**NFL’s Toughest Games**

**Problem Introduction**

Year after year the NFL season comes and goes. The winners earn praise from fans and media, while the losers are picking up the pieces of their shattered season. Most of the time success is measured on where a time finished in the ranks, or how far in the playoffs was reached. The best and the worst teams can play awesome games but still end up on the losing side of the battle. What if we could score how difficult the team’s seasons were over the years based on their performance and opponent’s performance game in and game out? I decided to find that those scores with the help of machine learning.

**Strategy to Solve the Problem**

Predicting wins and losses strictly from each statistical performance in each game is the first step for the analysis. Comparing the actual game outcomes to the predicted outcomes will produce a scoring metric used to determine toughest games based on the description of the game.

**Metrics**

Actual: Team won or lost the game

Predicted: Team was predicted to win or lose the game

Opponent Actual: Opponent won or lost the game

Opponent Predicted: Opponent was predicted to win or lose the game

**Game Category**

Toughest Games: Games that opponent was predicted to win regardless of actual outcome

Easy Games: Games that opponent was predicted to lose regardless of actual outcome

**Game Type Descriptions**

|  |  |  |  |
| --- | --- | --- | --- |
| Game Type | Description | Difficulty Score | Code |
| Hard Fought Loss | Team has optimal stats to win game but still lost because opponent also had optimal stats to win game | +1 | hard\_fought\_loss = games\_matched.query("actual == 'loss' and predicted == 'win' and opponent\_actual == 'win' and opponent\_predicted == 'win'") |
| Tough Loss | Team has optimal stats to win game but still lost while the opponent did not have optimal stats. | -1 | tough\_loss = games\_matched.query("actual == 'loss' and predicted == 'win' and opponent\_actual == 'win' and opponent\_predicted == 'loss'") |
| Loss Bad Game | Both teams have sub optimal stats and opponent wins | -1 | loss\_bad\_game = games\_matched.query("actual == 'loss' and predicted == 'loss' and opponent\_actual == 'win' and opponent\_predicted == 'loss'") |
| Expected Loss | Team has sub optimal stats to win the game and opponent has optimal stat, so opponent expected to win | +1 | expected\_loss = games\_matched.query("actual == 'loss' and predicted == 'loss' and opponent\_actual == 'win' and opponent\_predicted == 'win'") |
| Hard Fought Win | Team has optimal stats to win game and win the game even when their opponent has optimal stats to win the game as well | +1 | hard\_fought\_win = games\_matched.query("actual == 'win' and predicted == 'win' and opponent\_actual == 'loss' and opponent\_predicted == 'win'") |
| Lucky Win | Team has below optimal stats to win the game and still wins when their opponent did have optimal stats to win | +1 | lucky\_win = games\_matched.query("actual == 'win' and predicted == 'loss' and opponent\_actual == 'loss' and opponent\_predicted == 'win'") |
| Win Bad Game | Team has below optimal stats to win and still wins while their opponent also had sub optimal stats | -1 | win\_bad\_game = games\_matched.query("actual == 'win' and predicted == 'loss' and opponent\_actual == 'loss' and opponent\_predicted == 'loss'") |
| Expected Win | Team has optimal stats to win the game and opponent has below optimal stats, so expected outcome | -1 | expected\_win = games\_matched.query("actual == 'win' and predicted == 'win' and opponent\_actual == 'loss' and opponent\_predicted == 'loss'") |

Overall Difficulty Score: A team’s overall score for how tough their opponents were over the past 20 years.

**EDA**

The raw data imported is in the form of rows being each game and columns containing the information and performance statistics of the game by each team. 5357 rows, 39 columns. The data was transformed by giving each team its own row to analyze their performance in the game by itself. The transformation increased the rows to 10690 and decreased columns to 22.

The numerical columns were either had a normal distribution on a right skew. Column redzone had all zeros for converted but had the number of attempts recorded in the first few years. The data did not record the redzone conversion rate in those years, but the attempts were helpful. The model metric outcome was correct with wins and losses being equal. Two columns rushing yards and passing yards were questionable with some negatives in the columns, but after further evaluation they are not outliers or mistakes. Scores column was dropped because ultimately score decides the winner and loser regardless of the stats. This analysis I based purely on statistical performance because a team’s score may not represent how well a team played.

