Capstone 2 Project Proposal - Mortgage Loan Type Prediction

Alexandra Michel

Background/Motivation
There are many features that go into deciding whether someone will be approved or denied a home loan. There are different programs made available by different lenders, and a consumer may qualify for one or more programs based on their financial profile. While we cannot assume that historical data supports which program is the *best* option for a client, we can use it to predict which program will likely end up being the program they choose. For a system that shows a user all the loan options they may qualify for, it could cut down on time spent sifting through loan programs if we could predict with high probability which loan program is the most likely option, and present that to the consumer while sifting through the rest of the options to present.

Data Sources and Methods
Source of original data investigation was this dataset www.kaggle.com/dinu1763/mortgage-loan-approval/data# whose original source is from the Home Mortgage Disclosure Act (HMDA) government website. “The HMDA requires many financial institutions to maintain, report, and publicly disclose loan-level information about mortgages ... The public data are modified to protect applicant and borrower privacy.”
I started out an initial investigation using the Kaggle dataset (which was already pretty clean) but since the HMDA dataset is raw and much larger, I will focus my project on cleaning and preparing that set to use for my model.

The Data set
Each row in this data set is a closed loan transaction, and it contains attributes about the client like **`income, loan_amount, loan_purpose, debt_to_income_ratio, occupancy`**; attributes external to the client like **`population, minority_population_pct, median_family_income`**; and the target variable in this project that the model with look to predict: **`loan_type`**. I was able to clarify what the codes meant for each attribute by using this glossary provided by the HMDA: https://ffiec.cfpb.gov/documentation/2019/lar-data-fields/

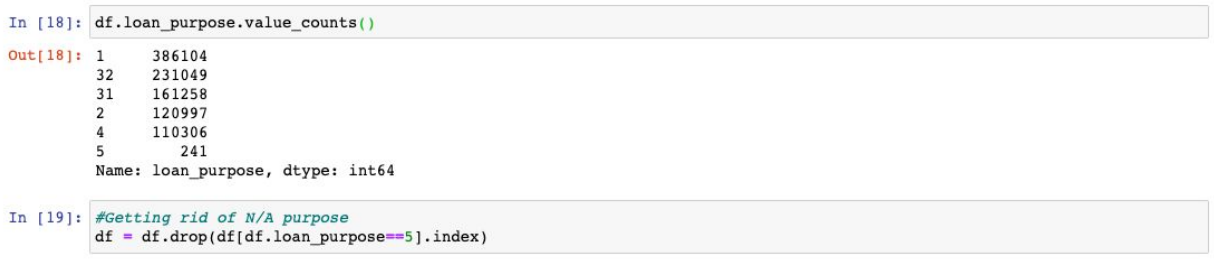
Data Cleaning
Before I began cleaning the data, I simplified the problem by making this a binary classification between FHA and Conventional loan programs. This is mainly because the other two programs, USDA and VA, are not as common in the dataset making up around 1% of the entries, and there are other ways of predicting that a consumer would be after one of these loan types. USDA loans are for rural properties and VA loans are available to veterans who meet certain criteria with the veterans administration.

We have to keep in mind that this prediction would be used once a consumer provides minimal information, and they would be looking to purchase or refinance a residential property. The information we would have initially from the client includes:
- client address,
- approximate income,
- total existing debt balances and payments,
- occupancy (primary, second or investment),
- loan purpose (purchase, refinance, cash-out refinance),
- lien status (applying for a 1st or 2nd mortgage),
- max loan amount, and whether it’s conforming or not

This prediction of loan type would ultimately come before pricing out rate or term information, so that won't be an input to the model. Also, we don't initially know the race, ethnicity, age or sex information so that will be dropped for now. I will likely over-simplify the input values in the model at first to see how the simple model runs, and may add attributes back in if I see how to gather that information from a client initially. If there end up being important features aside from these, we can try to extrapolate more information from what we know, or ask the client to provide more information.

