DS 7200: Final Report
By: Clarence Williams, Lingzhen Zhu, Ethan Nelson

Abstract

Taking inspiration from the NFL's embracing of data analytics in play selection, this project seeks to predict the offensive play selection on the fourth-down of NFL games. The dataset was sourced from the NFL Play-by-Play Python package using the 2000 to 2023 seasons. Thirty features were used to create a logistic regression, random forest, and Naïve Bayes model, with random forest having the best predictive accuracy and F1 score of 86% and 85%, respectively, on the withheld 2023 season test set. The best model tends to be conservative with fourth-down play selection because it encompasses over 20 years of play by play information.

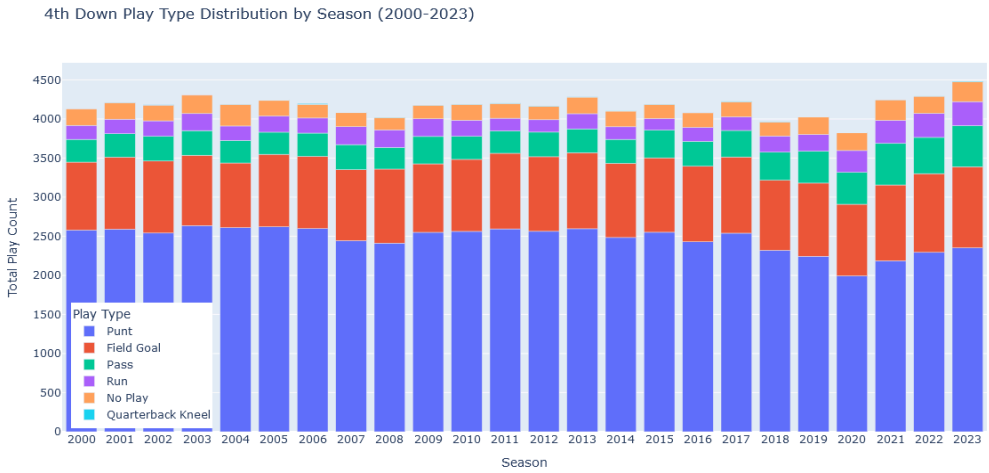
Data Processing

The data used in our project comes from an open-source Python package "nfl-data-py" (source: https://pypi.org/project/nfl-data-py/). It allows us to pull play-by-play NFL data with detailed information according to our desired time period. Our data processing workflow is as follows.

- Data Cleaning

We began by filtering the dataset to include only relevant variables and observations. Specifically, we focused on all available data from 2000 to 2023, only plays that occurred during the 4th down. As for the variables, we included the target variable (play_type), field position variables (yardline_100, ydstogo), game time variables (quarter_seconds_remaining, half_seconds_remaining, game_seconds_remaining), score difference variable (score_differential), environment variables (surface, roof, wind, temperature) where surface is the type of field, grass, astro turf , etc. and roof is the type of stadium, closed, open, outdoors, or dome. Finally, context related variables were included in our filtered dataset (home_team, away_team, season_type, pos_team, pos_team_score, pos_team_timeouts_remaining, def_team, def_team_score, def_team_timeouts_remaining drive, score, game_half, expected points (ep)),. These selected features could have a potential influence on the 4th down decision-making process.

To understand the dataset's overall structure, we plotted a distribution of play_type.



We could see that there are some records belonging to qb_kneel plays, which represent a time-wasting tactic typically used at the end of the game. These plays are uncommon and hold little strategic importance, so we excluded them as well.

Given the large dataset size, we leveraged distributed computing by converting the data into a Spark DataFrame for efficient processing.

