Project2_DAT514

Liam McFall, Erin Karnath, Cam Farrugia

4/23/2020

Read in Data and initital data exploration

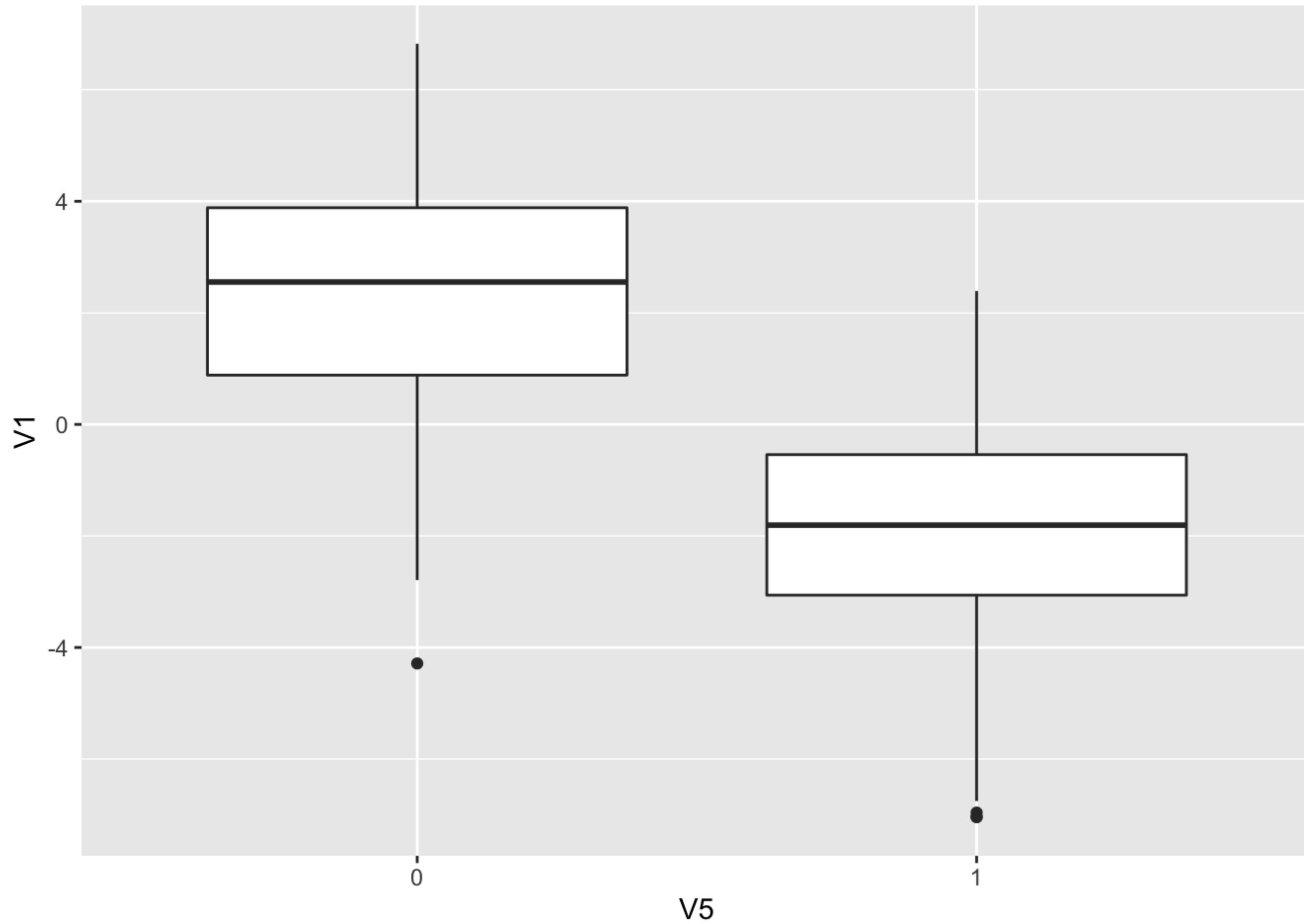
```
project<- read.csv("classif1.txt", header = FALSE)

# Data Exploration

project$V5 <- factor(project$V5)
str(project)
```

```
## 'data.frame':    1372 obs. of  5 variables:
##  $ V1: num  3.622 4.546 3.866 3.457 0.329 ...
##  $ V2: num  8.67 8.17 -2.64 9.52 -4.46 ...
##  $ V3: num  -2.81 -2.46 1.92 -4.01 4.57 ...
##  $ V4: num  -0.447 -1.462 0.106 -3.594 -0.989 ...
##  $ V5: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

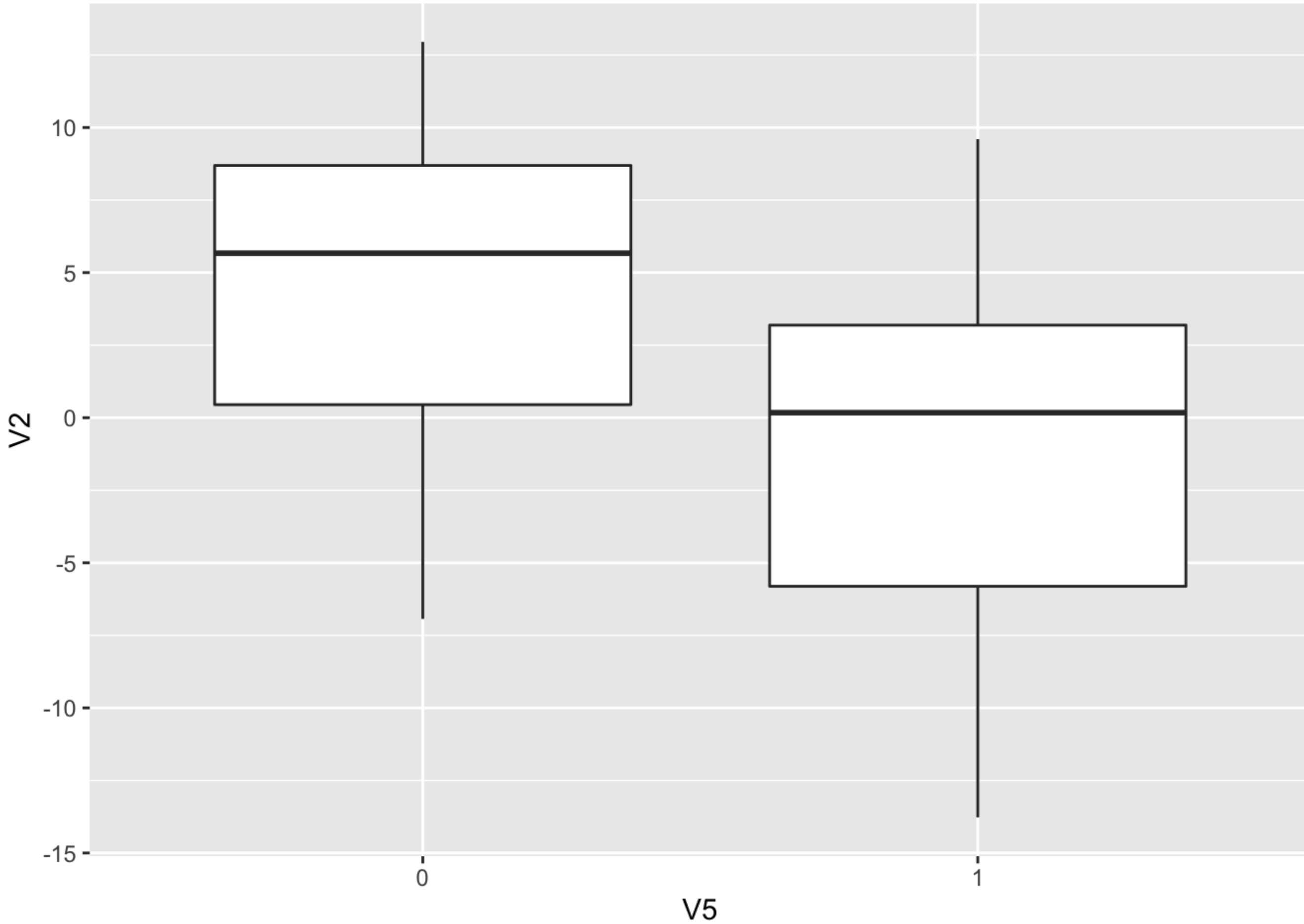
```
summary(project)
```

