EPL Data Analysis

SC1015 Mini Project

By A135 Grp 10:

Patel Dhairya Nayanbhai	U2221896A
Dominic Tay Jia Wen	U2223491J
Sum Yuan Sen	U2221346D

Table of Contents

Checkpoint 1: Introduction

Background on Topic

The English Premier League is the most-watched sports league in the world and is contested by 20 clubs which operates on a system of promotion and relegations.

As football evolves in the modern era where teams spent huge amount of money on player transfers, sponsorships etc. , it is clear that the stakes are high and teams are desperate to win.

Thus, this project aims to make use of the abundance of data that is ever present to us in today's technology-driven world to generate meaningful insights for teams to adopt.

Premier League

Our Objective

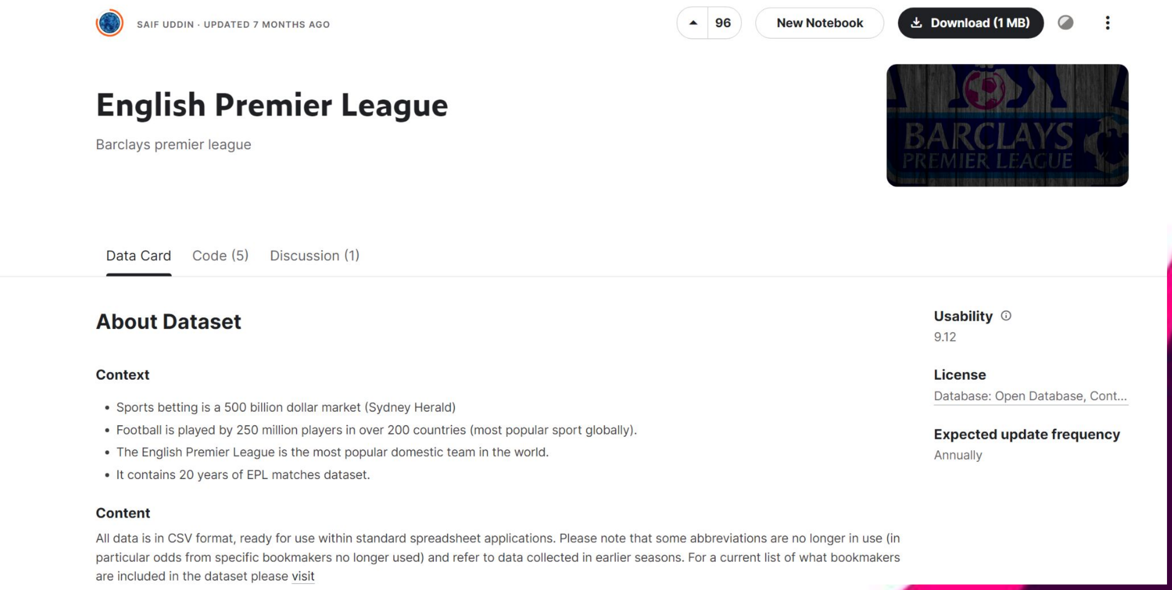
To investigate factors that will **maximise goals scored for a football team** in the *English Premier League*, using data from past 5 seasons of the EPL (***17/18*** to ***21/22***)

Premier League

Our Dataset

For our dataset, we use the ***Saif Uddin English Premier League Dataset*** from ***Kaggle***.

Seasons *17/18* to *22/23* were used.



Significance

1. Due to the competitive nature of the Premier League, teams need to **win** to get the maximum number of points for each match.

2. To win, team must **score more goals than opponent**.

Insights from data will help to generate strategies that maximise goals scored, help a team win and perform better in the long run.

Checkpoint 2: Data Preprocessing

Data Preparation & Cleaning

Our dataset started out with many miscellaneous data such as "*Referee*", "*Date*" etc.

Besides, other insignificant variables such as the red and yellow cards given out during a match are voided as well since its not within our scope of exploration.

As a result we streamline the data to that of home team related.

