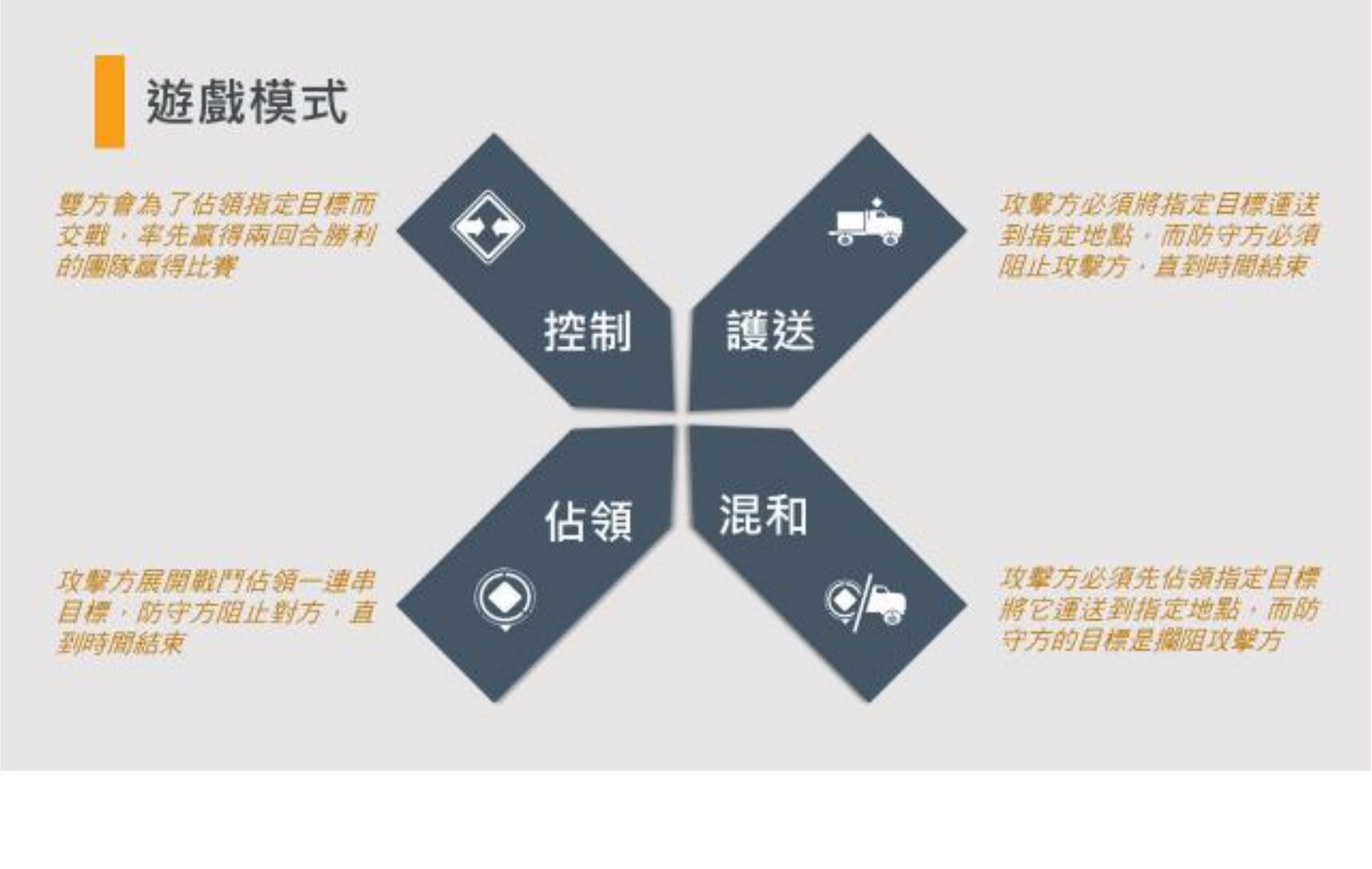
在鬥陣特攻中的會戰如何獲勝之分析

前言

鬥陣特攻(Overwatch)是一款由暴雪娛樂公司製作的第一人稱射擊遊戲，遊戲方式以 6vs6 為主，分成紅藍兩隊，在不同的地圖下，達成不同的條件的一方獲勝，一共有控制、護送、佔領、混和等模式。這次探討的目標是，在鬥陣特攻的戰鬥數據中找到如何獲勝的方式。



遊戲模式

雙方會為了佔領指定目標而交戰，率先贏得兩回合勝利的團隊贏得比賽

控制

護送

攻擊方必須將指定目標運送到指定地點，而防守方必須阻止攻擊方，直到時間結束

攻擊方展開戰鬥佔領一連串目標，防守方阻止對方，直到時間結束

佔領

混和

攻擊方必須先佔領指定目標將它運送到指定地點，而防守方的目標是攔阻攻擊方

數據收集

暴雪娛樂並沒有在官方網站公開玩家的生涯數據，通常要取得資料會由第三方網站取得，例如 Overbuff。但是在 2018/06/28 的更新檔，遊戲團隊決定可以讓玩家選擇是否公開自己的遊戲數據，導致這些第三方網站無法取得所有玩家的目前的遊戲資料。也就是說，從這類第三方網站擷取到的資料，很有可能是已經過期的數據，因此我決定不採用這種方式獲得的資料，轉而使用職業競賽的數據，這些數據不只可以得到不是過期的數據，並且數據的品質更好，畢竟職業競賽並不會像一般玩家競賽一樣可能會因為一些原因就不認真比賽。資料主要由 Winston's Lab 提供，收集了第一季職業聯賽、第二季職業選拔賽以及世界盃的每場賽事的數據，一共 21590 場會戰的數據。

何謂 "會戰"

在競技類的電腦遊戲裡面，幾乎所有的玩家都會聽過"會戰"這一個詞彙，是指雙方為了達成目標產生的衝突。但是其實大家都知道會戰的概念，我們對它卻沒有明確的定義，根據 Winston's Lab，在鬥陣特攻中的會戰定義如下

- 雙方拿下一個殺數，會戰就開始，如果兩個殺數的間隔超過 15 秒，則視為不同的會戰，雙方殺數多者為會戰獲勝方
- 若絕招在會戰前 12 秒發動，則視為會戰中的一部份，會戰結束後發動絕招則不視為會戰中的一部份

選擇變數以及資料

以下是原始資料中收集到的變數

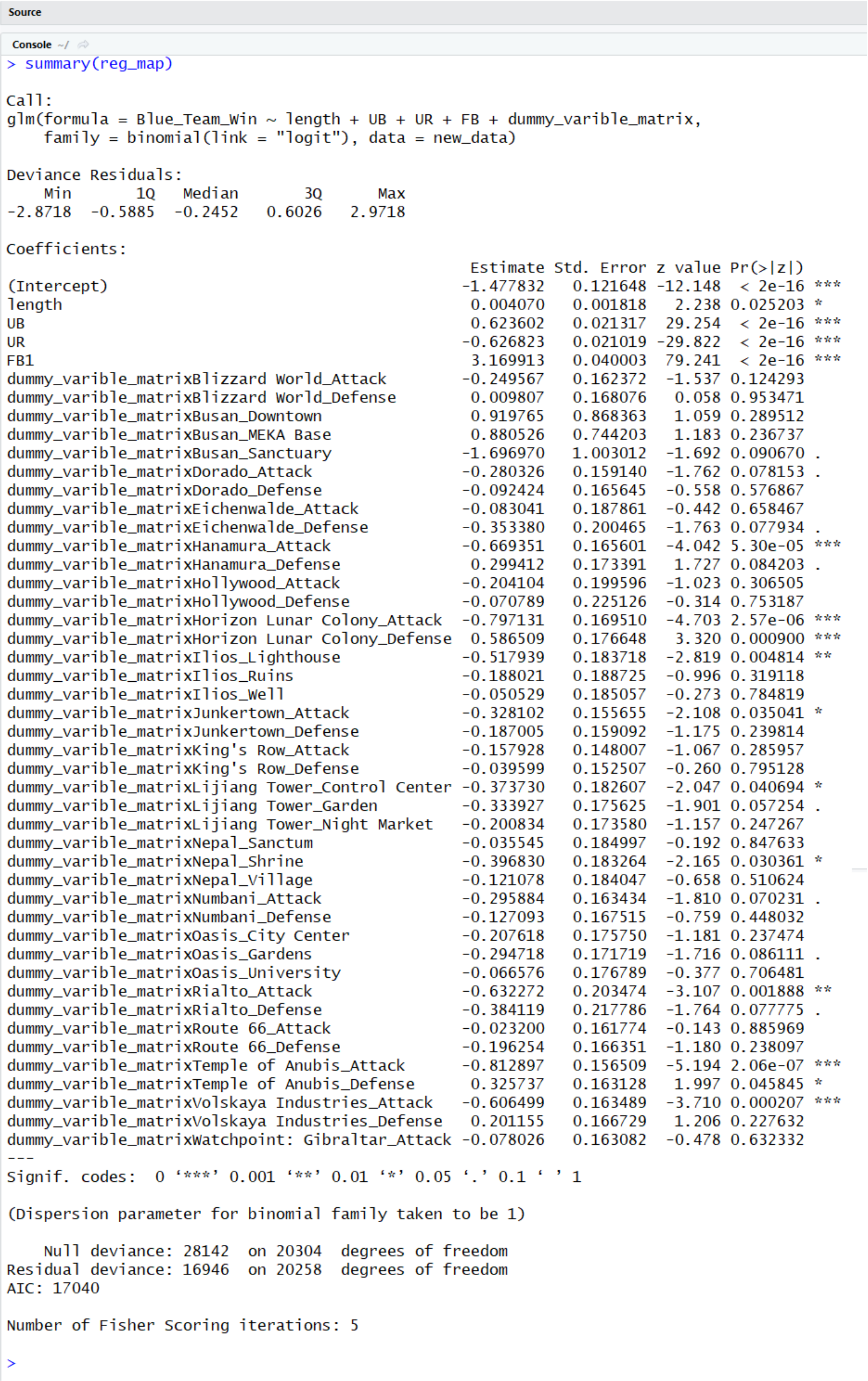
Length	會戰時間
UB	藍隊使用的絕招數量
UR	紅隊使用的絕招數量
FB	藍隊是否拿到會戰首殺
HP	地圖的小補血包數量
Mega_Hp	地圖的大補血包數量
Map_and_round_type	地圖_以及攻防類型(使用 Dummy Variables)
Blue_Team_win	藍隊是否在會戰中獲勝