The heat map below shows the correlation between all the numerical features. Most importantly the column to really focus in on is outcome win. The columns with the strongest correlation to winning were rushing attempts, possession, and rushing yards. The column with the strongest correlation to losing were turnovers, interceptions, and sacks.

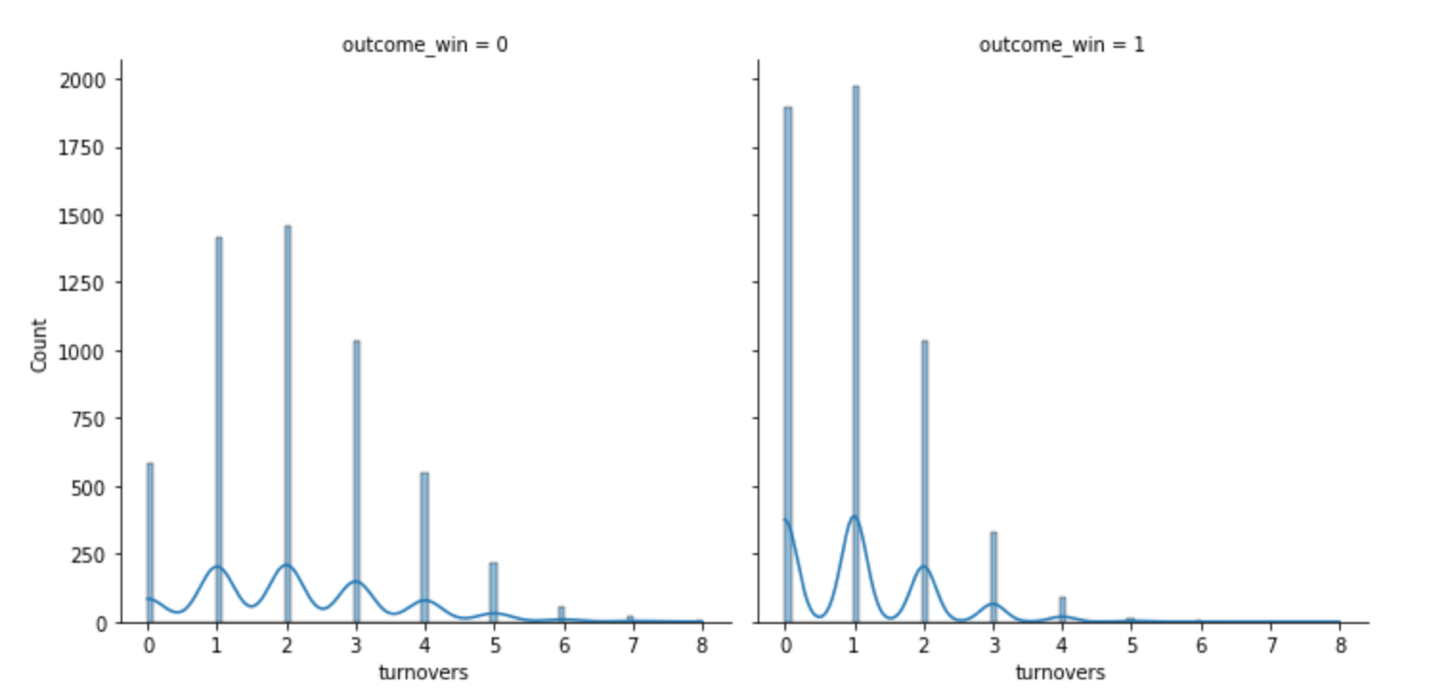
Chart, timeline, treemap chart

Description automatically generated

Comparing distribution between stats of winners and losers helps visualize the threshold between winning and losing. The charts below are the distribution comparison of rushing attempts and turnovers. Winning outcome rushing attempts distributions is more right then losing rushing attempts distribution meaning winning stats has higher number of rushing attempts. It’s easy to see that for turnovers, winners have a lot more 0-1 turnover games and barely go over 4 turnovers a game.

