1830004016
1830005022
1830021031
1830006195
1830006230

Prediction of the result of Pokémon one-on-one battles

Kaiyang LIU
Zinan LU
Geng
Yunzhi XIAO
Jiaying YIN

CONTENT

- Pokémon Overview
- Model Building
- Model Interpretation
- Discussion

Pokémon Overview

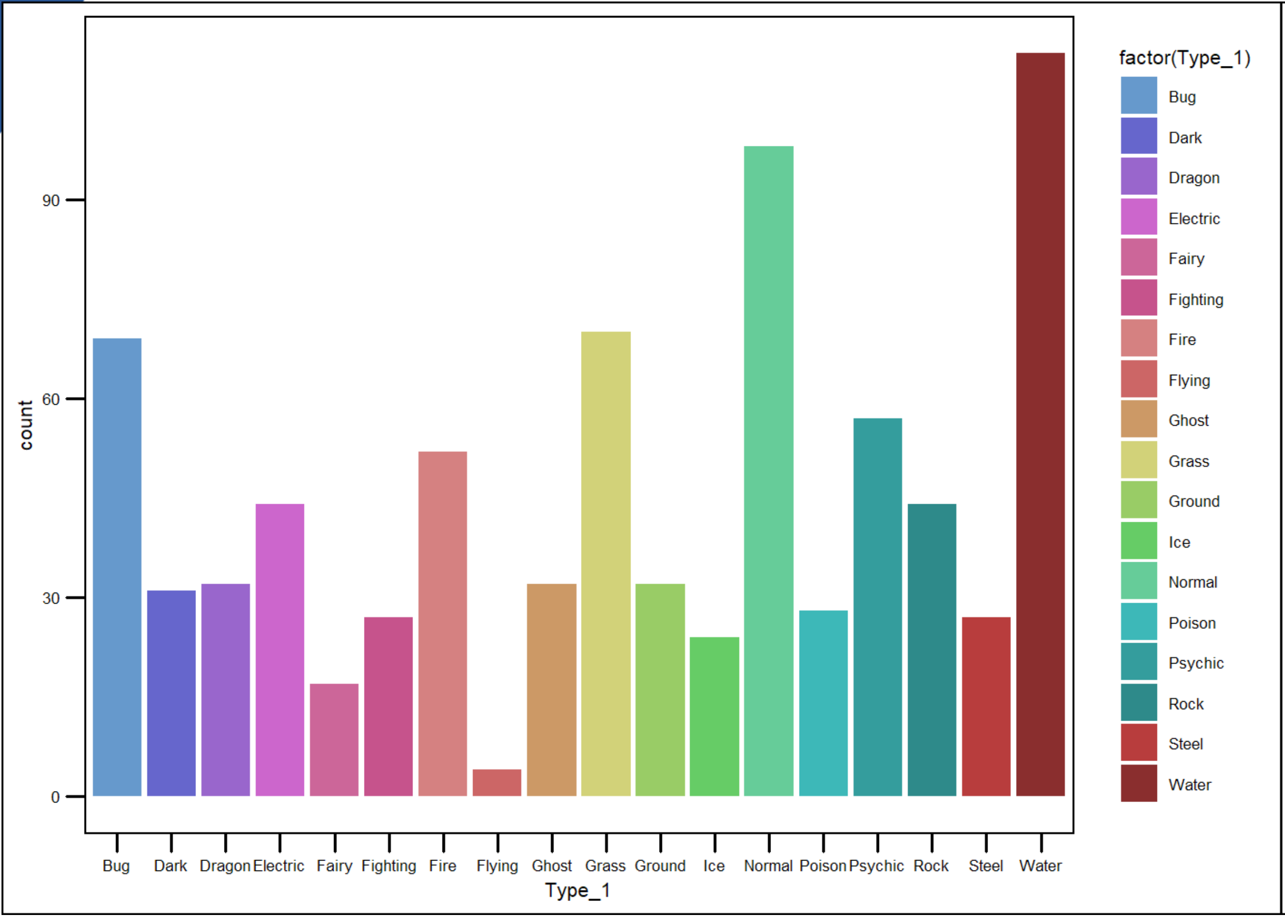
Variables

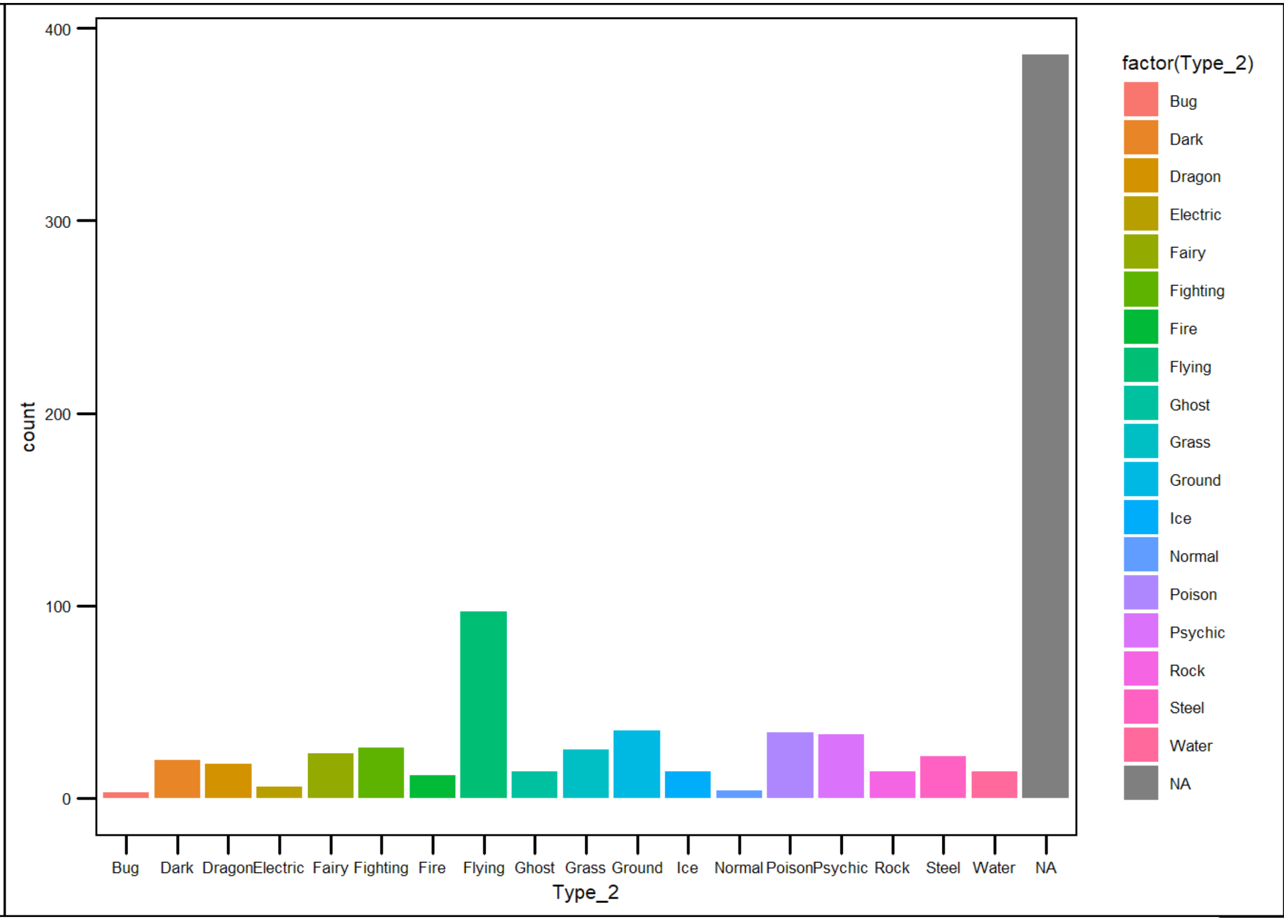
①	Name	character
②	Type 1	character
③	Type 2	character
④	HP	numeric
⑤	Attack	numeric
⑥	Defense	numeric
⑦	Speed	numeric
⑧	Sp.Atk	numeric
⑨	Sp.Def	numeric
⑩	Generation	numeric
⑪	Legendary	logic

