# PLAYGUESSR: COMMERCIAL APPLICATION OF MACHINE LEARNING IN FOOTBALL PLAY PREDICTION

Jason Wu[1], Evan Gunnell[2] and Yu Sun[2]

[1]Barrington High School, 616 W Main St, Barrington, IL 60010
[2]California State Polytechnic University, Pomona, CA, 91768

***ABSTRACT***

*The offensive strategy in American football strives to be enigmatic. A strong offense has a well rounded offensive playbook, rotating offensive plays in attempts to disrupt any predictive patterns. Therefore, it has always been in theinterest of defensive coordinators to offer accurate predictions of the upcoming play to minimize offensive yardage gain. A well advised defense can change its positioning and coverage schemes, given solely whether the next play will be a run or a pass. Although coaches have developed traditional heuristics for tendency-based play prediction, they are limited to patterns discerned by human consciousness.*

*This paper aims to take advantage of recent professional football databases in an attempt to develop a machine learning classification model for predicting opponent play-calling, as well as implement said model into a novel application aimed at deployment on all levels of play.*

*We conducted research on various classification models and features. Utilizing 10 past seasons from the National Football League (NFL), we devised random forest classification models with optimally chosen features. We also investigated the importance of Synthetic Minority Oversampling Technique (SMOTE) in training with inherently imbalanced datasets. The final model was able to achieve an NFL league average accuracy of 89.52%, with 91.69% as the highest team-specific accuracy. This accuracy is substantially higher than past projects of similar goals.*

***KEYWORDS***

*Football Analytics, Machine Learning, Classification, Play Prediction.*

## 1. INTRODUCTION

Football is distinctive in its compartmentalized, or "discrete", plays. Hence, football plays naturally contain features, such as the score differential, down, and time remaining. Similarly, most football plays can be classified into two: running—moving the ball upfield from the line of scrimmage—and passing—throwing the ball from behind the line of scrimmage to a receiver positioned downfield. All remaining plays are "special", such as those when the ball is not in play (kickoffs and points-after-touchdown) and those when the ball remains in play (punts, field-goals, and quarterback kneels) [7].

American football, like the majority of competitive sports, is based on ambiguity [2]. From a defensive standpoint, the offensive advantage relies heavily on the unpredictable nature of play-calling [3]. Defensive coordinators rely on traditional heuristics, usually on past opponent tendencies, to predict offensive plays. Simple predictions can lead to shifts in defensive

David C. Wyld et al. (Eds): COMIT, CRBL, BIOM, WiMNeT, SIP, AISO - 2021
pp. 145-164, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.111714

positioning and coverage schemes that greatly hinder offensive capabilities.

Given the readily accessible professional football datasets, as well as the inherent need for improved defensive prediction tools for in-game coaching assistance, this paper seeks to devise an accurate machine learning model for said predictions, as well as fully implement an accompanying mobile app for ready deployment at all levels of play [4].

Traditional prediction heuristics are indispensable to football [12]. Opponent behaviors are usually inferred through careful analysis of opponent game footage. However, such a method is somewhat ineffective, since much of the underlying patterns are indiscernible to humans, due to inability to consider a wide set of potential influential features. Furthermore, human bias and retroactive interference limit the extent to which team-specific tendencies are discovered. Moreover, tendency-based metrics, such as run/pass ratios, have wildly varying performances. For example, recent years saw the National Football League (NFL) shift to adopt a pass-heavy offense. However, teams with balanced playbooks will have run/pass ratios that approach 1, which, intuitively, lowers prediction accuracy.

Past papers have considered solely the prediction of two types of targets: run plays and pass plays, with varying success [6]. Although special offensive plays, such as punts, field-goals, and quarterback kneels are rare and self-revealing, they can still reveal certain attitudes from opposing teams that are not discernible by simple run/pass target sets. For example, an offensive coordinator who favors run/pass plays over punting on 4th down can be inferred to be riskier and prone to turnovers. Even more, run/pass-only target sets significantly reduce data for certain situations. For example, 4th down data is greatly reduced when punting plays are neglected, which skews the data-by-down distribution.

Past papers have also focused mainly on professional level deployment. The extensive datasets that result from NFL coverage allows for papers to implement models not easily generalizable to other levels of play. For example, one popular input factor commonly featured by past papers is Madden player ratings. Video game statistics, aside from the fact that they are tailored to serve diverging purposes that confound model development, are not easily replicable on lower levels, such as high-school and collegiate teams.

Current years saw an uprising in publicly maintained professional sports databases [7]. In this paper, we aim to accomplish the goal of accurately predicting opponent play-calling. We were inspired by past papers to utilize the expansive Kaggle dataset developed by Horowitz et al, detailing 2009-2018 NFL play-by-play statistics. Past papers have attempted to solely predict whether the upcoming offensive play will result in a run or a pass. However, by removing special plays, such as punts, a small yet significant portion of the dataset is lost [8]. Our paper is the first to recognize the usefulness of offensive special plays in the target set, since offensive special plays do not disrupt game continuity and function heavily similar to other offensive plays. Furthermore, we predicted that including special plays offers relevant information to both offenses and defenses alike regarding an opponent's situational behaviors. More elaboration is provided in the subsequent section.

Furthermore, we prioritized usability factors, such as time expenses, practicability, and functionality. Moreover, our application is aimed towards deployment on all levels of play, whereas previous papers have majorly focused on experimentation and development on the professional level with the NFL.

Much of our motives behind our evaluation metrics and experiments chosen are described in the following section, but our priorities are generally targeted towards responding to the challenges

described in Section 2. We chose 5-fold cross validation for evaluation, which allows a model trained on 8 past NFL seasons to be evaluated for 2 NFL seasons. This metric allows for confirmation that play-calling tendencies are persistent and independent of specific matchups. Similarly, models were trained with team-specific data, and the league average—across 32 teams—is recorded. This is done to ensure that our experiments are not skewed by teams with varying tendency strengths.

Our experiments target specific questions and challenges that guide the development of our machine learning model. First, we analyzed the potential application of Synthetic Minority Oversampling Technique (SMOTE) on the dataset, due to an inherent imbalance. We evaluated simple classification models trained on both pre-SMOTE and post-SMOTE data. Secondly, we explored various popular classification models and chose one with optimal accuracy and time expenses, again weighted with regard to cross validation scores. Third, we studied the influence of our feature set and greatly reduced the final feature set to maximize model accuracy.

The paper's structure is as follows: Section 2 describes a small subset of significant challenges that we encountered and overcame during the design and experiments; Section 3 dissects our solution into details by analyzing our implementation methods and the motives behind our choices; Section 4 presents our experiments, as well as an abundant collection of recorded data, that were targeted to correspond to the various challenges that we discuss in Section 2; Section 5 gives a brief analysis of recent related works, including a critique and a comparison with our paper. Finally, Section 6 gives a brief concluding summary, as well as defining areas of future work.

