Predicting Mortgage Approvals From Government Data

Ioannis Tolios

April 25, 2019

1 Executive Summary

In this report I am going to present my analysis of the HMDA mortgage application dataset, as well as the machine learning model I created for the aforementioned data. The dataset includes information about 500000 mortage applications. After doing basic exploratory analysis, visualization, as well as some data cleaning and processing, I managed to identify the most essential features of the dataset, i.e. those that provided the largest amount of information about the separation of the two label classes (mortgage accepted/rejected). Those features were subsequentely used for the creation of a binary classifier capable of predicting whether an application will be accepted or not, with an accuracy of 72%. The most important features of the dataset are the following:

- *lender* - A categorical with no ordering indicating which of the lenders was the authority in approving or denying this loan
- *loan_amount* - Size of the requested loan in thousands of dollars
- *msa_md* - A categorical with no ordering indicating Metropolitan Statistical Area/Metropolitan Division
- *state_code* - A categorical with no ordering indicating the U.S. state
- *applicant_income* - In thousands of dollars
- *number_of_owner-occupied_units* - Number of dwellings, including individual condominiums, that are lived in by the owner
- *minority_population* - Number of people that belong in a minority group, a new feature that was created for the purposes of this analysis
- *tract_family_income* - The tract median family income in dollars, another feature that was created for the purposes of this analysis

2 Exploratory Data Analysis

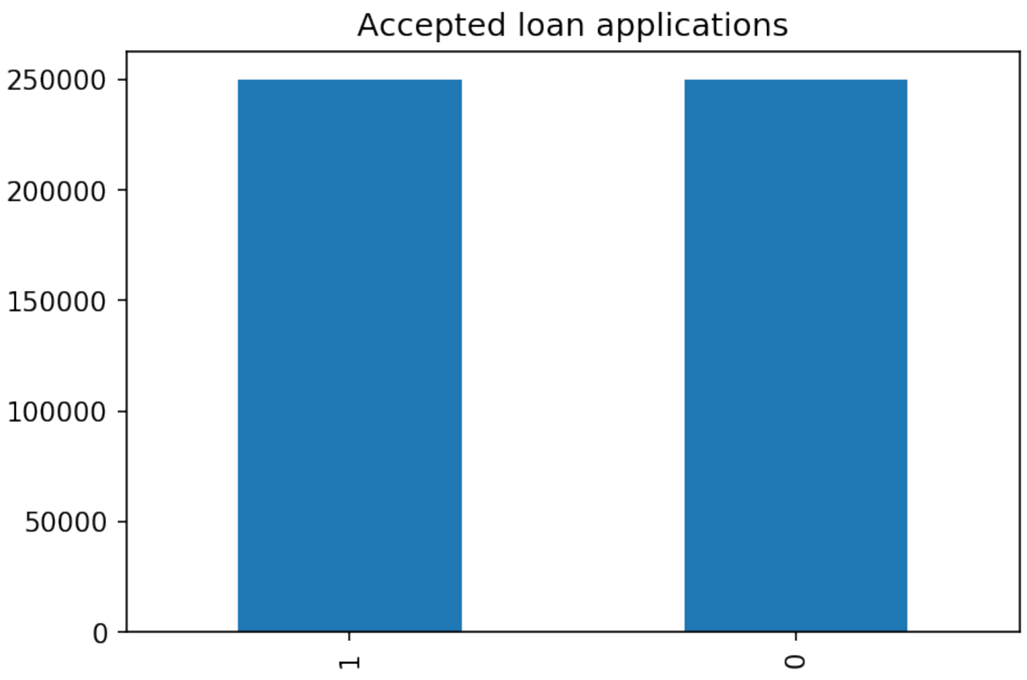
First of all, we are going to examine the total number of non-null values for each feature.

```
Out[37]: row_id                            500000
         loan_type                         500000
         property_type                     500000
         loan_purpose                      500000
         occupancy                         500000
         loan_amount                       500000
         preapproval                       500000
         msa_md                            500000
         state_code                        500000
         county_code                       500000
         applicant_ethnicity               500000
         applicant_race                    500000
         applicant_sex                     500000
         applicant_income                  460052
         population                        477535
         minority_population_pct           477534
         ffiecmedian_family_income         477560
         tract_to_msa_md_income_pct        477486
         number_of_owner-occupied_units    477435
         number_of_1_to_4_family_units     477470
         lender                            500000
         co_applicant                      500000
         row_id                            500000
         accepted                          500000
         dtype: int64
```

As we can see, some of the features have a fair number of missing values. Furthermore, we can read on the dataset description that some categorical features include categories that correspond to missing values, e.g. the -1 category of *msa_md* column indicates a missing value. We are going to remove those, so we have a clearer view of our dataset.