由於我要探討的是對 Win 這個應變數做迴歸分析，Win 這個變數代表藍隊是否在會戰中獲勝，若獲勝則為 1，若沒有獲勝則為 0，若平手則為 0.5。

我認為地圖補血包數量並不是一個影響會戰的重要變數，因為有些補包並不會真的常被使用到，並且通常會有輔助來補血，並不常用補包，再者地圖類型與補包數量有共線性，因此 Hp 以及 Mega_Hp 並不考慮放進模型中。Map 變數由於屬於類別型變數 ，以最後一筆資料的地圖：”直布羅陀基地-防守”為基準點，加上其他變數對”Blue_Team_win”變數進行 Logistic Regression。

另外，由於在定義上，會戰會出現平局的狀況，可是在實際上卻不一定是平局，例如雙方殺數相同，但是其中一方卻奪得目標，雖然在定義上是平局，但實際上卻不是。因此，我把平局的數據刪除掉，這樣接下來在做模型以及預測時，都能獲得更好的結果。一共刪除了 1285 筆平局的數據。

回歸模型

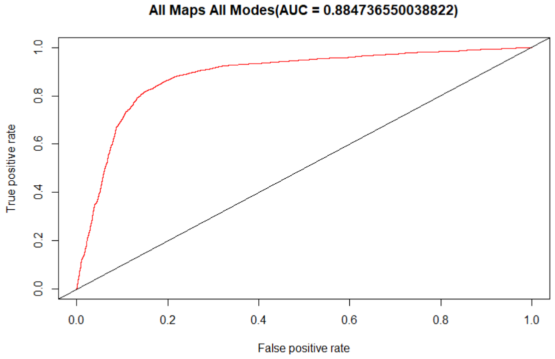


```
> summary(reg_map)

Call:
glm(formula = Blue_Team_Win ~ length + UB + UR + FB + dummy_varible_matrix,
    family = binomial(link = "logit"), data = new_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8718  -0.5885  -0.2452   0.6026   2.9718

Coefficients:
                                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                                      -1.477832   0.121648 -12.148  < 2e-16 ***
length                                            0.004070   0.001818   2.238 0.025203 *
UB                                                0.623602   0.021317  29.254  < 2e-16 ***
UR                                               -0.626823   0.021019 -29.822  < 2e-16 ***
FB1                                               3.169913   0.040003  79.241  < 2e-16 ***
dummy_varible_matrixBlizzard World_Attack        -0.249567   0.162372  -1.537 0.124293
dummy_varible_matrixBlizzard World_Defense        0.009807   0.168076   0.058 0.953471
dummy_varible_matrixBusan_Downtown                0.919765   0.868363   1.059 0.289512
dummy_varible_matrixBusan_MEKA Base               0.880526   0.744203   1.183 0.236737
dummy_varible_matrixBusan_Sanctuary              -1.696970   1.003012  -1.692 0.090670 .
dummy_varible_matrixDorado_Attack                -0.280326   0.159140  -1.762 0.078153 .
dummy_varible_matrixDorado_Defense               -0.092424   0.165645  -0.558 0.576867
dummy_varible_matrixEichenwalde_Attack           -0.083041   0.187861  -0.442 0.658467
dummy_varible_matrixEichenwalde_Defense          -0.353380   0.200465  -1.763 0.077934 .
dummy_varible_matrixHanamura_Attack              -0.669351   0.165601  -4.042 5.30e-05 ***
dummy_varible_matrixHanamura_Defense              0.299412   0.173391   1.727 0.084203 .
dummy_varible_matrixHollywood_Attack             -0.204104   0.199596  -1.023 0.306505
dummy_varible_matrixHollywood_Defense            -0.070789   0.225126  -0.314 0.753187
dummy_varible_matrixHorizon Lunar Colony_Attack  -0.797131   0.169510  -4.703 2.57e-06 ***
dummy_varible_matrixHorizon Lunar Colony_Defense  0.586509   0.176648   3.320 0.000900 ***
dummy_varible_matrixIlios_Lighthouse             -0.517939   0.183718  -2.819 0.004814 **
dummy_varible_matrixIlios_Ruins                  -0.188021   0.188725  -0.996 0.319118
dummy_varible_matrixIlios_Well                   -0.050529   0.185057  -0.273 0.784819
dummy_varible_matrixJunkertown_Attack            -0.328102   0.155655  -2.108 0.035041 *
dummy_varible_matrixJunkertown_Defense           -0.187005   0.159092  -1.175 0.239814
dummy_varible_matrixKing's Row_Attack            -0.157928   0.148007  -1.067 0.285957
dummy_varible_matrixKing's Row_Defense           -0.039599   0.152507  -0.260 0.795128
dummy_varible_matrixLijiang Tower_Control Center -0.373730   0.182607  -2.047 0.040694 *
dummy_varible_matrixLijiang Tower_Garden         -0.333927   0.175625  -1.901 0.057254 .
dummy_varible_matrixLijiang Tower_Night Market   -0.200834   0.173580  -1.157 0.247267
dummy_varible_matrixNepal_Sanctum                -0.035545   0.184997  -0.192 0.847633
dummy_varible_matrixNepal_Shrine                 -0.396830   0.183264  -2.165 0.030361 *
dummy_varible_matrixNepal_Village                -0.121078   0.184047  -0.658 0.510624
dummy_varible_matrixNumbani_Attack               -0.295884   0.163434  -1.810 0.070231 .
dummy_varible_matrixNumbani_Defense              -0.127093   0.167515  -0.759 0.448032
dummy_varible_matrixOasis_City Center            -0.207618   0.175750  -1.181 0.237474
dummy_varible_matrixOasis_Gardens                -0.294718   0.171719  -1.716 0.086111 .
dummy_varible_matrixOasis_University             -0.066576   0.176789  -0.377 0.706481
dummy_varible_matrixRialto_Attack                -0.632272   0.203474  -3.107 0.001888 **
dummy_varible_matrixRialto_Defense               -0.384119   0.217786  -1.764 0.077775 .
dummy_varible_matrixRoute 66_Attack              -0.023200   0.161774  -0.143 0.885969
dummy_varible_matrixRoute 66_Defense             -0.196254   0.166351  -1.180 0.238097
dummy_varible_matrixTemple of Anubis_Attack      -0.812897   0.156509  -5.194 2.06e-07 ***
dummy_varible_matrixTemple of Anubis_Defense      0.325737   0.163128   1.997 0.045845 *
dummy_varible_matrixVolskaya Industries_Attack   -0.606499   0.163489  -3.710 0.000207 ***
dummy_varible_matrixVolskaya Industries_Defense   0.201155   0.166729   1.206 0.227632
dummy_varible_matrixWatchpoint: Gibraltar_Attack -0.078026   0.163082  -0.478 0.632332
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 28142  on 20304  degrees of freedom
Residual deviance: 16946  on 20258  degrees of freedom
AIC: 17040

Number of Fisher Scoring iterations: 5

>
```

由以上報表可以看出來，一場會戰中影響勝負最大的其實是會戰首殺，這其實是相當合理的結果，會戰一開始如果失去了輔助或是攻擊角色，隊上的血量或是攻擊量就會受到影響，相當不容易拿下勝利。另外，可以看到某些地圖變數的攻擊方以及防守方的係數為一正一負，並且係數是負的都是地圖的攻擊方，原因在於攻擊方的抵達最後一個目標的距離比防守防還要長很多，因此造成攻擊方並不容易拿下會戰並贏得最後一個攻擊目標。

另外，我隨機從 20305 個樣本中找出 1000 個當作測試集，剩下的當作訓練集，畫出 ROC curve。這個模型的 ROC Curve 都在 X+Y = 1 這條直線的上方，並且 AUC 到達 0.885，代表這個模型預測的狀況相當不錯，另外，以 0.5 為切點，測試集的準確度也來到了 83%，以下為這個模型的