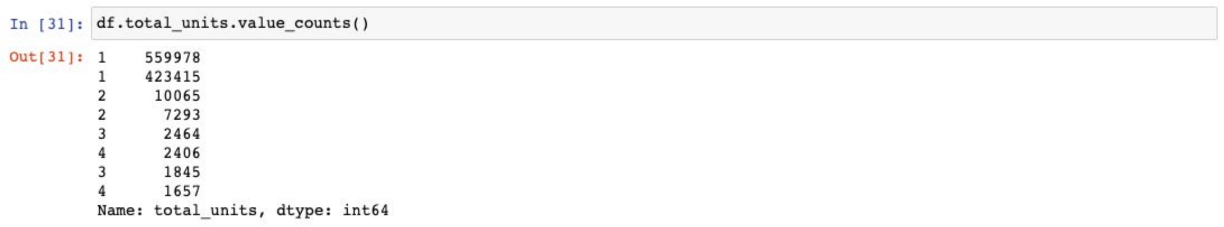
From industry knowledge, I know the debt-to-income (DTI) ratio is going to be a good distinguishing factor, so I dropped the rows which do not have that information present.
(I could fill this with an average value, but since we have over 1.6M entries in the dataset and only 12,723 have an "Exempt" value for DTI, I will just drop for now.)
I also got rid of reverse or business/commercial loans which were included in the original dataset, as this is strictly a model for residential mortgages that fully amortize.

Next, I went through each attribute and checked out the distribution of data in each to see if there was anything else I could clean up. I found a few more attributes that didn’t have much to contribute to the model, since the majority of the entries were skewed to one or more categories of the attribute.

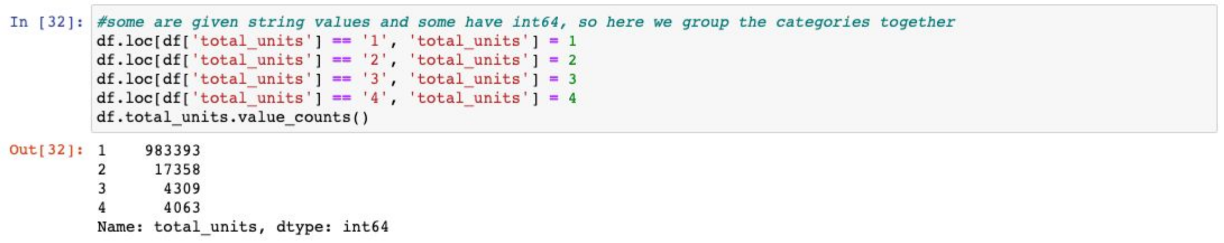


```
In [18]: df.loan_purpose.value_counts()

Out[18]: 1     386104
         32    231049
         31    161258
         2     120997
         4     110306
         5        241
         Name: loan_purpose, dtype: int64

In [19]: #Getting rid of N/A purpose
         df = df.drop(df[df.loan_purpose==5].index)
```

For example, with `loan_purpose` there weren’t many with ‘Not applicable’ and that doesn’t give good information about the loan purpose anyway, so I got rid of those entries. I also found a few minor things to clean up, like the `total_units` and `debt_to_income_ratio` features. For `total_units`, some entries had a string value and some had an int, so I converted all to ints.

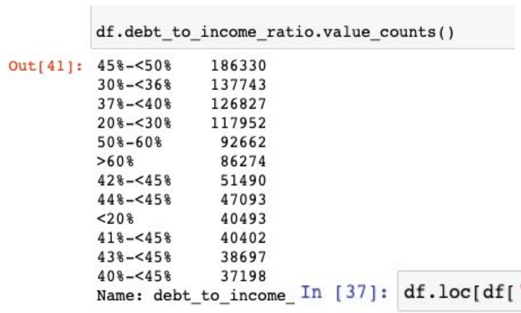


```
In [31]: df.total_units.value_counts()

Out[31]: 1    559978
         1    423415
         2     10065
         2      7293
         3      2464
         4      2406
         3      1845
         4      1657
         Name: total_units, dtype: int64
```

