

# PUBG Top 10% Placement Analysis

*Chance Robinson, Allison Roderick and William Arnost*

*Master of Science in Data Science, Southern Methodist University, USA*

## 1 Introduction

PlayerUnknown's Battlegrounds, colloquially referred to as PUBG, is an online battle royale-style shooter that is one of the best-selling video games of all time. The game's developer has created an API (application program interface) where others have collected data on over 65,000 matches and made that data available on the popular online competition site Kaggle. The project team will be utilizing logistic regression, linear and quadratic discriminant analyses, and random forest methods to attempt to create the best model for predicting whether or not an individual player in a solo match will place in the top 10 finishers or not.

## 2 Data Description

The source data is available on Kaggle.com under the competition PUBG Finish Placement Prediction. The files used in our analysis were transformed to fit the requirements of a binomial logistic regression classifier. The data has also been pre-split into training and test files for consistency when comparing our different model types. Additionally, as the percentage-based nature of the top 10% of players is inherently unbalanced, we've also downsampled the higher frequency data to match that over the lower frequency outcome of the top 10% of players.

`pubg_solo_game_types.csv`

- Filtered for solo only game types

`pubg_solo_game_types_test_full.csv`

- Pre-split for test data

`pubg_solo_game_types_train_full.csv`

- Pre-split for train data without downsampling for the unbalanced response variable

`pubg_solo_game_types_train_downsampled.csv`

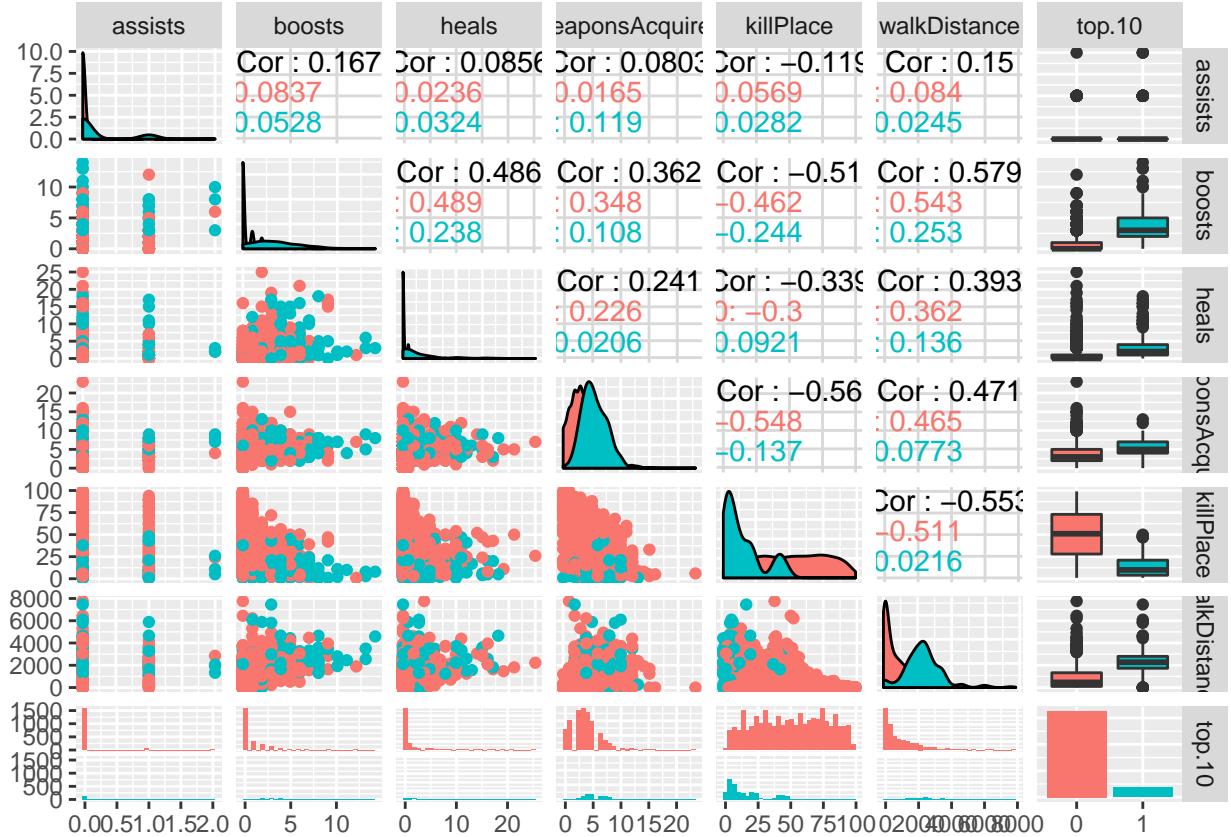
- Pre-split for train data with downsampling for the unbalanced response variable

### 2.1 Data Dictionary

| Column Name     | Description  |
|-----------------|--|
| DBNOs           | Number of enemy players knocked.   |
| assists         | Number of enemy players this player damaged that were killed by teammates.   |
| boosts          | Number of boost items used.  |
| damageDealt     | Total damage dealt. Note: Self inflicted damage is subtracted.   |
| headshotKills   | Number of enemy players killed with headshots.   |
| heals           | Number of healing items used.  |
| Id              | Players Id   |
| killPlace       | Ranking in match of number of enemy players killed.  |
| killPoints      | Kills-based external ranking of player. (Think of this as an Elo ranking where only kills matter.) If there is a value other than -1 in rankPoints, then any 0 in killPoints should be treated as a None.          |
| killStreaks     | Max number of enemy players killed in a short amount of time.  |
| kills           | Number of enemy players killed.  |
| longestKill     | Longest distance between player and player killed at time of death.  |
| matchDuration   | Duration of match in seconds.  |
| matchId         | ID to identify match. There are no matches that are in both the training and testing set.  |
| matchType       | String identifying the game mode that the data comes from.   |
| rankPoints      | Elo-like ranking of player.  |
| revives         | Number of times this player revived teammates.   |
| rideDistance    | Total distance traveled in vehicles measured in meters.  |
| roadKills       | Number of kills while in a vehicle.  |
| swimDistance    | Total distance traveled by swimming measured in meters.  |
| teamKills       | Number of times this player killed a teammate.   |
| vehicleDestroys | Number of vehicles destroyed.  |
| walkDistance    | Total distance traveled on foot measured in meters.  |
| weaponsAcquired | Number of weapons picked up.   |
| winPoints       | Win-based external ranking of player. (Think of this as an Elo ranking where only winning matters.) If there is a value other than -1 in rankPoints, then any 0 in winPoints should be treated as a None.          |
| groupId         | ID to identify a group within a match. If the same group of players plays in different matches, they will have a different groupId each time.  |
| numGroups       | Number of groups we have data for in the match.  |
| maxPlace        | Worst placement we have data for in the match. This may not match with numGroups, as sometimes the data skips over placements.   |
| winPlacePerc    | This is a percentile winning placement, where 1 corresponds to 1st place, and 0 corresponds to last place in the match. (to be removed from our binomial classifier so as not to influence our predictive results) |
| top.10          | The target of prediction. This is a percentile winning placement, where 1 corresponds to a top 10% placement a 0 in the lower 90%.   |

## 2.2 Exploratory Data Analysis

The data full data set has 65,000 matches, each with data on the players in that match and statistics based on that individual match with only minor exceptions. We have data on how much damage was dealt, how many kills a player got, and other information like distance travelled and weapons acquired. Below is a pairs plot for some of our core variables.



A complete set of pair plots and box plots are available in the EDA appendix.

A number of variables are not useful for prediction. Id, groupID, and matchId are identifiers. MatchType, DBNOs, and Revives are not relevant because we are focusing on the Solo match type. matchDuration, maxPlace, and numGroups are static across any given match, even if they vary individually, so there is no information about individual player performance. Finally, our response variable, top.10, is derived from winPlacePerc, so we cannot use it for prediction.

### 2.2.1 PCA

Principle Components Analysis can help us reduce numerical predictors to a few components. These groupings might help us find key variables to use for modeling

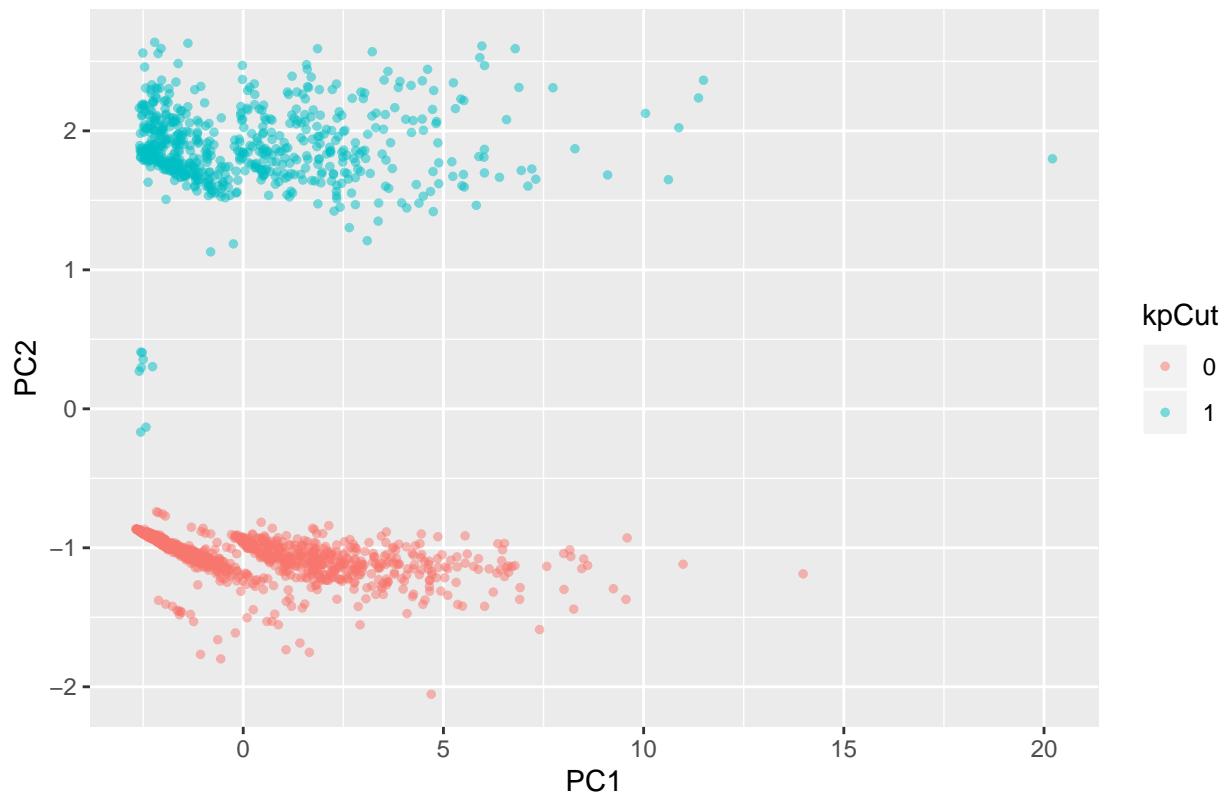
```
##          PC1        PC2        PC3
## assists  0.093464896  0.0341204724 -0.07248995
```

```

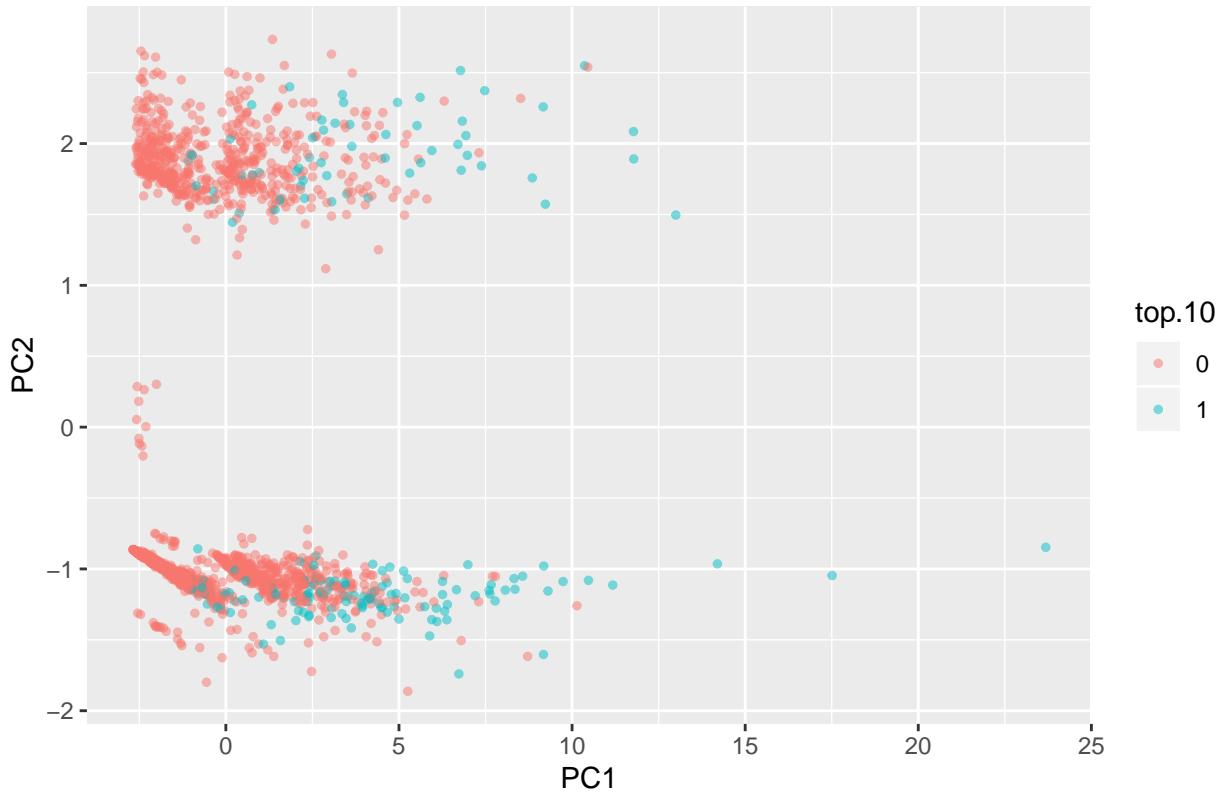
## boosts          0.284017732 -0.0187724920 -0.27993293
## damageDealt    0.354025626 -0.0006752358  0.16342595
## headshotKills  0.272432205  0.0058796339  0.19810172
## heals           0.172891815 -0.0314204736 -0.36832274
## killPlace       -0.328110631  0.0417139011  0.05106409
## killPoints      0.025713144  0.7026339764 -0.04743269
## kills           0.366152434  0.0064308394  0.19503487
## killStreaks     0.320309798  0.0146912776  0.20189572
## longestKill     0.282653385 -0.0104638751  0.06734322
## rideDistance    0.125792347 -0.0328766270 -0.36531789
## roadKills       0.058565079  0.0099295744  0.01992678
## swimDistance    0.038039360 -0.0055821852 -0.19718801
## teamKills       0.005310582 -0.0529793299 -0.10953494
## vehicleDestroys 0.054284130 -0.0255527144 -0.10946179
## walkDistance    0.245857081 -0.0176610089 -0.41203730
## weaponsAcquired 0.194128985 -0.0430771084 -0.35785321
## winPoints        0.015164472  0.7032363836 -0.05351414
## killsPK          0.327801875  0.0166732303  0.33144634
## damageKill       0.163661873 -0.0015941934  0.13894088

```

PCA of PUBG



## PCA of PUBG



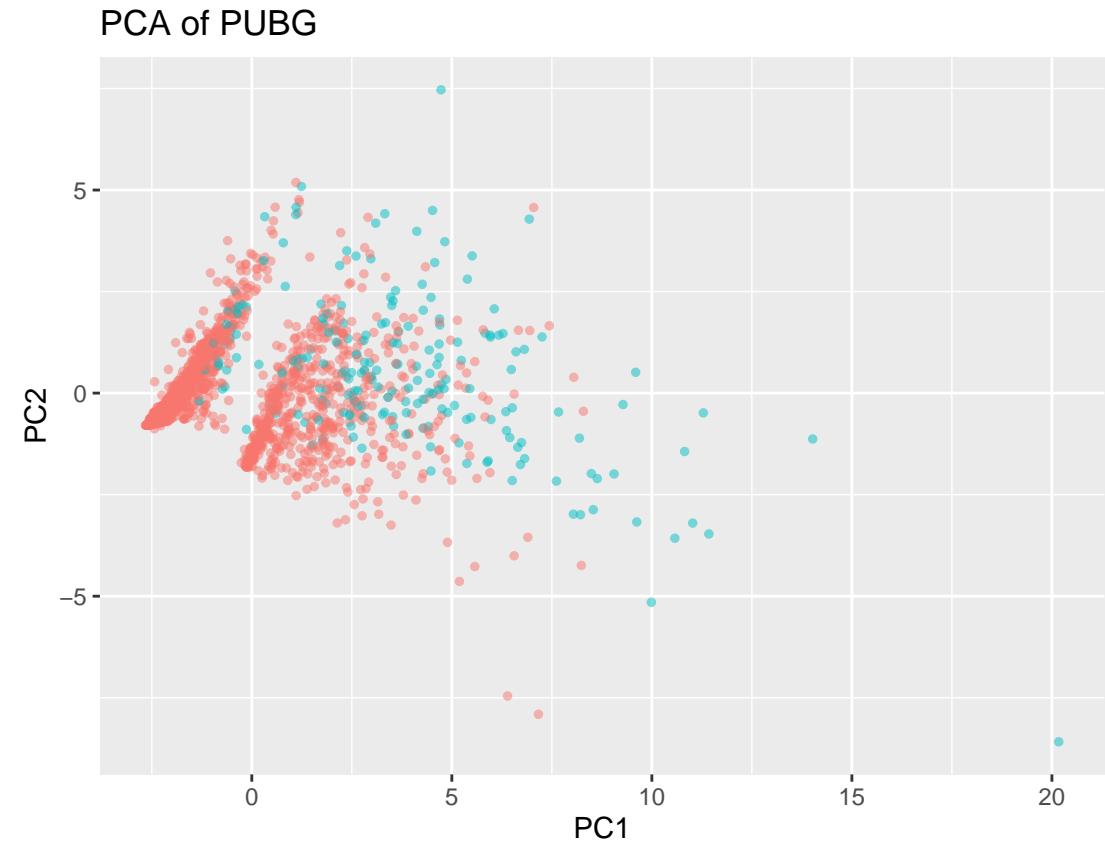
We see kills based measures are pretty strong in PC1, and killPoints/Winpoinnts dominate PC2. Looking at the graph, we get a strange separation. The killPoints variable can explain this, with one group having killPoints = 0 and the other being killPoints > 0. Lets try it one more time without killpoints.

|                    | PC1          | PC2         | PC3         |
|--------------------|--------------|-------------|-------------|
| ## assists         | 0.093288398  | 0.06828606  | 0.04474120  |
| ## boosts          | 0.284134067  | 0.28016254  | 0.07277247  |
| ## damageDealt     | 0.354186391  | -0.16308792 | -0.01079849 |
| ## headshotKills   | 0.272532648  | -0.19811583 | 0.01739273  |
| ## heals           | 0.173041804  | 0.36948689  | 0.05346143  |
| ## killPlace       | -0.328486849 | -0.05445813 | -0.06225663 |
| ## kills           | 0.366260701  | -0.19513648 | -0.01676471 |
| ## killStreaks     | 0.320325015  | -0.20251783 | -0.03788582 |
| ## longestKill     | 0.282828105  | -0.06637903 | 0.02458407  |
| ## rideDistance    | 0.125910114  | 0.36697769  | -0.30903984 |
| ## roadKills       | 0.058526280  | -0.02081518 | -0.39810562 |
| ## swimDistance    | 0.038043527  | 0.19662593  | 0.24080577  |
| ## teamKills       | 0.005560085  | 0.11493992  | -0.57603084 |
| ## vehicleDestroys | 0.054393599  | 0.11187784  | -0.54205238 |
| ## walkDistance    | 0.245894336  | 0.41172283  | 0.17098240  |
| ## weaponsAcquired | 0.194340503  | 0.36034996  | 0.10634883  |

```

## killsPK          0.327856167 -0.33177097 -0.07024240
## damageKill      0.163742912 -0.13812684  0.01166277

```



We are still getting some separation, but it has much improved. We are not getting 100\$ separation of the top.10 variable, so PCA alone is not giving us a clear answer as to what makes a top 10 player.

## 3 Objective I Analysis

### 3.1 Problem Statement

We would like to predict if a player finished in the top 10 of a PUBG match based on their stats during the match. Our response variable is `top.10`, which is 1 if the player finished in the top 10 and 0 if they did not. We will start with a logistic regression model before moving into more complex methods.

