Matthew Ball
Professor Shayok Chakraborty
CAP 5778
17 April 2023

Predicting Fourth Down Decisions in the NFL Using More Data Mining Techniques

I. Introduction

In this project, football analytics are researched to analyze essential decisions in football about whether a team should execute a play on fourth down or kick it away. A down is an attempt to move the ball ten yards in the direction of the opponent's goal line. A fourth down attempt is when the offense of a team has its final opportunity to perform another offensive play. If the play meets the distance that is needed, the team will either score or achieve a first down, and in a first down, the team can run four additional plays. However, if a fourth down attempt fails, the other team will get possession of the ball on the location where the play has taken place.

Another decision that can be made is to kick the ball on the fourth down, which takes the form of either punting or kicking a field goal. In punting, the team in possession punts the ball to the defending team, who takes possession of the ball. Since the average NFL punt lasts 45 yards, the defending team gets a more difficult possession to start the next play with [1]. This can have effects on how the play will turn out, which is important to the game since if the defense has the ball, they are more likely to score and get more points. This can affect the outcome of the game because it allows the defending team to be closer to winning, since they have more points and are therefore closer to achieving more points than the offense. Therefore, the decision to kick the ball on fourth down is an important one that could affect the results of a game and is therefore one of the scenarios that is analyzed as one of the things that could happen on a fourth down.

The other decision than punting on a kick is kicking a field goal, where the offensive team's kicker attempts to kick the football behind the endzone and through the goalposts. If this kick is successful, the offensive team gets 3 points. However, if it misses, the defending team gets possession of the ball at the kick's destination, meaning that a successful kick can only be done within 52 yards [2]. This decision is important because giving the other team the ball when they are in a good field position could give them an advantage and impact on the game's outcome. Because of the importance of these decisions, NFL teams often rely on statistics.

This project is a sequel to one that I worked on in Fall 2022 Data Mining. It was worked on in a group with Noah O'Connor, David Boyd, and Andrew Franklin. In that project, we tested with random forest, gradient boosting, K-nearest neighbors with/without SMOTE, and support vector machines with/without SMOTE. We found that gradient boosting had the best results with an F1 score of 0.886. Some parts of this report may be similar because of the two papers covering the same subject matter. In this project, I will test with more data mining techniques to find one with a greater F1 score than the best one in the original project.

II. Literature Survey

There are a few papers that we derived inspiration from when working on the previous project, which carried over to this one. One paper attempted to use data mining techniques to predict the outcome of soccer matches in a Portuguese league [3]. Like what is being done in this project, Nunes and Sousa used a classification algorithm, the one being a C4.5 decision tree. They also tuned the parameters for confidence and necessary instances for each leaf. However, they did not get optimal results because they did not have enough features. This project should have enough features to help predict fourth down decisions.

Another paper tried to use data mining techniques to determine the outcomes of soccer games [4]. This model uses data from the 2013-2016 English Premiere League games. Like this project, they took data from games and used different models, including decision trees, k-nearest neighbors, and Bayesian networks. They used data from previous games to calculate decisions, and in a similar manner, my model also uses data from different sports games, testing with different models. There are different models that

are used to calculate the results in different sports games, and the soccer paper uses it to find outcomes of soccer games, while my research involves predicting the fourth down decisions in football games.

The next paper also uses NFL game data, but to predict the outcomes of plays instead of the fourth down decisions [5]. They used a binary value of 0 or 1 for the classifiers, also using different models such as decision trees, k-nearest neighbors, and support vector machines to calculate outcomes of games. Like this project, they used football datasets, binary classification, and different models. A difference between the two projects is that my project works to predict the fourth down decisions, while the paper described in this paragraph uses football games to determine the outcomes of football games. Despite these differences, there are several similarities between the works in the ways that results are achieved in the classification research in sports.

Continuing onwards, these papers have further inspired the current project being worked on here. There is another example of a paper that uses data mining techniques to find what the best decisions are in sports. This paper uses different data mining models to predict the results of NFL and rugby games [14]. The models that they used included artificial neural networks, support vector machines, naïve bayes, decision trees, fuzzy logic, and logistic regression to get their results. While this paper uses data mining techniques to find the results of different games, this project works on predicting the decisions that are made on fourth down. However, their project and the project described here both utilize different data mining techniques.

The next paper uses data mining techniques to predict the performance of a part of a game, though it is a different part than the one described in this paper. Data mining techniques are used to determine the future performance of players in soccer [15]. McCullagh's paper only uses artificial neural networks, while this project tests a variety of data mining techniques to predict information pertaining to a game. In addition, McCullagh's paper tests with the information of an individual player to calculate how they will play in a game, while this project tests with information about the entire team's performance in a game to calculate what plays will be performed in fourth down decisions.

