Final Report

Who do I start? - How can past performance help predict future performance in NFL fantasy football

By Rory Breslin (with thanks to Springboard mentor Max Sopp)

1. Problem Statement

It is estimated that each year there are over 40 million people playing NFL fantasy football[1]. And for each week of the NFL season (except Week 17), these players have to make a decision on who to start in their team. That equates to nearly 640 million team decisions being made each year. As anyone who has played fantasy football can attest to, some decisions on who to play in your team are easy, while others are head-wrecking. Who you decide on can vary from week to week and often your reason for picking a player can change too – he had a more fantasy points week last week, he's playing against a terrible defence, he's at home with a better offense. With so much information to digest and can be easy to be overwhelmed.

The aim of this project is to create a model that would give fantasy owners a greater understanding of what statistics are most important when selecting their line ups and help predict which player is going to perform best in a given week.

2. Data Wrangling

The data used for the project contained information on NFL players' performance for the 2015-2019 seasons. Overall, this created a dataset of 23,715 rows and 40 columns. These columns contained an array of information relating to a player's team, opposition, date & time, game conditions, game score, and performance statistics for passing, rushing and receiving.
It was clear that the number of columns would need to be reduced eventually but we would leave this until the exploratory analysis stage when I would have a better understanding of what information was most useful to keep. Therefore, the initial focus of this section was reviewing the columns to ensure the data was in a format that was useful for the analysis. This led to changes in the columns related:

- Position – creating an 'OTH' group for positions with low counts
- Player Names – editing to remove team name from this field
- Team Names – reviewing and deciding to keep team name rather than abbreviated names
- Game Score – splitting into two separate columns for team and opposition

Lastly, I calculate fantasy points based on the scoring system in the fantasy league of interest. Overall these changes made the dataset wider – increasing to 60 columns.

[1] 'Fantasy football is a billion-dollar pastime. Covid-19 is wreaking havoc with it'. AJ Willingham. https://edition.cnn.com/2020/12/05/us/fantasy-football-coronavirus-challenges-trnd/index.html

3. Exploratory Data Analysis

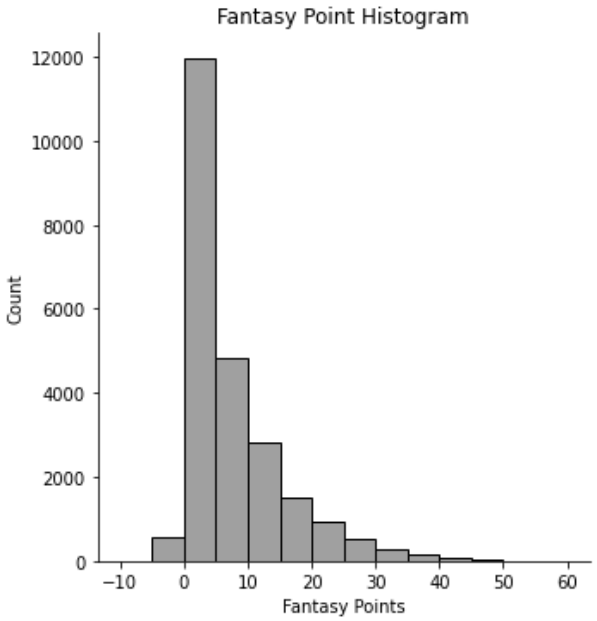
The target of this section was to gain a better understanding of our target variable, Fantasy Points. To do this, the section was broken into a number of sub-sections:

- Fantasy Points – looking at the distribution and outliers of this column
- Categorical Variables – looking at how counts and fantasy points differ by each categorical variable to see if this provides any insight
- Player and Game statistics – transformation of player and game statistics to allow them represent information that is known to a fantasy owner prior to kickoff
- Correlation Analysis – looking at the correlation between fantasy points and all those variables still remaining in dataset
- Clustering – apply an unsupervised cluster analysis on our dataset to see if it can uncover any trends that we were unable to identify in the above sections

a. Fantasy Points

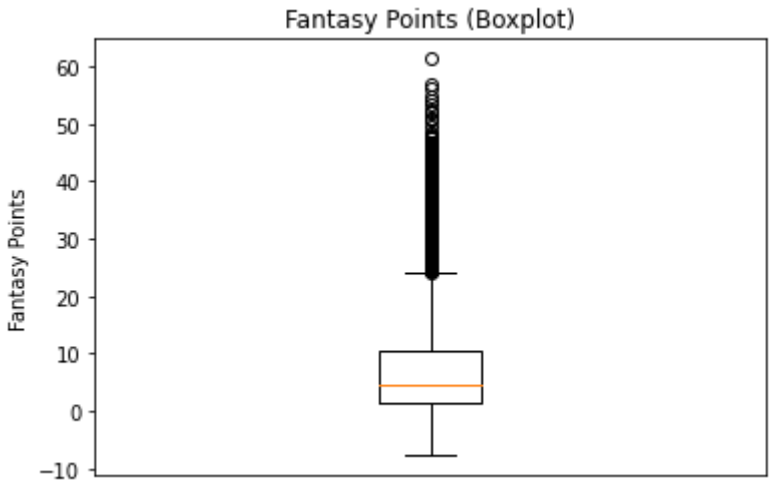
The first step of the EDA journey is to look at our target variable to get an understanding on the key metrics and distribution of this data. To start, we generated a histogram:

Figure 1: Histogram of Fantasy Points



This indicates that there are a lot of relatively low values, with 73% of data under 10 fantasy points. The data is skewed and has a long right tail, indicating outliers on the upper end. To further investigate this, I had a look at a boxplot of the data:

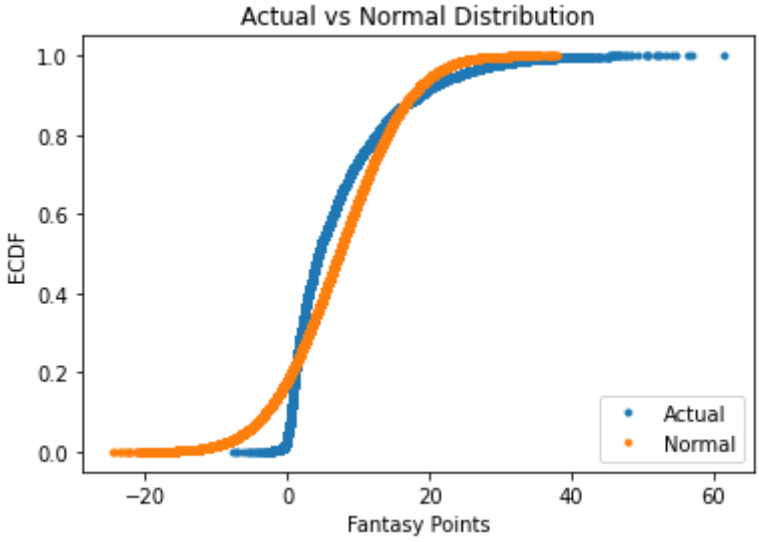
Figure 2: Boxplot of Fantasy Points



This further highlights that there are a large amount of outliers at the upper end of the dataset. At this point, I'm not of the view that we do not want to remove these outlier as these are the data points we're most interesting (i.e. we are want to know who is about to have on these high scoring fantasy weeks). We are less interested in the large number of low values in the dataset and we may need to be account for these later on.

The above so far does not indicate a Normal Distribution but it is important to test this. First, I created an Empirical Cumulative Distribution Function (ECDF) to the data and then generated a Normal Distribution ECDF using the mean (7.33) and standard deviation (7.94) of the fantasy points data. This graph (below) seems to show that the data does not follow a Normal Distribution.

Figure 3: ECDF of Fantasy Points vs Normal Distribution



Lastly, we tested the null hypothesis being of the data are normally distributed – with the lower the p-value in the result of this test, the more likely the data are to be non-normal. For our Fantasy Points data we received a p-value of 0 meaning – that regardless of what significance level we pick – the conclusion is that the data is not normally distributed.

b. Categorical Variables

There are a number of categorical fields in the dataset that may influence how many fantasy points a player will score. In this section, we looked at them individually to see if anything stands outs.

i. Position

Each player in the dataset is classified to a position with wide receiver (WR) being the most prominent (with 39% of the records) followed by running back (RB), tight end (TE), quarterback (QB) and other (being less than 0.1% of records).

