Find Pokemon Battle Combinations

Lucas Jamar

2020-05-31

To create a strategy for determining an optimal combination of pokemons, we first plot the predicted portion of remaining HP of player 1 against the price of our available pokemons for each opponent. We firstly notice that it is nearly impossible to beat Mewtwo. Secondly, although possible, beating Golem and Krabby require substantial spending. Beating Caterpie, Raichu, Venusaur is relatively inexpensive. Therefore, the 3 cheapest pokemons are reserved for Mewtwo, Golem and Krabby in the case we cannot defeat them. Since losing to Mewtwo is certain, we only consider combinations of pokemons where one of the 3 cheapest pokemons is fighting it. To ensure that an increase in spending is translated into an increase of the target variable, we select only rows where the predicted target variable is equal to the cumulative maximum of the predicted target variable for each opponent. This reduces the number of rows to about 100. Because the number of combinations is now much smaller, we can create all combinations of our remaining available pokemons. From this, we calculate the total price, the mean of the predicted variable and the number of unique combinations. We then select the combination with 6 unique IDs with the highest mean of the predicted variable and submit our results.

Determine rows to skip and column names in order to read submission data only.

```
library(data.table)
library(ggplot2)

skip_rows <- data.table::fread("data/04_features/pokemon.csv", select = "Set")
skip_rows <- skip_rows[Set != "submission", .N + 1]
column_names <- data.table::fread("data/04_features/pokemon.csv", nrows = 0)
column_names <- colnames(column_names)
```

Combine submission data and submission predictions.

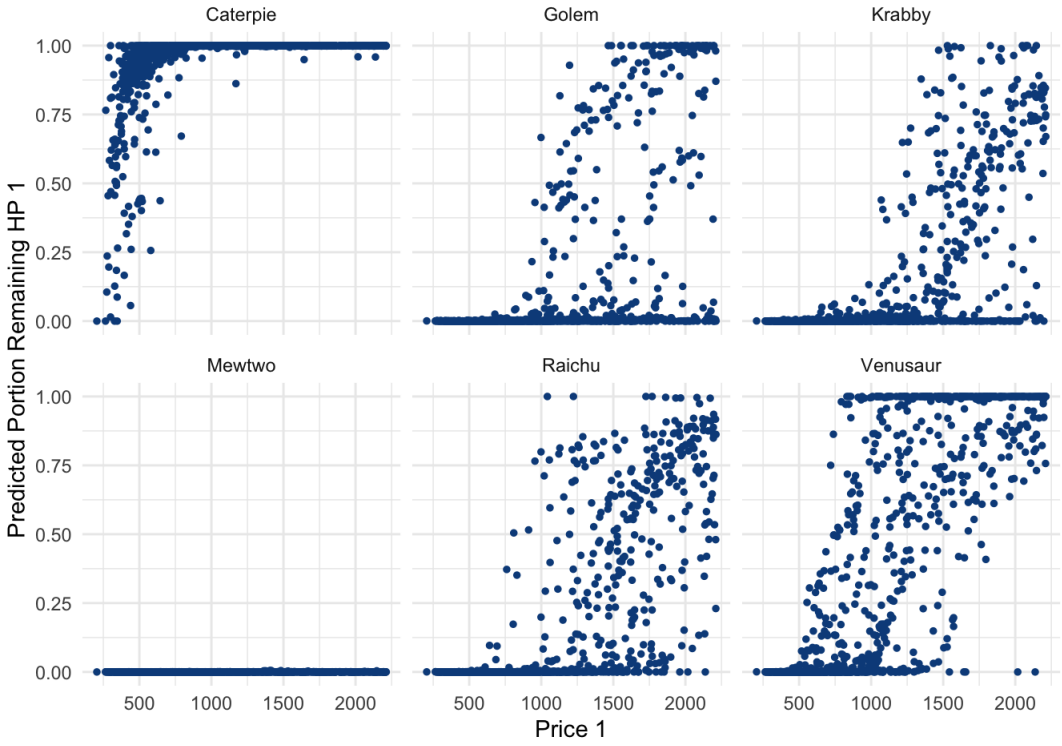
```
dt <- data.table::fread("data/04_features/pokemon.csv", skip = skip_rows, col.names = column_names)
predictions <- data.table::fread("data/07_model_output/submission_prediction.csv", select = "PredictedPortio
nRemainingHP_1")
dt <- cbind(dt, predictions)
rm(predictions)
```

