STA5703 Project Final Report

Justin Barry, Yuan Shao

December 6, 2018

1 Data Cleaning

In the data set, features ‘LTV’, ‘CoMonthlyRent’ and ‘CoMonthlyReliability’ are factor type, we convert them to numeric type.

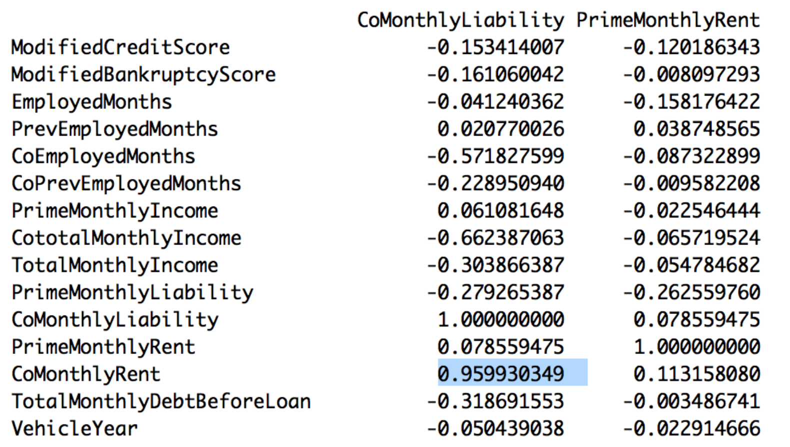
We have removed non-relevant features ”AppReceiveDate”, ”LoanNumber”.

In the “train.csv” there are 214 applications with missing values, and it’s just 0.2% of all observations. That is very small proportion, so instead of imputation we excluded all applications which have missing values.

Removing outliers: there are 5 data points’ “EmloyedMonths” is over 1000 months, and those 5 applications have been removed. Same way, outliers which “CoEmployedMonths” is more than 500 months have been excluded, “PrevEmployedMonths” over 400 months have been removed. Three outliers whose “PrimeMonthlyIncome” great than 500000 have been removed. For instance, one application with 666667 prime monthly income applied a loan for 35879, it’s very likely a typo or unusual case. With the same approach, we have cleaned all outliers in other features, such as “PrimeMonthlyIncome”, “VehicleMileage”, “PrimeMonthlyRent”.

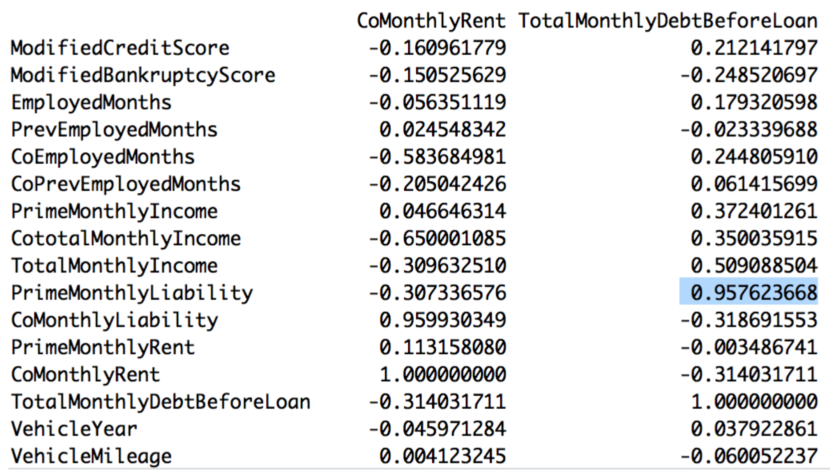
After cleaned data, there’are 109539 cases left from 109854, only small amount of data points have been removed, and we still have good amount of data points to train our models. We have fed data sets with and without outliers to a same model, and the model got around 1% higher accuracy when we input data without outliers. So it’s important to clean outliers before fitting data to models, as extreme values have negative influence on coefficients estimation.

Collinearity: as Figure 1 and Figure 2 showing some feature pairs have high collinearity, like “CoMonthlyLiability” and “CoMonthlyRent”, “TotalMonthlyDebtBeforeLoan” and “PrimeMonthlyLiability”. Since collinearity reduces the accuracy of the estimates of the model coefficients, we need to avoid it. One solution is to remove one feature from a pair, therefore attributions “CoMonthlyRent” and “PrimeMonthlyLiability” have been removed from data set.



