Fantasy Football Captain Prediction Model

Jordan Purser
22 August 2017
Udacity Machine Learning Engineer Nanodegree
Capstone Project

I. Definition

Project Overview

Fantasy sports, where sports fans compete by selecting fictional teams of their favorite players has proved to be a successful recipe the world over. From Football to baseball, cricket and basketball, most popular sports have an arena where friends, family, colleagues and alike can duke it out for bragging rites and prizes. In Australia, one of, if not the biggest fantasy sport is AFL Fantasy (Australian Rules Football). Over 100,000 actively participate in the AFL Fantasy competition each year and many (myself included) take it very seriously. With multiple competition cash and non-cash prizes, betting markets, locally organized prize pools and bragging rites, its not hard to understand why people are willing to go that extra mile to gain an edge on their competition.

The basic premise of the game is to select a team of players who will score points based on the statistics they accumulate in-game. For example, lets say Player A's stat line at the end of the game reads:

Kicks	Handballs	Marks	Goals
5	4	3	1

If kicks are worth 3 points, handballs 2 points, marks 3 points and goals 6 points, then these stats will equate to a score of:

(5 x 3) + (4 x 2) + (3 x 3) + (1 x 6) = 38 points

The aim of the game is of course to generate as many points as possible.

One interesting mechanic of AFL Fantasy is that of the weekly captain choice. The mechanic works as follows: each round the fantasy coach chooses a player in their team as captain and this player has their score doubled as a result. As you can probably tell, it is important to choose a high scoring captain. In this research project, I investigate the use of modern machine learning techniques for the purpose of AFL Fantasy captain selection. My approach uses historical scoring data as input to regression models that make score predictions for players ahead of each round. These predictions are then used as a ranking mechanism for captain selection.

The AFL Fantasy website (https://fantasy.afl.com.au) and the AFL website (www.afl.com.au) serve as the primary sources of historical player statistics for my analysis. AFL Fantasy enthusiast website DT Talk (dreamteamtalk.com) provides expert and community based research that serve as the benchmark for research in the field. In particular, the Calvin's Captains article series serves as inspiration for this project. For the purposes of this analysis, 'AFL' will be used to refer to the Australian Football League, 'fantasy coach' to the individual playing fantasy football, 'player' to the actual footballers, 'round' to a playing fixture round, 'game' to an individual football match, and 'season' to a playing fixture season. Predicted score refers to the output of a regression model and is used to construct a ranking. Actual score is the actual score the player scored in a given round and is used to construct the benchmarks and primary evaluation metric.

Problem Statement

One of the most important decisions in AFL fantasy is that of the weekly captain choice. The captain choice mechanic as stated above, works by allowing the coach to select 1 player prior to each round for the purpose of doubling that player’s score. The goal of this project is to create a model capable of accurately predicting a player’s score for the coming round for the purpose of ranking and selecting effective captain choices.

The model will receive historical data about the past performance of a player and will output a prediction of the player’s score for the coming round. Each round a prediction will be made for each player and then these predictions will be ranked. The top 5 predicted player’s actual scores will then be averaged to produce a metric representative of the performance of the model for the purpose of picking a captain.

1. The following steps will be conducted for the purpose of constructing this model:
2. Gather statistics from AFL Fantasy site and construct historical features to be used in model
3. Perform preprocessing on features
4. Train and tune models (Gradient Boosted Trees, Support Vector Machine, Neural Network) using training and validation data
5. Evaluate models with validation data and choose final model
6. Evaluate final model on test data and evaluate against benchmarks

Metrics

The primary metric for the evaluation of the model will be the average actual score of the top 5 predicted players, as determined by the model for each round in the test set.

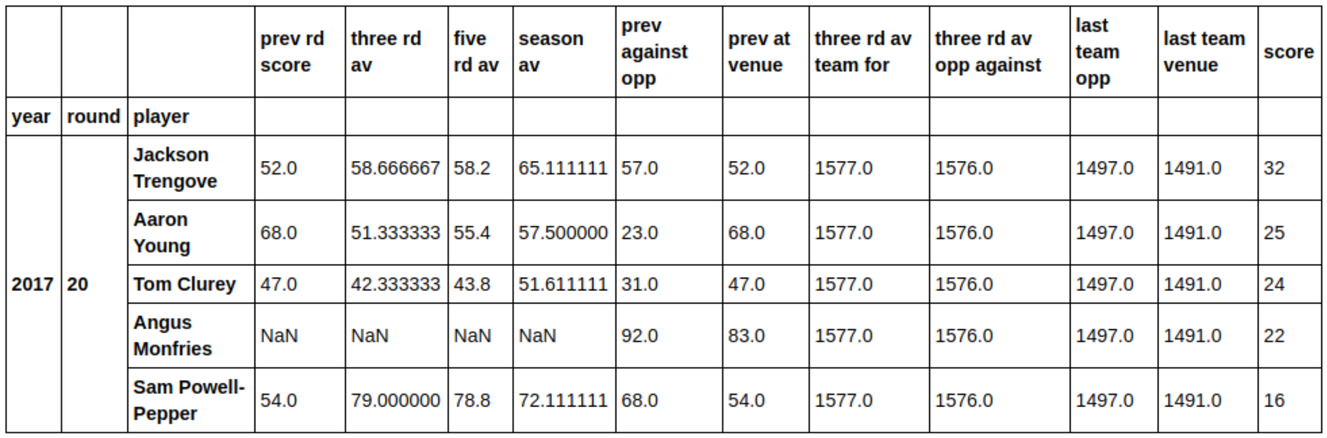
I chose to go with an average rather than simply the score of the top ranked player so that the metric is a bit more robust to outliers. This is also more reflective of the performance a fantasy coach using the model would achieve, as they would not always have access to the top ranked player each week. On top of this the metric will be directly interpretable as points scored. This will allow for a direct assessment of model performance and comparability with benchmarks.

For the purpose of evaluating the various machine learning models to be trialled, I will be using the R2 metric as generated on the validation dataset. The R2 metric was chosen because it is a great measure of the 'goodness of fit' of a model and because it is easy to understand and familiar for most readers.

II. Analysis

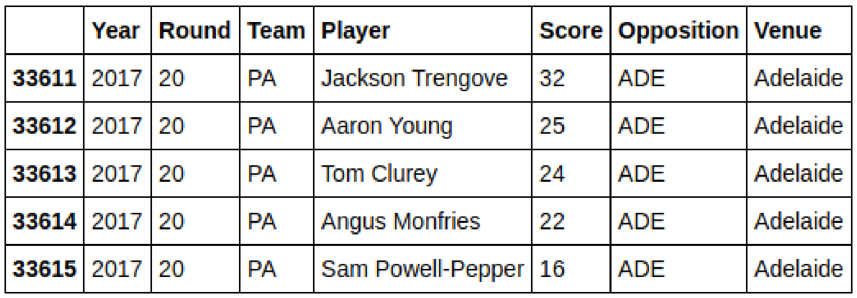
Data Exploration

The data set to be used for this analysis was self constructed from statistics data gathered from the the AFL Fantasy website (https://fantasy.afl.com.au) and fixture data recorded on the AFL website (www.afl.com.au). The data set is called 'capstone data final.csv' and is represented as follows:



			prev rd score	three rd av	five rd av	season av	prev against opp	prev at venue	three rd av team for	three rd av opp against	last team opp	last team venue	score
year	**round**	**player**											
2017	**20**	**Jackson Trengove**	52.0	58.666667	58.2	65.111111	57.0	52.0	1577.0	1576.0	1497.0	1491.0	32
		Aaron Young	68.0	51.333333	55.4	57.500000	23.0	68.0	1577.0	1576.0	1497.0	1491.0	25
		Tom Clurey	47.0	42.333333	43.8	51.611111	31.0	47.0	1577.0	1576.0	1497.0	1491.0	24
		Angus Monfries	NaN	NaN	NaN	NaN	92.0	83.0	1577.0	1576.0	1497.0	1491.0	22
		Sam Powell-Pepper	54.0	79.000000	78.8	72.111111	68.0	54.0	1577.0	1576.0	1497.0	1491.0	16

