Classification on the Kaggle Dataset Medals Data Set

Sam Ofiaza

The Kaggle Dataset Medals data set contains information on datasets including the medal it has received (if any) and how many votes, views, downloads, etc. it has.

Credit to Niek van der Zwaag for the data set (link (https://www.kaggle.com/datasets/niekvanderzwaag/kaggle-dataset-medals?select=dataset_medal_total.csv)).

Linear Models for Classification Overview

As opposed to linear models for regression, linear models for classification are those whose target is qualitative. The overall goal for these models is not to find a line of best fit, but to find a line that divides classes. These models have probabilistic results that are easy to interpret and are easy to calculate, update, and scale. However, they are inflexible and simple compared to other classification algorithms. They won't perform well if the data deviates too much from their simplistic assumptions. The two linear models for classification used in this notebook are Logistic Regression and Naive Bayes.

```
df <- read.csv("./dataset_medal_total.csv")
df <- df[, c(2, 5, 6, 7, 8)]
df$Medal <- factor(df$Medal, levels=c("None", "Bronze", "Silver", "Gold"))
str(df)
```

```
## 'data.frame':    42955 obs. of  5 variables:
##  $ Medal         : Factor w/ 4 levels "None","Bronze",..: 3 4 3 4 2 4 2 2 2 3 ...
##  $ Views         : int  69099 233338 63116 335275 18782 274664 18768 14891 25631 41586 ...
##  $ Votes         : int  96 513 254 893 58 688 41 31 36 82 ...
##  $ Votes_Advanced: int  29 90 47 158 16 135 6 11 6 21 ...
##  $ Downloads     : int  4466 25261 9294 60607 1895 40130 1903 1761 1415 2853 ...
```

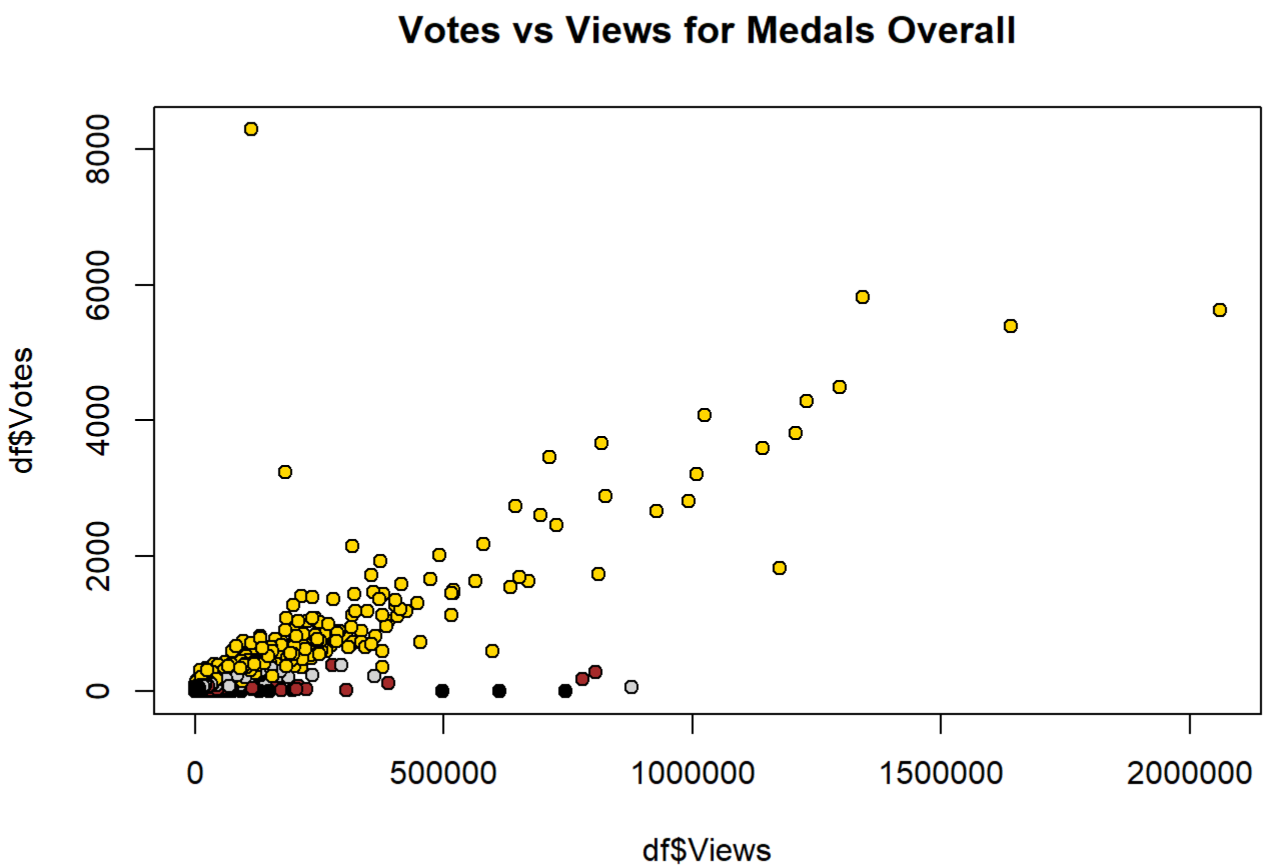
Data Summary

```
str(df)
```

```
## 'data.frame':    42955 obs. of  5 variables:
##  $ Medal         : Factor w/ 4 levels "None","Bronze",..: 3 4 3 4 2 4 2 2 2 3 ...
##  $ Views         : int  69099 233338 63116 335275 18782 274664 18768 14891 25631 41586 ...
##  $ Votes         : int  96 513 254 893 58 688 41 31 36 82 ...
##  $ Votes_Advanced: int  29 90 47 158 16 135 6 11 6 21 ...
##  $ Downloads     : int  4466 25261 9294 60607 1895 40130 1903 1761 1415 2853 ...
```