Fantasy point scoring differs greatly by position, with QB being the highest scoring position – averaging 16.1 points per game – and Other being the lowest averaging 1.4 points. The distribution seen earlier in the fantasy points column continues when broken out by position group, with a long right tail across each. However, this is less pronounced for the QB distribution.

Figure 4: Average Fantasy Points by Position Group

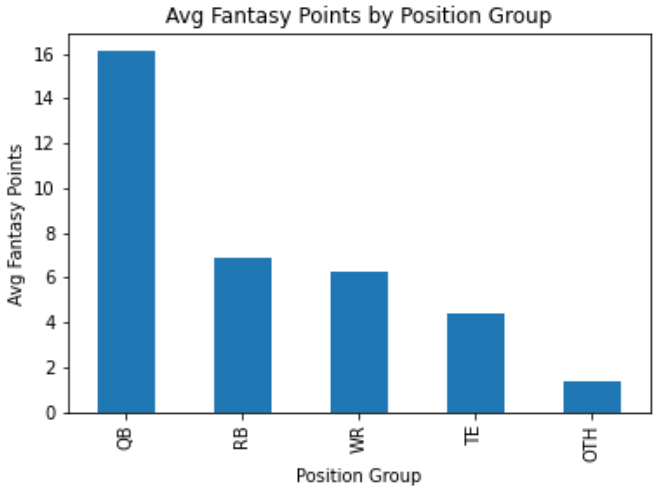
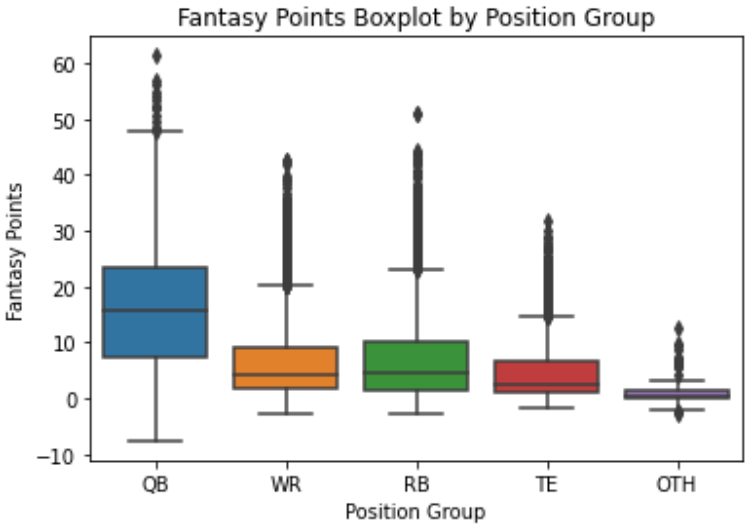


Figure 5: Fantasy Points Boxplot by Position Group



ii. Player Name

There are 1,012 unique players in the dataset. The counts and average fantasy points vary largely across each player. The highest performing player in the dataset is Patrick Mahomes (QB, Chiefs) who averages 28.9 points, followed by Andrew Luck (QB, Colts) and Deshaun Watson (QB, Texans) who both average 24.1 points. The lowest average is -2 points across several players.

iii. Team and Coaches

There are 32 teams in the NFL who all have a Head Coach (HC), Offensive Coordinator (OC) and Defensive Coordinator (DC). This dataset provides information on these for both the player's team as well as the opposition. Reviewing how each of these performed in relation to fantasy points showed that certain teams and coaches tend to have players who have better fantasy performances (i.e. New Orleans Saints and their HC Sean Payton have high average fantasy points for their players, whereas players playing against the Minnesota Vikings and their HC Mike Zimmer tend to score less on average).

While this is interesting and seems to show that which team and coach you play for or against is important, there are two issues that this type of data might present:

1. High dimensionality for modelling: To use this data for modelling purposes would mean creating dummy variables for n-1 unique values. Across the eight columns in the dataset, this would translate to an additional 458 columns to the dataset.
2. Duplication of information: Teams who have higher average fantasy points will also lead to the Head Coaches and Offensive Coordinators for those teams having higher fantasy points. While coaches and coordinators can move from team to team, there does seem to be a duplication of information that could be avoided.

iv. Game Setting

NFL games are played in a variety of locations and stadiums across the USA (and globe at times). For each of these games, there's a:

- Home team and an Away team
- Dome or open air stadium
- Grass or turf for the playing surface

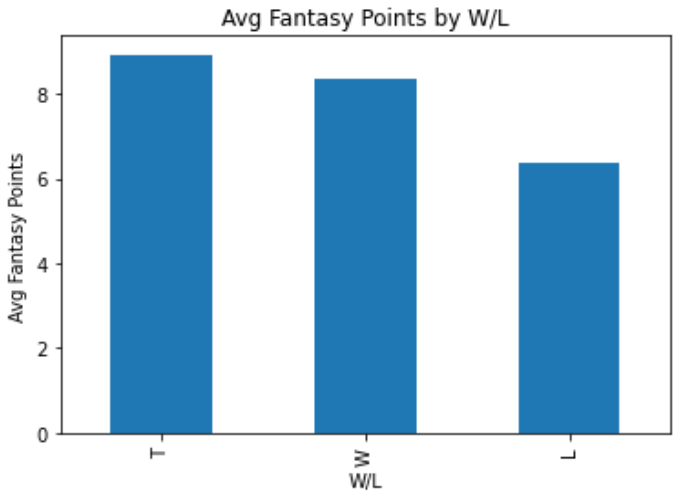
For each of these areas, there was a slight increase in average fantasy performance for:

- Players at home (7.6 vs 7.1 average fantasy points)
- Players in a dome (7.6 vs 7.3 average fantasy points)
- Players on turf (7.6 vs 7.2 average fantasy points)

v. Game Result

Players on average score more fantasy points in a win (8.35) or tie (8.91) than they do in a loss (6.39). However, ties are not common in the NFL with only 89 records in the dataset (0.3%).

Figure 6: Average Fantasy Points by Game Result



vi. Time & Date

There are a number of columns related to time & date in the dataset including:

- Time of Day
- Day of Week
- Week
- Month
- Season
- Date

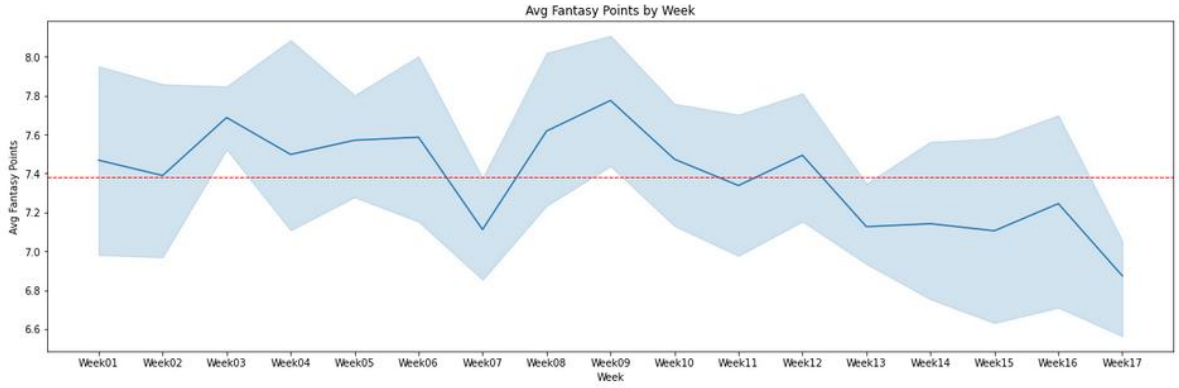
Counts across Time of Day are mainly for Noon, Afternoon and Night – but there are small number for games played in London and Mexico (321 records or 1.4%). Similar for Day of Week, the majority of records are for Sunday (19,928 or 84%) with remaining across Thursday, Saturday and Monday. For Week, Month and Season, there is a relatively even distribution across the various different unique values (17 weeks, 5 months, 5 seasons). Lastly, there are 248 unique date fields in the dataset.

