Fourth Down Decision Predictor

Anthony Clavette & Lexington Johnson

Abstract:

The goal of this project is to correctly predict when a team will go for it on fourth down. When the defense is prepared and stops a team on a fourth down attempt, it gives the ball to their offense which helps them win. We want to help defenses have a continuous awareness of the opposing team's probability of going for it on fourth down or kicking the ball away. Despite being just a game, professional sports results can have major impacts on people's jobs and communities, so the risky and poor decisions made by defensive coaches on fourth down must be nearly perfect. We found that the score differential, the yards to go for a first down, the time left in the game, and the team's position on the field most heavily contribute to their fourth down decision of kicking or going for it.

Introduction:

As technology becomes exponentially more advanced, we have seen a significant rise in innovation with offensive schemes than just ten years ago. Since the defense is always one step behind the offense, more time and algorithmic decision-making is required for defensive coordinators to prepare their side. Our model can provide defensive coaches with information that will help them be more aware of what contributes to the opposing team's fourth down decision. Professional football is a billion-dollar entertainment industry that floods United States economics, media, and pop culture. Game outcomes have been proven to be capable of causing major uproar in cities and on the internet, so these split-second decisions that coaches have to make can have an enormous impact on the NFL community. Typically, other works using this dataset or analyzing similar data focus on enhancing the offensive side of decision making throughout the game. On the other hand, our work focuses on preparing the defense and giving them more time for smaller details in fourth down scenarios. We can do this by determining what features of the game are the most important and revealing hidden situational patterns that are invisible to the naked eye.

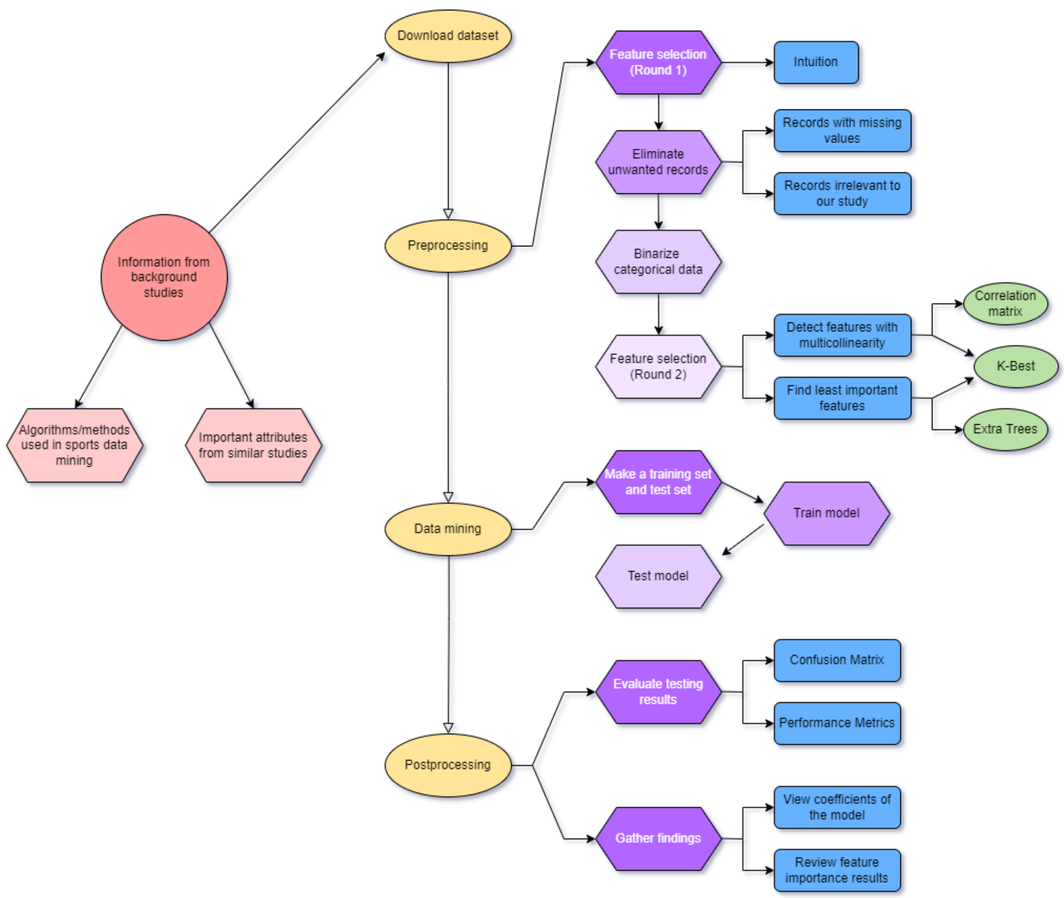
Background:

The sports data industry has stretched to expensive heights and made incredible advancements, which means many studies have solved problems similar to ours. Studies helpful to our progress include statistical studies on American football strategy, fourth down strategies, and the most important attributes to focus our model around. These studies also helped us adopt a plan and healthy habits for the project, such as how to properly prepare the dataset for usage in preprocessing algorithms and classifiers. These studies also gave us examples of how to analyze the results of a predictive model and what are the most meaningful performance metrics to a problem like ours.

To summarize a few background studies on fourth down decisions, their models all shared the following features: yards to first down, yard line on the field, and time remaining. Therefore, we were able to assume that these would have the largest weight in our model, and we should pay the most attention to them. Some of them focused on some more obscure elements to the problem, such as analyzing how fourth down decisions and success changes as the season goes on. Aside from predictive models, another useful study mapped recommended fourth down decisions according to success probability and the attributes mentioned above. This allowed us to incorporate a feature describing the team's win probability because this study outlined its correlation with fourth down success. Overall, reading studies of similar datasets or similar problems being solved gave us an advantage with understanding the importance of our dataset's features and gave useful insight on effective methodology and work progress.

Methodology:

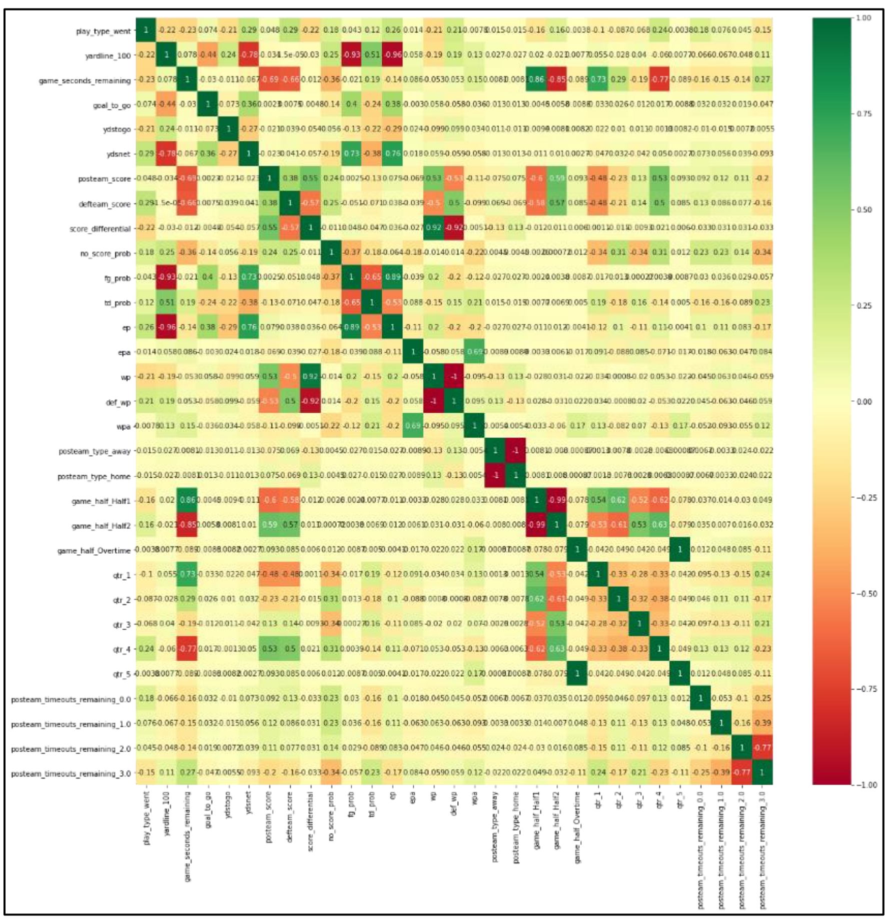
The framework for our research and processes is shown in the flowchart below. After reading background studies and downloading the dataset, our work had three distinct steps: preprocessing, data mining, and post processing.



