Classification Notebook

3/23/2023

Alejo Vinluan (abv210001)

Searching for Similarity - Classification

Overview

This R Notebook will explore Classification utilizing a loan dataset. The dataset will attempt to predict whether or not an applicant was approved or declined for the loan. The following notebook will utilize Logistic Regression, kNN, and Decision Trees as different models for prediction. Comparisons will be made after each model is utilized and an analysis about why each model produced their results.

Dataset Breakdown

LendingClub released the following dataset that breaks down whether or not an individual was approved for a loan. The columns of the dataset are:

- credit.policy - If the applicant was approved for a loan
- purpose - The purpose of the loan
- int.rate - The interest rate of the loan the applicant is judged by
- installment - The monthly rate of the loan if the applicant is approved
- log.annual.inc - Log of the annual income of the applicant
- dti - The debt-to-income ratio of the applicant
- fico - The FICO credit score of the applicant
- days.with.cr.line - The number of days the applicant has had a credit line
- revol.bal - The borrower's revolving balance (unpaid amount at the end of each credit card cycle)
- revol.util - The borrower's revolving line utilization rate (unpaid percentage of used credit vs. total credit)
- inq.last.6mnths - The amount of hard inquiries the applicant has had in the past 6 months
- delinq.2yrs - The number of times the applicant has had a delinquent payment in the last 2 years
- pub.rec - The number of times the applicant has had poor financial records (bankruptcy, tax lien, judgements)

The following data will be split into 80% train and 20% test.

Load Dataset

```r
# Load the dataset
loan_data <- read.csv("./data/loan_data.xls", stringsAsFactors=TRUE)
# source: https://www.kaggle.com/datasets/itssuru/loan-data

# Split the data for 80% train and 20% test
eighty <- sample(1:nrow(loan_data), nrow(loan_data)*0.8, replace=FALSE)
train  <- loan_data[eighty, ]
test   <- loan_data[-eighty, ]
```

Here is an example of the first 3 rows of this dataset:

```r
head(loan_data, 3)
```

```
##   credit.policy            purpose int.rate installment log.annual.inc   dti
## 1             1 debt_consolidation   0.1189      829.10       11.35041 19.48
## 2             1        credit_card   0.1071      228.22       11.08214 14.29
## 3             1 debt_consolidation   0.1357      366.86       10.37349 11.63
##   fico days.with.cr.line revol.bal revol.util inq.last.6mths delinq.2yrs
## 1  737          5639.958     28854       52.1              0           0
## 2  707          2760.000     33623       76.7              0           0
## 3  682          4710.000      3511       25.6              1           0
##   pub.rec not.fully.paid
## 1       0              0
## 2       0              0
## 3       0              0
```

Data Exploration

This is a view of the summary of the dataset.

```r
summary(loan_data)
```

```
##  credit.policy                 purpose        int.rate       installment
##  Min.   :0.000   all_other         :2331   Min.   :0.0600   Min.   : 15.67
##  1st Qu.:1.000   credit_card       :1262   1st Qu.:0.1039   1st Qu.:163.77
##  Median :1.000   debt_consolidation:3957   Median :0.1221   Median :268.95
##  Mean   :0.805   educational       : 343   Mean   :0.1226   Mean   :319.09
##  3rd Qu.:1.000   home_improvement  : 629   3rd Qu.:0.1407   3rd Qu.:432.76
##  Max.   :1.000   major_purchase    : 437   Max.   :0.2164   Max.   :940.14
##                  small_business    : 619
##  log.annual.inc        dti              fico        days.with.cr.line
##  Min.   : 7.548   Min.   : 0.000   Min.   :612.0   Min.   :  179
##  1st Qu.:10.558   1st Qu.: 7.213   1st Qu.:682.0   1st Qu.: 2820
##  Median :10.929   Median :12.665   Median :707.0   Median : 4140
##  Mean   :10.932   Mean   :12.607   Mean   :710.8   Mean   : 4561
##  3rd Qu.:11.291   3rd Qu.:17.950   3rd Qu.:737.0   3rd Qu.: 5730
##  Max.   :14.528   Max.   :29.960   Max.   :827.0   Max.   :17640
##
##    revol.bal         revol.util    inq.last.6mths    delinq.2yrs
```

```
##  Min.   :      0   Min.   :  0.0   Min.   : 0.000   Min.   : 0.0000
##  1st Qu.:   3187   1st Qu.: 22.6   1st Qu.: 0.000   1st Qu.: 0.0000
##  Median :   8596   Median : 46.3   Median : 1.000   Median : 0.0000
##  Mean   :  16914   Mean   : 46.8   Mean   : 1.577   Mean   : 0.1637
##  3rd Qu.:  18250   3rd Qu.: 70.9   3rd Qu.: 2.000   3rd Qu.: 0.0000
##  Max.   :1207359   Max.   :119.0   Max.   :33.000   Max.   :13.0000
##
##     pub.rec        not.fully.paid
##  Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.00000   Median :0.0000
##  Mean   :0.06212   Mean   :0.1601
##  3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :5.00000   Max.   :1.0000
##
```

