ML - Final Report

Mathieu Chiavassa, Alexander Parunov, Wangyang Ye

June 2018

1 Abstract

In our project, we evaluate different machine learning approaches to predict mortgage application decision. This is classification task with 8 classes. Our data set is public data related to Washington State home loans. Each year, these data are updated by banks and other financial institutions.
The prediction methods used in this project are: Naive Bayes, Random Forest and Support Vector Machines. We compare these machine learning methods and results, and select the best one by accuracy of prediction.

2 Data description and Pre-processing

2.1 Data description

The Home Mortgage Disclosure Act (HMDA) was enacted by congress in 1975 and it required the financial institutions which were either originating or re-financing home mortgage loans to report on the loan data. This rule applies to certain financial institutions, including banks, savings associations, credit unions, and other mortgage lending institutions. The major objective of this act is to make sure that government regulators can identify possible discriminatory lending patterns and to make sure that financial institutions are meeting community's lending needs. The goal of this project is to find a model that predicts well the action taken for the loan whether it is originated, denied, etc..

2.2 Pre-processing

The data set contains 466,566 instances with 47 variables (one of them is the response variable) about home loans in Washington State in 2016. After taking an initial glance on it, we see that it contains many NAs, redundant variables, outliers, incorrect values, not normalized variables, etc.. Which means the good and/or bad pre-processing can heavily affect the overall result. That's why it is crucial part of a problem.

2.2.1 Feature Selection

Initially we started with removing redundant and irrelevant variables. Following variables were discarded as we assumed they bring no essential meaning based on various reasons explained below:

- We don't need state abbreviation, name and year, our data set is always about Washington 2016.
- We don't need ids and sequence number.
- We don't need agency name, we can use just abbreviation.
- The number of the census tract for the property is not telling.
- We are not interested in application date whether it is before or after 1/1/2004, or the date is not available, so this variable is meaningless.
- The values of Hoepa_status_name are mostly Not HOEPA (Home Ownership and Equity Protection Act) loan, so this variable is not meaningful.
- preapproval_name is a explanation of preapproval class. We cannot use it for the prediction.
- Denial reasons can't predict the outcome of application, since it is highly correlated to the denied class.
- County name is not really making any sense, since it has no connection to applicant, just location of house.
- Co-applicant's sex, ethnicity and race names are irrelevant, since we don't want to build dependency between applicants. Also, additional race name for applicants are also removed since they are mostly empty values and thus meaningless.
- Rate spread had 98% of values as NAs, so we removed this column as well.
- The Metropolitian Statistical Area/Metropolitian Division (MSA/MD) where the property is located in is also removed.

After we got rid of these information, we had removed 27 variables and our data set became composed of 20 columns.
Then we decided to delete and change the names of the different categories of the categorical variables to make it easier to interpret. So some categories names were changed and others were removed. In total, we've changed the categories of 11 variables. In addition to that we declared outliers as missing values (NAs) and we performed imputation using chained equations (mice function in R). This imputation was done on Google Virtual Machine and the result has been saved locally as R object *hdma_cleaned.Rdata*, since it took around 17 hours to perform such imputation. Based on the imputed data, we removed some rows with abnormal values.

2.2.2 Feature Extraction

However, we strongly thought that it's useful to perform feature extraction as well, in order to discover new variable that could be useful.
We created 2 extra variables **Income Range** and **Loan Amount Range**, where each one buckets respective numerical variables into categorical based on their value. **Loan Amount Range** was split into 13 buckets on range (0,650] with spacing equal to 50. And **Income Range** was split into 5 buckets on range (0,250] with same spacing.
After performing feature selection and extraction we normalized numerical variables in order to give them same importance, and we corrected the skewed variables by applying logarithm. In addition to that since a lot of our variables are categorical we had to perform one-hot encoding to encode all categorical variables into binary. This step was necessary only for Support Vector Classifier, since it takes only numerical data as input. Which resulted into final data matrix of 68 predictive features. In conclusion we performed almost all the steps that are related to pre-processing stage.