Originally, the data was in the following form:



	Year	Round	Team	Player	Score	Opposition	Venue
33611	2017	20	PA	Jackson Trengove	32	ADE	Adelaide
33612	2017	20	PA	Aaron Young	25	ADE	Adelaide
33613	2017	20	PA	Tom Clurey	24	ADE	Adelaide
33614	2017	20	PA	Angus Monfries	22	ADE	Adelaide
33615	2017	20	PA	Sam Powell-Pepper	16	ADE	Adelaide

This data can be found in the file ‘capstone data raw.csv’ and the code used to construct the final dataset can be found in the 'Feature Engineering.ipynb' notebook provided.

The final dataset set consists of 33,616 unique player fantasy statistics recorded for every player from every round between round 1, 2014 and round 20, 2017. Each row represents a distinct combination of year, round and player, shown by the first three columns of data that make up the index for the data set. The remaining data is completely numerical and represent transformations of the 'score' data from the raw data set.

Each observation (row) can be considered a snapshot of a player prior to a given round, with each feature representing different combinations of past scoring performance. The features have been hand engineered rather than algorithmically created in order to make use of my domain knowledge. The final column represents the target variable we are trying to predict with our regression.

Some statistics and an explanation of the features follows:

	prev rd score	three rd av	five rd av	season av	prev against opp	prev at venue	three rd av team for	three rd av opp against	last team opp	last team venue	score
count	32569.00	30707.00	29122.00	32569.00	21061.00	26391.00	32428.00	32428.00	26884.00	29238.00	33616.00
mean	72.36	72.88	73.15	72.01	73.72	73.24	1578.63	1578.63	1569.25	1582.40	71.76
std	26.87	20.84	19.37	19.88	27.58	27.50	100.89	95.63	158.81	159.81	27.46
min	-3.00	4.00	4.00	2.00	-3.00	-3.00	1254.67	1303.00	1132.00	1132.00	-3.00
25%	53.82	57.67	59.00	58.31	54.00	54.00	1511.33	1510.33	1462.00	1473.00	52.00
50%	71.00	71.62	72.00	71.00	72.00	72.00	1578.83	1575.67	1560.00	1573.00	70.00
75%	90.00	87.00	86.40	85.64	92.00	91.00	1648.67	1638.33	1681.00	1690.00	90.00
max	194.00	160.00	156.40	180.00	200.00	200.00	1928.67	1921.00	2033.00	2048.00	200.00

➔ score - this is the target feature of our analysis and represents the actual score of the player in that round. The average value and standard deviation are as expected given my experience with fantasy football and the max and min values despite being outliers, are legitimate.

➔ prev rd score - contains the most recent score of the given player for that year. The construction logic is as follows: If no previous score is present (such as is the case in round 1 or if a player has been injured), the player's last year average score is used. If this is not available, (for example, a new rookie or a player that was injured for the entirety of last season) then a null value is recorded.
Since this feature makes use of the score data directly, it is curious that we see a mismatch between the maximum values for these two columns. After some investigation I found that this phenomenon was caused by nuances in the calculation of prev rd score. Because prev rd score uses the player's previous year average for the first game a player plays in a given year, the 200 value is never recorded directly because it occurs in the final round of a year.

➔ three rd av - contains the average of the three most recent scores of the player for the year. The construction logic is as follows: if three previous games do not exist in the year, the player's previous season average is used as many times as necessary to make up the total of three values to be averaged. If this average does not exist, null is recorded for this value.
Looking at the statistics, a minimum value of 4 seems like an unlikely 3 game average for a player to have. After investigation, once again it is the result of the methodology used to calculate this column. Here, we have a case where a player played a single game in the prior year for a grand total of 4 points (due to an injury). Since the prior year average is the only value used for the calculation of the first game of the year, this player records a three round average of 4.

➔ five_rd_av – methodology is exactly the same as the three_rd_av feature, except it uses the latest 5 games of the year for its calculation.

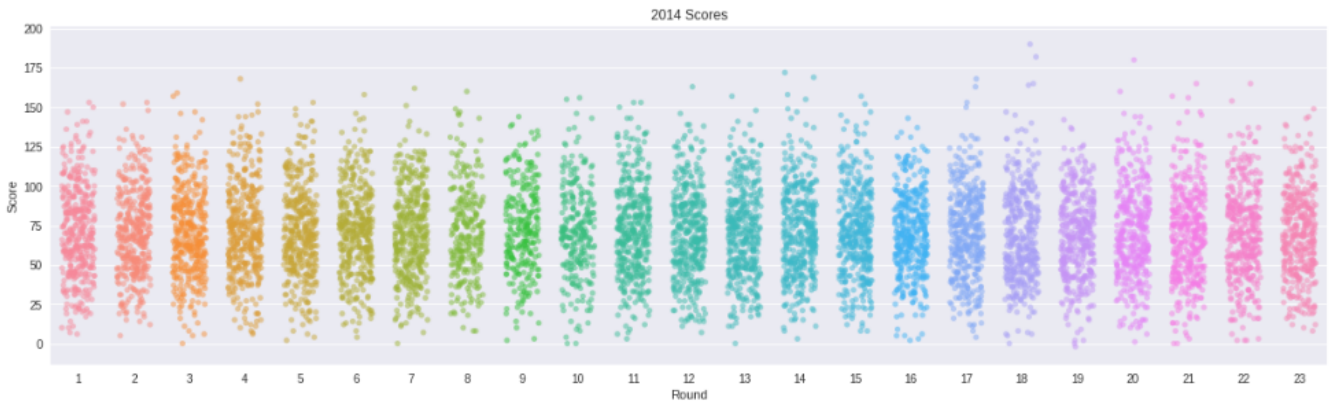
➔ season_av - calculates the rolling average score of a player for the current year. The construction logic is as follows: If no scores have been recorded, last year's average is used. If last year's average is not available, null is recorded.
Looking at the stats, a min season average of 2 and a max of 180 may seem a bit extreme, however when you consider that some observations will be based off of 1 game (the first game a player plays in the season), these values become acceptable. Further investigation confirms that these were indeed first round played scores.

