# p8106_midterm_wk2343

Gavin Ko

4/4/2020

## 1. Introduction

### Brief about Pokemon and Interested Question

Pokemon is a well desinged video game characters that have detailed numeric settings about their characteristics. Among all pokemons, some of them are classfied as "legendary". In this project, we wish to predict whether a pokemon is legendary by the properties of that specific pokemon.

### Data cleaning process

As characters in well designed video game, we assume that there's already tidy and organized statistics for each pokemon. Thankfully, we do find a dataset recording detailed information of all pokemons from generation 1 to 6 on kaggle (source: https://www.kaggle.com/alopez247/pokemon).

After downloading and reading in the dataset, Our dataset have 721 pokemons' data listed with 23 variables. These variables include their name, basic information like body weight, height and battle related information like attack, defense.

After a quick look on the dataset, we find that `total`, which indicates *species strength*, is simply the sum of `hp`, `attack`, `defense`, `sp_atk`, `sp_def` and `speed`. Also, to aviod collinearity issue, we would forfeit these six variables and keep `total` for further analysis. On the other hand, `type`, `color`, `egg_group` and `body_style` are classified as characters but should be factors instead. Therefore, we need to factorize them.

## 2. EDA

### Table 1 for grouped data

Since we're mainly interested in whether a pokemon is legendary or not, we can summarize the dataset grouped by legendary status.

We can find some interesting triats from this grouping summary:

1) There's a total of 46 legendary pokemon among 721 pokemons, which is around 6%.
2) **type**: The most popular type for legendary pokemons are Flying (19.6%), Psychic(17.4%) and Dragon(15.2%).
3) **total(species strength)**: while the mean of normal pokemons are around 400, legendary pokemon seems to have much higher average at 620.
4) **has_gender**: While most normal pokemons do have gender(94.5%), legendary pokemons are the opposite(13.0%). This make discussing it's male proportion(`pr_male`) not proper since the sample size is too small.
5) **egg_group**: For legendary pokemons, we have **ALL** of them with Undiscoverd egg group. Therefore, once we know that a specific pokemon has this kind of egg, they're highly possible to be legendary.
6) **height and weight**: The average height and weight of legendary pokemons (2.45m, 201kg) seem to be much larger than those of normal pokemons' (1.06m, 47kg).

7) **catch rate**: legendary pokemons owns much lower average catch rate (6.65%) compared to those of normal pokemons(> 100%).

For further analysis, we would focus on these variables to build the prediction model.

```
pokemon_data_final =
  pokemon_data %>%
   select(-number, -name, -hp, -attack, -defense, -sp_atk, -sp_def, -speed, -generation,
          -color, -pr_male, -body_style, -has_mega_evolution)
```

# 3. Prediction Model Building

Apparently, `legendary` is a binary status, so we need to build a non-linear classification model.

```
# create partition
set.seed(88)
rowTrain = createDataPartition(y = pokemon_data_final$is_legendary,
                               p = 2/3,
                               list = F)

ctrl = trainControl(method = "repeatedcv",
                    repeats = 5,
                    summaryFunction = twoClassSummary,
                    classProbs = T)
```

## Logistic Regression

### First Attempt - Full Predictors

```
# This doesn't work
#set.seed(88)
#model.glm = train(x = pokemon_data_final[rowTrain, -4],
                  #y = pokemon_data_final$is_legendary[rowTrain],
                  #method = "glm",
                  #metric = "ROC",
                  #trControl = ctrl)
```

It doesn't seems working with such a large set of categorical varibles. Due to limited knowledge I have, I can only limit the discussion to continuous and binary predictors and try again.

### Second Attempt - Some Predictors

As a result, I kept `total`, `has_gender`, `height`, `weight` and `catch_rate` as predictors.