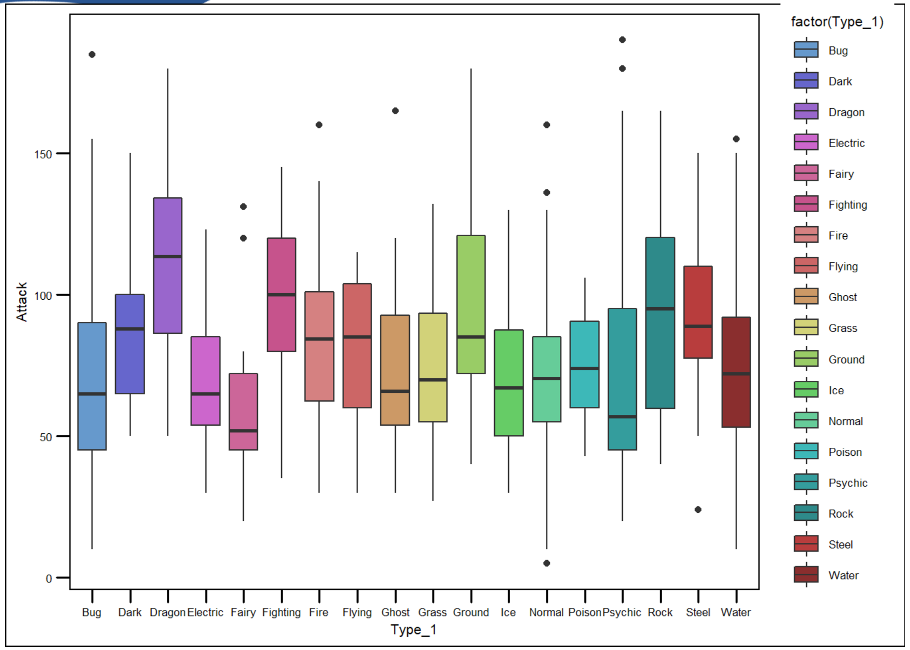
Variables

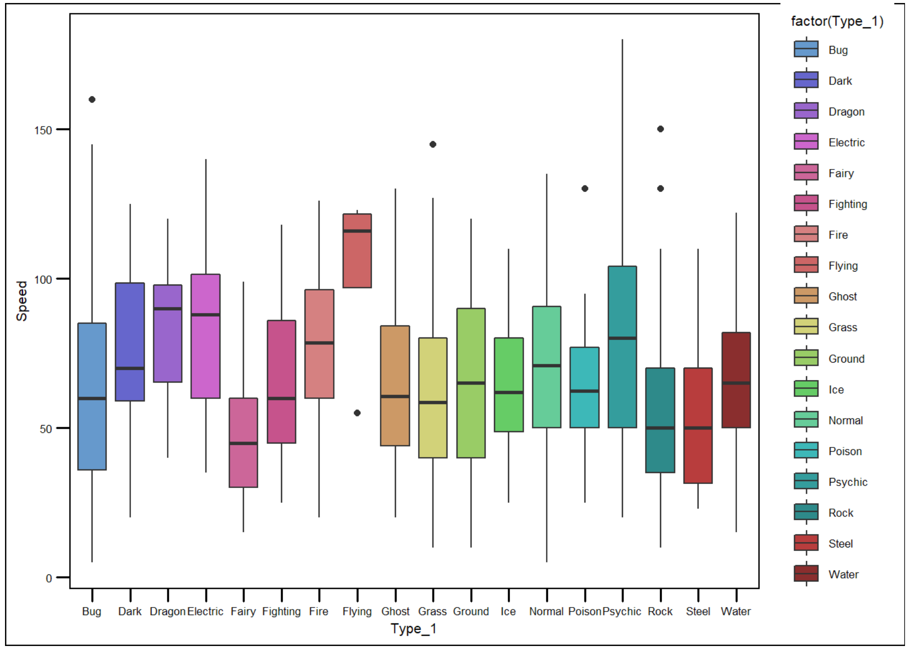
1. **Name:** The English name of the Pokémon
2. **Type 1:** Primary type, related to the nature (18 possible types)
3. **Type 2:** A Pokémon can have secondary type but not necessary
4. **HP:** Base health points of the Pokémon
5. **Attack:** Base attack of the Pokémon
6. **Defense:** Base defense of the Pokémon
7. **Speed:** Base speed of the Pokémon
8. **Sp.Atk:** Base special attack of the Pokémon
9. **Sp.Def:** Base special defense of the Pokémon
10. ~~**Generation:** The generation when the Pokémon was released~~
11. **Legendary:** Boolean indicating whether the Pokémon is legendary or not

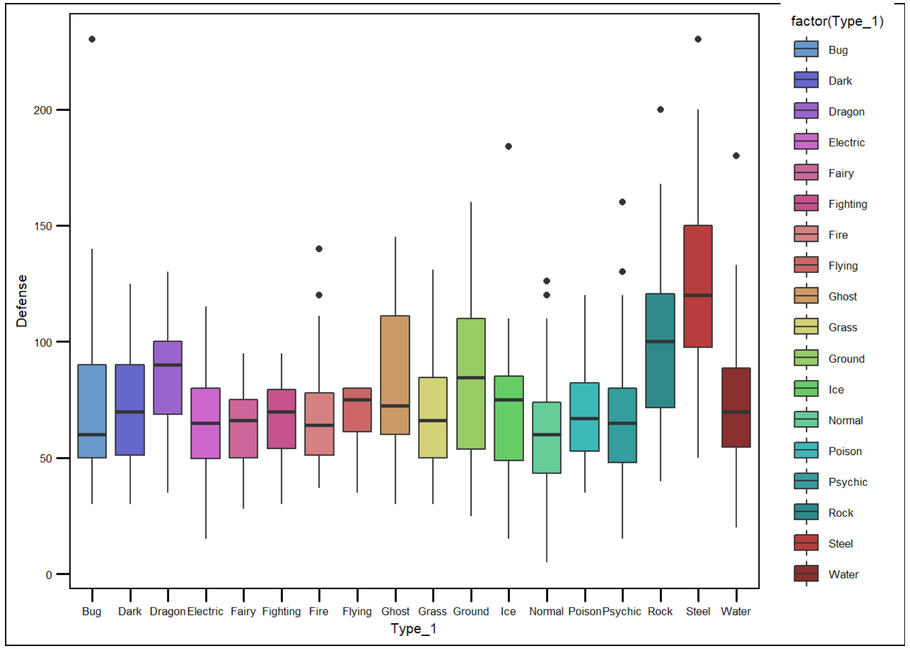
Data visualization

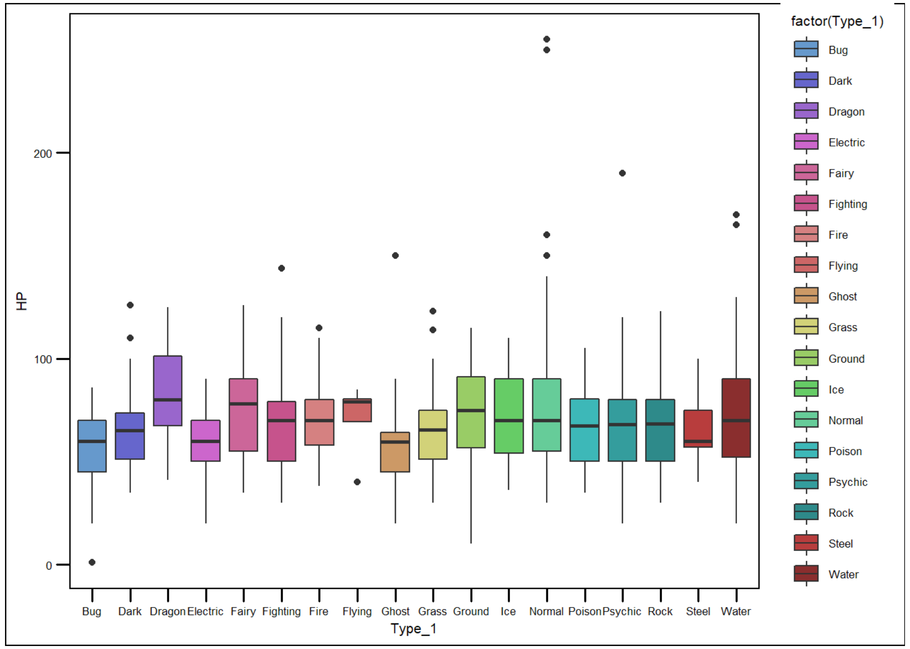





factor(Type_1)
Bug
Dark
Dragon
Electric
Fairy
Fighting
Fire
Flying
Ghost
Grass
Ground
Ice
Normal
Poison
Psychic
Rock
Steel
Water
Attack
0
50
100
150
Bug Dark Dragon Electric Fairy Fighting Fire Flying Ghost Grass Ground Ice Normal Poison Psychic Rock Steel Water
Type_1


factor(Type_1)
Bug
Dark
Dragon
Electric
Fairy
Fighting
Fire
Flying
Ghost
Grass
Ground
Ice
Normal
Poison
Psychic
Rock
Steel
Water
Speed
0
50
100
150
Bug Dark Dragon Electric Fairy Fighting Fire Flying Ghost Grass Ground Ice Normal Poison Psychic Rock Steel Water
Type_1


factor(Type_1)
Bug
Dark
Dragon
Electric
Fairy
Fighting
Fire
Flying
Ghost
Grass
Ground
Ice
Normal
Poison
Psychic
Rock
Steel
Water
Defense
0
50
100
150
200
Bug Dark Dragon Electric Fairy Fighting Fire Flying Ghost Grass Ground Ice Normal Poison Psychic Rock Steel Water
Type_1


factor(Type_1)
Bug
Dark
Dragon
Electric
Fairy
Fighting
Fire
Flying
Ghost
Grass
Ground
Ice
Normal
Poison
Psychic
Rock
Steel
Water
HP
0
100
200
Bug Dark Dragon Electric Fairy Fighting Fire Flying Ghost Grass Ground Ice Normal Poison Psychic Rock Steel Water
Type_1