Looking at Fantasy performance across these various attributes, there are many drastic differences between them. However, two pieces that do stand out are:

a) Decline in fantasy performance at end of season
   - Average fantasy points are lower in December and January then in September/October/November.
   - This trend is seen on a weekly basis with Weeks 13 onwards performing below average (see graph below)

b) Players fantasy performance dipped in 2017
   - Average fantasy points dropped to 6.9 points compared to other season which were 7.4 points or higher (on average)

Figure 7: Average Fantasy Points by Week



c. Player and Game Statistics

When looking at player and game statistics, there are two main issues encountered with regards to how it can be used to help us predict fantasy performance:

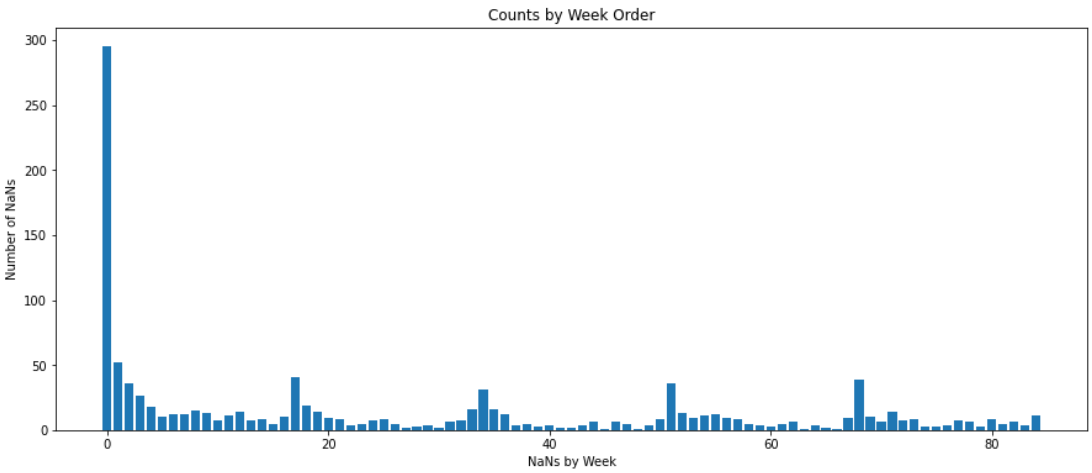
1. **Not independent:** Fantasy statistics were derived from raw player statistics, meaning they are not independent of each other
2. **Information not available at decision time:** Fantasy owners don't have access to the score of a game or how player performs until after the game – when it is too late

Rectifying this means transforming the data so that the information in the dataset is similar to what fantasy owner would have access at time of making decision on their line-up.

To do this, I developed a rolling average for the previous 4-weeks for all player and game statistics.

The creation of these rolling averages for previous means that the dataset now contains NaN values - 28,338 cells or 2.7% of the dataset. This is due to the process not being able to calculate an average for a player if no data existed yet for that player in the entire dataset (i.e. player playing his first NFL game). This is most common in first week of dataset where we had no player data to create averages from.

Figure 9: NaN values by Week

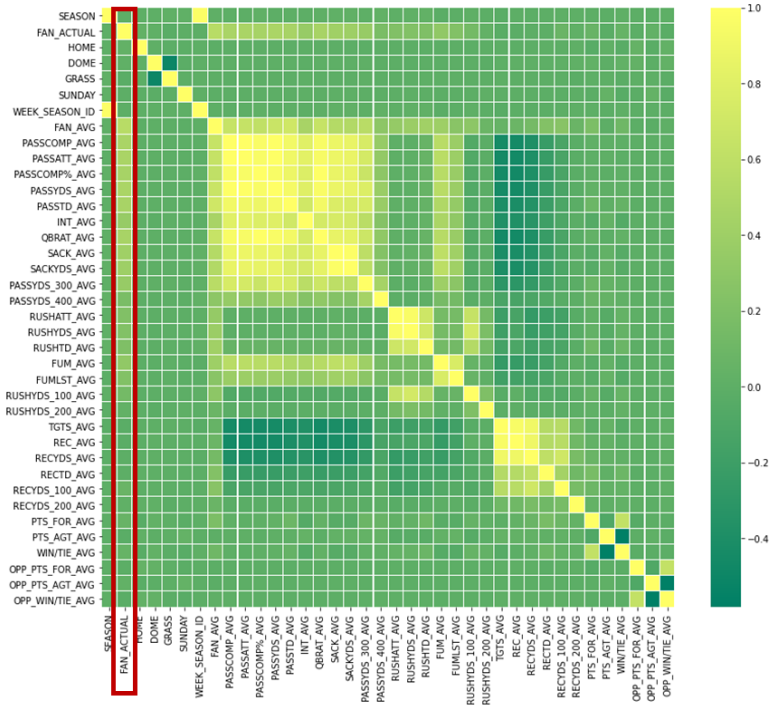


How we handle these values will be dealt with later in this documentation.

d. Correlations

In the last section, we transformed our player and game statistics to reflect performance over the previous for weeks. Now, we want to see if these statistics have any relationship with fantasy points.

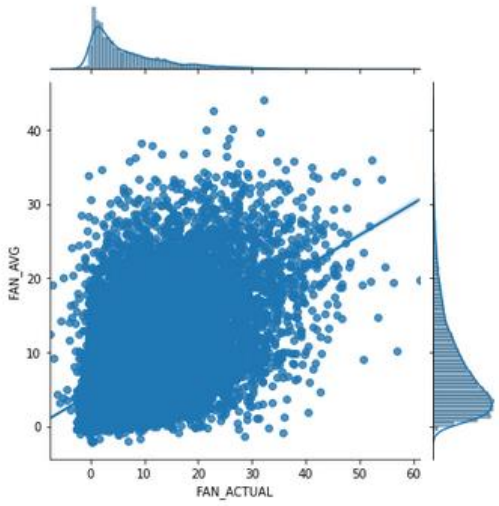
Figure 10: Correlation Heatmap of Numeric Data



The correlation heat map shows that fantasy performance over the previous 4 weeks seem to have the strongest correlation with how a player is going to perform this week. This relationship has a Pearson correlation coefficient of 0.56.

The joint plot below shows the linear relationship between the two variables as well showing that both have similar distributions with right tails.

Figure 11: Fantasy Points vs Average Fantasy Points (4-week prior) pair plot



i. Permutation Test

However, this observed correlation may just be by chance and a player's average fantasy performance over the previous 4-weeks may be completely independent of their performance that week. To see if this is the case or not, I needed to test the null hypothesis that these values are independent of each other.

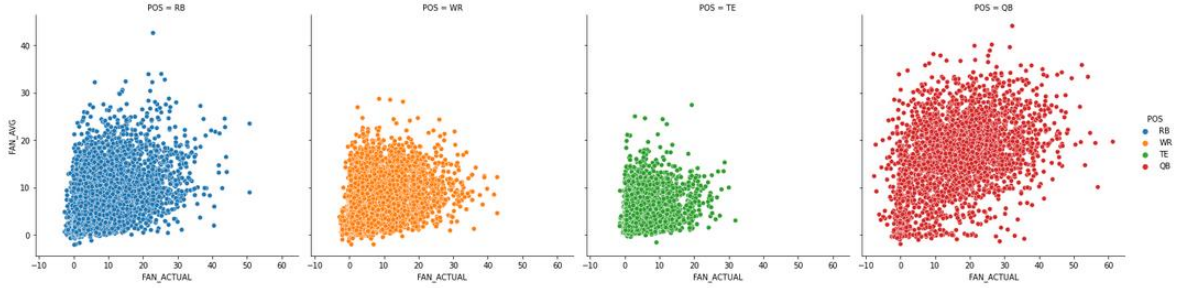
Since the data of Fantasy Points is not normally distributed, I used a non-parametric test – a ***Permutation test.*** This test permuted the fantasy 4-week average values and left the actual fantasy point total in a given week fixed. For each permutation, it calculated the correlation coefficient and assess how many of the permutation replicates have a Pearson correlation greater than the observed one.

In the 10,000 permutations ran, none of them generated a correlation coefficient higher than the one obtained in the actual dataset of 0.56. This gave us a p-value of 0.0 which allowed to reject the Null Hypothesis that Fantasy Performance in a given week is independent and uncorrelated with a player's average performance over the prior 4 weeks, at any significance level.

ii. Correlations by Position

Looking at the fantasy point relationship across other categorical variables produced a similar linear relationship to the one outlined above. However, the categorical variable of most interest from our analysis earlier was the Position variable.

