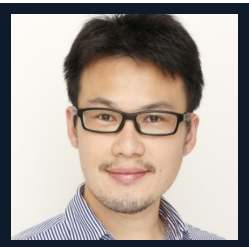
Predicting the Super Bowl LVI Winner

Mychael Gray

Data Science Intensive Capstone Project, November 1st 2021 Cohort

Springboard

Thanks to Springboard mentors



A Wang Ang, Principal Data Scientist

Michelle Jorgensen, Springboard Student Advisor

The Problem

According to data from the National Retail Federation (NRF), total spending for food, drinks, apparel, decorations, and other purchases for the day is expected to reach…

Americans spent:

16+ billions

or…

$85.36 per person.

Are there more important factors and qualities of a NFL Super Bowl team and can all of these factors be used to predict a winner accurately, from the 2022 season?

Who might care?

Investors/Gamblers

Businesses

Consumers & Enthusiasts

…and many more…

What factors might affect Score?

- Home or Away game?
- Timing: 1st game or last game?
- Rushing yards
- Passing yards
- Opponent

kaggle

NFL

Data Source

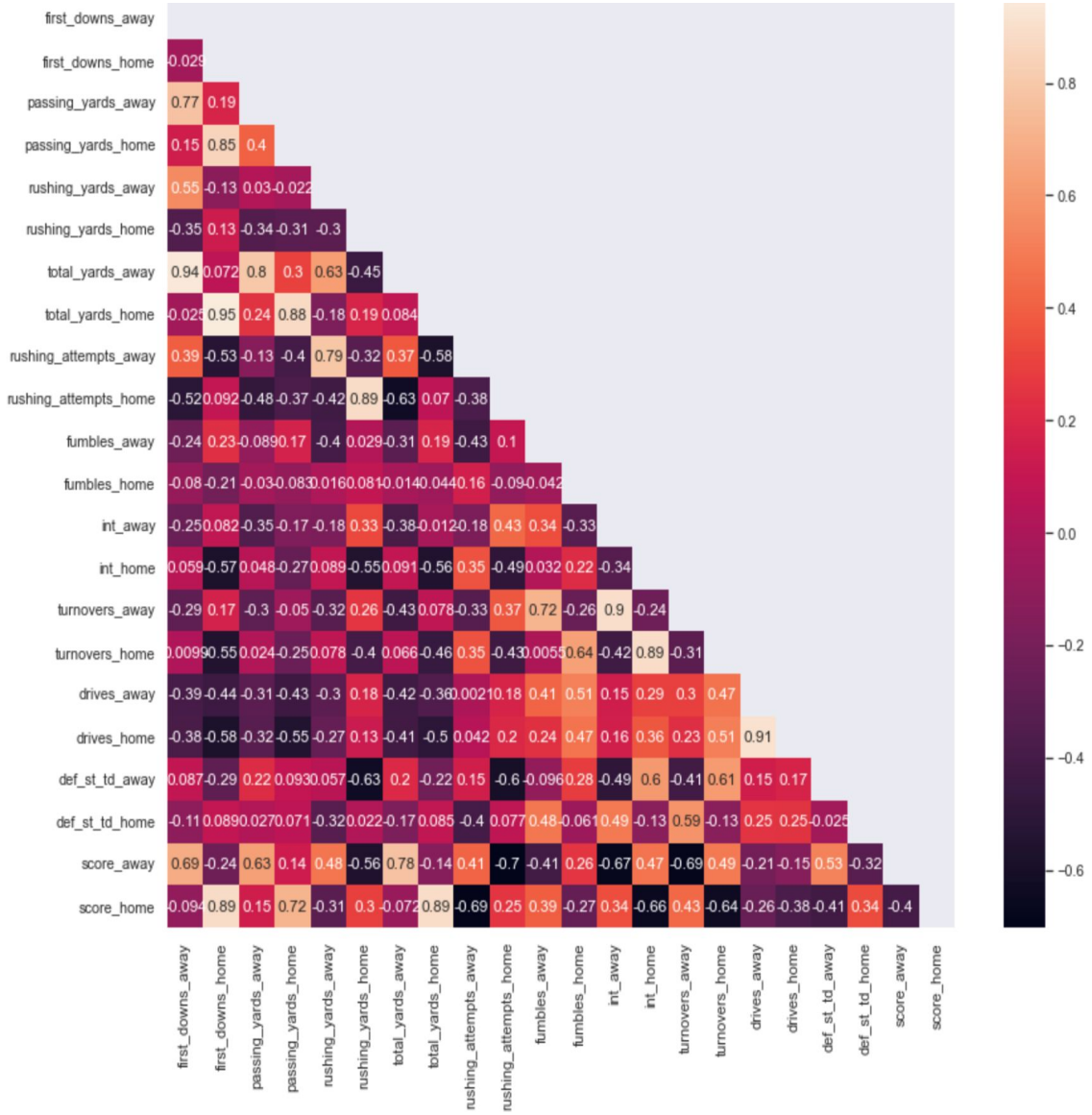
- Technical Factors:
- Time-Series Analysis
- Coaching
- Fumbles
- Sacks
- Star Power
- Team Chemistry

Industry

Some Historical Indicators

	date	away	home	first_downs_away	first_downs_home	third_downs_away	third_downs_home	four
0	2002-09-05	49ers	Giants	13	21	4-12	9-16	
1	2002-09-08	Jets	Bills	18	26	2-8	7-17	
2	2002-09-08	Vikings	Bears	19	20	5-13	7-13	
3	2002-09-08	Chargers	Bengals	27	13	6-10	4-11	
4	2002-09-08	Chiefs	Browns	24	24	5-11	4-11	

"score_away" and 'score_home' are potential targets for the data. The end score of the winner for each observation is what can be modeled. The other columns are potential features.