	CoMonthlyLiability	PrimeMonthlyRent
ModifiedCreditScore	-0.153414007	-0.120186343
ModifiedBankruptcyScore	-0.161060042	-0.008097293
EmployedMonths	-0.041240362	-0.158176422
PrevEmployedMonths	0.020770026	0.038748565
CoEmployedMonths	-0.571827599	-0.087322899
CoPrevEmployedMonths	-0.228950940	-0.009582208
PrimeMonthlyIncome	0.061081648	-0.022546444
CototalMonthlyIncome	-0.662387063	-0.065719524
TotalMonthlyIncome	-0.303866387	-0.054784682
PrimeMonthlyLiability	-0.279265387	-0.262559760
CoMonthlyLiability	1.000000000	0.078559475
PrimeMonthlyRent	0.078559475	1.000000000
CoMonthlyRent	0.959930349	0.113158080
TotalMonthlyDebtBeforeLoan	-0.318691553	-0.003486741
VehicleYear	-0.050439038	-0.022914666

Figure 1: Collinearity pairs 1



	CoMonthlyRent	TotalMonthlyDebtBeforeLoan
ModifiedCreditScore	-0.160961779	0.212141797
ModifiedBankruptcyScore	-0.150525629	-0.248520697
EmployedMonths	-0.056351119	0.179320598
PrevEmployedMonths	0.024548342	-0.023339688
CoEmployedMonths	-0.583684981	0.244805910
CoPrevEmployedMonths	-0.205042426	0.061415699
PrimeMonthlyIncome	0.046646314	0.372401261
CototalMonthlyIncome	-0.650001085	0.350035915
TotalMonthlyIncome	-0.309632510	0.509088504
PrimeMonthlyLiability	-0.307336576	0.957623668
CoMonthlyLiability	0.959930349	-0.318691553
PrimeMonthlyRent	0.113158080	-0.003486741
CoMonthlyRent	1.000000000	-0.314031711
TotalMonthlyDebtBeforeLoan	-0.314031711	1.000000000
VehicleYear	-0.045971284	0.037922861
VehicleMileage	0.004123245	-0.060052237

Figure 2: Collinearity pairs 2

2 Model Selection and Fitting

We use logistic regression as it's a classifier with binary output and can take mixed data input, which fits well to our case. After cleaned the data set, there are 31 predictors left, and we assume those predictors have different effects on the response/LoanStatus: some are important some can be excluded in our model.

There are various ways to select features, such as best subset selection, forward/backward step-wise selection, ridge and lasso. We applied lasso and cross-validation when we fitted our model.

We have used package glmnet, and converted predictors into matrices and response into vectors with model.matrix() function, as glmnet() function takes matrix and vector as input. Then we split cleaned data set into two halves, one for training and the other for testing.

Second we performed cross-validation and computed the associated test error to tune the parameter lambda. Figure 20 is a plot of test error on different lambda values. After we found this best lambda value with smallest validation test error, we fitted our model with whole training data set, best lambda value. Figure 21 shows some coefficients of the model, some coefficients are zero, since lasso shrinked some features.

Once model was built, we did prediction with the model and test data set. Figure 5 is the confusion matrix of prediction and test data responses, we can see the accuracy is 81%.

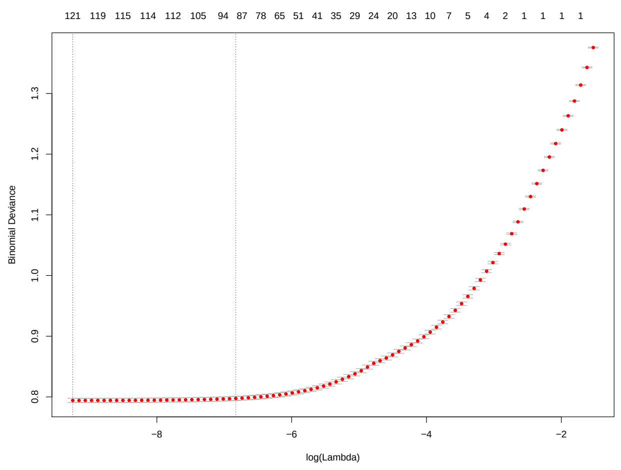
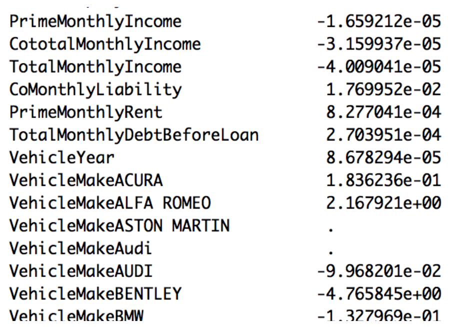
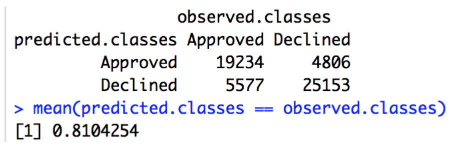


