model developers to make changes or improve model performance accordingly.

Deployment strategies

Data Repository: The data resources will be stored in a Database server and can be updated with new observations automatically or manually. The new complaints raised on the

CFPB data platform can be linked to the database to continuously update the complaints database without any manual intervention. However, the demographic dataset and income tax returns dataset can be stored in the Database and can be updated manually. The complaints data needs to be updated regularly, however, the demographics and income tax returns data can be updated annually into the databases. Tableau dashboard will be shared across the business users and decision makers through a server, so that the key performance indicators can be monitored continuously. A systematic feedback loop is created between the users and developers to optimize the prediction models and user interface.

Cost/benefit/ROI analysis/Value Capture

Predicting the responses provided by the financial institutions will assist in prioritize the complaints and improve the response time and reduce the amount of resources allocated to scrutinize the complaints received. Financial institutions can now focus more on the complaints, which includes financial payoff, to improve customer service and work on the factors leading to financial payoff. The analysis also facilitates the institutions to identify demographics that lead to serious complaints and can revise or add extra terms and conditions to offer services at particular demographic region to reduce the probability of raising serious complaints. Also, products and services terms and conditions can be revised and added to improve the product or service quality, based on analyzing the issues being encountered in these regions. This will assist the financial institutions in taking precautionary steps to avoid the issues raised for mortgage related products or services.

Conclusion

After conducting the secondary research of CFPB and financial institutions functionality, and evaluation of products or services offered to consumers, it is concluded that the undertaken project will add value to all the stakeholders. Pre-information of company response, seriousness of complaints, and potential issues, will assist consumers to take better decisions while opting for services, facilitate financial institutions to improve customer service and optimize usage of resources, and support CFPB to maintain integrity among the stakeholders. Deployment of models in real business operations, while considering the business constraints, will allow the stakeholders to function efficiently, however, continuous monitoring, improvement, and viability check of outcomes are necessary to make it a success.

References

CFPB Data Analysis

1. Fonseka, W. R., Nadeesha, D. G., Thakshila, P. M., Jeewandara, N. A., Wijesinghe, D. M., Sahabandu, R. V., & Asanka, P. P. (2016). Use of data warehousing to analyze customer complaint data of Consumer Financial Protection Bureau of United States of America. 2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS). doi:10.1109/iciafs.2016.7946520

Mortgage Issues

2. Kang, K. (2018). Mortgage Loan Prediction:Bayesian Machine Learning Approach. Financial Stability Studies, 19(1), 99-129. doi:10.26588/kdic.2018.19.1.004

Complaint Analysis

3. Garding, S., & Bruns, A. (2015). Analysis of Customers' Complaint Channel Choice and Complaint Behaviour. SpringerBriefs in Business Complaint Management and Channel Choice, 35-74. doi:10.1007/978-3-319-18179-0_4

Heuristics and Biases

4. Nicholls, N. (1999). Cognitive Illusions, Heuristics, and Climate Prediction. Bulletin of the American Meteorological Society, 80(7), 1385-1397. doi:10.1175/1520-0477(1999)0802.0.co;2

5. Heuristics and Biases in the Intuitive Projection of Retail Sales Author(s): Anthony D. Cox and John O. Summers Source: Journal of Marketing Research, Vol. 24, No. 3 (Aug., 1987), pp. 290-297

6. Doswell, C. A. (2004). Weather Forecasting by Humans—Heuristics and Decision Making. Weather and Forecasting,19(6), 1115-1126. doi:10.1175/waf-821.1

7. Goodwin, P., & Wright, G. (1994). Heuristics, biases and improvement strategies in judgmental time series forecasting. Omega, 22(6), 553-568. doi:10.1016/0305-0483(94)90047-7

Appendix

List of Tables

Table A1: CFPB consumer complaints data dictionary

FIELD NAME	DESCRIPTION
Date received	The date the CFPB received the complaint. For example, “05/25/2013.”
Product	The type of product the consumer identified in the complaint. For example, “Checking or savings account” or “Student loan.”
Sub-product	The type of sub-product the consumer identified in the complaint. For example, “Checking account” or “Private student loan.”
Issue	The issue the consumer identified in the complaint. For example, “Managing an account” or “Struggling to repay your loan.”
Sub-issue	The sub-issue the consumer identified in the complaint. For example, “Deposits and withdrawals” or “Problem lowering your monthly payments.”
Consumer complaint narrative	Consumer complaint narrative is the consumer-submitted description of “what happened” from the complaint. Consumers must opt-in to share their narrative. We will not publish the narrative unless the consumer consents and consumers can opt-out at any time. The CFPB takes reasonable steps to scrub personal information from each complaint that could be used to identify the consumer.

Company public response	The company’s optional, public-facing response to a consumer’s complaint. Companies can choose to select a response from a pre-set list of options that will be posted on the public database. For example, “Company believes the complaint is the result of an isolated error.”
Company	The complaint is about this company. For example, “ABC Bank.”
State	The state of the mailing address provided by the consumer.
ZIP code	The mailing ZIP code provided by the consumer. This field may: i) include the first five digits of a ZIP code; ii) includes the first three digits of a ZIP code (if the consumer consented to publication of their complaint narrative); or iii) be blank (if ZIP codes have been submitted with non-numeric values, if there are less than 20,000 people in a given ZIP code, or if the complaint has an address outside of the United States).
Tags	Data that supports easier searching and sorting of complaints submitted by or on behalf of consumers. For example, complaints, where the submitter reports the age of the consumer as 62 years or older, are tagged “Older American.” Complaints submitted by or on behalf of a servicemember or the spouse or dependent of a servicemember are tagged “Servicemember.” Servicemember includes anyone who is active duty, National Guard, or Reservist, as well as anyone who previously served and is a veteran or retiree.
Consumer consent provided?	Identifies whether the consumer opted in to publish their complaint narrative. We do not publish the narrative unless the consumer consents and consumers can opt-out at any time.

Submitted via	How the complaint was submitted to the CFPB. For example, “Web” or “Phone.”
Date sent to the company	The date the CFPB sent the complaint to the company.
Company response to consumer	This is how the company responded. For example, “Closed with an explanation.”
Timely response?	Whether the company gave a timely response. For example, “Yes” or “No.”
Consumer disputed?	Whether the consumer disputed the company’s response.
Complaint ID	The unique identification number for a complaint.

Table A2: Demographics data dictionary

VARIABLE NAME	**DESCRIPTION**
ZIP	Area Zip Code
houses	Number of houses in the ZIP code
cost_of_living	Cost of living index
m_prop_tax_w_mtg	Median real estate property taxes paid for housing units with mortgages (in dollar value)

m_prop_tax_w_out_mtg	Median real estate property taxes paid for housing units with no mortgage (in dollar value)
male	Number of Males
female	Number of Females
unemp	Unemployment rate (in percent)
commute	Mean travel time to work (in minutes)
est_m_house_val	Estimated median house/condo value (in dollar value)
m_res_age	Median resident age (in years)
est_m_house_income	Estimated median household income (in dollar value)
houses_w_mtg	Number of Housing units with a mortgage
houses_w_out_mtg	Number of Housing units without mortgage
m_gross_rent	Median gross rent (in dollar value)

Table A3: Statistics of Income (SOI) data dictionary (Income tax Return)

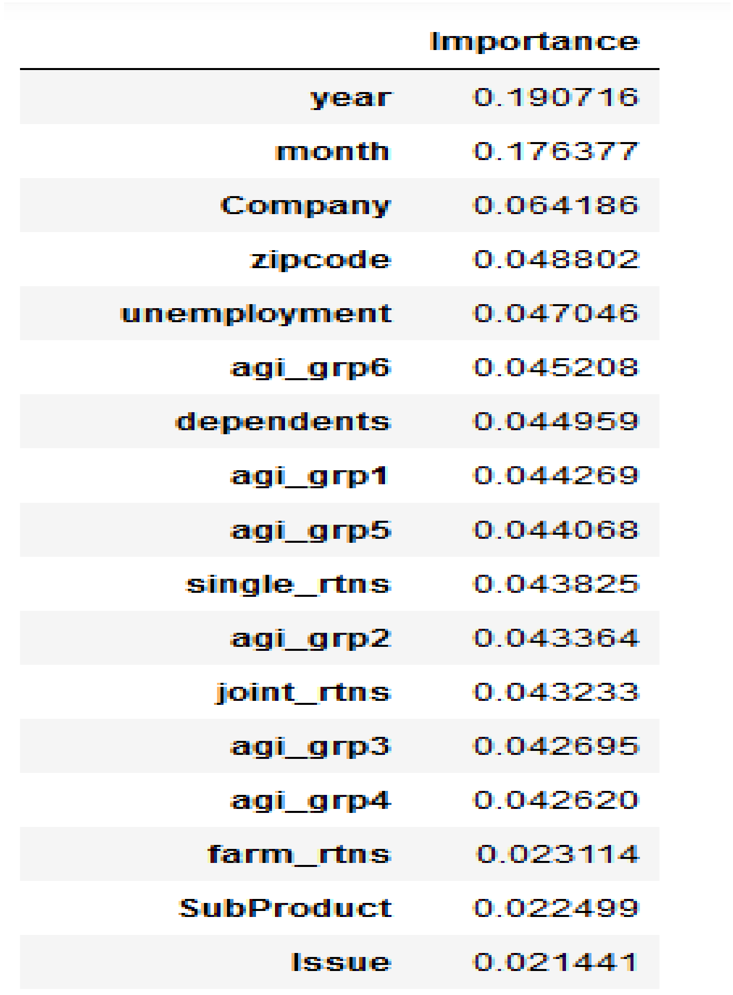
VARIABLE NAME	DESCRIPTION	VALUE/LINE REFERENCE
STATE	The State associated with the ZIP code	Two-digit State abbreviation code
ZIPCODE	5-digit Zip code	
AGI_STUB	Size of adjusted gross income	1 = $1 under $25,000 2 = $25,000 under $50,000 3 = $50,000 under $75,000 4 = $75,000 under $100,000 5 = $100,000 under $200,000 6 = $200,000 or more
N1	Number of returns	
MARS1	Number of single returns	Filing status is single
MARS2	Number of joint returns	Filing status is married filing jointly
MARS4	Number of head of household returns	Filing status is head of household
PREP	Number of returns with paid preparer's signature	
NUMDEP	Number of dependents	1040:6c
A00100	Adjust gross income (AGI) [2]	1040:37 / 1040A:21 / 1040EZ:4
SCHF	Number of farm returns	1040:18

e

Table A4: Correlation between Variables in Demographic dataset

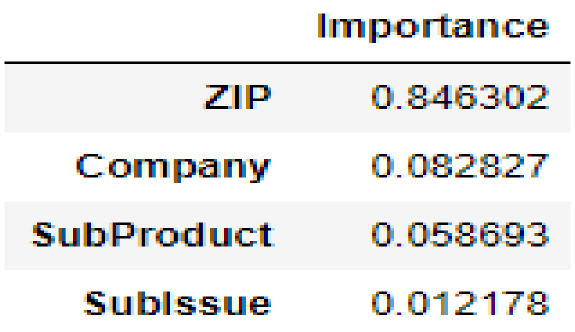
	houses	percent_r	cost_of_li	pop_dens	m_prop_t	m_prop_t	male	female	unemp	commute	est_m_ho	m_res_ag	est_m_ho	houses_w	houses_w	m_gross_r
houses	1	0.400478	0.486778	0.424206	0.339125	0.3434	0.966421	0.96793	-0.00069	-0.05586	0.231095	-0.37846	0.188045	0.847876	0.81096	0.370948
percent_r	0.400478	1	0.365023	0.500439	0.086255	0.087889	0.311189	0.297945	0.181736	-0.37524	0.041004	-0.50539	-0.33578	0.020976	0.121746	-0.028
cost_of_li	0.486778	0.365023	1	0.626997	0.595443	0.583064	0.455061	0.460513	-0.06993	0.167855	0.633411	-0.23554	0.424576	0.459375	0.222237	0.685769
pop_dens	0.424206	0.500439	0.626997	1	0.309042	0.267385	0.376037	0.380872	0.014073	-0.04784	0.302426	-0.26992	0.069655	0.248975	0.189032	0.318247
m_prop_t	0.339125	0.086255	0.595443	0.309042	1	0.935423	0.295312	0.296673	-0.34326	0.085613	0.835664	-0.03323	0.782511	0.414389	0.147968	0.712169
m_prop_t	0.3434	0.087889	0.583064	0.267385	0.935423	1	0.296801	0.301703	-0.33561	0.078012	0.839359	-0.02014	0.785129	0.430274	0.13955	0.722524
male	0.966421	0.311189	0.455061	0.376037	0.295312	0.296801	1	0.992226	0.01824	0.000287	0.173671	-0.43695	0.194792	0.879327	0.819637	0.371698
female	0.96793	0.297945	0.460513	0.380872	0.296673	0.301703	0.992226	1	0.024703	0.007432	0.178117	-0.42531	0.197814	0.891222	0.821584	0.37523
unemp	-0.00069	0.181736	-0.06993	0.014073	-0.34326	-0.33561	0.01824	0.024703	1	0.030985	-0.30282	-0.16922	-0.44519	-0.12302	0.080961	-0.28923
commute	-0.05586	-0.37524	0.167855	-0.04784	0.085613	0.078012	0.000287	0.007432	0.030985	1	0.06715	0.179917	0.245912	0.131445	-0.10445	0.223359
est_m_ho	0.231095	0.041004	0.633411	0.302426	0.835664	0.839359	0.173671	0.178117	-0.30282	0.06715	1	0.094506	0.721113	0.290004	0.076211	0.675709
m_res_ag	-0.37846	-0.50539	-0.23554	-0.26992	-0.03323	-0.02014	-0.43695	-0.42531	-0.16922	0.179917	0.094506	1	0.125812	-0.24391	-0.24637	-0.11256
est_m_ho	0.188045	-0.33578	0.424576	0.069655	0.782511	0.785129	0.194792	0.197814	-0.44519	0.245912	0.721113	0.125812	1	0.462184	0.085367	0.777884
houses_w	0.847876	0.020976	0.459375	0.248975	0.414389	0.430274	0.879327	0.891222	-0.12302	0.131445	0.290004	-0.24391	0.462184	1	0.680909	0.54655
houses_w	0.81096	0.121746	0.222237	0.189032	0.147968	0.13955	0.819637	0.821584	0.080961	-0.10445	0.076211	-0.24637	0.085367	0.680909	1	0.171735
m_gross_	0.370948	-0.028	0.685769	0.318247	0.712169	0.722524	0.371698	0.37523	-0.28923	0.223359	0.675709	-0.11256	0.777884	0.54655	0.171735	1

Table A5: Response Prediction Random Forest Variable Importance (part II)



	Importance
year	0.190716
month	0.176377
Company	0.064186
zipcode	0.048802
unemployment	0.047046
agi_grp6	0.045208
dependents	0.044959
agi_grp1	0.044269
agi_grp5	0.044068
single_rtns	0.043825
agi_grp2	0.043364
joint_rtns	0.043233
agi_grp3	0.042695
agi_grp4	0.042620
farm_rtns	0.023114
SubProduct	0.022499
Issue	0.021441

Table A6: Response Prediction Random Forest Variable Importance (part I)