```
##        V1                V2                V3                V4         
##  Min.   :-7.0421   Min.   :-13.773   Min.   :-5.2861   Min.   :-8.5482  
##  1st Qu.:-1.7730   1st Qu.: -1.708   1st Qu.:-1.5750   1st Qu.:-2.4135  
##  Median : 0.4962   Median :  2.320   Median : 0.6166   Median :-0.5867  
##  Mean   : 0.4337   Mean   :  1.922   Mean   : 1.3976   Mean   :-1.1917  
##  3rd Qu.: 2.8215   3rd Qu.:  6.815   3rd Qu.: 3.1793   3rd Qu.: 0.3948  
##  Max.   : 6.8248   Max.   : 12.952   Max.   :17.9274   Max.   : 2.4495  
##  V5     
##  0:762  
##  1:610  
##         
##         
##         
## 
```

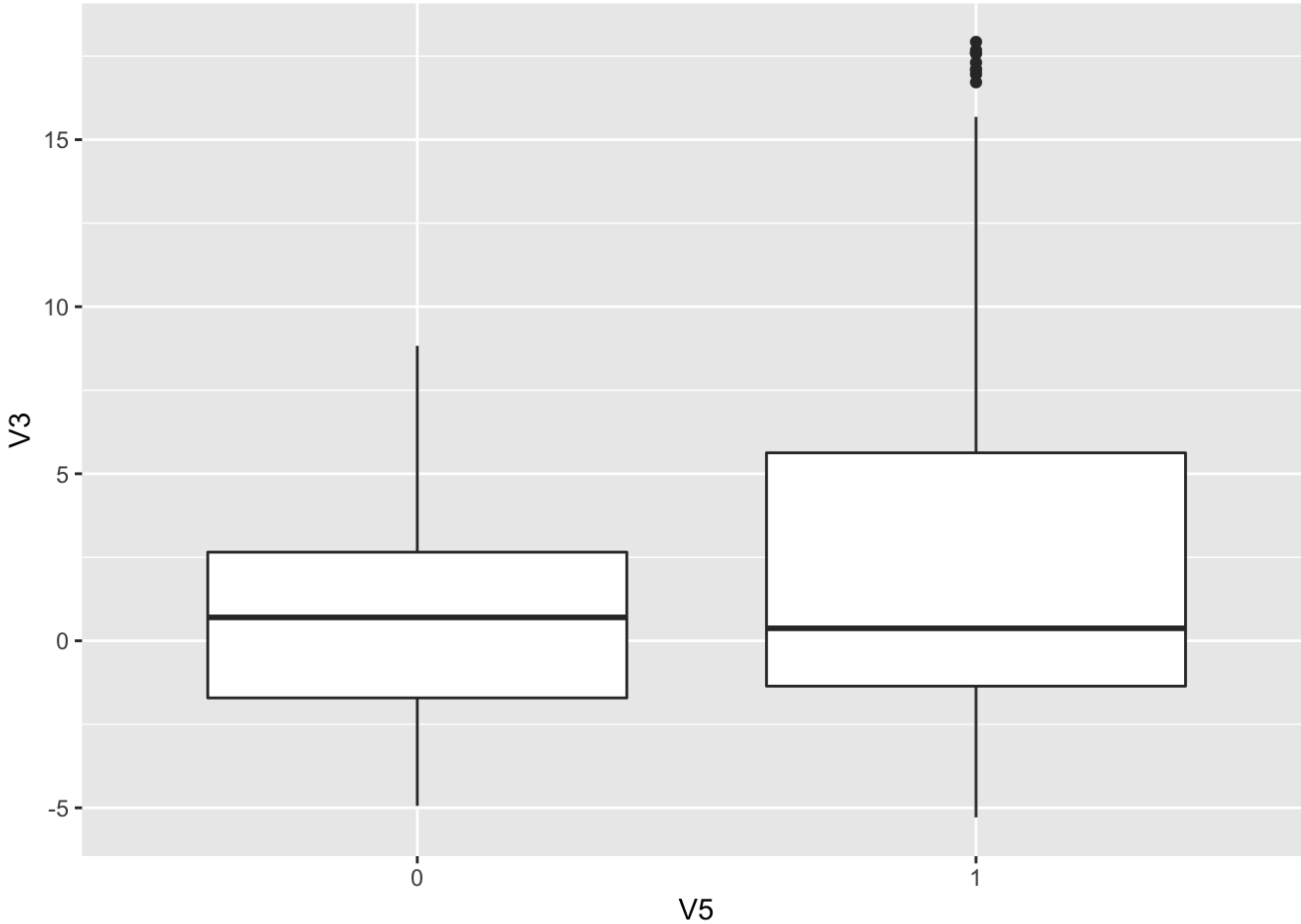
```
attach(project)
library(tidyverse)