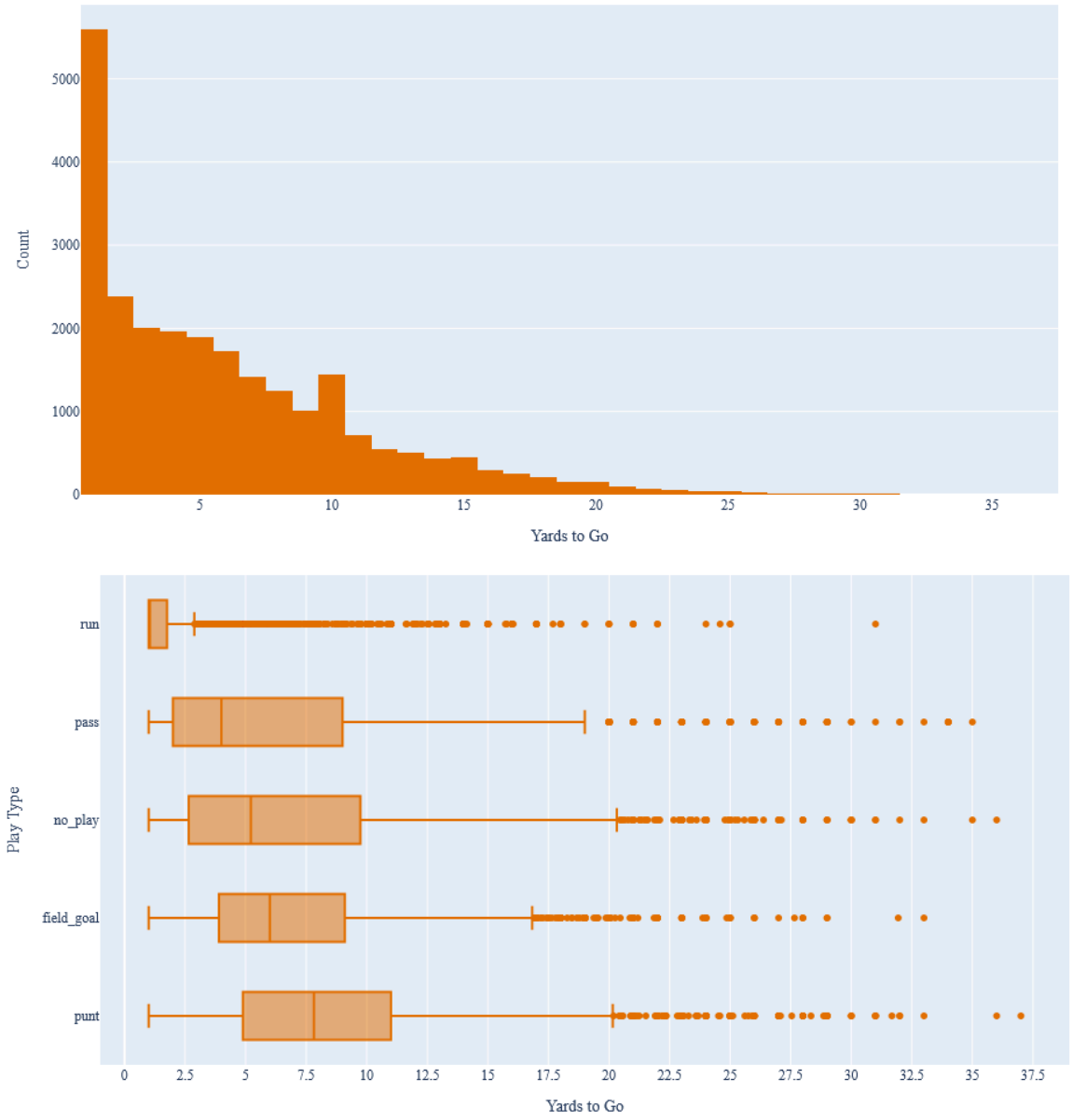
- Data Preprocessing

First, we did a train-test split, the data was divided into a training set (60%) and a test set (40%). Next, we addressed the class imbalance issue. To ensure our model does not develop biases toward majority classes, we used the RandomUnderSampler from the imblearn package to reduce the number of samples in the majority classes (punt and field_goal) to match the smaller categories. This method randomly removes samples from the majority of classes to balance the dataset. We then applied the SMOTENC function from imblearn to oversample all classes except pass, SMOTENC generates synthetic samples for minority classes by interpolating between existing samples.