Figure 3: Test error vs lambda



```
PrimeMonthlyIncome              -1.659212e-05
CototalMonthlyIncome            -3.159937e-05
TotalMonthlyIncome              -4.009041e-05
CoMonthlyLiability               1.769952e-02
PrimeMonthlyRent                 8.277041e-04
TotalMonthlyDebtBeforeLoan       2.703951e-04
VehicleYear                      8.678294e-05
VehicleMakeACURA                 1.836236e-01
VehicleMakeALFA ROMEO            2.167921e+00
VehicleMakeASTON MARTIN          .
VehicleMakeAudi                  .
VehicleMakeAUDI                 -9.968201e-02
VehicleMakeBENTLEY              -4.765845e+00
VehicleMakeBMW                  -1.327969e-01
```

Figure 4: Model coefficients



```
                observed.classes
predicted.classes Approved Declined
         Approved    19234     4806
         Declined     5577    25153
> mean(predicted.classes == observed.classes)
[1] 0.8104254
```

Figure 5: Confusion Matrix

3 Next Steps

To improve accuracy, to try other methods like extreme gradient boosting. To evaluate model performance.

4 Categorical Features Description

Following features are treated as categorical features. Some are qualitative in nature: LoanStatus, Source, EmploymentStatus, VehicleMake, IsNewVehicle, RequestType, OccupancyStatus, MemberIndicator, CoApplicantIndicator. And we converted features CoMonthlyLiability and CoMonthlyRent to categorical features. After explored those variables, we decided including following key features in our model, below is a detail description of each feature.

LoanStatus: This is the response variable of our model, which has two value –'Approved' and 'Declined' to show if an application is accepted or not. There are 49466 cases are approved and 60388 cases were denied.

Source: From the figure 1 we can see, 'LENDER' has higher chance(over50%) to get accepted, and 'CONSUMER' and 'GATEWAY' has higher chance(over 50%) to get declined. It seems important at moment so we will include this feature in our model.

EmploymentStatus: From figure 2 we can see, different employ status has different approve rate. For instance, 'Employed' has higher declined rate than 'Retired'. Therefore this feature is selected.

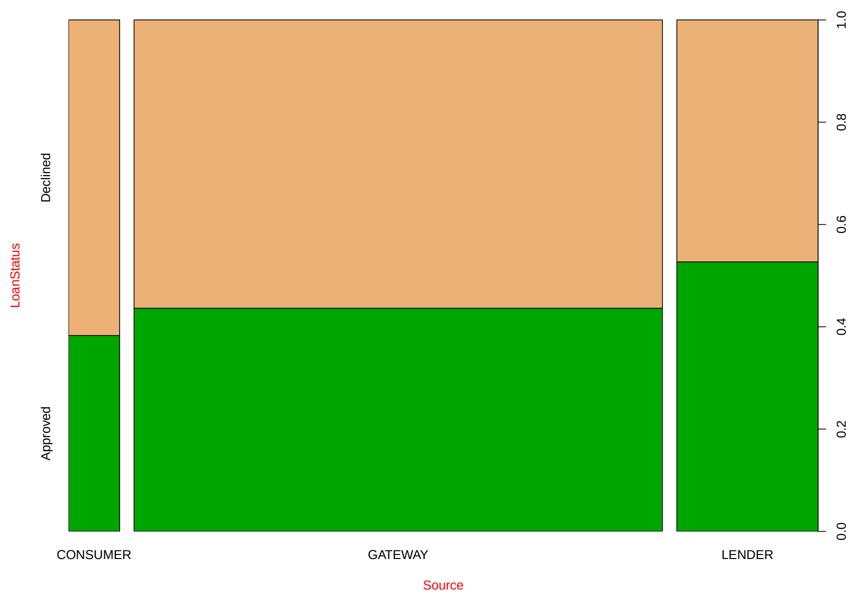
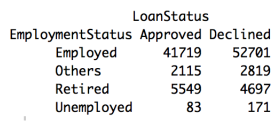
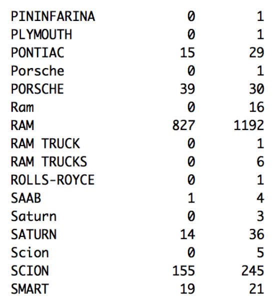


Figure 6: Source



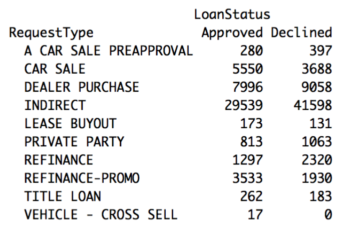
```
                       LoanStatus
EmploymentStatus Approved Declined
      Employed      41719    52701
      Others         2115     2819
      Retired        5549     4697
      Unemployed       83      171
```

Figure 7: EmploymentStatus Table



```
PININFARINA          0      1
PLYMOUTH             0      1
PONTIAC             15     29
Porsche              0      1
PORSCHE             39     30
Ram                  0     16
RAM                827   1192
RAM TRUCK            0      1
RAM TRUCKS           0      6
ROLLS-ROYCE          0      1
SAAB                 1      4
Saturn               0      3
SATURN              14     36
Scion                0      5
SCION              155    245
SMART               19     21
```