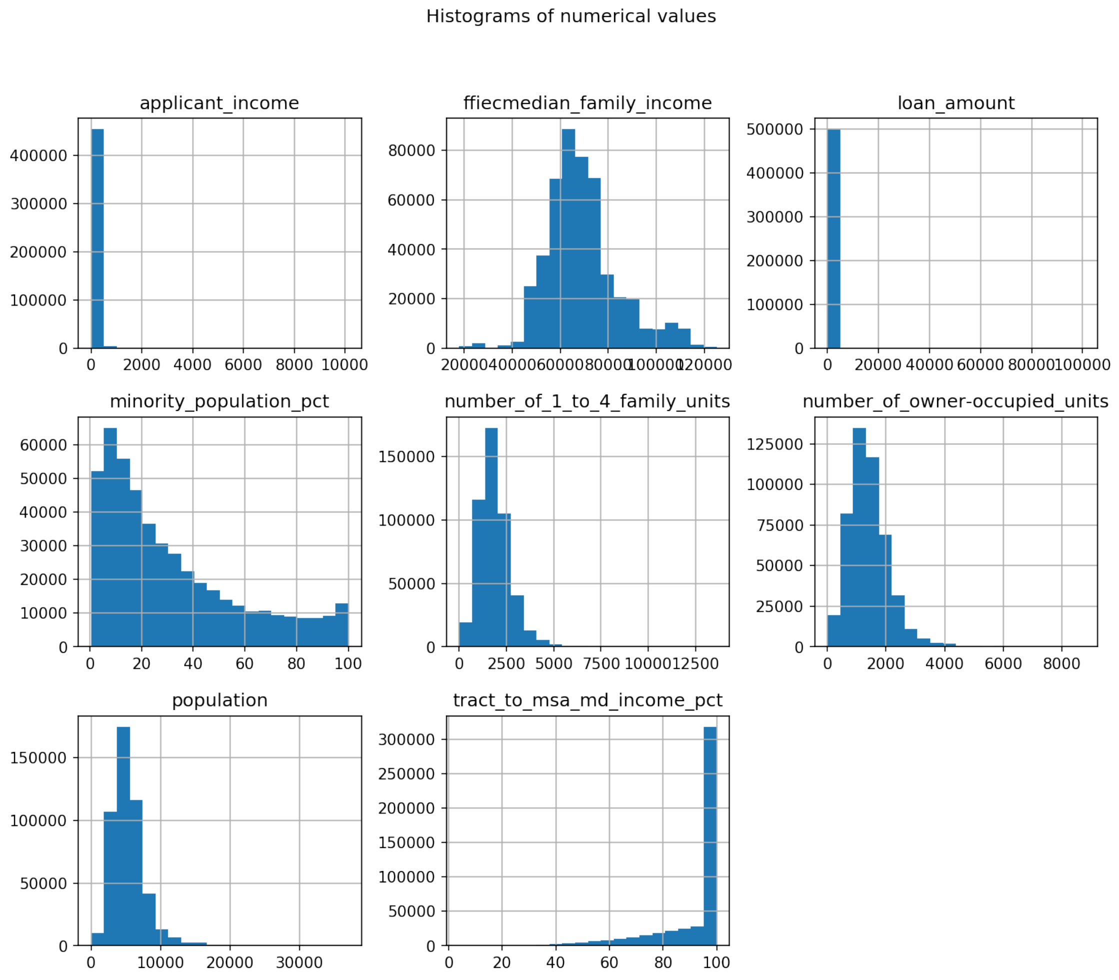
```
Out[38]: loan_type                        500000
         property_type                    500000
         loan_purpose                     500000
         occupancy                        497811
         loan_amount                      500000
         preapproval                       88891
         msa_md                           423018
         state_code                       480868
         county_code                      479534
         applicant_ethnicity              436883
         applicant_race                   434460
         applicant_sex                    458682
         applicant_income                 460052
         population                       477535
         minority_population_pct          477534
         ffiecmedian_family_income        477560
         tract_to_msa_md_income_pct       477486
         number_of_owner-occupied_units   477435
         number_of_1_to_4_family_units    477470
         lender                           500000
         co_applicant                     500000
         accepted                         500000
         dtype: int64
```

After removing the aforementioned categories, we see that certain features have significantly fewer values. This has to be dealt with before we create the binary classifier, as machine learning models can't accept null values on a dataset. There are various ways of dealing with this problem, such as imputing missing values with the mean/median of the feature, or more advanced methods like the K-Nearest Neighbors algorithm. One feature (*preapproval*) will be dropped, because it has a very large number of missing values.



```
Out[39]: 1    0.500228
         0    0.499772
         Name: accepted, dtype: float64
```