Before:

```
In [87]:  final.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1560 entries, 0 to 379
Data columns (total 22 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      1560 non-null   object
 1   Referee   1560 non-null   object
 2   HomeTeam  1560 non-null   object
 3   AwayTeam  1560 non-null   object
 4   FTHG      1560 non-null   int64
 5   FTAG      1560 non-null   int64
 6   FTR       1560 non-null   object
 7   HTHG      1560 non-null   int64
 8   HTAG      1560 non-null   int64
 9   HTR
 10  HS
 11  AS
 12  HST
 13  AST
 14  HC
 15  AC
 16  HF
 17  AF
 18  HR
 19  AR
 20  HY
 21  AY
dtypes: int6
memory usage
```

After:

```
In [102]:  HomeData.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1560 entries, 0 to 379
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   HS      1560 non-null   int64
 1   HST     1560 non-null   int64
 2   HF      1560 non-null   int64
 3   HC      1560 non-null   int64
 4   FTHG    1560 non-null   int64
 5   HTHG    1560 non-null   int64
dtypes: int64(6)
memory usage: 85.3 KB
```

Premier League

Exploratory Data Analysis

Exploring the data such as ***Shots, Shot on Target, Corners, Fouls*** that happened during a match in the *Premier League* and whether it affects the **number of goals scored.**

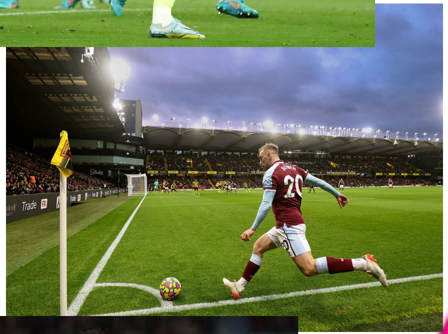
In [79]:

```
#Extracting variables that could affect home goals
HomeData = pd.DataFrame(final[["HS", "HST", "HF", "HC", "FTHG", "HTHG"]])
HomeData
```

Out[79]:

	HS	HST	HF	HC	FTHG	HTHG
0	27	10	9	9	4	2
1	6	2	6	3	0	0
2	19	6	16	8	2	0
3	14	4	7	12	0	0
4	9	4	13	6	1	1
...	...	...	...	...	...	...
375	6	3	12	3	1	1
376	12	6	10	3	4	0
377	29	8	6	5	3	1
378	24	5	5	13	3	0
379	9	0	13	3	0	0

1560 rows × 6 columns


Premier League

Exploratory Data Analysis

We noted that the data are all *Numeric* and *Univariate* in nature.

In [80]: `HomeData.describe()`

Out[80]:

	HS	HST	HF	HC	FTHG	HTHG
count	1560.000000	1560.000000	1560.000000	1560.000000	1560.000000	1560.000000
mean	13.523077	4.626282	10.491667	5.683974	1.474359	0.658974
std	5.656580	2.602501	3.429665	3.028960	1.305972	0.831768
min	1.000000	0.000000	0.000000	0.(		
25%	9.000000	3.000000	8.000000	3.(		
50%	13.000000	4.000000	10.000000	5.(		
75%	17.000000	6.000000	13.000000	8.(		
max	35.000000	15.000000	23.000000	18.(		

