Predicting Quarterback 2nd Contract Value

Paul Sabin

Contents

0.1 1. Introduction

This analysis builds a **random forest regression model** to predict a quarterback's **second contract value (APY as a percentage of salary cap)**.
College and NFL performance metrics, physical traits, and draft data are used as predictors.

We use **k-fold cross-validation** to estimate performance and produce: - Variable importance plots
- Cross-validated metrics (RMSE, MAE, Bias)
- Visualization of predicted vs actual performance
- Review of largest over- and under-predictions

0.2 2. Data Loading and Preprocessing

The entire script is in this file:

```r
source("predict_qb_contract.R")
```

```
## read in cfb qb usage data from  2013
## read in cfb qb usage data from  2014
## read in cfb qb usage data from  2015
## read in cfb qb usage data from  2016
## read in cfb qb usage data from  2017
## read in cfb qb usage data from  2018
## read in cfb qb usage data from  2019
## read in cfb qb usage data from  2020
## read in cfb qb usage data from  2021
## read in cfb qb usage data from  2022
## read in cfb qb usage data from  2023
## read in cfb qb usage data from  2024
```

To confirm the structure:

```r
data_to_model %>%
  glimpse()
```

```
## Rows: 111
## Columns: 33
## $ gsis_id             <chr> "00-0030998", "00-0031064", "00-0031076", "00-00312~
## $ player              <chr> "Keith Wenning", "Tom Savage", "David Fales", "Tedd~
## $ college             <chr> "Ball State", "Pittsburgh", "San Jose State", "Loui~
## $ conference          <chr> "MAC", "ACC", "MWC", "AAC", "ACC", "MWC", "SEC", "A~
## $ birth_date          <date> 1991-02-14, 1990-04-26, 1990-10-04, 1992-11-10, 19~
## $ draft_year          <int> 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 201~
## $ draft_number        <dbl> 194, 135, 183, 32, 120, 36, 163, 214, 3, 1, 147, 98~
## $ undrafted           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ apy_cap_pct         <dbl> 0.003, 0.008, 0.004, 0.034, 0.004, 0.150, 0.001, 0.~
## $ height              <dbl> 74, 76, 73, 74, 78, 75, 73, 76, 77, 76, 75, 73, 73,~
## $ weight              <dbl> 223, 230, 219, 215, 250, 215, 207, 221, 245, 231, 2~
## $ rookie_age          <dbl> 24.54483, 26.35181, 24.90897, 22.80630, 24.16975, 2~
## $ usg_overall         <dbl> 0.543, 0.497, 0.503, 0.519, 0.544, 0.591, 0.472, 0.~
## $ usg_pass            <dbl> 0.973, 0.945, 0.907, 0.973, 0.872, 0.920, 0.896, 0.~
## $ usg_rush            <dbl> 0.067, 0.077, 0.063, 0.096, 0.255, 0.068, 0.086, 0.~
## $ usg_1st_down        <dbl> 0.504, 0.390, 0.441, 0.399, 0.464, 0.557, 0.431, 0.~
## $ usg_2nd_down        <dbl> 0.519, 0.517, 0.553, 0.568, 0.544, 0.608, 0.457, 0.~
## $ usg_3rd_down        <dbl> 0.768, 0.748, 0.595, 0.801, 0.745, 0.694, 0.697, 0.~
## $ usg_standard_downs  <dbl> 0.518, 0.414, 0.434, 0.432, 0.478, 0.536, 0.419, 0.~
## $ usg_passing_downs   <dbl> 0.609, 0.673, 0.662, 0.728, 0.672, 0.740, 0.599, 0.~
## $ passing_att         <dbl> 454, 376, 487, 382, 391, 605, 347, 504, 351, 422, 3~
## $ passing_cmp         <dbl> 296, 230, 312, 268, 224, 424, 225, 335, 239, 276, 2~
## $ passing_pct         <dbl> 0.652, 0.612, 0.641, 0.702, 0.573, 0.701, 0.648, 0.~
## $ passing_yds         <dbl> 3933, 2834, 4189, 3523, 2861, 4866, 3075, 3528, 328~
## $ passing_td          <dbl> 34, 21, 33, 28, 16, 48, 26, 21, 22, 24, 21, 12, 30,~
## $ passing_int         <dbl> 6, 9, 13, 4, 13, 7, 9, 7, 7, 17, 5, 5, 16, 6, 11, 2~
## $ passing_ypa         <dbl> 8.7, 7.5, 8.6, 9.2, 7.3, 8.0, 8.9, 7.0, 9.3, 8.4, 8~
## $ rushing_att         <dbl> 40, 72, 48, 57, 159, 40, 53, 83, 79, 49, 148, 83, 8~
## $ rushing_yds         <dbl> 45, -208, 7, 54, 295, 117, 186, 267, 179, 80, 548, ~
## $ rushing_td          <dbl> 5, 3, 2, 0, 4, 2, 7, 6, 5, 3, 8, 4, 2, 0, 5, 14, 4,~
## $ rushing_ypc         <dbl> 1.1, -2.9, 0.1, 0.9, 1.9, 2.9, 3.5, 3.2, 2.3, 1.6, ~
## $ rushing_long        <dbl> 11, 12, 16, 20, 26, 17, 57, 53, 20, 28, 29, 21, 35,~
## $ rush_pct            <dbl> 0.08097166, 0.16071429, 0.08971963, 0.12984055, 0.2~
```

