NFL Big Data Bowl 2021

Group 3 || Matthew Fritze, Krupal Patel, Mital Patel, Junyi Peng, Mohit Ramnani, Yanyan Zhang

Introduction

The NFL is a multi-billion dollar organization that entertains millions of people around the world. Data analysis offers an opportunity for teams to discover patterns in their strategies, players, and seasons. In this report we investigate how they relate to successful outcomes, and we discuss why it might be hard to find strategies with clear advantages. We hypothesize that any strong offensive strategies would trigger defensive adaptations - resulting in no silver-bullet solutions to football strategy.

Objectives

The main objectives of this research study are:

1. Data cleaning: explore and review the data in Spark using queries learned from the class. Prepare the data for building machine learning models using Spark's MLLib library.

2. Machine learning: explain the correlation, feature importance and any possible causation between different play input features and the play result features by modeling use different algorithms and different input features.

Data Preparation

The data source for this project is provided through a current Kaggle competition called NFL Big Data Bowl 2021 sponsored by the NFL. The dataset contains games, players, plays and tracking data for all passing plays during the 2018 regular season. A passing play is one in which a pass was thrown, quarterback was sacked or a penalty was called. The data consists of over 50 key variables across 4 different tables and a detailed description of the variables can be found in the appendix of this report. The dataset was 2.17GB across 20 csv files which were manually uploaded to Databricks platform and then converted and saved to a table for faster reads.

The data was of high quality with few null values and since data was provided by the NFL itself it is deemed to be accurate with few errors. Procuring the data was relatively straightforward as it involved downloading a zip file from Kaggle with all the necessary files. Databricks notebooks were used to read the csv files once they were uploaded to the platform.

Data ingestion into Databricks was simple and the attached notebook was used to prepare the data for analysis. (https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/7115721934622905/2539233089939245/3573834079610223/latest.html. The tracking data was provided as 17 separate csv files, one for each week of the regular NFL season. This was merged into one single table.

Although the data is of a high quality there is some missing data. Lineman (both offensive and defensive) data are not provided by the NFL. Also, the dataset is focused on passing plays therefore running plays data are not included.

Snapshots of the data tables are provided below:

Players data

	nflId	height	weight	birthDate	collegeName	position	displayName
1	2539334	72	190	1990-09-10	Washington	CB	Desmond Trufant
2	2539653	70	186	1988-11-01	Southeastern Louisiana	CB	Robert Alford
3	2543850	69	186	1991-12-18	Purdue	SS	Ricardo Allen
4	2555162	73	227	1994-11-04	Louisiana State	MLB	Deion Jones
5	2555255	75	232	1993-07-01	Minnesota	OLB	De'Vondre Campbell
6	2555543	73	216	1995-07-26	Florida	FS	Keanu Neal
7	2556445	70	211	1992-10-20	Florida	CB	Brian Poole

Showing the first 1000 rows.

Games data

	gameId	gameDate	gameTimeEastern	homeTeamAbbr	visitorTeamAbbr	week
1	2018090600	09/06/2018	20:20:00	PHI	ATL	1
2	2018090901	09/09/2018	13:00:00	CLE	PIT	1
3	2018090902	09/09/2018	13:00:00	IND	CIN	1
4	2018090903	09/09/2018	13:00:00	MIA	TEN	1
5	2018090900	09/09/2018	13:00:00	BAL	BUF	1
6	2018090905	09/09/2018	13:00:00	NE	HOU	1
7	2018090907	09/09/2018	13:00:00	NYG	JAX	1

Showing all 253 rows.