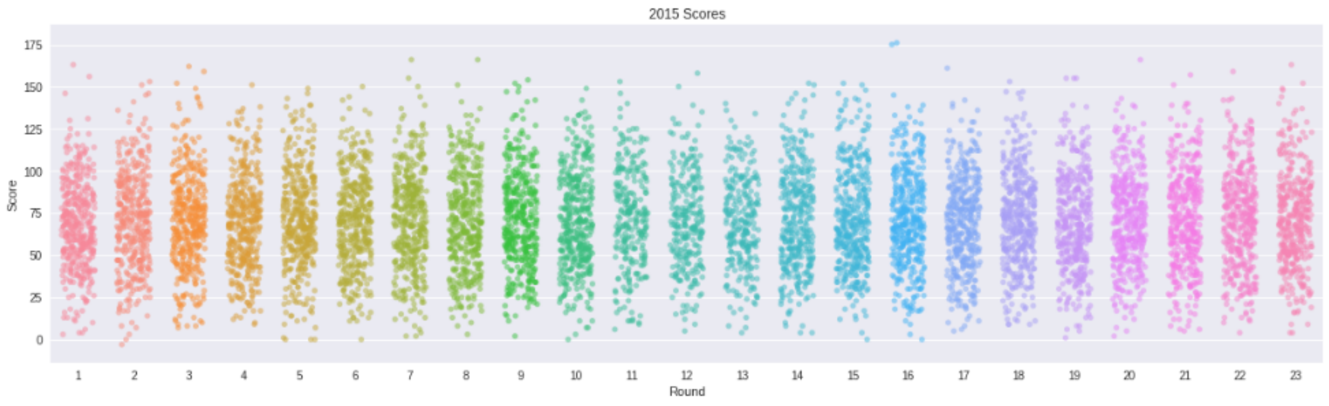
➔ prev against opp - is the most recent previous score of the player against their current round opposition. The construction logic is as follows: If not available, null is recorded. Unlike previous features, the search for the most recent game against opposition is not limited to the current year.

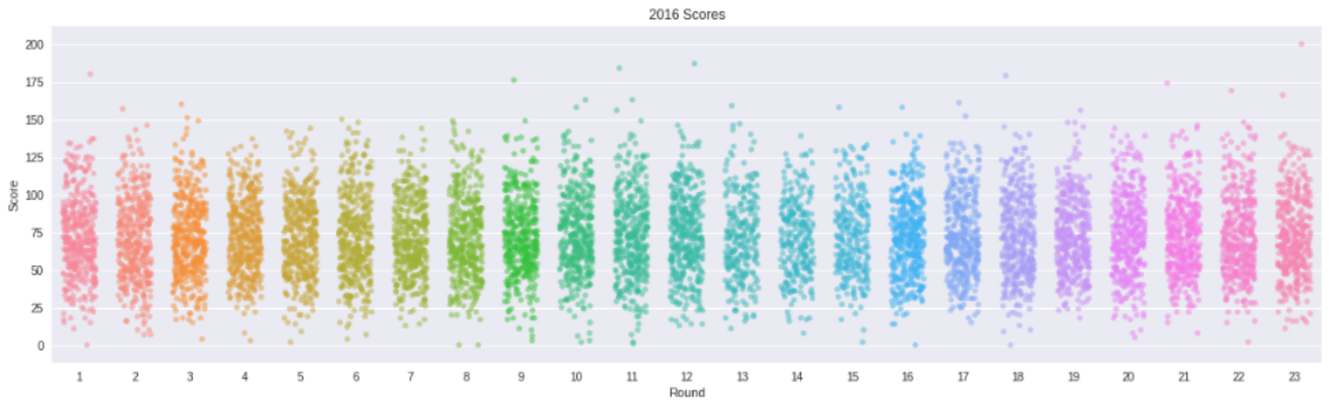
- ➔ prev at venue - most recent prior previous score of the player at their current round venue. The construction logic is as follows: If not available, null is recorded. Like above, the search for the most recent game at venue is not limited to the current year.
- ➔ three rd av team for - averages the team score over the most recent three rounds in a given year. The construction logic is as follows: A team score is calculated on a per round basis by adding the scores of all players in that (real life) team for that round. The most recent three previous team scores are taken and averaged, where the team is the team of the player in question. If three rounds are not available, the team's prior year average is used as many times as necessary in the calculation of the average. If last year's average team score is not available, null is recorded.
- ➔ three rd av opp against - averages the team scores of the opposition's last three opposition teams. You can think of this feature as a proxy for how easy the opposition is to score against. The construction logic is as follows: the team scores of the last three teams to play the opposition are averaged. If three rounds are not available, last year's average team points against for the opposition is used as many times as necessary in the calculation of the average. If last year's average team points against is not available, null is recorded.
- ➔ last team opp - the player's most recent team score against the current round opposition. The construction logic is as follows: If not available, null is recorded. The search for the last team score against opposition is not limited to the current year.
- ➔ last_team_venue - same as above, except player's last team score at current round venue is used.

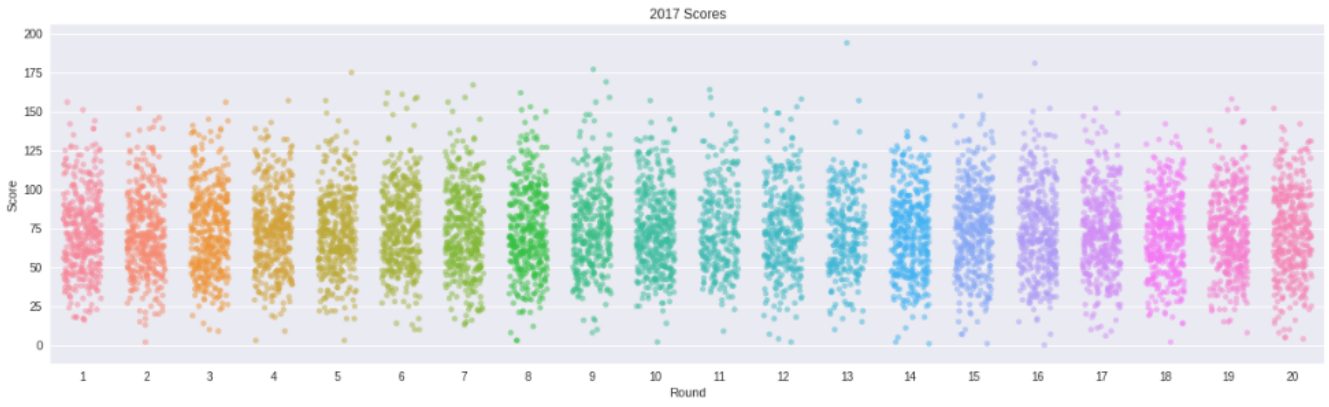
Exploratory Visualization

The plot below shows the distribution of scores across each round for each year. This plot highlights exactly what it is we are attempting to predict, the score. As you can see by the darker areas of the plot, the majority of the scores each week fall between 25 and 125 points. the plot also shows that this remains constant across rounds and years. Multiple scores outside of this range occur each round but are comparatively rarer. It is these scores above the 125 mark that represent that which we are attempting to identify.









If the model is accurate enough, then theoretically we should be able to produce distributions like those above for each round and rank in order to identify those players with high scoring potential.
The code used to generate these visualizations can be found in ‘Exploratory Data Analysis.ipynb’.

Algorithms and Techniques

The following machine learning algorithms have been chosen for evaluation for the purpose of constructing a AFL fantasy score prediction model:

- ➔ Gradient Boosted Trees
- ➔ Support Vector Machines
- ➔ Neural Networks

The above algorithms were chosen due to the breadth of approaches they represent and historical success when applied to supervised learning problems. Specifically:

Gradient Boosting has been shown to produce outstanding performance across a wide array of problem domains on the data science competition platform Kaggle (Gorman, 2017). Its also a ensemble method, unlike the other algorithms to be evaluated and when built upon decision trees, it tends not to overfit. Although a computationally intensive model, this shouldn’t be a problem as the dataset is small.

Support Vector Machines are another high performance model that have successfully been applied to a wide range of problem domains including similar problems in sports statistics prediction (Lutz, 2015). It tend to be robust to outliers and work well when the number of features is not significantly larger than the number of observations.

Neural networks represent the state of the art model for tasks as diverse as self driving cars to stock market price prediction. Neural networks are complex and take a lot of data to train. They can also be prone to overfitting if appropriate measures are not taken. Using a neural network represents a complex approach towards tackling the problem and the small amount of data available may prove to be a limiting factor in its success. Nevertheless, the indisputable performance of neural networks across a wide range of domains demands its inclusion in this investigation.