## 2. CHALLENGES

In order to build the prediction model, a few challenges have been identified as follows.

### 2.1. Choosing optimal dataset features & targets

The dataset that we utilized to conduct research featured an extensive list of features, 255 in total [9]. However, we distinguished only 15 features to have potential relevance to influence play calling. Althoughthere was an overabundance of potential targets, ranging from total WPA (win percentage added) to two-point conversion probability, we ultimately decided on play-type [10]. That is, whether the play would result in a pass attempt or a run attempt. Our rationale behind this being that run/pass play predictions are actionable data for the defense. In addition, we added all possible offensive plays, such as punts, field-goals, and quarterback kneels (only excluding special plays that take place after scoring, such as extra points and kickoffs). Past papers have erred in the ruling that only run/pass predictions are constructive to the defense. These erroneous perspectives limit the dataset, especially on 4th down, where punts are most likely and most frequent. In the perspective of commercial potentials, a client should be able to input arbitrary factors for predictions beyond the next play. For example, if a team is predicted to attempt a field-goal later in an offensive drive, an offensive coordinator can determine the intensity of the next offensive drive, based on the score differential. Another example considers the end of the game (or when the score differential is very high), when the leading team is likely to run out the play clock with the quarterback kneeling when in possession of the ball. However, this practice varies with teams, and some will favor offensive drives in an attempt to score until the end of the game, which comes with associated risks of turnovers. Coordinators equipped with hypothetical predictions about opponent endgame behaviors can then adjust the offensive drive intensities, as well as defensive prevention schemes, among many other controllable factors, to take advantage.

A greater challenge arises from choosing relevant features. An overabundance of factors can not only confound the model and decrease the predictive accuracy, but can also significantly slow down the client-side processes, hence diminishing commercial practicability. Unnecessary factors not only make training the model more timely, but can also decrease the quality of client-side usage. More features meansmore inputs during in-game scenarios. Time is a significantly limiting variable, given that NFL regulationsonly allow 40 seconds in between each play. Variables that consume time yet do not yield comparable increases in prediction accuracy must not be equated into the final product.

Our solution is a multi-layered analysis using feature importances [11]. All 15 potential features were first gauged for predictive influence, and although some factors distinguished themselves on the extremes of the spectrums, others were prone to confoundment due to the mixing of variables. This warranted a simple manual grid-search using a small subset of potential factors similar in influence. In the end, the 15 potential features were minimized into a small subset of 7 showing maximum accuracy; each feature was considered simple for client-side entry, hence increasing efficiency.

## 2.2. Competing for user attention with other professional products/past papers

We have found several past papers, alongside numerous community projects, with similar goals of predicting football play-calling, with the oldest dating from 2015. However, they were primarily concerned with implementation solely on the professional level. Hence, they incorporated numerous NFL-specific features, such as Madden player ratings, that cannot be easily generalized to other levels of play. Our paper is distinguishable from past papers in that we seek to bring play prediction to all levels of play, using simple yet sufficient models [12].
Furthermore, we did not find any similar commercial product that shares a common target. Some challenges that may arise from the app-implementation include the creation of a streamlined and efficient UI, capable of importing large amounts of training data, as well as providing a high amount of customizability for the consumer's preferences. Similarly, the former challenge warrants compatibility with existing popular game-film sharing systems, and the latter would require the ability of choosing specific datasets and features to predict from. An example of the latter would be using the dataset of all past opponent games versus using solely the dataset of past opponent games against a specific team, since play-calling variability varies drastically between play-levels. Furthermore, organizations may have ethical regulations that may conflict with the use of said technologies on the field.

## 2.3. Determining how to measure model accuracy

The dataset we utilized was expansive, providing data from 10 NFL seasons. This allowed a high variability of metrics for measuring the model's accuracy. The challenge arises from the consideration of real-life scenarios. For a commercial product, a possible utilization is providing predictions during an ongoing game. Hence, a possible accuracy metric is comparing model predictions versus actual results in a specific game. However, another consideration arises from the potential high variability of a team's playbook throughout the season, or, on a greater scope, throughout the franchise's short-term history. Hence, we decided that an appropriate accuracy metric would be cross validation with 5 folds. Roughly speaking, 5 fold cross validation allows a model trained with solely 80% of the data to be tested across 2 NFL seasons, which amounts to around 32-46 games.

We believe that using such a metric, in conjunction with the resulting model accuracies, has led to discoveries of generalized play-calling trends and tendencies throughout the NFL. For example, the fact that our model achieved a high validity through 2 NFL suggests that play-

calling tendencies are rather innate, and not significantly influenced by a specific opponent, hence reducing the perceived variability in NFL play-calling.

## 2.4. Finding data and SMOTE

Traditionally, professional sports data has been long privatized, with only a small fraction of relevant data making its way to the public through media outlets. However, recent years saw the rise of publicly maintained sites documenting a wide range of professional sports data, from player drafts and trades to detailed post-game performance reports. This, in conjunction with recently developed web-scraping libraries targeting professional sports databases, form the premise that made this paper possible.

Due to extensive testing by previous papers on past NFL seasons, our original intention was to switch focus to a different level of play. We settled upon high-school game films as the most appropriate candidate, since college football is heavily similar to the professional level, and levels below high-school are hard to find due to the scarcity and obscurity of their datasets. However, high-school data being the candidate would result in significant limitations in research capabilities. Due to the constrictions of high-school level data, we are provided with less features, as well as less data. Moreover, the level of dynamism in high-school level play is, intuitively, higher than those at the professional levels. Personnel are almost completely replaced each year, and the skill and performance vary significantly per team. With our main intention being the development of a sound model, we chose to avoid these self-imposed limitations, and instead chose past NFL seasons as our research dataset.

Another challenge also arises from the fact that the dataset is intuitively skewed. Visualization of the types of data reveals that the play types are heavily skewed. Running and passing plays More details can be found in later sections describing methodologies.

## 2.5.Choosing optimal model & optimizing hyperparameters (accuracy and speed)

Once the metric for measuring accuracy was established, we turned to evaluating different metrics using said accuracy metric. This took place before the features were optimized. We chose several popular classifier models, since there was a high variability in performance and efficiency. Models quick to train with data usually performed less adequately against those with longer training times. Due to the fact that model retraining is infrequent and not required, we prioritized accuracy over efficiency. In the end, the model with the best accuracy was a random forest classifier. Interestingly, the traditional heuristics that coaches rely on are heavily similar with the model's decision trees. However, the decision tree is limited inthe expansiveness of the feature set that it can utilize.