### 3.2 Establishing the Initial Model

During EDA, we discovered a number of variables that would not be useful for prediction, either because they were ID columns or match characteristics that were not relevant to individual player performance. Removing those, our first model contains all the remaining variables.

```

##  

## Call:  

##   glm(formula = top.10 ~ assists + boosts + heals + teamKills +  

##         weaponsAcquired + damageDealt + headshotKills + kills + killStreaks +  

##         roadKills + vehicleDestroys + killPlace + killPoints + rankPoints +  

##         winPoints + longestKill + swimDistance + rideDistance + walkDistance,  

##         family = binomial(link = "logit"), data = train_obj1)  

##  

## Deviance Residuals:  

##       Min      1Q  Median      3Q     Max  

## -3.5064 -0.2621  0.0017  0.4644  5.1751  

##  

## Coefficients:  

##              Estimate Std. Error z value Pr(>|z|)  

## (Intercept) -5.335e+00 4.679e-01 -11.402 < 2e-16 ***  

## assists      6.767e-01 7.031e-02   9.624 < 2e-16 ***  

## boosts       3.279e-01 1.265e-02  25.930 < 2e-16 ***  

## heals        -3.521e-02 6.748e-03 -5.217 1.82e-07 ***  

## teamKills    -2.027e+00 2.315e-01 -8.757 < 2e-16 ***  

## weaponsAcquired 4.750e-02 8.215e-03  5.782 7.36e-09 ***  

## damageDealt   6.655e-05 2.758e-04   0.241  0.80935  

## headshotKills 4.061e-02 3.300e-02   1.231  0.21841  

## kills        -5.179e-02 3.254e-02  -1.591  0.11151  

## killStreaks   -1.210e+00 5.375e-02 -22.509 < 2e-16 ***  

## roadKills     -2.286e-01 1.223e-01  -1.870  0.06148 .  

## vehicleDestroys -4.657e-01 1.674e-01  -2.783  0.00539 **  

## killPlace      -9.337e-02 1.817e-03 -51.380 < 2e-16 ***  

## killPoints     -1.578e-03 2.606e-04  -6.055 1.41e-09 ***  

## rankPoints     3.995e-03 3.056e-04  13.074 < 2e-16 ***  

## winPoints      5.282e-03 3.949e-04  13.376 < 2e-16 ***  

## longestKill    -1.896e-03 4.007e-04  -4.732 2.22e-06 ***  

## swimDistance   1.503e-03 3.705e-04   4.056 5.00e-05 ***  

## rideDistance   1.316e-04 1.011e-05  13.026 < 2e-16 ***  

## walkDistance   8.866e-04 2.384e-05  37.197 < 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: 36895  on 26613  degrees of freedom  

## Residual deviance: 15916  on 26594  degrees of freedom  

## AIC: 15956  

##
```

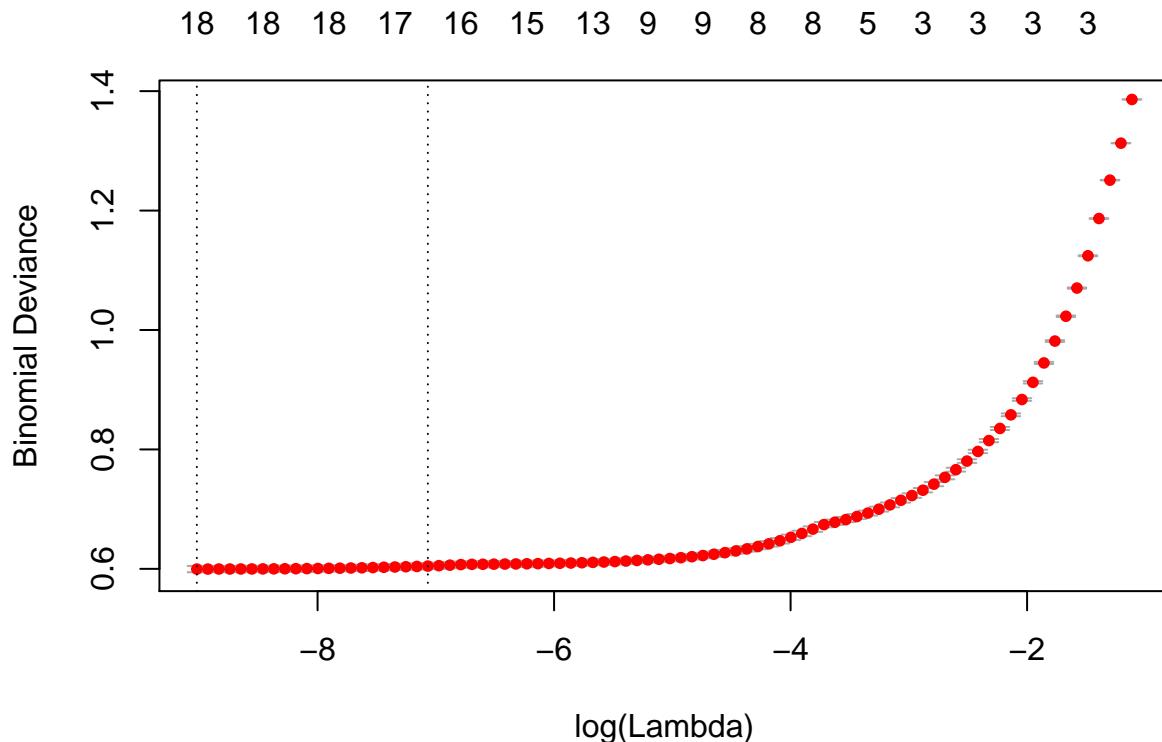
```
## Number of Fisher Scoring iterations: 6
```

We can see some insignificant variables, and the vif (Appendix: Logistic Model) will show some correlation. This model performance of this model is as follows:

| Model   | AUC   | Sensitivity | Specificity | Error Rate (1 - Accuracy) |
|---------|-------|-------------|-------------|---------------------------|
| Initial | 0.941 | 0.891       | 0.843       | 15.2%                     |

### 3.3 Variable Selection

Next, we use lasso to narrow these down to the most important variables. First, we used cv.glmnet to search for the ideal value of lambda for the lasso regression (Appendix: Logistic Regression). We selected the lambda.1se value, which will hopefully eliminate the most variables. Using that value for lambda, lasso produced this output:



```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                               s0
## (Intercept)      -0.6733424558
## (Intercept)      .
## assists          0.6465602204
```

```

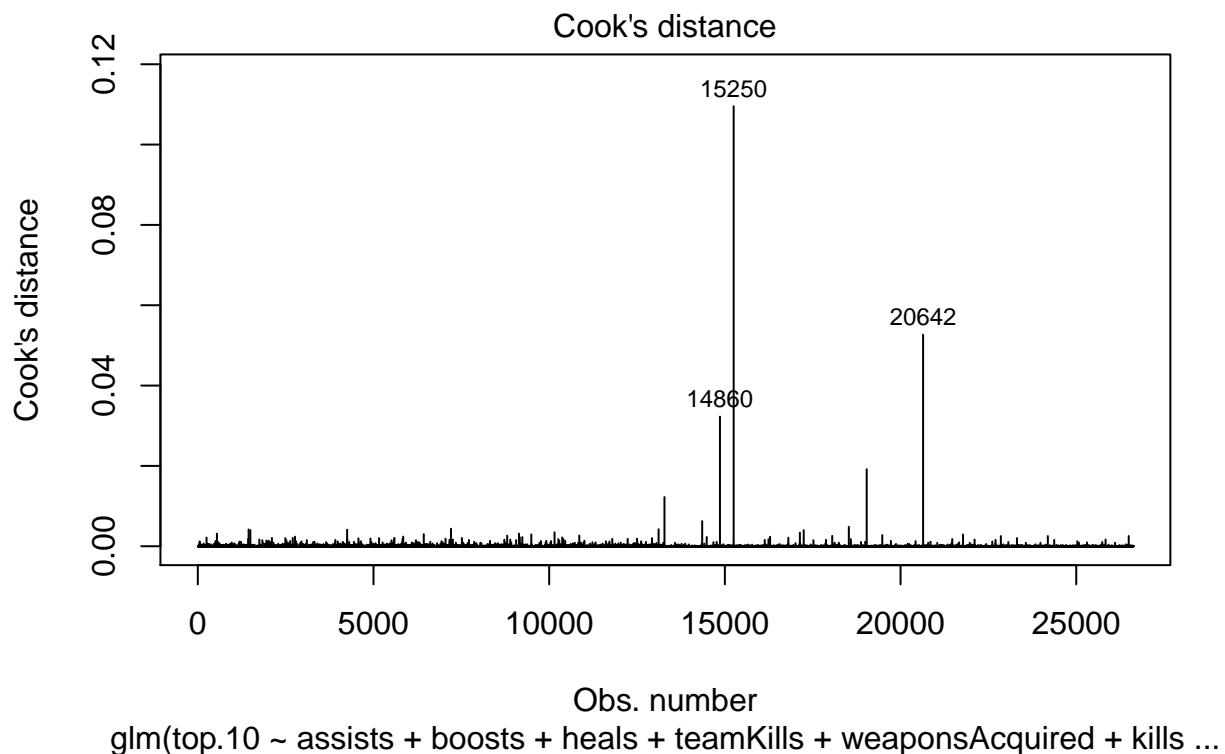
## boosts          0.3247910863
## heals          -0.0291791887
## teamKills      -1.9070440684
## weaponsAcquired 0.0445136944
## damageDealt     .
## headshotKills   .
## kills           -0.0262597549
## killStreaks     -1.1450786717
## roadKills       -0.1971398791
## vehicleDestroys -0.4170561352
## killPlace        -0.0876942534
## killPoints       .
## rankPoints       0.0008663913
## winPoints        0.0008998702
## longestKill      -0.0014955118
## swimDistance     0.0013710884
## rideDistance     0.0001344144
## walkDistance     0.0008663011

```

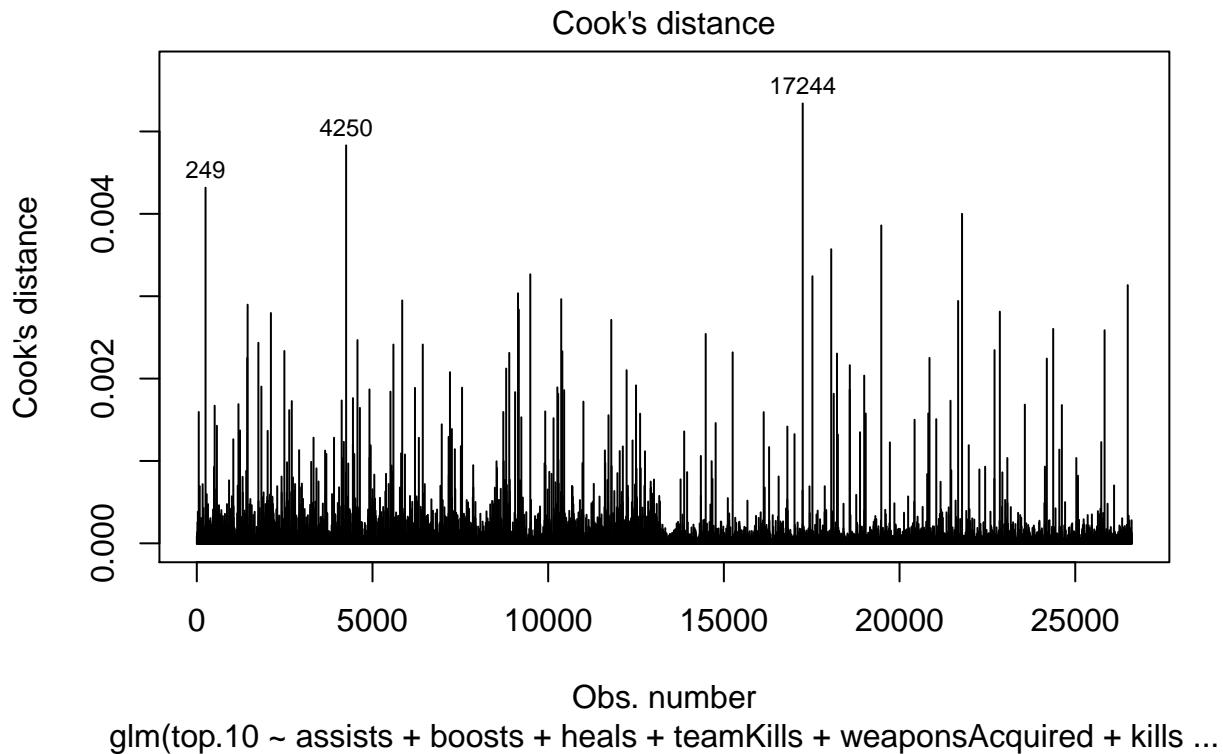
### 3.4 Final Iterations

The selection eliminated damageDealt, headshotKills, and killPoints. We created a model using just the variables from lasso and found two issues. First, winPoints and rankPoints have high VIFs. Removing them individually found that when one is missing, the other is no longer significant. We decided to remove them both. Second, the cooksD plot shows 3 potentially influential points. All three points have high values ( $> p99$  threshold) for roadKills and killStreaks, and I also found through adding and removing variables that swimDistance contributes influential points. We chose to remove these variables, as a very small part of the population has high values in these metrics, and it negatively effects the model.

Here we can see influential points from the Cook's D chart.



After removing roadKills , killStreaks, and swimDistance



Final Model Coefficients

```
##  
## Call:  
## glm(formula = top.10 ~ assists + boosts + heals + weaponsAcquired +  
##       killPlace + walkDistance, family = binomial(link = "logit"),  
##       data = train_obj1)  
##  
## Deviance Residuals:  
##      Min        1Q     Median        3Q       Max  
## -3.3664  -0.3759  -0.0174   0.4865   2.6092  
##  
##  
## Coefficients:  
##                               Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -5.934e-01  5.982e-02 -9.919 < 2e-16 ***  
## assists      6.225e-01  6.666e-02   9.339 < 2e-16 ***  
## boosts       3.067e-01  1.158e-02  26.493 < 2e-16 ***  
## heals        -2.276e-02  6.449e-03 -3.529 0.000418 ***  
## weaponsAcquired 7.900e-02  7.742e-03  10.205 < 2e-16 ***  
## killPlace    -6.216e-02  1.192e-03 -52.172 < 2e-16 ***
```

```

## walkDistance      7.415e-04  2.073e-05  35.775  < 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: 36895  on 26613  degrees of freedom
## Residual deviance: 17365  on 26607  degrees of freedom
## AIC: 17379
##
## Number of Fisher Scoring iterations: 6

```

### 3.5 Comparing Competing Models

| Model       | AUC   | Sensitivity | Specificity | Error Rate (1 - Accuracy) |
|-------------|-------|-------------|-------------|---------------------------|
| Initial     | 0.941 | 0.891       | 0.843       | 15.2%                     |
| Lasso       | 0.941 | 0.892       | 0.843       | 15.2%                     |
| Inf. Points | 0.936 | 0.892       | 0.842       | 15.3%                     |
| Final       | 0.932 | 0.863       | 0.841       | 15.6%                     |

### 3.6 Model Interpretation

We prefer the final model because it simplifies the previous versions without significant loss in accuracy. The variable interpretations are found in the table below.

| Variable        | Interpretation   |
|-----------------|--|
| assists         | The log-odds of placing in the top 10 increase by 0.6225 for each additional assist  |
| boosts          | The log-odds of placing in the top 10 increase by 0.3067 for each additional boost   |
| heals           | The log-odds of placing in the top 10 decrease by -0.0228 for each additional heal   |
| weaponsAcquired | The log-odds of placing in the top 10 increase by 0.0790 for each additional weapon acquired                               |
| killPlace       | The log-odds of placing in the top 10 increase by 0.0622 for each additional rank in kills (lower is better for killPlace) |
| walkDistance    | The log-odds of placing in the top 10 increase by 0.0007 for each additional meter walked                                  |

### 3.7 Conclusion

Players who do well in kill placement and who travel far tend to win games. Travel distance probably is a proxy for time alive, as players eliminated early would probably not travel far. Acquiring in game items like boosts or weapons also help. The negative coefficient on heals is odd, perhaps indicating that taking more damage is detrimental to placement. We will try to improve on the final model using more complex methods.

## 4 Objective II Analysis

### 4.1 Question of Interest

As we stated in the Problem Statement, we would like to predict if a player finished in the top 10 of a PUBG match based on their stats during the match. The next three methods of prediction we will present are Logistic Regression using more complicated terms than what were presented in Objective 1, Linear Discriminant Analysis (LDA), and Random Forest. These different techniques are all going to help us answer the question: What model is has the highest predictive accuracy for classifying players as top 10 finishers?

### 4.2 Model Selection

#### 4.2.1 Logistic Regression with complex terms

##### 4.2.1.1 Objective

In the first logistic regression we used terms that were easier to interpret, now we will add more complex terms. The final model from the previous section was:

top.10 ~ assists + boosts + heals + weaponsAcquired + killPlace + walkDistance

Additional terms considered in the more complex model are as follows: \* Dichotomized variables: Two level versions of the original variables, with cutoffs determined by EDA to try to separate top.10 from the rest of the group

- Binned Variables: These have more levels than the dichotomized versions, to try to capture more complexity
- Ratios: Key variables are compared against each other in ratio format. For instance, does damage dealt per kill make a difference?
- Interactions: multiplicative relationships between key variables. For instance, does increased damage matter if the player is above the boosts cut point?
- Full definitions of new variables can be found in Appendix: Complex Logistic Regression

##### 4.2.1.2 Model Selection

We ran a total of 8 additional logistic models, with the following results:

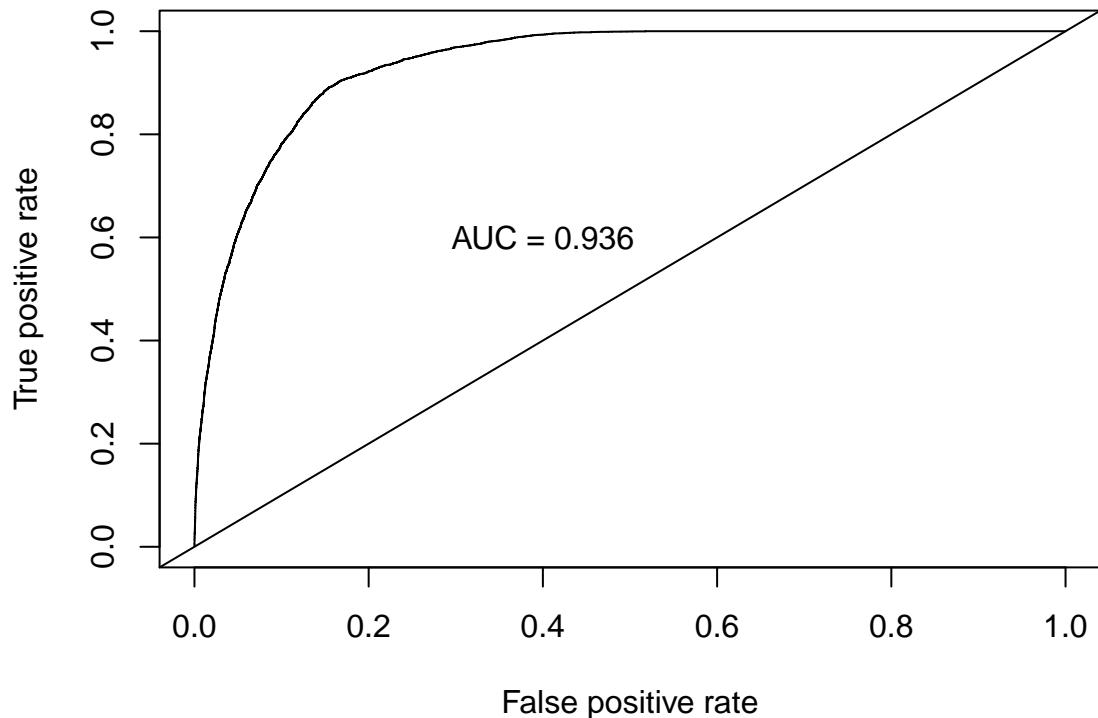
| Model    | AUC   | Sensitivity | Specificity | Error Rate (1 - Accuracy) |
|----------|-------|-------------|-------------|---------------------------|
| Model 01 | 0.932 | 0.873       | 0.836       | 16.0%                     |
| Model 02 | 0.932 | 0.862       | 0.841       | 15.7%                     |
| Model 03 | 0.932 | 0.885       | 0.832       | 16.2%                     |
| Model 04 | 0.936 | 0.889       | 0.827       | 16.6%                     |
| Model 05 | 0.932 | 0.864       | 0.841       | 15.6%                     |

| Model      | AUC   | Sensitivity | Specificity | Error Rate (1 - Accuracy) |
|------------|-------|-------------|-------------|---------------------------|
| Model 06   | 0.939 | 0.888       | 0.836       | 15.8%                     |
| Model 07   | 0.936 | 0.895       | 0.840       | 15.4%                     |
| Model 08   | 0.936 | 0.894       | 0.838       | 15.6%                     |
| Obj1 Final | 0.932 | 0.863       | 0.841       | 15.6%                     |

The most successful model was number 7, which had the best accuracy with no warnings. The model coefficients and performance are in the figures below:

```
##
## Call:
## glm(formula = top.10 ~ boosts + assists + weaponsAcquired + killPlace +
##       walkDistance + killsPK, family = binomial(link = "logit"),
##       data = train_obj1)
##
## Deviance Residuals:
##    Min      1Q   Median      3Q     Max
## -3.5511 -0.3216 -0.0002  0.4906  2.9651
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)            2.866e-01 7.051e-02  4.064 4.82e-05 ***
## boosts                 3.581e-01 1.159e-02 30.896 < 2e-16 ***
## assists                7.086e-01 6.731e-02 10.526 < 2e-16 ***
## weaponsAcquired       6.123e-02 7.994e-03  7.659 1.87e-14 ***
## killPlace              -7.958e-02 1.471e-03 -54.091 < 2e-16 ***
## walkDistance           6.951e-04 2.127e-05 32.683 < 2e-16 ***
## killsPK                -7.938e-01 3.420e-02 -23.213 < 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: 36895  on 26613  degrees of freedom
## Residual deviance: 16839  on 26607  degrees of freedom
## AIC: 16853
##
## Number of Fisher Scoring iterations: 6
```

## Ordinary Logistic



```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction      0      1
##           0 41066    600
##           1  7813   5103
##
##          Accuracy : 0.8459
## 95% CI : (0.8428, 0.8489)
## No Information Rate : 0.8955
## P-Value [Acc > NIR] : 1
##
##          Kappa : 0.4715
##
## McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.89479
## Specificity : 0.84016
## Pos Pred Value : 0.39509
## Neg Pred Value : 0.98560
## Prevalence : 0.10448
```

```

##           Detection Rate : 0.09349
##     Detection Prevalence : 0.23663
##     Balanced Accuracy : 0.86747
##
##     'Positive' Class : 1
##

```

The additional variable for this model is killsPK, or kills per kilometer. It is a ratio designed to separate players who are aggressive and engaging other players during the match vs. those who travel far but play a more stealthy strategy. The model suggests each additional unit of killsPK would decrease the log odds of entering the top 10 by 0.794

#### 4.2.1.3 Conclusions for Logistic Regression

Of the models we tried, none achieved an increase in accuracy that would justify the additional complexity. We also ran into issues with complex variables. The rank deficient error suggest some values of the combination of two categorical variables did not have any observations assigned. In this case it would make sense to code a special variable to account for this. The other warning suggests some combination of variables allowed for perfect separation of top.10, but which variables was not clear from the EDA. For logistic regression, the best model we found was the one at the end of objective one.

### 4.2.2 Linear Discriminant Analysis

Our next prediction tool is Linear Discriminant Analysis (LDA) for classifying the player as Top 10 or not. We have taken a subset of the continuous variables from our EDA to build the LDA off of. Before running the LDA, we will cover two things: a LASSO call to eliminate less important variables and assumption checking.

#### 4.2.2.1 LASSO

The LASSO call plus manual variable selection reduced the predictors considered for the LDA model to: boosts, heals, killPlace, killStreaks, longestKill, matchDuration, rideDistance, swimDistance, teamKills, walkDistance, weaponsAcquired. See see the Appendix - LDA for the output of the LASSO call.

#### 4.2.2.2 Assumption Checking

LDA performs optimally when the assumptions of MANOVA are met. That is,

1. The predictors are normally distributed for each class of the response.
2. The covariance matrices for each class of the response are homogeneous.

When we check the first assumption for the predictors that are to be included in the LDA model, we see that the assumption is not met, as shown in the histograms of the variables as shown in the Appendix - LDA.

Most of the predictors are right skewed. The variable matchDuration is bimodal. To remedy this, we tried transforming the variables, but it did not help our overall prediction accuracy. However, because issues of normality exist, we will explore QDA as well as LDA.

We also checked the homogeneity of correlation matrices, shown in the Appendix - LDA. We find that, overall, there are no major departures from homogeneity. The variables walkDistance and killPlace show the greatest deviances from homogenous correlations between bottom 90 and top 10 placements. Consequently, we tried removing those variables from the model. However, removing those variables reduced our prediction accuracy.

Thus, we will proceed with the variables selected and see if LDA or QDA performs better.

