CISC 5790
Dr. Zhao Yijun
Abe Raouh and Benjamin Castle
GitHub Link

Final Project Report

I . Introduction

The NFL Big Data Bowl 2025 provides a rich dataset to explore the application of machine learning algorithms in predicting various aspects of football plays. This project applies data mining techniques to analyze the dataset and implement classification models for specific performance predictions. The report details the steps to preprocess the data, implement these classification algorithms, evaluate their performance, and interpret the results.

This project uses metrics such as total player movement, per-position movement, and additional derived features to predict simple play outcomes - whether a play will be a run or pass. This approach combines data-driven insights with football strategy to make meaningful predictions and provide actionable knowledge for teams and analysts.

II. Football Concepts for Context

This competition, hosted by the National Football League (NFL) on Kaggle, aims to learn more about the actions teams and players take before the snap, which is the moment a play begins when a member of the offense passes the ball backward through their legs to another member of their team. The pre-snap phase is, at most, 40 seconds long and occurs before every football play. During this time, coaches and players work together to decide on the formation they will employ and the play they will run. Notably, lots of movement happens in this window: generally, teams will huddle or circle, up while one team member relays the instructions given to

them by the coach(es). After this huddle breaks, each member of the team must then find their way to their designated position. This contributes to the majority of the movement for most positions. Some positions, like the wide receiver (WR) or running back (RB), will, comparatively more often, see a lot more movement after this point from several sources. These two positions are also the key focal points of our classification question: will the play the offense runs be passing or running play?

III. Data Exploration and Preprocessing

The dataset comprises files capturing different aspects of football plays, including game metadata, player statistics, play-by-play details, and incredibly detailed player movements.

→ *Overview of the Dataset:*

The key datasets provided include:

- *games.csv: Contains game-level information, such as team names, scores, and the game date.*
- *plays.csv: Provides play-by-play information, including play types (run or pass), game context (down, distance), and metrics like expected points and win probability.*
- *players.csv: Contains player-specific details such as names, positions, heights, and weights.*
- *player_play.csv: Offers player-level statistics (e.g., rushing yards, receiving yards) for each play.*
- *tracking_week_[week].csv: The core dataset, which tracks player movements on the field in real-time using x and y coordinates, speed, acceleration, and orientation.*

For this project, the player_play.csv dataset and columns such as pff_runConceptSecondary and penaltyNames were excluded as they did not provide meaningfully to our analysis.

→ *Data Cleaning and Preprocessing:*

The first step in preparing this dataset for analysis involved clearing and refining the data. The focus was on data consistency. While examining the dataset, some rows contained missing values in key columns like player positions (x,y), speed (s), or game-related information. Since these rows were very few compared to the dataset, they were dropped without significantly affecting the data (only about 5 were dropped from over 16000 samples).

Filtering columns was also necessary to focus the analysis. To predict the play types (run vs pass), some columns like a quarter, down, yardsToGo, and pff_passCoverage were very important in our prediction, while others like penaltyNames and pff_runConceptSecondary were dropped.

Finally, during the data analysis, we observed that the target variable – play type – had a slight class imbalance with a ratio of approximately 1.5:1 in favor of passing plays. This imbalance was accounted for during model building by implying the class weighting technique (adjusting model training to penalize misclassification for the minority class).

IV. Feature Engineering

Feature engineering played an important role in transforming the raw dataset into a format suitable for model training and evaluation. The process involved selecting relevant features, calculating new metrics, and merging datasets to produce a comprehensive, enriched

dataset. Below is a breakdown of the steps to produce this dataset (please refer to the Python script “create_combined_data.py”).

As mentioned in the pre-processing step, only the most relevant columns were retained from the original dataset:

- *players.csv: Player-level details such as their ID, name, and positions they played*
- *games.csv: Game-level metadata such as homeTeamAbbr and visitorTeamAbbr*
- *plays.csv: Play-level information such as passResult, rushLocationType, expectedPointsAdded, and yardsToGo*

To analyze player movements before the ball was snapped, we filtered the tracking data to include only rows with frames labeled “BEFORE_SNAP” and rows where the player’s club was the same as the team that had possession of the ball.

```
tracking_with_position = tracking_with_position[
  (tracking_with_position["club"] == tracking_with_position["possessionTeam"])
  & (tracking_with_position["frameType"] == "BEFORE_SNAP")
]
```

Several features were engineered to quantify team and player movement during plays. This was a crucial part of our prediction algorithm, which will be discussed in more detail in the next sections. The first feature calculated was the total distance traveled by the possession team, which was calculated as follows:

```
total_distance_possession_team = (
  tracking_with_position.groupby(["gameId", "playId"])["dis"]
  .sum()
  .reset_index()
  .rename(columns={"dis": "totalDistanceTraveledByPossessionTeam"})
)
```

The second was the distance traveled by each position type (eg. QB, WR, RB) during each play and was calculated as follows:

```
distance_by_position = (
  tracking_with_position.groupby(["gameId", "playId", "position"])["dis"]
  .sum()
  .reset_index()
)

# Pivot the data to get distances for each position as columns
distance_by_position_pivot = distance_by_position.pivot(
  index=["gameId", "playId"], columns="position", values="dis"
).fillna(0)

# Rename columns for clarity
distance_by_position_pivot.columns = [
  f"distance_{pos}" for pos in distance_by_position_pivot.columns
]
distance_by_position_pivot = distance_by_position_pivot.reset_index()
```

Finally, the newly calculated features, and the existing chosen features were both combined to form the dataset combined_data.csv.