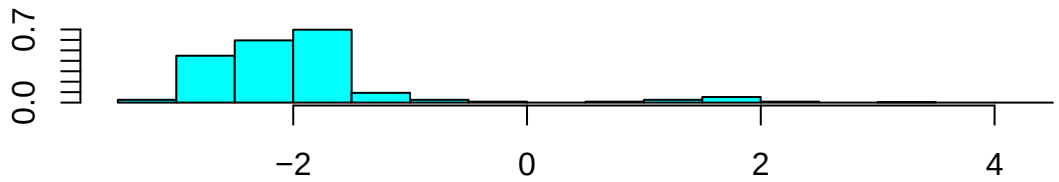
```
set.seed(88)
model.glm <- train(x = pokemon_data_final[rowTrain,c(3,5,8:10)],
y = pokemon_data_final$is_legendary[rowTrain],
method = "glm",
metric = "ROC",
trControl = ctrl)
```

## LDA Method

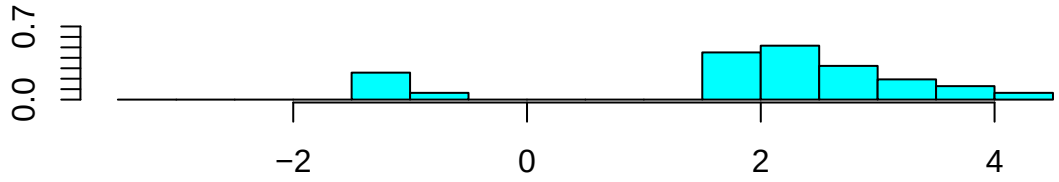
Since our response variable `is_legendary` is a binary outcome, it can be treated as categorical and we can make this porblem a classification question. Under this scenario, we can apply linear discriminant analysis.

```r
# Model building
lda.fit <- lda(is_legendary ~ total + has_gender + height_m + weight_kg + catch_rate,
               data = pokemon_data_final, subset = rowTrain)

plot(lda.fit)
```




group False




group True

## QDA Method

Another approach to classification problems is quadratic discriminant analysis.

```r
# Model building
qda.fit <- qda(is_legendary ~ total + has_gender + height_m + weight_kg + catch_rate,
               data = pokemon_data_final, subset = rowTrain)
```

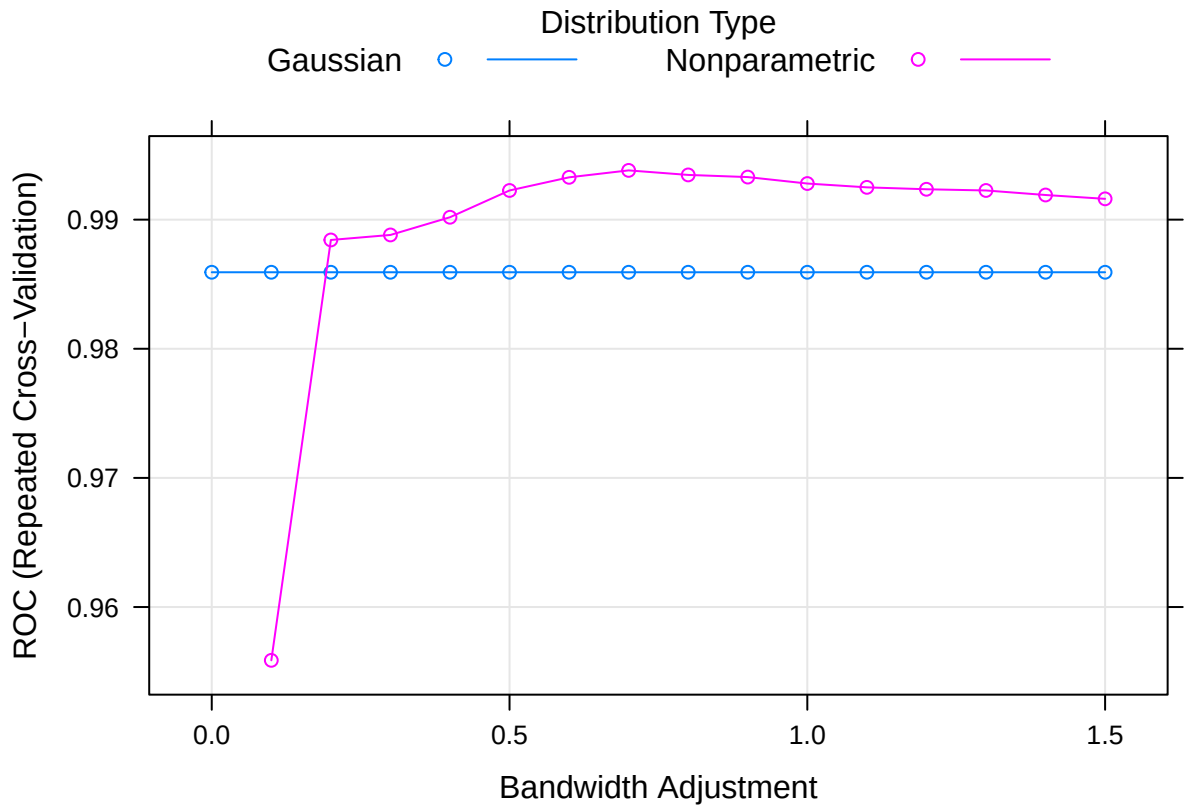
## NB Method

```r
# Model building
nbGrid <- expand.grid(usekernel = c(FALSE,TRUE),
                      fL = 2,
                      adjust = seq(0, 1.5, by = .1))

model.nb <- train(x = pokemon_data_final[rowTrain, c(3,5,8,9,10)],
                  y = pokemon_data_final$is_legendary[rowTrain],
                  method = "nb",
                  tuneGrid = nbGrid,
                  metric = "ROC",
```

```
                  trControl = ctrl)

plot(model.nb)
```




