SUPER BOWL LX

Who gets into the Playoffs?

Data Collection & Processing

- Pro-Football Reference: The premier website for football statistics and historical numbers
    - We manually scraped the website for a host of offensive, defensive, and miscellaneous statistics, dating back to the year 2000, giving us 25 years of data to use (49 statistics per year)

Year	Team	Points Scored/Game	...	Int%	QBHits/Game	TFL/Game
2024	Arizona Cardinals	23.53	...	1.7	4.47	5.12
2024	Atlanta Falcons	22.88	...	2.1	4.35	4.65
2024	Baltimore Ravens	30.47	...	1.9	7.18	4.82
2024	Buffalo Bills	30.88	...	2.8	5.53	5.47
2024	Carolina Panthers	20.06	...	1.7	3.76	3.82

- After this process, every total statistic has to be reduced to a per/game number because of the shift in season-length (NFL turned to a 17 game schedule in the 2021 season)

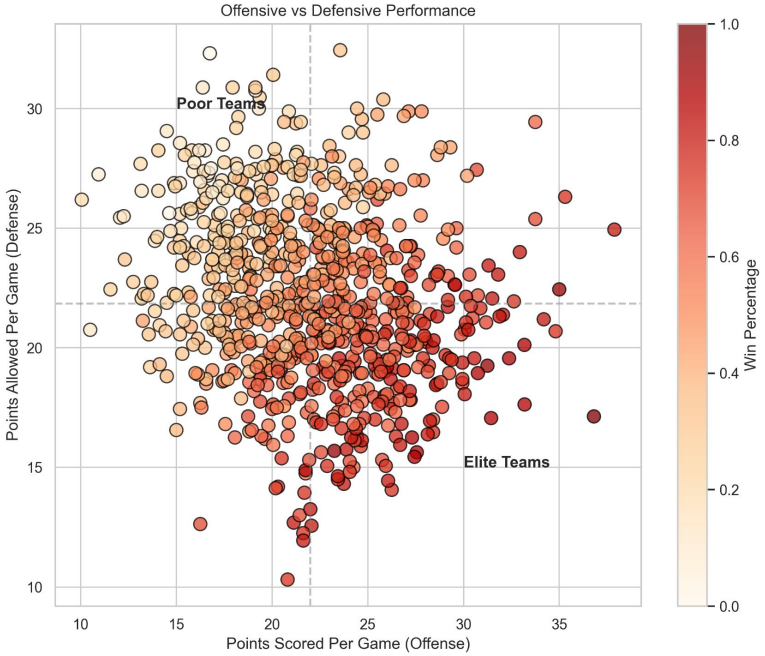
Before

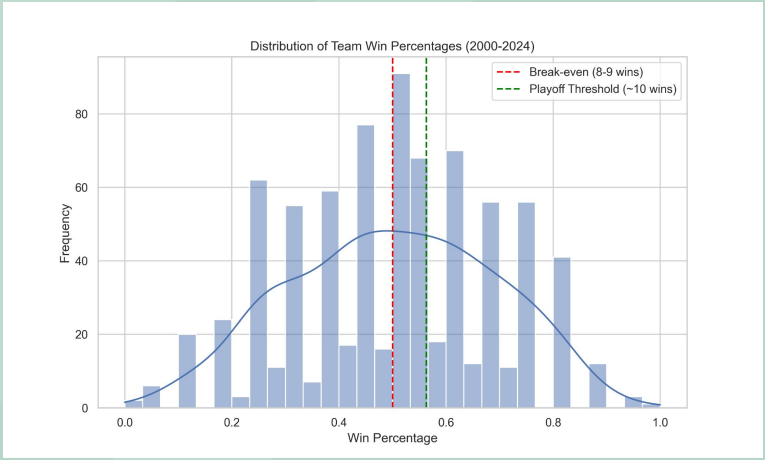
TO	1stD	Cmp	Pass Att	Pass Yds	Pass TD	Int
15	410	399	551	4474	39	12
8	360	329	520	3875	30	6
11	393	318	477	4035	41	4

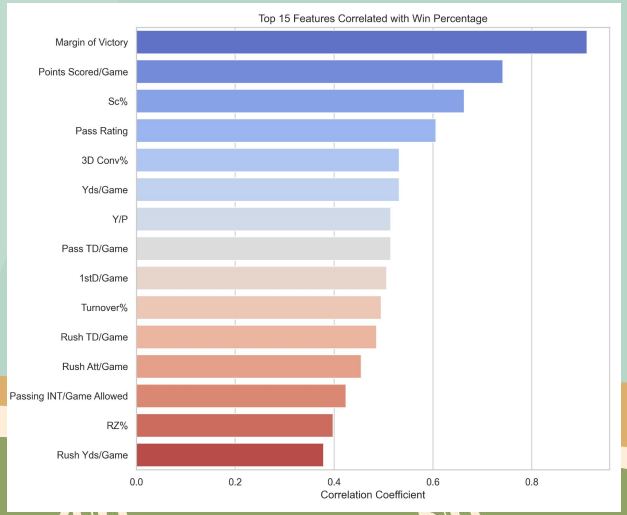
After

TO/Game	1stD/Game	Cmp/Game	Pass Att/Game	Pass Yds/Game	Pass TD/Game	Int/Game
0.88	24.12	23.47	32.41	263.18	2.29	0.71
0.47	21.18	19.35	30.59	227.94	1.76	0.35
0.65	23.12	18.71	28.06	237.35	2.41	0.24

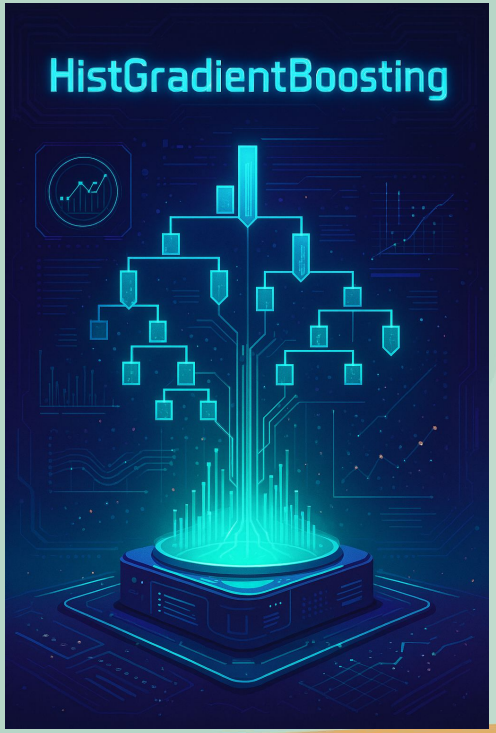
EDA Insights







What Model was used for this Stage?



- Builds an ensemble of decision trees sequentially (boosting)
- Each new tree tries to correct errors (residuals) from previous trees
- Uses gradient descent to minimize a specified loss function
- Histograms bin continuous features to speed up split finding and reduce memory usage
- Chooses splits that most reduce the loss, not necessarily balancing tree depth

Benefits of HistGradientBoostingRegressor

01

SPEED & SCALABILITY

- Trains extremely fast using histogram binning, despite large datasets (Quicker than random forest)
- Ideal for iterating quickly during feature engineering and cross-validation.

02

PATTERN RECOGNITION

- Captures complex, nonlinear relationships between NFL metrics without manual feature crossing (Linear regression cannot do this)

03

HANDLES MANY FEATURES

- Designed for wide, numeric tabular datasets (Random forest too slow)
- No need for scaling or heavy preprocessing.