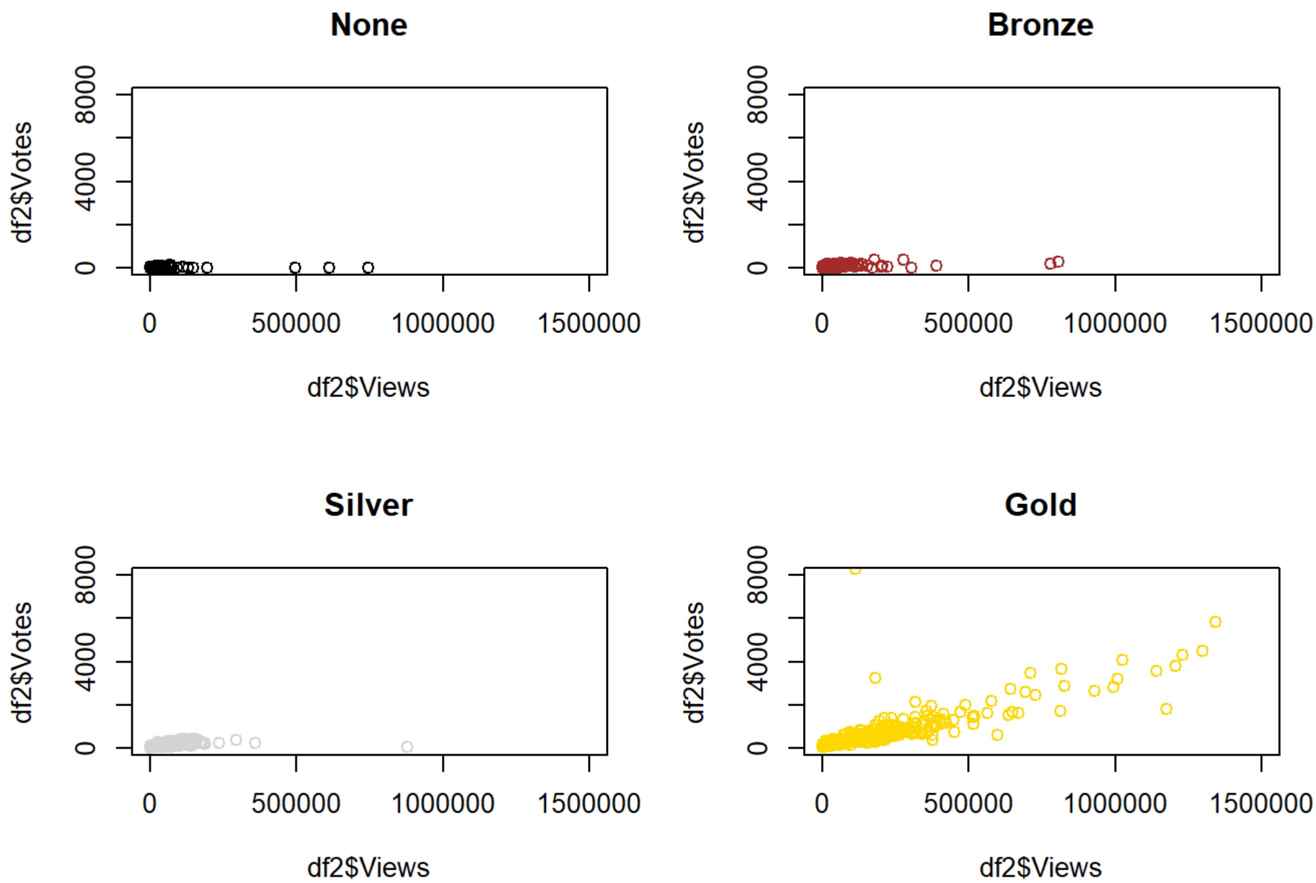
Data Visualization

```
plot(df$Views, df$Votes, pch=21, bg=c("black", "brown", "lightgray", "gold")[unclass(df$Medal)],
main="Votes vs Views for Medals Overall")
```