Figure 8: Vehicle Make: Accepted vs Declined



```
                          LoanStatus
RequestType              Approved Declined
  A CAR SALE PREAPPROVAL      280      397
  CAR SALE                   5550     3688
  DEALER PURCHASE            7996     9058
  INDIRECT                  29539    41598
  LEASE BUYOUT                173      131
  PRIVATE PARTY               813     1063
  REFINANCE                  1297     2320
  REFINANCE-PROMO            3533     1930
  TITLE LOAN                  262      183
  VEHICLE - CROSS SELL         17        0
```

Figure 9: RequestType: Accepted vs Declined

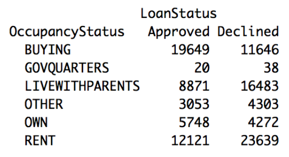
VehicleMake: Figure 3 are a sample of some makes, from it we can see the brands which are not so popular(less than 10 applications) and not luxury were tend to be declined. Some brand have two values because of upper/lower case difference, like 'Ram' and 'RAM'. We will consider to combine those values, make them all to upper case.

RequestType: As figure 4 displays, approve rates differ between values. For example, applications from 'INDIRECT' had just around 40% approve rate while 'CAR SALE' had around 60% approve rate.

OccupancyStatus: 'BUYING' higher the chance to be approved while 'LIVEWITHPARENTS' and 'RENT' had negative impact to approve. Those are expected, as generally home owner should have better financial situation than people who live with parents or rent a place. See details in figure 5.

MemberIndicator: An application with CFE membership had higher approve rate, more than 60%.

CoMonthlyLiability: When value equal to 'NULL', has little higher change to fail.



```
                    LoanStatus
OccupancyStatus    Approved Declined
  BUYING              19649    11646
  GOVQUARTERS            20       38
  LIVEWITHPARENTS      8871    16483
  OTHER                3053     4303
  OWN                  5748     4272
  RENT                12121    23639
```

Figure 10: OccupancyStatus: Accepted vs Declined

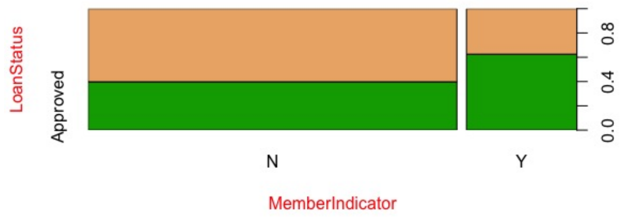


Figure 11: LoanStatus MemberIndicator

CoMonthlyRent: When equal to 'NULL', has little higher change to fail.

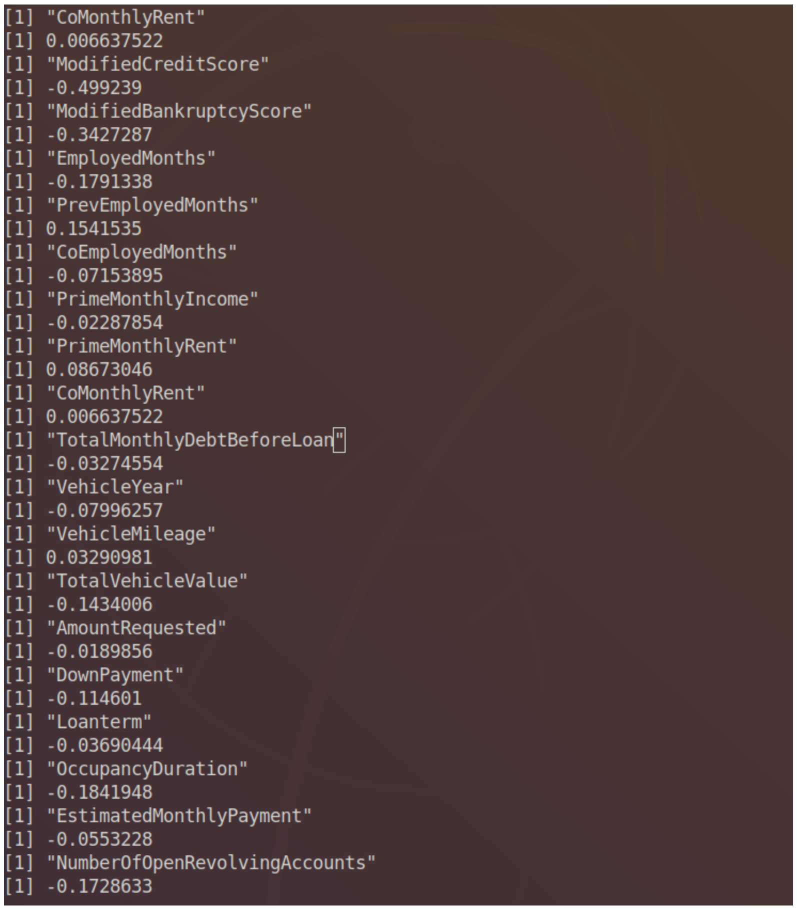
5 Numerical Feature Description

5.1 Polyserial correlated features

Polyserial correlation was used to infer a relationship between the the numeric features and the categorical feature LoanStatus. We calculated polyserial correlation for each numerical column against the response variable LoanStatus, the results can be seen in Figure 12

