Classification on CS:GO matches

Andrew Gerungan

Logistic regression is actually a method of classification rather than regression. It is a type of linear classification that takes probabilities within the range of [0,1] for the sigmoid function. It's strength is mostly in that it helps separate classes if linearly separable, doesn't only work with quantitative data, is computationally inexpensive, and outputs a probabilistic result that can be used for various applications. The main weakness of logistic regression is that it's prone to underfitting since it can't really process more complex, non-linear boundaries well.

Setting everything up

The code below loads in the dataset "economy.csv" which has stats about various CS:GO matches and their associated map, team matchup, and some stats about each round's economy

```
df <- read.csv("economy.csv")
str(df)
```

```
'data.frame':   43234 obs. of  99 variables:
 $ date      : chr  "2020-03-01" "2020-03-01" "2020-03-01" "2020-02-29" ...
 $ match_id  : int  2339402 2339402 2339402 2339401 2339401 2339400 2339400 2339400 2339873 2339873 ...
 $ event_id  : int  4901 4901 4901 4901 4901 4901 4901 4901 5234 5234 ...
 $ team_1    : chr  "G2" "G2" "G2" "Natus Vincere" ...
 $ team_2    : chr  "Natus Vincere" "Natus Vincere" "Natus Vincere" "Astralis" ...
 $ best_of   : chr  "5" "5" "5" "3" ...
 $ X_map     : chr  "Nuke" "Dust2" "Mirage" "Dust2" ...
 $ t1_start  : chr  "t" "ct" "t" "t" ...
 $ t2_start  : chr  "ct" "t" "ct" "ct" ...
 $ X1_t1     : num  4350 3900 4150 4150 4200 4550 4450 4550 4250 4250 ...
 $ X2_t1     : num  1100 7400 14300 18050 10000 ...
 $ X3_t1     : num  22100 23250 2000 21000 22000 ...
 $ X4_t1     : num  9350 28500 24800 25850 24500 ...
 $ X5_t1     : num  25750 31900 9000 25000 27550 ...
 $ X6_t1     : num  10400 31700 23150 25000 29350 ...
 $ X7_t1     : num  24600 18950 21850 27250 31950 ...
 $ X8_t1     : num  8150 30200 23700 26150 31850 ...
 $ X9_t1     : num  26700 28650 10450 26300 31750 ...
 $ X10_t1    : num  23400 30350 26250 27850 32850 ...
 $ X11_t1    : num  4300 30150 8800 26750 32150 ...
 $ X12_t1    : num  25900 11100 24950 27450 31750 ...
 $ X13_t1    : num  11950 23700 12100 27850 31850 ...
 $ X14_t1    : num  24850 8550 24350 18300 33050 ...
 $ X15_t1    : num  21900 26350 18250 27850 33150 ...
 $ X16_t1    : num  4150 4050 4300 4000 4250 4250 4000 3650 3450 4250 ...
 $ X17_t1    : num  10650 9400 19400 21100 3000 ...
 $ X18_t1    : num  27300 21900 8900 9100 21150 ...
 $ X19_t1    : num  27000 12700 NA 3100 11750 ...
 $ X20_t1    : num  30950 11300 NA 22100 28050 ...
 $ X21_t1    : num  NA 3100 NA 23350 25900 ...
 $ X22_t1    : num  NA 26250 NA NA NA ...
 $ X23_t1    : num  NA 21300 NA NA NA ...
 $ X24_t1    : num  NA 23950 NA NA NA ...
 $ X25_t1    : num  NA 27450 NA NA NA ...
 $ X26_t1    : num  NA 27550 NA NA NA ...
 $ X27_t1    : num  NA 28050 NA NA NA ...
 $ X28_t1    : num  NA 26250 NA NA NA ...
 $ X29_t1    : num  NA 26250 NA NA NA ...
