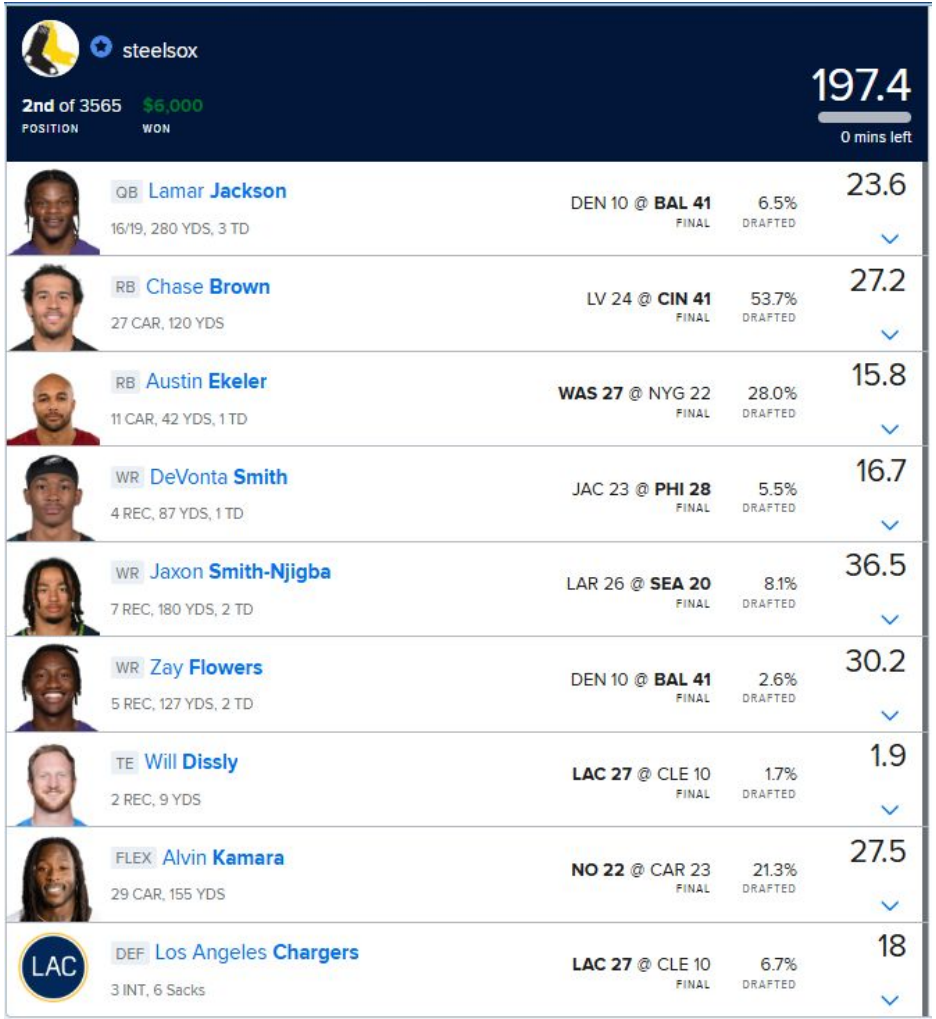
NFL Daily Fantasy Model and Lineup Generator

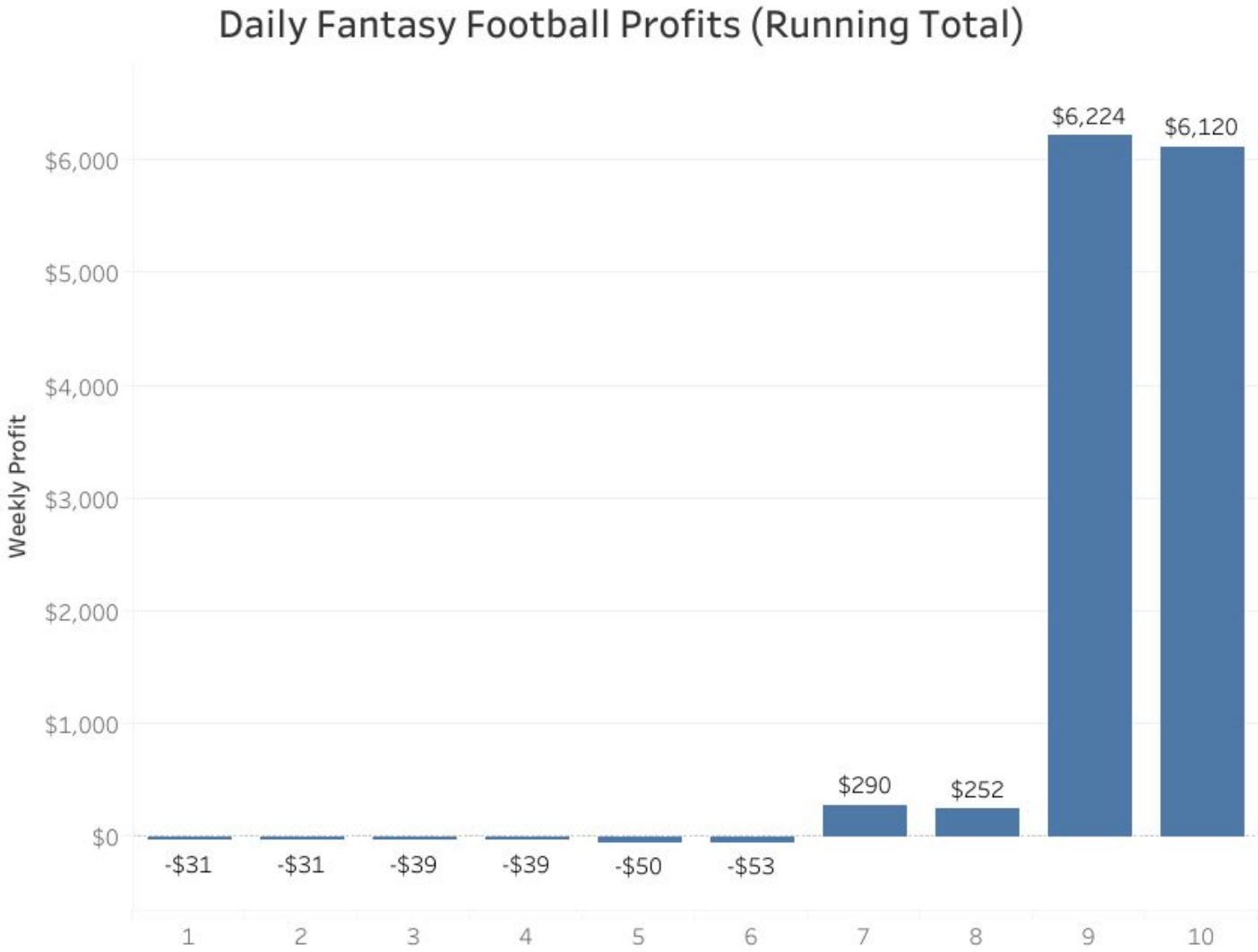
How I won $6,000



FanDuel Week 9
Entry: $33
Winnings: $6,000
2nd in field of 3,565

FanDuel Week 7
Entry: $5
Winnings: $150
1st in field of 238

FanDuel Week 7
Entry: $3
Winnings: $150
6th in field of 4,761


Daily Fantasy Football Profits (Running Total)
Weekly Profit
$6,000
$5,000
$4,000
$3,000
$2,000
$1,000
$0
-$31
-$31
-$39
-$39
-$50
-$53
$290
$252
$6,224
$6,120
1
2
3
4
5
6
7
8
9
10

What is daily fantasy football?

Season-long fantasy

- Draft a team before season
- Basically the same team all season
- Can trade players
- Can pick up players on waivers
- Season-long commitment
- Pay entry fee at beginning of season
- Top teams win money at end of season

Daily fantasy

Pick new players each time

No season-long commitment

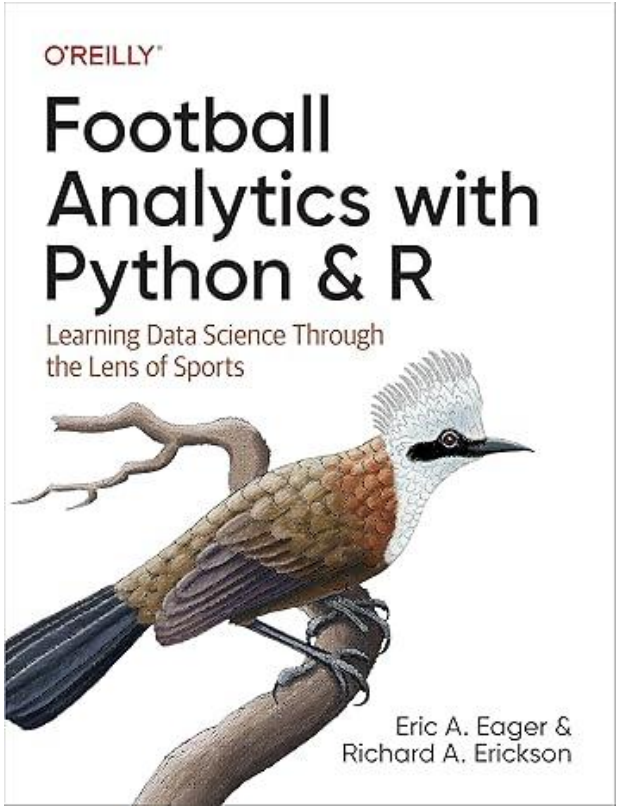
Can win or lose money every week

Salary cap

Last Presentation

- Last presentation: Major League Baseball Game Outcome Model
- September 2023
- Binary classification
- All data scraped
- Live predictions output in Jupyter Notebook
- Not regularly used on new data

Where's the data?