The highest absolute values of the polyserial correlation indicate the strongest relationships between the response and predictor. The values are bound between -1 and 1. From the results, we can see that ModifiedCreditScore has the strongest relationship of -0.499. Following closely is ModifiedBankruptcyScore with a score of -0.342. Other values worth mentioning are TotalVehicleValue, DownPayment, OccupancyDuration, NumberOfOpenRevolvingAccounts, PrevEmployedMonths, and EmployedMonths. There scores can be seen in Figure 12.

Here, we provide some visuals of the key features to get a better idea of how they are described. The histograms for ModifiedCreditScore vs LoanStatus(Declined) and ModifiedBankruptcyScore vs LoanStatus(Declined) can be seen in Figure 13 and Figure 14, respectively. The histograms for ModifiedCreditScore vs LoanStatus(Accepted) and ModifiedBankruptcyScore vs LoanStatus(Accepted) can be seen in Figure 15 and Figure 16, respectively. Comparing the histograms of ModifiedCreditScore vs LoanStatus(Approved) with ModifiedCreditScore vs LoanStatus(Declined), we can see that the average ModifiedCreditScore is higher for "Approved" then it is for "Declined". This makes sense intuitively and probably means there is some predictive power for this feature. The same idea applies to ModifiedBankruptcyScore. We can see that the average bankruptcy score is lower for "Approved" then it is for "Declined." This also makes sense. Additionally, the summaries of ModifiedCreditScore and ModifiedBankruptcyScore can be seen in Figure 17 and Figure 18.



```
[1] "CoMonthlyRent"
[1] 0.006637522
[1] "ModifiedCreditScore"
[1] -0.499239
[1] "ModifiedBankruptcyScore"
[1] -0.3427287
[1] "EmployedMonths"
[1] -0.1791338
[1] "PrevEmployedMonths"
[1] 0.1541535
[1] "CoEmployedMonths"
[1] -0.07153895
[1] "PrimeMonthlyIncome"
[1] -0.02287854
[1] "PrimeMonthlyRent"
[1] 0.08673046
[1] "CoMonthlyRent"
[1] 0.006637522
[1] "TotalMonthlyDebtBeforeLoan"
[1] -0.03274554
[1] "VehicleYear"
[1] -0.07996257
[1] "VehicleMileage"
[1] 0.03290981
[1] "TotalVehicleValue"
[1] -0.1434006
[1] "AmountRequested"
[1] -0.0189856
[1] "DownPayment"
[1] -0.114601
[1] "Loanterm"
[1] -0.03690444
[1] "OccupancyDuration"
[1] -0.1841948
[1] "EstimatedMonthlyPayment"
[1] -0.0553228
[1] "NumberOfOpenRevolvingAccounts"
[1] -0.1728633
```

