PERSONALIZED MEDICINE: REDEFINING CANCER TREATMENT

EDA & XGB MODEL by Tyag Raj

A Project report for Edwisor.com, 2017

Table of Contents

1. Problem Statement

Once sequenced, a cancer tumor can have thousands of genetic mutations. But the challenge is distinguishing the mutations that contribute to tumor growth (drivers) from the neutral mutations (passengers). Currently this interpretation of genetic mutations is being done manually. This is a very time-consuming task where a clinical pathologist has to manually review and classify every single genetic mutation based on evidence from text-based clinical literature. This Project focuses to automatically classify genetic mutations that contribute to cancer tumor growth (so-called “drivers”) in the presence of mutations that are don’t affect the tumors (“passengers”).

1.1 Exploratory Data Analysis

This is an Exploratory Data Analysis for the Personalized Medicine: Redefining Cancer Treatment.

The data comes in 4 different files. Two csv files and two text files:

- training/test variants: These are csv catalogues of the gene mutations together with the target value Class, which is the (manually) classified assessment of the mutation. The feature variables are Gene, the specific gene where the mutation took place, and Variation, the nature of the mutation. The test data of course doesn’t have the Class values. This is what we have to predict. These two files each are linked through an ID variable to another file each, namely:
- training/test text: Those contain an extensive description of the evidence that was used (by experts) to manually label the mutation classes.

The text information holds the key to the classification problem and will have to be understood/modelled well to achieve a useful accuracy.

Set the working directory to load the files and load the required libraries in to R environment.

```
library('ggplot2') # visualization
library('corrplot') # visualisation
library('dplyr') # data manipulation
library('readr') # data input
library('tibble') # data wrangling
library('tidyr') # data wrangling
library('stringr') # string manipulation
library('tidytext') # text mining
library('SnowballC') # text analysis
library('wordcloud') # test visualisation
```

Reading in the text and variant files:

```
train_txt_dump <- tibble(text = read_lines('training_text', skip = 1))
train_txt <- train_txt_dump %>%
                        separate(text, into = c("ID", "txt"), sep = "\\|\\|")
train_txt <- train_txt %>%
                mutate(ID = as.integer(ID))
test_txt_dump <- tibble(text = read_lines('test_text', skip = 1)
)
test_txt <- test_txt_dump %>%
                separate(text, into = c("ID", "txt"), sep = "\\|\\|")
test_txt <- test_txt %>%
                 mutate(ID = as.integer(ID))

train <- read_csv('training_variants')
test  <- read_csv('test_variants')
```

Starting this EDA with a look at the variants data files, which are more easily accessible through common visualisation tools.

1.2 Overview of the data:

```
train <- train %>%
          mutate(Gene = factor(Gene),
          Variation = factor(Variation),
          Class = factor(Class))

test <- test %>%
         mutate(Gene = factor(Gene),
          Variation = factor(Variation))

summary(train, maxsum = 9)
```

```
glimpse(train)
## Observations: 3,321
## Variables: 4
## $ ID        <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15...
## $ Gene      <fctr> FAM58A, CBL, CBL, CBL, CBL, CBL, CBL, CBL, CBL, CBL...
## $ Variation <fctr> Truncating Mutations, W802*, Q249E, N454D, L399V, V...
## $ Class     <fctr> 1, 2, 2, 3, 4, 4, 5, 1, 4, 4, 4, 4, 4, 4, 5, 4, 1, ...
nrow(train)
## [1] 3321
nrow(test)
## [1] 5668
sum(is.na(train))
## [1] 0
sum(is.na(test))
## [1] 0
train %>%
  group_by(Gene) %>%
  summarise(ct = n()) %>%
  arrange(desc(ct))
## # A tibble: 264 x 2
##      Gene    ct
##    <fctr> <int>
##  1  BRCA1   264
##  2   TP53   163
##  3   EGFR   141
##  4   PTEN   126
##  5  BRCA2   125
##  6    KIT    99
##  7   BRAF    93
##  8    ALK    69
##  9  ERBB2    69
## 10 PDGFRA    60
## # ... with 254 more rows
```

```
test %>%
  group_by(Gene) %>%
  summarise(ct = n()) %>%
  arrange(desc(ct))
## # A tibble: 1,397 x 2
##     Gene    ct
##   <fctr> <int>
## 1     F8   134
## 2   CFTR    57
## 3     F9    54
## 4   G6PD    46
## 5    GBA    39
## 6     AR    38
## 7    PAH    38
## 8   CASR    37
## 9   ARSA    30
## 10  BRCA1    29
## # ... with 1,387 more rows
train %>%
  group_by(Variation) %>%
  summarise(ct = n()) %>%
  arrange(desc(ct))
## # A tibble: 2,996 x 2
##             Variation    ct
##                <fctr> <int>
## 1 Truncating Mutations    93
## 2           Deletion    74
## 3       Amplification    71
## 4            Fusions    34
## 5      Overexpression     6
## 6                G12V     4
## 7                E17K     3
## 8                Q61H     3
## 9                Q61L     3
## 10               Q61R     3
## # ... with 2,986 more rows
```