ggplot(data = project, aes(x = V5)) +
  geom_boxplot(aes(y = V1))
```



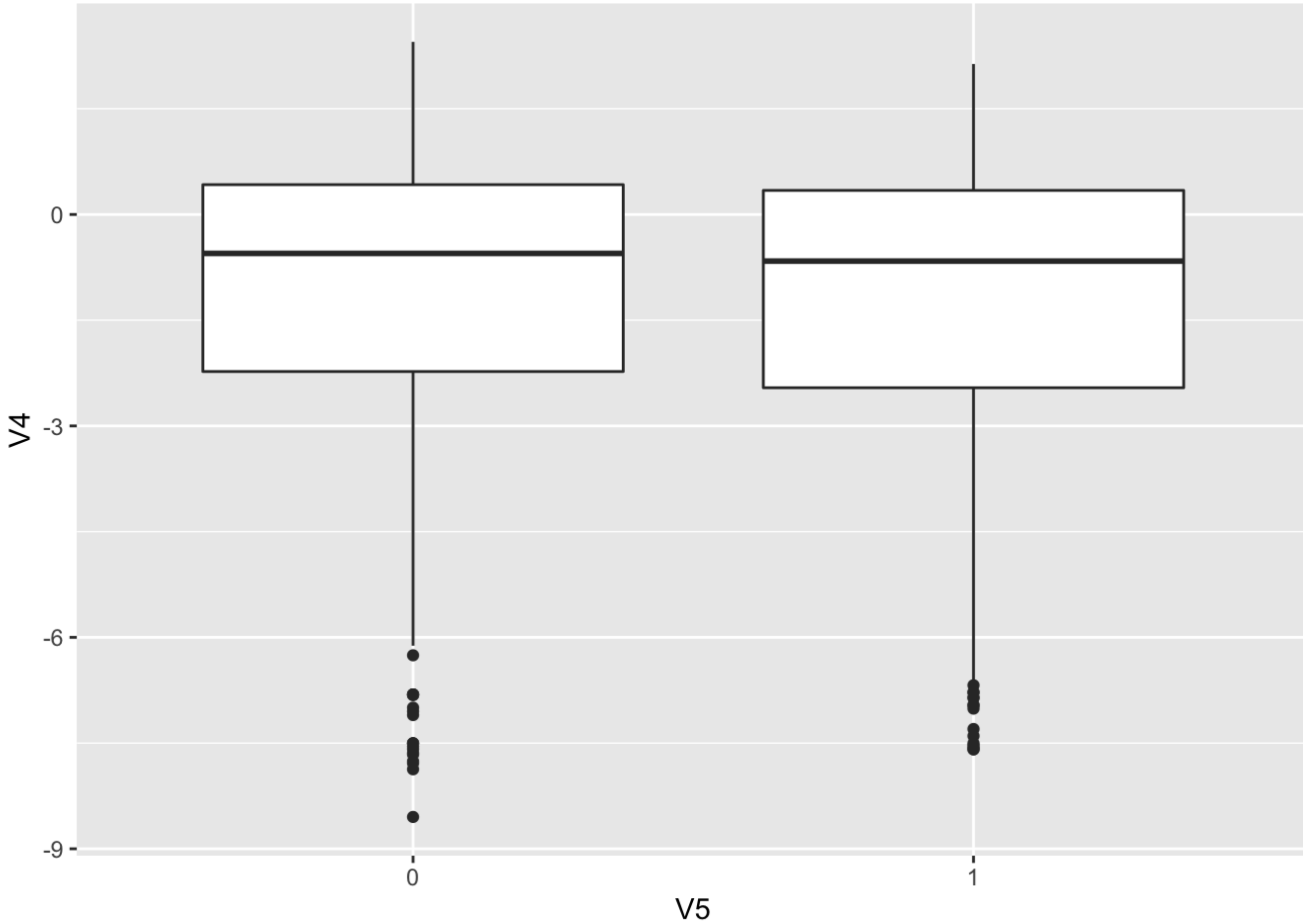

