# Multivariable Logistic Binary Classifier - Delinquency Prediction

The panel data-set contains commercial customers’ financial information and days past due indicator from 2000 to 2020. The goal is to build a model to predict when customers will be 90+ days past due **(90+DPD)** on payments.

## Prepare Training Data

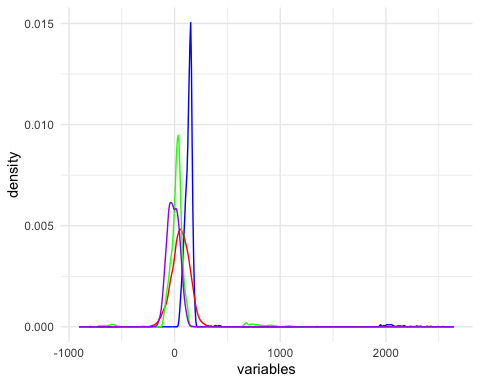
setwd("/Users/jacobrichards/Desktop/DS\_DA\_Projects/Delinquency\_Prediction")  
train <- read.csv(file="/Users/jacobrichards/Desktop/DS\_DA\_Projects/Delinquency\_Prediction/Data\_Files/FITB\_train.csv",header=TRUE)  
test <- read.csv(file="/Users/jacobrichards/Desktop/DS\_DA\_Projects/Delinquency\_Prediction/Data\_Files/FITB\_test.csv",header=TRUE)

From checking the distribution of the data, you can see that the distribution of feature 3 (displayed by the blue curve) has many values in the extreme right tail.

library(ggplot2)  
ggplot() + geom\_density(data=train, aes(x=feature\_3), color="blue") +  
 geom\_density(data=train, aes(x=feature\_2), color="red") +  
 geom\_density(data=train, aes(x=feature\_1), color="green") +  
 geom\_density(data=train, aes(x=feature\_4), color="purple") +  
 theme\_minimal() + xlab("variables")

## Warning: Removed 136 rows containing non-finite outside the scale range  
## (`stat\_density()`).

## Warning: Removed 87 rows containing non-finite outside the scale range  
## (`stat\_density()`).



The many outliers adds noise that disrupts our model’s ability to capture the trend of the data, so we removed the top and bottom percentiles from feature 3. This is known as **Winsorization**.

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

train$key <- row.names(train)  
feature\_3\_winsor <- data.frame(feature\_3 = train$feature\_3, key = row.names(train))  
feature\_3\_winsor\_clean <- na.omit(feature\_3\_winsor)  
  
feature\_3\_winsor\_clean <- feature\_3\_winsor\_clean %>%  
 mutate(z\_score = (feature\_3 - mean(feature\_3)) / sd(feature\_3),percentile = ecdf(feature\_3)(feature\_3) \* 100)  
  
feature\_3\_winsor\_df <- feature\_3\_winsor\_clean[!(feature\_3\_winsor\_clean[, 4] < 1 | feature\_3\_winsor\_clean[, 4] > 99), ]  
  
non\_matching\_keys <- anti\_join(train, feature\_3\_winsor\_df, by = "key")$key  
  
train <- train %>% mutate(feature\_3 = ifelse(key %in% non\_matching\_keys, NA, feature\_3))  
  
colnames(train)[3] <- "feature\_3\_winsor"

We need to fill in the blanks from the values we just removed, so we will replace them with the median of the non-outliers of feature 3.

train[is.na(train[,3]),3] <- median(feature\_3\_winsor\_clean$feature\_3)  
  
colnames(train)[3] <- "feature\_3\_impute"  
  
test[is.na(test[,3]),3] <- median(feature\_3\_winsor\_clean$feature\_3)  
colnames(test)[3] <- "feature\_3\_impute"

Feature 2 has missing values, so for each missing value for we will fill in the blank with the next year’s value or the previous year’s (if the next year is also missing) corresponding to the same ID.

train$date <- format(as.Date(train$date, format = "%Y-%m-%d"), "%Y")  
  
train <- train %>%  
 arrange(id, date) %>% # Sort by id and date  
 group\_by(id) %>%  
 mutate(feature\_2 = ifelse(is.na(feature\_2),  
 lead(feature\_2, order\_by = date), # Try next year  
 feature\_2)) %>%  
 mutate(feature\_2 = ifelse(is.na(feature\_2),  
 lag(feature\_2, order\_by = date), # Try previous year  
 feature\_2))  
  
colnames(train)[2] <- "feature\_2\_impute"  
  
  
test <- test %>%  
 arrange(id, date) %>%   
 group\_by(id) %>%  
 mutate(feature\_2 = ifelse(is.na(feature\_2),  
 lead(feature\_2, order\_by = date), # Try next year  
 feature\_2)) %>%  
 mutate(feature\_2 = ifelse(is.na(feature\_2),  
 lag(feature\_2, order\_by = date), # Try previous year  
 feature\_2))  
  
colnames(test)[2] <- "feature\_2\_impute"  
  
train <- na.omit(train)  
test <- na.omit(test)  
  
your\_tibble <- head(train,5)  
library(knitr)  
library(kableExtra)