Plenty of data!

nflfastR

- Every play since 1999
- 372 features
- Cloned into Python
  - Nfl_data_py
  - Used for training data

3 Models

- Defense
- Quarterback
- FLEX (RB, WR, TE)

5 Algorithms

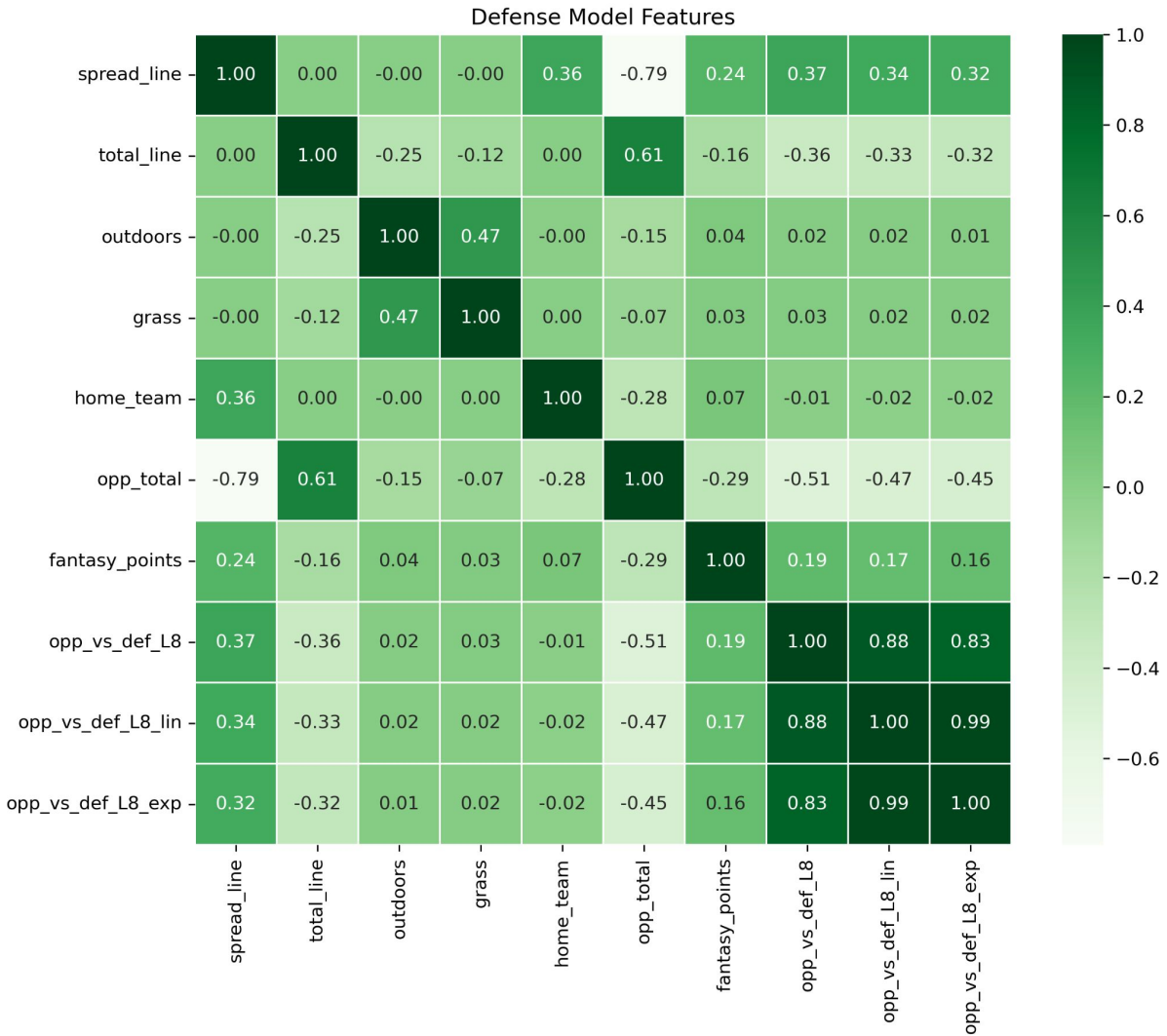
- K Nearest Neighbors
- Linear Regression
- Random Forest
- Gradient Boost
- XG Boost

Loss function:
Root Mean Squared Error

Last 8 games (L8)

- Features engineered to incorporate means over the previous 8 games that a player has played
- For team-wise features, last eight games a team has played.

Defense Model


Defense Model Features
spread_line 1.00 0.00 -0.00 -0.00 0.36 -0.79 0.24 0.37 0.34 0.32
total_line 0.00 1.00 -0.25 -0.12 0.00 0.61 -0.16 -0.36 -0.33 -0.32
outdoors -0.00 -0.25 1.00 0.47 -0.00 -0.15 0.04 0.02 0.02 0.01
grass -0.00 -0.12 0.47 1.00 0.00 -0.07 0.03 0.03 0.02 0.02
home_team 0.36 0.00 -0.00 0.00 1.00 -0.28 0.07 -0.01 -0.02 -0.02
opp_total -0.79 0.61 -0.15 -0.07 -0.28 1.00 -0.29 -0.51 -0.47 -0.45
fantasy_points 0.24 -0.16 0.04 0.03 0.07 -0.29 1.00 0.19 0.17 0.16
opp_vs_def_L8 0.37 -0.36 0.02 0.03 -0.01 -0.51 0.19 1.00 0.88 0.83
opp_vs_def_L8_lin 0.34 -0.33 0.02 0.02 -0.02 -0.47 0.17 0.88 1.00 0.99
opp_vs_def_L8_exp 0.32 -0.32 0.01 0.02 -0.02 -0.45 0.16 0.83 0.99 1.00
spread_line
total_line
outdoors
grass
home_team
opp_total
fantasy_points
opp_vs_def_L8
opp_vs_def_L8_lin
opp_vs_def_L8_exp
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