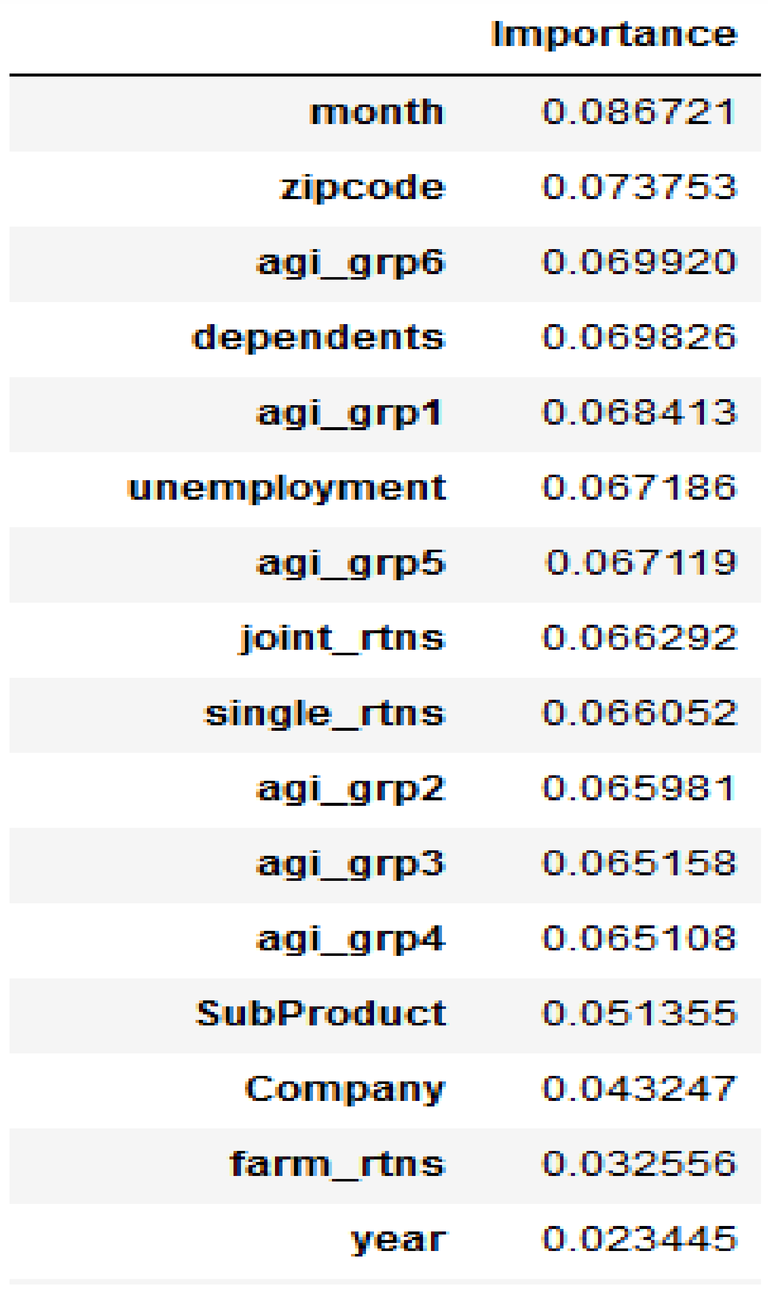


	Importance
ZIP	0.846302
Company	0.082827
SubProduct	0.058693
SubIssue	0.012178

Table A7: Seriousness Prediction Decision Tree over sampling Variable Importance

Variable	**Importance**
Company	760.53
Month	127.44
Issue	108.43
percent_rent	96.23
m_res_age	84.98
commute	57.50
pop_density	36.16
Sub.product	22.92
cost_of_living	17.43
m_gross_rent	13.40
houses_w_mtg	11.82
male	0.00
est_m_house_val	0.00
houses_w_out_mtg	0.00

Table A8: Mortgage Issue Prediction Random Forest Variable Importance



	Importance
month	0.086721
zipcode	0.073753
agi_grp6	0.069920
dependents	0.069826
agi_grp1	0.068413
unemployment	0.067186
agi_grp5	0.067119
joint_rtns	0.066292
single_rtns	0.066052
agi_grp2	0.065981
agi_grp3	0.065158
agi_grp4	0.065108
SubProduct	0.051355
Company	0.043247
farm_rtns	0.032556
year	0.023445

List of Figures

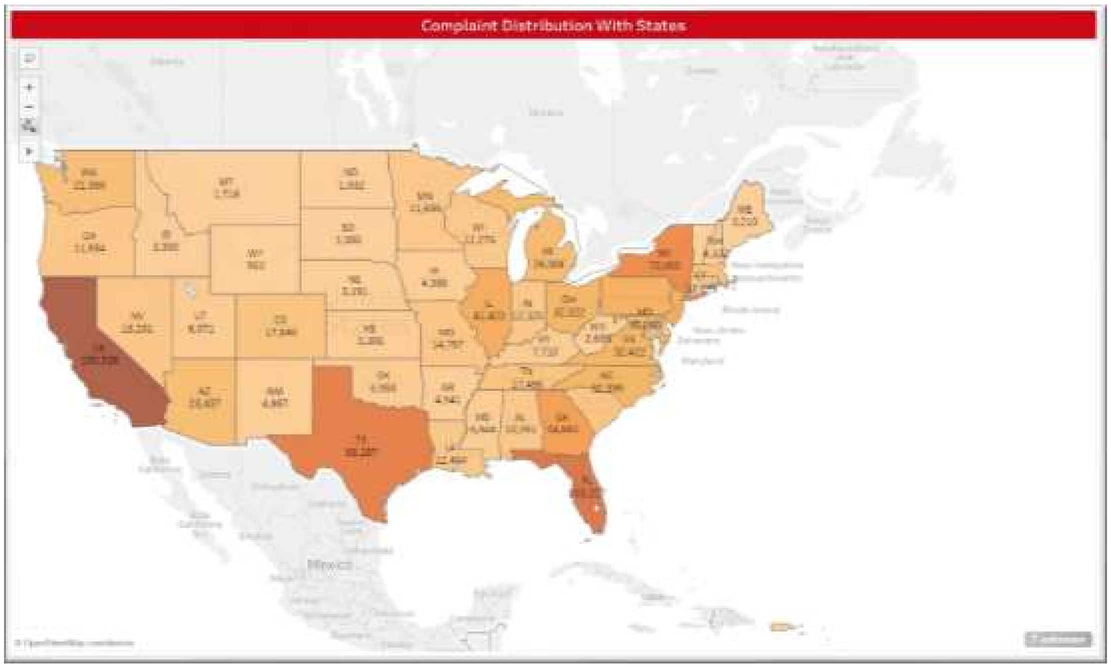


Figure A1: Distribution of Consumer Complaints throughout the Country.

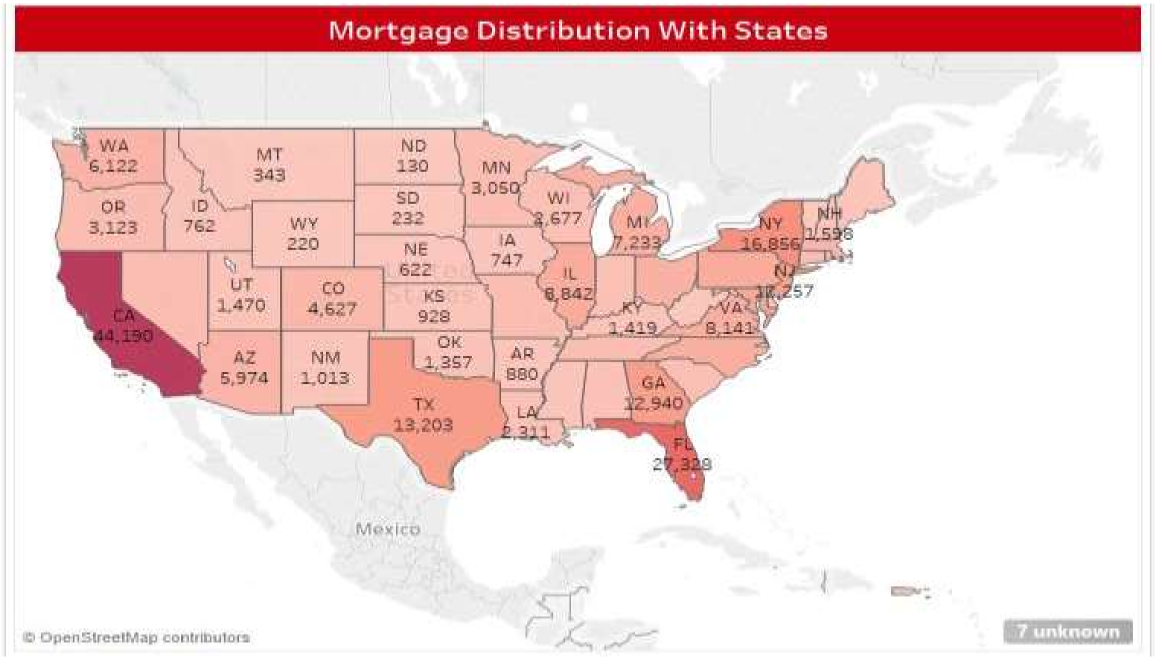


Figure A2: Distribution of Mortgage Consumer Complaints throughout the Country

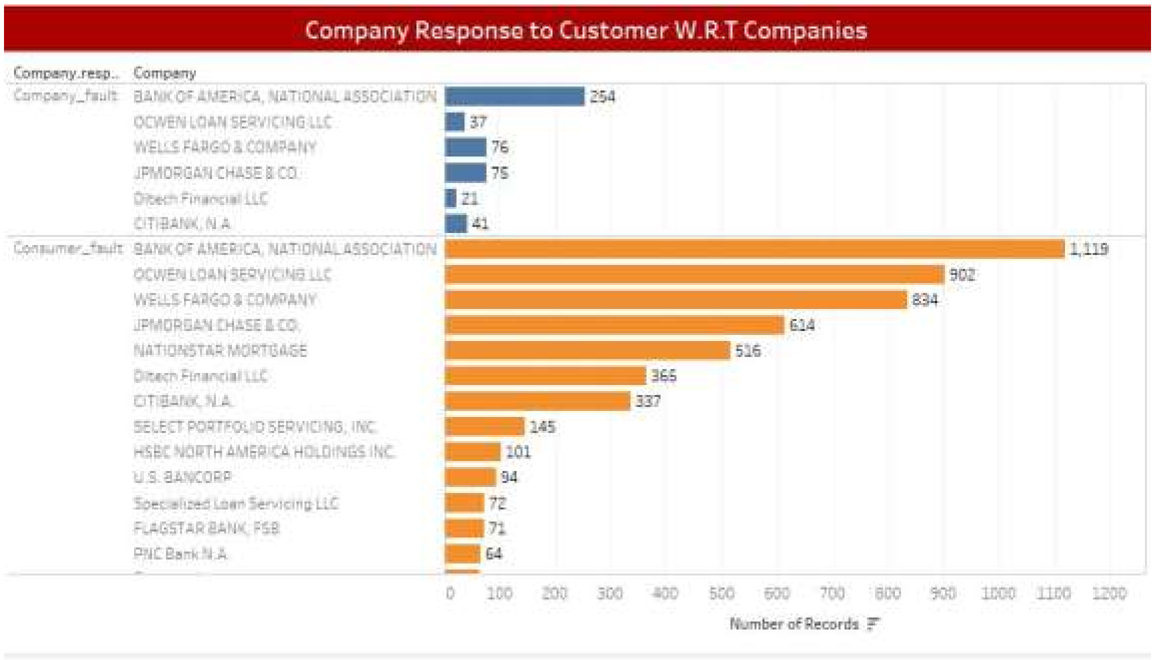


Figure A3: Company response to the customer varies with respect to Companies.

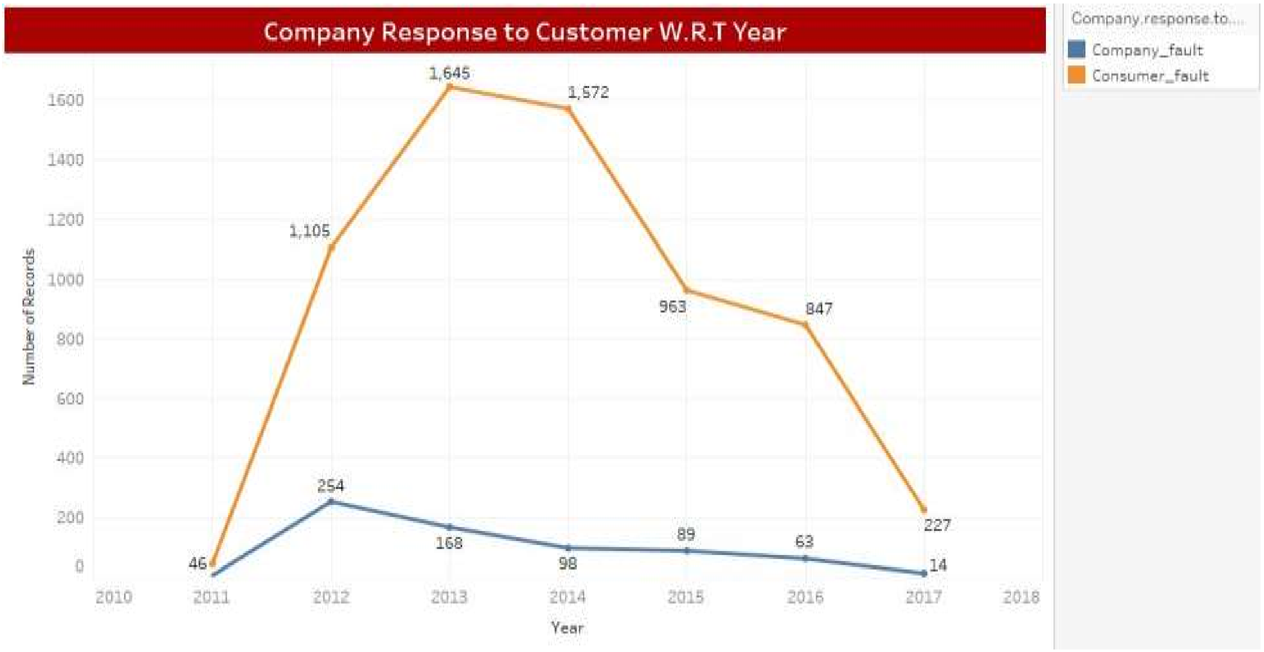


Figure A4: Company response to the customer varies with respect to different Year

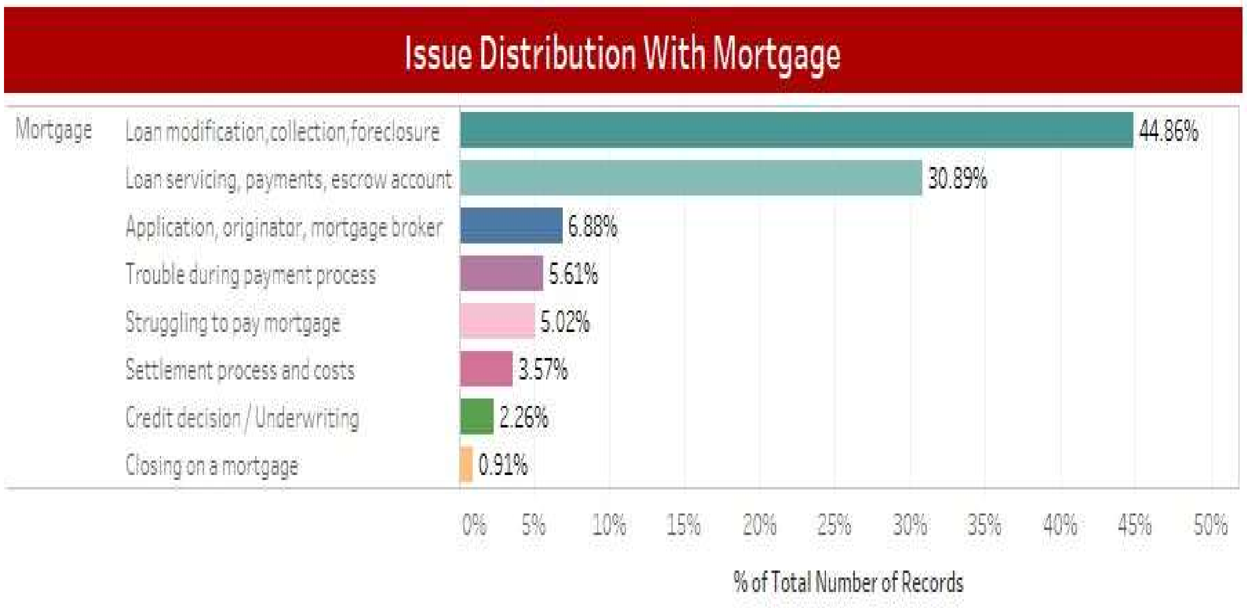


Figure A5: Distribution of issues of Mortgage Complaints.

