hw\_week4

Ben Wilson

June 8, 2019

# Week 3 Homework

*Advanced Data Prep and Regression*

## Setup

The packages included to run this notebook are:

* tidyverse - *for convenience*
* tree - *for the regression tree*
* rpart - *an alternative for the regression tree*
* randomForest - *for the random forest*
* caret - *for the confusion matrix*
* ROCR - *for the ROC curve*

## 9.1

**Using the same crime data set uscrime.txt as in Question 8.2, apply Principal Component Analysis and then create a regression model using the first few principal components. Specify your new model in terms of the original variables (not the principal components), and compare its quailty to that of your solution to Question 8.2.**

uscrime <- read.delim("data/uscrime.txt")

First, let’s pull in the answer from last week. I tested out several different models and found Po1 + M.F + Wealth were the 3 best features for the model.

# Po1, M.F, and Wealth  
crime\_lm6 <- lm(Crime ~ Po1 + M.F + Wealth, data = uscrime)

Now, let’s compress the target variables.

# Divide data into predictors and target variable  
uscrime\_p <- select(uscrime, -Crime)  
uscrime\_t <- select(uscrime, Crime)

Compress those predictors!  
*Don’t forget to scale and center.*

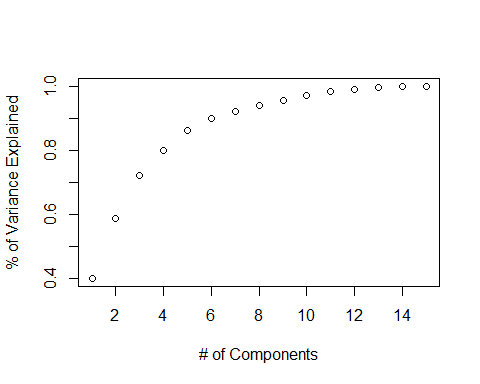
uscrime\_pca <- prcomp(uscrime\_p, center = TRUE, scale = TRUE)  
uscrime\_pca