Plot predicted portion remaining HP_1 against price_1 for each opponent.

```
plt <- ggplot2::ggplot(dt) +
  aes(x = Price_1, y = PredictedPortionRemainingHP_1) +
  geom_point(size = 1L, colour = "#0c4c8a") +
  theme_minimal() +
  facet_wrap(vars(Name_2)) +
  labs(x = "Price 1", y= "Predicted Portion Remaining HP 1")
ggplot2::ggsave(filename = "data/08_reporting/price_1_predicted_result.png", plot = plt)
```

```
## Saving 7 x 5 in image
```

```
print(plt)
```



Order available pokemons from cheapest to priciest. Order each price group from highest to lowest predicted HP_1. Calculate cumulative max price and previous cumulative max price. Fill missing previous cumulative max values with -1

```
data.table::setorder(dt, Price_1, -PredictedPortionRemainingHP_1)
dt[, CumMaxPredictedPortionRemainingHP_1 := cummax(PredictedPortionRemainingHP_1), by = Name_2]
dt[, PreviousCumMaxPredictedPortionRemainingHP_1 := data.table::shift(CumMaxPredictedPortionRemainingHP_1,1)
, by = Name_2]
data.table::setnafill(dt, cols = "PreviousCumMaxPredictedPortionRemainingHP_1", fill = -1)
```

Reserve the 3 cheapest pokemons for Mewtwo, Krabby and Golem since they are the hardest to fight. 100% chance of losing to Mewtwo so we only keep rows of Mewtwo with the 3 cheapest pokemons. We do not allow Caterpie, Raichu and Venusaur to fight the 3 cheapest.

```
third_cheapest <- sort(unique(dt$Price_1))[3]
three_cheapest <- dt[Name_2 %in% c("Mewtwo","Krabby","Golem") & Price_1 <= third_cheapest]
dt <- dt[Name_2 != "Mewtwo"]
dt <- dt[!(Name_2 %in% c("Caterpie","Raichu","Venusaur") & Price_1 <= third_cheapest)]
```

Apart from the 3 cheapest, keep only rows where Price_1 maximises the predicted remaining HP_1. Ensure cummax != previous cummax so that there are no repeated 1s. Combine again with 3 cheapest and ensure uniqueness.

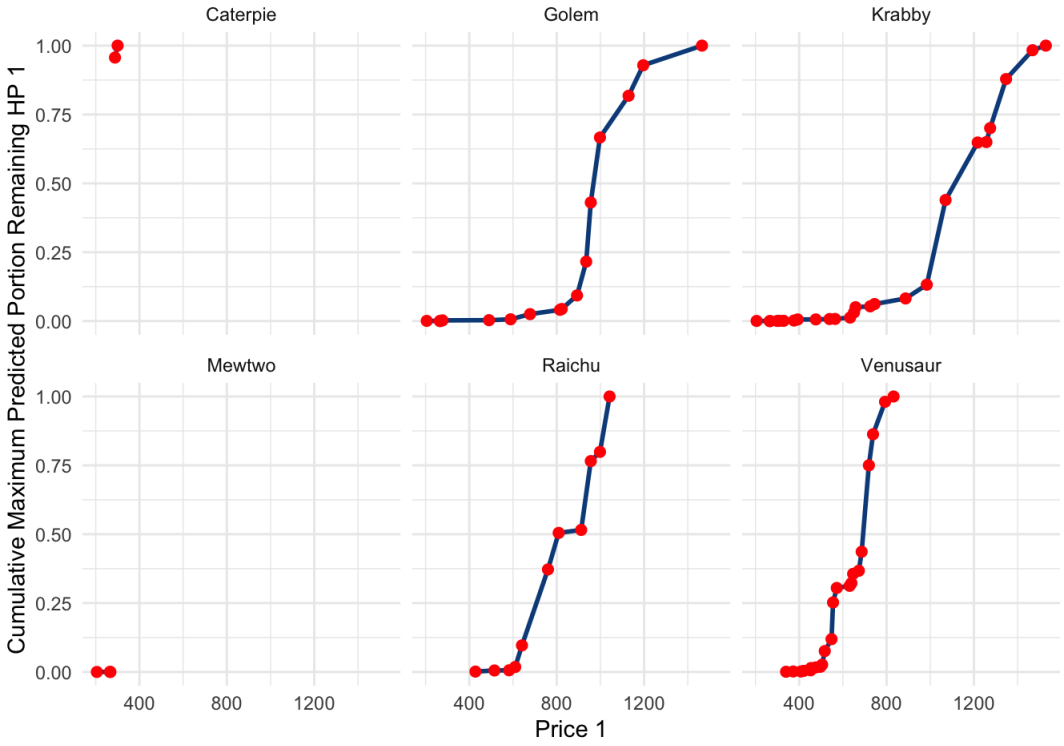
```
dt <- dt[(CumMaxPredictedPortionRemainingHP_1 == PredictedPortionRemainingHP_1 &
  CumMaxPredictedPortionRemainingHP_1 != PreviousCumMaxPredictedPortionRemainingHP_1)]
dt <- data.table::rbindlist(list(dt, three_cheapest))
dt <- unique(dt)
dt[, PreviousCumMaxPredictedPortionRemainingHP_1 := NULL]
```

