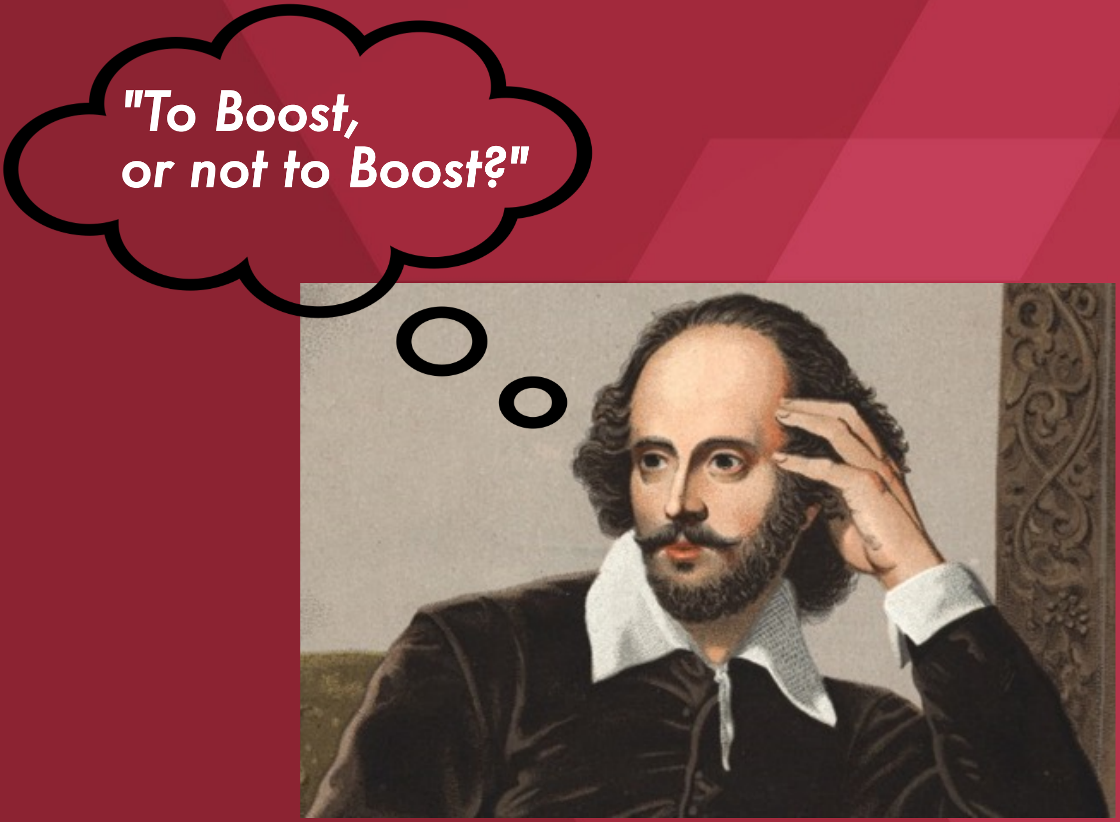


An Analysis of Gradient-Boosting Methods

Romith Challa, Troy Jennings, Adi Bose

COMP 4442, Fall 2021

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Research Question:

Does an increase in sophistication of tree-based boosting algorithms impact the accuracy & efficiency of NFL play predictions?

Methodology:

- Split multiple, consecutive years of processed **play-by-play** data into *training* & *test* sets
- Run a basic **decision-tree** model for baseline classification performance
- Implement a standard **gradient-boosting** algorithm & an **extreme-gradient boosting** algorithm (XG-Boost)
- Evaluate performance based on:
  - **Accuracy** rate of predicting a multiclass categorical output (RUN vs. PASS vs. FIELD-GOAL vs. PUNT)
  - Relative **time** to run models

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

Data Source and Definitions

Variable	Description	Data Type	Type
yardline_100	Yards until the goal line	Integer from [0, 100]	Independent
half_seconds_remaining	Seconds remaining in the half	Integer from [0, 1800]	Independent
down	The current play down	Integer from [1, 4]	Independent
ydstogo	Yards until first down	Integer from [0, 100]	Independent
posteam_timeouts	Offensive team timeouts remaining	Integer from [0, 3]	Independent
score_differential	The difference in score between teams	Integer from $(-\infty, \infty)$	Independent
play_type	The resulting type of play	Categorical: run, pass, field_goal, punt	Dependent

Source: *https://www.kaggle.com/maxhorowitz/nflplaybyplay2009to2016*

Data was scraped and uploaded onto **Kaggle** by the Carnegie Mellon Sports Analytics Club

EDA and Data Preparation

Pre-Processing

Culled the dataset from 250+ columns & 400K+ observations to a manageable 7 columns & 4 seasons of interest

Removed edge-cases: timeouts, penalties, XPA

Explored variable distributions to determine optimal datatypes to feed into model

Encoded **play_type** (outcome) and created dummy vectors for **down** (categorical predictor)

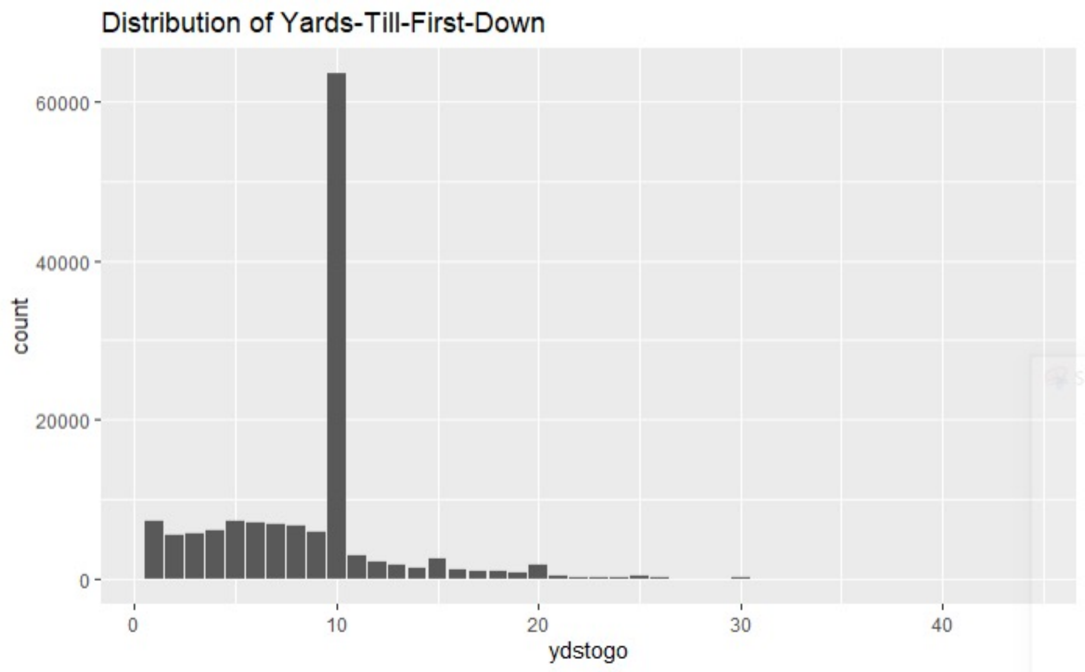
Finalized Predictors:
down, ydstogo, yardline_100, score_differential, half_seconds_remaining, posteam_timeouts

