DTSC 691
Machine Learning
Project Proposal
Nicole Gordon

Goals of the project

This project aims to combine machine learning with software development to create a web application that allows users to easily interact with a trained model. The machine learning model will predict the outcome of NFL games based on a subset of in-game statistics. The model will take in the current game conditions (eg. current score, time remaining, yards to go, etc.) as predictors and the response will be which team is more likely to win the game. This type of model would be useful for both sportsbooks and individual teams. Sportsbooks could use similar models to help set the spread, while teams and other sports analysts could use it to track their win-probability throughout the game.

In addition to creating a model with a simple user interface, the research goal of this project is to determine which features are most important for predicting the outcome of a game. Any given game could have hundreds to tens of thousands of data points, depending on how in-depth the statistics are. A recent development in the field of football analytics is the NFL's Next Gen Stats player tracking. A high fidelity tracking system captures the movement of players, which provides increasingly complicated data compared to static statistics due to the four-dimensional nature (three spatial dimensions plus time). Although this data and analysis is outside the scope of this project, it highlights the enormous scope of the data that is available to analysts and provides the motivation to identify the most predictive features.

The data for this will come from a dataset compiled by a group of statistical researchers from Carnegie Mellon University which is available on Kaggle and Github. Although NFL entities like Next Gen Stats take extensive game data, there is a lack of comprehensive, publicly available NFL data, especially compared to other major sports. This makes it more difficult for analysts not affiliated with the league to perform research in football analytics. However, a team at Carnegie Mellon created a package to scrape and clean data from the NFL website and compiled it into data sets containing information about play-by-play statistics and final game outcomes.

Using this data, I will build a regression model to predict how likely the team is to win the game. The dataset has over 100 columns, so feature engineering and dimensionality reduction will

need to be performed to extract the most important features. Multiple model types will be tested and grid searches will be used to find the optimal hyperparameters before selecting the final model. The models will be evaluated based on how accurate they are at predicting game outcomes. Once the model has been chosen, trained, and evaluated, an interface will be created that will make the model accessible to users.

The final product will be a web application where a user can input the current game conditions and a prediction of the game outcome will be output. Although the UI will be simple and easy to navigate, the goal is to create a UX design that is also presentable and professional.

Data description

The data for this project was collected by the NFL but was compiled by a team from Carnegie Mellon by using an API provided by the NFL. The team includes Maksim Horowitz, Ron Yurko, and Sam Ventura. The data was collected over the 2009-2019 NFL seasons. The full data includes regular season, preseason, and postseason. This project will only use regular season data and will only aim to make predictions on regular season games. Preseason games are often not representative of regular season games because they are often used to evaluate potential players, so the rosters vary greatly compared to the regular starters during the season. Postseason games are also not representative of regular season games because of the added stakes of the game and the single-elimination playoff structure, which may influence what decisions are made.

The data will be accessed from Github and the games_data and play_by_play_data folders will be used. Data for each year is contained in a separate csv file, so they will be read in and concatenated to form the full data sets. According to Kaggle the data has already been cleaned and parsed, so there should be only small amounts of cleaning and imputation necessary, but this will be double checked. So far I have not found a data dictionary.

The play_by_play_data has 256 columns. Many of these columns detail the result of the play or analysis by the authors. This project is only interested in the current game state, so the data will be subset and the 26 of the columns will be used. These are:

‘play_id', 'game_id', 'home_team', 'away_team', 'posteam',
'posteam_type', 'defteam', 'side_of_field', 'yardline_100',
'game_date', 'quarter_seconds_remaining', 'half_seconds_remaining',
'game_seconds_remaining', 'game_half', 'quarter_end', 'drive', 'sp',
'qtr', 'down', 'goal_to_go', 'time', 'yrdln', 'ydstogo',
'total_home_score', 'total_away_score', 'score_differential'

There are 498,393 total observations in the play_by_play data. This data may be further subset during the initial data exploration, but the initial data will contain all variables related to the current game state.

The play_by_play_data contains “advanced metrics such as expected point and win probability values” (Kaggle) that were calculated by the team but these columns will not be used in my analysis. There is some legacy R code and code from a workshop they organized on Github but these will also not be used.The code from the workshop focuses on quarterback and receiver performance, which is not a part of my project.

The games_data has 10 columns. The data will be subset based on the columns that identify the game details and the outcome of the game. These six columns are:

‘game_id’, 'home_team', 'away_team', 'week', 'season', 'home_score', 'away_score'.

There are 2,816 total observations in the games_data.
Both the play_by_play_data and games_data share the game_id primary key, so these two tables will likely be joined on that column.

Software

Spyder will be used to develop the ML model in python. Flask will be used for the backend of the web app to create an API endpoint for the model. React or Angular will be used to create the UI. I use Angular in my current job but would also like to learn React.

Analysis plan & Model Specifications

Analysis description

Multiple machine learning models will be developed, tuned, trained and tested in Python in order to find the most optimal features and model to predict game outcomes. This process will include data cleaning, feature engineering, dimensionality reduction, grid searches, cross-validation, and accuracy metrics. To begin the project, the first model will be used only as a starting point in order to create the endpoint in Flask. Then the frontend will be created in React or Angular. Once a suitable model is found, it will be implemented in the backend and connected to the front end.

Week 1 goals

- ☑ ~~Find data~~
- ☑ ~~Develop project proposal~~

Week 2 goals

- ☑ ~~Set up Github repository~~
- ☑ ~~Clean data~~
- ☐ Preprocess data
- ☐ Feature engineering to narrow number of features
- ☐ Create an initial model. Just need an example model to connect to the endpoint.

Week 3 goals

- [ ] Create the endpoint
- [ ] Develop the basic UI structure

Week 4 goals

- [ ] Add formatting to the UI
- [ ] Integrate the backend into the front end

Week 5 goals

- [ ] Create multiple kinds of models
- [ ] Begin narrowing model selection and tuning hyperparameters

Week 6 goals

- [ ] Continue hyperparameter tuning on chosen model
- [ ] Test the final model

Week 7 goals

- [ ] Compile project into jupyter notebook
- [ ] Create video walkthrough

Delivery plan

I plan to communicate the results of the project using a jupyter notebook. The bulk of the project will be tuning, training, and testing machine learning models, and these are best presented in a code notebook. A jupyter notebook can also include screenshots of the web app which is useful for showcasing the end product and how the user interacts with the model.tr

Link to the google doc:
https://docs.google.com/document/d/1dVwonWPOA8GtiKYWg2LZU65nj7Jbtb7DzW87eTqUHtQ/edit?usp=sharing