### Comparison of Training/ Testing Performance

```
# prediction performance
glm.pred <- predict(model.glm, newdata = pokemon_data_final[-rowTrain,], type = "prob")
lda.pred <- predict(lda.fit, newdata = pokemon_data_final[-rowTrain,], type = "prob")
qda.pred <- predict(qda.fit, newdata = pokemon_data_final[-rowTrain,], type = "prob")
nb.pred  <- predict(model.nb, newdata = pokemon_data_final[-rowTrain,], type = "prob")
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 4

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 30

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 35

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 37

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
```

```
## observation 40
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 44
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 46
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 47
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 48
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 52
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 57
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 69
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 80
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 82
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 89
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 95
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 103
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 106
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 107
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 113
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 120
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 123
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 124
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 131
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 144
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 152
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 158

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 159

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 160

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 161

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 165

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 166

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 167

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 169

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 177

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 206

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 207

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 208

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 211

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 217

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 218

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 235

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 239
```

```
# roc curve building
roc.glm <- roc(pokemon_data_final$is_legendary[-rowTrain], glm.pred[, 2], levels = c("False", "True"))
```

```
## Setting direction: controls < cases
```

```
roc.lda <- roc(pokemon_data_final$is_legendary[-rowTrain], lda.pred$posterior[ ,2], levels = c("False",
```

```
## Setting direction: controls < cases
```

```
roc.qda <- roc(pokemon_data_final$is_legendary[-rowTrain], qda.pred$posterior[ ,2], levels = c("False",
```

