Classification

David Favela Corella

Contents

Linear Models for Classification

Linear models for classification refers to models that follow a linear trend in the data set but are to be classified according to an attribute. These models tend to have as a target a binary column, which will make it easier to make predictions based on a 0/1 input. Some of the strengths these algorithm have include being easy to implement and interpret. Also, they work well with small data sets. Some weaknesses include being limited to the binary selection, which leads to transforming the data to suit the model.

Logistic Regression Model

Based on classification of data. While Linear regression allowed for quantitative data, logistic regression focuses on classifying and predicting a data set based on a certain factor. A relatively easy model to construct would imply having a binary attribute from which we'd be able to predict different attributes. It's strengths feature being relatively easy to compute and giving out a nice probability output. Its weakness is its inability to perform outside of binary data.

Load the data from files

```
df <- read.csv("bank.csv", header=TRUE)
str(df)
```

```
## 'data.frame':    10000 obs. of  12 variables:
##  $ customer_id     : int  15634602 15647311 15619304 15701354 15737888 15574012 15592531 15656148 15
##  $ credit_score    : int  619 608 502 699 850 645 822 376 501 684 ...
##  $ country         : chr  "France" "Spain" "France" "France" ...
##  $ gender          : int  0 0 0 0 0 1 1 0 1 1 ...
##  $ age             : int  42 41 42 39 43 44 50 29 44 27 ...
##  $ tenure          : int  2 1 8 1 2 8 7 4 4 2 ...
```

```
##  $ balance         : num  0 83808 159661 0 125511 ...
##  $ products_number : int  1 1 3 2 1 2 2 4 2 1 ...
##  $ credit_card     : int  1 0 1 0 1 1 1 1 0 1 ...
##  $ active_member   : int  1 1 0 0 1 0 1 0 1 1 ...
##  $ estimated_salary: num  101349 112543 113932 93827 79084 ...
##  $ churn           : int  1 0 1 0 0 1 0 1 0 0 ...
```

Data cleaning & Separate data

We only care about four distinct attributes: credit score, gender, estimated salary, and the credit card We transform the gender into F or M, from the binary data provided. We also make sure the credit score and estimated salary remains numeric, while the gender and credit card status become factors.

```
df <- df[,c(2,4,11,9)]
df$credit_score <- as.numeric(df$credit_score)
df[df$gender == 0,]$gender <- "F"
df[df$gender == 1,]$gender <- "M"
df$gender <- as.factor(df$gender)
df$estimated_salary <- as.numeric(df$estimated_salary)
df$credit_card <- as.factor(df$credit_card)
head(df)
```