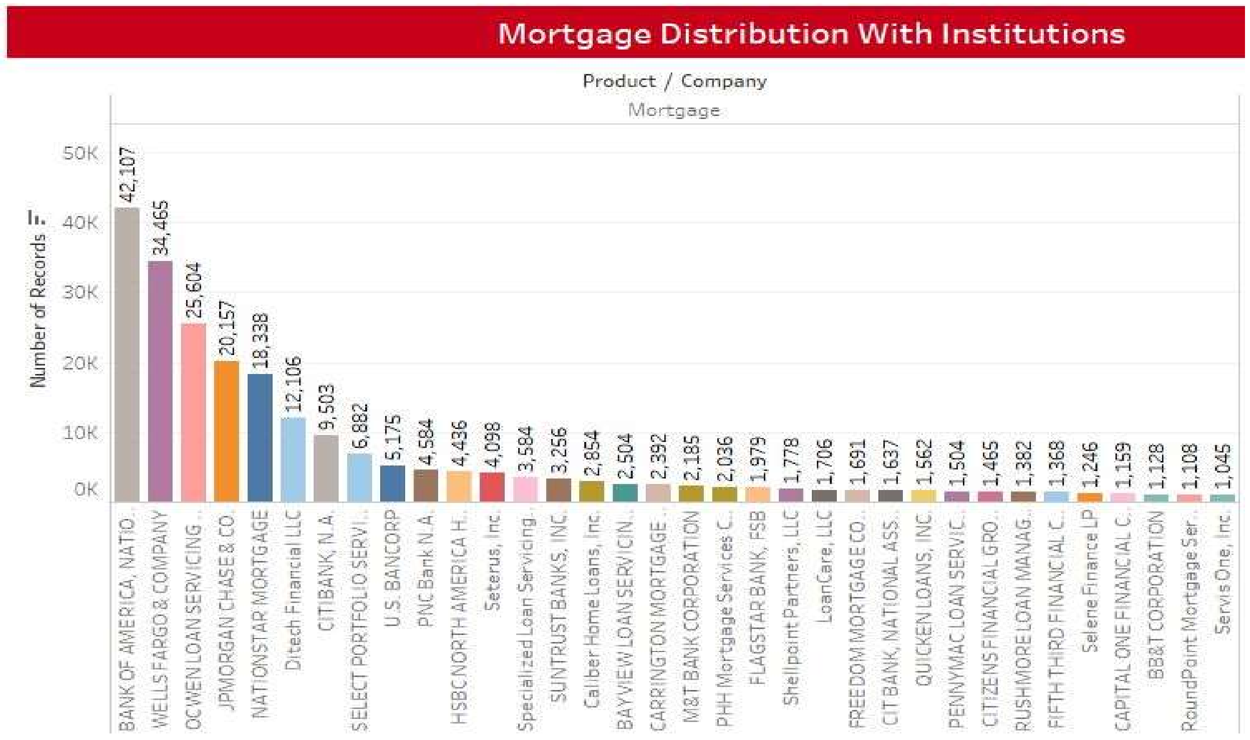


Figure A6: Distribution of Mortgage Complaints with respect to different institutions.

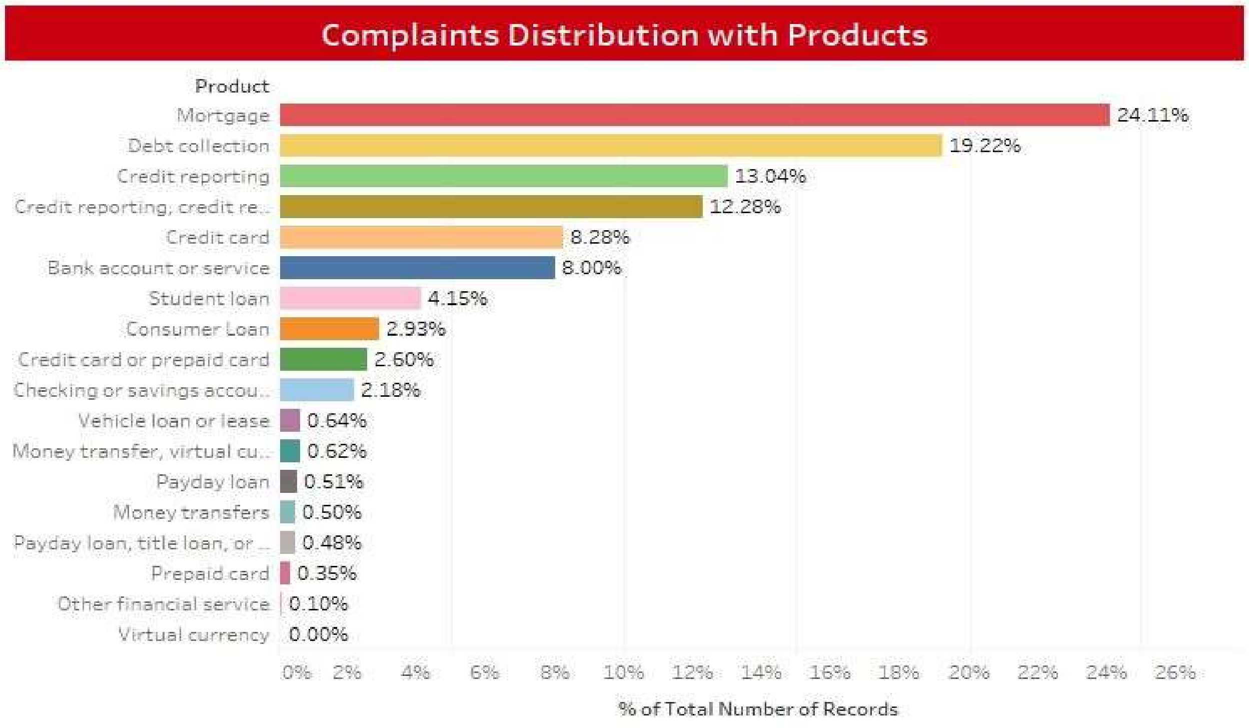


Figure A7: Distribution of Complaints with respect to different Products.

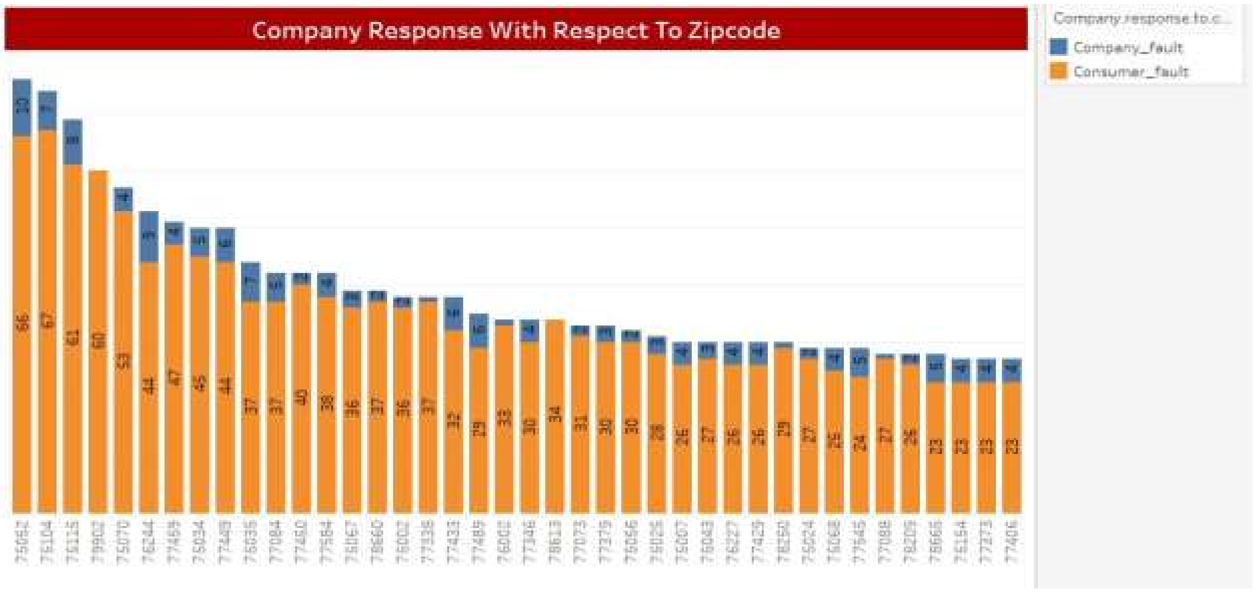


Figure A8: Variation in Company Response with respect to different zip codes

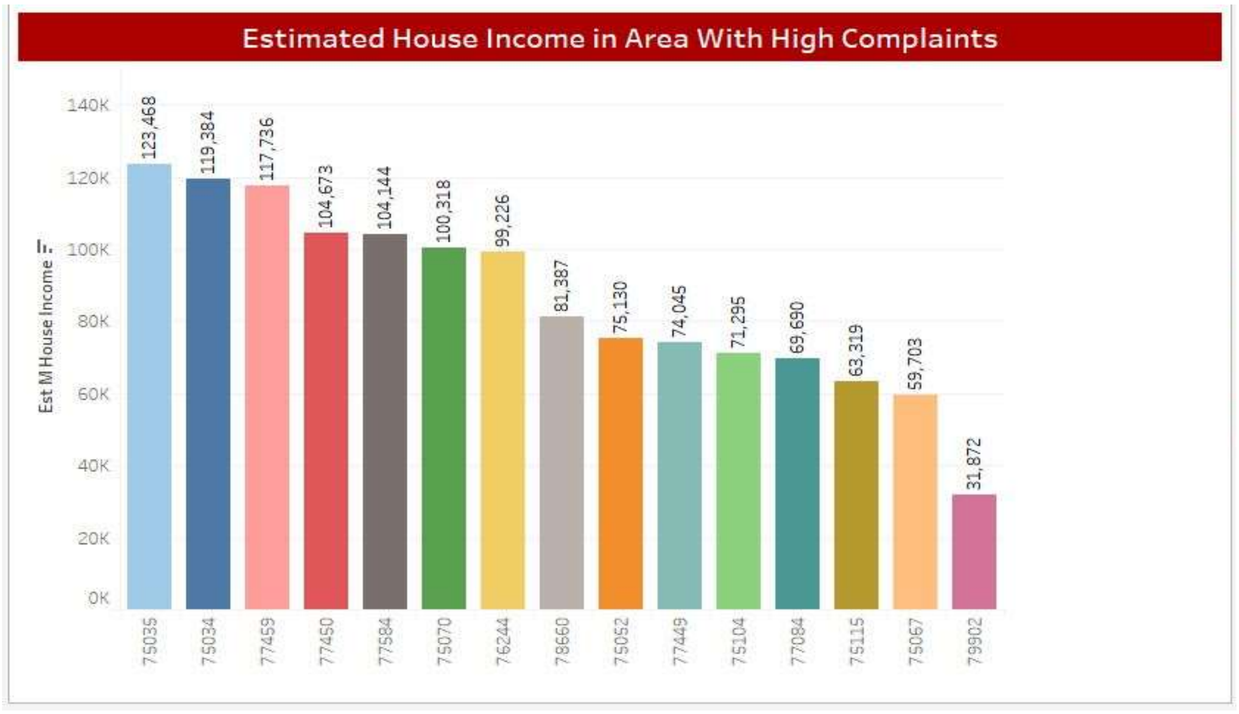


Figure A9: Estimated House Income Varies in Area with High Complaints.

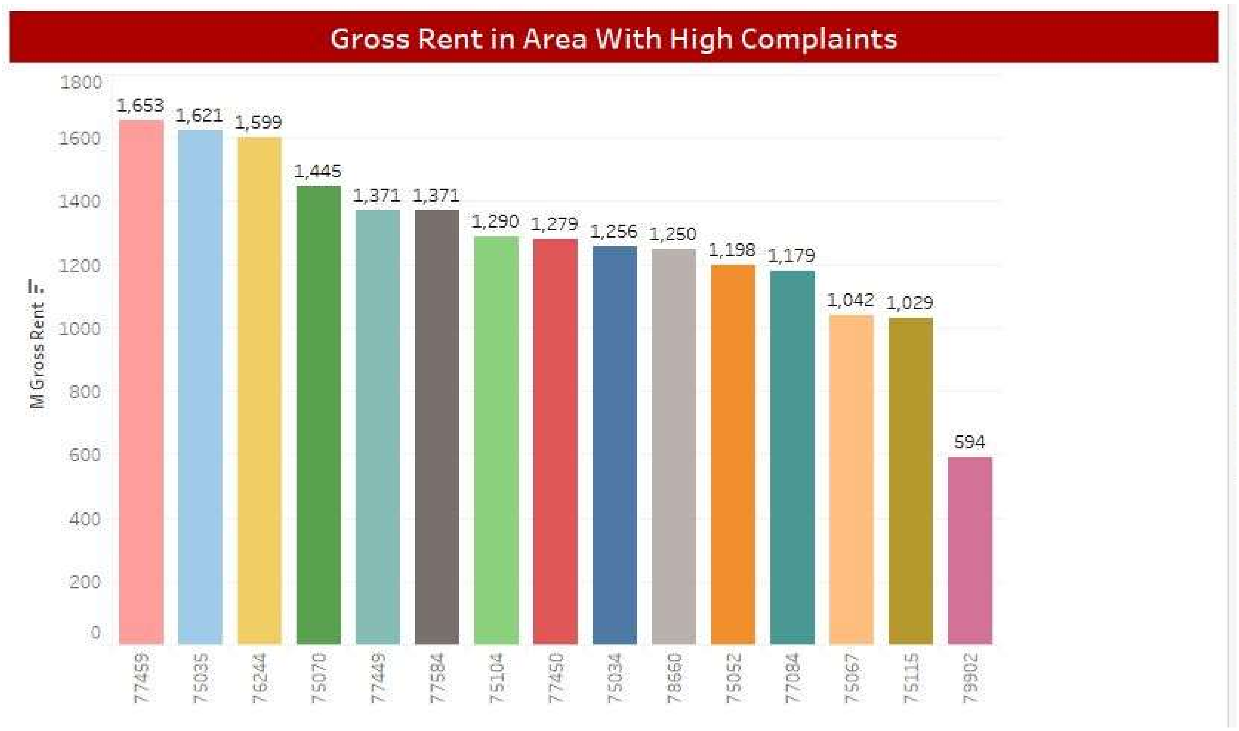


Figure A10: Gross Rent Varies in Area with High Complaints.