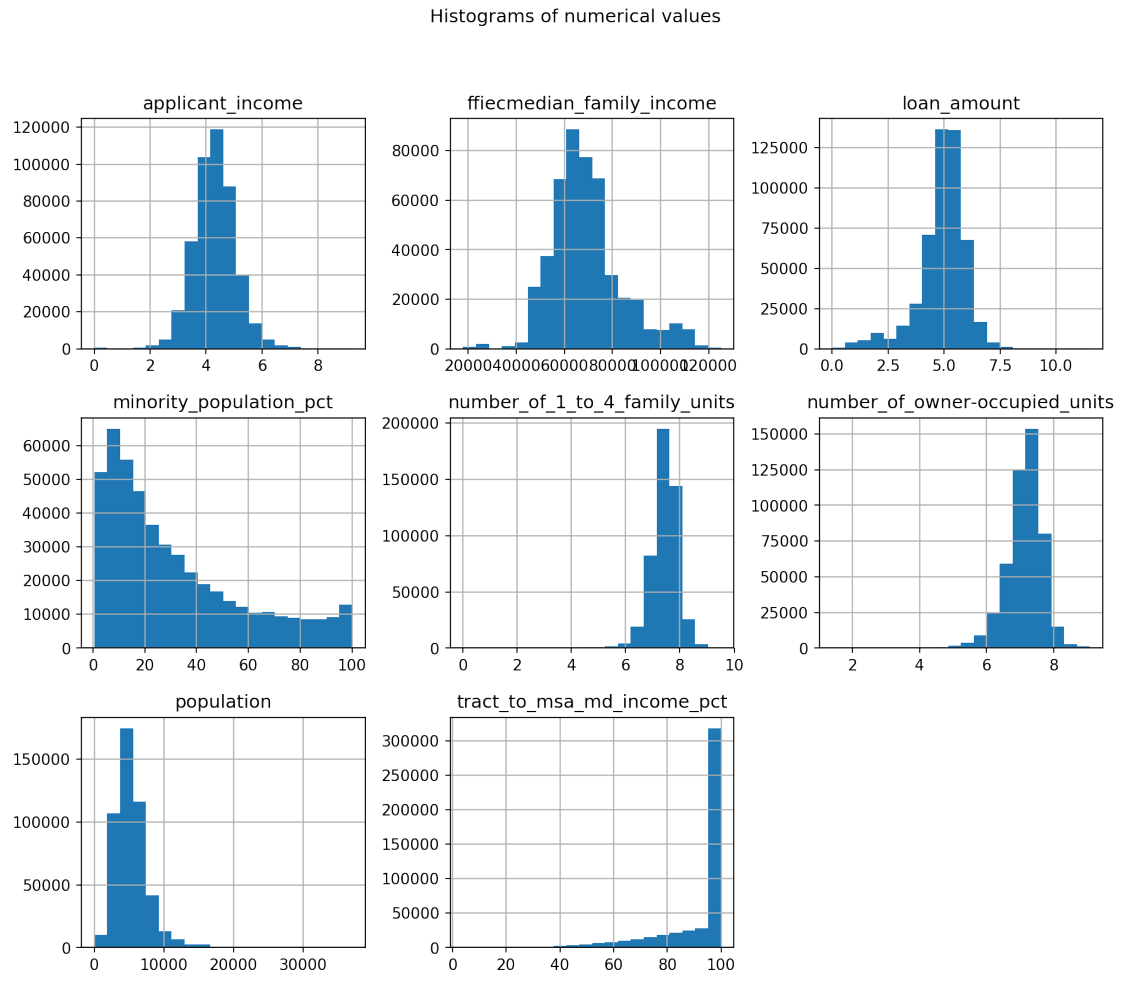
The classes of our dataset are almost perfectly balanced. This is helpful, as it simplifies the creation of the machine learning model. Most classifiers perform better, and have a higher accuracy when the classes are balanced.



```
Skewness of numerical columns:
```

```
Out[40]: loan_amount                        76.552786
         applicant_income                   22.277181
         population                          2.864237
         minority_population_pct             1.009139
         ffiecmedian_family_income           0.773280
         tract_to_msa_md_income_pct         -1.963872
         number_of_owner-occupied_units      1.881743
         number_of_1_to_4_family_units       2.016264
         dtype: float64
```