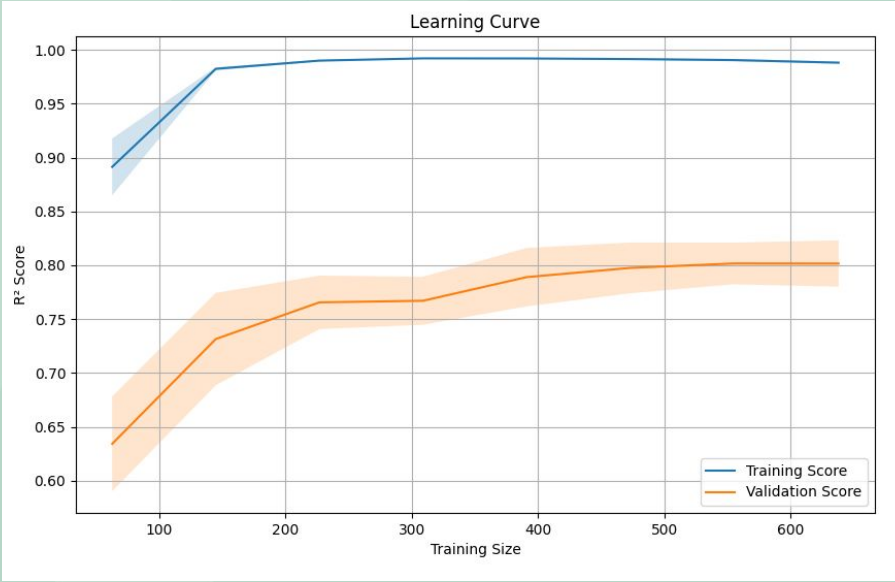
04

BUILT-IN REGULARIZATION

- Strong regularization prevents overfitting automatically, unlike linear regression:
  - Early stopping
  - Shrinkage

Hyperparameter Optimization (RandomizedSearchCV)

Hyperparameter	Min	Max	Notes	Best
learning_rate	0.01	0.11	Continuous uniform distribution	0.02
max_iter	500	2,000	Discrete integer range	591
max_depth	3	None	Discrete options: [None, 3, 5, 7]	7
min_samples_leaf	10	40	Discrete integer range	24
l2_regularization	0	1	Continuous uniform distribution	0.305



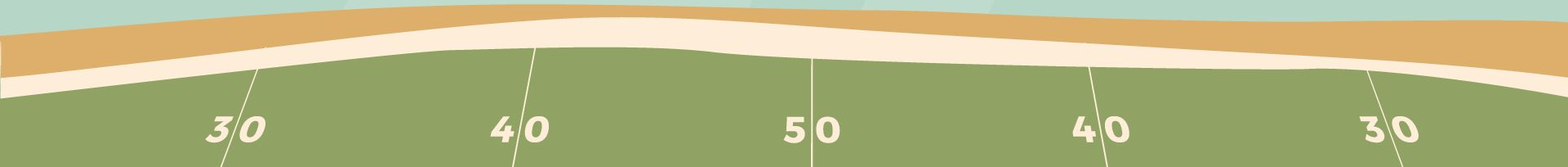
Metrics

- R^2: 0.9879 (Near-perfect fit on training data)
- RMSE: 0.0211 (Average prediction error of just 2.1%)
- MAE: 0.0163 (Typical error within 1.6 percentage points)

Model Training Pt.3

Obtained Mean and Standard Deviation the Model

- Cv_r2_mean (Average): ***0.083***
  - *Model explains 80.3% of the variance across seasons*
  - *Captures the meaningful relationships across offense, defense, and misc stats that account for win%.*
- Cv_r2_std (Standard Deviation): ***0.023***
  - *Reflects how stable the R^2 scores were across the cross-validation*
  - *Model performs similarly regardless of which subset of data it trains on (indicating good generalization)*



Model Pitfalls

Missing Information (19.7% unexplained)

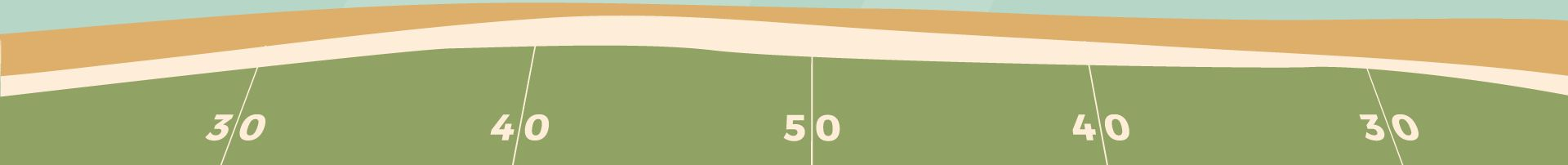
- Player injuries and roster changes mid-season
- Coaching adjustments and strategic shifts
- Division tiebreakers
- Luck factor (Negative point differential but winning record)

Assumptions

- Treats all data equally, but NFL rules/strategy evolve
- High multicollinearity between offensive & defensive stats may create unstable predictions
- Predicts win% but doesn't simulate actual division standings (head-to-head tiebreakers/seeding)

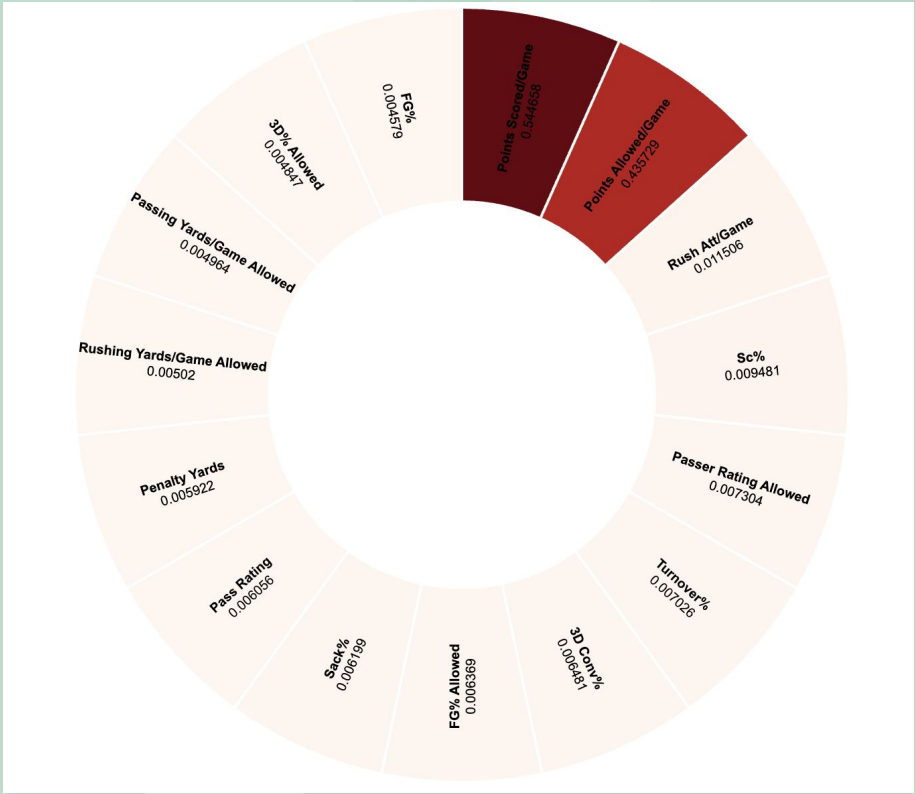
Next Steps

- Use forward-chaining CV (train on past → test on future) for realistic accuracy
- Add uncertainty: "Bills have 85% playoff probability" vs. "Bills make playoffs"
- Weight recent seasons more heavily to capture modern NFL trends



Results

Points Scored/Game	0.544658
Points Allowed/Game	0.435729
Rush Att/Game	0.011506
Sc%	0.009481
Passer Rating Allowed	0.007304
Turnover%	0.007026
3D Conv%	0.006481
FG% Allowed	0.006369
Sack%	0.006199
Pass Rating	0.006056
Penalty Yards	0.005922
Rushing Yards/Game Allowed	0.005020
Passing Yards/Game Allowed	0.004964
3D% Allowed	0.004847
FG%	0.004579
RZPct% Allowed	0.004565
Penalties - Yards	0.004474
Pass Deflections/Game	0.004349
RZ%	0.004276
TO%	0.003771

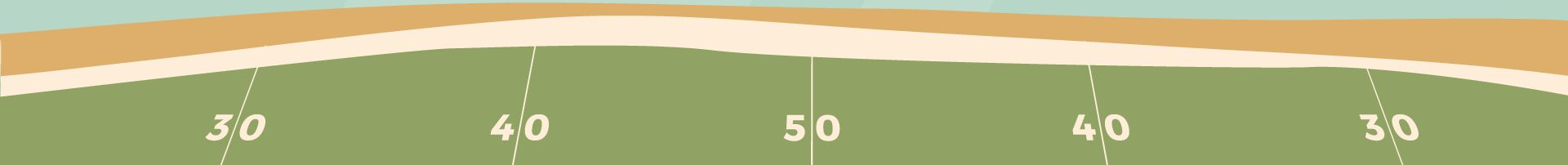


Playoff Standings

Who makes the Superbowl?