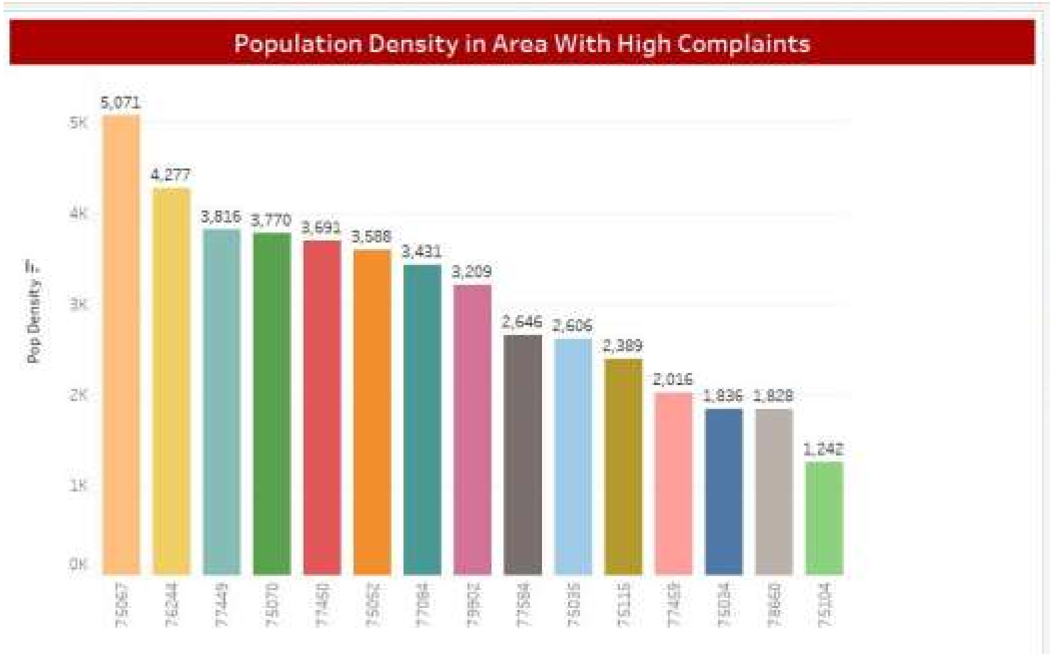


Figure A11: Population Density Varies in Area with High Complaints

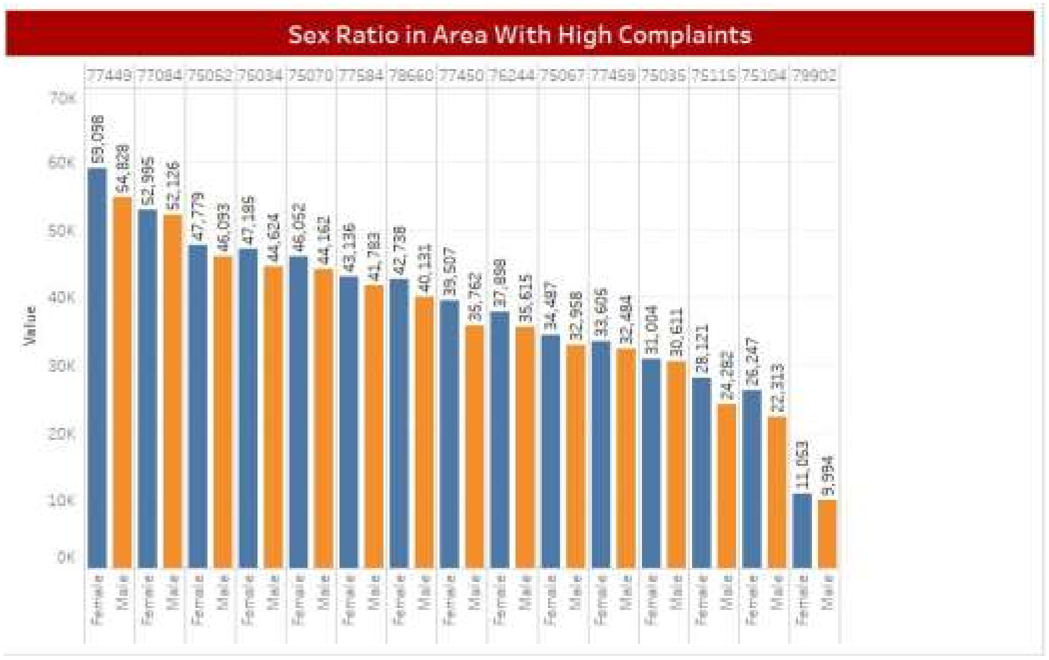


Figure A12: Variation in Sex Ratio in Areas with High Complaints.

o

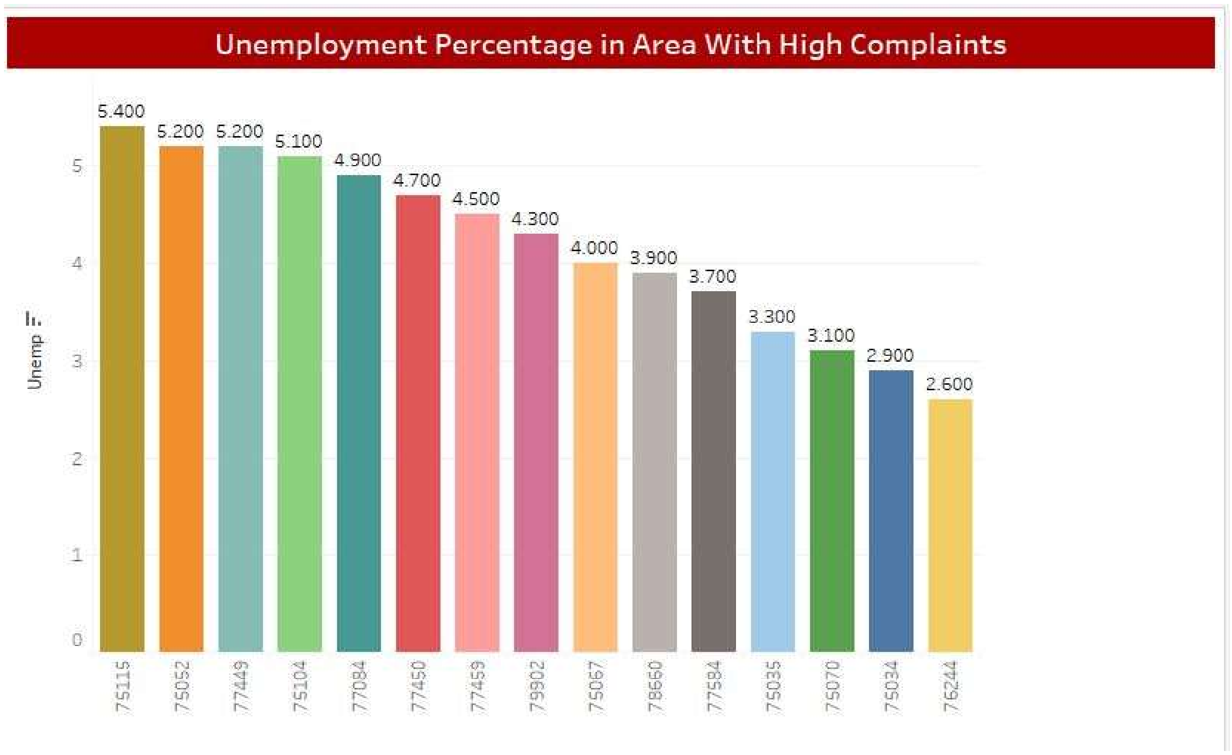


Figure A13: Variation in Unemployment Percentage in Areas with High Complaints.

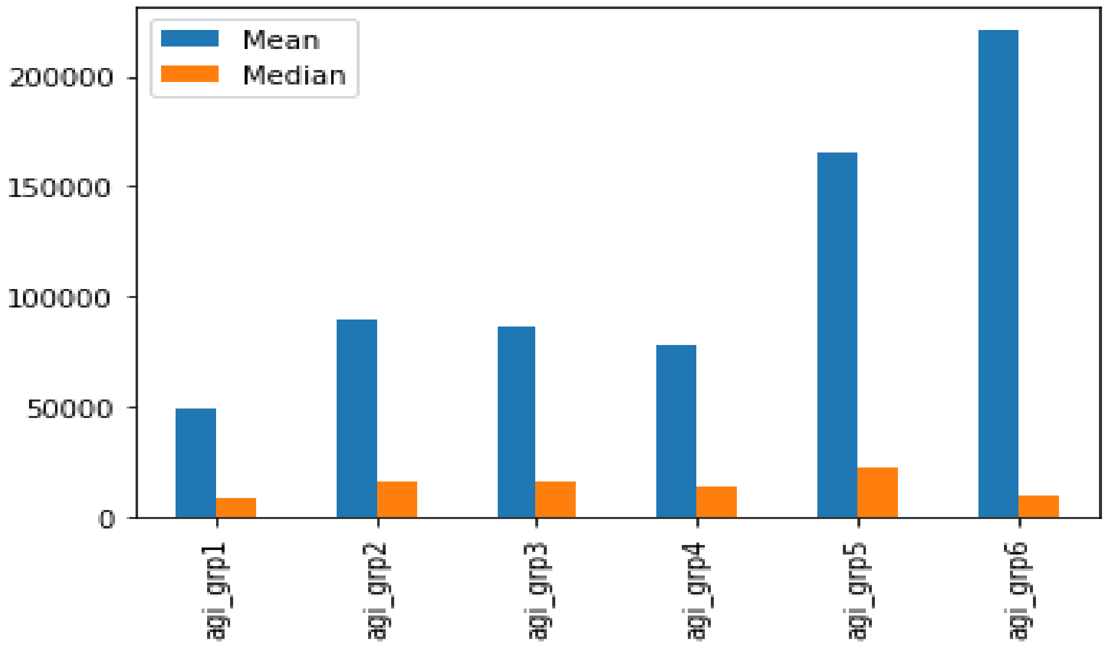


Figure A14: Gross income range wise mean and median Income

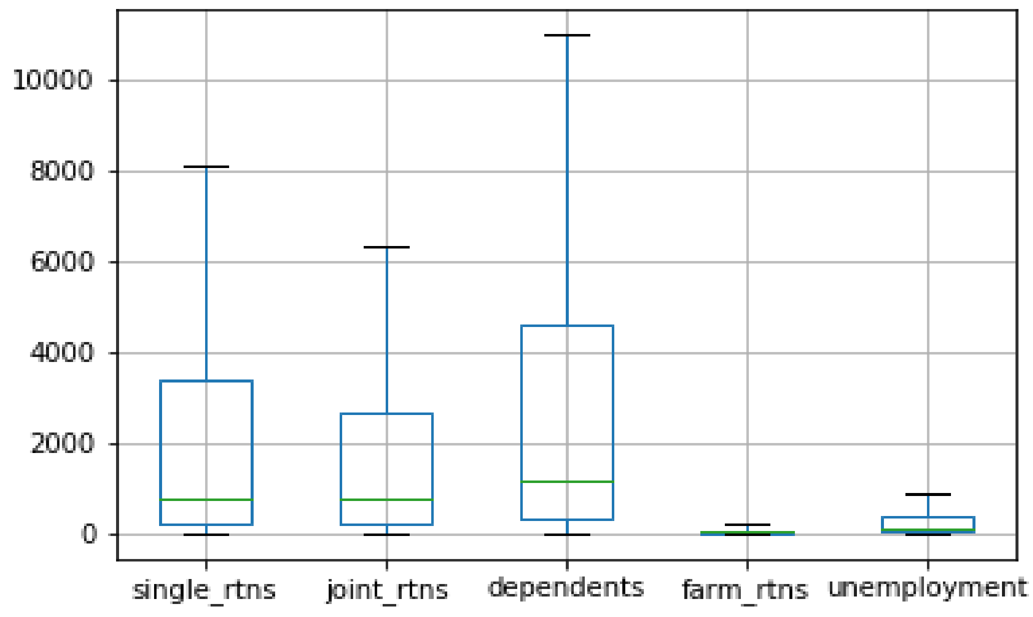


Figure A15: Interquartile range for number of dependents as per Income tax return group