With so many models naturally comes many parameters to tune. The following outlines both the constant and the tunable parameters for each model:

- ➔ Gradient Boosting: For this model, decision trees will be used as the underlying method. The loss function to be used is least squares. The following parameters are to be tuned: learning rate and n_estimators.
- ➔ Support Vector Machine: For this model, both a linear and rbf kernel will be evaluated. The following parameters are to be tuned: penalty factor (C), and the width of penalty free zone (epsilon).
- ➔ Neural Network: For this model I will use the standard neural network structure with mean squared error as the loss function and gradient decent as the optimizer. ReLu will be used as the activation function. Parameters to be tuned include: learning rate, number of layers, width of layers, batch size, and training iterations.

Techniques to be used include cross validation and feature normalization. Cross validation represents best practice for tuning model parameters as well as provides a way to evaluate the different models without incorporating the testing set. Feature normalization helps with the speed of model training and minimizes any unfair treatment of features due to scale.

Benchmark

There currently exists a number of different heuristics for choosing a captain employed by AFL fantasy coaches. 3 of the most popular are:

1. Choosing a high averaging player
2. Choosing a in-form player
3. Choosing a captain mentioned in the 'Calvin's Captains' article produced by DT Talk

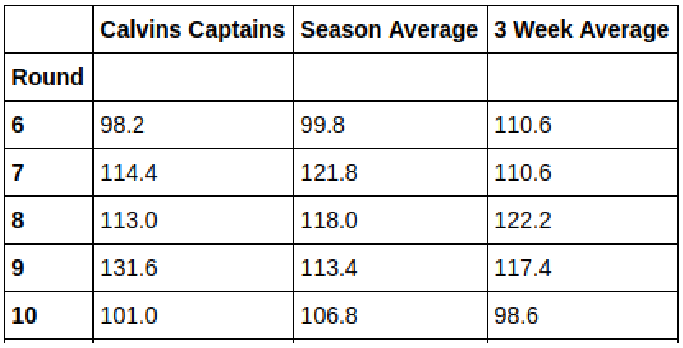
For the purposes of my analysis all three will serve as benchmarks for evaluating my model, as each is sufficiently unique and reflective of popular contemporary captain choice approaches. Outperformance of these benchmarks would indicate a model superior to that used by almost all AFL Fantasy coaches.

The respective benchmarks are constructed as follows:

1. Prior to the beginning of each round, identify the top 5 players according to season average score. Calculate the average actual score of these players for that round.
2. Prior to the beginning of each round, identify the top 5 players according to 3 week average score. Calculate the average actual score of these players for that round.
3. Prior to the beginning of each round, identify the top 5 ranked captain choices as outlined in the 'Calvin's Captains' article produced by DT Talk. Calculate the average actual score of these players for that round.

Top 5 averages were chosen to comply with the evaluation methodology of the model. The same rationals of minimizing the affect of outliers, representing the approach of fantasy coaches, and being directly interpretable as a score remain here. Also doing this means that the benchmarks will be directly comparable with the output of the model.

The following is an excerpt of the benchmark data:



	Calvins Captains	Season Average	3 Week Average
Round			
6	98.2	99.8	110.6
7	114.4	121.8	110.6
8	113.0	118.0	122.2
9	131.6	113.4	117.4
10	101.0	106.8	98.6

For the exact methodology and code used to construct the benchmarks, please see included 'Benchmark Construction.ipynb' notebook. Data for the benchmarks can be found in ‘Benchmarks.csv’. Raw data used to construct the benchmarks is contained in ‘Season Average Benchmark.csv’, ‘3 Week Benchmark.csv’, and ‘Calvins Captains Benchmark.csv’ respectively.

III. Methodology

Data Preprocessing

The preprocessing required for the data set was as follows:

1. Remove columns with missing values
   Since the standard Sci-kit Learn algorithms to be used in the following section won't run on missing input, it was necessary to make a decision on missing values in the data set. Since appropriate proxies for much of the missing data have already been included during the dataset

construction, the remaining missing values represent datapoints who's value was inappropriate to estimate based on available information and techniques. Since the likely approach taken by a fantasy coach when faced with a player lacking sufficient historical data would be to exclude the player from consideration as captain, I have chosen to remove these datapoints from consideration by the models.

2. Create training/validation/testing splits of data
   In order to tune the parameters of the selected models without introducing bias by training on the testing set, a validation set was created. The training set will be used to train the models, the validation set to tune parameters/hyperparameters and select the final model, and the testing set to create the rankings and evaluate the final model. The data was split by round rather than a number/percentage of observations to ensure that the splits did not occur mid-round. The training set consists of 64 rounds (11721 observations), the validation set of 10 rounds (2985 observations) and the testing set of 15 rounds (4305 observations).
3. Extract target variable from data set
   Self explanatory. The target values (score) is separated from the input features.
4. Normalize features
   In order to ensure that features are not treated differently because of magnitude and to speed up training, feature normalization was implemented. The normalization process is applied on a feature by feature basis and involved the calculation of a standard score. This involves the subtraction of the feature mean from each value, then division by the feature's standard deviation.

$$z = \frac{x - \mu}{\sigma}$$

The exact code used for the above preprocessing steps can be found in the 'Preprocessing and Model Implementation.ipynb' notebook.

Implementation and Refinement

The implementation and refinement process can be broken down into two major components, which can each be further broken down into subcomponents:

1. The implementation and evaluation of models
   - Import and initialize models
   - Establish parameter search space
   - Train and tune models using training data and grid search
   - Construct tuned model using optimal parameters
   - Evaluate tuned model using validation set

2. The generation of predictions and construction of evaluation metrics using optimal model
   - Select optimal model based on validation set results
   - Use model to generate predictions on test set
   - Evaluate model test set predictions
   - Add predictions to features and construct data for evaluation

Firstly, the Gradient Boosting model was implemented. For this model I made use of Scikit-Learn's GradientBoostingRegressor() model from the sklearn.ensemble module. The model was initialized with default parameters which included the desired loss function: least squares. Next, a parameter grid search was constructed with Scikit-Learn’s GridSearchCV() object from the sklearn.model_selection module. The following parameters values were specified for tuning:

```
{'learning_rate': [0.01, 0.03, 0.1, 0.3], 'n_estimators': [10, 100, 200, 500, 1000]}
```

Grid search was implemented with a R2 scoring function and returned an optimal parameter choice of

```
{'learning_rate': 0.01, 'n_estimators': 500}
```

A model was then fit to the training data using these parameters and then evaluated on the validation set. The R2 score for the validation set was 0.3239.

Next the Support Vector Machine model was implemented. This time I made use of Scikit-Learn’s SVR() model from the sklearn.svm module. The model was initialized with default parameters. Another parameter grid search was constructed, this time with the following parameter values:

```
{'C': [0.1, 1, 10], 'epsilon': [0.01, 0.1, 1], 'kernel': ['linear', 'rbf']}
```

The optimal parameter selection was:

```
{'epsilon': 1, 'kernel': 'linear', 'C': 0.1}
```

The model was then fit to the training data using these parameters and then evaluated on the validation set. The R2 score for the validation set was 0.3319.

