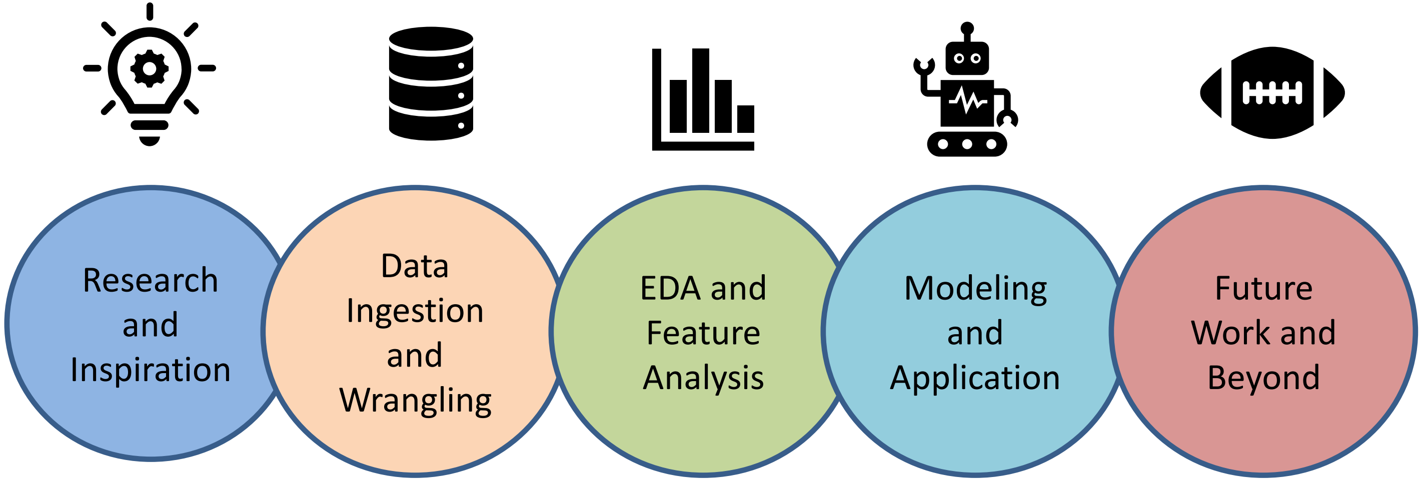
Predicting NFL Game Outcomes

Data Science Certificate - Cohort 17 - Georgetown University

Peter Coiley, Chris Hurd, Chris Whitcomb

Agenda



Research and Inspiration
Data Ingestion and Wrangling
EDA and Feature Analysis
Modeling and Application
Future Work and Beyond

Research and Inspiration

Background Research

The NFL’s Analytics Revolution has Arrived

"Football is still well behind baseball and basketball when it comes to embracing advanced metrics, but teams have made significant progress in recent years. Those who do not adapt will be left behind”

- theringer.com

“The point we made with our coaches is: We have all this information but so does everyone else. What advantage does it give us to get it? None. It’s what we do with it, the way we use it.”

- Kevin Colbert, Steelers general manager

Hypothesis and Motivation

Motivation

- Professional sports are increasingly relying on advanced data analytics and machine learning to provide an edge that will increase their chance of winning.

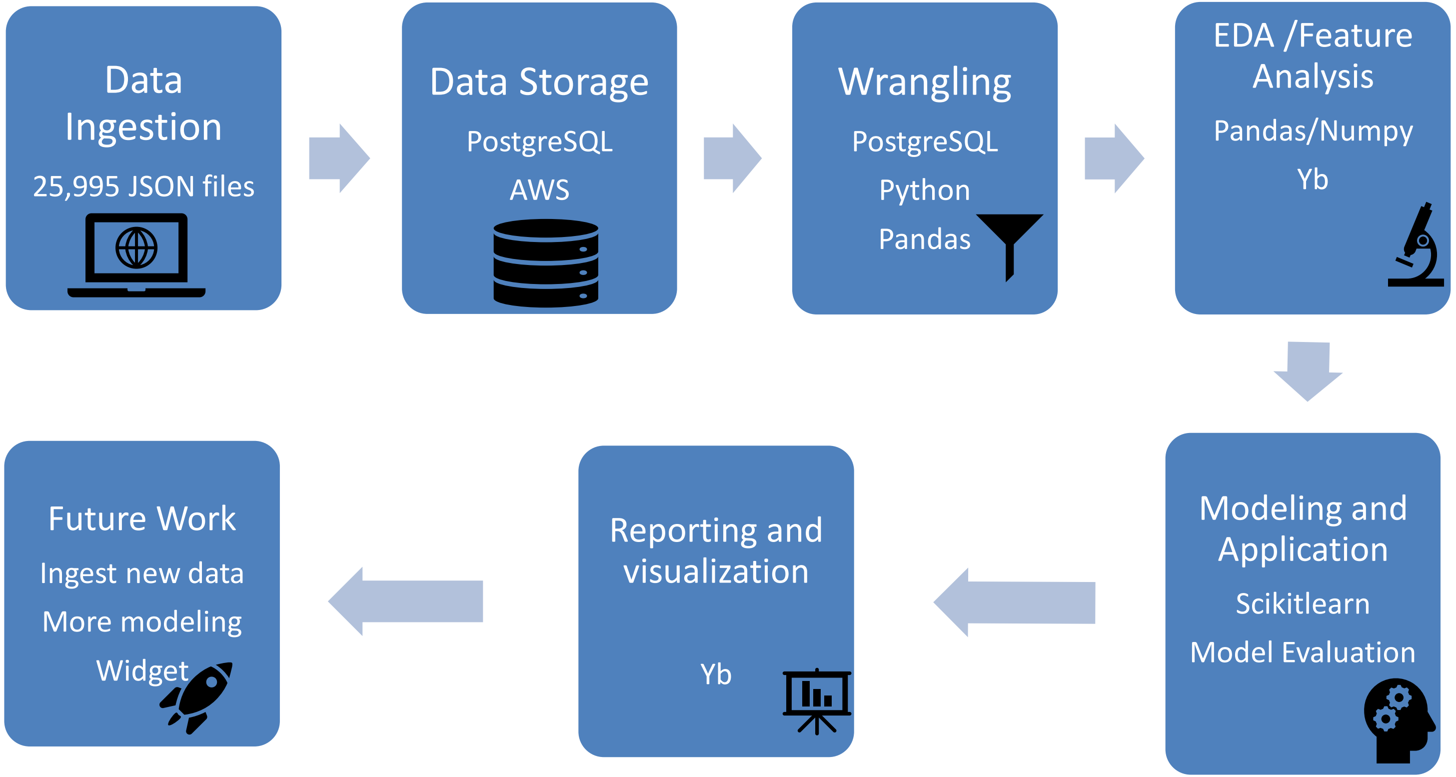
Business Purpose

- Build a data product that can accurately predict the results of NFL games

Hypothesis

- Using data from NFL games between 2002 - 2018, we attempted to determine which features and attributes related to NFL game data best predicted a winning outcome.
- Selected our features from game data that consisted of results from previously played matchups.

Data Science Pipeline



Data Ingestion & Wrangling

Our Data Science Toolkit

Yellowbrick

PostgreSQL

jupyter

NumPy

matplotlib
Version 3.1.1

aws

scikit learn

pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

SQLAlchemy

.py

GitHub

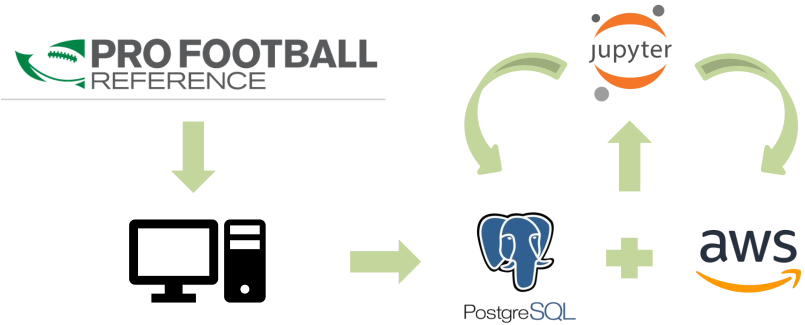
python™

PRO FOOTBALL REFERENCE

Dataset

- Instances: 1,062,249
- Seasons: 69
- Games: 14,176
- Players: 25,995
- Size: 1.15GB

Data Ingestion, Storage, & Wrangling Process




```python
def get_players_for_letter(self, letter):

    """Get a list of player links for a letter of the alphabet.
        Site organizes players by first letter of last name.
        Args:
            - letter (str): letter of the alphabet uppercased
        Returns:
            - player_links (str[]): the URLs to get player profiles
    """

    response = self.get_page(PLAYER_LIST_URL.format(letter))
    soup = BeautifulSoup(response.content, 'html.parser')

    players = soup.find('div', {'id': 'div_players'}).find_all('p')
    player_links = []


    for player in players:

        if len(player.contents[1].split(' ')) == 2:
            player_career_end = int(player.contents[1].split(' ')[1].split('-')[1])

            if player_career_end >= 1950:
                player_links.append(BASE_URL.format(player.contents[0].contents[0].get('href')))
```

Data Wrangling

Merging Data	Organizing Data	Feature Engineering	Encoding
• Transformed our data from 25,995 json files into 1 PostgreSQL table.	• Summed key values to convert player statistics into team statistics • Split games into teams, home team and away team. • Brought home team statistics and away team statistics together into 1 table.	• Used game statistics as a base to create team averages. • Used team averages to develop novel features: Winning %, Point Differential, Turnover Differential, and Power Ranking	• It was necessary to convert multiple fields in our data frame from strings values to numeric values.