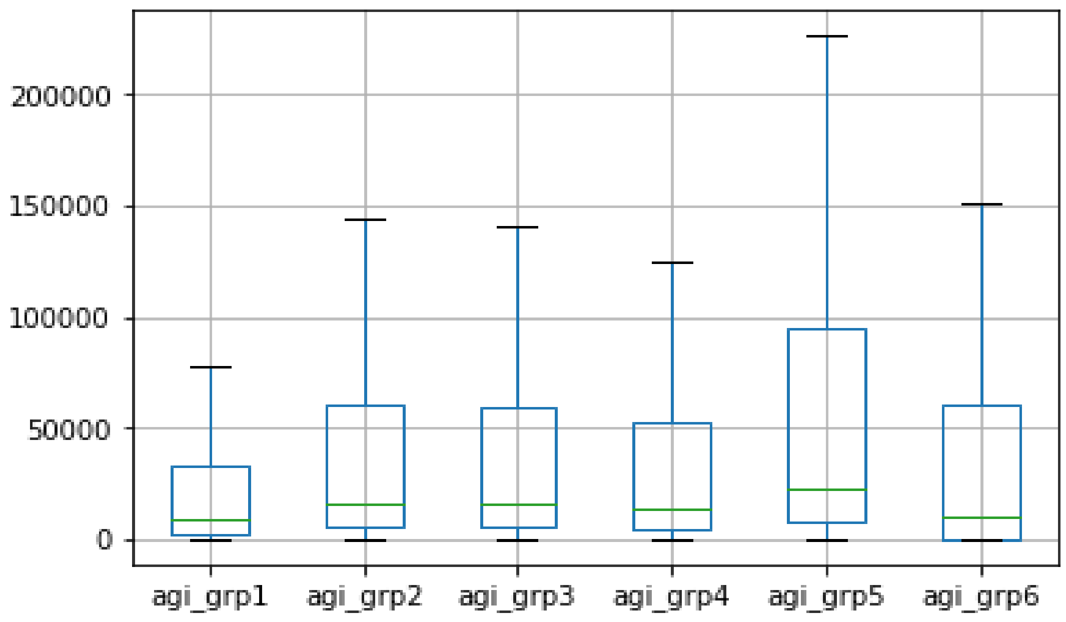


Figure A16: Interquartile range for aggregate adjusted annual gross Income

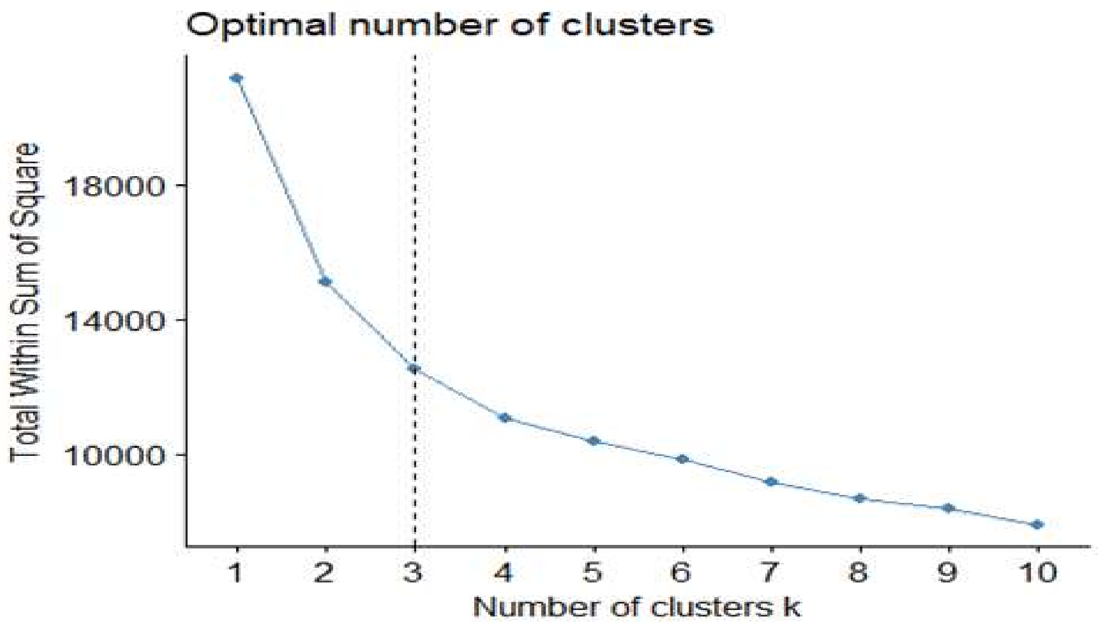


Figure A17: Scree plot for cluster analysis

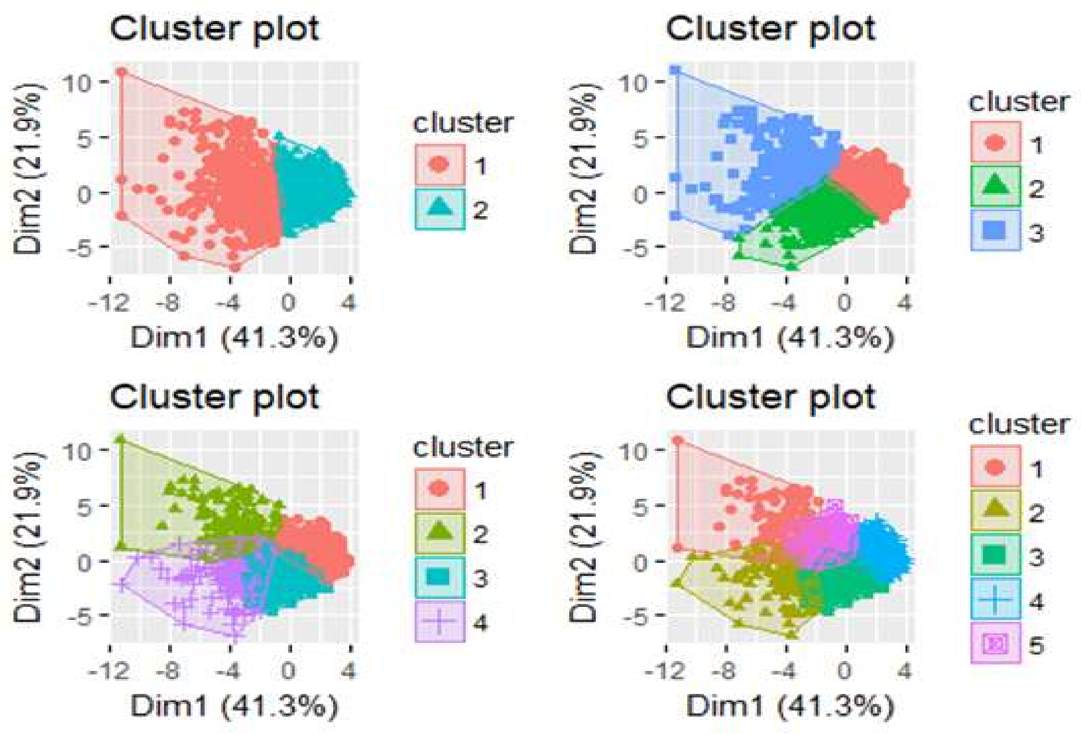


Figure A18: Cluster representation for different cluster sizes

r

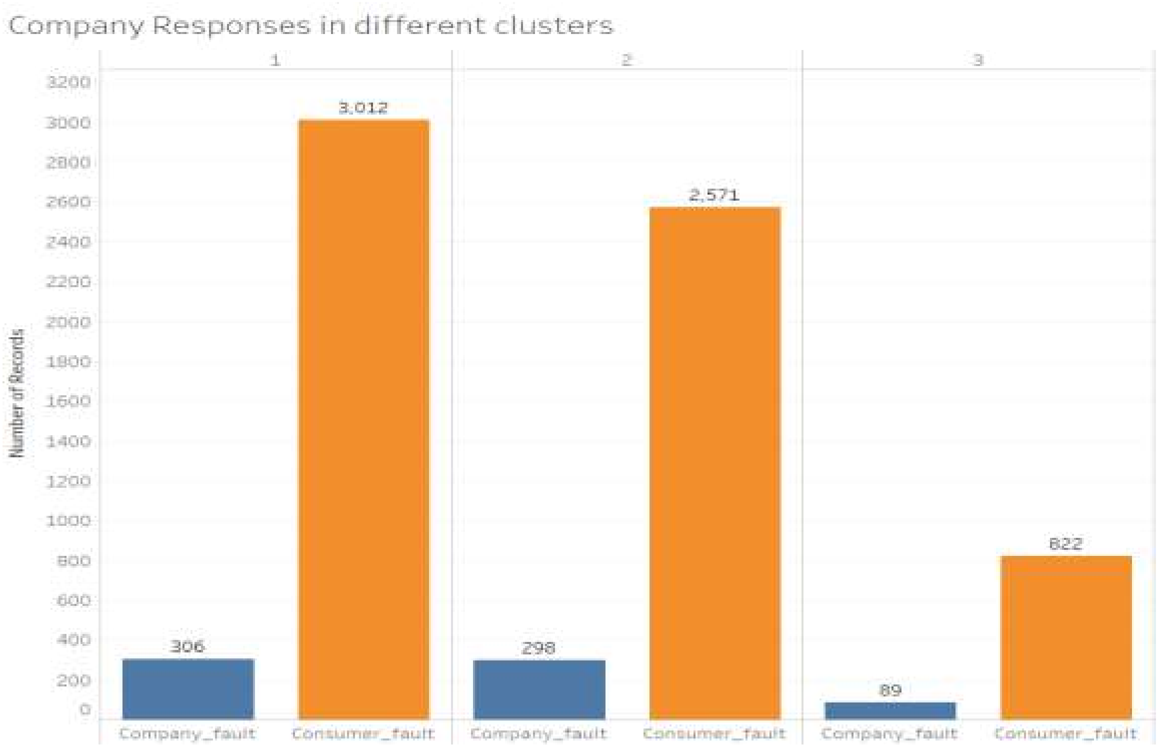


Figure A19: Company response for different clusters

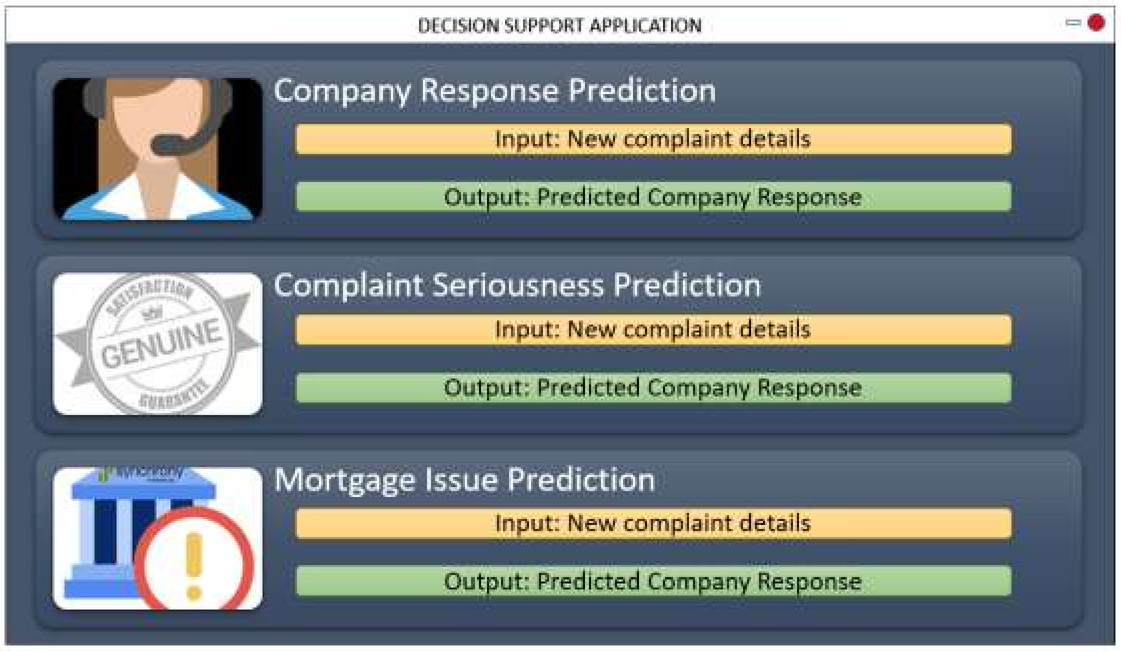

DECISION SUPPORT APPLICATION
Company Response Prediction
Input: New complaint details
Output: Predicted Company Response
Complaint Seriousness Prediction
Input: New complaint details
Output: Predicted Company Response
Mortgage Issue Prediction
Input: New complaint details
Output: Predicted Company Response

Figure A20: Sample User interface for prediction and decision support

Consumer Complaints Analysis Source Code

Python Script for Web-Scraping Demographics dataset

```
#Import the required libraries
from lxml import html
import requests
import random
import time
import pandas as pd
from fake_useragent import UserAgent

#Global declaration of user agent required for get() request
ua = UserAgent()

#Import the unique datasets present in our complaints database
filename = 'uniquezip_tx.csv'
data = pd.read_csv(filename, converters={0:str})

#Convert the dataframe into list
mnop = data['ZIP'].values.tolist()

#Change the subset values, in order to scrape data, like 1-500 records per day
mn = mnop[1435:1847]

#Create a nested list for gathering the scraped data for each zip
final = []
final_1 = []
for l in range(len(mn)):
	final.append(final_1)

#for loop for multiple requests
for k in range(len(mn)):
	agent = {'User-Agent': ua.random}
	url = 'http://www.city-data.com/zips/'+mn[k]+'.html'
	response = requests.get(url,headers=agent)

	#Decode the website content using ISO 8859
	decode_response = response.content.decode('ISO-8859-1')
	tree = html.fromstring(decode_response)

	#Obtain the required data from the html element using xpath
	head_key = tree.xpath('//div[@id="body"][@class="row"]//following-sibling::b/following::text()'

	#Create empty list to keep appending required values from the head_key
	ab_list = []

	for i in range(len(head_key)):
		#Houses and condos
		if("Houses and condos:" in head_key[i]):
		#ab_list.append(head_key[i])
		ab_list.append(head_key[i+1])
```

```
#Housing units with mortgage
if("Housing units in zip code" in head_key[i]):
#ab_list.append(head_key[i])
ab_list.append(head_key[i+1])

#Housing units without mortgage
if("Houses without a mortgage" in head_key[i]):
1
#ab_list.append(head_key[i])
ab_list.append(head_key[i+1])

#Estimated median house/condo value
if("Estimated median house/condo value in 2016" in head_key[i]):
#ab_list.append(head_key[i])
ab_list.append(head_key[i+1])

#cost of living index
if("Mar. 2016 cost of living index in zip code" in head_key[i]):
#ab_list.append(head_key[i])
ab_list.append(head_key[i+1])

#Estimated median household income in 2016:
if("Estimated median household income in 2016: " in head_key[i]):
#ab_list.append(head_key[i])
ab_list.append(head_key[i+2])

#Median gross rent
if("Median gross rent in 2016:" in head_key[i]):
#ab_list.append(head_key[i])
ab_list.append(head_key[i+1])

#real estate property taxes paid for housing units with mortgages
if("Median real estate property taxes paid for housing units with mortgages" in head_key[i]):
#ab_list.append(head_key[i])
ab_list.append(head_key[i+1])

#real estate property taxes paid for housing units with no mortgage
if("Median real estate property taxes paid for housing units with no mortgage" in head_key[i]):
#ab_list.append(head_key[i])
ab_list.append(head_key[i+1])

#Unemployment rate
if("Unemployed:" in head_key[i]):
#ab_list.append(head_key[i])
ab_list.append(head_key[i+1])

#Males
if("Males:" in head_key[i]):
#ab_list.append(head_key[i])
ab_list.append(head_key[i+1])

#Females
if("Females:" in head_key[i]):
```

```
                #ab_list.append(head_key[i])
                ab_list.append(head_key[i+1])

                #resident age
                if("Median resident age:" in head_key[i]):
                #ab_list.append(head_key[i])
                ab_list.append(head_key[i+2])

                #commute
                if("Mean travel time to work (commute):" in head_key[i]):
                #ab_list.append(head_key[i])
                ab_list.append(head_key[i+1])
                2

                #Population density
                if("Population density:" in head_key[i]):
                #ab_list.append(head_key[i])
                ab_list.append(head_key[i+1])

                #Population density
                if("% of renters here:" in head_key[i]):
                #ab_list.append(head_key[i])
                ab_list.append(head_key[i+1])
        x = ab_list.copy()

        #Cleansing the obtained values
        for i in range(len(x)):
                x[i] = x[i].replace('$','')
                x[i] = x[i].replace('minutes','')
                x[i] = x[i].replace('years','')
                x[i] = x[i].replace('\r\n','')
                x[i] = x[i].replace(' ','')
                x[i] = x[i].replace('\xa0','')
                x[i] = x[i].replace(',','')
                x[i] = x[i].replace('%','')

        #Further cleasning activity
        y = x.copy()
        for i in range(len(x)):
                if("(" in x[i]):
                        y[i] = "(".join(x[i].split("(")[:-1])
        final[k]=y.copy()

        #Generate a random time delay between each web request
        timeDelay = random.randrange(0, 25)
        time.sleep(timeDelay)

#Create a new dataframe and export to csv file format
new_df = pd.DataFrame(columns=['houses','percent_rent', 'cost_of_living','pop_density', 'm_prop_tax_'male',
'female', 'unemp', 'commute', 'est_m_house_val', 'm_res_age', 'est_m_house_'houses_w_mtg',
'houses_w_out_mtg', 'm_gross_rent'], data=final, index=mn)
```

```
new_df.to_csv("ZipTX/zip_tx_v2.csv")
```