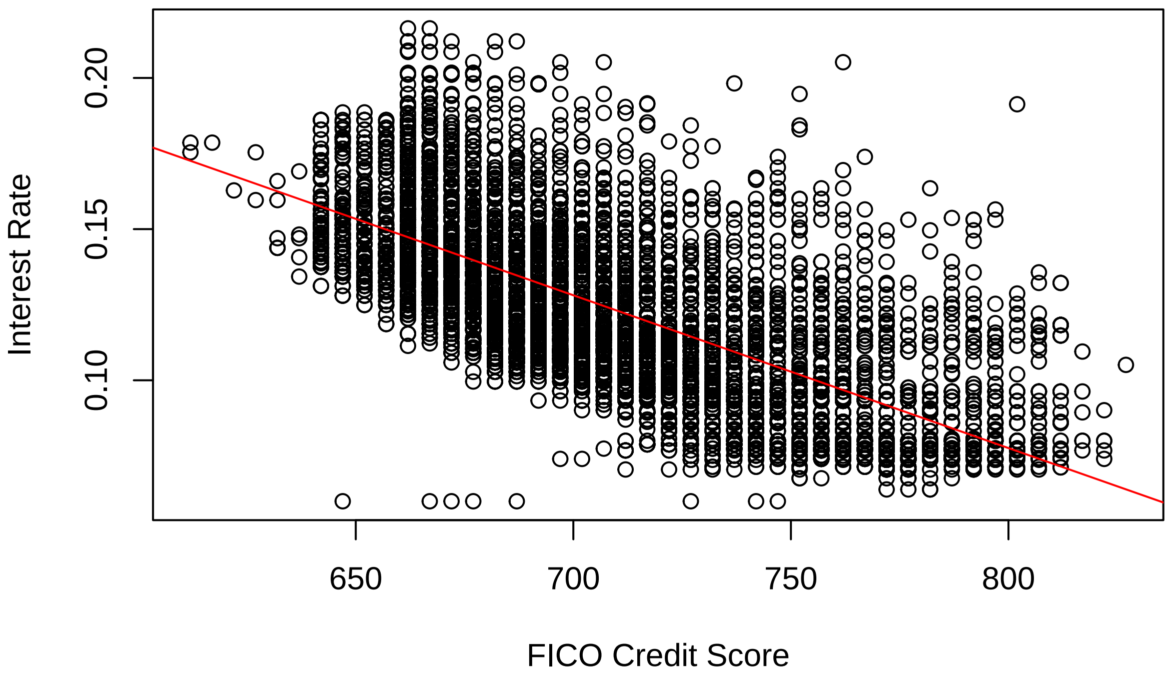
Statistics that are dervied from summary are that:

- 80.5% of applicants are approved for their loan
- The average interest rate is 12.26%
- The median credit score of the applicants is 707
- The average 12.6% debt-to-income rate suggests that each applicant has 12.6% of the value of their income as debt

Data Visualization

This will show some relationships within the dataset. First, there is a relationship between an applicant's projected interest rate and their credit score. This is inversely related since a higher credit score generally leads to lower interest rates.