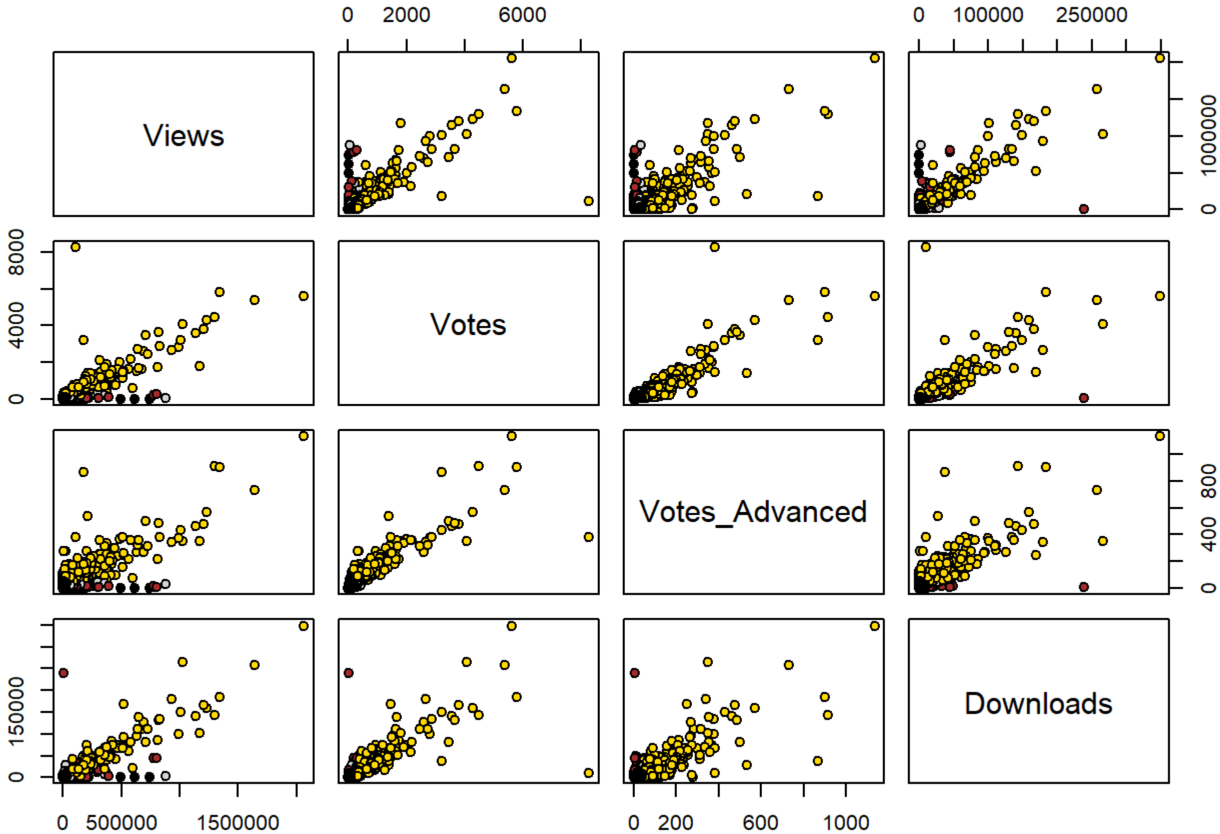


Clearer View of each Medal's Distribution

```
par(mfrow=c(2,2))
df2 <- df[df$Medal == "None",]
plot(df2$Views, df2$Votes, main="None", col="black", xlim=c(0,1500000), ylim=c(0,8000))
df2 <- df[df$Medal == "Bronze",]
plot(df2$Views, df2$Votes, main="Bronze", col="brown", xlim=c(0,1500000), ylim=c(0,8000))
df2 <- df[df$Medal == "Silver",]
plot(df2$Views, df2$Votes, main="Silver", col="lightgray", xlim=c(0,1500000), ylim=c(0,8000))
df2 <- df[df$Medal == "Gold",]
plot(df2$Views, df2$Votes, main="Gold", col="gold", xlim=c(0,1500000), ylim=c(0,8000))
```




```
pairs(df[2:5], pch=21, bg=c("black", "brown", "lightgray", "gold")[unclass(df$Medal)])
```



Overall Medal Distribution

```
table(df$Medal)/nrow(df)
```

```
##
##      None    Bronze    Silver      Gold
## 0.7949715 0.1676871 0.0264230 0.0109184
```

Creation of One vs Many sub data sets

```
gold <- df
gold$Medal <- as.factor(ifelse (df$Medal=="Gold", 1, 0))

silver <- df
silver$Medal <- as.factor(ifelse (df$Medal=="Silver", 1, 0))

bronze <- df
bronze$Medal <- as.factor(ifelse (df$Medal=="Bronze", 1, 0))

none <- df
none$Medal <- as.factor(ifelse (df$Medal=="None", 1, 0))

fun1 <- function(df, i) {
  train <- df[i,]
  glm1 <- glm(Medal~., data=train, family="binomial")
  glm1
}

fun2 <- function(glm, df, i) {
  test <- df[-i,]
  probs <- predict(glm1, newdata=test)
  pred <- ifelse(probs>0.5, 1, 0)
  acc <- mean(pred==test$Medal)
  print(paste("accuracy = ", acc))
  table(pred, test$Medal)
}
```

Divide into 80/20 train/test

```
set.seed(4321)
i <- sample(1:nrow(df), nrow(df)*0.8, replace=FALSE)
```

Creation of Linear Models for each Medal Type

```
glm1 <- fun1(gold, i)
```

```
## Warning: glm.fit: algorithm did not converge
```