PostgreSQL

pandas

Merging the Data

PRO-FOOTBALL-REFERENCE.COM

PostgreSQL

```python
import psycopg2
import json
import os

directory = os.fsencode(r'C:\Users\phoki\Documents\Georgetown\Data\stats_data\stats_data')

conn = psycopg2.connect(database='nfl_stats', user='postgres', password ='georgetown', host='nflstats.cb6meldrm5db.us-east-1.rds.amazonaws.

cur = conn.cursor()

cur.execute("""CREATE TABLE player_logs (
    player_id INTEGER,
    year INTEGER,
    game_id VARCHAR(255),
    date DATE,
    age FLOAT,
    team VARCHAR(255),
    game_location VARCHAR(255),
    opponent VARCHAR(255),
    game_won VARCHAR(255),
    player_team_score INTEGER,
    opponent_score INTEGER,
    passing_attempts INTEGER,
    passing_completions INTEGER,
    passing_yards FLOAT,
    passing_rating FLOAT,
    passing_touchdowns INTEGER,
    passing_interceptions INTEGER,
    passing_sacks INTEGER,
    passing_sacks_yards_lost FLOAT,
    rushing_attempts INTEGER,
    rushing_yards FLOAT,
    rushing_touchdowns INTEGER,
```

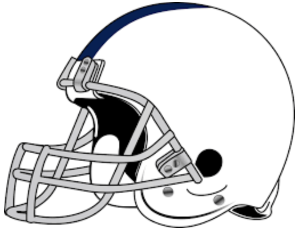
{"player_id": 2415, "year": "2001", "game_id": "200111040sdg", "date": "2001-11-04", "game_number": "8", "age": "22.293", "team": "SDG", "game_location": "H", "opponent": "KAN", "game_won": false, "player_team_score": "20", "opponent_score": "25", "passing_attempts": 15, "passing_completions": 27, "passing_yards": 221, "passing_rating": 94.8, "passing_touchdowns": 1, "passing_interceptions": 0, "passing_sacks": 2, "passing_sacks_yards_lost": 12, "rushing_attempts": 2, "rushing_yards": 18, "rushing_touchdowns": 0, "receiving_targets": 0, "receiving_receptions": 0, "receiving_yards": 0, "receiving_touchdowns": 0, "kick_return_attempts": 0, "kick_return_yards": 0, "kick_return_touchdowns": 0, "punt_return_attempts": 0, "punt_return_yards": 0, "punt_return_touchdowns": 0, "defense_sacks": 0, "defense_tackles": 0, "defense_tackle_assists": 0, "defense_interceptions": 0, "defense_interception_yards": 0, "defense_interception_touchdowns": 0, "defense_safeties": 0, "point_after_attemps": 0, "point_after_makes": 0, "field_goal_attempts": 0, "field_goal_makes": 0, "punting_attempts": 0, "punting_yards": 0, "punting_blocked": 0}

Organizing the Data

Summed Player Date into Team Data, Split Games Into Teams, Brought Everything Back Together…

```
create table ALL_HOME_GAME_STATS as (

select CAST(sum(passing_yards) AS INT) as passing_yards,
CAST(sum(rushing_yards) AS INT) as rushing_yards,
CAST(sum(passing_interceptions) AS INT) as turnovers_given,
CAST(sum(defense_interceptions) AS INT) as turnovers_taken,
player_team_score, opponent_score,
game_id, team, game_won, game_location, season_year from player_logs
where season_year > 2001 and game_location='H'
group by game_id, player_team_score, opponent_score, team, game_won, game_location, season_year )
```

VS



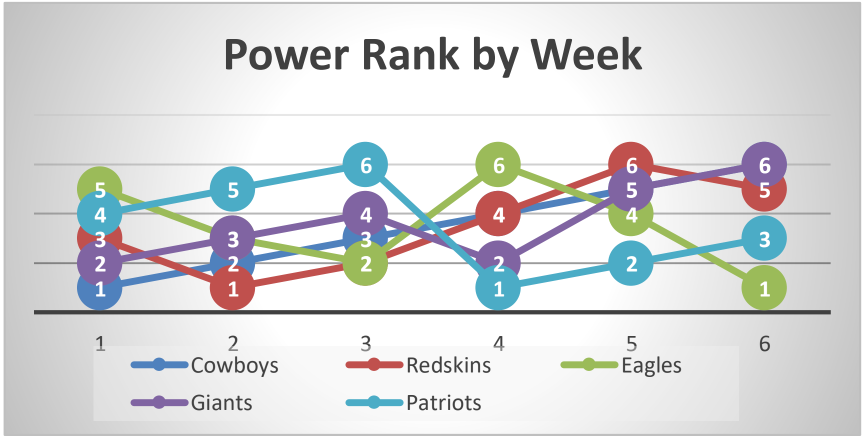
Feature Engineering

Sample Code

```python
def average_stats(data_frame):
    #Changing colums from strings to numerical values in both data frames...
    count = len(data_frame.index)
    index=0
    while index < count:
        data_frame.at[index,"game_id"]=(index+1)
        data_frame.at[index,"game_date"]=(index+1)
        data_frame.at[index,"team"]=1
        data_frame.at[index,"opp_team"]=0
        index+=1

    #Checking for NaN values
    counter = data_frame["week_number"].values.tolist()
    counter.sort()
    ztt = []
    for i in counter:
        if not math.isnan(i):
            ztt.append(i)
    for i in range(len(ztt)):
        ztt[i] = int(ztt[i])

    counter = ztt

    new_df = data_frame.head(4)

    for y in counter[:-1]:
        if y in [1,2,3]:
            continue
        if y >= 4:
            test = data_frame.head(y)
            z = test.mean()
            new_df = new_df.append(z, ignore_index=True)

    return new_df
```

Transformed teams’ game statistics to teams' average statistics. This allowed us to base an instance off of how teams had been performing through a specific point in their season.

Developed novel features to enhance and strengthen our model’s performance.



Encoding

It was necessary to convert multiple fields in our data frame from string values to numeric values.

We did this in multiple ways:

- pandas.DataFrame.at
- pandas.DataFrame.map()

Our largest hurdle came with encoding NaN values!

stackoverflow

Sample Code

```
count = len(data_frame.index)
index=0
while index < count:
    data_frame.at[index,"game_id"]=(index+1)
    data_frame.at[index,"game_date"]=(index+1)
    data_frame.at[index,"team"]=1
    data_frame.at[index,"opp_team"]=0
    index+=1
```

```
# Step 4 change game_location and outcome to a numerical value of 1 and 0
# Home Games will be 1 and Away 0
# Wins will be 1 and Loss will be 0

    for team in team_list:
        final_team_df[team]['game_location'] = final_team_df[team]['game_location'].map({'H': 1, 'A': 0})
        final_team_df[team]['outcome'] = final_team_df[team]['outcome'].map({'true': 1, 'false': 0})
```

DEN[1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. nan]

DEN[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18]

Exploratory Data Analysis

Exploratory Data Analysis Process

Iterative Process

- From raw data, developed initial feature set → Extracted data into Jupyter notebook for initial EDA and initial feature exploration
- Developed novel features into new model feature sets → Examined data and new features
- Developed final feature set → Final EDA modeling and feature exploration

Initial Data Analysis - Overview

Started with 22 features - including rankings and novel feature “power ranking”

→

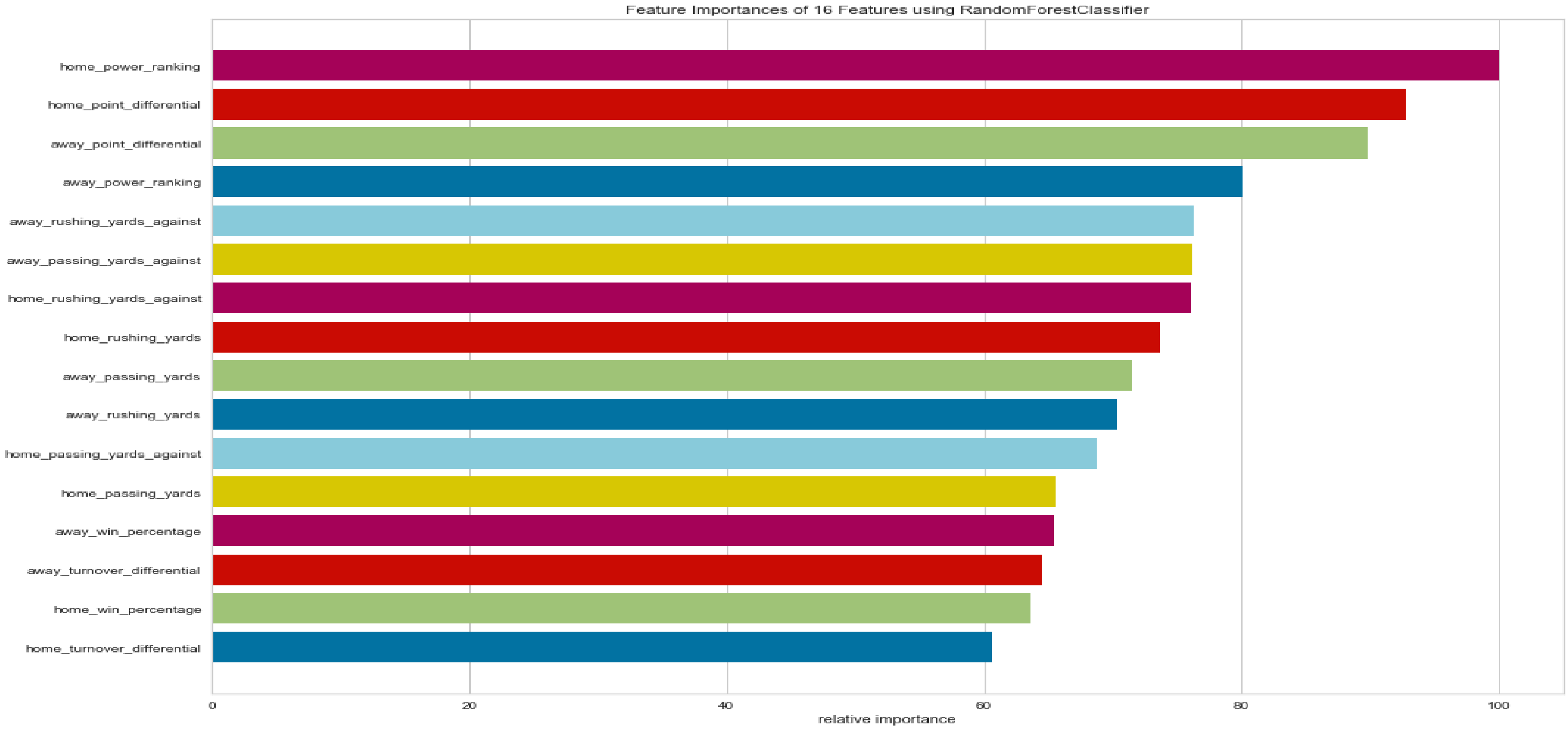
Based on initial analysis decided to combine multiple features and created 2 new features point differential and turnover differential.