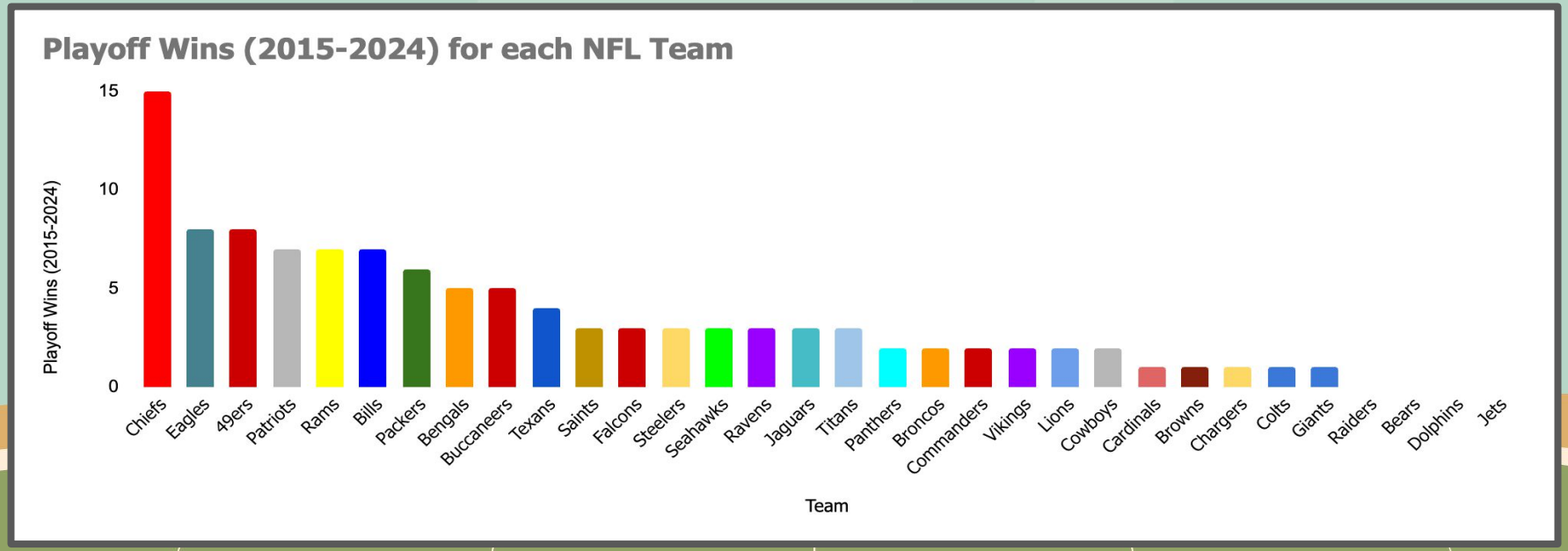
Simulating The NFL Playoffs

- After simulating Regular Season results, we extend pipeline to the Playoff Bracket
- Variables to consider:
  - Single Elimination ⟶ High Variance
  - Teams can play maximum of 3 Playoff Games ⟶ Small Sample Size
  - Upsets (Lower seeded teams prevailing) occur often ⟶ Which statistics influence expected Playoff performance?
    - Regular Season W/L% is not enough



Gathering Further Data

- *Individual* playoff games = **high variance**, but *distribution of playoff* ***wins*** = **low variance** (most teams get 0; max is 3)
- Recent decade best reflects modern playoff structure.
- Playoff model uses 10 years while regular season model uses 25 years because larger sample size improves stability



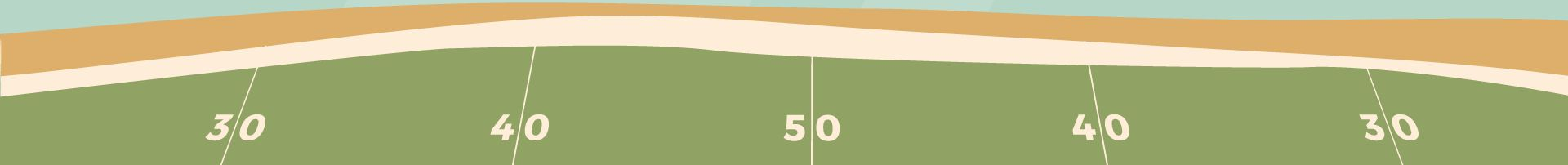
How Model was Trained

- *Train a GLM (Tweedie) model using regular-season stats to predict playoff wins.*
- *Merges Regular Season and Playoff Stats into one table*
- *Model learns how offense, defense, and misc efficiency relate to postseason success*

```
X, y, features = prepare_features_and_target(df, numeric_cols, target_col)

print(f"Training playoff model (target={target_col}, power={parsed.power}, alpha={parsed.alpha})")

#Initialize Tweedie GLM Model
model = build_model_tweedie(power=parsed.power, alpha=parsed.alpha)

#Train the model and compute cross-validated R² scores to evaluate predictive ability
model, cv_scores = train_and_evaluate(X, y, power=parsed.power, alpha=parsed.alpha)

joblib.dump(model, parsed.out)
joblib.dump(features, parsed.features_out)
```

- Apply the trained model to 2025 team stats to estimate playoff strength.
- Feeds each team's 2025 efficiency profile into the trained GLM to generate predictions.

```
X = snapshot[features]
expected = model.predict(X)

out = snapshot[["Team"]].copy()
if "Year" in snapshot.columns:
    out["Year"] = snapshot["Year"]
else:
    out["Year"] = TARGET_YEAR
out["expected_playoff_wins"] = expected
```

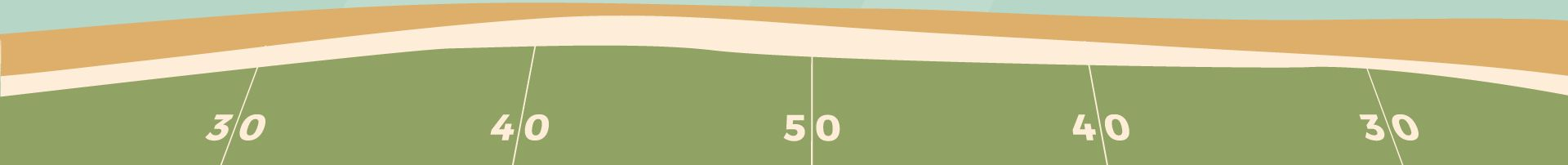


Interpreting Playoff Model Results

Cross Validation: repeatedly splits the playoff dataset into training and testing groups to measure how well the model generalizes. These R^2 values represent how accurately the model predicted unseen seasons across these splits.