2.3 Data Visualization and Exploration

Before tackling classification, besides exploring the plots of the variables, we decided to perform clustering in order *"to see"* the overall pattern of data. Before we start with the clustering process, the first problem we encountered is the high number of dimensions. Hence, we decided to perform the clustering in the factorial space rather than work directly with raw data. Some benefits of this method are:

- it takes into account the structural component of data and discard the noise.
- it reduces the curse of dimensionality.
- it works with orthogonal factors (avoiding multi-collinearity).
- it embeds the points in a Euclidean space.

In order to work with factorial space, the well-known Principal Component Analysis (PCA) was applied. Afterwards, we applied Hierarchical clustering (HC) using the coordinates given by the PCA. Then we consolidate the clustering using K-means, setting the initial centers as the resulting centers come from HC. The final clustering is as figure 1.

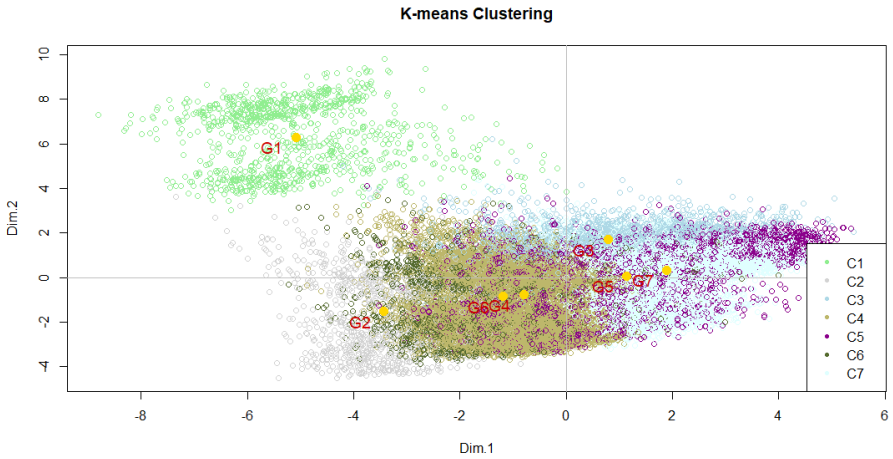

Figure 1: Result of clustering

As we can see, in total there are 7 clusters. We can observe that C1 is the most separated among all the clusters. The C2 and C3 are also relatively separated from others but with overlap. Unfortunately the rest of cluster are difficult to distinguish, specially the G4 and G6 which are almost completely overlapped. This causes big troubles for classifier, since data looks non-separable. Therefore, the error rate of those classes could be quite high. The brief description of each cluster is as follow, the results were obtained using R function *catdes*:

- C1: Applicant of this cluster tend to apply for small loan amount (87% of them apply for a loan up to 50k). 99% of the loan are conventional. More than half of the loans are intended for house improvement. The ethnicity of applicants is mostly Not Hispanic/Latino. The mortgages are mainly secured by subordinate lien. 59% of applications are originated.

- C2: Manufactured properties count 99.7% of the total applications within this cluster. 93% applicants of this cluster are White. Most of the mortgages is secured by first lien. The ethnicity of applicants is mostly Not Hispanic/Latino. 95% of the property are occupied by owner. Loan purposes are mainly for home purchasing and refinancing. Near half of applicants are originated.

- C3: The properties of this cluster are all not occupied by the owner. The major part of applications are conventional. Most of the mortgages is secured by first lien. 99% of the properties are for 1-4 family. The ethnicity of applicants is mostly Not Hispanic/Latino. Loan amounts are mainly in the range 50-200k and the incomes are in the range 100-250k.

- C4: Applicants mostly have an income up to 100k. Most of the applicants are White. The properties are all occupied by owner, they are of the type

for 1-4 family and all of the mortgages are secured by first lien. The loan amounts mainly vary from 100-300k. There's no big difference among different actions taken for the loan.