 $ X30_t1    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ X1_t2     : num  4250 3500 4200 4250 4200 4200 3950 4450 4050 4200 ...
 $ X2_t2     : num  20250 18550 21200 9300 17950 ...
 $ X3_t2     : num  24600 7300 24150 2100 9050 ...
 $ X4_t2     : num  29300 1500 27050 19300 1000 ...
 $ X5_t2     : num  30650 21800 31350 27250 22850 ...
 $ X6_t2     : num  31450 12000 30650 9850 7500 ...
 $ X7_t2     : num  32050 21050 31050 24800 24400 ...
 $ X8_t2     : num  31450 24450 28150 12200 24700 ...
 $ X9_t2     : num  32050 6850 29650 27550 9700 ...
 $ X10_t2    : num  30950 26850 30950 26100 27150 ...
 $ X11_t2    : num  29300 23100 31550 7450 21850 ...
 $ X12_t2    : num  30200 25650 32950 27650 21900 ...
 $ X13_t2    : num  31550 26800 32250 24500 8150 ...
 $ X14_t2    : num  32350 26750 31650 19100 25200 ...
 $ X15_t2    : num  33050 28250 33350 18050 20450 ...
 $ X16_t2    : num  4250 4000 4350 4150 4300 4100 4150 4250 4200 4200 ...
 $ X17_t2    : num  20300 18850 14300 15100 19300 ...
 $ X18_t2    : num  8250 15850 22850 18450 27400 ...
 $ X19_t2    : num  1300 23000 NA 23150 31700 ...
 $ X20_t2    : num  22200 26850 NA 28150 29650 ...
 $ X21_t2    : num  NA 29100 NA 27850 12150 ...
 $ X22_t2    : num  NA 26300 NA NA NA ...
 $ X23_t2    : num  NA 26850 NA NA NA ...
 $ X24_t2    : num  NA 19050 NA NA NA ...
 $ X25_t2    : num  NA 3500 NA NA NA ...
 $ X26_t2    : num  NA 26450 NA NA NA ...
 $ X27_t2    : num  NA 27450 NA NA NA ...
 $ X28_t2    : num  NA 27500 NA NA NA ...
 $ X29_t2    : num  NA 29050 NA NA NA ...
 $ X30_t2    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ X1_winner : num  2 2 2 1 2 2 2 2 1 1 ...
 $ X2_winner : num  2 1 2 1 1 1 2 2 2 1 ...
 $ X3_winner : num  2 1 2 1 1 2 2 2 1 2 ...
 $ X4_winner : num  2 1 2 2 1 1 1 1 2 2 ...
 $ X5_winner : num  2 1 2 1 1 1 2 1 2 2 ...
 $ X6_winner : num  2 1 2 1 1 2 1 2 2 2 ...
 $ X7_winner : num  2 2 1 1 2 1 1 2 1 1 ...
 $ X8_winner : num  2 1 2 1 1 1 2 2 2 1 ...
 $ X9_winner : num  1 1 2 1 1 2 1 1 2 2 ...
 $ X10_winner: num  2 2 2 1 1 2 1 2 2 2 ...
 $ X11_winner: num  2 2 2 1 1 2 1 2 1 2 ...
 $ X12_winner: num  2 2 2 1 1 2 2 2 1 2 ...
 $ X13_winner: num  2 2 2 1 1 2 1 1 1 1 ...
 $ X14_winner: num  2 2 2 1 1 2 1 1 1 2 ...
 $ X15_winner: num  2 2 2 1 1 2 1 1 2 1 ...
 $ X16_winner: num  2 2 1 1 2 1 2 2 2 1 ...
 $ X17_winner: num  1 1 2 2 2 2 2 2 2 1 ...
 $ X18_winner: num  1 2 2 2 2 2 1 2 1 2 ...
 $ X19_winner: num  1 2 NA 2 1 2 1 1 2 2 ...
 $ X20_winner: num  2 2 NA 2 1 2 1 1 2 2 ...
 $ X21_winner: num  NA 2 NA 1 1 2 2 2 1 2 ...
 $ X22_winner: num  NA 1 NA NA NA 1 1 2 1 2 ...
 $ X23_winner: num  NA 1 NA NA NA 1 2 2 1 1 ...
 $ X24_winner: num  NA 1 NA NA NA 1 1 1 1 2 ...
 $ X25_winner: num  NA 1 NA NA NA 1 2 1 1 2 ...
 $ X26_winner: num  NA 1 NA NA NA 1 1 1 1 NA ...
 $ X27_winner: num  NA 2 NA NA NA 1 1 1 2 NA ...
 $ X28_winner: num  NA 2 NA NA NA 2 NA 2 1 NA ...
 $ X29_winner: num  NA 2 NA NA NA NA NA NA 1 NA ...
 $ X30_winner: num  NA NA NA NA NA NA NA NA NA NA ...
```