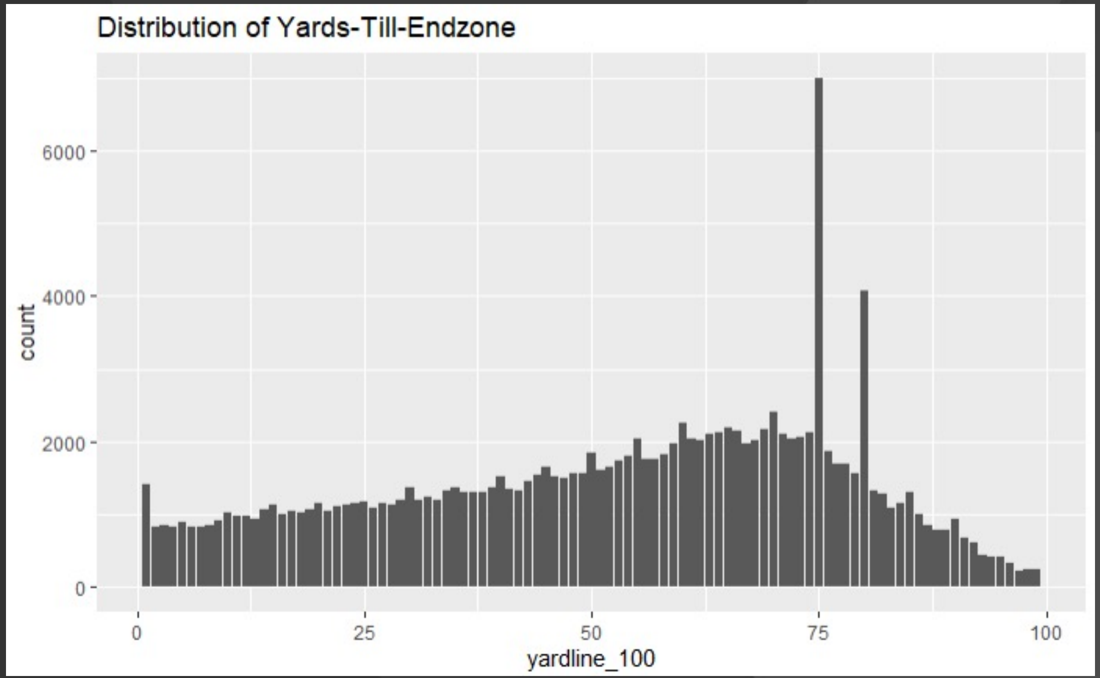
Training set: play-by-play data from 3 seasons (2015 – 2017)

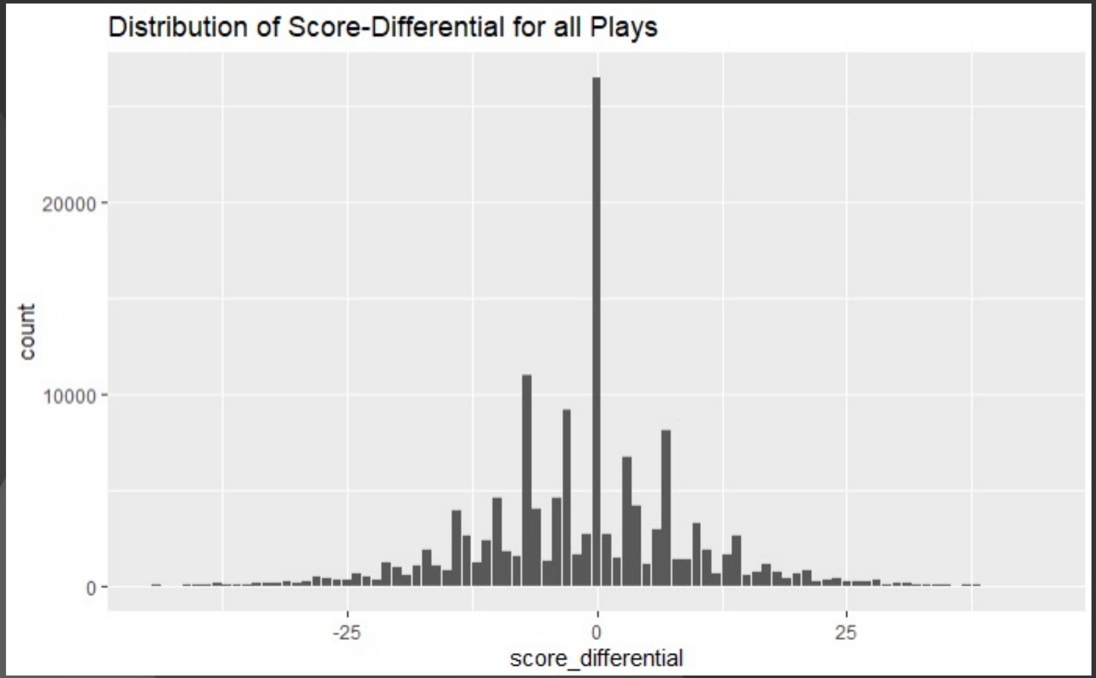
Testing set: play-by-play from 1 season (2018)