Figure 12: Caption

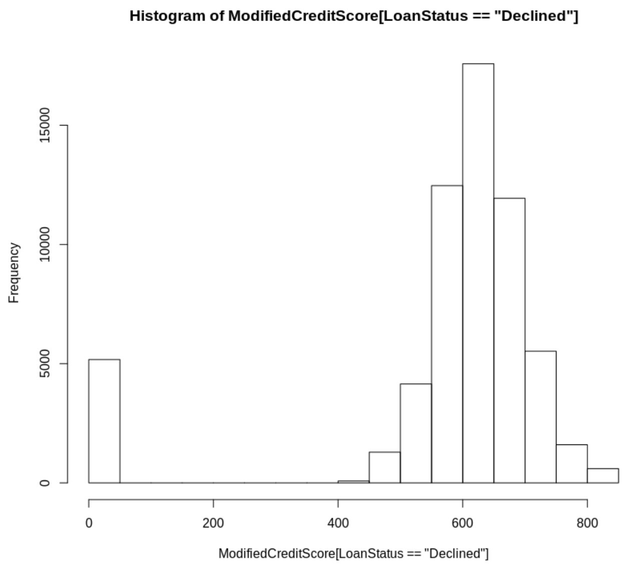


Figure 13: Caption

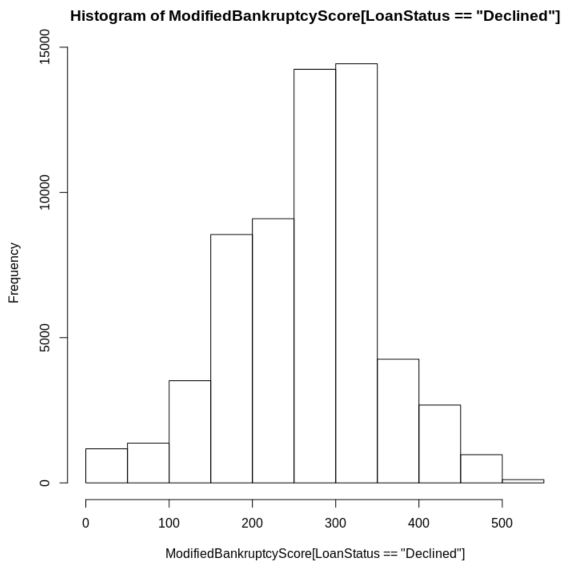


Figure 14: Caption

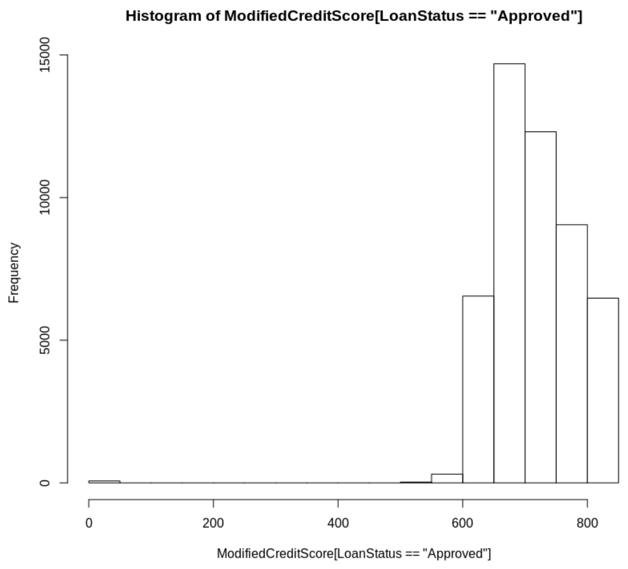


Figure 15: Caption

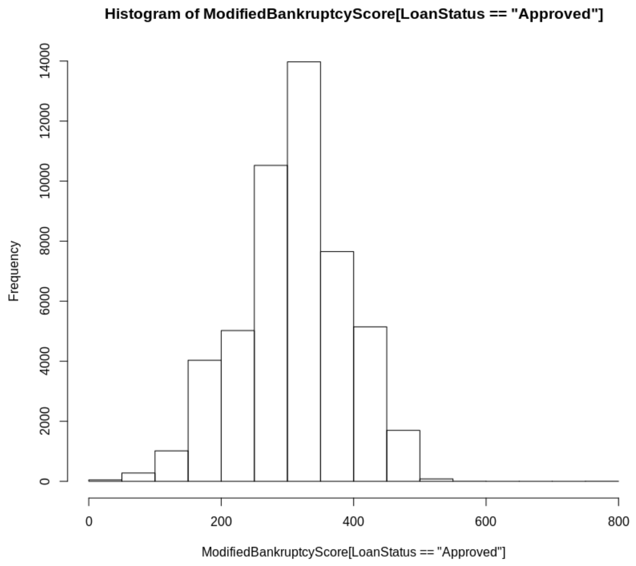
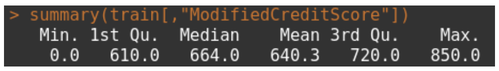
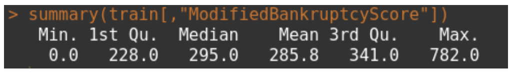


Figure 16: Caption



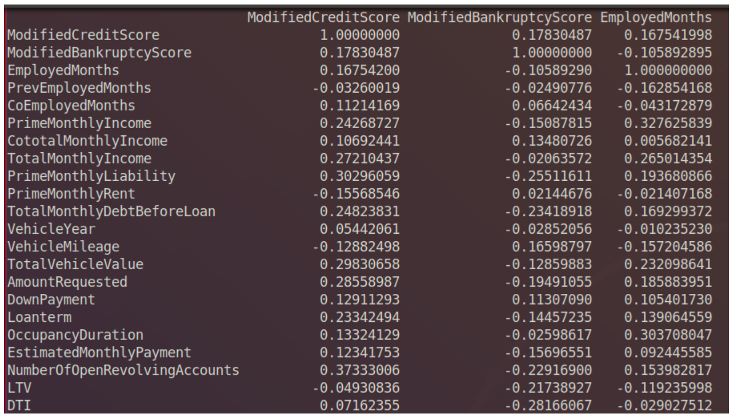
```
> summary(train[,"ModifiedCreditScore"])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0   610.0   664.0   640.3   720.0   850.0
```

Figure 17: Caption



```
> summary(train[,"ModifiedBankruptcyScore"])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    0.0   228.0   295.0   285.8   341.0   782.0 
```