Figure 12: Fantasy Points vs Average Fantasy Points (4-week prior) scatter plot by Position



Above shows the linear relationship across each position group which is similar to the overall data. When we look at correlation of fantasy points and average fantasy points for previous 4-weeks, we get slightly lower Pearson correlation coefficients:

- QB: 0.43
- RB: 0.48
- WR: 0.35
- TE: 0.32

This led to the question that – although previous fantasy points was the most correlated feature for the overall dataset – was it the most correlated by position group? The table below shows that while previous fantasy performance seems to be the best indicator for QBs and RBs, fantasy points is not one of the top 3 most correlated features for WRs and TEs with Targets, Receptions and Receiving Yards being more correlated.

Table 1: Correlation Coefficient by Position

QB	RB	WR	TE
0.43 – Fantasy Points	0.48 – Fantasy Points	0.38 – Targets	0.37 - Targets
0.42 – Pass Yards	0.47 – Rushing Attempts	0.37 – Receiving Yards	0.37 – Receiving Yards
0.40 – Pass Completions	0.40 – Rushing Yards	0.37 – Receptions	0.36 – Receptions

e. Clusters

The last significant bit of analysis in the EDA section was looking to use unsupervised learning to gain additional insight from the data. For this, we focused on the raw player statistics for passing, rushing, receiving and fantasy (19 columns) to see if there was any grouping of data that we were unaware of that might help us in understanding the dataset better.

For this, I used a K-means algorithm and created a pipeline that included scaling the data before running the algorithm. For this algorithm to work optimally, we need to find the correct number of clusters (k) to included in the model. To do this, I completed the following:

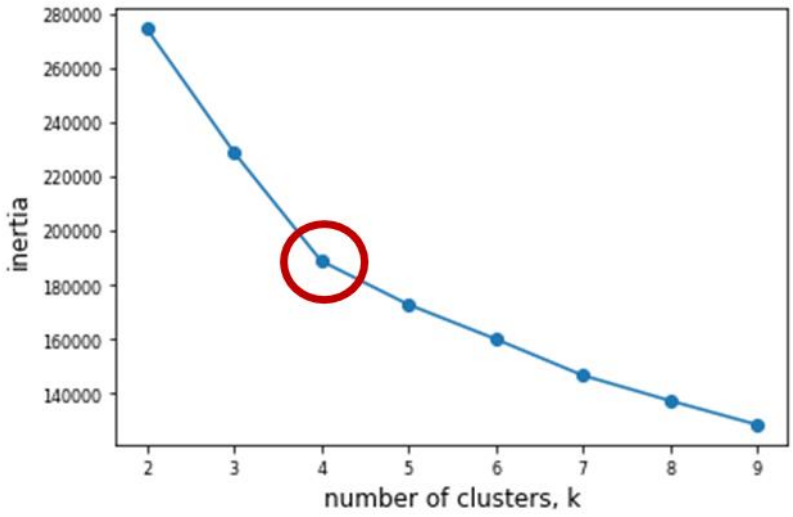
- Use the Elbow and Silhouette Methods to identify the optimal number of clusters
- Use PCA visualisation to see if this can show the clusters to use and whether they make sense
- Investigate the cluster vs our existing dataset to see how well it explains are main focus which is fantasy performance

Following this analysis, it became clear that the 4 cluster was the optimal number.

i. Elbow Method

The Elbow Method shows the sum-of-squares error for each cluster in a range of 2 to 10, with aim to identify the k that increasing the it does model the data much better (this is considered the elbow). The elbow for our data is when k=4 (see graph below).

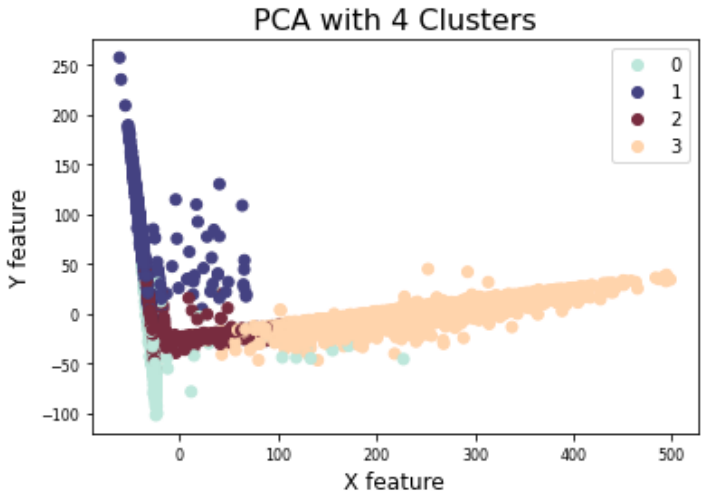
Figure 13: Model Performance by Cluster Size



ii. PCA Analysis

To get a better understanding of what the cluster looks like, it would be good to visualise it. This can be done using a PCA to reduce the dimensions of our dataset into our 2 dimension which we can then display in a scatter plot. This is being done now purely for visualisation purposes but PCA could potentially be useful later on in our analysis if we find that we have too many features and / or they are highly correlated.

Figure 14: PCA scatterplot of 4-cluster model by top 2 features



iii. Compare by Features

PCA is good to visualise but because this removes the features to create this visualisation, it can often be difficult to interpret. With that in mind, we looked at a number of different charts to see how fantasy points and overall counts vary by cluster and position.