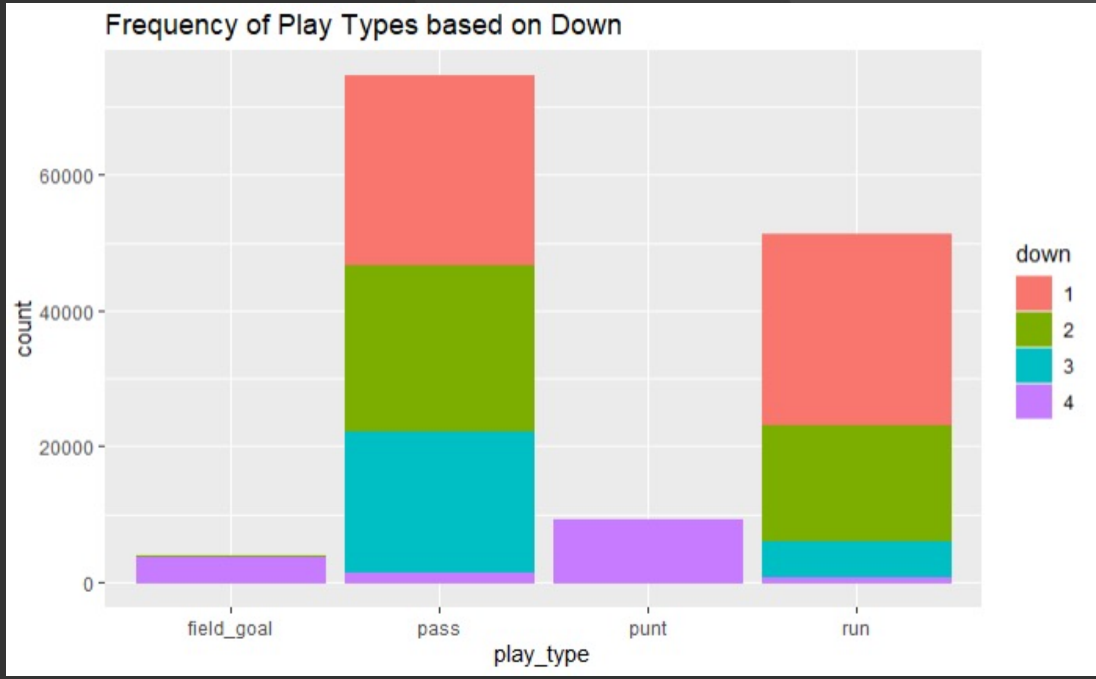
Data-Partitioning

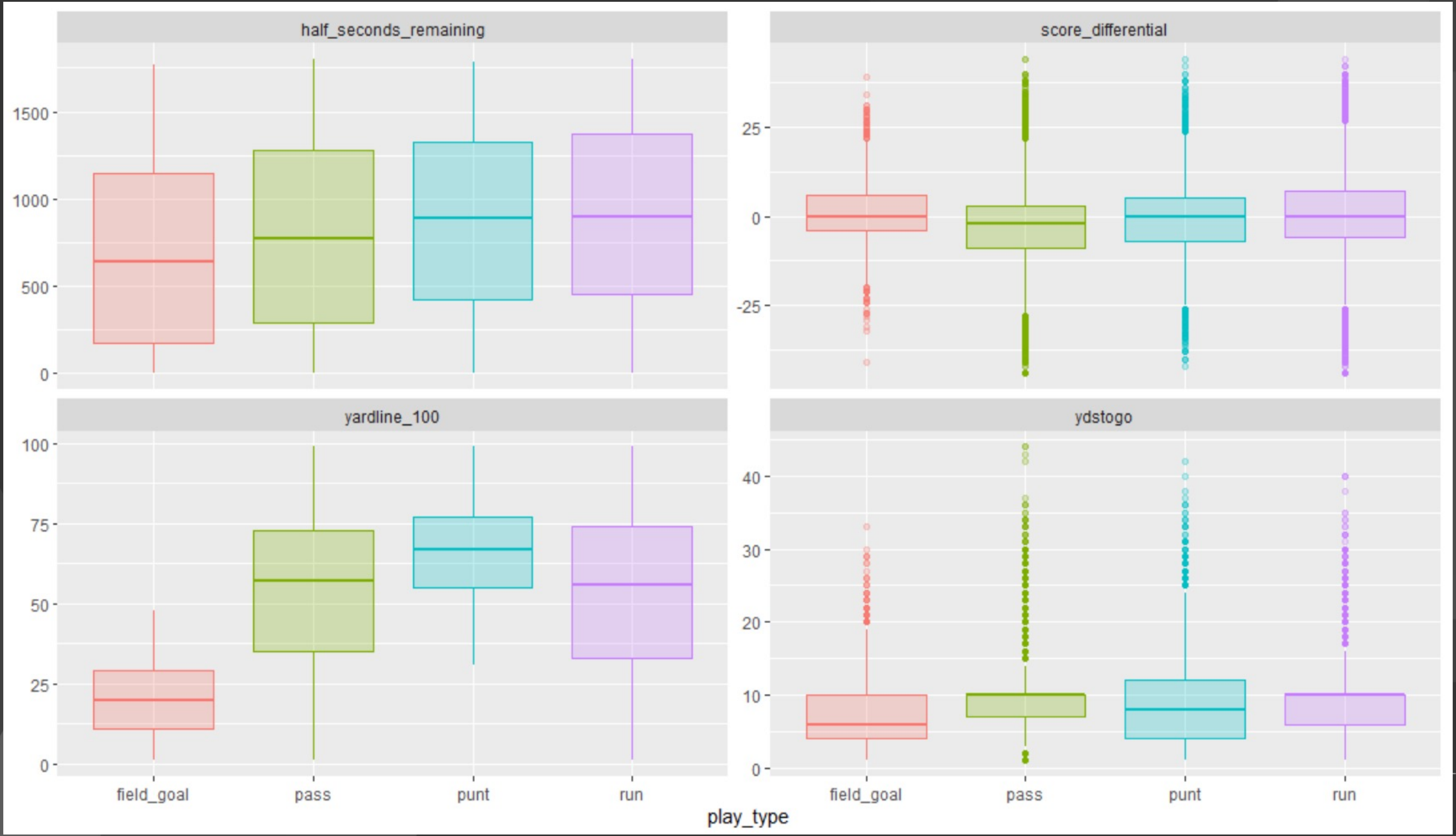
UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE


Distribution of Yards-Till-First-Down
count
60000
40000
20000
0
0
10
20
30
40
ydstogo


Distribution of Yards-Till-Endzone
count
6000
4000
2000
0
0
25
50
75
100
yardline_100


Distribution of Score-Differential for all Plays
count
20000
10000
0
-25
0
25
score_differential


Frequency of Play Types based on Down
count
60000
40000
20000
0
field_goal
pass
punt
run
play_type
down
1
2
3
4



half_seconds_remaining
score_differential
yardline_100
ydstogo
0
500
1000
1500
-25
0
25
0
25
50
75
100
0
10
20
30
40
field_goal
pass
punt
run
field_goal
pass
punt
run
play_type

How do tree-based boosting algorithms address our research question?

Decision Tree

- A single model of decision-making branches to reach a predicted classification
- Serves as a "baseline" model for our study

Gradient-Boosting

- Combine multiple models that sequentially learns from past decision trees
- Leads to higher accuracy but can be time-intensive

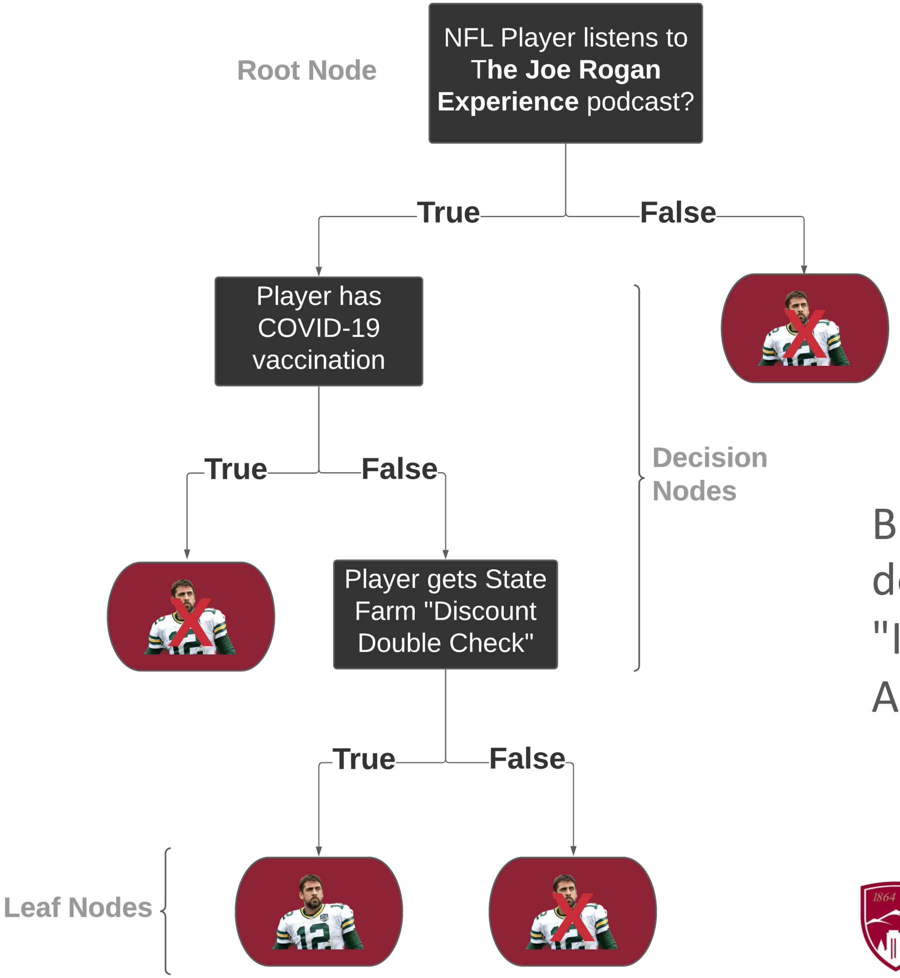
Extreme Gradient-Boosting (XG-Boost)