ROC Curve 以及 Confusion matrix。

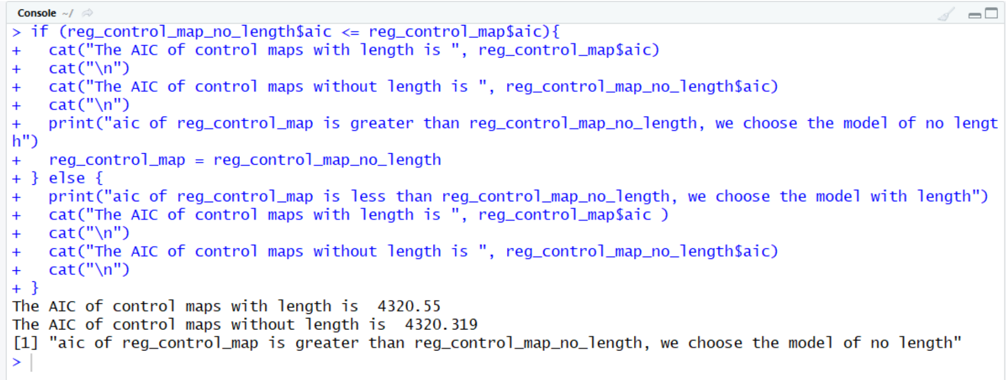


Cutoff = 0.5, accuracy = 0.828

實際狀況 預測	贏	輸
贏	380	90
輸	82	448

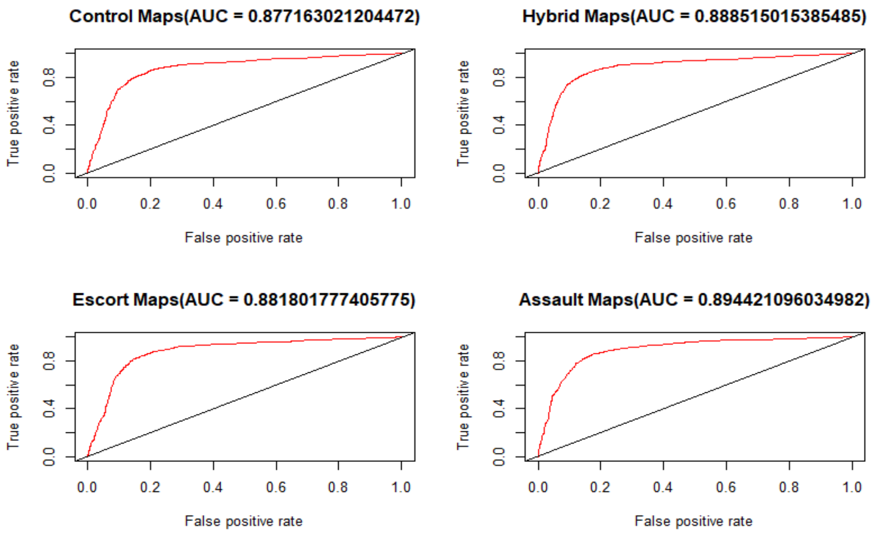
Length 變數

由於我認為會戰的時間”length”變數在護送、混和、佔領模式中並不會對勝負造成很大的影響。因此我決定把不同的模式分開來做模型，但是我並不確定在控制模式中，”length”變數會對勝負造成影響，因此我對控制模式做了兩種模型，一種是把”length”變數放入，另一種是沒有”length”變數放進模型中，結果沒有”length”變數的那個模型的 AIC 比”length”變數的模型的 AIC 小 0.23，雖然並沒有差很多，但是在少一個變數的狀況下 fitting 的狀況沒有差很多，我還是決定選沒有”length”變數的模型來做不同地圖類型下的 ROC curve。



```
> if (reg_control_map_no_length$aic <= reg_control_map$aic){
+   cat("The AIC of control maps with length is ", reg_control_map$aic)
+   cat("\n")
+   cat("The AIC of control maps without length is ", reg_control_map_no_length$aic)
+   cat("\n")
+   print("aic of reg_control_map is greater than reg_control_map_no_length, we choose the model of no lengt
h")
+   reg_control_map = reg_control_map_no_length
+ } else {
+   print("aic of reg_control_map is less than reg_control_map_no_length, we choose the model with length")
+   cat("The AIC of control maps with length is ", reg_control_map$aic )
+   cat("\n")
+   cat("The AIC of control maps without length is ", reg_control_map_no_length$aic)
+   cat("\n")
+ }
The AIC of control maps with length is  4320.55
The AIC of control maps without length is  4320.319
[1] "aic of reg_control_map is greater than reg_control_map_no_length, we choose the model of no length"
> 
```

四種模型的 ROC Curve



Control Maps(AUC = 0.877163021204472)

Hybrid Maps(AUC = 0.888515015385485)

Escort Maps(AUC = 0.881801777405775)

Assault Maps(AUC = 0.894421096034982)

Control Maps(Accuracy = 0.816)

實際狀況 預測	贏	輸
贏	192	39
輸	53	216

Escort Maps(Accuracy = 0.822)

實際狀況 預測	贏	輸
贏	223	51
輸	38	188

Hybrid Maps(Accuracy = 0.836)

實際狀況 預測	贏	輸
贏	204	45
輸	37	214

Assault Maps(Accuracy = 0.836)

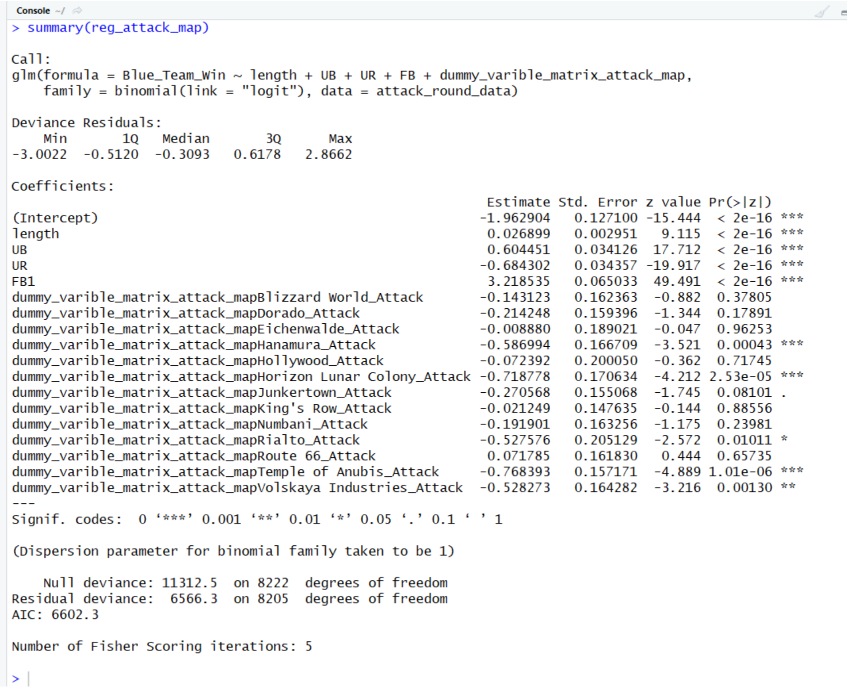
實際狀況 預測	贏	輸
贏	214	38
輸	44	204

以上是把不同遊戲模式分開來之後，各找了 500 個樣本當作測試集，做出來的 ROC Curve 和 confusion matrix，由上圖可知，AUC 跟原先的並沒有差很多，也就是說有沒有分開來做模型，預測的狀況都不會有明顯的差異，而且事實上 confusion matrix 也證實了相同的結果，因此這應該還不是比較好的做法。

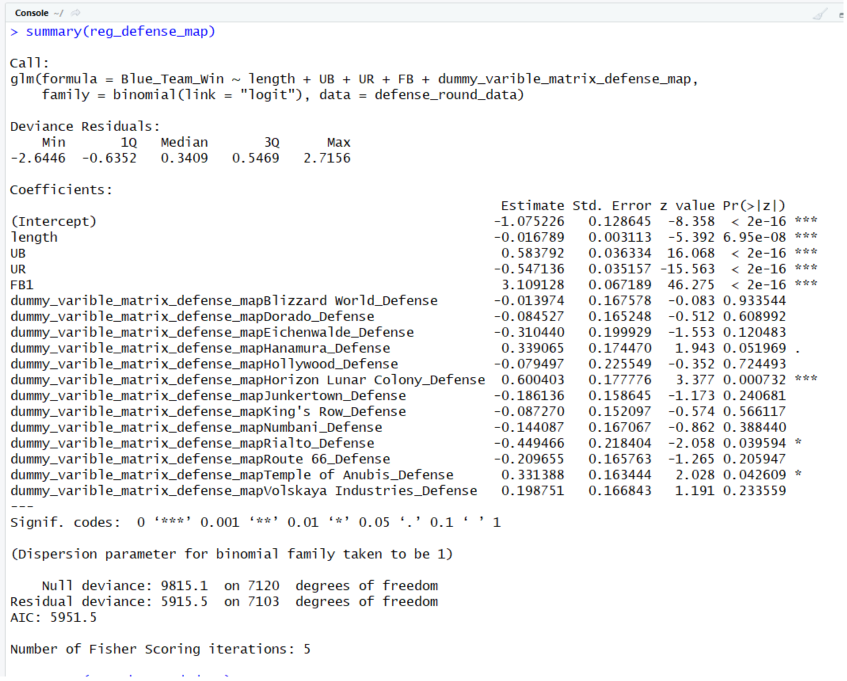
不同的資料分法

我認為前面的分法只針對地圖類型確實不夠好，因為對於進攻方以及防守方，"length"變數的影響方向是不同的，在進攻時，會戰時間愈長，愈容易獲勝，在防守時則相反，會戰時間拖得愈久，愈難獲勝，不過在控制模式由於雙方並沒有進攻方和防守方之分。會發生這種現象的原因在於遊戲機制，防守方在最後一個目標快被爭奪下來時，通常選擇會做一種策略叫做“續點“，就是讓其中一名選手踩在目標點附近，試圖拖住時間甚至找到反敗為勝的方式，但通常進攻方還是會取得勝利，不過不代表續點沒有意義，它具有戰略上的意義，在遊戲機制中，如果拖到延長賽，雖然輸掉會戰，但是另一方在這場遊戲最多就只能獲得平手，無法獲勝。因此分不同的地圖不如分成進攻方以及防守方，還有控制模式的地圖來分別做模型，這樣"length"變數才會有意義。

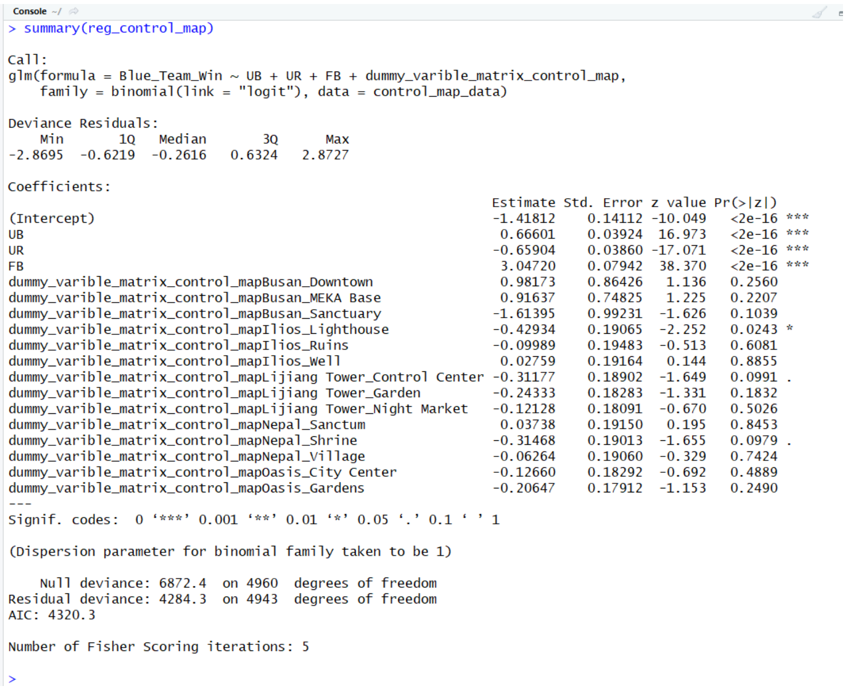
以下是分成進攻、防守、控制模式分別做出的模型



```
> summary(reg_attack_map)

Call:
glm(formula = Blue_Team_Win ~ length + UB + UR + FB + dummy_varible_matrix_attack_map,
    family = binomial(link = "logit"), data = attack_round_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0022  -0.5120  -0.3093   0.6178   2.8662

Coefficients:
                                                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                                                -1.962904   0.127100 -15.444  < 2e-16 ***
length                                                      0.026899   0.002951   9.115  < 2e-16 ***
UB                                                          0.604451   0.034126  17.712  < 2e-16 ***
UR                                                         -0.684302   0.034357 -19.917  < 2e-16 ***
FB1                                                         3.218535   0.065033  49.491  < 2e-16 ***
dummy_varible_matrix_attack_mapBlizzard World_Attack       -0.143123   0.162363  -0.882  0.37805
dummy_varible_matrix_attack_mapDorado_Attack               -0.214248   0.159396  -1.344  0.17891
dummy_varible_matrix_attack_mapEichenwalde_Attack          -0.008880   0.189021  -0.047  0.96253
dummy_varible_matrix_attack_mapHanamura_Attack             -0.586994   0.166709  -3.521  0.00043 ***
dummy_varible_matrix_attack_mapHollywood_Attack            -0.072392   0.200050  -0.362  0.71745
dummy_varible_matrix_attack_mapHorizon Lunar Colony_Attack -0.718778   0.170634  -4.212 2.53e-05 ***
dummy_varible_matrix_attack_mapJunkertown_Attack           -0.270568   0.155068  -1.745  0.08101 .
dummy_varible_matrix_attack_mapKing's Row_Attack           -0.021249   0.147635  -0.144  0.88556
dummy_varible_matrix_attack_mapNumbani_Attack              -0.191901   0.163256  -1.175  0.23981
dummy_varible_matrix_attack_mapRialto_Attack               -0.527576   0.205129  -2.572  0.01011 *
dummy_varible_matrix_attack_mapRoute 66_Attack              0.071785   0.161830   0.444  0.65735
dummy_varible_matrix_attack_mapTemple of Anubis_Attack     -0.768393   0.157171  -4.889 1.01e-06 ***
dummy_varible_matrix_attack_mapVolskaya Industries_Attack  -0.528273   0.164282  -3.216  0.00130 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 11312.5  on 8222  degrees of freedom
Residual deviance:  6566.3  on 8205  degrees of freedom
AIC: 6602.3

Number of Fisher Scoring iterations: 5

> 
```