- Exploratory Data Analysis

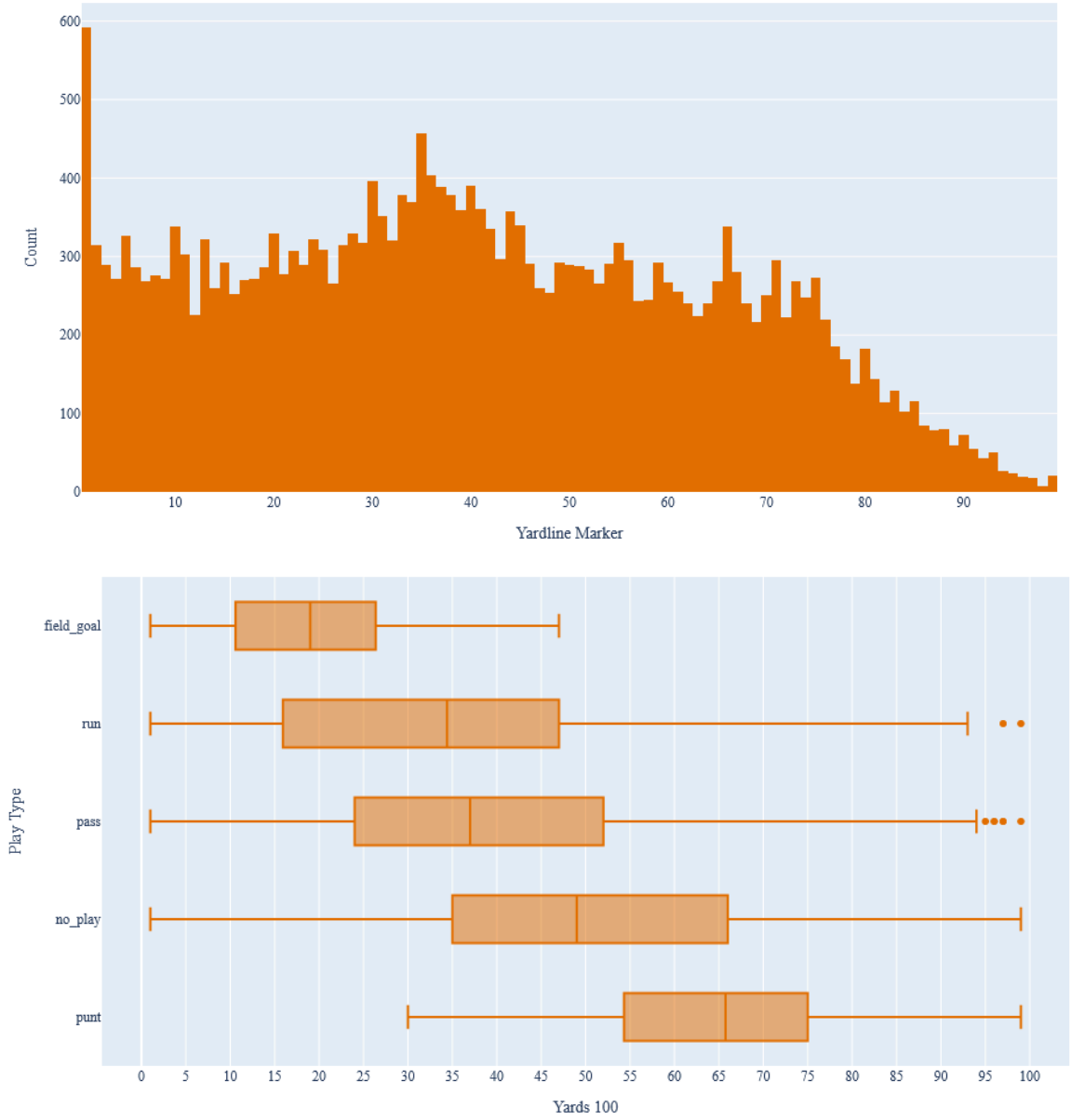
We conducted exploratory data analysis (EDA) on key variables to understand their distributions and relationships with play_type.

Yards to Go:



The histogram and box plots above highlight that the most common distances teams face on 4th down is 1-5 yards. This distance is considered short distance and indicates higher odds of attempting a conversion through running or passing the ball. There are a few teams facing longer distances which may then favor punting or attempting a field goal.

Yardline Marker:



This histogram shows the distribution of yardline_100, representing the ball's position relative to the opponent's end zone. In short, this is how far the offensive team must go until they achieve a touchdown. The box plot shows teams closer to their own end zone tend to punt, while those in the opponent's red zone (20-40 yard range) favor field_goal.