## Standard deviations (1, .., p=15):  
## [1] 2.45335539 1.67387187 1.41596057 1.07805742 0.97892746 0.74377006  
## [7] 0.56729065 0.55443780 0.48492813 0.44708045 0.41914843 0.35803646  
## [13] 0.26332811 0.24180109 0.06792764  
##   
## Rotation (n x k) = (15 x 15):  
## PC1 PC2 PC3 PC4 PC5  
## M -0.30371194 0.06280357 0.1724199946 -0.02035537 -0.35832737  
## So -0.33088129 -0.15837219 0.0155433104 0.29247181 -0.12061130  
## Ed 0.33962148 0.21461152 0.0677396249 0.07974375 -0.02442839  
## Po1 0.30863412 -0.26981761 0.0506458161 0.33325059 -0.23527680  
## Po2 0.31099285 -0.26396300 0.0530651173 0.35192809 -0.20473383  
## LF 0.17617757 0.31943042 0.2715301768 -0.14326529 -0.39407588  
## M.F 0.11638221 0.39434428 -0.2031621598 0.01048029 -0.57877443  
## Pop 0.11307836 -0.46723456 0.0770210971 -0.03210513 -0.08317034  
## NW -0.29358647 -0.22801119 0.0788156621 0.23925971 -0.36079387  
## U1 0.04050137 0.00807439 -0.6590290980 -0.18279096 -0.13136873  
## U2 0.01812228 -0.27971336 -0.5785006293 -0.06889312 -0.13499487  
## Wealth 0.37970331 -0.07718862 0.0100647664 0.11781752 0.01167683  
## Ineq -0.36579778 -0.02752240 -0.0002944563 -0.08066612 -0.21672823  
## Prob -0.25888661 0.15831708 -0.1176726436 0.49303389 0.16562829  
## Time -0.02062867 -0.38014836 0.2235664632 -0.54059002 -0.14764767  
## PC6 PC7 PC8 PC9 PC10  
## M -0.449132706 -0.15707378 -0.55367691 0.15474793 -0.01443093  
## So -0.100500743 0.19649727 0.22734157 -0.65599872 0.06141452  
## Ed -0.008571367 -0.23943629 -0.14644678 -0.44326978 0.51887452  
## Po1 -0.095776709 0.08011735 0.04613156 0.19425472 -0.14320978  
## Po2 -0.119524780 0.09518288 0.03168720 0.19512072 -0.05929780  
## LF 0.504234275 -0.15931612 0.25513777 0.14393498 0.03077073  
## M.F -0.074501901 0.15548197 -0.05507254 -0.24378252 -0.35323357  
## Pop 0.547098563 0.09046187 -0.59078221 -0.20244830 -0.03970718  
## NW 0.051219538 -0.31154195 0.20432828 0.18984178 0.49201966  
## U1 0.017385981 -0.17354115 -0.20206312 0.02069349 0.22765278  
## U2 0.048155286 -0.07526787 0.24369650 0.05576010 -0.04750100  
## Wealth -0.154683104 -0.14859424 0.08630649 -0.23196695 -0.11219383  
## Ineq 0.272027031 0.37483032 0.07184018 -0.02494384 -0.01390576  
## Prob 0.283535996 -0.56159383 -0.08598908 -0.05306898 -0.42530006  
## Time -0.148203050 -0.44199877 0.19507812 -0.23551363 -0.29264326  
## PC11 PC12 PC13 PC14 PC15  
## M 0.39446657 0.16580189 -0.05142365 0.04901705 0.0051398012  
## So 0.23397868 -0.05753357 -0.29368483 -0.29364512 0.0084369230  
## Ed -0.11821954 0.47786536 0.19441949 0.03964277 -0.0280052040  
## Po1 -0.13042001 0.22611207 -0.18592255 -0.09490151 -0.6894155129  
## Po2 -0.13885912 0.19088461 -0.13454940 -0.08259642 0.7200270100  
## LF 0.38532827 0.02705134 -0.27742957 -0.15385625 0.0336823193  
## M.F -0.28029732 -0.23925913 0.31624667 -0.04125321 0.0097922075  
## Pop 0.05849643 -0.18350385 0.12651689 -0.05326383 0.0001496323  
## NW -0.20695666 -0.36671707 0.22901695 0.13227774 -0.0370783671  
## U1 -0.17857891 -0.09314897 -0.59039450 -0.02335942 0.0111359325  
## U2 0.47021842 0.28440496 0.43292853 -0.03985736 0.0073618948  
## Wealth 0.31955631 -0.32172821 -0.14077972 0.70031840 -0.0025685109  
## Ineq -0.18278697 0.43762828 -0.12181090 0.59279037 0.0177570357  
## Prob -0.08978385 0.15567100 -0.03547596 0.04761011 0.0293376260  
## Time -0.26363121 0.13536989 -0.05738113 -0.04488401 0.0376754405

The prcomp function returns us 15 principal components for our 15 predictor variables. If we keep all 15, then we are not accomplishing our goal of compressing the data while still explaining the variance.

Let’s plot out the cumulative variance explained by adding an additional principal component. To set a goal before we run this, let’s say we want the principal components to account for at least 90% of the variance in our data.

plot(  
 cumsum(uscrime\_pca$sdev^2/sum(uscrime\_pca$sdev^2)),   
 ylab = "% of Variance Explained",   
 xlab = "# of Components"  
)



Eyeballing the cumulative explained variance plot shows us that we can use the first **6** components to explain 90% of the variability in our data.

keep\_pcs <- 6  
  
# uscrime\_c for compressed  
uscrime\_c <- uscrime\_pca$x[,1:keep\_pcs] %>%   
 cbind(uscrime\_t)  
  
# train model on compressed data  
crime\_lm\_pc <- lm(Crime ~ ., data = uscrime\_c)

# PCA Model  
summary(crime\_lm\_pc)

##   
## Call:  
## lm(formula = Crime ~ ., data = uscrime\_c)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -377.15 -172.23 25.81 132.10 480.38   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 905.09 35.35 25.604 < 2e-16 \*\*\*  
## PC1 65.22 14.56 4.478 6.14e-05 \*\*\*  
## PC2 -70.08 21.35 -3.283 0.00214 \*\*   
## PC3 25.19 25.23 0.998 0.32409   
## PC4 69.45 33.14 2.095 0.04252 \*   
## PC5 -229.04 36.50 -6.275 1.94e-07 \*\*\*  
## PC6 -60.21 48.04 -1.253 0.21734   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 242.3 on 40 degrees of freedom  
## Multiple R-squared: 0.6586, Adjusted R-squared: 0.6074   
## F-statistic: 12.86 on 6 and 40 DF, p-value: 4.869e-08

