Fair or Not: Evidence from HMDA Data

Anji Zhao (az529), Chenghao Li (cl2567), Yiwei Zhang (yz2454)

***Abstract*— In this paper, we constructed an empirical analysis to apply machine learning techniques to classify mortgage applications using the HMDA dataset. Moreover, we explored whether race plays an important role in the mortgage application process. We believe that the algorithms can not only be used to classify mortgage applications, but can also provide some insights for the regulatory departments to check the fairness of loan-granting procedure.**

I. INTRODUCTION

Housing mortgage loan granting procedure relies on applicants' background information, and it is important for the financial institutions to predict whether an applicant should be approved or not in order to minimize their credit risk exposure. As we all know, in 2008, subprime mortgage default was the direct reason that caused the global financial crisis. Hence, it is meaningful to develop a good model to classify these mortgage applications. However, as machine learning algorithms usually achieve their goals by seeking existing patterns in the data, we do not know whether the algorithms classify the mortgage application "fairly". As mentioned in the Equal Credit Opportunity Act, discrimination in credit applications based on race, color, religion, national origin, sex is unlawful. Different races, ethnicity, genders are supposed to have equal opportunities to get approved. Therefore, it is also important for us to ensure that our models are fairly classifying mortgage applications.

In this project, HMDA data from District of Columbia is used to explore the fairness of loan granting procedure. D.C. has a highly diversified population, which provides less biased dataset for this project. Dataset has all loan application records ranging from 2007 to 2012. Note that 2007 is during the global financial crisis, therefore, data in 2007 may provide some different insights.

II. EXPLORATORY DATA ANALYSIS

A. Data Characteristics

We selected and downloaded the HMDA dataset for District of Columbia from 2007 to 2012. The dataset has 36 feature columns and 254,976 records of loan application. For every loan application record, outcome is nominal with value 1 to 5, with each representing being approved, rejected, etc. The data also contains some nominal value variables including loan agency names, loan types, property types, and most importantly for this project, applicants' race and ethnicity. Other than these categorical features, there are two features with continuous values, which are loan amount and applicants' income.

The data is complete by itself without any missing values, there are, however, certain features having values of "No information provided" or "Not applicable", these values will be considered missing values in this study. In addition, some features are involved with co-applicants' information, while most of the applications are filed alone, which also makes the dataset messy. We picked 17 most relevant features to conduct this study.

Below is the features being used in this project:

- Categorical: Year, Agency, Loan Type, Property Type, Loan Purpose, Owner Occupancy, Loan Amount, Preapproval Status, Applicant Ethnicity, Co-applicant Ethnicity, Applicant Race, Co-applicant Race, Applicant Sex, Co-applicant Sex, Purchaser Type
- Continuous: Applicant income, Loan Amount
- Output: Action taken

B. Data Visualization

Correlation matrix serves as a great preview tool to see the relationships between features. We used heatmap to visualize the matrix:

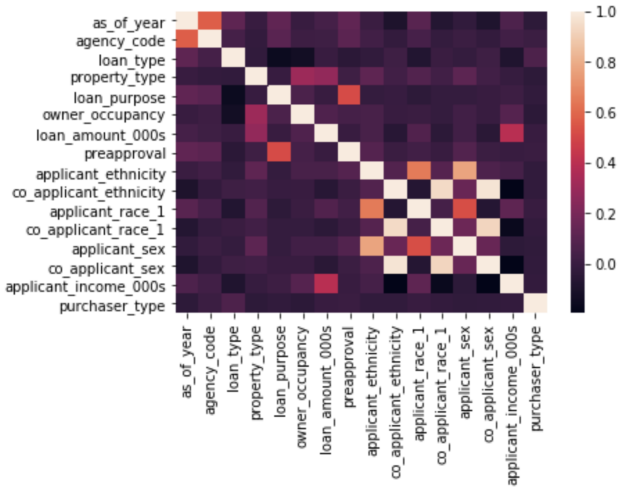

Fig. 1. Correlation matrix visualization between all features

We observe that except (co-)applicant race and ethnicity have relatively strong correlation, correlations are low among almost all other features. Hence, co-linearity is not significant between those features.