Following work was done on optimizing hyperparameters using a grid search cross validation. There were several popular and significantly influential settings that we considered, ranging from the number of trees in an ensemble to the max depth reached by each tree. The grid search sets were limited with respect to time restraints. For example, even though an increase in n-estimators may boost model accuracy, its tradeoff is significantly slower training times and potentially renders the possibility of in-game model retraining futile, not to mention slowing down prediction processes. Similarly, hyperparameters, such as max depth, have the potential to increase efficiency at the cost of model accuracy. We found that default hyperparameters kept the time costs sufficiently low, while not sacrificing accuracy. More of our optimization methodologies can be found in the next section.

## 3. SOLUTION

A mobile app [PlayGuessr] trains a machine learning classification model using the play conditions (eg. down, time, score differential) as inputs and play type as outputs (eg. run, pass, punt). Firstly, we implemented the app to receive the client's desired training dataset, as well as any prediction inputs. For ease of customizability, the training data is categorized by games. This can allow the client to select, with specificity, the data that the model will train with.

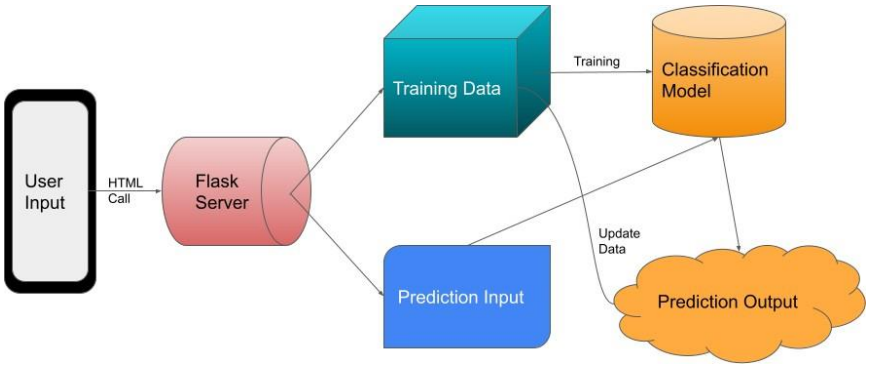


Figure 1. Overview of the app

As Figure 1 demonstrates, the model trains with the client-selected datasets and can readily take inputs from the client. The client's inputs are sent to the model, which returns a play type prediction from the client' s inputs. After the play, the client is prompted to record the actual play type in the app. The app maintains a dataset of the past plays during an ongoing game and gives the client the option to retrain the model with said plays.

PlayGuessr is implemented as follows. Firstly, the backend is divided into three main components. The training set is produced from user-provided data. We used the Pandas library for Python to parse our intended features and convert Data Frames into training sets. SMOTE was performed on the training data, with a K-Neighbors value of 2. The second backend component is our machine learning model. After obtaining our training data, we fitted a random forest classifier model using said data. The model is now ready to make predictions. Thirdly, the Flask module for Python web framework was utilized to set up a server for our predictive algorithm.

A design prototype (Figure 3) is provided below, demonstrating the chronological relationships between each screen.

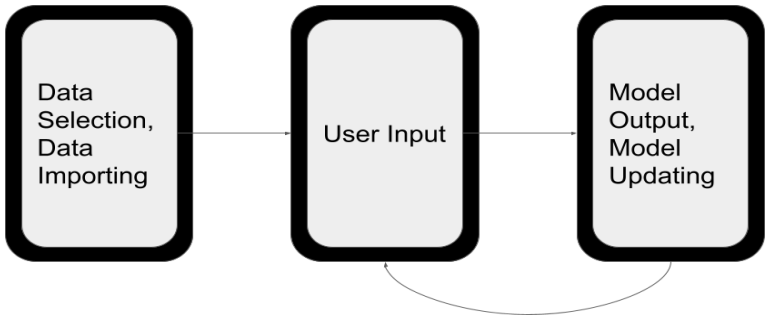


Figure 2. Design prototype

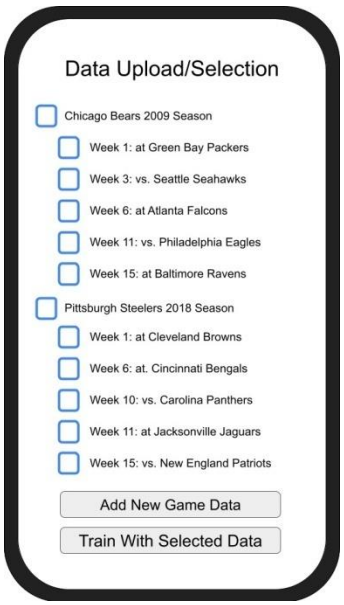


Figure 3. Uploading and Selecting Training Set

The app will first prompt the user to choose their datasets to use for model training (Figure 3). After the training set is established, the backend will fit the model accordingly, and predictions are ready to be made. The user will be taken to the input screen (Figure 4).


Figure 4. Input Screen

The input screen consists of six out of the seven features, due to net yards per offensive drive hard to keep track by the user. Instead of traditional textboxes, we have implemented buttons for features with discrete values. After the features are selected, the user will prompt the app to send a server command to run our predictive algorithm. The results will be returned in the final output screen (Figure 5).

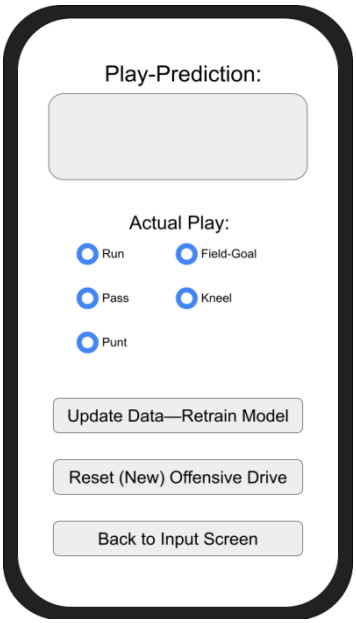


Figure 5. Output Screen

After the play, the user will be prompted to provide the actual resulting play. The actual output is kept in a temporary list in the backend, alongside the provided input features. After every prediction, the user will be provided the ability to update the training dataset with new information from the ongoing game. Due to the time expenses associated with retraining a random forest classification model, it will be up to the user to update the model at a time of convenience. Moreover, the net yards feature will be implemented in the output screen, allowing the user to end an offensive drive to effectively reset the net yardage to zero. The user will be returned to the input screen, where the next offensive prediction can be made again.