Figure 18: Caption



```
                              ModifiedCreditScore ModifiedBankruptcyScore EmployedMonths
ModifiedCreditScore                    1.00000000              0.17830487    0.167541998
ModifiedBankruptcyScore                0.17830487              1.00000000   -0.105892895
EmployedMonths                         0.16754200             -0.10589290    1.000000000
PrevEmployedMonths                    -0.03260019             -0.02490776   -0.162854168
CoEmployedMonths                       0.11214169              0.06642434   -0.043172879
PrimeMonthlyIncome                     0.24268727             -0.15087815    0.327625839
CototalMonthlyIncome                   0.10692441              0.13480726    0.005682141
TotalMonthlyIncome                     0.27210437             -0.02063572    0.265014354
PrimeMonthlyLiability                  0.30296059             -0.25511611    0.193680866
PrimeMonthlyRent                      -0.15568546              0.02144676   -0.021407168
TotalMonthlyDebtBeforeLoan             0.24823831             -0.23418918    0.169299372
VehicleYear                            0.05442061             -0.02852056   -0.010235230
VehicleMileage                        -0.12882498              0.16598797   -0.157204586
TotalVehicleValue                      0.29830658             -0.12859883    0.232098641
AmountRequested                        0.28558987             -0.19491055    0.185883951
DownPayment                            0.12911293              0.11307090    0.105401730
Loanterm                               0.23342494             -0.14457235    0.139064559
OccupancyDuration                      0.13324129             -0.02598617    0.303708047
EstimatedMonthlyPayment                0.12341753             -0.15696551    0.092445585
NumberOfOpenRevolvingAccounts          0.37333006             -0.22916900    0.153982817
LTV                                   -0.04930836             -0.21738927   -0.119235998
DTI                                    0.07162355             -0.28166067   -0.029027512
```

Figure 19: Correlation values for select predictors

5.2 Correlated features

From these results, it may useful to find what features are correlated with the features with high polyserial values. We can then see how these features affect the model. To do this, we can set a correlation threshold, and write code that searches for features above this correlation threshold. We can remove those features from the model and see what our training/test error is. We can iterate the above steps and compare error rates. A sample of the correlation values can be seen in Figure 19

6 Methodology

Our next step is feature selection, and we would like to apply lasso and stepwise selection for this task. Feature selection is choosing only key features and omit trivial features, to increase model interpretability, especially for our case – 35 predictors. Another reason is to increase model accuracy, remove unnecessary variables and choose the suitable model complexity which can achieve low test error rate. Mechanics such as cross-validation, BIC and adjust R^2 will be used.

As this is classification case, and with many features, we will use logistic regression(maybe polynomial form when need), and address following potential problems and solve them: Non-linearity of the response-predictor relationship, Correlation of error terms, outliers and Collinearity.

7 Model Selection and Fitting

We use logistic regression as it's a classifier with binary output and can take mixed data input, which fits well to our case. After cleaned the data set, there are 31 predictors left, and we assume those predictors have different effects on the response/LoanStatus: some are important some can be excluded in our model.

There are various ways to select features, such as best subset selection, forward/backward step-wise selection, ridge and lasso. We applied lasso and cross-validation when we fitted our model.

We have used package glmnet, and converted predictors into matrices and response into vectors with model.matrix() function, as glmnet() function takes matrix and vector as input. Then we split cleaned data set into two halves, one for training and the other for testing.

Second we performed cross-validation and computed the associated test error to tune the parameter lambda. Figure 20 is a plot of test error on different lambda values. After we found this best lambda value with smallest validation test error, we fitted our model with whole training data set, best lambda value. Figure 21 shows some coefficients of the model, some coefficients are zero, since lasso shrinked some features.

Once model was built, we did prediction with the model and test data set. First we set prediction criteria to 0.5: Applications would be predicted as "Approved" when prediction probability is greater than 0.5. Figure 22 is the confusion matrix of prediction and test data responses with a criteria 0.5, we can see the accuracy is 80.7%. Type 1 error rate is 9.1% and type 2 error rate is 10.2%.

Sometimes institutions care more type 1 error than type 2 error and would like to have a low type 1 error rate, to reduce risks/loss of approving not qualified applications. Besides trying to improve overall accuracy, we can also set higher prediction criteria, which would reduce type 1 error and increase type 2 error. We tried different criteria and found 0.7 is a good choice, as there is no big difference for the whole accuracy, and type 1 error rate reduced from 10.1% to 4.1%. The compensation is that type 2 error rate went up. So which criteria to choose is based on institution's priorities. On the other hand if they prefer lower type 2 error rate, we can reduce the criteria. Figure 23 shows confusion matrix when prediction criteria is 0.7.

8 Future Work

To improve accuracy, to try other methods like randome forest, extreme gradient boosting.