We then isolate the columns that we are concerned about, those being the Team matchup, what type of series it is (best of 1, best of 3, best of 5), what map the match is on, and which side each team starts on. We then double check that there aren't any nulls in the dataset

```
df <- df[,c(4,5,6,7,8,9)] #grab team_1, team_2, best_of, X_map, t1_start, t2_start


df$team_1 <- factor(df$team_1)
df$team_2 <- factor(df$team_2)
df$best_of <- factor(df$best_of)
df$X_map <- factor(df$X_map)
df$t1_start <- factor(df$t1_start)
df$t2_start <- factor(df$t2_start)

sapply(df, function(x) sum(is.na(x)==TRUE))
```

```
  team_1   team_2  best_of    X_map t1_start t2_start 
       0        0        0        0        0        0 
```

Now we split our data into train/test (80/20)

```
set.seed(1234)
i <- sample(1:nrow(df), .8*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Data Exploration

We run a summary on the dataset and each of its columns before checking the head/tail of best_of and X_map; I already have an idea of a relationship I want to check - some maps don't see a lot of play outside of bo5s since certain maps are banned a lot. I want to see if I can determine what type of series the match is based on the map played - this should work for some of the more fringe maps (hopefully). I also checked the dim of the dataset and the head, and tail of the two columns

```
summary(train)
```

```
     team_1              team_2      best_of        X_map     
 AGO    :  393   HAVU       :  342   1: 9758   Mirage  :6987  
 HAVU   :  328   AGO        :  334   2: 1004   Inferno :6807  
 Tricked:  315   Tricked    :  286   3:23289   Train   :4892  
 Heroic :  305   mousesports:  279   5:  533   Overpass:4255  
 AVANGAR:  303   pro100     :  265   o:    3   Nuke    :3311  
 forZe  :  301   Liquid     :  262             Cache   :3079  
 (Other):32642   (Other)    :32819             (Other) :5256  
 t1_start   t2_start  
 ct:17358   ct:17229  
 t :17229   t :17358  
```

```
dim(train)
```

```
[1] 34587     6
```

```
summary(train$team_1)
```

```
                AGO                HAVU             Tricked 
                393                 328                 315 
             Heroic             AVANGAR               forZe 
                305                 303                 301 
             Spirit                FaZe          Virtus.pro 
                301                 293                 287 
   ALTERNATE aTTaX              Liquid              Nemiga 
                283                 276                 276 
          Astralis             Windigo               TYLOO 
                271                 266                 265 
              FURIA                 NRG              Chiefs 
                264                 259                 256 
             Sprout         Singularity               North 
                253                 251                 248 
        mousesports          Complexity       Natus Vincere 
                243                 240                 239 
                 G2           Renegades           Grayhound 
                238                 238                 232 
               LDLC            Valiance              Cloud9 
                232                 225                 223 
             fnatic                 NiP               OpTic 
                222                 221                 221 
             pro100         HellRaisers             Epsilon 
                219                 215                 214 
                BIG                  SJ                ViCi 
                205                 202                 192 
            Kinguin             TeamOne          Luminosity 
                191                 190                 188 
               ENCE                Envy     Movistar Riders 
                183                 182                 176 
           EURONICS               Nexus             eUnited 
                172                 172                 171 
              ORDER              MVP PK                PACT 
                168                 162                 162 
             DETONA          Fragsters      Space Soldiers 
                159                 158                 158 
        Ground Zero              Isurus            Vitality 
                157                 157                 155 
             Gambit              5POWER         DreamEaters 
                151                 150                 147 
                W7M             Imperial                INTZ 
                147                 146                 146 
        Red Reserve             GODSENT               Chaos 
                145                 142                 140 
          BOOT-d[S]           Winstrike               Ghost 
                139                 139                 138 
       Swole Patrol Quantum Bellator Fire       Tainted Minds 
                137                 134                 134 
              x-kom                  SK               AVANT 
                133                 132                 131 
               Bpro                MIBR              Sharks 
                126                 126                 126 
           Endpoint    Copenhagen Flames               Rogue 
                124                 123                 122 
             Unique                 EPG         Izako Boars 
                120                 118                 117 
            GoodJob               PRIDE       Vega Squadron 
                116                 115                 114 
            eXtatus          GamerLegion              Mythic 
                110                 110                 109 
            Bravado               3DMAX               Syman 
                108                 107                 107 
              Flash              Legacy         Lucid Dream 
                106                 100                  97 
          Alpha Red            Heretics               EHOME 
                 96                  95                  94 
            (Other) 
              16294 
```