- Optimizes for speed, while retaining predictive power of standard gradient-boosting
- Allows for more control in fine-tuning parameters

Background — Decision Trees

Principles of Decision Trees (for Classification)

Begin	Begin with a root node
Find Attributes	Find the best attributes in the dataset from all unused attributes
Select Attribute	Select the attribute with the most purity (least uncertainty) and generate a new decision node
Make Split	Split the data by the selected attribute to create a new subset of the data
Continue Splitting	Continue to recur on each subset until no more splits can be made
Predict	Make predictions

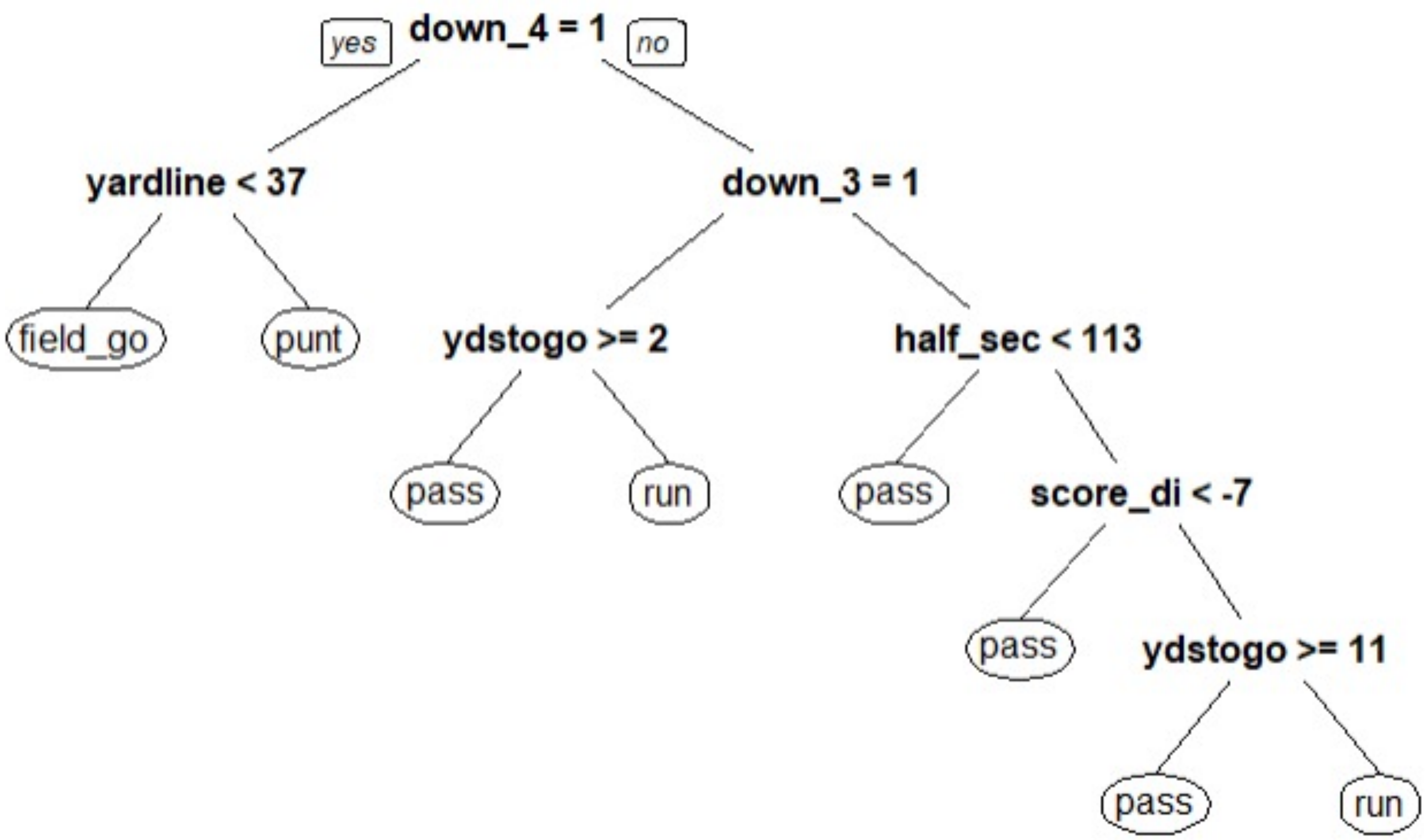


Binary classification decision tree for "Is an NFL player Aaron Rodgers?"

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL OF ENGINEERING & COMPUTER SCIENCE