The following figures (Figures 6, 7, 8) demonstrate the various functions utilized in the processes described above, primarily to parse and construct training data, as well as evaluate models.

```
def classify_play(play):
    if play == "run":
        return 0
    elif play == "pass":
        return 1
    elif play == "punt":
        return 2
    elif play == "field_goal":
        return 3
    elif play == "qb_kneel":
        return 4
```

Figure 6. Code for Play Type Vectorization

```
    x, y = oversample.fit_resample(in_data, out_data)

    return x, y

def test_team(team_name):
    # Handling exceptions (teams changing teams & location)
    if team_name == "JAX":...
    elif team_name == "SD":...
    elif team_name == "STL":...
    else:
        input_data, output_data = gen_data(team_name, init_data)

    # SMOTE
    oversample = SMOTE(k_neighbors=2)
    x, y = oversample.fit_resample(input_data, output_data)

    # Model Fitting
    model = RandomForestClassifier(random_state=0)
    model.fit(x, y)

    # return model.feature_importances_
    # return avg(cross_val_score(post_model, x, y, cv=5))
    return model
```

Figure 7. Code for Experimentation

```
def gen_data(team_name, initial_data):
    # Processing the NFL dataset (checking for valid values)
    initial_data = initial_data[(initial_data["posteam"] == team_name)
                                & ((initial_data["play_type"] == "run") | (initial_data["play_type"] == "pass")
                                   | (initial_data["play_type"] == "punt") | (initial_data["play_type"] == "field_goal")
                                   | (initial_data["play_type"] == "qb_kneel"))
                                & ((initial_data["down"] == 1) | (initial_data["down"] == 2)
                                   | (initial_data["down"] == 3) | (initial_data["down"] == 4))
                                & ((initial_data["game_half"] == "Half1") | (initial_data["game_half"] == "Half2"))
                                & ((initial_data["half_seconds_remaining"] > 0) & (initial_data["game_seconds_remaining"] > 0))]

    input_data_ids = ["down", "yardline_100", "shotgun", "half_seconds_remaining", "game_seconds_remaining", "ydsnet", "ydstogo"]
    output_data_ids = "play_type"

    home_games = initial_data[initial_data["home_team"] == team_name]
    home_games = home_games[input_data_ids]

    away_games = initial_data[initial_data["away_team"] == team_name]
    away_games = away_games[input_data_ids]

    home_plays = initial_data[initial_data["home_team"] == team_name]
    home_plays = home_plays[output_data_ids]

    away_plays = initial_data[initial_data["away_team"] == team_name]
    away_plays = away_plays[output_data_ids]

    home_plays = home_plays.apply(classify_play)
    away_plays = away_plays.apply(classify_play)

    in_data = home_games.values.tolist()
    in_data.extend(away_games.values.tolist())

    out_data = home_plays.values.tolist()
    out_data.extend(away_plays.values.tolist())
```

Figure 8. Code for Parsing Data

# 4. EXPERIMENTATION

## Experiment 1

Note: Code snippets will be provided to detail the exact experiments taking place. We created two functions, gen_data (used to generate training data, given a specific team, see Figure 9), and test_team (performing SMOTE [when necessary] and fitting model with training data, able to return the model, CV scores, and feature importances). Please refer to Section 3 for the

implementation of said programs. These two functions may be slightly altered with our following experiments.

The first experiment seeks to determine the relevancy of SMOTE, or Synthetic Minority Oversampling Technique. Intuitively, running and passing plays constitute the majority of plays called, significantly overshadowing situational plays (punts, field-goals, and quarterback kneels). The following table (Table 1) and PyPlot graph (Figure 9) describes the total occurrences of each play type over the 2009-2018 seasons. Another PyPlot Graph is an average of each play type's occurrences averaged per NFL team. Given an imbalanced dataset, we wanted to test the effectiveness of SMOTE in model accuracy, using a basic SVM classification model as our metric (using all available features).We applied SMOTE to our training data using a baseline K-Neighbors value of 2. We trained one model with the SMOTE undergone data, also keeping a control model trained with non-SMOTE data. We then compared average 5-fold cross validation results from the two models. This was done using the Chicago Bears, but results can be generalized to other teams as well, and the degree of efficacy is predicted to be dependent upon the varying play type ratios of each team. Those with higher disparities are predicted to benefit more from SMOTE. A Pyplot of the post-SMOTE training data for play types can be found below as well.

Table 1. Pyplot of the post-SMOTE training data

| **Play Type** | **Total Occurrences** | **Average Occurrences** | **Percent of Total Plays** |
|---|---|---|---|
| Run | 126,663 | 3,958.22 | 37.22% |
| Pass | 177,933 | 5,560.41 | 52.29% |
| Punt | 22,832 | 713.50 | 6.71% |
| Field-Goal | 9,249 | 289.03 | 2.72% |
| Quarterback Kneel | 3,603 | 112.59 | 1.06% |

