FinTech HW6

Alyson Brown

2/21/2019

Part 1: Credit Modeling

Q1

```
# Q1
load("/Users/alysonbrown/Desktop/training_10.Rda")
load("/Users/alysonbrown/Desktop/testing_11.Rda")

# Feature handling
training_10$STATE <- as.factor(training_10$STATE)
training_10$OCC_STAT <- as.factor(training_10$OCC_STAT)

testing_11$STATE <- as.factor(testing_11$STATE)
testing_11$OCC_STAT <- as.factor(testing_11$OCC_STAT)

names(training_10)
```

```
##  [1] "LOAN_ID"        "ORIG_CHN"       "ORIG_RT"        "ORIG_AMT"
##  [5] "ORIG_TRM"       "ORIG_DTE"       "FRST_DTE"       "OLTV"
##  [9] "OCLTV"          "NUM_BO"         "DTI"            "CSCORE_B"
## [13] "FTHB_FLG"       "PURPOSE"        "PROP_TYP"       "NUM_UNIT"
## [17] "OCC_STAT"       "STATE"          "ZIP_3"          "MI_PCT"
## [21] "CSCORE_C"       "MI_TYPE"        "RELOCATION_FLG" "CSCORE_MN"
## [25] "ORIG_VAL"       "Delq.90.days"
```

Q2

Delinquency by Various Characteristics

```
# Q2 Bullet 1
set1 = select(training_10, CSCORE_B, OLTV, OCLTV, DTI)
describeBy(set1, training_10$Delq.90.days)
```

```
##
##  Descriptive statistics by group
## group: 0
##          vars       n   mean    sd median trimmed   mad min max range
## CSCORE_B    1 1658597 768.06 39.39    779  773.11 32.62 431 850   419
## OLTV        2 1659977  65.81 17.19     71   67.52 13.34   1  97    96
## OCLTV       3 1659977  67.02 17.11     72   68.58 11.86   1 148   147
## DTI         4 1656123  31.22  9.96     32   31.41 11.86   1  64    63
##           skew kurtosis   se
## CSCORE_B -1.12     0.83 0.03
## OLTV     -0.80    -0.02 0.01
## OCLTV    -0.83     0.08 0.01
## DTI      -0.12    -0.67 0.01
## ------------------------------------------------------
## group: 1
##          vars    n   mean    sd median trimmed   mad min max range  skew
## CSCORE_B    1 7567 724.22 50.41    724  725.32 57.82 457 825   368 -0.20
## OLTV        2 7593  74.01 12.91     78   75.18  5.93  10  97    87 -1.18
## OCLTV       3 7593  74.97 12.95     79   76.18  8.90  10 105    95 -1.15
## DTI         4 7539  36.26  8.66     38   36.81  8.90   2  64    62 -0.54
##          kurtosis   se
## CSCORE_B    -0.52 0.58
## OLTV         2.17 0.15
## OCLTV        2.10 0.15
## DTI          0.17 0.10
```

The delinquent group had lower average credit scores (724 vs. 768), higher average OLTVs (74 vs. 66) & average OCLTVs (75 vs. 67), and higher average DTIs (36 vs. 31).

Since the difference of all variables based on the delinquency are obvious, we can guess that credit score, loan-to-value ratio, combined loan-to-value ratio, and deebt-to-income ratio are correlated with high delinquency.

Delinquency by State

```
#summarize delinquency by states
mytable <- table(training_10$STATE, training_10$Delq.90.days)
z = prop.table(mytable, 1) # row percentages
```

We can see that VI has the highest delinquency which is 0.031128405 and GU has lowest delinquency which is 0. It is interesing to note that the two “states” with significantly higher delinquency than the others are Puerto Rico and the Virgin Islands.

```
library(usmap)
library(ggplot2)
```