Lastly, the neural network model was implemented. The model used was Scikit-Learn’s MLPRegressor() from the sklearn.neural_network module, which was initialized with ReLu activation functions and stochastic gradient decent as the optimization method. A parameter grid search was again implemented to evaluate the following parameter choices:

```
{'hidden_layer_sizes': [(5,), (20,), (20, 5), (5, 3)], 'alpha': [0.003, 0.0001, 0.0003],
 'batch_size': [16, 32], 'max_iter': [1000, 10000]}
```

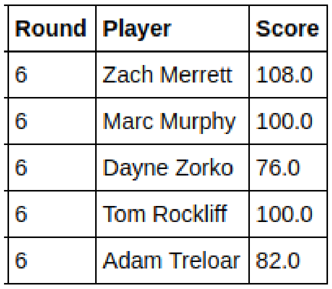
The optimal parameter selection was:

```
{'alpha': 0.0001,
 'batch_size': 16,
 'hidden_layer_sizes': (5,),
 'max_iter': 10000}
```

The model was then fit to the training data using these parameters and then evaluated on the validation set. The R2 score for the validation set was 0.2672.

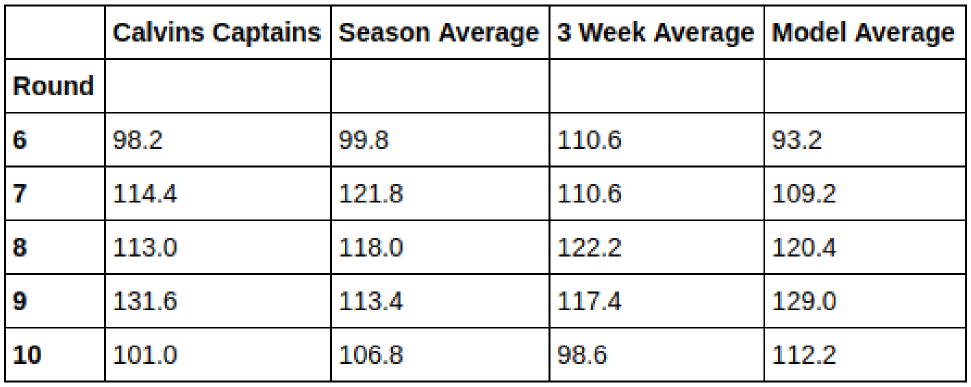
Based on the validation set results, the SVM model was chosen as the final model. Next, I used this model to perform predictions on the test set. Using these predictions, R2 score on the test set was 0.3052.

Finally, I added the prediction data to the testing dataset (dataframe containing both features and targets) in order to construct the evaluation metric. As stated above, the primary evaluation metric is the construction of an average of the top 5 predicted player’s actual scores for each round in the test set. In order to construct this, I first grouped the data by round and then extracted the scores of the top 5 players as ranked by their predicted score. Note that the predicted score is used for ranking. The actual score is used for the calculation of the evaluation metric. An excerpt of this data can be seen below:



Round	Player	Score
6	Zach Merrett	108.0
6	Marc Murphy	100.0
6	Dayne Zorko	76.0
6	Tom Rockliff	100.0
6	Adam Treloar	82.0

This data was then grouped by round and averaged to produce the evaluation metric. It was then added to the benchmark data constructed by the same process:



	Calvins Captains	Season Average	3 Week Average	Model Average
Round				
6	98.2	99.8	110.6	93.2
7	114.4	121.8	110.6	109.2
8	113.0	118.0	122.2	120.4
9	131.6	113.4	117.4	129.0
10	101.0	106.8	98.6	112.2

The exact code used to implement the models and create the evaluation metric data can be found in the ‘Preprocessing and Model Implementation.ipynb’ notebook. The top 5 predicted data can be found in ‘Model Prediction Top 5s.csv’ and the evaluation data in ‘Evaluation Data.csv’.

In conducting the implementation of my model, I did run into a few setbacks. Firstly, I had to forgo the use of the Naive Bayes model specified in my project proposal due to its lack of suitability for regression problems. I initially selected this model based on a poor understanding of its use case. Next, I had issues getting GridSearchCV() to work with my predefined validation set. It was my initial intention to perform both the parameter tuning and model selection evaluation with the validation set, however after many failed attempts, I eventually settled tuning my parameters on the training set using GridsearchCV() and leaving the validation set for the model selection choice. The final challenge occurred when attempting to implement the neural network. Initially I attempted to use Google’s Tensorflow library to implement this model, but it proved difficult and error prone. Instead I opted for Scikit-Learn’s MLPRegressor() model from the sklearn.neural_network module as it proved much easier to use and remained consistent with the above model implementations.

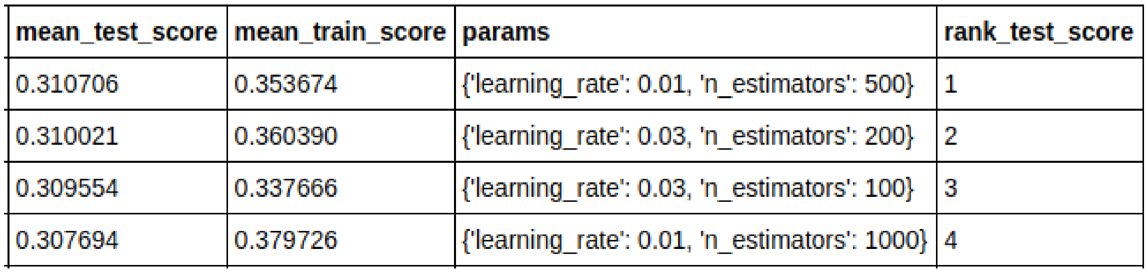
IV. Results

Model Evaluation and Validation

The final model, as defined above is a Support Vector Machine that has been trained on the training data with the following parameters:

{'epsilon': 1, 'kernel': 'linear', 'C': 0.1}

The final model parameter choices were chosen after grid search analysis conducted on the training set. Analyzing the grid search results shows similar performance with other parameter choices indicating that the model is robust with respect to parameter choice:



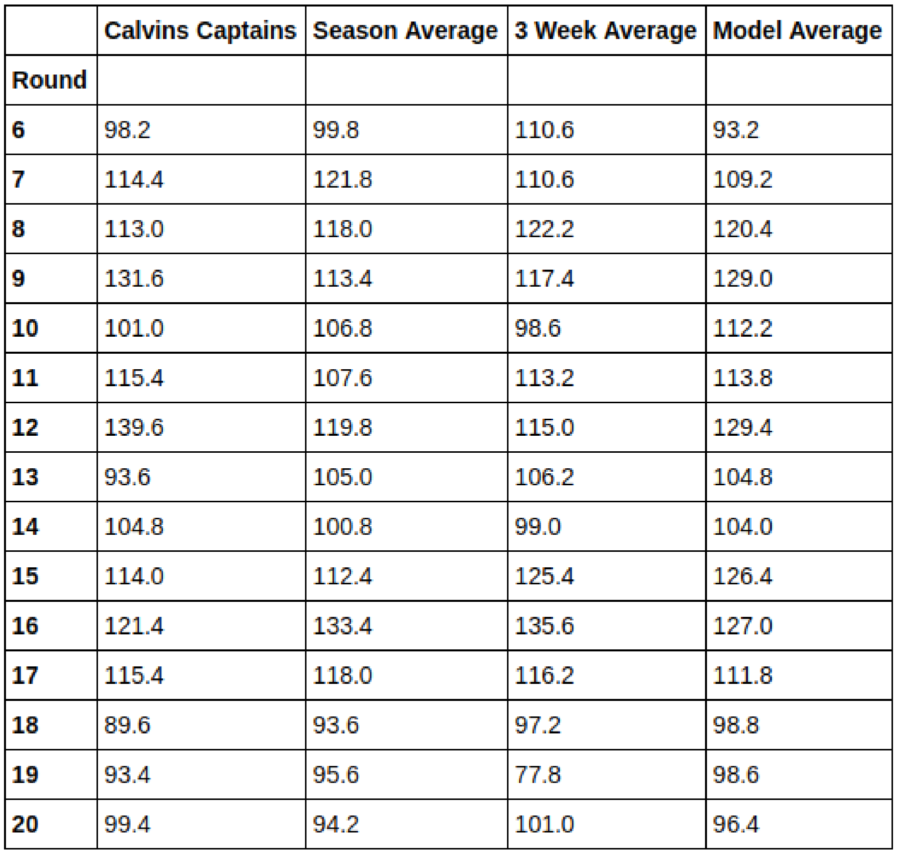
mean_test_score	mean_train_score	params	rank_test_score
0.310706	0.353674	{'learning_rate': 0.01, 'n_estimators': 500}	1
0.310021	0.360390	{'learning_rate': 0.03, 'n_estimators': 200}	2
0.309554	0.337666	{'learning_rate': 0.03, 'n_estimators': 100}	3
0.307694	0.379726	{'learning_rate': 0.01, 'n_estimators': 1000}	4