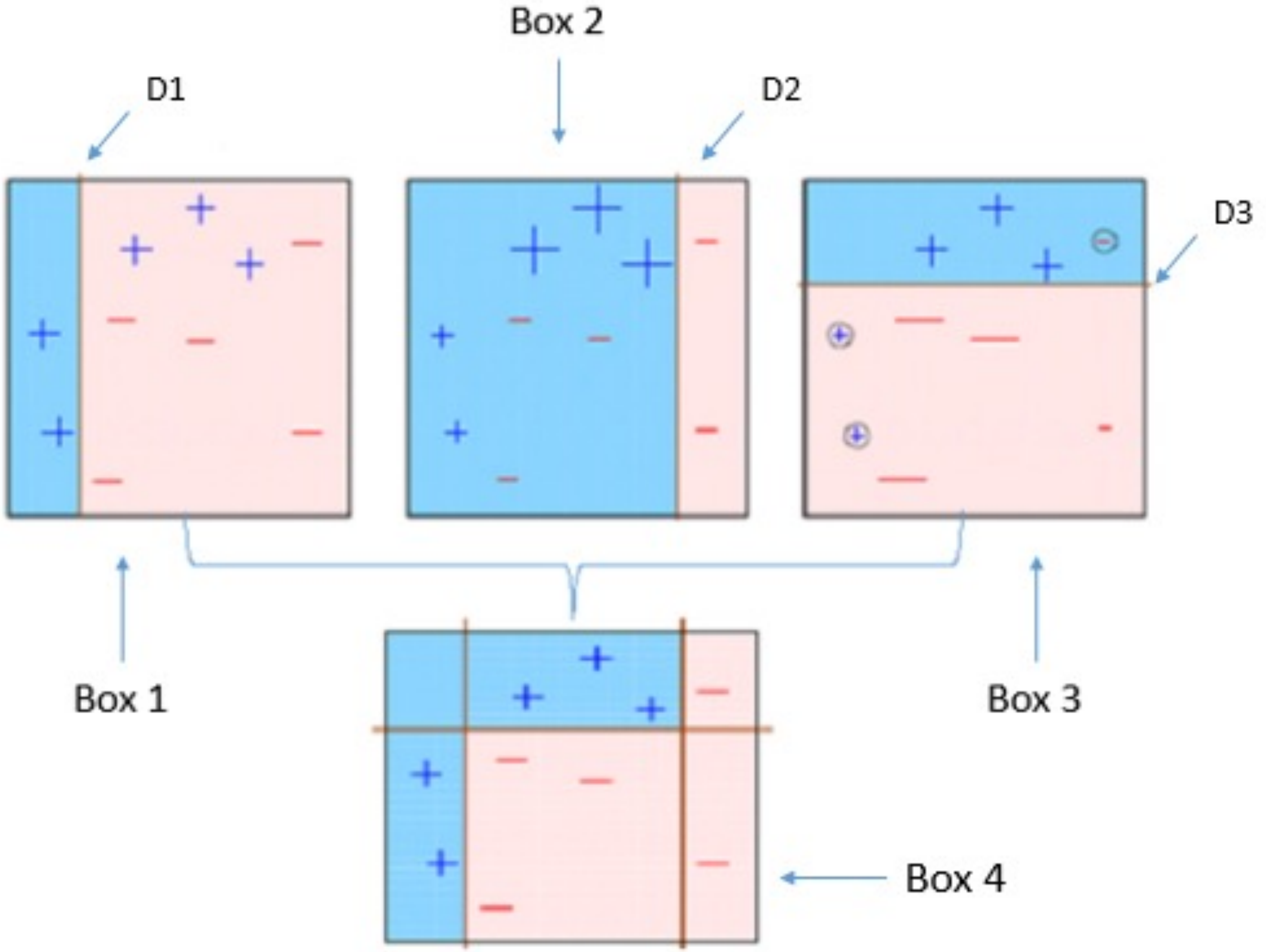
Decision Tree

Baseline Decision Tree



Primary Method: Boosting

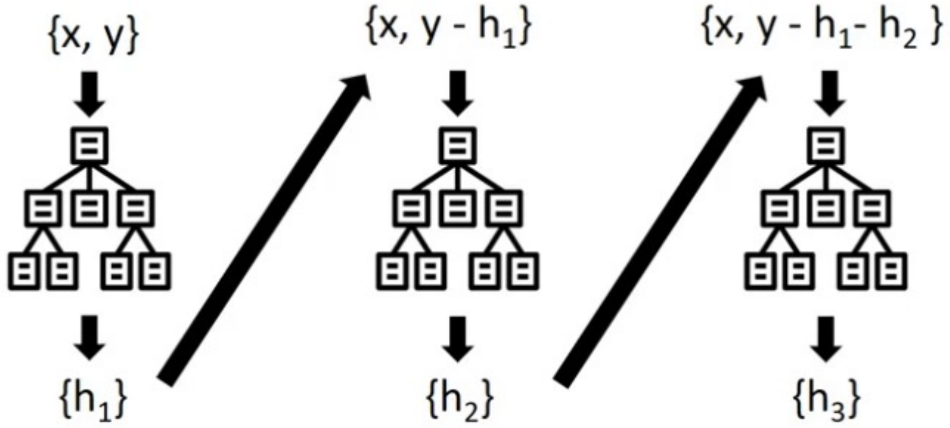
Boosting – An ensemble method using multiple Decision Trees



Source: https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/beginners-tutorial-on-xgboost-parameter-tuning-r/tutorial/

Primary Method: Boosting

Gradient-Boosting



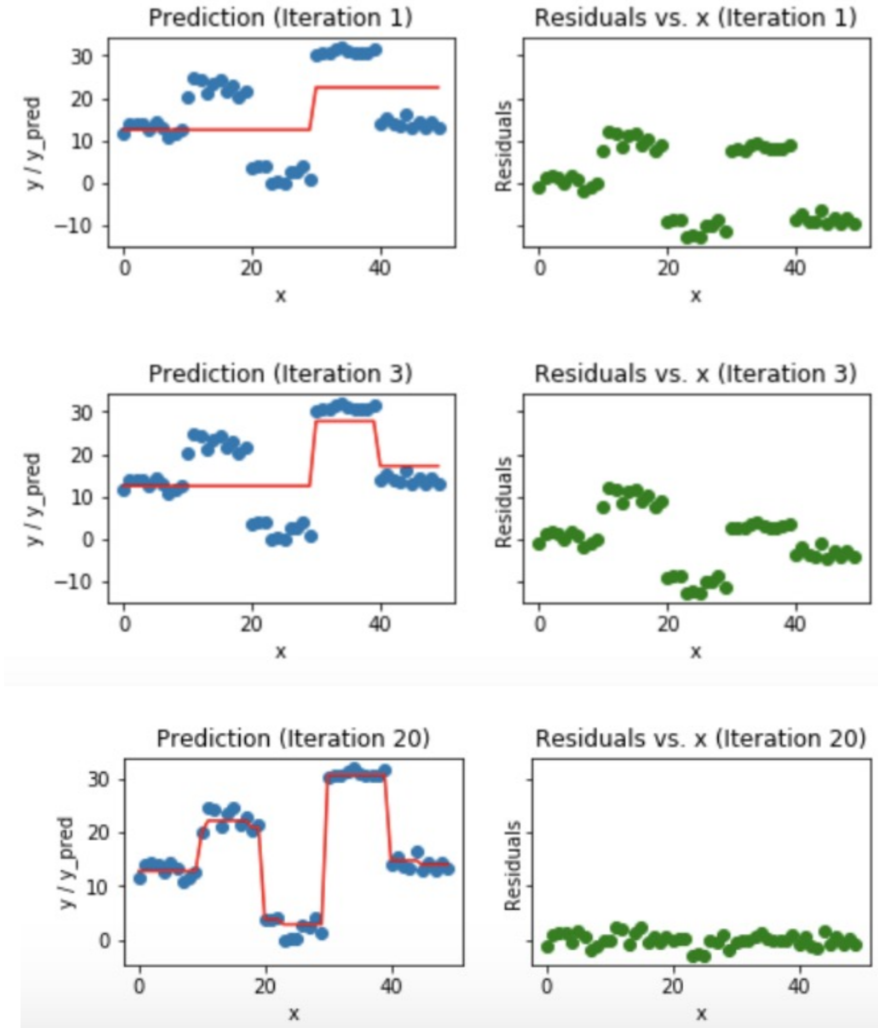
Source: https://sefiks.com

- Base model
- Compute residual errors
    - Differentiable loss function (classification: **logloss**)
    - Gradient-Descent
- Parameters to avoid overfitting and output a final model with low variance and low bias

#1

#3

#20



Source: https://blog.mlreview.com/gradient-boosting-from-scratch-1e317ae4587d

Primary Method: Boosting

XG-Boost (Extreme Gradient-Boosting)