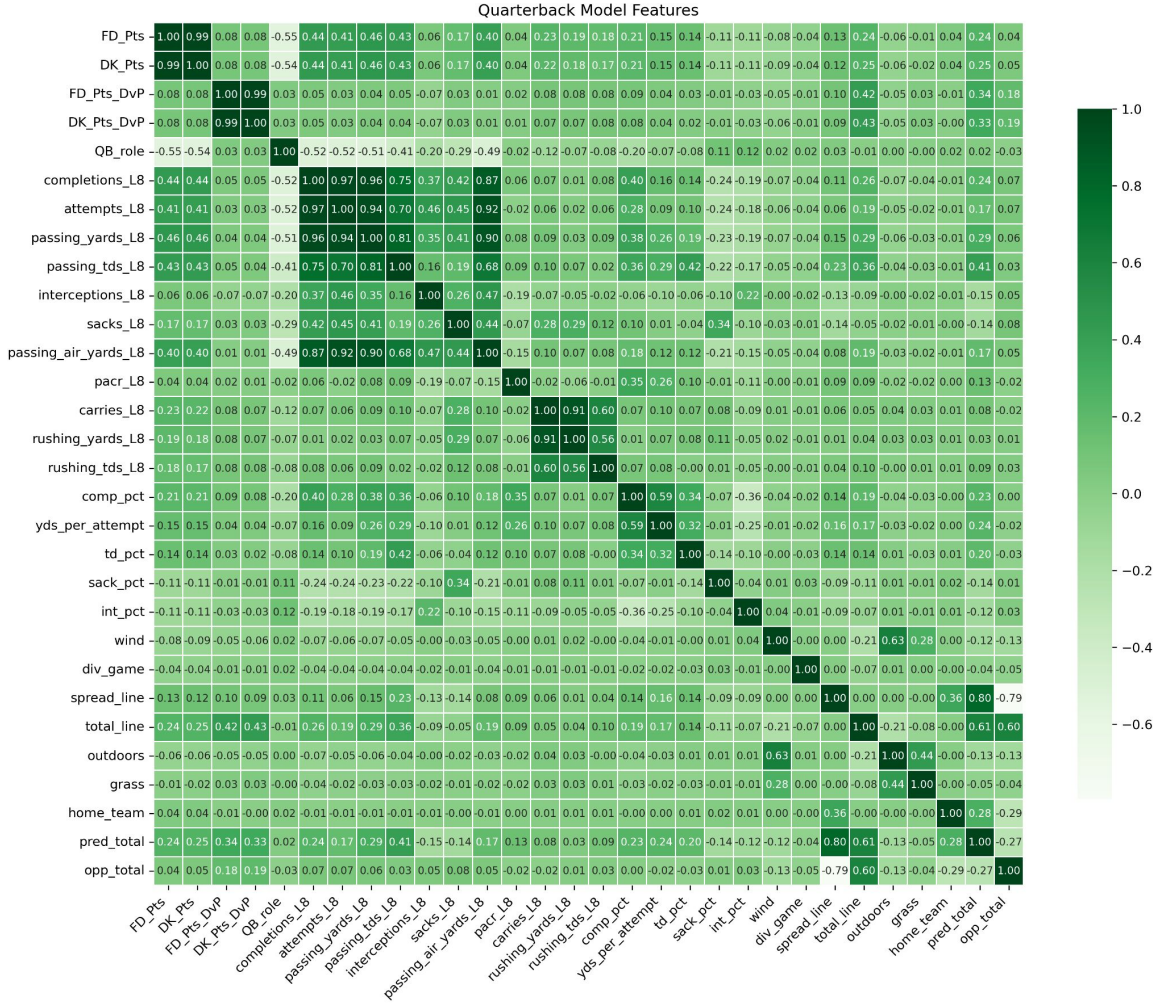
Defense Model Features

- **opp_total:** Predicted point total for opponent
- **spread_line:** Predicted margin, winners positive, losers negative
- **opp_vs_def_L8:** Mean points scored by defenses against opponent
- **opp_vs_def_L8_lin:** Linear weights
- **opp_vs_def_L8_exp:** Exponential weights
- **total_line:** Predicted point total for game, both teams
- **home_team:** Binary
- **outdoors:** Binary
- **grass:** Binary

Defense Model

- 12,064 samples
- Grid Search CV (5-fold)
- Standard Scaler
- Ridge Regression
  - Param grid: np.linspace(0.1, **100**, 50)
- Gradient Boost
  - Param grid: {'n_estimators': [100, 200, **300**], 'learning_rate': [**0.01**, 0.1, 0.2], 'max_depth': [**2**, 3, 4]}
- XG Boost
  - Param grid: {'n_estimators': [100, 200, **300**], 'learning_rate': [**0.01**, 0.1, 0.2], 'max_depth': [**2**, 3, 4]}

Quarterback Model


Quarterback Model Features
FD_Pts
DK_Pts
FD_Pts_DvP
DK_Pts_DvP
QB_role
completions_L8
attempts_L8
passing_yards_L8
passing_tds_L8
interceptions_L8
sacks_L8
passing_air_yards_L8
pacr_L8
carries_L8
rushing_yards_L8
rushing_tds_L8
comp_pct
yds_per_attempt
td_pct
sack_pct
int_pct
wind
div_game
spread_line
total_line
outdoors
grass
home_team
pred_total
opp_total
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

Quarterback Model

- QB_role feature: 1 if starting QB, 2 or 3 otherwise
    - Use play-by-play data to see which QB took the most snaps
- 5,522 data points
    - Data doesn't start until 2006. Certain data we needed not kept until that time
    - Held out odd-numbered years
- 27 features
- Standard Scaler
- Randomized Search CV
- Different models for FanDuel and DraftKings because scoring systems are different

Quarterback Model

- Random Forest
  - FanDuel: param grid: {'max_depth': [3, **6**, 9, 12, 15], 'n_estimators': [100, **300**, 500, 700], 'min_samples_split': [2, **5**, 10], 'min_samples_leaf': [1, 2, **4**]}
  - DraftKings: param grid: {'max_depth': [3, **6**, 9, 12], 'n_estimators': [100, 300, **500**], 'min_samples_split': [**2**, 5, 10], 'min_samples_leaf': [1, 2, **4**]}
- Gradient Boost
  - FanDuel: param grid: {'n_estimators': [**100**, 200, 300], 'learning_rate': [0.01, **0.05**, 0.1, 0.2], 'max_depth': [**2**, 3, 4]}
  - DraftKings: param grid: { 'n_estimators': [**100**, 300, 500], 'learning_rate': [0.001, 0.01, 0.05, **0.1**, 0.2, 0.3], 'max_depth': [**2**, 4, 6]}

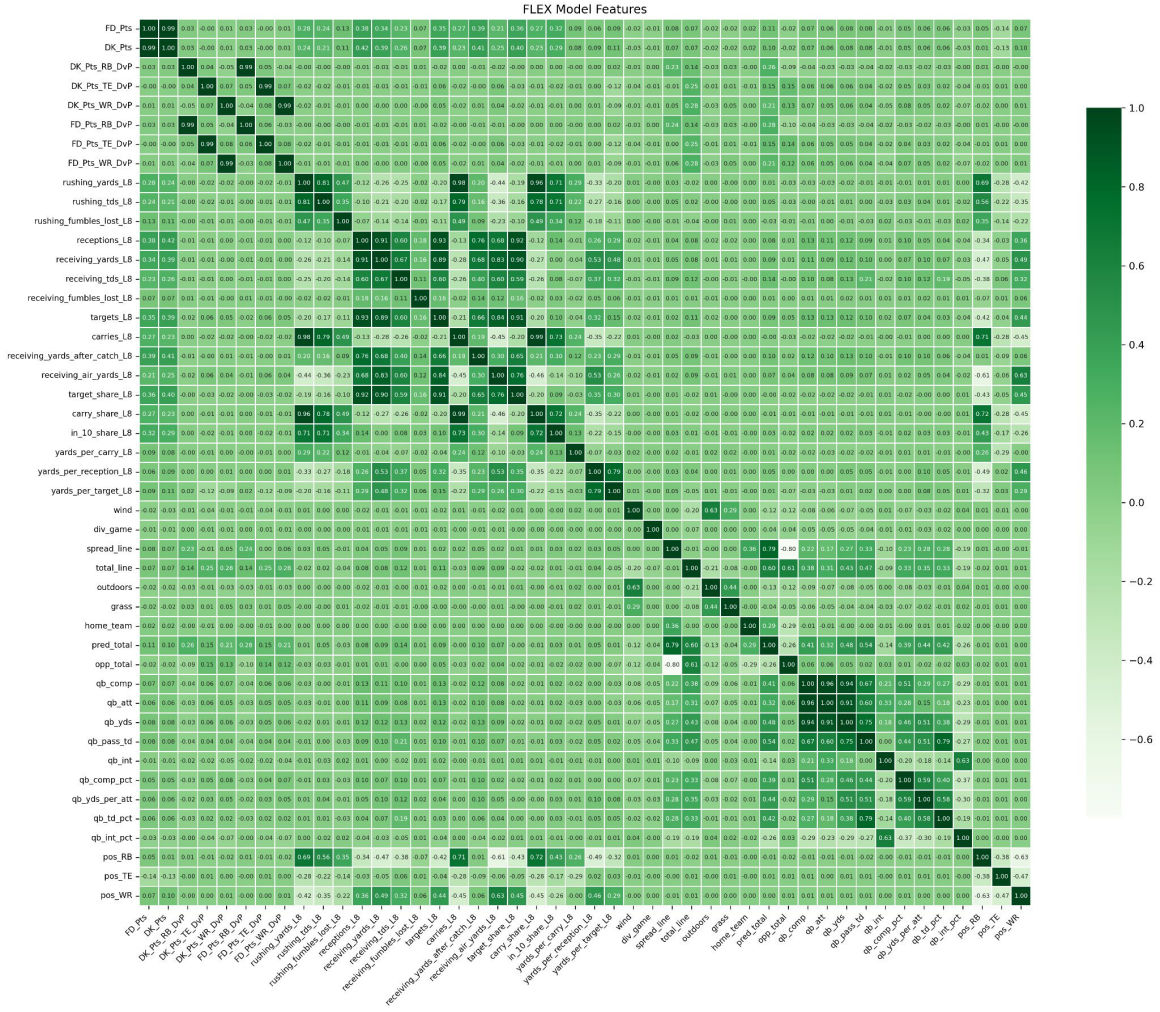
Quarterback Model

- XG Boost

  FanDuel: param grid {'n_estimators': [**100**, 200, 300], 'learning_rate': [0.01, **0.05**, 0.1, 0.2, 0.3], 'max_depth': [**2**, 4, 6], 'subsample': [**0.6**, 0.8, 1.0], 'colsample_bytree': [0.6, 0.8, **1.0**], 'gamma': [0, 0.1, **0.3**], 'reg_alpha': [0, **0.01**, 0.1], 'reg_lambda': [1, **1.5**, 2]}

  DraftKings: param grid {'n_estimators': [**100**, 200, 300], 'learning_rate': [0.01, **0.05**, 0.1, 0.2, 0.3], 'max_depth': [**2**, 4, 6], 'subsample': [0.6, **0.8**, 1.0], 'colsample_bytree': [0.6, 0.8, **1.0**], 'gamma': [0, **0.1**, 0.3], 'reg_alpha': [0, **0.01**, 0.1], 'reg_lambda': [**1**, 1.5, 2]}

