AML - Project Deliverable 3

Participants: Milin Kaur Saini , Yonghan Xie , Sebastian Manuel Hereu , Harshith Belagur , Ingrid Xin Pan

Notebooks:
Notebook with models,
Notebook w/ data exploration & master table creation

Executive Summary

Our goal was to predict whether the home team in an NBA game would win or lose given a set of features. The data sets we used are available on Kaggle (link) The approach that worked best was CatBoost, which resulted in 64% accuracy.

Datasets:
We used two datasets, both of which are available on Kaggle. The first dataset contains information about over 64,000 NBA games in a large table. For each game, the table contains information such as the players involved, the home team, the away team, and the result of the game.

While the above dataset was useful for obtaining data about specific games, it lacked team-level statistics. Thus, the second dataset was essential for obtaining advanced statistics to be used in our game predictions.

After appropriate processing, we joined the game table from the first dataset and several tables from the second dataset to create a master table with which all models were built. Each row in the master table represents a game.

Preprocessing

Dealing with Null Values
One of the features was 'official_id' which represents the individual who was officiating the game. This feature may have had predictive power as officials can be biased toward certain teams or players.

In the data set, roughly 10% of the rows were missing the 'official_id' and this was handled by dropping these rows given this represented a relatively small proportion of the data.

Dropping Non-Predictive Columns
There were several columns that were non-predictive such as 'away_lg_team' where the value was 'NBA' for each row, or features such as 'away_abbreviation_team' where the value was just an abbreviation of the team name (stated in a separate column). Given the lack of predictive power and/or high collinearity, these columns were dropped.

Encoding Categorical Data
We created new features which were lists of player IDs for the home and away teams (feature names: 'home_team_players' and 'away_team_players') that were present for each game. This was not a feature present in the original Kaggle set and required some data manipulation to produce these.

This feature was in the form {4096, 4292, 4102, 4582, 4365, 4237, 4400}, with each number representing a player ID, and the set of numbers representing the players for each game for each team. Because ML models cannot inherently handle data in this format, we used a function called a MultiLabelBinarizer to convert these into one hot encoded format.

Additionally, we one hot encoded other categorical features with no inherent order such as the following:

- **'home_team'** - string representing the name of the home team
- **'away_team'** - string representing the name of the away team
- **'obpm_comparison'** - string with two possible values: (1) 'Home OBPM > Away OBPM', or (2) 'Home OBPM <= Away OBPM' where OBPM stands for Offensive Box Plus Minus (reference). At a high level, OBPM calculates a player's contribution to their team's offense when they are on the court and this statistic represents the aggregate OBPM of all players who played in a game.
- **'dpbm_comparison'** - string with two possible values: (1) 'Home DBPM > Away DBPM', or (2) 'Home DBPM <= Away DBPM' where DBPM stands for Defensive Box Plus Minus (reference). It's conceptually the same as OBPM, except for defense rather than offense.