→

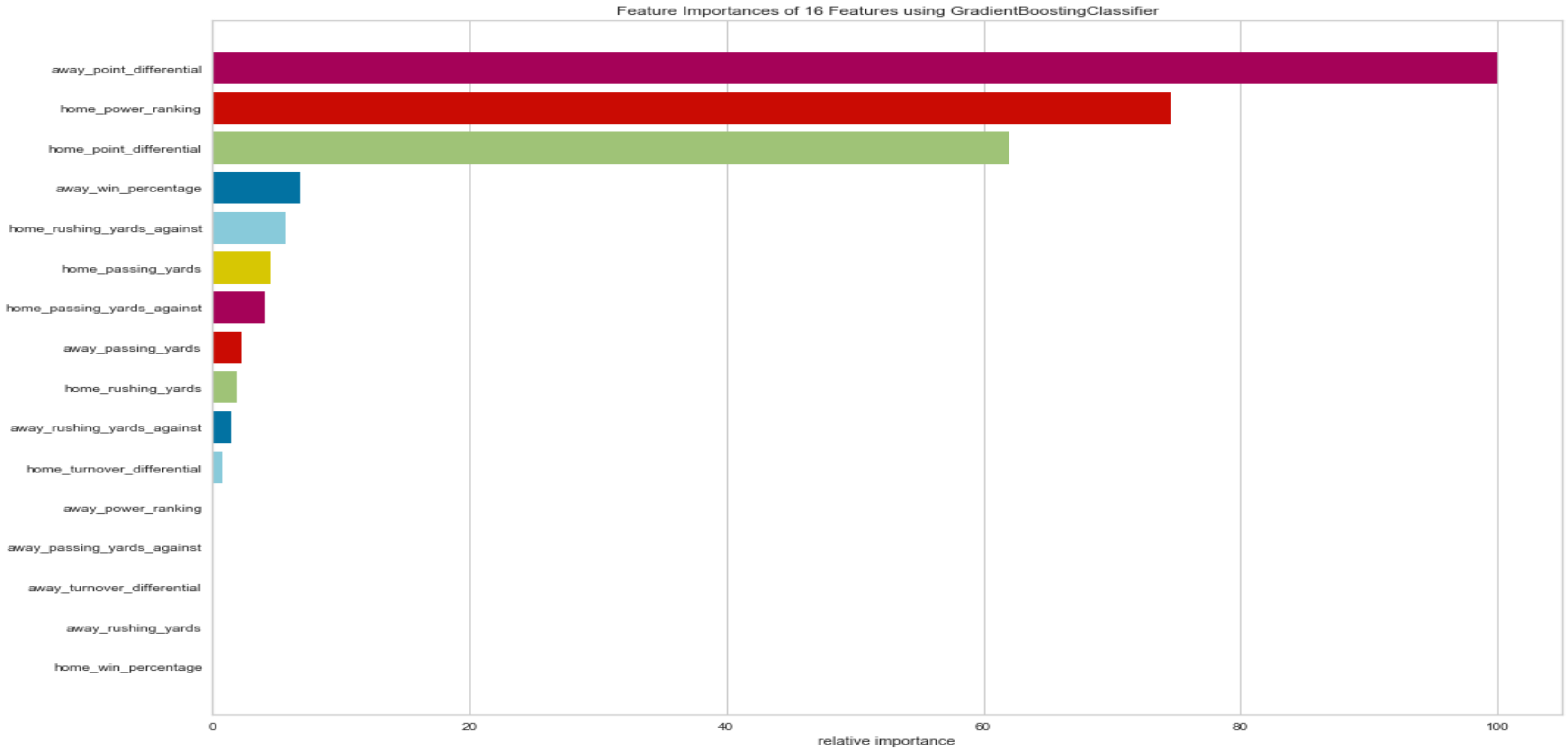
Core Features for each team:

1. Rushing Yards For
2. Rushing Yards Against
3. Passing Yards For
4. Passing Yards Against
5. Point Differential
6. Turnover Differential
7. Power Ranking
8. Winning Percentage

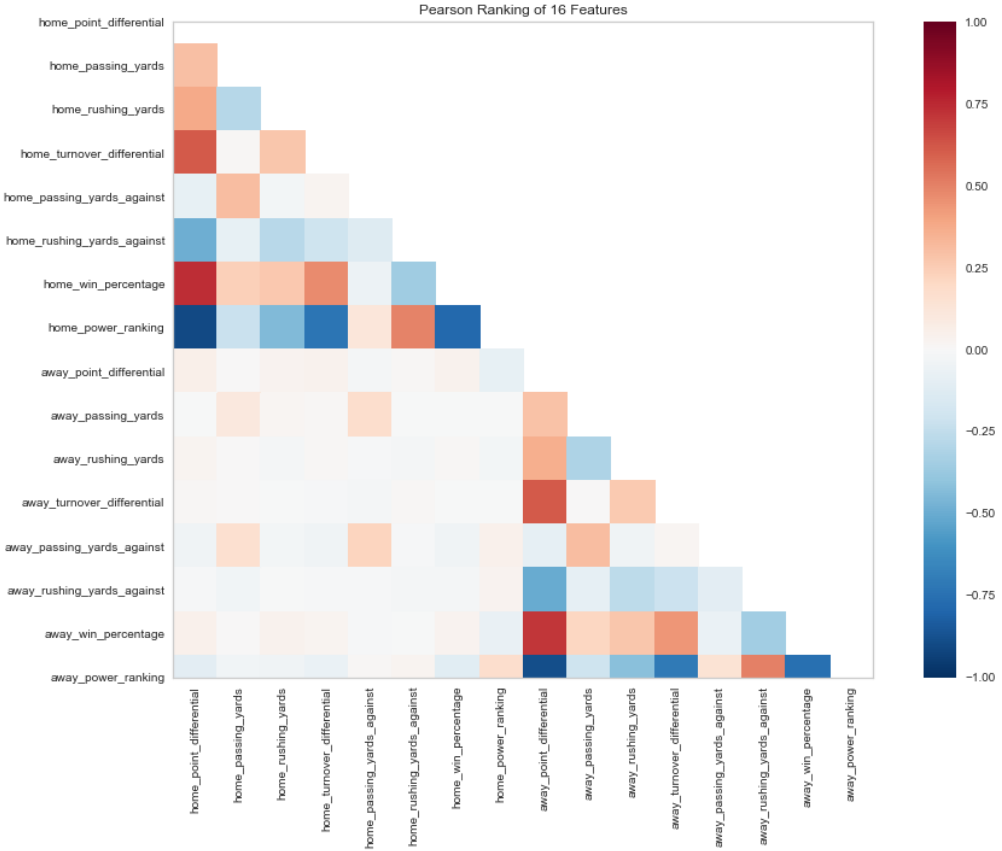
Feature Importance - Random Forest



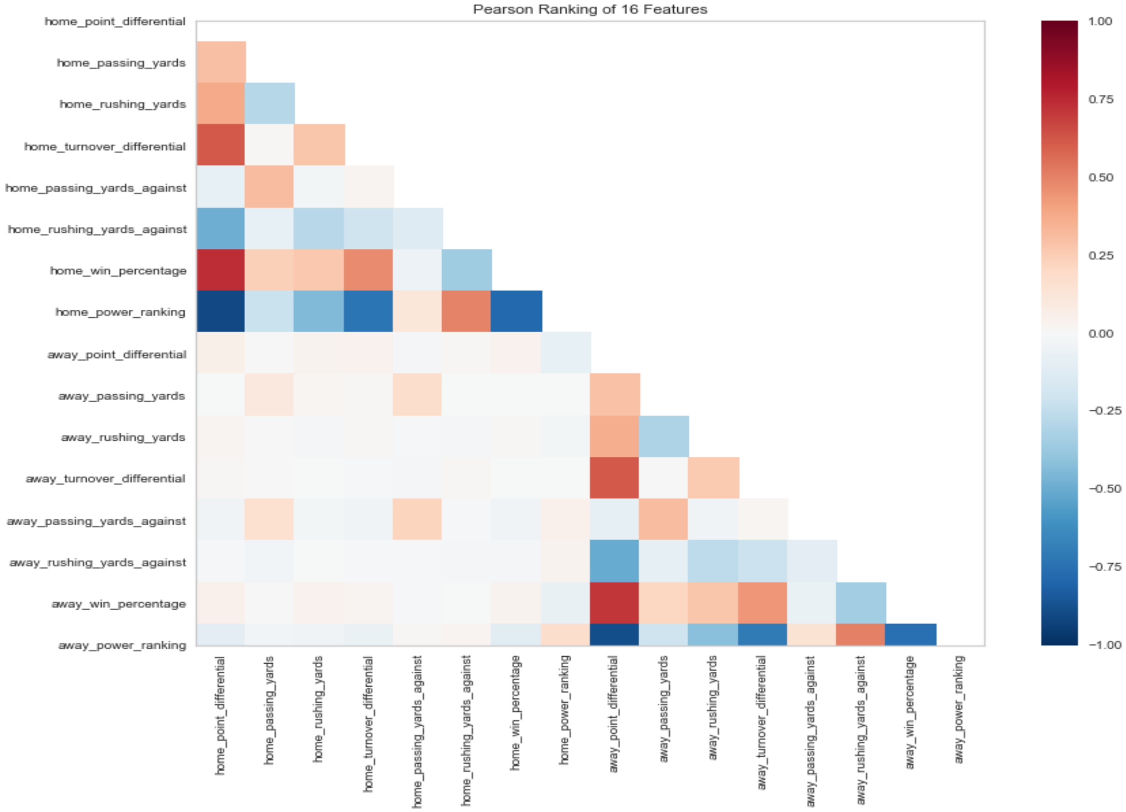
Feature Importance - Gradient Boosting



Feature Analysis - Pearson's Correlation for Random Forest



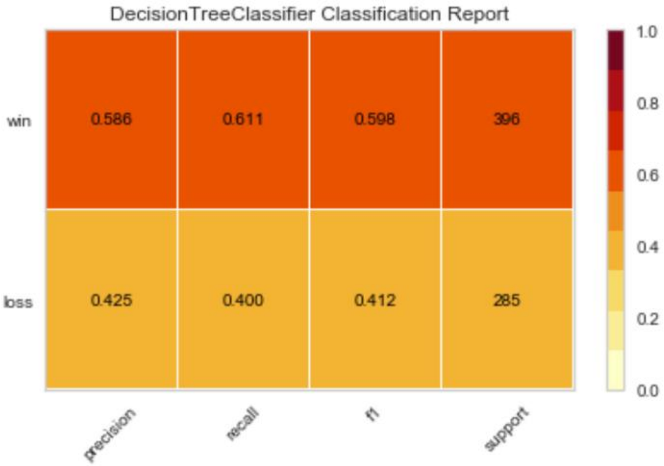
Feature Analysis - Pearson's Correlation for Gradient Boosting

