Predicting NFL Team Formations and Play Calling Final Report

Dhruv Raval (dkraval2, Undergraduate), Min Zhang (mz85, MCS), Ryan Thammakhoune (rct4, MCS)
University of Illinois Department of Computer Science

Abstract

We aim to predict play outcomes in NFL games by analyzing pre-snap behavior. Using the NFL Big Data Bowl 2025 dataset, we explore multiple machine learning models including decision trees, random forest, and logistic regression. This report covers our techniques and findings.

Introduction

Within any sport, the ability to reliably predict your opponent's next move will provide you with an immense advantage on the field and can be the difference between a win or loss. This trend can be seen in football, where historically, NFL coaches and players have used their knowledge of the game and footage of their opposing team to predict their opponents' formations and plays. However, as statistical modeling and machine learning techniques have improved, we aim to automate this process using historical pre-snap data from the NFL Big Data Bowl 2025 dataset. Our goal is to predict the play type, using player movement and game context data.

Motivation

This project is interesting because it merges real-world sports strategy in a highly complex sport like football with a significant amount of data. For an offensive or defensive coordinator, play calling involves complex pattern recognition and years of experience within the game. With the large amount of data collected in the NFL Big Data Bowl 2025 dataset, this makes football an ideal domain for classification.

Dataset Overview

As mentioned earlier, in this project, we leverage the NFL Big Data Bowl 2025 dataset—a publicly released competition dataset hosted on Kaggle that comprises millions of pieces of game data from the 2022 NFL Season. The dataset is composed of 3 smaller datasets: player_play (354,727 observations), play (16,124 observations), and player (1,697 observations). Both the player_play and play datasets are comprised of 50 features, of which 8 features within the player_play dataset are non-numerical like the team abbreviation and name of route ran, while the remaining 42 are numerical features like the NFL ID and number of rushing yards. Of the 50 features within the play dataset, 21 features are non-numerical like the play description, possession team name, and game clock, while the remaining 29 features are numerical like the quarter, which takes the following discrete values and distribution: [1 (27.38%), 2 (27.14%), 3 (22.34%), 4 (22.31%), 5 (0.83%)]. The player data consist of 7 features, with only two being numerical (player's NFL ID and their weight) and the remaining 5 being non-numerical like the name of the player, the players position, and their height. All 3 dataset files were processed prior to use in order to clean and extract significant features, which will be discussed later.

Related Literature

For this specific dataset, there have been a multitude of attempts on Kaggle in predicting plays with a multitude of modeling techniques. Of these attempts, the simplest model used was logistic regression [Ulis]. This approach, after applying a 5-fold cross validation, led to an accuracy of 71.4%, 78.9% sensitivity, 60.9% specificity, and an AUC of 0.76. Within the same attempt, a random forest model was also created. The random forest model yielded slightly higher accuracy than logistic regression with an accuracy of 72.45%, 78.85% sensitivity, and 63.58% specificity. Another attempt utilized gradient boosted trees (XGBoost) with better results than the logistic regression and random forest approach [Thirumaligai]. Within the XGBoost model, features like the game clock, yards to go, which down it was, and 19 other features (22 total) were utilized. The resulting model produced an accuracy of 73.80% and an F1 score of 0.7276. However, of the models used on Kaggle for this dataset, the one that appears to yield the best results is an ensemble model harnessing a random forest classifier, a gradient boosting classifier, and a logistic regression model with stochastic gradient descent learning [Pym]. Each model is then ensembled together using soft voting for the final prediction. This results in an F1 score of 0.91.

More generally, outside of the NFL Big Data Dataset, other attempts have been made to predict plays in the NFL using machine learning methods. In a 2019 paper from Udgam Goyal, performed at MIT, they use a play-by-play dataset from 2009 - 2018 to predict plays from each NFL team. Specifically, they narrowed down their features to only data available prior to the snap, similar to our plan. They selected features relating to the game status – drive, quarter, down, time, yard line, score etc. They also used features relating to the probability of scoring from different plays – no score prob, field goal prob, safety prob, touchdown prob, win prob, which are already provided in their dataset. They also use ratios such as run-to-pass ratios and rankings, such as run, pass, and overall offensive and defensive rankings and Pro Football Focus (PFF) grades. In our problem, we are less focused on individual teams and more focused on generalized predictions, and we do not have access to player and team grades. However, we used similar features such as game status and pass and run data. In terms of models, this study utilized logistic regression, neural networks, decision trees, and random forests. In their results, the random forest had a 75.3% accuracy, followed by the neural network at 75.8%, followed by the decision tree at 71.5%, and finally, the logistic regression model at 71.1%. As we will discuss later, we use similar models.