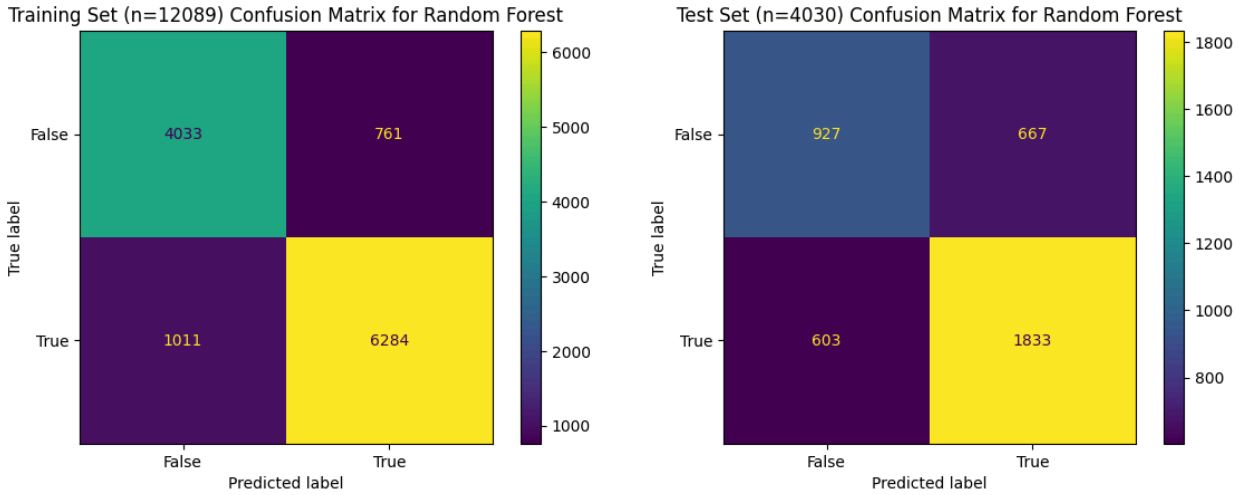
V. Model Building

Models were built in Python using Scikit-learn's machine learning tools. The models which consistently produced the best results were RandomForestClassifiers, using the offensive position distance sums, per play, as features, and isDropback, the boolean indicating if the play resulted in a run (False) or pass (True), as the indicator. A RandomForestClassifier was built with 100 trees, each post-pruned to a max depth of 12. Then, the TunedThresholdClassifierCV was applied, which uses 5-fold cross-validation to tune the decision threshold of the tree to optimize a chosen scoring criterion. The criterion "f1_weighted", which seeks to maximize the F1 score with samples inversely weighted by proportion, generated the best results in this case. A

LinearRegression was also attempted on a subset of continuous data with expectedPointsAdded as the indicator, but there were no pairwise trends at all, so the model was unable to produce any meaningful results.

VI. Results and interpretations

The initial RandomForest produced a training accuracy of 86% and a test accuracy of 69%, with the following confusion matrices:



The results from the TunedThresholdClassifierCV confirmed our results were validated: the best scoring function, weighted F1 score, produced an accuracy nearly identical for both training (85% accuracy) and test (69% accuracy) data from the initial RandomForestClassifier, using a split threshold of 0.482. Therefore, we can be reasonably confident that our model is valid. In these results, we see the modern NFL's proclivity for pass plays displayed prominently. It is also important to note that we are most concerned with class-1 accuracy: the penalty for misreading a running play is a few extra yards, but these plays tend to be contained. Misreading a passing play, however, can be disastrous, due to their inherent larger scale: one can throw a ball much farther and faster than one can run. Therefore, this model accurately accounts for this inherent imbalance in consequence by prioritizing the class-1 accuracy.

This model would be useful for both defenses and on broadcasts. Each team is allowed to have 1 player on the field at a time with an earpiece with one-way communication from the coach. If the coach is relaying information about the likely outcome of a play, the defensive player with the earpiece can use this information to quickly readjust their team or alert them of the likely outcome of the play. Even a split second of foresight can change the outcome of a play wildly in an environment as fast-paced as professional football. Additionally, the NFL's broadcasts on various networks have made an increasing effort to include advanced statistics in onscreen graphics. The model works off of a running sum of movement, so there could very easily be such a sum, by position, being fed into the model at every frame to compute a classification and a probability. A common criticism of American football is the pace: plays are short and breaks are constant and long, with a 3 hour TV window often only having 15-20 minutes of actual live game time. The most consistent type of break is the 40 second long pre-snap phase between plays. This graphic would help make the often monotonous pre-snap phase more engaging for the audience.