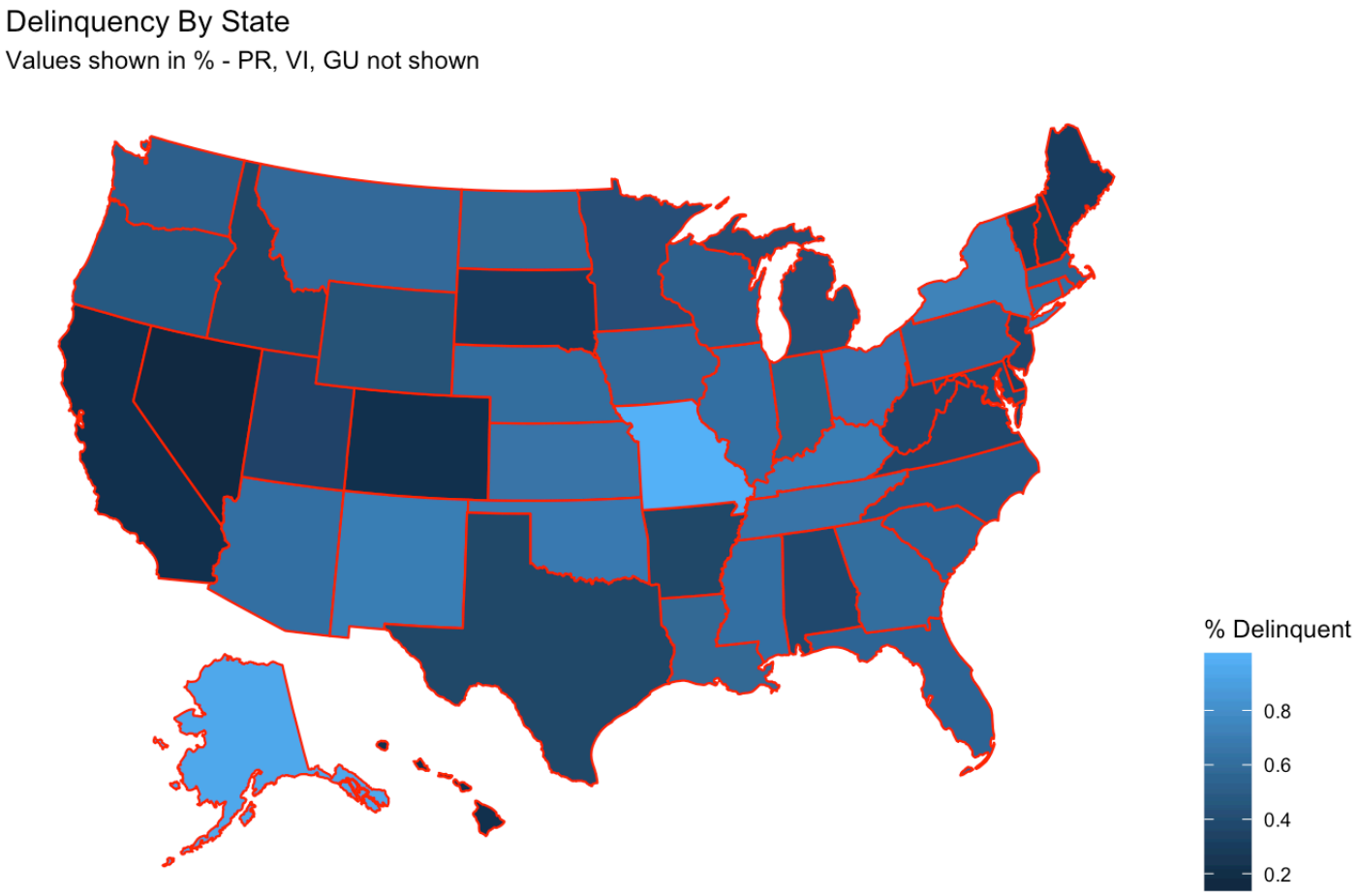
```
##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```
z = z[-12,]
z = z[-40,]
z = z[-47,] #12, 41, 49

statepop$del = z[,2]*100

plot_usmap(data = statepop, values = "del", lines = "red") +
  scale_fill_continuous(name = "% Delinquent", label = scales::comma) +
  theme(legend.position = "right")+
  labs(title = "Delinquency By State", subtitle = "Values shown in % - PR, VI, GU not
shown") +
  theme(legend.position = "right")
```