FLEX Model


FLEX Model Features
FD_Pts
DK_Pts
DK_Pts_RB_DvP
DK_Pts_TE_DvP
DK_Pts_WR_DvP
FD_Pts_RB_DvP
FD_Pts_TE_DvP
FD_Pts_WR_DvP
rushing_yards_L8
rushing_tds_L8
rushing_fumbles_lost_L8
receptions_L8
receiving_yards_L8
receiving_tds_L8
receiving_fumbles_lost_L8
targets_L8
carries_L8
receiving_yards_after_catch_L8
receiving_air_yards_L8
target_share_L8
carry_share_L8
in_10_share_L8
yards_per_carry_L8
yards_per_reception_L8
yards_per_target_L8
wind
div_game
spread_line
total_line
outdoors
grass
home_team
pred_total
opp_total
qb_comp
qb_att
qb_yds
qb_pass_td
qb_int
qb_comp_pct
qb_yds_per_att
qb_td_pct
qb_int_pct
pos_RB
pos_TE
pos_WR
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

FLEX Model

- 40,039 data points (even-numbered years)
- 44 features
- Standard Scaler
- Randomized Search CV
- Different models for FanDuel and DraftKings because scoring systems are different

FLEX Model

- Random Forest
  - FanDuel: param grid {'max_depth': [3, 6, **9**, 12], 'n_estimators': [100, 250, **400**], 'min_samples_split': [2, 5, **10**], 'min_samples_leaf': [1, 2, **4**]}
  - DraftKings: param grid {'max_depth': [6, **9**, 12], 'n_estimators': [**250**, 400], 'min_samples_split': [2, 5, **10**], 'min_samples_leaf': [1, 2, **4**]}

FLEX Model

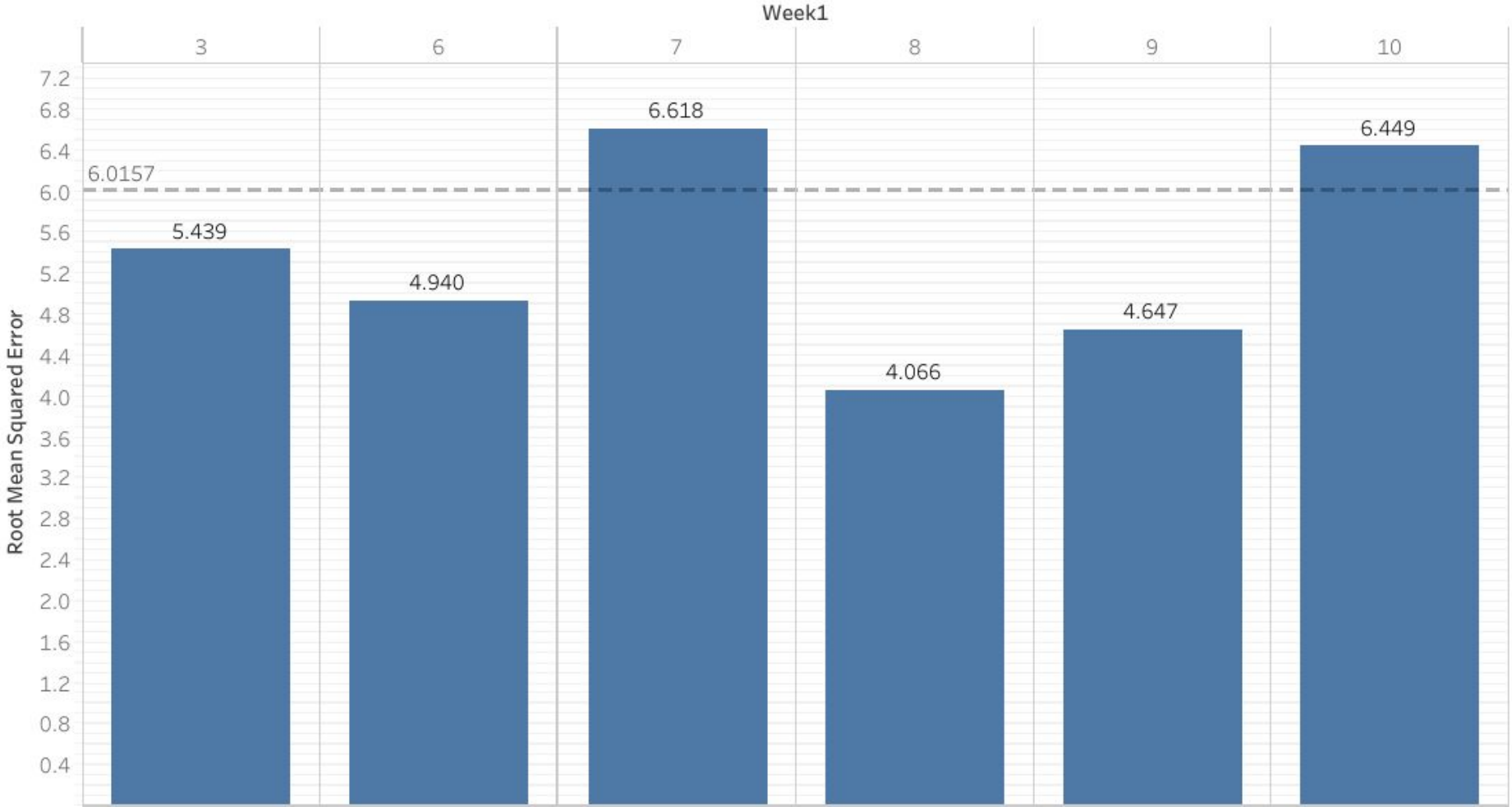
- Gradient Boost
  - FanDuel: param grid {'n_estimators': [100, 300, **500**], 'learning_rate': [0.001, **0.01**, 0.05, 0.1, 0.2, 0.3], 'max_depth': [**4**, 6, 8]}
  - DraftKings: param grad {'n_estimators': [100, 300, **500**], 'learning_rate': [0.001, **0.01**, 0.05, 0.1, 0.2, 0.3], 'max_depth': [**4**, 6, 8]}

FLEX Model

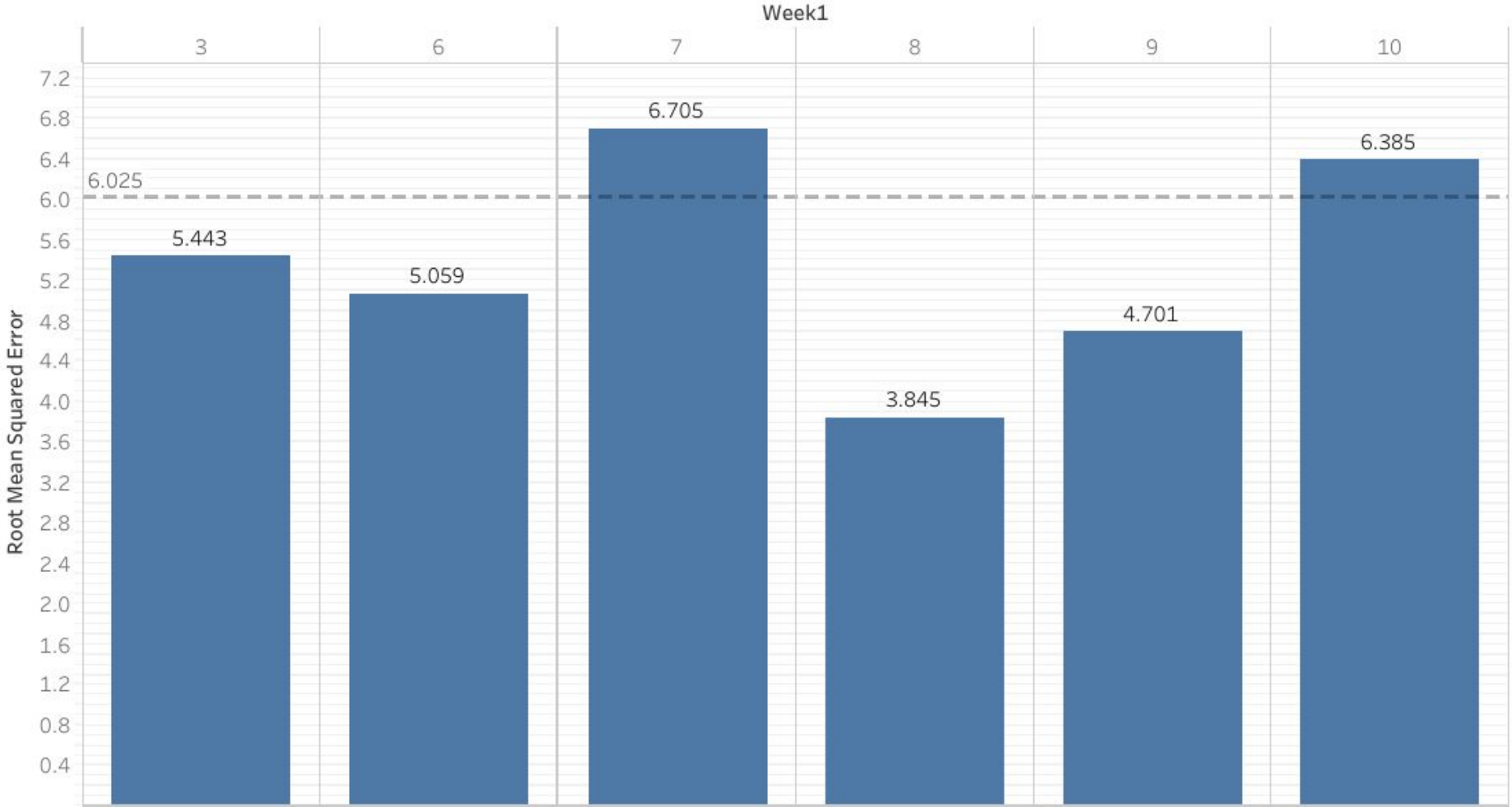
- XG Boost
  - FanDuel: param grid { 'n_estimators': [300, **500**, 700], 'learning_rate': [**0.01**, 0.05, 0.1, 0.2, 0.3], 'max_depth': [2, **4**, 6], 'subsample': [0.6, **0.8**, 1.0], 'colsample_bytree': [0.6, 0.8, **1.0**], 'gamma': [**0**, 0.1, 0.3], 'reg_alpha': [0, 0.01, **0.1**], 'reg_lambda': [1, **1.5**, 2]}
  - DraftKings: param grid {'n_estimators': [300, **500**, 700], 'learning_rate': [**0.01**, 0.05, 0.1, 0.2, 0.3],'max_depth': [**4**, 6, 8], 'subsample': [0.6, **0.8**, 1.0], 'colsample_bytree': [**0.6**, 0.8, 1.0], 'gamma': [0, **0.1**, 0.3], 'reg_alpha': [**0**, 0.01, 0.1], 'reg_lambda': [1, **1.5**, 2]}