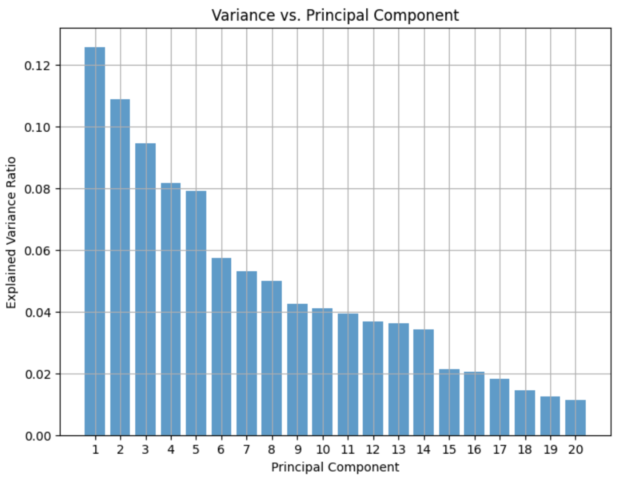
In the article NFL Play Prediction, a new measure for success was created called “progress” [Teich, et. al.]. This was used to create a new label for every NFL play that helps gauge how effective the play was. For our play prediction model, it may be important to create several such measures to gauge play effectiveness since there isn’t an explicit play effectiveness statistic. While this article found that decision trees had the highest accuracy at 67.14%, it was argued that Support Vector Machines had the best performance because of their high precision at 67.62%.

Methodology

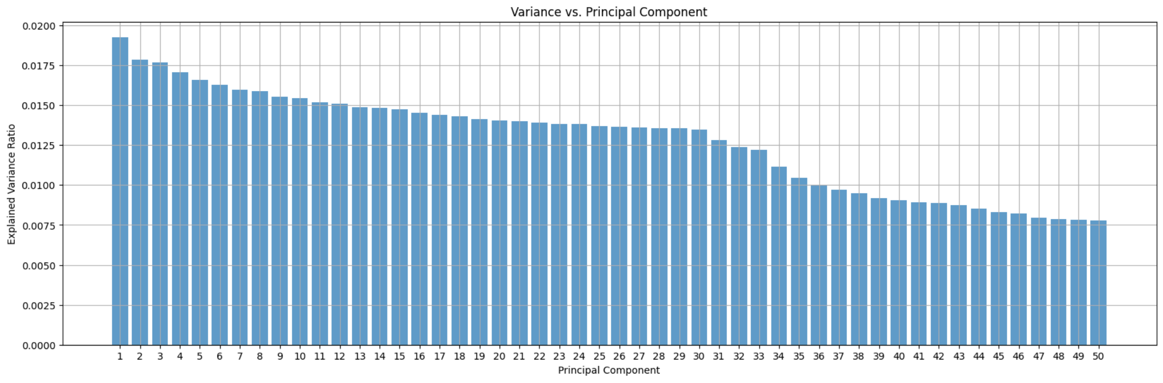
For the preprocessing stage, we need to link player information, specifically which quarterback is associated with each play within the game. This is done through creating two additional features within the plays dataset that will store the NFL ID and name of the quarterback based on the game ID and play ID for each observation within the plays dataset. In addition, we added a feature to identify the team's position on the field on a scale of 1-99, denoting the absolute yardline. We then converted the game clock feature from a categorical feature with the normal time format (hh:mm) to a numerical feature by having the game clock feature as the number of seconds left in the quarter. The final feature we added was a binary feature based on whether the play was a pass or a run, which was done through analyzing if there was a pass location recorded for that observation. After the new features are added, we analyze the occurrences of null/empty entries within the dataset. However, we found that the empty entries within certain features were informational, so no processing was performed to remove them for numerical features. For categorical features, we filled the empty entries with the string “None” prior to performing one hot encoding on those features. We also went through and removed features that contained information that was recorded post-snap and the IDs as we want our model to only use information available pre-snap in order to predict the play.

PCA

Once the data was cleaned, we had 17 variables remaining (1 response and 16 explanatory variables), which included categorical features like the name of the quarterback, the possession team, and the defensive team and numerical features like the game clock, which quarter it was, and yards to a first down. We then performed PCA to explore the reduction of dimensionality of our data. However, PCA does not work as well with categorical data. We first visualized the top 20 principal components with categorical data removed as shown below.



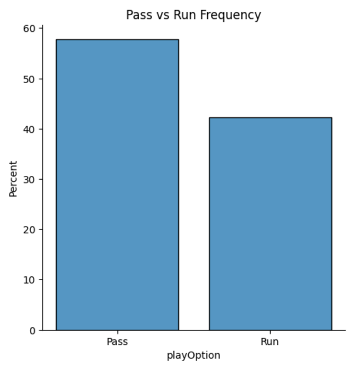
If we convert binary categorical features with one hot encoding, we get the following result.