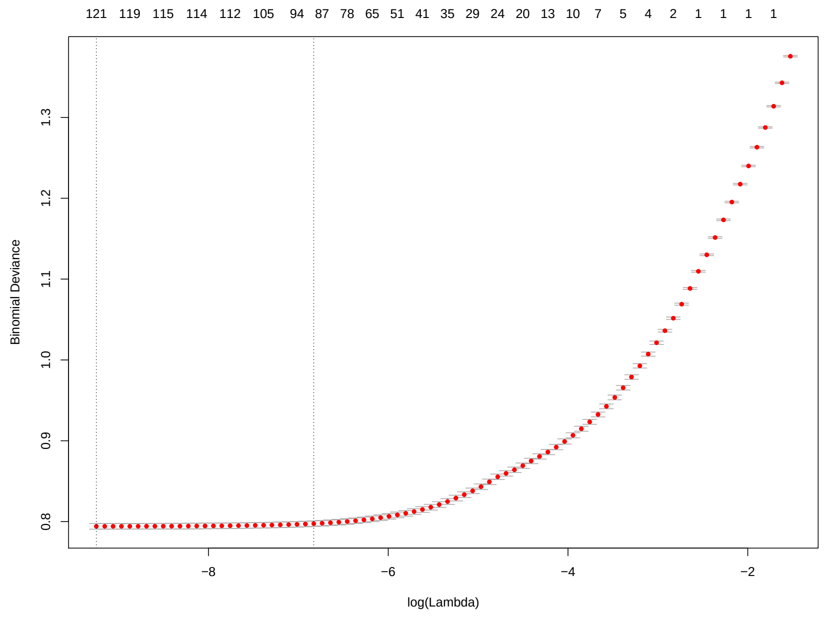
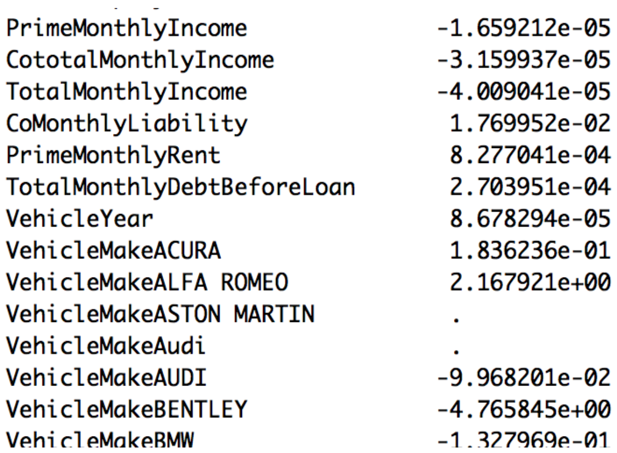
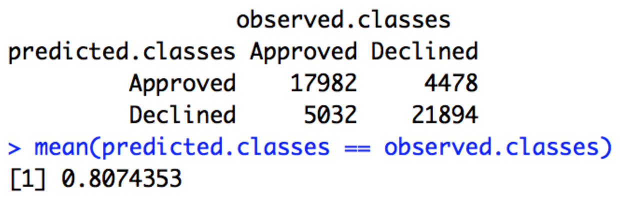


Figure 20: Test error vs lambda



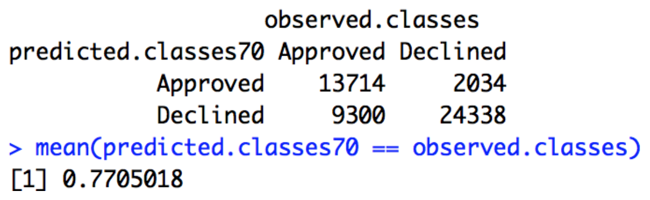
```
PrimeMonthlyIncome                -1.659212e-05
CototalMonthlyIncome              -3.159937e-05
TotalMonthlyIncome                -4.009041e-05
CoMonthlyLiability                 1.769952e-02
PrimeMonthlyRent                   8.277041e-04
TotalMonthlyDebtBeforeLoan         2.703951e-04
VehicleYear                        8.678294e-05
VehicleMakeACURA                   1.836236e-01
VehicleMakeALFA ROMEO              2.167921e+00
VehicleMakeASTON MARTIN            .
VehicleMakeAudi                    .
VehicleMakeAUDI                   -9.968201e-02
VehicleMakeBENTLEY                -4.765845e+00
VehicleMakeBMW                    -1.327969e-01
```

Figure 21: Model coefficients



```
                 observed.classes
predicted.classes Approved Declined
         Approved    17982     4478
         Declined     5032    21894
> mean(predicted.classes == observed.classes)
[1] 0.8074353
```

Figure 22: Confusion Matrix with prediction criteria 0.5



```
                     observed.classes
predicted.classes70 Approved Declined
           Approved    13714     2034
           Declined     9300    24338
> mean(predicted.classes70 == observed.classes)
[1] 0.7705018
```

Figure 23: Confusion Matrix with prediction criteria 0.7