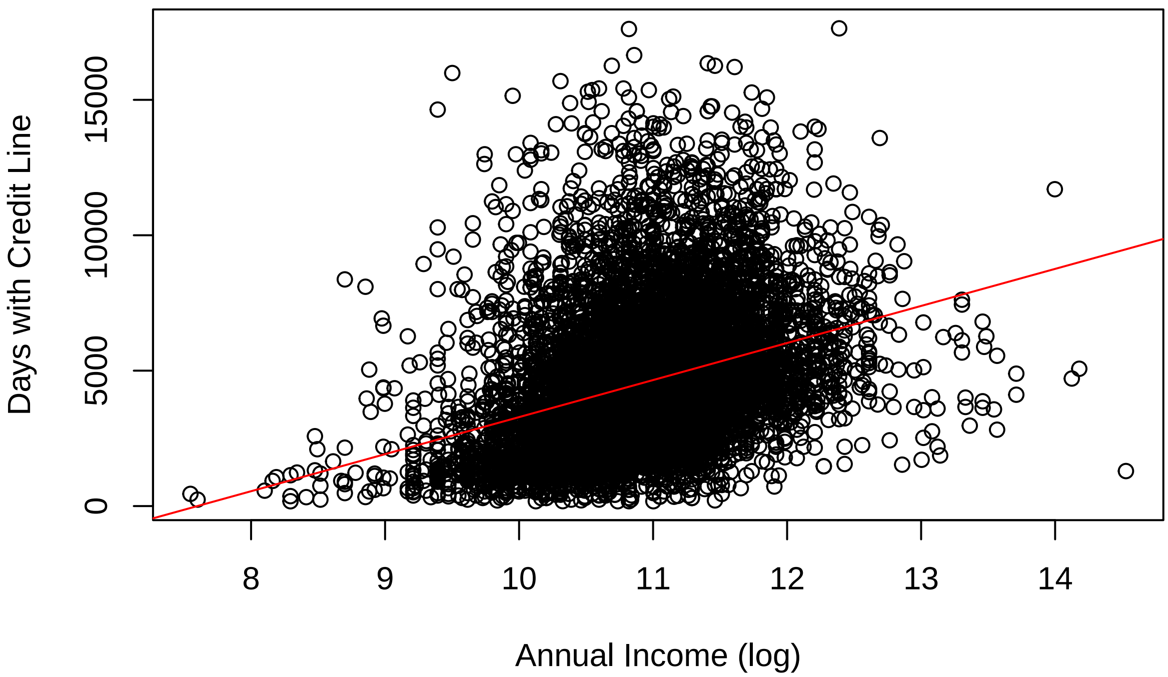
```
# Create scatter plot
plot(loan_data$fico,loan_data$int.rate, xlab="FICO Credit Score", ylab="Interest Rate")
# Create regression line
abline(lm(loan_data$int.rate ~ loan_data$fico), col="red")
```



There are outliers within the data, but the regression line shows a clear decrease in interest rate as credit score increases.

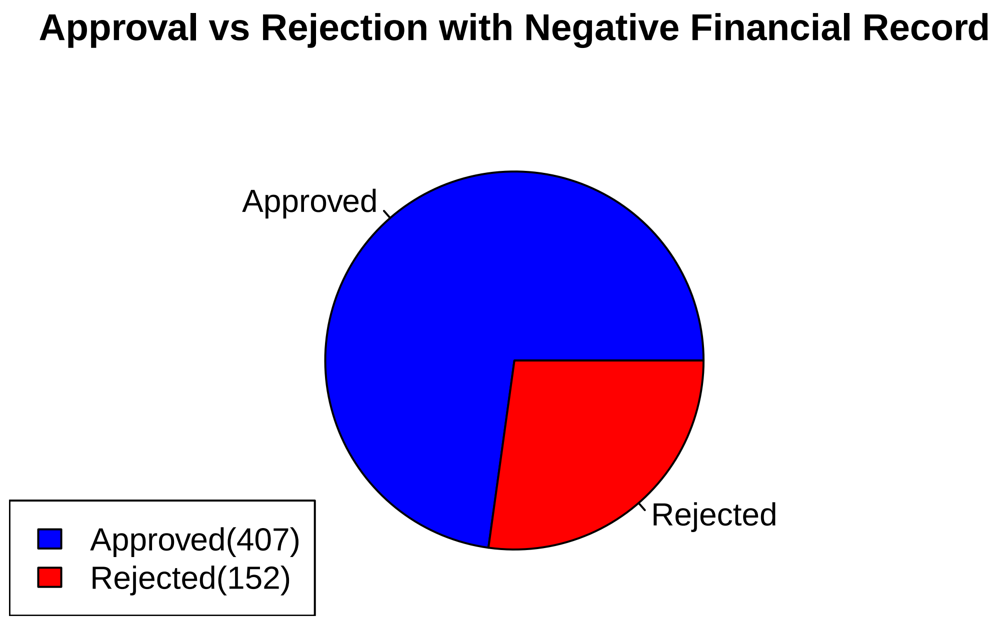
Another relationship within the dataset is the relationship between annual income and days with a credit line. The data suggests that applicants with a higher income generally have had longer days with a credit line.

```
plot(loan_data$log.annual.inc, loan_data$days.with.cr.line, xlab="Annual Income (log)", ylab="Days with
# Create regression line
abline(lm(loan_data$days.with.cr.line ~ loan_data$log.annual.inc), col="red")
```