GRADIENT-BOOSTING

Not optimized

Can be slow

First-Order gradients

Less flexibility

THE "CHAD" XG-BOOST

Tree-pruning via depth-first search

Regularization to avoid over-fitting

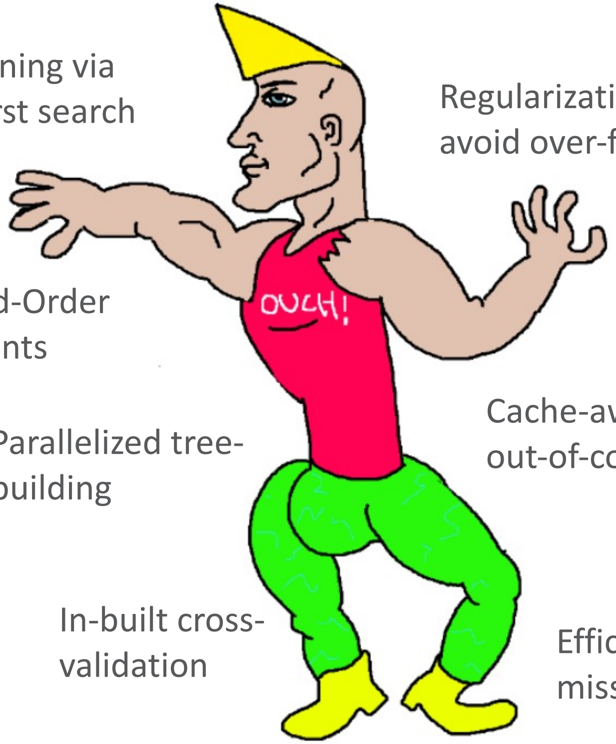
Second-Order Gradients

Cache-awareness & out-of-core computing

Parallelized tree-building

In-built cross-validation

Efficiently handles missing data



Primary Method: Boosting

XG-Boost Implementation

Data Requirement	Approach to Satisfy Requirement
Factor variable for classification problems	- Predictor Factors: dummy-vectorized down - Remaining variables kept in numeric form
Only numeric vectors (must include 0 for converted factors)	Model matrix form of predictors and outcome converted to specialized matrix in xgb package (xgb.DMatrix)
Input data structure for model: sparse matrix (cells containing 0 not stored, so enforcing memory-efficiency)	Encoded Outcome Variable: fg → 0 | pass → 1 | punt → 2 | run → 3

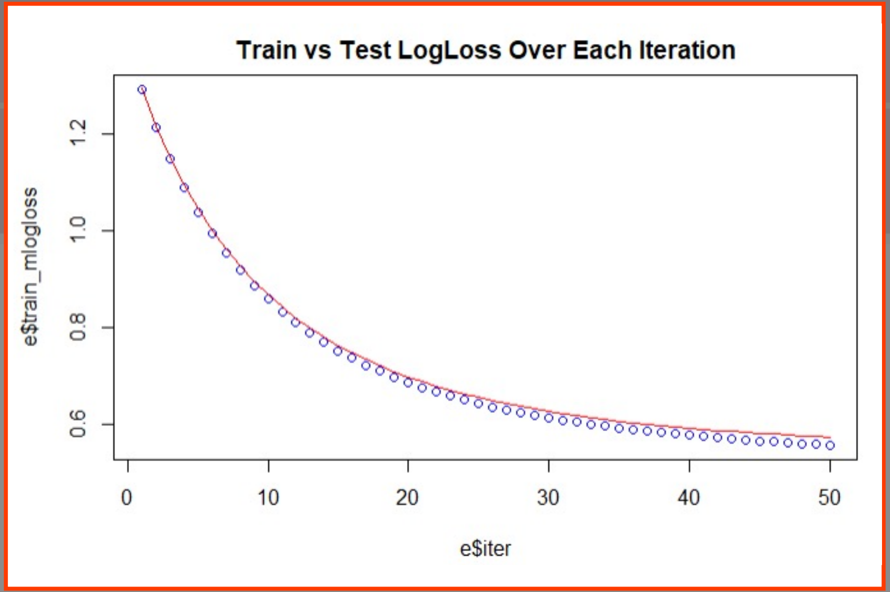
XG-Boost handles the following data characteristics: **correlated features** and **null values**