Chart, histogram

Description automatically generated



**Modeling**

The first step in modeling is data preprocessing by doing data transformation to get the dataset in workable shape. The next phase was converting conversion ratios to decimals for columns, third downs, fourth downs, comp att, and redzone. The decimals are better for the model to work with. I also split penalties and sacks into amounts as one column and yards as another. The penalties divided by penalty yards didn’t quite make sense because 1 penalty can be 25 yards and 3 penalties can also be 25 yards. The weight of 3 penalties verse 1 should be more costly despite the yards being the same. Converting outcome win column to binary for data exploration then back to classifier for the model was a necessary step for analysis understanding. The dataset was split into training and testing sets.

The type of model used is classification to predict if the team had sufficient statistical performance to win the game. The following algorithms were tested and ranked based on accuracy.

* Support Vector Machines
  + Kernel: Linear
  + Accuracy: 0.8244
* Random Forest
  + Optimal Depth: 13
  + Accuracy: 0.8205
* Decision Tree
  + Optimal Depth: 6
  + Accuracy: 0.7972
* K-Nearest Neighbors
  + Max Neighbor: 21
  + Accuracy: 0.7148

**Hyperparameter Tuning**

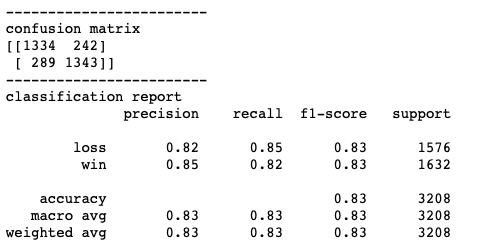
Testing consisted of cross validating each algorithm by its parameters to find the optimal setting to maximize accuracy. Then comparing the most accurate results from each algorithm to each other to find the best of the best.

* SVM parameter tuning was of the kernels, poly, rbf ,and linear
* Decision Tree and Random Forest both were tuned and tested on depths up to 25
* k-nearest neighbors tuning was on a range from 5 neighbors to 25 neighbors.

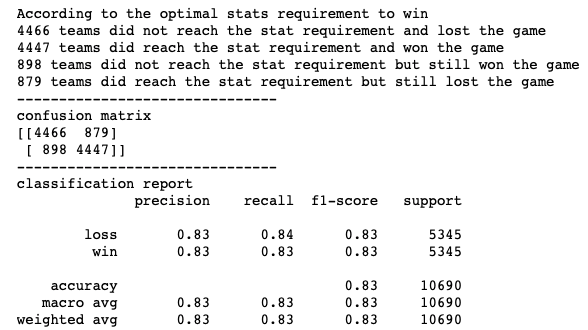
**Results**

Train Model for optimal statistical performance to win game.

The selected algorithm was Support Vector Machines with a linear kernel. The accuracy for kernel rbf was .76, kernel poly accuracy was .77 and kernel linear .82, making linear the optimal parameter. Below is the model result after fitting x train and y train and validating with the test set.



For final results, run the entire dataset through the model to compare the actual results to the predicted results based on the optimal statistical performance threshold to win the game.



The results are used to rank the teams with the most difficult games over the past 20 years. This is achieved by comparing the teams actual and predicted results to their opponents actual and predicted results. The comparisons determine the toughest games by game type descriptions.