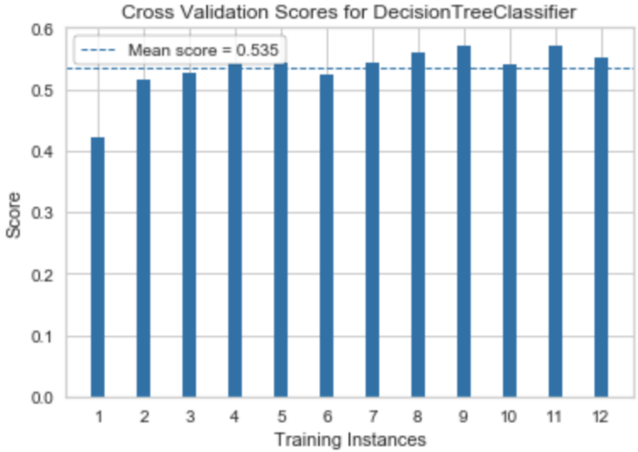


Modeling & Application

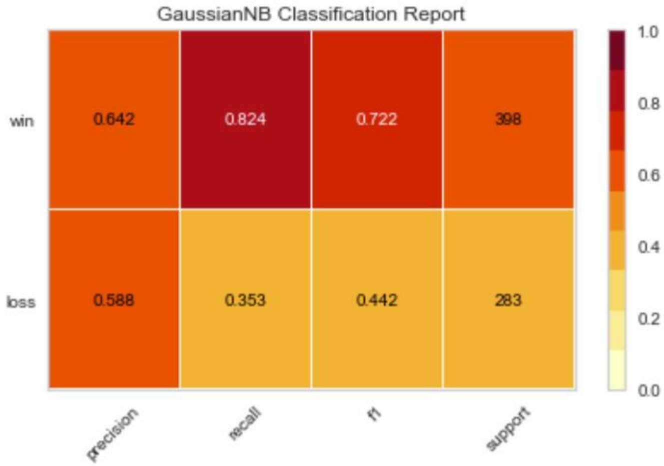
Classification Reports and CV – Baseline (examples)