Results:

- Two Values Obtained: Mean and Standard Deviation of Testing/Training the Model Many Times
  - Cv_r2_mean (Average): ***0.061***
    - *Model finds predictive signal (value above 0) despite noisy playoff outcomes*
    - *Playoff wins are highly skewed (most teams finish with 0 wins)*
  - Cv_r2_std (Standard Deviation): ***0.213***
    - *Playoff outcomes fluctuate dramatically year-to-year*
    - *Reflects inherent unpredictability of postseason football*



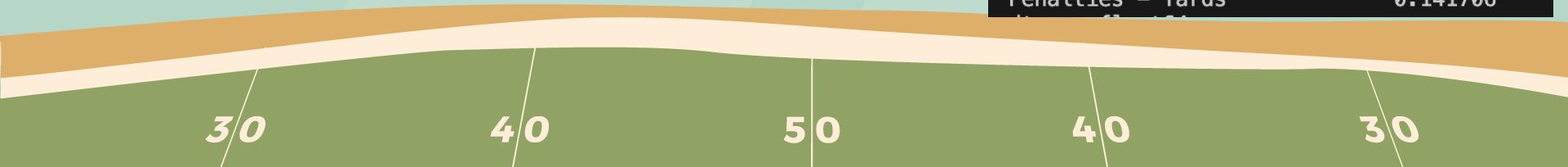
Which Stats Actually Predict Playoff Success?

The model provides the correlation between the statistics we inputted and playoff success.

Noteworthy Takeaways:

- Passer Rating (+0.42), Yds/Game (+0.41), Pass Att/Game (+0.16), and Pass TD/Game (+0.15) have a clear positive correlation with playoff success
  - Could suggest that teams with successful Pass-Heavy Offenses are disposed towards playoff success
- Passing INT/Game Allowed (+0.45), Passing Yards/Game Allowed (-0.35), Penalties (-0.21)
  - Playoff success is heavily influenced by interceptions generated, limiting opposing passing offense, and playing disciplined football
- Multicollinearity

```
Top stats correlated with playoff success:
Y/P                          -0.583263
Cmp/Game                     -0.532283
Int%                         -0.461285
Passing INT/Game Allowed      0.449366
Pass Rating                   0.424776
Yds/Game                      0.408889
Rush Att/Game                -0.405285
Passing Yards/Game Allowed   -0.352852
W - L %                       0.335857
Pass Yds/Game                 0.297787
Rush TD/Game                  0.230974
Points Scored/Game           -0.219679
Penalties                    -0.213464
Passer Rating Allowed         0.212242
Margin of Victory            -0.211125
Rush Yds/Game                 0.209797
Pass Att/Game                 0.155664
Pass TD/Game                  0.150230
Total Yards/Game Allowed      0.142362
Penalties - Yards             0.141706
```



End Result of Playoff Model Pt. 1

How the NFL Playoffs are Structured:

- *7 seeds from each conference*
- *1st seed advances to second (Divisional) round*
- *2 vs 7, 3 vs 6, 4 vs 5*
- *After each round, highest remaining seed plays lowest remaining seed*

Expected Playoff Wins:

- *Model learns the relationship between regular-season stats and playoff wins*
- *Model feeds each team's projected regular-season stats into the GLM.*
- *Provides an expected value: probability-weighted measure of playoff strength*
- *Used to decide match ups in bracket*

End Result of Playoff Model Pt. 2

- *Teams with >1 Exp PW considered legitimate Super Bowl contenders*
- *How strong the team tends to be in the playoffs, not what will happen in a single game*

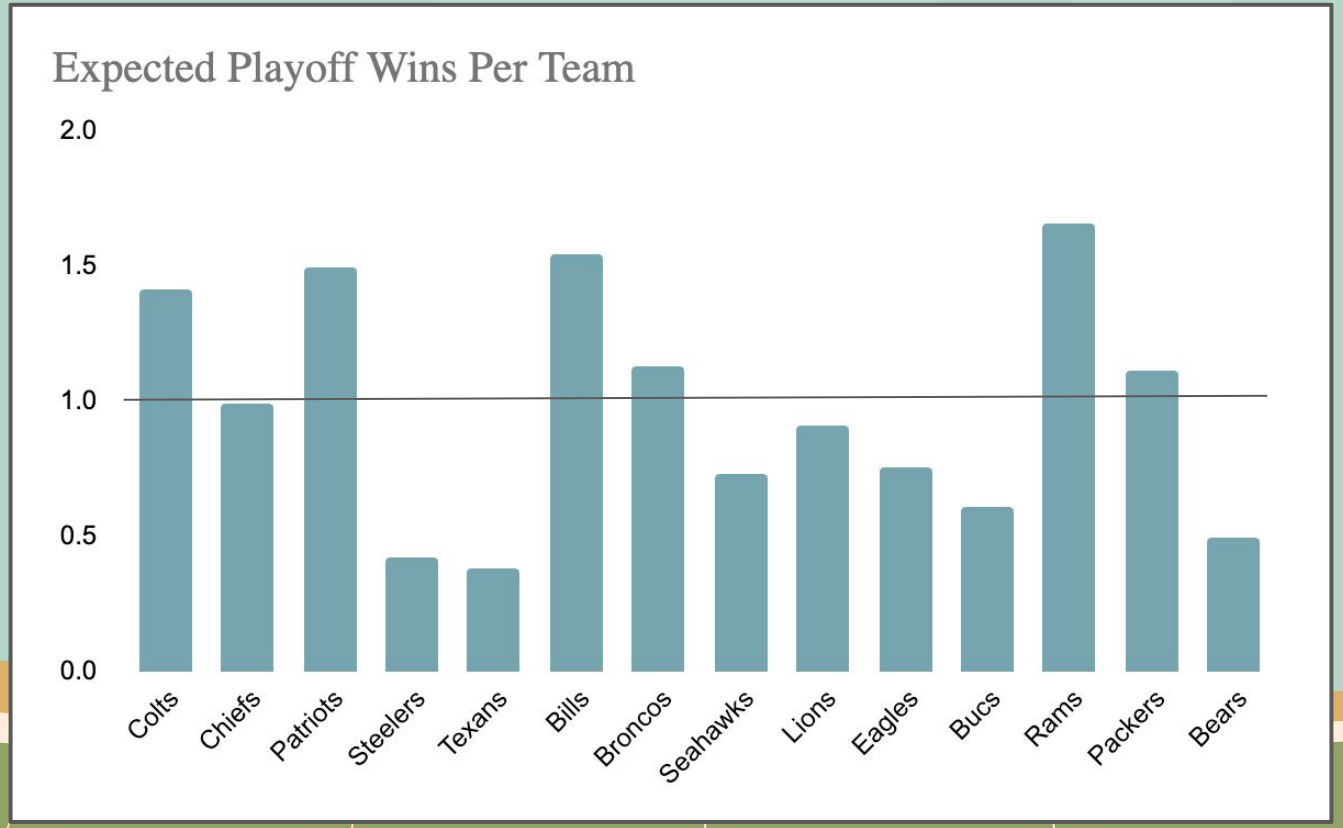
======= AFC Seeds =======

	Seed	Team	Division	pred_wins	pred_final_win_pct	expected_playoff_wins
0	1	Indianapolis Colts	AFC South	13	0.787	1.414513
1	2	Kansas City Chiefs	AFC West	12	0.723	0.994698
2	3	New England Patriots	AFC East	12	0.701	1.492351
3	4	Pittsburgh Steelers	AFC North	10	0.569	0.426668
4	5	Houston Texans	AFC South	12	0.718	0.381032
5	6	Buffalo Bills	AFC East	12	0.678	1.542834
6	7	Denver Broncos	AFC West	10	0.576	1.132492