Plays data

	gameId	playId	playDescription	quarter	down	yardsToGo	possessionTeam	playType	yardlineSide
1	2018090600	75	(15:00) M.Ryan pass short right to J.Jones pushed ob at ATL 30 for 10 yards (M.Jenkins).	1	1	15	ATL	play_type_pass	ATL
2	2018090600	146	(13:10) M.Ryan pass incomplete short right to C.Ridley (J.Mills, J.Hicks).	1	1	10	ATL	play_type_pass	PHI
3	2018090600	168	(13:05) (Shotgun) M.Ryan pass incomplete short left to D.Freeman.	1	2	10	ATL	play_type_pass	PHI
4	2018090600	190	(13:01) (Shotgun) M.Ryan pass deep left to J.Jones to PHI 6 for 33 yards (R.Darby).	1	3	10	ATL	play_type_pass	PHI
5	2018090600	256	(10:59) (Shotgun) M.Ryan pass incomplete short right to D.Freeman.	1	3	1	ATL	play_type_pass	PHI
6	2018090600	320	(10:10) (Shotgun) N.Foles pass short left to N.Agholor to PHI 8 for 4 yards (R.Alford).	1	2	8	PHI	play_type_pass	PHI
7	2018090600	344	(9:24) (Shotgun) N.Foles pass incomplete short left to D.Sproles (R.Allen).	1	3	4	PHI	play_type_pass	PHI

Showing the first 1000 rows.

Analysis

MLLib - Random Forest Classifier

Using the plays data table a multi-class Random Forest Classifier was built with Spark MLLib in an attempt to predict the pass result of a given play given other features in the dataset. A combination of numerical and categorical features were selected to predict the labels in the ‘passResult’ column.

<u>Data and Features:</u>

The plays data consists of 19,239 instances of pass plays during the 2018 regular season. Ten features were selected for the model, which are listed as follows: 'quarter', 'down', 'yardsToGo', 'absoluteYardlineNumber', 'offenseFormation', 'personnelO', 'defendersInTheBox', 'numberOfPassRushers', 'personnelD', 'typeDropback'.

Rows with missing values were dropped from the analysis since there was no reasonable method to fill the values and only a small proportion of the instances had null values. Our final dataset contained 18,506 instances, which were separated into a 80/20 train/test split.

The label, 'passResult', contained 4 unique classes: C (completion pass), I (incomplete pass), S (quarterback sack) and IN (intercepted pass). The dataset was unbalanced with most of the passes being completed. The value counts of the full dataset is shown in the table below.

```
passResult|count|
+----------+-----+
|        IN|  405|
|         S| 1259|
|         I| 5591|
|         C|11251|
```

Pipeline:

The Pipeline function was used to prepare and transform the final list of features for model training. The pipeline consisted of four stages. StringIndexer to convert a string column of labels to a column of label indices. One Hot encoder to map a column of categorical features into binary column vectors. Vector Assembler to merge all the feature columns into a single feature vector that can be used for model training. The last stage was the Random Forest Classifier to fit the features to the machine learning algorithm.

Model selection/ hyper parameter tuning:

The Random Forest Classifier algorithm was chosen to fit and predict the pass result labels. The hyper parameters of the Random Forest Classifier were tuned using the CrossValidator function which was passed a param grid of various hyper-parameters

such as maxDepth, numTrees and bootstrap. The number of folds for cross validation were set to 3, the default value. The best model hyperparameters when tested on the train set were maxDepth=5, numTrees=5, bootstrap=True.

Results:

The best Random Forest Classifier model was tested on the test set with the following results.

Summary Stats

Accuracy	0.601
Precision	0.424
Recall	0.012
F1 Score	0.023

Interpreting the results, it seems the model was unsuccessful in accurately classifying the labels based on the features provided. The classifier was unable to find any meaningful patterns in the data that can be used to make useful predictions. Model kept outputting the label for the class with the highest value count, which was C (pass completion). Although the model has a 61 percent accuracy score, the F1 Score is only 2.3 percent. Possible causes of the low F1 score are bad/incomplete data and teams adjusting to their opponents strategies over time causing optimal plays to be neutralized.

The analysis for Random Forest Classifier can be found at the published Databricks notebook in the appendix.

MLLib - Linear Regression

Data & Features & Pipeline:

Using the same features as the random forest classifier, we attempted to use linear regression to predict the number of yards gained on the play which is encoded as a float in the dataset. The pipeline used LinearRegression as the only stage, using gridsearch ElasticNetParam as the estimator params map, and using RegressionEvaluator as the evaluation measure.