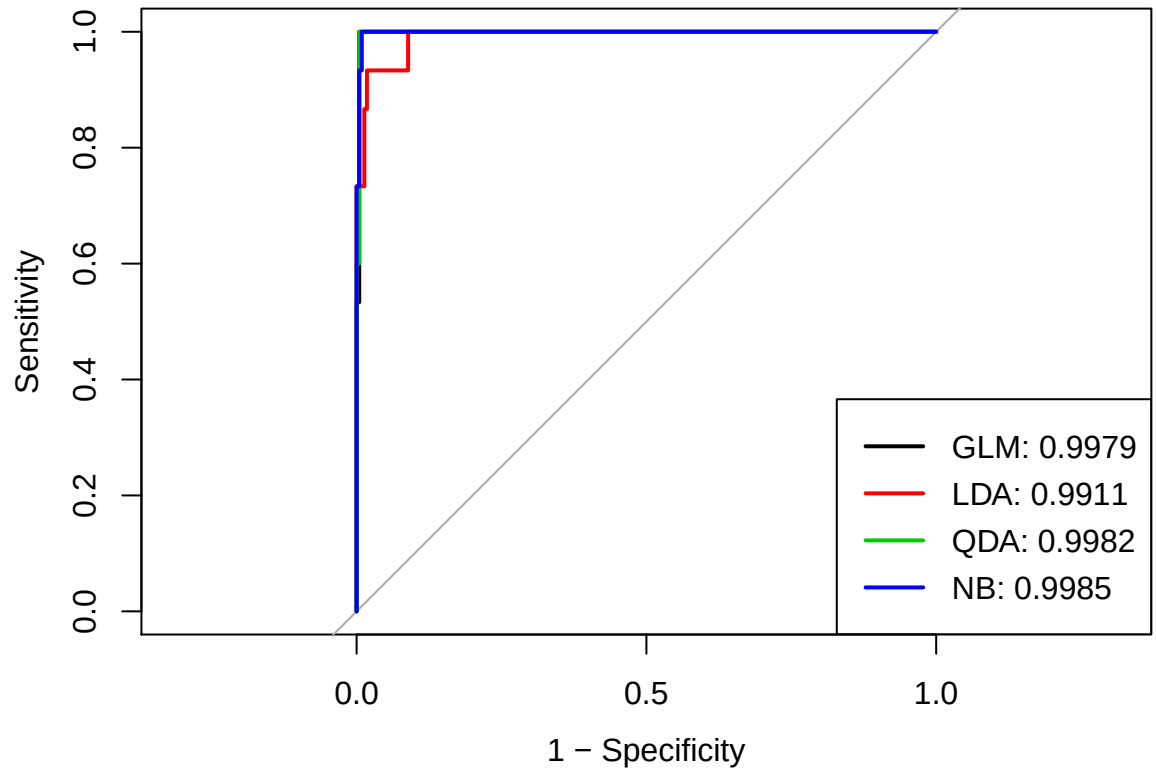
```
## Setting direction: controls < cases
```

```
roc.nb <- roc(pokemon_data_final$is_legendary[-rowTrain], nb.pred[, 2], levels = c("False", "True"))
```

```
## Setting direction: controls < cases
```

```
# auc
auc <- c(roc.glm$auc[1], roc.lda$auc[1], roc.qda$auc[1], roc.nb$auc[1])

# roc curve comparison
plot(roc.glm, legacy.axes = TRUE)
plot(roc.lda, col = 2, add = TRUE)
plot(roc.qda, col = 3, add = TRUE)
plot(roc.nb, col = 4, add = TRUE)
modelNames <- c("GLM","LDA","QDA","NB")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc, 4)), col = 1:4, lwd = 2)
```




# 4. Conclusion

## Important Predictors

It's not easy to tell which predictor is more important under the complicated mathematical structure of discrimination analysis. However, we can use logistic regression model as a reference of the importance of each predictors.

```
summary(model.glm$finalModel)
```

```
##
## Call:
```

```
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.19048  -0.00410  -0.00016   0.00000   1.06732
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -26.564727  12.263639  -2.166  0.03030 *
## total            0.047479   0.020525   2.313  0.02071 *
## has_genderTrue  -0.801533   1.035287  -0.774  0.43880
## height_m         0.772251   0.427424   1.807  0.07080 .
## weight_kg       -0.003798   0.003453  -1.100  0.27134
## catch_rate      -0.154592   0.054792  -2.821  0.00478 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 229.954  on 480  degrees of freedom
## Residual deviance:  30.403  on 475  degrees of freedom
## AIC: 42.403
##
## Number of Fisher Scoring iterations: 12
```

Accordingly, the two major components in predicting whether a pokemon is legendary are having gender or not, height and catch rate. This is consistent to what we've observed in exploratory data analysis. On the other hand, species strength(`total`) doesn't seem to have a huge effect on determining a pokemon to be legendary. This might be a result of the large scale of species strength.

## Model Comparison

All of the models in use have extremely high accuracy with AUC $> 0.99$. Among them, Naive Bayes Model stands out as the best model in AUC aspect. This is kind of contradictory to intution cause NB approach are suppose to be more suitable for larger p. Therefore, dispite the high AUC value, I would choose QDA as the final model.