======= NFC Seeds =======

	Seed	Team	Division	pred_wins	pred_final_win_pct	expected_playoff_wins
0	1	Seattle Seahawks	NFC West	13	0.760	0.728853
1	2	Detroit Lions	NFC North	12	0.733	0.910131
2	3	Philadelphia Eagles	NFC East	10	0.582	0.757764
3	4	Tampa Bay Buccaneers	NFC South	10	0.574	0.610024
4	5	Los Angeles Rams	NFC West	13	0.752	1.654685
5	6	Green Bay Packers	NFC North	12	0.678	1.117581
6	7	Chicago Bears	NFC North	9	0.543	0.494513

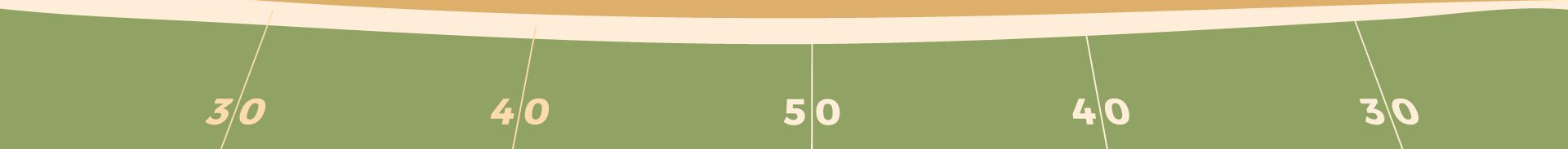
Visualization of Expected Playoff Wins



Comparison to Preseason Betting Odds

- ***Note**: Model had knowledge of first 11 of 17 weeks of NFL Season*
- 1 seed Indianapolis Colts are not in the Top 20 Preseason Super Bowl Odds
    - *Breakout of QB Daniel Jones, RB Jonathan Taylor and Colts Offense*
- 3 seed New England Patriots hardly in Top 20 Preseason Super Bowl Odds
    - *Breakout of QB Drake Maye, Easy Strength of Schedule*
- Super Bowl participants Buffalo Bills and Los Angeles Rams were Preseason ranked at #2 and #9
- #3 Preseason ranked Baltimore Ravens not included in Simulated Playoff Bracket
    - *Several injuries including former MVP QB Lamar Jackson*
- *#1 Preseason ranked Philadelphia Eagles knocked out in the first round*
    - *Change in Offensive Coordinator and unanticipated Offensive struggles*

New England Patriots	+8000	8.5
Dallas Cowboys	+6000	7.5
Seattle Seahawks	+6000	8.5
Arizona Cardinals	+5000	8.5
Chicago Bears	+4000	8.5
Pittsburgh Steelers	+4000	8.5
Houston Texans	+3500	9.5
Tampa Bay Buccaneers	+3000	9.5
Los Angeles Chargers	+2800	9.5
Denver Broncos	+2500	9.5
Minnesota Vikings	+2500	9.5
Cincinnati Bengals	+2000	9.5
Los Angeles Rams	+2000	9.5
San Francisco 49ers	+2000	10.5
Washington Commanders	+1800	9.5
Detroit Lions	+1200	10.5
Green Bay Packers	+1200	10.5
Kansas City Chiefs	+800	11.5
Baltimore Ravens	+700	11.5
Buffalo Bills	+700	12.5
Philadelphia Eagles	+700	11.5



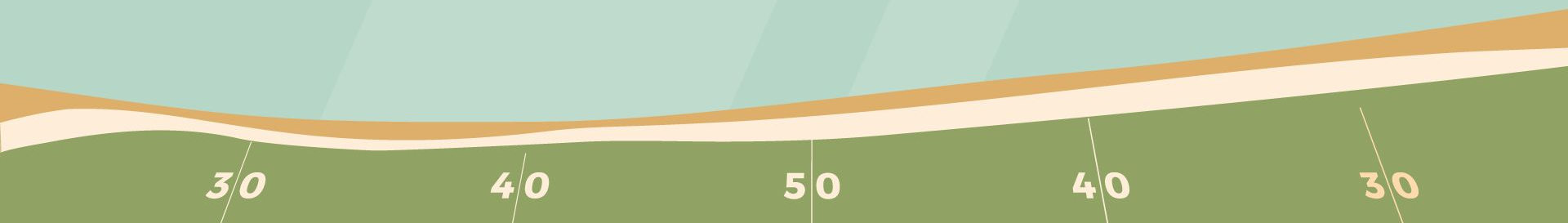
End Result of Playoff Model Pt. 3

Who wins the Super Bowl?

Overview

This part of the project takes in the two teams predicted to play in the Super Bowl, and predicts the probability that each wins.

1. Data Preprocessing

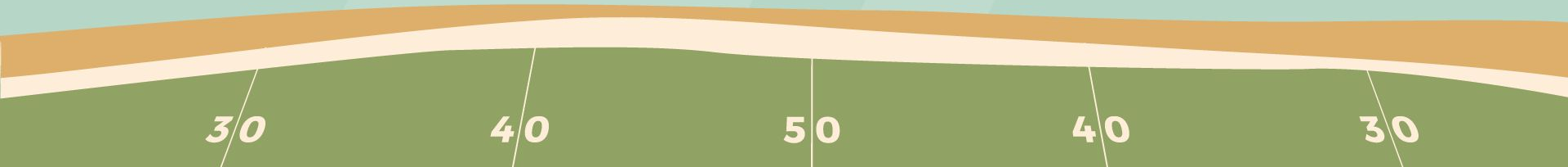


Datasets

- We will work with three data sets:
- Season-level statistics for every NFL team from 2020 - 2024
- Super Bowl matchups from 2001 - 2025, and the corresponding champion
- Predictions of stats for NFL teams in 2025 based off of data up to week 11

Attributes

- 49 features for season-level statistics
- To prevent ambiguity, we labeled each Super Bowl with a ‘NFL Year’ that corresponds to the year of NFL that the Super Bowl follows



	Year	Team	Points Scored/Game	...	Int%	QBHits/Game	TFL/Game
0	2024	Arizona Cardinals	23.53	...	1.7	4.47	5.12
1	2024	Atlanta Falcons	22.88	...	2.1	4.35	4.65
2	2024	Baltimore Ravens	30.47	...	1.9	7.18	4.82
3	2024	Buffalo Bills	30.88	...	2.8	5.53	5.47
4	2024	Carolina Panthers	20.06	...	1.7	3.76	3.82

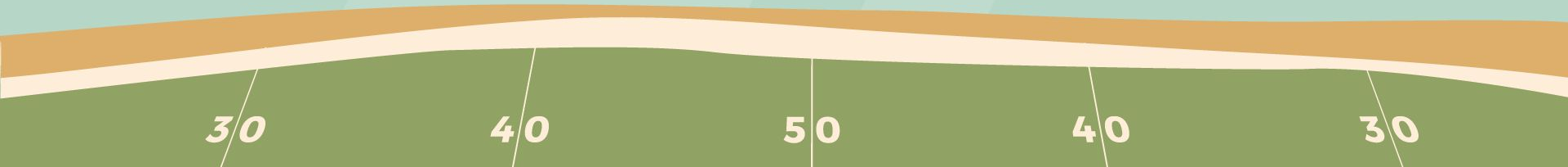
[5 rows x 51 columns]
(798, 51)

	Year	NFL Year	Winner	Loser
0	2025	2024	Philadelphia Eagles	Kansas City Chiefs
1	2024	2023	Kansas City Chiefs	San Francisco 49ers
2	2023	2022	Kansas City Chiefs	Philadelphia Eagles
3	2022	2021	Los Angeles Rams	Cincinnati Bengals
4	2021	2020	Tampa Bay Buccaneers	Kansas City Chiefs

(25, 4)