- Feature Engineering

Our model incorporates a combination of categorical (home_team, away_team, posteam, defteam, season_type, game_half, and play_type) and numerical variables (yardline_100, ydstogo, score_differential, quarter_seconds_remaining, and half_seconds_remaining).

For categorical variables, we first converted them into numerical indices using StringIndexer, then we applied one-hot encoding via OneHotEncoder to avoid numerical relationships between categories. For numerical variables, we standardized using MaxAbsScaler to normalize the scale of features, ensuring they contribute equally to the model.

Models

- Logistic Regression

Logistic regression was the initial baseline model. The dataset was fed through the pipeline described below. The pipeline consisted of processing categorical features first using a string indexer. Next, the string indexer output was one hot encoded. Then, the one-hot encoded categorical features and numeric features were combined and fed through the feature assembler to make the features column. This features column was standardized using a max abs scaler to scale it on the range [-1, 1] before being fed to the logistic regression model.

Hyperparameter tuning was accomplished using a grid search for the regularization parameter (regParam), elastic net parameter (elasticNetParam), and the number of iterations. The final parameters of the best logistic regression were 25 iterations and 0 for both regularization and elastic net parameters.

The model's performance was evaluated on the 2023 season test dataset to see how it performed against completely unseen data. The overall performance of the logistic regression model is as follows: accuracy of 0.7900, F1 score of 0.7506, weighted precision of 0.7515, and weighted recall of 0.7900.

- Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs to the class that is the mode of the predictions from individual trees. This approach is well-suited for multi-class classification problems like ours due to its ability to handle complex feature interactions and non-linear relationships. This model will follow the same pipeline as the previous logistic regression model, and for the hyperparameter tuning, we use the following parameters: numTrees (Number of decision trees in the forest), maxDepth (Maximum depth of each tree), and impurity (Splitting criterion usually gini or entropy). The final parameters of our best model are: numTrees: 50, maxDepth: 15, impurity: entropy. The training and validation were performed on 80% of the training data, and the remaining 20% was used to evaluate the model performance during tuning.

On the 2023 season test set, the random forest model had an accuracy of 0.8562, F1-Score of 0.8031, weighted precision of 0.7846, and weighted recall of 0.8396.

- Naïve Bayes

The third model created was Naïve Bayes. The same data pipeline was used to create the model as was used for the previous two with one modification. A MinMaxScaler was tweaked to have minimum and maximum values of 1 and 2 rather than the normal values of 0 to 1. This change was to account for an error that occurred where PySpark said Naïve Bayes was unable to handle negative values. When the min and max values were left at 0 and 1 respectively, some resulting values were still slightly negative, thus throwing an error. Shifting the range slightly fixed this error. After undergoing this data pipeline, a hyperparameter search for the smoothing parameter was conducted with values ranging from [0.05, 1]. As Naïve Bayes only has one parameter, this was the only aspect that varied from the tested Naïve Bayes models. The hyperparameter search was conducted both using the regular play_type label class distribution and a balanced distribution using under and oversampling.

Accuracy and F1 Score were used to evaluate which configuration of the model was best. First, the model was tested on a data set that contained data from all seasons. Under these metrics, the best model with regular sampling of the play_type label class has a sampling parameter of 0.05. This model had an accuracy of 0.6942, F1-Score of 0.5688, weighted precision of 0.4819, and weighted recall of 0.6942. When using over and undersampling techniques to balance the distribution of play_type, the best smoothing parameter was 0.65. This model had an accuracy of 0.5459, F1-Score of 0.6205, weighted precision of 0.7821, and weighted recall of 0.5459. When comparing the two sampling techniques, regular sampling resulted in the best model.

When tested on only data from the 2023 season, the Naïve Bayes model without class balancing had an accuracy of 0.5462, F1-Score of 0.3860, weighted precision of 0.2984, and weighted recall of 0.5462.

Results

The results from each model using the 2023 season test set is summarized in the table below.