During preprocessing, we removed over 200 attributes by intuition and then needed to use algorithms to be able to keep reducing dimensionality. Supported by Pandas and Scikit-learn, we used a correlation matrix, a K-Best classifier, and an Extra Trees classifier to reveal the least important features and any multicollinearity that may exist. To begin the data mining step, we used built-in methods from Scikit-learn to build our training and test sets, build a logistic regression model, train the model, and test the model on the test data. In order to meaningfully interpret our trained logistic regression model, we wanted to view the results of our testing and information we found about our attributes. To evaluate the results of our model, we used algorithms from Scikit-learn to view the confusion matrix, performance metrics, and ROC/AUC. Lastly, we looked at the model's weight coefficient for each attribute and we reviewed the results of the feature importance algorithms from preprocessing.

Data:

The dataset was originally a 700 MB collection of every NFL play that happened from 2009-2018. There were 255 attributes describing the nature of all 450,000+ records in the dataset, which would be a lot of data for us to efficiently process. Fortunately, many attributes and records were completely irrelevant to our study and could be removed, we just had to find them. We started with feature selection, removing the attributes that are obviously unnecessary and do not contribute to the independent variable we were targeting. We were able to remove about 200 attributes this way. To clean the data more before running it through any preprocessing algorithms, we removed all records with missing values and also irrelevant records (unfinished plays, penalties, or plays that happened on first, second, and third down). To go deeper into feature selection, we produced the following correlation matrix to find any interesting correlations or multicollinearity among attributes:



The correlation matrix helped us eliminate some attributes that we noticed had almost identical correlations with other attributes. For example, although it might seem obvious beforehand, it made us realize we do not need multiple metrics for game time (quarter, half, seconds remaining, etc). They all have essentially the same meaning and same correlation with other attributes, so we were able to use only the ones with the highest importance.

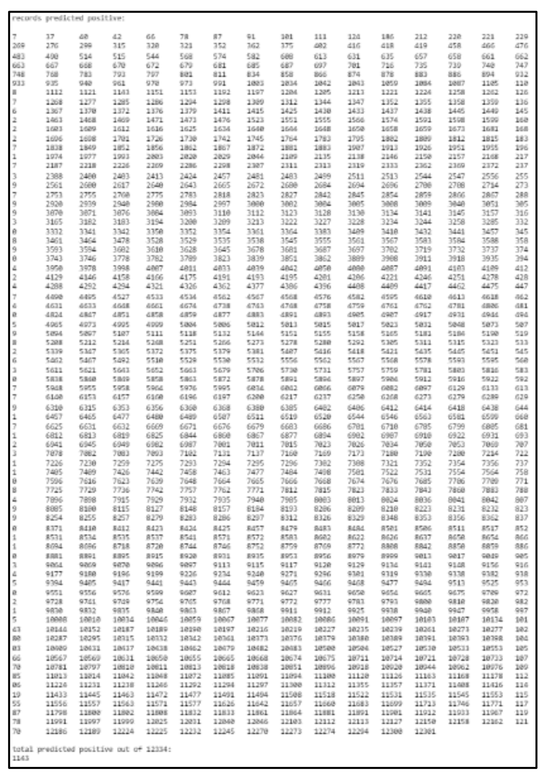
Next, we used Scikit-learn's K-Best classifier to finally isolate the 10 most important attributes, but we ended up just using it on every remaining attribute because the classifier summary printed the score of each attribute's importance:

	Specs	Score
6	defteam_score	3436.191539
4	ydsnet	3350.287771
11	ep	2672.937305
24	qtr_4	2334.754878
1	game_seconds_remaining	2112.573880
0	yardline_100	1978.869548
7	score_differential	1899.891932
14	def_wp	1798.618299
13	wp	1798.618291
3	ydstogo	1740.271925
26	posteam_timeouts_remaining_0.0	1238.926234
8	no_score_prob	1221.888147
19	game_half_Half2	1008.864013
18	game_half_Half1	1001.325608
29	posteam_timeouts_remaining_3.0	860.851087
10	td_prob	522.958799
21	qtr_1	384.236406
22	qtr_2	286.970809
27	posteam_timeouts_remaining_1.0	217.955604
2	goal_to_go	208.584533
23	qtr_3	171.839073
5	posteam_score	85.185034
28	posteam_timeouts_remaining_2.0	76.919492
9	fg_prob	69.708212
16	posteam_type_away	8.124723
17	posteam_type_home	8.124723
12	epa	7.288137
15	wpa	2.297683
20	game_half_Overtime	0.544613
25	qtr_5	0.544613

The scores proved more multicollinearity in our dataset, which can be seen by attributes with identical scores as others: 'def_wp' and 'wp', 'posteam_type_away' and 'posteam_type_home', 'game_half_overtime', and 'qtr_5' are three pairs of attributes that have identical scores. This visual also gave us insight on how unimportant some attributes are by looking at their importance score compared to the top 20 ranked attributes. For example, 'qtr_5' and 'game_half_overtime' are binary indicators of the game being in overtime and had a score of less than 1, while the top 20 attributes had scores ranging from 200 to 3500. After completing this final round of eliminating features, the dataset was ready to be split into training and test sets.

Experiments:

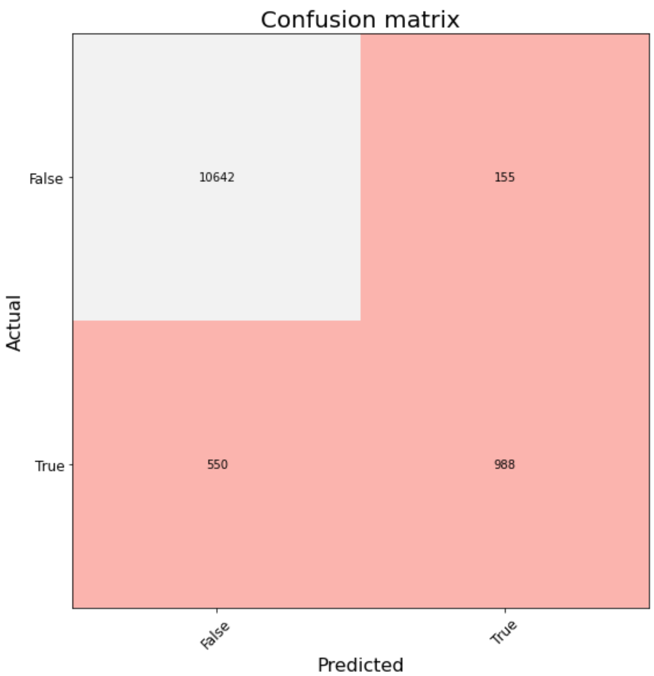
First, we used Spyder IDE to efficiently carry out Python scripts for the first round of preprocessing, which removed 200+ columns from the dataset and exported it to replace the original file. Using this new and significantly smaller dataset, the experimental procedure was then carried out on Jupyter Notebook, a Python-supporting IDE that interactively runs notebook files on a local host. Throughout the semester, updates to code were stored on GitHub to organize relevant steps throughout the project and to keep records for potential mistakes. After the necessary preprocessing, data sampling was not necessary because the remaining 37,000 rows and 25 columns were not dangerously large for our code. Using Scikit-learn's train_test_split() method, 67% of this dataset was randomly selected for the training set, and the remaining 33% was the test set; the fit() method was then used to train the model, and the predict() method was used to test it:



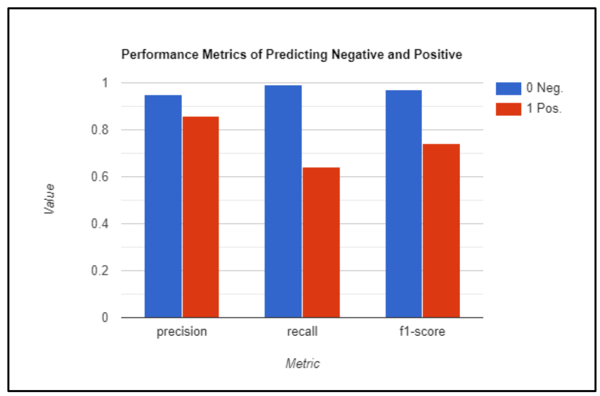

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.33, random_state=0)
classifier = LogisticRegression(solver='lbfgs', max_iter=4500, random_state=0)
classifier.fit(X_train, Y_train)
predicted_y = classifier.predict(X_test)