The model itself was chosen over its competition based on it's validation set results. The validation set was only used for the purpose of choosing a model and therefore represented untrained data for the model. The resulting validation score of 0.3239 is similar to the training scores, indicating that the model is not overfit. Model performance on the unseen test set further confirms this with a score of 0.3052. The model's performance on the unseen validation and testing sets confirm that the model is robust to unseen data and that its output can be trusted. Although a R2 score of 0.3 leaves a lot on the table with regards to predictive power, given the stochastic nature of the value we are attempting to predict as well as the limited data we are using to predict it, I think it performs quite well. Even with incorporating the suggestions below (see improvements section), a phenomenon like fantasy football performance still has too much inherent randomness to be predicted with high accuracy. The predictions themselves however, should not be relied upon as reliable indicators of current round scoring.

After performing sensitivity analysis that involving scalling up, scalling down, and switching the sign of the testing dataset, it unfortunately has to be said that the model does not stand up. In the model's defense data like this although numerically possible, is not really practically possible. The model has already been tested on the realistic range of possible values in the testing set and stood up robustly, however this sensitivity analysis serves as a warning to those wishing to use the model and highlights that the model is unlikely to stand up to any sort of significant change to the way the game is played or score is calculated. For the code and results of the sensitivity analysis, see 'Preprocessing and Model Implementation.ipynb'.

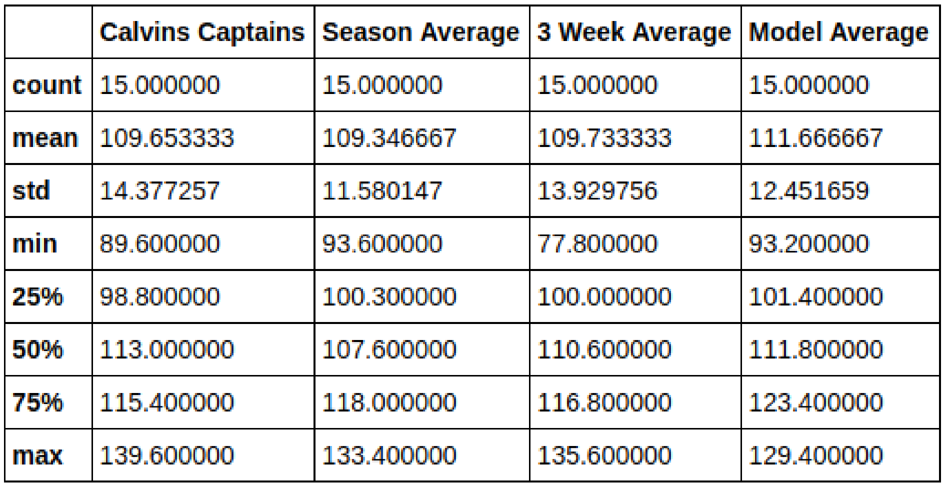
Justification

The following table displays the average score of the top 5 players as defined by each methodology for each of the testing rounds:



	Calvins Captains	Season Average	3 Week Average	Model Average
Round				
6	98.2	99.8	110.6	93.2
7	114.4	121.8	110.6	109.2
8	113.0	118.0	122.2	120.4
9	131.6	113.4	117.4	129.0
10	101.0	106.8	98.6	112.2
11	115.4	107.6	113.2	113.8
12	139.6	119.8	115.0	129.4
13	93.6	105.0	106.2	104.8
14	104.8	100.8	99.0	104.0
15	114.0	112.4	125.4	126.4
16	121.4	133.4	135.6	127.0
17	115.4	118.0	116.2	111.8
18	89.6	93.6	97.2	98.8
19	93.4	95.6	77.8	98.6
20	99.4	94.2	101.0	96.4

and the statistics for the respective models are:



	Calvins Captains	Season Average	3 Week Average	Model Average
count	15.000000	15.000000	15.000000	15.000000
mean	109.653333	109.346667	109.733333	111.666667
std	14.377257	11.580147	13.929756	12.451659
min	89.600000	93.600000	77.800000	93.200000
25%	98.800000	100.300000	100.000000	101.400000
50%	113.000000	107.600000	110.600000	111.800000
75%	115.400000	118.000000	116.800000	123.400000
max	139.600000	133.400000	135.600000	129.400000

As you can see the machine learning model manages to record an average score in excess of the benchmarks across the test set. The model is competitive across all quartiles and also records the second smallest standard deviation of scores. These results indicate that the machine learning model provides an adequate alternative to the captain choice selection problem that can compete with current, ubiquitous methods for selecting captains.

Whether or not the model outperforms the benchmarks is a bit more of a difficult question. Although recording a higher average than the other models, the small sample size means this outcome could be the result of chance. The model also records the worst maximum average score, underperforming the other models by quite significantly. One would need a much larger sample size to claim outperformance with any sort of confidence.

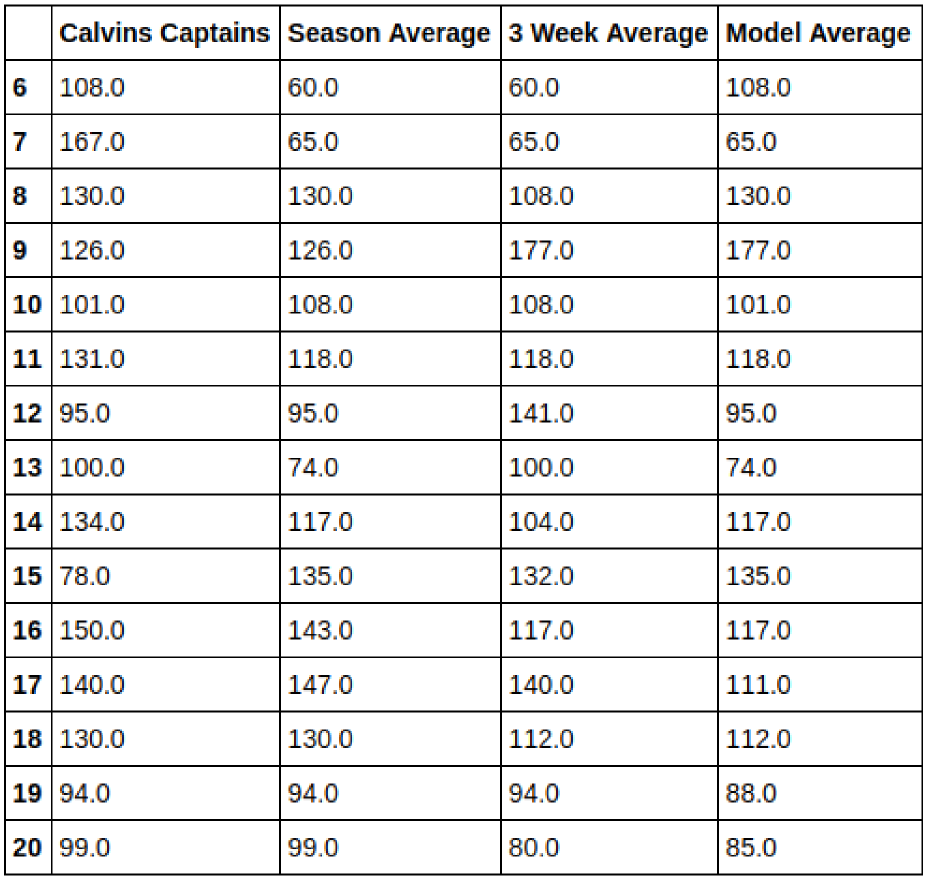
Overall, I view the model as a success but not as a solution to the captain choice problem. The outperformance of the machine learning model is too small and and recorded across too few observations to claim that it represents a new standard to captain selection. What this result does do is open the door for further research. This result serves as a proof of concept and makes me extremely excited to continue this research. Considering the result in within the context of the limited features used means improvement could be significant. Overall I am very pleased with the result which achieved everything I'd hoped it would.

