Model #6

Ethan Allavarpu (UID: 405287603)

10/28/2020

Transforming and Cleaning the Data

```
training <- read.csv("training.csv", stringsAsFactors = TRUE)
training$class <- factor(training$class)
levels(training$class) <- c("NG", "OG", "TSG")
outlier <- function(data) {
  low <- mean(data) - 3 * sd(data)
  high <- mean(data) + 3 * sd(data)
  which(data < low | data > high)
}
library(ggplot2)
scatter <- function(var) {
  ggplot(training, aes_string(var, "class")) +
    geom_jitter(width = 0.05, height = 0.1, size = 0.1,
                colour = rgb(0, 0, 0, alpha = 1 / 3))
}
scat_plot <- lapply(names(training)[-99], scatter)
library(gridExtra)
# grid.arrange(grobs = scat_plot[1:20], ncol = 4)
# grid.arrange(grobs = scat_plot[21:40], ncol = 4)
# grid.arrange(grobs = scat_plot[41:60], ncol = 4)
# grid.arrange(grobs = scat_plot[61:80], ncol = 4)
# grid.arrange(grobs = scat_plot[81:98], ncol = 4)
outlier_index <- sort(table(unlist(lapply(training[,-99], outlier))), decreasing = TRUE)
outlier_index[1:100]
```

```
 915 1280 2918  517 1914 2182 3052 1173 2215 3049  259  740 1749 1979 2998  417
  24   24   24   22   22   22   20   19   19   19   18   18   18   18   18   17
 441  806 2297  422  635 1258 1570 2278 2518 2729   80  150 2694  169  276  341
  17   17   17   16   16   16   16   16   16   16   15   15   15   14   14   14
1528 1556 1726 1809 1911 1955 2071 2624 2641 3120 3142   73  277  364  751 1244
  14   14   14   14   14   14   14   14   14   14   14   13   13   13   13   13
1330 2329 2787  343 1138 1171 1188 1372 1460 2031 2251 2968 2983 3166  352  634
  13   13   13   12   12   12   12   12   12   12   12   12   12   12   11   11
 907  923 1096 1858 2636  588 1137 1317 1463 1561 1740 1991 2487 2540 2555 2621
  11   11   11   11   11   10   10   10   10   10   10   10   10   10   10   10
2815 3029   74  144  657  789  857 1267 1610 1932 2022 2093 2142 2534 2666 2721
  10   10    9    9    9    9    9    9    9    9    9    9    9    9    9    9
2848 2900 3027  155
   9    9    9    8
```

```r
training <- training[-as.numeric(names(outlier_index)[1:50]),]
sort(training$Missense_TO_Silent_Ratio, decreasing = TRUE)[1:10]
```

```
 [1] 384.98658 172.91420 135.59623  71.09712  23.21809  21.81193  20.37791
 [8]  19.42402  19.38769  15.84808
```

```r
training <- training[-which(training$Missense_TO_Silent_Ratio > 100), ]
sort(training$Missense_KB_Ratio, decreasing = TRUE)[1:10]
```

```
 [1] 2063.9413 1296.6625 1060.0601  952.3810  931.4227  726.8519  594.7603
 [8]  593.3610  581.5085  516.8084
```

```r
training <- training[-which(training$Missense_KB_Ratio > 2000), ]
sort(training$LOF_TO_Silent_Ratio, decreasing = TRUE)[1:10]
```

```
 [1] 81.177835  9.030120  6.470238  5.582840  4.741460  4.558252  4.176630
 [8]  4.058140  4.039062  4.021930
```

```r
training <- training[-which(training$LOF_TO_Silent_Ratio > 5), ]
sort(training$Gene_expression_Z_score, decreasing = TRUE)[1:10]
```

```
 [1] 19.720  9.210  7.080  6.883  6.590  6.280  5.321  5.316  3.161  2.767
```

```r
training <- training[-which(training$Gene_expression_Z_score > 4), ]
sort(training$dN_to_dS_ratio, decreasing = TRUE)[1:10]
```

```
 [1] 20.950  3.649  3.446  3.372  2.574  2.194  2.183  2.102  1.921  1.744
```

```r
training <- training[-which(training$dN_to_dS_ratio > 5),]
sort(training$Silent_KB_Ratio, decreasing = TRUE)[1:10]
```

```
 [1] 474.4745 193.1684 174.0558 171.0362 166.4971 160.2273 158.7697 148.5800
 [9] 143.6782 135.2657
```

```r
training <- training[-which(training$Silent_KB_Ratio > 200), ]
sort(training$Lost_start_and_stop_fraction, decreasing = TRUE)[1:10]
```

```
 [1] 0.333 0.167 0.118 0.087 0.074 0.071 0.071 0.068 0.067 0.067
```

```r
training <- training[-which(training$Lost_start_and_stop_fraction > 0.2),]
sort(training$Synonymous_Zscore, decreasing = FALSE)[1:10]
```

```
 [1] -20.5110 -10.9780 -10.2960  -9.7346  -9.3720  -8.8090  -8.4062  -8.3918
 [9]  -8.1076  -8.1076
```

```r
training <- training[-which(training$Synonymous_Zscore < -15), ]
numeric_training <- training[,-99]

n_zeroes <- rep(NA, nrow(numeric_training))

for(i in seq_len(nrow(numeric_training))){
  row_i_zeroes <- 0
  for(j in seq_len(ncol(numeric_training))){
    if(round(numeric_training[i,j], digits = 5) == 0){
      row_i_zeroes <- row_i_zeroes + 1
    }
  }
  n_zeroes[i] <- row_i_zeroes
}
```

```
training <- training[n_zeroes <= 50, ]
```

```
library(dplyr)
```

```

Attaching package: 'dplyr'

The following object is masked from 'package:gridExtra':

    combine

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
#function to calculate wca
score <- function (conf_mat) {
  print(sum(diag(conf_mat) * c(1, 20, 20)))
  print(sum(diag(conf_mat) * c(1, 20, 20)) / sum(apply(conf_mat, 2, sum) * c(1, 20, 20)))
}

# Set new threshold to account for unbalanced data
classify <- function(probs) {
  if (any(probs[2:3] > 0.05)) {
    subset <- probs[2:3]
    output <- which(subset == max(subset))
    if (length(output) > 1) {
        output <- sample(1:2, 1)
    }
  } else {
    output <- 0
  }
  output
}
```