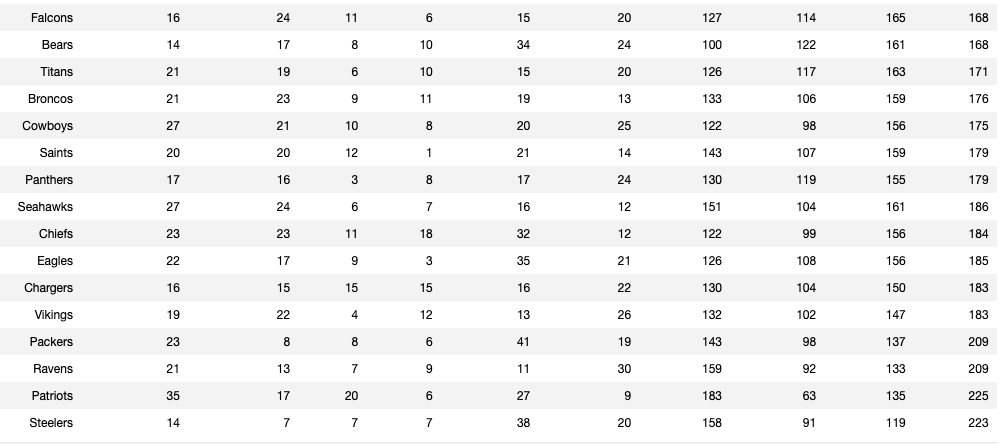
Game Type Results

|  |  |  |
| --- | --- | --- |
| Game Type | Description | Total Games |
| Hard Fought Loss | team has optimal stats to win game but still lost because opponent also had optimal stats to win game. | 608 |
| Hard Fought Win | team has optimal stats to win game and actually win the game even when their opponent has optimal stats to win the game as well. | 608 |
| Tough Loss | team has optimal stats to win game but still lost while the opponent did not have optimal stats | 271 |
| Lucky Win | team has below optimal stats to win the game and still wins when their opponent did have optimal stats to win. | 271 |
| Loss Bad Game | both teams have sub optimal stats and opponent wins | 627 |
| Win Bad Game | team has below optimal stats to win and still wins while their opponent also had sub optimal stats | 627 |
| Expected Loss | team has sub optimal stats to win the game and opponent has optimal stat, so opponent expected to win. | 3839 |
| Expected Win | team has optimal stats to win the game and opponent has below optimal stats, so expected outcome | 3839 |
| Grand Total | All Outcomes | 10690 |

Individual Team Results

Table

Description automatically generatedTable

Description automatically generatedTable

Description automatically generated

Top 10 Teams with Highest Difficulty Score

* Jaguars: 83 score
* Browns: 79 score
* Raiders: 68 score
* Lions: 48 score
* Jets: 42 score
* Dolphins: 37 score
* Rams: 33 score
* Texans: 31 score
* Washington: 27 score
* Colts: 25 score

The results determined that the Jaguars played in the most difficult games based on the scoring metric. The above table provides a breakdown of each team’s game type amount that led to the difficulty score. Below is the scoring metric breakdown of the top 10 highest difficulty scores

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Team | Hard Fought Win | Hard Fought Loss | Lucky Win | Tough Loss | Win Bad Game | Lose Bad Game | Expected Win | Expected Loss | Difficulty Score |
| Jaguars | +12 | +31 | +9 | -8 | -8 | -14 | -92 | +153 | 83 |
| Browns | +13 | +21 | +7 | -15 | -14 | -18 | -75 | +160 | 79 |
| Raiders | +11 | +16 | +14 | -9 | -15 | -22 | -83 | +156 | 68 |
| Lions | +16 | +18 | +7 | -8 | -20 | -39 | -69 | +145 | 48 |
| Jets | +18 | +24 | +6 | -9 | -9 | -19 | -108 | +139 | 42 |
| Dolphins | +17 | +10 | +11 | -5 | -29 | -20 | -99 | +142 | 37 |
| Rams | +15 | +14 | +8 | -9 | -18 | -14 | -109 | +146 | 33 |
| Texans | +15 | +39 | +3 | -8 | -12 | -17 | -113 | +124 | 31 |
| Wash. | +10 | +27 | +3 | -8 | -14 | -24 | -103 | +136 | 27 |
| Colts | +44 | +18 | +21 | -5 | -16 | -7 | -133 | +103 | 25 |

**Conclusion/Reflection**

The steps to finding each teams difficulty score are the following

1. Split each game from the original dataset to two data points by team and performance.
2. Preprocess data for model
3. Train and test model with optimal classification algorithm based on accuracy
4. Run entire dataset through model
5. Analyze results by comparing actual records to predicted records.
6. Use scoring metric to rank the teams by most difficult games played.

Step 3 in the process proved to be most interesting to me. Selecting the best model out of all the the possible classification algorithms was challenging. Each having their own pros and cons testing and cross validating really helped in the decision making. The choice of using SVM came down to a decimal edge over random forest, although random forest could have been the optimal algorithm as well. SVM is just slightly better model because it draws the line between winning and losing, while separating the points into the two sectors.

**Improvements**

More football game statistics such as field goals, number of punts by a team, or another feature could provide better results in determining difficult games. Score was a feature I strongly debated to keep or drop. Overall, I decided to drop the column, but for a comparison analysis the score feature should remain for evalation. Another improvement could be fine tuning the classifier algorithms even more so that random