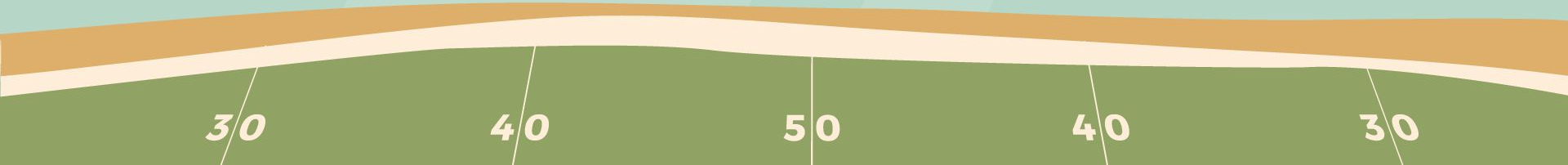
	Team	Year	Points Scored/Game	Points Allowed/Game	Margin of Victory	...	Pass Deflections/Game	Sk%	Int%	QBHits/Game	TFL/Game
0	Arizona Cardinals	2025	22.4	25.6	-3.2	...	5.2	5.2	1.7	4.8	4.7
1	Atlanta Falcons	2025	19.5	23.9	-4.4	...	4.5	10.7	2.8	5.8	5.2
2	Baltimore Ravens	2025	25.2	25.1	0.1	...	4.2	4.0	1.7	5.9	4.3
3	Buffalo Bills	2025	29.2	22.9	6.3	...	3.2	8.7	2.6	5.7	5.6
4	Carolina Panthers	2025	18.8	22.6	-3.8	...	3.8	4.5	2.1	3.5	3.7

[5 rows x 51 columns]



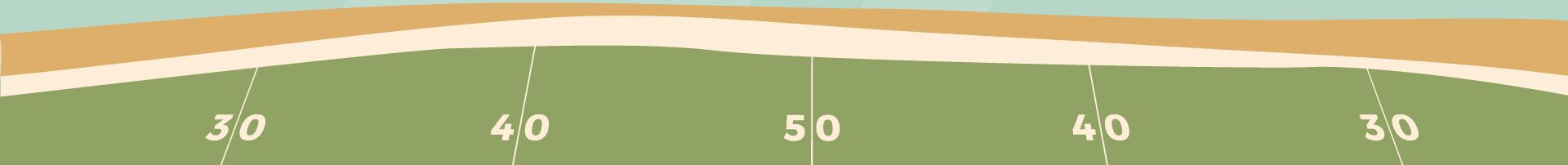
Preprocessing

- Since the Super Bowl is played between two teams, we construct difference features to be passed into the model.
- Based on these inputs, the model should learn to label the data as 1 (winning team) or 0 (losing team)
- We organized the training data with three lists: rows, labels, and groups:
- “Rows” stores the computed difference feature matrix, “labels” stores the outcome (1/0), and “groups” groups the training data according to the NFL Year.

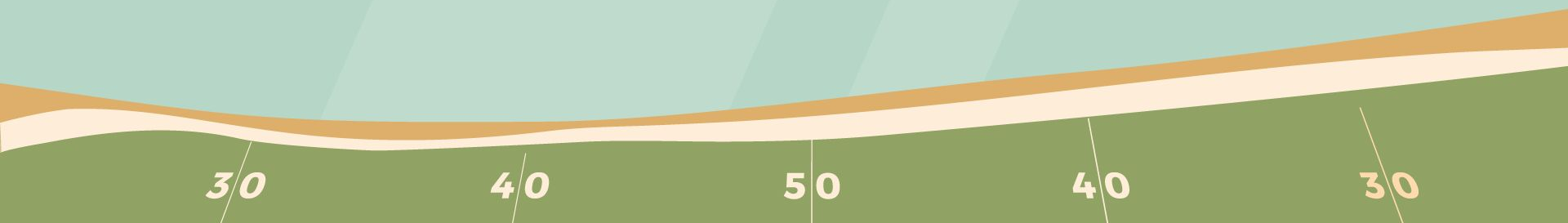


```
rows, labels, groups = [], [], []

for _, row in sb_train.iterrows():
    nfl_year = int(row["NFL Year"]) #we use nfl season start year instead of the actual superbowl year for indexing
    winner, loser = row["Winner"], row["Loser"]

    A = nfl_data[(nfl_data["Year"] == nfl_year) & (nfl_data["Team"] == winner)].iloc[0]
    B = nfl_data[(nfl_data["Year"] == nfl_year) & (nfl_data["Team"] == loser)].iloc[0]

    diff = A[feature_cols].astype(float) - B[feature_cols].astype(float) #create difference features
```

	Points Scored/Game	Points Allowed/Game	Margin of Victory	Strength of Schedule	W - L %	...	Pass Deflections/Game	Sk%	Int%	QBHits/Game	TFL/Game
0	4.59	-1.36	5.9	-2.4	-0.058	...	1.11	0.6	0.1	-1.47	-0.36
1	-4.59	1.36	-5.9	2.4	0.058	...	-1.11	-0.6	-0.1	1.47	0.36
2	-7.06	-0.24	-6.9	-1.2	-0.059	...	-0.12	2.1	-2.1	0.53	0.42
3	7.06	0.24	6.9	1.2	0.059	...	0.12	-2.1	2.1	-0.53	-0.42
4	7.71	-1.00	8.7	-0.3	0.412	...	0.00	-3.0	-1.3	-0.35	-0.47
5	-7.71	1.00	-8.7	0.3	-0.412	...	-0.00	3.0	1.3	0.35	0.47
6	0.00	-0.24	0.3	2.0	0.118	...	0.53	1.1	0.9	-0.23	0.29
7	-0.00	0.24	-0.3	-2.0	-0.118	...	-0.53	-1.1	-0.9	0.23	-0.29
8	1.19	-0.44	1.7	0.9	-0.187	...	0.38	1.8	-0.5	1.06	2.75
9	-1.19	0.44	-1.7	-0.9	0.187	...	-0.38	-1.8	0.5	-1.06	-2.75
10	-1.75	-0.13	-1.7	-0.2	-0.063	...	-0.63	-1.3	0.4	0.19	-1.31
11	1.75	0.13	1.7	0.2	0.063	...	0.63	1.3	-0.4	-0.19	1.31
12	-5.69	-3.69	-2.0	-1.4	-0.125	...	-0.06	-2.4	-0.4	0.37	-1.25



2. Choosing a model



Model Comparison

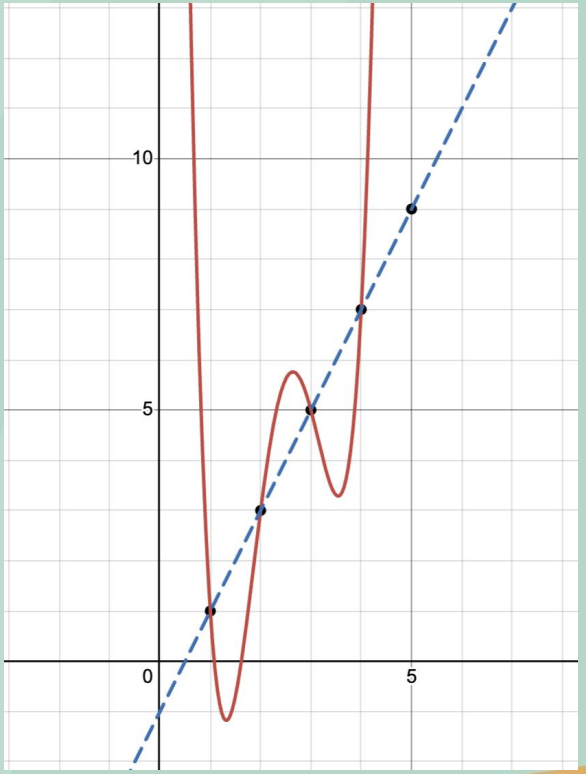
Logistic Regression

- Linear decision boundary with low model complexity
- Should be more stable on smaller data sets
- Feature weights may be interpretable

Boosted Trees

- Ensemble of decision trees reduce bias
- Nonlinear model which can capture complex relations
- We try to reduce overfitting caused by the smaller data set by setting a lower maximum depth (2), and also use fewer trees (50).