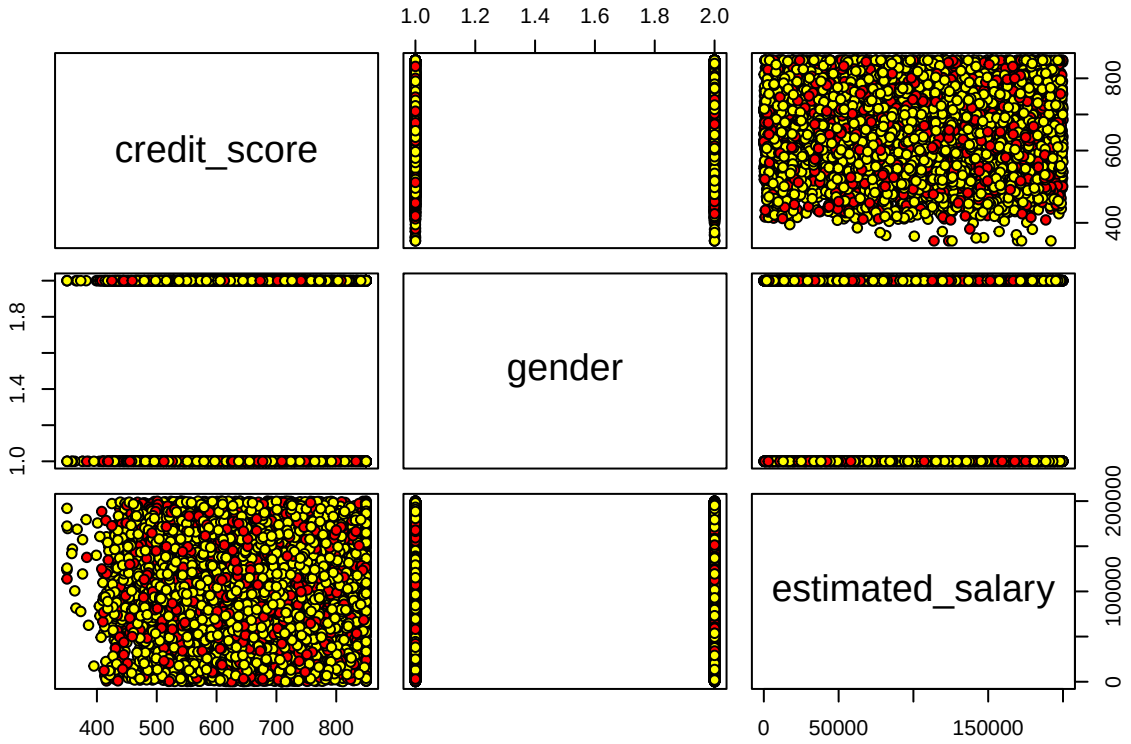
```
##   credit_score gender estimated_salary credit_card
## 1          619      F        101348.88           1
## 2          608      F        112542.58           0
## 3          502      F        113931.57           1
## 4          699      F         93826.63           0
## 5          850      F         79084.10           1
## 6          645      M        149756.71           1
```

```
str(df)
```

```
## 'data.frame':    10000 obs. of  4 variables:
##  $ credit_score    : num  619 608 502 699 850 645 822 376 501 684 ...
##  $ gender          : Factor w/ 2 levels "F","M": 1 1 1 1 1 2 2 1 2 2 ...
##  $ estimated_salary: num  101349 112543 113932 93827 79084 ...
##  $ credit_card     : Factor w/ 2 levels "0","1": 2 1 2 1 2 2 2 2 1 2 ...
```

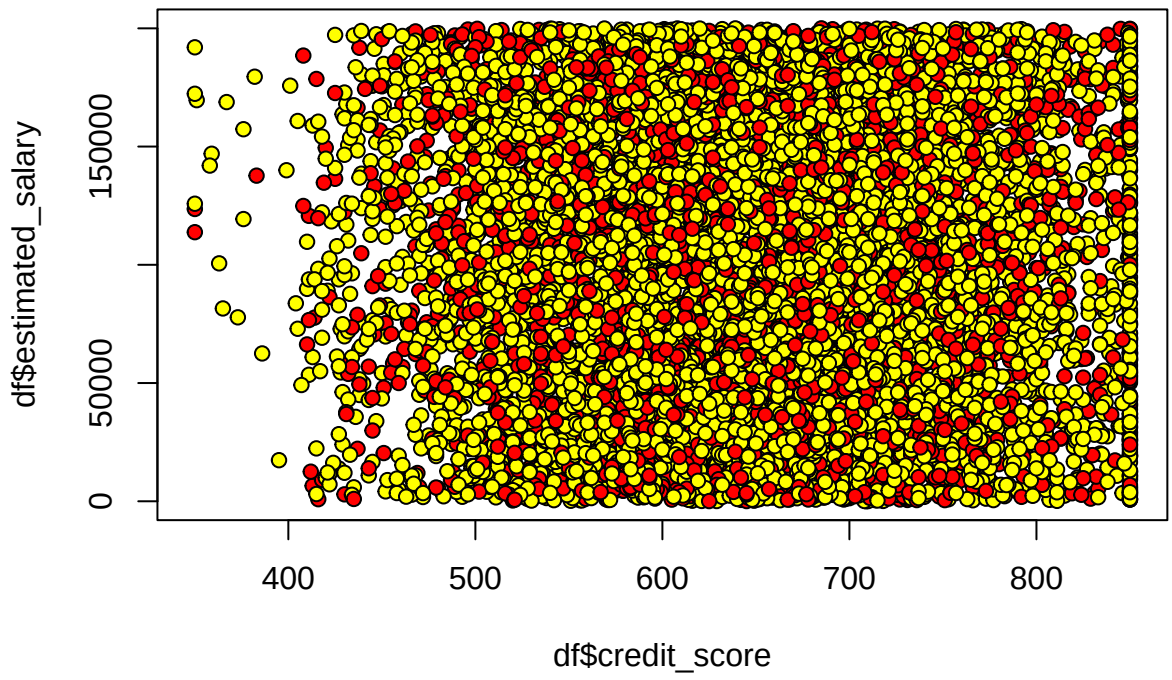
We run a plot of pairs on the data, dividing the red plots on those without a credit card and those on yellow as the ones with credit cards.

```
pairs(df[1:3], pch = 21, bg = c("red", "yellow")[unclass(df$credit_card)])
```