Multinom (Logistic Regression)

```
library(dplyr)
library(caret)
```

```
Loading required package: lattice
```

```
set.seed(43)
# vars <- training %>% select(Broad_H3K9ac_percentage, N_LOF, pLOF_Zscore,
#                              Missense_Entropy,
#                             N_Splice, LOF_TO_Total_Ratio, VEST_score,
#                             BioGRID_log_degree,
#                             Broad_H3K79me2_percentage, FamilyMemberCount,
#                             S50_score_replication_timing, Gene_expression_Z_score,
#                             Polyphen2, Broad_H3K36me3_percentage, class)

vars_index <- createDataPartition(training$class, p = 0.76,
                                  list = FALSE)
vars_train <- training[vars_index, ] #was formerly vars[vars_index,]...
vars_test <- training[-vars_index, ]

mn <- nnet::multinom(class ~ ., data = vars_train, model = TRUE)
```

```
# weights:  300 (198 variable)
initial  value 2529.005489
iter  10 value 1080.678526
iter  20 value 987.136589
iter  30 value 885.515823
iter  40 value 781.246074
iter  50 value 700.655896
iter  60 value 658.140282
iter  70 value 566.093553
iter  80 value 446.705622
iter  90 value 349.145460
iter 100 value 260.176659
final  value 260.176659
stopped after 100 iterations
```

```
tidymn <- broom::tidy(mn) %>% arrange(p.value)
terms <- tidymn$term[-(1:2)]
terms_unique <- unique(terms)
top_14 <- terms_unique[1:14]
cat(top_14, sep = ",")
```

```
LOF_TO_Silent_Ratio,Splice_TO_Silent_Ratio,Missense_TO_Silent_Ratio,LOF_TO_Benign_Ratio,Splice_TO_Benig
```

```
#set.seed(9) gives 0.82
#set.seed(2) gives 0.77
#set.seed(12) 0.81
#set.seed(3275) gives 0.76
#set.seed(999) gives 0.77
#set.seed(1235) gives 0.84
#set.seed(712) gives 0.79
#set.seed(100) gives 0.799
#set.seed(200) gives 0.71
set.seed(712)
```

```r
vars_mn <- training %>% select(
  LOF_TO_Silent_Ratio,Splice_TO_Silent_Ratio,Missense_TO_Silent_Ratio,LOF_TO_Benign_Ratio,Splice_TO_Beni
)

vars_index <- createDataPartition(vars_mn$class, p = 0.76,
                                  list = FALSE)
vars_train <- training[vars_index, ] # it's being overridden
vars_test <- training[-vars_index, ]

mn <- nnet::multinom(class ~ ., data = vars_train, model = TRUE)
```

```
# weights:  300 (198 variable)
initial  value 2529.005489
iter  10 value 1083.612637
iter  20 value 992.226859
iter  30 value 898.663909
iter  40 value 786.702868
iter  50 value 722.496434
iter  60 value 680.350546
iter  70 value 589.189119
iter  80 value 445.320264
iter  90 value 337.815570
iter 100 value 244.883308
final  value 244.883308
stopped after 100 iterations
```

```r
tests <- read.csv("test.csv")
preds <- predict(mn, newdata = vars_test, type = "prob")
predclass <- apply(preds, 1, classify)
tbl <- table(predclass, vars_test$class)
score(tbl)
```

```
[1] 1569
[1] 0.7924242
```

```r
tbl
```

```

predclass  NG  OG TSG
        0 569   2   3
        1  44  28   6
        2  47   5  22
```

```r
test_preds <- predict(mn, newdata = tests, type = "prob")
test_preds <- apply(test_preds, 1, classify)
csv_file <- data.frame("id" = tests$id,
                       "class" = test_preds)
write.csv(csv_file, "modelpredictions7_REFINED_LOG.csv", row.names = FALSE)
```