There are data mining papers that involve predicting info from other games. Data mining techniques are used to determine the future results in basketball games [16]. The programmers of this project used Java and the RapidMiner tool to calculate the results with a combination of naïve bayes and linear regression. In this project, individual models are used to predict fourth down decisions, instead of using an already-existing data mining infrastructure. This gives the different models individual weight to find which one is the best, instead of choosing an existing implementation choosing from multiple models to find the most accurate result. In addition, the current project involves predicting the decisions on fourth downs, while the basketball paper predicts overall decisions.

There is a data mining paper that discusses different prediction methods for analyzing the results of college sports games. These techniques are not mentioned by name, but there are different variants of techniques, including visualization methods, association rules, and many others [17]. Yu and company do not go into much detail about how these methods work and how they are used to predict the results of college sports game. However, this paper will go into further detail about the algorithms involved in sports predictions, particularly in the field of finding the decisions that are made on the fourth down in sports games. While Yu and company's paper is a literature review, this paper is more of an experimental paper, including the methods for data mining being described in further detail, including the equations involved and an analysis of the features.

Data mining techniques have also been used to predict the score and winning probability of cricket games. The data mining techniques utilized here include linear regression and naïve bayes, to predict this important info for cricket games that are useful for plays and how they will turn out in the cricket space [18]. Similarly, the model involved in this paper involves different data mining techniques, although there are more than two different techniques that are used to calculate the different fourth down decisions in football. There are multiple models used in both papers, but my paper involves predicting the outcomes of fourth down decisions, while the other paper involves the prediction of scores and winning decisions in fourth down decisions. This paper also had a detailed description of the algorithms used, and in a similar manner, my paper will describe the different algorithms used in data mining as well as show the equations

that are involved, to help readers further understand the data mining algorithms and make them reproducible.

III. Methodology

To help with classification and aid prediction, the runs and passes were set to be 1 and the punts and kicks were set to be 0. As the prediction of fourth down decisions were a binary classification problem, and to continue with the research posed in last semester's project, multiple models were used to determine the fourth down decisions. These models included ridge classifier, naïve bayes, linear discriminant analysis, logistic regression, and bagging with gradient boosting. To calculate the accuracy of these models, an average f1 score over 10 runs was used, along with giving the user the ability to decide between using hyperparameter tuning and not using hyperparameter tuning. These are also compared with the Fall 2022 top model of gradient boosting, to be able to plot the accuracy of these models in comparison to the best model in the previous semester's project, to see if the goal of finding the most accurate model has been achieved.

The main dataset used for training models contained data from games from 2009 to 2018, but to use the same, most recent data as the models in the previous project, only the data from 2018 is used. The over 200 columns in the original dataset were shortened to 21 columns including important information pertaining to the game, including the game ID, the team with possession (abbreviated to posteam), quarter, whether the team is home or away (abbreviated to postteam_type), the team on defense (abbreviated as defteam), the down, the number of seconds remaining in the quarter, the distance in yards that the team with possession is away from the opponent's endzone (abbreviated as yardline_100), drive (the number of plays used by the offensive team until the other team gets the ball), distance from first down line (abbreviated as ydstogo), total yards gained on drive (abbreviated as ydsnet), play type, number of timeouts remaining for the team with possession and the team on defense, the score the team with possession and team on defense have at the beginning of the play, probability of there being no scoring for the rest of the half (abbreviated as no_score_prob), probability of the team with possession achieving a field goal (abbreviated as fg_prob), probability of the team on defense achieving a touchdown (abbreviated as td_prob), expected points added by the team with possession in the play (abbreviated as epa), and the win probability for the team on offense (abbreviated as wp).

Several fourth down conversion statistics were included to represent 2017 and join with the 2018 data, since these pieces of 2018 data were not available when the original project was being worked on and it was closest to the date and the team roster. Another piece of 2017 data included involves the statistics of whether a team makes a kick. This piece of 2017 data was included because if a team's kicker cannot successfully make a kick, achieving the first down is more likely in the case of a play meeting the needed distance in a fourth down. The rushing and passing statistics for 2017 were also included because if a team has a strong offense, they are more likely to attempt a field goal.

There was some data cleaning performed after the datasets were obtained. The 2018 play-by-play datasets and the additional 2017 datasets stored the teams differently since the play-by-play data categorized teams by name while the 2017 datasets labeled teams by city. A dictionary was created to map the team city with team name to correctly merge the datasets. When the two datasets were merged, some kick stats had letters in them, so those letters were removed, and those strings were turned into floats to properly represent the data in models. Some columns were dropped from the merged set, including 1-19 attempts to misses, the team on offense, the team on defense, whether the team is home or away, the play type, and the team names. Since different models were used, they are described below:

i. Ridge Classifier [6][7]

The ridge classifier's implementation consists of three phases. The first phase involved with the ridge classifier is the initialization phase, which involves initializing different parameters. The first one of these parameters is the alpha, which is the regularization constant that is used to reduce the variation of estimations and improve classification. The second parameter is the number of iterations used in the solvers. The sklearn implementation of the ridge classifier automatically chooses the solver based on the data – these solvers include cholskey, cholskey kernel, sparce cg, and lsqr.