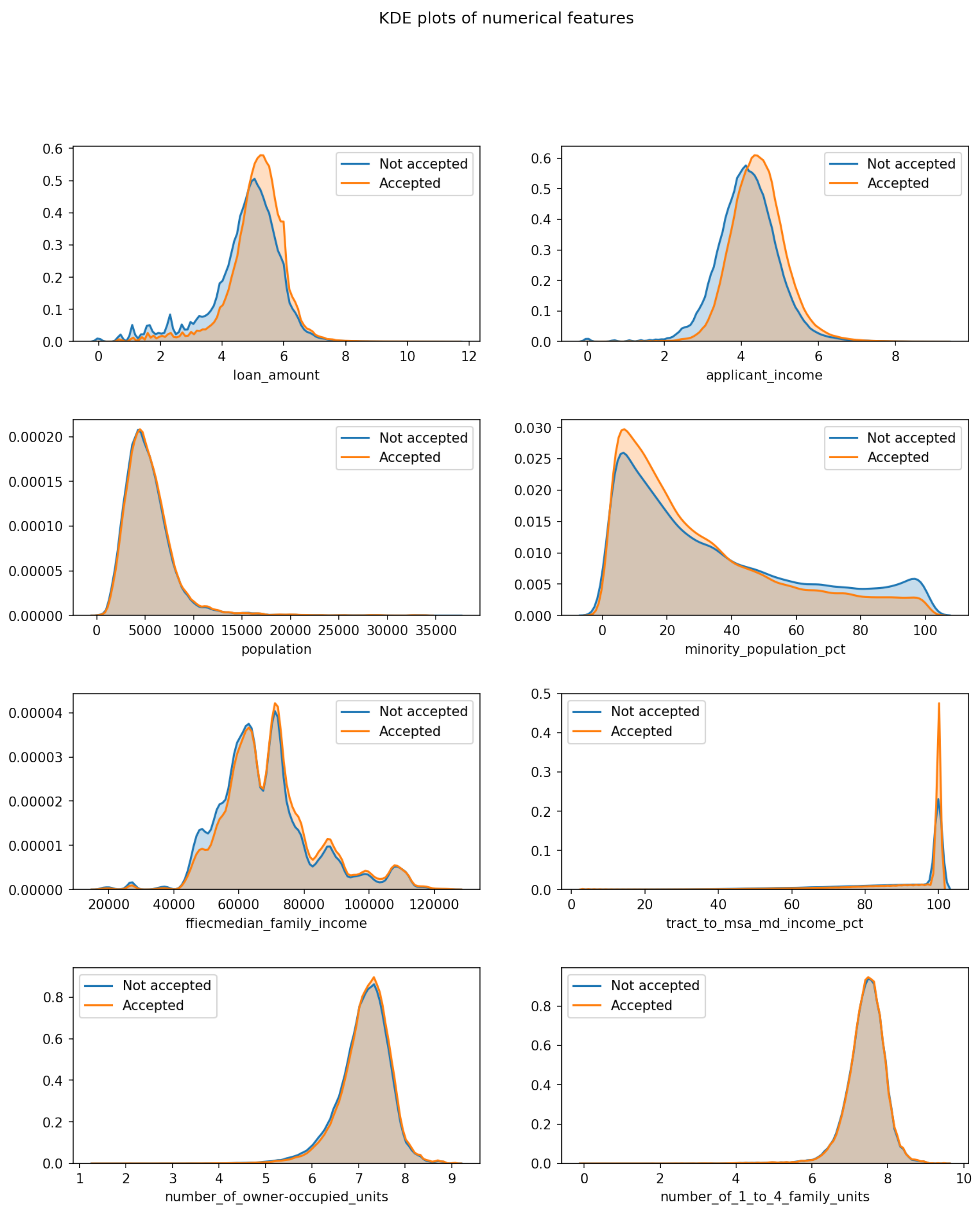
As we can see on the histograms, as well as the list of skewness values, a few of the numerical features, especially *loan_amount* and *applicant_income* are highly skewed. We are going to fix that by applying the logarithmic function on them, as it is helpful to have features with low skewness and a distribution that is close to normal. This will enable us to create more meaningful visualizations, as well as a better machine learning model.



```
Skewness of numerical columns after applying log function:
```