Distributions on the features or the output space (y) are important as well in order to have a basic understanding of the data. For continuous features like applicant income and loan amount, there are some extreme outliers, which can be seen from their distributions in Figure 2 and Figure 3.

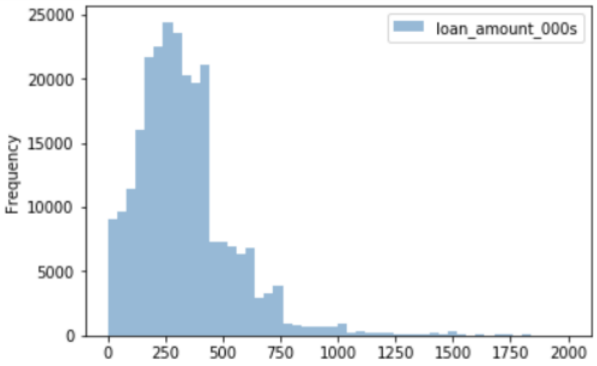

Fig. 2. Histogram of loan amount being applied

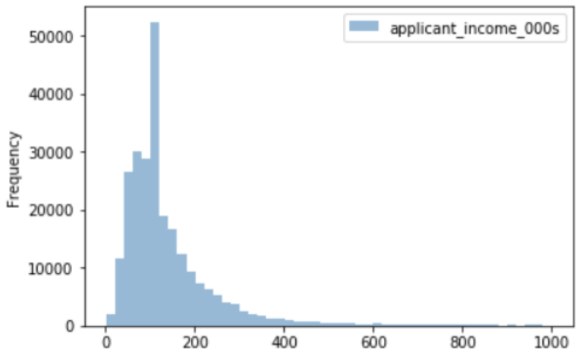

Fig. 3. Histogram of applicants' income level

We can observe that most loan amount being applied are around 250,000 dollars, and majority of applicants have annual income around 100,000 dollars. But some loan amount exceeds 1,750,000 dollars while some applicants claimed to have an income level over 800,000 dollars.

For output space, to have a first check of the fairness of the loan granting process, it is ideal to group records by sex or race, and plot the bar charts separately.

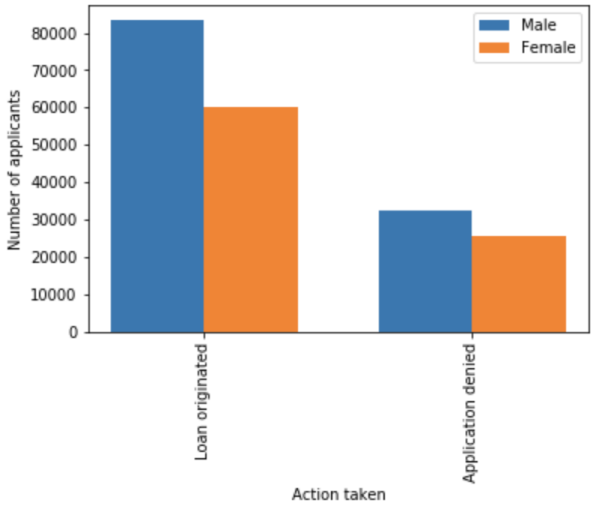

Fig. 4. Comparison between male and female

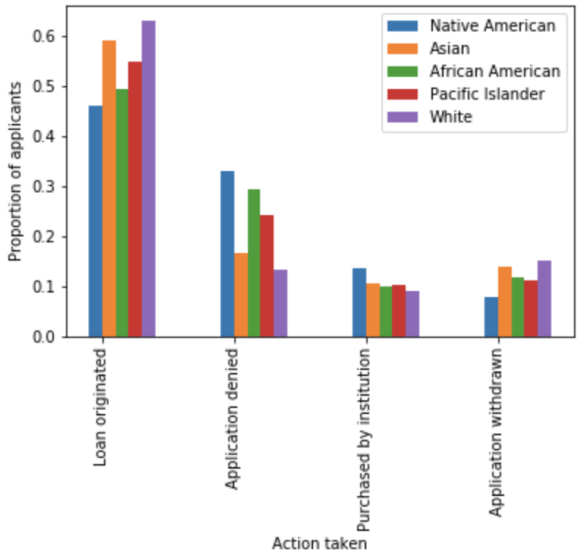

Fig. 5. Comparison among races

Based on Figure 4, we cannot see much difference in pattern for male applicants and female applicants, their distributions are similar. For distributions of difference races, however, we observe that African American, Pacific Islander and Native American have higher denial rates compared to White and Asian.