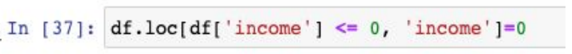


```
In [32]: #some are given string values and some have int64, so here we group the categories together
         df.loc[df['total_units'] == '1', 'total_units'] = 1
         df.loc[df['total_units'] == '2', 'total_units'] = 2
         df.loc[df['total_units'] == '3', 'total_units'] = 3
         df.loc[df['total_units'] == '4', 'total_units'] = 4
         df.total_units.value_counts()

Out[32]: 1    983393
         2     17358
         3      4309
         4      4063
         Name: total_units, dtype: int64
```

For `debt_to_income_ratio`, there were too many categories because only some were binned into ranges of percentages. I binned the rest to come up with these categories for the data:

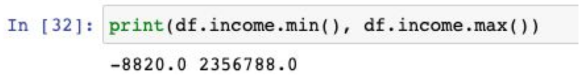


```
          df.debt_to_income_ratio.value_counts()

Out[41]:  45%-<50%    186330
          30%-<36%    137743
          37%-<40%    126827
          20%-<30%    117952
          50%-60%      92662
          >60%         86274
          42%-<45%     51490
          44%-<45%     47093
          <20%         40493
          41%-<45%     40402
          43%-<45%     38697
          40%-<45%     37198
          Name: debt_to_income_
```



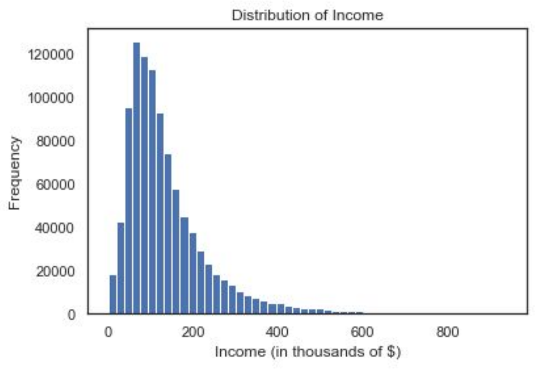
```
In [37]: df.loc[df['income'] <= 0, 'income']=0
```

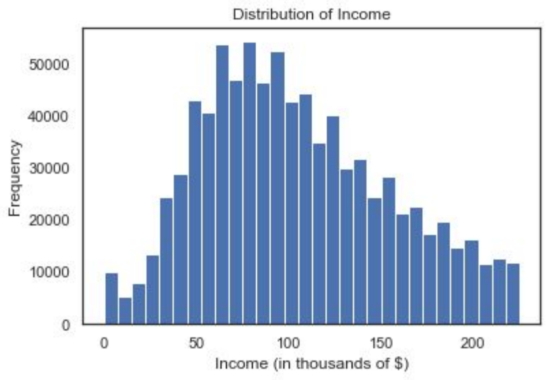
The `income` attribute ended up having a much wider range of values than I thought it would have. I'm not sure why there would be a negative income value in the dataset, but in order to make sense of it when we go to apply it to another consumer dataset, I'm going to change these income values to '0.0'.



```
In [32]: print(df.income.min(), df.income.max())

         -8820.0 2356788.0
```

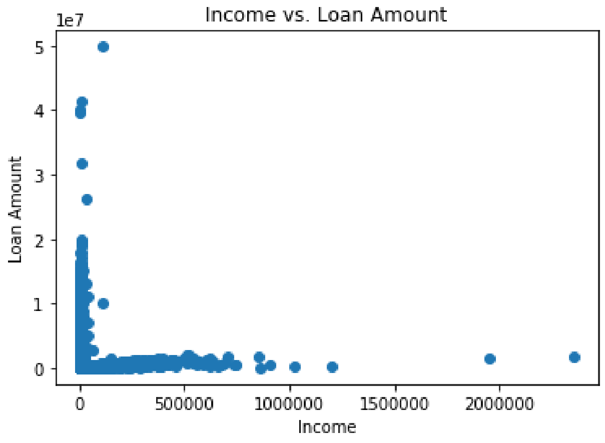
On the higher end of this range, I saw that there were over 1600 entries with income values of over $5,000,000/year based on the fact that this is supposed to be measured in thousands of dollars. The following figures show the full distribution of income, and the distribution of the lowest 900,000 income values to get a closer look at it.