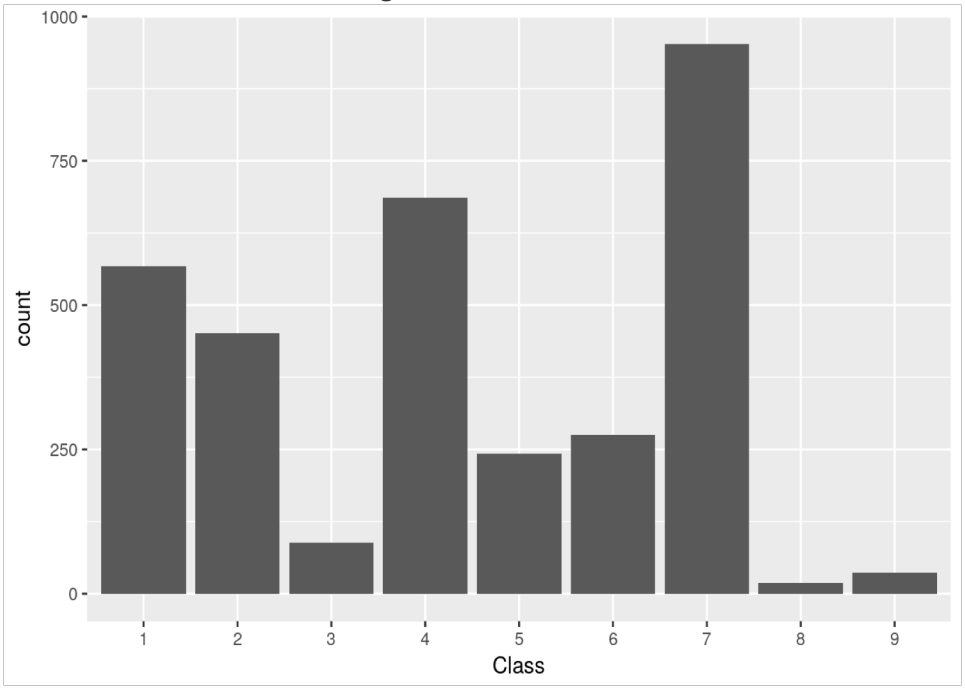
```
test %>%
  group_by(Variation) %>%
  summarise(ct = n()) %>%
  arrange(desc(ct))
## # A tibble: 5,628 x 2
##             Variation    ct
##                 <fctr> <int>
##  1 Truncating Mutations    18
##  2           Deletion    14
##  3       Amplification     8
##  4             Fusions     3
##  5               G44D     2
##  6               A101V     1
##  7               A1020P     1
##  8               A1028V     1
##  9               A1035V     1
## 10               A1038V     1
## # ... with 5,618 more rows
```