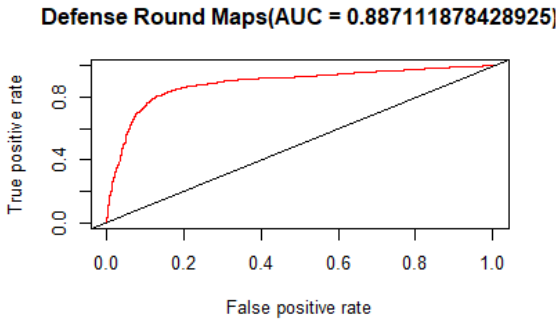
```
> summary(reg_defense_map)

Call:
glm(formula = Blue_Team_Win ~ length + UB + UR + FB + dummy_varible_matrix_defense_map,
    family = binomial(link = "logit"), data = defense_round_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6446  -0.6352   0.3409   0.5469   2.7156

Coefficients:
                                                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                                                  -1.075226   0.128645  -8.358  < 2e-16 ***
length                                                       -0.016789   0.003113  -5.392 6.95e-08 ***
UB                                                            0.583792   0.036334  16.068  < 2e-16 ***
UR                                                           -0.547136   0.035157 -15.563  < 2e-16 ***
FB1                                                           3.109128   0.067189  46.275  < 2e-16 ***
dummy_varible_matrix_defense_mapBlizzard World_Defense       -0.013974   0.167578  -0.083 0.933544
dummy_varible_matrix_defense_mapDorado_Defense               -0.084527   0.165248  -0.512 0.608992
dummy_varible_matrix_defense_mapEichenwalde_Defense          -0.310440   0.199929  -1.553 0.120483
dummy_varible_matrix_defense_mapHanamura_Defense              0.339065   0.174470   1.943 0.051969 .
dummy_varible_matrix_defense_mapHollywood_Defense            -0.079497   0.225549  -0.352 0.724493
dummy_varible_matrix_defense_mapHorizon Lunar Colony_Defense  0.600403   0.177776   3.377 0.000732 ***
dummy_varible_matrix_defense_mapJunkertown_Defense           -0.186136   0.158645  -1.173 0.240681
dummy_varible_matrix_defense_mapKing's Row_Defense           -0.087270   0.152097  -0.574 0.566117
dummy_varible_matrix_defense_mapNumbani_Defense              -0.144087   0.167067  -0.862 0.388440
dummy_varible_matrix_defense_mapRialto_Defense               -0.449466   0.218404  -2.058 0.039594 *
dummy_varible_matrix_defense_mapRoute 66_Defense             -0.209655   0.165763  -1.265 0.205947
dummy_varible_matrix_defense_mapTemple of Anubis_Defense      0.331388   0.163444   2.028 0.042609 *
dummy_varible_matrix_defense_mapVolskaya Industries_Defense   0.198751   0.166843   1.191 0.233559
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 9815.1  on 7120  degrees of freedom
Residual deviance: 5915.5  on 7103  degrees of freedom
AIC: 5951.5

Number of Fisher Scoring iterations: 5
```

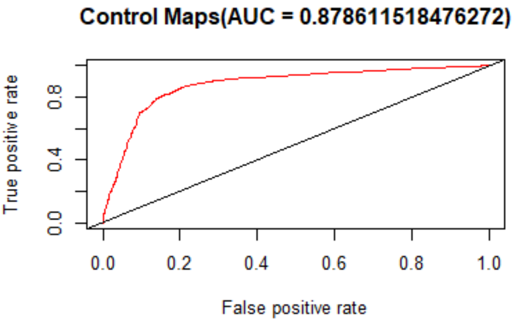


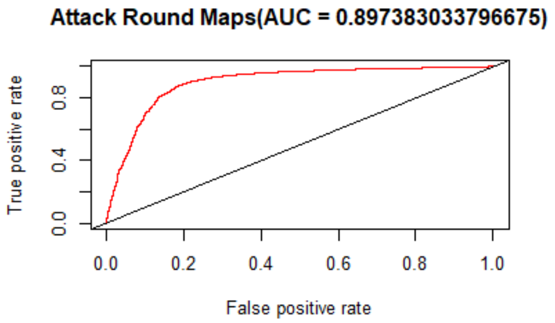
```
> summary(reg_control_map)

Call:
glm(formula = Blue_Team_Win ~ UB + UR + FB + dummy_varible_matrix_control_map,
    family = binomial(link = "logit"), data = control_map_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8695  -0.6219  -0.2616   0.6324   2.8727

Coefficients:
                                                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                                                  -1.41812    0.14112 -10.049   <2e-16 ***
UB                                                            0.66601    0.03924  16.973   <2e-16 ***
UR                                                           -0.65904    0.03860 -17.071   <2e-16 ***
FB                                                            3.04720    0.07942  38.370   <2e-16 ***
dummy_varible_matrix_control_mapBusan_Downtown                0.98173    0.86426   1.136   0.2560
dummy_varible_matrix_control_mapBusan_MEKA Base               0.91637    0.74825   1.225   0.2207
dummy_varible_matrix_control_mapBusan_Sanctuary              -1.61395    0.99231  -1.626   0.1039
dummy_varible_matrix_control_mapIlios_Lighthouse             -0.42934    0.19065  -2.252   0.0243 *
dummy_varible_matrix_control_mapIlios_Ruins                  -0.09989    0.19483  -0.513   0.6081
dummy_varible_matrix_control_mapIlios_Well                    0.02759    0.19164   0.144   0.8855
dummy_varible_matrix_control_mapLijiang Tower_Control Center -0.31177    0.18902  -1.649   0.0991 .
dummy_varible_matrix_control_mapLijiang Tower_Garden         -0.24333    0.18283  -1.331   0.1832
dummy_varible_matrix_control_mapLijiang Tower_Night Market   -0.12128    0.18091  -0.670   0.5026
dummy_varible_matrix_control_mapNepal_Sanctum                 0.03738    0.19150   0.195   0.8453
dummy_varible_matrix_control_mapNepal_Shrine                 -0.31468    0.19013  -1.655   0.0979 .
dummy_varible_matrix_control_mapNepal_Village                -0.06264    0.19060  -0.329   0.7424
dummy_varible_matrix_control_mapOasis_City Center            -0.12660    0.18292  -0.692   0.4889
dummy_varible_matrix_control_mapOasis_Gardens                -0.20647    0.17912  -1.153   0.2490
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6872.4  on 4960  degrees of freedom
Residual deviance: 4284.3  on 4943  degrees of freedom
AIC: 4320.3

Number of Fisher Scoring iterations: 5

> 
```

從上面的報表可以看出進攻方在 length 係數是正的，在防守方式負的，證實前面講到的續點的現象，而三個模型中最重要的變數還是首殺。另外，還可以觀察到佔領模式的地圖：Hanamura, Temple of Aunbis, Volskaya Industry, Horizon Lunar Colony, 四張地圖中的係數都是正的，也代表由於防守方到最後一個目標地點的距離比進攻方近，防守方比較容易獲勝，因此進攻方必須展開多次的會戰才有機會攻下目標。我認為這樣的模型解釋能力會比較好，但是由於把原本的資料分成三個來做模型，數據數量遠比原先的資料少，如果能再收集更多的數據會更好。

三種模型的 ROC Curve







從上面三個 ROC Curve 中可以看出 3 個 AUC 並沒有跟最前面的模型差很多，而且在 Confusion Matrix 上，在正確率表現也沒有差太多，我認為原因在於把原本的資料分成三個來做模型，訓練集遠比原先的資料少，預測做出來的正確率並不會很高，不過如果資料筆數可以到達跟原始資料的 20000 筆，應該也能做出預測正確率更高的模型。以下是三種模型的 Confusion Matrix。

Attack Round Type(Accuracy = 0.822)

實際狀況 預測	贏	輸
贏	205	44
輸	45	206

Defense Round Type(Accuracy = 0.818)