```
ggplot(data = project, aes(x = V5)) +
  geom_boxplot(aes(y = V2))
```



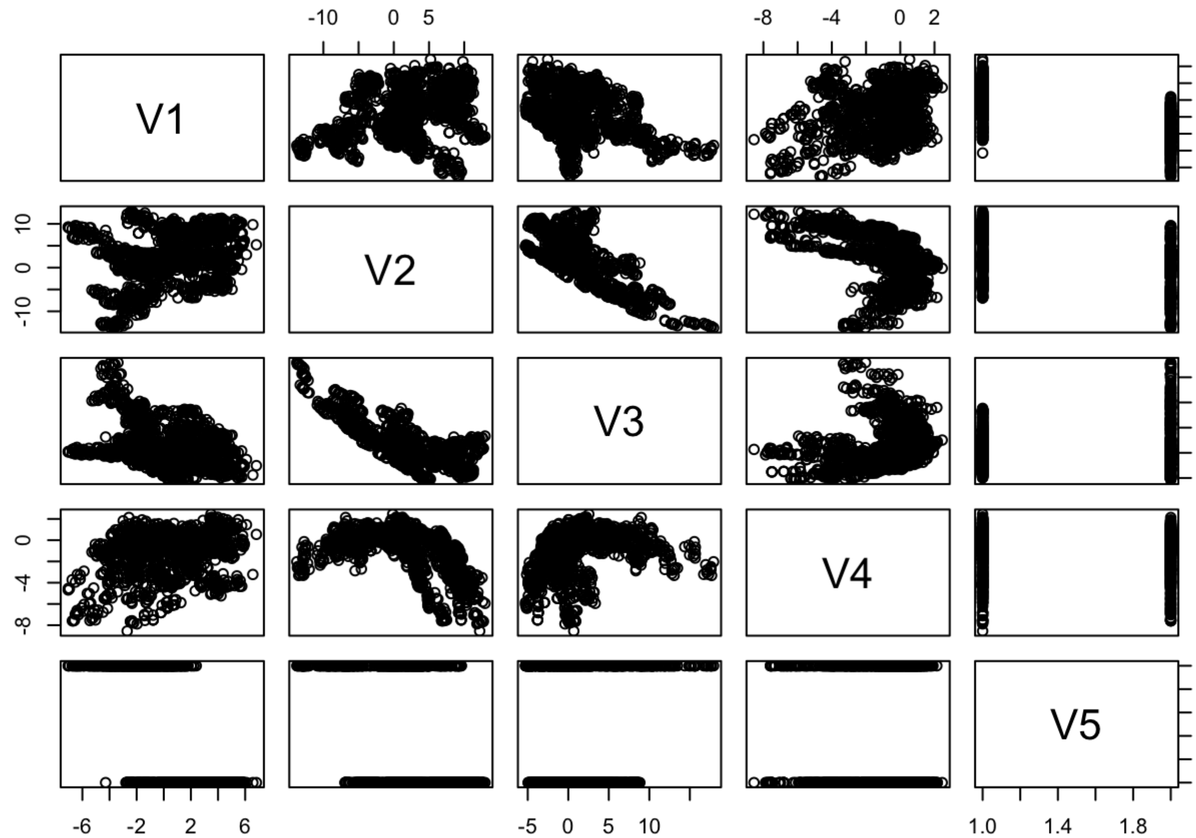

```
ggplot(data = project, aes(x = V5)) +
  geom_boxplot(aes(y = V3))
```



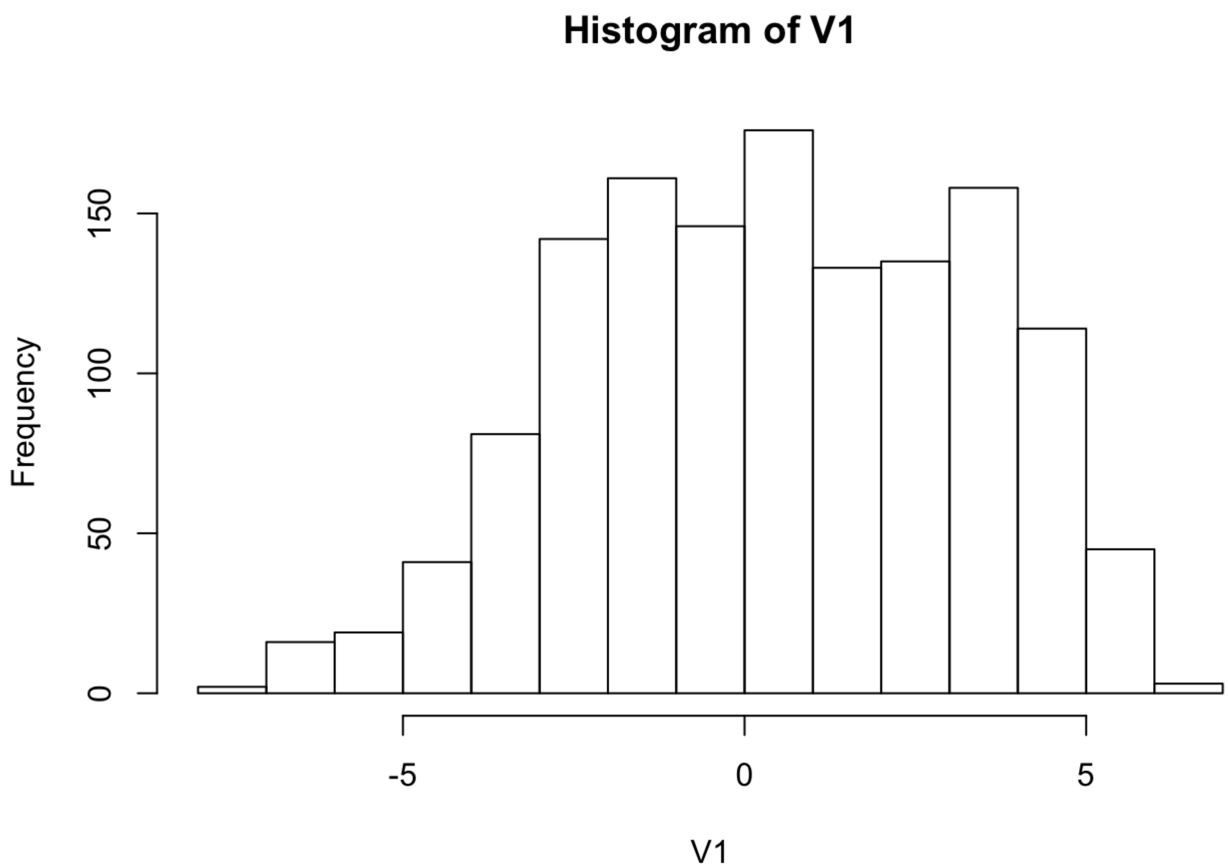