#### 4.2.2.3 LDA Results

LDA has a prediction accuracy of 0.9206, with a sensitivity of 0.57479 and a specificity of 0.96099. The area under the ROC curve is 0.941. Output of the LDA call and the ROC curve are shown in Appendix - LDA. The confusion matrix is shown below for comparison with QDA.

```
## Confusion Matrix and Statistics
##
##
##          0      1
## 0 46972 2425
## 1 1907 3278
##
##          Accuracy : 0.9206
##                 95% CI : (0.9183, 0.9229)
##    No Information Rate : 0.8955
##    P-Value [Acc > NIR] : < 2.2e-16
##
##          Kappa : 0.5582
##
##  Mcnemar's Test P-Value : 3.998e-15
##
##          Sensitivity : 0.57479
##          Specificity : 0.96099
##    Pos Pred Value : 0.63221
##    Neg Pred Value : 0.95091
##          Prevalence : 0.10448
##          Detection Rate : 0.06006
##    Detection Prevalence : 0.09499
##          Balanced Accuracy : 0.76789
##
##          'Positive' Class : 1
##
```

#### 4.2.2.4 QDA Results

QDA has a prediction accuracy of 0.8779, with a sensitivity of 0.68262 and a specificity of 0.90071. The area under the ROC curve is 0.912 .Output of the QDA call and the ROC curve are shown in Appendix - LDA. The confusion matrix is shown below for comparison with LDA.

```
## Confusion Matrix and Statistics
##
##          0      1
## 0 46972 2425
## 1 1907 3278
##
##          Accuracy : 0.9206
##                 95% CI : (0.9183, 0.9229)
##    No Information Rate : 0.8955
##   P-Value [Acc > NIR] : < 2.2e-16
##
##          Kappa : 0.5582
##
##  Mcnemar's Test P-Value : 3.998e-15
##
##          Sensitivity : 0.57479
##          Specificity : 0.96099
##    Pos Pred Value : 0.63221
##    Neg Pred Value : 0.95091
##        Prevalence : 0.10448
##        Detection Rate : 0.06006
##  Detection Prevalence : 0.09499
##     Balanced Accuracy : 0.76789
##
##          'Positive' Class : 1
##
```

#### 4.2.2.5 LDA Conclusion

Although the QDA is better at predicting the top 10 placements that were true top 10 than LDA (QDA sensitivity of 0.68 > LDA sensitivity of 0.57), QDA predicts many more incorrect top 10 placements than LDA does (1907 LDA false positives < 4853 QDA false positives). Because LDA has a better overall accuracy (0.9206 for LDA > 0.8779 for QDA), we think the LDA is a stronger model for prediction than QDA.

### 4.2.3 Random Forest

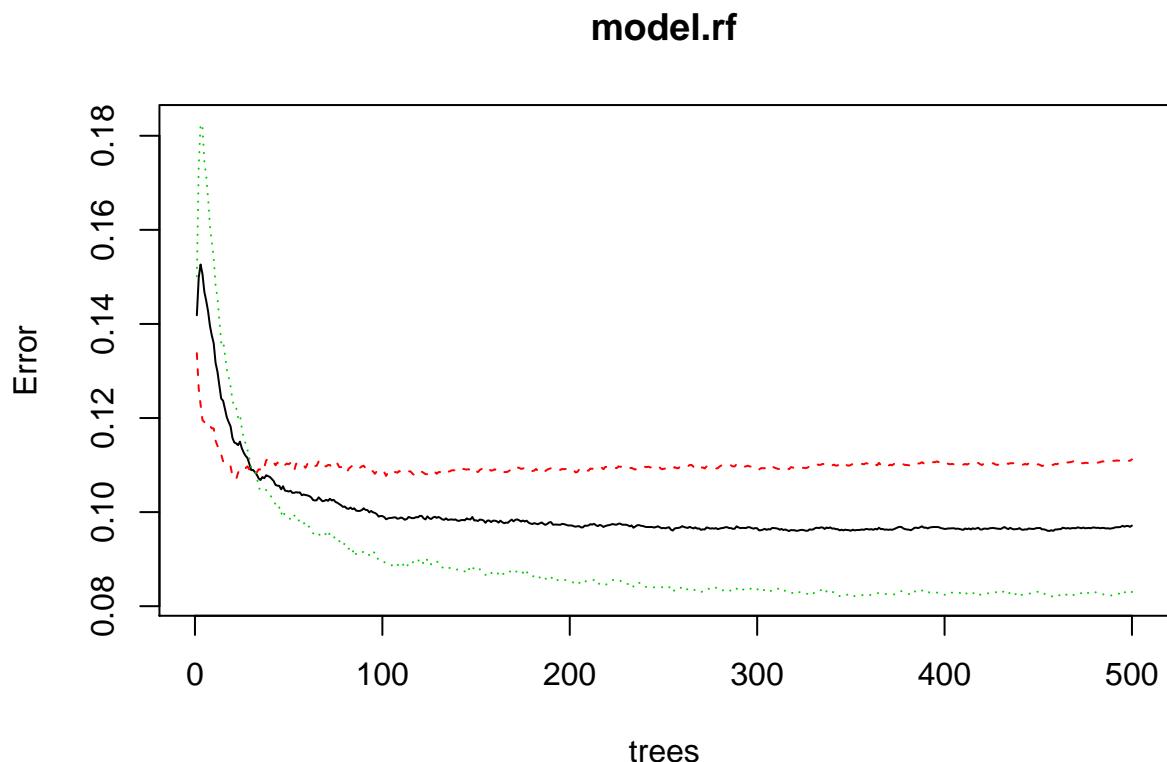
We then tried our first non-parametrical model with Random Forest, which averages out the results of many Decision Trees to provide the lowest error rates across all of the permutations attempted.

#### 4.2.3.1 Assumption Checking

Random Forest methods do not have the same level of assumption restrictions as many of the parametrical models we reviewed. Fortunately, our data set had no null values to speak of which may have posed more of a problem.

#### 4.2.3.2 Error Rates by No. of Trees

The model could likely be tuned further as you can see there's no additional payoff from having the `ntree` value set so high. In fact, it could be harming the performance by setting it to such a high value as you can see the error rates starting to trend upwards at the far-right edge of the graph.



#### 4.2.3.3 Variable Importance

Similar features were confirmed as having the best predictive outcomes from the variable importance output. (i.e... Kill Place, Walking Distance, etc..) edge of the graph.

| Column.Name     | Overall.Score |
|-----------------|---------------|
| assists         | 43.62334      |
| boosts          | 1157.86767    |
| damageDealt     | 587.86497     |
| headshotKills   | 61.26146      |
| heals           | 209.00441     |
| killPlace       | 5172.69584    |
| killPoints      | 155.16722     |
| kills           | 635.76974     |
| killStreaks     | 159.01334     |
| longestKill     | 386.08015     |
| matchDuration   | 105.38803     |
| maxPlace        | 326.37764     |
| numGroups       | 367.23582     |
| rankPoints      | 289.94210     |
| rideDistance    | 270.77805     |
| roadKills       | 13.37229      |
| swimDistance    | 96.74467      |
| teamKills       | 22.79068      |
| vehicleDestroys | 13.58180      |
| walkDistance    | 2802.62385    |
| weaponsAcquired | 261.50104     |
| winPoints       | 155.96105     |

#### 4.2.3.4 Random Forest Results

The out-of-bag estimate of the error rate was 9.71% which was consistent with our finding from classifying the test data not used to train the model with. This should allow us to better gauge what real-world performance would be like with new data to apply the model to.

When running our model against the hold out test data set, we received the following performance metrics.

| Model | AUC   | Sensitivity | Specificity | Error Rate (1 - Accuracy) |
|-------|-------|-------------|-------------|---------------------------|
| RF    | 0.906 | 0.918       | 0.894       | 10.4%                     |

#### 4.2.3.5 Random Forest Conclusion

The Random Forest approach required that the model be tuned with parameters better suited to the provided data set.

The following arguments were important to address bias/ variance issues that were observed when using the default settings. Trial and error techniques along with tuning libraries in `caret` and native to the `randomForest` library allowed us to better optimize the results. We were able to get a roughly 90/90 split between our sensitivity and specificity performance measures. Or rather, when it was in the top 10%, the model was able to accurately predict that it was roughly 90% of the time. The same could be said for the specificity metric.

From the library documentation:

- `ntree` : Number of trees to grow
- `mtry` : Number of variables randomly sampled as candidates at each split.
- `cutoff` : A vector of length equal to number of classes.

### 4.3 Comparing Competing Models

### 4.4 Model Interpretation

### 4.5 Conclusion

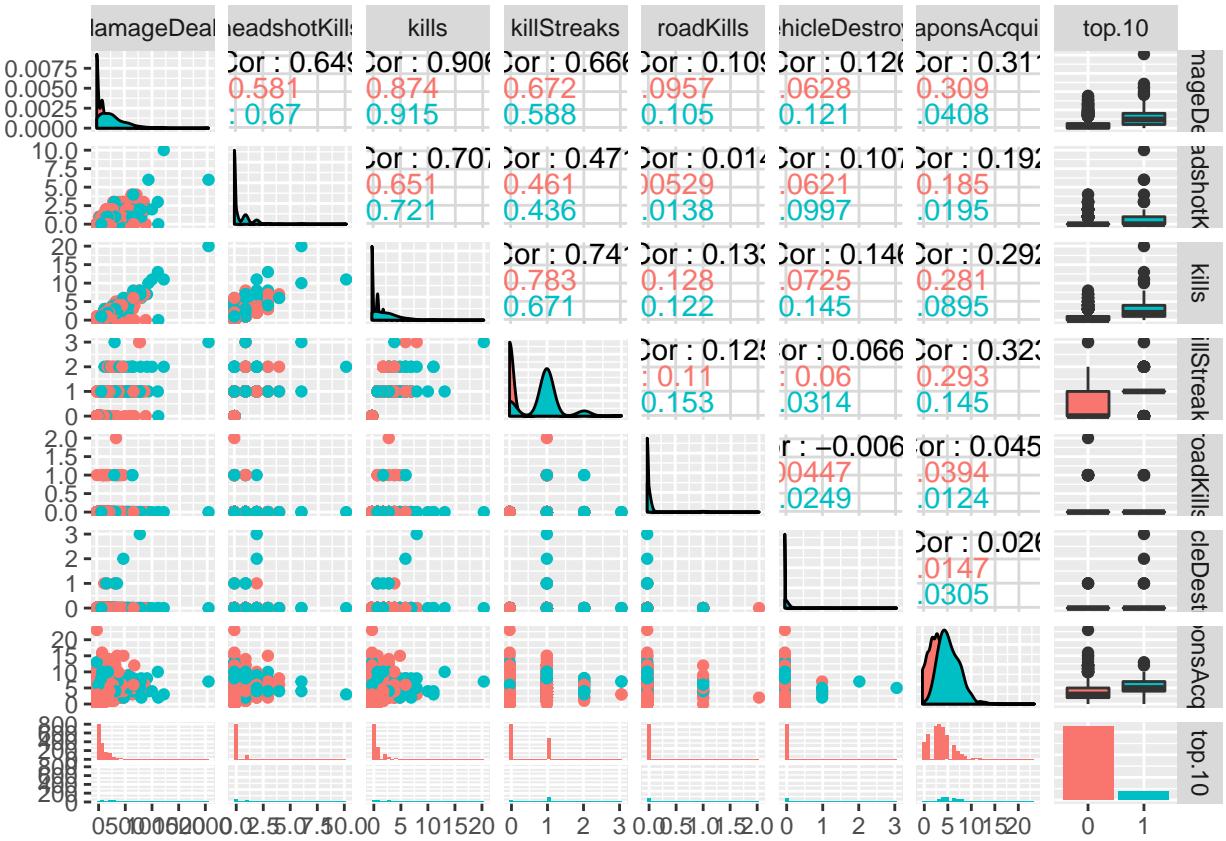


## 5 Appendix

### 5.1 Exploratory Data Analysis

#### 5.1.1 Pairs Plots

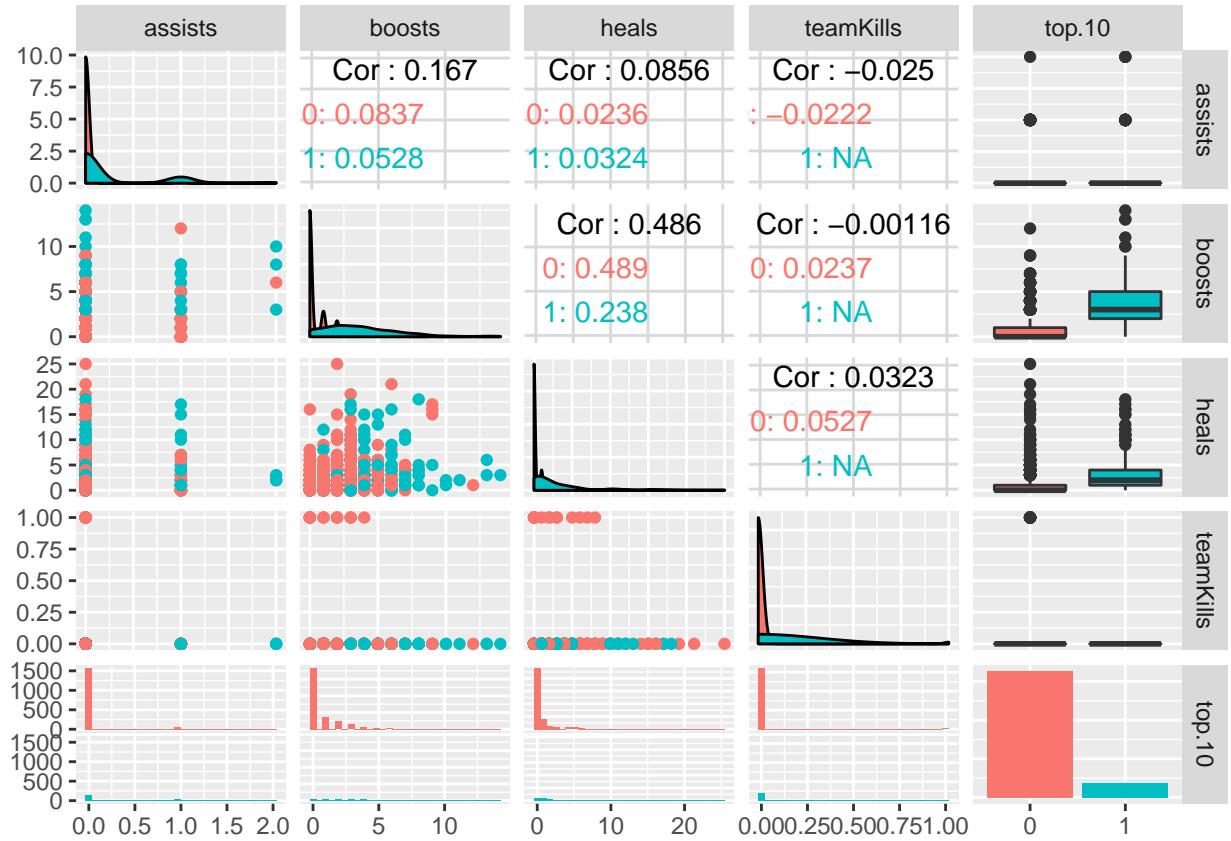




```
## Warning in cor(x, y, method = method, use = use): the standard deviation is
## zero
```

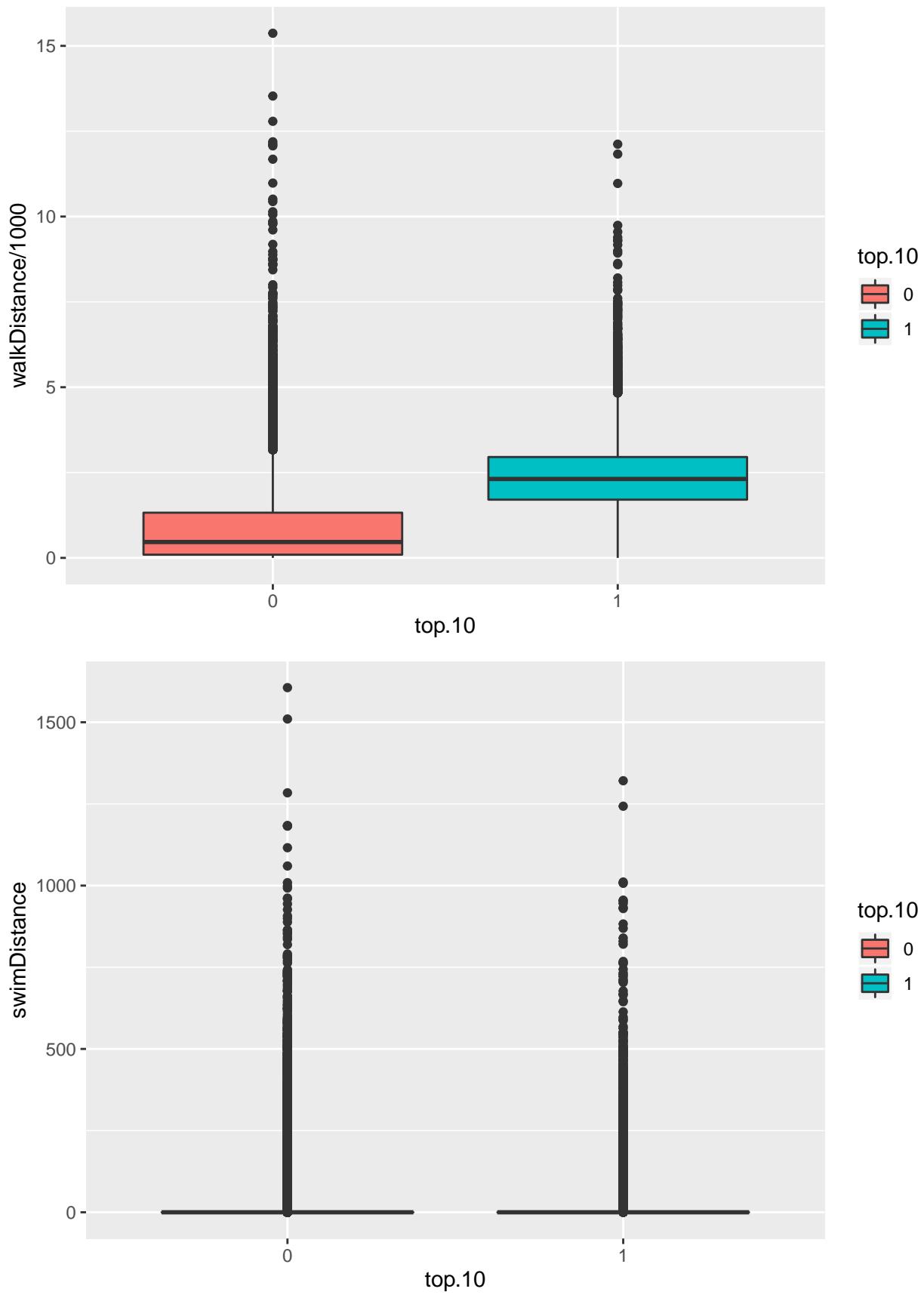
```
## Warning in cor(x, y, method = method, use = use): the standard deviation is
## zero
```

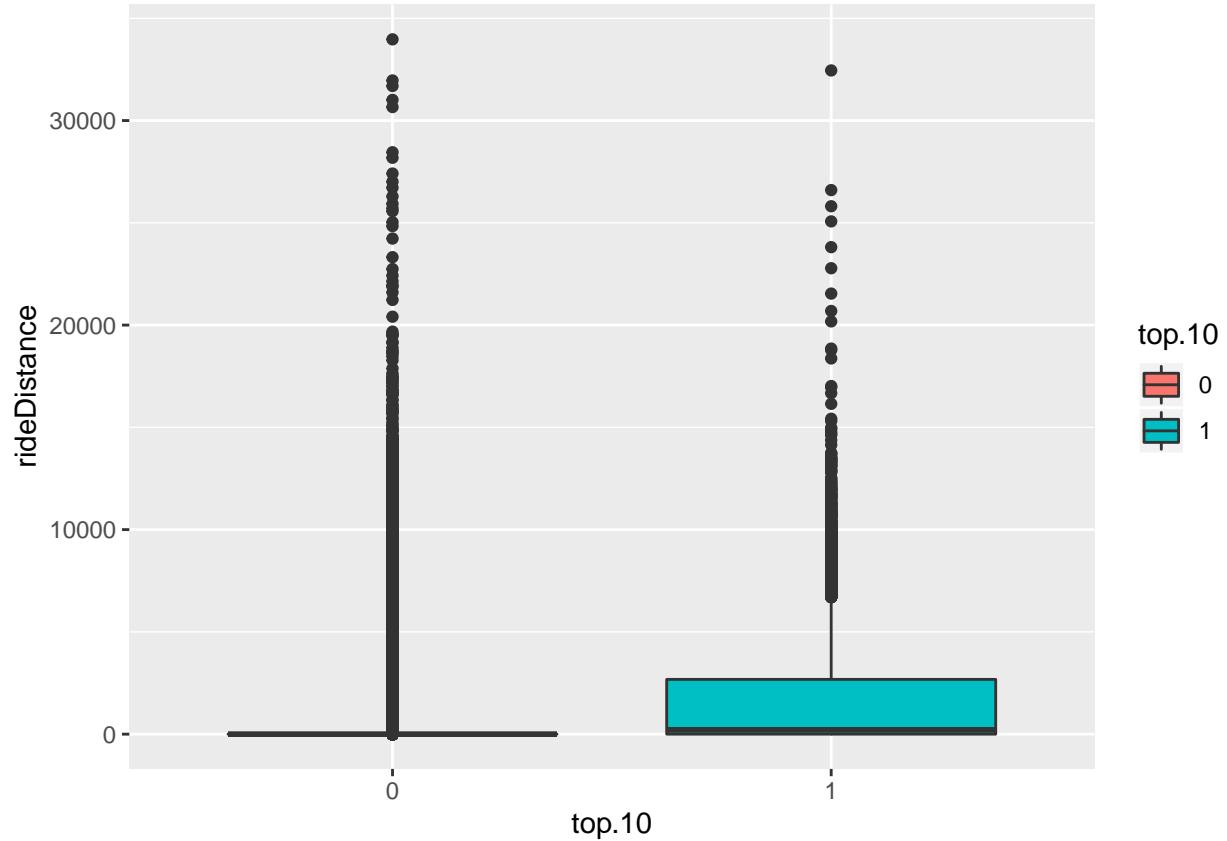
```
## Warning in cor(x, y, method = method, use = use): the standard deviation is
## zero
```