Figure 15: Records count by Cluster

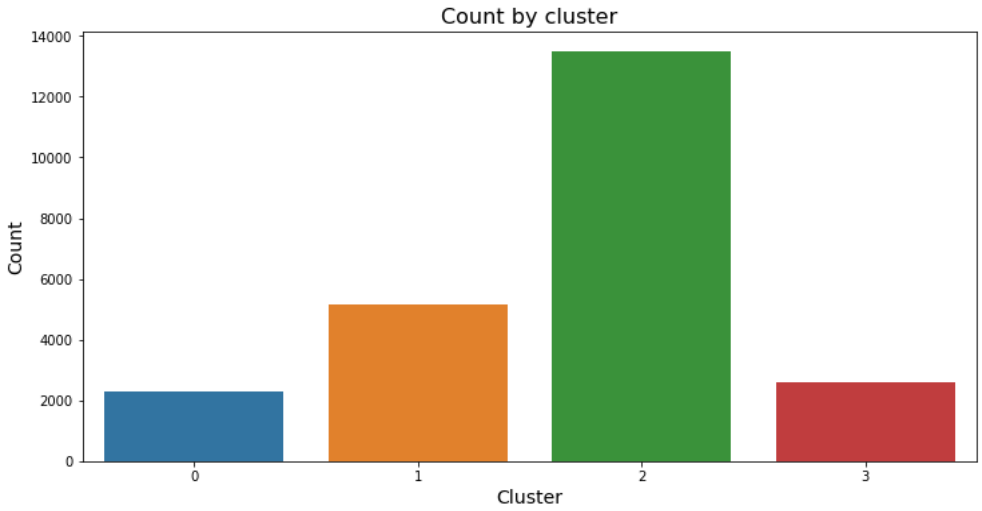


Figure 16: Average Fantasy Points by Cluster

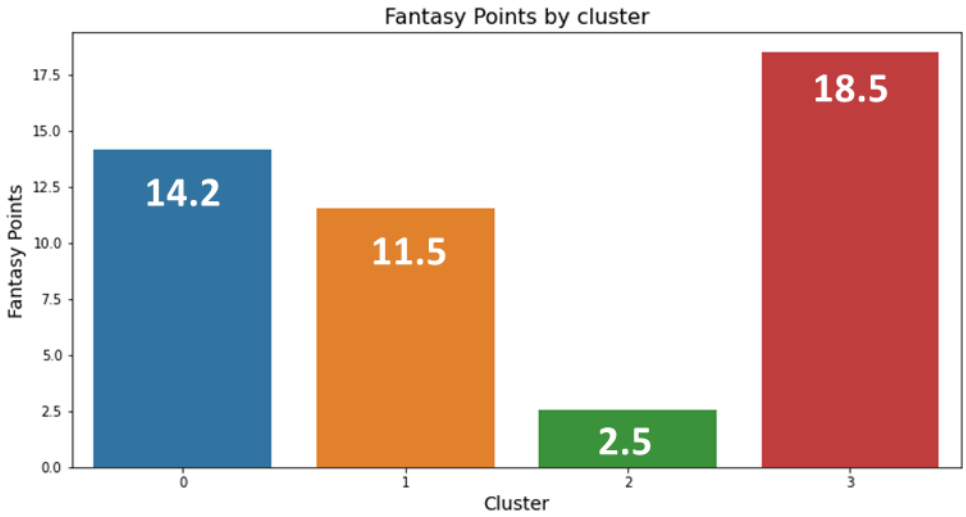
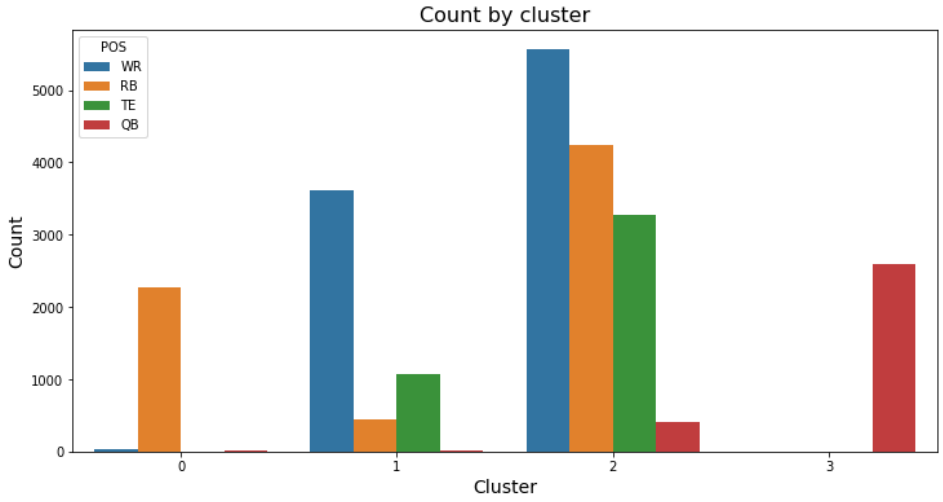


Figure 17: Records count by Cluster and Position



These graphs begin to paint a picture of what each cluster represents which I've looked to summarise below.

Table 2: Summary of Main Cluster Results and Findings

Cluster	Scoring	Count	Prominent Position	Main Statistics
0	2nd	4th	RB	Rushing
1	3rd	2nd	WR/TE	Receiving
2	4th	1st	All	None
3	1st	3rd	QB	Passing

These cluster are very interesting in that it separates out players who do not perform well into their own cluster and puts the remaining players into groups defined on whether they tend to accumulate Passing, Rushing or Receiving statistics.

Another side note is that the analysis did make some interesting classifications including assigning 7 QBs (Johnny Manziel, Jacoby Brissett, Marcus Mariota, Cam Newton, Josh Allen, Lamar Jackson, Taysom Hill) to the rushing cluster in certain games.

f. Tidy data

Following the above analysis, there were several changes that were applied to the dataset.

i. Drop Columns

The analysis above showed that certain columns in the original dataset are no longer required. For this reason, the following columns were dropped:

- 'PLAYER' column
    - Over 1,100 unique values means there are too many dummy variables to create to include in data for modelling
    - It is the statistics of the player, rather than the actual name that are likely to be the better predictor of future fantasy performance
- 'TEAM', 'HC', 'OC', 'DC', 'OPP', 'OPP HC', 'OPP OC', 'OPP DC' columns
    - Columns contain interesting information showing how certain teams and coaches produce better fantasy players. However, it is believed that this information can also be captured in the rolling average for points for / against and win rate (i.e. teams that score and win more will have better performing fantasy players). Therefore, to reduce dimensionality of dataset for modelling (remember including all would create 458 additional columns) and avoid duplication of information, we remove these columns and aim to use points for, points against, and win/tie rate as a proxy for this information
- 'WEEK' column
    - There looks to be a decline in fantasy performance toward the end of the season and this seems to be captured by both the Month and Week columns
    - As Month column has less unique values (5) than Week (17), the decision is to include Month column to attempt capture this trend
    - However, there is possibility for other analysis to use dummy variables for Weeks that may better capture this trend
- 'DATE' column
    - There are 248 unique fields and time element should be covered through Season and Month columns
- Raw player and game statistics columns
    - Following the creation of rolling averages for previous 4-weeks, the raw statistics in which they were calculated from are no longer required
- Fantasy points columns (except total)
    - These columns were used to calculate fantasy points during the Data Wrangling stage but no longer required as part of the analysis
    - The only exception is the raw fantasy points total, which we will keep as this is our target variable

ii. Drop Rows with NaN values

A number of options were considered including:

1. Drop values: In a real life situation, we would have no information on a player's previous performance (as they would have not played a game) and therefore have no information to project performance from.
2. Replace with zero: Assume that player not playing means he has not played and therefore his previous performance is 0.

3. Replace with estimate: Look to replace these values with an estimate (i.e. historic average performance of players in their first week playing)

Looking at data showed the issue is mainly in the first week of the dataset due to the creation of rolling averages but were cautious to make any assumptions about data (i.e. put to zero or use average) due to the impact this might have on the results of our modelling in the future.
Final decision was to drop all rows that contain a missing value. This will reduce the rows in the data set by 1,107 or 4.7%.
To enhance how we handle missing data with this data set in the future, I would need to run an analysis to understand the best assumptions for:

- First week of dataset (where we are assuming a player or team stats exist but not available in data set)
- All other weeks of dataset (where we are assuming a player is making his debut and no stats exist yet for player)

iii. Drop Other Rows

When reviewing Position column, decision was made to drop the 'Other' position type from dataset due to its low average fantasy points (1.39) and small number of rows (138).

iv. Create binary columns

There are a number of columns we want to keep in the dataset for but need to convert to make binary to support later modelling. This includes the following columns:

- Turn 'H/A' into 'HOME' binary column
  - 1 (when home) and 0 (when away)
- Turn 'ROOF' into 'DOME' binary column
  - 1 (when Dome stadium) and 0 (when Open stadium)
- Turn 'SURFACE' into 'GRASS' binary column
  - 1 (when Grass stadium) and 0 (when Turf stadium)
- Turn 'W/L' into 'WIN/TIE' binary column
  - Players in Wins and Ties had a similar and higher average fantasy points than those losses so makes to group these together – ties are too small to have separate dummy variable for them
  - 1 (when Win/Tie) and 0 (when Loss)
- Turn 'DAY' into 'SUNDAY' binary column
  - 84% of games are played on Sunday with only a small portion of players playing on Thursday, Saturday or Monday
  - 1 (when Sunday) and 0 (when Other)

v. Create dummy variables

There are a number of columns we want to keep in the dataset for but need to convert to dummy variables to support later modelling. This includes the following columns:

- 'TIME' column
  - First, we need to group London and Mexico into other distinct values in 'TIME' column – these account for only 321 or 1.4% of records and therefore does not make sense to create as separate dummy variables for them and increase dimensionality

    - As London games are usually played at 6pm GMT, the equivalent of 1pm ET, we will look to group London into 'Noon' group
    - Mexico games are played at night so we will group Mexico into the 'Night' group
    - Finally, we create dummy variables for:
        - TIME_Night
        - TIME_Day
    - Afternoon is held out to avoid multicollinearity
- 'SEASON' column
    - Create dummy variables for:
        - SEASON_2016
        - SEASON_2017
        - SEASON_2018
        - SEASON_2019
    - 2015 is held out to avoid multicollinearity
- 'MONTH' column
    - Create dummy variables for:
        - MONTH_January
        - MONTH_November
        - MONTH_October
        - MONTH_September
    - December is held out to avoid multicollinearity
- 'POSITION' column
    - Create dummy variables for:
        - POS_RB
        - POS_TE
        - POS_WR
    - QB is held out to avoid multicollinearity

These changes leave us with a dataset of 22,410 rows and 50 columns.

4. Pre-processing and Feature Engineering

Following our EDA, we now have a dataset of 22,410 rows and 50 columns. We now need to get this data into a shape to better support modelling.
Some of this was already done in the last section in regards to creating dummy variables for our categorical fields. In the coming sections, we will look to apply:

- Train / Test split
- Scaling
- Feature engineering

The aim of these steps is to provide a dataset that is ready to modelled.

a. Train / Test split

Our first step is to split our data between a training and testing sets. This will allow us to build our model on the training set and then evaluate it on the test set. The aim of this is to avoid us overfitting our model and allowing to generalise to new data when it becomes available – like a new season.