```
ggplot(data = project, aes(x = V5)) +
  geom_boxplot(aes(y = V4))
```



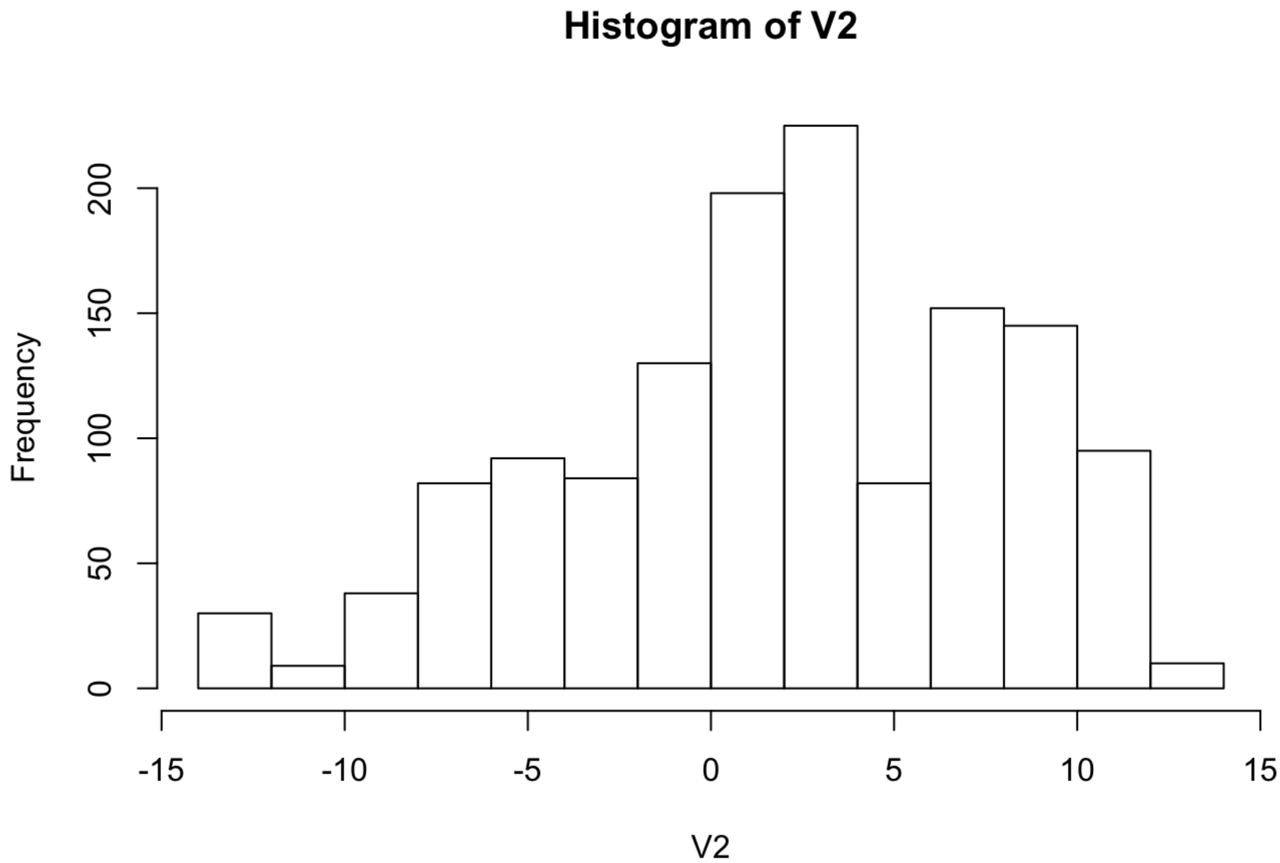

```
pairs(project)
```


V1
V2
V3
V4
V5
-10
0
5
-8
-4
0
2
4
0
-6
10
0
-10
15
5
-5
0
-4
-8
1.6
1.0
-6
-2
2
6
-5
0
5
10
1.0
1.4
1.8

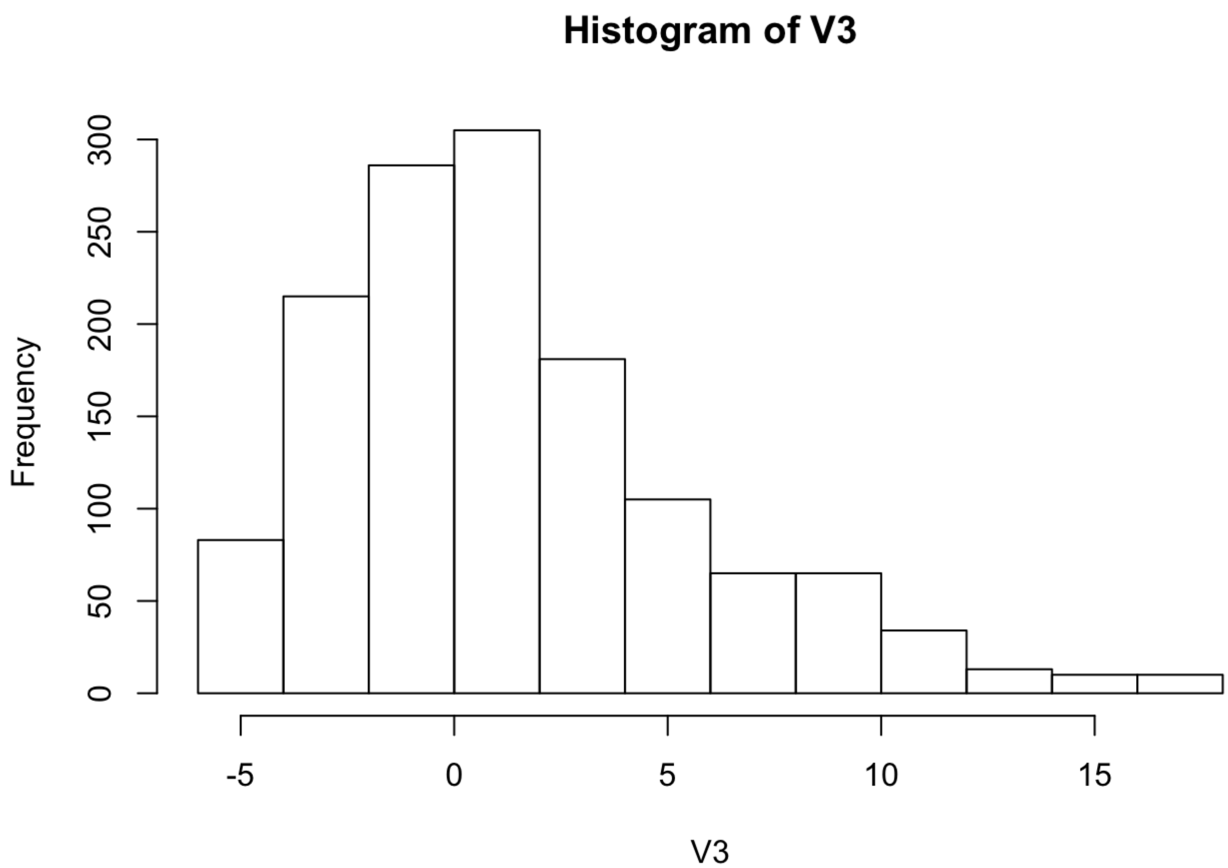
```
hist(V1)
```



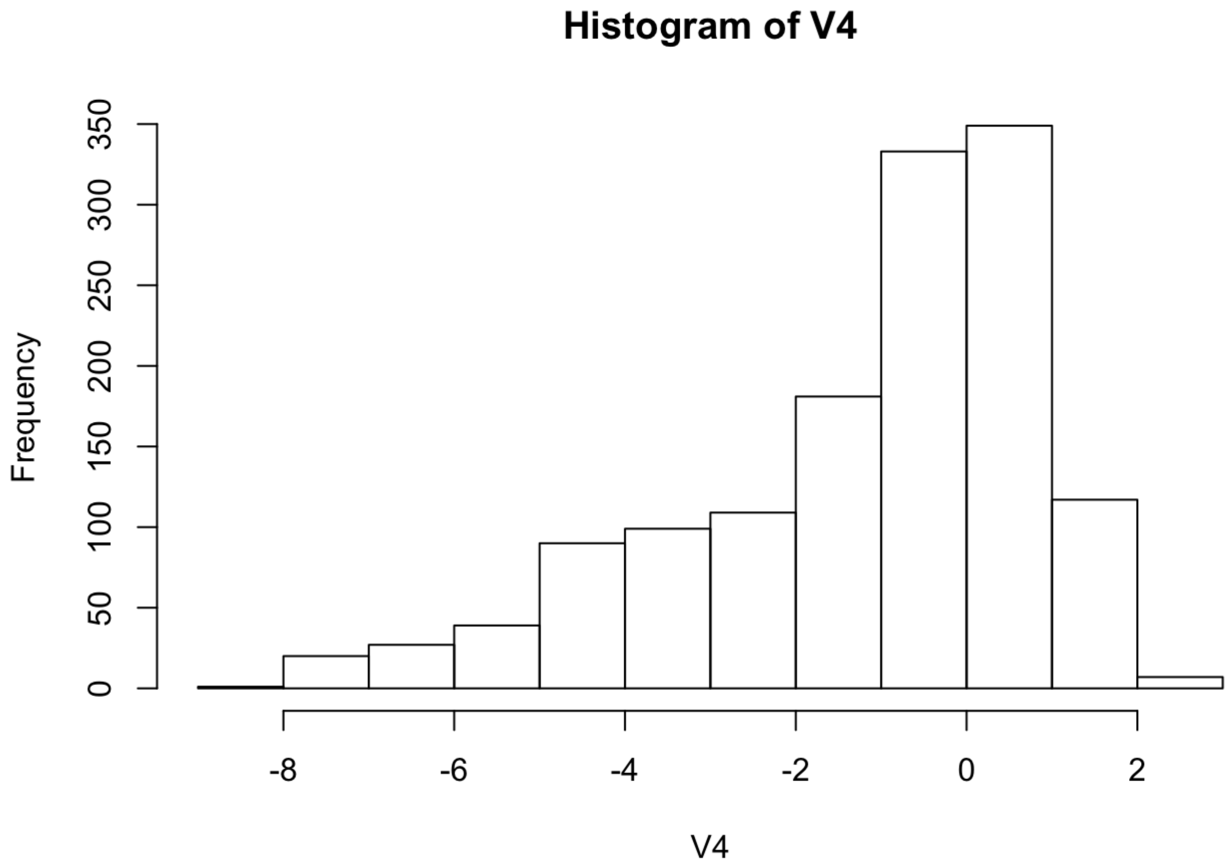