Summary Stats

MSE	112.23
MAE	7.46
RMSE Squared	10.59
R Squared	0.00014
Explained Variance	0.4097

As in the case of the random forest model, the linear regression model was a poor predictor for net yards gained. Further analysis could be done by breaking down the yards gained by each offensive formation, as yards-gained is more likely to be linearly related to individual play-types.

MLLib - Decision Tree Regression

Data and Features:

The following modeling is trying to explore and explain whether there's any relation between the playtype variables and the outcome (playResult), and the feature importance of the playtype variables for determining and predicting the outcomes. A decision tree regressor is chosen to be the algorithm used for the modeling.

The playtype column contains three different variables which are play_type_sack, play_type_pass and play_type_unknown. Those three different variables stand for three different strategies or outcomes of dropping back.

Cmd 5

```
1 display(nfl_plays.groupBy("playType").count().selectExpr("playType"))
```

▸ (2) Spark Jobs

	playType
1	play_type_sack
2	play_type_pass
3	play_type_unknown

Showing all 3 rows.

A sack occurs when the quarterback is tackled behind the line of scrimmage before he can throw a forward pass, when the quarterback is tackled behind the line of scrimmage in the "pocket" and his intent is unclear, or when a passer runs out of bounds behind the line of scrimmage due to defensive pressure.

Dropping back to pass is a passing style employed in American football. The quarterback initially takes a three-step drop, backpedaling into the pocket to make a pass.

In other situations where it's hard to differentiate the outcomes or the game has been interrupted due to different reasons, it will be categorized into play_type_unknown.

Hypothesis:

Based on personal experience, the dropping back strategies are important because they play important roles in deciding the tempo and the progress of the game especially

during the earlier stages. Theoretically, the development during the early games are usually the defining factor of the trend of the play style through the whole game.

Both offenceFormation and playType columns are indexed for the following process.

Cmd 6

```
nfl_plays_1 = StringIndexer(inputCol="offenseFormation", outputCol="offenseFormationIndex").fit(df_new).transform(df_new)
nfl_plays = StringIndexer(inputCol="playType", outputCol="playTypeIndex").fit(nfl_plays_1).transform(nfl_plays_1)
display(nfl_plays)
```

▸ (5) Spark Jobs
▸ nfl_plays_1: pyspark.sql.dataframe.DataFrame = [gameId: string, playId: string ... 26 more fields]
▾ nfl_plays: pyspark.sql.dataframe.DataFrame

During the vector assembling process, the other input chosen is the offenceFormation which stands for the formation used by the possession team. It is chosen due to its crucial role in deciding the play style during and right before the scoring time point.

```
from pyspark.ml.feature import VectorAssembler
vectorAssembler = VectorAssembler(inputCols = ["offenseFormationIndex","playTypeIndex"], outputCol = 'features')
nfl_plays_final = vectorAssembler.transform(nfl_plays)
nfl_plays_final1 = nfl_plays_final.select(['features', 'playResult'])
nfl_plays_final1.show(3)
```

▸ (1) Spark Jobs
▾ nfl_plays_final: pyspark.sql.dataframe.DataFrame

Results:

As a result, this model has a rmse of 10.0165 which is not perfect but acceptable considering the random nature of sports. If one model or certain algorithms perform extremely well on certain sports using certain inputs, the results of those sports are certainly predictable which could represent that some cheating or unethical factors are involved.

Cmd 10

```
from pyspark.ml.regression import DecisionTreeRegressor
dt = DecisionTreeRegressor(featuresCol ='features', labelCol = 'playResult')
dt_model = dt.fit(training)
dt_predictions = dt_model.transform(test)
dt_evaluator = RegressionEvaluator(
    labelCol="playResult", predictionCol="prediction", metricName="rmse")
rmse = dt_evaluator.evaluate(dt_predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)

#This model has a rmse of 10.0165 which is not perfect but accetable considering the random nature of the sport.
```

```
▸ (8) Spark Jobs
▸ dt_predictions:  pyspark.sql.dataframe.DataFrame = [features: udt, playResult: integer ... 1 more fields]
Root Mean Squared Error (RMSE) on test data = 10.0165
Command took 4.52 seconds -- by junyi.peng@ryerson.ca at 12/1/2020, 6:12:05 PM on Quickstart
```

The feature importance of playType, however, is over 94% which should be considered surprisingly high. This acknowledges the hypothesis that the play type strategies indeed play important roles in explaining and predicting the play results which in this case, the net yards gained by the offense including penalty yardage in football games.