實際狀況 預測	贏	輸
贏	195	56
輸	35	214

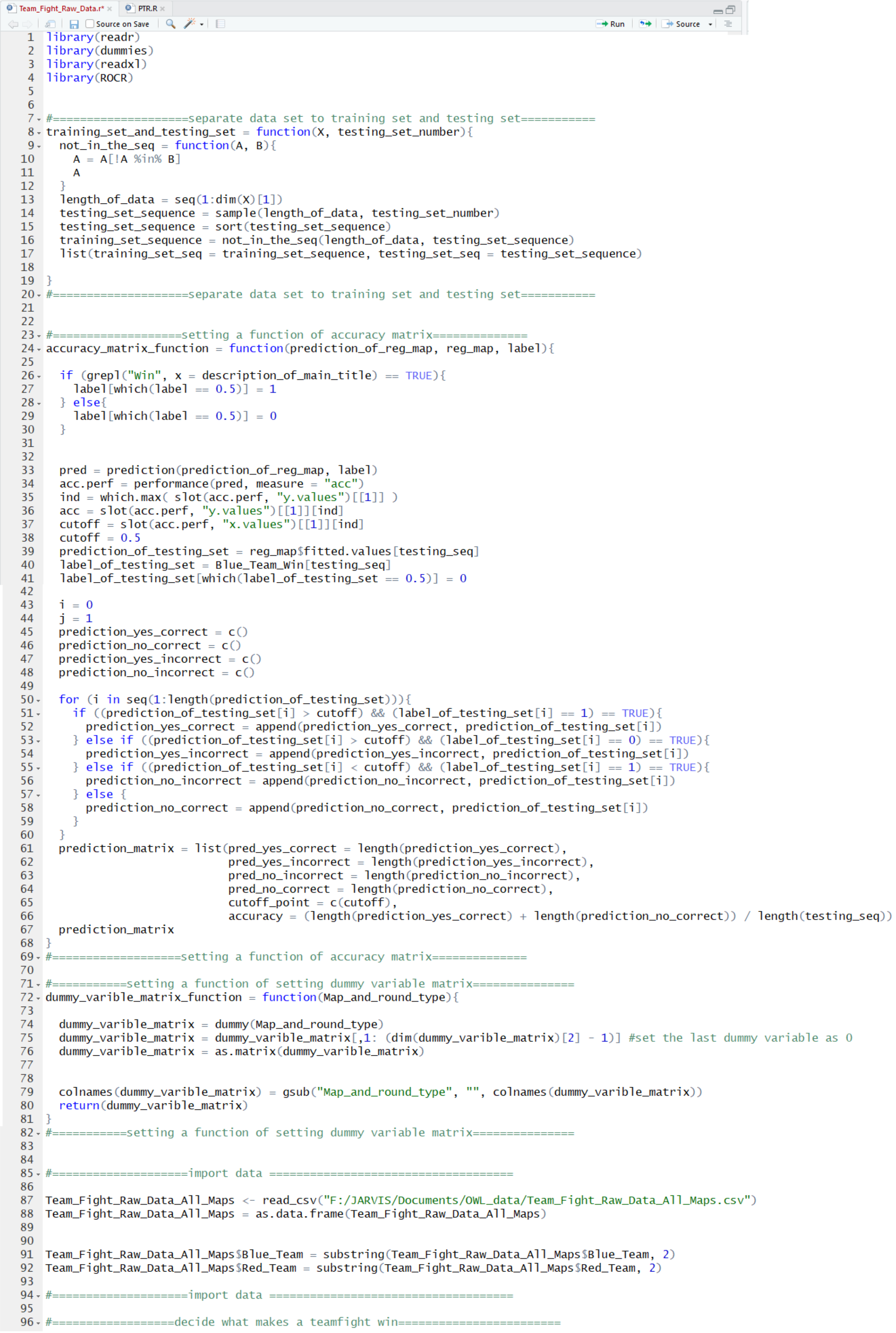
Control Maps(Accuracy = 0.82)

實際狀況 預測	贏	輸
贏	215	48
輸	42	195

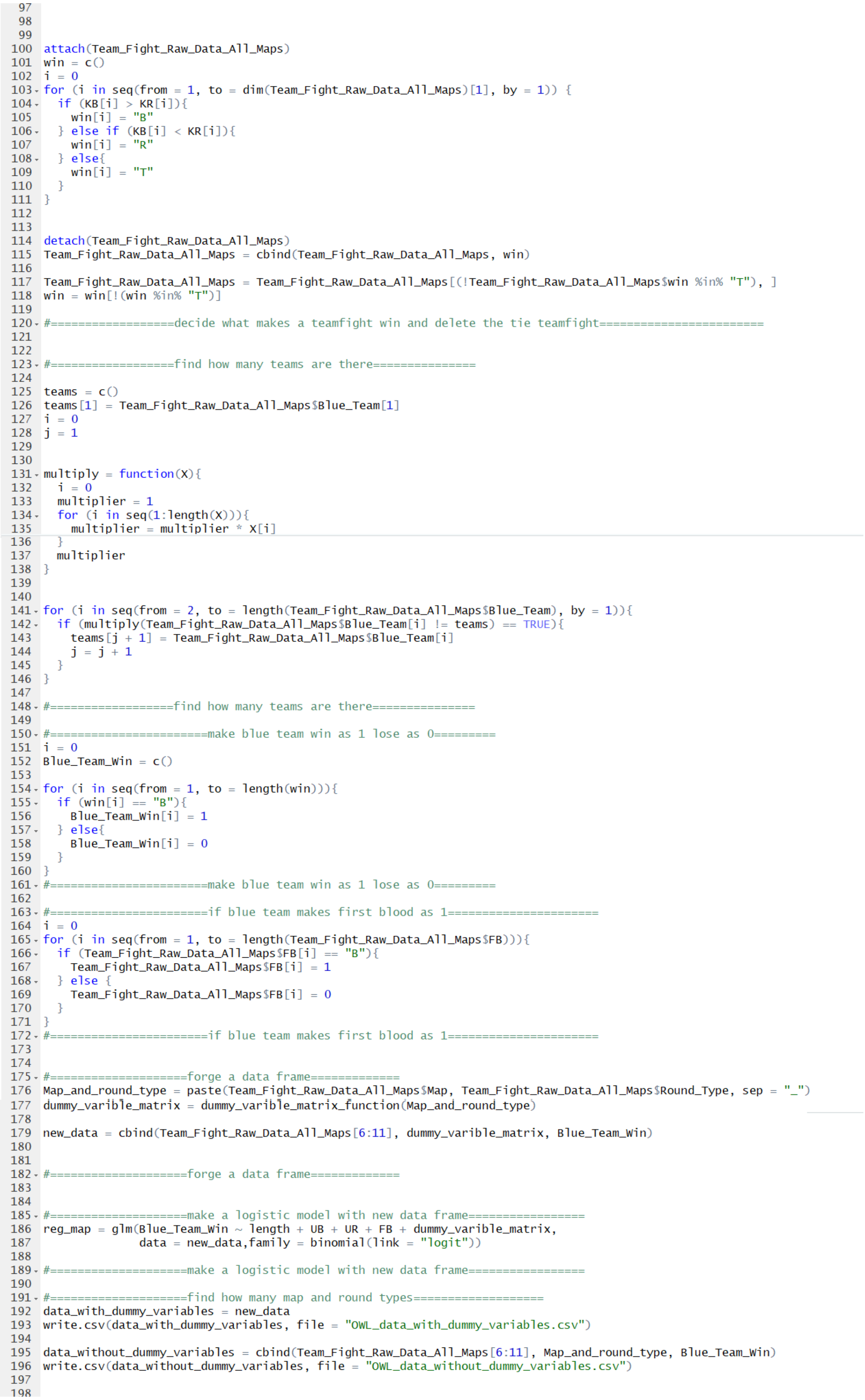
結論

1. 事實上影響會戰勝負的關鍵在於是否拿到首殺，係數都超過 3，若是在比賽中拿下首殺，odds 會是沒拿下首殺的 8.15 倍
2. 在實務上，我們可以藉由是否拿下首殺直接作為是不是能贏下會戰的判斷
3. 有沒有對不同的模式分開來做模型，預測的狀況都不會有明顯的差異
4. 對於分成進攻方和防守方，變數比較具有解釋力
5. 如果暴雪娛樂能在提供更多數據提供我們使用，例如使用多少大補血包和小補血包、總輸出量、總補血量、復活時間、到會戰點的距離等等，能讓模型更加準確