# Old Model  
summary(crime\_lm6)

##   
## Call:  
## lm(formula = Crime ~ Po1 + M.F + Wealth, data = uscrime)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -581.83 -169.25 17.68 158.53 563.19   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -2.578e+03 1.319e+03 -1.954 0.0572 .   
## Po1 1.255e+02 2.179e+01 5.759 8.18e-07 \*\*\*  
## M.F 3.234e+01 1.378e+01 2.347 0.0236 \*   
## Wealth -1.451e-01 6.819e-02 -2.128 0.0391 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 266.6 on 43 degrees of freedom  
## Multiple R-squared: 0.556, Adjusted R-squared: 0.525   
## F-statistic: 17.95 on 3 and 43 DF, p-value: 1.06e-07

The PCA model outperformed the original model across all summary statistics.

The features were more significant.  
The standard error decreased.  
The p-value decreased.

These are all signs that the PCA model is a more accurate model.

However, we lost some interperability with all the scaling, centering, and compressing. Let’s translate the data back to it’s source format.

# ct for Compressed translation  
# Step 1 - Rotate the data back to their original number lines.   
# This changes it from Principal Components to the original column names, like "Wealth" and "Po1".  
uscrime\_ct <- uscrime\_pca$x[,1:keep\_pcs] %\*% t(uscrime\_pca$rotation[,1:keep\_pcs])  
  
# Step 2 - Put the data back on it's original scale.  
uscrime\_ct <- scale(  
 uscrime\_ct,   
 center = FALSE,  
 scale = 1 / uscrime\_pca$scale  
)  
  
# Step 3 - Recenter the data.  
uscrime\_ct <- scale(  
 uscrime\_ct,   
 center = -1 \* uscrime\_pca$center,  
 scale = FALSE  
)  
  
# Step 4 - Cast the data back to a data frame.  
uscrime\_ct <- as.data.frame(uscrime\_ct)  
  
glimpse(uscrime\_ct)

## Observations: 47  
## Variables: 15  
## $ M <dbl> 14.91590, 14.31160, 14.73283, 12.55880, 13.72291, 12.57...  
## $ So <dbl> 1.156706603, 0.186983875, 0.965209619, 0.043318923, 0.0...  
## $ Ed <dbl> 8.676431, 11.198136, 9.028447, 11.491545, 11.745811, 11...  
## $ Po1 <dbl> 5.896157, 9.928460, 4.185161, 15.439532, 10.106595, 11....  
## $ Po2 <dbl> 5.539784, 9.320017, 3.956494, 14.368242, 9.644777, 11.5...  
## $ LF <dbl> 0.5063873, 0.5849868, 0.5348132, 0.5965743, 0.5926502, ...  
## $ M.F <dbl> 96.11987, 101.64875, 96.34515, 99.14320, 99.75173, 96.4...  
## $ Pop <dbl> 40.120109, 21.137555, 25.431581, 119.568007, 15.391495,...  
## $ NW <dbl> 26.05723824, 8.78269589, 21.06320301, 12.18950317, 3.88...  
## $ U1 <dbl> 0.10329039, 0.09991706, 0.09456736, 0.09804812, 0.07922...  
## $ U2 <dbl> 4.105854, 3.396407, 3.395429, 4.148868, 2.424254, 3.004...  
## $ Wealth <dbl> 3817.021, 5696.875, 3642.237, 6769.108, 5981.928, 6606....  
## $ Ineq <dbl> 25.71936, 18.02267, 25.65144, 16.17669, 15.87332, 12.86...  
## $ Prob <dbl> 0.08058657, 0.03365183, 0.08455710, 0.01742998, 0.04330...  
## $ Time <dbl> 25.43133, 26.67956, 23.10526, 32.82974, 22.38854, 22.51...