Some interesting additional metrics include the number of round wins (simple count of times model was the highest scoring in a round):

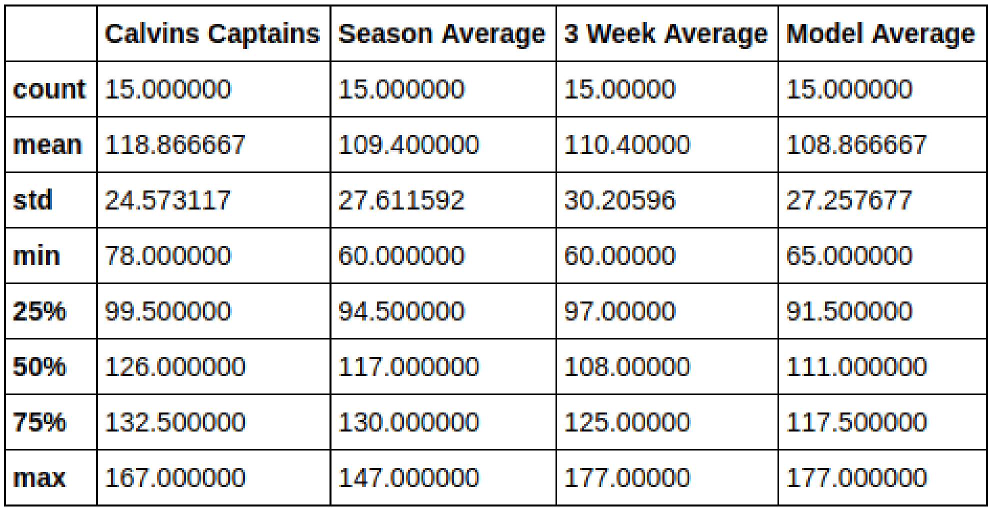
Calvins Captains	Season Average	3 Week Average	Model Average
4	2	5	4

Interestingly, the 3 Week Average benchmark records the most amount of round wins. Taking note of the statistics above, it also manages to record the single worst score of only 77.8 points.

Another interesting metric is the performance of the models based only on their top selections:



	Calvins Captains	Season Average	3 Week Average	Model Average
6	108.0	60.0	60.0	108.0
7	167.0	65.0	65.0	65.0
8	130.0	130.0	108.0	130.0
9	126.0	126.0	177.0	177.0
10	101.0	108.0	108.0	101.0
11	131.0	118.0	118.0	118.0
12	95.0	95.0	141.0	95.0
13	100.0	74.0	100.0	74.0
14	134.0	117.0	104.0	117.0
15	78.0	135.0	132.0	135.0
16	150.0	143.0	117.0	117.0
17	140.0	147.0	140.0	111.0
18	130.0	130.0	112.0	112.0
19	94.0	94.0	94.0	88.0
20	99.0	99.0	80.0	85.0



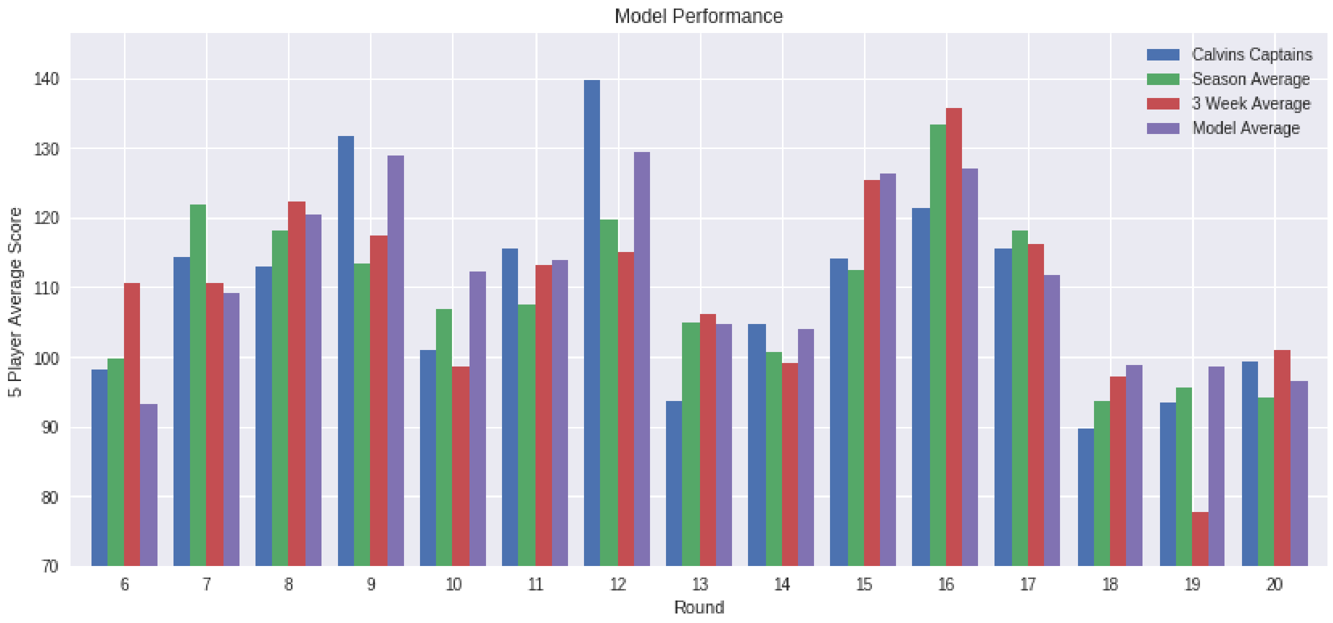
	Calvins Captains	Season Average	3 Week Average	Model Average
count	15.000000	15.000000	15.00000	15.000000
mean	118.866667	109.400000	110.40000	108.866667
std	24.573117	27.611592	30.20596	27.257677
min	78.000000	60.000000	60.00000	65.000000
25%	99.500000	94.500000	97.00000	91.500000
50%	126.000000	117.000000	108.00000	111.000000
75%	132.500000	130.000000	125.00000	117.500000
max	167.000000	147.000000	177.00000	177.000000

As you can see here, the machine learning model significantly underperforms, recording the lowest average score despite simultaneously recording the equal highest single score. As far as picking the single best captain for the round is concerned, Calvin's Captains is the clear winner.