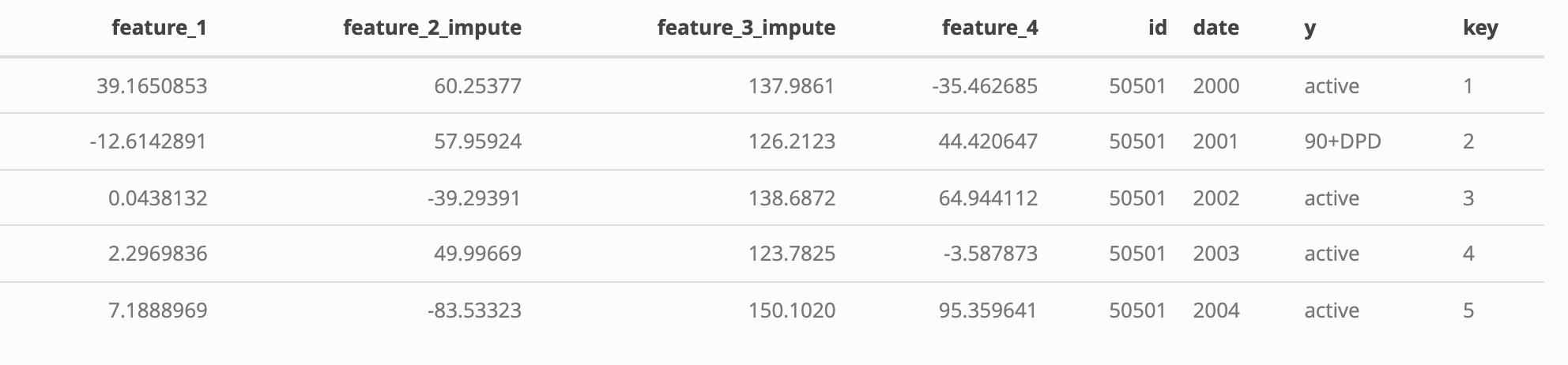
##   
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':  
##   
## group\_rows

kable(your\_tibble, format = "html") %>%  
 kable\_styling(position = "center") %>%  
 save\_kable(file = "~/Desktop/DS\_DA\_Projects/Delinquency\_Prediction/ReadMe\_files/figure-gfm/t4.png", zoom = 2)

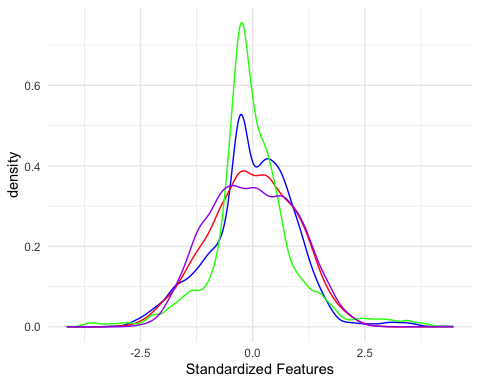
## save\_kable will have the best result with magick installed.

knitr::include\_graphics("~/Desktop/DS\_DA\_Projects/Delinquency\_Prediction/ReadMe\_files/figure-gfm/t4.png")



Our features (variables) all represent different financial measurements quantified by different units. In order for these variables to be measured uniformly, we can reassign each value with its corresponding z-score within its respective variable distribution.

library(dplyr)  
train <- train %>%  
 mutate(across(c(feature\_1, feature\_2\_impute, feature\_3\_impute, feature\_4),   
 ~ (.x - mean(.x, na.rm = TRUE)) / sd(.x, na.rm = TRUE)))  
  
colnames(train) <- c("feature\_1\_standard","feature\_2\_standard","feature\_3\_standard","feature\_4\_standard","id","date","y","key")  
  
test <- test %>%  
 mutate(across(c(feature\_1, feature\_2\_impute, feature\_3\_impute, feature\_4),   
 ~ (.x - mean(.x, na.rm = TRUE)) / sd(.x, na.rm = TRUE)))  
  
colnames(test) <- c("feature\_1\_standard","feature\_2\_standard","feature\_3\_standard","feature\_4\_standard","id","date","y")  
  
ggplot() +   
 geom\_density(data = train, aes(x = feature\_3\_standard), color = "blue") +  
 geom\_density(data = train, aes(x = feature\_2\_standard), color = "red") +  
 geom\_density(data = train, aes(x = feature\_1\_standard), color = "green") +  
 geom\_density(data = train, aes(x = feature\_4\_standard), color = "purple") +  
 theme\_minimal() +  
 labs(x = "Standardized Features")



The preparation of the training data is complete.