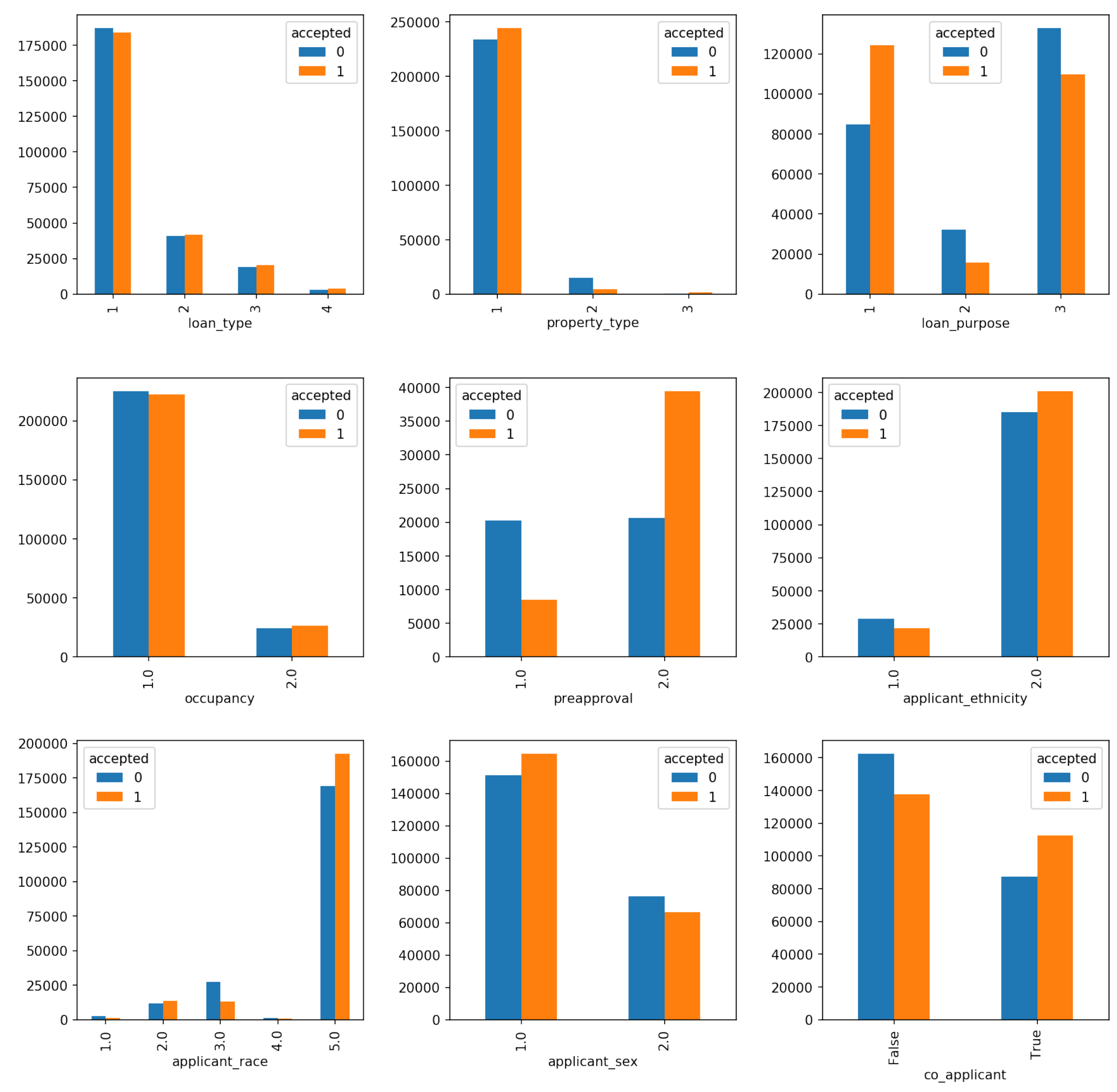
```
Out[41]: loan_amount                       -1.190548
         applicant_income                   0.026275
         population                         2.864237
         minority_population_pct            1.009139
         ffiecmedian_family_income          0.773280
         tract_to_msa_md_income_pct        -1.963872
         number_of_owner-occupied_units    -1.076900
         number_of_1_to_4_family_units     -1.568488
         dtype: float64
```

We can see that after applying the logarithmic function, the numerical features are significantly less skewed, and their distribution is closer to normal as expected.