C. Feature Engineering

We applied the techniques learned in the class and did several feature engineering work. We will discuss each of them in detail below:

- **Missing Values:** We explored the features and found the variable applicant_income_000s contains 7517 missing values. Since this variable indicates the applicant's income level and from the visualization part we know it contains extreme outliers. As a result, we chose to fill the missing values with the median, which is 112.
- **One-hot Encoding:** We created dummy variables for the categorical features. Specifically, as discussed in the lecture, we chose one-hot encoding to transform all the categorical features to binary values by introducing additional dummy variables.
- **Handling Outliers:** We handled outliers in the data set. We know that the two real value variables, applicant income and loan amount contains extreme outliers and the causes of these outliers are most likely to be incorrect information. So we dropped the entries which have applicant's income or loan amount that is three standard deviations above the mean.
- **Merge Classes:** The objective of this project is to study loan approval vs denial. Thus, some actions taken by the financial institutions can be mapped into the two binary classes: Approved or Denied. Also, some classes have very few samples or their definitions are very similar to some other classes, so we merged the classes. Specifically, we mapped the observations

of "Preapproval request approved but not accepted", "Application approved but not accepted", "Loan purchased by institutions" into the "Loan originated" class. Also, we added the observations of "File closed for incompleteness" and "Preapproval request denied by financial institution" into the "Application denied by financial institution" class. In this way, we made the output y a binary variable and made the problem a binary classification problem.

- **Rebalancing Classes:** We used Synthetic Minority Oversampling Technique (SMOTE)[2] methods to oversample the minority classes in order to balance the data set. SMOTE method is based on nearest-neighbor judged by Euclidean distance, and it synthesizes new samples to balance minority class with the dominating class. Balancing the classes is important to prevent algorithms from overfitting on the majority classes.
- **Normalizing the Data:** We normalized the data using MinMaxScaler. We chose MinMaxScaler because it preserves the original data distribution. Moreover, it does not transform one-hot encoded features which might create disastrous result. Mathematically, MinMaxScaler can be formulated as:

$$x_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{1}$$

After doing all these feature engineering work, we split the data into training set and testing set (0.7:0.3).

D. Overfit/Underfit Problem

Three methods were used to prevent our models from overfitting/underfitting the data.

- Minority classes were resampled to balance the dataset such that accuracy score would be sensical, also to make the model learn enough information from the minority classes.
- Automated 5-fold cross-validation was used to tune parameters, and to prevent overfitting/underfitting by choosing the best sets of parameters.
- We tested each model by adding different regularizers and applied grid search approaches to search for the best parameters. Regularization in general can prevent algorithms from overfitting effectively.

III. Model Implementation

A. Logistic Regression

Logistic Regression is one of the fundamental classification methods that have been discussed in class. It is often used for binary classifications since Sigmoid function gives probabilities of the two classes between 0 and 1. L1 and L2 regularization methods were tried to penalize coefficients and further prevent overfitting. We tested the regularization parameter λ in the set of $\{0.01,\ 0.1,\ 1,\ 10\}$. Also, we set the maximum iterations number to be 1000 in order to make the algorithm fully converged.

The objective functions for L1 and L2 regularization methods are:

$$\mathcal{L}(\omega) = \Sigma_{i=1}^{n} log(1 + y_i(X\omega)) + \lambda \Sigma_{j=1}^{p} |\omega_j| \tag{2}$$

$$\mathcal{L}(\omega) = \Sigma_{i=1}^{n} log(1 + y_i(X\omega)) + \lambda \Sigma_{j=1}^{p} \omega_j^2 \tag{3}$$

Logistic Regression is designed to be a very efficient algorithm so it computes very fast. In this model, we used grid search to tune the parameter λ. We used 5-fold cross validation to make our results more reliable.