## 10.1

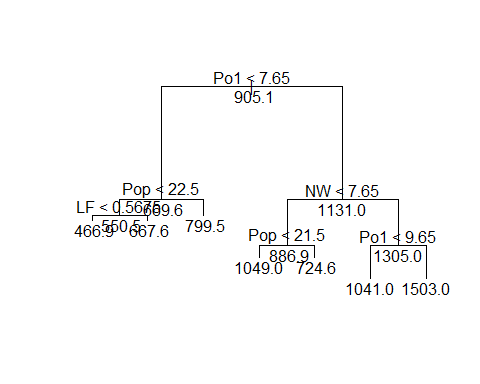
**Using the same crime data set uscrime.txt as in Question 8.2 and 9.1, find the best model you can using**

1. **a regression tree model**
2. **a random forest model**

**In R, you can use the tree package or the rpart package, and the randomForest package. For each model, describe one or two qualitative takeaways you get from analyzing the results (i.e., don’t just stop when you have a good model, but interpret it too).**

We will start with building out a single tree.

crime\_tree <- tree(  
 Crime ~ .,  
 data = uscrime  
)  
plot(crime\_tree)  
text(crime\_tree, all = TRUE, pretty = 0)



This plot shows us the decision points in the tree.

Now that we understand how to interpret a single tree, let’s grow a full forest.

set.seed(1234) # Allows you to replicate results for 'Random' Forest  
crime\_rf <- randomForest(  
 Crime ~ ., # Formula  
 data = uscrime, # Source data set  
 ntree = 250, # How many trees we grow  
 nodesize = 5, # How tall we allow the tree to grow  
 replace = TRUE, # If we allow the sampling of the same data point for a single tree  
 importance = TRUE, # Performance details  
 localImp = TRUE # Performance details  
)

In our model we created 250 different trees. These trees were all brought back together in an ensemble. A side effect of the ensemble is that we lose some interpretability of the model. We cannot point to a particular tree and a particular branch to say “this is why the prediction provided value *x*.”

There model does return a few stats to help interpret its peformance.

# Mean-Squared Error of the model  
mean(crime\_rf$mse)

## [1] 85340.83

# R Squared score - % of variance explained by model  
mean(crime\_rf$rsq)

## [1] 0.4170817

crime\_rf$importance

## %IncMSE IncNodePurity  
## M 5153.3803 216561.21  
## So 860.1521 21739.96  
## Ed 5168.6408 202858.09  
## Po1 38781.8025 1201041.73  
## Po2 27545.4174 1127990.26  
## LF 2837.6166 332583.84  
## M.F -782.7835 236231.68  
## Pop -203.0019 360270.53  
## NW 17816.1756 561507.53  
## U1 147.3488 114366.08  
## U2 1610.1519 155364.24  
## Wealth 7377.2022 647937.07  
## Ineq 992.6367 245142.48  
## Prob 15884.0223 736907.16  
## Time 2573.0537 225581.90

# Display the standard errors  
crime\_rf$importanceSD

## M So Ed Po1 Po2 LF M.F   
## 1403.0765 369.3548 1789.6333 4232.7521 3990.6604 1769.7942 1052.2382   
## Pop NW U1 U2 Wealth Ineq Prob   
## 1050.4038 2644.3263 790.1807 1188.9149 2197.6812 1302.2722 2577.8764   
## Time   
## 994.2672

Wrapping up, between the 4 models we have trained *(Linear Regression, PCA Linear Regression, Tree Regression, and Random Forest Regression)* the PCA Linear Regression model performed the best.

However, I may select the Tree Regression to brief to leadership. The decision tree structure is easy for non-analytical teammates to follow. It also becomes actionable. The team can strategically target a node on the tree.

## 10.2

**Describe a situation or problem from your job, everyday life, current events, etc., for which a logistic regression model would be appropriate. List some (up to 5) predictors that you might use.**

There is a site called [Numerai](https://numer.ai) that blends the ideas of high frequency traders and Kaggle data competitions. Each week a new round opens up and thousands of data scientists/quants train models to submit. The source data is transformed in such a way that the modeler cannot know what signals or stocks they are modeling to trade.

Numerai gets lots of models that they can create an ensemable out of to buy and trade these stocks. Then you, as the modeler, get a kick back for how well your model performs and the uniqueness of the model.