All scalers and models stored in pickle files.

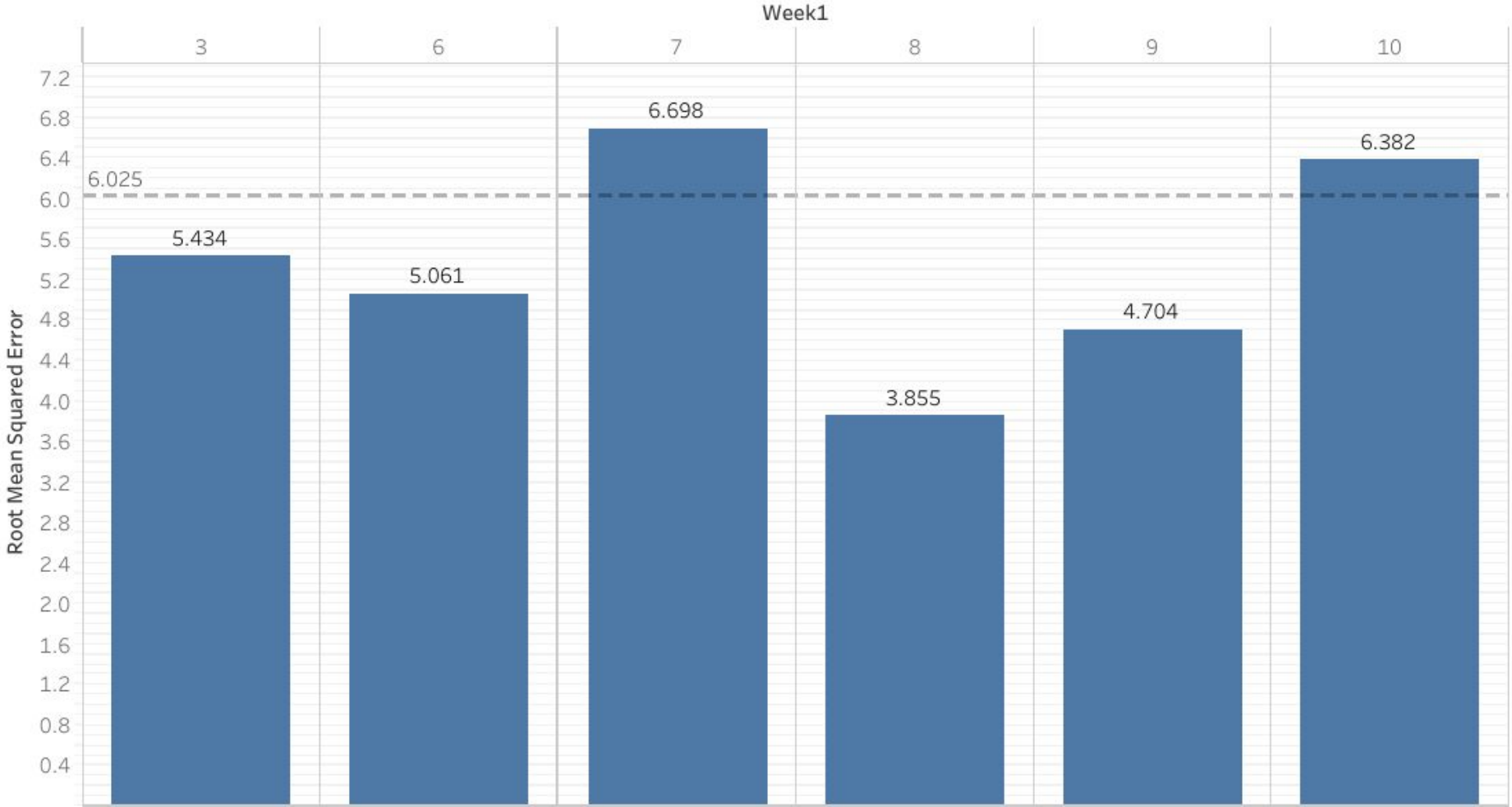
FanDuel Defense Ridge Model



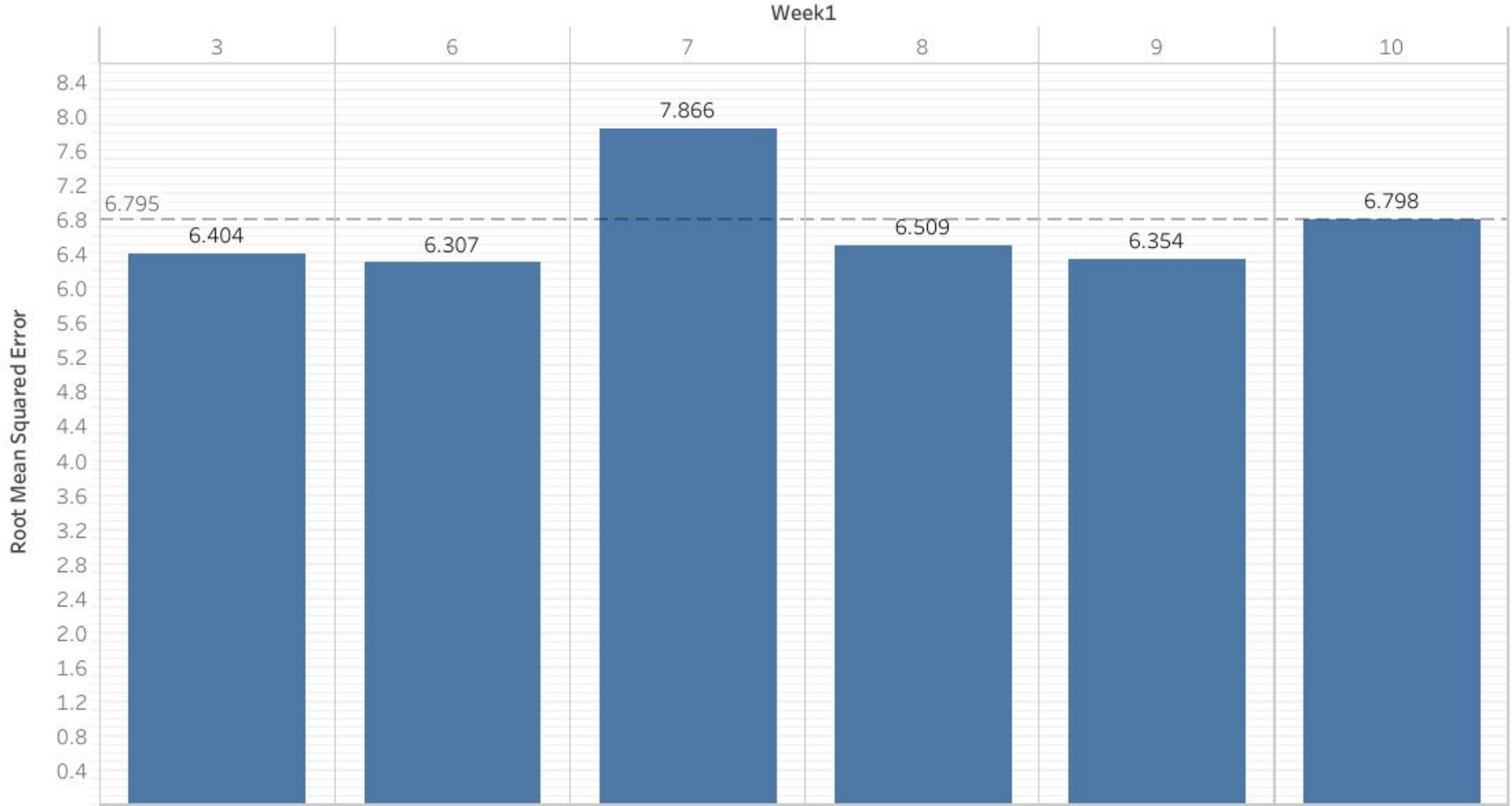
FanDuel Defense Gradient Boost Model



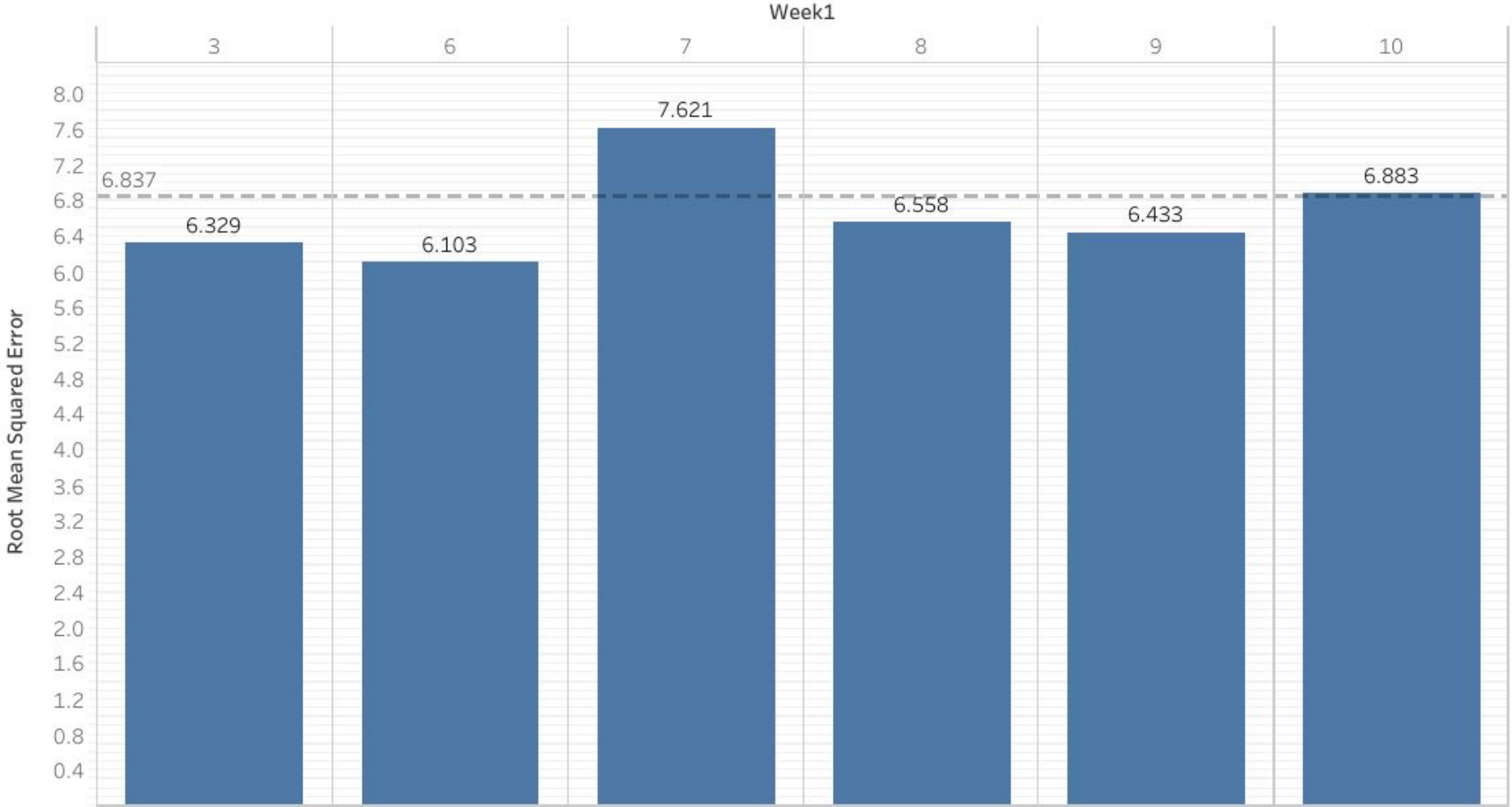
FanDuel Defense XG Boost Model



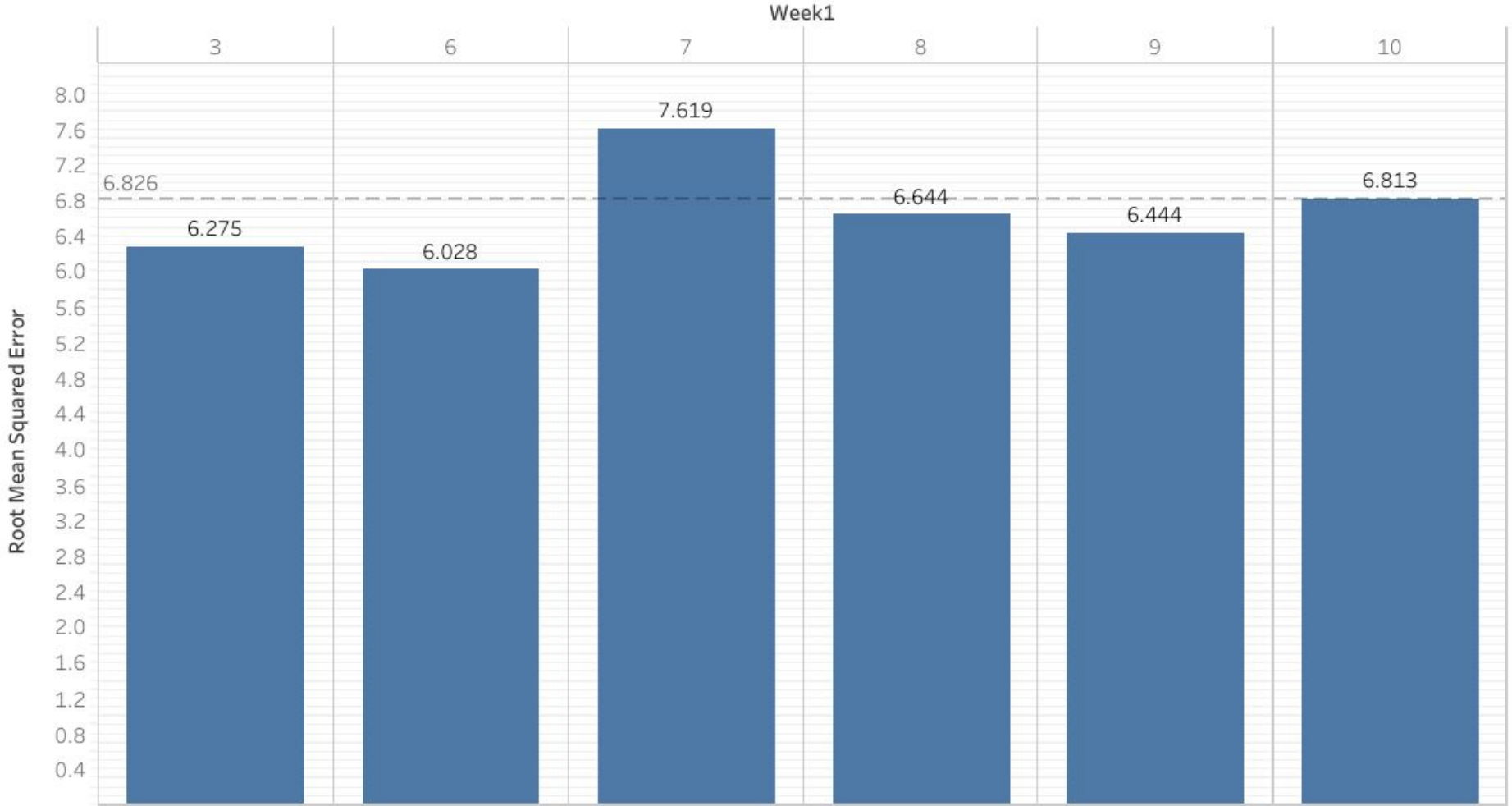
FanDuel QB Random Forest Model

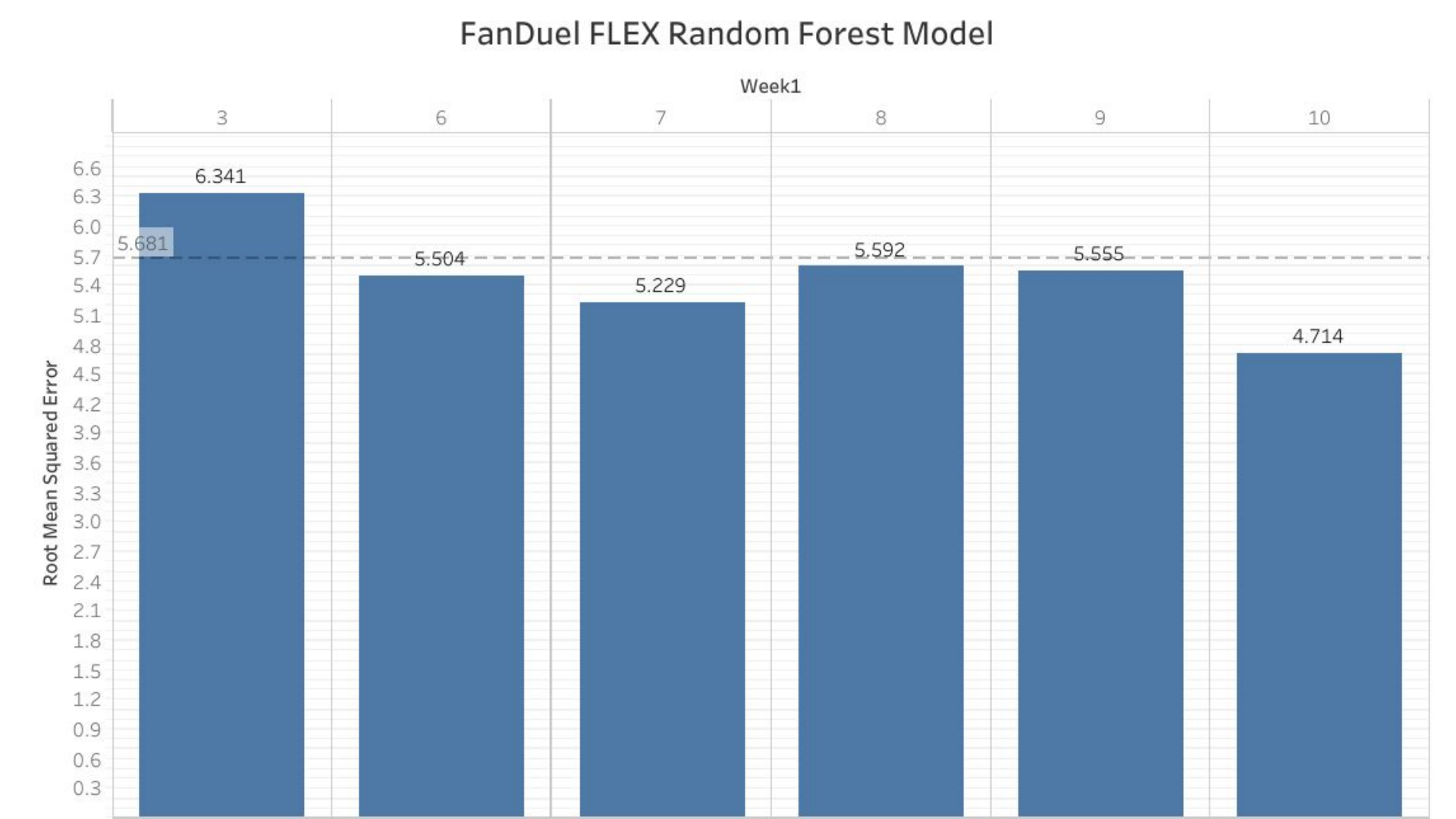


FanDuel QB Gradient Boost Model

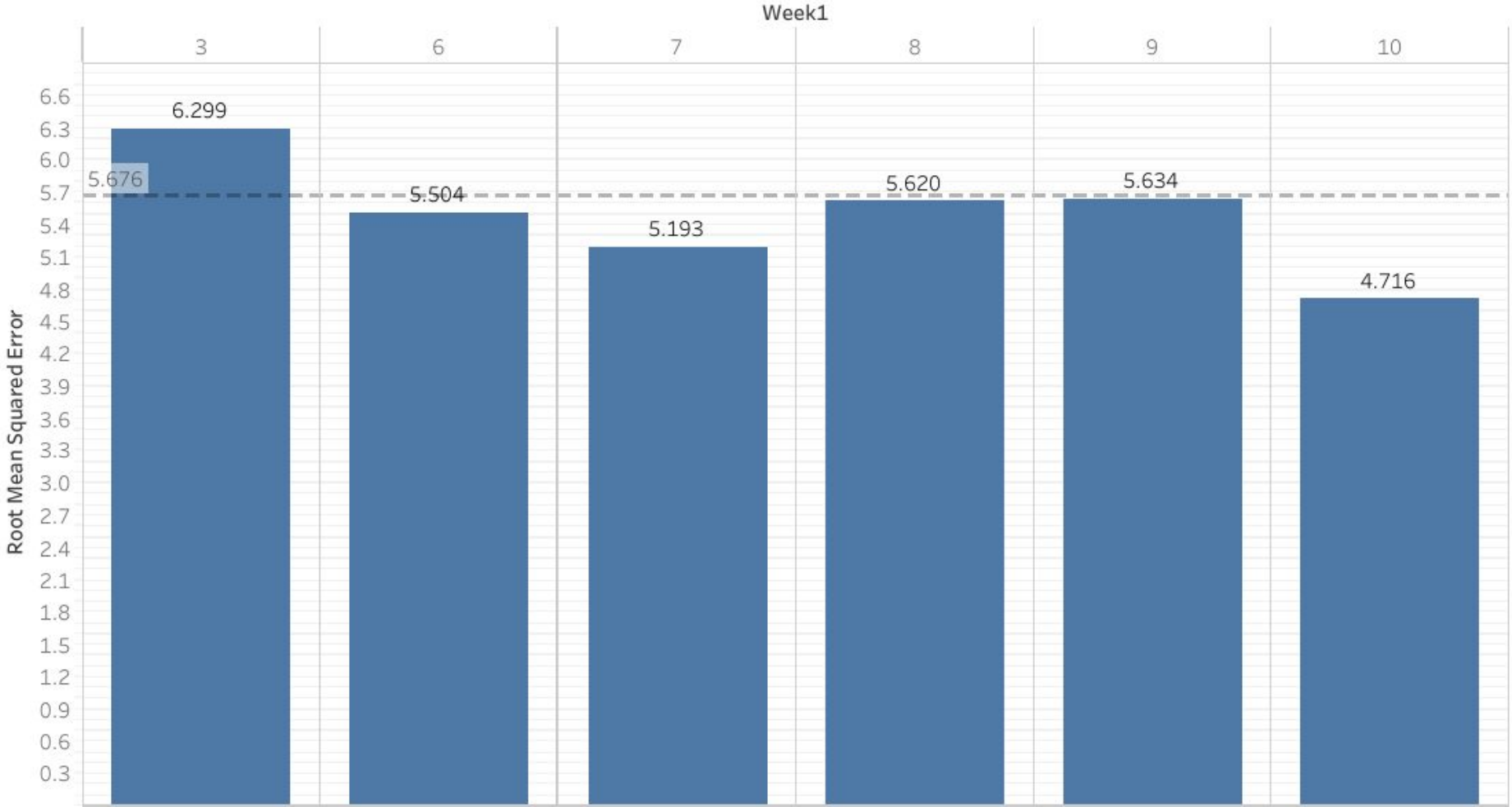


FanDuel QB XG Boost Model

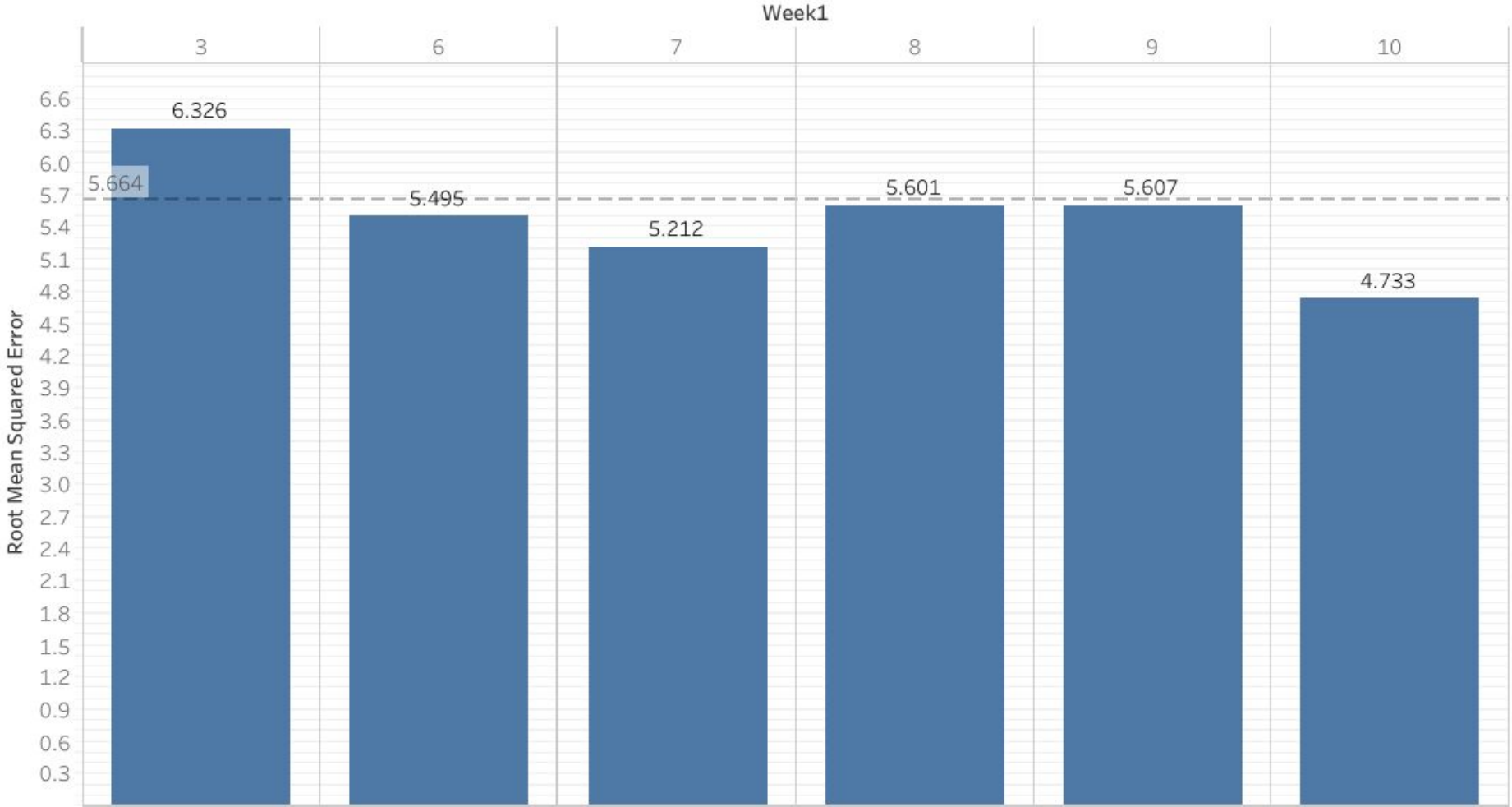



FanDuel FLEX Random Forest Model
Week1
3
6
7
8
9
10
Root Mean Squared Error
6.6
6.3
6.0
5.7
5.4
5.1
4.8
4.5
4.2
3.9
3.6
3.3
3.0
2.7
2.4
2.1
1.8
1.5
1.2
0.9
0.6
0.3
5.681
6.341
5.504
5.229
5.592
5.555
4.714

FanDuel FLEX Gradient Boost Model



FanDuel FLEX XG Boost Model



Live Predictions

Process for live predictions

- A little bit of R (R Studio)
  - Updates in nflfastR more reliable than in nfl_data_py
