Machine Learning Project - NFL Prediction

Jiaming Deng
dengji@tcd.ie
22302794

Tingzhou Wan
twan@tcd.ie
22329065

Yifan Zhu
yizhu@tcd.ie
22301710

***Index Terms*—Machine Learning, NFL**

I. Introduction

This project explores the prediction of winning/lossing of NFL teams given the information regarding the games in previous seasons and teams and compares the performance/accuracy of the model be selected with elo-predictions. This demonstrates how to predict the outcome in a sports league. A model is capable of performing such task could provide interesting insights for the people watching the games, and useful application for preseason training plan scheduling. The input to our application contains the elo information and team information from NFL, the outputs are the prediction of winning/lossing of teams in NFL.

II. Datasets and Features

A. Data Source and Preprocessing

$FiveThirtyEightDatabase$[1] was used as a source for sports games data in csv format. It provides all the detailed information of games played on National Football League (NFL) since 1969. While this raw data is not clean, the scale of data is large enough to use in this project and we decided to use the data between 2000 to 2022, which is nearly 5000 pieces.

The steps to clean up the original data are as follows.

1. Fill in all blank data with NAN.

2. Set data with decimal point as floating point number.

3. Set the full name of the team to the abbreviated name and uniform these shortname.

4. Convert boolean data to 0/1.

5. Convert the data with special characters into pure data. For example, from the 18th week, the English name is used as the code of the week instead of the number. Therefore, we need to change the English name to the corresponding number of the week.

6. Merge useful columns of different files into one file.

1 https://en.wikipedia.org/wiki/FiveThirtyEight

B. Feature Extraction

We have considered a lot of methods to complete feature extraction, explored many factors affecting the outcome of football matches, and found a lot of information to help us complete feature extraction.

There are nine features extracted.

1. home team is more popular
2. away team is more popular
3. whether it scores more than the average
4. victory rate of the home team this season
5. victory rate of the away team this season
6. victory rate of the home team last season
7. victory rate of the away team last season
8. whether the home team and the away team belong to the same division
9. elo probability

All these features are extracted from the raw data. Determine whether the home team or the away team is more popular according to the team name preferred by the audience; Determine whether the team score is higher than the average score according to the average score data and the score of the game on the spot; According to the statistics of the number of wins and losses of a team in a season, determine the victory rate of home and away teams; Judge whether the match areas of the two opposing teams are consistent according to their match areas; Other features are directly provided by the raw data.

C. Visualization

In order to select features and useful columns in the dataset, visualization is an important step to finish. By displaying the basic information of

the raw data, we can get the key information of the dataframes and make a better preparation for the preprocessing task of the raw data. Pandas dataframe and Seaborn package are the main tools we use for visualizing the data and our outputs. In the parts of modeling and prediction, the visualizaton would help us to have a better understanding of the results and which model would be the best to predict the results.

Below is the visualization for the correlation of all the variables and we can see the winning rates of both home team and away teams are the 4 variables have the most correlations.

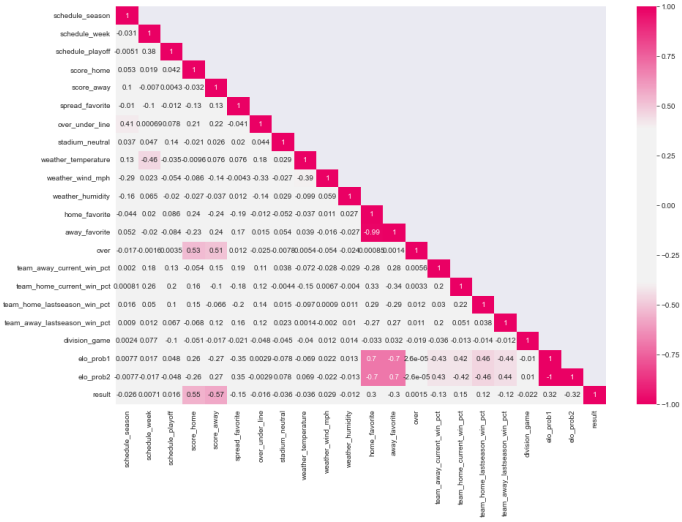


Fig. 1. Correlations

Below is the visualization for the winning rates of both home teams and guest teams in current seasons and last seasons. They both demonstrate that home teams tend to have higher winning rate than guest teams.