The best parameter we found using logistic regression is $\lambda = 10$ for both models using l_1 and l_2 regularizer. Also, we found that within 1000 number of iterations, all the algorithms converges with different λ. Moreover, models with different λ achieved very similar accuracy scores, which indicates that further tuning the regularization strength parameter will not make much improvement. The accuracy score achieved is 85.30% and the confusion matrix is shown below.

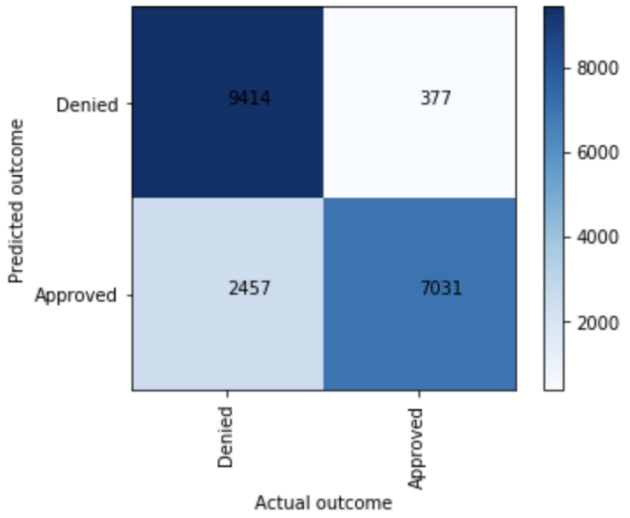

Fig. 6. Confusion Matrix of Logistic Regression

Based on the confusion matrix, False Positive Rate and False Negative Rate were 20.70% and 5.09%, respectively.

B. K-Nearest Neighbors Classifier

K-Nearest Neighbors algorithm is a non-parametric classification algorithm that does not need prior training to make prediction due to its own nature. It predicts the "unknown" data point to be the same category as its K nearest neighbors, where distance is determined by a specified distance function. The only parameter to choose for KNN algorithm is the number of neighbors, i.e., "K", which in our case is chosen from 1, 9, 31. Mathematically, KNN algorithm can be expressed as:

$$\mathcal{P}(\hat{y} = j|X) = \frac{1}{K} \Sigma_{i \in \mathcal{N}_i} I(y_i = j) \tag{4}$$

where $\mathcal{N}_i$ is the K nearest neighbors set, and $\hat{y}$ is predicted to be of class j which maximizes the $\mathcal{P}$ above.

KNN algorithm requires computing Euclidean distances between data points, which is typically inefficient. Dimension reduction is often used before applying KNN algorithm because Eculidean distance in high dimension does not contain much information. In our case, however, dimension reduction was not applied to maintain interpretability.

After the best K, whose value is 9, was chosen and applied to the test dataset. The accuracy score achieved by KNN algorithm was 81.3%.

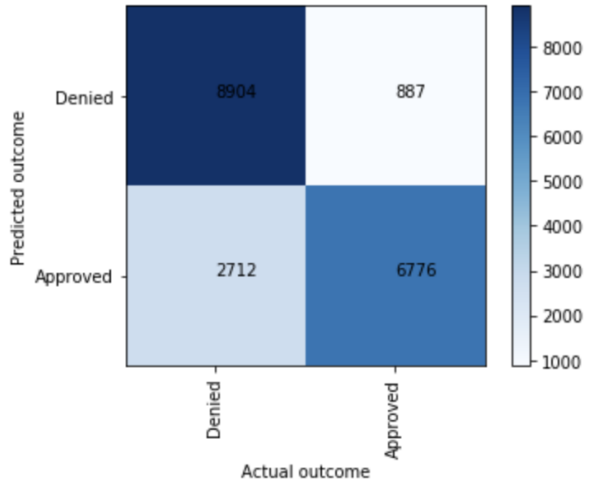

Fig. 7. Confusion Matrix of KNN Classification

KNN algorithm has its disadvantages besides the one mentioned above, and that is it does not handle categorical data well as Euclidean distance doesn't capture distance between different features. That might explain why accuracy achieved was lower than logistic regression. KNN also has higher False Positive Rate (23.35%) and higher False Negative Rate (11.58%). Therefore, KNN is inferior to Logistic regression based on the results.

For more information about KNN algorithm, one can refer to Chapter 2 in the book, *An Introduction to Statistical Learning*[1].