For this, we first need to confirm what our target variable is. Following our EDA, there are two potential options – Fantasy Points or Cluster groups. The decision on this will have an impact on the type of modelling we do later as Fantasy Points is a continuous variable (meaning a Regression problem) where as Clusters are categorical (meaning a Classification problem). Our focus to-date has been mainly on the **Fantasy Points** columns and therefore will use this as **our target variable (y)**. However, cluster grouping could provide interesting insight for future analysis.

The remaining 48 columns in the dataset will set as the features (X) for our model.

With our target variable and features confirmed, we split the data – 75% for training and 25% for test.

b. Scaling the data

Our feature data contains a wide variety of values and scales that will need to standardized before we model our data. We did this using the following steps:

1. Remove Dummy Variables
2. Standarise continuous variables (using StandardScaler in sklearn)
3. Concatenate Dummy variables back into scaled continuous data

This approach allows us to scale our continuous data varies significantly from feature to feature (i.e. Passing Yards has a range of 0-403 where as Fumbles Lost has range of 0-2) but keeps intact our dummy variables to remain at scale of 0 to 1.

c. Feature Elimination

We still have 48 features remaining to use for our model. This is still quite high and we would like to reduce the number of feature if possible. I used three techniques to reduce the number of features – remove low variance features, remove highly correlated features, and a recursive feature elimination model.

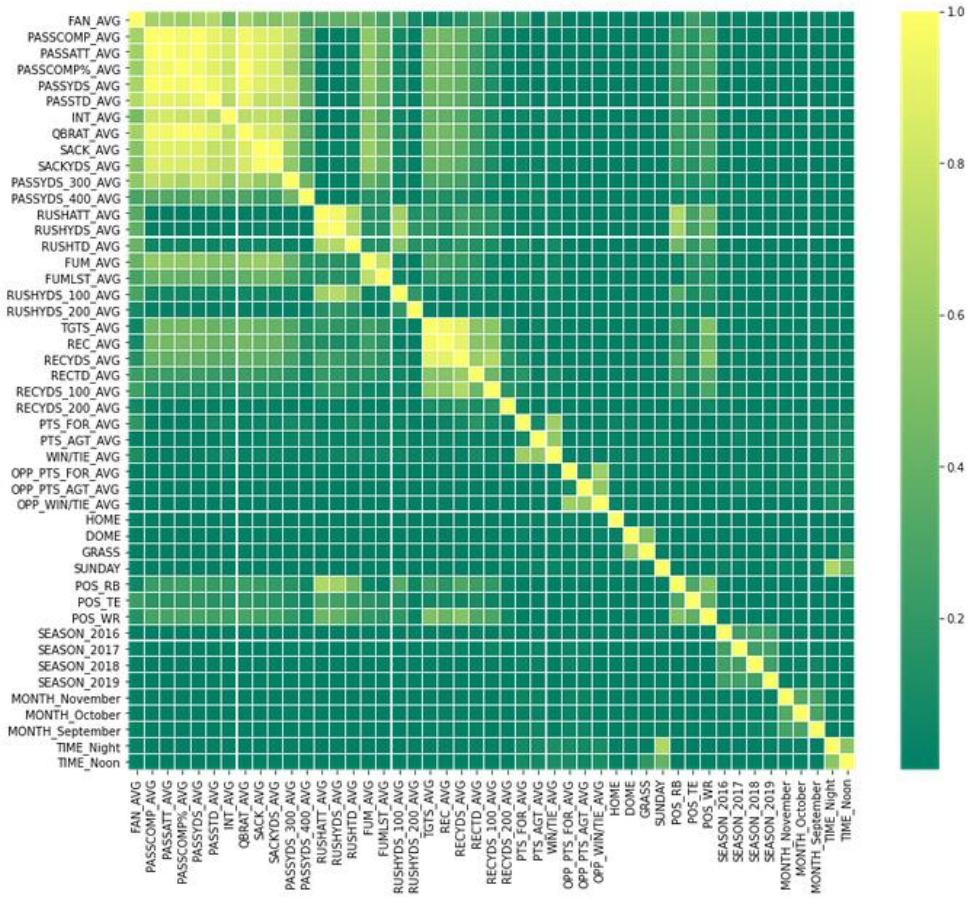
i. Remove Low Variance Features

- VarianceThreshold is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet some threshold.
- Applied a threshold of 0.1, which means we are looking for columns where approximately 90% of the values are similar.
- Identifies one feature to remove - 'MONTH_January'.

ii. Remove Highly Correlated Features

- If two or more than two features are mutually correlated, they convey redundant information to the model and hence only one of the correlated features should be retained to reduce the number of features[2].
- Looked to find variables which have a correlation (or pearson coefficient) greater than 0.8.
- Identifies 11 features to remove, reducing features to 36.

Figure 18: Correlation Heatmap of Features

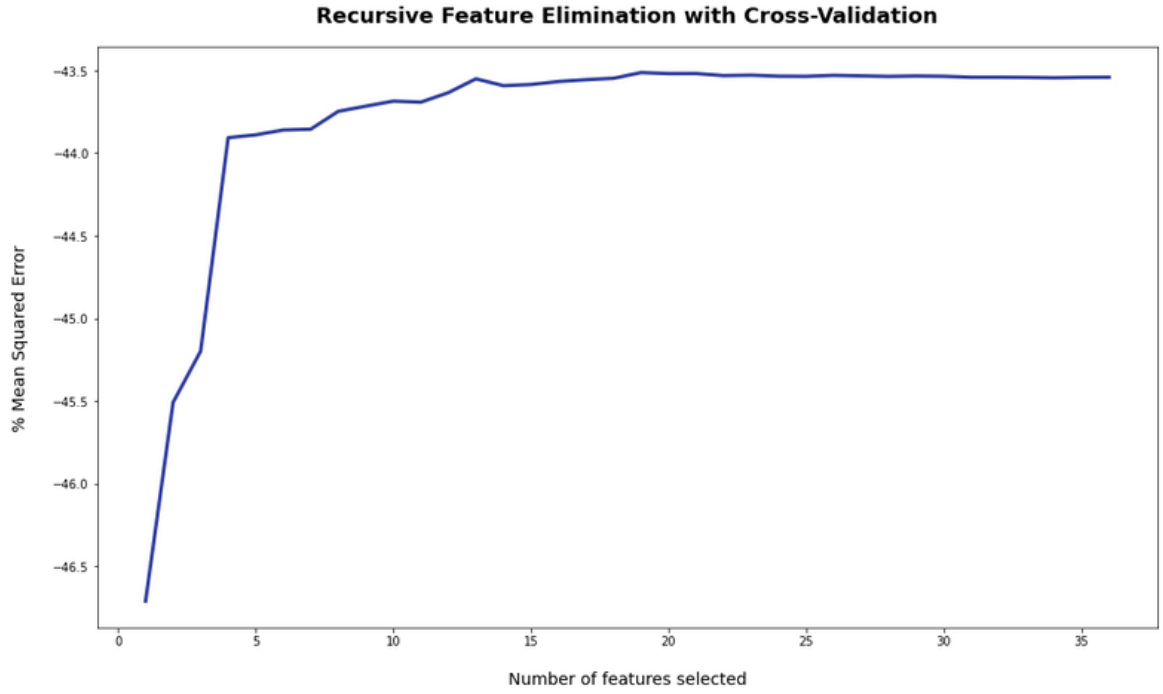


[2] 'Applying Filter Methods in Python for Feature Selection'. Usman Malik. https://stackabuse.com/applying-filter-methods-in-python-for-feature-selection/

iii. Recursive Feature Elimination

- To further highlight the most important features in our data for modelling, we applied a RFECV (Recursive Feature Elimination with Cross-Validation) model
- RFE (Recursive Feature Elimination) models look to rank features through assigning weights to features (e.g., the coefficients of a linear model) and then recursively consider smaller and smaller sets of features by dropping the lowest ranking features each time. This process is then repeated until the desired number of features is reached.[3]
- The applied a RFECV model applied to our feature data used a SVR estimator, had a "linear" kernel and reguralisation parameter of 1.
- The model identifies 19 features as being optimal number.