Cross Validation



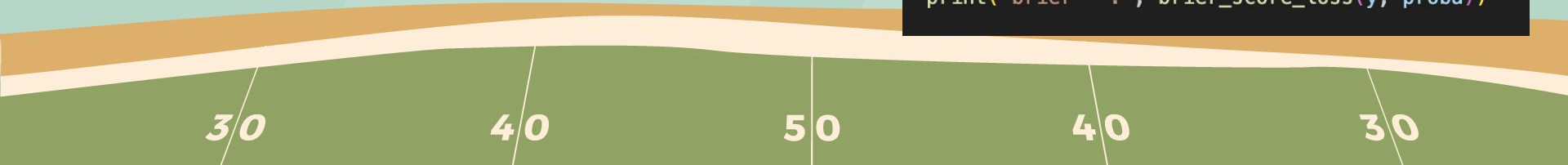
30 40 50 40 30

Leave One Group Out CV

- In LOGO, each group of data will be left out once as the test set while the model is trained on all of the remaining data, repeatedly.
- This reduces the bias of the evaluation in comparison to splitting the data into train/test only once.
- For our data set the groups are already sorted using NFL Years.

- We use a scikit-learn library:

```
# Cross validation
logo = LeaveOneGroupOut()

proba = cross_val_predict(
    logit_pipe, X, y,
    cv=logo, groups=groups,
    method="predict_proba"
)[:, 1]

pred = (proba >= 0.5).astype(int)

print("accuracy:", accuracy_score(y, pred))
print("logloss :", log_loss(y, proba))
print("brier   :", brier_score_loss(y, proba))
```



CV Results

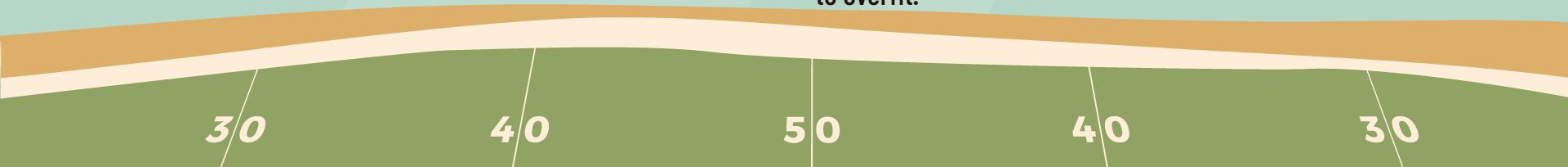
- Accuracy measures the correctness of the model, converting the predicted probabilities into the target outcomes 1/0.
- Log-Loss and Brier Score measures how much the predicted probabilities differ from the actual outcome.
- Logistic Regression (with regularization) outperforms Boosted Trees on all three criteria.

Logistic Regression vs Boosted Trees

```
accuracy: 0.68
logloss : 0.634625523473115 5
brier   : 0.220474024040230 7
```

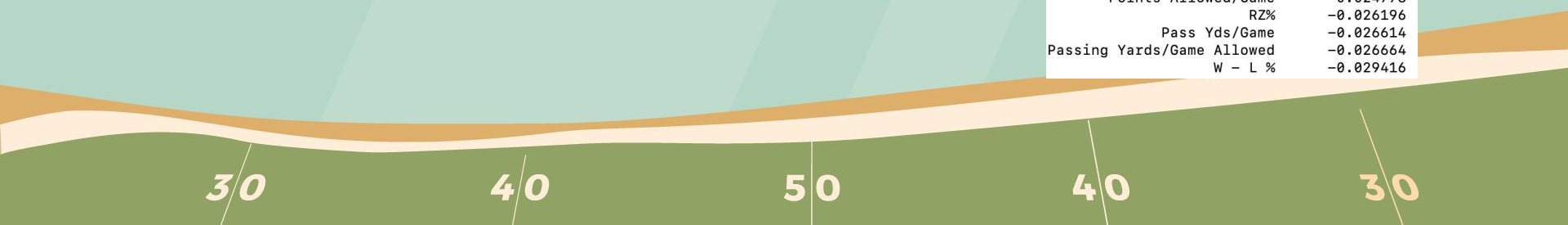
```
Accuracy : 0.54
Log Loss : 0.7958221007635176
Brier    : 0.2810811385787553
```

- This is probably because of the smaller data set and the lack of need for a overly complex relation causing Boosted Trees to overfit.

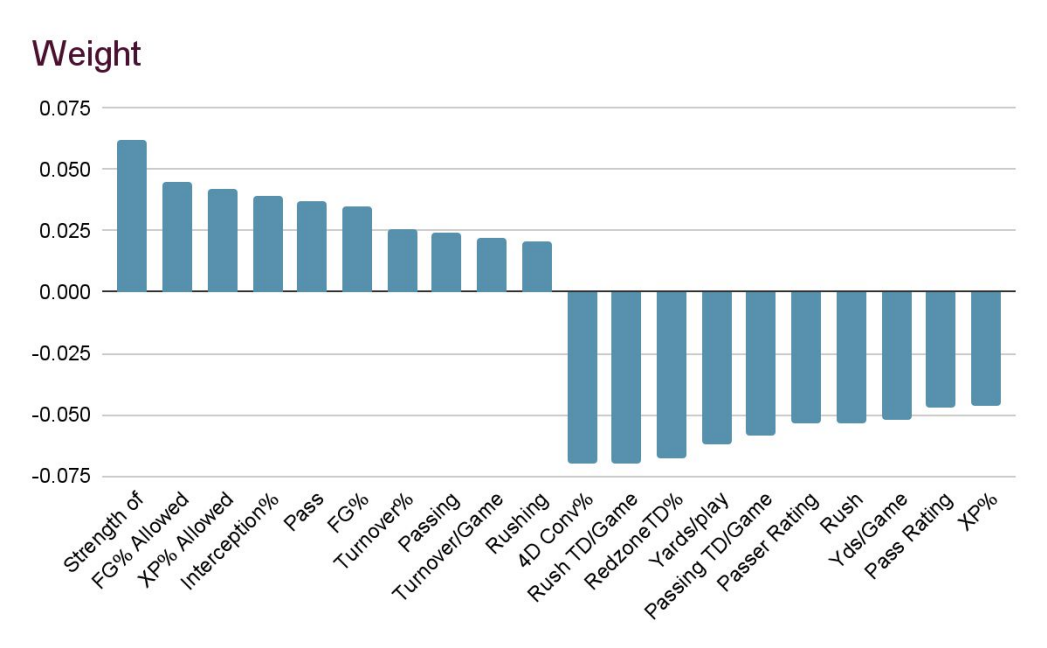


3. Results

Feature	Weight (coef)
Strength of Schedule	0.062420
FG% Allowed	0.045198
XP% Allowed	0.042410
Int%	0.039420
Pass Deflections/Game	0.037146
FG%	0.034643
Turnover%	0.025662
Passing INT/Game Allowed	0.024583
TO/Game	0.022002
Rushing Yards/Game Allowed	0.020732
TO%	0.017199
Pass Att/Game	0.016241
Int/Game	0.009320
Rushing TD/Game Allowed	0.006572
Sk%	0.000060
Cmp/Game	-0.006178
Scoring% Allowed	-0.006459
Penalties - Yards	-0.006895
Rush Att/Game	-0.007635
Total Yards/Game Allowed	-0.008718
Margin of Victory	-0.015261
3D% Allowed	-0.016875
3D Conv%	-0.018012
Pass TD/Game	-0.019813
TFL/Game	-0.024701
Points Allowed/Game	-0.024778
RZ%	-0.026196
Pass Yds/Game	-0.026614
Passing Yards/Game Allowed	-0.026664
W - L %	-0.029416