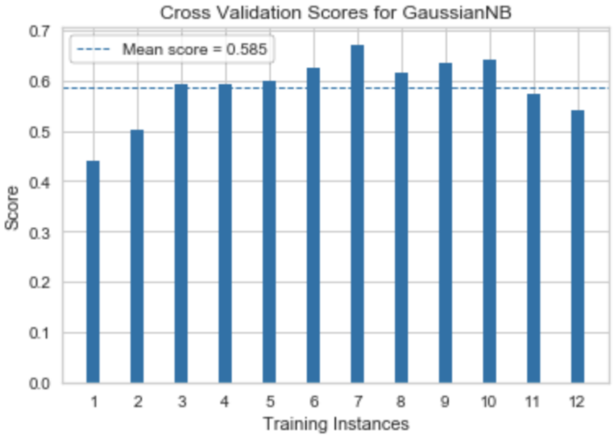
Decision Tree





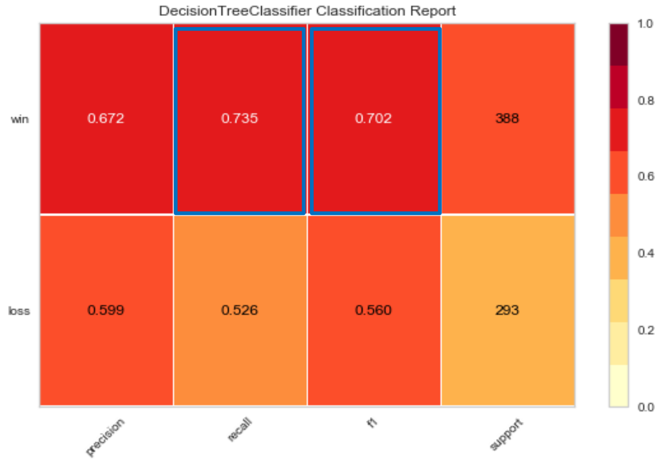
Gaussian NB



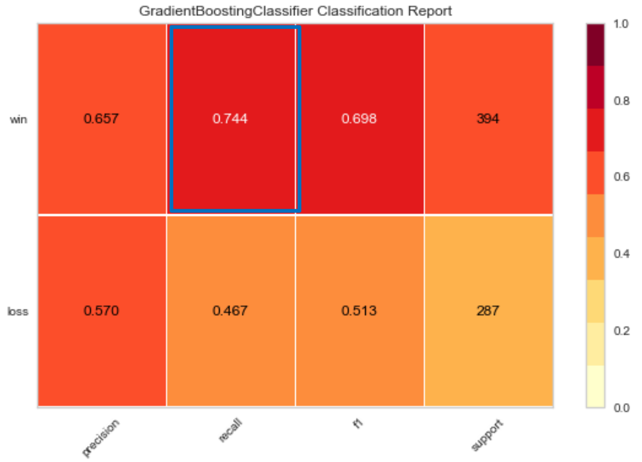


Classification Reports - Final Features

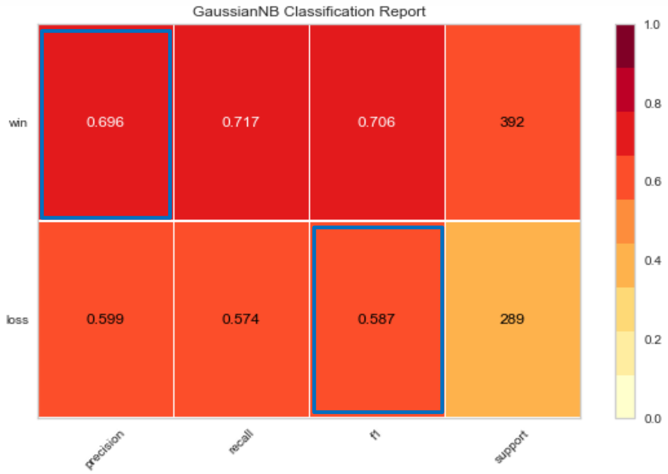
Decision Tree



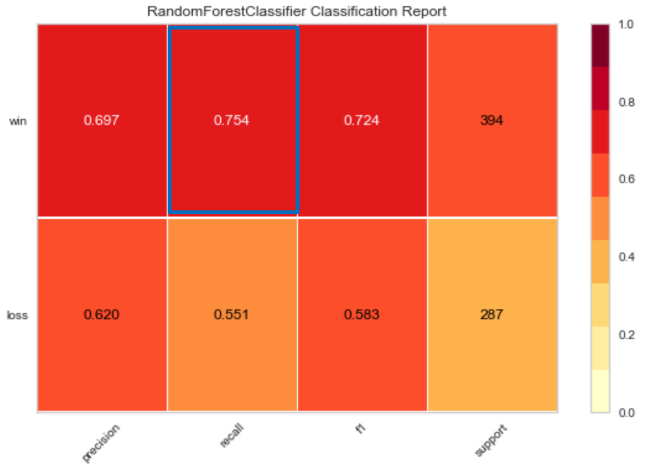
Gradient Boosting Classifier



Gaussian NB

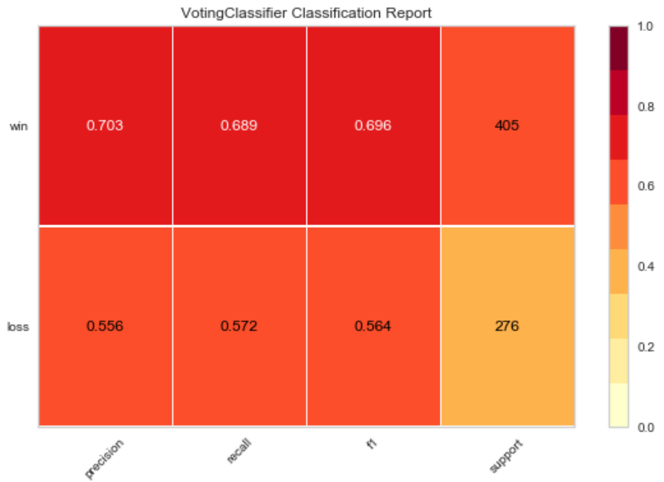


Random Forest Classifier

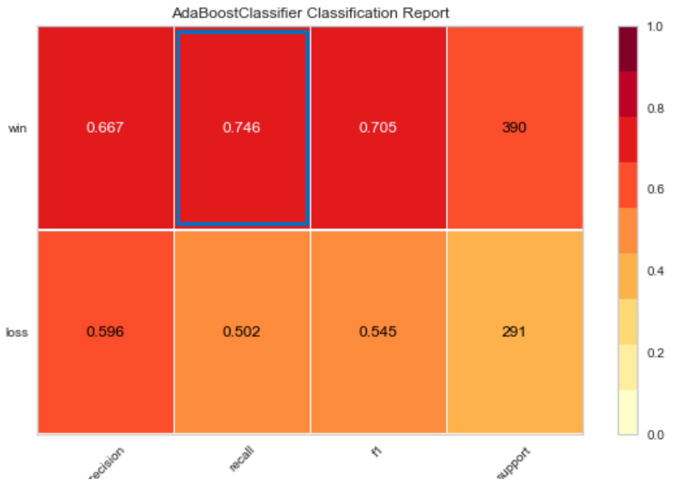


Classification Reports – Final Features

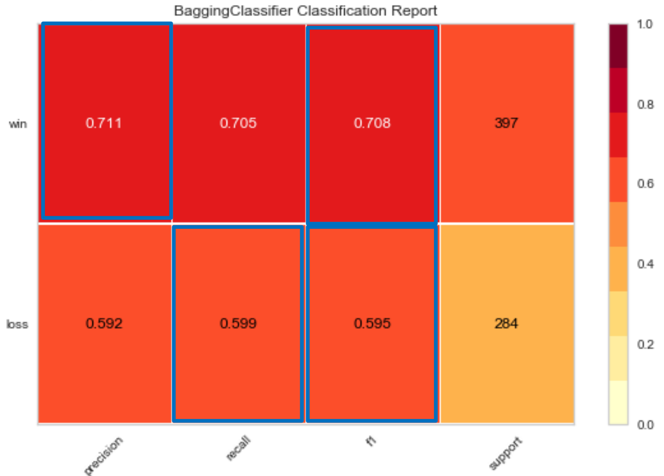
Voting Classifier



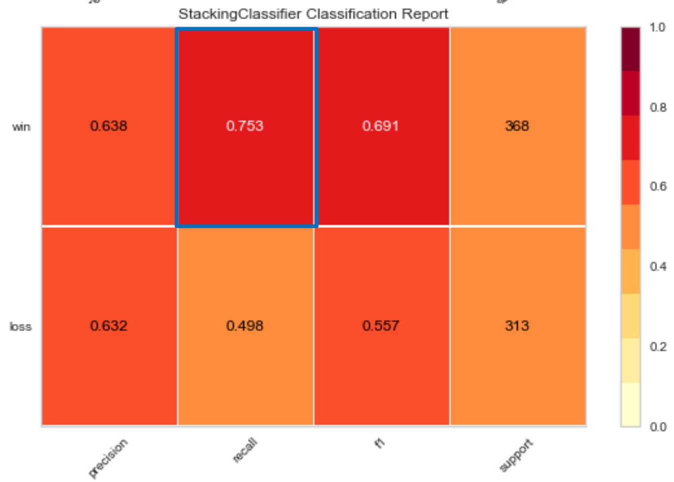
AdaBoost Classifier



Bagging Classifier

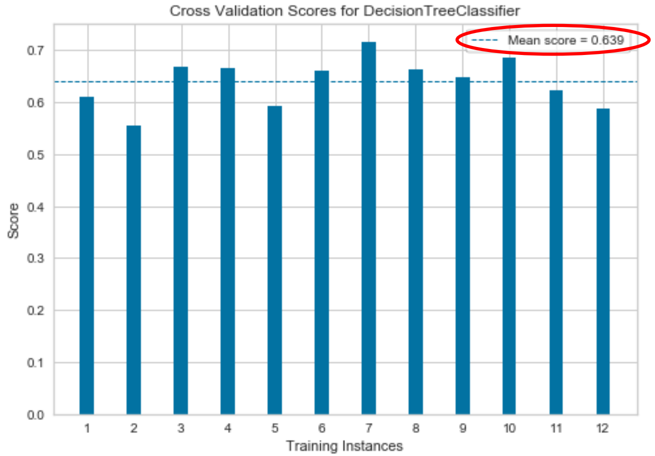


Stacking Classifier

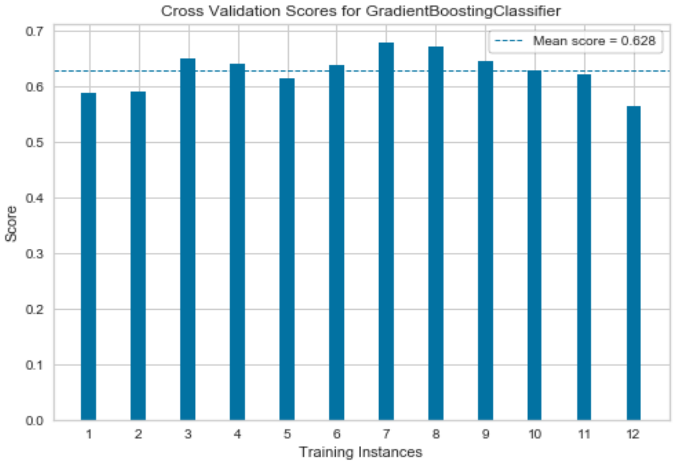


Cross Validation

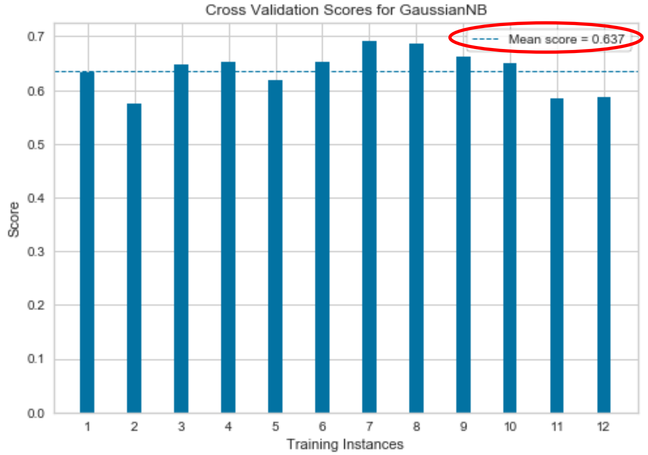
Decision Tree



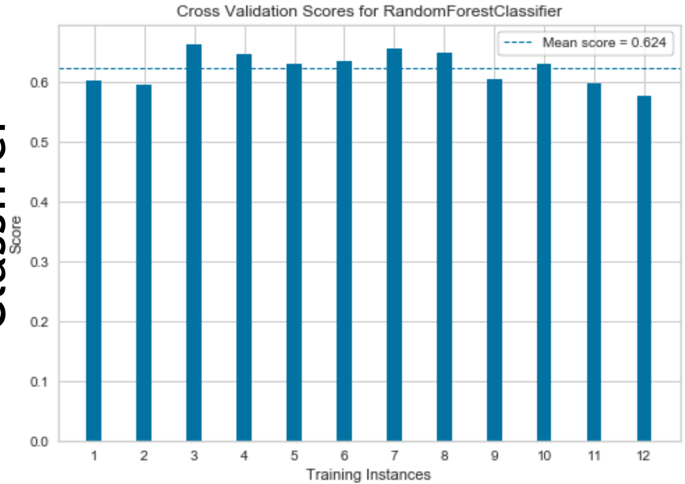