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

```
summary(glm1)
```

```
##
## Call:
## glm(formula = Medal ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.005466   0.000000   0.000000   0.000000   0.004098
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1.204e+03  6.493e+03  -0.185    0.853
## Views           5.189e-05  7.519e-03   0.007    0.994
## Votes          -2.339e-02  2.570e+00  -0.009    0.993
## Votes_Advanced  2.437e+01  1.317e+02   0.185    0.853
## Downloads      -2.269e-05  2.761e-02  -0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4.0801e+03  on 34363  degrees of freedom
## Residual deviance: 1.7358e-04  on 34359  degrees of freedom
## AIC: 10
##
## Number of Fisher Scoring iterations: 25
```

```
glm2 <- fun1(silver, i)
```

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

```
summary(glm2)
```

```
##
## Call:
## glm(formula = Medal ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.1970  -0.1879  -0.1788   4.0686
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -4.112e+00  4.217e-02 -97.517  < 2e-16 ***
## Views           8.078e-06  1.491e-06   5.418 6.03e-08 ***
## Votes          -1.879e-02  1.331e-03 -14.110  < 2e-16 ***
## Votes_Advanced  1.024e-01  3.892e-03  26.300  < 2e-16 ***
## Downloads       2.195e-05  1.988e-05   1.104     0.27
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8289.1  on 34363  degrees of freedom
## Residual deviance: 7071.4  on 34359  degrees of freedom
## AIC: 7081.4
##
## Number of Fisher Scoring iterations: 7
```

```
glm3 <- fun1(bronze, i)
```

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

```
summary(glm3)
```

```
##
## Call:
## glm(formula = Medal ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.0119  -0.5656  -0.5476  -0.5280   3.4085
##
## Coefficients:
##                  Estimate Std. Error  z value Pr(>|z|)
## (Intercept)    -1.842e+00  1.684e-02 -109.411  < 2e-16 ***
## Views           9.268e-06  1.809e-06    5.123 3.01e-07 ***
## Votes          -2.459e-02  1.320e-03  -18.630  < 2e-16 ***
## Votes_Advanced  1.088e-01  3.538e-03   30.753  < 2e-16 ***
## Downloads       3.481e-05  1.705e-05    2.041   0.0412 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 31090  on 34363  degrees of freedom
## Residual deviance: 29560  on 34359  degrees of freedom
## AIC: 29570
##
## Number of Fisher Scoring iterations: 6
```

```
glm4 <- fun1(none, i)
```

```
## Warning: glm.fit: algorithm did not converge
```

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

```
summary(glm4)
```

```
##
## Call:
## glm(formula = Medal ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
##  -8.49    0.00    0.00    0.00    8.49
##
## Coefficients:
##                  Estimate Std. Error    z value Pr(>|z|)
## (Intercept)     6.303e+14  3.783e+05  1.666e+09   <2e-16 ***
## Views          -1.578e+09  3.046e+01 -5.182e+07   <2e-16 ***
## Votes           9.613e+12  8.140e+03  1.181e+09   <2e-16 ***
## Votes_Advanced -1.521e+14  4.426e+04 -3.436e+09   <2e-16 ***
## Downloads      -3.491e+10  2.408e+02 -1.450e+08   <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: 34820  on 34363  degrees of freedom
## Residual deviance: 21194  on 34359  degrees of freedom
## AIC: 21204
##
## Number of Fisher Scoring iterations: 25
```

Logistic Regression Model Summary Analysis

The estimate column in the coefficients table indicates the change in log odds for each predictor in receiving the model's corresponding positive class (its specific medal). The four summaries correspond to the four classes gold, silver, bronze, and none respectively. For example, in the above summary for receiving no medals, receiving one more view will decrease the log odds of receiving no medal by -1.578e9. Each additional vote will increase the log odds of receiving no medal by 9.613e12. The p-value for each predictor in each model is significant in determining its model's medal except every predictor in gold's model and Downloads in silver and bronze's models. A lower residual deviance value compared to the null deviance value indicates that adding the predictors helped the model compared to just having the intercept. It seems that this occurred in every model, particularly in gold's case where the difference is several orders of magnitude higher compared to the rest. All of this indicates that the models are all good.