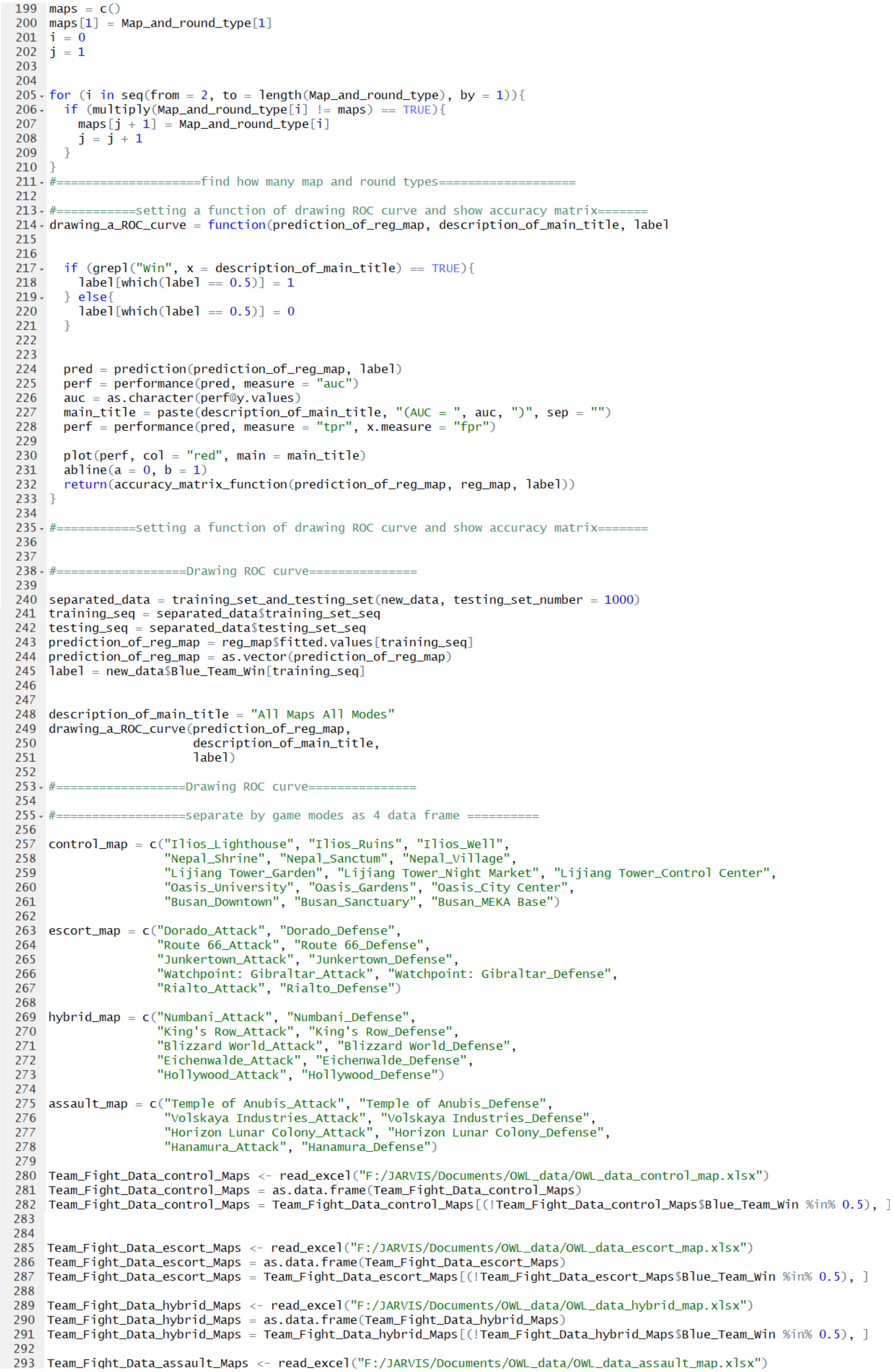
程式碼



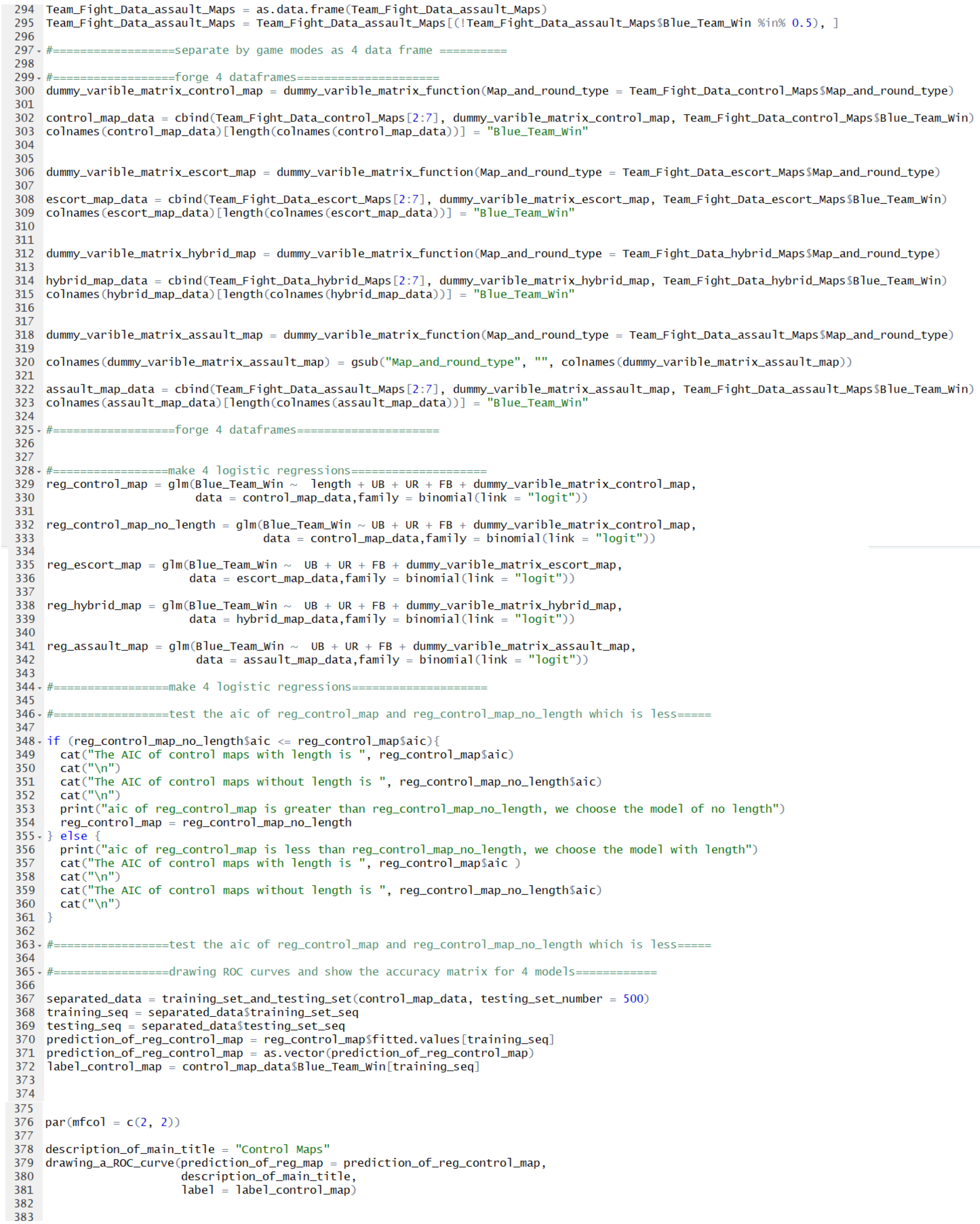
```r
library(readr)
library(dummies)
library(readxl)
library(ROCR)


#====================separate data set to training set and testing set===========
training_set_and_testing_set = function(X, testing_set_number){
  not_in_the_seq = function(A, B){
    A = A[!A %in% B]
    A
  }
  length_of_data = seq(1:dim(X)[1])
  testing_set_sequence = sample(length_of_data, testing_set_number)
  testing_set_sequence = sort(testing_set_sequence)
  training_set_sequence = not_in_the_seq(length_of_data, testing_set_sequence)
  list(training_set_seq = training_set_sequence, testing_set_seq = testing_set_sequence)

}
#====================separate data set to training set and testing set===========


#===================setting a function of accuracy matrix==============
accuracy_matrix_function = function(prediction_of_reg_map, reg_map, label){

  if (grepl("Win", x = description_of_main_title) == TRUE){
    label[which(label == 0.5)] = 1
  } else{
    label[which(label == 0.5)] = 0
  }


  pred = prediction(prediction_of_reg_map, label)
  acc.perf = performance(pred, measure = "acc")
  ind = which.max( slot(acc.perf, "y.values")[[1]] )
  acc = slot(acc.perf, "y.values")[[1]][ind]
  cutoff = slot(acc.perf, "x.values")[[1]][ind]
  cutoff = 0.5
  prediction_of_testing_set = reg_map$fitted.values[testing_seq]
  label_of_testing_set = Blue_Team_Win[testing_seq]
  label_of_testing_set[which(label_of_testing_set == 0.5)] = 0

  i = 0
  j = 1
  prediction_yes_correct = c()
  prediction_no_correct = c()
  prediction_yes_incorrect = c()
  prediction_no_incorrect = c()

  for (i in seq(1:length(prediction_of_testing_set))){
    if ((prediction_of_testing_set[i] > cutoff) && (label_of_testing_set[i] == 1) == TRUE){
      prediction_yes_correct = append(prediction_yes_correct, prediction_of_testing_set[i])
    } else if ((prediction_of_testing_set[i] > cutoff) && (label_of_testing_set[i] == 0) == TRUE){
      prediction_yes_incorrect = append(prediction_yes_incorrect, prediction_of_testing_set[i])
    } else if ((prediction_of_testing_set[i] < cutoff) && (label_of_testing_set[i] == 1) == TRUE){
      prediction_no_incorrect = append(prediction_no_incorrect, prediction_of_testing_set[i])
    } else {
      prediction_no_correct = append(prediction_no_correct, prediction_of_testing_set[i])
    }
  }
  prediction_matrix = list(pred_yes_correct = length(prediction_yes_correct),
                           pred_yes_incorrect = length(prediction_yes_incorrect),
                           pred_no_incorrect = length(prediction_no_incorrect),
                           pred_no_correct = length(prediction_no_correct),
                           cutoff_point = c(cutoff),
                           accuracy = (length(prediction_yes_correct) + length(prediction_no_correct)) / length(testing_seq))
  prediction_matrix
}
#===================setting a function of accuracy matrix==============

#===========setting a function of setting dummy variable matrix===============
dummy_varible_matrix_function = function(Map_and_round_type){

  dummy_varible_matrix = dummy(Map_and_round_type)
  dummy_varible_matrix = dummy_varible_matrix[,1: (dim(dummy_varible_matrix)[2] - 1)] #set the last dummy variable as 0
  dummy_varible_matrix = as.matrix(dummy_varible_matrix)


  colnames(dummy_varible_matrix) = gsub("Map_and_round_type", "", colnames(dummy_varible_matrix))
  return(dummy_varible_matrix)
}
#===========setting a function of setting dummy variable matrix===============


#====================import data ====================================

Team_Fight_Raw_Data_All_Maps <- read_csv("F:/JARVIS/Documents/OWL_data/Team_Fight_Raw_Data_All_Maps.csv")
Team_Fight_Raw_Data_All_Maps = as.data.frame(Team_Fight_Raw_Data_All_Maps)


Team_Fight_Raw_Data_All_Maps$Blue_Team = substring(Team_Fight_Raw_Data_All_Maps$Blue_Team, 2)
Team_Fight_Raw_Data_All_Maps$Red_Team = substring(Team_Fight_Raw_Data_All_Maps$Red_Team, 2)

#====================import data ====================================

#==================decide what makes a teamfight win========================
```