```
summary(train$team_2)
```

```
               HAVU                 AGO             Tricked 
                342                 334                 286 
        mousesports              pro100              Liquid 
                279                 265                 262 
            AVANGAR              Nemiga             Windigo 
                260                 260                 260 
         Virtus.pro              Sprout                  G2 
                250                 241                 236 
             Spirit               forZe     ALTERNATE aTTaX 
                226                 222                 220 
                 SJ              fnatic              Heroic 
                219                 217                 213 
             Cloud9                 NiP         HellRaisers 
                211                 211                 208 
        Singularity              5POWER     Movistar Riders 
                196                 194                 194 
                BIG                LDLC                FaZe 
                191                 191                 190 
         Complexity               North              Chiefs 
                189                 189                 187 
              OpTic           Renegades             Epsilon 
                187                 184                 180 
          Grayhound            Valiance          Luminosity 
                179                 176                 175 
              FURIA               ORDER             Kinguin 
                171                 168                 165 
      Natus Vincere         Red Reserve                Envy 
                165                 165                 163 
              TYLOO                 NRG              MVP PK 
                163                 162                 161 
        DreamEaters            Astralis             TeamOne 
                159                 158                 153 
               PACT              Gambit              Isurus 
                152                 150                 150 
            GODSENT               Nexus           Winstrike 
                144                 144                 141 
              x-kom                INTZ               Chaos 
                139                 138                 137 
             Sharks            Endpoint            Imperial 
                137                 136                 134 
  Copenhagen Flames             eUnited               PRIDE 
                133                 132                 131 
Quantum Bellator Fire               Syman               Ghost 
                131                 131                 130 
               MIBR            EURONICS         Ground Zero 
                130                 129                 128 
        Izako Boars               AVANT            Vitality 
                127                 123                 122 
          BOOT-d[S]               EHOME       Tainted Minds 
                120                 120                 119 
                EPG           Fragsters               Rogue 
                118                 118                 117 
                W7M                ENCE        Swole Patrol 
                117                 116                 116 
            eXtatus      Space Soldiers             GoodJob 
                112                 112                 110 
             Mythic                  SK         GamerLegion 
                109                 109                 108 
             Unique                Bpro           Nordavind 
                108                 106                 104 
              Vexed              Legacy                ViCi 
                 99                  98                  97 
      x6tence Galaxy               3DMAX               Flash 
                 95                  92                  90 
      Vega Squadron            Heretics              Beyond 
                 90                  89                  86 
            (Other) 
              18486 
```

```
summary(train$best_of)
```

```
    1     2     3     5     o 
 9758  1004 23289   533     3 
```

```
summary(train$X_map)
```

```
      Cache Cobblestone     Default       Dust2     Inferno 
       3079        1609         159        2921        6807 
     Mirage        Nuke    Overpass       Train     Vertigo 
       6987        3311        4255        4892         567 
```

```
summary(train$t1_start)
```

```
   ct     t 
17358 17229 
```

```
summary(train$t2_start)
```

```
   ct     t 
17229 17358 
```

```
head(train$X_map)
```

```
[1] Inferno     Cobblestone Train       Overpass    Overpass   
[6] Inferno    
10 Levels: Cache Cobblestone Default Dust2 Inferno ... Vertigo
```