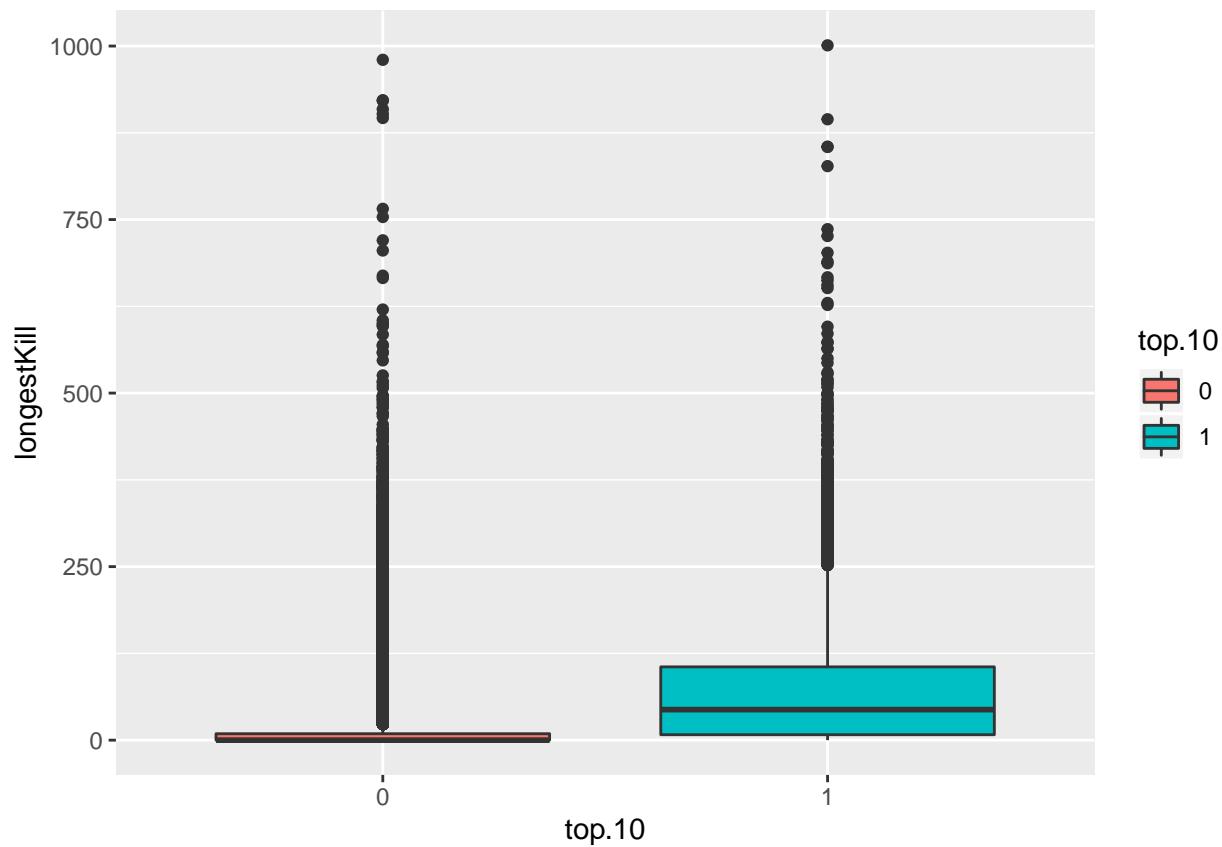


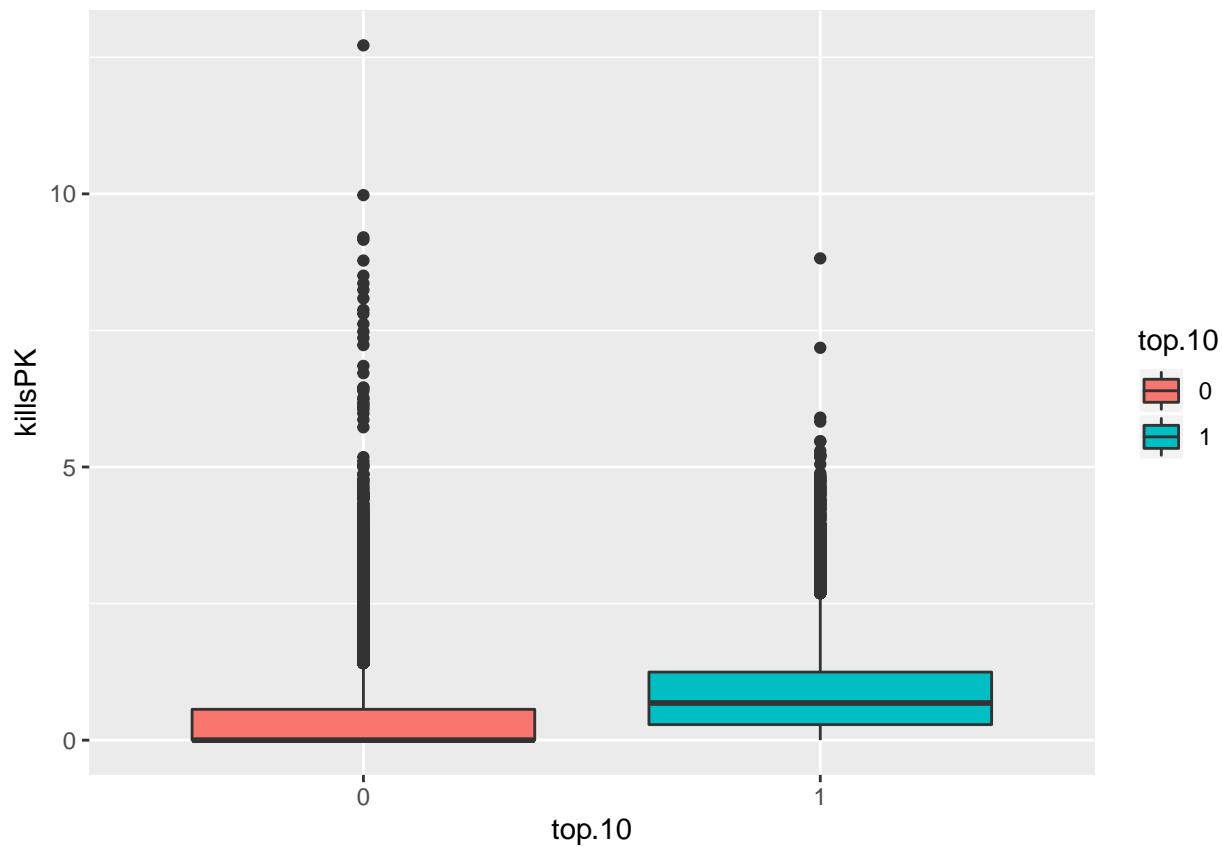


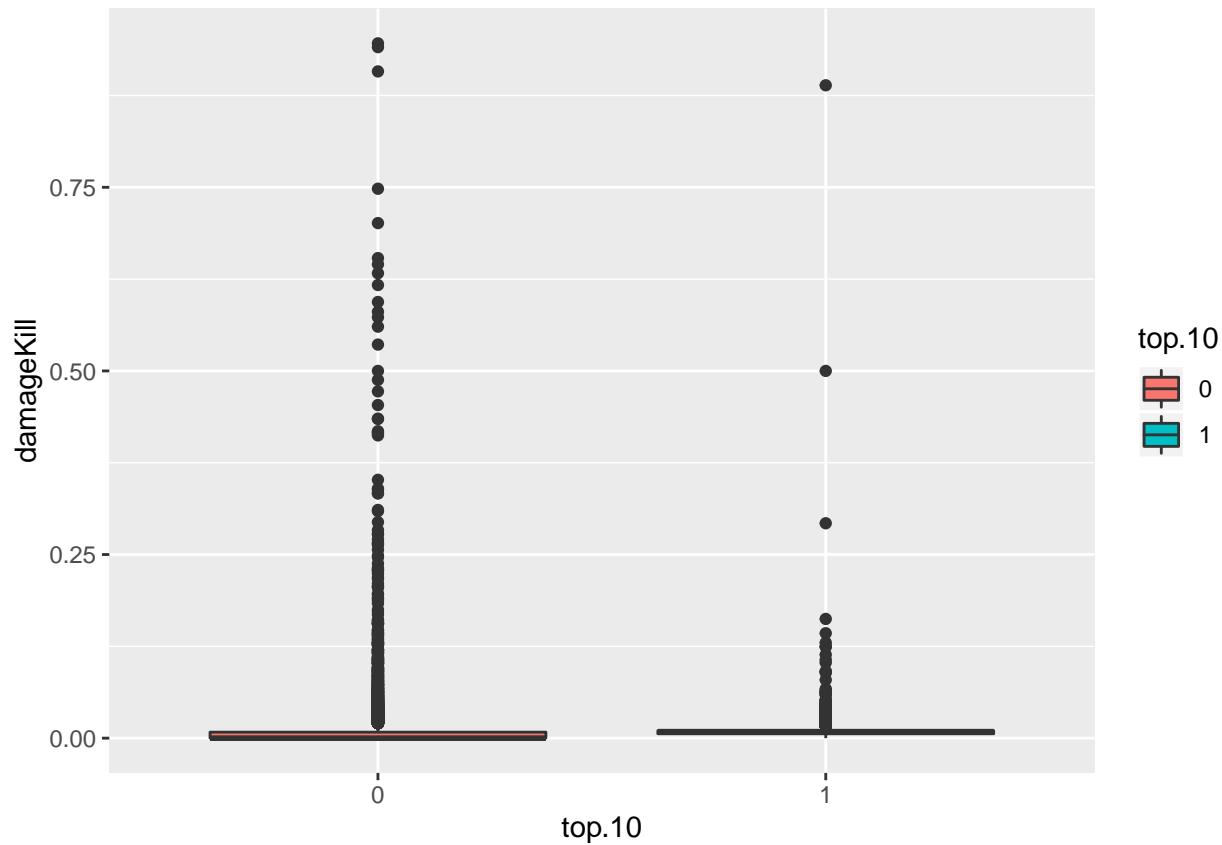
### 5.1.2 Box Plots

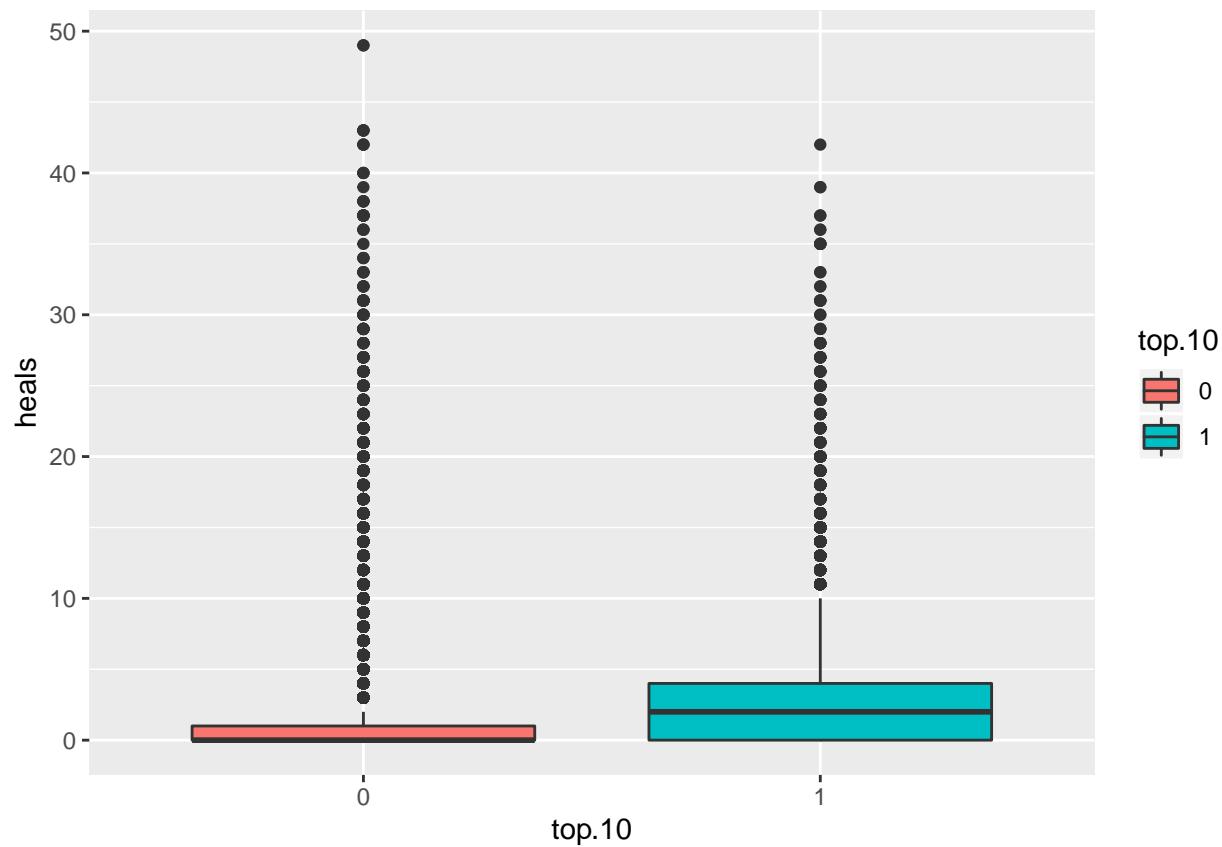


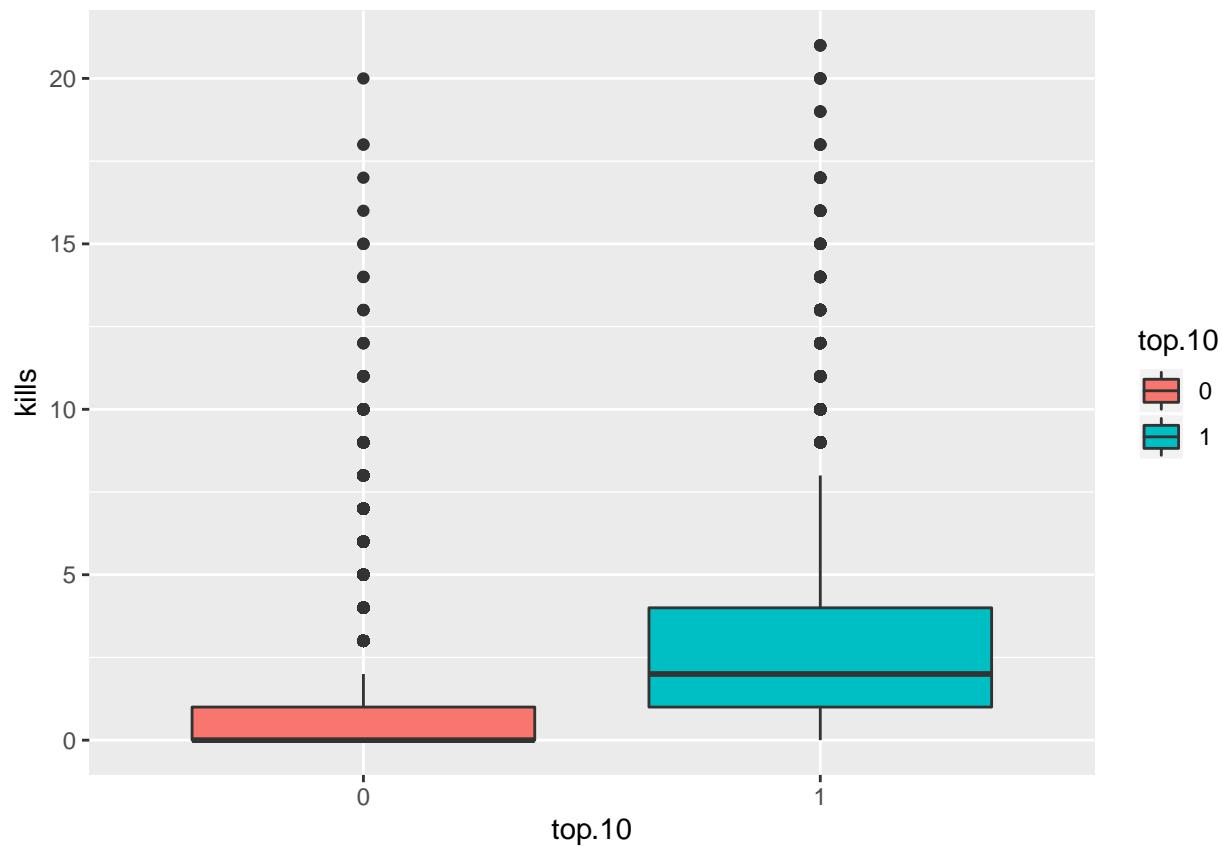


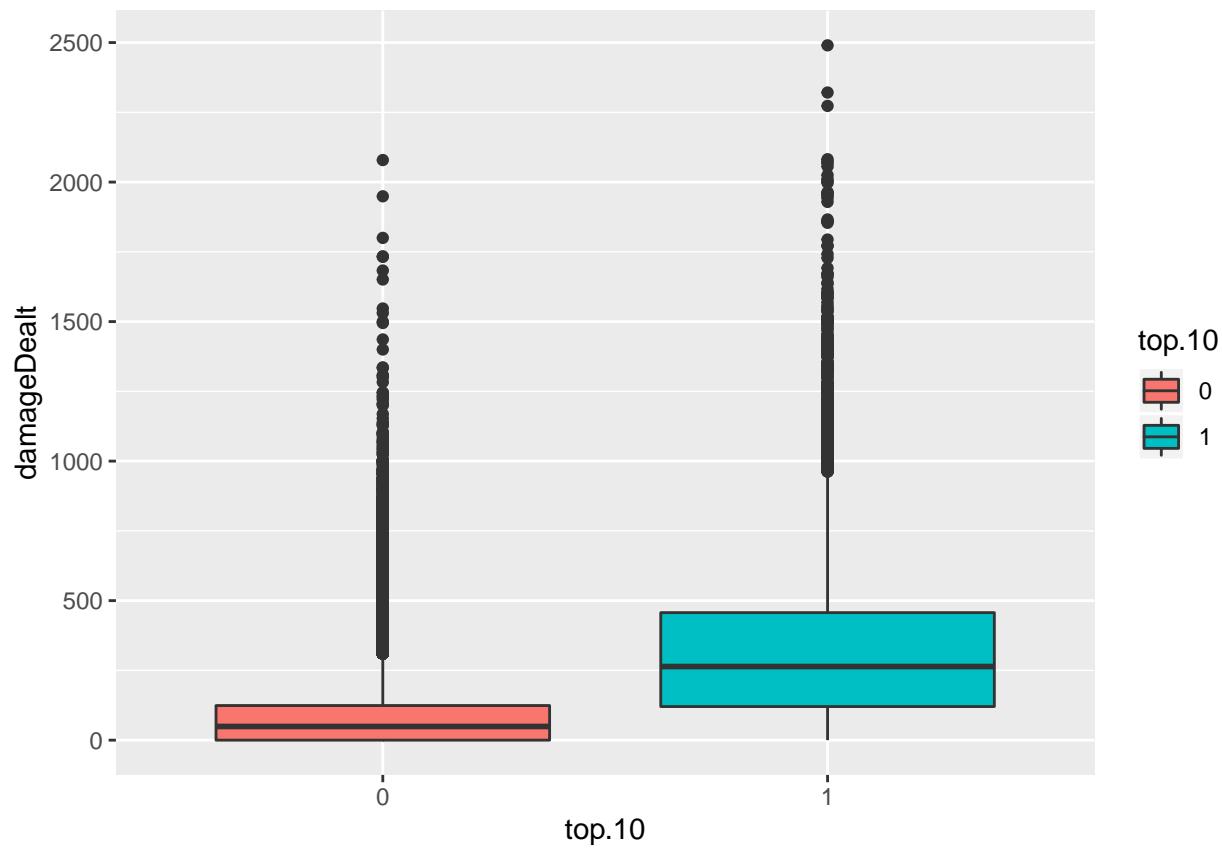


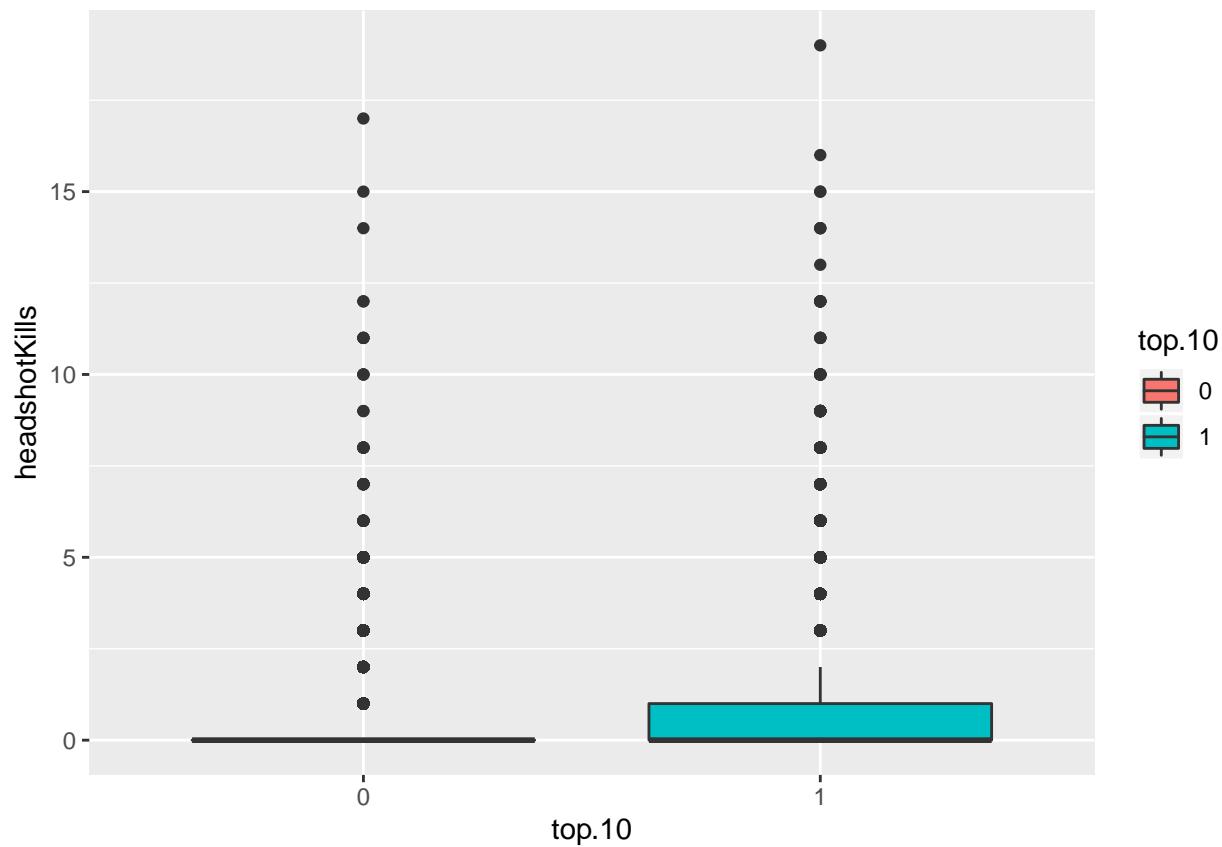


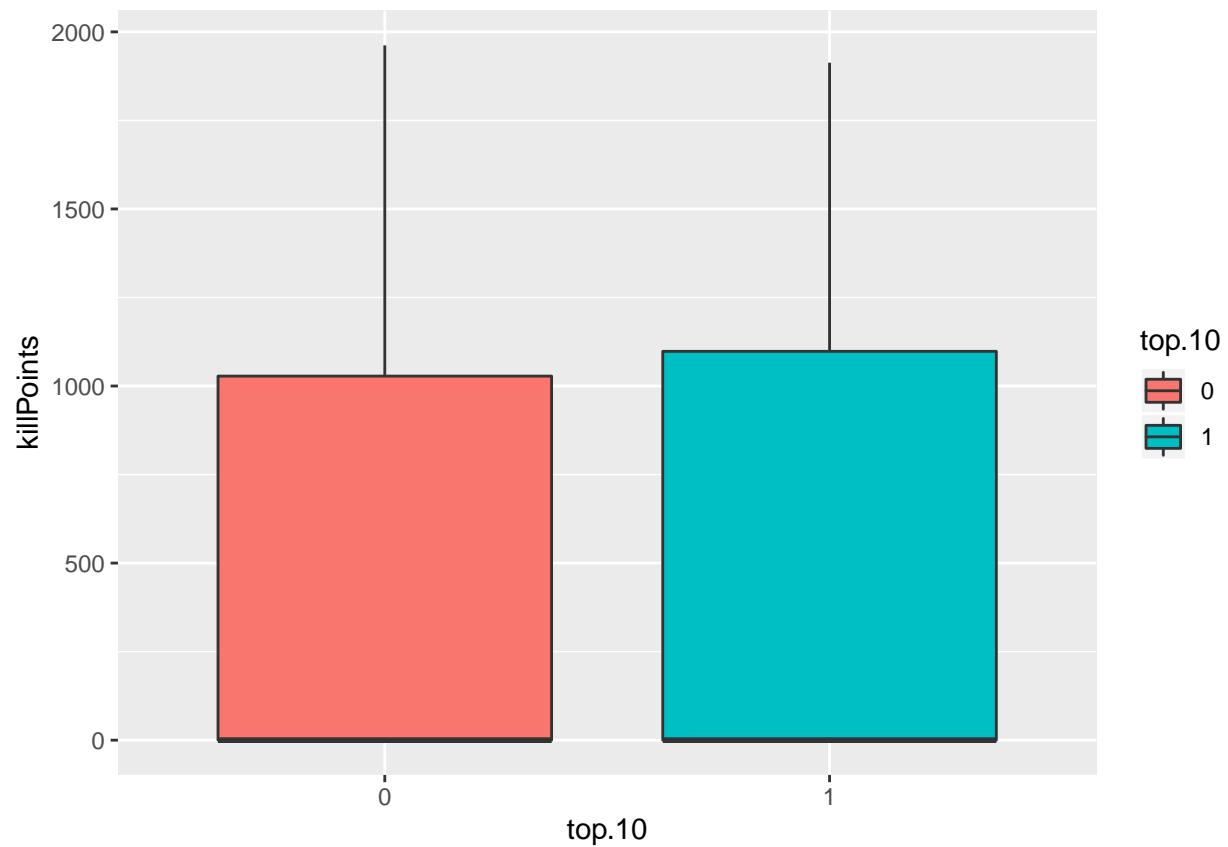


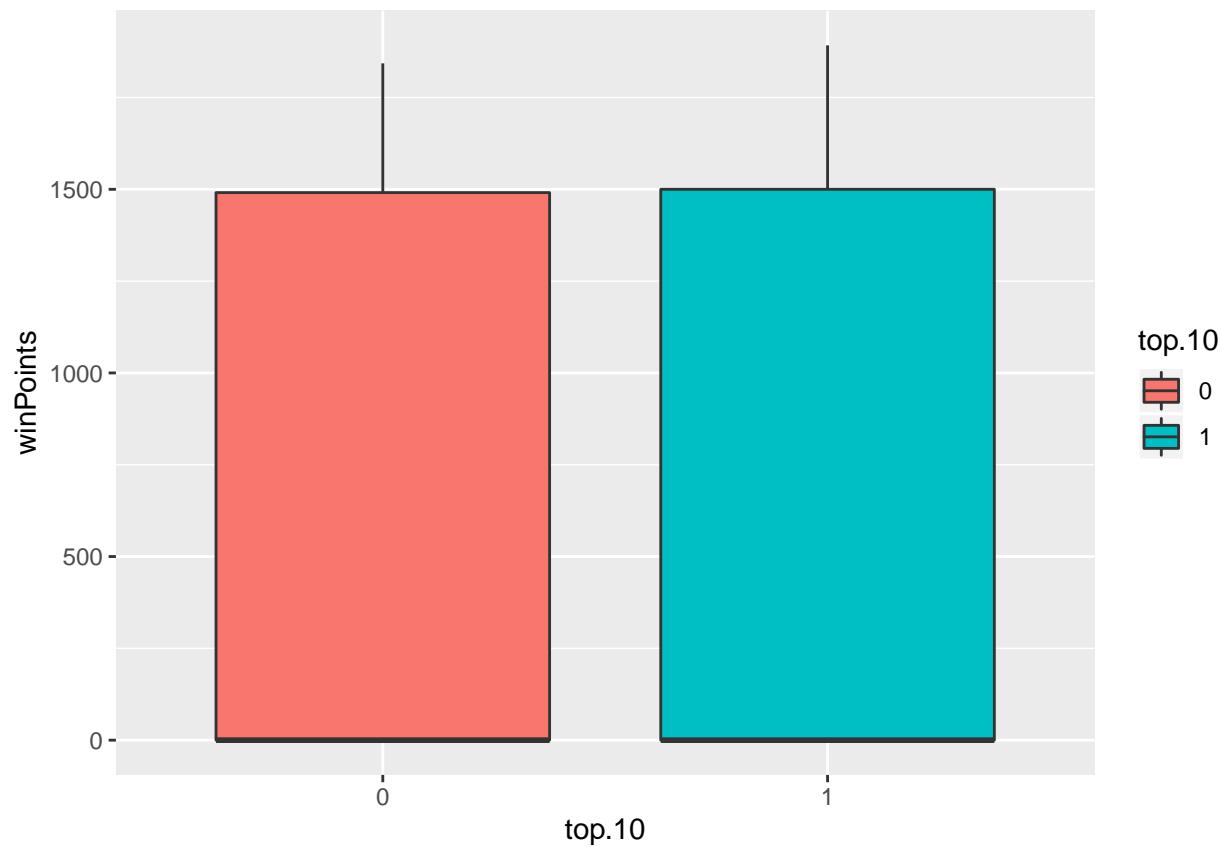


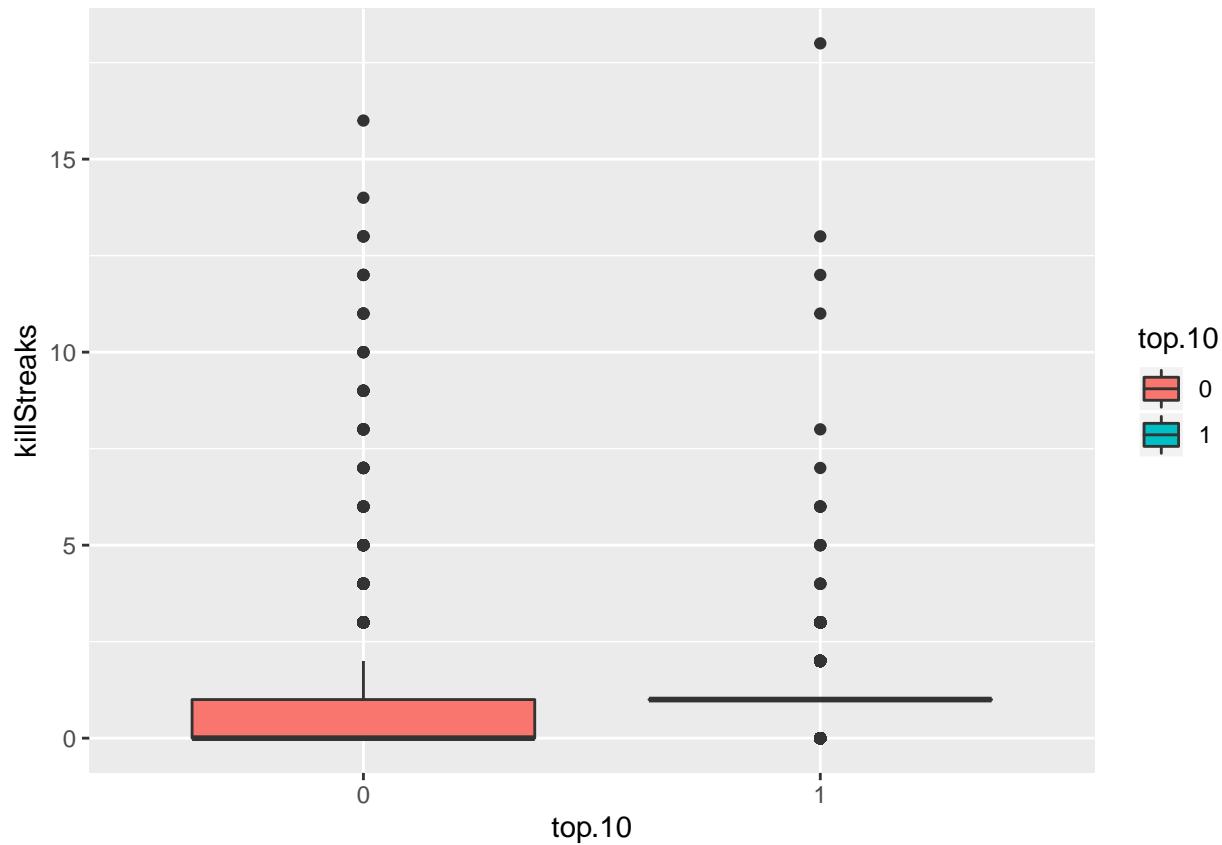


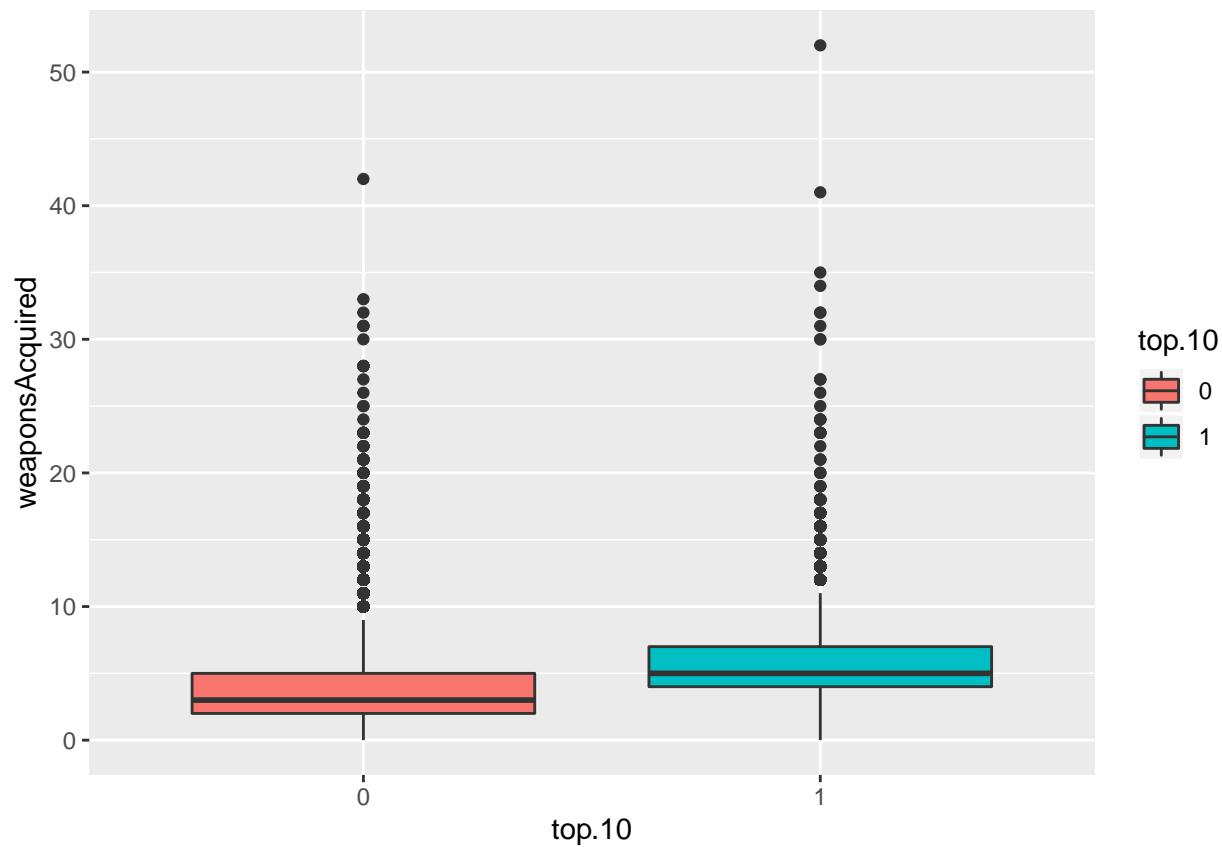


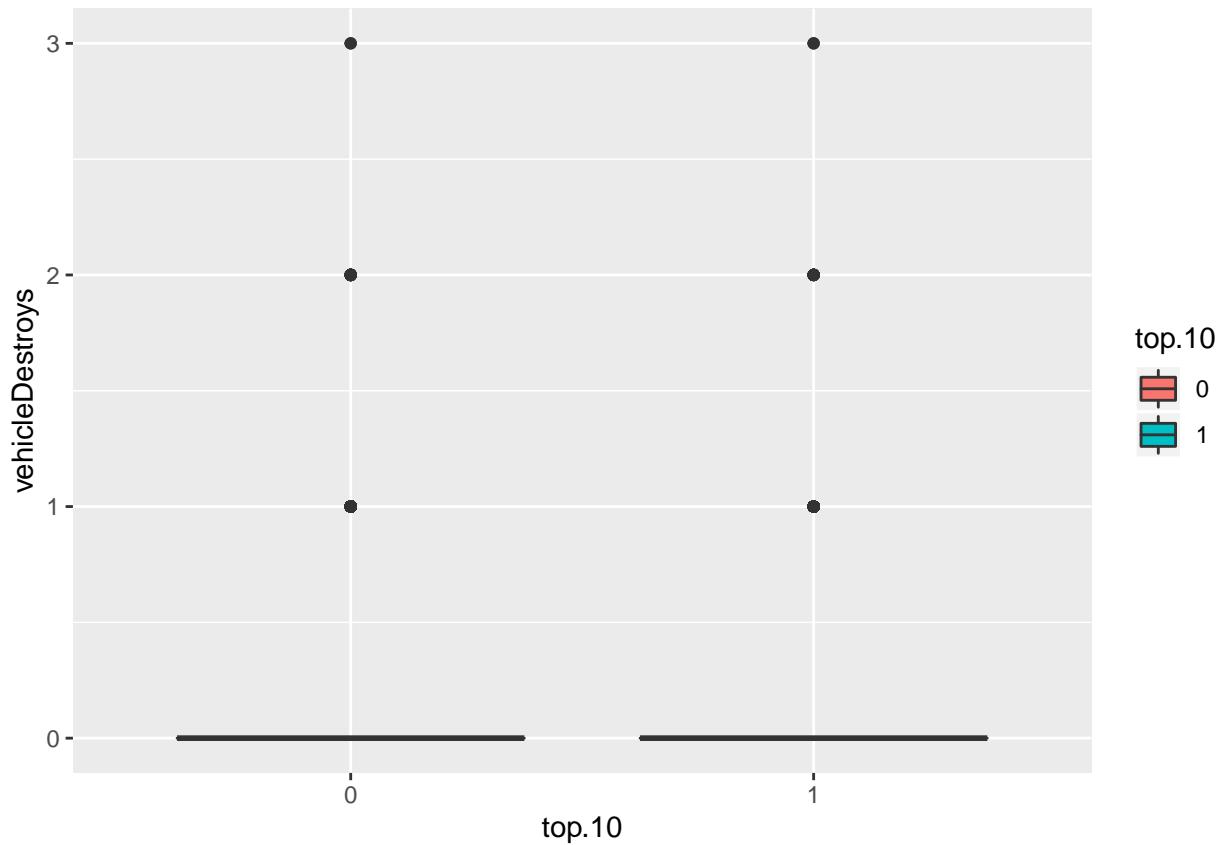












### Logistic Regression

#### 5.1.2.1 Original logistic model with VIF

```
##
## Call:
## glm(formula = top.10 ~ assists + boosts + heals + teamKills +
##      weaponsAcquired + damageDealt + headshotKills + kills + killStreaks +
##      roadKills + vehicleDestroys + killPlace + killPoints + rankPoints +
##      winPoints + longestKill + swimDistance + rideDistance + walkDistance,
##      family = binomial(link = "logit"), data = train_obj1)
##
## Deviance Residuals:
##    Min      1Q   Median      3Q     Max
## -3.5064 -0.2621  0.0017  0.4644  5.1751
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.335e+00  4.679e-01 -11.402 < 2e-16 ***
## assists      6.767e-01  7.031e-02   9.624 < 2e-16 ***
## boosts       3.279e-01  1.265e-02  25.930 < 2e-16 ***
##
```

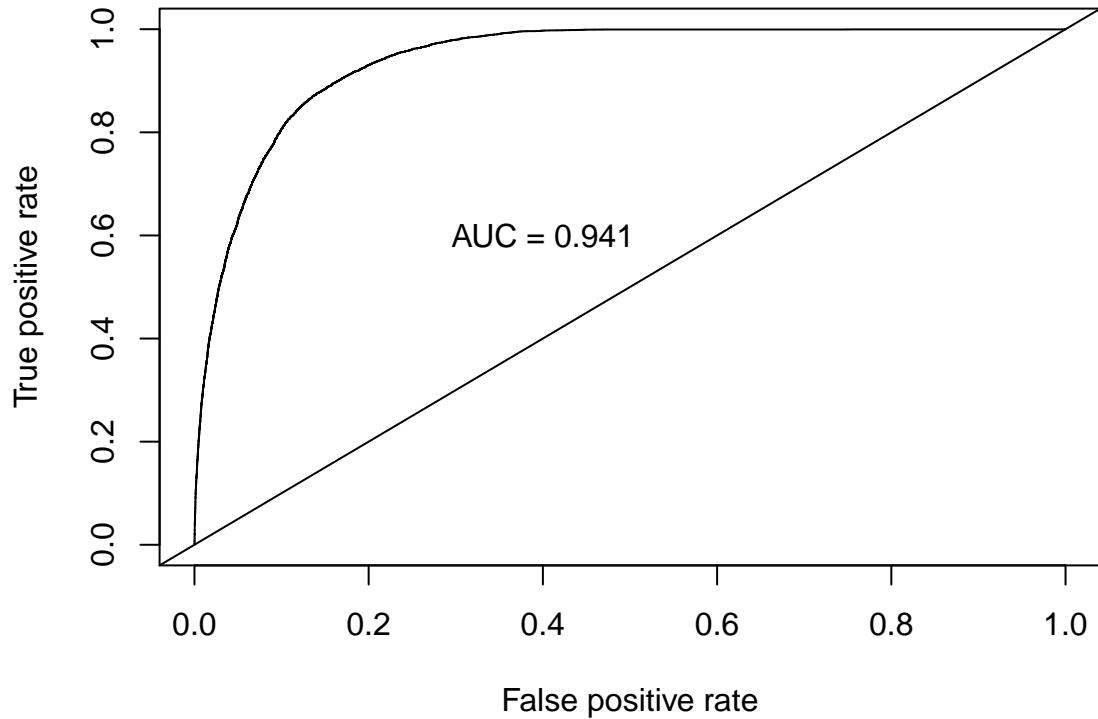
```

## heals      -3.521e-02  6.748e-03 -5.217 1.82e-07 ***
## teamKills -2.027e+00  2.315e-01 -8.757 < 2e-16 ***
## weaponsAcquired 4.750e-02  8.215e-03  5.782 7.36e-09 ***
## damageDealt   6.655e-05  2.758e-04  0.241  0.80935
## headshotKills 4.061e-02  3.300e-02  1.231  0.21841
## kills       -5.179e-02  3.254e-02 -1.591  0.11151
## killStreaks   -1.210e+00  5.375e-02 -22.509 < 2e-16 ***
## roadKills    -2.286e-01  1.223e-01 -1.870  0.06148 .