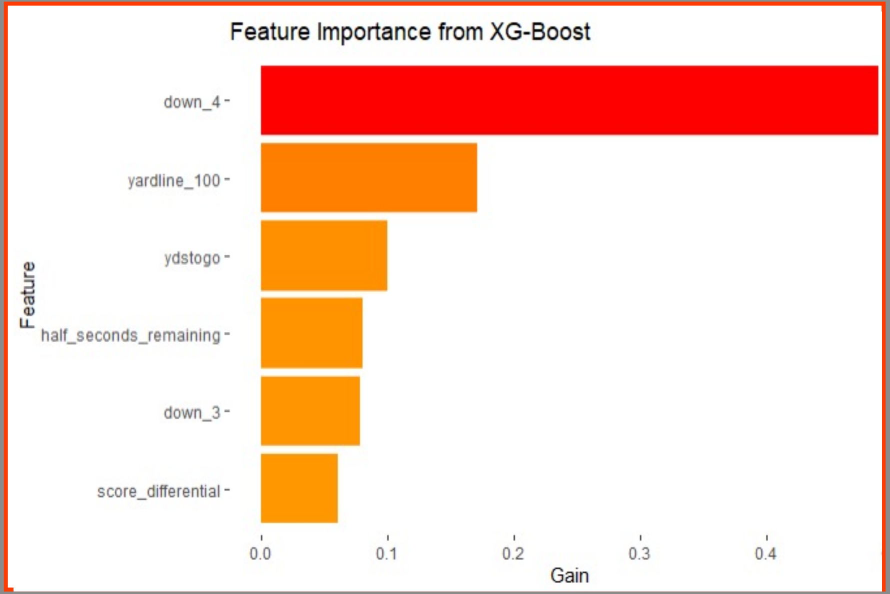
Primary Method: Boosting

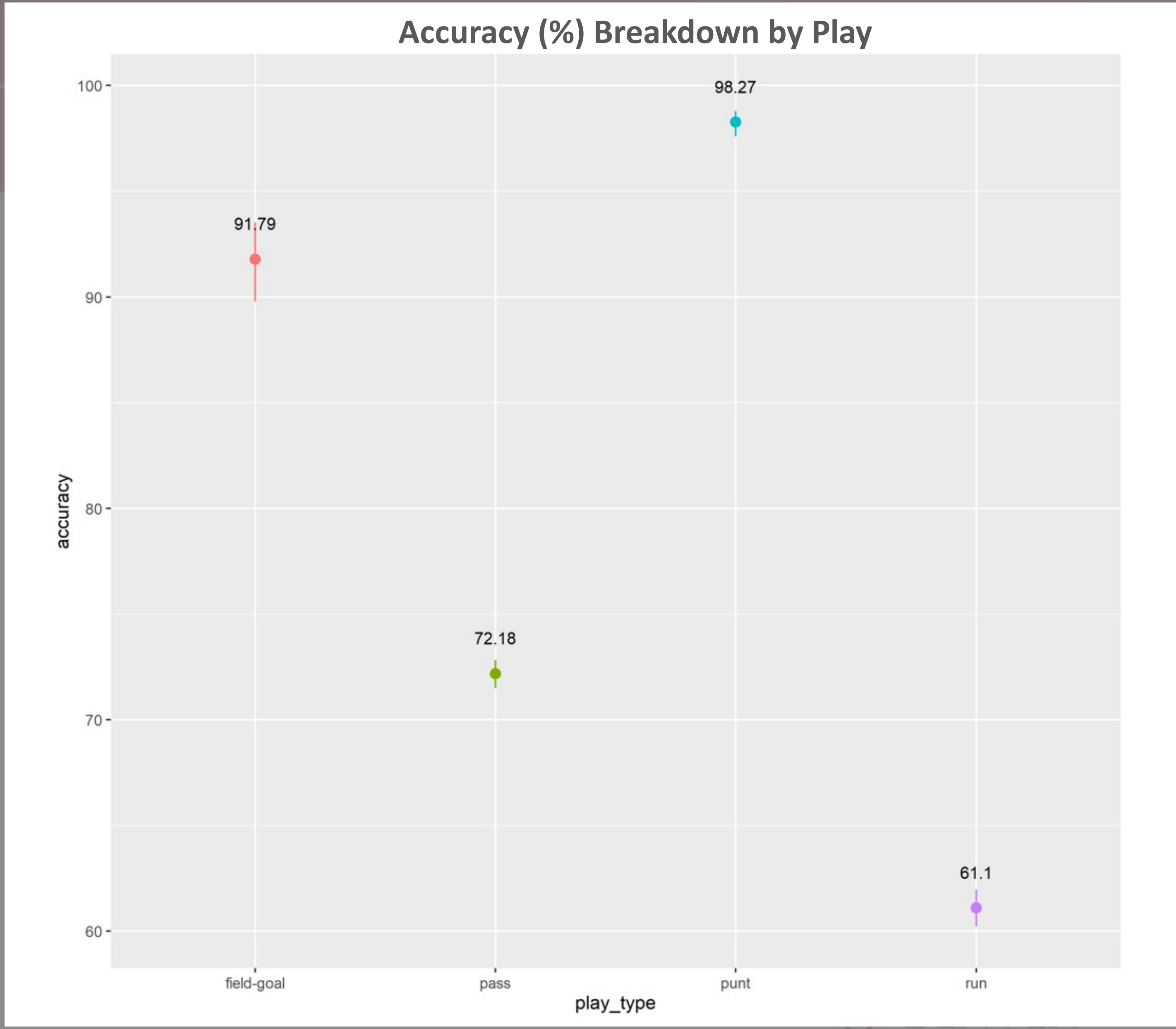
XG-Boost Implementation

Parameter	Option Used	Description
objective	multi:softprob	Outputs predicted probability of observations belonging to each class
eval_metric	mlogloss	Type of metric for validation for each tree; negative log-likelihood for classifications
eta	0.1	Learning rate [0, 1]; lower eta avoids overfitting
max_depth	5	Maximum depth of each tree; lower max_depth avoids overfitting
lambda	1	Regularization term; higher lambda leads to reduce overfitting
gamma	1	Minimum loss reduction to make partition in tree; larger gamma avoids overfitting
nrounds	50	Number of iterations

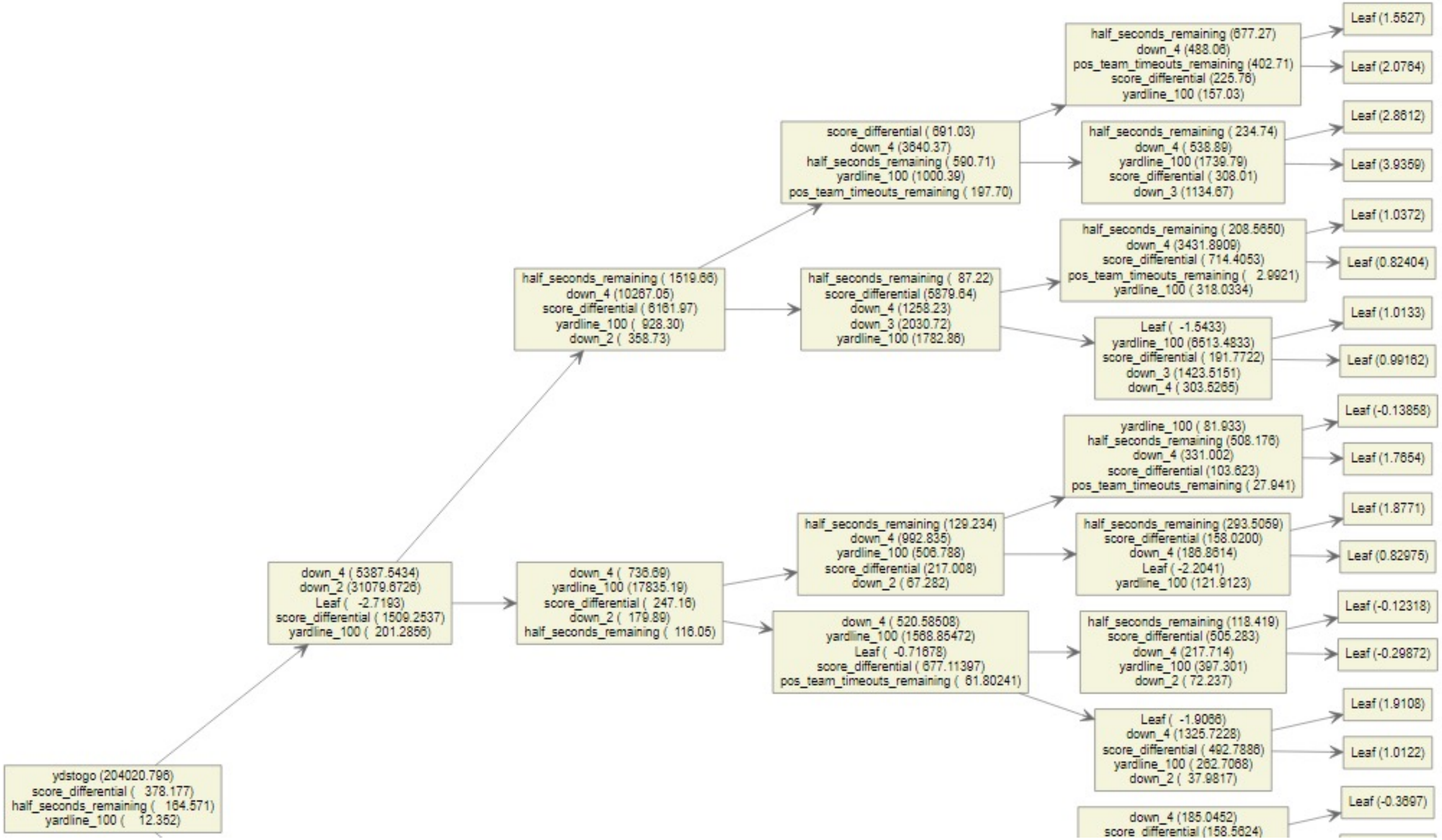
- Other notable parameters: **min_child_weight, colsample_bytree, subsample**
- **Grid Search** for hyper-parameter tuning