Finally, the percentage of applicants who have had negative financial records (such as bankruptcies or leins) and had their loan approved or decline can be viewed in the following pie chart:

```
# Seperate the data into values where public record is greater than 0
neg_record <- subset(loan_data, pub.rec > 0)
neg_record$credit.policy <- ifelse(neg_record$credit.policy == 0, "Rejected", "Approved")
table_data <- table(neg_record$credit.policy)
pie(table_data, col = c("blue", "red"), main = "Approval vs Rejection with Negative Financial Record")
legend("bottomleft", legend = paste(names(table_data), "(", table_data, ")", sep = ""), fill = c("blue"
```



According to the chart, there are 407 approvals and 152 rejections when the applicant has a negative financial record. That is a 63% approval rating compared to the total of 80.5% approvals for all applicants.

Logistic Regression

This section will explore logistic regression and analyze the performance of the logistic regression model. The model utilizes binomial logistic regression and addresses all other columns as functions of the predictor variable.

```
set.seed(123)
logistic_regression <- glm(credit.policy ~ ., data = train, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
lr_prediction <- predict(logistic_regression, newdata = test, type = "response")
lr_prediction_result <- ifelse(lr_prediction >= 0.5, 1, 0)
conf_matrix <- table(test$credit.policy, lr_prediction_result)
conf_matrix
```

```
##    lr_prediction_result
##        0    1
##   0  247  141
##   1   53 1475
```

Utilizing logistic regression, we've calculated the following:

- True Negative - 234
- False Positive - 127
- False Negative - 61
- True Positive - 1494

The accuracy, precision, recall, and specifcity can be calculated below.

```
TN <- 234
FP <- 127
FN <- 61
TP <- 1494

# Calculate accuracy
accuracy <- (TP + TN) / (TP + TN + FP + FN)
accuracy <- accuracy * 100
accuracy
```

```
## [1] 90.18789
```

```
# Calculate sensitivity
sensitivity <- TP / (TP + FN)
sensitivity <- sensitivity * 100
sensitivity
```

```
## [1] 96.07717
```

```
# Calculate specificity
specificity <- TN / (TN + FP)
specificity <- specificity * 100
specificity
```

```
## [1] 64.81994
```

The confusion matrix shows that there is a 90% accuracy in the logistic regression model. The sensitivity of 96% shows that there is a high percent of true positive classifications in the model. The specificity at 64.82% shows that the predictions on True Negative classifications are low in accuracy.

kNN Classification Model

This section will explore classification by using kNN. kNN is based on similarity. It finds the k closest data points to the current object it is attempting to predict, then makes a prediction based on the outcome of the k closest points. The model below only utilizes k = 1, which means it compares data to the closest neighbor.

```
library(class)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3

## Loading required package: lattice
```

```
train_no_purpose <- train[,-2]
test_no_purpose <- test[,-2]
knn_predictions <- knn(train_no_purpose[, -1], test_no_purpose[, -1], train_no_purpose[, 1], 1)
confusion_matrix <- confusionMatrix(knn_predictions, as.factor(test_no_purpose$credit.policy))
print(confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  148  229
##          1  240 1299
##
##                Accuracy : 0.7552
##                  95% CI : (0.7353, 0.7743)
##     No Information Rate : 0.7975
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.234
##
##  Mcnemar's Test P-Value : 0.6443
##
##             Sensitivity : 0.38144
##             Specificity : 0.85013
##          Pos Pred Value : 0.39257
##          Neg Pred Value : 0.84405
##              Prevalence : 0.20251
##          Detection Rate : 0.07724
##    Detection Prevalence : 0.19676
##       Balanced Accuracy : 0.61579
##
##        'Positive' Class : 0
##
```