On a close up, we can see that when comparing credit score with estimated salary, there seems to be a higher amount of people with credit cards with a credit score of more than 800.

```
plot(df$credit_score, df$estimated_salary, pch=21, bg=c("red","yellow")
     [unclass(df$credit_card)])
```



Check if there are any missing values on the data.

```
sapply(df, function(x) sum(is.na(x)==TRUE))
```

```
##     credit_score           gender estimated_salary      credit_card
##                0                0                0                0
```

Set the train and test sets for the models

We set the seed and divide the sets into 80/20 for train and test.

```
set.seed(1234)
i <- sample(1:nrow(df), 0.80*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Logistic regression

Create the model based on the credit card.

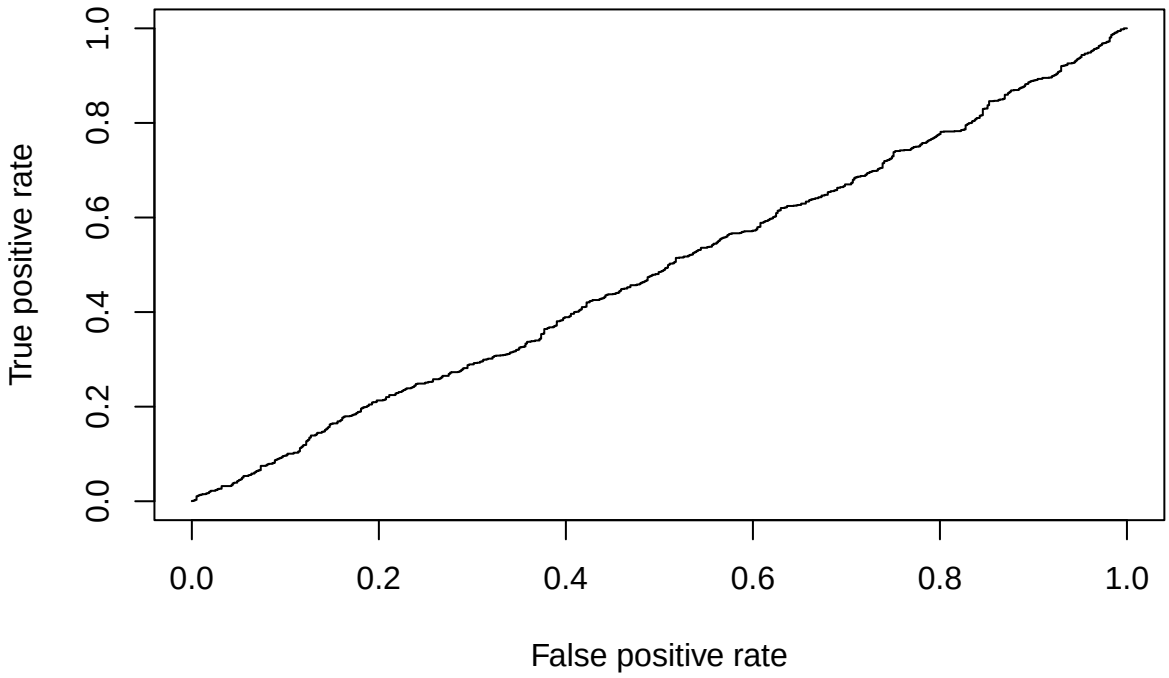
```
glm1 <- glm(credit_card ~ ., data=train, family="binomial")
summary(glm1)
```