DANIEL FELIX RITCHIE SCHOOL
OF ENGINEERING & COMPUTER SCIENCE


ydstogo (204020.796)
score_differential (378.177)
half_seconds_remaining (164.571)
yardline_100 (12.352)
down_4 (5387.5434)
down_2 (31079.6726)
Leaf (-2.7193)
score_differential (1509.2537)
yardline_100 (201.2856)
half_seconds_remaining (1519.66)
down_4 (10267.05)
score_differential (6161.97)
yardline_100 (928.30)
down_2 (358.73)
down_4 (736.69)
yardline_100 (17835.19)
score_differential (247.16)
down_2 (179.89)
half_seconds_remaining (116.05)
score_differential (691.03)
down_4 (3640.37)
half_seconds_remaining (590.71)
yardline_100 (1000.39)
pos_team_timeouts_remaining (197.70)
half_seconds_remaining (87.22)
score_differential (5879.64)
down_4 (1258.23)
down_3 (2030.72)
yardline_100 (1782.86)
half_seconds_remaining (129.234)
down_4 (992.835)
yardline_100 (506.788)
score_differential (217.008)
down_2 (67.282)
down_4 (520.58508)
yardline_100 (1568.85472)
Leaf (-0.71678)
score_differential (677.11397)
pos_team_timeouts_remaining (61.80241)
half_seconds_remaining (677.27)
down_4 (488.06)
pos_team_timeouts_remaining (402.71)
score_differential (225.76)
yardline_100 (157.03)
half_seconds_remaining (234.74)
down_4 (538.89)
yardline_100 (1739.79)
score_differential (308.01)
down_3 (1134.67)
half_seconds_remaining (208.5650)
down_4 (3431.8909)
score_differential (714.4053)
pos_team_timeouts_remaining (2.9921)
yardline_100 (318.0334)
Leaf (-1.5433)
yardline_100 (6513.4833)
score_differential (191.7722)
down_3 (1423.5151)
down_4 (303.5265)
yardline_100 (81.933)
half_seconds_remaining (508.176)
down_4 (331.002)
score_differential (103.623)
pos_team_timeouts_remaining (27.941)
half_seconds_remaining (293.5059)
score_differential (158.0200)
down_4 (186.8614)
Leaf (-2.2041)
yardline_100 (121.9123)
half_seconds_remaining (118.419)
score_differential (505.283)
down_4 (217.714)
yardline_100 (397.301)
down_2 (72.237)
Leaf (-1.9066)
down_4 (1325.7228)
score_differential (492.7886)
yardline_100 (262.7068)
down_2 (37.9817)
down_4 (185.0452)
score_differential (158.5624)
Leaf (1.5527)
Leaf (2.0764)
Leaf (2.8612)
Leaf (3.9359)
Leaf (1.0372)
Leaf (0.82404)
Leaf (1.0133)
Leaf (0.99162)
Leaf (-0.13858)
Leaf (1.7654)
Leaf (1.8771)
Leaf (0.82975)
Leaf (-0.12318)
Leaf (-0.29872)
Leaf (1.9108)
Leaf (1.0122)
Leaf (-0.3697)

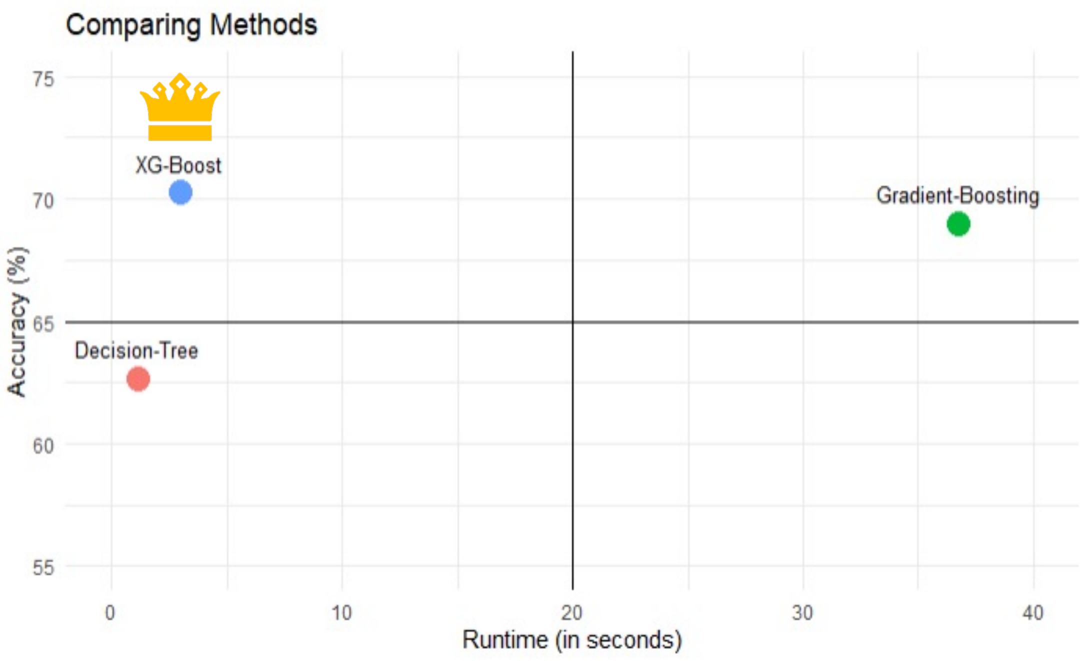
Does an increase in sophistication of tree-based boosting algorithms impact the accuracy & efficiency of NFL play predictions?

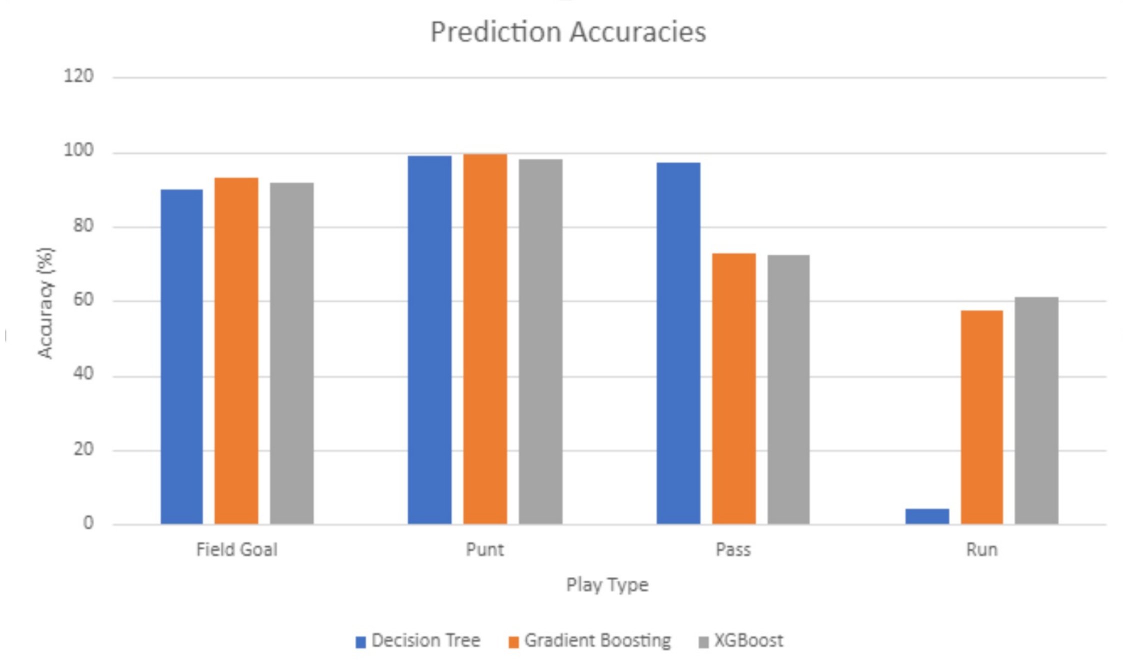
Method	Classification Accuracy	Runtime
Decision Tree (Baseline)	62.64 %	1.1 sec
Gradient-Boosting	69.01 %	36.8 sec (50 iterations)
XGBoost	70.25 %	2.9 sec (50 iterations)

UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL
OF ENGINEERING & COMPUTER SCIENCE

Conclusions

Summary of Model Analysis





UNIVERSITY of DENVER
DANIEL FELIX RITCHIE SCHOOL
OF ENGINEERING & COMPUTER SCIENCE


John Edwards
@John_B_Edwards
was inspired by this to share my own, personal cheat sheet i rely on for my projects in case it's helpful
JOHN EDWARDS'
PATENTED MACHINE
LEARNING ALGORITHM
CHEAT SHEET
START
USE XGBOOST