Logistic Regression

Data Preprocessing

Name	Squirtle
Type 1	Water
Type 2	NA
HP	44
Attack	48
Defense	65
Sp. Atk	50
Sp. Def	64
Speed	43
Generation	1
Legendary	FALSE

V.S

Name	Bulbasaur
Type 1	Grass
Type 2	Poison
HP	45
Attack	49
Defense	49
Sp. Atk	65
Sp. Def	65
Speed	45
Generation	1
Legendary	FALSE

Take the relative difference of the numeric variables
Calculate the type relationship parameter

Building Model

</>

```
> glm(winner_first_label~Diff_arrack + Diff_defense +
Diff_sp_defense + Diff_sp_attack + Diff_speed + Diff_HP +
First_pokemon_legendary + Second_pokemon_legendary +
type_relationship, data = train, family = ‘binomial’)

Coefficients:
      (Intercept)        Diff_attack       Diff_defense    Diff_sp_defense
        -1.055374           0.009622           0.001625           0.001308
   Diff_sp_attack         Diff_speed            Diff_HP  type_relationship
        -0.001128           0.065027           0.002242           0.793978

> sum(diag(table(test$winner_first_label,
test$predicted)))/nrow(test)

[1] 0.8831907
```

Model Evaluation and Diagnostics

- Goodness of Fit

```
> mylogit = glm(winner_first_label~Diff_arrack + Diff_defense + Diff_sp_defense + Diff_sp_attack + Diff_speed + Diff_HP + First_pokemon_legendary + Second_pokemon_legendary + type_relationship, data = train, family = ‘binomial’)
> mylogit_one = glm(winner_first_label~Diff_arrack + Diff_defense + Diff_sp_defense + Diff_sp_attack + Diff_speed + Diff_HP + type_relationship, data = train, family = ‘binomial’)

> anova(mylogit_one, mylogit, test = ‘Chisq’)
```

```
Analysis of Deviance Table

Model 1: winner_first_label ~ Diff_attack + Diff_defense + Diff_sp_defense +
    Diff_sp_attack + Diff_speed + Diff_HP + type_relationship
Model 2: winner_first_label ~ Diff_attack + Diff_defense + Diff_sp_defense +
    Diff_sp_attack + Diff_speed + Diff_HP + First_pokemon_legendary +
    Second_pokemon_legendary + type_relationship
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1     37493      27115
2     37491      27107  2   7.5584  0.02284 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Statistical Tests for Individual Predictors

Wald Test

The Wald statistic is defined as (e.g. Wasserman (2006): [All of Statistics], pages 153, 214-215):

$$W = \frac{(\hat{\beta} - \beta_0)}{\widehat{se}\ (\hat{\beta})} \sim \mathcal{N}\ (0,1)$$

or

$$W = \frac{\left(\hat{\beta} - \beta_0\right)^2}{\widehat{var}\ (\hat{\beta})} \sim \mathcal{X}_1^2$$

Wald Test

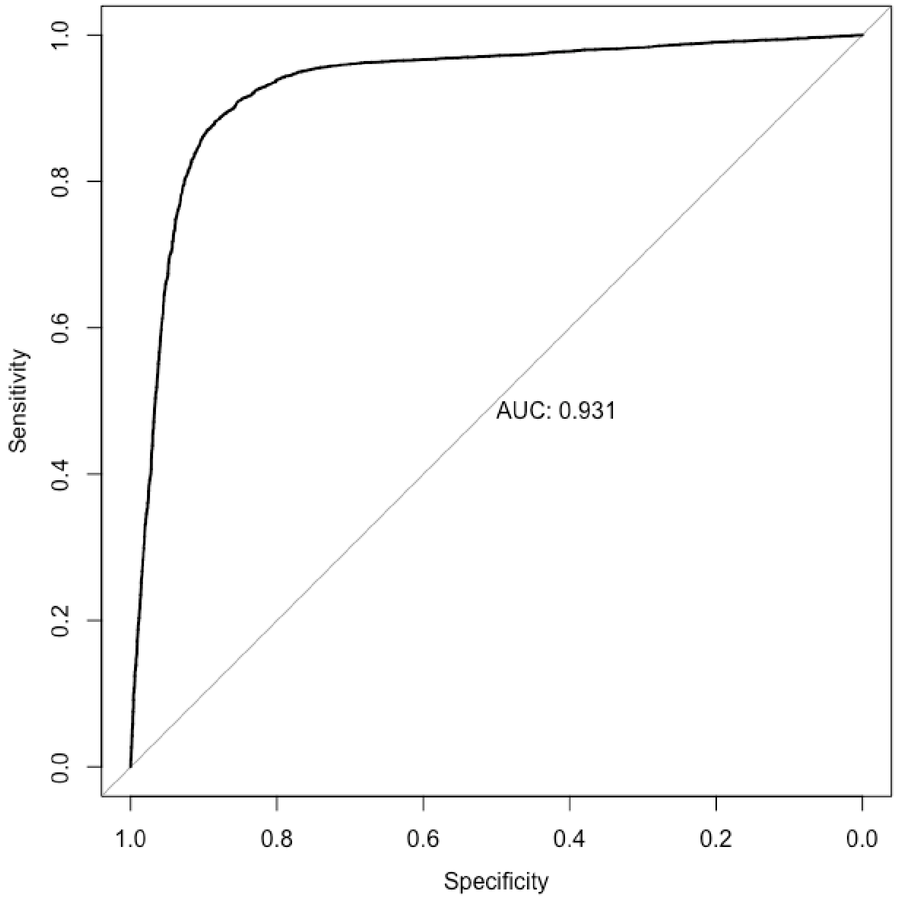
```
> summary(mylogit_one)
```

```
> summary(mylogit_one)

Call:
glm(formula = winner_first_label ~ Diff_attack + Diff_defense +
    Diff_sp_defense + Diff_sp_attack + Diff_speed + Diff_HP +
    type_relationship, family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.1457  -0.5364  -0.0818   0.5348   4.3381

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)       -1.0553738  0.0383047 -27.552  < 2e-16 ***
Diff_attack        0.0096219  0.0004580  21.009  < 2e-16 ***
Diff_defense       0.0016250  0.0004460   3.644 0.000268 ***
Diff_sp_defense    0.0013079  0.0005350   2.445 0.014504 *
Diff_sp_attack    -0.0011282  0.0004601  -2.452 0.014195 *
Diff_speed         0.0650266  0.0007346  88.522  < 2e-16 ***
Diff_HP            0.0022421  0.0004966   4.515 6.32e-06 ***
type_relationship  0.7939779  0.0338008  23.490  < 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: 51870  on 37500  degrees of freedom
Residual deviance: 27115  on 37493  degrees of freedom
AIC: 27131

Number of Fisher Scoring iterations: 6
```

Validation of Predicted Values

- ROC Curves

ROC curves in logistic regression are used for determining the best cutoff value for predicting whether a new observation is a "failure" (0) or a "success" (1)



Cross Validation

</>

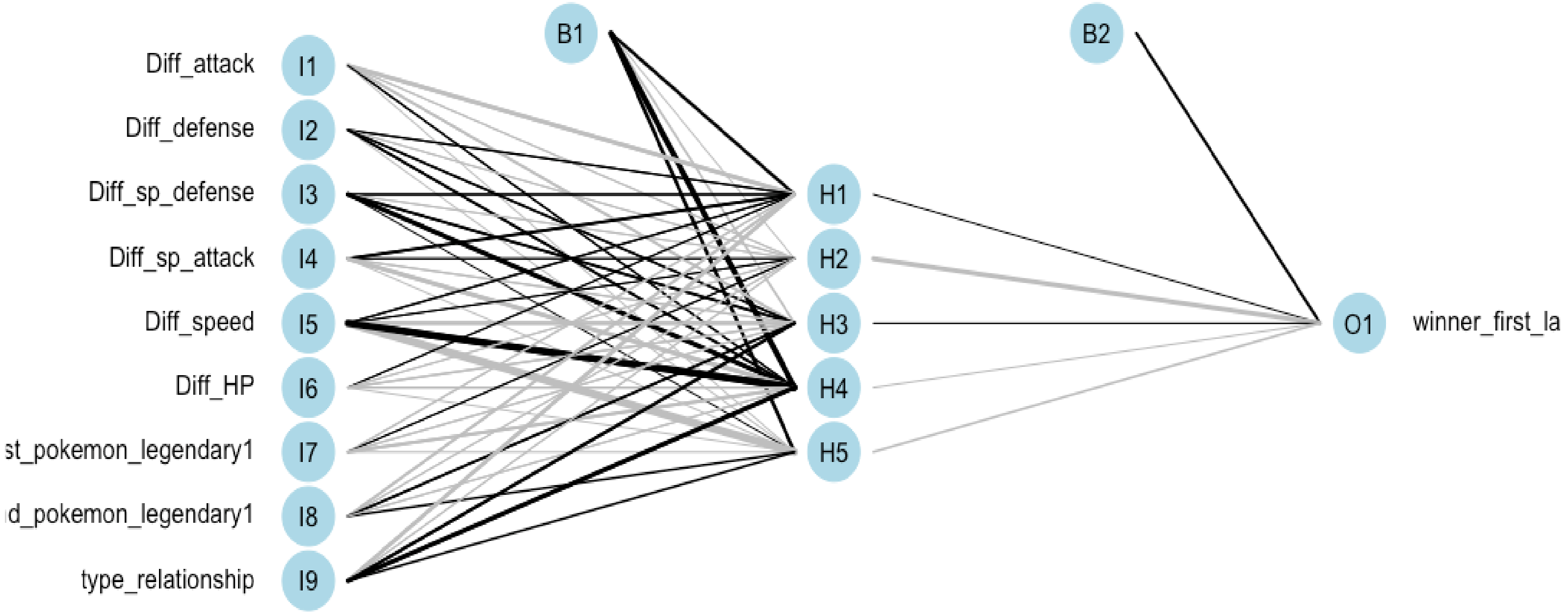
```
> ctrl = trainControl(method = ‘repeatedcv’,
number = 10, savePredictions = T)