## vehicleDestroys -4.657e-01  1.674e-01 -2.783  0.00539 **
## killPlace     -9.337e-02  1.817e-03 -51.380 < 2e-16 ***
## killPoints    -1.578e-03  2.606e-04 -6.055 1.41e-09 ***
## rankPoints    3.995e-03  3.056e-04 13.074 < 2e-16 ***
## winPoints     5.282e-03  3.949e-04 13.376 < 2e-16 ***
## longestKill   -1.896e-03  4.007e-04 -4.732 2.22e-06 ***
## swimDistance   1.503e-03  3.705e-04  4.056 5.00e-05 ***
## rideDistance   1.316e-04  1.011e-05 13.026 < 2e-16 ***
## walkDistance   8.866e-04  2.384e-05 37.197 < 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: 36895  on 26613  degrees of freedom
## Residual deviance: 15916  on 26594  degrees of freedom
## AIC: 15956
##
## Number of Fisher Scoring iterations: 6

##      assists      boosts      heals      teamKills
##      1.056570    1.457881   1.190480   1.012404
## weaponsAcquired  damageDealt  headshotKills      kills
##      1.036598    7.068001   1.996675   9.407337
## killStreaks      roadKills  vehicleDestroys  killPlace
##      2.283736    1.056648   1.030448   2.142370
## killPoints       rankPoints  winPoints      longestKill
##      54.408050   124.018858  204.493531   1.703710
## swimDistance     rideDistance  walkDistance
##      1.016534    1.137564   1.387638

```

## Random Forest



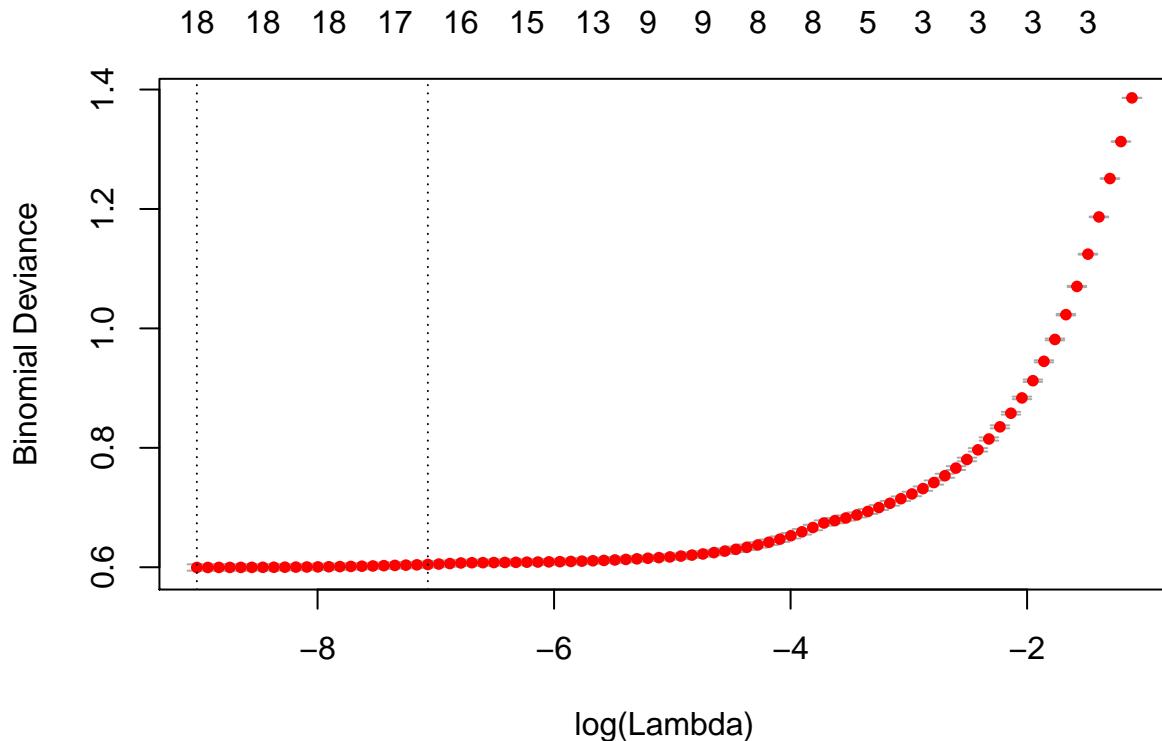
```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction      0      1
##           0 41211   619
##           1 7668   5084
##
##          Accuracy : 0.8482
## 95% CI : (0.8451, 0.8512)
## No Information Rate : 0.8955
## P-Value [Acc > NIR] : 1
##
##          Kappa : 0.4752
##
## McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.89146
##          Specificity : 0.84312
## Pos Pred Value : 0.39868
## Neg Pred Value : 0.98520
## Prevalence : 0.10448
```

```

##          Detection Rate : 0.09314
##    Detection Prevalence : 0.23363
##    Balanced Accuracy : 0.86729
##
##    'Positive' Class : 1
##

```

### 5.1.2.2 Full Lasso Variable Reduction



```

## [1] 0.0001208403

## 21 x 1 sparse Matrix of class "dgCMatrix"
##                               1
## (Intercept) -4.3850198039
## (Intercept) .
## assists      0.6746223715
## boosts       0.3272759414
## heals        -0.0342697309
## teamKills    -2.0062930907
## weaponsAcquired 0.0470239834
## damageDealt   .