Delinquency by Month

```
#summarize delinquency by month
mytable2 <- table(training_10$ORIG_DTE, training_10$Delq.90.days)
prop.table(mytable2, 1) # row percentages
```

```
##
##                      0           1
##   01/2010 0.993133559 0.006866441
##   02/2010 0.993964817 0.006035183
##   03/2010 0.994016040 0.005983960
##   04/2010 0.994461432 0.005538568
##   05/2010 0.993892045 0.006107955
##   06/2010 0.994898400 0.005101600
##   07/2010 0.995564448 0.004435552
##   08/2010 0.996195467 0.003804533
##   09/2010 0.996504034 0.003495966
##   10/2010 0.996573909 0.003426091
##   11/2010 0.996706704 0.003293296
##   12/2010 0.996450027 0.003549973
```

We can see that January, May and February are the first months with highest delinquent. January is the highest month.

Baseline Logit Model:

```
#Q3
fit <- glm(Delq.90.days~ ORIG_RT, data=training_10, family = binomial)
summary(fit)
```

```
##
## Call:
## glm(formula = Delq.90.days ~ ORIG_RT, family = binomial, data = training_10)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8679  -0.1037  -0.0777  -0.0641   3.9324
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -12.7505     0.1155 -110.44   <2e-16 ***
## ORIG_RT       1.5443     0.0232   66.57   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 97033  on 1667569  degrees of freedom
## Residual deviance: 92647  on 1667568  degrees of freedom
##   (38 observations deleted due to missingness)
## AIC: 92651
##
## Number of Fisher Scoring iterations: 8
```

```
fit_preds = predict(fit, newdata = testing_11)
```

Better Logit Model:

CSCORE B, OLT V , OCLT V , DTI, OCC STAT, and STATE, but without the interest rate (ORIG RT)

```
#Q4
set2 = select(training_10, CSCORE_B, OLTV, OCLTV, OCC_STAT, DTI, STATE, Delq.90.days)
set3 = select(testing_11, CSCORE_B, OLTV, OCLTV, OCC_STAT, DTI, STATE, Delq.90.days)
# Remove NAs
training_set = set2[complete.cases(set2),]
testing_set = set3[complete.cases(set3),]

fit2 <- glm(Delq.90.days~ CSCORE_B + OLTV + OCLTV + DTI + OCC_STAT + STATE, data=trai
ning_set, family = binomial)
summary(fit2)
```

```
##
## Call:
## glm(formula = Delq.90.days ~ CSCORE_B + OLTV + OCLTV + DTI +
```

```
##     OCC_STAT + STATE, family = binomial, data = training_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5437  -0.0987  -0.0655  -0.0442   4.1601
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.1460117  0.3347273  12.386  < 2e-16 ***
## CSCORE_B    -0.0181113  0.0002387 -75.874  < 2e-16 ***
## OLTV         0.0260857  0.0029745   8.770  < 2e-16 ***
## OCLTV        0.0069166  0.0030011   2.305 0.021185 *
## DTI          0.0400914  0.0013627  29.421  < 2e-16 ***
## OCC_STATP   -0.0902930  0.0455615  -1.982 0.047504 *
## OCC_STATS   -0.6208677  0.0824305  -7.532 5.00e-14 ***
## STATEAL      1.0352229  0.2789658   3.711 0.000206 ***
## STATEAR      0.6561132  0.2937910   2.233 0.025531 *
## STATEAZ      0.2325658  0.2855356   0.814 0.415364
## STATECA     -0.2079515  0.2718923  -0.765 0.444373
## STATECO     -0.2113699  0.2889948  -0.731 0.464537
## STATECT      0.8113254  0.2819109   2.878 0.004003 **
## STATEDC      0.1951585  0.3627823   0.538 0.590612
## STATEDE      0.8796665  0.3137224   2.804 0.005048 **
## STATEFL      0.4842326  0.2752499   1.759 0.078536 .
## STATEGA      0.6631809  0.2774191   2.391 0.016824 *
## STATEGU     -8.0362494 42.1007311  -0.191 0.848618
## STATEHI     -0.1807292  0.3580578  -0.505 0.613735
## STATEIA      0.1524484  0.2914025   0.523 0.600867
## STATEID      0.6609090  0.3017415   2.190 0.028501 *
## STATEIL      0.6095328  0.2726899   2.235 0.025400 *
## STATEIN      0.5919131  0.2797884   2.116 0.034381 *
## STATEKS      0.7309895  0.2914064   2.508 0.012125 *
## STATEKY      0.8161248  0.2861949   2.852 0.004349 **
## STATELA      0.3543145  0.2851804   1.242 0.214081
## STATEMA      0.1817578  0.2785838   0.652 0.514121
## STATEMD      0.3960342  0.2817436   1.406 0.159827
## STATEME      0.6026334  0.3260235   1.848 0.064539 .
## STATEMI      0.2369716  0.2825090   0.839 0.401576
## STATEMN      0.4238177  0.2827591   1.499 0.133909
## STATEMO      0.7933983  0.2765945   2.868 0.004125 **
## STATEMS      0.9207812  0.2891351   3.185 0.001450 **
## STATEMT      0.6737903  0.3063229   2.200 0.027835 *
## STATENC      0.6857276  0.2763620   2.481 0.013092 *
## STATEND     -0.6926222  0.4895199  -1.415 0.157098
## STATENE      0.0687911  0.3142482   0.219 0.826722
## STATENH      0.3086250  0.3207301   0.962 0.335920
## STATENJ      0.8581296  0.2737818   3.134 0.001722 **
```

```
## STATENM      0.8563985  0.2942628   2.910 0.003611 **
## STATENV      0.3759995  0.3057811   1.230 0.218833
## STATENY      0.6004197  0.2731771   2.198 0.027955 *
## STATEOH      0.6668364  0.2753447   2.422 0.015443 *
## STATEOK      0.6056605  0.2859806   2.118 0.034189 *
## STATEOR      0.5702905  0.2853496   1.999 0.045655 *
## STATEPA      0.5818489  0.2740621   2.123 0.033749 *
## STATEPR      1.3804033  0.2939447   4.696 2.65e-06 ***
## STATERI      0.7997791  0.3158991   2.532 0.011349 *
## STATESC      0.5252912  0.2850934   1.843 0.065399 .
## STATESD     -0.0277127  0.3581129  -0.077 0.938317
## STATETN      0.6841799  0.2792832   2.450 0.014295 *
## STATETX     -0.1684787  0.2734333  -0.616 0.537789
## STATEUT      0.1161330  0.2970016   0.391 0.695784
## STATEVA      0.1734588  0.2800447   0.619 0.535655
## STATEVI      1.8066462  0.4569321   3.954 7.69e-05 ***
## STATEVT      0.2544173  0.3629626   0.701 0.483337
## STATEWA      0.6052436  0.2760875   2.192 0.028364 *
## STATEWI      0.4140228  0.2760203   1.500 0.133622
## STATEWV      0.3919315  0.3384355   1.158 0.246836
## STATEWY      0.1744628  0.3541971   0.493 0.622325
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 96155  on 1662316  degrees of freedom
## Residual deviance: 85693  on 1662257  degrees of freedom
## AIC: 85813
##
## Number of Fisher Scoring iterations: 12
```

```
fit2_preds = predict(fit2, newdata = testing_11)
```