## Building The Model

Building a logistic regression model from features 1 to 4 as continuous independent variables and column y as the binary dependent variable (true/false).

Given historical data of customers’ financial information and whether or not they were **90+ days past due** on payments, we can produce a model that will generate a probability that a customer will be **90+ days past due** on payments.

For explanation of logistic regression binary classifiers see the following invaluable resource: <https://seantrott.github.io/binary_classification_R/>

When the model produces a probability of an individual being **90+DPD**, we will have to decide at what probability we draw the conclusion that the individual will be **90+DPD**. The value chosen has great impact on the accuracy of the model.

## Fitting The Model

**decision threshold:** The minimum predicted probability value from the model that a customer will be **90+ DPD** at which it is concluded that the customer will be **90+ DPD**.

Since the outcome of whether or not there will be late payments is known in our testing data; we can asses the accuracy of the model by evaluating the model on the testing data and directly comparing the predicted outcomes to the actual outcomes, as well assesing the impact of the **decision threshold** we used on the model’s accuracy.

The testing data is completely distinct from the data used to produce the model, so the accuracy results of the model being evaluated on it are representative of the model’s accuracy being evaluated of future data. The following analysis are those accuracy results.

To asses the accuracy of the model and find the optimal decision threshold we produce the ROC curve.

library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

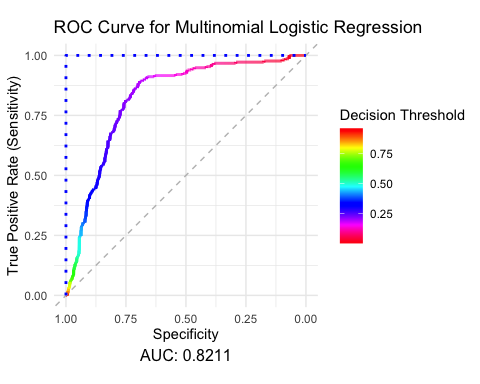
test$predicted\_y <- predict(delinquency\_model, newdata = test, type = "class")  
 test$y\_numeric <- as.numeric(as.character(factor(test$y, levels = c("90+DPD", "active"), labels = c(1, 0))))  
 test$Probability <- predict(delinquency\_model, newdata = test, type = "probs")  
 options(digits = 4)  
   
roc\_curve <- roc(response = test$y\_numeric, predictor = test$Probability)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

roc\_metrics <- coords(roc\_curve, x = "all", ret = c("threshold", "sensitivity", "specificity"))  
  
auc\_value <- auc(roc\_curve)  
  
roc\_data <- data.frame(TPR = roc\_metrics$sensitivity,FPR = roc\_metrics$specificity,Threshold = roc\_metrics$threshold)  
  
ggplot(roc\_data, aes(x = FPR, y = TPR, color = Threshold)) +  
 geom\_line(size = 1) +  
 geom\_abline(slope = 1, intercept = 1, linetype = "dashed", color = "gray") +   
 geom\_line(data = data.frame(FPR = c(1, 1, 0), TPR = c(0, 1, 1)), aes(x = FPR, y = TPR),   
 color = "blue", size = 1, linetype = "dotted") +  
 labs(title = "ROC Curve for Multinomial Logistic Regression",  
 x = "Specificity",  
 y = "True Positive Rate (Sensitivity)",  
 caption = paste("AUC:", round(auc\_value, 4)),  
 color = "Decision Threshold") +  
 scale\_color\_gradientn(colors = rev(rainbow(100))) +  
 coord\_fixed() +  
 scale\_x\_reverse() + # Reverse the x-axis  
 ylim(0, 1) +  
 theme\_minimal() +  
 theme(plot.caption = element\_text(hjust = 0.5, size = 12))

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



### The ROC curve is a plot of the models prediction accuracy ratios Sensitivity (y-axis) by Specificity (x-axis) as a result of the decision threshold chosen which is displayed by color gradient.

**Sensitivity:** *what proportion of individuals who were* ***90+ DPD*** *did the model correctly predict as being* ***90+ DPD.***

**Specificity:** *what proportion of individuals who were* ***not 90+ DDP*** *did the model correctly predict as being* ***not 90+ DPD.***

**AUC:** The area under the ROC curve, AUC, is used as a metric for overall model performance as the ROC curve is the result of accuracy metrics from the entire range of decision thresholds. The blue dotted line is a perfect model containing 100% of the area under its curve. The grey line is if you were to predict the outcome by tossing a coin and thus naturally the area under it’s curve is 50%.

The **AUC** of our model is 0.8211.