C. Random Forest Classifier

As briefly discussed in the lecture, Random Forest algorithm fits multiple decision trees and to cast the majority vote as the prediction. Bootstrap is embedded in the algorithm to resample data with replacement, which reduces variance of the output but still gives unbiased estimator. The model randomly chooses a subset of features to fit decision tree models to make sure no feature dominates others. By rule of thumb, number of features to randomly choose from the original set of features should be square root of number of features in the original set[3].

Three hyperparameters needed to be chosen. Criterion, which is similar to loss function by definition, can be chosen to be Gini-indnex or Cross-entropy; number of decision trees to model was chosen from 10, 50, 100, 300 and 500; the max depth of every tree was chosen from 10, 20 and 50. Grid search was implemented with 5-fold cross-validation to choose hyperparameters. Gini-index and Cross-entropy are defined as below:

$$G = 1 - \Sigma_{i=1}^{K}(\hat{p}_i)^2 \tag{5}$$

$$E = -\Sigma_{i=1}^{K}\hat{p}_i log\hat{p}_i \tag{6}$$

where $K = 2$ in our case since its binary classification, $\hat{p}_i$ represents the probability of that point being in class i.

The best set of hyperparameters indicated that Cross-entropy should be used as criterion, 300 decision trees should be modeled, and max depth of the trees should be 20. With this set of hyperparameters, the model achieved 85.8% accuracy score based on test data.

For a random decision tree in the decision forest, we extracted one portion of it to visualize (Fig 8):

The resulting confusion matrix is given below:

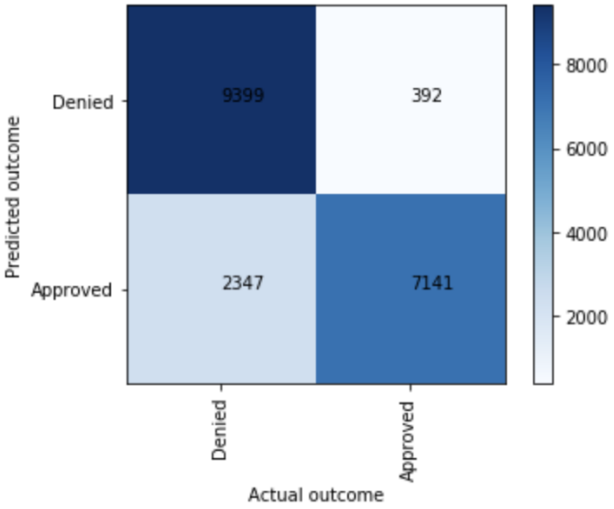

Fig. 9. Confusion Matrix of Random Forest Classification

One major advantage of random forest algorithm is that it automatically ranks importance of each feature, which is often used to select features and reduce dimensionality of the data. The rank of importance is given below:

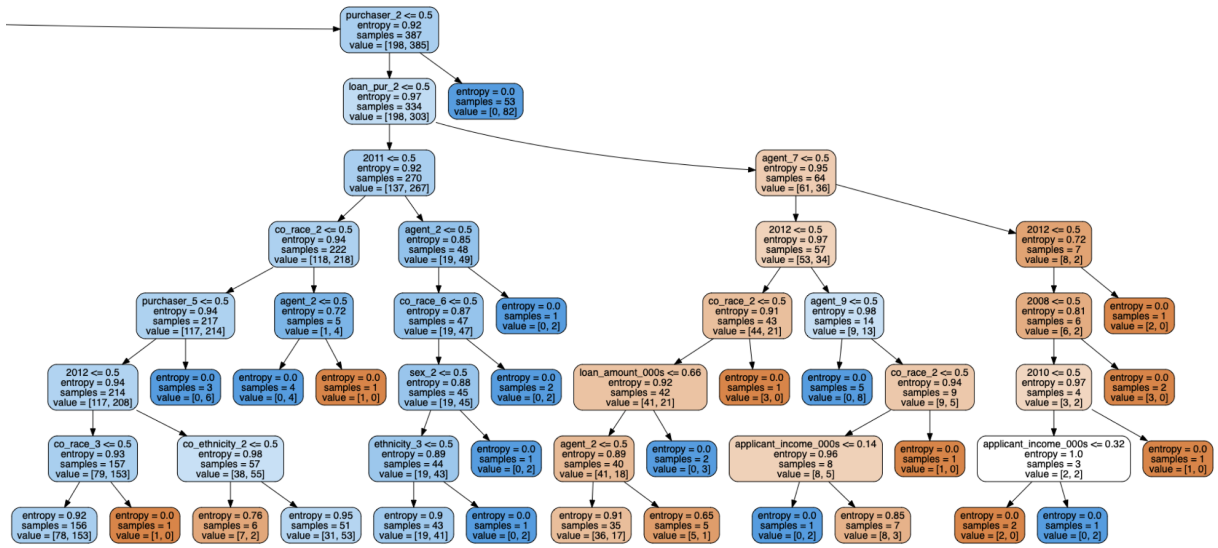

Fig. 8. Visualization of a decision tree in random forest

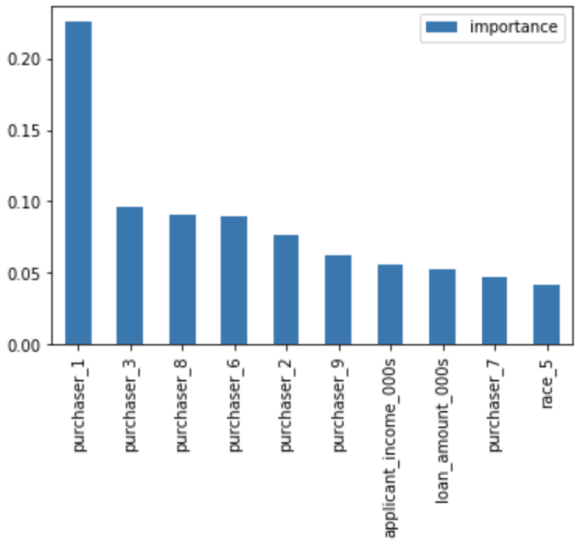

Fig. 10. Top 10 important features

One can see that Loan purchase is the most important feature identified by random forest algorithm. As one of the objectives of this project, fairness of the model requires that model should not take protected features into consideration, but applicants' race is the 10-th most important feature according to random forest model, more discussion about fairness of the model can be found in next section.

D. Support Vector Machine

Finally, we fitted Support Vector Machine (SVM) algorithm to the dataset. As discussed in class, SVM is effective in dealing with high dimensional data, and it is also more comfortable with categorical variables than KNN algorithm. Moreover, kernel function for SVM can be versatile or even customized. In our case, radial basis function was used as the kernel function as it is a popular choice for kernel function of SVM classifier. Radial basis function is defined as below:

$$K(x^{(i)}, x^{(j)}) = exp(-\gamma||x^{(i)} - x^{(j)}||^2), \gamma > 0 \qquad (7)$$

where γ is the parameter sets the spread of the kernel. This is not mentioned in class, for more information about kernel function choice, one can refer to book: *An Introduction to Statistical Learning with Applications in R*[4].

Then SVM learns a linear classifier:

$$f(x) = \Sigma_i^N \alpha_i K(x, x^{(i)}) \qquad (8)$$

by solving optimization problem with respect to α_i.

Since RBF finds SVM in an infinite dimension, it is impossible to visualize the result. L2 regularization was used because it produces more accurate results than other regularizers. Similar as before, regularization strength was determined by coefficient "C"; γ which is the constant coefficient in RBF, scales the influence and directly affects the result, was also chosen through cross-validation.

Cross-validation showed that lower strength of regularization and small scale influence were better for prediction. Feeding the fitted model with test data, accuracy score was 85.57%. False Positive Rate was 20.25%, and False Negative Rate was 5.28%.