```
hist(V2)
```




```
hist(V3)
```




```
hist(V4)
```



Initial data sampling, splitting into a training and testing set using a 70/30 split.

```
# Data sampling

set.seed(7)
train <- sample(nrow(project), nrow(project) * .7)
project.train <- project[train,]
project.test <- project[-train,]

results <- data.frame(Model = character(), Test.Accuracy = numeric(), Train.Test.Split =
character(), stringsAsFactors = FALSE)
```

Logit model

```
log.fit <- glm(V5 ~ ., data = project.train, family = "binomial")
```

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

```
summary(log.fit)
```

```
##
## Call:
## glm(formula = V5 ~ ., family = "binomial", data = project.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4313   0.0000   0.0000   0.0002   2.0399
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   7.1549     1.7780   4.024 5.72e-05 ***
## V1           -8.1177     2.0792  -3.904 9.46e-05 ***
## V2           -4.0816     1.0603  -3.850 0.000118 ***
## V3           -5.2284     1.3550  -3.858 0.000114 ***
## V4           -0.4821     0.4060  -1.187 0.235065
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1317.273  on 959  degrees of freedom
## Residual deviance:   35.295  on 955  degrees of freedom
## AIC: 45.295
##
## Number of Fisher Scoring iterations: 12
```

```
preds <- predict(log.fit, type = 'response')

project.train.log <- project.train %>%
  mutate(probs = preds) %>%
  mutate(pred = ifelse(probs > .5, 1, 0)) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Training Accuracy
sum(project.train.log$same == TRUE)/nrow(project.train.log)
```

```
## [1] 0.990625
```

```
# Test

preds <- predict(log.fit, newdata = project.test, type = 'response')
project.test.log <- project.test %>%
  mutate(probs = preds) %>%
  mutate(pred = ifelse(probs > .5, 1, 0)) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Test Accuracy
results[1,] <- c('logit.test', sum(project.test.log$same == TRUE)/nrow(project.test.lo
g), '70/30')

# Test confusion Matrix
table(true = project.test.log[,5], pred = project.test.log[,"pred"])
```

```
##      pred
## true   0   1
##    0 225   0
##    1   3 184
```

LDA

```
set.seed(7)
lda.fit <- lda(V5~., data = project.train)
lda.fit
```

```
## Call:
## lda(V5 ~ ., data = project.train)
##
## Prior probabilities of groups:
##        0        1
## 0.559375 0.440625
##
## Group means:
##          V1         V2        V3        V4
## 0  2.305713  4.1124680 0.8016314 -1.142657
## 1 -1.883848 -0.7954217 1.9738657 -1.280306
##
## Coefficients of linear discriminants:
##              LD1
## V1 -0.8310396111
## V2 -0.4557457935
## V3 -0.5955229640
## V4 -0.0006491046
```