Cmd 11

```
dt_model.featureImportances
#The feature importnace of playType, however, is over 94% which should be considered surpirsingly high. This acknowledges the hypothesis that
the play type strategies indeed play import roles in determining the play results in football games.
```

```
Out[105]: SparseVector(2, {0: 0.0546, 1: 0.9454})
Command took 0.10 seconds -- by junyi.peng@ryerson.ca at 12/1/2020, 6:13:04 PM on Quickstart
```

MLLib - Logistic Regression

The following analysis is trying to predict how players speed, acceleration, weight, height and age affect the performance on their offense plays. Our hypothesis is that players' speed, acceleration, weight and height will lead to a higher likelihood of successful offense plays, whereas age will have a negative impact on the result of the offense plays.

Data and Features:

This part of the analysis uses the players data and the tracking information. We performed some data cleaning on the data:

1. We created a new column '*offenseResult*' and assigned value 1 (offense successful) when 'event' value is '*pass_arrived*', '*handoff*', '*pass_forward*' or '*pass_outcome_caught*' and value 0 (offense failed) for the rest of the 'event' values.

2. Data in 'players' used 2 different ways to present players height, in'' and ft'-in''. We transformed players heights in ft'-in'' format to in''.

3. To use player age as an independent variable, we unified the format of column 'birthDate' from 'players' table to 'MM-dd-yyyy' and calculated players' age based on the unified '*birthDate*' column.

4. We have removed values from the data frame where the 'event' and 'nflId' values are null.

Model:

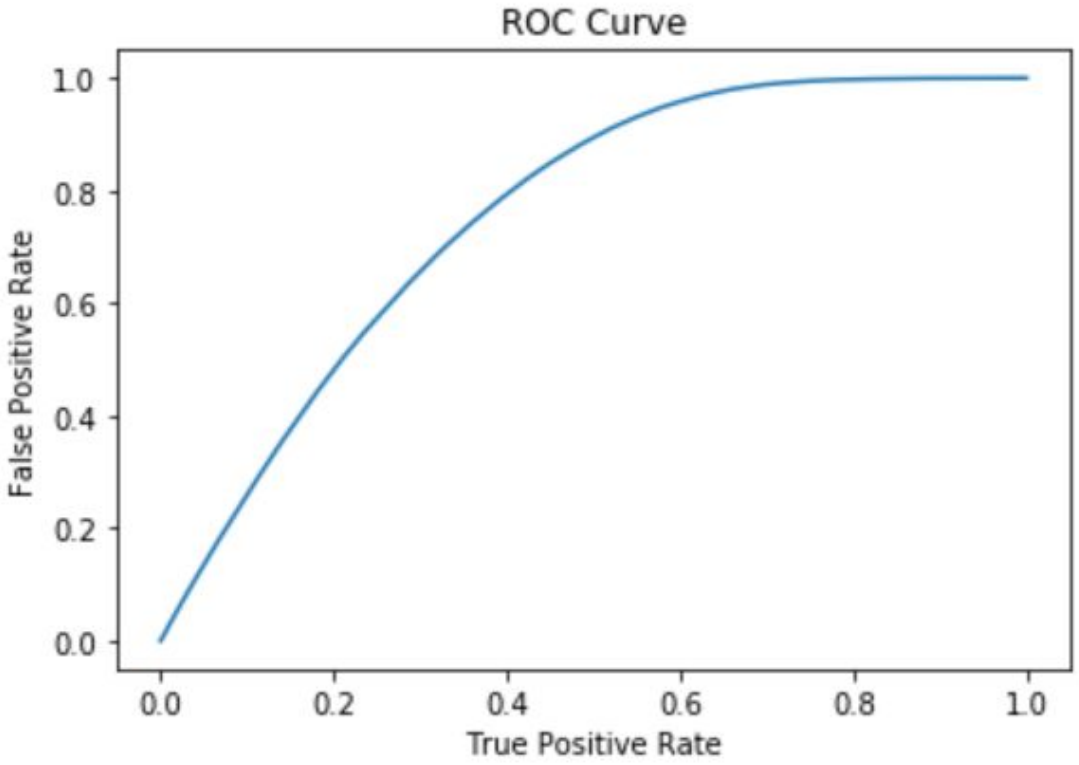
We used a logistic regression model that used 'weight', 'height', 'age', 's' and 'a' as independent variables and 'offenseResult' as the target variable.