R Script for data modeling using complaints data integrated with Demographics dataset

```
#### Loading required libraries
library(dplyr)
library(cluster)
library(stringr)
library(factoextra)
library(caret)
library(rpart)
library(rpart.plot)
library(DMwR)
library(ROSE)
library(ebmc)
library(rusboost)
library(gridExtra)
library(e1071)
library(nnet)
library(corrplot)

# Import CFPB Consumer dataset
data <- read.csv("ConsumerComplaintsAlltab.csv", sep = "\t")
str(data)

#remove complaint text, Tags, consumer_consent, submitted_via.
data_nt <- data[-c(6,11,12,13)]
str(data_nt)

#Demographics dataset
demo_tx <- read.csv("WebScraping/ZipTX/zip_tx_v1.csv")
demo_tx_na <- na.omit(demo_tx)
names(demo_tx_na)[1] <- "ZIP.code"
summary(demo_tx_na)


# Clustering on demographic dataset
demo_tx_na_rn <- demo_tx_na[,-1]
demo_tx_scale <- demo_tx_na[,-1]
demo_tx_scale <- scale(demo_tx_scale)
rownames(demo_tx_scale) <- demo_tx_na[,1]
demo_tx_eucl <- dist(demo_tx_scale, method = "euclidean")

#Scree plot
fviz_nbclust(demo_tx_scale, kmeans, method = "wss") + geom_vline(xintercept = 3, linetype =
2)

#K means Clustering
set.seed(123)
km.res <- kmeans(demo_tx_scale, 3, nstart = 50)
demo_tx_clust <- cbind(demo_tx_na_rn, cluster=km.res$cluster)
```

w

```
#Visualizing different cluster sizes
km.res.2 <- kmeans(demo_tx_scale, 2, nstart = 50)
km.res.3 <- kmeans(demo_tx_scale, 3, nstart = 50)
km.res.4 <- kmeans(demo_tx_scale, 4, nstart = 50)
km.res.5 <- kmeans(demo_tx_scale, 5, nstart = 50)
p1 <- fviz_cluster(km.res.2, data = demo_tx_scale, geom = "point")
p2 <- fviz_cluster(km.res.3, data = demo_tx_scale, geom = "point")
p3 <- fviz_cluster(km.res.4, data = demo_tx_scale, geom = "point")
p4 <- fviz_cluster(km.res.5, data = demo_tx_scale, geom = "point")
grid.arrange(p1, p2, p3, p4, nrow = 2)

#Cluster statistics
aggregate(demo_tx_na_rn, by = list(cluster=km.res$cluster), mean)
demo_tx_clust <- cbind(demo_tx_na_rn, cluster=km.res$cluster)

#cluster size
km.res$size

#cluster means
km.res$centers

#visualization
fviz_cluster(km.res, data = demo_tx_scale, geom = "point")

#Merge with original TEXAS dataset
data_zip_merge_tx <- data_nt %>% filter(data_nt$State=="TX")
demo_tx_zipclust_prep <- cbind(ZIP.code=demo_tx_na$ZIP.code, demo_tx_na_rn,
cluster=km.res$cluster)
merged_clust_tx <- merge(data_zip_merge_tx, demo_tx_zipclust_prep, by="ZIP.code", all.x =
T)
merged_clust_tx_na <- merged_clust_tx %>% filter(!is.na(cluster))
str(merged_clust_tx_na)

# Data preparation for predicting company response

#remove date, public response, state, date sent, consumer disputation, complaint id
tx_clust <- merged_clust_tx_na[-c(7, 9, 10, 12, 13, 31)]
tx_clust[,"Month"] <- as.factor(format(as.Date(tx_clust$Date.received,
format="%m/%d/%Y"),"%m"))
#tx_clust[,"Year"] <- as.factor(format(as.Date(tx_clust$Date.received,
format="%m/%d/%Y"),"%Y"))

#remove data received
tx_clust <-tx_clust[-2]
str(tx_clust)

#replace sub issue
for(i in 2:5){
  tx_clust[,i] <- as.character(tx_clust[,i])
}
x1 <- tx_clust$Sub.product==""
```

```
x2 <- tx_clust$Sub.issue==""

tx_clust$Sub.product[x1] <- tx_clust$Product[x1]
tx_clust$Sub.issue[x2] <- tx_clust$Issue[x2]

for(i in 2:5){
  tx_clust[,i] <- as.factor(tx_clust[,i])
}

#combine outcome categories
tx_clust$Company.response.to.consumer <-
as.character(tx_clust$Company.response.to.consumer)

x1 <- tx_clust$Company.response.to.consumer == "Closed"
tx_clust$Company.response.to.consumer[x1] = "Consumer_fault"

x1 <- tx_clust$Company.response.to.consumer=="Closed with explanation"
tx_clust$Company.response.to.consumer[x1] = "Consumer_fault"

x1 <- tx_clust$Company.response.to.consumer=="Closed without relief"
tx_clust$Company.response.to.consumer[x1] = "Consumer_fault"

x1 <- tx_clust$Company.response.to.consumer == "Closed with monetary relief"
tx_clust$Company.response.to.consumer[x1] = "Company_fault"

x1 <- tx_clust$Company.response.to.consumer=="Closed with non-monetary relief"
tx_clust$Company.response.to.consumer[x1] = "Company_fault"

x1 <- tx_clust$Company.response.to.consumer=="Closed with relief"
tx_clust$Company.response.to.consumer[x1] = "Company_fault"

x1 <- tx_clust$Company.response.to.consumer == "Company_fault" |
tx_clust$Company.response.to.consumer == "Consumer_fault"
tx_clust$Company.response.to.consumer[!x1] = "Late_Response"

final_tx_resp_na <- tx_clust %>%
filter(tx_clust$Company.response.to.consumer!="Late_Response")

final_tx_resp_na$Company.response.to.consumer <-
as.factor(final_tx_resp_na$Company.response.to.consumer)

#-------------------------------------------------------------------------------------Mortgage data
tx_clust_rep_mortgage <- final_tx_resp_na %>% filter(Product=="Mortgage")
tx_clust_rep_mortgage_issues <- tx_clust_rep_mortgage %>% filter(Issue=="Loan
modification,collection,foreclosure" | Issue=="Loan servicing, payments, escrow account")

final_tx_resp_na_mortgage <- tx_clust_rep_mortgage_issues[-c(2)]

for(i in c(1,2,3,4,5,6,24)){
  final_tx_resp_na_mortgage[,i]<-as.character(final_tx_resp_na_mortgage[,i])
  final_tx_resp_na_mortgage[,i]<-as.factor(final_tx_resp_na_mortgage[,i])
}
```

y

```
## Modeling

#Remove complaint ID
final_tx <- final_tx_resp_na_mortgage[,-7] #final_tx_prep[-c(1,2, 7, 10, 14, 15)]
str(final_tx)
for(i in c(1,2,3,4,5,6,23)){
  final_tx[,i]<-as.character(final_tx[,i])
  final_tx[,i]<-as.factor(final_tx[,i])
}

for(i in 7:22){
  final_tx[,i]<-as.numeric(final_tx[,i])
}
final_tx[,7:22] <- scale(final_tx[,7:22])

#--------------------------------------------------------------------------------------data exports
texas_mortgage <- final_tx_resp_na_mortgage[,-7]
#write.csv(texas_mortgage, "texas_mortgage.csv")

ref_final_tx <- final_tx

#correlation
x <- final_tx
for(i in 1:ncol(x)){
  x[,i] <- as.numeric(x[,i])
}
x <- scale(x)
#write.csv(cor(x), "Correlation.csv")

#Excluding multi collinear variables
final_tx <- final_tx[,-c(1, 4, 7, 11, 12, 14, 15, 19)]

#--------------------------------------------------------------------------------------Train and test
set.seed(123)
partition<-createDataPartition(final_tx$Company.response.to.consumer,p=0.80,list = FALSE)
train_tx<-final_tx[partition,]
test_tx<-final_tx[-partition,]


#----------------------------------------------------------------decision tree model

tx_model_resp <- rpart(Company.response.to.consumer ~ ., data=train_tx, method = "class")

#rpart.plot(tx_model_resp,digits = 2, split.fun=split.fun, faclen = 3)
#rpart.plot(tx_model_resp,digits = 2,fallen.leaves = TRUE,type = 3,extra = 1)

varImp(tx_model_resp)

wp1 <- predict(tx_model_resp,test_tx, type = "class")
```

```
confusionMatrix(wp1, test_tx$Company.response.to.consumer, positive = "Company_fault",
mode = "everything")

roc.curve(test_tx$Company.response.to.consumer, wp1, plotit = F)

#----------------------------------------------------------------------OVer sampling

train_over <- ovun.sample(Company.response.to.consumer ~ ., data=train_tx, method = "over",
N=9000, seed=123)$data

table(train_over$Company.response.to.consumer)

tx_model_resp <- rpart(Company.response.to.consumer ~ ., data=train_over, method = "class")

#rpart.plot(tx_model_resp,digits = 2, split.fun=split.fun, faclen = 3)

varImp(tx_model_resp)

wp1 <- predict(tx_model_resp,test_tx, type = "class")

confusionMatrix(wp1, test_tx$Company.response.to.consumer, positive = "Company_fault",
mode = "everything")

roc.curve(test_tx$Company.response.to.consumer, wp1, plotit = F)

#accuracy.meas(test_tx$Company.response.to.consumer, wp1[,2])

#roc.curve(test_tx$Company.response.to.consumer, wp1, plotit = F)


#----------------------------------------------------------------------Undersampling


train_under <- ovun.sample(Company.response.to.consumer ~ ., data=train_tx, method =
"under", N=972, seed=123)$data

table(train_under$Company.response.to.consumer)

tx_model_resp <- rpart(Company.response.to.consumer ~ ., data=train_under, method =
"class")

#rpart.plot(tx_model_resp,digits = 2, split.fun=split.fun, faclen = 3)

varImp(tx_model_resp)

wp1 <- predict(tx_model_resp,test_tx, type = "class")

confusionMatrix(wp1, test_tx$Company.response.to.consumer, positive = "Company_fault")

roc.curve(test_tx$Company.response.to.consumer, wp1, plotit = F)

#accuracy.meas(test_tx$Company.response.to.consumer, wp1[,2])
```

```
#roc.curve(test_tx$Company.response.to.consumer, wp1[,2], plotit = F)

#-------------------------------------------------------------------Both

train_both <- ovun.sample(Company.response.to.consumer ~ ., data=train_tx, method = "both",
p=0.5, seed = 123)$data
table(train_both$Company.response.to.consumer)

tx_model_resp <- rpart(Company.response.to.consumer ~ ., data=train_both, method = "class")

#rpart.plot(tx_model_resp,digits = 2, faclen = 3)

varImp(tx_model_resp)

wp1 <- predict(tx_model_resp,test_tx, type = "class")

confusionMatrix(wp1, test_tx$Company.response.to.consumer, positive = "Company_fault",
mode="everything")

roc.curve(test_tx$Company.response.to.consumer, wp1, plotit = F)

#accuracy.meas(test_tx$Company.response.to.consumer, wp1[,2])

#roc.curve(test_tx$Company.response.to.consumer, wp1[,2], plotit = F)


#-------------------------------------------------------------------ROSE
train_rose <- ROSE(Company.response.to.consumer ~ ., data=train_tx, seed = 123)$data

table(train_rose$Company.response.to.consumer)

tx_model_resp <- rpart(Company.response.to.consumer ~ ., data=train_rose, method = "class")

#rpart.plot(tx_model_resp,digits = 2, split.fun=split.fun, faclen = 3)

varImp(tx_model_resp)

wp1 <- predict(tx_model_resp,test_tx, type = "class")

confusionMatrix(wp1, test_tx$Company.response.to.consumer, positive = "Company_fault")

roc.curve(test_tx$Company.response.to.consumer, wp1, plotit = F)

#accuracy.meas(test_tx$Company.response.to.consumer, wp1[,2])

#roc.curve(test_tx$Company.response.to.consumer, wp1[,2], plotit = F)


#-------------------------------------------------------------------------DMwR SMOTE

set.seed(123)
```

```
train_SMOTE <- SMOTE(Company.response.to.consumer ~ ., data = train_tx, perc.over = 100,
perc.under = 200)
table(train_SMOTE$Company.response.to.consumer)

tx_model_resp <- rpart(Company.response.to.consumer ~ ., data=train_SMOTE, method =
"class")

#rpart.plot(tx_model_resp,digits = 2, split.fun=split.fun, faclen = 3)

varImp(tx_model_resp)

wp1 <- predict(tx_model_resp,test_tx, type = "class")

confusionMatrix(wp1, test_tx$Company.response.to.consumer, positive = "Company_fault")

roc.curve(test_tx$Company.response.to.consumer, wp1, plotit = F)


#accuracy.meas(test_tx$Company.response.to.consumer, wp1[,2])

#roc.curve(test_tx$Company.response.to.consumer, wp1[,2], plotit = F)


# SVM
tx_model_resp <- svm(Company.response.to.consumer ~ ., data=train_tx)

#varImp(tx_model_resp)

wp1 <- predict(tx_model_resp,test_tx, type = "class")

confusionMatrix(wp1, test_tx$Company.response.to.consumer, positive = "Company_fault")

roc.curve(test_tx$Company.response.to.consumer, wp1, plotit = F)

#nnet
num_tx_mortgage <- final_tx
num_tx_mortgage_y <- num_tx_mortgage[4]
num_tx_mortgage_x <- num_tx_mortgage[,-4]

for(i in (1:ncol(num_tx_mortgage_x))){
  num_tx_mortgage_x[,i] <- as.numeric(num_tx_mortgage_x[,i])
}

num_tx_mortgage_x <- scale(num_tx_mortgage_x)
num_tx <- cbind(num_tx_mortgage_x, num_tx_mortgage_y)

set.seed(123)
partition<-createDataPartition(num_tx$Company.response.to.consumer,p=0.70,list = FALSE)
train_tx<-num_tx[partition,]
test_tx<-num_tx[-partition,]
train_SMOTE <- SMOTE(Company.response.to.consumer ~ ., data = train_tx, perc.over = 100,
perc.under = 200)
```

```
table(train_SMOTE$Company.response.to.consumer)
train_over <- ovun.sample(Company.response.to.consumer ~ ., data=train_tx, method = "over",
N=9000, seed=123)$data
table(train_over$Company.response.to.consumer)

tx_model_resp <- nnet(Company.response.to.consumer ~ ., data=train_SMOTE, size = 10)

#varImp(tx_model_resp)

wp1 <- predict(tx_model_resp,test_tx, type = "class")

confusionMatrix(as.factor(wp1), test_tx$Company.response.to.consumer, positive =
"Company_fault")

roc.curve(test_tx$Company.response.to.consumer, as.factor(wp1), plotit = F)

tx_model_resp <- rpart(Company.response.to.consumer ~ ., data=train_over, method = "class")

#rpart.plot(tx_model_resp,digits = 2, split.fun=split.fun, faclen = 3)

varImp(tx_model_resp)

wp1 <- predict(tx_model_resp,test_tx, type = "class")

confusionMatrix(wp1, test_tx$Company.response.to.consumer, positive = "Company_fault")

roc.curve(test_tx$Company.response.to.consumer, wp1, plotit = F)
```