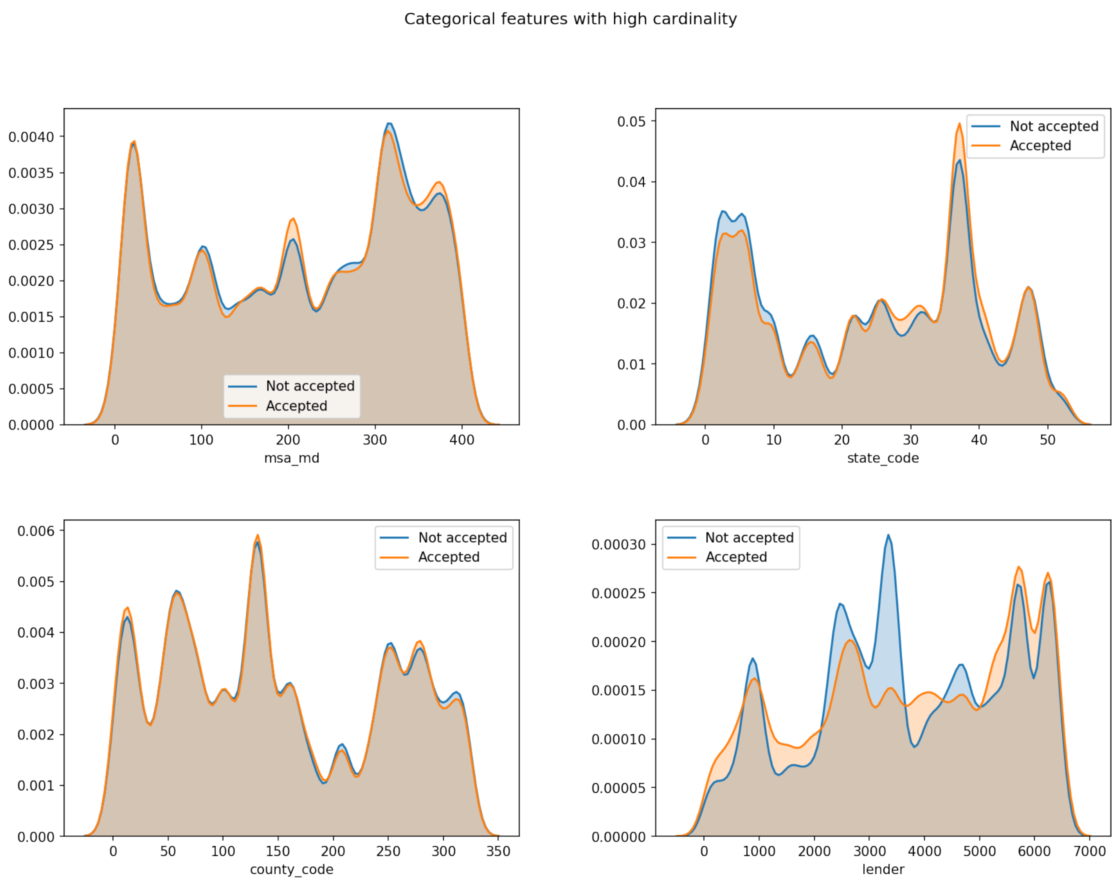


Next, we're going to examine the KDE plots of the numerical features. I created different plots for each class of the label, so we can visually determine which features provide more information about the separation of those classes. As we can see *loan_amount*, *applicant_income*, *minority_population_pct* and *ffiecmedian_family_income* are the most important numerical features, and will be used for the creation of the binary classifier.

Categorical features with low cardinality

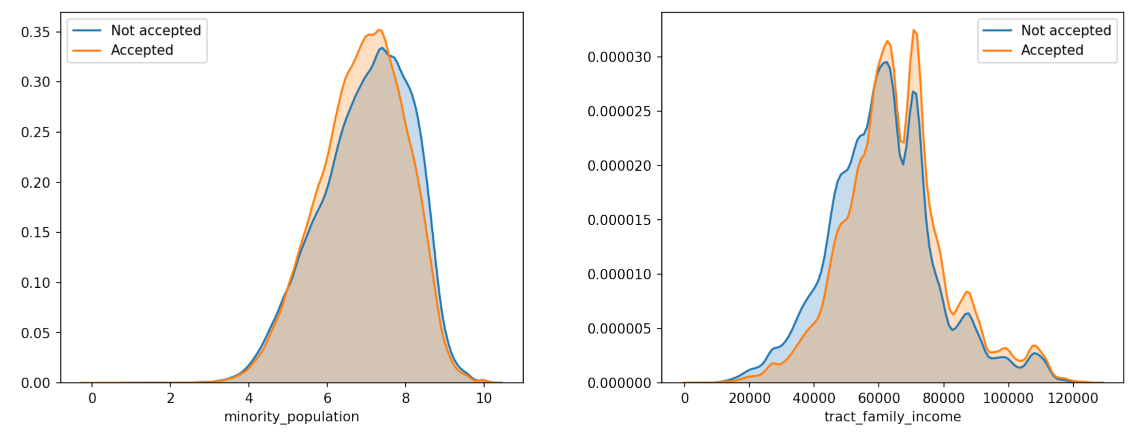


We are now going to visually examine class separation on the categorical features. I divided those features in two groups, depending on the number of categories, i.e. low and high cardinality. I decided to do this for a number of reasons, including the fact that features with low cardinality can be visualized using bar plots, while this isn't appropriate for those with high cardinality. As we can see, most of the low-cardinality features provide information about class separation, except for *occupancy*. It must be noted that there are some evident signs of discrimination, specifically based on sex and race. The mortgage applications of white citizens had a higher approval rate compared to african americans. Furthermore, women had a lower approval rate compared to men. Making inferences and trying to identify any causes for those differences, is beyond the scope of this analysis. Regardless of that, using data that is influenced by prejudice and discrimination, raises ethical concerns that will be examined later in further detail.