Figure 19: Recursive Feature Elimination (with CV) performance by number of features



These three steps allowed us to reduce our numbers of features from 48 to 19. This along with deciding on our target variable (fantasy points) has given us the following data to support our modelling:

Target Variable

- FAN_ACTUAL

Feature Variables

1. FAN_AVG
2. PASSCOMP_AVG
3. PASSYDS_300_AVG
4. RUSHATT_AVG
5. RUSHTD_AVG

[3] 'sklearn.feature_selection.RFE'. Sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html#sklearn.feature_selection.RFE

6. FUMLST_AVG
7. TGTS_AVG
8. RECTD_AVG
9. WIN/TIE_AVG
10. OPP_PTS_AGT_AVG
11. HOME
12. DOME
13. GRASS
14. SUNDAY
15. POS_TE
16. POS_WR
17. MONTH_September
18. TIME_Night
19. TIME_Noon

5. Modelling

Now, it is time to apply machine learning models to the data that has been crafted in the previous sections. As mentioned earlier, our target variable (Fantasy Points) is continuous and making this a regression problem. With this in mind, we will be focusing our modelling on four different types of regression models:

- Linear Regression
- SGD Regressor
- ElasticNet
- Random Forest Regressor

For each of these models, we looked to tune their hyper-parameters to identify the optimal model. Following this, we will evaluate the models on the training and test data using a three metrics – Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and R-squared.

a. The Models

i. Linear Model

Our first model to apply to the data was an Ordinary least squares (OLS) regression. This is a linear model that is fitted by minimizing the sum of the squares in the difference between the observed and predicted values of the dependent variable configured as a straight line.[4]

This model is fitted to the data and ready to be evaluated. Unlike the other models we will look, the OLS estimator does not have many hyperparameters to tune and therefore not considered at this time.

ii. SGD Regressor

The next model applied to the data was a Stochastic Gradient Descent (SGD) Regressor. This is a linear model that is fitted by minimizing a regularized empirical loss with SGD - the gradient of the loss is estimated one sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate).[5]

Initially, we set this model up using default values for the hyperparameters but then look to tune these hyperparameters using RandomisedSearchCV. For this model, we look at five hyperparameters to tune:

- Loss function ('squared_loss')
- Regularisation penalty ('l1')
- Regularisation value or alpha (0.01)
- Learning rate type ('adaptive')
- Initial Learning Rate (0.01)

[4] 'Ordinary Least Squares Regression'. https://www.encyclopedia.com/social-sciences/applied-and-social-sciences-magazines/ordinary-least-squares-regression

[5] 'sklearn.linear_model.SGDRegressor'. Sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html?highlight=sgd%20regressor#sklearn.linear_model.SGDRegressor

For each hyperparameter, 3 to 8 options were included in the process and this identified as the best parameter fore each (in bracket above).

This model (with optimal hyperparameters) is then fitted to our training data to be evaluated.

iii. ElasticNet

ElasticNet is a linear regression model trained with both l1 and l2 -norm regularization of the coefficients. This combination allows for learning a sparse model where few of the weights are non-zero like Lasso, while still maintaining the regularization properties of Ridge. Elastic-net is useful when there are multiple features which are correlated with one another.[6]
Initially, we set this model up using default values for the hyperparameters but then look to tune these hyperparameters using ElasticNetCV. For this model, we look at two hyperparameters to tune:

- L1 ratio (1)
- Regularisation value or alpha (0.01)

We selected a range of seven values (0.1 to 1) for the L1 ratio and range of eight values (0.0001 to 1) for regularisation value / alpha to find the best combination for modelling on our data. This identified an L1 ratio of 1 and an alpha of 0.01 as the best values.
This model (with optimal hyperparameters) is then fitted to our training data to be evaluated.

iv. Random Forest Regressor

The last model applied to the data was a random forest regression model. This is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.[7]

Initially, we set this model up using default values for the hyperparameters but then look to tune these hyperparameters using GridSearchCV. For this model, we look at three hyperparameters to tune:

- Number of estimators (1,000)
- Depth of decision trees (10)
- Number of features per decision tree ('sqrt')

Applying this show that the best results are found using 1,000 estimators at a max depth of 10 and using square root of the total number of features in dataset per decision tree.

This model (with optimal hyperparameters) is then fitted to our training data to be evaluated.

[6] '1.1 Linear Models'. Sklearn. https://scikit-learn.org/stable/modules/linear_model.html#elastic-net
[7] 'sklearn.ensemble.RandomForestRegressor'.Sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html?highlight=randomforestregressor#sklearn.ensemble.RandomForestRegressor

b. Evaluation

For evaluation of our models, we will look at three metrics:

1. **Root mean squared error (RMSE)**
    - This represents the sample standard deviation of difference predicted values and observed values[8]
2. **Mean absolute error (MAE)**
    - This represents the average absolute difference between the predicted values and the observed values
3. **R-Squared (R2)**
    - This shows how well features in model explain the variability in our target variable (fantasy points)

For each model, we look to calculate these metrics on the training data, training data (using cross-validation) and the test data.

i. Training Data

First we applied our models to our training data to see how well it performed at predicting fantasy points versus the observed values in the data. The table below shows the performance by model and metric.

Table 3: Training Data Performance by Model and Metric

Model	RMSE	MAE	R-Squared
Linear	6.39	4.65	0.364
SGD Regressor	6.39	4.66	0.363
ElasticNet	6.39	4.65	0.363
Random Forest	5.66	4.24	0.501

Random Forest Regression model performed best of all models when applied to all training data with the lowest RMSE and MAE (indicating it had the lowest prediction error) and the highest R-squared (meaning it explained the degree of variability in fantasy point performance).

ii. Training Data (with cross-validation)

The previous section evaluated performance on the training data from which each model was fitted on. This approach could be prone to overfitting of the model meaning better metrics might not translate or generalise well to new data. To test this out before we apply the models to our test data, we applied a 5-fold cross-validation of model performance on the training data for each metric – the results of which are in the table below.

Table 4: Training Data (with CV) Performance by Model and Metric

Model	RMSE	MAE	R-Squared
Linear	6.40	4.66	0.361
SGD Regressor	6.41	4.66	0.358
ElasticNet	6.68	5.03	0.304
Random Forest	6.65	4.90	0.310

[8] 'Choosing the Right Metric for Evaluating Machine Learning Models – Part 1'. Alvira Swalin. https://www.kdnuggets.com/2018/04/right-metric-evaluating-machine-learning-models-1.html

Applying the cross-validation shows that the Random Forest and ElasticNet models do not perform as well as previously, with much higher metrics across the board. However, there is very little change to the Linear and SGD models metrics and now perform better relative to other models with the lowest RMSE, lowest MAE and the highest R-squared. This seems to show that the Random Forest model in particular may overfit, whereas the Linear and SGD regressor are more biased with less variance.

iii. Test Data

Lastly, we will apply are models to our test data. This data has been held out until this point to provide us with a set of clean data that we could use for final evaluation of our model. The results of this are in the table below.

Table 5: Test Data Performance by Model and Metric

Model	RMSE	MAE	R-Squared
Linear	6.45	4.69	0.350
SGD Regressor	6.45	4.69	0.350
ElasticNet	6.45	4.69	0.350
Random Forest	6.44	4.71	0.350

The results above show us that there is very little to differentiate models. Random Forest model performed best in relation to RMSE while the other models perform better in relation to MAE. R-squared consistent across all models.

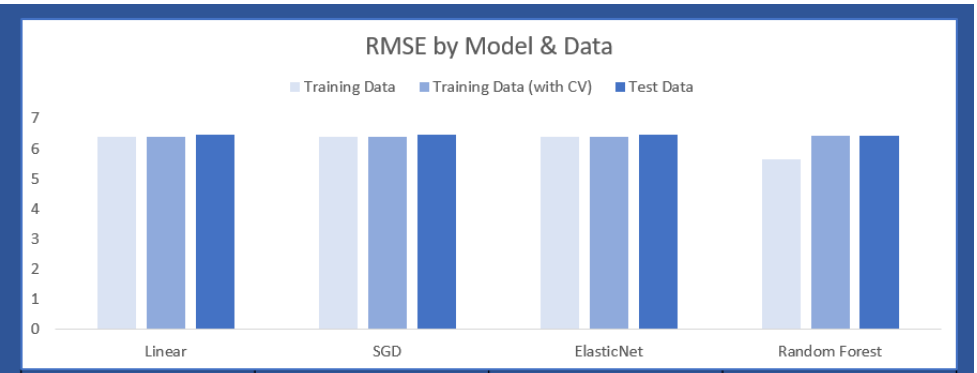
c. Conclusions

Our choice of model depends on which metric is deemed the best for the our dataset – as the results are inconclusive.

Considering that RMSE will penalise higher differences between predicted and actual values – and this something we want to minimise when predicting fantasy points – this is the metric we will concentrate on for our model decision. This would means we would select the Random Forest as our final model.