Python script for prediction company response (Objective 1 – Part 1)

```
# coding: utf-8
# In[2]:
# Importing the necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.cross_validation import KFold
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import learning_curve
from sklearn import tree
import pydotplus
import graphviz
from IPython.display import Image
# In[20]:
# Importing the dataset
url = 'C:\\Users\\Rusan\\Dropbox\\CapstoneProject\\DataSet\\TaxComplaint.csv'
complaint = pd.read_csv(url, parse_dates=['Date received', 'Date sent to company'])
complaint.info()
# In[21]
# Renaming some of the columns
complaint = complaint.rename(columns = {'Date received':'DateReceived', 'Sub-
product':'SubProduct',
                              'Company public response':'CompanyPublicResponse',
                              'Consumer consent provided?':'ConsumerConsentProvided',
                              'Submitted via':'SubmittedVia',
                              'Date sent to company':'DateSentToCompany',
                              'Company response to consumer':'CompanyResponseToConsumer',
                              'Timely response?':'TimelyResponse',
                              'Consumer disputed?':'ConsumerDisputed'})
# In[22]:
# Drop the columns with null values
complaint = complaint.drop(columns=['CompanyPublicResponse','ConsumerConsentProvided'])
# In[7]:
# check the number of features for each class in our dataset
pd.value_counts(complaint['SubProduct'], sort=True)
# In[6]:
pd.value_counts(complaint['Issue'], sort=True)
# In[7]:
pd.value_counts(complaint['Company'], sort=True)
# In[23]:
# Just taking top 5 companies with most complaints for the analysis
complaint = complaint[(complaint['Company']=='BANK OF AMERICA, NATIONAL
ASSOCIATION')|
      (complaint['Company']=='WELLS FARGO & COMPANY')|
      (complaint['Company']=='OCWEN LOAN SERVICING LLC')|
      (complaint['Company']=='JPMORGAN CHASE & CO.')|
      (complaint['Company']=='NATIONSTAR MORTGAGE')]
# In[9]:
pd.value_counts(complaint['TimelyResponse'], sort=True)
# In[9]:
pd.value_counts(complaint['CompanyResponseToConsumer'], sort=True)
# In[11]:
```

```
pd.value_counts(complaint['ConsumerDisputed'], sort=True)
# In[24]:
# Replacing the dataset categorical variables into numeric codes
complaint['month'] = complaint['DateReceived'].dt.month
# In[25]:
complaint['Issue'] = complaint['Issue'].map({'Loan modification,collection,foreclosure': 1,
        'Loan servicing, payments, escrow account': 2, 'Application, originator, mortgage broker':3,
        'Settlement process and costs':4,'Credit decision / Underwriting':5,'Other':6})
# In[26]:
complaint['SubProduct'] = complaint['SubProduct'].map({'Second mortgage': 1,
        'Reverse mortgage': 2, 'VA mortgage':3,
        'FHA mortgage':4,'Conventional adjustable mortgage (ARM)':5,'Conventional fixed
mortgage':6,
        'Home equity loan or line of credit':7,'Other mortgage':8})
# In[27]:
complaint['Company'] = complaint['Company'].map({'BANK OF AMERICA, NATIONAL
ASSOCIATION':1,
        'WELLS FARGO & COMPANY':2,'OCWEN LOAN SERVICING LLC':3, 'JPMORGAN
CHASE & CO.':4,
        'NATIONSTAR MORTGAGE':5})
# In[28]:
complaint['TimelyResponse'] = complaint['TimelyResponse'].map({'Yes':1, 'No':0})
# In[29]:
complaint['CompanyResponseToConsumer'] =
complaint['CompanyResponseToConsumer'].map({'Closed with explanation':1,
                                             'Closed with non-monetary relief':2,
                                             'Closed without relief':3,
                                             'Closed':4,
                                             'Closed with monetary relief':5,
                                             'Closed with relief':6})
# In[30]:
complaint['ConsumerDisputed'] = complaint['ConsumerDisputed'].map({'Yes':1, 'No':0})
# In[37]:
# Drop redundant columns
complaint =
complaint.drop(columns=['State','DateReceived','DateSentToCompany','SubmittedVia'])
complaint_wo_na = complaint[np.isfinite(complaint['CompanyResponseToConsumer'])]
complaint = complaint.drop(columns=['CompanyResponseToConsumer'])
# In[32]:
# Creating the required functions
def evaluateModel(estimator, X_test, y_test, name):
   """This function takes the tuned classifier and produces the accuracy, RSME, Confusion
Matrix, and
   Classification Report for the classifier supplied"""
   print("Classifer: ",name)
```

```
    print("\nAccuracy: {0:.4f}".format(estimator.score(X_test, y_test)))
    # Make prediction and print the confusion matrix
    y_pred = estimator.predict(X_test)
    # RSME value
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    print("\nRoot Mean Squared Error: {0:.4f}".format(rmse))
    print('\nConfusion Matrix:')
    print(confusion_matrix(y_test, y_pred))
    print('\nClassification Report:')
    print(classification_report(y_test, y_pred))
# The following function is from:
# http://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=1, train_sizes=np.linspace(.1, 1.0, 5)):
    """
    Generate a simple plot of the test and traning learning curve.
    Parameters
    ----------
    estimator : object type that implements the "fit" and "predict" methods
        An object of that type which is cloned for each validation.
    title : string
        Title for the chart.
    X : array-like, shape (n_samples, n_features)
        Training vector, where n_samples is the number of samples and
        n_features is the number of features.
    y : array-like, shape (n_samples) or (n_samples, n_features), optional
        Target relative to X for classification or regression;
        None for unsupervised learning.
    ylim : tuple, shape (ymin, ymax), optional
        Defines minimum and maximum yvalues plotted.
    cv : integer, cross-validation generator, optional
        If an integer is passed, it is the number of folds (defaults to 3).
        Specific cross-validation objects can be passed, see
        sklearn.cross_validation module for the list of possible objects
    n_jobs : integer, optional
        Number of jobs to run in parallel (default 1).
    """
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()
    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")
    plt.legend(loc="best")
    return plt
# ## Predicting Issues
# Create arrays for the predictors and the target variable
y = complaint['Issue'].values
X = preprocessing.scale(complaint.drop('Issue', axis=1).values)
# In[22]:
# Split into training and test set : we keep 80% of our data as training set and 20% as test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=123,
stratify=y)
# In[23]:
# KFold Cross-Validation
fold = KFold(len(X_train), n_folds=5, shuffle=True)
# In[24]:
# Decision Tree Classifier
# Create the classifier
decTree = DecisionTreeClassifier()
# Tuning hyper-parameters
params = {'criterion': ['gini', 'entropy']}
decTreeEst = GridSearchCV(estimator=decTree, cv=fold, param_grid=params)
# In[39]:
# Fit the classifer to the training data
decTreeEst.fit(X_train, y_train)
# In[26]:
# Creating the classifier with tuned hyper-parameter
bestDecTreeModel = DecisionTreeClassifier(criterion=decTreeEst.best_estimator_.criterion)
# In[27]:
# Evaluating the model
evaluateModel(decTreeEst,X_test, y_test, 'Decision Tree Classifier')
plot_learning_curve(bestDecTreeModel, 'Decision Tree Classifier', X_test, y_test)
# In[44]:
```

```
## Visualizing the Decision Tree
# Fit the best model
bestDecTreeModel.fit(X_train, y_train)
# Create DOT Data
dot_data = tree.export_graphviz(bestDecTreeModel, out_file=None,
                                feature_names=colName,
                                class_names=list(['1','2','3','4','5','6']))
# Create GraphViz object
graph = pydotplus.graph_from_dot_data(dot_data)
# Draw graph
Image(graph.create_png())
# In[59]:
# Score the best Decision Tree model
colName = complaint.columns
colName = colName[np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,15,16,17,18])]
pd.DataFrame(bestDecTreeModel.feature_importances_, index=colName,
        columns = ['Importance']).sort_values('Importance', ascending=False)
# In[33]:
# Random Forest Classifier
# Create the classifier
rForest = RandomForestClassifier()
# In[34]
# Tuning hyper-parameters
params = {'n_estimators': [10, 50, 100, 200, 500], 'criterion': ['gini', 'entropy']}
rForestEst = GridSearchCV(estimator=rForest, cv=fold, param_grid=params)
# In[35]:
# Fit the classifer to the training data
rForestEst.fit(X_train, y_train)
# In[62]:
# Creating the classifier with tuned hyper-parameter
bestRandForestModel =
RandomForestClassifier(n_estimators=rForestEst.best_estimator_.n_estimators,
                                        criterion=rForestEst.best_estimator_.criterion)
# In[63]:
# Score the best Random Forest model
bestRandForestModel.fit(X_train, y_train)
colName = complaint.columns
colName = colName[np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,15,16,17,18])]
pd.DataFrame(bestRandForestModel.feature_importances_,
        index=colName, columns = ['Importance']).sort_values('Importance', ascending=False)
# In[37]:
# Evaluating the model
evaluateModel(rForestEst,X_test, y_test, 'Random Forest Classifier')
plot_learning_curve(bestRandForestModel, 'Random Forest Classifier', X_test, y_test)
# ## Predicting Company Response to Consumer
# In[39]:
# Create arrays for the predictors and the target variable
y = complaint_wo_na['CompanyResponseToConsumer'].values
X = preprocessing.scale(complaint_wo_na.drop('CompanyResponseToConsumer',
axis=1).values)
# In[45]:
# Split into training and test set : we keep 80% of our data as training set and 20% as test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=123,
stratify=y)
# KFold Cross-Validation
fold = KFold(len(X_train), n_folds=5, shuffle=True)
# In[48]:
# Decision Tree Classifier
# Create the classifier
decTree = DecisionTreeClassifier()
# Tuning hyper-parameters
params = {'criterion': ['gini', 'entropy']}
decTreeEst = GridSearchCV(estimator=decTree, cv=fold, param_grid=params)
# Fit the classifer to the training data
decTreeEst.fit(X_train, y_train)
# Creating the classifier with tuned hyper-parameter
bestDecTreeModel = DecisionTreeClassifier(criterion=decTreeEst.best_estimator_.criterion)
# Predict with the best model
bestDecTreeModel.fit(X_train, y_train)
# In[50]:
# Evaluating the model: Decision Tree Classification
evaluateModel(decTreeEst,X_test, y_test, 'Decision Tree Classification')
plot_learning_curve(bestDecTreeModel, 'Decision Tree Classification', X_test, y_test)
# Score the best Random Forest model
colName = complaint_wo_na.columns
colName = colName[np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,18,19])]
pd.DataFrame(bestDecTreeModel.feature_importances_,
            index=colName, columns = ['Importance']).sort_values('Importance', ascending=False)
# In[51]:
# Random Forest Classifier
# Create the classifier
rForest = RandomForestClassifier()
# Tuning hyper-parameters
params = {'n_estimators': [10, 50, 100, 200, 500], 'criterion': ['gini', 'entropy']}
rForestEst = GridSearchCV(estimator=rForest, cv=fold, param_grid=params)
# Fit the classifer to teh training data
rForestEst.fit(X_train, y_train)
# Creating the classifier with tuned hyper-parameter
bestRForestModel =
RandomForestClassifier(n_estimators=rForestEst.best_estimator_.n_estimators,
                                        criterion=rForestEst.best_estimator_.criterion)
# Predict with the best model
bestRForestModel.fit(X_train, y_train)
# In[53]:
# Evaluating the model: Random Forest Classification
evaluateModel(rForestEst,X_test, y_test, 'Random Forest Classification')
plot_learning_curve(bestRForestModel, 'Random Forest Classification', X_test, y_test)
# Score the best Random Forest model
colName = complaint_wo_na.columns
colName = colName[np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,18,19])]
pd.DataFrame(bestRForestModel.feature_importances_,
        index=colName, columns = ['Importance']).sort_values('Importance', ascending=False)
```