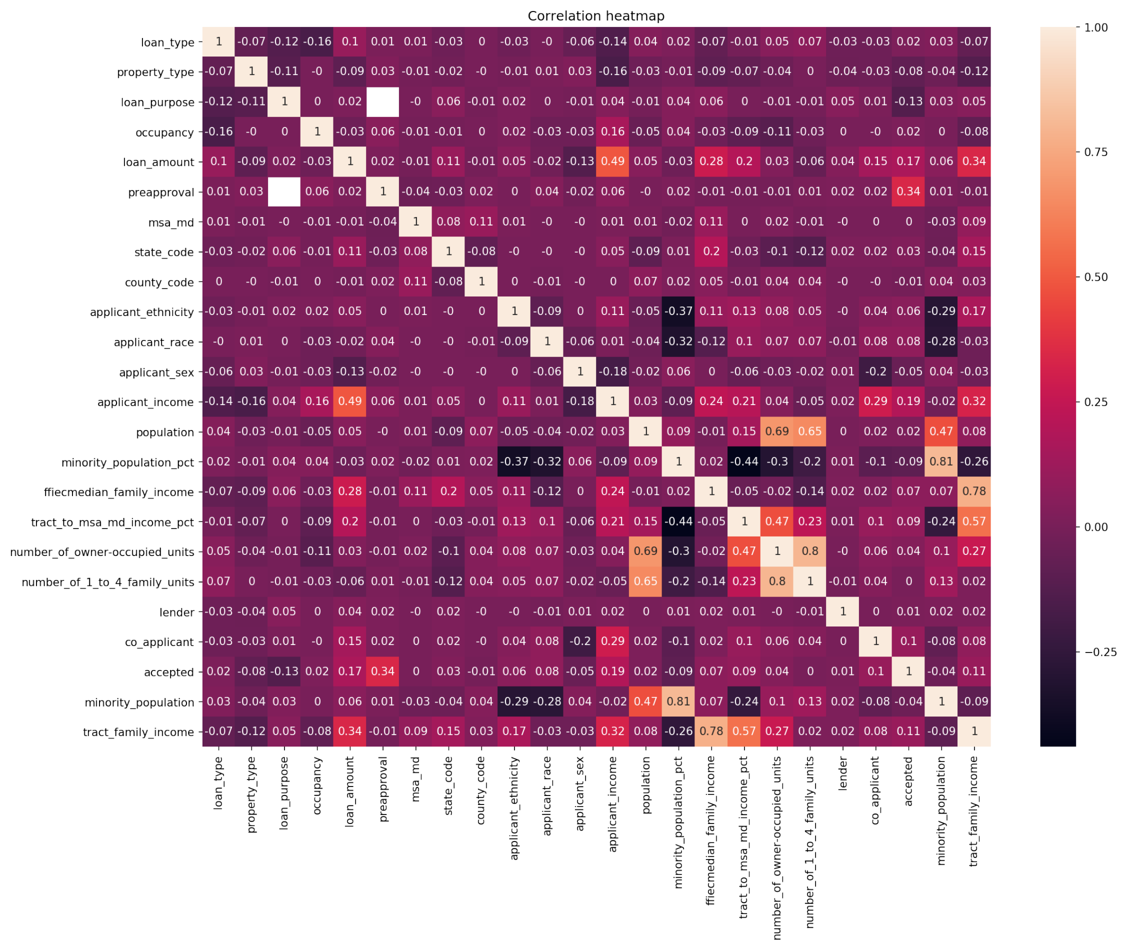


For the high-cardinality categorical features, I decided to use KDE plots. We can see that *lender* and *state_code* provide adequate information about the separation of the target classes.

New features



I decided to create two new features, i.e. *minority_population* and *tract_family_income*. The first feature is the product of *minority_population_pct* and *population*, and contains the actual number of people that belong in a minority group. The second is the product of *tract_to_msa_md_income_pct* and *ffiecmedian_family_income*, and contains the tract median family income in dollars. It is evident from the KDE plots of those new features, that they both provide useful information about class separation, so they will be used in the binary classifier.



The correlation heatmap helps us visually identify any features that are highly correlated. Including those in the binary classifier is redundant, because the information they provide is highly overlapping. We can see that the new features we created earlier, are highly correlated with those that comprise them. This is fairly self-evident and reasonable, so I decided to drop the original features. There is a number of other features that are highly correlated as well, but those had already been dismissed because they wouldn't contribute useful information for the training of the binary classifier.