Modeling Overview

Type: Supervised learning

Regression Model: Predicting Score

No missing data & feature names updated

Tools: Python 3, scikit learn, and matplotlib

Modeling Steps

Data pre-processing steps:

1. Check data types
2. Create dummy or indicator features for categorical variables
3. Data splitting into training and test sets **(70% / 30%)**
4. Standardize the magnitude of numeric features using a scaler

Optional: Cross validation (CV) for hyperparameter tuning:

- CV/Time Series Splits: 205
- Using scikit-learn's grid search method
- Evaluation metric: MSE or Mean Squared Error

Feature Selection:
Ridge was used (but Lasso could be used) and plotted for visualization.

These steps can be piped together using sklearn pipeline class

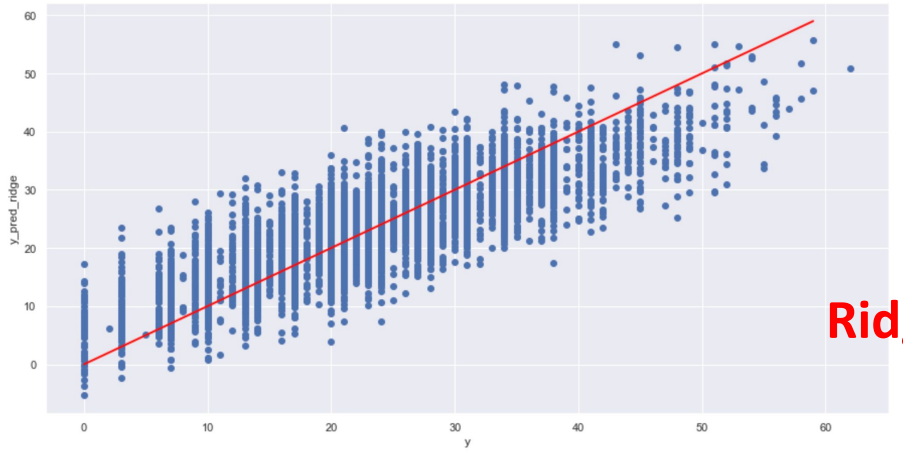
Testing using the same pipe (excluding cross validation)

Performance evaluation using holdout dataset (30% of the whole data)

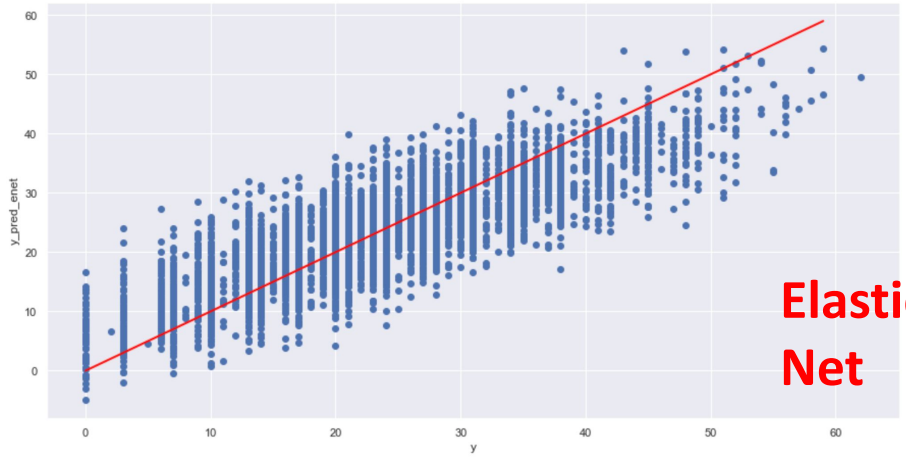
Regression Algorithms Used (Three Models):