```
##
## Call:
## glm(formula = credit_card ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6030  -1.5439   0.8247   0.8396   0.8640
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       8.942e-01  1.758e-01   5.087 3.63e-07 ***
## credit_score      2.548e-05  2.554e-04   0.100    0.921
## genderM           4.696e-02  4.926e-02   0.953    0.340
## estimated_salary -5.725e-07  4.264e-07  -1.343    0.179
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9684.1  on 7999  degrees of freedom
## Residual deviance: 9681.3  on 7996  degrees of freedom
## AIC: 9689.3
##
## Number of Fisher Scoring iterations: 4
```

###ROC on Logistic Regression

The ROC test lets us know how good the data fits on the threshold. We can see that it increases along the graph linearly, which is not a bad thing but it would be more reassuring if the positive rate reached 1 early on.

```
library(ROCR)
p <- predict(glm1, newdata=test, type="response")
pr <- prediction(p, test$credit_card)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```



Test and predict the model

We then use the test data set to test out the model. We see that the accuracy of the model is of .70, which is not bad but also not amazing.

```
probs <- predict(glm1,  newdata=test, type="response")
pred <- ifelse(probs> 0.5, 1, 0)
acc <- mean(pred == test$credit_card)
print(paste("accuracy = ", acc))
```

```
## [1] "accuracy =  0.7015"
```

```
table(pred, test$credit_card)
```

```
##
## pred    0    1
##    1  597 1403
```

Using the GGally library

GGally gives us plots of the logistic model for credit card. It lets us know that there is a trend in several graphs, like in credit score against credit card, credit score against gender, which we didn't look into in this model.

```r
library("GGally")
```