0.3 3. Modeling Approach

We model `apy_cap_pct` using all variables **from conference onward** as predictors. Before modeling the log-transformation is taken then converted back before evaluation.

A **random forest** is used because it handles nonlinearities, interactions, and mixed variable types naturally.

0.4 4. Model Specification and Training

```r
# Define the recipe
rf_recipe <- recipe(apy_cap_pct ~ ., data = data_to_model) |>
  update_role(gsis_id, player, college, birth_date, draft_number, new_role = "ID") |>
  step_rm(gsis_id, player, college, birth_date) |>
  step_dummy(all_nominal_predictors()) |>
  step_zv(all_predictors()) |>
  step_normalize(all_numeric_predictors())

# Random forest model spec
rf_spec <- rand_forest(mtry = tune(), min_n = tune(), trees = 500) |>
  set_mode("regression") |>
  set_engine("ranger", importance = "permutation")

# Workflow
rf_workflow <- workflow() |>
  add_model(rf_spec) |>
  add_recipe(rf_recipe)

# Cross-validation folds
folds <- vfold_cv(data_to_model, v = 5)

# Tune model
rf_tune <- tune_grid(
  rf_workflow,
  resamples = folds,
  grid = 10,
  metrics = metric_set(rmse, mae, rsq)
)

# Select best parameters
best_params <- select_best(rf_tune, metric = "rmse")

# Finalize workflow
final_rf_workflow <- finalize_workflow(rf_workflow, best_params)

# === Fit resamples to get out-of-fold predictions ===
cv_fit <- fit_resamples(
  final_rf_workflow,
  resamples = folds,
  control = control_resamples(save_pred = TRUE)
)

# --- Collect out-of-fold predictions ---
cv_preds <- collect_predictions(cv_fit)
```

```r
# Join with original data (if needed for player names)
cv_preds <- cv_preds |>
  left_join(data_to_model |>
              dplyr::mutate(rowid = 1:n()) |>
              dplyr::select(rowid, player), by = c(".row" = "rowid"))

# === Evaluation Metrics ===
cv_metrics <- cv_preds |> metrics(truth = apy_cap_pct, estimate = .pred)
cv_bias <- cv_preds |> summarise(bias = mean(.pred - apy_cap_pct))
```

0.5 5. Model Evaluation

0.5.1 5.1 Cross-Validated Metrics

```r
cv_metrics %>%
  dplyr::select(-.estimator) |>
  bind_rows(
    tibble(.estimate = cv_bias$bias, .metric = "bias")
    ) %>%
  kable(digits = 4, caption = "Cross-Validated Performance Metrics") %>%
  kable_styling(full_width = FALSE)
```

Table 1: Cross-Validated Performance Metrics

.metric	.estimate
rmse	0.0739
rsq	0.1040
mae	0.0349
bias	-0.0273

0.5.2 5.2 Predicted vs Actual Plot

```r
cv_preds %>%
  ggplot(aes(y = apy_cap_pct, x = .pred)) +
  geom_point(alpha = 0.7) +
  geom_smooth(method = 'glm', color = 'red', se = FALSE) +
  geom_abline(slope = 1, intercept = 0, color = "black") +
  xlim(0, 0.25) + ylim(0, 0.25) +
  labs(
    title = "Predicted vs Actual Contract Value",
    x = "Predicted APY (as % of Cap)",
    y = "Actual APY (as % of Cap)"
  ) +
  theme(aspect.ratio = 1)
```