1. Linear Regression – Using Ridge
2. Linear Regression – using Elastic Net
3. Linear Regression – using Random Forests

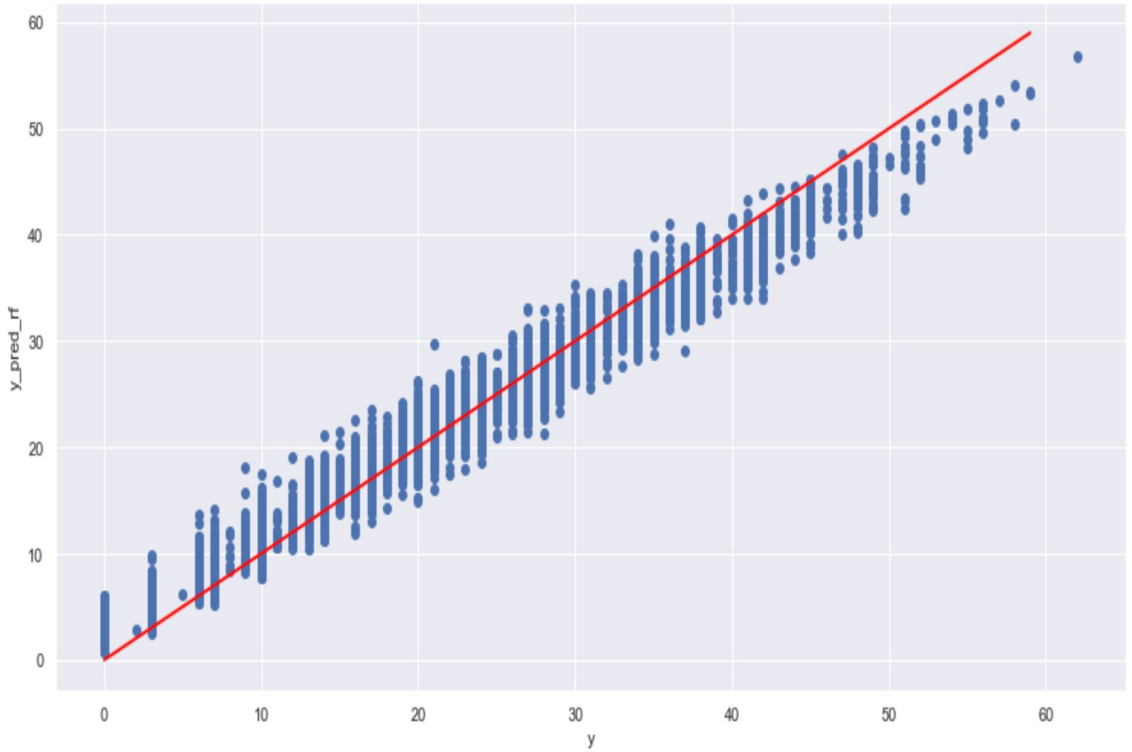
Model Comparisons



Ridge



Elastic Net



Random Forest Regressor

Linear Regression (using sklearn), Ridge, Elastic Net, and Random Forests

Details on the Best Model (ET)

Making a Ridge Regression model: Third model

Sklearn has a Ridge() function built into the linear_model module.

```
4]: from sklearn.linear_model import Ridge

# Split data if not split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)

# Initialize ridge model, normalize=True parameter assures all variables are scaled.
ridge = Ridge(alpha=0.1, normalize=True)

# We can use the scaled data from above to fit
ridge.fit(X_train, y_train)

# Make predictions
ridge_pred= ridge.predict(X_test)

# Score predictions
ridge.score(X_test, y_test)
```

```
4]: 0.8817195230671278
```

- By using the Ridge Regression Model we were able to increase the predictions of the predictive power to about 88%.

- By using a Lasso Regression Model we have the potential of increasing the predictions of the predictive power to about 91% while managing the features used.

Making a Lasso Regression model: Fourth model

Sklearn has a Lasso() function built into the linear_model module.

```
from sklearn.linear_model import Lasso

# Split data if not split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)

# Initialize ridge model, normalize=True parameter assures all variables are scaled.
lasso = Lasso(alpha=0.1, normalize=True)

# We can use the scaled data from above to fit
lasso.fit(X_train, y_train)

# Make predictions
lasso_pred= lasso.predict(X_test)

# Score predictions
lasso.score(X_test, y_test)
```

```
0.909819917007543
```

While the Ridge and Lasso models scored high, the Random Forest model scored the best but the time it took to build the model was an issue.

Assumptions, Limitations and Disclaimers

- We assume that all observations are independent, are assumed to be unrelated games.
- Used only 5357 observations/games and past data that is not current.
- The model will behave poorly if we try to predict prices of games that are scheduled too far in the future or other seasons.
- The game/team data does not contain all data (several sources can be included in the future).

More Ideas to Improve the Model in Future

- Engineer more features related with team statistics such as financial impact, fan sentiment, environmental/social impact (politics or weather), etc.
- Answer which teams are more volatile in weekly stats per game or more stable and why?
- Use social network and news media to extract sentiments about teams and sports/industry wide health to get more features

Conclusions

- All sources of datasets contributed to the predictive power of the model.
- Out of 3 supervised classification models, the slow Randomized Forests Trees provided the best results.
- Out of 39 features, we used 51 features (including dummies for teams) for the best model with historical performances data (which we engineered).
- With 70%-30% splitting, the test data set gave MSE = 5.08 and predicted a score of Rams: 26.7175, Bengels: 26.58 (very close to the actual score.

Thank you!

Mychael Gray

Email: graymychael@gmail.com

Linkedin: https://www.linkedin.com/in/graymychael/

Github: https://github.com/graymychael/

Project report:
https://github.com/graymychael/NFL_Super_Bowl_LVI_winner_prediction