We used a pipeline model which consists of a Vector Assembler, Min Max Scaler and Logistic Regressor for our model. The model also used a 75/25 split for the training and test datasets.

The result of the logistic regression is as follows:

Logit(p) = - 2.42 + 5.73 * s + 14.98 * a + 0.34 * height + 0.47 * weight + 0.52 * age

The AUC score of the above model is around 0.76.



Suggesting a Defense Lineup

Using the plays data to predict a defensive lineup for a given number of players at each offensive position. Only ‘passR result’ values IN(Incomplete) and S(Quarterback Sack) were considered for the data set as those two results are positive outcomes defensively.

Initially, all available defensive formations were used however, there were way too many of them so they were narrowed down to the top 10 most frequently used formations. And since a number of players in offensive formation were mostly repeating, they did not seem enough for feature columns, hence ‘absoluteYardlineNumber’ was also included, And finally, the 10 defensive formations were given a unique numeric value from 1 to 10(column named personnelD_num).

Feature Columns: 'num_offense_runningbacks', 'num_offense_tightends', 'num_offense_widereceivers', 'absoluteYardlineNumber'

Target Columns: ‘personnelD_num’

This data was then separated into 70/30 train/test splits.
The Random Forest Classifier was run and the best model hyperparameters on train were numTrees = 20 and numFeatures = 4

Result:

```
MSE:  5.34886445597414
MAE:  2.0209893036519397
RMSE Squared:  2.312761218970549
R Squared:  0.07961493433130429
Explained Variance:  0.4672452677139267
```

The model with the best hyperparameters was used for the test but the results were not satisfactory with a prediction accuracy of just greater than 1%. It was concluded that a set defensive formation cannot be predicted based on just those features since every team uses a different tactic and what they choose to go with also might depend on the situation of the game.

	(sum(equal) / sum(1))
1	0.010723860589812333

Conclusions

Finding consistent advantages turned out to be a hard problem. This is inline with our original hypothesis that any successful strategies would be correspondingly counter-strategized.

In conclusion, all models performed comparably mediocre and acceptable as we hypothesized due to the nature of the American football. If one model or certain algorithms perform extremely well on certain sports using certain inputs, the study of different strategies and formation would have no purpose. We did, however, find some explainable statistical relations between certain features with high feature importance with certain result features which could be used for further analysis and study for NFL professionals.

Appendix

Published Notebooks

Data cleaning and feature creation:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/4876776748177010/3052620058281825/7840256590229964/latest.html

Random Forest Classifier predicting passResult:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/7115721934622905/1623734348589260/3573834079610223/latest.html

Linear Regression predicting net yards gained
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/4876776748177010/3052620058281844/7840256590229964/latest.html

Decision tree regressor:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/2831789717620731/2413801639644454/4938613764438363/latest.html

Random Forest Classifier predicting personnelD based on personnelO and AbsoluteYardlineNumber :
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/6826459341672661/4402010555934432/6118366557028857/latest.html

Logistic Regression:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/8255965852721784/3621600656749198/3242671813138915/latest.html

Data variable description

Game data

- `gameId`: Game identifier, unique (numeric)
- `gameDate`: Game Date (time, mm/dd/yyyy)
- `gameTimeEastern`: Start time of game (time, HH:MM:SS, EST)
- `homeTeamAbbr`: Home team three-letter code (text)
- `visitorTeamAbbr`: Visiting team three-letter code (text)
- `week`: Week of game (numeric)