```
head(train$best_of)
```

```
[1] 1 3 1 1 2 3
Levels: 1 2 3 5 o
```

```
tail(train$X_map)
```

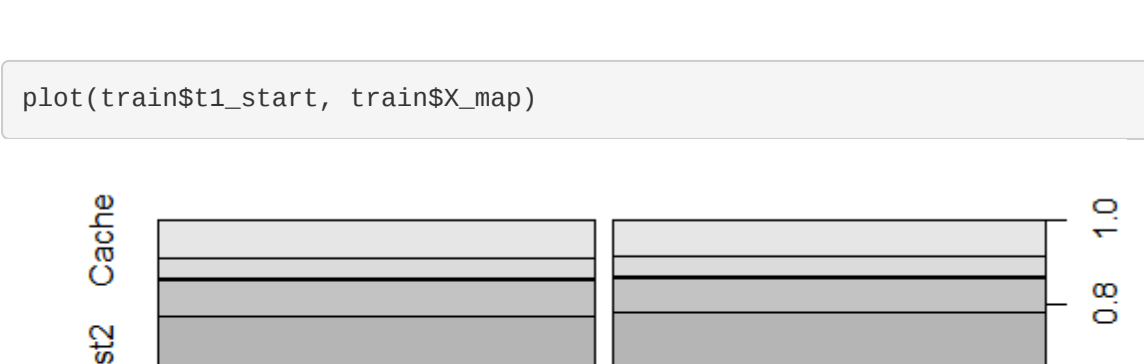
```
[1] Cobblestone Inferno     Mirage      Overpass    Inferno    
[6] Nuke       
10 Levels: Cache Cobblestone Default Dust2 Inferno ... Vertigo
```

```
tail(train$best_of)
```

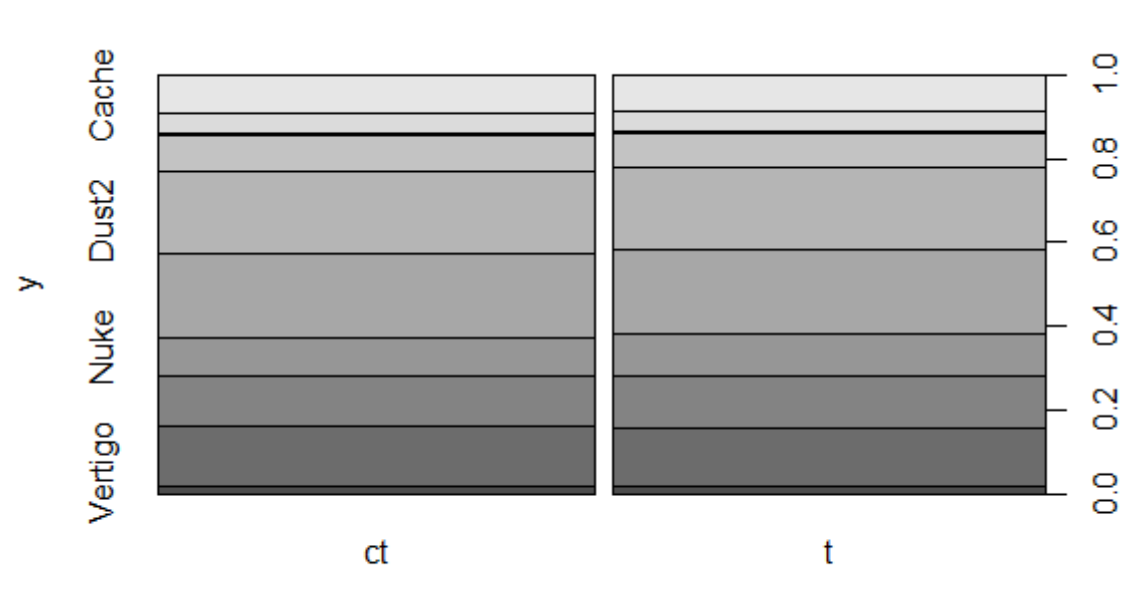
```
[1] 3 1 3 3 3 3
Levels: 1 2 3 5 o
```

We then plot map against the type of series, along with t1_start against the map, since the map may influence which side team 1 starts