The confusion matrix shows us the following statistics:

- True Negative - 117
- False Positive - 231
- False Negative - 240
- True Positive - 1328

```
TN <- 117
FP <- 231
FN <- 240
TP <- 1328
```

```r
# Calculate accuracy
accuracy <- (TP + TN) / (TP + TN + FP + FN)
accuracy <- accuracy * 100
accuracy
```

```
## [1] 75.41754
```

```r
# Calculate sensitivity
sensitivity <- TP / (TP + FN)
sensitivity <- sensitivity * 100
sensitivity
```

```
## [1] 84.69388
```

```r
# Calculate specificity
specificity <- TN / (TN + FP)
specificity <- specificity * 100
specificity
```

```
## [1] 33.62069
```

The data shows us that we have a 75.42% accuracy. While the sensitivity is relatively high at 84.69%, the specificity is low at 33.62%. This suggests that we have a high percentage of True Positive classifications. However, the number of True Negative classifications are extremely low. This could be due to the k = 1 utilization. Since there are high percentage of approvals, the True Negative accuracy decreases since the closest k point could be an approval.

Decision Trees

The final model being utilized are Decision Trees. Every child in the tree would be a decision from one of the predictor variables. At the leaf of the tree, a decision is made whether or not there was a loan approval.

```r
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.2.3
```

```r
tree_model <- rpart(credit.policy ~ ., data = train, method = "class")
tree_predictions <- predict(tree_model, newdata = test, type = "class")
confusionMatrix(table(tree_predictions, test$credit.policy))
```

```
## Confusion Matrix and Statistics
##
##
## tree_predictions    0    1
##                0  367    4
##                1   21 1524
##
##                Accuracy : 0.987
##                  95% CI : (0.9808, 0.9915)
```

```
##     No Information Rate : 0.7975
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9589
##
##  Mcnemar's Test P-Value : 0.001374
##
##             Sensitivity : 0.9459
##             Specificity : 0.9974
##          Pos Pred Value : 0.9892
##          Neg Pred Value : 0.9864
##              Prevalence : 0.2025
##          Detection Rate : 0.1915
##    Detection Prevalence : 0.1936
##       Balanced Accuracy : 0.9716
##
##        'Positive' Class : 0
##
```

The confusion matrix from the decision tree gives us the following statistics:

- True Negative - 346
- False Positive - 1
- False Negative - 29
- True Positive - 1533

```r
TN <- 346
FP <- 1
FN <- 29
TP <- 1533

# Calculate accuracy
accuracy <- (TP + TN) / (TP + TN + FP + FN)
accuracy <- accuracy * 100
accuracy
```

```
## [1] 98.4285
```

```r
# Calculate sensitivity
sensitivity <- TP / (TP + FN)
sensitivity <- sensitivity * 100
sensitivity
```

```
## [1] 98.14341
```

```r
# Calculate specificity
specificity <- TN / (TN + FP)
specificity <- specificity * 100
specificity
```

```
## [1] 99.71182
```

According to the data, there is a 98.43% accuracy, a 98.14% sensitivity, and a 99.71% specificity. These numbers are significantly higher than the other 2 models and display superior accuracy during prediction.

Comparisons

As a breakdown, here are the results of each model again:

Logistic Regression

Accuracy: 90.19%

Sensitivity: 96.08%

Specificity: 64.82%

Logistic regression is fairly accurate. However, there is a lower rate of True Negative detection. The reason for these statistics may be that the training data contains a significantly greater amount of positive cases where an applicant was approved for a loan than negative cases.

kNN

Accuracy: 75.42%

Sensitivity: 84.69%

Specificity: 33.62%

The kNN model has an extremely low specificity in which there are a lot less True Negatives being accurately detected. Since the model I used above utilized a k = 1, this means the kNN was only searching for the nearest neighbor of the current node. Since the dataset had significantly more people approved for a loan than denied, the nearest neighbor was most likely approved for a loan.

Decision Trees

Accuracy: 98.43%

Sensitivity: 98.14%

Specificity: 99.71%

The decision tree model was the most accurate of all, with an accuracy rate of 98.43%. Since both sensitivity and specificity were also high, the model had a high overall accuracy. The interactions between each of the predictors were captured accurately by the decision tree and was overall a good model for this dataset.