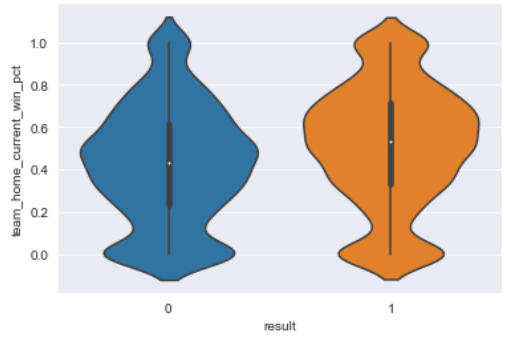


Fig. 2. Current Seasons

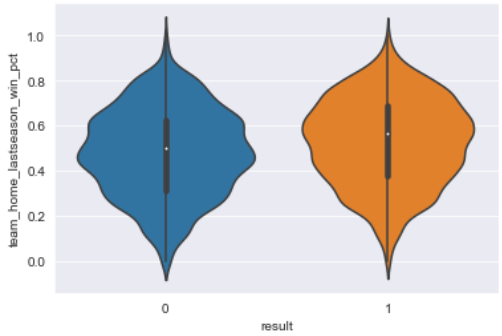


Fig. 3. Last Seasons

III. Methods

In this project, we totally prepared six prediction models. Through the k-fold cross-validation to compare their accuracy, and finally select the best-fit model as the NFL Prediction Machine Learning Model.

A. LogisticRegression

Logistic regression is a generalized linear model that introduces nonlinear factors through the Sigmoid function, so it is often used to solve binary classification problems (0 or 1). In the scenario, losing the game is 0, and winning the game is 1, so the logistic regression model perfectly fits the scenario.

B. k-Nearest Neighbours Regression (kNN)

K-Nearest Neighbours Regression is a non-parametric method that approximates the association between an independent variable and a continuous outcome in an intuitive manner by averaging observations within the same neighborhood. The size of the neighborhood can be chosen using cross-validation so that the size that minimizes the mean squared error is chosen.

C. Gaussian Naive Bayes

Naive Bayes is a probabilistic machine learning algorithm for many classification functions that is based on Bayes' theorem. Gaussian Naive Bayes is an extension of Naive Bayes. Its typical applications are document classification, spam filtering, prediction, etc. The algorithm includes features in its model that are independent of each other. Any modification of one eigenvalue does not directly affect the values of any other eigenvalues of the

algorithm. It is based on probabilistic models, algorithms can be easily coded, and predictions can be made quickly in real time.

D. Random Forest

Random Forest is a common machine learning algorithm that combines the outputs of multiple decision trees to arrive at a single result. The Random Forest algorithm has three main hyper-parameters that need to be set before training. These include node size, number of trees, and number of features sampled. Random forest classifiers can be used to solve regression or classification problems.

The random forest algorithm consists of an ensemble of decision trees, each tree in the ensemble consists of a data sample drawn from the training set and a replacement data sample, called a bootstrap sample. Of this training sample, one-third is reserved as test data, known as an out-of-bag sample. Inject another instance of randomness through feature bagging, adding more diversity to the dataset and reducing correlation between decision trees.

E. Decision Tree

Decision trees are a non-parametric supervised learning algorithm for classification and regression tasks. It is a hierarchical tree structure consisting of a root node, branches, internal nodes and leaf nodes.

A decision tree starts with a root node, which does not have any incoming branches. The outgoing branches of the root node then feed the internal nodes (also known as decision nodes). Both types of nodes perform evaluation based on available capabilities to form homogeneous subsets, which are represented by leaf nodes or terminal nodes. Leaf nodes represent all possible outcomes within the data-set.

F. Linear Support Vector

Similar to SVC with kernel='linear', but built using liblinear rather than libsvm, giving it more flexibility in terms of the penalties and loss functions that can be used. As a result, it should scale to large numbers of samples more effectively.

Both dense and sparse input are supported by this class, and the multiclass support is handled using a one-vs-the-rest strategy.

G. Voting Classifier

Voting classifier is a machine learning model, which is trained on the set of many models and predicts the output based on the highest probability of the selected category as the output. It simply summarizes the results of each classifier passed to the voting classifier, and predicts the output category based on the highest majority of votes.

IV. Experiments

A. Find the best-fit model

In this part, We are going to compare the accuracy of 7 models through k-fold cross-validation and (ROC,AOC) Score, and finally find the best-fit model to predict the outcome of NFL games

- **Collecting the data:** The data is from the CSV file which has been cleaned in last part. A total of ten features are used and associated with the outcome of the game. The training data is from 2001 to 2016 in Schedule Season, the test data is from 2017 to 2018.
- **Preparing the model:** Firstly, creating a two-dimensional array, initializing these seven models including adjusting parameters, and put them into this two-dimensional array. Then, using 5-fold cross-validation, the accuracy scoring method is "roc-auc", and the accuracy corresponding model is also put into a two-dimensional array. Finally, sort this array and get the accuracy of each model, and output the best fit model. From the output, we can see the ROC, AUC scores of the LRG (logistic regression model) are the highest. Thus, we decided to apply logistic regression model as our prediction model to predict the outcome of the games.