The second phase of the ridge classifier's implementation in sklearn is the fit phase. The model creates a coefficient vector that matches the data matrix and the vector of fourth down decisions. The last phase of the ridge classifier's implementation is the prediction phase. This is where the classes are generated based on the data fed to the coefficient matrix. The matrix corresponding to the data is multiplied by a matrix of feature weights, and the results of this multiplication are the weights for the classes. The maximum value among these weights decides what class is predicted for the data, and thus the fourth down decisions.

ii. Naïve Bayes [8]

The Naïve Bayes classifier is a variant on Bayesian networks that makes two assumptions: the attributes behave independently of each other and that unlisted variables do not affect predictions. The variant of Naïve Bayes that is being utilized in this project is Gaussian Naïve Bayes, where the values of the attributes are distributed via a Gaussian distribution. The Gaussian distribution is described in this manner:

$$p(x_i|c) = \frac{1}{\sqrt{2\pi\sigma_{c,i}^2}} exp^{\left(-\frac{(x_i-\mu_{c,i})^2}{2\sigma_{c,i}^2}\right)}$$

In this equation, the variable *i* represents the attribute in the dataset, the variable $\mu_{c,i}$ represents the mean and the variable $\sigma_{c,i}^2$ is the variance, and *c* is the class label.

iii. Linear Discriminative Analysis [9]

The linear discriminative analysis classifier involves reducing the matrix to a smaller matrix of lower dimensions. The low-dimensional feature space is selected based on the Fisher criterion, maximizing the distances between class means and simultaneously minimizing the intraclass variances. Maximizing the Fisher score is the same thing as maximizing the Rayleigh coefficient where the decision function weight vector is *w:*

$$w^* = arg\max_w \left\{\frac{w^T S_b w}{w^T S_w w}\right\}$$

The between-class variance is $S_b = (\frac{1}{n}) \sum_{k=1}^{K} n_k(\mu_k - \mu)(\mu_k - \mu)^T$ where *n* is the number of training samples, *k* is a class, *K* is the number of classes, *T* is the transposing symbol, n_k is the number of training samples for a class, μ_k is the mean vector for a class, and μ is the mean vector for the data. The within-class variance is $S_w = (\frac{1}{n}) \sum_{k=1}^{K} \sum_{i \epsilon I_k} (x_i - \mu_k)(x_i - \mu_k)^T$ where *i* is a feature, I_k is the identity matrix in the feature space for a class, and x_i is a vector of data containing that feature.

iv. Logistic Regression [10]

The logistic regression algorithm uses the features to help determine the class. The probability that a piece of data is in a class (represented by the variable γ) is calculated as follows:

$$\gamma = \frac{\exp\left(B_0 + B_1 x_1 + \cdots + B_p x_p\right)}{1 + \exp\left(B_0 + B_1 x_1 + \cdots + B_p x_p\right)}$$

In this algorithm, the vector B_p represents the set of coefficient variables, which are determined by the likelihood where, given the data has the corresponding feature, it is of the class. The vector x_p represents the individual features of the data.

The implementation of logistic regression that is used here is LogisticRegressionCV, which on top of the main part of the algorithm conducts another cross-validation algorithm since the algorithm on its own cannot produce an F1 score. This cross-validation algorithm involves splitting the dataset into subsets of equal sizes. At each iteration, a fold is chosen as the test set, while the others remain as training sets. After the iterations are complete, the performances are averaged. The number of folds and iterations are given by the user of the algorithm in the parameters of the implementation.

v. Bagging with Gradient Boosting [11][12][7]

The bagging algorithm involves, for each estimator, creating a replicate of the training set and constructing a classifier. The majority vote decision of all the classifiers is decided to be the class. The classifier that is being used for this project is gradient boosting, which was determined to be the most effective classifier because it had the highest F1 score. The gradient boosting algorithm was described in the original paper, but because it is an important classifier for the bagging algorithm here, it will be described in more detail.