```

```

## headshotKills      0.0333622719
## kills              -0.0440824065
## killStreaks        -1.1987929481
## roadKills          -0.2242257360
## vehicleDestroys   -0.4592145376
## killPlace           -0.0923410748
## killPoints          -0.0011652837
## rankPoints          0.0033626431
## winPoints           0.0043254560
## longestKill         -0.0018193615
## swimDistance        0.0014831701
## rideDistance        0.0001321729
## walkDistance        0.0008825722

## [1] 0.0008525043

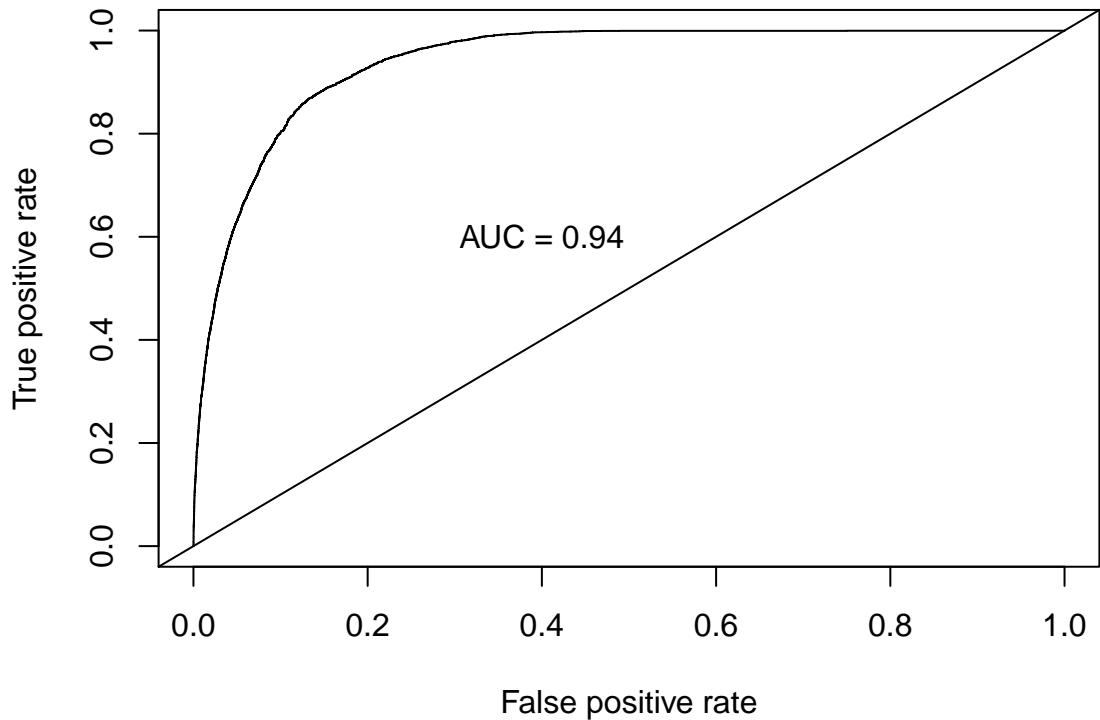
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                                         1
## (Intercept) -0.6697250551
## (Intercept) .
## assists      0.6465141635
## boosts       0.3247444750
## heals        -0.0291869321
## teamKills    -1.9071380400
## weaponsAcquired 0.0445041552
## damageDealt  .
## headshotKills .
## kills         -0.0263675957
## killStreaks   -1.1451581898
## roadKills     -0.1971800303
## vehicleDestroys -0.4170606837
## killPlace     -0.0877153549
## killPoints    .
## rankPoints    0.0008643892
## winPoints     0.0008978790
## longestKill   -0.0014961791
## swimDistance  0.0013709914
## rideDistance  0.0001344471
## walkDistance  0.0008664751

## 21 x 1 sparse Matrix of class "dgCMatrix"
##                                         s0
## (Intercept) -0.6733424558
## (Intercept) .

```

```
## assists          0.6465602204
## boosts          0.3247910863
## heals           -0.0291791887
## teamKills       -1.9070440684
## weaponsAcquired 0.0445136944
## damageDealt      .
## headshotKills    .
## kills            -0.0262597549
## killStreaks      -1.1450786717
## roadKills        -0.1971398791
## vehicleDestroys -0.4170561352
## killPlace        -0.0876942534
## killPoints        .
## rankPoints       0.0008663913
## winPoints        0.0008998702
## longestKill      -0.0014955118
## swimDistance     0.0013710884
## rideDistance     0.0001344144
## walkDistance     0.0008663011
```

## Random Forest



##

```

##      0      1
## 48879  5703

## Confusion Matrix and Statistics
##
##          Reference
## Prediction      0      1
##           0 43486   972
##           1  5393  4731
##
##          Accuracy : 0.8834
##                 95% CI : (0.8807, 0.8861)
##    No Information Rate : 0.8955
##    P-Value [Acc > NIR] : 1
##
##          Kappa : 0.5358
##
## McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.82956
##          Specificity : 0.88967
##    Pos Pred Value : 0.46731
##    Neg Pred Value : 0.97814
##          Prevalence : 0.10448
##          Detection Rate : 0.08668
##    Detection Prevalence : 0.18548
##    Balanced Accuracy : 0.85961
##
##          'Positive' Class : 1
##

##
## Call:
## glm(formula = top.10 ~ assists + boosts + heals + teamKills +
##       weaponsAcquired + kills + killStreaks + roadKills + vehicleDestroys +
##       killPlace + rankPoints + winPoints + longestKill + swimDistance +
##       rideDistance + walkDistance, family = binomial(link = "logit"),
##       data = train_obj1)
##
## Deviance Residuals:
##      Min      1Q      Median      3Q      Max
## -3.5204 -0.2625   0.0021   0.4641   5.1861
##
## Coefficients:
```

```

##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.884e+00 4.571e-01 -10.683 < 2e-16 ***
## assists      6.792e-01 6.870e-02   9.886 < 2e-16 ***
## boosts       3.257e-01 1.260e-02  25.842 < 2e-16 ***
## heals        -3.475e-02 6.734e-03 -5.160 2.47e-07 ***
## teamKills    -2.034e+00 2.308e-01 -8.811 < 2e-16 ***
## weaponsAcquired 4.852e-02 8.216e-03  5.905 3.52e-09 ***
## kills         -4.373e-02 1.800e-02 -2.430 0.01510 *
## killStreaks   -1.212e+00 5.305e-02 -22.839 < 2e-16 ***
## roadKills     -2.466e-01 1.211e-01 -2.036 0.04177 *
## vehicleDestroys -4.814e-01 1.672e-01 -2.880 0.00398 **
## killPlace      -9.255e-02 1.765e-03 -52.432 < 2e-16 ***
## rankPoints     3.695e-03 2.989e-04 12.360 < 2e-16 ***
## winPoints      3.764e-03 3.018e-04 12.472 < 2e-16 ***
## longestKill    -1.755e-03 3.962e-04 -4.429 9.47e-06 ***
## swimDistance   1.534e-03 3.714e-04  4.130 3.63e-05 ***
## rideDistance   1.304e-04 1.009e-05 12.934 < 2e-16 ***
## walkDistance   8.889e-04 2.378e-05 37.382 < 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: 36895 on 26613 degrees of freedom
## Residual deviance: 15953 on 26597 degrees of freedom
## AIC: 15987
##
## Number of Fisher Scoring iterations: 6

##          assists        boosts        heals      teamKills
## 1.011919      1.453020      1.189852      1.012041
## weaponsAcquired      kills      killStreaks      roadKills
## 1.036222      2.912014      2.236310      1.053813
## vehicleDestroys      killPlace      rankPoints      winPoints
## 1.028168      2.026555     118.368517     118.867443
## longestKill      swimDistance      rideDistance      walkDistance
## 1.677517      1.015841      1.136265      1.381498

```

### 5.1.2.3 Final Model for Objective 1

```

##
## Call:
## glm(formula = top.10 ~ assists + boosts + heals + weaponsAcquired +

```

```

##      killPlace + walkDistance, family = binomial(link = "logit"),
##      data = train_obj1)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -3.3664 -0.3759 -0.0174  0.4865  2.6092
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -5.934e-01  5.982e-02 -9.919 < 2e-16 ***
## assists                6.225e-01  6.666e-02  9.339 < 2e-16 ***
## boosts                 3.067e-01  1.158e-02 26.493 < 2e-16 ***
## heals                  -2.276e-02  6.449e-03 -3.529 0.000418 ***
## weaponsAcquired     7.900e-02  7.742e-03 10.205 < 2e-16 ***
## killPlace              -6.216e-02  1.192e-03 -52.172 < 2e-16 ***
## walkDistance            7.415e-04  2.073e-05 35.775 < 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: 36895  on 26613  degrees of freedom
## Residual deviance: 17365  on 26607  degrees of freedom
## AIC: 17379
##
## Number of Fisher Scoring iterations: 6

```

### 5.1.3 Complex Logistic Regression

#### 5.1.3.1 Variable Definitions

```

#train_obj1 <- train %>% mutate(
#  DurationCut = as.factor(ifelse(matchDuration > 1600, 1, 0)),
#  kpCut = as.factor(ifelse(killPoints > 0, 1, 0)),
#  wpCut = as.factor(ifelse(winPoints > 0, 1, 0)),
#  rpCut = as.factor(ifelse(rankPoints > -1, 1, 0)),
#  boostCut = as.factor(ifelse(boosts >= 2, 1, 0)),
#  healCut = as.factor(ifelse(heals >= 2, 1, 0)),
#  walkCut = as.factor(ifelse(walkDistance >= 1250, 1, 0)),
#  weaponCut = as.factor(ifelse(weaponsAcquired >= 3, 1, 0)),
#  killCut = as.factor(ifelse(kills >= 2, 1, 0)),
#  assistCut = as.factor(ifelse(assists >= 2, 1, 0)),
#  damageCut = as.factor(ifelse(damageDealt >= 250, 1, 0)),

```

```

# streakCut = as.factor(ifelse(killStreaks >= 1, 1, 0)),
# killsPK = kills/(walkDistance/1000+1),
# damageKill = kills/(damageDealt+1)
#)

```

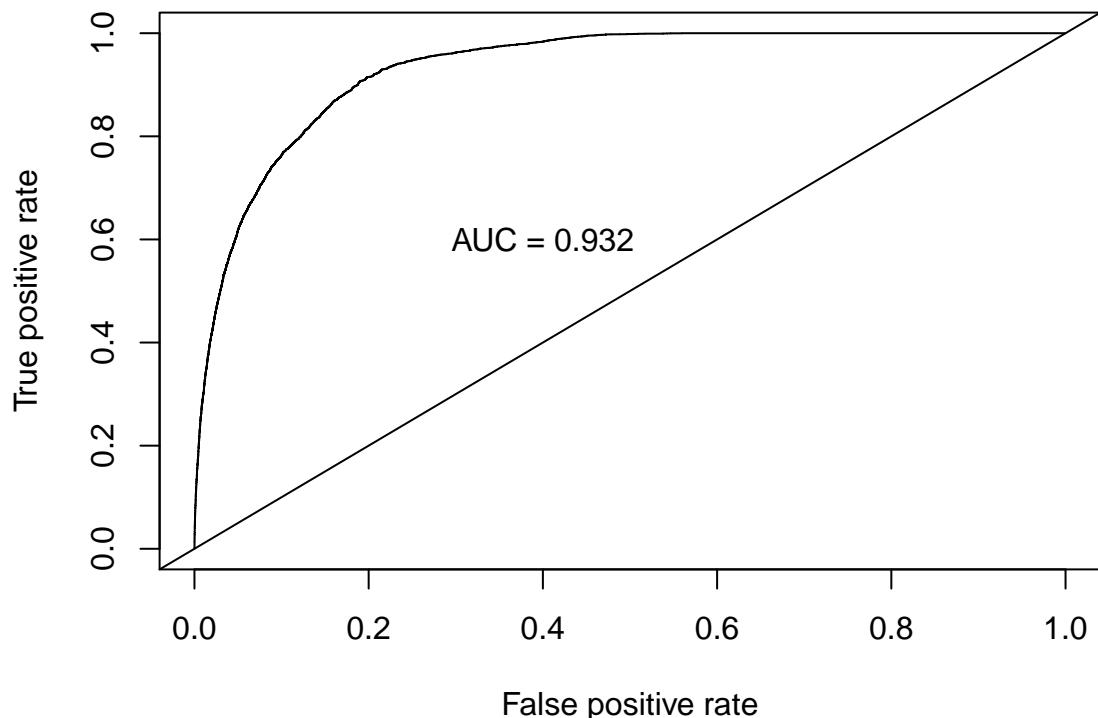
### 5.1.3.2 Full Model List

```

##
## Call:
## glm(formula = top.10 ~ boosts + assists + weaponsAcquired + killPlace +
##      walkDistance + boostCut:killPlace, family = binomial(link = "logit"),
##      data = train_obj1)
##
## Deviance Residuals:
##       Min      1Q   Median      3Q     Max
## -3.2453 -0.3495 -0.0074  0.4957  2.6710
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -4.788e-01  6.177e-02 -7.751 9.10e-15 ***
## boosts                 2.398e-01  1.263e-02 18.993 < 2e-16 ***
## assists                6.233e-01  6.676e-02  9.336 < 2e-16 ***
## weaponsAcquired        7.619e-02  7.777e-03  9.796 < 2e-16 ***
## killPlace               -6.813e-02  1.476e-03 -46.155 < 2e-16 ***
## walkDistance            7.288e-04  2.068e-05 35.241 < 2e-16 ***
## killPlace:boostCut1    1.303e-02  1.703e-03   7.652 1.98e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 36895  on 26613  degrees of freedom
## Residual deviance: 17319  on 26607  degrees of freedom
## AIC: 17333
##
## Number of Fisher Scoring iterations: 6

```

## Random Forest



```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction      0      1
##           0 40845   727
##           1  8034  4976
##
##          Accuracy : 0.8395
##          95% CI : (0.8364, 0.8426)
##  No Information Rate : 0.8955
##  P-Value [Acc > NIR] : 1
##
##          Kappa : 0.4522
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.87252
##          Specificity  : 0.83563
##  Pos Pred Value : 0.38248
##  Neg Pred Value : 0.98251
##          Prevalence  : 0.10448
```

```

##          Detection Rate : 0.09117
##      Detection Prevalence : 0.23836
##      Balanced Accuracy : 0.85408
##
##      'Positive' Class : 1
##

##  

## Call:  

## glm(formula = top.10 ~ boosts + assists + weaponsAcquired + killPlace +  

##      walkDistance + kpCut:killPlace, family = binomial(link = "logit"),  

##      data = train_obj1)  

##  

## Deviance Residuals:  

##      Min        1Q    Median        3Q       Max  

## -3.3510  -0.3752  -0.0173   0.4860   2.6179  

##  

## Coefficients:  

##              Estimate Std. Error z value Pr(>|z|)  

## (Intercept) -5.951e-01  5.983e-02 -9.946 <2e-16 ***  

## boosts      2.943e-01  1.096e-02  26.854 <2e-16 ***  

## assists     6.196e-01  6.666e-02   9.294 <2e-16 ***  

## weaponsAcquired 7.865e-02  7.739e-03  10.164 <2e-16 ***  

## killPlace    -6.259e-02  1.298e-03 -48.225 <2e-16 ***  

## walkDistance 7.334e-04  2.057e-05  35.646 <2e-16 ***  

## killPlace:kpCut1 1.079e-03  1.400e-03   0.771   0.441  

## ---  

## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  

##  

## (Dispersion parameter for binomial family taken to be 1)  

##  

## Null deviance: 36895  on 26613  degrees of freedom  

## Residual deviance: 17377  on 26607  degrees of freedom  

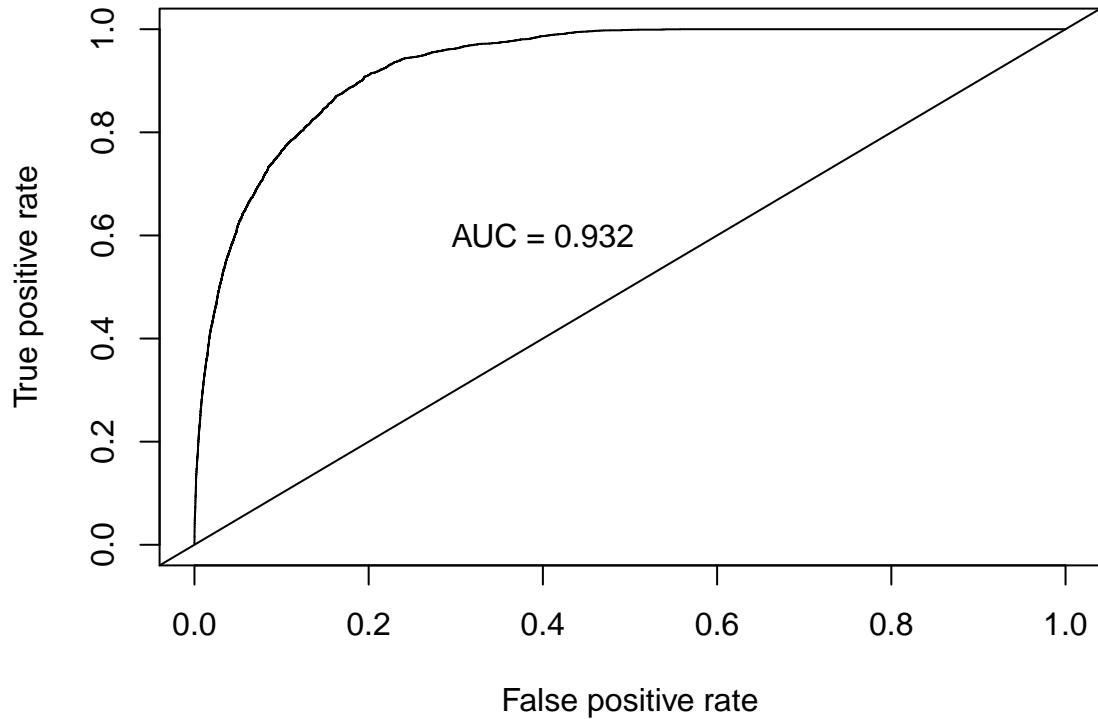
## AIC: 17391  

##  

## Number of Fisher Scoring iterations: 6

```

## Random Forest



```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction      0      1
##           0 41103   785
##           1  7776  4918
##
##          Accuracy : 0.8432
##                 95% CI : (0.8401, 0.8462)
##    No Information Rate : 0.8955
##    P-Value [Acc > NIR] : 1
##
##          Kappa : 0.4562
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.8624
##          Specificity  : 0.8409
##    Pos Pred Value : 0.3874
##    Neg Pred Value : 0.9813
##          Prevalence : 0.1045
```

```

##          Detection Rate : 0.0901
##      Detection Prevalence : 0.2326
##      Balanced Accuracy : 0.8516
##
##      'Positive' Class : 1
##

##  

## Call:  

## glm(formula = top.10 ~ boosts + assists + weaponsAcquired + killPlace +  

##      walkDistance + walkDistance * killPlace, family = binomial(link = "logit"),  

##      data = train_obj1)  

##  

## Deviance Residuals:  

##      Min        1Q    Median        3Q        Max  

## -3.9357  -0.3234  -0.0021   0.5071   2.7891  

##  

## Coefficients:  

##                               Estimate Std. Error z value Pr(>|z|)  

## (Intercept)           -1.125e-01 7.102e-02 -1.584   0.113  

## boosts                3.150e-01 1.108e-02 28.434 <2e-16 ***  

## assists               6.477e-01 6.671e-02  9.709 <2e-16 ***  

## weaponsAcquired       6.775e-02 7.839e-03  8.643 <2e-16 ***  

## killPlace              -8.444e-02 2.314e-03 -36.484 <2e-16 ***  

## walkDistance           4.135e-04 3.191e-05 12.957 <2e-16 ***  

## killPlace:walkDistance 1.432e-05 1.183e-06 12.110 <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: 36895  on 26613  degrees of freedom  

## Residual deviance: 17219  on 26607  degrees of freedom  

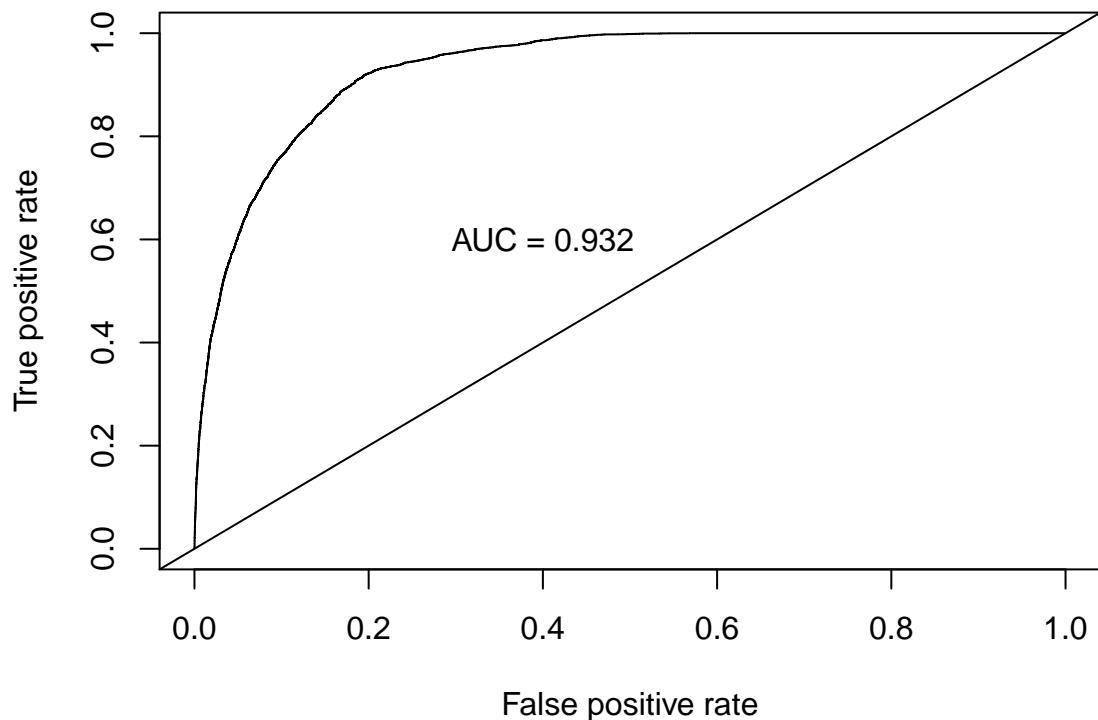
## AIC: 17233  

##  

## Number of Fisher Scoring iterations: 7