- Odds
  - The Odds API
- Wind
  - Scrape RotoGrinders (Beautiful Soup)

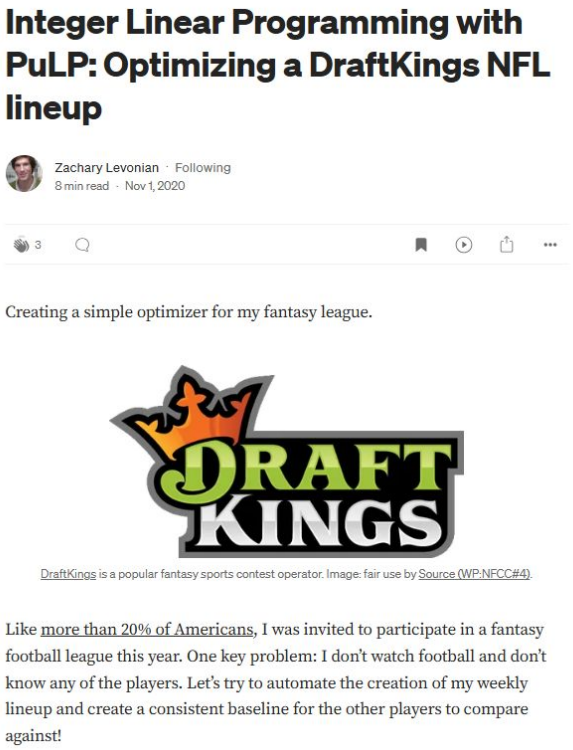
Process for live predictions

- 11 Jupyter Notebooks run sequentially
  - Download player lists from FD and DK
  - Assemble datasets for live predictions
  - One notebook contains dict of starting QBs
  - Check summary statistics, decide to drop or fill missing values
  - Predictions
  - Model Evaluation (just once)
- 3 Key Points During Week
  - Wednesday: Initial predictions and lineups
  - Friday night: Predictions adjusted with final injury status (questionable, doubtful, out)
  - Sunday morning (1-2 hours before kickoff): Final decisions made on which players are active and inactive. Predictions and lineups adjusted if necessary

Target Engineering?

- How are missing values handled?
  - In most cases, imputed with mean.
  - Min-priced players unlikely to play have data similar to average players