	Accuracy	F1
Logistic Regression	0.7900	0.7506
Random Forest	**0.8562**	**0.8031**
Naïve Bayes	0.5462	0.3860

Random forest was the best of the created models. Random forest, logistic regression, and Naïve Bayes had accuracies of 0.8562, 0.7900, and 0.5462 respectively. While random forest and logistic regression had comparable performance, Naïve Bayes struggled the most. A potential reason for the underwhelming performance by Naïve Bayes could be its assumption of independence among its features. In our case, there is likely strong correlation between each of the predictor variables, leading to worse performance.

- Random Forest Model Sensitivity

We explored decreasing the number of trees and the max depth to see how our best model, random forest, is sensitive to parameter changes. All metrics are given by using the 2023 season test set. Decreasing the number of trees from the best value of 100 to 75 does not appreciably impact the results. In this case, the model had an accuracy of 0.8356, F1-Score of 0.7987, weighted precision of 0.7800, and weighted recall of 0.8356. Furthermore, by decreasing the number of trees from 75 to 25, the evaluation metrics slightly improve, but the increase is not notable. For 15 trees, the model had an accuracy of 0.8400, F1-Score of 0.8035, weighted precision of 0.7840, and weighted recall of 0.8400.

When the max depth was decreased from 15 to 10, the model had an accuracy of 0.8098, F1-Score of 0.7611, weighted precision of 0.7573, and weighted recall of 0.8098. Decreasing the max depth from 15 to 5 most impacts the model's evaluation metrics. In this case, all metrics decreased, with accuracy decreasing to 0.7559, F1-Score decreasing to 0.6592, weighted precision decreasing to 0.6581, and weighted recall decreasing to 0.7559.

Conclusions

The goal of this project was to predict the play type of NFL teams in 4th down situations using a combination of situational factors from the nfl-data-py dataset. We successfully tested logistic regression, random forest, and Naïve Bayes models. The random forest model proved to be the most successful, correctly predicting the play type of NFL data with an accuracy of ~85%. This project could be improved in 2 distinct ways. First, additional predictor variables could be identified to improve the performance of each model. Potential new predictor variables include the actual players who will conduct the given play types, offensive and defensive ratings of the teams, and positional tracking data of the players. A second approach to improving the models performance would be to explore additional model architectures including neural networks, gradient boosted trees, or reinforcement learning. The latter method would reorient the objective to be the play type that maximizes the win likelihood or number of points that a team can earn.

Contributions

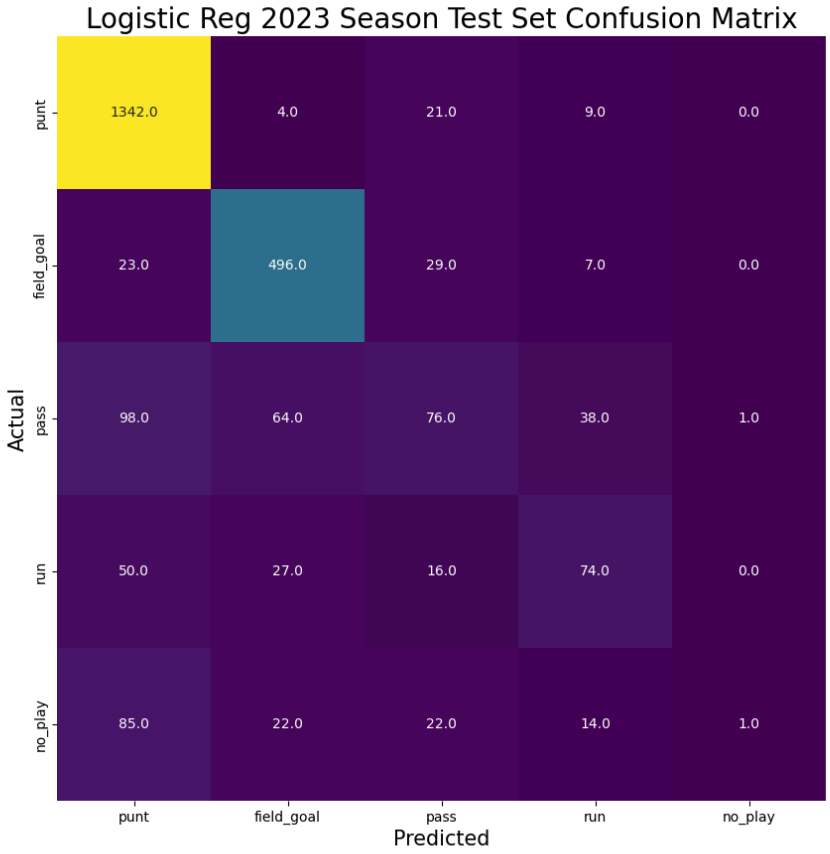
Clarence Williams: Logistic Regression Model