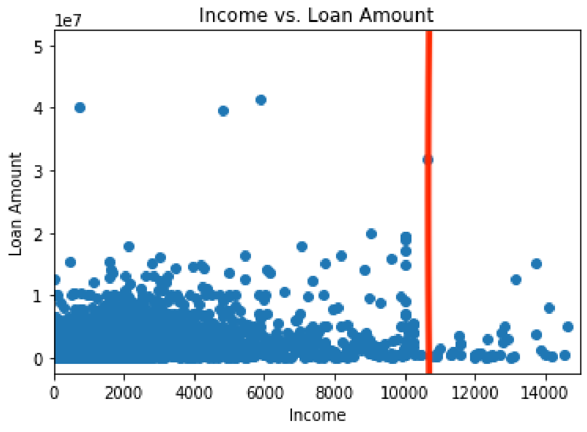




Exploratory Data Analysis

Here I was most interested in looking at the continuous values plotted against each other to see if there were patterns that looked interesting. I found that there were applicants with very large incomes applying for loan amounts that were below 1% of what their annual income supposedly was.





After a lot of investigating, I determined that it made the most sense to take the entries that reported more than $5,000,000/yr in annual gross income, and scale the income down by 1,000 as a correction. Seeing as even the top 1% of Californians earn on average $1.7M/yr[1], with a lower threshold of $500K to be in that 1%, it seemed unreasonable to even have entries with income higher than $3M/yr. Once this was cleaned up,

- The average loan_amount is: **$362308.42** and the median loan_amount is: **$285000.0**
- The mean income is: **$154270.0/yr** and the median income is: **$110000.0/yr**

Which is much more reflective of California average income, rather than the average of $340K I was seeing beforehand.

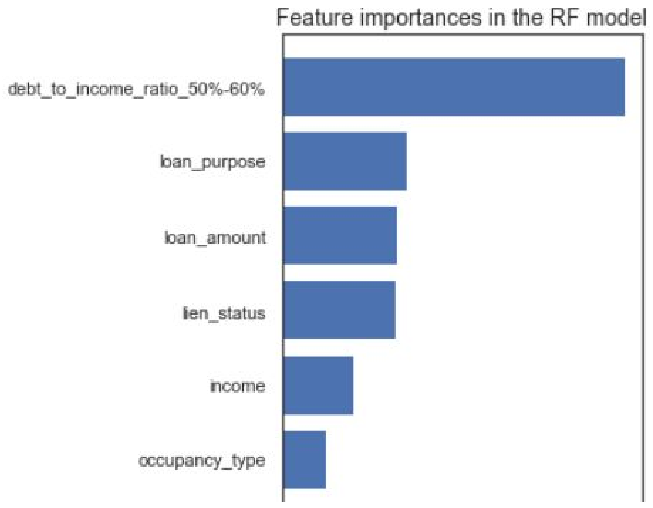
[1] https://www.epi.org/multimedia/unequal-states-of-america/#/California

I know there is probably more I could do here, but this was really the bulk of my investigation before heading into the ML modeling part of the project!

RandomForest Model

I began by splitting the data into independent attributes 'X' and the dependent attribute loan_type as 'y'. Then, I split both into training (80%) and test (20%) sets. I started the parameter tuning process by first just running the RandomForestClassifier model with n_estimators=100 and max_depth=10 to get a baseline look at how the model did. I measured how well it did by using a ROC curve and computing the area under the curve to see how many True/False positives the model predicts against the test data. From that first run through, I got the following as the most important features making up 85% of the prediction:

debt_to_income_ratio_50%-60%	0.358624
loan_purpose	0.131003
loan_amount	0.120223
lien_status	0.118441
income	0.075409
occupancy_type	0.0465



It's slightly unsurprising to me that the most important feature involves the loans for applicants with DTI between 50-60%, though I'm glad to see it reflected in the results. Conventional loans tend to have a more conservative DTI limit on approvals, and the loans being approved at 50-60% are most likely all FHA. This is giving the model good information to determine when to put the loan in the "FHA bucket". The other top 5 features are also intuitive as to why they would be important.