The inputs for the gradient boosting algorithm include the input data, the number of iterations, the loss function, and the base-learner model. The sklearn implementation uses decision trees as the base-learner model, and the log loss function as the log function:

$$L_{log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p))$$

In this log loss function, *y* is the true class label and *p* is the predicted class label. In the gradient boosting algorithm, the first function estimate on the first step of the gradient is initialized with a constant. For each iteration, the following actions are performed. The negative gradient $g_t(x)$ is calculated:

$$g_t(x) = E_y \left[\frac{L_{log}(y, p)}{p} | x \right]_{p=\hat{f}^{(t-1)}(x)}$$

In this algorithm, *x* represents the current piece of data, E_y represents the estimator for the class, and $\hat{f}^{(t-1)}(x)$ represents the prediction from the previous step on the gradient. Afterwards within this iteration, the best gradient descent step size (p_t) is calculated:

$$p_t = arg \min_p \sum_{i=1}^{N} L_{log}\left[y_i, \hat{f}_{t-1}(x_i) + \rho h(x_i, \theta_t)\right]$$

In this algorithm, *p* is the potential steps for the gradient, *N* is the number of potential steps, y_i is the true class label at this estimator, and $h(x_i, \theta_t)$ is the decision tree model, with the parameters being the data matrix at this estimator and the parameters for the step. Afterwards, the current step is set to be the previous step with the product of the step-size and the decision tree model (with parameters being the data matrix and parameters for step) added onto it.

IV. Implementation

For the project implementation, Python 3.10.7 was used, as well as several libraries within it, which will be described. To start, I used Pandas to create dataframes from the data in the CSV files. The 2018 play data and the additional pieces of 2017 data were merged in the way discussed above. This is where the first challenge came – on the original project, the null columns were removed after the kick columns were created, but since this was using a newer version of Python, the column creation did not work. This was solved by removing the null columns before creating the new kicking columns, allowing the newer interpreter to properly process the data. The sklearn train_test_split() function was used to split the data into a traditional 70% training and 30% testing split before using each run of a model with them. The models were all implemented through sklearn using the functions of the same name, except for logistic regression, which used the LogisticRegressionCV function since the normal logistic regression function was unable to calculate F1 scores.

Originally, it was planned that Bayesian networks would be among the different models that would be used in this project. However, upon implementing it with Pomegranate, it would crash the Python environment, and upon re-implementing it with pgmpy, it was discovered that there was not enough data for the Bayesian network to properly create nodes for the classes. Therefore, Bayesian networks were scrapped from the project in favor of using the Ridge classifier.

As for more original plans that ended up being scrapped, one of the graphs that was made for the program during the initial graph production in late March was the quarter decision. That decision was immediately shown to be unimportant in the first test runs of feature importance, so that ended up being replaced by the offense and defense timeouts.

Another challenge found during implementation was the timeline – the plan was to spend until the end of March getting up to hyperparameter tuning and doing the graphs of the different important features based on what was assumed to be important. After the midterm presentation had passed, there was a week to calculate feature importances and then a couple more weeks to work on the presentation and final report. Calculating the feature importances, debugging the program, and getting the necessary tables together for the presentation ended up taking more time than originally anticipated, leading to some of the most important features being graphed, as well as a feature that was shown to be important in some debugging runs but not shown in the final tables being graphed by the program. In retrospect, more time should have been placed on doing the feature importances on the timeline, which could be done by swapping out the time meant for creating graphs and the time meant for calculating feature importances on the timeline. To compensate for the graph that could not be made, in the explanation for the final experimental results, the graph for the feature that was not as important in the final run as was in the test runs was still used to help infer the impacts of other features.

In this project, the user can choose between using hyperparameter tuning and not using hyperparameter tuning. For hyperparameter tuning, the method that was used involved sklearn's GridSearchCV. GridSearchCV determines what parameters are used in hyperparameter tuning by deciding them based on a grid [13]. This determines the best parameters that would be used for the models.

Models	Parameters
Ridge Classifier	{}
Naive Bayes	{}
Linear Discriminant Analysis	{}
Logistic Regression	{'cv': 3, 'scoring': 'f1', 'class_weight': 'balanced', 'n_jobs': -1}
Bagging	{'base_estimator': GradientBoostingClassifier(min_samples_leaf=3, n_estimators=300)}
Gradient Boosting	{}

Table 1

Table 1 shows the parameters that were used without hyperparameter tuning. Shown above, the default parameters were used for all the models except for logistic regression and bagging. Logistic regression used 3 scores for cross-validation, f1 scoring, used a balanced class weight, and used all the cores on the CPU. Bagging used the top model of the previous project (gradient boosting classifier with a limit of 3 nodes on tree, 3 samples on the leaf, and 300 boosting stages) as its estimator.