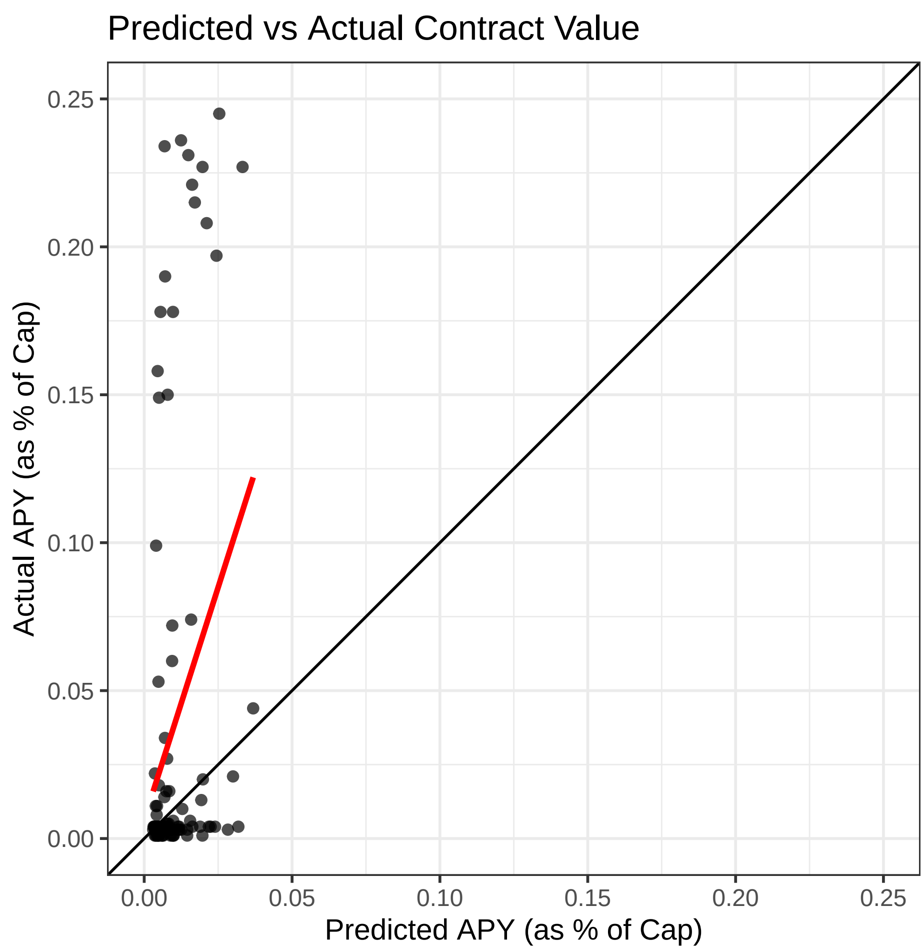

Figure 1: Predicted vs Actual apy_cap_pct

0.5.3 5.3 Biggest Over/Under Predictions

```
cv_preds %>%
  mutate(error = .pred - apy_cap_pct, abs_error = abs(error)) %>%
  arrange(desc(abs_error)) %>%
  select(player, apy_cap_pct, .pred, error) %>%
  head(10) %>%
  kable(digits = 4, caption = "Largest Model Misses (Over/Under Predictions)") %>%
  kable_styling(full_width = FALSE)
```

Table 2: Largest Model Misses (Over/Under Predictions)

player	apy_cap_pct	.pred	error
Justin Herbert	0.234	0.0069	-0.2271
Josh Allen	0.236	0.0125	-0.2235
Joe Burrow	0.245	0.0254	-0.2196
Lamar Jackson	0.231	0.0149	-0.2161
Patrick Mahomes	0.227	0.0197	-0.2073
Kyler Murray	0.221	0.0162	-0.2048
Trevor Lawrence	0.215	0.0171	-0.1979
Jalen Hurts	0.227	0.0333	-0.1937
Tua Tagovailoa	0.208	0.0211	-0.1869
Brock Purdy	0.190	0.0071	-0.1829

0.6 6. Variable Importance

```
final_rf_fit <- fit(final_rf_workflow, data = data_to_model)
final_rf_fit %>%
  extract_fit_parsnip() %>%
  vip(num_features = 20)
```

0.7 7. Summary

This report used a random forest to predict quarterback second-contract values as a proportion of the salary cap.