Following are the findings:

- There are 3321 different IDs in the training set containing 264 different Gene expressions with 2996 different Variations. There are 9 different Classes indicated by integer levels.
- The Gene and Variation features contain character strings of various lengths.
- There is 70% more test data than train data. The data description tells us that “Some of the test data is machine-generated to prevent hand labeling.”, which should explain this otherwise curious imbalance.
- There are no missing values in the variants data.
- The most frequent Genes in the train vs test data are complete different. In addition, the test data seems to contain significantly more different Genes and fewer high-frequency Genes than the train data. To some extent, this might be an effect of the added machine-generate entries in the test data (by adding many different random levels). Thereby, the difference in frequency might mirror the true fraction of effective test data over train data.
- In contrast, the most frequent Variations in train vs test are largely identical; although, again, the corresponding frequencies are lower in the test data (by a factor of 5 - 10).

```
train %>%
  ggplot(aes(Class)) +
  geom_bar()
```

Here we see how the *Class* target is distributed in the train data:



Following are the findings:

- Class levels 3, 8, and 9 are notably under-represented.
- Levels 5 and 6 are of comparable, medium-low frequency.
- Levels 1, 2, and 4 are of comparable, medium-high frequency.
- Level 7 is clearly the most frequent one.

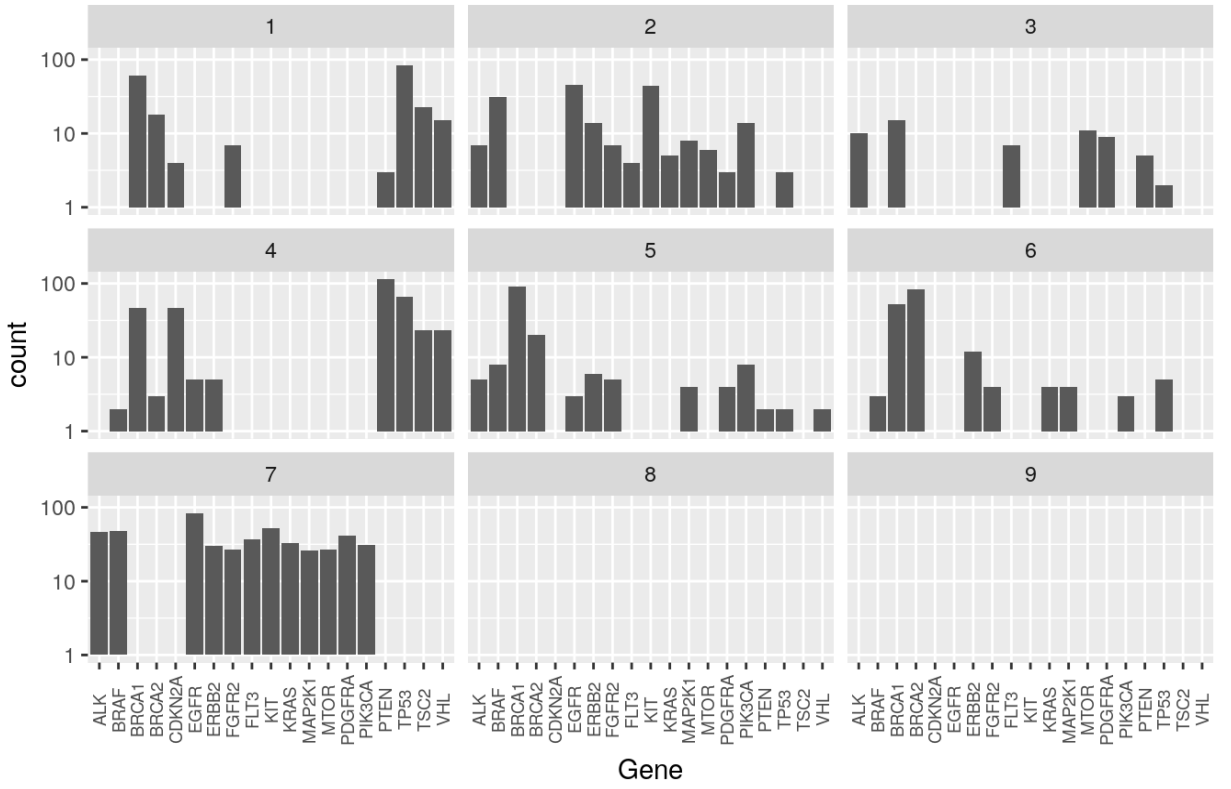
1.3 Feature interactions

Now we want to examine how the features interact with each other and with the target Class variable.

1.3.1 Gene vs Class

First, we will look at the frequency distribution of the overall most frequent Genes for the different Classes. Note the logarithmic frequency scale.

```
train %>%
  filter(Gene %in% str_c(top_gene$Gene)) %>%
  ggplot(aes(Gene)) +
  geom_bar() +
  scale_y_log10() +
  theme(axis.text.x  = element_text(angle=90, vjust=0.5, size=7)) +
  facet_wrap(~ Class)
```

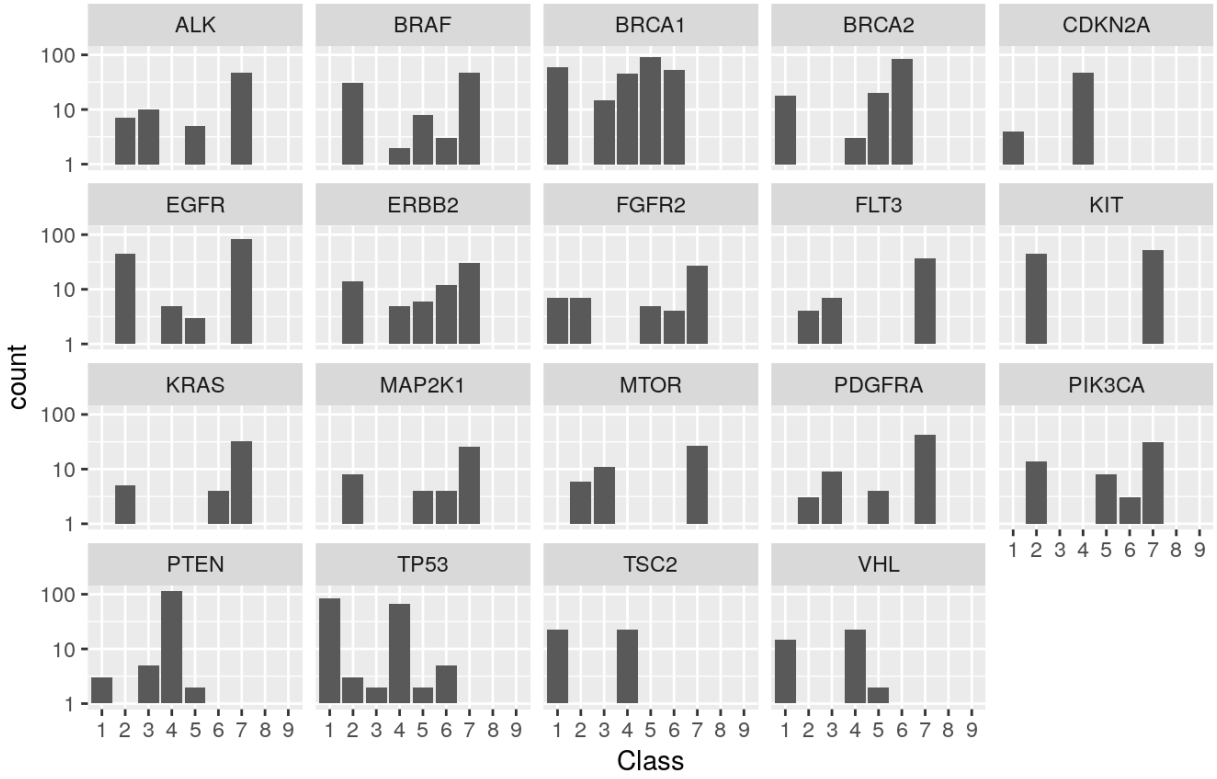


We see immediately that there are significant differences:

- Some Genes, like “PTEN”, are predominantly present in a single Class (here: 4).
- Other Genes, like “TP53”, are mainly shared between 2 classes (here: 1 and 4).
- Classes 8 and 9 contain none of the most frequent Genes.

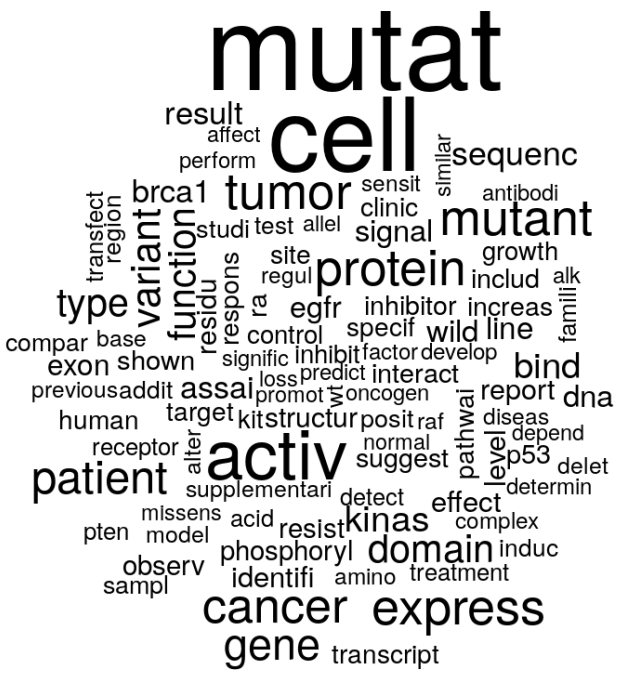
Here's what it looks like for the Classes sorted by Genes (again log counts):

```
train %>%
  filter(Gene %in% str_c(top_gene$Gene)) %>%
  ggplot(aes(Class)) +
  geom_bar() +
  scale_y_log10() +
  facet_wrap(~ Gene)
```



We see immediately that there are significant differences:

- Some Genes, like "PTEN", are predominantly present in a single Class (here: 4).
- Other Genes, like "TP53", are mainly shared between 2 classes (here: 1 and 4).
- Classes 8 and 9 contain none of the most frequent Genes.



With Word cloud We can see the most frequent terms in the text data.