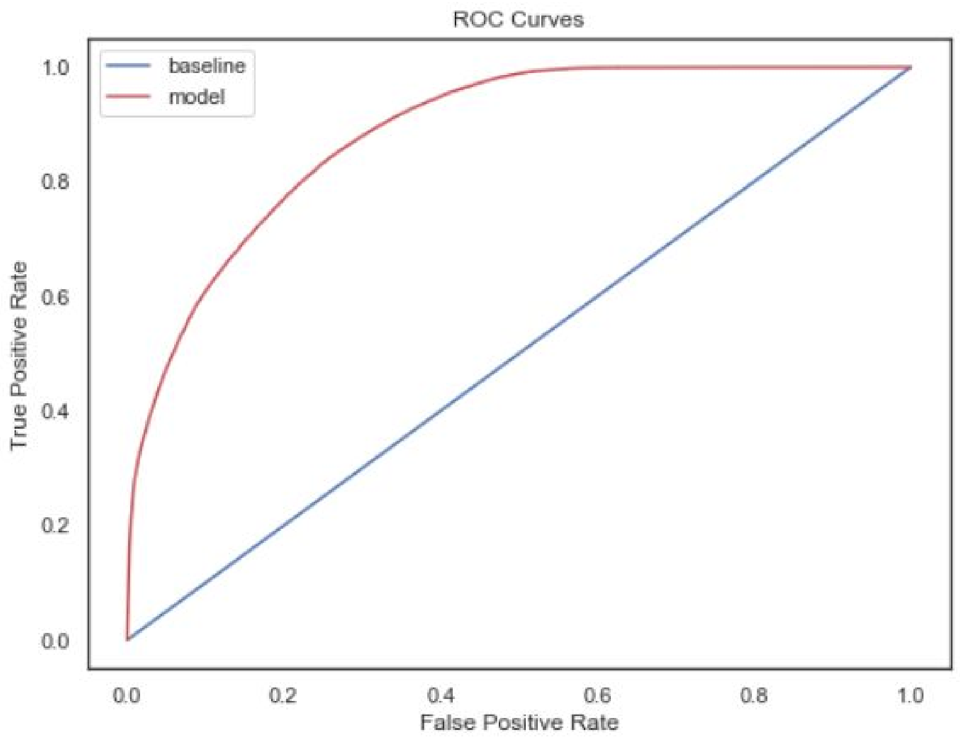
The ROC score was 89% on both the training and test data for this first run. I played around with taking away some less important features which could be information that is a bit harder to access (like knowing if the loan the consumer will end up with will have an interest-only component, or knowing exactly how many units the property will have ahead of time). When I dropped less important features I didn't see any change in accuracy and was able to simplify the model.

Ultimately, I arrived at using the following features for the model:
`-loan_amount`
`-income`
`-debt_to_income`
`-lien_status`
`-loan_purpose`
`-construction_method`
`-conforming_loan_limit`

The only information I dropped from the model which could possibly help it perform better is the **`ffiec_msa_md_median_family_income`**, **`tract_to_msa_income_percentage`**, and the **`tract_minority_population_percent`**. These are data points which depend on the tract and location of the subject property, so I'm not sure if they are attainable ahead of time, unless the assumption is made that the consumer currently lives in the same area or tract as the home they are looking to purchase. In the case of refinancing, this information may be easier to pull.

With n_estimators=500 and max_depth=20, the RandomForestClassifier performed with a 0.89 ROC score on the test data and 0.93 ROC score on the training data, using the area under the ROC curve as the measure. I also measured the area under the precision-recall curve, which quantifies the number of correct positive predictions made.

```
Precision Baseline: 0.9 Test: 0.93 Train: 0.93
Roc Baseline: 0.5 Test: 0.89 Train: 0.93
```



Next steps:

- Explore how to possibly incorporate the attributes I took out that didn't have low importances
- Look at an actual dataset with the same features on transactions from one mortgage brokerage, and create an ROC curve against that as the test data to see how well the model does.
- It may not make sense to have a determined loan amount at the step of the application process before this prediction happens, but if I can calculate an estimate of the max loan amount, then it could still work well.

Link to my GitHub repository:
https://github.com/xandramichel/Springboard-Capstone-2/blob/master/Capstone%202%20HMDA.ipynb