> mylogit_n = (winner_first_label~Diff_arrack +
Diff_defense + Diff_sp_defense + Diff_sp_attack +
Diff_speed + Diff_HP + First_pokemon_legendary +
Second_pokemon_legendary + type_relationship,
data = train, family = ‘binomial’, method = ‘glm’,
trControl = ctrl, tunelength = 100)

> pred1 = predict(mylogit_n, test)

> confusionMatrix(data = pred1,
test$winner_first_label)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 5911  780
         1  688 5120

               Accuracy : 0.8826
                 95% CI : (0.8768, 0.8881)
    No Information Rate : 0.528
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.7642

 Mcnemar's Test P-Value : 0.01755

            Sensitivity : 0.8957
            Specificity : 0.8678
         Pos Pred Value : 0.8834
         Neg Pred Value : 0.8815
             Prevalence : 0.5280
         Detection Rate : 0.4729
   Detection Prevalence : 0.5353
      Balanced Accuracy : 0.8818

       'Positive' Class : 0
```

Neural Network

Building Neural Network

```
> prrr = predict(nn,
temp2[-split,], type =
'class')

> sum(diag(table(prrr,
test$winner_first_label)))
/nrow(test)

[1] 0.9479158
```



Cross Validation

</>

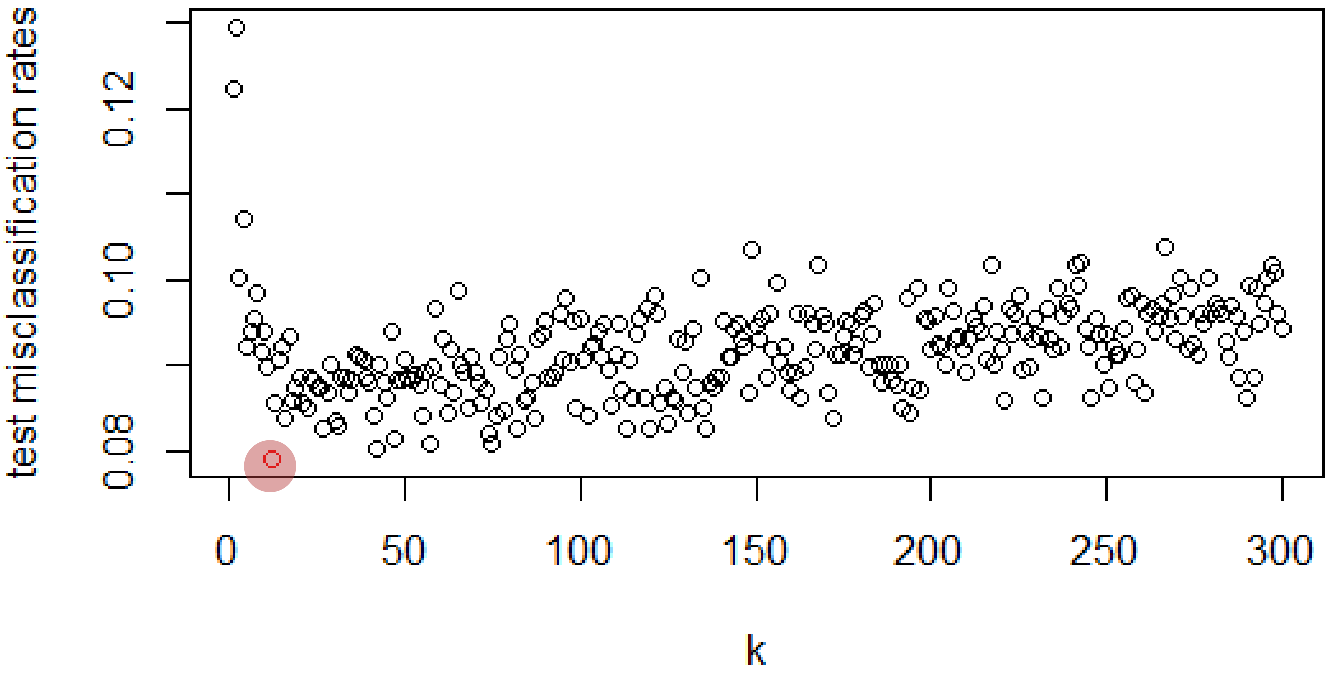
```
rate = rep(0,100)
for(a in 1:100){
  split <- createDataPartition(y=temp1$winner_first_label, p = 0.9, list = FALSE)
  train <- temp1[split,]
  test <- temp1[-split,]
  nn = nnet(winner_first_label ~ Diff_attack + Diff_defense + Diff_sp_defense + Diff_sp_attack + Diff_speed + Diff_HP +
              First_pokemon_legendary+Second_pokemon_legendary + type_relationship,
            data = temp2,subset = split,size = 5,rang = 0.1,decay = 5e-4,maxit = 10000)

  prrr = predict(nn,temp2[-split,],type = 'class')
  rate[a] = sum(diag(table(prrr, test$winner_first_label)))/nrow(test)

  mean(rate)
}
> rate
  [1] 0.9580 0.9464 0.9548 0.9550 0.9568 0.9540 0.9592 0.9600 0.9452 0.9480 0.9538 0.9536 0.9506 0.9520 0.9568 0.9552 0.9536 0.9556 0.9530 0.9482
 [21] 0.9512 0.9480 0.9572 0.9500 0.9564 0.9552 0.9560 0.9572 0.9578 0.9434 0.9560 0.9554 0.9538 0.9466 0.9582 0.9468 0.9516 0.9468 0.9520 0.9552
 [41] 0.9562 0.9512 0.9522 0.9444 0.9606 0.9490 0.9536 0.9454 0.9562 0.9526 0.9558 0.9498 0.9460 0.9504 0.9582 0.9524 0.9514 0.9578 0.9478 0.9506
 [61] 0.9554 0.9406 0.9474 0.9546 0.9570 0.9434 0.9550 0.9468 0.9556 0.9488 0.9514 0.9432 0.9564 0.9564 0.9572 0.9548 0.9544 0.9558 0.9524 0.9500
 [81] 0.9556 0.9538 0.9526 0.9596 0.9584 0.9558 0.9476 0.9494 0.9574 0.9516 0.9520 0.9532 0.9606 0.9520 0.9564 0.9548 0.9468 0.9442 0.9574 0.9564
> mean(rate)
[1] 0.952784
```

KNN

How to choose k?

Use cross validation to find out the k value that return us the min error.