Key takeaways: - The model achieves weak predictive accuracy (see RMSE and MAE above). - Key predictors often include age (selection bias), passing numbers, and running efficiency. - The largest outliers are quarterbacks who ended up being among the best in the NFL because they get paid so much better than anyone else.

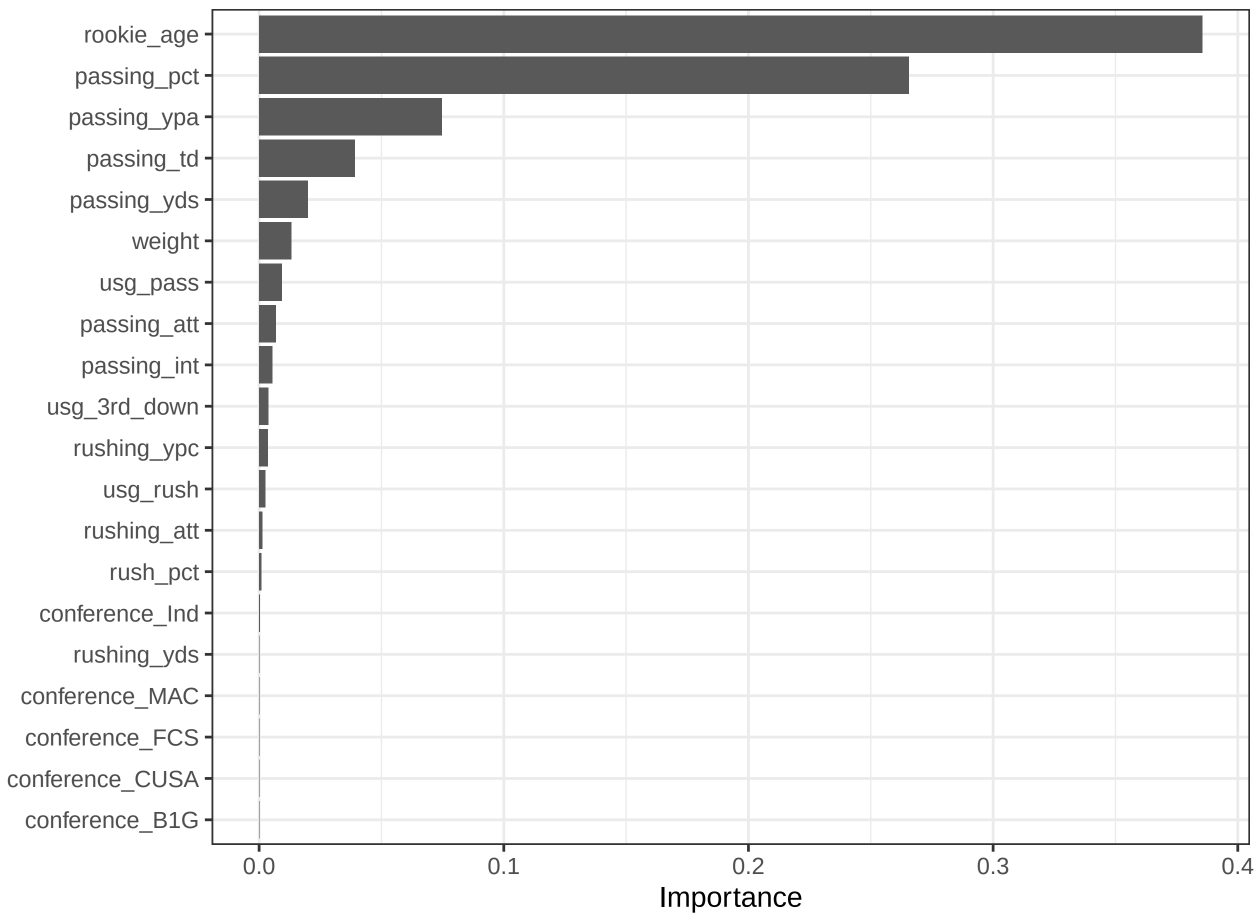

Figure 2: Top 20 Most Important Features