Models	Parameters
Ridge Classifier	{'alpha': 0.1}
Naive Bayes	{}
Linear Discriminant Analysis	{'solver': 'svd'}
Logistic Regression	{'Cs': 100, 'class_weight': 'balanced', 'cv': 3, 'n_jobs': -1, 'penalty': 'l2', 'scoring': 'f1', 'solver': 'lbfgs'}
Bagging	{'base_estimator': GradientBoostingClassifier(min_samples_leaf=3, n_estimators=300), 'n_estimators': 10}
Gradient Boosting	{'max_depth': 3, 'min_samples_leaf': 3, 'n_estimators': 300}

Table 2

Table 2 shows the parameters that were used after hyperparameter tuning. The regularization strength for the ridge classifier was 0.1. The parameters for naïve bayes could not be hyperparameter tuned because the only parameter for GaussianNB was the data. The solver for linear discriminant analysis was singular value decomposition. Logistic regression used 100 inverses of regularization strength, a balanced class weight, 3 folds for cross-validation, all the CPUs, a penalty of L2, a scoring of F1, and a solver of LBFGS. Bagging used the top model of the previous project (gradient boosting classifier with a limit of 3 nodes per tree, 3 samples on the leaf, and 300 boosting stages) as its

estimator, using 10 bags. Gradient boosting had the same best parameters as the last project, with 3 nodes per tree, 3 samples on the leaf, and 300 boosting stages.

Upon testing, it was found out that some of the values found in the results were close and difficult to gauge, so it was decided to use the Tabulate library to create tables with the different pieces of data to show the numbers. In addition, to help determine what parameters the models used for feature importance would have, a dictionary of the parameters for the models was returned alongside the average F1 scores over 10 runs. With the graphing involved with these projects, the graphing was done using Seaborn and the results were showcased using Pyplot.

There were several assumptions used in the implementation of this project. To start, the datasets have features removed from them – the assumption is that none of these features have too large of an impact on what decisions are made on the fourth down. This assumption is made because having a smaller feature set is important for training the data and processing it, and having too many features would make things too complex. Another assumption is that, since each model is trained on its own training/testing set for each run, the differences between each set are small enough that they make the model results a worthy comparison. This is because of the way the program is structured, that there are 10 runs for each training and testing set, and therefore the random 70% 30% training-testing split means that the models will have different hyperparameters, F1 scores, and feature importances on each run.

Another assumption is that the importance of features is not skewed by the values to the point that it makes the predictions untrustworthy. This is because some of the features shown in the result have much larger results than others. Another assumption is that when the bagging classifier uses the tuned gradient boosting as its classifier, the contrast between it and other non-tuned classifiers in its test does not overly skew the results. The reason behind this assumption is that with the bagging classifier, the tuned value is the number of estimators and not the classifier that is in the bags, where the goal of this project is to find a classifier with a higher F1 score than the one in the project of the previous semester, which has tuning used in it. Also, this assumption was noticed too late into the research's development to refactor the code and re-test it without the bagging estimator's gradient boosting classifier having its tuning affected by the parameters of the main gradient boosting algorithm. Also, there is a very rare chance that has occurred in 1 in every 20 or so tests debugging where the gradient boosting classifier outperforms the bagging classifier – however, since this is so rare, it is likely because there is a different train-test set for each model and each run of the model, making the two models perform differently for variants of sets.

V. Experimental Results

Models	F1 Scores
Ridge Classifier	0.588737
Naive Bayes	0.574103
Linear Discriminant Analysis	0.711195
Logistic Regression	0.251023
Bagging	0.903611
Gradient Boosting	0.894635

Table 3

Models	F1 Scores
Ridge Classifier	0.567927
Naive Bayes	0.570016
Linear Discriminant Analysis	0.713431
Logistic Regression	0.259665
Bagging	0.894251
Gradient Boosting	0.888304

Table 4

Tables 3 and 4 show the F1 scores of the different models. Table 3 is without hyperparameter tuning while Table 4 has hyperparameter tuning. On both tables, bagging has the highest F1 score involved. It is noted that the F1 scores on the hyperparameter tuning side have lower scores than the no hyperparameter tuning side – this does not show that going without hyperparameter tuning has superior results than being with it. Rather, it is a result of the different models being tested involving different runs, and different data is gotten from the different train-test splits for use.

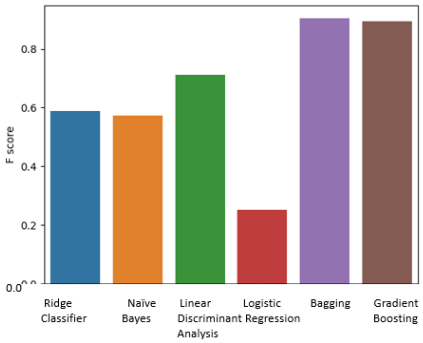

Figure 1 (left)

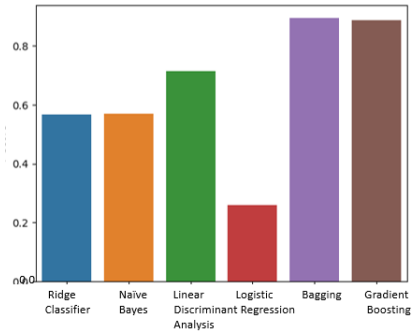

Figure 2 (right)

For each model, the average F1 score over 10 runs was taken, with tests being done without hyperparameter tuning (shown on Figure 1) and with hyperparameter tuning (shown on Figure 2). Amongst the five models that were tested with, the best ones involved the use of gradient boosting – bagging with gradient boosting, as well as the Fall 2022 gradient boosting model. Three of the other models, ridge classifier, naïve bayes, and linear discriminant analysis, did well, but not as well as the first two classifiers mentioned. The last model, logistic regression, had significantly worse performance than the others. There was barely a difference in performance between the tuned models and the untuned models – it makes the two hours it takes to run tuning versus the ten minutes it takes to run without tuning seem not worth the cost.