Logistic regression works well in this scenario because they want a binary decision - Buy or Sell. However, they ask that you provide the results in a confidence percentage instead. This way they can set their own threshold for when to trust your model.

Due to Numerai cleaning, prepping, and obscuring the source the feature names are not so exciting. The data set includes predictors like:

* Feature\_1
* Feature\_2
* …
* Feature\_50

Either way, it’s a fun clean place to practice your modelling skills and possibly earn some change on the side!

## 10.3

1. **Using the GermanCredit data set germancredit.txt from …, use logisitc regression to find a good predictive model for whether credit applicants are good credit risks or not. Show your model (factors used and their coefficients), the software output, and the quality of fit. You can use the glm function in R.**
2. **Because the model gives a result between 0 and 1, it requires a threshold probablity to separate between “good” and “bad” answers. In this data set, they estimate that incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad. Determine a good threshold probability based on your model.**

gercredit <- read.table("data/germancredit.txt")  
glimpse(gercredit)

## Observations: 1,000  
## Variables: 21  
## $ V1 <fct> A11, A12, A14, A11, A11, A14, A14, A12, A14, A12, A12, A11...  
## $ V2 <int> 6, 48, 12, 42, 24, 36, 24, 36, 12, 30, 12, 48, 12, 24, 15,...  
## $ V3 <fct> A34, A32, A34, A32, A33, A32, A32, A32, A32, A34, A32, A32...  
## $ V4 <fct> A43, A43, A46, A42, A40, A46, A42, A41, A43, A40, A40, A49...  
## $ V5 <int> 1169, 5951, 2096, 7882, 4870, 9055, 2835, 6948, 3059, 5234...  
## $ V6 <fct> A65, A61, A61, A61, A61, A65, A63, A61, A64, A61, A61, A61...  
## $ V7 <fct> A75, A73, A74, A74, A73, A73, A75, A73, A74, A71, A72, A72...  
## $ V8 <int> 4, 2, 2, 2, 3, 2, 3, 2, 2, 4, 3, 3, 1, 4, 2, 4, 4, 2, 4, 3...  
## $ V9 <fct> A93, A92, A93, A93, A93, A93, A93, A93, A91, A94, A92, A92...  
## $ V10 <fct> A101, A101, A101, A103, A101, A101, A101, A101, A101, A101...  
## $ V11 <int> 4, 2, 3, 4, 4, 4, 4, 2, 4, 2, 1, 4, 1, 4, 4, 2, 4, 3, 2, 2...  
## $ V12 <fct> A121, A121, A121, A122, A124, A124, A122, A123, A121, A123...  
## $ V13 <int> 67, 22, 49, 45, 53, 35, 53, 35, 61, 28, 25, 24, 22, 60, 28...  
## $ V14 <fct> A143, A143, A143, A143, A143, A143, A143, A143, A143, A143...  
## $ V15 <fct> A152, A152, A152, A153, A153, A153, A152, A151, A152, A152...  
## $ V16 <int> 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2, 3, 1, 1...  
## $ V17 <fct> A173, A173, A172, A173, A173, A172, A173, A174, A172, A174...  
## $ V18 <int> 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2...  
## $ V19 <fct> A192, A191, A191, A191, A191, A192, A191, A192, A191, A191...  
## $ V20 <fct> A201, A201, A201, A201, A201, A201, A201, A201, A201, A201...  
## $ V21 <int> 1, 2, 1, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1, 2, 1, 2, 1, 1, 2, 1...

V21 is our target variable, but the logisitic regression models need the target variable to be 0 or 1, not 1 or 2.