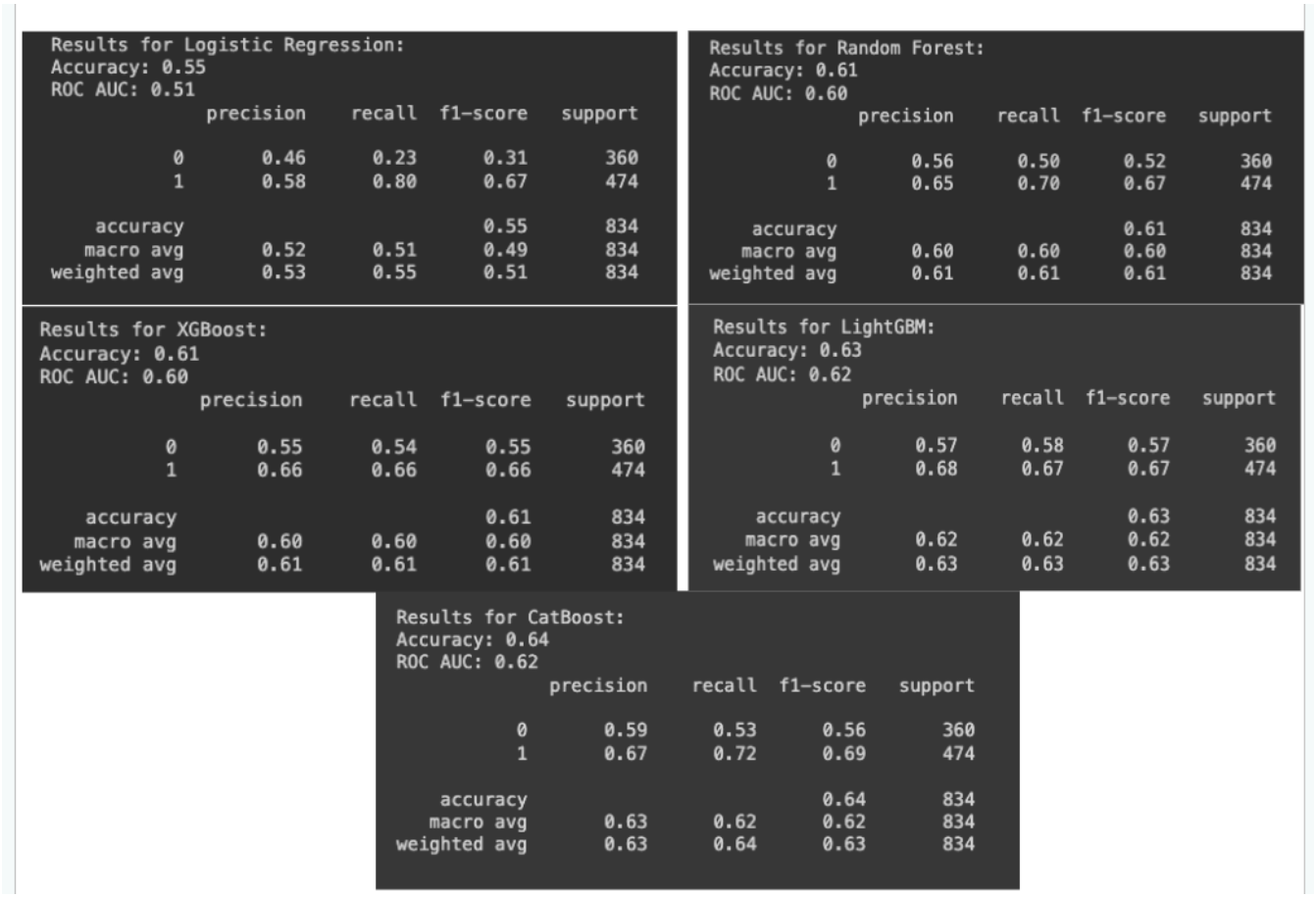
Handling Collinearity
To handle features that had high covariance with each other, we ran a correlation analysis of each pair of features. We dropped features that had a higher than 80% correlation with another feature. In total, we dropped 324 features, leaving 1325 features.

Other Processing
We converted dates in the 'game_date' column to an integer format. Additionally, we converted columns with values of 'TRUE' and 'FALSE' to 1 and 0 format.

Splitting Into Test and Train Sets
Given the data was not imbalanced (the home team had a 55% chance of winning), we opted for a random, rather than stratified split.

Models and Results

We tried a number of models on this data set. The accuracy of each is highlighted. Note that because we are not dealing with imbalanced classes, we think that accuracy is the appropriate metric to highlight.



Given that CatBoost had the highest accuracy, we went one step further with it to get the top 50 most important features and retrained the model on this subset of features. This resulted in an accuracy of 62%, which was not better than the original 64%.

Future work

If we were not time bound, other steps we may have tried would have included (1) doing PCA to reduce dimensionality rather than drop features which may have preserved more information though at the cost of interpretability, (2) train a neural network and create ‘player embeddings’. Similar to Word2Vec, we would create dense vector representations of players based on their statistics. Analogous to document embeddings, we could create team embeddings based on individual player embeddings.