Bagging with gradient boosting was the model that performed the best, with an F1 score of 0.904 without hyperparameter tuning and 0.894 with hyperparameter tuning. Gradient boosting, the model from Fall 2022, performed very well, with an F1 score of 0.894 without hyperparameter tuning and 0.888 with hyperparameter tuning. Linear discriminant analysis, naïve bayes, and ridge classifier all performed well. Linear discriminant analysis had an F1 score of 0.711 without hyperparameter tuning and an F1 score of 0.713 with hyperparameter tuning, naïve bayes had an F1 score of 0.574 without hyperparameter tuning and 0.570 with hyperparameter tuning, and ridge classifier had an F1 score of 0.589 without hyperparameter tuning and 0.568 with hyperparameter tuning. Logistic regression had the worst result, barely able to make an F1 score of 0.3 regardless of whether it was hyperparameter tuned, making it useless in predicting fourth down decisions.

Features	Feature Importance
epa	0.432442
td_prob	0.136565
ydsnet	0.12495
ydstogo	0.0880497
wp	0.0527934
no_score_prob	0.0515755
fg_prob	0.026864
yardline_100	0.0267782
drive	0.00721029
posteam_timeouts_remaining	0.00461364

Table 4

Features	Feature Importance
epa	0.428497
td_prob	0.123215
ydsnet	0.0984935
wp	0.0812082
ydstogo	0.0799731
no_score_prob	0.0584937
yardline_100	0.0301952
fg_prob	0.0282232
posteam_timeouts_remaining	0.0184842
quarter_seconds_remaining	0.0115277

Table 5

Table 4 and 5 show the importance of the different features in fourth down decisions. Since bagging with gradient boosting was the classifier with the most accurate results, it was used to calculate the importance of which features would be most useful for predicting decisions made on fourth down in football games. To find the feature importances, the feature importances attribute for gradient boosting was averaged among all the estimators for bagging with numpy arrays.

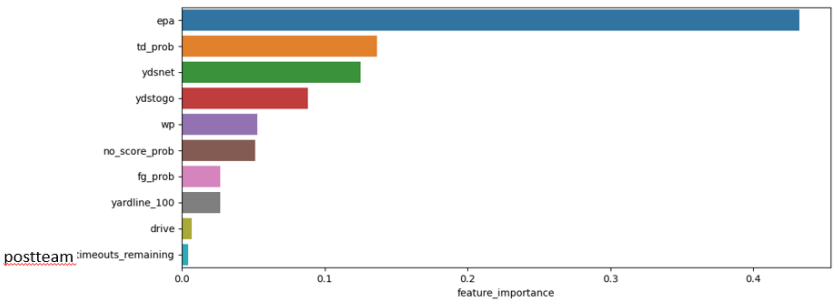

Figure 3

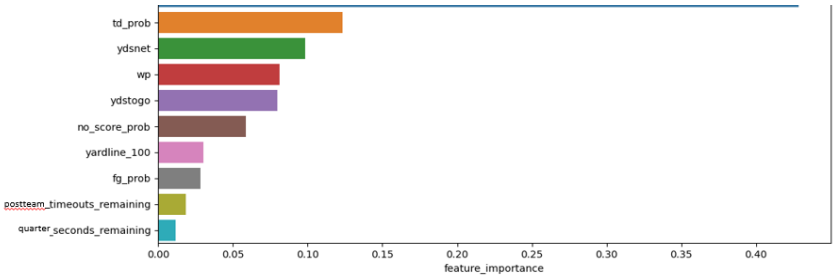

Figure 4

Figure 3 shows the feature importances for bagging with gradient boosting without hyperparameter tuning, and Figure 4 shows the feature importances with hyperparameter tuning. The figures show the expected points added for an incoming play as the most important feature. The expected points added can range between negative and positive values depending on the game. For instance, if a game is going poorly and the offense is losing yardage on the field, the expected points added might equate to a negative value, while if the team is doing well in a game, it might equate to a positive value. Other features that are important include the yards to go, the number of offense team timeouts remaining, and the win probability.