Random Forest Model (using h2o):

```
library(h2o)

## Create an H2O cloud
h2o.init(nthreads=-1,            ## -1: use all available threads
  max_mem_size = "2G")    ## specify the memory size for the H2O cloud
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         11 hours 35 seconds
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.22.1.1
##     H2O cluster version age:    1 month and 27 days
##     H2O cluster name:           H2O_started_from_R_alysonbrown_adc446
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.42 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         XGBoost, Algos, AutoML, Core V3, Core V4
##     R Version:                  R version 3.5.1 (2018-07-02)
```

```
h2o.removeAll() # Clean slate - just in case the cluster was already running
```

```
## [1] 0
```

```
y='Delq.90.days'
x=c('CSCORE_B','OLTV','OCLTV','DTI','OCC_STAT','STATE')

df <- as.h2o(training_10)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
test_set <- as.h2o(testing_11)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
splits = h2o.splitFrame(df, 0.8)
train = splits[[1]]
valid = splits[[2]]

## Random Forest Model
rf1 <- h2o.randomForest(training_frame = train, validation_frame = valid, x=x, y=y, m
odel_id = "rf_1", ntrees = 200, stopping_rounds = 2, score_each_iteration = T, seed =
1000000)
```

```
##
  |
  |                                                                      |   0%
  |
  |=                                                                     |   1%
  |
  |=                                                                     |   2%
  |
  |==                                                                    |   2%
  |
  |==                                                                    |   3%
  |
  |==                                                                    |   4%
  |
  |===                                                                   |   4%
  |
  |===                                                                   |   5%
  |
  |====                                                                  |   6%
  |
  |=====                                                                 |   7%
  |
  |=====                                                                 |   8%
  |
  |======                                                                |   8%
  |
  |======                                                                |   9%
  |
  |======                                                                |  10%
  |
```

```
  |=======                                                               |  10%
  |
  |=======                                                               |  11%
  |
  |=======                                                               |  12%
  |
  |========                                                              |  12%
  |
  |========                                                              |  13%
  |
  |=========                                                             |  14%
  |
  |==========                                                            |  15%
  |
  |==========                                                            |  16%
  |
  |======================================================================| 100%
```

```
summary(rf1)
```

```
## Model Details:
## ==============
##
## H2ORegressionModel: drf
## Model Key:  rf_1
## Model Summary:
##   number_of_trees number_of_internal_trees model_size_in_bytes min_depth
## 1              32                        32             4134997        20
##   max_depth mean_depth min_leaves max_leaves mean_leaves
## 1        20   20.00000       8395      10234  9189.68750
##
## H2ORegressionMetrics: drf
## ** Reported on training data. **
## ** Metrics reported on Out-Of-Bag training samples **
##
## MSE:  0.004867903
## RMSE:  0.06977036
## MAE:  0.009127136
## RMSLE:  0.04974002
## Mean Residual Deviance :  0.004867903
##
##
## H2ORegressionMetrics: drf
## ** Reported on validation data. **
##
## MSE:  0.004669142
```

```
## RMSE:  0.06833112
## MAE:  0.009041454
## RMSLE:  0.04834598
## Mean Residual Deviance :  0.004669142
##
##
##
##
## Scoring History:
##             timestamp   duration number_of_trees training_rmse
## 1 2019-02-24 22:30:12  0.002 sec               0            NA
## 2 2019-02-24 22:30:14  1.827 sec               1       0.08294
## 3 2019-02-24 22:30:16  3.669 sec               2       0.08225
## 4 2019-02-24 22:30:18  5.576 sec               3       0.08049
## 5 2019-02-24 22:30:20  7.401 sec               4       0.07908
##   training_mae training_deviance validation_rmse validation_mae
## 1           NA                NA              NA             NA
## 2      0.00918           0.00688         0.08227        0.00911
## 3      0.00922           0.00676         0.07561        0.00904
## 4      0.00918           0.00648         0.07304        0.00906
## 5      0.00916           0.00625         0.07156        0.00901
##   validation_deviance
## 1                  NA
## 2             0.00677
## 3             0.00572
## 4             0.00533
## 5             0.00512
##
## ---
##              timestamp          duration number_of_trees training_rmse
## 28 2019-02-24 22:31:09        56.485 sec              27       0.07006
## 29 2019-02-24 22:31:11        58.543 sec              28       0.07000
## 30 2019-02-24 22:31:13  1 min  0.610 sec              29       0.06994
## 31 2019-02-24 22:31:15  1 min  2.761 sec              30       0.06988
## 32 2019-02-24 22:31:17  1 min  4.922 sec              31       0.06982
## 33 2019-02-24 22:31:19  1 min  7.080 sec              32       0.06977
##    training_mae training_deviance validation_rmse validation_mae
## 28      0.00913           0.00491         0.06842        0.00905
## 29      0.00913           0.00490         0.06841        0.00905
## 30      0.00913           0.00489         0.06839        0.00905
## 31      0.00913           0.00488         0.06836        0.00904
## 32      0.00913           0.00487         0.06836        0.00904
## 33      0.00913           0.00487         0.06833        0.00904
##    validation_deviance
## 28             0.00468
## 29             0.00468
## 30             0.00468
```

```
## 31             0.00467
## 32             0.00467
## 33             0.00467
##
## Variable Importances: (Extract with `h2o.varimp`)
## =================================================
##
## Variable Importances:
##   variable relative_importance scaled_importance percentage
## 1 CSCORE_B        19240.292969          1.000000   0.315598
## 2    STATE        13852.536133          0.719975   0.227223
## 3      DTI        13085.242188          0.680096   0.214637
## 4    OCLTV         6658.621582          0.346077   0.109221
## 5     OLTV         6643.756836          0.345304   0.108977
## 6 OCC_STAT         1484.032471          0.077131   0.024343
```

```
finalRf_predictions<-h2o.predict(object = rf1, newdata = test_set)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
head(finalRf_predictions)
```

```
##        predict
## 1 1.120157e-01
## 2 5.240607e-05
## 3 0.000000e+00
## 4 1.859491e-03
## 5 7.268715e-03
## 6 1.644737e-04
```