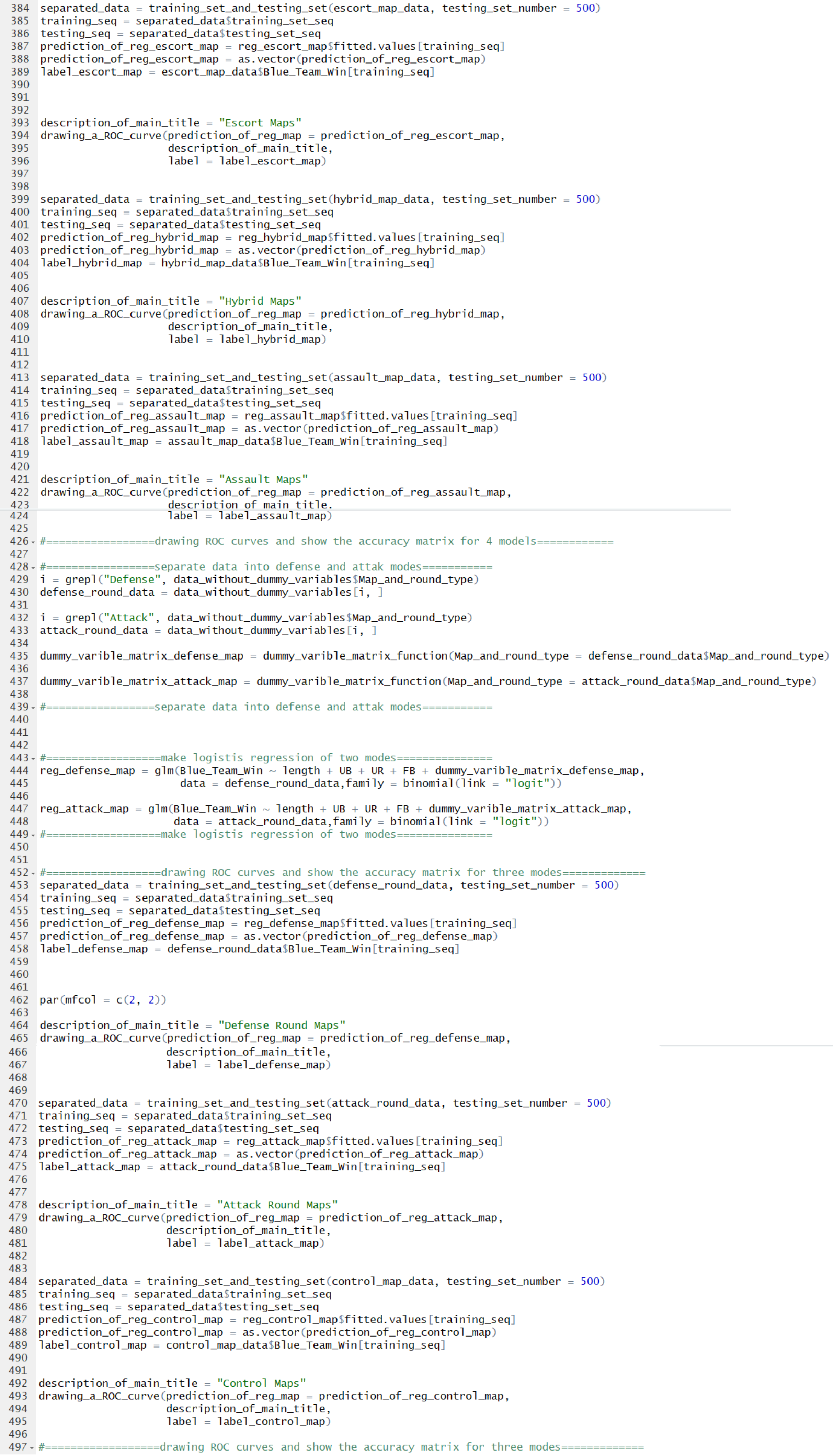
```r



attach(Team_Fight_Raw_Data_All_Maps)
win = c()
i = 0
for (i in seq(from = 1, to = dim(Team_Fight_Raw_Data_All_Maps)[1], by = 1)) {
  if (KB[i] > KR[i]){
    win[i] = "B"
  } else if (KB[i] < KR[i]){
    win[i] = "R"
  } else{
    win[i] = "T"
  }
}


detach(Team_Fight_Raw_Data_All_Maps)
Team_Fight_Raw_Data_All_Maps = cbind(Team_Fight_Raw_Data_All_Maps, win)

Team_Fight_Raw_Data_All_Maps = Team_Fight_Raw_Data_All_Maps[(!Team_Fight_Raw_Data_All_Maps$win %in% "T"), ]
win = win[!(win %in% "T")]

#==================decide what makes a teamfight win and delete the tie teamfight========================


#==================find how many teams are there===============

teams = c()
teams[1] = Team_Fight_Raw_Data_All_Maps$Blue_Team[1]
i = 0
j = 1


multiply = function(X){
  i = 0
  multiplier = 1
  for (i in seq(1:length(X))){
    multiplier = multiplier * X[i]
  }
  multiplier
}


for (i in seq(from = 2, to = length(Team_Fight_Raw_Data_All_Maps$Blue_Team), by = 1)){
  if (multiply(Team_Fight_Raw_Data_All_Maps$Blue_Team[i] != teams) == TRUE){
    teams[j + 1] = Team_Fight_Raw_Data_All_Maps$Blue_Team[i]
    j = j + 1
  }
}

#==================find how many teams are there===============

#=======================make blue team win as 1 lose as 0=========
i = 0
Blue_Team_Win = c()

for (i in seq(from = 1, to = length(win))){
  if (win[i] == "B"){
    Blue_Team_Win[i] = 1
  } else{
    Blue_Team_Win[i] = 0
  }
}
#=======================make blue team win as 1 lose as 0=========

#=======================if blue team makes first blood as 1=====================
i = 0
for (i in seq(from = 1, to = length(Team_Fight_Raw_Data_All_Maps$FB))){
  if (Team_Fight_Raw_Data_All_Maps$FB[i] == "B"){
    Team_Fight_Raw_Data_All_Maps$FB[i] = 1
  } else {
    Team_Fight_Raw_Data_All_Maps$FB[i] = 0
  }
}
#=======================if blue team makes first blood as 1=====================


#====================forge a data frame============
Map_and_round_type = paste(Team_Fight_Raw_Data_All_Maps$Map, Team_Fight_Raw_Data_All_Maps$Round_Type, sep = "_")
dummy_varible_matrix = dummy_varible_matrix_function(Map_and_round_type)

new_data = cbind(Team_Fight_Raw_Data_All_Maps[6:11], dummy_varible_matrix, Blue_Team_Win)


#====================forge a data frame============


#====================make a logistic model with new data frame================
reg_map = glm(Blue_Team_Win ~ length + UB + UR + FB + dummy_varible_matrix,
              data = new_data,family = binomial(link = "logit"))

#====================make a logistic model with new data frame================

#====================find how many map and round types==================
data_with_dummy_variables = new_data
write.csv(data_with_dummy_variables, file = "OWL_data_with_dummy_variables.csv")

data_without_dummy_variables = cbind(Team_Fight_Raw_Data_All_Maps[6:11], Map_and_round_type, Blue_Team_Win)
write.csv(data_without_dummy_variables, file = "OWL_data_without_dummy_variables.csv")