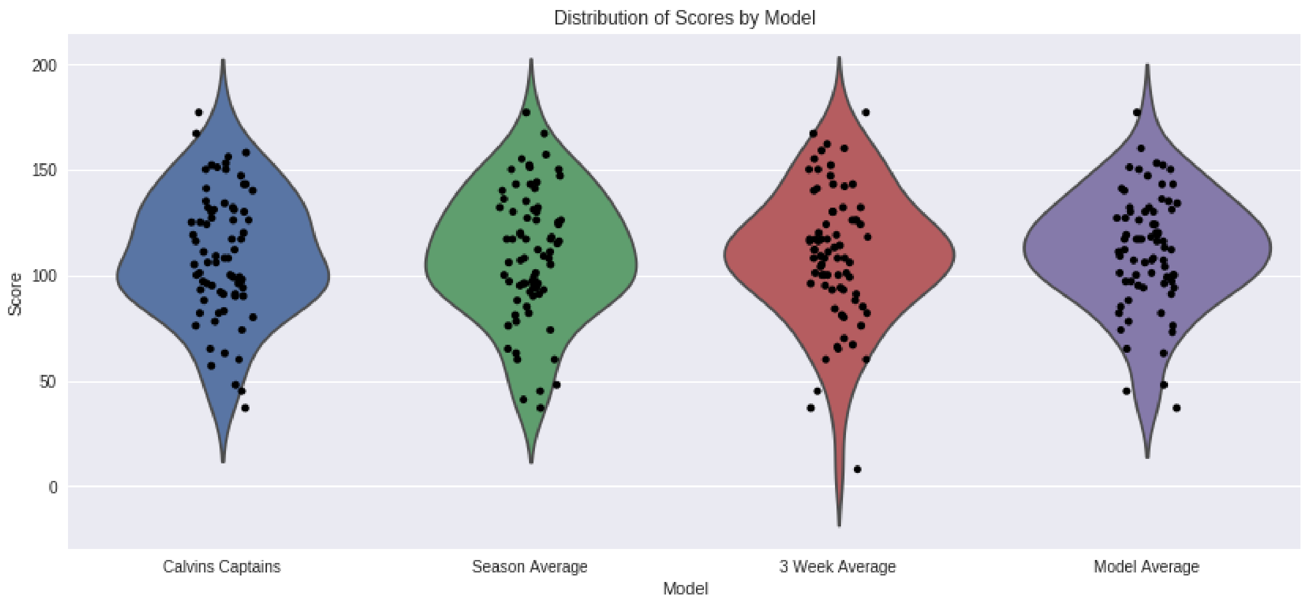
V. Conclusion

Free-Form Visualization

The below plot is a visual representation of the evaluation data. Each bar represents the average performance of the top 5 suggested captains for each model across each testing round.



This visualization really helps to illustrate the similarity in performance across models. Although the variation across rounds can be quite large, the model's performances tends to be quite correlated. Another interesting visualization shows the distribution of the scores of players selected by each model:



This visualization gives meaning to the quartile data discussed above in the justification section. You can see clearly that the 3 week average model has the longest downside tail, and that all model's distributions are concentrated around the 110 point mark.

Reflection

In this project, I investigated whether modern machine learning models can be used to predict AFL Fantasy scores for the purpose of ranking captain choices. For this purpose, I constructed a dataset of historical score and fixture data from the AFL and AFL Fantasy websites and used it to engineer historical score features for use in my models. I constructed machine learning models consisting of Gradient Boosted Trees, Support Vector Machines and Neural Networks and trained these models on the data. The models were evaluated using the cross validation technique and a final model was chosen for the purpose of generating predictions on the testing data. Lastly, these predictions were used to rank players and then used to construct a model for evaluation, consisting of an average of the top 5 player's actual scores.

Overall, I was able to construct a model capable of performing on par with currently used methods of captain selection. Whilst this result does not represent a solution to the captain choice problem, I am very pleased with this result and believe that further improvements are capable with the application of more advanced techniques and acquisition of more data. I am encouraged by my results and will continue my research in a bid to improve my AFL Fantasy results.

The project however was not without it's difficulties, the first of which encountered when attempting to engineer the features for input to my models. This proved to be a long and arduous task involving much thought and investigation into the nuances of the calculations. Working on or procrastinating about this section was the single most time consuming component of the project.
My second major challenge came when attempting to implement the models. Firstly, I had to forgo the use of the Naive Bayes model specified in my project proposal due to its lack of suitability for regression problems. The initial selection of this model was based on a poor understanding of the use case of Naive Bayes and was probably the source of many misunderstandings in the review process for

the project proposal section of the capstone project. The second problem faced in model construction was getting GridSearchCV() to play nice with my predefined validation set. It was my initial intention to perform both the parameter tuning and model selection evaluation with the validation set, however after many failed attempts to employ this with Scikit-Learn's GridSearchCV() function I eventually settled on allowing the parameter tuning to occur on the training set. The final challenge occurred when attempting to implement the neural network in Tensorflow. Due to Tensorflow's high level of complexity, it proved too difficult and time consuming to implement the neural network and so a decision was made to use Scikit-Learn's MLPregressor() model instead. Despite these setbacks, this experience was character building and I am looking forward to my next research challenge.

Improvement

During the conduction of this research project there were many opportunities to identify possible avenues for improvement of the final model. Some of course, represent the immortal problems of the data scientist (such as more data) but others are more nuanced. The following is a list of possible avenues for improvement as well as their rational:

- ➔ More Data – having more data is universally appealing for data scientists, however in this case it is likely to only help marginally. The game of Australian Rules Football changes rapidly with a myriad of rule updates and the constant change of coaching tactics so data becomes stale every 5 years or so.
- ➔ More Features – the obtainment of useful features outside of historical player scoring would likely lead to the single most significant improvement in model performance of anything on this list. Features representative of injury data, game significance, time on ground, preseason form, rookie form, suspension information and even expert opinion could be incorporated into the dataset to improve predictive power. Most of these additional features are incorporated qualitatively in the 'Calvin's Captains' benchmark model and would no doubt have a positive impact if included in my dataset.
- ➔ Different Models – Other regression models could be used, including multiple model solutions (stacking). Although I used what I believed to be the most suitable models for the problem, other models could be trialled and in particular, combined to produce better results.
- ➔ Further Parameter Tuning – Particularly in the case of the neural network model, further improvement is likely possible as a result of more tuning.
- ➔ Feature Engineering – the approach taken in the project was to hand engineer features using my domain knowledge of the problem, however computational methods for feature construction have been shown to be incredibly successful across a wide range of domains.
- ➔ Collaboration – there's nothing like working with someone else to open your eyes to new possibilities.

References

AFL Website, viewed 30 July 2017, <http://www.afl.com.au/>

AFL Fantasy Website, viewed 30 July 2017 <https://fantasy.afl.com.au/>

DT Talk Website, Calvin’s Captains Articles, viewed 15 August 2017
<http://dreamteamtalk.com/category/2017/weekly-features/calvins-captains/>

Lutz, Roman 2015, 'Fantasy Football Prediction', ARXIV, viewed 28 July 2017,
<https://arxiv.org/abs/1505.06918>

Gorman, B 2017, ‘A Kaggle master explains gradient boosting’, No Free Hunch blog, 15 August 2017,
<http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/>