Model	LRG	GNB	DTC	SVC
ROC-AUC Score	0.692	0.690	0.687	0.686

Model	VOTE	RFC	KNB
ROC-AUC Score	0.684	0.668	0.581

The best fit model is: LRG
The score is: 0.692349
Time used is 53856 ms

- **Training the model:** In this step, we start to train the model, and fit the data, and finally get

y-prediction and predict the probability that the input sample belongs to each category.

- **Drawing the confusion matrix:** We perform performance evaluation on classification models by confusion matrix and calculate the number of TP, FP, TN, FN

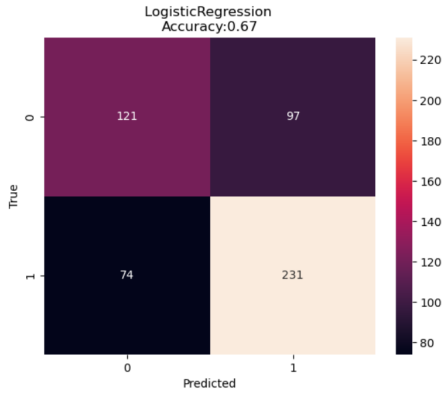


Fig. 4. Confusion Matrix of Logistic Regression

- **Finding the FPR and TPR for the threaholds of the classification and plot ROC curve:** ROC(Receiver Operating Characteristic) curve is used to evaluate the predictive ability of the model. Its main two indicators are FPR and TPR. When the TPR is higher and the FPR is lower at the same time (the ROC curve is steeper), the performance of the model will be better. In this plot, we can see the curve is steep and the value of AUC is 0.66, which is an efficient score. Thus, the application of the logistic regression model is efficient in the prediction of NFL games.

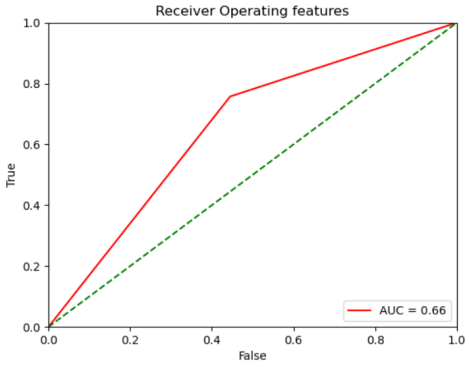


Fig. 5. ROC of Logistic Regression

- **Comparing the ROC-AUC Score and Brier Score of LRG model and Elo-prediction:** The Brier score is used to predict the maximum possible difference between the probability and the actual result, therefore, the smaller the value, the better. Also, the Brier score is only used for classification results that are structured as binary. According to the comparing with elo-prediction, we can see that the ROC-AUC Score of LRG model is higher than ELO-Model and the Brier Score is lower than ELO-Model. Thus, LRG model is more efficient and precise than ELO-Model

LRG Model	Eloprob1	Result
ROC-AUC Score	0.702	0.679
Brier Score	0.214	0.218

V. Results/Discussion

In this part, We are going to use real data and cases to compare with elo's predictions. In the same scenario, output the number of games, the probability of successful prediction, the number of successful predictions, and the number of predictions respectively. From this table, it shows that the probability of successful prediction in LRG model is 0.6965 which is higher than ELO-predictions' 0.6875.

According to above, we can conclude that by comparing the ROC-AUC scores and Brier scores of the two models, or comparing the probability of successful predictions through actual cases and data, the performance and accuracy of the LRG model are higher than those of the ELO model. Thus, Logistic Regression Model could be the best-fit model in NFL games' predictions.

Model	Win Percentage	Corrects	Predictions
LRG model	0.696	280	402
ELO model	0.687	242	352

VI. Summary

In this project, the task of predicting the outcomes of the games in NFL is tackled. We used multiple approaches for features extracting from the raw datasets. And 7 machine learning models were used for comparison of the performance in this task.

In our results of these models, Logistic Regression has the best performance in predicting the results of

the games. Among the results from other models, the performance of other models like Gaussian Naive Bayes and Voting Classifier tend to be slightly worse than that of Logistic Regression.

During the process of finishing the project, we found it hard to complete the feature engineering since there are various statistical factors in our raw datsaets and we needed to get the ones have great impact on the results of the games. Also, training the models using substantial features took much time. The accuracy of our algorithm is still not high enough for reference in the real world. In the futrue works, more hypertunning steps can be taken to boost the performance of our model to have a higher accuracy in predicting the NFL games in the future.