- C5: 60% of applicant are Asian. All the properties are occupied by owner and of type for 1-4 family. And all mortgages are secured by first lien. More than half of the applications were originated. The loam amounts mainly vary from 100-300k.

- C6: The applicants are all Hispanic/Latino and they are mostly White. The properties is for 1-4 family and 96% of them are occupied by owner. Almost all mortgages are secured by first lien. Applicants mostly have an income up to 100k. More than half applications are originated and 20% of the applications are denied.

- C7: Applicants mostly have an income up to 200k and most of them are White. Loan amounts mainly vary from 250-450k and great part of the loans are conventional. Properties are all occupied by owner and support 1-4 family. 63% of the applications were originated.

From the above description of categories, i.e. clusters, it's hard to say if there is any obvious pattern which we can make use. In addition to that many clusters are highly overlapping each other. Which makes it hard to predict all 8 classes with high accuracy.

3 Methods and Results

In order to hold 8 classes classification problem we decided to use 3 classifiers: **Naive Bayes**, **Random Forest**, and **Support Vector Machines**. However since our data set is very large (466k instances) we decided to get a random sample of 30k and perform training, model selection and testing using this subset of data, mainly for speed and efficiency purposes. And for the model selection, cross-validation will be used for the evaluation of models in a standard way. The proportion of each class in the subset is mostly equal to the proportion of each class in the original data set. It means that if, for example, class *approved* was 10% of original data set, the subset data set of size 30k will contain approximately 3000 instances of class *approved*.

3.1 Naive Bayes

We decided to start with the simplest Naive Bayes classifier. As a matter of fact, Naive Bayes classifier does a good job at classification with test error of **37%**, considering the fact that it takes no time to train classifier. However no parameter can be tuned, like with Random Forest and SVC since Naive Bayes method mainly calculates probabilities. The main reason why we decided to chose this classifier is to understand the base probabilities of selecting various classes. And it takes almost no time to *"train"* Naive Bayes classifier. In

addition to that Naive Bayes performs better for categorical variables rather than numerical, and the data set contains a lot of categorical variables. And importantly, the numerical variables do satisfy normal distribution assumption. It was deducted from observing bell curve of their density plots.

3.2 Random Forest

In compare to SV classifier the main parameter to tune in random forest algorithm execution is the number of trees. So from the Table 1 we see that the from the 398 trees, the values of OOB do not decrease a lot, but the training time is largely increased. Thus, based on OOB and the training time, we chose the optimal number of trees as **398 trees**. The main reason of why we decided to use Random Forest is it intrinsically suited for multiclass problems and it proved to working well with a mixture of categorical and numerical variables. Upon performing the test with tuned parameters we found out that the test error is **34%**.

	Number of Trees	OOB
1	10	0.392
2	16	0.373
3	25	0.362
4	40	0.350
5	63	0.345
6	100	0.339
7	158	0.338
8	251	0.335
9	398	0.333
10	631	0.333
11	1000	0.331
12	1585	0.332
13	2512	0.333
14	3981	0.332

Table 1: OOB for different number of trees

3.3 Support Vector Machines

The main reason why we decided to use support vector classifier is because of the non-linear classification problem we faced. And this solved by so-called kernel trick, implicitly mapping inputs into high-dimensional feature space. Also after performing onehot encoding we get a sparse matrix with many feature values equal to 0. And since it solves quadratic optimization problem, the local minima is guaranteed to be be global minima.
However, the most important part in building SVC is the tuning parameters part. We've selected different set of parameters to test and then chose the best one based on test error.