From these results, using PCA might not help our models. If we remove categorical data values, we risk losing important features. If we convert categorical features using one hot encoding, we still have a high amount of principal components with high variance. Thus, we decided to run our models without dimensionality reduction.

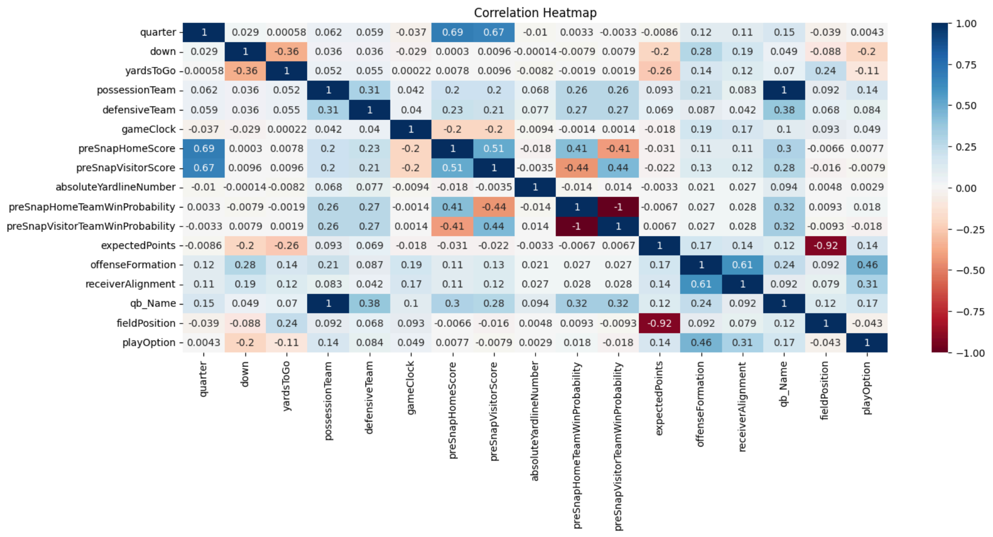
Class Imbalance Exploration

We also explored class imbalance in our dataset. We specifically wanted to see if there was a major class imbalance within our target feature, playOption. In order to achieve this, we plotted the difference between the amount of passes and runs as the outcome of a play. From the plot we created (listed below), we can see that although there are more passes than runs within the data, the disparity between the two classes is not large, so we can conclude there is no major imbalance.



Feature Selection

We then created a heatmap to visualize the correlation between features. From our correlation matrix, we can see that there appears to be some correlation between the features and the target feature (playOption).



However, in order to ensure that the correlation is significant, we calculate p-values in order to extract the numerical features that are significant. With a p-value threshold of 0.05, we found the following significant features: ['down', 'yardsToGo', 'gameClock', 'preSnapHomeTeamWinProbability', 'preSnapVisitorTeamWinProbability', 'expectedPoints', 'fieldPosition'].

For non-numerical features a χ^2 test is done between them and the target feature. To which we derive the p-value from to determine their significance. With a p-value threshold of 0.05, the following significant features: ['possessionTeam', 'defensiveTeam', 'offenseFormation', ’receiverAlignment', 'qb_Name'].

Model Implementation

With the features selected, we implemented two baseline models: a logistic regression model and decision tree model. We then implemented a random forest classifier and an ensemble model with soft voting with data that was split using the holdout method and random sampling (without replacement). We performed a 75-25 train-test split, and within the train split we performed an additional 80-20 train-validation split. With our base models implemented, we performed hyperparameter tuning on models like the decision tree and random forest classifier to prevent overfitting. In our experiments, with default parameters, both models yielded 100% training accuracy with low validation and test accuracy. We tuned our hyperparameters through iteratively testing different max tree depths for both decision tree and random forest classifier and recording the max tree depth that yields the highest validation accuracy. For the random forest classifier, we repeated this process to tune the number of trees within the forest. We also adjusted the weights used within the ensemble models in order to place more weight on the ensemble models that individually performed better.

We then utilized 5-fold cross validation in order to compare the performance of the 4 models based on metrics like accuracy, F1 score, precision, and recall. Based on these metrics and their average across each of the 5 folds, we determined our best performing model.