To illustrate the meaning of this curve, take for example the accuracy results if you were to select a decision threshold of 0.50 represented as cyan blue: Your specificity would be about 94%, which is good since you did not falsely predict too many late payments. However, your sensitivity would only be 25%, such that you only successfully predicted 25% of the late payments. Conversely if you had selected a decision threshold of .10 represented as pink-red you would successfully predict 94% of late payments but only 25% of individuals who were not late on payments were correctly identified as such by the model.

Therefore, the **decision threshold** we choose is a trade-off between successfully predicting late payments and successfully predicting non-late payments.

The **decision threshold** which balances these goals is visually evident in the following plot.

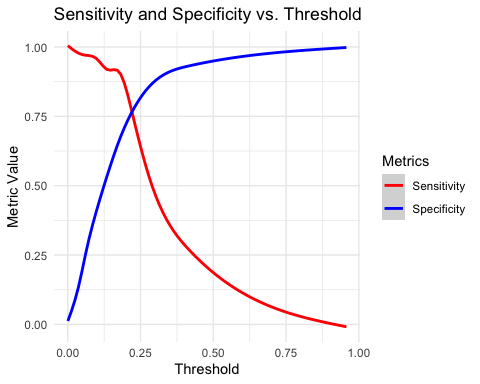
ggplot(roc\_metrics, aes(x = threshold)) +  
 geom\_smooth(aes(y = sensitivity, color = "Sensitivity")) +  
 geom\_smooth(aes(y = specificity, color = "Specificity")) +  
 labs(title = "Sensitivity and Specificity vs. Threshold",  
 x = "Threshold", y = "Metric Value") +  
 scale\_color\_manual(name = "Metrics", values = c("Sensitivity" = "red", "Specificity" = "blue")) +  
 theme\_minimal()

## `geom\_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'

## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_smooth()`).

## `geom\_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'

## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_smooth()`).

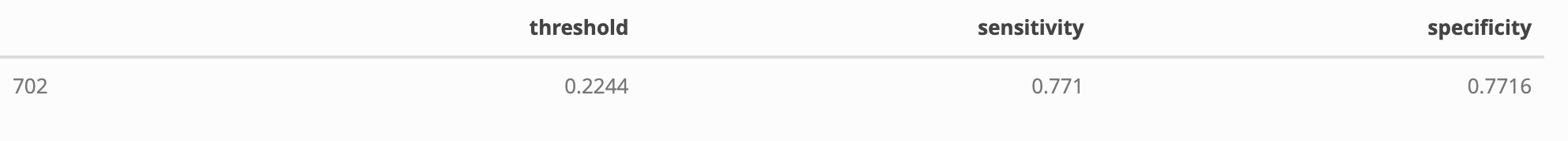


The balanced **decision threshold** is visually apparent by the intersection of **Sensitivity** and **Specificity** and by the following simple calculation.

optimal\_threshold <- roc\_metrics$threshold[which.min(abs(roc\_metrics$sensitivity - roc\_metrics$specificity))]  
your\_tibble <- roc\_metrics[roc\_metrics[,1] ==optimal\_threshold,]  
library(knitr)  
library(kableExtra)  
  
kable(your\_tibble, format = "html") %>%  
 kable\_styling(position = "center") %>%  
 save\_kable(file = "~/Desktop/DS\_DA\_Projects/Delinquency\_Prediction/ReadMe\_files/figure-gfm/t2.png", zoom = 2)

## save\_kable will have the best result with magick installed.

knitr::include\_graphics("~/Desktop/DS\_DA\_Projects/Delinquency\_Prediction/ReadMe\_files/figure-gfm/t2.png")



Confusion matrix displaying the results of balanced decision threshold evaluated on the testing data.

test$predicted\_class <- ifelse(test$Probability >= roc\_metrics$threshold[which.min(abs(roc\_metrics$sensitivity - roc\_metrics$specificity))], 1, 0)  
  
library(caret)

## Loading required package: lattice

conf\_matrix <- confusionMatrix(  
 factor(test$predicted\_class, levels = c(0, 1)),  
 factor(test$y\_numeric, levels = c(0, 1)))  
  
confusion\_table <- as.data.frame.matrix(conf\_matrix$table)  
rownames(confusion\_table) <- c("Actual: Non-delinquent", "Actual: Delinquent")  
colnames(confusion\_table) <- c("Predicted: Non-delinquent", "Predicted: Delinquent")  
  
  
library(kableExtra)  
library(webshot2)  
  
kable(confusion\_table, format = "html") %>%  
 kable\_styling(position = "center") %>%  
 save\_kable(file = "~/Desktop/DS\_DA\_Projects/Delinquency\_Prediction/ReadMe\_files/figure-gfm/t3.png", zoom = 2)

## save\_kable will have the best result with magick installed.

knitr::include\_graphics("~/Desktop/DS\_DA\_Projects/Delinquency\_Prediction/ReadMe\_files/figure-gfm/t3.png")