Gradient Boosting Classifier



Gaussian NB

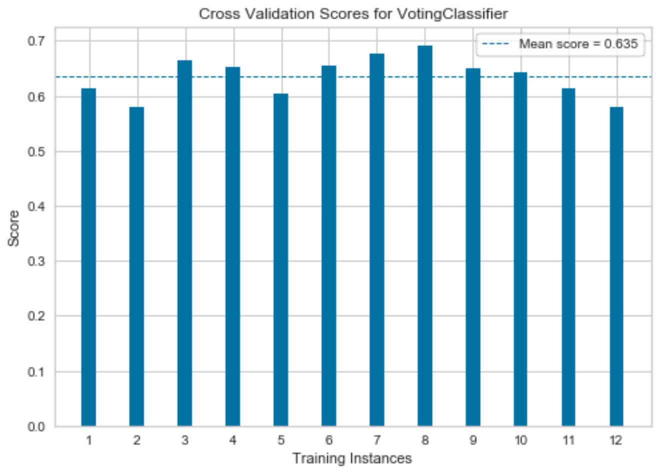


Random Forest Classifier

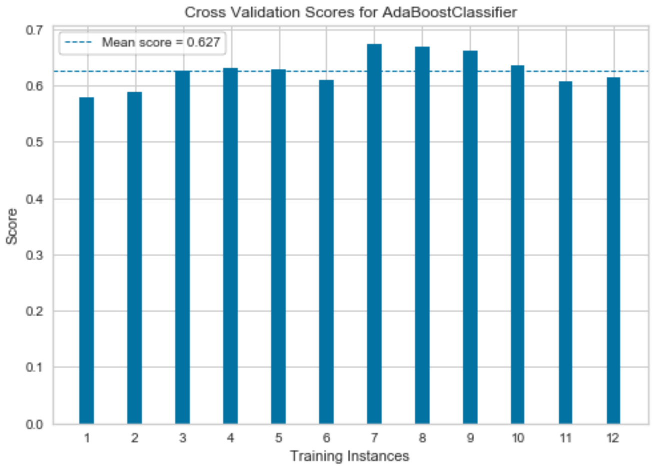


Cross Validation

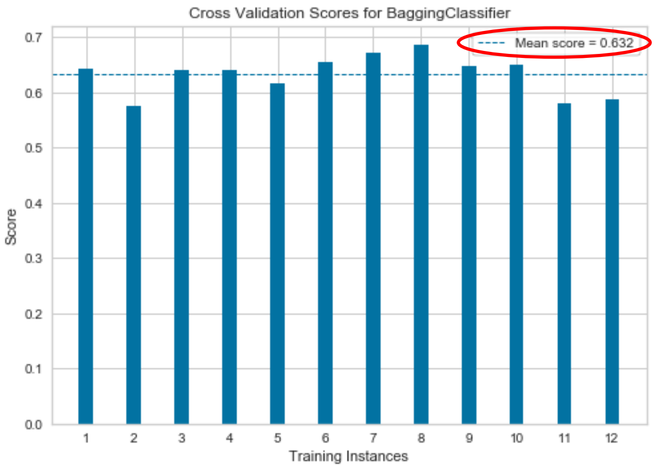
Voting Classifier



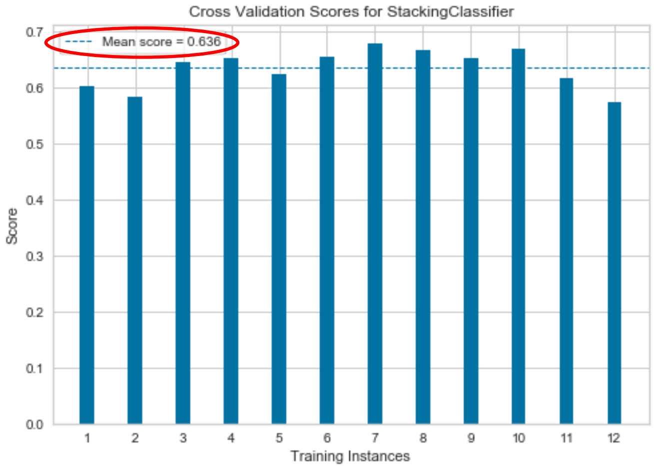
AdaBoost Classifier



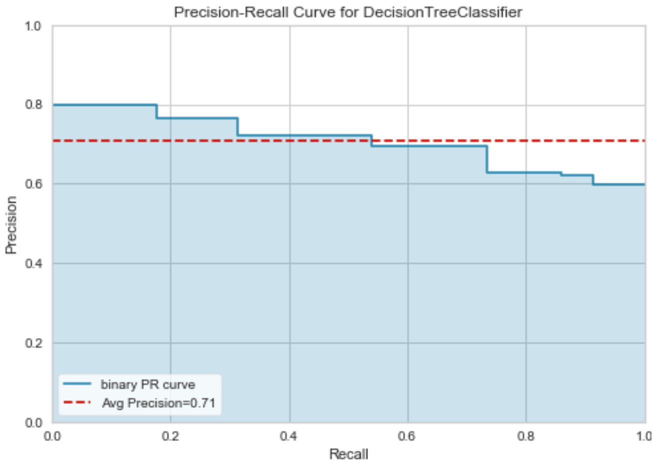
Bagging Classifier

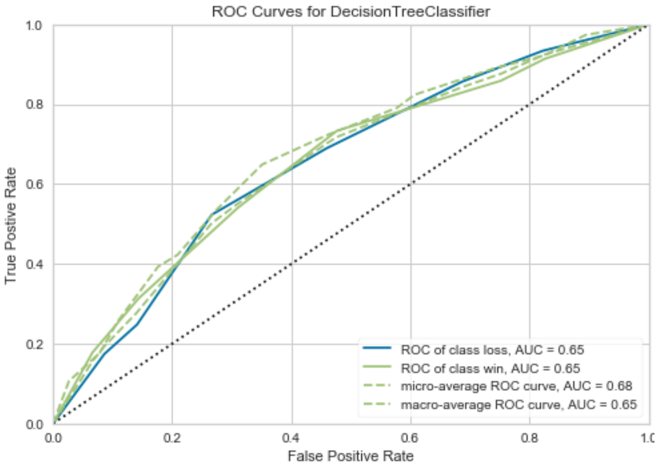


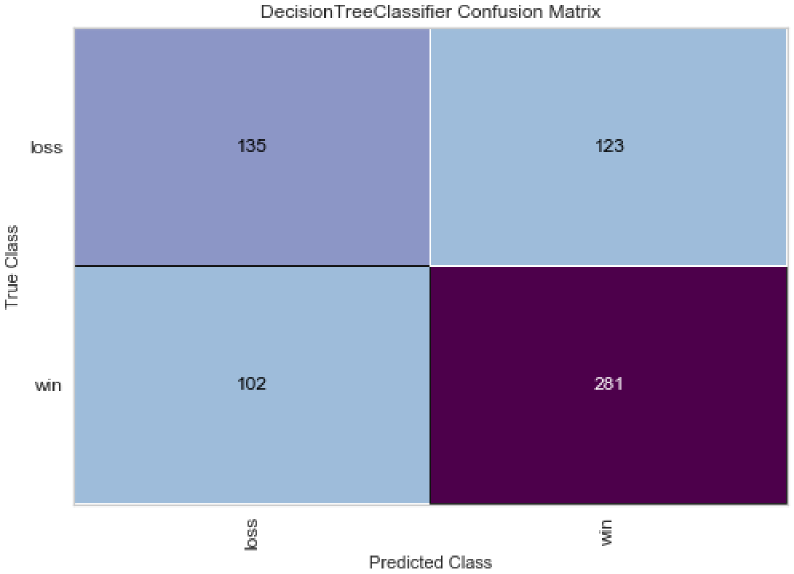
Stacking Classifier



Decision Tree Classifier: 2002-2017 Data



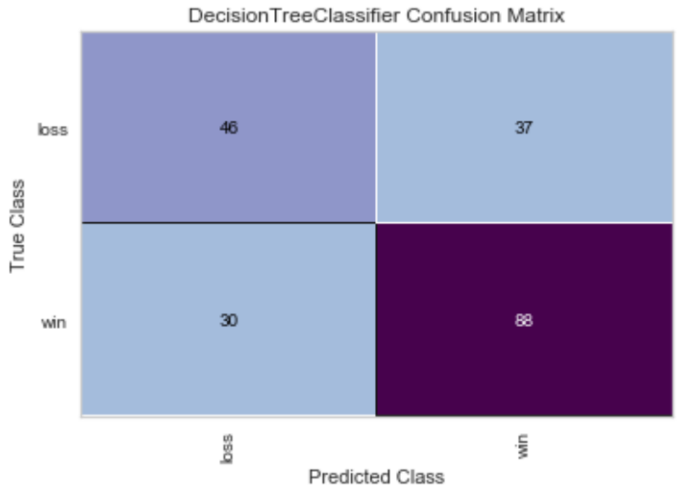


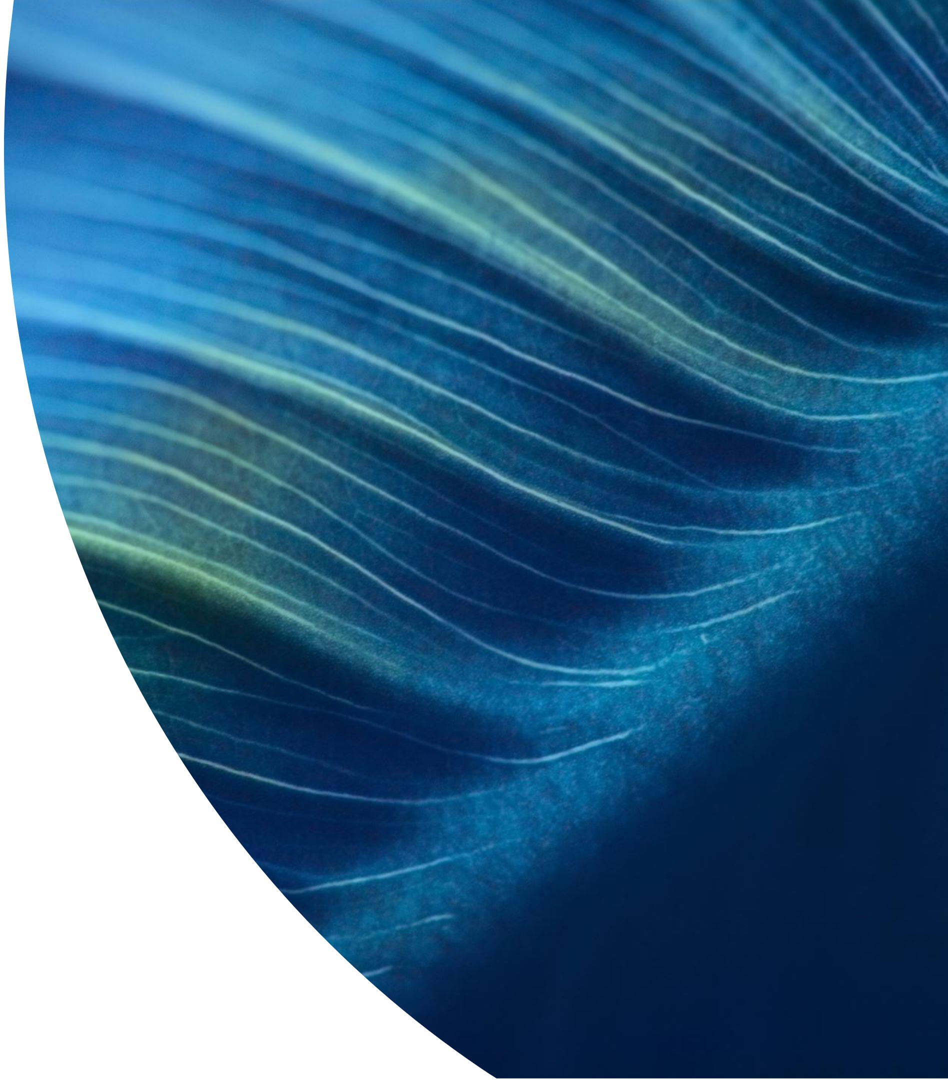


Decision Tree Classifier: Predicting on 2018 Data

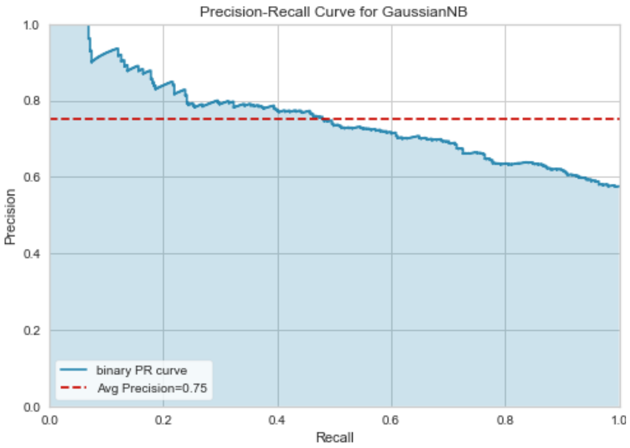
We used the model to predict the 2018 results

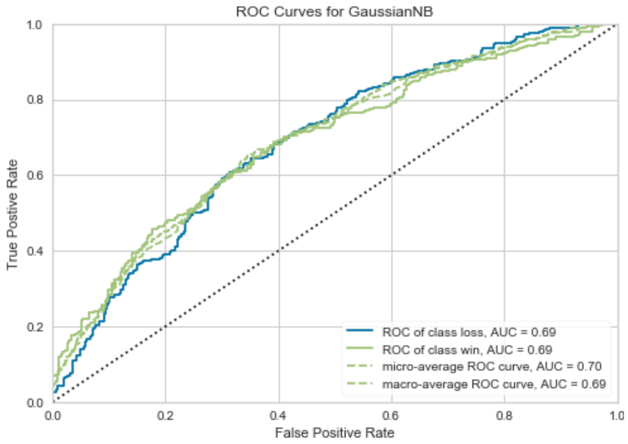
The results show that we have an overall **accuracy of 66.6**%. The **recall rate** for predicting wins is 74.6%