```
</>
> kchoose = which.min(error)

> kchoose

[1] 12
```



Model evaluation

- Cross validation & Accuracy

```
> knn.opt = knn(train = train[,2:10], test = test
[,2:10], cl = train[,1], k = 12)
> table = CrossTable(x = test[,1], y = knn.opt,
dnn = c("Actual", "Predicted"), prop.chisq = F)
> accuracy = sum(diag(table$t))/sum(table$t)

> accuracy
[1] 0.9124

> mean(error)
[1] 0.08974
```

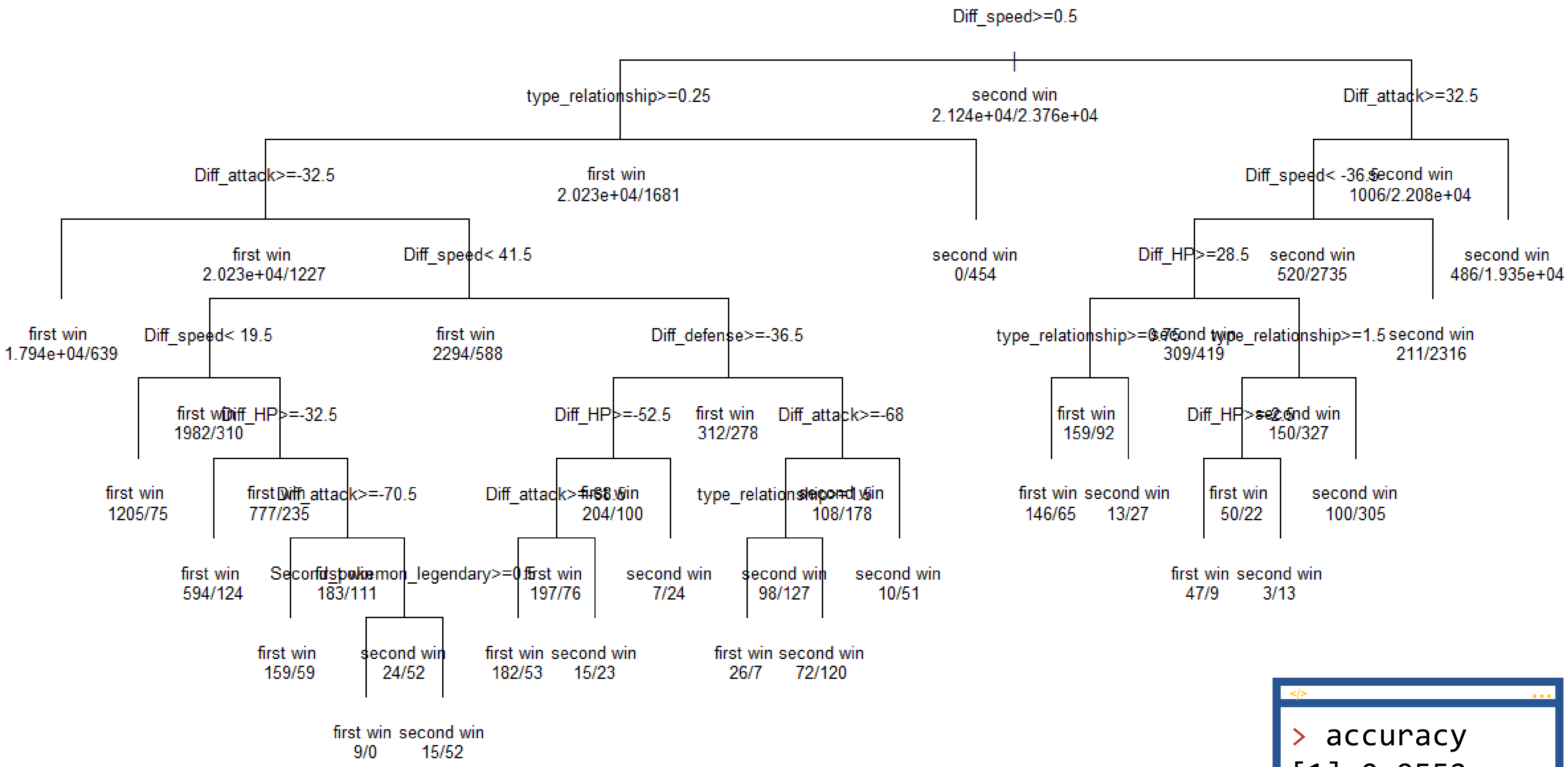
```
Total Observations in Table:  5000

             | Predicted
      Actual |   first win |  second win |   Row Total |
-------------|-------------|-------------|-------------|
   first win |        2168 |         198 |        2366 |
             |       0.916 |       0.084 |       0.473 |
             |       0.900 |       0.076 |             |
             |       0.434 |       0.040 |             |
-------------|-------------|-------------|-------------|
  second win |         240 |        2394 |        2634 |
             |       0.091 |       0.909 |       0.527 |
             |       0.100 |       0.924 |             |
             |       0.048 |       0.479 |             |
-------------|-------------|-------------|-------------|
Column Total |        2408 |        2592 |        5000 |
             |       0.482 |       0.518 |             |
-------------|-------------|-------------|-------------|
```

KNN method when k = 12,
Accuracy = 0.9124

Decision Tree

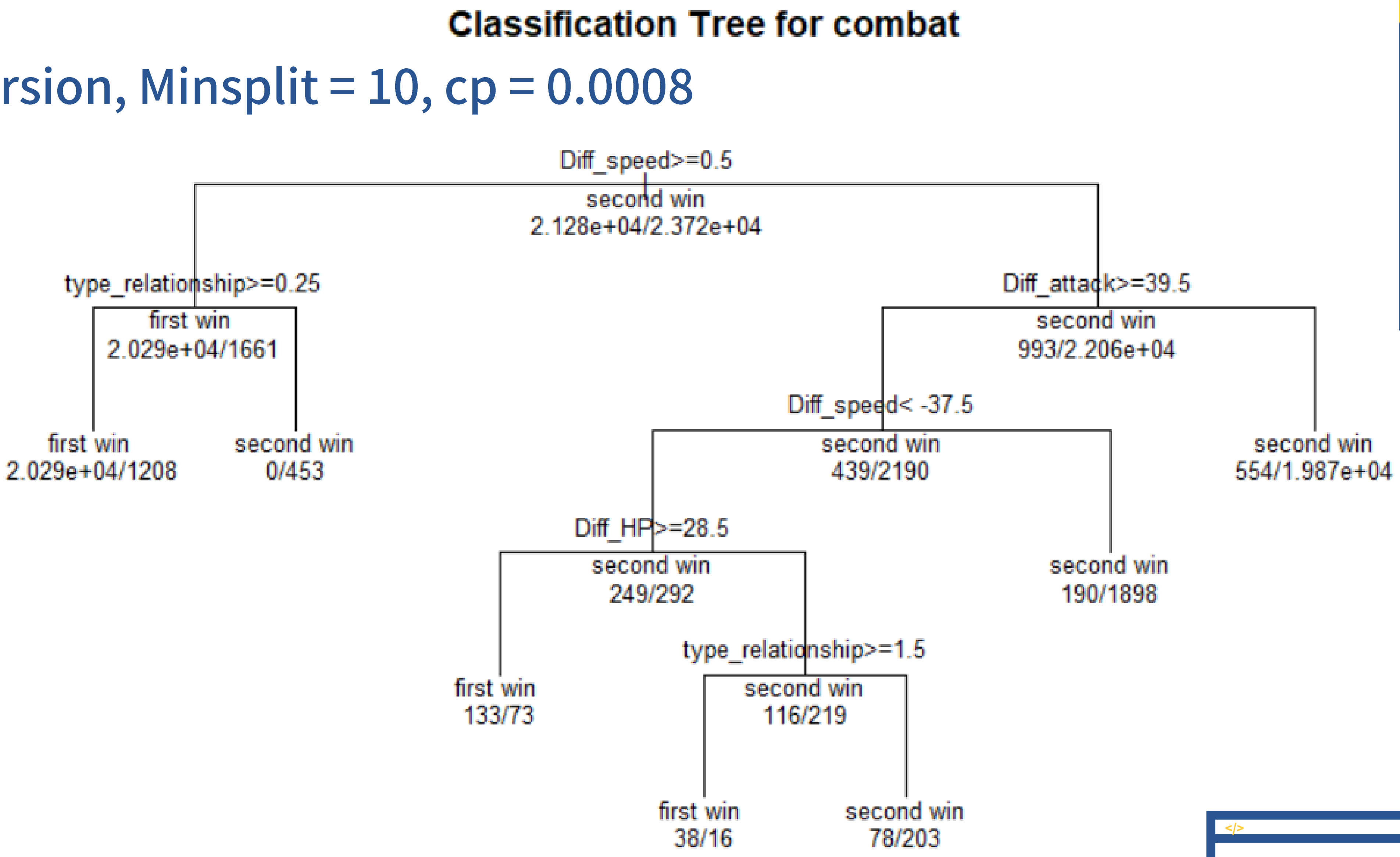
Classification Tree for combat

Minsplit = 10, cp = 0.0003



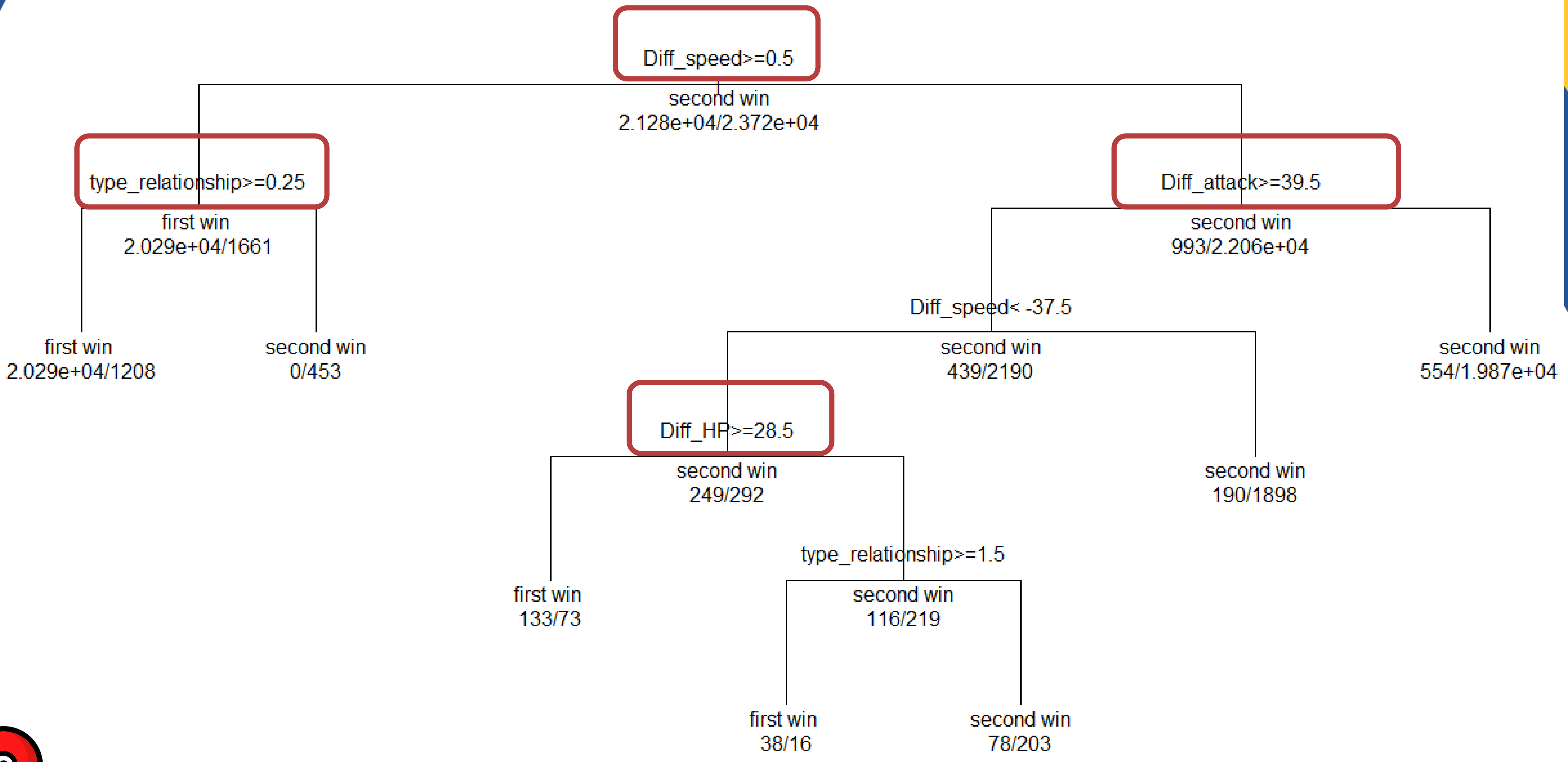