In the dataset documentation 1 is considered a good loan and 2 is considered a bad loan. Let’s recode 2 to 0 for our model.

gercredit <- mutate(gercredit, V21 = ifelse(V21 == 2, 0, 1))

credit\_glm\_all <- glm(  
 V21 ~ .,   
 family = binomial(link = "logit"),   
 data = gercredit  
)  
summary(credit\_glm\_all)

##   
## Call:  
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = gercredit)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.6116 -0.7095 0.3752 0.6994 2.3410   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.005e-01 1.084e+00 -0.369 0.711869   
## V1A12 3.749e-01 2.179e-01 1.720 0.085400 .   
## V1A13 9.657e-01 3.692e-01 2.616 0.008905 \*\*   
## V1A14 1.712e+00 2.322e-01 7.373 1.66e-13 \*\*\*  
## V2 -2.786e-02 9.296e-03 -2.997 0.002724 \*\*   
## V3A31 -1.434e-01 5.489e-01 -0.261 0.793921   
## V3A32 5.861e-01 4.305e-01 1.362 0.173348   
## V3A33 8.532e-01 4.717e-01 1.809 0.070470 .   
## V3A34 1.436e+00 4.399e-01 3.264 0.001099 \*\*   
## V4A41 1.666e+00 3.743e-01 4.452 8.51e-06 \*\*\*  
## V4A410 1.489e+00 7.764e-01 1.918 0.055163 .   
## V4A42 7.916e-01 2.610e-01 3.033 0.002421 \*\*   
## V4A43 8.916e-01 2.471e-01 3.609 0.000308 \*\*\*  
## V4A44 5.228e-01 7.623e-01 0.686 0.492831   
## V4A45 2.164e-01 5.500e-01 0.393 0.694000   
## V4A46 -3.628e-02 3.965e-01 -0.092 0.927082   
## V4A48 2.059e+00 1.212e+00 1.699 0.089297 .   
## V4A49 7.401e-01 3.339e-01 2.216 0.026668 \*   
## V5 -1.283e-04 4.444e-05 -2.887 0.003894 \*\*   
## V6A62 3.577e-01 2.861e-01 1.250 0.211130   
## V6A63 3.761e-01 4.011e-01 0.938 0.348476   
## V6A64 1.339e+00 5.249e-01 2.551 0.010729 \*   
## V6A65 9.467e-01 2.625e-01 3.607 0.000310 \*\*\*  
## V7A72 6.691e-02 4.270e-01 0.157 0.875475   
## V7A73 1.828e-01 4.105e-01 0.445 0.656049   
## V7A74 8.310e-01 4.455e-01 1.866 0.062110 .   
## V7A75 2.766e-01 4.134e-01 0.669 0.503410   
## V8 -3.301e-01 8.828e-02 -3.739 0.000185 \*\*\*  
## V9A92 2.755e-01 3.865e-01 0.713 0.476040   
## V9A93 8.161e-01 3.799e-01 2.148 0.031718 \*   
## V9A94 3.671e-01 4.537e-01 0.809 0.418448   
## V10A102 -4.360e-01 4.101e-01 -1.063 0.287700   
## V10A103 9.786e-01 4.243e-01 2.307 0.021072 \*   
## V11 -4.776e-03 8.641e-02 -0.055 0.955920   
## V12A122 -2.814e-01 2.534e-01 -1.111 0.266630   
## V12A123 -1.945e-01 2.360e-01 -0.824 0.409743   
## V12A124 -7.304e-01 4.245e-01 -1.721 0.085308 .   
## V13 1.454e-02 9.222e-03 1.576 0.114982   
## V14A142 1.232e-01 4.119e-01 0.299 0.764878   
## V14A143 6.463e-01 2.391e-01 2.703 0.006871 \*\*   
## V15A152 4.436e-01 2.347e-01 1.890 0.058715 .   
## V15A153 6.839e-01 4.770e-01 1.434 0.151657   
## V16 -2.721e-01 1.895e-01 -1.436 0.151109   
## V17A172 -5.361e-01 6.796e-01 -0.789 0.430160   
## V17A173 -5.547e-01 6.549e-01 -0.847 0.397015   
## V17A174 -4.795e-01 6.623e-01 -0.724 0.469086   
## V18 -2.647e-01 2.492e-01 -1.062 0.288249   
## V19A192 3.000e-01 2.013e-01 1.491 0.136060   
## V20A202 1.392e+00 6.258e-01 2.225 0.026095 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1221.73 on 999 degrees of freedom  
## Residual deviance: 895.82 on 951 degrees of freedom  
## AIC: 993.82  
##   
## Number of Fisher Scoring iterations: 5

In the summary of the model, we can see the coefficients used with all the features, their standard error, and the significance that they contribute to the model.

