NFL Data Analysis

Jenna, Brendan, Jacob, Maxwell

Project Overview

- **Goal:** Create data visualizations to help answer group hypotheses and allow curious users to examine historical NFL data.
- **Project Kickoff**:
  - Decided on the Data Visualization Track for project 3.
  - Initial idea for data source was ESPN fantasy football data. Determined the extracting and transforming of the data would be too time consuming (ESPN makes it difficult to access site data).
- **Team roles and responsibilities**:
  - Database/Server- Brendan
  - Web Visualizations- Jenna/Jacob/Maxwell

NFL Data

- Utilized Kaggle data set that contained over 16k games going back to the commencement of the NFL.
- Data overview:
  - A “teams” table contained all teams and their home stadiums. Included both current and former teams that may have changed names or locations (i.e. Houston Oilers are now Tennessee Titans, etc.)
  - A “games” table that contained, date, schedule, scores, betting data (spread/over under) and weather data including temperature and wind speed.

Data Flow/Technologies Used

1. Data source, two csv's from Kaggle (games & teams data)
2. Loaded into Pandas dataframes- minor transformations made: Removed pre-2002 data, edited column names, changed some data types for usability (dates)
3. Loaded data into an sqlite file, one table for each csv.
4. Created flask server to serve games and teams data via API.
5. Created 3 separate utilizing accessing these api endpoints.
6. **Special Technology used:** Chart.js. This is a Javascript library that is an alternative to Plotly.

Hypotheses

- **Weather Impacts**:
  - How do teams from warm-weather states fair in the cold? (Under 32 degrees)
- **Scoring Margin Differences**:
  - Has the margin of victory gotten slimmer over time?
- **Home Wins vs. Losses**:
  - How do teams fair at their home stadium?

Demo Time!

Server-Side: Lessons Learned (Brendan)

- Lesson 1: Always set up the backend APIs completely before starting front end work.
- Lesson 2: Don't assume python libraries play nice with each other; sqlite file loading was difficult. Ultimately I had to abandon automap base and use SQL query strings to obtain the data.
- Lesson 3: Ensure as much data transformation as possible happens on the server side.

```
#Connect to database to get data.
conn = sqlite3.connect('Resources/football.db')

games = pd.read_sql('SELECT * FROM games', conn)
print(games.head())

teams = pd.read_sql('SELECT * FROM teams', conn)
print(teams.head())

warm_teams = pd.read_sql('SELECT * FROM games WHERE weatherTemp
print(warm_teams.head())

games['scheduleDate'] = pd.to_datetime(games['scheduleDate'])
games = games[games['scheduleDate'] >= '2002-06-01' ]
```

Lessons Learned! (Jacob)

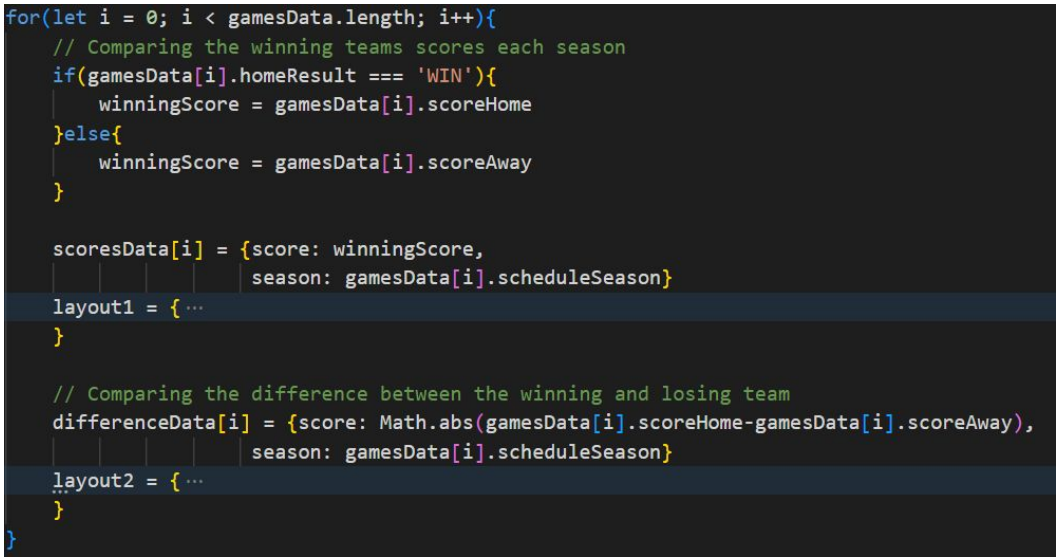
- Combining code for a custom dropdown menu while implementing a javascript library I was unfamiliar with was challenging.
- Using code from several sources and attempting to get them to work together lead to a lot of debugging issues.
- One of the main issues was getting the data to properly populate within an updating pie chart.
- I was finally able to get it working with Thomas' help.

https://youtu.be/XCAUnA4FyHU?si=wVOjLIVkcH8EHa8K
https://youtu.be/5-ptp9tRApM?si=w0z61DQMi1xhiYbN
https://cdnjs.com/libraries/font-awesome
https://fontawesome.com/search?ic=free

```
87              let input = teamName;
88              console.log(input);
89
90              const table = data.split('\n').slice(1);
91              let wins = table.filter(function(row){
92                const columns = row.split(',');
93                const date = columns[0];
94                const teamScore = columns[6];
95                const team = columns[7];
96                const gameResult = columns[9];
97
98                return gameResult === "WIN" && team === input;
99              }).length
100             console.log(`${wins}`);
101
102
103             let loss = table.filter(function(row){
104               const columns = row.split(',');
105               const date = columns[0];
106               const teamScore = columns[6];
107               const team = columns[7];
108               const gameResult = columns[9];
109
110               return gameResult === "LOSS" && team === input;
111             }).length
112             console.log(`${loss}`);
113
114             myChart.data.datasets = [{data: [wins, loss]}];
115             myChart.update();
```

Lessons Learned! (Max)

- Learning the scope of a “var” vs “let”.
- Condensing loops that are pulling data from the same object.
- Figuring out where Plotly attributes went inside of variables.



```
for(let i = 0; i < gamesData.length; i++){
    // Comparing the winning teams scores each season
    if(gamesData[i].homeResult === 'WIN'){
        winningScore = gamesData[i].scoreHome
    }else{
        winningScore = gamesData[i].scoreAway
    }

    scoresData[i] = {score: winningScore,
                     season: gamesData[i].scheduleSeason}
    layout1 = {···
    }

    // Comparing the difference between the winning and losing team
    differenceData[i] = {score: Math.abs(gamesData[i].scoreHome-gamesData[i].scoreAway),
                     season: gamesData[i].scheduleSeason}
    layout2 = {···
    }
}
```

Lessons Learned!(Jenna)

- Initially, I was including too many elements in my for loop, causing the data to count the amount of wins and losses multiple times
- I had difficulties placing the destroy function for the chart
- The ultimate goal of this dashboard was to visualize how much home field advantage might affect the outcomes of simply wins vs losses by team

```
function updateChart(TeamName) {
  d3.json(gamesapi).then(gameData => {
    console.log("Games Data:", gameData);

    let wins = 0;
    let losses = 0;
    let ties = 0;

    for (let i = 0; i < gameData.length; i++) {
      let game = gameData[i];

      if (game.teamHome === TeamName) {
        if (game.homeResult === 'WIN') { wins++ }
        if (game.homeResult === 'TIE') { ties++ }
        if (game.homeResult === 'LOSS') { losses++ }
      }
    }

    if (lineChart) {
      lineChart.destroy();
    }
```

Questions?