Lingzhen Zhu: Random Forest Model

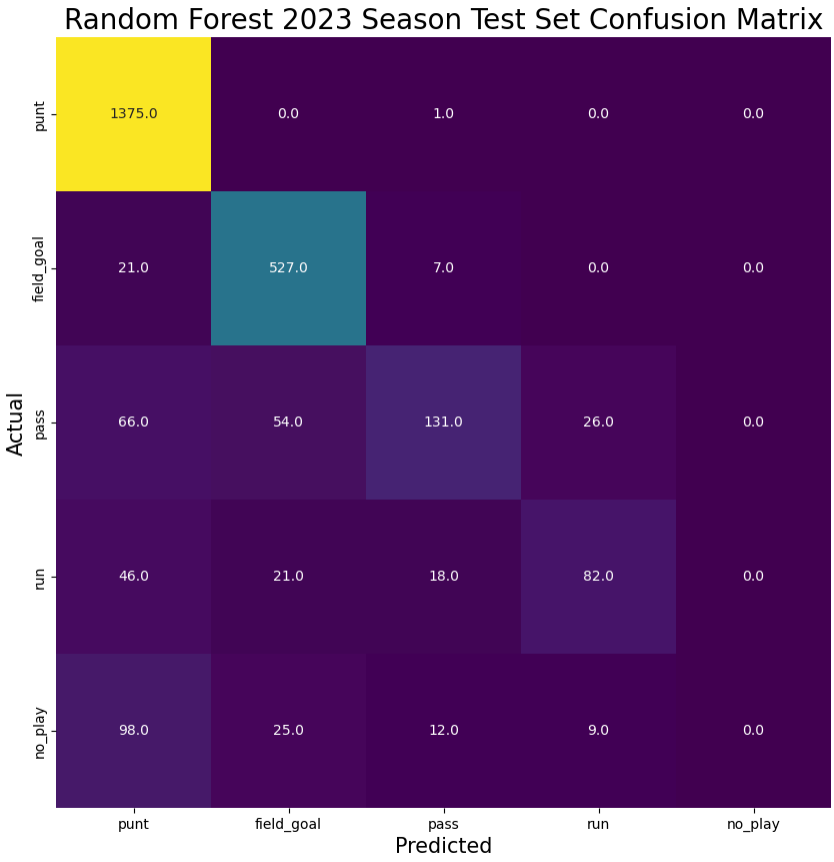
Ethan Nelson: Naïve Bayes Model, EDA Notebook