Results

	Train	**Val**	**Test**
Logistic Regression	0.723899	0.723853	0.724138
Decision Tree	0.727207	0.731707	0.727611
Random Forest	0.890428	0.741215	0.733565
Ensemble	0.807008	0.733774	0.730588

With all four models trained and tuned, we found that the two baseline models of logistic regression and decision tree, we were able to achieve similar results to comparable attempts of the same models on Kaggle with the dataset using the previously mentioned train-val-test splits. Specifically, the logistic regression tuned to use sag solver (the sag solve is able to handle larger datasets more efficiently) achieved a training accuracy of 72.3% and a testing accuracy of 72.4%. The decision tree had slightly better performance in terms of accuracy at 72.7% and a test accuracy of 72.8% once the max depth of the tree was tuned to be limited to 3 as letting the tree build out without a depth limiter yielded an overfitted model with 100% training accuracy and very low validation and test accuracy. Beyond the two baseline models, the random forest model was hyperparameter tuned to have max depth of 23 and 98 trees within the forest and the ensemble model ensembled the tuned version of the other 3 models and utilized soft voting with weights of [1, 2, 3] for the logistic regression, decision tree, and random forest models. These two tuned models both yielded better accuracy than the baseline models, with the random forest classifier having higher training and testing accuracy at 89.0% and 73.4%.

For a comprehensive performance analysis, we performed 5-fold cross validation and recorded the average accuracy, F1 score, precision, and recall across all 5 folds for each model in the table below.

	Logistic Regression	**Decision Tree**	**Random Forest**	**Ensemble**
Average Accuracy	0.724509	0.727362	0.734308	0.733688
Average F1	0.651182	0.656587	0.670888	0.668377
Average Precision	0.700227	0.701846	0.703866	0.705295
Average Recall	0.608703	0.616906	0.640977	0.635227

Conclusion

Results show that offensive play calling in the NFL can be predicted with machine learning models using only pre-snap data, achieving test accuracies exceeding 70% across all models evaluated. Our experiments demonstrated that tree-based models like decision trees and random forests outperformed logistic regression, and that ensemble models combining these classifiers can offer an increase in performance compared to the individual models. Despite exploring dimensionality reduction through PCA, we found it ineffective due to the loss of categorical context and the high variance retained across many components. Our final models maintained strong generalization performance through proper feature selection, categorical encoding, and extensive cross-validation. These results affirm the potential for data-driven approaches to inform decision making in football.

Ablation Study/Parameter Study

We conducted a series of ablation studies and parameter tuning experiments to understand the effect of individual model parameters and ensure that performance improvements were not the result of overfitting. To test the effectiveness of feature selection, we tried removing the least significant features like "fieldPosition" and "expectedPoints" and ran the models. This resulted in a reduction in test accuracy ranging from 2-5% across our models, suggesting that keeping all features would provide the most accurate results. Running our models with default parameters led to poor results. Without limiting the max depth of the decision tree model, we were able to achieve 100% training accuracy, but extremely poor testing accuracy. Tuning the max depth to 3 significantly improved generalization, yielding a balanced performance across all splits. Similarly, for the random forest model, we tested a range of tree counts and depths and found optimal performance with 98 estimators and a max depth of 23. Our ensemble model benefited from weighted soft voting, with increased weight given to models with better validation metrics.

Future Work

Several possibilities remain for future work. First, our framework can be adapted to predict other football-related targets such as offensive and defensive formations, play direction, or play success metrics. Additionally, deeper machine learning methods such as neural networks hold potential to further improve performance. Using a neural network might help us learn more complex, nonlinear relationships that we did not capture. Finally, integrating external data, such as weather conditions, crowd noise, coaching tendencies, or player ratings (sourced from PFF or other data providers) could provide additional context to further refine our predictions.

Contributions

Although everyone worked on each part of the project, each team member made major contributions or was tasked to work on the following sections.

Dhruv Raval	Min Zhang	Ryan Thammakhoune
Related literature research, data cleaning, and class imbalance verification	Base model training and hyperparameter tuning	PCA, visualization, ablation study, and feature selection

References

Goyal, U. (2019, December 19). Leveraging Machine Learning to Predict Playcalling Tendencies in the NFL. Massachusetts Institute of Technology. https://dspace.mit.edu/bitstream/handle/1721.1/129909/1237411720-MIT.pdf?sequence=1&isAllowed=y

Pym. (2025, January 6). NFL Play Prediction Using Ensemble Learning. Kaggle. https://www.kaggle.com/code/moneymoneymoney12138/nfl-play-prediction-using-ensemble-learning

Teich, B., Lutz, R., & Kassarnig, V. (2016, January 4). NFL play prediction. University of Massachusetts Amherst. https://arxiv.org/abs/1601.00574

Thirumaligai, S. (2025, January 11). NFL Big Data Bowl - Play Prediction Attempt. Kaggle. https://www.kaggle.com/code/sriharithirumaligai/nfl-big-data-bowl-play-prediction-attempt

Ulis, M. (2025, January 6). NFL Big Data Bowl 2025 - Michael Ulis. Kaggle. https://www.kaggle.com/code/michaelulis/nfl-big-data-bowl-2025-michael-ulis