print("records predicted positive: \n")
count=0
for x in range(len(predicted_y)):
    if (predicted_y[x] == 1):
        count = count + 1
        print(x, end="\t")
print("\n\ntotal predicted positive out of 12334: ")
print(count)
```

Results:

The model predicted that the team would attempt a play on fourth down (positive outcome) for 1143 instances of the test set's 12,344 total records, with 94% accuracy. The confusion matrix and performance metrics are shown below:

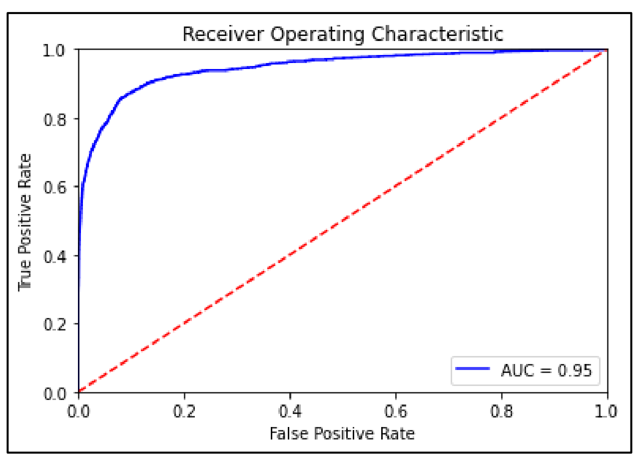



```
from sklearn.metrics import classification_report
print(classification_report(Y_test, predictions))

              precision    recall  f1-score   support

           0       0.95      0.99      0.97     10797
           1       0.86      0.64      0.74      1538

    accuracy                           0.94     12335
   macro avg       0.91      0.81      0.85     12335
weighted avg       0.94      0.94      0.94     12335
```



These results show that the model clearly proved its competence, but they also show weak spots in it as well. Looking at the precision, recall, and f1-score for the two outcome predictions, we can see how much better the model was for predicting negative outcomes than positive outcomes. This means the high accuracy of the model (0.94) can be linked to the dataset having a vast majority of true negatives and the model's ability to correctly predict them.

The graph on the right is the ROC curve, which shows the TPR plotted against the FPR. This curve visualizes the model's discriminative ability, and the area under the curve is used as a measure for the trait. The area under our curve was 0.95, which shows a consistent proficiency in distinguishing between positive and negative outcomes.



As a result of these findings, we can conclude that although our model is better at correctly predicting negative outcomes than positive outcomes, it is still very good at distinguishing between both classes.

Conclusion:

We had some interesting results that came from data exploration and testing our model, but nothing was too shocking. We had a few ideas of what to expect in our findings because our background studies gave us an idea of important and unimportant features and we are also just familiar with American football.

Early on, an obstacle we faced was using Weka with our dataset and trying to familiarize ourselves with the software. We were intrigued by Weka’s organized GUI, built-in libraries and algorithms, model summary output, and especially the visualization methods. Overall, the software seemed optimal for taking this project to a higher level with every resource we could ever need. The obstacle was format issues of some attribute values in our dataset. We spent hours troubleshooting by trying different methods of discretizing, binarizing, and other forms of preprocessing our data because Weka refused to build a logistic regression model using our dataset. After about a week of research and trying different solutions, we started using Pandas and Scikit-learn with Python instead; we had our data cleaned and our first logistic regression model built within just one long day of working on it. On the other hand, there were also challenges we were not able to solve but have the potential to extend and improve this project further. The biggest challenge is transforming “irrelevant” records into attributes. Specifically, instead of just removing all plays that happened on first/second/third down, we would only remove the ones that did not eventually reach fourth down. For each fourth down being examined, our goal was to store the results of the preceding first/second/third downs as another feature. We believe that the individual outcomes of first/second/third down are relevant to the outcome of their corresponding fourth down, so we look forward to figuring out the solution to this challenge in the near future.