```

## Random Forest



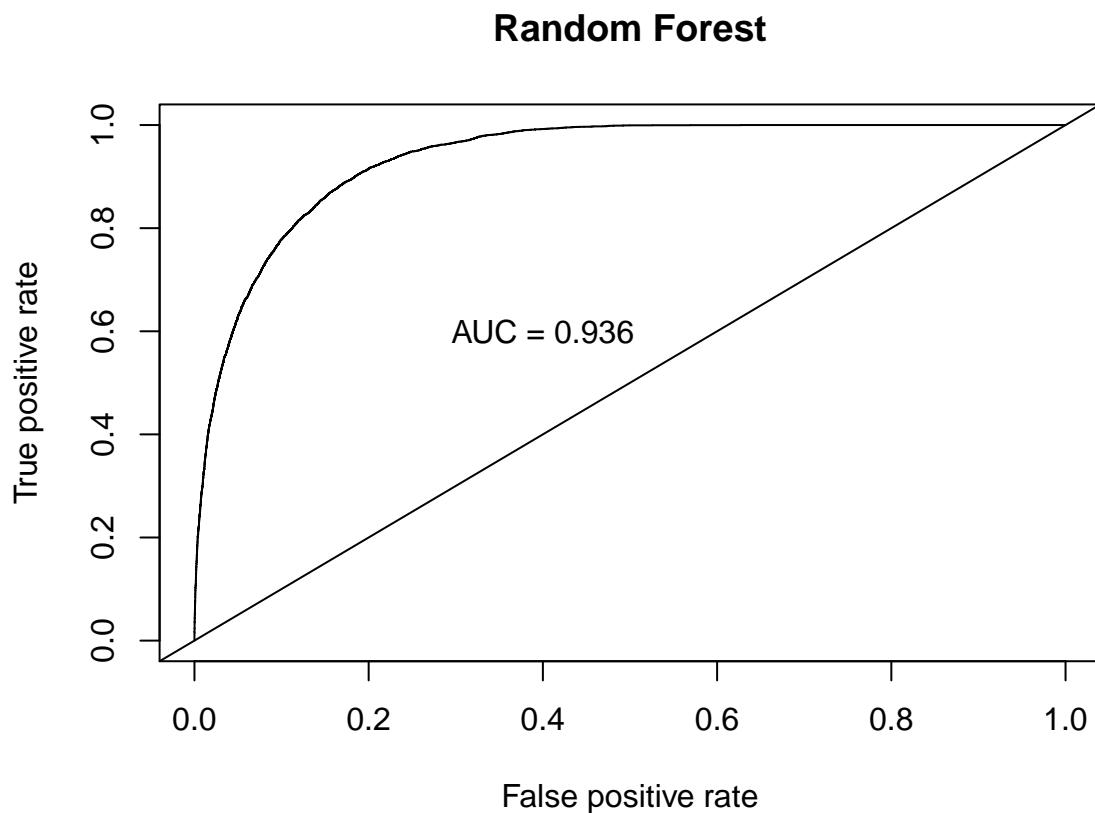
```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction      0      1
##           0 40687   654
##           1  8192  5049
##
##          Accuracy : 0.8379
##          95% CI : (0.8348, 0.841)
##  No Information Rate : 0.8955
##  P-Value [Acc > NIR] : 1
##
##          Kappa : 0.4532
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.8853
##          Specificity  : 0.8324
##  Pos Pred Value  : 0.3813
##  Neg Pred Value  : 0.9842
##          Prevalence  : 0.1045
```

```

##          Detection Rate : 0.0925
##    Detection Prevalence : 0.2426
##    Balanced Accuracy : 0.8589
##
##          'Positive' Class : 1
##
##          Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -3.1290 -0.2718  0.1166  0.4911  2.9315
##
##          Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -1.8990163  0.1206717 -15.737 < 2e-16 ***
## boosts                 0.2958458  0.0113382  26.093 < 2e-16 ***
## assists                 0.6267201  0.0669282   9.364 < 2e-16 ***
## weaponsAcquired        0.0374587  0.0090652   4.132 3.59e-05 ***
## killPlace                -0.0532964  0.0012909 -41.285 < 2e-16 ***
## walkDistance              0.0006996  0.0001289   5.426 5.78e-08 ***
## walkBin(500,1e+03]        0.8116992  0.1374167   5.907 3.49e-09 ***
## walkBin(1e+03,1.5e+03]     1.4153735  0.1718194   8.238 < 2e-16 ***
## walkBin(1.5e+03,2e+03]     1.6105048  0.2222116   7.248 4.24e-13 ***
## walkBin(2e+03,2.5e+03]     1.5435278  0.2788404   5.536 3.10e-08 ***
## walkBin(2.5e+03,3e+03]     1.4517335  0.3393104   4.278 1.88e-05 ***
## walkBin(3e+03,3.5e+03]     1.4327575  0.4041083   3.545 0.000392 ***
## walkBin(3.5e+03,4e+03]     1.0128776  0.4709296   2.151 0.031492 *
## walkBin(4e+03,4.5e+03]     0.9021646  0.5469129   1.650 0.099033 .
## walkBin(4.5e+03,5e+03]     0.4101287  0.6243381   0.657 0.511244
## walkBin(5e+03,2e+04]     -0.0089293  0.7459099  -0.012 0.990449
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 34792  on 25102  degrees of freedom
## Residual deviance: 15745  on 25087  degrees of freedom
##   (1511 observations deleted due to missingness)
## AIC: 15777

```

```
##  
## Number of Fisher Scoring iterations: 7
```



```
## Confusion Matrix and Statistics  
##  
##           Reference  
## Prediction      0      1  
##           0 37382   611  
##           1  7796  4869  
##  
##           Accuracy : 0.834  
##                 95% CI : (0.8308, 0.8373)  
##     No Information Rate : 0.8918  
##     P-Value [Acc > NIR] : 1  
##  
##           Kappa : 0.4543  
##  
## McNemar's Test P-Value : <2e-16  
##  
##           Sensitivity : 0.88850  
##           Specificity  : 0.82744
```

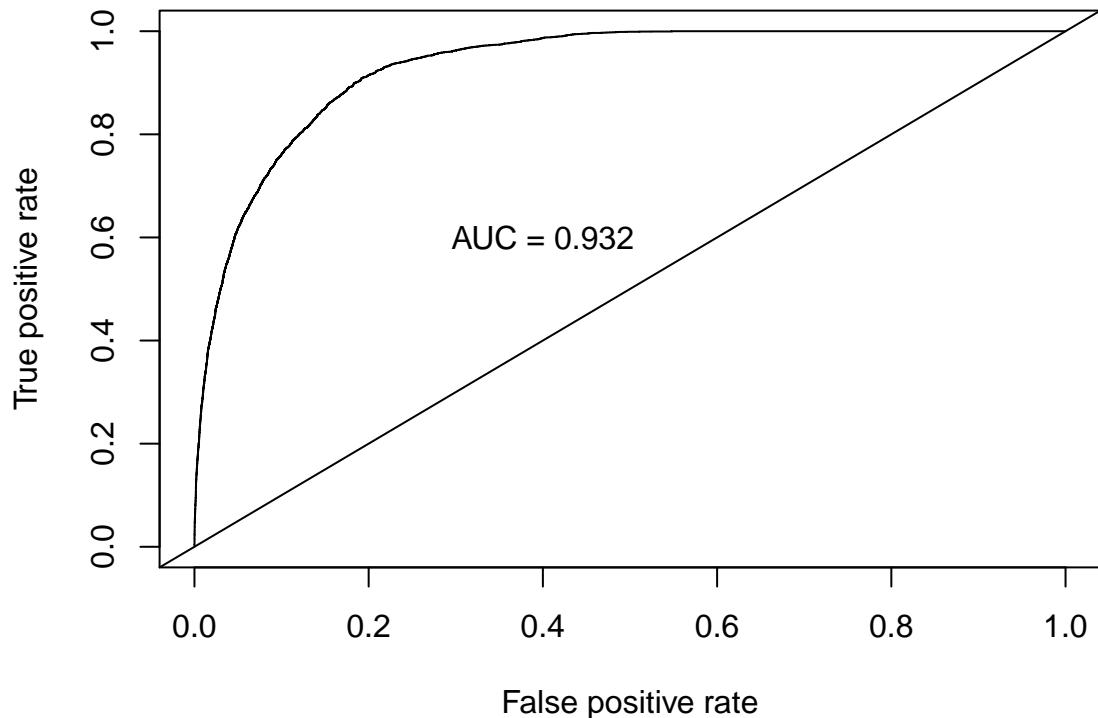
```

##          Pos Pred Value : 0.38445
##          Neg Pred Value : 0.98392
##          Prevalence : 0.10818
##          Detection Rate : 0.09612
##          Detection Prevalence : 0.25001
##          Balanced Accuracy : 0.85797
##
##          'Positive' Class : 1
##

##
## Call:
## glm(formula = top.10 ~ boosts + assists + weaponsAcquired + killPlace +
##      walkDistance + kpCut + kpCut:killPlace, family = binomial(link = "logit"),
##      data = train_obj1)
##
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -3.3065  -0.3720  -0.0150   0.4891   2.6470
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -4.829e-01  6.256e-02 -7.719 1.17e-14 ***
## boosts                 2.992e-01  1.103e-02  27.134 < 2e-16 ***
## assists                6.274e-01  6.671e-02   9.405 < 2e-16 ***
## weaponsAcquired       7.646e-02  7.762e-03   9.851 < 2e-16 ***
## killPlace              -6.674e-02  1.494e-03 -44.660 < 2e-16 ***
## walkDistance            7.525e-04  2.096e-05  35.907 < 2e-16 ***
## kpCut1                 -4.442e-01  7.149e-02  -6.213 5.21e-10 ***
## killPlace:kpCut1       1.371e-02  2.457e-03   5.582 2.38e-08 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 36895  on 26613  degrees of freedom
## Residual deviance: 17339  on 26606  degrees of freedom
## AIC: 17355
##
## Number of Fisher Scoring iterations: 6

```

## Random Forest



```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction      0      1
##           0 41140    774
##           1  7739   4929
##
##          Accuracy : 0.844
##                 95% CI : (0.841, 0.8471)
##     No Information Rate : 0.8955
##     P-Value [Acc > NIR] : 1
##
##          Kappa : 0.4586
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.8643
##          Specificity  : 0.8417
##     Pos Pred Value : 0.3891
##     Neg Pred Value : 0.9815
##          Prevalence  : 0.1045
```

```

##          Detection Rate : 0.0903
##      Detection Prevalence : 0.2321
##      Balanced Accuracy : 0.8530
##
##      'Positive' Class : 1
##

##  

## Call:  

## glm(formula = top.10 ~ boosts + assists + weaponsAcquired + killPlace +  

##      walkDistance + damageCut:streakCut, family = binomial(link = "logit"),  

##      data = train_obj1)  

##  

## Deviance Residuals:  

##      Min        1Q    Median        3Q       Max  

## -3.6348  -0.2734  -0.0068   0.4847   2.5701  

##  

## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error z value Pr(>|z|)  

## (Intercept) -1.077e+00 7.153e-02 -15.062 <2e-16 ***  

## boosts      3.599e-01 1.174e-02  30.645 <2e-16 ***  

## assists     7.779e-01 6.885e-02  11.299 <2e-16 ***  

## weaponsAcquired 5.215e-02 8.279e-03   6.299  3e-10 ***  

## killPlace    -9.611e-02 1.810e-03 -53.093 <2e-16 ***  

## walkDistance 9.176e-04 2.355e-05  38.968 <2e-16 ***  

## damageCut0:streakCut0 2.188e+00 7.842e-02  27.904 <2e-16 ***  

## damageCut1:streakCut0 1.638e+00 1.941e-01   8.438 <2e-16 ***  

## damageCut0:streakCut1 4.937e-01 5.585e-02   8.839 <2e-16 ***  

## damageCut1:streakCut1        NA         NA         NA         NA  

## ---  

## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  

##  

## (Dispersion parameter for binomial family taken to be 1)  

##  

## Null deviance: 36895  on 26613  degrees of freedom  

## Residual deviance: 16238  on 26605  degrees of freedom  

## AIC: 16256  

##  

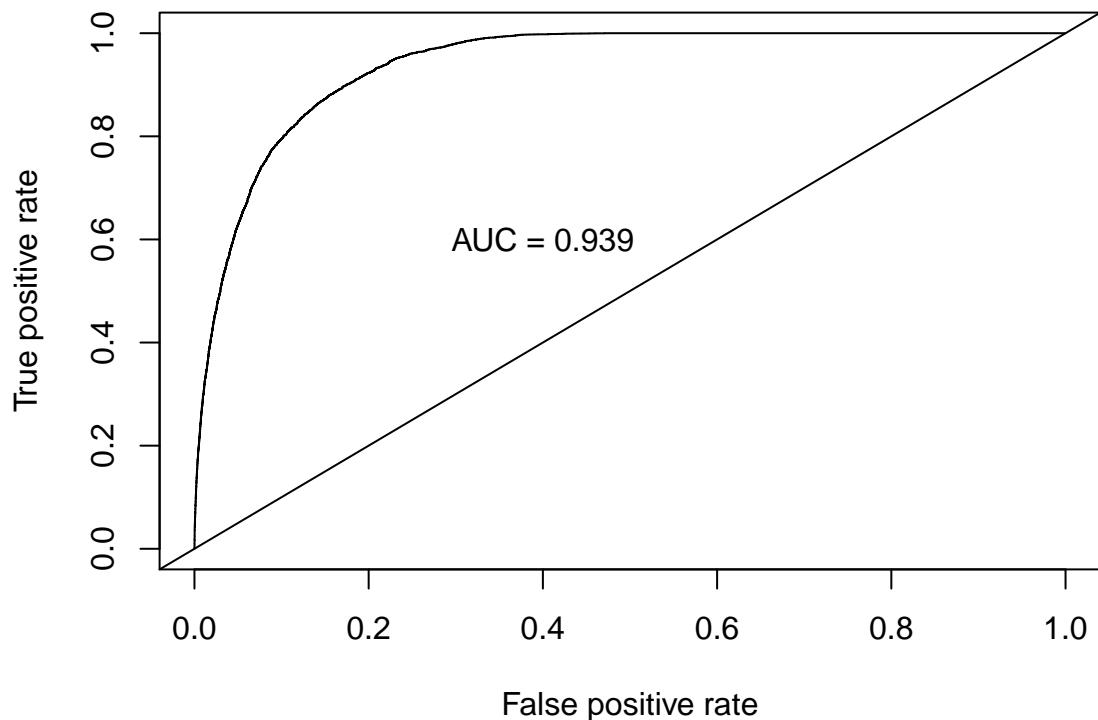
## Number of Fisher Scoring iterations: 6

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type  

## == : prediction from a rank-deficient fit may be misleading

```

## Random Forest



```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction      0      1
##           0 40879   639
##           1  8000  5064
##
##          Accuracy : 0.8417
## 95% CI : (0.8386, 0.8448)
## No Information Rate : 0.8955
## P-Value [Acc > NIR] : 1
##
##          Kappa : 0.4613
##
## McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.88795
## Specificity : 0.83633
## Pos Pred Value : 0.38763
## Neg Pred Value : 0.98461
## Prevalence : 0.10448
```

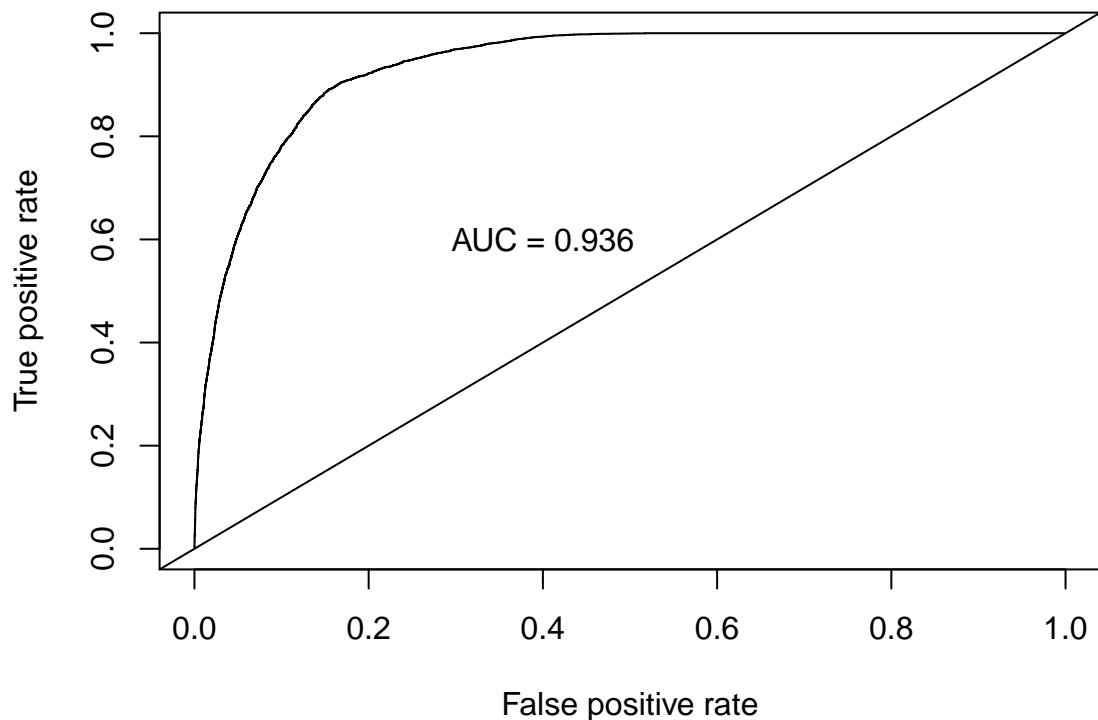
```

##           Detection Rate : 0.09278
##     Detection Prevalence : 0.23935
##     Balanced Accuracy : 0.86214
##
##           'Positive' Class : 1
##

## Call:
## glm(formula = top.10 ~ boosts + assists + weaponsAcquired + killPlace +
##       walkDistance + killsPK, family = binomial(link = "logit"),
##       data = train_obj1)
##
## Deviance Residuals:
##      Min        1Q     Median        3Q        Max
## -3.5511  -0.3216  -0.0002   0.4906   2.9651
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)            2.866e-01 7.051e-02  4.064 4.82e-05 ***
## boosts                 3.581e-01 1.159e-02 30.896 < 2e-16 ***
## assists                7.086e-01 6.731e-02 10.526 < 2e-16 ***
## weaponsAcquired       6.123e-02 7.994e-03  7.659 1.87e-14 ***
## killPlace               -7.958e-02 1.471e-03 -54.091 < 2e-16 ***
## walkDistance            6.951e-04 2.127e-05 32.683 < 2e-16 ***
## killsPK                -7.938e-01 3.420e-02 -23.213 < 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: 36895  on 26613  degrees of freedom
## Residual deviance: 16839  on 26607  degrees of freedom
## AIC: 16853
##
## Number of Fisher Scoring iterations: 6

```

## Random Forest



```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction      0      1
##           0 41066    600
##           1  7813   5103
##
##          Accuracy : 0.8459
##          95% CI : (0.8428, 0.8489)
##  No Information Rate : 0.8955
##  P-Value [Acc > NIR] : 1
##
##          Kappa : 0.4715
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.89479
##          Specificity  : 0.84016
##  Pos Pred Value  : 0.39509
##  Neg Pred Value  : 0.98560
##          Prevalence  : 0.10448
```

```

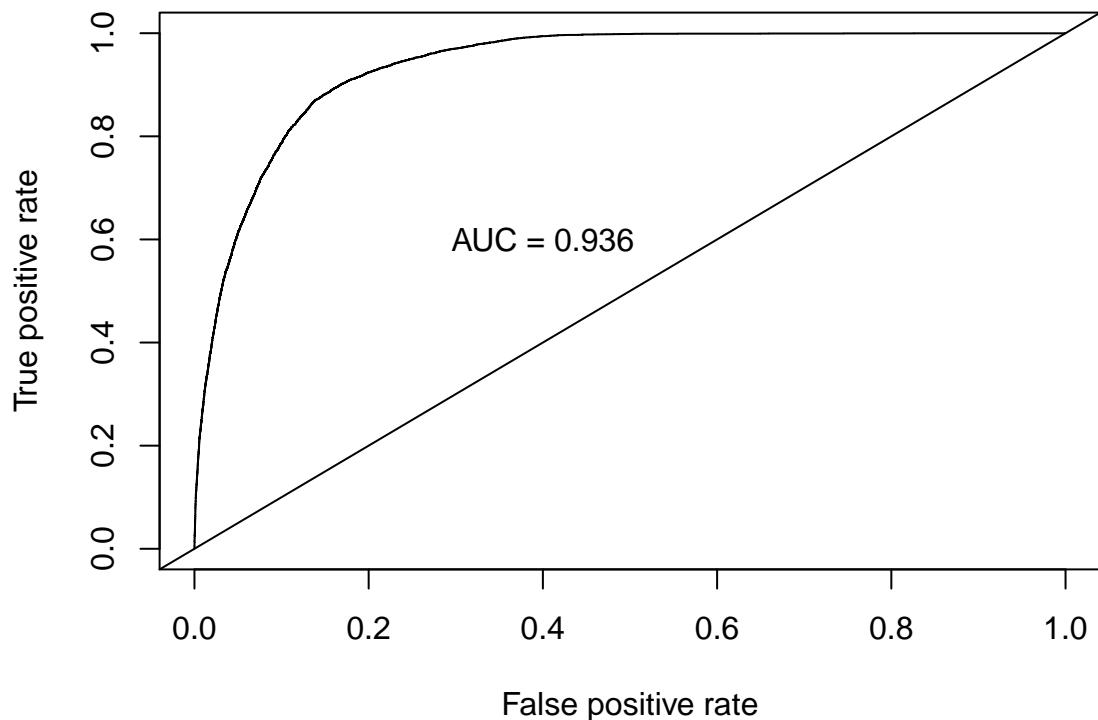
##          Detection Rate : 0.09349
##    Detection Prevalence : 0.23663
##    Balanced Accuracy : 0.86747
##
##    'Positive' Class : 1
##

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

##
## Call:
## glm(formula = top.10 ~ boosts + assists + weaponsAcquired + killPlace +
##     walkDistance + damageKill + killsPK, family = binomial(link = "logit"),
##     data = train_obj1)
##
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -3.5382  -0.3095   0.0055   0.4915   8.4904
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)           4.936e-01  7.445e-02   6.630 3.36e-11 ***
## boosts                3.565e-01  1.160e-02  30.740 < 2e-16 ***
## assists               6.536e-01  6.773e-02   9.650 < 2e-16 ***
## weaponsAcquired      5.806e-02  8.057e-03   7.206 5.74e-13 ***
## killPlace              -8.295e-02 1.534e-03 -54.067 < 2e-16 ***
## walkDistance          7.371e-04  2.203e-05  33.454 < 2e-16 ***
## damageKill            -3.780e+01  4.138e+00 -9.133 < 2e-16 ***
## killsPK                -6.874e-01  3.653e-02 -18.820 < 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: 36895  on 26613  degrees of freedom
## Residual deviance: 16738  on 26606  degrees of freedom
## AIC: 16754
##
## Number of Fisher Scoring iterations: 7