Appendix

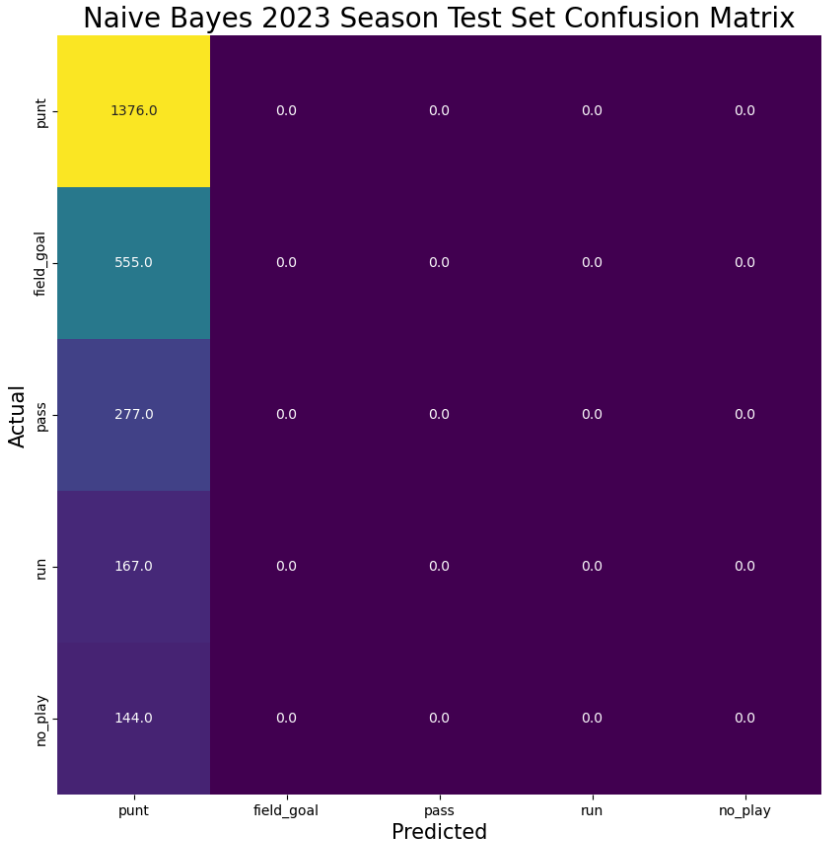
Logistic Regression Confusion Matrices



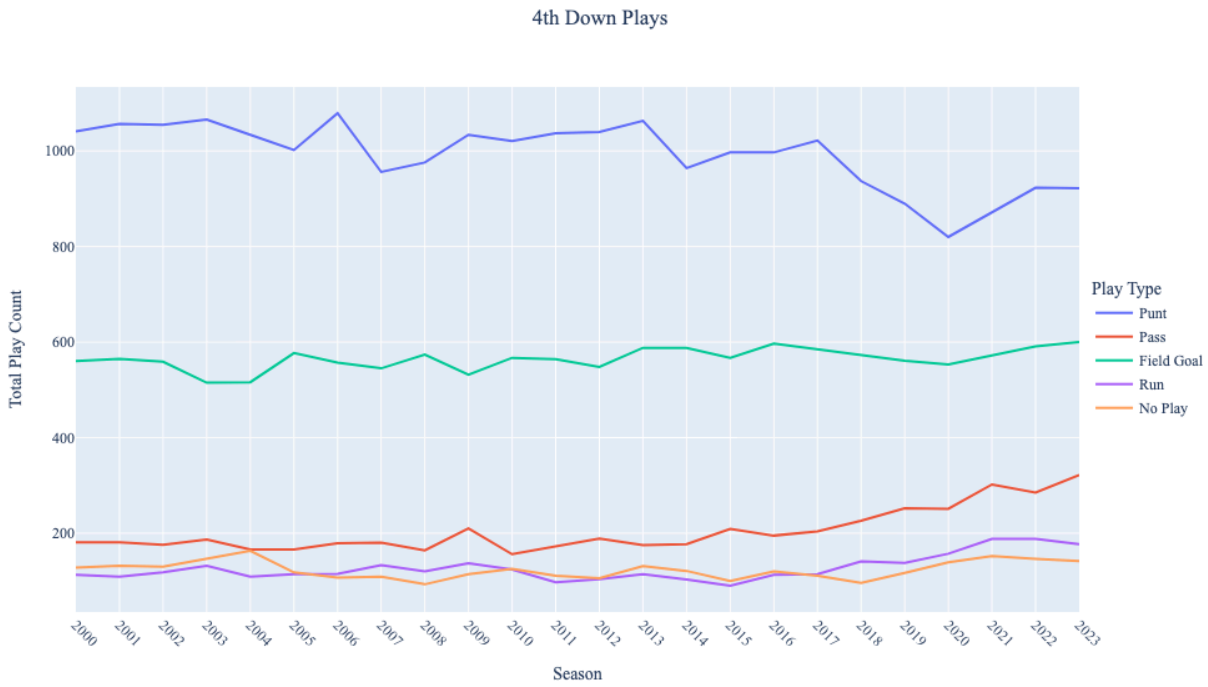
Random Forest Confusion Matrices



Naïve Bayes Confusion Matrices



Play Type Count Line Graph


4th Down Plays
Total Play Count
1000
800
600
400
200
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
Season
Play Type
Punt
Pass
Field Goal
Run
No Play