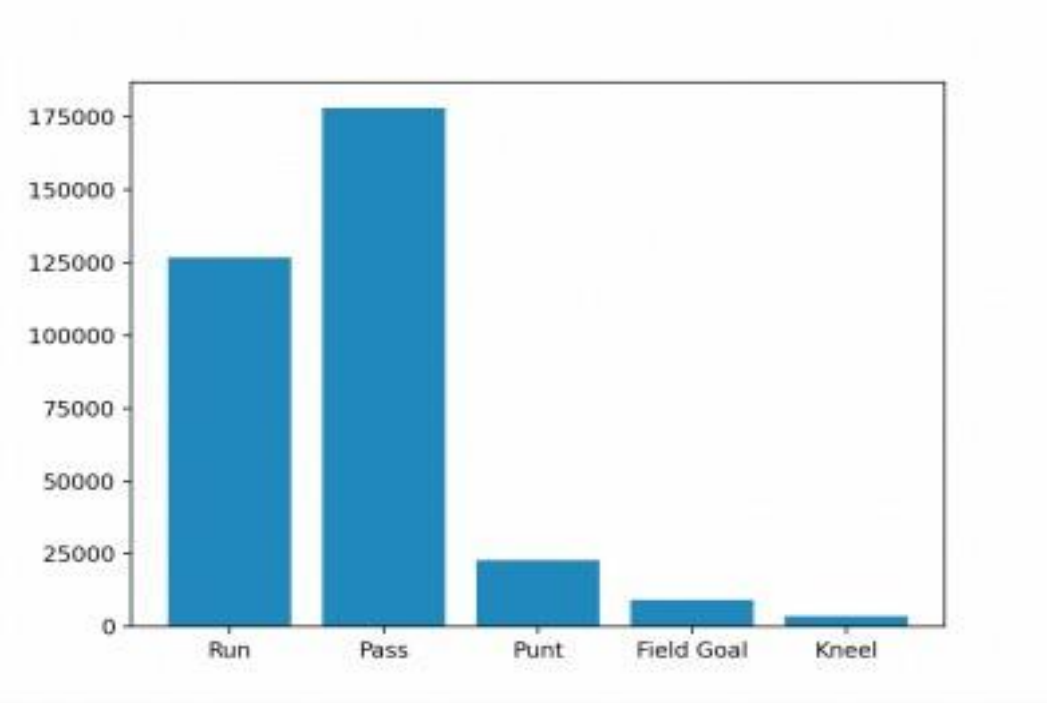


Figure 9. Total Occurrences in the NFL Per Play Type

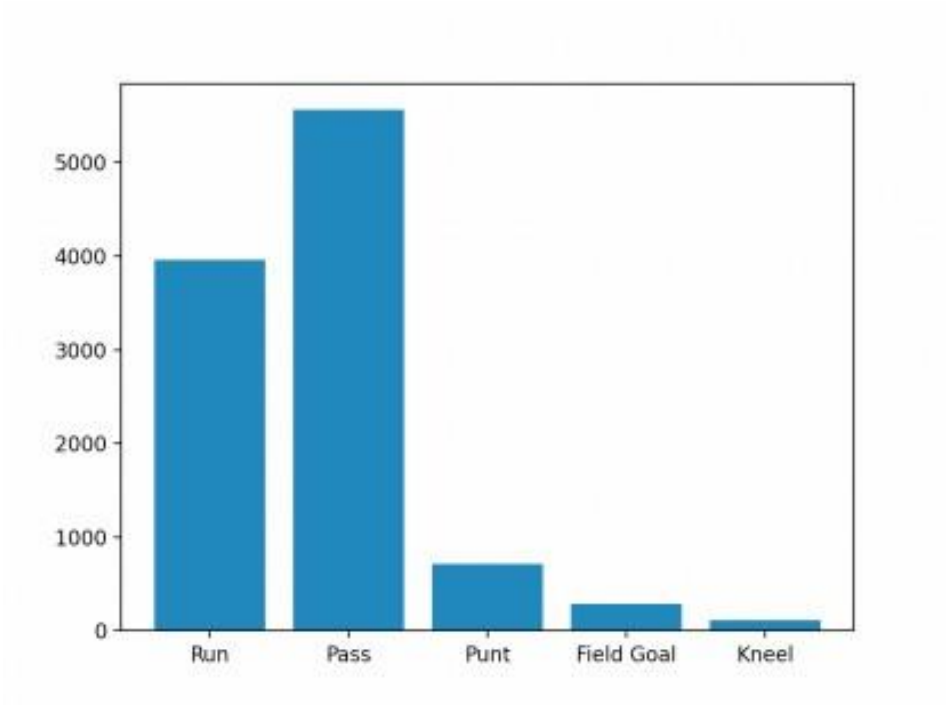


**Figure 10. Average Occurrences Per Team Per Play Type**

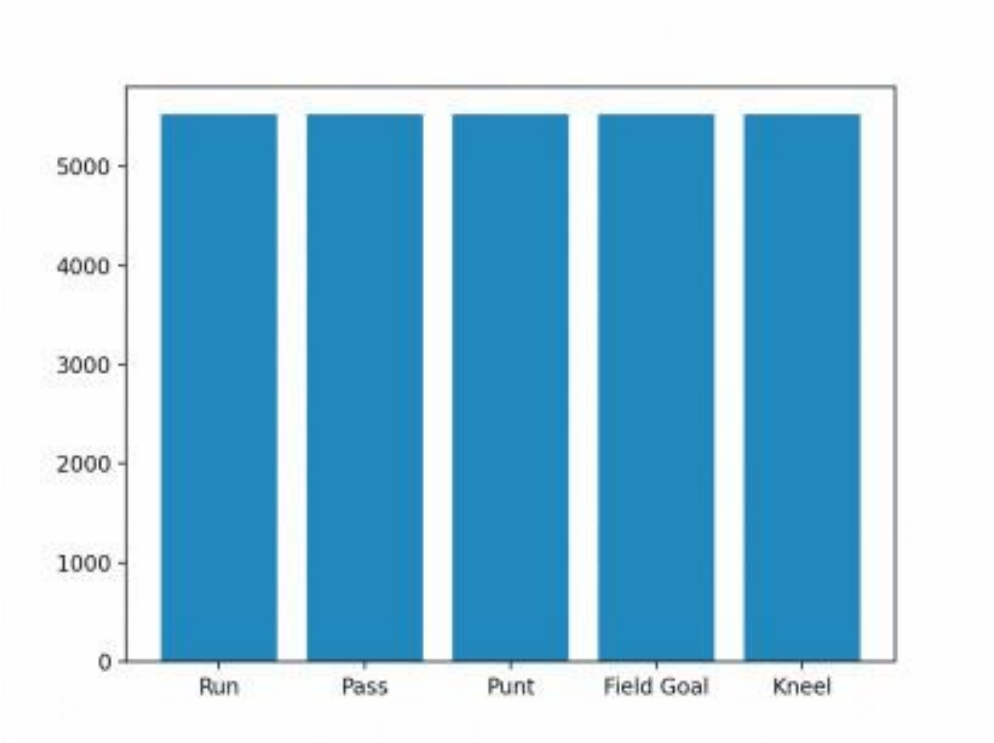


Figure 11. Post-SMOTE Distribution for the Chicago Bears

Table 2. Table of accuracy

| Control SVM Accuracy | Experimental SVM Accuracy (SMOTE) |
|---|---|
| 51.73% | 57.41% |

The results are as followed: Above, Figure 11 demonstrates the resulting distribution for the Chicago Bears after SMOTE was performed. Furthermore, Table 2 (above) describes the resulting model accuracies for both pre-SMOTE and post-SMOTE models. Code for the experiment can be found in Figure 12.

```
# gen_data() returns dataset without performing SMOTE
pre_input, pre_output = gen_data('CHI', init_data)# before SMOTE
model = svm.SVC().fit(pre_input, pre_output)
print(avg(cross_val_score(model, pre_input, pre_output, cv=5)))
# perform SMOTE
oversample = SMOTE(k_neighbors=2)
post_input, post_output = oversample.fit_resample(pre_input, pre_output)
model = svm.SVC().fit(post_input, post_output)
print(avg(cross_val_score(model, post_input, post_output, cv=5)))
```

Figure 12. Code for Experiment 1

As the results demonstrate, SMOTE was capable of elevating a basic SVM classification model's

accuracy by 10.97%. We discovered that SMOTE is essential for boosting accuracy due to the imbalance nature of our datasets. Further experimentation with increasing K-Neighbor values did not yield any significant nor constant upwards trends in model accuracy, and the K-Neighbor value was determined to be at two in order to compensate for future time expenses. The following experiment explores the SMOTE-induced accuracy enhancements with different popular classification models.