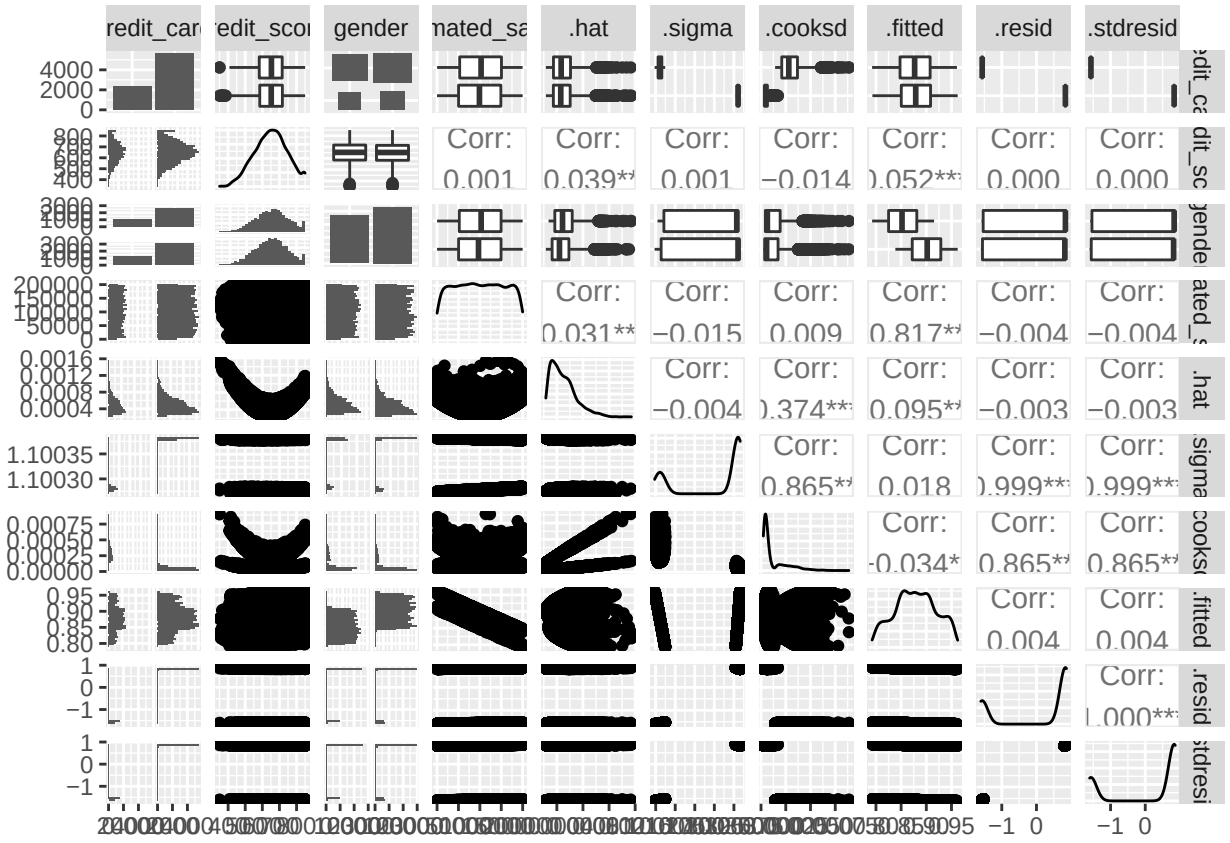
```
## Loading required package: ggplot2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
par(mfrow=c(1,2))
ggpairs(glm1)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Bayes Algorithmn

The Naive Bayes algorithm is a classifier algorithm, similar to the logistic regression algorithm. It is easier to implement and interpret. Some of its strengths come from the fact that it works well with small data sets and it can handle high dimensions well. Some issues it has is that is it reliable but doesn't perform as well in large data sets. It also can make guesses for the test values that were not present in the training data set.

Here we set up the Naive Bayes model from the e1071 library.

```
library(e1071)
nb1 <- naiveBayes(credit_card~., data=train, family = "binomial")
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace, family = "binomial")
##
## A-priori probabilities:
## Y
##      0      1
## 0.2935 0.7065
##
## Conditional probabilities:
```

```
##    credit_score
## Y       [,1]     [,2]
##   0 650.5170 96.19211
##   1 650.7219 96.13926
##
##    gender
## Y           F         M
##   0 0.4667802 0.5332198
##   1 0.4548832 0.5451168
##
##    estimated_salary
## Y        [,1]     [,2]
##   0 101694.61 57789.20
##   1  99774.95 57540.51
```

Evaluate and predict on the test data

We run the test cases and find out that the accuracy of the model is the same as the logistic model created previosly.

```
p1 <- predict(nb1, newdata=test, type="class")
table(p1, test$credit_card)
```

```
##
## p1     0    1
##   0    0    0
##   1  597 1403
```

```
mean(p1==test$credit_card)
```

```
## [1] 0.7015
```

Extracting probabilities

Here we can see some of the probabilities that the model created for being a credit card user or not.

```
p1_raw <- predict(nb1, newdata=test, type="raw")
head(p1_raw)
```

```
##              0         1
## [1,] 0.2995505 0.7004495
## [2,] 0.2968491 0.7031509
## [3,] 0.2790646 0.7209354
## [4,] 0.3024347 0.6975653
## [5,] 0.2916033 0.7083967
## [6,] 0.2961055 0.7038945
```

Results from the models

The results gotten from both models were equal. Both resulted with the accuracy of their algorithms being 0.7015 I believe we got these results because the data sets were clearly defined and both algorithms work towards the same objective, which is to create a model to predict this binary data.

The classifiers used in these algorithms include the credit card, credit score, gender, and estimated salary. The credit card and gender are both binary values. 0 is no credit card/woman, while 1 is credit card/man. The credit score and estimated salary are both numeric values from which we can formulate our predictions. It is also to be noted that the credit card status is the attribute we base the models on.