```
> accuracy
[1] 0.9552
```

Simplify version, Minsplit = 10, cp = 0.0008




```
> accuracy
[1] 0.9484
```

Pruned Classification Tree for combat



Example

- Speed difference >= 0.5 → A faster than B
- Type relationship Difference >= 0.25 →

    ① If No, which means <0.25, for example

    A belong to ground type, B belong to fly type, then B wins

    ② If Yes, which means >= 0.25, for example

    A belong to Fire type, B also belong to Fire type,

Model evaluation

Cross validation

```
> accuracy =
sum(diag(table$t))/sum(table$t
)
> accuracy
[1] 0.9484


> mean(error)
[1] 0.04876
```

```
Total Observations in Table:  5000

             | Predicted
      Actual |  first win | second win |  Row Total |
-------------|------------|------------|------------|
   first win |       2263 |        103 |       2366 |
             |      0.956 |      0.044 |      0.473 |
             |      0.944 |      0.040 |            |
             |      0.453 |      0.021 |            |
-------------|------------|------------|------------|
  second win |        133 |       2501 |       2634 |
             |      0.050 |      0.950 |      0.527 |
             |      0.056 |      0.960 |            |
             |      0.027 |      0.500 |            |
-------------|------------|------------|------------|
Column Total |       2396 |       2604 |       5000 |
             |      0.479 |      0.521 |            |
-------------|------------|------------|------------|
```

Supporting Vector Machine

SVM

- Ideas:

  separate the combat result (the first Pokémon win or lose) according to these new differencing variables.

</>

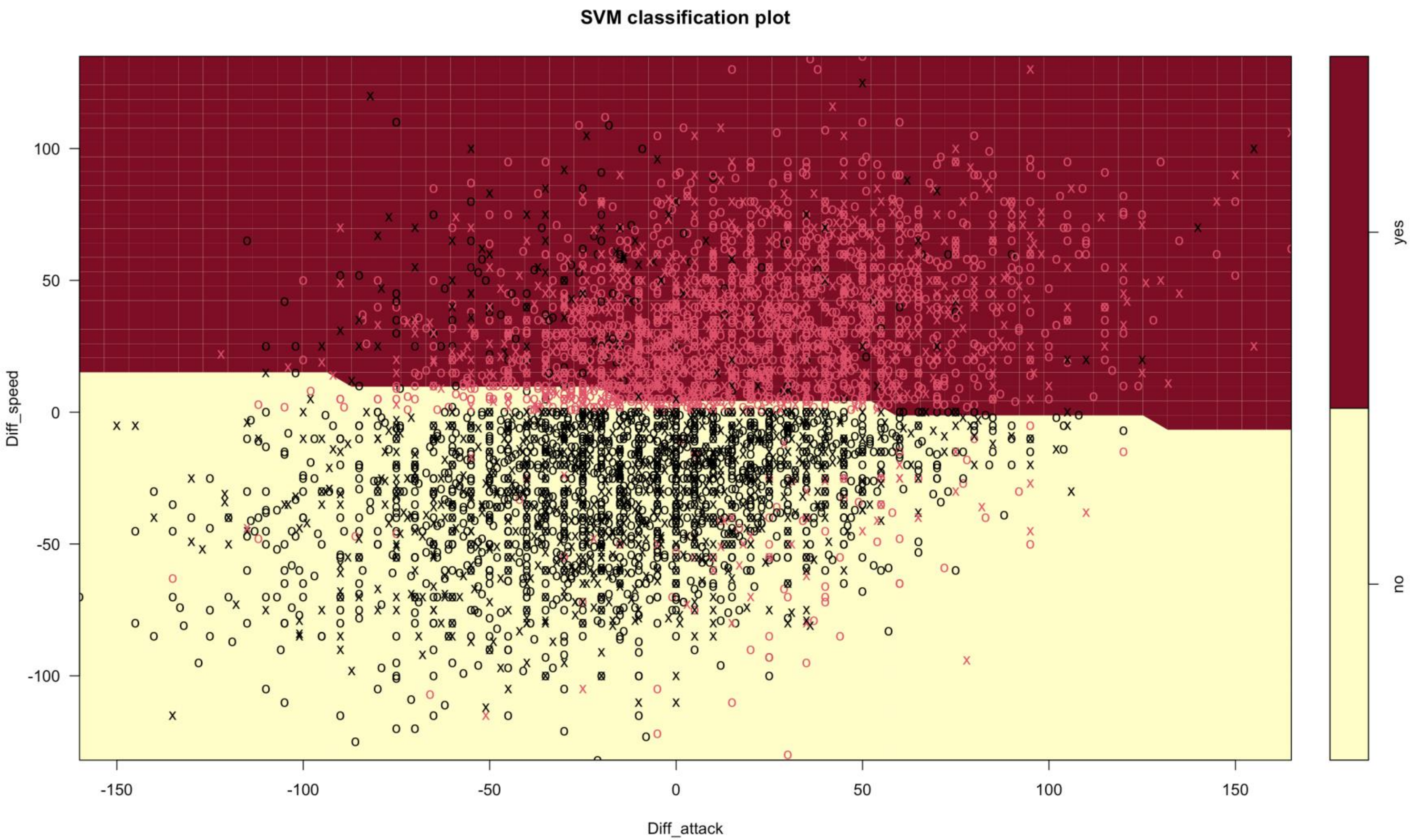
```
> SVM_model = svm(winner_first_label~.,data = df_train,kernel="linear", cost = 5, scale = T)
> SVM_test = predict(SVM_model, df_test[-c(1)])
> table_SVM = table(SVM_test,df_test[,1])
> SVM_accuracy[i] = sum(diag(table_SVM))/dim(df_test)[1]