Neural Network Model

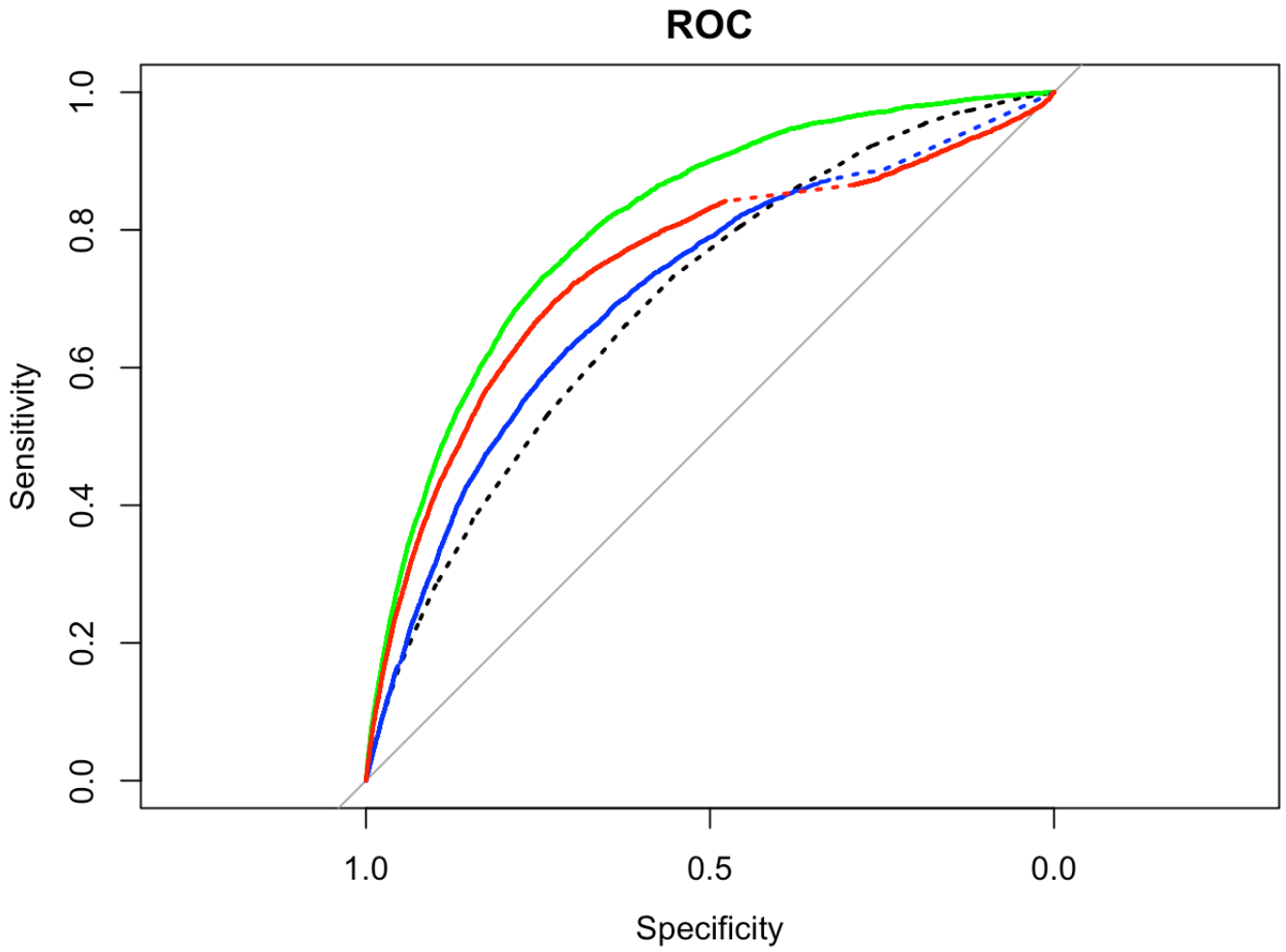
```
h2o.no_progress()  # Disable progress bars for Rmd

train_set <- as.h2o(training_10)
dl_fit1 <- h2o.deeplearning(x = x, y = y, training_frame = train_set,model_id = "dl_f
it1", hidden = c(20,20), seed = 1)
nn_preds <- h2o.predict(object = dl_fit1, newdata = test_set)
```