From the nature of our data set we selected radial basis kernel (RBF) for our SVC, since our data is not easily linearly separable as showed in cluster plot in Data Visualization section. The two parameters that needed to be tuned are gamma and cost. We decided to start with gammas and took set of following gammas: $\gamma = [0.125, 0.25, 0.5, 1, 2, 4, 8, 16]$ while keeping cost $C = 10$. The result of test error for each γ is depicted in table 2:

γ	error (%)
0.125	33.5
0.25	35.6
0.5	36.4
1	39
2	41
4	41.4
8	41.9
16	42

Table 2: Test errors for various gammas

From the table 2 we see that lower gamma γ gives lower test error. So we ran one more model with $\gamma = 0.05$ and got 32% of test error. So we selected this gamma as our optimal parameter.
After selecting optimal gamma γ based on the lowest test error, we continue training several models keeping this gamma $\gamma = 0.05$ as constant and varying costs: [1, 3.17, 10, 31.63, 100, 316.22, 1000]. The result of test error for each cost C is depicted in table 3:

C	error (%)
1	33.25
3.17	32.97
10	32.8
31.63	33.35
100	33.7
316.22	34.7
1000	36.8

Table 3: Test errors for various costs

From the table 3 we see that optimal cost $C = 10$. Thus our tuned parameters for SV Classifier are $\gamma = 0.05$ and $C = 10$. We will use these parameters for cross-validating SV Classifier in the next section.

4 Model Selection

After tuning and adjusting parameters for various models, we can move forward with model selection. In order to do so, we decided to apply 10-fold cross-validation as our validation protocol for each model.

4.1 Naive Bayes

The results of 10-fold cross-validation for each fold is depicted in Table 4. Where training and validation errors were rounded up to 3 decimals. As we see from the table the average **Validation Error** is **37%**, with 95% CI of **(35.1%, 38.9%)**. Taking into account the complexity of data set and simplicity of Naive Bayes model, the validation error is not very big. Also considering the fact that it takes almost no time to train this model, since it does not really train the model but just assigns probabilities.

fold	TR error	VA error
1	0.369	0.372
2	0.367	0.367
3	0.369	0.379
4	0.369	0.365
5	0.366	0.370
6	0.366	0.365
7	0.367	0.379
8	0.366	0.382
9	0.369	0.348
10	0.368	0.373
Average	0.368	0.37

Table 4: 10 fold cross-validation for Naive Bayes Classifier

4.2 Random Forest

The results of 10-fold cross validation for each fold is depicted in Table 5. Where training and validation errors were rounded up to 3 decimals. As we see from the table the average *Validation Error* is **33%** with 95% CI of **(31.1%, 34.9%)**. The average *Validation Error* is again better than the one of **Naive Bayes** and almost same as in **Support Vector Classifier**. The one advantage of **Random Forests** is that, it takes less time to train than **SVC.**

	TR.error	VA.error
1	0.3352	0.3317
2	0.3355	0.3271
3	0.3343	0.3300
4	0.3336	0.3283
5	0.3375	0.3292
6	0.3344	0.3342
7	0.3367	0.3396
8	0.3316	0.3396
9	0.3330	0.3413
10	0.3358	0.3446
Average	0.3348	0.3345

Table 5: 10 fold cross-validation for Random Forest

4.3 Support Vector Machines

The results of 10-fold cross validation for each fold is depicted in Table 5 below. Where training and validation errors were rounded up to 3 decimals. As we see from the table the average **Validation Error** is **32.8%** with 95% CI of **(0.309%, 34.7%)**. We were able to improve our model compared to Naive Bayes. The support vector classifier had **radial** kernel with cost $C = 10$ and gamma $\gamma = 0.05$. Even though it took more time to train model compared to Naive Bayes, the drastic improvement in accuracy is worth it.

fold	TR error	VA error
1	0.297	0.326
2	0.298	0.328
3	0.297	0.326
4	0.299	0.340
5	0.300	0.321
6	0.296	0.330
7	0.296	0.335
8	0.299	0.315
9	0.297	0.331
10	0.299	0.336
Average	0.298	0.328

Table 6: 10 fold cross-validation for SV Classifier

4.4 Results

In conclusion, we can say that, based on the results and reasons presented above, we can select both **Random Forests** and **SV Classifier** as our final models. As result of training the SVC with the best parameters and the whole training set, we got **32.8%** of test error with 95% CI of **(31.61%, 33.99%)**.