> mean(SVM_accuracy)
[1] 0.9096201
```

Note:
'df_train' & 'df_test': training data and test data respectively
the parameter 'cost': penalty term in the Lagrange formulation
the parameter 'scale' : to scale the data

SVM

- linear

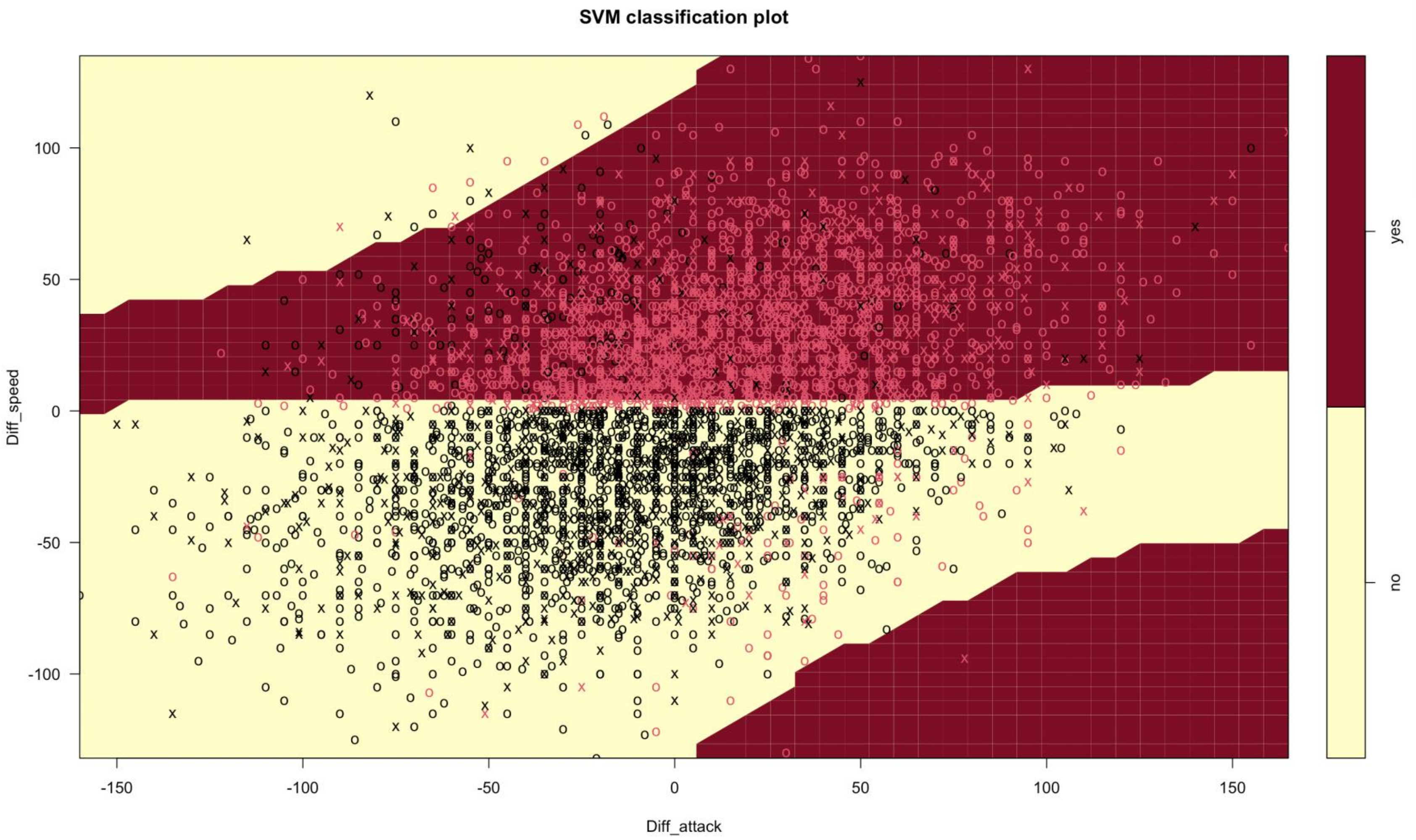
```
</>
> SVM_accuracy
[1] 0.899416
```



SVM

- polynomial

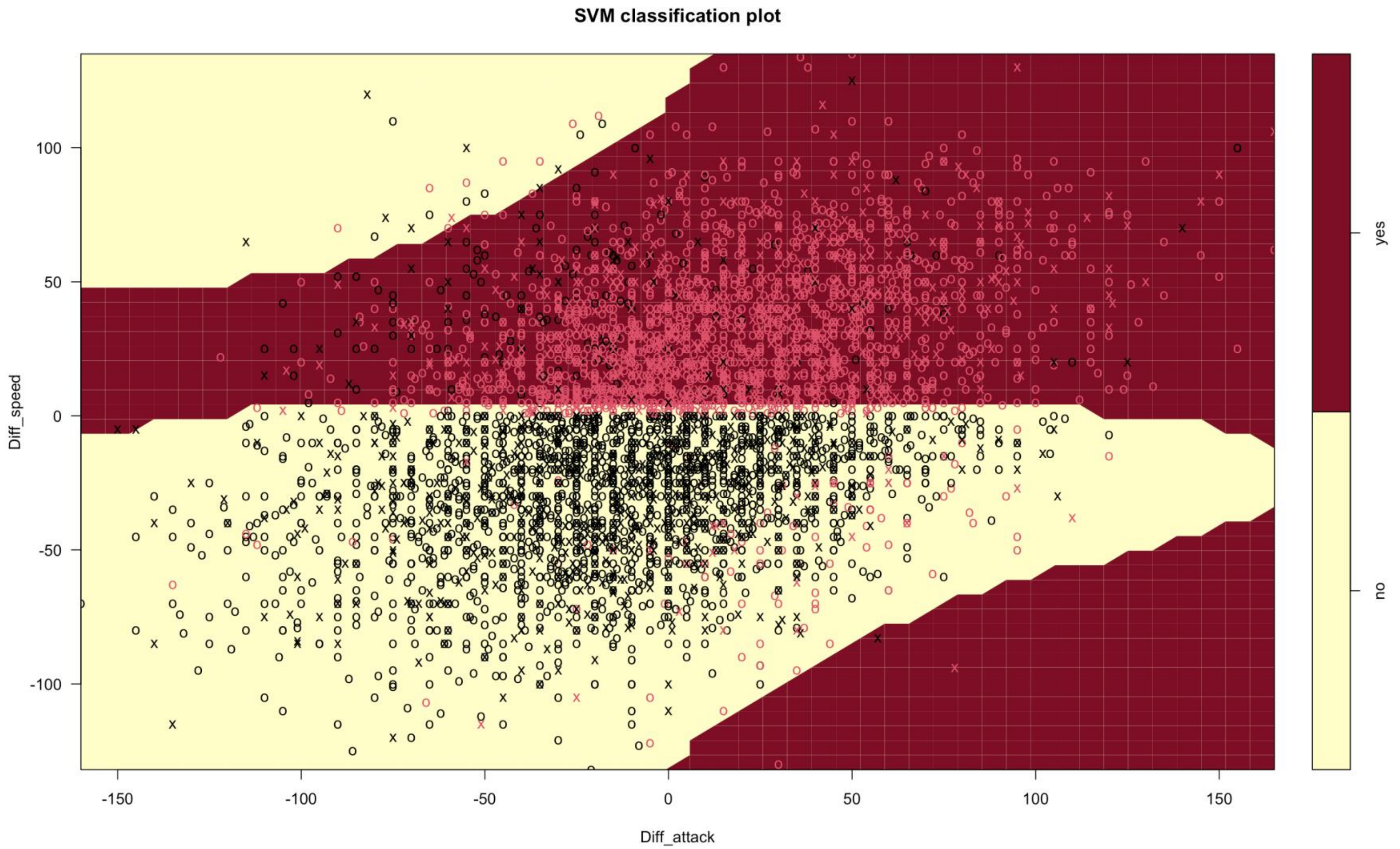
```
> SVM_accuracy
[1] 0.9384123
```



SVM

- radial

```
</>
> SVM_accuracy
[1] 0.9394121
```



Linear Discriminant Analysis

LDA

- Ideas:

  LDA tries to retain most of the between-class variance in the data

```
> LDA_model = lda(winner_first_label~.,data = df_train)
> LDA_test = predict(LDA_model, newdata = df_test[-c(1)])
> table_LDA = table(LDA_test$class, df_test[,1])
> LDA_accuracy[i] = sum(diag(table_LDA))/dim(df_test)[1]