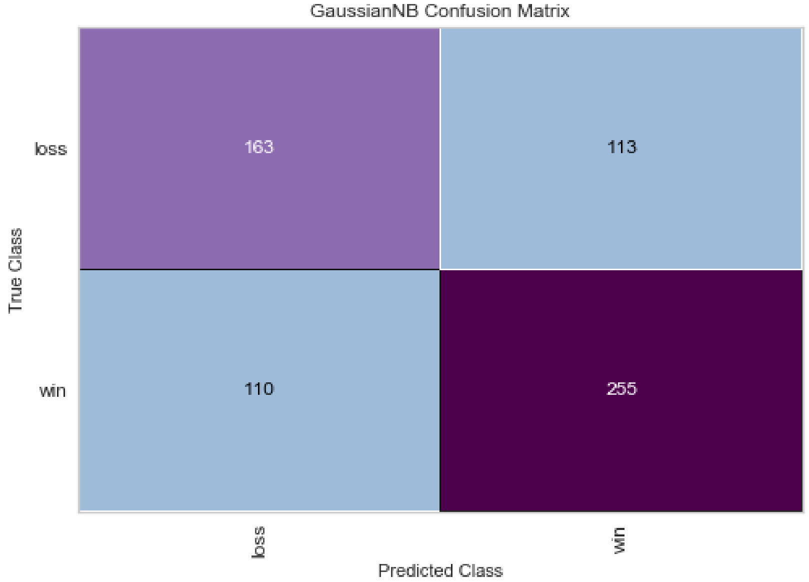




Gaussian NB: 2002-2017 Data



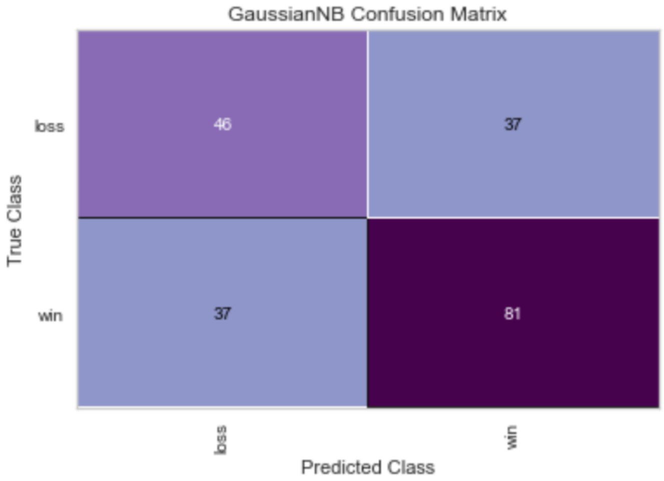


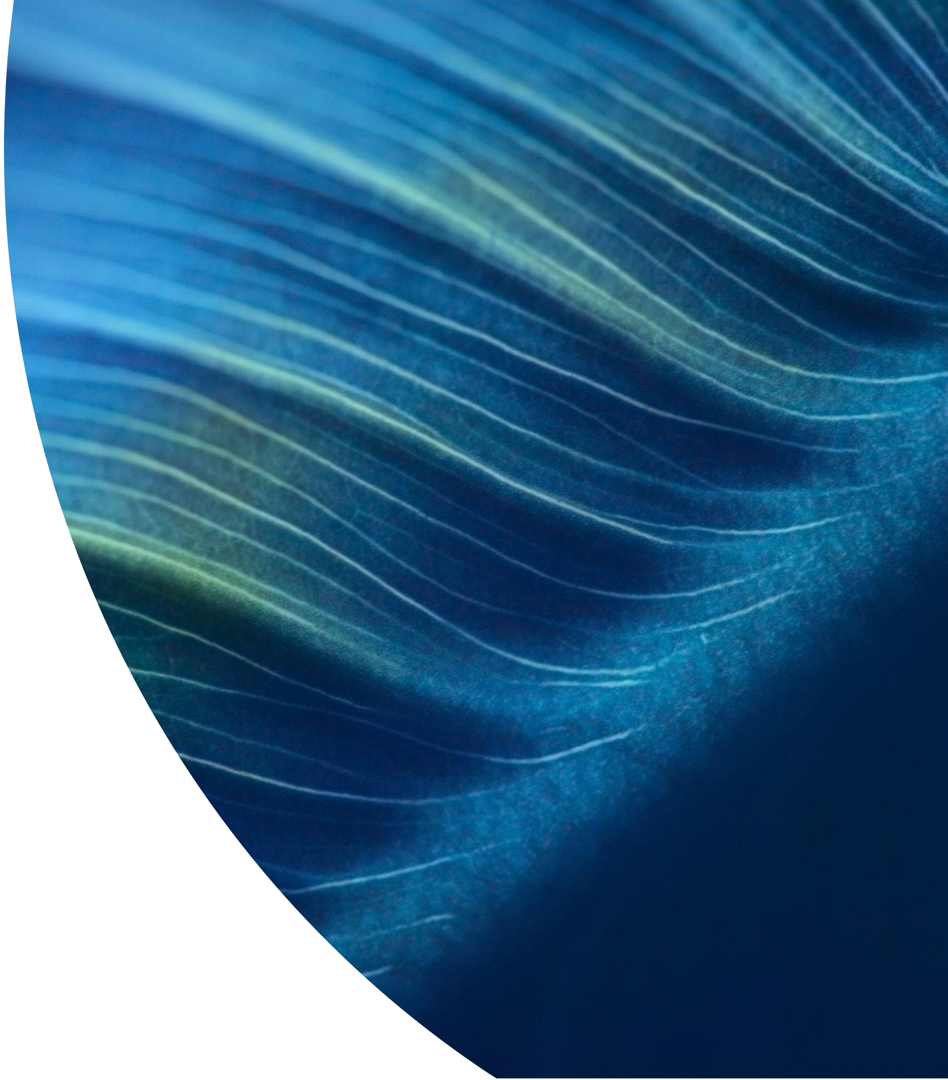


Gaussian NB: Predicting on 2018 Data

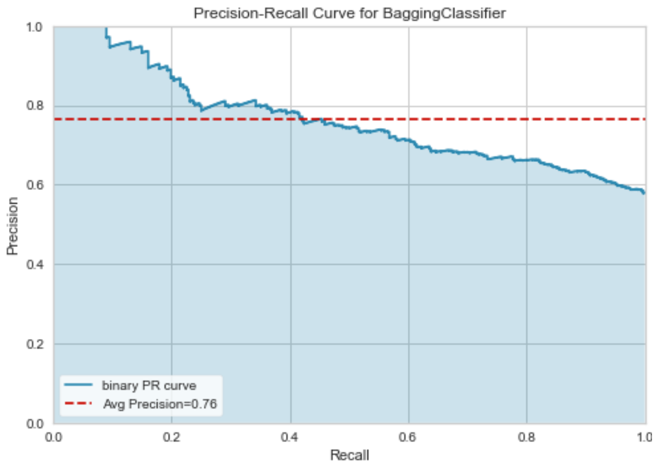
We used the model to predict the 2018 results

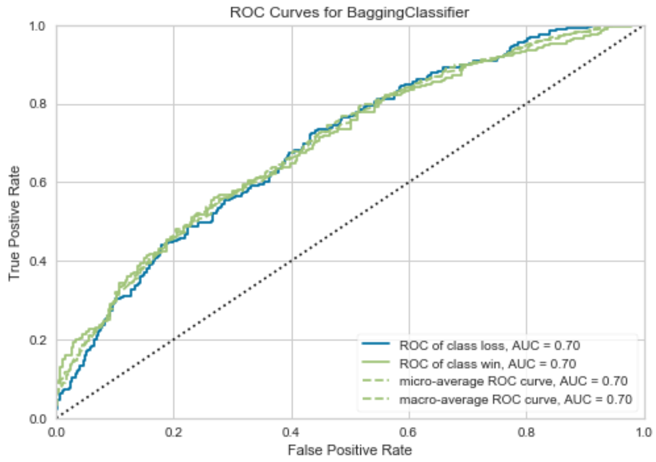
The results show that we have an overall **accuracy of 63.2**%. The **recall rate** for predicting wins is 68.6%

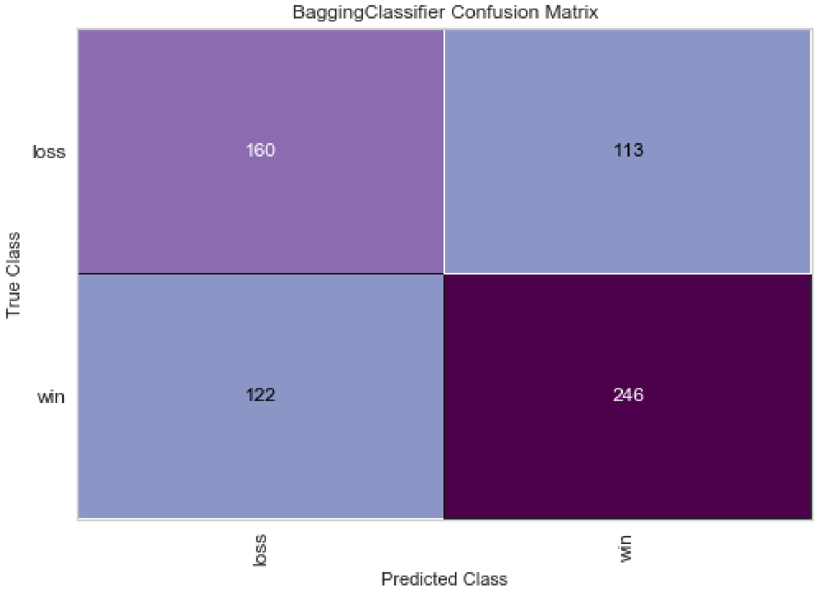




Bagging Classifier: 2002-2017 Data



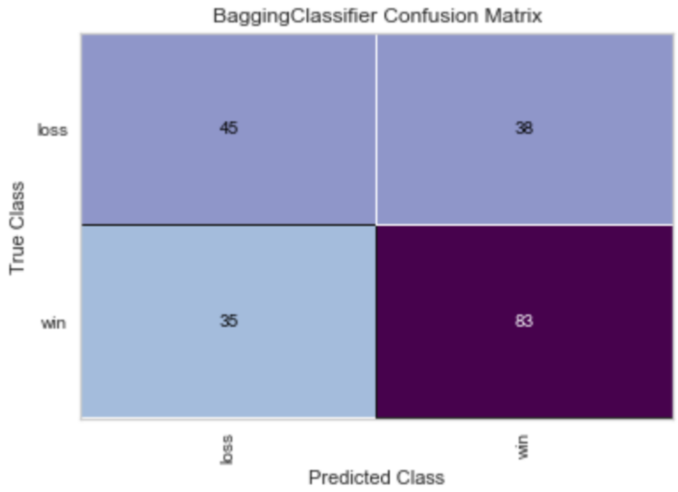


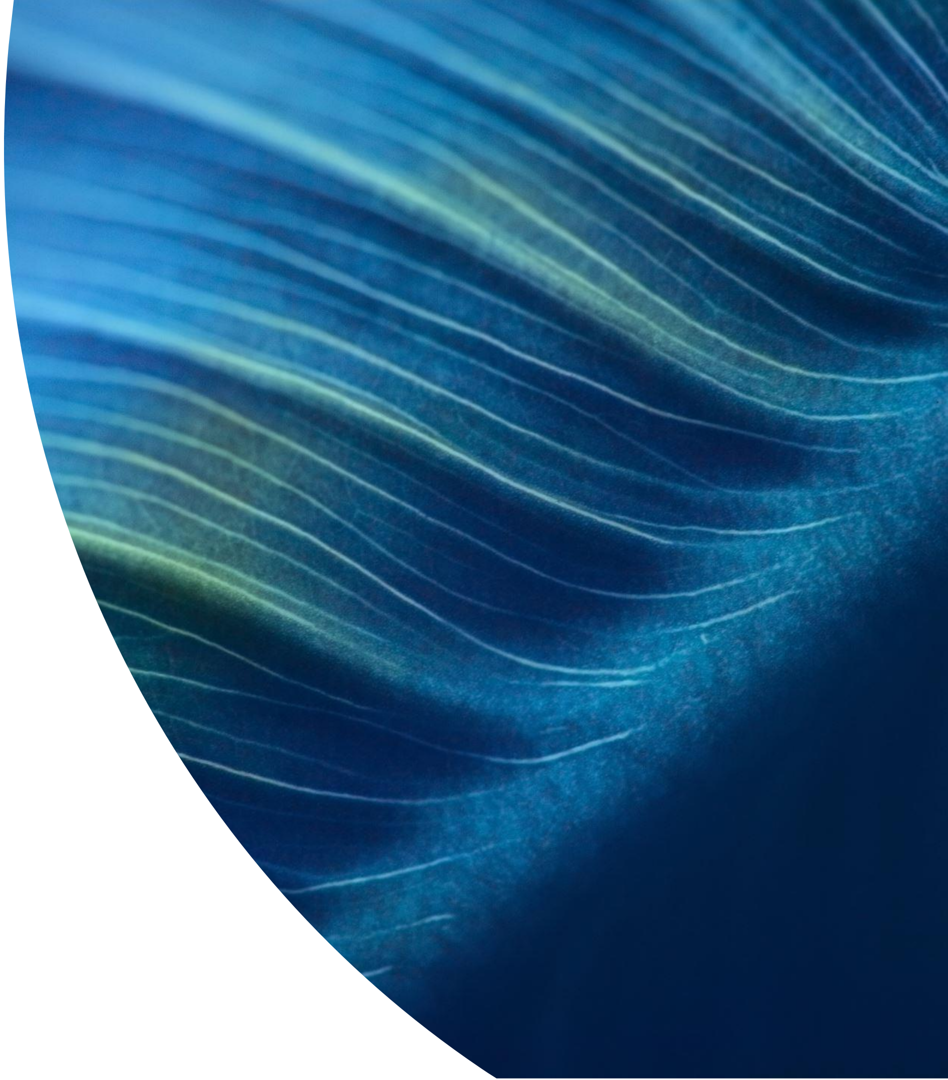


Bagging Classifier: Predicting on 2018 Data

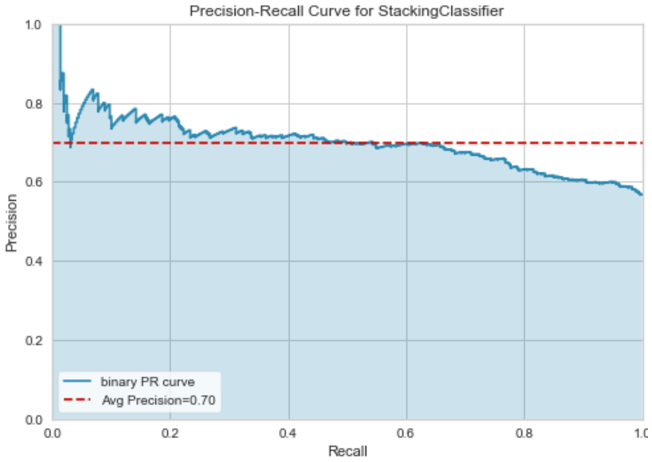
We used the model to predict the 2018 results