```



```r
maps = c()
maps[1] = Map_and_round_type[1]
i = 0
j = 1


for (i in seq(from = 2, to = length(Map_and_round_type), by = 1)){
  if (multiply(Map_and_round_type[i] != maps) == TRUE){
    maps[j + 1] = Map_and_round_type[i]
    j = j + 1
  }
}
#===================find how many map and round types===================

#===========setting a function of drawing ROC curve and show accuracy matrix=======
drawing_a_ROC_curve = function(prediction_of_reg_map, description_of_main_title, label


  if (grepl("Win", x = description_of_main_title) == TRUE){
    label[which(label == 0.5)] = 1
  } else{
    label[which(label == 0.5)] = 0
  }


  pred = prediction(prediction_of_reg_map, label)
  perf = performance(pred, measure = "auc")
  auc = as.character(perf@y.values)
  main_title = paste(description_of_main_title, "(AUC = ", auc, ")", sep = "")
  perf = performance(pred, measure = "tpr", x.measure = "fpr")

  plot(perf, col = "red", main = main_title)
  abline(a = 0, b = 1)
  return(accuracy_matrix_function(prediction_of_reg_map, reg_map, label))
}

#===========setting a function of drawing ROC curve and show accuracy matrix=======


#==================Drawing ROC curve===============

separated_data = training_set_and_testing_set(new_data, testing_set_number = 1000)
training_seq = separated_data$training_set_seq
testing_seq = separated_data$testing_set_seq
prediction_of_reg_map = reg_map$fitted.values[training_seq]
prediction_of_reg_map = as.vector(prediction_of_reg_map)
label = new_data$Blue_Team_Win[training_seq]


description_of_main_title = "All Maps All Modes"
drawing_a_ROC_curve(prediction_of_reg_map,
                    description_of_main_title,
                    label)

#==================Drawing ROC curve===============

#==================separate by game modes as 4 data frame ==========

control_map = c("Ilios_Lighthouse", "Ilios_Ruins", "Ilios_Well",
                "Nepal_Shrine", "Nepal_Sanctum", "Nepal_Village",
                "Lijiang Tower_Garden", "Lijiang Tower_Night Market", "Lijiang Tower_Control Center",
                "Oasis_University", "Oasis_Gardens", "Oasis_City Center",
                "Busan_Downtown", "Busan_Sanctuary", "Busan_MEKA Base")

escort_map = c("Dorado_Attack", "Dorado_Defense",
               "Route 66_Attack", "Route 66_Defense",
               "Junkertown_Attack", "Junkertown_Defense",
               "Watchpoint: Gibraltar_Attack", "Watchpoint: Gibraltar_Defense",
               "Rialto_Attack", "Rialto_Defense")

hybrid_map = c("Numbani_Attack", "Numbani_Defense",
               "King's Row_Attack", "King's Row_Defense",
               "Blizzard World_Attack", "Blizzard World_Defense",
               "Eichenwalde_Attack", "Eichenwalde_Defense",
               "Hollywood_Attack", "Hollywood_Defense")

assault_map = c("Temple of Anubis_Attack", "Temple of Anubis_Defense",
                "Volskaya Industries_Attack", "Volskaya Industries_Defense",
                "Horizon Lunar Colony_Attack", "Horizon Lunar Colony_Defense",
                "Hanamura_Attack", "Hanamura_Defense")

Team_Fight_Data_control_Maps <- read_excel("F:/JARVIS/Documents/OWL_data/OWL_data_control_map.xlsx")
Team_Fight_Data_control_Maps = as.data.frame(Team_Fight_Data_control_Maps)
Team_Fight_Data_control_Maps = Team_Fight_Data_control_Maps[(!Team_Fight_Data_control_Maps$Blue_Team_Win %in% 0.5), ]


Team_Fight_Data_escort_Maps <- read_excel("F:/JARVIS/Documents/OWL_data/OWL_data_escort_map.xlsx")
Team_Fight_Data_escort_Maps = as.data.frame(Team_Fight_Data_escort_Maps)
Team_Fight_Data_escort_Maps = Team_Fight_Data_escort_Maps[(!Team_Fight_Data_escort_Maps$Blue_Team_Win %in% 0.5), ]

Team_Fight_Data_hybrid_Maps <- read_excel("F:/JARVIS/Documents/OWL_data/OWL_data_hybrid_map.xlsx")
Team_Fight_Data_hybrid_Maps = as.data.frame(Team_Fight_Data_hybrid_Maps)
Team_Fight_Data_hybrid_Maps = Team_Fight_Data_hybrid_Maps[(!Team_Fight_Data_hybrid_Maps$Blue_Team_Win %in% 0.5), ]

Team_Fight_Data_assault_Maps <- read_excel("F:/JARVIS/Documents/OWL_data/OWL_data_assault_map.xlsx")
```



```
Team_Fight_Data_assault_Maps = as.data.frame(Team_Fight_Data_assault_Maps)
Team_Fight_Data_assault_Maps = Team_Fight_Data_assault_Maps[(!Team_Fight_Data_assault_Maps$Blue_Team_Win %in% 0.5), ]

#==================separate by game modes as 4 data frame ==========

#==================forge 4 dataframes=====================
dummy_varible_matrix_control_map = dummy_varible_matrix_function(Map_and_round_type = Team_Fight_Data_control_Maps$Map_and_round_type)

control_map_data = cbind(Team_Fight_Data_control_Maps[2:7], dummy_varible_matrix_control_map, Team_Fight_Data_control_Maps$Blue_Team_Win)
colnames(control_map_data)[length(colnames(control_map_data))] = "Blue_Team_Win"


dummy_varible_matrix_escort_map = dummy_varible_matrix_function(Map_and_round_type = Team_Fight_Data_escort_Maps$Map_and_round_type)

escort_map_data = cbind(Team_Fight_Data_escort_Maps[2:7], dummy_varible_matrix_escort_map, Team_Fight_Data_escort_Maps$Blue_Team_Win)
colnames(escort_map_data)[length(colnames(escort_map_data))] = "Blue_Team_Win"


dummy_varible_matrix_hybrid_map = dummy_varible_matrix_function(Map_and_round_type = Team_Fight_Data_hybrid_Maps$Map_and_round_type)

hybrid_map_data = cbind(Team_Fight_Data_hybrid_Maps[2:7], dummy_varible_matrix_hybrid_map, Team_Fight_Data_hybrid_Maps$Blue_Team_Win)
colnames(hybrid_map_data)[length(colnames(hybrid_map_data))] = "Blue_Team_Win"


dummy_varible_matrix_assault_map = dummy_varible_matrix_function(Map_and_round_type = Team_Fight_Data_assault_Maps$Map_and_round_type)

colnames(dummy_varible_matrix_assault_map) = gsub("Map_and_round_type", "", colnames(dummy_varible_matrix_assault_map))

assault_map_data = cbind(Team_Fight_Data_assault_Maps[2:7], dummy_varible_matrix_assault_map, Team_Fight_Data_assault_Maps$Blue_Team_Win)
colnames(assault_map_data)[length(colnames(assault_map_data))] = "Blue_Team_Win"

#==================forge 4 dataframes=====================


#=================make 4 logistic regressions====================
reg_control_map = glm(Blue_Team_Win ~  length + UB + UR + FB + dummy_varible_matrix_control_map,
                      data = control_map_data,family = binomial(link = "logit"))

reg_control_map_no_length = glm(Blue_Team_Win ~ UB + UR + FB + dummy_varible_matrix_control_map,
                                data = control_map_data,family = binomial(link = "logit"))

reg_escort_map = glm(Blue_Team_Win ~  UB + UR + FB + dummy_varible_matrix_escort_map,
                     data = escort_map_data,family = binomial(link = "logit"))

reg_hybrid_map = glm(Blue_Team_Win ~  UB + UR + FB + dummy_varible_matrix_hybrid_map,
                     data = hybrid_map_data,family = binomial(link = "logit"))

reg_assault_map = glm(Blue_Team_Win ~  UB + UR + FB + dummy_varible_matrix_assault_map,
                      data = assault_map_data,family = binomial(link = "logit"))

#=================make 4 logistic regressions====================

#=================test the aic of reg_control_map and reg_control_map_no_length which is less=====

if (reg_control_map_no_length$aic <= reg_control_map$aic){
  cat("The AIC of control maps with length is ", reg_control_map$aic)
  cat("\n")
  cat("The AIC of control maps without length is ", reg_control_map_no_length$aic)
  cat("\n")
  print("aic of reg_control_map is greater than reg_control_map_no_length, we choose the model of no length")
  reg_control_map = reg_control_map_no_length
} else {
  print("aic of reg_control_map is less than reg_control_map_no_length, we choose the model with length")
  cat("The AIC of control maps with length is ", reg_control_map$aic )
  cat("\n")
  cat("The AIC of control maps without length is ", reg_control_map_no_length$aic)
  cat("\n")
}

#=================test the aic of reg_control_map and reg_control_map_no_length which is less=====

#=================drawing ROC curves and show the accuracy matrix for 4 models============

separated_data = training_set_and_testing_set(control_map_data, testing_set_number = 500)
training_seq = separated_data$training_set_seq
testing_seq = separated_data$testing_set_seq
prediction_of_reg_control_map = reg_control_map$fitted.values[training_seq]
prediction_of_reg_control_map = as.vector(prediction_of_reg_control_map)
label_control_map = control_map_data$Blue_Team_Win[training_seq]



par(mfcol = c(2, 2))

description_of_main_title = "Control Maps"
drawing_a_ROC_curve(prediction_of_reg_map = prediction_of_reg_control_map,
                    description_of_main_title,
                    label = label_control_map)


```



```
separated_data = training_set_and_testing_set(escort_map_data, testing_set_number = 500)
training_seq = separated_data$training_set_seq
testing_seq = separated_data$testing_set_seq
prediction_of_reg_escort_map = reg_escort_map$fitted.values[training_seq]
prediction_of_reg_escort_map = as.vector(prediction_of_reg_escort_map)
label_escort_map = escort_map_data$Blue_Team_Win[training_seq]



description_of_main_title = "Escort Maps"
drawing_a_ROC_curve(prediction_of_reg_map = prediction_of_reg_escort_map,
                    description_of_main_title,
                    label = label_escort_map)


separated_data = training_set_and_testing_set(hybrid_map_data, testing_set_number = 500)
training_seq = separated_data$training_set_seq
testing_seq = separated_data$testing_set_seq
prediction_of_reg_hybrid_map = reg_hybrid_map$fitted.values[training_seq]
prediction_of_reg_hybrid_map = as.vector(prediction_of_reg_hybrid_map)
label_hybrid_map = hybrid_map_data$Blue_Team_Win[training_seq]


description_of_main_title = "Hybrid Maps"
drawing_a_ROC_curve(prediction_of_reg_map = prediction_of_reg_hybrid_map,
                    description_of_main_title,
                    label = label_hybrid_map)


separated_data = training_set_and_testing_set(assault_map_data, testing_set_number = 500)
training_seq = separated_data$training_set_seq
testing_seq = separated_data$testing_set_seq
prediction_of_reg_assault_map = reg_assault_map$fitted.values[training_seq]
prediction_of_reg_assault_map = as.vector(prediction_of_reg_assault_map)
label_assault_map = assault_map_data$Blue_Team_Win[training_seq]


description_of_main_title = "Assault Maps"
drawing_a_ROC_curve(prediction_of_reg_map = prediction_of_reg_assault_map,
                    description_of_main_title,
                    label = label_assault_map)

#=================drawing ROC curves and show the accuracy matrix for 4 models============

#=================separate data into defense and attak modes==========
i = grepl("Defense", data_without_dummy_variables$Map_and_round_type)
defense_round_data = data_without_dummy_variables[i, ]

i = grepl("Attack", data_without_dummy_variables$Map_and_round_type)
attack_round_data = data_without_dummy_variables[i, ]

dummy_varible_matrix_defense_map = dummy_varible_matrix_function(Map_and_round_type = defense_round_data$Map_and_round_type)

dummy_varible_matrix_attack_map = dummy_varible_matrix_function(Map_and_round_type = attack_round_data$Map_and_round_type)

#=================separate data into defense and attak modes==========



#==================make logistis regression of two modes==============
reg_defense_map = glm(Blue_Team_Win ~ length + UB + UR + FB + dummy_varible_matrix_defense_map,
                      data = defense_round_data,family = binomial(link = "logit"))

reg_attack_map = glm(Blue_Team_Win ~ length + UB + UR + FB + dummy_varible_matrix_attack_map,
                     data = attack_round_data,family = binomial(link = "logit"))
#==================make logistis regression of two modes==============


#==================drawing ROC curves and show the accuracy matrix for three modes============
separated_data = training_set_and_testing_set(defense_round_data, testing_set_number = 500)
training_seq = separated_data$training_set_seq
testing_seq = separated_data$testing_set_seq
prediction_of_reg_defense_map = reg_defense_map$fitted.values[training_seq]
prediction_of_reg_defense_map = as.vector(prediction_of_reg_defense_map)
label_defense_map = defense_round_data$Blue_Team_Win[training_seq]



par(mfcol = c(2, 2))

description_of_main_title = "Defense Round Maps"
drawing_a_ROC_curve(prediction_of_reg_map = prediction_of_reg_defense_map,
                    description_of_main_title,
                    label = label_defense_map)


separated_data = training_set_and_testing_set(attack_round_data, testing_set_number = 500)
training_seq = separated_data$training_set_seq
testing_seq = separated_data$testing_set_seq
prediction_of_reg_attack_map = reg_attack_map$fitted.values[training_seq]
prediction_of_reg_attack_map = as.vector(prediction_of_reg_attack_map)
label_attack_map = attack_round_data$Blue_Team_Win[training_seq]


description_of_main_title = "Attack Round Maps"
drawing_a_ROC_curve(prediction_of_reg_map = prediction_of_reg_attack_map,
                    description_of_main_title,
                    label = label_attack_map)


separated_data = training_set_and_testing_set(control_map_data, testing_set_number = 500)
training_seq = separated_data$training_set_seq
testing_seq = separated_data$testing_set_seq
prediction_of_reg_control_map = reg_control_map$fitted.values[training_seq]
prediction_of_reg_control_map = as.vector(prediction_of_reg_control_map)
label_control_map = control_map_data$Blue_Team_Win[training_seq]


description_of_main_title = "Control Maps"
drawing_a_ROC_curve(prediction_of_reg_map = prediction_of_reg_control_map,
                    description_of_main_title,
                    label = label_control_map)

#==================drawing ROC curves and show the accuracy matrix for three modes============
```