## Experiment 2

We have discovered that SMOTE significantly boosts a model's accuracy due to the imbalance in our datasets. However, different classification models respond differently to SMOTE, and their increases in accuracy also vary. This experiment aims to find the most optimal post-SMOTE model. We considered 5 popular classification models: SVM, Random Forest, Naive Bayes (Gaussian NB), Passive Aggressive, andGradient Boosting. We chose to remain with the Chicago Bears for our testing data. For features, we have included all available features as a baseline. All hyperparameters are set at default values, as provided by Scikit-Learn, except for a constant and uniform random state. The following table shows results for all 5 models. Each model is trained on both pre-SMOTE and post-SMOTE data (with K-neighbors remaining at 2) and evaluated by a 5-fold cross validation metric. The percent increase in performance is also recorded in Table 3. Note: the code from Experiment 1 was reused with slight modifications by substituting various models in place of SVM.

Table 3. Results for 5 models

| Model | Pre-SMOTE | Post-SMOTE | % Increase |
|---|---|---|---|
| SVM | 51.73% | 57.41% | 10.97% |
| Random Forest | 71.67% | 88.82% | 23.94% |
| Naive Bayes | 65.75% | 81.30% | 23.66% |
| Passive Aggressive | 48.74% | 57.50% | 17.98% |
| Gradient Boosting | 74.04% | 87.95% | 18.79% |

Our data demonstrates that there is a high variability between the effects of SMOTE on final model accuracy. We decided upon Random Forest as our model of choice, as it yielded the highest post SMOTE accuracy. This experiment further supports our previous experiment demonstrating the pivotal role of SMOTE. Gradient boosting is another highly favorable candidate with the second highest post-SMOTE accuracy. Had our experiment been done solely on non-SMOTE datasets, we would have chosen gradient boosting as the most optimal model, since it has the highest pre-SMOTE accuracy.

## Experiment 3

The previous experiments have yielded a desirable data set and a competent model. The final experiment seeks to define the optimal subset of features with the highest accuracy and intuitive convenience. We predict that due to different teams having varying priorities, an optimal feature subset forthe Chicago Bears might not be optimal for teams differing in values and priorities. To

mitigate team-specific feature selection, we performed feature analysis on all 32 NFL teams with all available features. We used random forest classifiers along with post-SMOTE datasets. The team-specific data is then averaged to obtain an appropriate generalized feature list for the entire league.

The below table (Table 4) and PyPlot (Figure 13) demonstrates the relative average importance of ourfeature set. Note: the function test_team was modified to return the feature importances of the random forest classification model trained from the team's data.

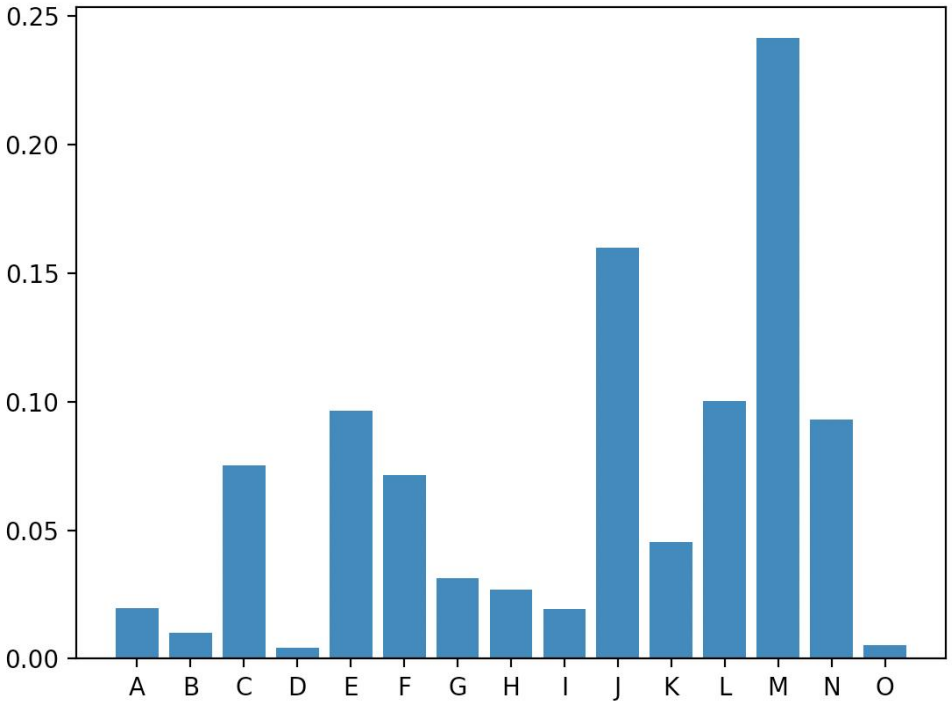


Figure 13. The relative average importance of our feature set chart

Table 4. Average Importances for All Features

| Feature | Importance (%) |
|---|---|
| A. Timeouts Remaining | 1.96% |
| B. Quarter | 1.02% |
| C. Quarter Seconds Remaining | 7.52% |
| D. Half | 0.42% |
| E. Half Seconds Remaining | 9.65% |
| F. Game Seconds Remaining | 7.16% |
| G. Score Differential | 3.12% |

| | |
|---|---|
| H. Offensive Team Score | 2.69% |
| I. Defensive Team Score | 2.69% |
| J. Yardline | 15.99% |
| K. Yards to Go | 4.52% |
| L. Net Yards (On Drive) | 10.03% |
| M. Down | 24.14% |
| N. Shotgun (Formation) | 9.31% |
| O. No Huddle | 0.52% |

From the average feature importances, our set of 15 features can be ranked by their percentages. The ordergoes as follows: down, yardline, net yards on a drive, half seconds remaining, shotgun, quarter seconds remaining, game seconds remaining, yards to go, score differential, offensive/defensive team score, timeouts remaining, quarter, no huddle, half.

We then trained the model 15 times, starting with solely the most influential feature, adding the next most influential feature every iteration, as well as recording the league average accuracy for each iteration. We seek to set a pruning point in our features set to discard any irrelevant features bringing insufficient boosts in accuracy in return for added inconvenience for client-side management.