These two models can be used to fit the training set of the complete one which contains 466,566 instances. Indeed, after model selection we trained a model with the complete dataset with again splitting train-test data as 80%-20% using **Random Forest**, and we got pretty much same *accuracy* of **68%**. Which is a solid proof of our above described analysis. This *test error* of **32%** is absolutely acceptable for such data, since it is not a matter of life and death whether your mortgage loan will be accepted or denied by the bank.

5 Conclusion and Discussion

5.1 Performance of the models

We started experimenting with the simplest Naive Bayes model and the test error rate was sufficient enough to accept. However, it was necessary to train other more sophisticated models in order to improve accuracy. Naive Bayes classifier was basically chosen pretty much to compare if we can improve other models with training.
The second model we trained was Random Forest, since it doesn't require too many parameters to adjust, just number of trees. And this algorithm is naturally suited for multi-class problems as well as performs good on data sets with mixture of categorical and numerical variables, which is exactly our case. We used 80% of the subset data as training set, and for several number of trees, we check the OOB error for each of them and chose the most optimal taking into account both accuracy and training time. The chosen number of trees was 398 since choosing number higher than that does not make much sense. Because the error rate doesn't decrease significantly but training time will increase, since model becomes more complex (i.e. more trees to build). We must note that it was worth to train Random Forest, since the overall *accuracy* was increased to **68%** of full data set.
The third and final model we decided to use is Support Vector Machines since our data set is not linearly separable, as seen in figure 1, and besides that, the number of dimensions is relatively high. Thus we chose radial (RBF) kernel, even though we tried linear and polynomial kernel, however it didn't give as good results. Overall SVC was a good choice, however we had to put more effort here, since there are 2 parameters to tune: γ and cost C. We first found optimal γ while keeping cost C as constant, and after using this optimal γ we found optimal cost C. And our final set of parameters were $\gamma = 0.05$ and $C = 10$. Then using those parameters we trained data and tested in the same manner as with Random Forest and got slightly better *accuracy* of **69%** on full data set. Overall both **SVC** and **Random Forest** can be used as final model. Even though **SVC** has slightly better performance in terms of test error, **Random Forest** can be preferred since it takes less time to train and it is is simpler to understand and interpret.

5.2 Possible Improvement of Accuracy

As a matter of fact it seems that overall accuracy of **68%-69%** is close to the best we can do. There are 2 classes which are completely overlapping, as shown at plot 1 and error rate is around 50%-60%. It seems like the model selects those 2 classes at random. Thus even tuning parameters or feature engineering won't solve this problem. Also, the accuracy of some classes with small amount of observation could also be improved.
The nature of the data is still very complicated and messy. In addition to that we have 8 class classification problem, which makes it even harder for any classifier. Luckily we predict just a mortgage application decision, i.e. it is not a matter of life and death. Thus such accuracy is completely acceptable and can be used for further work.

5.3 Future Work

We have tried most if not all classification models studied during the course and achieved quite promising results. Disregarding the fact stated above, that accuracy is close to top of what we could do, we still think that other more complicated models can be applied. For example SVC with self-designed kernel, so that those 2 overlapping classes can be mapped to another hyperspace where they become easily separable with some hyper-plane. It was not done in this project, because of limited knowledge of kernel engineering. It is way too advance at the moment.
Another possible option can be applying Deep Learning models and see if results are improved. It wasn't applied in this project, because R doesn't seem like a good environment for doing Deep Learning and there are not many libraries written for R. So the whole project, mostly pre-processing part should be rewritten in Python. In addition to that, we don't have a big computational power to train models on quite big data set.

6 References

Official Data Set
https://www.kaggle.com/miker400/washington-state-home-mortgage-hdma2016

Description of Data Set
https://cfpb.github.io/api/hmda/fields.html