VII. Contribution

Jiaming Deng – Searched and obtained the raw data, process, clean up and merge the raw data. Added several new features into the data. Did some experiments with LogisticRegression model. Worked on code, report parts concerning these topics as well as contributed to the Distribution, Summary, Code sections.

Tingzhou Wan – Worked on code, experiment, report for KNN model. Used Pandas Dataframe and Seaborn pakcage to visualize the 3 raw datasets and the preprocessed dataset. Worked on explanatory data analysis of the preprocessed dataset and the visualization of the results. Contributed to introduction,summary, appendix sections of the report.

Yifan Zhu – Worked on the code, experiment, report for Gauussian Naive Bayes, Random Forest, Decision Tree, Linear Support Vector, Voting Classifer. Contributed to Discussion section of the report.

VIII. Code

This link → Github Link, contains the data, code for extracting the required input feature and preprocessing them. It also contains the code used for testing the different models like Random Forest, SVM, Logistic Regression and code for visualization.

IX. Appendix

Model Shortname	Model Fullname
LRG	Logistic Regression
GNB	Gaussian Naive Bayes
DTC	Decesion Tree
SVC	Support Vector Classification
VOTE	Voting Classifier
RFC	Random Forest Classifier
KNB	KNeighbors Classifier

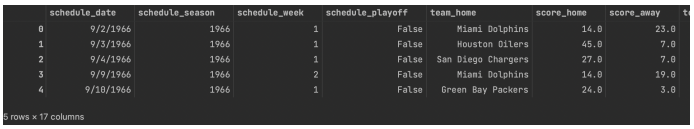
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17367 entries, 0 to 17366
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   date       17367 non-null  object
 1   season     17367 non-null  int64
 2   neutral    17367 non-null  int64
 3   playoff    603 non-null    object
 4   team1      17367 non-null  object
 5   team2      17367 non-null  object
 6   elo1_pre   17367 non-null  float64
 7   elo2_pre   17367 non-null  float64
 8   elo_prob1  17367 non-null  float64
 9   elo_prob2  17367 non-null  float64
 10  elo1_post  17275 non-null  float64
 11  elo2_post  17275 non-null  float64
 12  score1     17275 non-null  float64
 13  score2     17275 non-null  float64
dtypes: float64(8), int64(2), object(4)
memory usage: 1.9+ MB
```

Fig. 6. elo raw data information

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44 entries, 0 to 43
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   team_name               44 non-null     object
 1   team_name_short         44 non-null     object
 2   team_id                 44 non-null     object
 3   team_id_pfr             44 non-null     object
 4   team_conference         44 non-null     object
 5   team_division           35 non-null     object
 6   team_conference_pre2002 44 non-null     object
 7   team_division_pre2002   42 non-null     object
dtypes: object(8)
memory usage: 2.9+ KB
```

Fig. 7. teams raw data information

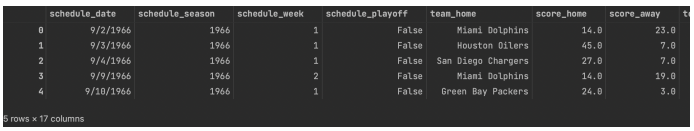
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   team_name        32 non-null     object
 1   team_name_short  32 non-null     object
 2   team_id          32 non-null     object
 3   team_id_pfr      32 non-null     object
 4   team_conference  32 non-null     object
 5   team_division    32 non-null     object
dtypes: object(6)
memory usage: 1.6+ KB
```



	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	t
0	9/2/1966	1966	1	False	Miami Dolphins	14.0	23.0	
1	9/3/1966	1966	1	False	Houston Oilers	45.0	7.0	
2	9/4/1966	1966	1	False	San Diego Chargers	27.0	7.0	
3	9/9/1966	1966	2	False	Miami Dolphins	14.0	19.0	
4	9/10/1966	1966	1	False	Green Bay Packers	24.0	3.0	

5 rows × 17 columns

Fig. 9. games raw data head



	schedule_date	schedule_season	schedule_week	schedule_playoff	team_home	score_home	score_away	t
0	9/2/1966	1966	1	False	Miami Dolphins	14.0	23.0	
1	9/3/1966	1966	1	False	Houston Oilers	45.0	7.0	
2	9/4/1966	1966	1	False	San Diego Chargers	27.0	7.0	
3	9/9/1966	1966	2	False	Miami Dolphins	14.0	19.0	
4	9/10/1966	1966	1	False	Green Bay Packers	24.0	3.0	

5 rows × 17 columns

Fig. 10. games raw data head