The following table (Table 5) describes our findings, as well as a PyPlot graph (Figure 17) that charts theprogress in accuracy boosting per feature added. Code for the experiment can be found in Figure 14.

```
teams = ["ARI", "ATL", "BAL", "BUF", "CAR", "CHI", "CIN", "CLE",
         "DAL", "DEN", "DET", "GB", "HOU", "IND", "JAX", "KC",
         "OAK", "SD", "STL", "MIA", "MIN", "NE", "NO", "NYG",
         "NYJ", "PHI", "PIT", "SF", "SEA", "TB", "TEN", "WAS"]
features_ranked = ['down', 'yardline_100', 'ydsnet', 'half_seconds_remaining', 'shotgun',
                   'quarter_seconds_remaining', 'game_seconds_remaining', 'ydstogo',
                   'score_differential', 'posteam_score', 'defteam_score', 'timeouts',
                   'qtr', 'no_huddle', 'half']
current_features = list()
accuracies = list()
for i in range(len(features_ranked)):
    current_features.append(features_ranked[i])
    league_accuracy = list()
    for team_name in teams:
        league_accuracy.append(avg(test_team(team_name)))
    accuracies.append(avg(league_accuracy))
print(avg(accuracies))
```

Figure 14. Experiment 3 code

Table 5. The progress in accuracy boosting per feature added

| Feature Added | Cumulative Model Accuracy | Accuracy Difference | Increase In Percentage |
|---|---|---|---|
| A. Down | 44.56% | N/A | N/A |
| B. Yardline | 79.30% | +34.74% | +77.97% |
| C. Net Yards | 84.71% | +5.42% | +6.83% |
| D. Half Seconds Remaining | 86.34% | +1.62% | +1.92% |
| E. Shotgun | 88.81% | +2.47% | +2.86% |
| F. Quarter SecondsRemaining | 88.96% | +0.14% | +0.16% |
| G. Game Seconds Remaining | 89.04% | +0.082% | +0.092% |
| H. Yards to Go | 89.62% | +0.58% | +0.65% |
| I. Score Differential | 89.59% | -0.028% | -0.031% |
| J. Offensive Team Score | 89.40% | -0.19% | -0.22% |
| K. Defensive Team Score | 89.16% | -0.24% | -0.27% |
| L. Timeouts Remaining | 89.06% | -0.10% | -0.12% |
| M. Quarter | 89.02% | -0.036% | -0.041% |
| N. No Huddle | 88.94% | -0.082% | -0.092% |
| O. Half | 88.92% | -0.023% | -0.026% |

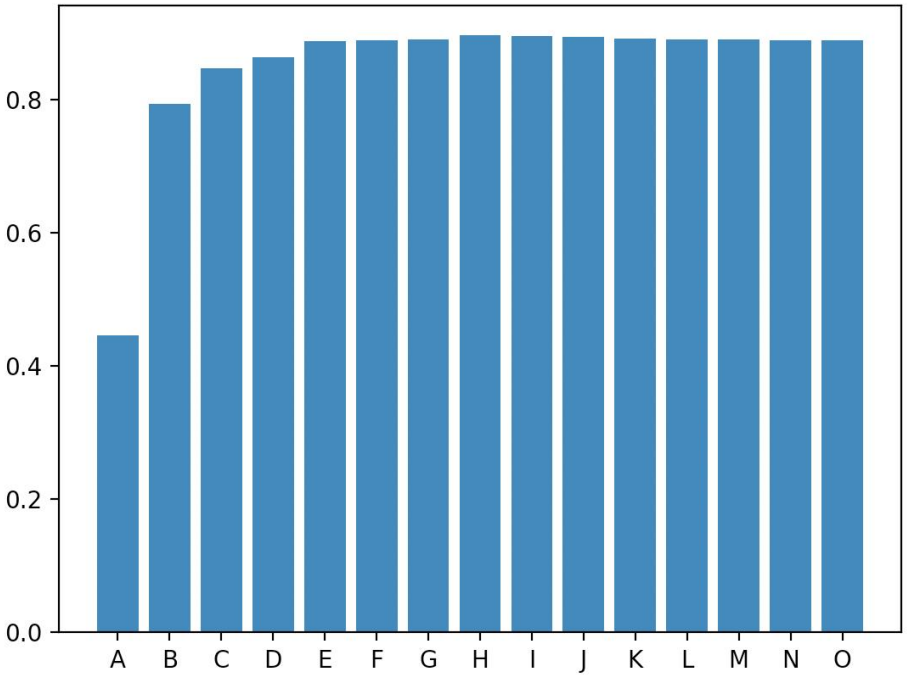


Figure 15. The progress in accuracy boosting per feature added chart

The graph (Figure 19) shows that model accuracy quickly rose and plateaued. The graph also reveals that model accuracy was maximized when the feature set consisted of factors A-H. The table confirms the absolute maxima of 89.62%. Furthermore, each successive addition of features resulted in a decrease in model accuracy. The results have defined our final feature set as follows: down, yardline, net yards, half seconds remaining, shotgun, quarter seconds remaining, game seconds remaining, yards to go.

## Discussion

The first experiment was designed in response to the imbalanced nature of our dataset. As previously discussed, the imbalance in the dataset arises from the comparatively rare occurrences of offensive special plays, involving punts, field-goals, and quarterback kneels. Hence, our challenge was whether advanced techniques that remedy imbalanced classification is practicable and desirable in our project. The experiment validated our hypothesis, and the subsequent experiment further supported our predictions, with all models showing significant yet varying degrees of improvement after the training data had undergone SMOTE.

Our second experiment was tailored towards the challenge of choosing the optimal model. Past papershave been exceptionally widespread in their classification models of choice, and we evaluated a small set of popular classification models for their accuracy. The result demonstrated that random forest and gradient boosting classifiers were the strongest candidates, with gradient boosting holding the highest accuracy with pre-SMOTE training data, and random forest with post-SMOTE training data. Both classification models are good candidates, yet time complexities, as well as our utilization of SMOTE, drove us to choose random forest as our model of choice.

Our final experiment targeted the challenge of optimal feature selection in an effort to maximize model accuracy as well as minimize potential confounding usage. The accuracy vs feature set graph defined a clear absolute extremum. Although alternative evaluating metrics, such as analysis of variance with multiple factors, may be better for feature selection, our method of evaluation is less computationally expensive and also does not lie far from the optimal feature set.

## 5. RELATED WORKS

Joash Fernandes et al (2020) sought to design an intuitive and interpretable model for predicting run/pass plays in the NFL. The authors utilized a significantly smaller dataset compared to ours, consisting of the 2013-2016 NFL seasons. The authors considered the following features: year, quarter, minute, second, down, yards to go, yardline, and offensive formation; these are heavily similar to our feature set. The authors also derived many features, including the previous play, home/away game, point differential, passing and completion proportions, average yards per play type, and average position group scores (weighted by the position player count on the field). The paper experimented with complex models, assessing classification trees, K-nearest neighbors, random forests, and neural networks, which constitutes a more expansive list than our models. Similar to our model, the authors chose a decision tree model, however, with limited splits.

The finalized model achieved a lower accuracy of 65.3% with only three variables—down, point differential, and yards to go—and ten splits. Although the model was trained from league-wide data, team-specific trees scored accuracies between 64.7% to 82.5%, which had significantly higher variance than our results.