However, it is important to note, that due to how close the metrics are this does not mean this is the clear cut best model.

Figure 20: RMSE Performance by Model and Data Type



	Linear	SGD	ElasticNet	Random Forest
Training	6.39	6.39	6.39	5.66
Training (with CV)*	6.39	6.40	6.40	6.44
Test	6.45	6.45	6.45	6.44

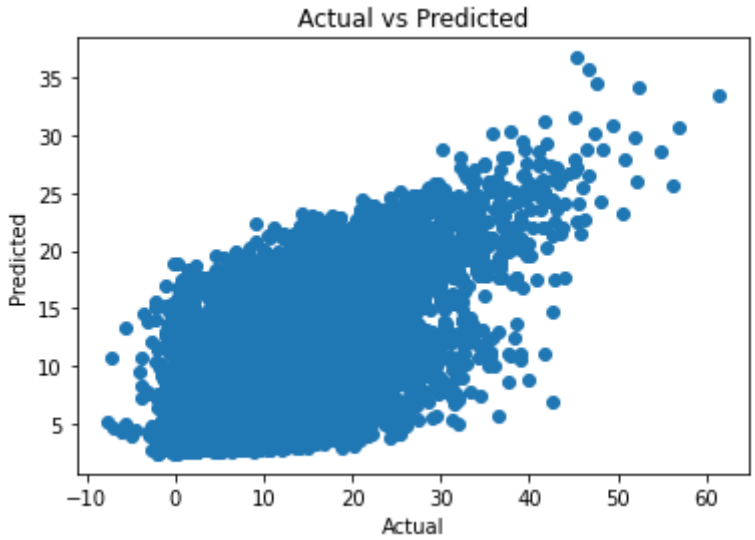
Focusing on the results of Random Forest model, this provided us the following metrics:

- RMSE = 6.44
- MAE = 4.71
- R-squared = 0.35

These indicate that on average the predictions from our model will error by +/- 4.71 fantasy points (based on MAE) and that the model explains 35% of the variability of given player's fantasy performance. The scatter plot below shows how predicted fantasy points differed from actual observed fantasy points.

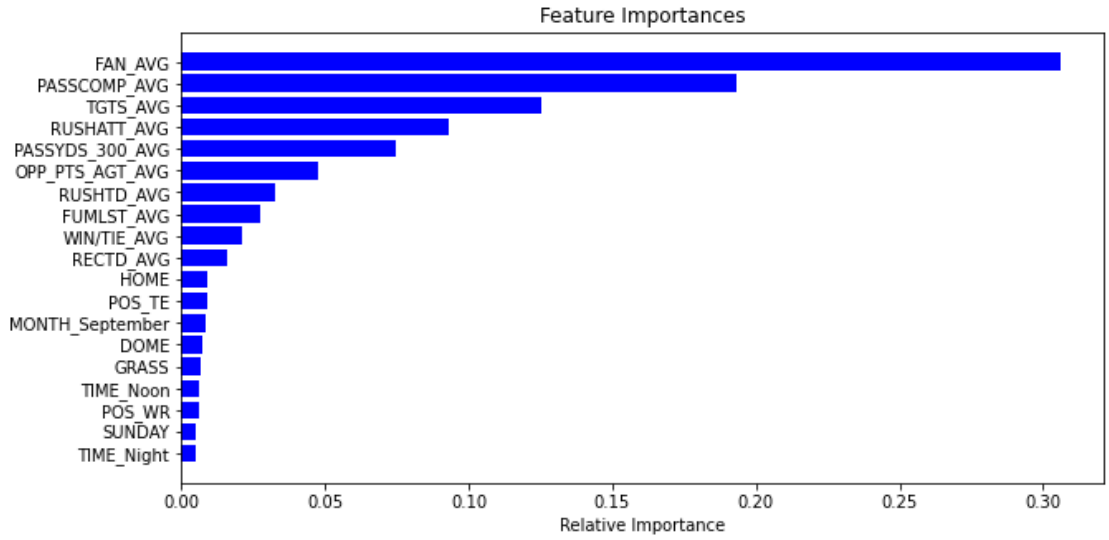
While a good starting point, there is still plenty of room for improvement in the model to reduce the error rate and generate more accurate predictions, as one point can make a massive difference in a fantasy football game.

Figure 21: Actual vs Predicted Fantasy Points for Random Forest Model



The model is also able to identify which of the features in the model are most significant – as shown in the graph below.

Figure 22: Feature Importance for Random Forest Model



As we had expected when we did our EDA, previous fantasy performance (over 4-weeks) is the most important feature in the dataset. If nothing else and you are unsure who to start in a given week in fantasy football, picking the player with the highest 4-week average fantasy performance would be a good rule of thumb.

Following this, we see the next three most important features are Pass Completions, Targets and Rushing Attempts. These make sense as they are important statistics for specific position group or cluster that were identified earlier the analysis – Passes for QBs, Targets for WRs / TEs, Rushing Attempts for RBs.

Lastly, a lot of the details about a game – be it the time of the game or the stadium type / surface – don't seem to be massive influences on a given player's fantasy performance.

6. Next Steps

This project and model is a good first step in attempting to understand and predict Fantasy Football performance. However, for future models and studies, the aim would be to reduce the error on the final model. This model currently gives a MAE of 4.71 which states that on average a prediction will +/- 4.71 the actual performance. While this can provide a good steer to fantasy owners on who to start in a given week, many fantasy players know that one point can be the difference between joy and despair on a given NFL weekend – so reducing this number to support fantasy decisions would be hugely beneficial.

In terms of how this analysis could be enhanced going forward there a number options that could be explored:

a. Explore more rolling averages

For this project, I wanted to explore how previous performance could help explain future performance. For this, I choose an arbitrary time period of 4 weeks as the previous to examine but this has two problems:

- It says that the performance in all weeks before this 4-week period do not matter – which we do not know for certain
- It states that each week in this 4-week period is weighted equally – which might not be the best way to weight performance

For future analysis, there is possibility to use different number of weeks and weights to see if there are better way to optimise these variables.

b. Better use of clusters

While I performed a cluster analysis on the data the yielded interesting results, I did not end up incorporating this into the final model. Going forward there are a number of ways that this could be factored into the dataset:

- Classification problem:
    - Instead of using fantasy points as the target variable, use the clusters as the target variable to identify if a player is correctly identified into the right group
    - This option gives less precision in terms of identifying a player's exact score but this should allow you to gain certainty on whether the player you are selecting is going to be a viable starting player or not
- Add cluster to rolling average
    - Instead of using clusters as target variable, look to apply to previous performance to say the % of weeks a player was in each of the clusters (i.e. 50% was in Rushing group and 50% in low scoring group) to see if this is a better indicator of future performance than previous fantasy points
- Filter dataset
    - One cluster identified a group of players that are very low scoring
    - As our focus is on understanding which players will be high scoring fantasy players, removing these from dataset may lead to a more accurate model
    - This would impact on the number of records in the dataset significantly so this would have to be taken into consideration also

c. Position Focus

A lot of the EDA seemed to suggest that performance differs by position and that different metrics are important for different positions (i.e. Targets for WRs vs Rushing Attempts for RBs). Focusing on position level analysis may reduce the size of the dataset but could produce more meaningful mode and results.

d. New data

The data we had was for the 2015 to 2019 seasons. Expanding this could to include more seasons and player records if possible would add to the robustness of the modelling.

Also, looking to access new data sources that could provide new information to the model to better a player's performance. This could include information on a player's snap count, the performance of team QB, performance of team offensive line, or salary.

e. Different approach for handling NaNs

For this analysis, we discarded over +1,000 rows containing NaN values that were generated through the development of the 4-week rolling average metrics. To hold onto this information in the future, we could look to provide a default value for these missing values based on historic performance for players playing in their 1st week in dataset by position.

f. Different Feature Engineering

For our feature engineering, we focused on a Recursive Feature Elimination model to reduce the size of our dataset. In the future, it might make sense to do a Principal Component Analysis (PCA) to see if this leads to better modelling results. However, it must be noted that taking this approach would lead to less interpretability of the results of the model (i.e. we wouldn't be able to tell which feature, like historic fantasy performance, is the most important).

g. Different Models

While we applied a number of models in our analysis, there is always scope to add more. With this in mind we could look add in new models like Support Vector model to see how they perform.