Creation of Naive Bayes Model

```
library(e1071)
train <- df[i,]
test <- df[-i,]
nb1 <- naiveBayes(train[,2:5], train[,1], laplace=laplace)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = train[, 2:5], y = train[, 1], laplace = laplace)
##
## A-priori probabilities:
## train[, 1]
##       None     Bronze     Silver       Gold
## 0.79545455 0.16779188 0.02601560 0.01073798
##
## Conditional probabilities:
##           Views
## train[, 1]       [,1]       [,2]
##     None     1349.465   7278.995
##     Bronze   6635.063  20358.373
##     Silver  25885.208  45253.719
##     Gold   149020.005 221021.073
##
##           Votes
## train[, 1]       [,1]       [,2]
##     None     3.527309   5.100222
##     Bronze  22.068505  24.463222
##     Silver  86.185682  74.700950
##     Gold   544.441734 821.098884
##
##           Votes_Advanced
## train[, 1]       [,1]       [,2]
##     None     1.156759   1.137269
##     Bronze   8.778529   3.682692
##     Silver  29.060403   7.841282
##     Gold   115.387534 118.427073
##
##           Downloads
## train[, 1]       [,1]       [,2]
##     None     128.2520   354.6392
##     Bronze   783.3562  1932.5280
##     Silver  3273.6577  4628.8914
##     Gold   21531.6829 32937.9815
```

Naive Bayes Model Summary Analysis

The A-priori probabilities are the estimated probabilities of each class before using any predictor. Unsurprisingly, receiving no medal is most likely for most datasets. Each conditional probability table for each predictor indicates how the binomial distribution of that predictor appears. The columns [,1] and [,2] are the mean and variance, respectively.

Calculation of Metrics for both the Logistic Regression and Naive Bayes Models

```
fun2(glm1, gold, i)
```

```
## [1] "accuracy =  1"
```

```
##
## pred    0    1
##    0 8491    0
##    1    0  100
```

```
fun2(glm2, silver, i)
```

```
## [1] "accuracy =  0.960307298335467"
```

```
##
## pred    0    1
##    0 8250  241
##    1  100    0
```

```
fun2(glm3, bronze, i)
```

```
## [1] "accuracy =  0.821091840297986"
```

```
##
## pred    0    1
##    0 7054 1437
##    1  100    0
```

```
fun2(glm4, none, i)
```

```
## [1] "accuracy =  0.195320684437202"
```

```
##
## pred    0    1
##    0 1678 6813
##    1  100    0
```

```
p1 <- predict(nb1, newdata=test)
A <- table(p1, test$Medal)
A[order(A[,1], decreasing=TRUE), order(A[,1], decreasing=TRUE)]
```

```
##
## p1       None Bronze Silver Gold
##   None   6665    220      0   22
##   Bronze  144   1173     29    0
##   Silver    3     42    204   23
##   Gold      1      2      8   55
```

Analysis of Prediction Results on Test Data

The one vs many approach used in Logistic Regression worked perfectly for gold, very well for silver, decent for bronze, and horribly for none based on accuracy and the confusion matrices. The confusion matrix for none in particular contains a high amount of false positives. The Naive Bayes results seem better overall by losing accuracy for gold and silver and gaining accuracy for none and bronze compared to the results for logistic regression. I believe that Logistic Regression performed poorly because it was unable to distinguish clear boundaries between the none class, which is ~80% of the data, and the other classes.

Strengths and Weaknesses of Logistic Regression and Naive Bayes

Logistic Regression has easy to interpret probabilistic results and can be optimized to reduce error with gradient descent. It is better than Naive Bayes when used with large data sets. It does not do well when the classes are muddied together and cannot be cleanly separated with linear boundaries. Naive Bayes works very well for being so simple and can be easy to implement despite the fact that its assumption of independent predictors is often untrue. However, its results can be beaten with more complex models. It has higher bias and lower variance compared to Logistic Regression.

Benefits and Drawbacks of the Classification Metrics

Accuracy is how often the model correctly predicts the data and is simple and commonly used. The confusion matrix goes into more detail with how often true positives, false positives, true negatives, and false negatives were calculated and can give a better picture as to the performance of the model.