Looking closely at the features you may notice that the categorical features have been expanded into each of their options. V1 became V1A12, V1A13, and V1A14. Whereas V2 remained V2 because it is a numerical feature.

This idea of breaking a categorical feature into one column for each category and labeling it as a binary *yes* or *no* is known as **one-hot encoding**.

The summary also provides the AIC score, which is a measure of the models complexity. In this case, the AIC is high, which is to be expected since we threw the EVERY feature at it.

Now that we have our model trained, we can feed the data through it to get predictions.

credit\_p\_all <- predict(credit\_glm\_all, gercredit, "response")  
head(credit\_p\_all)

## 1 2 3 4 5 6   
## 0.9647683 0.3677376 0.9719376 0.7481979 0.2479989 0.7376644

The prediction returns the model’s confidence in a positive classification. With these percentages we can set a threshold for when we will accept a positive classification.

Since false positives *(Type 1 Error)* are 5x as costly as a false negative *(Type 2 Error)*, we will want to set a higher threshold.

A confusion matrix is a good tool to check how many type 1 vs. type 2 errors you are making.

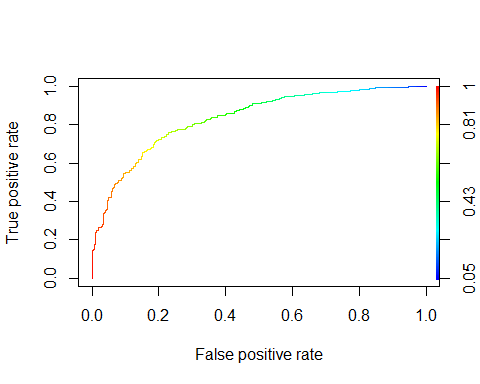
# You must be 70% sure this is a good idea before we will accept it!  
threshold <- 0.7  
  
# This was bit of an odd step. I was not expecting to have to convert these to a factor.  
# I also needed to install the `e1071` package before the confusion matrix would work.  
pred\_factor <- factor(as.numeric(credit\_p\_all > threshold))  
target\_factor <- factor(gercredit$V21)  
  
confusionMatrix(  
 data = pred\_factor,  
 reference = target\_factor  
)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 232 176  
## 1 68 524  
##   
## Accuracy : 0.756   
## 95% CI : (0.7281, 0.7823)  
## No Information Rate : 0.7   
## P-Value [Acc > NIR] : 4.795e-05   
##   
## Kappa : 0.4732   
##   
## Mcnemar's Test P-Value : 7.386e-12   
##   
## Sensitivity : 0.7733   
## Specificity : 0.7486   
## Pos Pred Value : 0.5686   
## Neg Pred Value : 0.8851   
## Prevalence : 0.3000   
## Detection Rate : 0.2320   
## Detection Prevalence : 0.4080   
## Balanced Accuracy : 0.7610   
##   
## 'Positive' Class : 0   
##

The confusion matrix just gives us a snapshot of setting a single threshold. What if we want to look at all the possible thresholds and their true positive / false positive tradeoff?

Well for that, we can use a Reciever Operating Characteristic or ROC curve.

credit\_pred\_all <- prediction(credit\_p\_all, gercredit$V21)  
credit\_perf\_all <- performance(credit\_pred\_all, "tpr", "fpr")  
plot(credit\_perf\_all, colorize = TRUE)



I found the color very helpful on this plot for highlighting the ideal threshold.

At the extremes, a threshold of 1, meaning the model needs to be **100%** certain that this is really a good idea, we get 0 positives.  
A threshold of 0, meaning we let every data point come through as a positive “good idea” we are right about 100% of the true positives, but we also have 100% of the true negatives.

The sweet spot for our case looks to be around the orangish-yellow area. ~.8 threshold. This will let us capture 60% of the proverbial good eggs and only let ~15% of bad eggs through.

After talking with management, they may still want to increase the threshold. You will be turning away a lot of good customers, but you won’t have as much risk with a bad one.

*I could go on and refine the model to get more Area Under the Curve (AUC), but I believe I’ve demonstrated my ability to refine a model in the questions above and that I have met the requirements for this question. The same tactics to optimize a regression model apply to a classification model.*