```
plot(train$X_map, train$best_of)
```




```
plot(train$t1_start, train$X_map)
```



…definitely going with map and type of series

Logistic Regression

We're going to go ahead and run logistic regression on the type of series, seeing if the map is a good predictor

```
glm1 <- glm(best_of~X_map, data=train, family="binomial")
summary(glm1)
```

```

Call:
glm(formula = best_of ~ X_map, family = "binomial", data = train)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.9482  -1.5226   0.7640   0.8557   0.9404  

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)    
(Intercept)       0.78270    0.03884  20.152  < 2e-16 ***
X_mapCobblestone -0.19588    0.06492  -3.017  0.00255 ** 
X_mapDefault     13.78337   70.00611   0.197  0.84392    
X_mapDust2        0.47226    0.05909   7.992 1.32e-15 ***
X_mapInferno      0.02097    0.04686   0.448  0.65450    
X_mapMirage       0.03351    0.04671   0.717  0.47315    
X_mapNuke         0.42140    0.05666   7.438 1.02e-13 ***
X_mapOverpass     0.29937    0.05246   5.707 1.15e-08 ***
X_mapTrain        0.13531    0.05011   2.700  0.00693 ** 
X_mapVertigo      0.95259    0.12389   7.689 1.48e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 41155  on 34586  degrees of freedom
Residual deviance: 40774  on 34577  degrees of freedom
AIC: 40794

Number of Fisher Scoring iterations: 13
```

The deviance is pretty bad on this logistic regression model, let's see how it works with the test set

```
probs <- predict(glm1, newdata=test, type="response")
pred <- ifelse(probs>0.5, 1, 0)
acc <- mean(pred==test$best_of)
print(paste("accuracy=", acc))
```

```
[1] "accuracy= 0.278362437839713"
```

```
table(pred, test$best_of)
```

```

pred    1    2    3    5    o
   1 2407  258 5866  115    1
```

Wow the accuracy is bad; also it somehow predicts a possible best-of-0 series, which is pretty amusing. We're gonna utilize a confusion matrix to get a couple more stats on the model though.

```
library(caret)
confusionMatrix(as.factor(pred), reference=test$best_of)
```

```
 警告:  Levels are not in the same order for reference and data. Refactoring data to match.
```

```
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    5    o
         1 2407  258 5866  115    1
         2    0    0    0    0    0
         3    0    0    0    0    0
         5    0    0    0    0    0
         o    0    0    0    0    0

Overall Statistics
                                          
               Accuracy : 0.2784          
                 95% CI : (0.2689, 0.2879)
    No Information Rate : 0.6784          
    P-Value [Acc > NIR] : 1               
                                          
                  Kappa : 0               
                                          
 Mcnemar's Test P-Value : NA              

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 5
Sensitivity            1.0000  0.00000   0.0000   0.0000
Specificity            0.0000  1.00000   1.0000   1.0000
Pos Pred Value         0.2784      NaN      NaN      NaN
Neg Pred Value            NaN  0.97016   0.3216   0.9867
Prevalence             0.2784  0.02984   0.6784   0.0133
Detection Rate         0.2784  0.00000   0.0000   0.0000
Detection Prevalence   1.0000  0.00000   0.0000   0.0000
Balanced Accuracy      0.5000  0.50000   0.5000   0.5000
                      Class: o
Sensitivity          0.0000000
Specificity          1.0000000
Pos Pred Value             NaN
Neg Pred Value       0.9998844
Prevalence           0.0001156
Detection Rate       0.0000000
Detection Prevalence 0.0000000
Balanced Accuracy    0.5000000
```

```
#ROC doesn't work since not binary
```

Yeah this model is pretty bad, both the Kappa and P-value are really bad. Let's try using the Naive-Bayes algorithm instead