3 Machine Learning Modelling

For the creation of the binary classifier, I experimented with various algorithms and techniques. First of all I tested the accuracy of some typical classification algorithms, such as Logistic Regression, K-Nearest Neighbors and Support Vector Machines. The accuracy I got with them was below 70%, and that was unacceptable. I also tried more sophisticated algorithms and machine learning libraries, like LightGBM and Keras/Tensorflow. Eventually, I got the highest accuracy (about 72.5%) with the XGBoost library. I used the XGBoost Classifier for the machine learning model, as well as various functions of the scikit-learn library for data preprocessing, metrics, and model selection. First of all, I created a *prepare_data* function that dealt with missing values by imputing them with the median/mode value of each feature, depending on whether it was a numerical or categorical one. It also applied the logarithmic function to the skewed numerical features, as mentioned earlier. Afterwards, I applied one-hot encoding to the categorical features, as it is typical in machine learning. I ran into a problem though, as the encoded high-cardinality features resulted in the dramatic increase of the dataset size. This was normal, as thousands of features were added to it, but led to memory problems and increased the time needed to train the machine learning models. After doing some research, I decided to apply target encoding to the high-cardinality features, instead of one-hot encoding, and that solved the problem. After that, I used the StandardScaler class of scikit-learn to scale my dataset, as well as the RandomizedSearchCV class to do a combination of cross-validation and parameter tuning. RandomizedSearchCV tests a sample of hyperparameter combinations for a given estimator, and helps us choose the best one.

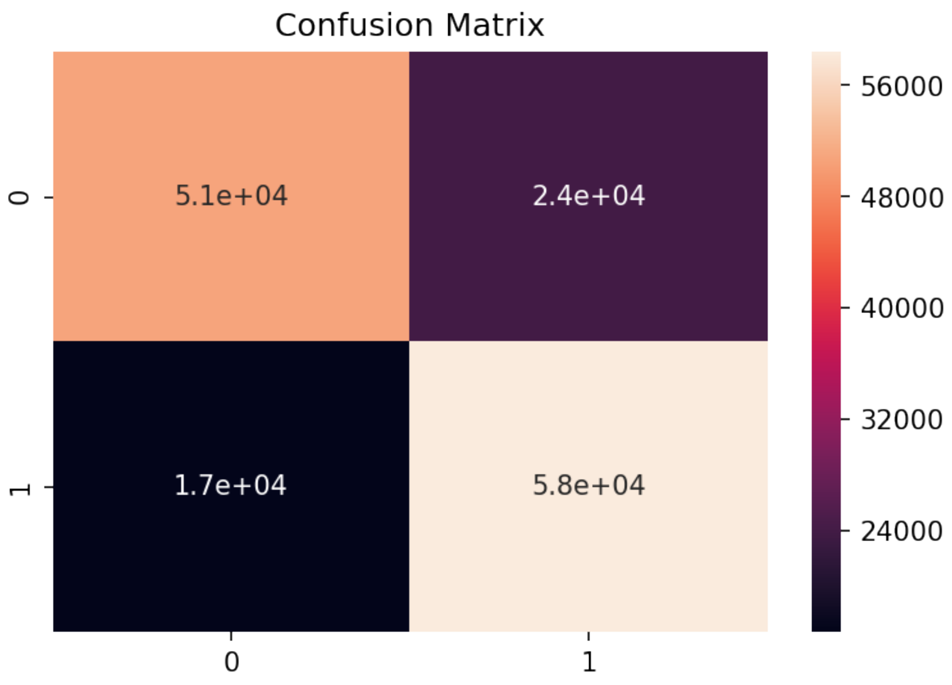
```
Best estimator:
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
   colsample_bytree=0.6, gamma=0, learning_rate=0.02, max_delta_step=0,
   max_depth=8, min_child_weight=8, missing=None, n_estimators=600,
   n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
   reg_alpha=0.2, reg_lambda=1, scale_pos_weight=1, seed=None,
   silent=True, subsample=0.7)


The accuracy is: 0.7282733333333333

Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.68      0.71     74876
           1       0.71      0.78      0.74     75124

   micro avg       0.73      0.73      0.73    150000
   macro avg       0.73      0.73      0.73    150000
weighted avg       0.73      0.73      0.73    150000
```



RandomizedSearchCV helped us find the best hyperparameter combination, which was then evaluated using various metrics. As we can see from the classification report, as well as the confusion matrix, the classifier is fairly accurate, but outputs a number of false positive and false negative values as well.

4 Ethical Concerns And Conclusion

The accuracy of our binary classifier is satisfactory. It could either be improved by adding more useful features in the dataset, or by exploring the performance of other algorithms and techniques. As it was mentioned before though, our data analysis highlighted the discrimination against specific groups of people in the approval of mortgages, i.e. based on race, nationality and sex. Including that information in a real-world data science project, would possibly reinforce and exacerbate that problem. I would personally be reluctant to do it, as it is ethically questionable. Of course in a business environment, making such decisions might not be my responsibility, so I would discuss that concern with a manager, or the head of my department. Regardless, our goal in this project was simply to maximize the accuracy of a classifier that would not be deployed to production, so all of the relevant features were included in the machine learning model.