Lee et al (2016) set out to build upon past research to develop a model featuring more diverse features, primarily formation and player ratings. The paper utilized past NFL data from the 2011-2014 seasons, which is significantly smaller than the size of our training set, although featuring more details. The authors chose the following features from their dataset: score difference, quarter, time left in quarter, down, yards to go, player counter per position, formation, out of position players, turnovers, and home/away games. The authors settled on team-on-league type data. Moreover, the feature set contains many derived features, such as positional group rankings.

The authors experimented with four different models: logistic regression, linear discriminant analysis, gradient boosting machine, and random forest, most of which were assessed in our paper as well. Gradientboosting scored the highest, with a 75.7% accuracy, which is in agreement with our findings. However, the authors seeked to combine models in order to achieve an optimal accuracy, and they reported that a combined model with gradient boosting having a 60% and random forest a 40% weight showed the best performance, with an accuracy of 75.9%. The team was also able to achieve a game-specific accuracy ranging from 47.2% to 94.6%, as well as a season-specific accuracy ranging from 67.6% to 86%, which isalso more wildly variant than our results.

Ota (2017) presents a unique viewpoint by advising against using raw statistics, due to lack of specificscenarios and resulting discontinuous models. Instead, Ota advocates for a more ideal “aggregate” model.

Ota’s study utilized solely one NFL season, from 2016, which is significantly and concerningly smaller. The author included the following features: down, yards to go, score differential, game time remaining, and yardline. The list is marginally smaller than ours, as well as containing all static and observable variables.

The study performed experimentation on a neural network with customizable hyperparameters, which we did not assess. It must be clarified that models were trained separately per down. The average accuracy, 68.9%, was higher than the baseline run/pass ratio driven model accuracy, which scored 61.3%. It was noted that as the downs incremented, the model accuracy rose considerably, with 3rd down scoring the highest, at 86.8%.

The study was extended into research of situational-based models. Logistic regressions, linear regressions, support vector machines, and random forests were tested for accuracy, all of which were repeated in our study. Ota chose to reduce the feature set even more, with only team, down, distance, and yardline remaining. Holistic average accuracy proved to be at 65.67%, only a slight improvement over the naïve tendency-based model, which scored 62.86%. Both experiments yielded lesser accuracies than our results.

While Ota put forth two perspectives—aggregate and situational—both suggest that their practicality is not fully realized. The aggregate model is too broad in its reach to offer specific insight, and much of the general trends revealed are too intuitive to offer insight. Perhaps if the aggregate model was more specified towards a certain area, it would serve more value to coordinators [13].

## 6. CONCLUSIONS

In this paper, we addressed the potential of football in-game predictions. We experimented with dataset engineering techniques, such as SMOTE, to satisfy our eccentric target set that deviates from previous studies in its inclusion of offensive special plays. We also experimented with different machine learning classification models, and found that random forest and gradient boosting both have high-performances. We chose to advance with random forest due to time complexities, and in an effort to further minimize potential confounding variables during deployment, we sought to reduce our feature set using feature importances.

The experiments proved to be highly successful in responding to the encountered challenges. The experiment design was focused around both team-specific and league aggregate evaluations to minimize potential bias with outlying teams. The experimental results show that our model had a clear advantage over previous models, with a 89.52% accuracy across the league. This confirmed that offensive play-calling tendencies are independent of the defense and deeply rooted in the offensive team itself.

### Limitations

Overall, we were unable to predict and evaluate the effects of deployment in other levels of play. The NFL dataset is expansive and may not be attainable in the same manner, whether it be personnel longevity or league expansiveness, in college and high-school level play.

Furthermore, hardware limitations prevented experiments from reaching further complexity. Hyperparameter tuning and analysis of variance were excluded for time expenses.

Even more, the dataset presented limiting feature sets. We also noticed that the dataset featured erroneous data, primarily for score differential, although the error is marginal and rare.

The choice to incorporate more plays into our label set meant that certain models, such as those that govern two-target predictions (logistic regression, for example) were made unavailable.

## Future Works

A major area remains on the practicability of our model's deployment in lower level play. Further research will be done on whether the conditions of high-school and college football, such as higher player turnover rates and smaller datasets, warrants adjustments to our model.

Furthermore, more features can be evaluated for their influences. Some appealing features include gameday conditions, and specific offensive formations. The label set can also experiment with specificity. For example, expanding the target set to include all possible offensive plays, or providing multi-class outputs for different levels of specificity. Even more, deep learning models, especially those of recurrent neural networks, have yet to be tested.

## REFERENCES

[1] Lee, Peter, Ryan Chen, and Vihan Lakshman. "Predicting offensive play types in the National Football League."(2016): 1-5.
[2] Pincivero, Danny M., and Tudor O. Bompa. "A physiological review of American football." Sports Medicine23.4 (1997): 247-260.
[3] Ota, Karson L. Football play type prediction and tendency analysis. Diss. Massachusetts Institute of Technology, 2017.
[4] Ötting, Marius. "Predicting play calls in the National Football League using hidden Markov models." arXiv preprint arXiv:2003.10791 (2020).
[5] Czaczkes, Benjamin, and Yoav Ganzach. "The natural selection of prediction heuristics: Anchoring and adjustment versus representativeness." Journal of Behavioral Decision Making 9.2 (1996): 125-139.
[6] Yashiro, Kotaro, and Yohei Nakada. "Computational Method for Optimal Attack Play Consisting of Run Plays and Hand-pass Plays for Seven-a-side Rugby." 2020 IEEE International Symposium on Multimedia (ISM). IEEE, 2020.
[7] Joash Fernandes, Craig, et al. "Predicting plays in the National Football League." Journal of Sports Analytics 6.1 (2020): 35-43.
[8] Cook, Sally. How to Speak Football: From Ankle Breaker to Zebra: An Illustrated Guide to Gridiron Gab.Flatiron Books, 2016.
[9] Ferryman, James, and Ali Shahrokni. "Pets2009: Dataset and challenge." 2009 Twelfth IEEE internationalworkshop on performance evaluation of tracking and surveillance. IEEE, 2009.
[10] Shen, Ting, et al. "Decision supporting model for one-year conversion probability from MCI to AD using CNN and SVM." 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2018.
[11] Maina, James, and Kunihito Matsui. "Elastic multi-layered analysis using DE-integration." Publications of the Research institute for Mathematical Sciences 41.4 (2005): 853-867.
[12] Goyal, Udgam. Leveraging machine learning to predict playcalling tendencies in the NFL. Diss. Massachusetts Institute of Technology, 2020.
[13] Gray, Philip K., and Stephen F. Gray. "Testing market efficiency: Evidence from the NFL sports betting market." The Journal of Finance 52.4 (1997): 1725-1737.