Plot cummax predicted portion remaining HP_1 against price_1 for each opponent.

```
plt <- ggplot2::ggplot(dt) +
  aes(x = Price_1, y = PredictedPortionRemainingHP_1) +
  geom_line(size = 1L, colour = "#0c4c8a") +
  geom_point(size = 2L, colour = "red") +
  theme_minimal() +
  facet_wrap(vars(Name_2)) +
  labs(x = "Price 1", y = "Cumulative Maximum Predicted Portion Remaining HP 1")
ggplot2::ggsave(filename = "data/08_reporting/price_1_cummax_predicted_result.png", plot = plt)
```

```
## Saving 7 x 5 in image
```

```
print(plt)
```



Make rows of combinations of predicted portion remaining hp_1 per opponent. Then calculate the average remaining HP_1 (row mean) and keep that column only.

```
combinations <- dt[, .(list(PredictedPortionRemainingHP_1)), by = Name_2]$V1
combinations <- expand.grid(combinations)
data.table::setDT(combinations)
combinations[, MeanPredictedPortionRemainingHP_1 := rowMeans(.SD)]
combinations = combinations[, .(MeanPredictedPortionRemainingHP_1)]
```

Make rows of combinations of price_1 per opponent. Then calculate total price and combine with mean HP_1

```
combinations_price_1 <- dt[, .(list(Price_1)), by = Name_2]$V1
combinations_price_1 <- expand.grid(combinations_price_1)
data.table::setDT(combinations_price_1)
combinations_price_1[, TotalPrice := rowSums(.SD)]
combinations <- cbind(combinations, combinations_price_1[, .(TotalPrice)])
rm(combinations_price_1)
```

Make rows of combinations of pokemons IDs per opponent. Then rename the columns by opponent and combine with mean hp_1 and total price.

```
combinations_ids <- dt[, .(list(SelectedPokemonID)), by = Name_2]
opponents <- combinations_ids$Name_2
combinations_ids <- combinations_ids$V1
combinations_ids <- expand.grid(combinations_ids)
data.table::setDT(combinations_ids)
data.table::setnames(combinations_ids, colnames(combinations_ids), opponents)
combinations <- cbind(combinations_ids, combinations)
rm(combinations_ids)
```

Keep only the combinations that are cheaper than 3500. Order combinations from highest to lowest mean predicted hp 1, followed by from cheapest to most expensive.

```
combinations <- combinations[TotalPrice <= 3500]
data.table::setorder(combinations, -MeanPredictedPortionRemainingHP_1, TotalPrice)
```

Ensure that Golem, Krabby and Mewtwo are not fighting the same 3 cheap pokemons.

```
combinations[, Index := .I]
combinations[, UniqueIDs := data.table::uniqueN(c(Golem,Krabby,Mewtwo)), by = Index]
combinations <- combinations[UniqueIDs == 3]
```

Calculate the number of unique IDs, if the best combination has less than 6 unique IDs, print a warning message. Then, keep only the

combinations with 6 unique IDs, select the top combination and melt it.

```
combinations[, UniqueIDs := data.table::uniqueN(c(Caterpie,Golem,Krabby,Mewtwo,Raichu,Venusaur)), by = Index
]
if(combinations[1, UniqueIDs] < 6) {
  print("Warning: Best combination is not unique. Selecting less optimal unique combination")
}
combinations <- combinations[UniqueIDs == 6]
combinations[, c("UniqueIDs", "Index") := NULL]
combinations <- combinations[1]
combinations <- data.table::melt.data.table(combinations,
  measure.vars = opponents,
  value.name = "ID",
  variable.name = "Name_2"
)
```

Merge onto original data and keep only the ones with matching IDs

```
dt[combinations, on = .(Name_2 = Name_2), ID := ID]
dt <- dt[SelectedPokemonID == ID]
```

Merge onto submission data and write submission to CSV

```
submission <- data.table::fread("data/01_raw/Submission.csv")
submission[dt, on = .(Name_2 = Name_2), SelectedPokemonID := ID]
data.table::fwrite(submission, "data/07_model_output/Submission.csv")
```