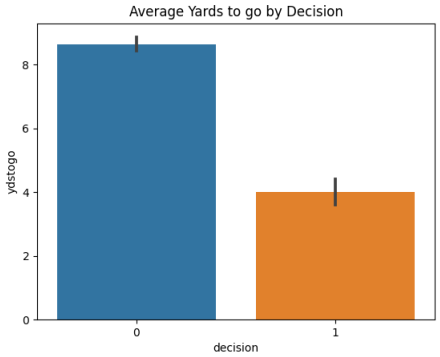

Figure 5

Figure 5 shows how important the number of yards from the first down line is to the game. This shows that teams are more likely to make a run or pass when they are closer to the first down line, and if they are farther away from it, and therefore closer to the goal, the team is more likely to go for a field goal or punt the ball. In this graph, 0 represents field goals and punts and 1 represents runs and passes.

Figure 6 shows how important the number of remaining offense timeouts for a game is for the fourth down decisions. If the offensive team has timeouts remaining, then they are more likely to go for the field goal compared to if they do not have any timeouts left. This is because if they have timeouts left, they are more likely to be willing to take risks to score on the round. Running or passing is a less risky move because there is less of a chance for players to risk making illegal moves and getting timed out on a play. Conversely, if the defensive team has fewer than the maximum number of timeouts left, the offensive team is more likely to run or pass since the other team is playing rougher and could risk disrupting play.

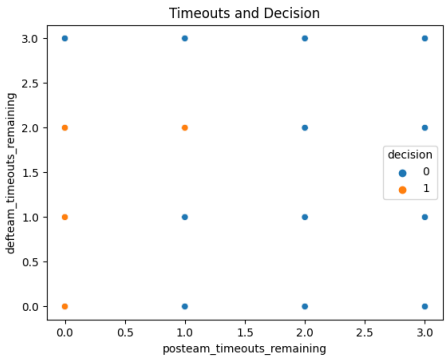

Figure 6

Figure 7 shows how win probability is important to the decision that a team makes on fourth down. If a team has a high win probability, they are more likely to go for the field

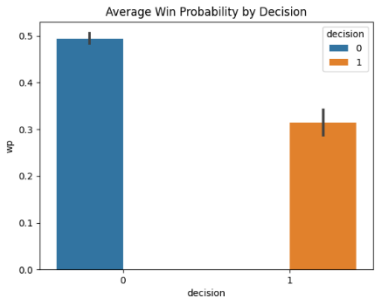

Figure 7

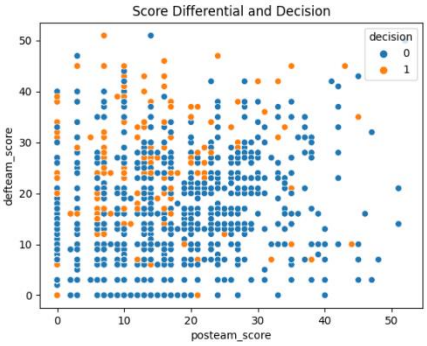

Figure 8

goal or punt compared to if a team has a low probability. This is because while a field goal leads to a higher score, there is more risk of the other team getting the ball. Therefore, running or passing on fourth down would be a safer way to get points, especially if the opponent team has a higher win probability and is more likely to take the ball if the play fails.

Another feature that was shown to be important to the model is the probability of there being no scoring for the rest of the half. Getting the feature importances for the presentation was one of the last things done on the project, so there was not enough time to create a graph of such a probability and properly test it out before the deadline. However, a team could get a rough estimate of how likely they are to score based on how they are doing so far in the game, which is gauged by the offensive team score and defensive team score as shown in Figure 8, which was made earlier in the project's development on the assumption that it would be an important feature in the final test run, since there were many practice runs of the program during debugging, with defensive team score showing up as an important feature in some of them. Attempts to run or pass are concentrated on the top left side of the graph, meaning that the team on defense is winning and the team on offense is more likely to have a lower scoring probability for the rest of the game. This means that the team on offense is more likely to make safer decisions to score so they can close the gap between themselves and the defense. In addition, there is a cluster of dots of different types around the y=x line, meaning that it is a close game and both teams have a high probability of getting scores.

vi. Conclusion and Future Research Directions

The aim of this project is to find a model that can be used to predict fourth down decisions that would be more useful than the top one in the previous semester's project. Data was used from the previous semester's datasets and merged in a similar manner. The classification models that were used included ridge classifier, naïve bayes, linear discriminative analysis, logistic regression, and bagging with gradient boosting. Using hyperparameter tuning did not have as much of an effect on the results as was initially expected, not being worth the higher amount of time it took to run compared to going without hyperparameter tuning. After averaging the F1 score for each model over ten runs, the model that proved to be more successful than gradient boosting was bagging with gradient boosting, with an average F1 score of 0.904 without hyperparameter tuning and 0.894 with hyperparameter tuning.

Finding models that accurately predict decisions is important for teams to make important decisions on fourth down, which can be crucial for how the game turns out. As the model has determined, some of the most important features in the game that help to aid fourth down decisions include the expected points added, the number of yards left on a first down, the number of remaining offense timeouts, and the probability of there being no scoring for the rest of the game.