  - Lineup generator will tend to roster useless players
- Drop min-priced players ($4K FanDuel)
  - Week 11 FanDuel predictions dataset goes from 796 samples to 391
- **Drop injured players from predictions dataset**
  - Adjust predictions of other players on team based on increased opportunities

Lineup Generation



Integer Linear Programming with PuLP: Optimizing a DraftKings NFL lineup

Zachary Levonian · Following
8 min read · Nov 1, 2020

3

Creating a simple optimizer for my fantasy league.

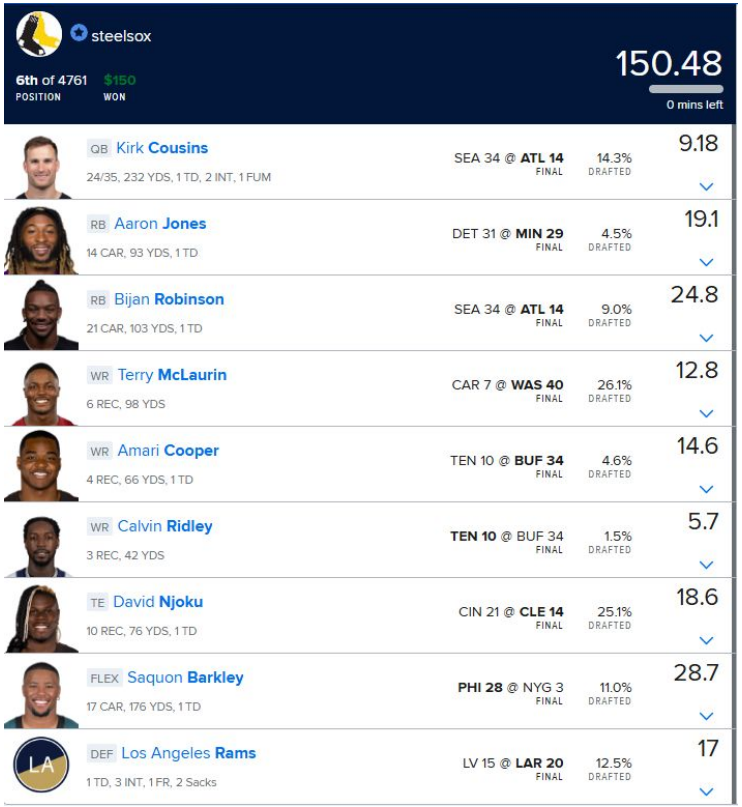
DraftKings is a popular fantasy sports contest operator. Image: fair use by Source (WP:NFCC#4).

Like more than 20% of Americans, I was invited to participate in a fantasy football league this year. One key problem: I don't watch football and don't know any of the players. Let's try to automate the creation of my weekly lineup and create a consistent baseline for the other players to compare against!

Find maximum predicted points while staying within salary cap and position restrictions.

https://zwlevonian.medium.com/integer-linear-programming-with-pulp-optimizing-a-draftkings-nfl-lineup-5e7524dd42d3

Lineup Generation



Future Work

- More command line
- More SQL (sqlite)
  - Map out database (ended up with a lot of CSVs)
- Incorporate entire project in Streamlit
- Set up Streamlit app for bulk lineup generation
- Revenge Game binary variable
- Variable to indicate how many games a player has missed
  - Impact on other predictions would be different
  - Might reduce inclusion of fringe players in lineups
- Snap count variable
- Account for recently traded players as well as injuries
- Deep Learning

Thank you!

Mike Batista

- GitHub repo: https://github.com/mbmontana785/NFL_Daily_Fantasy_Model
- LinkedIn: https://www.linkedin.com/in/mbmontana/
- Email: steelsox79@gmail.com