The confusion matrix is given below:

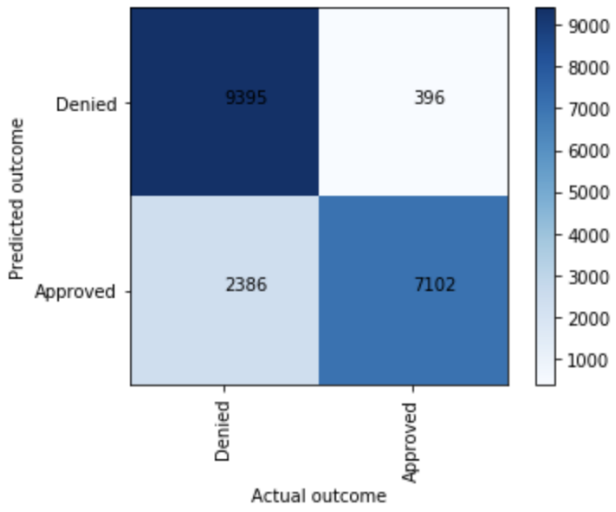

Fig. 11. Confusion Matrix of SVM Classification

IV. Result & Fairness Analysis

A. Result Analysis & WMD

TABLE I
Classification Results

Classifier	Scores
Logistic Regression	85.3%
KNN	81.3%
Random Forest	85.8%
SVM	85.6%

Overall, best accuracy was achieved by Random Forest algorithm with 85.8% accuracy score. This is reasonable since Random Forest is a very complex nonlinear model. SVM and Logistic Regression also achieved very similar accuracy scores, which indicates that this may be the boundary of this classification problem. If we would like to further improve the performance, more data is necessary in order to get more information.

However, we can see from the results that all of our models have a relatively high False Positive Rates. In the real life, False Positive means we issue mortgage to someone who has a high default probability. This can be very costly for the financial institutions. One possible way to solve the problem could be customizing loss functions which penalizes false positive predictions more.

We are fairly confident that our project does not produce a Weapon of Math Destruction. We have split the training and testing data so the outcome of our model can be easily tested. Also, although our model still made some wrong predictions which might have negative consequences on the real life, there is still much room for improvement since we have a lot of data (only a very small subset of the HMDA data are used in our project) available. Finally, our model does not create self-fulfilling (or defeating) feedback loops since the prediction results do not change people's behavior of applying mortgages.

B. Fairness Analysis

Another purpose of this project is to analyze the fairness of our fitted models. Fairness is important because our algorithms could have a big impact if we applied it in the real life. As we mentioned above, false positive predictions can do harm to the financial institutions. One the other hand, false negative predictions also limit the opportunities of getting a mortgage for those who are actually qualified to get. In this project, we focus on analyzing whether race plays an important role on the mortgage application procedure. That is to say, we set the "Race" to be our protected attribute on which discrimination is prohibited. In order to analyze the fairness, we split our training and testing set according to the applicants' race. Then we ran our fitted models separately on different race groups and calculated the fairness metric. Below are the results and a brief analysis.

- **Fairness Metric: Demographic Parity**

TABLE II
Fairness Metric: Demographic Parity

Classifier	White Approval Rate	Black Approval Rate
Logistic Regression	51.0%	24.6%
KNN	53.5%	22.2%
Random Forest	51.9%	24.8%
SVM	50.1%	24.6%

We first checked whether our algorithms satisfied the demographic parity. By definition, an algorithms satisfies the demographic parity if the prediction is independent of the protected attribute:

$$\mathrm{P}\hat{y}|a=1 \quad = \quad \mathrm{P}\hat{y}|a=0 \quad = \quad \mathrm{P}\hat{y} \tag{9}$$

In our case, we would like to check whether our predicted result of mortgage application is independent of an applicant's race. And from TABLE 2 we can see that for all four trained models, the white applicants have an approval rate around 50% while the black applicants have an approval rate around 25%, which indicates that white applicants are almost twice likely to get their application approved than black applicants. This result have shown that our prediction is not independent of applicants' race. In other words, our algorithms could be "unfair". However, as we have discussed in the lecture, demographic parity is not a perfect indicator of fairness because it does not take the base rate in different groups into consideration. It could be the case that the black people living in the Columbia Area who applied mortgage have a high default probability in general. Since, we could not observe the base rate directly, it is hard to tell whether the difference between the two groups reflects a sign of discrimination.

