Model Training

Lucas Jamar

2020-05-31

Using the optimal parameters found during our grid search, we train our final lightGBM L2 regressor of the difference between the true and expected portion of remaining HP of player 1. Once again, Name_2 was removed from the list of features. We then use our new model to make predictions on the combinations of submission and available pokemons data and we compute some feature importance metrics.

Read in the data with features

```
library(MLmetrics)
library(lightgbm)
library(data.table)

train_dt <- data.table::fread("data/04_features/pokemon.csv")
```

Features to keep for modelling

```r
feature_columns <- c(
  "Name_1",
  "Level_1",
  "Price_1",
  "HP_1",
  "Attack_1",
  "Defense_1",
  "Sp_Atk_1",
  "Sp_Def_1",
  "Speed_1",
  "Legendary_1",
  "Level_2",
  "Price_2",
  "HP_2",
  "Attack_2",
  "Defense_2",
  "Sp_Atk_2",
  "Sp_Def_2",
  "Speed_2",
  "Legendary_2",
  "WeatherAndTime",
  "MaxStrength_1",
  "MaxStrength_2",
  "Type_1",
  "Type_2",
  "WeatherInfluence_1",
  "WeatherInfluence_2",
  "Modifier_1",
  "Modifier_2",
  "Damage_1",
  "Damage_2",
  "Sp_Damage_1",
  "Sp_Damage_2",
  "MaxDamage_1",
  "MaxDamage_2",
  "ExpectedRounds_1_ToDefeat_2",
  "ExpectedRounds_2_ToDefeat_1",
  "ExpectedRounds",
  "ExpectedRemainingHP_1",
  "ExpectedRemainingHP_2",
  "ExpectedBattleResult",
  "ExpectedPortionRemainingHP_1",
  "RatioLevel",
  "RatioPrice",
  "RatioHP",
  "RatioAttack",
  "RatioDefense",
  "RatioSp_Atk",
  "RatioSp_Def",
  "RatioSpeed",
  "RatioWeatherInfluence",
  "RatioDamage",
  "RatioSp_Damage",
  "RatioMaxDamage",
  "RatioExpectedRounds_ToDefeat",
  "RatioExpectedRemainingHP",
  "RatioMaxStrength"
)
```

Separate available data from submission data

```r
submission_dt <- train_dt[Set == "submission"]
train_dt <- train_dt[Set != "submission"]
```

Separate target from features

```r
train_result <- train_dt[, .(Set, BattleOutcome, PortionRemainingHP_1, ExpectedPortionRemainingHP_1, TrueMin
usExpectedPortionRemainingHP_1)]
submission_result <- submission_dt[, .(ExpectedPortionRemainingHP_1)]
```

Keep only features for model training

```r
keep_columns <- function(df, columns) {
  columns <- colnames(df)[!colnames(df) %in% columns]
  df[, (columns) := NULL]
}
keep_columns(train_dt, feature_columns)
keep_columns(submission_dt, feature_columns)
```

Determine which features are categorical

```r
features <- colnames(train_dt)
categorical_features <- features[lapply(train_dt, class) == "character"]
```

Encode categorical variables using LGB encoder

```r
train_dt <- lightgbm::lgb.prepare_rules(data = train_dt)
rules <- train_dt$rules
submission_dt <- lightgbm::lgb.prepare_rules(data = submission_dt, rules = rules)
train_dt <- as.matrix(train_dt$data)
submission_dt <- as.matrix(submission_dt$data)
```

Transform train data to LGB dataset

```r
dtrain <- lightgbm::lgb.Dataset(
  label = train_result$TrueMinusExpectedPortionRemainingHP_1,
  data = train_dt,
  categorical_feature = categorical_features,
  free_raw_data = FALSE
)

valids <- list(train = dtrain)
```

Parameters chosen from grid search

```r
parameter <- list(
  nthread = -1,
  boosting = "goss",
  num_iterations = 8999,
  learning_rate = 0.1,
  feature_fraction = 0.95,
  num_leaves = 30,
  seed = 12345
)
```

Train on full available data

```r
compute_time <- Sys.time()
lgb_pokemon <- lightgbm::lgb.train(
  data = dtrain,
  objective = "regression",
  valids = valids,
  params = parameter,
  eval = c("rmse", "mae")
)
```

Make predictions for training data followed by predictions for submission data

```r
train_result$PredictedPortionRemainingHP_1 <- predict(lgb_pokemon, train_dt)
train_result[, PredictedPortionRemainingHP_1 := PredictedPortionRemainingHP_1 + ExpectedPortionRemainingHP_1
]
train_result[PredictedPortionRemainingHP_1 < 0, PredictedPortionRemainingHP_1 := 0]
train_result[PredictedPortionRemainingHP_1 > 1, PredictedPortionRemainingHP_1 := 1]
submission_result$PredictedPortionRemainingHP_1 <- predict(lgb_pokemon, submission_dt)
submission_result[, PredictedPortionRemainingHP_1 := PredictedPortionRemainingHP_1 + ExpectedPortionRemainin
gHP_1]
submission_result[PredictedPortionRemainingHP_1 < 0, PredictedPortionRemainingHP_1 := 0]
submission_result[PredictedPortionRemainingHP_1 > 1, PredictedPortionRemainingHP_1 := 1]
```

Save model & predictions

```
lightgbm::lgb.save(lgb_pokemon, "data/07_model_output/lgb_pokemon.model")
data.table::fwrite(train_result, "data/07_model_output/train_prediction.csv")
data.table::fwrite(submission_result, "data/07_model_output/submission_prediction.csv")
```

Save importance matrix

```
importance_matrix <- lightgbm::lgb.importance(model = lgb_pokemon)
data.table::fwrite(importance_matrix, "data/07_model_output/feature_importance.csv")
```

Calculate and print regression metrics

```
results <- train_result[, .(RMSE = MLmetrics::RMSE(PredictedPortionRemainingHP_1, PortionRemainingHP_1),
                            MAE = MLmetrics::MAE(PredictedPortionRemainingHP_1, PortionRemainingHP_1),
                            R2 = MLmetrics::R2_Score(PredictedPortionRemainingHP_1, PortionRemainingHP_1),
                            compute_time = Sys.time() - compute_time)]
print(results)
```

Calculate and print regression metrics per class of BattleOutcome

```
results <- train_result[, .(RMSE = MLmetrics::RMSE(PredictedPortionRemainingHP_1, PortionRemainingHP_1),
                            MAE = MLmetrics::MAE(PredictedPortionRemainingHP_1, PortionRemainingHP_1),
                            R2 = MLmetrics::R2_Score(PredictedPortionRemainingHP_1, PortionRemainingHP_1)),
                            by = BattleOutcome]
print(results)
```