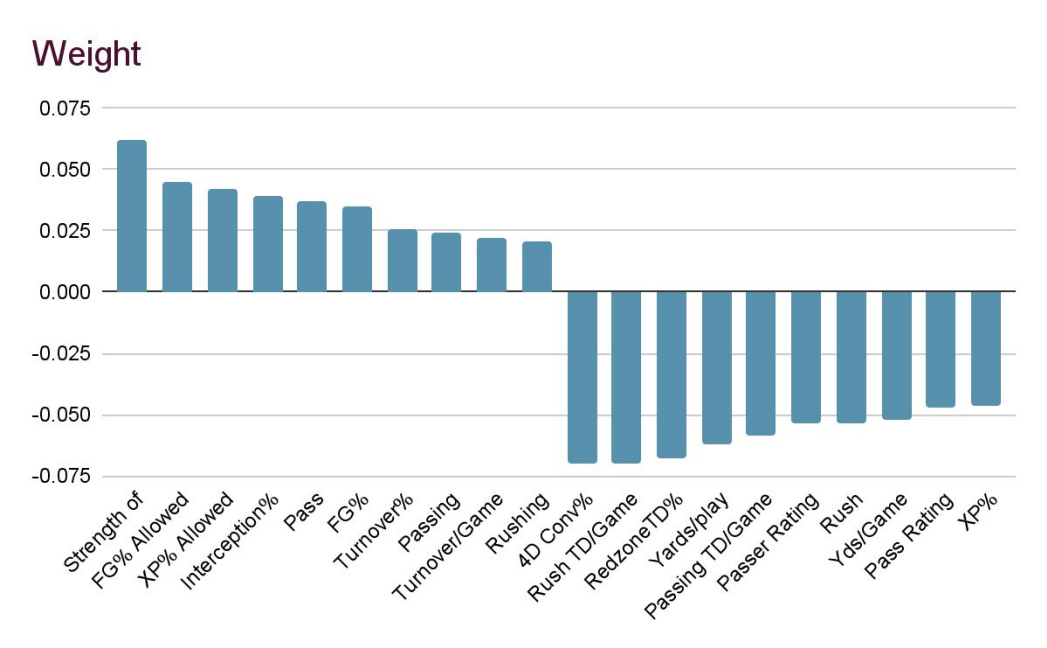
Weight Contribution



Interpretation

- Some 'positive' stats have negative weights, while some 'negative' stats have positive weights.
- 2 Explanations:
  - This model is only based on Super Bowl outcomes and not regular season
  - Statistical correlations in smaller datasets could be counter-intuitive

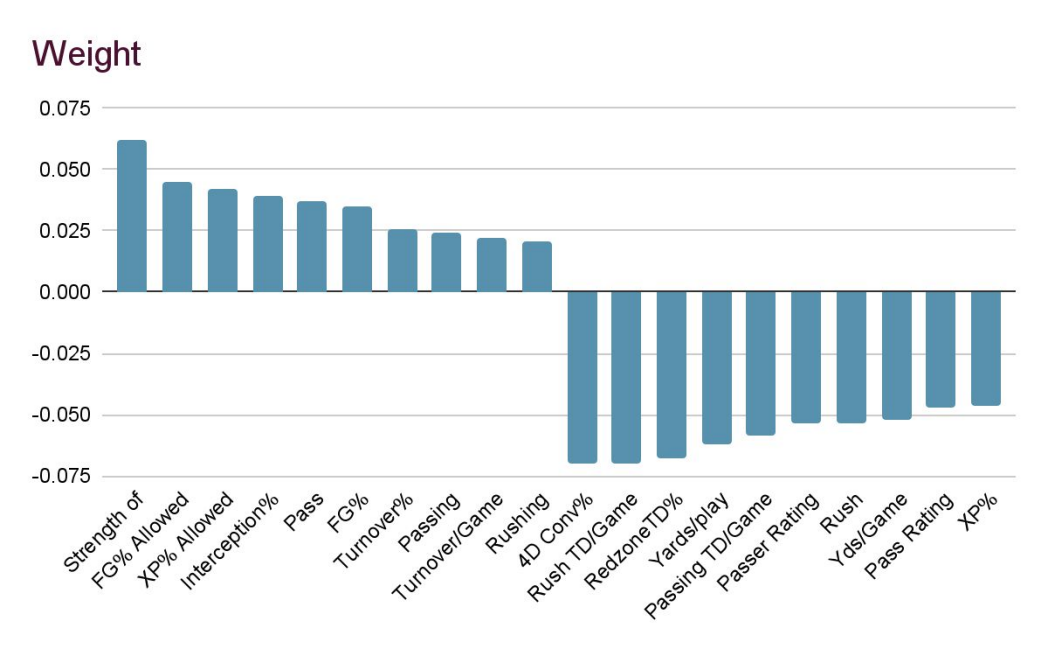
Weight Contribution



Interpretation

- Strength of Schedule: Qualifying despite stronger opponents demonstrates team strength
- Interception, Turnover: More aggressive or risk-taking play styles may be rewarded in a final
- FG% allowed: Possibly because strong defense forces touchdown opportunities into field goal attempts.

Weight Contribution



Interpretation

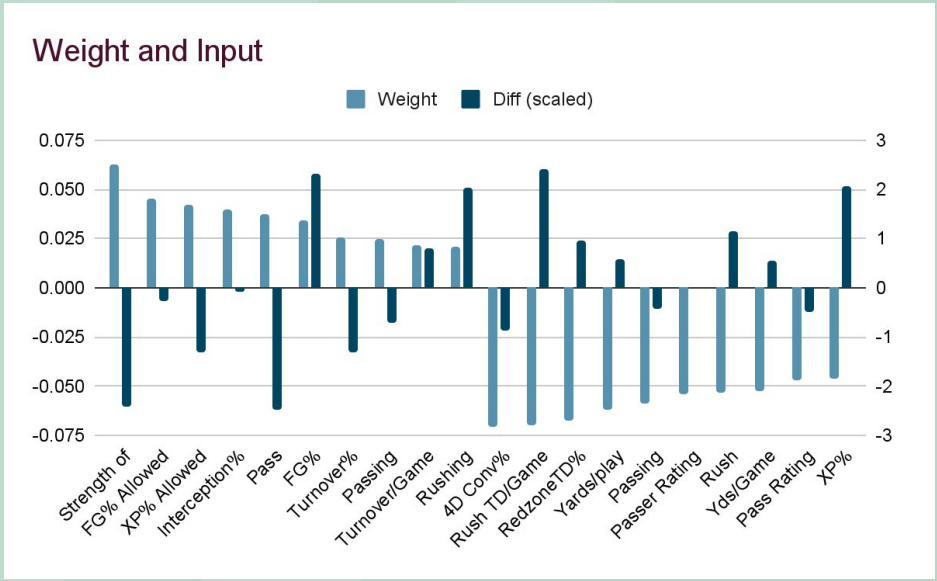
- Redzone/Passing Touchdowns allowed: Worse defense
- 4th Down Conversion%: Stronger teams may not need to rely on this metric
- Again, these data only correlate with Super Bowl Winners, and does not necessarily relate to playing better football

Super Bowl Prediction

Bills vs Rams

```
teamA = "Buffalo Bills"
teamB = "Los Angeles Rams"
```

- The difference between each feature of Buffalo Bills and Los Angeles Rams were calculated, and then standardized.
- Each difference feature contributes by the product of its weight with the scaled difference.
- Result?



Super Bowl Winner

Rams

Super Bowl Winner

Rams

- Probability of winning for Los Angeles Rams = 0.66

- This arises from the sum of the contributions of each individual difference feature inputted into the sigmoid function


THANK
YOU
85
30
50
40
30