- **Fairness Metric: Equalized Odds**
  As an alternative way of ensuring fairness, we then checked whether our models satisfied equalized odds. An algorithm is said to satisfy equalized odds if the predication $\hat{y}$ is independent of the protected attribute a conditional on the outcome y:

TABLE III
FAIRNESS METRIC: EQUALIZED ODDS

Classifier	White TPR	Black TPR	White FPR	Black FPR
Logistic Reg	91.9%	99.1%	27.1%	14.1%
KNN	86.7%	92.4%	30.3%	18.8%
Random Forest	91.9%	99.4%	26.5%	14.1%
SVM	93.0%	99.7%	27.8%	14.1%

$$\hat{y} \perp a|y \quad \Longleftrightarrow P(\hat{y}|y, a = 1) = P(\hat{y}|y, a = 0) \quad (10)$$

In other words, the true positive, true negative, false positive, and false negative rates should be the same for both groups. We calculated the True Positive Rate (TPR) and False Positive Rate (FPR) for both groups. The other two rates can be easily inferred from the table. Note that, if the TPR are the same for both groups, then a weak condition of equalized odds: equality of opportunity holds. The results are quite surprising. We found that the black group actually has a higher TPR than the white group. For the Logistic Regression, SVM and Random Forest model, the black group has a TPR even higher than 99%. This indicates that for those whose applications should be approved ($y = 1$), our models are in favor of the black group. After looking at the race distribution in the testing data, we thought this could be due to the fact that there are much more observations of the white group. Therefore, our models might have a large probability of making wrong predictions for the white group.

Another interesting results came from the FPR. We found that for all the four fitted models, the white group has a much larger FPR, almost twice, than the black group. This indicates that for those whose application should not be approved ($y = 0$), white people are more likely to get their mortgage granted. This means that our models to some extent are in favor of the white group. But it is still hard to say whether the black group is discriminated.

V. CONCLUSION AND FUTURE WORKS

In conclusion, we applied four machine learning algorithms, namely logistic regression, KNN, random forest and SVM to classify the mortgage application given each individual applicant's information. After tuning the parameters, our best model random forest achieves an accuracy score of 85.8%.

Also, we checked whether our models are "fair" in term of the applicants' race. We set the applicants' race as the protected attribute and calculated the fairness metrics. Results show that for demographic parity and the FPR, the black group might be discriminated against. But the TPR for the black group is higher than the white group. Further investigation of the base rate for each groups is needed to draw other more reliable conclusions.

We are fairly confident that our model can be easily generalized well to new data and applied in the real life with some further improvement. Here are some of the possible ways for improvement: Firstly, we noticed that for our best model random forest, adding further regularization terms or increasing the complexity of the model made little improvement, which indicates that this could the boundary of the problem given the existing data. We might want to include more data points in the training set in order to improve the performance because algorithms can always learn more information with larger data set.

Secondly, we found that our model performed relatively worse in term of the False Positive Rate. As we have discussed above, for the mortgage application, false positive is more costly for the financial institutions in the real life since it means that we issued a mortgage to someone with a high default risk. Hence, we might want to customize the loss functions in order to penalize false positive predictions more.

Lastly, the fairness of our models is still hard to tell. This is partly due to the fact that we could not observe the base rates for different groups directly. Therefore, further investigation of the data is needed. If more evidence of the unfairness of our models can be shown, we should certainly adjust them to address the fairness problem. But there could be a trade-off between accuracy and fairness as we discussed in the lecture.

REFERENCES

[1] G. James, D. Witten, T. Hastie and R. Tibshirani. "K-Nearest Neighbors," in *An Introduction to Statistical Learning with Applications in R*, 7th ed. New York, NY: Springer, 2015, pp. 39-42.

[2] N. Chawla, K. Bowyer, L. Hall and W. Kegelmeyer. *SMOTE: Synthetic Minority Over-sampling Technique*. Journal of Artificial Intelligence Research 16, 2002, pp. 321–357.

[3] G. James, D. Witten, T. Hastie and R. Tibshirani. "Random Forests," in *An Introduction to Statistical Learning with Applications in R*, 7th ed. New York, NY: Springer, 2015, pp. 319-321.

[4] G. James, D. Witten, T. Hastie and R. Tibshirani. "Support Vector Machines," in *An Introduction to Statistical Learning with Applications in R*, 7th ed. New York, NY: Springer, 2015, pp. 319-321.