Player data

- `nflId`: Player identification number, unique across players (numeric)
- `height`: Player height (text)
- `weight`: Player weight (numeric)
- `birthDate`: Date of birth (YYYY-MM-DD)
- `collegeName`: Player college (text)
- `position`: Player position (text)
- `displayName`: Player name (text)

Play data

- `gameId`: Game identifier, unique (numeric)
- `playId`: Play identifier, not unique across games (numeric)
- `playDescription`: Description of play (text)
- `quarter`: Game quarter (numeric)
- `down`: Down (numeric)
- `yardsToGo`: Distance needed for a first down (numeric)
- `possessionTeam`: Team on offense (text)

- `playType`: Outcome of dropback: sack or pass (text)
- `yardlineSide`: 3-letter team code corresponding to line-of-scrimmage (text)
- `yardlineNumber`: Yard line at line-of-scrimmage (numeric)
- `offenseFormation`: Formation used by possession team (text)
- `personnelO`: Personnel used by offensive team (text)
- `defendersInTheBox`: Number of defenders in close proximity to line-of-scrimmage (numeric)
- `numberOfPassRushers`: Number of pass rushers (numeric)
- `personnelD`: Personnel used by defensive team (text)
- `typeDropback`: Dropback categorization of quarterback (text)
- `preSnapHomeScore`: Home score prior to the play (numeric)
- `preSnapVisitorScore`: Visiting team score prior to the play (numeric)
- `gameClock`: Time on clock of play (MM:SS)
- `absoluteYardlineNumber`: Distance from end zone for possession team (numeric)
- `penaltyCodes`: NFL categorization of the penalties that ocurred on the play. For purposes of this contest, the most important penalties are Defensive Pass Interference (DPI), Offensive Pass Interference (OPI), Illegal Contact (ICT), and Defensive Holding (DH). Multiple penalties on a play are separated by a `;` (text)
- `penaltyJerseyNumber`: Jersey number and team code of the player commiting each penalty. Multiple penalties on a play are separated by a `;` (text)
- `passResult`: Outcome of the passing play (`C`: Complete pass, `I`: Incomplete pass, `S`: Quarterback sack, `IN`: Intercepted pass, text)
- `offensePlayResult`: Yards gained by the offense, excluding penalty yardage (numeric)
- `playResult`: Net yards gained by the offense, including penalty yardage (numeric)
- `epa`: Expected points added on the play, relative to the offensive team. Expected points is a metric that estimates the average of every next scoring outcome given the play's down, distance, yardline, and time remaining (numeric)
- `isDefensivePI`: An indicator variable for whether or not a `DPI` penalty ocurred on a given play (TRUE/FALSE)

Tracking data

Each of the 17 `week[week].csv` files contain player tracking data from all passing plays during Week `[week]` of the 2018 regular season. Nearly all plays from each `[gameId]` are included; certain plays or games with insufficient data are dropped. Each team and player plays no more than 1 game in a given week.

- `time`: Time stamp of play (time, yyyy-mm-dd, hh:mm:ss)
- `x`: Player position along the long axis of the field, 0 - 120 yards. See Figure 1 below. (numeric)
- `y`: Player position along the short axis of the field, 0 - 53.3 yards. See Figure 1 below. (numeric)
- `s`: Speed in yards/second (numeric)
- `a`: Acceleration in yards/second^2 (numeric)
- `dis`: Distance traveled from prior time point, in yards (numeric)
- `o`: Player orientation (deg), 0 - 360 degrees (numeric)
- `dir`: Angle of player motion (deg), 0 - 360 degrees (numeric)
- `event`: Tagged play details, including moment of ball snap, pass release, pass catch, tackle, etc (text)
- `nflId`: Player identification number, unique across players (numeric)
- `displayName`: Player name (text)
- `jerseyNumber`: Jersey number of player (numeric)
- `position`: Player position group (text)
- `team`: Team (away or home) of corresponding player (text)
- `frameId`: Frame identifier for each play, starting at 1 (numeric)
- `gameId`: Game identifier, unique (numeric)
- `playId`: Play identifier, not unique across games (numeric)
- `playDirection`: Direction that the offense is moving (text, left or right)
- `route`: Route ran by offensive player (text)