```
preds <- predict(lda.fit, type = 'response')

project.train.lda <- project.train %>%
  mutate(pred = preds$class) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Training accuracy

sum(project.train.lda$same == TRUE)/nrow(project.train.lda)
```

```
## [1] 0.975
```

```
# Test

preds <- predict(lda.fit, project.test, type = 'response')

project.test.lda <- project.test %>%
  mutate(pred = preds$class) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Test accuracy

results[2,] <- c('lda.test', sum(project.test.lda$same == TRUE)/nrow(project.test.lda),
'70/30')

# Test confusion Matrix
table(true = project.test[,5], pred = preds$class)
```

```
##     pred
## true   0   1
##    0 217   8
##    1   0 187
```

QDA

```
set.seed(7)
qda.fit <- qda(V5~., data = project.train)
qda.fit
```

```
## Call:
## qda(V5 ~ ., data = project.train)
##
## Prior probabilities of groups:
##        0        1
## 0.559375 0.440625
##
## Group means:
##          V1         V2        V3        V4
## 0  2.305713  4.1124680 0.8016314 -1.142657
## 1 -1.883848 -0.7954217 1.9738657 -1.280306
```

```r
preds <- predict(qda.fit, type = 'response')

project.train.qda <- project.train %>%
  mutate(pred = preds$class) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Training accuracy

sum(project.train.qda$same == TRUE)/nrow(project.train.qda)
```

```
## [1] 0.9864583
```

```r
# Test

preds <- predict(qda.fit, project.test, type = 'response')

project.test.qda <- project.test %>%
  mutate(pred = preds$class) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Test accuracy

results[3,] <- c('qda.test', sum(project.test.qda$same == TRUE)/nrow(project.test.qda),
'70/30')

# Test confusion Matrix
table(true = project.test[,5], pred = preds$class)
```

```
##     pred
## true   0   1
##    0 223   2
##    1   0 187
```

Bagging

```r
set.seed(7)
bag.fit <- randomForest(V5 ~ ., data = project.train,
                        mtry=4,importance =TRUE)

bag.fit
```

```
##
## Call:
##  randomForest(formula = V5 ~ ., data = project.train, mtry = 4,      importance = TRU
E)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 0.94%
## Confusion matrix:
##     0   1 class.error
## 0 531   6 0.011173184
## 1   3 420 0.007092199
```

```
preds <- predict(bag.fit, project.train, type = 'response')

project.train.bag <- project.train %>%
  mutate(pred = preds) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Training accuracy

sum(project.train.bag$same == TRUE)/nrow(project.train.bag)
```

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

```
# Test

preds <- predict(bag.fit, project.test, type = 'response')

project.test.bag <- project.test %>%
  mutate(pred = preds) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Test accuracy

results[4,] <- c('bag.test', sum(project.test.bag$same == TRUE)/nrow(project.test.bag),
'70/30')

# Test confusion Matrix
table(true = project.test[,5], preds)
```

```
##     preds
## true   0   1
##    0 223   2
##    1   3 184
```

Random Forest

```
set.seed(7)

rf.fit <- randomForest(V5 ~ ., data = project.train,
                        importance =TRUE)

rf.fit
```

```
##
## Call:
##  randomForest(formula = V5 ~ ., data = project.train, importance = TRUE)
##               Type of random forest: classification
##                     Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.73%
## Confusion matrix:
##     0   1 class.error
## 0 532   5 0.009310987
## 1   2 421 0.004728132
```

```
preds <- predict(rf.fit, project.train, type = 'response')

project.train.rf <- project.train %>%
  mutate(pred = preds) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Training accuracy

sum(project.train.rf$same == TRUE)/nrow(project.train.rf)
```

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

```
# Test

preds <- predict(rf.fit, project.test, type = 'response')

project.test.rf <- project.test %>%
  mutate(pred = preds) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Test accuracy

results[5,] <- c('rf.test', sum(project.test.rf$same == TRUE)/nrow(project.test.rf), '7
0/30')

# Test confusion Matrix
table(true = project.test[,5], preds)
```

```
##     preds
## true   0   1
##    0 223   2
##    1   0 187
```

Model Evaluation

```
results_ordered <- results[order(results$Test.Accuracy, decreasing = TRUE),]
print(results_ordered)
```

```
##        Model     Test.Accuracy Train.Test.Split
## 3   qda.test 0.995145631067961            70/30
## 5    rf.test 0.995145631067961            70/30
## 1 logit.test 0.992718446601942            70/30
## 4   bag.test 0.987864077669903            70/30
## 2   lda.test 0.980582524271845            70/30
```

In this run of the models all of the models perform extremely well against the test data. The best performing models are QDA and a Random Forest using all available predictors with no manipulation of the variables. Both models have a Test Accuracy of 99.5%. We decided to run the mdoels again with a more even split between training and testing data to ensure that the models are not being overfit.

Initial data sampling, splitting into a training and testing set using a 50/50 split.

```
# Data sampling

set.seed(7)
train <- sample(nrow(project), nrow(project) / 2)
project.train <- project[train,]
project.test <- project[-train,]
```

Logit model

```
log.fit <- glm(V5 ~ ., data = project.train, family = "binomial")
```

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

```
summary(log.fit)
```

```
##
## Call:
## glm(formula = V5 ~ ., family = "binomial", data = project.train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.40854   0.00000   0.00000   0.00002   1.90670
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   9.5408     3.7173   2.567  0.01027 *
## V1          -10.0763     3.7693  -2.673  0.00751 **
## V2           -4.7874     1.7553  -2.727  0.00638 **
## V3           -6.3879     2.3704  -2.695  0.00704 **
## V4           -0.3401     0.4676  -0.727  0.46705
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 932.630  on 685  degrees of freedom
## Residual deviance:  22.936  on 681  degrees of freedom
## AIC: 32.936
##
## Number of Fisher Scoring iterations: 13
```

```
preds <- predict(log.fit, type = 'response')

project.train.log <- project.train %>%
  mutate(probs = preds) %>%
  mutate(pred = ifelse(probs > .5, 1, 0)) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Training Accuracy
sum(project.train.log$same == TRUE)/nrow(project.train.log)
```

```
## [1] 0.9897959
```

```
# Test

preds <- predict(log.fit, newdata = project.test, type = 'response')
project.test.log <- project.test %>%
  mutate(probs = preds) %>%
  mutate(pred = ifelse(probs > .5, 1, 0)) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Test Accuracy
results[6,] <- c('logit.test', sum(project.test.log$same == TRUE)/nrow(project.test.lo
g), '50/50')