ROC Curve

```
library(pROC)
testing_11$Delq.90.days <- as.numeric(testing_11$Delq.90.days)

roc_benchmark = roc(testing_11$Delq.90.days, fit_preds)
roc_logistic = roc(testing_11$Delq.90.days, fit2_preds)
roc_rf = roc(testing_11$Delq.90.days, as.matrix(finalRf_predictions))
roc_nn = roc(testing_11$Delq.90.days, as.matrix(nn_preds))
plot(roc_benchmark, col = "black", lty = 3, main = "ROC", xlim = c(1.0,0), ylim = c(0
,1.0))
plot(roc_logistic, col = "green", lty = 3, xlim = c(1.0,0), add = TRUE)
plot(roc_rf, col = "blue", lty = 3, xlim = c(1.0,0), add = TRUE)
plot(roc_nn, col = "red", lty = 3, xlim = c(1.0,0), add = TRUE)
```



In the code, comment on the results of the ROC curve comparison. Which model seems to be the best out of sample? Can you provide an explanation of the observed differences?

Logistic Regression with many features is the best model in this case based on the ROC curve. It is closely followed by the deep learning neural net model. The random forest and baseline logit model both performed relatively poorly. There are many possible reasons that a logit model would produce more accurate results than other models, such as overfitting or in the case of the baseline logit model, not enough information. It's also possible that decision trees were not able to find good splits on the differenct features.

I am thinking of this slide

Another important factor to consider is that the data is highly imbalanced, with about 99% of the data from non-delinquent loans. While this is probably representative of the real world, it is is tougher to accurately classify a loan as delinquent, when there is such a high prior probability (99%) that the loan is not delinquent. (If unfamiliar with prior probabilities, it essentially encapulates the idea that, with no outside knowledge about a loan, there's a 99% chance it's not delinquent simply based on the historical data, and would therefore require very compelling evidence that it strongly resembles a delinquent loan of the past - of which there is little data to begin with.)

Imbalanced data such as this makes classification in any model difficult, but it would especially be a challenge for decision trees, as trees are looking for splits in the features that have allow for grouping a set of loans as delinquent or not - which is tough when so few are actually delinquent relative to the whole population. Logistic regression, however, looks at the entire set of features together, and tries to find the most similar loans in the historical data. It calculates similarity scores, then probabilities, and then chooses some cutoff for the probability for which it classifies a loan as delinquent, meaning that you could adjust such a cutoff to account for the overwhelming prior pobability that a loan is not delinquent - which is another reason that logistic regression (with enough information/features is the best model in this case.)

I believe that the deep learning model was second place was because it would not have the same previously mentioned issues of decision trees/random forests, but would possibly overfit the training data since kernels within the neural net are going over the data many times and constantly updating the beta coefficients for prediction. Such a model could possibly over-complicate the solution and overfit the training data.