> mean(LDA_accuracy)
[1] 0.8782244
```

LDA

Bayesian

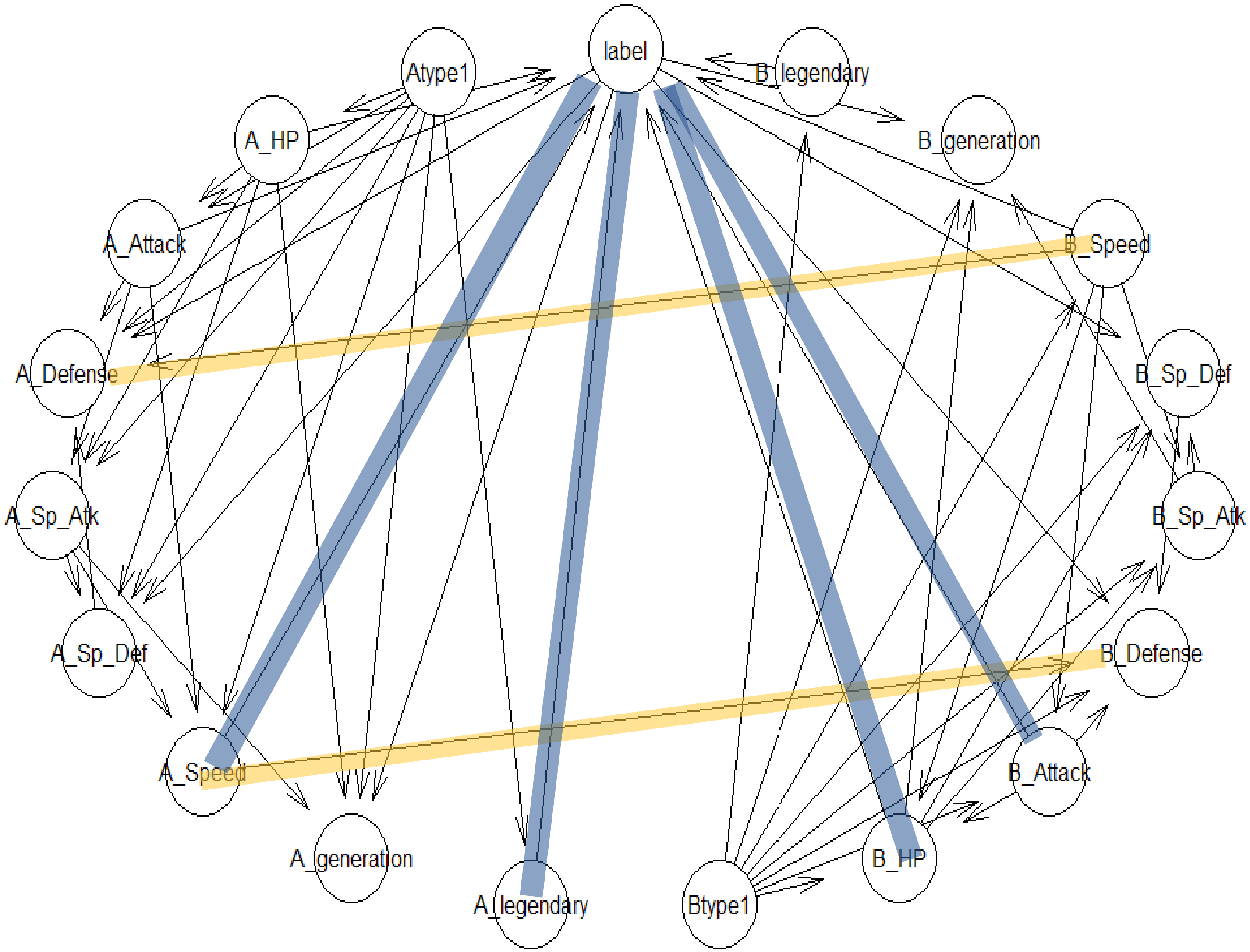
Bayesian Network

</>

```
> library(bnlearn)
> bn = hc(info)
```

```
  Bayesian network learned via Score-based methods

  model:
   [Atype1][Btype1][A_HP|Atype1][A_legendary|Atype1][B_Speed|Btype1][B_legendary|Btype1][A_Attack|
Atype1:A_HP]
   [B_Attack|Btype1:B_Speed][A_Sp_Atk|Atype1:A_HP:A_Attack][B_HP|Btype1:B_Attack:B_Speed]
   [A_Speed|Atype1:A_Attack:A_Sp_Atk][B_Sp_Atk|Btype1:B_HP:B_Speed]
   [label|A_HP:A_Attack:A_Speed:A_legendary:B_HP:B_Attack:B_Speed:B_legendary][A_Sp_Def|Atype1:A_H
P:A_Sp_Atk:label]
   [B_Sp_Def|Btype1:B_HP:B_Sp_Atk:label][A_Defense|Atype1:A_Attack:A_Sp_Def:B_Speed:label]
   [B_Defense|A_Speed:Btype1:B_Attack:B_Sp_Def:label]
  nodes:                                 17
  arcs:                                  46
    undirected arcs:                     0
    directed arcs:                       46
  average markov blanket size:           9.29
  average neighbourhood size:            5.41
  average branching factor:              2.71

  learning algorithm:                    Hill-Climbing
  score:                                 BIC (cond. Gauss.)
  penalization coefficient:              5.409889
  tests used in the learning procedure:  1008
  optimized:                             TRUE
```

Bayesian Network



Bayesian Network

```
> fit <- bn.fit(bn, data = info)
> cpquery(fit, event=(Atype1==“Rock”), evidence =
 (A_Legendary==“TRUE”)& (A_generation==“1”)

[1] 0.2127
```

Naïve Bayesian

```
> f = naiveBayes(label~.,data = train)

> CrossTable(test$label,p,prop.r = F,prop.c = F,prop.t = T,prop.chisq = F)
> accuracy

[1] 0.809
```

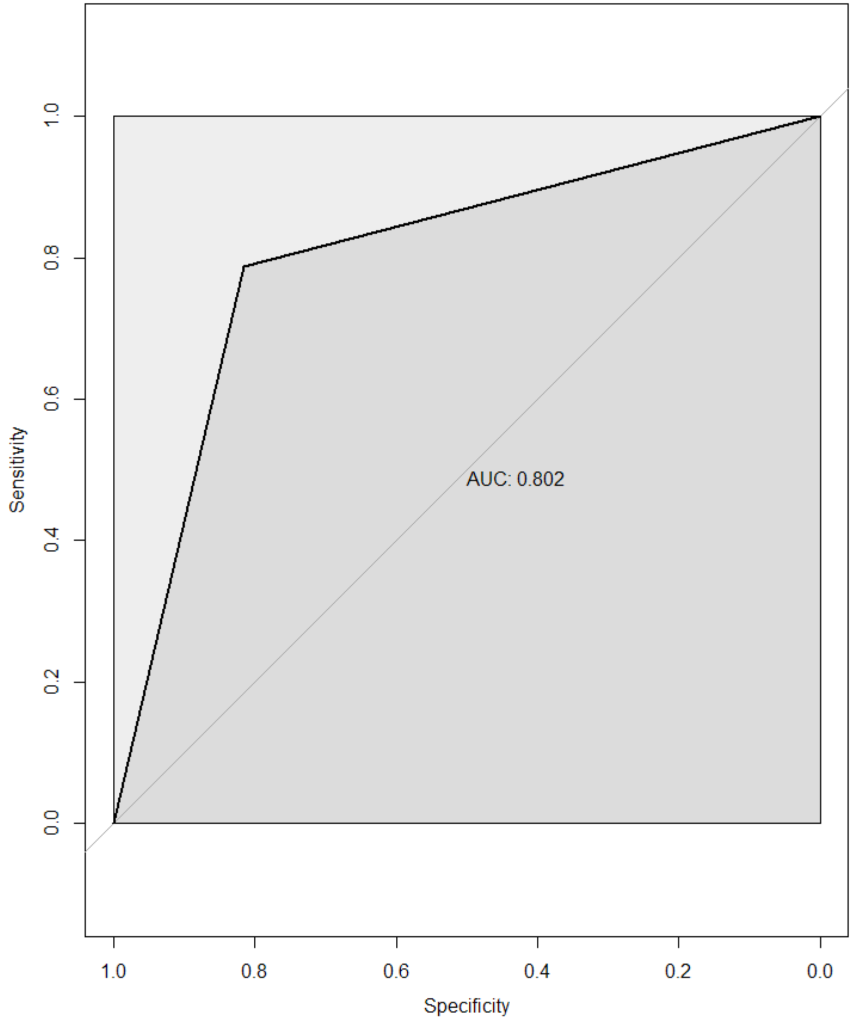
Total Observations in Table: 10000

test$label	p: 0	1	Row Total
0	4277 0.428	989 0.099	5266
1	927 0.093	3807 0.381	4734
Column Total	5204	4796	10000

Naïve Bayesian

</>

```
> library(pROC)
> pred <- ordered(p)
> pre <- roc(test$label, pred)


> plot(pre,print.auc = T,auc.polygon
= T,max.auc.polygon = T)
```



Interpretation

- Significant factor
  - Logistic regression: <u>Speed</u>, <u>attack</u>, type, HP, defense
  - Bayes network: <u>Speed</u>, HP, <u>attack</u>, dependency
  - Decision tree: <u>Speed</u>, type, <u>attack</u>
- Speed and Attack seems to be the most decisive factor
- High accuracy of different models

Discussion

- Variable “Secondary type” is abandoned
- Use relative difference to estimate the difference of the two combating Pokémon on different variables
- Speed plays the decisive role.
- Base statistics (Ignore the evolution of the Pokémon)
- The type relationship actually should be a multiplicative coefficient of attack, defense, special attack, special defence
- Player’ s operation
  - Operation proficiency
  - Use props to improve Pokémon’ s attributes
  - Different kinds of attack/defense have different effects