There are multiple research directions that could be gone through in the future. Including data based on previous games, compared to just data in 2018, would help to see more trends in data and create less of a skew for specific team rosters. Models could also be used to help predict other decisions because if a team is doing well in previous points in the game, they are more likely to make successful decisions on fourth down. If access to such data was available, data could be found from how the current season is going so far, to make the most out of the current roster and help with moves on the spot. In the future, even more models could possibly be looked through to determine better ways to predict fourth down decisions.

References

[1] “2018 NFL Punting Stats.” NFL.com, www.nfl.com/stats/player-stats/category/punts/2018/reg/all/puntingaveragestats/desc.

[2] “Football Field Goal Range.” Rookieroad.com, www.rookieroad.com/football/strategy/field-goal-range.

[3] Nunes, Sérgio, and Marco Sousa. “Applying Data Mining Techniques to Football Data from European Championships.” Semantic Scholar, 2006, www.semanticscholar.org/paper/Applying DataMining-Techniques-to-Football-Data-Nunes-Sousa/fa0e6fc6d2143d0faca96a1f542772bef296667f.

[4] Che Mohamad Firdaus, Che Mohd Roslil, et al. “A Comparative Study of Data Mining Techniques on Football Match Prediction.” IOPscience, IOP Publishing, 1 May 2018, iopscience.iop.org/article/10.1088/1742-6596/1020/1/012003.

[5] Teach, Brendan, et al. “NFL Play Prediction.” College of Information and Computer Sciences, University of Massachusetts Amherst, 4 Jan. 2016, arxiv.org/pdf/1601.00574.pdf.

[6] Anagh Singh, Shiva Prakash B., et al. “A Comparison of Linear Discriminant Analysis and Ridge Classifier on Twitter Data.” *2016 International Conference on Computing, Communication, and Automation (ICCCA),* IEEE, April 2016, doi: 10.1109/CCAA.

[7] Pedregosa, Fabian, et al. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research,* vol. 12, 2011, pp. 2825-2830.

[8] Jahromi, Ali H. and Mohammad Taheri. “A non-parametric mixture of Gaussian naïve bayes classifiers based on local independent features.” 2017 Artificial Intelligence and Signal Processing Conference (AISP), IEEE, October 2017, doi: 10.1109/AISP.2017.8324083.

[9] Bandos, Tatyana V. and Lorenzo Bruzzone. “Classification of Hyperspectral Images with Regularized Linear Discriminant Analysis.” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47 no. 3, IEEE, February 2009, doi: 10.1109/TGRS.2008.2005729.

[10] Raphael Couronné, Phillipp Probst, et al. “Random forest versus logistic regression: a large-scale benchmark experiment.” *BMC Bioinformatics,* vol. 19 no. 270, 2018, doi: 10.1186/s12859-018-2264-5.

[11] Estaban Alfaro, Matias Gámez, et al. “Adabag: An R Package for Classification with Boosting and Bagging.” *Journal of Statistical Software,* August 2013, doi: 10.18637/jss.v054.i02.

[12] Natakin, Alexey and Alois Knoll. “Gradient boosting machines, a tutorial.” *Frontiers in Neurorobotics,* vol. 7, December 2013, doi: 10.3389/fnbot.2013.00021.

[13] Ranjan GSK, Amar Verma, et al. “K-Nearest Neighbors and GridSearchCV Based Real Time Fault Monitoring System for Industries.” 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), IEEE, March 2019, doi: 10.1109/I2CT45611.2019.9033691.

[14] Maral Hagighat, Hamid Rastegari, et al. “A Review of Data Mining Techniques for Result Prediction in Sports.” Pennsylvania State University, citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=37767ab1f559dda7a8d239aec0af1c9485d3d426.

[15] John McCullagh. “Data Mining in Sport: A Neural Network Approach.” Pennsylvania State University, citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=db63e35ce3e4ec99a1c2c73060c34ddc75256ebf.

[16] Dragan Miljković, Ljubiša Gajić, et al. “The Use of Data Mining for Basketball Matches Outcomes Prediction.” 2010 IEEE 8th International Symposium on Intelligent Systems and Informatics, IEEE, September 2010, doi: 10.1109/SISY.2010.5647440.

[17] Kewei Yu, Ting Yuan, et al. “Application of Data Mining Technologies in Sports Data Analysis in Colleges and Universities.” 2021 International Conference on Information Technology and Contemporary Sports (ICS), IEEE, January 2021, doi: 10.1109/TCS52929.2021.00073.

[18] Tejinder Singh, Vishal Singla, et al. “Score and Winning Prediction in Cricket through Data Mining.” 2015 International Conference on Soft Computing Technologies and Implementations (ICSCTI), IEEE, October 2015, doi: 10.1109/ICSCTI.2015.7489605.