# Test confusion Matrix
table(true = project.test.log[,5], pred = project.test.log[,"pred"])
```

```
##     pred
## true   0   1
##    0 359   4
##    1   4 319
```

LDA

```
set.seed(7)
lda.fit <- lda(V5~., data = project.train)
lda.fit
```

```
## Call:
## lda(V5 ~ ., data = project.train)
##
## Prior probabilities of groups:
##         0         1
## 0.5816327 0.4183673
##
## Group means:
##          V1         V2        V3        V4
## 0  2.383398  4.1677802 0.7467721 -1.029451
## 1 -1.808978 -0.8035846 1.8425322 -1.179726
##
## Coefficients of linear discriminants:
##             LD1
## V1 -0.847225526
## V2 -0.471776977
## V3 -0.623171889
## V4 -0.008809756
```

```
preds <- predict(lda.fit, type = 'response')

project.train.lda <- project.train %>%
  mutate(pred = preds$class) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Training accuracy

sum(project.train.lda$same == TRUE)/nrow(project.train.lda)
```

```
## [1] 0.9766764
```

```
# Test

preds <- predict(lda.fit, project.test, type = 'response')

project.test.lda <- project.test %>%
  mutate(pred = preds$class) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Test accuracy

results[7,] <- c('lda.test', sum(project.test.lda$same == TRUE)/nrow(project.test.lda),
'50/50')

# Test confusion Matrix
table(true = project.test[,5], pred = preds$class)
```

```
##     pred
## true   0   1
##    0 347  16
##    1   1 322
```

QDA

```
set.seed(7)
qda.fit <- qda(V5~., data = project.train)
qda.fit
```

```
## Call:
## qda(V5 ~ ., data = project.train)
##
## Prior probabilities of groups:
##         0         1
## 0.5816327 0.4183673
##
## Group means:
##          V1         V2        V3        V4
## 0  2.383398  4.1677802 0.7467721 -1.029451
## 1 -1.808978 -0.8035846 1.8425322 -1.179726
```

```
preds <- predict(qda.fit, type = 'response')

project.train.qda <- project.train %>%
  mutate(pred = preds$class) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Training accuracy

sum(project.train.qda$same == TRUE)/nrow(project.train.qda)
```

```
## [1] 0.983965
```

```
# Test

preds <- predict(qda.fit, project.test, type = 'response')

project.test.qda <- project.test %>%
  mutate(pred = preds$class) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Test accuracy

results[8,] <- c('qda.test', sum(project.test.qda$same == TRUE)/nrow(project.test.qda),
'50/50')

# Test confusion Matrix
table(true = project.test[,5], pred = preds$class)
```

```
##     pred
## true   0   1
##    0 354   9
##    1   0 323
```

Bagging

```
set.seed(7)
bag.fit <- randomForest(V5 ~ ., data = project.train,
                        mtry=4,importance =TRUE)

bag.fit
```

```
##
## Call:
##  randomForest(formula = V5 ~ ., data = project.train, mtry = 4,      importance = TRU
E)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 1.02%
## Confusion matrix:
##     0   1 class.error
## 0 394   5 0.012531328
## 1   2 285 0.006968641
```

```
preds <- predict(bag.fit, project.train, type = 'response')

project.train.bag <- project.train %>%
  mutate(pred = preds) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Training accuracy

sum(project.train.bag$same == TRUE)/nrow(project.train.bag)
```

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

```
# Test

preds <- predict(bag.fit, project.test, type = 'response')

project.test.bag <- project.test %>%
  mutate(pred = preds) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Test accuracy

results[9,] <- c('bag.test', sum(project.test.bag$same == TRUE)/nrow(project.test.bag),
'50/50')

# Test confusion Matrix
table(true = project.test[,5], preds)
```

```
##     preds
## true   0   1
##    0 354   9
##    1   5 318
```

Random Forest

```
set.seed(7)

rf.fit <- randomForest(V5 ~ ., data = project.train,
                        importance =TRUE)

rf.fit
```

```
##
## Call:
##  randomForest(formula = V5 ~ ., data = project.train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 1.02%
## Confusion matrix:
##     0   1 class.error
## 0 395   4  0.01002506
## 1   3 284  0.01045296
```

```
preds <- predict(rf.fit, project.train, type = 'response')

project.train.rf <- project.train %>%
  mutate(pred = preds) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Training accuracy

sum(project.train.rf$same == TRUE)/nrow(project.train.rf)
```

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

```
# Test

preds <- predict(rf.fit, project.test, type = 'response')

project.test.rf <- project.test %>%
  mutate(pred = preds) %>%
  mutate(same = ifelse(pred == V5, TRUE, FALSE))

# Test accuracy

results[10,] <- c('rf.test', sum(project.test.rf$same == TRUE)/nrow(project.test.rf), '5
0/50')

# Test confusion Matrix
table(true = project.test[,5], preds)
```

```
##     preds
## true   0   1
##    0 359   4
##    1   1 322
```

Model Evaluation

```
results_ordered <- results[order(results$Test.Accuracy, decreasing = TRUE),]
print(results_ordered)
```

```
##         Model     Test.Accuracy Train.Test.Split
## 3    qda.test 0.995145631067961            70/30
## 5     rf.test 0.995145631067961            70/30
## 1  logit.test 0.992718446601942            70/30
## 10    rf.test 0.992711370262391            50/50
## 6  logit.test 0.988338192419825            50/50
## 4    bag.test 0.987864077669903            70/30
## 8    qda.test 0.986880466472303            50/50
## 2    lda.test 0.980582524271845            70/30
## 9    bag.test 0.979591836734694            50/50
## 7    lda.test 0.975218658892128            50/50
```

After running the model with a different split ratio of the training and test sets, we have decided that the simple Random Forest using all of the available predictor variables is the best model. This is due to the fact that it has a 99.5% Test Accuracy using a 70/30 split which is tied for the best with QDA using the same split, but QDA loses almsot a full percentage point of accuracy when using a 50/50 split, whereas the Random Forest is still ranked as the best model by Test Accuracy when using the 50/50 data split. In general all of these models are great performers though, with the worst performing model being LDA with the lowest accuracy of 97.5%.