```
library(e1071)
nb1 <- naiveBayes(best_of~X_map, data=train)
nb1
```

```

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
           1            2            3            5            o 
2.821291e-01 2.902825e-02 6.733455e-01 1.541041e-02 8.673779e-05 

Conditional probabilities:
   X_map
Y          Cache  Cobblestone      Default        Dust2
  1 0.0989956958 0.0589260094 0.0000000000 0.0664070506
  2 0.1215139442 0.0438247012 0.0009960159 0.0796812749
  3 0.0838163940 0.0412641161 0.0021898750 0.0923182618
  5 0.0731707317 0.0544090056 0.2007504690 0.0806754221
  o 0.0000000000 0.0000000000 0.0000000000 0.0000000000
   X_map
Y        Inferno       Mirage         Nuke     Overpass
  1 0.2157204345 0.2195121951 0.0782947325 0.1103709777
  2 0.1842629482 0.2201195219 0.0926294821 0.1155378486
  3 0.1899609258 0.1952423891 0.1038258405 0.1295890764
  5 0.1744840525 0.1425891182 0.0675422139 0.0806754221
  o 0.0000000000 0.3333333333 0.0000000000 0.3333333333
   X_map
Y          Train      Vertigo
  1 0.1430621029 0.0087108014
  2 0.1354581673 0.0059760956
  3 0.1416119198 0.0201812014
  5 0.1144465291 0.0112570356
  o 0.3333333333 0.0000000000
```

These probabilities look better (maybe?), so let's put it against the test set

```
p1 <- predict(nb1, newdata=test, type="class")
table(p1, test$best_of)
```

```

p1     1    2    3    5    o
  1    0    0    0    0    0
  2    0    0    0    0    0
  3 2407  257 5852   98    1
  5    0    1   14   17    0
  o    0    0    0    0    0
```

```
acc2 <- mean(p1==test$best_of)
print(paste("accuracy=", acc2))
```

```
[1] "accuracy= 0.678732508384411"
```

This accuracy is much better (the Naive-Bayes algorithm is way better) and actually correctly shows that maps are across bo3s and bo5s. Let's also take a look at the raw probabilities

```
p1_raw <- predict(nb1, newdata=test, type="raw")
head(p1_raw)
```

```
              1          2         3          5            o
[1,] 0.23074579 0.02808817 0.7302923 0.01087284 9.060699e-07
[2,] 0.28536386 0.02780049 0.6741619 0.01246934 2.044154e-04
[3,] 0.30924035 0.02717789 0.6499189 0.01366240 4.407226e-07
[4,] 0.28536386 0.02780049 0.6741619 0.01246934 2.044154e-04
[5,] 0.28536386 0.02780049 0.6741619 0.01246934 2.044154e-04
[6,] 0.05782142 0.00592554 0.3022025 0.63403274 1.777662e-05
```

The raw probabilities also look much better in comparison to the logistic regression from before, so overall we can conclude that this model produced through the Naive-Bayes algorithm is more accurate

Summary

The Naive Bayes algorithm is a classifier that is used most often for more sophisticated algorithms as a base but is also a fine classifier by itself. It's main strengths lie in it's simplicity - it's easy to both implement and interpret. It also works extremely well with small data sets and handles high dimensions admirably. However, it's weakness lie in that it's outperformed by other classifiers for larger data sets, it creates guess values that do not occur in the training data, and if the predictors aren't independent, its assumptions may limit its own performance.

Metrics Utilized

The accuracy value derived from the mean found between the predicted and actual values in the column help show, well, how accurate the model produced is.

The confusion matrix was utilized to mainly identify the p-value and the Kappa value of the model created. This is useful in identifying how balanced a data set is, since the Kappa value adjusts for the distribution. As seen in the first model created, it helped show that that model was not that accurate.

One of the metric omitted is the ROC, which plots true positive rate versus false positive rate; the reason it was omitted is one of its weaknesses: it doesn't work on non-binary values. It does provide valuable metrics on prediction and performance though.

The raw probabilities extracted in the Naive Bayes algorithm is useful in observing the raw information about the model, helping with understanding the model.