```

## Random Forest



```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction      0      1
##           0 40958   602
##           1  7921  5101
##
##          Accuracy : 0.8438
##          95% CI : (0.8408, 0.8469)
##  No Information Rate : 0.8955
##  P-Value [Acc > NIR] : 1
##
##          Kappa : 0.4674
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.89444
##          Specificity  : 0.83795
##  Pos Pred Value  : 0.39172
##  Neg Pred Value  : 0.98551
##          Prevalence  : 0.10448
```

```

##          Detection Rate : 0.09346
##    Detection Prevalence : 0.23858
##    Balanced Accuracy : 0.86619
##
##    'Positive' Class : 1
##

```

## 5.1.4 LDA

### 5.1.4.1 LASSO

```

# Reduce variables to relevant continuous variables
train1<-train[,c(5:6,8:10,12:15,19,21:27,29:30)]
test1<-test[,c(5:6,8:10,12:15,19,21:27,29:30)]
# train2 <- train1[,c(1,4:19)]
# test2 <- test1[,c(1,4:19)]
train2_alt <- train1[,c(1,4:17,19)]
test2_alt <- test1[,c(1,4:17,19)]

# Get data in format for LASSO
x=model.matrix(top.10~,train2_alt)[,-1]
y=as.numeric(train2_alt$top.10)

xtest<-model.matrix(top.10~,test2_alt)[,-1]
ytest<-as.numeric(test2_alt$top.10)

grid=10^seq(10,-2, length =100)
lasso.mod=glmnet(x,y,alpha=1, lambda =grid)

set.seed(23) #removes kills roadKills vehicleDestroys
lda.cv.out=cv.glmnet(x,y,alpha=1,family="binomial") #alpha=1 performs LASSO
# plot(lda.cv.out)

# simplest model
lda.lasso.model.coef<-coef(cv.out, cv.out$lambda.1se)
# lda.lasso.model.coef

lda.lasso.model.coef

## 16 x 1 sparse Matrix of class "dgCMatrix"
##                               1
## (Intercept) 3.091783e+00
## boosts      2.672920e-01

```

```

## heals           -2.784586e-03
## killPlace      -8.033586e-02
## kills          .
## killStreaks    -1.117635e+00
## longestKill    -5.733176e-05
## matchDuration  -1.609689e-03
## rankPoints     1.034136e-05
## rideDistance   1.948066e-04
## roadKills      .
## swimDistance   5.967117e-04
## teamKills      -1.509184e+00
## vehicleDestroys .
## walkDistance   8.511812e-04
## weaponsAcquired 5.746021e-02

# Removing the kills, roadKills, and vehicleDestroys as shown above
# Also removing rankPoints because
# (a) the majority of the players in these data aren't ranked and
# (b) the Kaggle description says "This ranking is inconsistent and is being deprecated in the API's next
train_final <- train2_alt[,c(-4,-8,-10,-13)]

```

#### 5.1.4.2 Assumption Checking

```

train_final_no <- train_final[which(train_final$top.10==0),]
train_final_yes <- train_final[which(train_final$top.10==1),]

# nrow(train_final)
# nrow(train_final_no)
# nrow(train_final_yes)

max_cols<-ncol(train_final)
no_matrix<-as.matrix(train_final_no[, -max_cols])
yes_matrix<-as.matrix(train_final_yes[, -max_cols])

no_hist<-multi.hist(no_matrix)
yes_hist<-multi.hist(yes_matrix)
# par(mfrow=c(1,1))

no_hist

## NULL

```

```

yes_hist

## NULL

#http://www.sthda.com/english/wiki/ggplot2-quick-correlation-matrix-heatmap-r-software-and-data-visuali
get_upper_tri <- function(cormat){
  cormat[lower.tri(cormat)]<- NA
  return(cormat)
}

custom_corr_plot <- function(cormat){

  upper_tri <- get_upper_tri(cormat)
  # Melt the correlation matrix
  melted_cormat <- melt(upper_tri, na.rm = TRUE)
  # Create a ggheatmap
  ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value))+
    geom_tile(color = "white")+
    scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                         midpoint = 0, limit = c(-1,1), space = "Lab",
                         name="Pearson\nCorrelation") +
    theme_minimal() # minimal theme
    theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                      size = 12, hjust = 1))+ 
    coord_fixed()

  p<-ggheatmap +
    geom_text(aes(Var2, Var1, label = value), color = "black", size = 4) +
    theme(
      axis.title.x = element_blank(),
      axis.title.y = element_blank(),
      panel.grid.major = element_blank(),
      panel.border = element_blank(),
      panel.background = element_blank(),
      axis.ticks = element_blank(),
      legend.justification = c(1, 0),
      legend.position = c(0.6, 0.7),
      legend.direction = "horizontal")+
    guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                                 title.position = "top", title.hjust = 0.5))

  return(p)
}

```

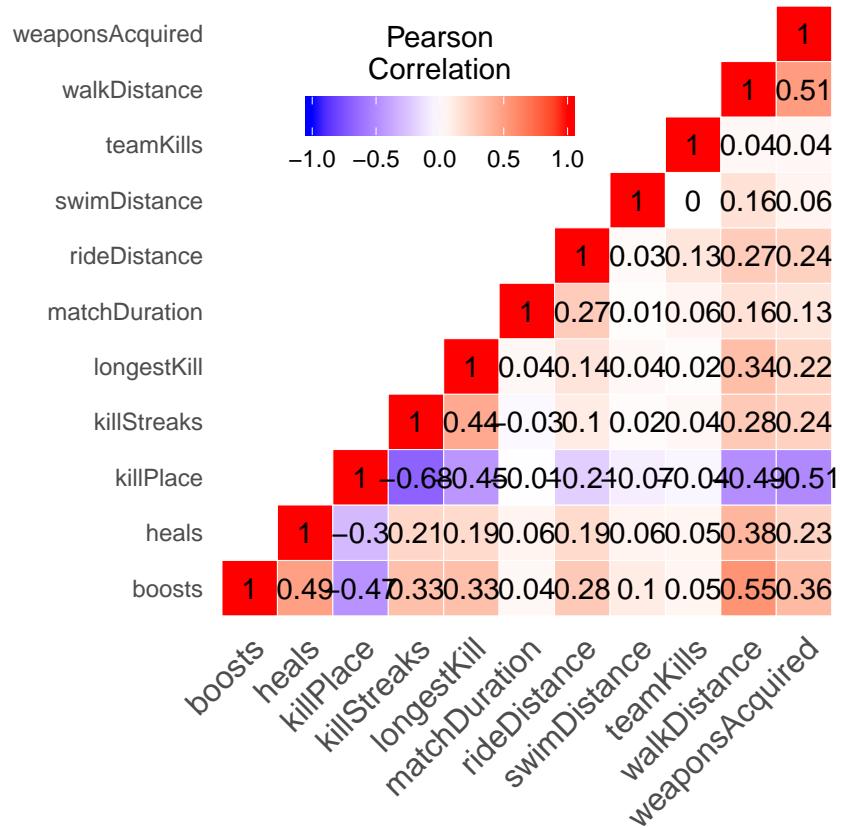


Figure 1: (#fig:code.lda.no )Correlation Matrix for Bottom 90 of Training Dataset

```

}

cormat <- round(cor(no_matrix),2)
lda.no <- custom_corr_plot(cormat)

cormat <- round(cor(yes_matrix),2)
lda.yes <- custom_corr_plot(cormat)

```

```
lda.no
```

```
lda.yes
```

#### 5.1.4.3 LDA Results

```

# Run LDA
lda <- lda(top.10 ~ . , data=train_final, prior=c(.9,.1))

```

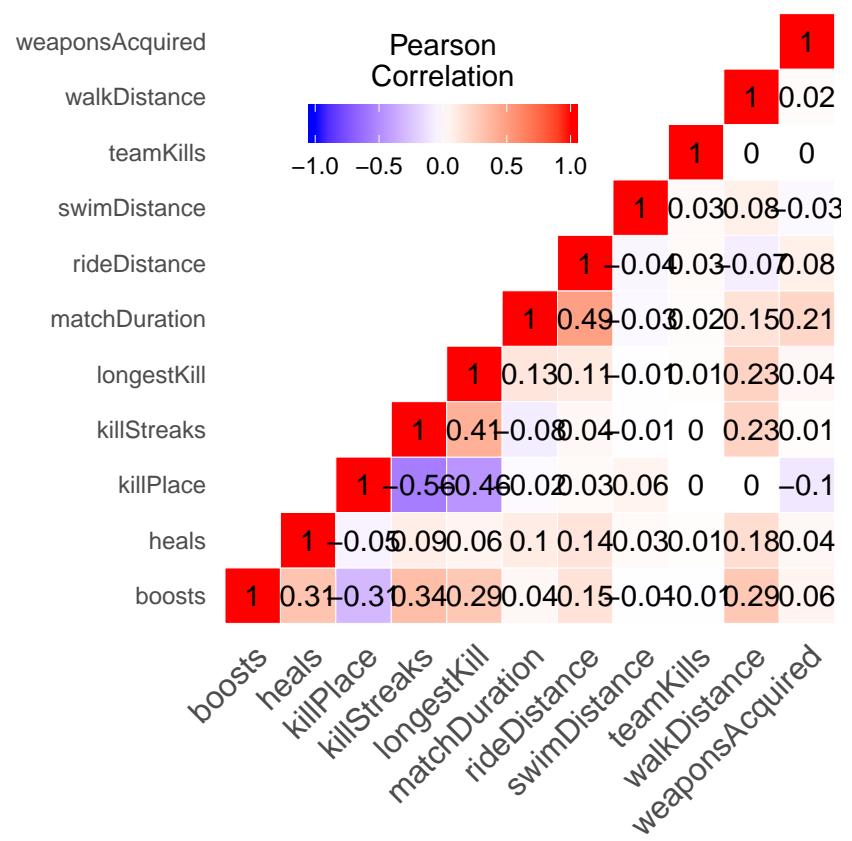


Figure 2: (#fig:code.lda.yes )Correlation Matrix for Top 90 of Training Dataset

```

lda

## Call:
## lda(top.10 ~ ., data = train_final, prior = c(0.9, 0.1))
##
## Prior probabilities of groups:
##   0   1
## 0.9 0.1
##
## Group means:
##           boosts      heals killPlace killStreaks longestKill matchDuration
## 0 0.7381078 0.8007815 50.25445 0.3915984 15.00272 1681.215
## 1 3.9244758 2.7710228 13.42301 0.8962200 70.77070 1680.553
##   rideDistance swimDistance teamKills walkDistance weaponsAcquired
## 0      515.6142     5.365372 0.016908394    823.7333     3.517848
## 1     1666.6767    16.136750 0.003005937    2351.6425     5.710528
##
## Coefficients of linear discriminants:
##                               LD1
## boosts                  0.1636559657
## heals                  -0.0071779101
## killPlace                -0.0313509817
## killStreaks               -0.4645084746
## longestKill                0.0003963294
## matchDuration             -0.0007443145
## rideDistance              0.0001033605
## swimDistance              0.0006812726
## teamKills                 -1.0147862026
## walkDistance              0.0004217550
## weaponsAcquired            0.0155413057

predict <- predict(lda,newdata=test)

test_final <- test
test_final$predcited_place <- as.vector(predict$class)
test_final$top.10<-as.factor(test_final$top.10)
test_final$predcited_place<-as.factor(test_final$predcited_place)

xtab<-table(test_final$predcited_place,test_final$top.10)
confusionMatrix(xtab, positive="1")

## Confusion Matrix and Statistics

```

```

## 
##          0      1
##  0 46972 2425
##  1 1907 3278
##
##          Accuracy : 0.9206
##          95% CI : (0.9183, 0.9229)
##  No Information Rate : 0.8955
##  P-Value [Acc > NIR] : < 2.2e-16
##
##          Kappa : 0.5582
##
##  Mcnemar's Test P-Value : 3.998e-15
##
##          Sensitivity : 0.57479
##          Specificity : 0.96099
##          Pos Pred Value : 0.63221
##          Neg Pred Value : 0.95091
##          Prevalence : 0.10448
##          Detection Rate : 0.06006
##          Detection Prevalence : 0.09499
##          Balanced Accuracy : 0.76789
##
##          'Positive' Class : 1
## 

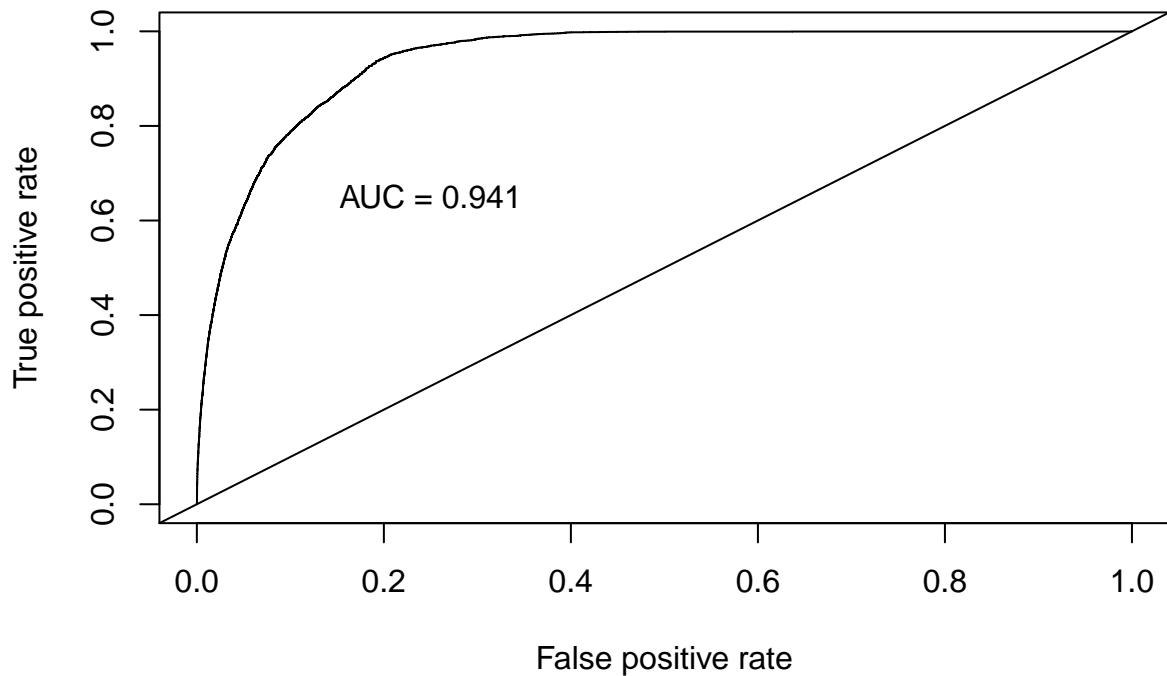
predict.posteriors <- as.data.frame(predict$posterior)

# Evaluate the model
pred <- prediction(predict.posteriors[,2], test$top.10)
roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred, measure = "auc")
auc.train <- auc.train@y.values

# Plot
plot(roc.perf, main="ROC Curve - LDA")
abline(a=0, b= 1)
text(x = .25, y = .65 ,paste("AUC = ", round(auc.train[[1]],3), sep = ""))

```

## ROC Curve – LDA



### 5.1.4.4 QDA Results

```
qda <- qda(top.10 ~ ., data=train_final, prior=c(.9,.1))

qda

## Call:
## qda(top.10 ~ ., data = train_final, prior = c(0.9, 0.1))
##
## Prior probabilities of groups:
##   0   1
## 0.9 0.1
##
## Group means:
##           boosts      heals killPlace killStreaks longestKill matchDuration
## 0  0.7381078 0.8007815  50.25445    0.3915984     15.00272    1681.215
## 1  3.9244758 2.7710228  13.42301    0.8962200     70.77070    1680.553
##   rideDistance swimDistance teamKills walkDistance weaponsAcquired
## 0      515.6142      5.365372 0.016908394     823.7333      3.517848
## 1     1666.6767     16.136750 0.003005937    2351.6425      5.710528
```

```

predict.qda <- predict(qda,newdata=test)

# test_final <- test
test_final$predcited_place.qda <- as.vector(predict$class)
# test_final$top.10<-as.factor(test_final$top.10)
test_final$predcited_place.qda<-as.factor(test_final$predcited_place.qda)
# test_final$predcited_place<-as.factor(test_final$predcited_place)
# str(test_final)

xtab.qda<-table(test_final$predcited_place.qda,test_final$top.10)
confusionMatrix(xtab.qda, positive="1")

## Confusion Matrix and Statistics
##
##
##          0      1
##    0 46972  2425
##    1 1907   3278
##
##          Accuracy : 0.9206
##                  95% CI : (0.9183, 0.9229)
##      No Information Rate : 0.8955
##      P-Value [Acc > NIR] : < 2.2e-16
##
##          Kappa : 0.5582
##
##  Mcnemar's Test P-Value : 3.998e-15
##
##          Sensitivity : 0.57479
##          Specificity  : 0.96099
##      Pos Pred Value : 0.63221
##      Neg Pred Value : 0.95091
##          Prevalence  : 0.10448
##      Detection Rate : 0.06006
##      Detection Prevalence : 0.09499
##          Balanced Accuracy : 0.76789
##
##      'Positive' Class : 1
##

predict.posteriors.qda <- as.data.frame(predict.qda$posterior)

# Evaluate the model

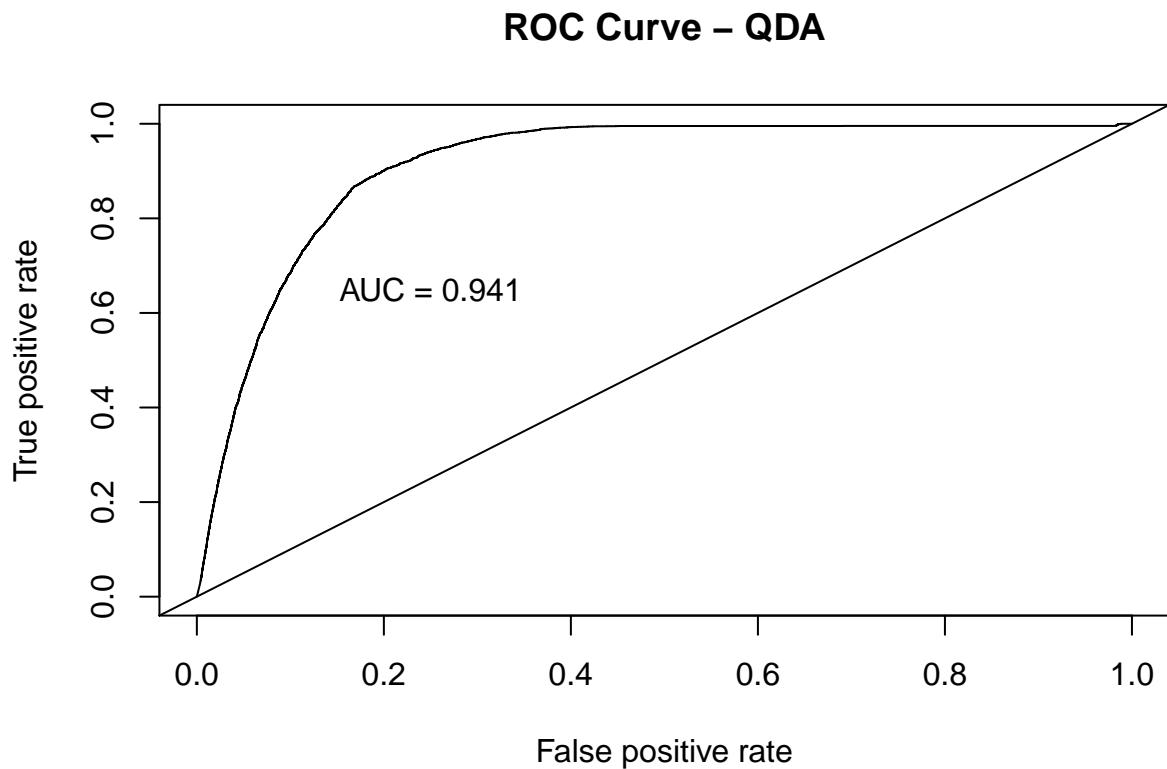
```

```

pred.qda <- prediction(predict.posteriors.qda[, 2], test$top.10)
roc.perf.qda = performance(pred.qda, measure = "tpr", x.measure = "fpr")
auc.train.qda <- performance(pred.qda, measure = "auc")
auc.train.qda <- auc.train.qda@y.values

# Plot
plot(roc.perf.qda, main="ROC Curve - QDA")
abline(a=0, b= 1)
text(x = .25, y = .65 ,paste("AUC = ", round(auc.train[[1]],3), sep = ""))

```



## 5.1.5 Random Forest

### 5.1.5.1 Model

```

# results='hide', message=FALSE, include=FALSE, echo=FALSE
rf_cols_to_remove = c("Id", "groupId", "matchId", "matchType", "DBNOs", "revives", "winPlacePerc")

train.rf <- train %>%
  dplyr::select(-rf_cols_to_remove) %>%

```

```

mutate(kills = kills * 100 / numGroups) %>% # Normalized Kills
mutate(matchDuration = as.factor(ifelse(matchDuration < mean(matchDuration), "Low", "High"))) %>%
mutate(top.10 = factor(top.10, labels = c("No", "Yes")))

test.rf <- test %>%
dplyr::select(-rf_cols_to_remove) %>%
mutate(kills = kills * 100 / numGroups) %>% # Normalized Kills
mutate(matchDuration = as.factor(ifelse(matchDuration < mean(matchDuration), "Low", "High"))) %>%
mutate(top.10 = factor(top.10, labels = c("No", "Yes")))

set.seed(1234)
model.rf <- randomForest(as.factor(top.10) ~ .,
                           data = train.rf,
                           ntree = 500,
                           mtry = 12,
                           cutoff = c(0.4, 1-0.40))

model.rf

```

### 5.1.5.2 Tuning Plot

```
plot(model.rf)
```

### 5.1.5.3 Variable Importance

```

# varImp(model.rf)

rf.imp.variables <- data.frame(varImp(model.rf))

# rf.imp.variables[order(rf.imp.variables$Overall, decreasing = T),]

kable(data.frame('Column Name' = rownames(rf.imp.variables),
                 'Overall Score'= rf.imp.variables$Overall),
      "latex", booktabs = T) %>%
kable_styling(position = "center")

```

### 5.1.5.4 Predict

```
prd.rf.train <- predict(model.rf, train.rf)
prd.rf.test <- predict(model.rf, test.rf)
```

#### 5.1.5.5 Confusion Matrix

```
confusionMatrix(data=prd.rf.test,
                 reference=test.rf$top.10, "Yes")
```

#### 5.1.5.6 ROC Curve

```
plotRoc <- function(preds, truth) {
  pred <- prediction(preds, truth)
  roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
  auc.train <- performance(pred, measure = "auc")
  auc.train <- auc.train@y.values
  #Plot ROC
  par(mar=c(4,4,4,4))
  plot(roc.perf,main="Random Forest")
  abline(a=0, b= 1) #Ref line indicating poor performance
  text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
  table(test$top.10, useNA = "ifany")
}

plotRoc(as.integer(prd.rf.test),as.integer(test.rf$top.10))

# ?randomForest
```