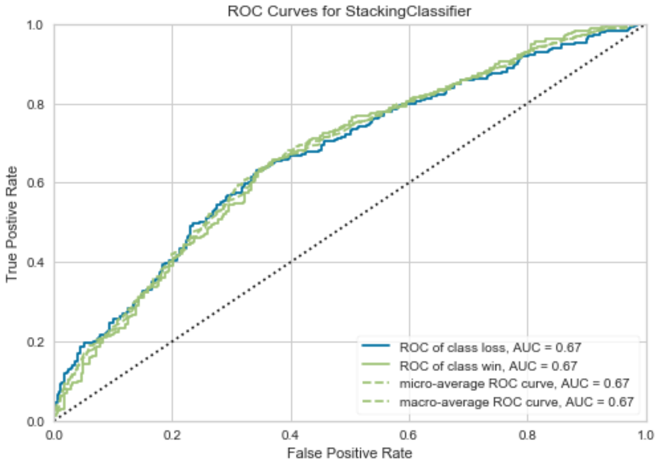
The results show that we have an overall **accuracy of 63.7**%. The **recall rate** for predicting wins is 70.3%

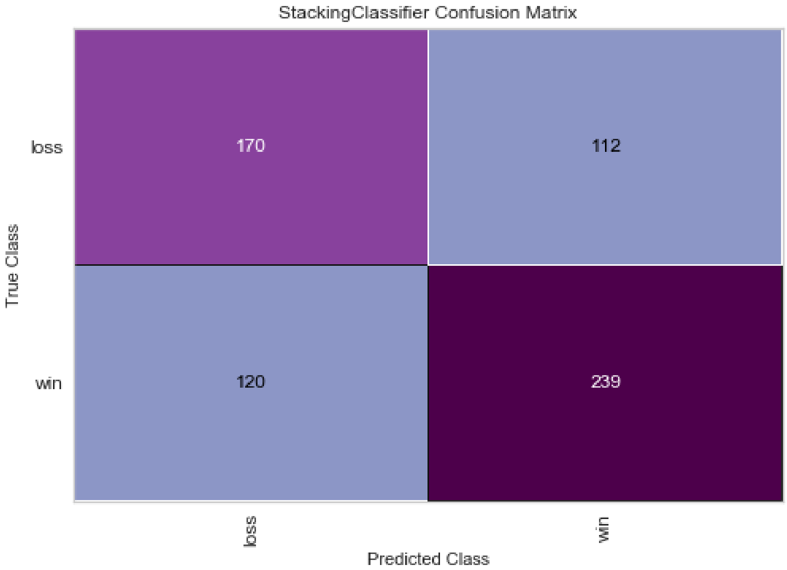




Stacking Classifier: 2002-2017 Data



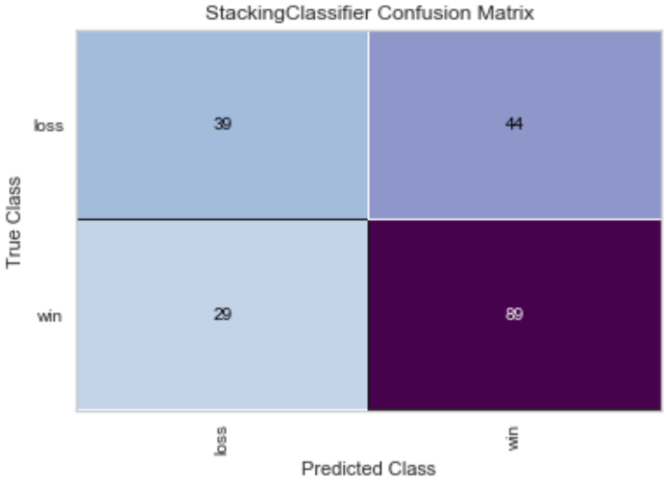


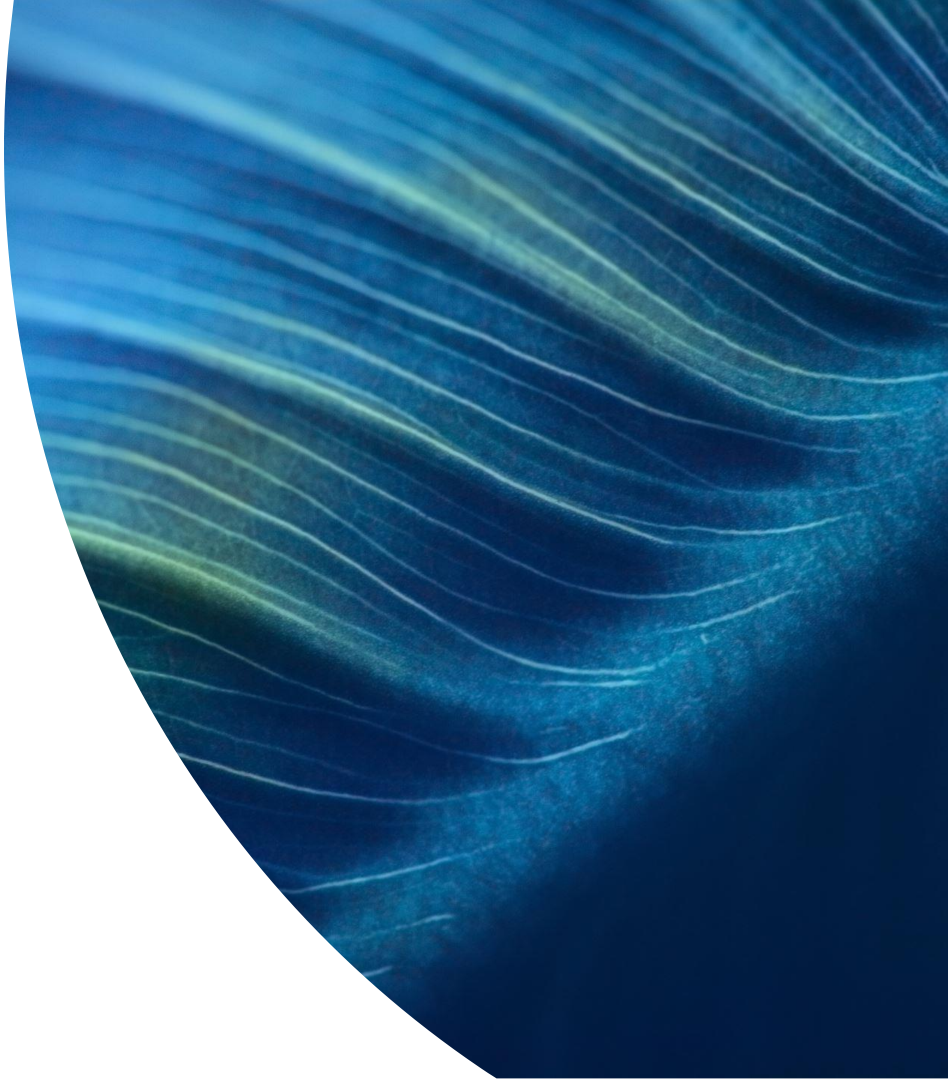


Stacking Classifier: Predicting on 2018 Data

We used the model to predict the 2018 results

The results show that we have an overall **accuracy of 63.7**%. The **recall rate** for predicting wins is 75.4%





Future Direction and Limitations

Future Direction and Lessons Learned

Future Direction

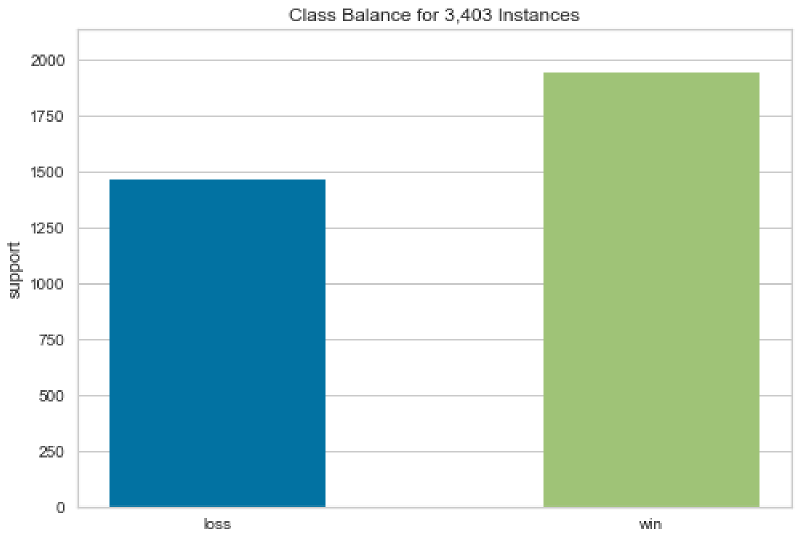
- Acquire more data
  - Weather, dome vs. outdoors, increased granularity
- Improve current data quality
  - Lacked fumble data due to error with PFR website scraping
- Factor in player attributes
  - How many All-Pros / Pro Bowlers? Average age?

Lessons Learned

- Remember to ask questions when it goes beyond level of ability
- Utilize version control better
- Project management and task delegation
- Strategy for teaching others as you go to bring everyone along

Limitations

- Home_outcome = imbalance of wins vs. losses (Apparent bias in predicting wins over losses?)
- No fumble data meant turnover_differential only partly helpful



Thank you to all the instructors and everyone involved. The experience was truly invaluable. Next stop Vegas baby!

Q&A

Modeling Demonstration

if there is time