Python script for prediction company response (Objective 1 – Part 2)

```
# coding: utf-8
# In[2]:
# Importing the necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.cross_validation import KFold
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import learning_curve
from sklearn import tree
import pydotplus
import graphviz
from IPython.display import Image
# In[20]:
# Importing the dataset
url = 'C:\\Users\\Rusan\\Dropbox\\CapstoneProject\\DataSet\\TaxComplaint.csv'
complaint = pd.read_csv(url, parse_dates=['Date received', 'Date sent to company'])
complaint.info()
# In[21]:
# Renaming some of the columns
complaint = complaint.rename(columns = {'Date received':'DateReceived', 'Sub-
product':'SubProduct',
                                'Company public response':'CompanyPublicResponse',
                                'Consumer consent provided?':'ConsumerConsentProvided',
                                'Submitted via':'SubmittedVia',
                                'Date sent to company':'DateSentToCompany',
                                'Company response to consumer':'CompanyResponseToConsumer',
                                'Timely response?':'TimelyResponse',
                                'Consumer disputed?':'ConsumerDisputed'})
# In[22]:
# Drop the columns with null values
complaint = complaint.drop(columns=['CompanyPublicResponse','ConsumerConsentProvided'])
# In[7]:
# check the number of features for each class in our dataset
pd.value_counts(complaint['SubProduct'], sort=True)
# In[6]:
pd.value_counts(complaint['Issue'], sort=True)
# In[7]
pd.value_counts(complaint['Company'], sort=True)
# In[23]:
# Just taking top 5 companies with most complaints for the analysis
complaint = complaint[(complaint['Company']=='BANK OF AMERICA, NATIONAL
ASSOCIATION')|
     (complaint['Company']=='WELLS FARGO & COMPANY')|
     (complaint['Company']=='OCWEN LOAN SERVICING LLC')|
     (complaint['Company']=='JPMORGAN CHASE & CO.')|
     (complaint['Company']=='NATIONSTAR MORTGAGE')]
# In[9]:
pd.value_counts(complaint['TimelyResponse'], sort=True)
# In[9]:
pd.value_counts(complaint['CompanyResponseToConsumer'], sort=True)
# In[11]
pd.value_counts(complaint['ConsumerDisputed'], sort=True)
# In[24]:
# Replacing the dataset categorical variables into numeric codes
complaint['month'] = complaint['DateReceived'].dt.month
# In[25]:
complaint['Issue'] = complaint['Issue'].map({'Loan modification,collection,foreclosure': 1,
      'Loan servicing, payments, escrow account': 2, 'Application, originator, mortgage broker':3,
      'Settlement process and costs':4,'Credit decision / Underwriting':5,'Other':6})
# In[26]:
complaint['SubProduct'] = complaint['SubProduct'].map({'Second mortgage': 1,
      'Reverse mortgage': 2, 'VA mortgage':3,
      'FHA mortgage':4,'Conventional adjustable mortgage (ARM)':5,'Conventional fixed
mortgage':6,
      'Home equity loan or line of credit':7,'Other mortgage':8})
# In[27]:
```

```
complaint['Company'] = complaint['Company'].map({'BANK OF AMERICA, NATIONAL
ASSOCIATION':1,
        'WELLS FARGO & COMPANY':2,'OCWEN LOAN SERVICING LLC':3, 'JPMORGAN
CHASE & CO.':4,
        'NATIONSTAR MORTGAGE':5})
# In[28]:
complaint['TimelyResponse'] = complaint['TimelyResponse'].map({'Yes':1, 'No':0})
# In[29]:
complaint['CompanyResponseToConsumer'] =
complaint['CompanyResponseToConsumer'].map({'Closed with explanation':1,
                                            'Closed with non-monetary relief':2,
                                            'Closed without relief':3,
                                            'Closed':4,
                                            'Closed with monetary relief':5,
                                            'Closed with relief':6})

# In[30]:
complaint['ConsumerDisputed'] = complaint['ConsumerDisputed'].map({'Yes':1, 'No':0})
# In[37]:
# Drop redundant columns
complaint =
complaint.drop(columns=['State','DateReceived','DateSentToCompany','SubmittedVia'])
complaint_wo_na = complaint[np.isfinite(complaint['CompanyResponseToConsumer'])]
complaint = complaint.drop(columns=['CompanyResponseToConsumer'])
# In[32]:
# Creating the required functions
def evaluateModel(estimator, X_test, y_test, name):
   """This function takes the tuned classifier and produces the accuracy, RSME, Confusion
Matrix, and
   Classification Report for the classifier supplied"""
   print("Classifer: ",name)
   print("\nAccuracy: {0:.4f}".format(estimator.score(X_test, y_test)))
   # Make prediction and print the confusion matrix
   y_pred = estimator.predict(X_test)
   # RSME value
   rmse = np.sqrt(mean_squared_error(y_test, y_pred))
   print("\nRoot Mean Squared Error: {0:.4f}".format(rmse))
   print('\nConfusion Matrix:')
   print(confusion_matrix(y_test, y_pred))
   print('\nClassification Report:')
   print(classification_report(y_test, y_pred))
# The following function is from:
# http://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                  n_jobs=1, train_sizes=np.linspace(.1, 1.0, 5)):
```

```
"""
Generate a simple plot of the test and traning learning curve.
Parameters
----------
estimator : object type that implements the "fit" and "predict" methods
    An object of that type which is cloned for each validation.
title : string
    Title for the chart.
X : array-like, shape (n_samples, n_features)
    Training vector, where n_samples is the number of samples and
    n_features is the number of features.
y : array-like, shape (n_samples) or (n_samples, n_features), optional
    Target relative to X for classification or regression;
    None for unsupervised learning.
ylim : tuple, shape (ymin, ymax), optional
    Defines minimum and maximum yvalues plotted.
cv : integer, cross-validation generator, optional
    If an integer is passed, it is the number of folds (defaults to 3).
    Specific cross-validation objects can be passed, see
    sklearn.cross_validation module for the list of possible objects
n_jobs : integer, optional
    Number of jobs to run in parallel (default 1).
"""
plt.figure()
plt.title(title)
if ylim is not None:
    plt.ylim(*ylim)
plt.xlabel("Training examples")
plt.ylabel("Score")
train_sizes, train_scores, test_scores = learning_curve(
    estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)
plt.grid()
plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                 train_scores_mean + train_scores_std, alpha=0.1,
                 color="r")
plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.1, color="g")
plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
         label="Training score")
plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
         label="Cross-validation score")
    plt.legend(loc="best")
    return plt
# ## Predicting Issues
# Create arrays for the predictors and the target variable
y = complaint['Issue'].values
X = preprocessing.scale(complaint.drop('Issue', axis=1).values)
# In[22]:
# Split into training and test set : we keep 80% of our data as training set and 20% as test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=123,
stratify=y)
# In[23]:
# KFold Cross-Validation
fold = KFold(len(X_train), n_folds=5, shuffle=True)
# In[24]:
# Decision Tree Classifier
# Create the classifier
decTree = DecisionTreeClassifier()
# Tuning hyper-parameters
params = {'criterion': ['gini', 'entropy']}
decTreeEst = GridSearchCV(estimator=decTree, cv=fold, param_grid=params)
# In[39]:
# Fit the classifer to the training data
decTreeEst.fit(X_train, y_train)
# In[26]:
# Creating the classifier with tuned hyper-parameter
bestDecTreeModel = DecisionTreeClassifier(criterion=decTreeEst.best_estimator_.criterion)
# In[27]:
# Evaluating the model
evaluateModel(decTreeEst,X_test, y_test, 'Decision Tree Classifier')
plot_learning_curve(bestDecTreeModel, 'Decision Tree Classifier', X_test, y_test)
# In[44]:
## Visualizing the Decision Tree
# Fit the best model
bestDecTreeModel.fit(X_train, y_train)
# Create DOT Data
dot_data = tree.export_graphviz(bestDecTreeModel, out_file=None,
                                feature_names=colName,
                                class_names=list(['1','2','3','4','5','6']))
# Create GraphViz object
graph = pydotplus.graph_from_dot_data(dot_data)
# Draw graph
Image(graph.create_png())
# In[59]:
# Score the best Decision Tree model
colName = complaint.columns
colName = colName[np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,15,16,17,18])]
pd.DataFrame(bestDecTreeModel.feature_importances_, index=colName,
        columns = ['Importance']).sort_values('Importance', ascending=False)
# In[33]:
# Random Forest Classifier
# Create the classifier
rForest = RandomForestClassifier()
# In[34]:
# Tuning hyper-parameters
params = {'n_estimators': [10, 50, 100, 200, 500], 'criterion': ['gini', 'entropy']}
rForestEst = GridSearchCV(estimator=rForest, cv=fold, param_grid=params)
# In[35]:
# Fit the classifer to the training data
rForestEst.fit(X_train, y_train)
# In[62]:
# Creating the classifier with tuned hyper-parameter
bestRandForestModel =
RandomForestClassifier(n_estimators=rForestEst.best_estimator_.n_estimators,
                       criterion=rForestEst.best_estimator_.criterion)
# In[63]:
# Score the best Random Forest model
bestRandForestModel.fit(X_train, y_train)
colName = complaint.columns
colName = colName[np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,15,16,17,18])]
pd.DataFrame(bestRandForestModel.feature_importances_,
        index=colName, columns = ['Importance']).sort_values('Importance', ascending=False)
# In[37]:
# Evaluating the model
evaluateModel(rForestEst,X_test, y_test, 'Random Forest Classifier')
plot_learning_curve(bestRandForestModel, 'Random Forest Classifier', X_test, y_test)
# ## Predicting Company Response to Consumer
# In[39]:
# Create arrays for the predictors and the target variable
y = complaint_wo_na['CompanyResponseToConsumer'].values
X = preprocessing.scale(complaint_wo_na.drop('CompanyResponseToConsumer',
axis=1).values)
# In[45]:
# Split into training and test set : we keep 80% of our data as training set and 20% as test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=123,
stratify=y)
# KFold Cross-Validation
fold = KFold(len(X_train), n_folds=5, shuffle=True)
# In[48]:
# Decision Tree Classifier
# Create the classifier
```

```
decTree = DecisionTreeClassifier()
# Tuning hyper-parameters
params = {'criterion': ['gini', 'entropy']}
decTreeEst = GridSearchCV(estimator=decTree, cv=fold, param_grid=params)
# Fit the classifer to the training data
decTreeEst.fit(X_train, y_train)
# Creating the classifier with tuned hyper-parameter
bestDecTreeModel = DecisionTreeClassifier(criterion=decTreeEst.best_estimator_.criterion)
# Predict with the best model
bestDecTreeModel.fit(X_train, y_train)
# In[50]:
# Evaluating the model: Decision Tree Classification
evaluateModel(decTreeEst,X_test, y_test, 'Decision Tree Classification')
plot_learning_curve(bestDecTreeModel, 'Decision Tree Classification', X_test, y_test)
# Score the best Random Forest model
colName = complaint_wo_na.columns
colName = colName[np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,18,19])]
pd.DataFrame(bestDecTreeModel.feature_importances_,
          index=colName, columns = ['Importance']).sort_values('Importance', ascending=False)
# In[51]:
# Random Forest Classifier
# Create the classifier
rForest = RandomForestClassifier()
# Tuning hyper-parameters
params = {'n_estimators': [10, 50, 100, 200, 500], 'criterion': ['gini', 'entropy']}
rForestEst = GridSearchCV(estimator=rForest, cv=fold, param_grid=params)
# Fit the classifer to teh training data
rForestEst.fit(X_train, y_train)
# Creating the classifier with tuned hyper-parameter
bestRForestModel =
RandomForestClassifier(n_estimators=rForestEst.best_estimator_.n_estimators,
                                criterion=rForestEst.best_estimator_.criterion)
# Predict with the best model
bestRForestModel.fit(X_train, y_train)
# In[53]:
# Evaluating the model: Random Forest Classification
evaluateModel(rForestEst,X_test, y_test, 'Random Forest Classification')
plot_learning_curve(bestRForestModel, 'Random Forest Classification', X_test, y_test)
# Score the best Random Forest model
colName = complaint_wo_na.columns
colName = colName[np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,18,19])]
pd.DataFrame(bestRForestModel.feature_importances_,
          index=colName, columns = ['Importance']).sort_values('Importance', ascending=False)
```

Python script for predicting mortgage issues (Objective 3)

```
# coding: utf-8
# In[43]:
# Importing the necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.cross_validation import KFold
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import learning_curve
from sklearn import tree
import pydotplus
import graphviz
from IPython.display import Image
# In[2]:
# Importing the dataset
url = 'C:\\Users\\Rusan\\Dropbox\\CapstoneProject\\DataSet\\TaxComplaint.csv'
complaint = pd.read_csv(url, parse_dates=['Date received', 'Date sent to company'])
complaint.info()
# In[3]:
# Renaming some of the columns
complaint = complaint.rename(columns = {'Date received':'DateReceived', 'Sub-
product':'SubProduct',
                                 'Company public response':'CompanyPublicResponse',
                                 'Consumer consent provided?':'ConsumerConsentProvided',
                                 'Submitted via':'SubmittedVia',
                                 'Date sent to company':'DateSentToCompany',
                                 'Company response to consumer':'CompanyResponseToConsumer',
                                 'Timely response?':'TimelyResponse',
                                 'Consumer disputed?':'ConsumerDisputed'})
# In[4]:
# Drop the columns with null values
complaint = complaint.drop(columns=['CompanyPublicResponse','ConsumerConsentProvided'])
# In[5]:
# check the number of features for each class in our dataset
pd.value_counts(complaint['SubProduct'], sort=True)
# In[6]:
```

```
pd.value_counts(complaint['Issue'], sort=True)
# In[7]:
pd.value_counts(complaint['Company'], sort=True)
# In[8]:
# Just taking top 5 companies with most complaints for the analysis
complaint = complaint[(complaint['Company']=='BANK OF AMERICA, NATIONAL
ASSOCIATION')|
        (complaint['Company']=='WELLS FARGO & COMPANY')|
        (complaint['Company']=='OCWEN LOAN SERVICING LLC')|
        (complaint['Company']=='JPMORGAN CHASE & CO.')|
        (complaint['Company']=='NATIONSTAR MORTGAGE')]
# In[9]:
pd.value_counts(complaint['TimelyResponse'], sort=True)
# In[10]:
pd.value_counts(complaint['CompanyResponseToConsumer'], sort=True)
# In[11]:
pd.value_counts(complaint['ConsumerDisputed'], sort=True)
# In[12]:
# Replacing the dataset categorical variables into numeric codes
complaint['month'] = complaint['DateReceived'].dt.month
# In[13]:
complaint['Issue'] = complaint['Issue'].map({'Loan modification,collection,foreclosure': 1,
        'Loan servicing, payments, escrow account': 2, 'Application, originator, mortgage broker':3,
        'Settlement process and costs':4,'Credit decision / Underwriting':5,'Other':6})
# In[14]:
complaint['SubProduct'] = complaint['SubProduct'].map({'Second mortgage': 1,
        'Reverse mortgage': 2, 'VA mortgage':3,
        'FHA mortgage':4,'Conventional adjustable mortgage (ARM)':5,'Conventional fixed
mortgage':6,
        'Home equity loan or line of credit':7,'Other mortgage':8})
# In[15]:
complaint['Company'] = complaint['Company'].map({'BANK OF AMERICA, NATIONAL
ASSOCIATION':1,
        'WELLS FARGO & COMPANY':2,'OCWEN LOAN SERVICING LLC':3, 'JPMORGAN
CHASE & CO.':4,
        'NATIONSTAR MORTGAGE':5})
# In[16]:
complaint['TimelyResponse'] = complaint['TimelyResponse'].map({'Yes':1, 'No':0})
# In[17]:
complaint['CompanyResponseToConsumer'] =
complaint['CompanyResponseToConsumer'].map({'Closed with explanation':1,
                                                    'Closed with non-monetary relief':2,
                                                    'Closed without relief':3,
                                                    'Closed':4,
                                                    'Closed with monetary relief':5,
                                                        'Closed with relief':6})
# In[18]:
complaint['ConsumerDisputed'] = complaint['ConsumerDisputed'].map({'Yes':1, 'No':0})
# In[19]:
# Drop redundant columns
complaint =
complaint.drop(columns=['State','DateReceived','DateSentToCompany','SubmittedVia','Compan
yResponseToConsumer'])
# In[20]:
# Creating the required functions
def evaluateModel(estimator, X_test, y_test, name):
    """This function takes the tuned classifier and produces the accuracy, RSME, Confusion
Matrix, and
    Classification Report for the classifier supplied"""
    print("Classifer: ",name)
    print("\nAccuracy: {0:.4f}".format(estimator.score(X_test, y_test)))
    # Make prediction and print the confusion matrix
    y_pred = estimator.predict(X_test)
    # RSME value
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    print("\nRoot Mean Squared Error: {0:.4f}".format(rmse))
    print('\nConfusion Matrix:')
    print(confusion_matrix(y_test, y_pred))
    print('\nClassification Report:')
    print(classification_report(y_test, y_pred))
# The following function is from:
# http://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=1, train_sizes=np.linspace(.1, 1.0, 5)):
    """
    Generate a simple plot of the test and traning learning curve.
    Parameters
    ----------
    estimator : object type that implements the "fit" and "predict" methods
        An object of that type which is cloned for each validation.
    title : string
        Title for the chart.
    X : array-like, shape (n_samples, n_features)
        Training vector, where n_samples is the number of samples and
        n_features is the number of features.
    y : array-like, shape (n_samples) or (n_samples, n_features), optional
        Target relative to X for classification or regression;
        None for unsupervised learning.
    ylim : tuple, shape (ymin, ymax), optional
        Defines minimum and maximum yvalues plotted.
    cv : integer, cross-validation generator, optional
        If an integer is passed, it is the number of folds (defaults to 3).
        Specific cross-validation objects can be passed, see
        sklearn.cross_validation module for the list of possible objects
    n_jobs : integer, optional
        Number of jobs to run in parallel (default 1).
    """
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()
    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")
    plt.legend(loc="best")
    return plt
# In[21]:
# Create arrays for the predictors and the target variable
y = complaint['Issue'].values
X = preprocessing.scale(complaint.drop('Issue', axis=1).values)
# In[22]:
# Split into training and test set : we keep 80% of our data as training set and 20% as test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=123,
stratify=y)
# In[23]:
# KFold Cross-Validation
fold = KFold(len(X_train), n_folds=5, shuffle=True)
# In[24]:
# Decision Tree Classifier
# Create the classifier
```

```
decTree = DecisionTreeClassifier()
# Tuning hyper-parameters
params = {'criterion': ['gini', 'entropy']}
decTreeEst = GridSearchCV(estimator=decTree, cv=fold, param_grid=params)
# In[39]:
# Fit the classifer to the training data
decTreeEst.fit(X_train, y_train)
# In[26]:
# Creating the classifier with tuned hyper-parameter
bestDecTreeModel = DecisionTreeClassifier(criterion=decTreeEst.best_estimator_.criterion)
# In[27]:
# Evaluating the model
evaluateModel(decTreeEst,X_test, y_test, 'Decision Tree Classifier')
plot_learning_curve(bestDecTreeModel, 'Decision Tree Classifier', X_test, y_test)
# In[44]:
## Visualizing the Decision Tree
# Fit the best model
bestDecTreeModel.fit(X_train, y_train)
# Create DOT Data
dot_data = tree.export_graphviz(bestDecTreeModel, out_file=None,
                                feature_names=colName,
                                class_names=list(['1','2','3','4','5','6']))
# Create GraphViz object
graph = pydotplus.graph_from_dot_data(dot_data)
# Draw graph
Image(graph.create_png())
# In[59]:
# Score the best Decision Tree model
colName = complaint.columns
colName = colName[np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,15,16,17,18])]
pd.DataFrame(bestDecTreeModel.feature_importances_, index=colName,
        columns = ['Importance']).sort_values('Importance', ascending=False)
# In[33]:
# Random Forest Classifier
# Create the classifier
rForest = RandomForestClassifier()
# In[34]:
# Tuning hyper-parameters
params = {'n_estimators': [10, 50, 100, 200, 500], 'criterion': ['gini', 'entropy']}
rForestEst = GridSearchCV(estimator=rForest, cv=fold, param_grid=params)
# In[35]:
# Fit the classifer to the training data
rForestEst.fit(X_train, y_train)
# In[62]:
# Creating the classifier with tuned hyper-parameter
```

```
bestRandForestModel =
RandomForestClassifier(n_estimators=rForestEst.best_estimator_.n_estimators,
                                    criterion=rForestEst.best_estimator_.criterion)
# In[63]:
# Score the best Random Forest model
bestRandForestModel.fit(X_train, y_train)
colName = complaint.columns
colName = colName[np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,15,16,17,18])]
pd.DataFrame(bestRandForestModel.feature_importances_,
          index=colName, columns = ['Importance']).sort_values('Importance', ascending=False)
# In[37]:
# Evaluating the model
evaluateModel(rForestEst,X_test, y_test, 'Random Forest Classifier')
plot_learning_curve(bestRandForestModel, 'Random Forest Classifier', X_test, y_test)
```