1.4 Model – XGB Model

I have used XGBoost to build a model for this project. XGBoost is the most popular machine learning algorithm these days. Regardless of the data type (regression or classification), it is well known to provide better solutions than other ML algorithms.

```
# setup working directory
setwd("/Users/tyagraj/desktop/Project2")
# load the following libraries
library(data.table)
library(Matrix)
library(xgboost)
library(caret)
library(stringr)
library(tm)
library(syuzhet)

# LabelCount Encoding function
labelCountEncoding <- function(column){
  return(match(column,levels(column)[order(summary(column,maxsum=nlevels(column)))]))
}

# Load Text files
train_text <- do.call(rbind,strsplit(readLines('training_text'),'||',fixed=T))
train_text <- as.data.table(train_text)
train_text <- train_text[-1,]
colnames(train_text) <- c("ID", "Text")
train_text$ID <- as.numeric(train_text$ID)
test_text <- do.call(rbind,strsplit(readLines('test_text'),'||',fixed=T))
test_text <- as.data.table(test_text)
test_text <- test_text[-1,]
colnames(test_text) <- c("ID", "Text")
test_text$ID <- as.numeric(test_text$ID)

# Load Variant Files
train <- fread("training_variants", sep=",", stringsAsFactors = T)
test  <- fread("test_variants", sep=",", stringsAsFactors = T)
# Merging Vairant and Text Files
train <- merge(train,train_text,by="ID")
test  <- merge(test,test_text,by="ID")
# Removing the files that are not required
rm(test_text,train_text);gc()
```

```
# Adding a dummy Class to test variable
test$Class <- -1

# Binding train and test
data <- rbind(train,test)

# Removing the files that are not required
rm(train,test);gc()

# Basic text features
data$nchar <- as.numeric(nchar(data$Text))
data$nwords <- as.numeric(str_count(data$Text, "\\S+"))

# TF-IDF
txt <- Corpus(VectorSource(data$Text))
txt <- tm_map(txt, content_transformer(tolower))
txt <- tm_map(txt, removePunctuation)
txt <- tm_map(txt, removeWords, stopwords("english"))
txt <- tm_map(txt, stemDocument, language="english")
txt <- tm_map(txt, removeNumbers)
txt <- tm_map(txt, stripWhitespace)

dtm <- DocumentTermMatrix(txt, control = list(weighting = weightTfIdf))
dtm <- removeSparseTerms(dtm, 0.95)
data <- cbind(data, as.matrix(dtm))

# LabelCount Encoding for Gene and Variation
data$Gene <- labelCountEncoding(data$Gene)
data$Variation <- labelCountEncoding(data$Variation)

# Sentiment analysis
sentiment <- get_nrc_sentiment(data$Text)
data <- cbind(data,sentiment)

# Set seed
set.seed(1012)
cvFoldsList <- createFolds(data$Class[data$Class > -1], k=5, list=TRUE, returnTrain=FALSE)

# To sparse matrix
varnames <- setdiff(colnames(data), c("ID", "Class", "Text"))
train_sparse <- Matrix(as.matrix(sapply(data[Class > -1, varnames, with=FALSE],as.numeric)),
sparse=TRUE)
test_sparse <- Matrix(as.matrix(sapply(data[Class == -1, varnames,
with=FALSE],as.numeric)), sparse=TRUE)
y_train <- data[Class > -1,Class]-1
test_ids <- data[Class == -1,ID]
```

```
dtrain <- xgb.DMatrix(data=train_sparse, label=y_train)
dtest <- xgb.DMatrix(data=test_sparse)

# Params for xgboost
param <- list(booster = "gbtree",
            objective = "multi:softprob",
            eval_metric = "mlogloss",
            num_class = 9,
            eta = .2,
            gamma = 1,
            max_depth = 5,
            min_child_weight = 1,
            subsample = .7,
            colsample_bytree = .7
)
# Cross validation - for determining CV scores & optimal amount of rounds
xgb_cv <- xgb.cv(data = dtrain,
               params = param,
               nrounds = 1000,
               maximize = FALSE,
               prediction = TRUE,
               folds = cvFoldsList,
               print_every_n = 5,
               early_stopping_round = 100)
rounds <- which.min(xgb_cv$evaluation_log[, test_mlogloss_mean])
# Train model
xgb_model <- xgb.train(data = dtrain,
                     params = param,
                     watchlist = list(train = dtrain),
                     nrounds = rounds,
                     verbose = 1,
                     print_every_n = 5)
# Feature importance
names <- dimnames(train_sparse)[[2]]
importance_matrix <- xgb.importance(names,model=xgb_model)
xgb.plot.importance(importance_matrix[1:30,],20)
# Predict and output csv
preds <- as.data.table(t(matrix(predict(xgb_model, dtest), nrow=9, ncol=nrow(dtest))))
colnames(preds) <-
c("class1","class2","class3","class4","class5","class6","class7","class8","class9")
write.table(data.table(ID=test_ids, preds), "submission.csv", sep=",", dec=".", quote=FALSE,
row.names=FALSE)
```