In [11]: `HomeData.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1560 entries, 0 to 379
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   HS      1560 non-null   int64
 1   HST     1560 non-null   int64
 2   HF      1560 non-null   int64
 3   HC      1560 non-null   int64
 4   FTHG    1560 non-null   int64
 5   HTHG    1560 non-null   int64
dtypes: int64(6)
memory usage: 85.3 KB
```

Premier League

Exploratory Data Analysis

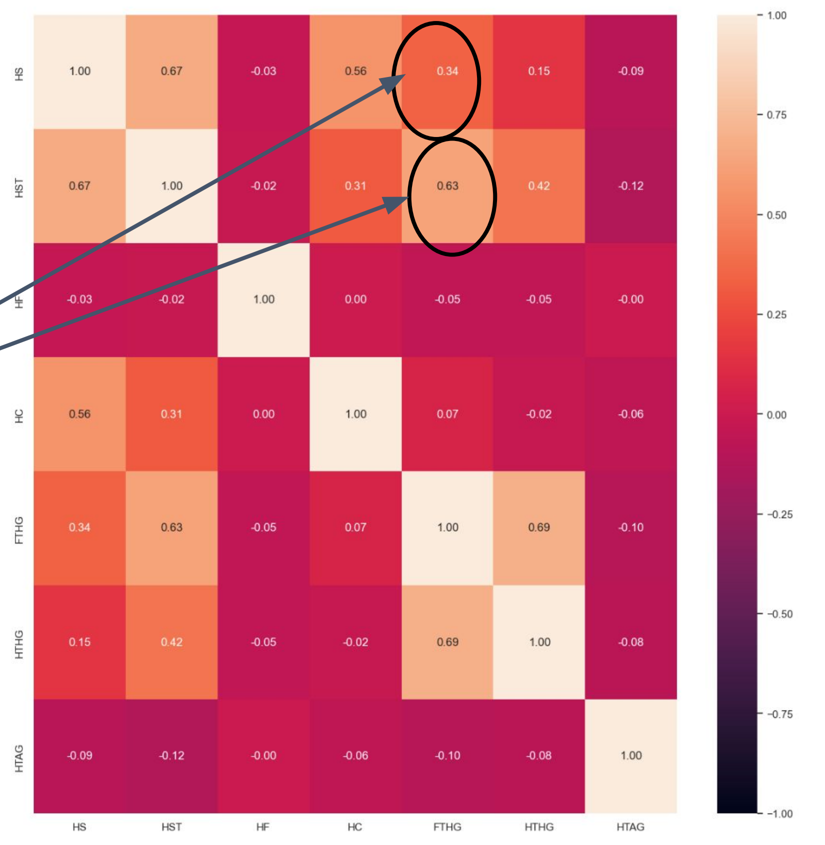
A ***Correlation Heatmap*** was then used to observed which variables stands out when correlating to **goals scored.**

As seen here the Variables **“HS” (R^2 = 0.34)** and **“HST” (R^2 = 0.64)** were the most correlated to **goals scored**

In [23]:

```
f, axes = plt.subplots(1, 1, figsize = (15,15))
sb.heatmap(HomeData.corr(), vmin = -1, vmax = 1, annot = True, fmt = ".2f")
```

Out[23]: <Axes: >



Exploratory Data Analysis

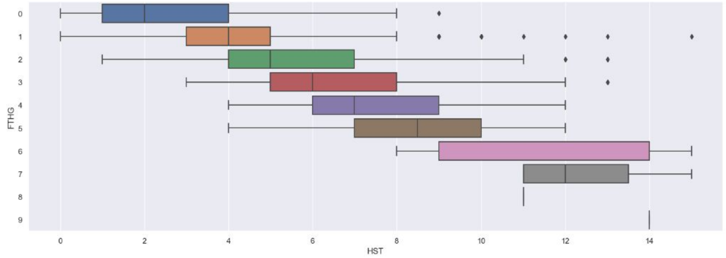
And since the variables “HS” and “HST” are univariate, we proceeded to use a ***Box & Whiskers Plot*** to have a better look on the distribution.

Home Shots on Target - “HST”:

In [25]:
```
HomeShotsOnTarget = final['HST']
jointDF2 = pd.concat([HomeShotsOnTarget, HomeGoals], axis = 1)
f = plt.figure(figsize=(18, 6))
sb.boxplot(x = "HST", y = "FTHG", data = jointDF2, orient = "h")
jointDF2.corr()
```

Out[25]:

	HST	FTHG
HST	1.00000	0.62818
FTHG	0.62818	1.00000

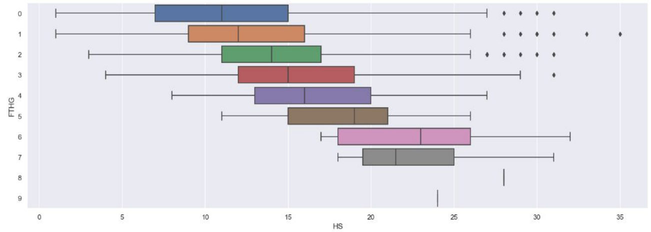


Home Shots - “HS”:

In [81]:
```
HomeShots = final['HS']
jointDF2 = pd.concat([HomeShots, HomeGoals], axis = 1)
f = plt.figure(figsize=(18, 6))
sb.boxplot(x = "HS", y = "FTHG", data = jointDF2, orient = "h")
jointDF2.corr()
```

Out[81]:

	HS	FTHG
HS	1.000000	0.341059
FTHG	0.341059	1.000000



Premier League

Checkpoint 3: Model Training

Model Selection

1. **Decision Tree (Baseline)**

2. **K-Nearest Neighbors**

3. **Random Forest (a max depth of 4 to prevent long time training and fitting)**

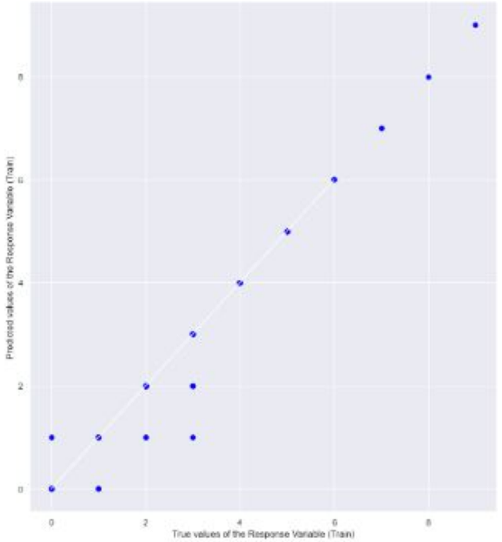
<u>How we train, fit and evaluate the models:</u>

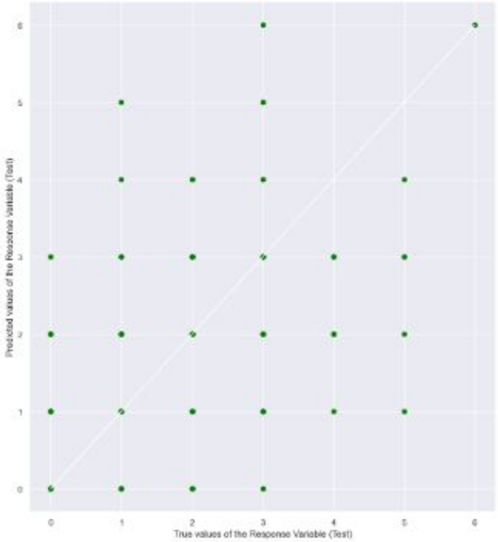
- We train and fit each of the model over 100 times to get a much more accurate average score value
- Values used for comparison is the accuracy score of the model on the test values

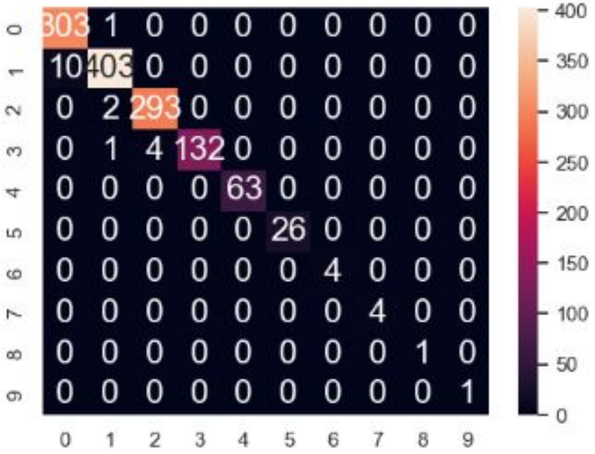
Sample result of Decision Tree

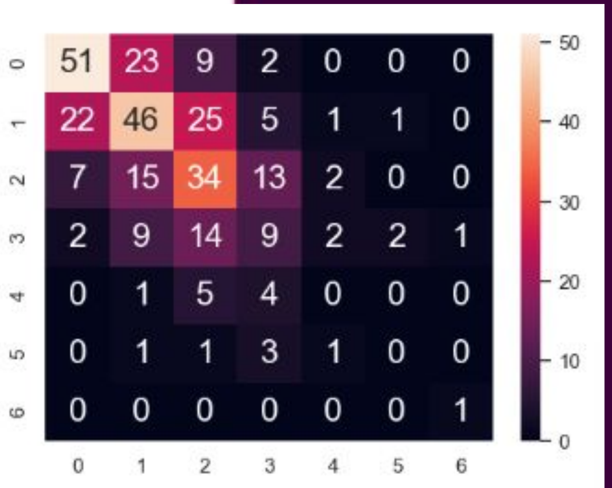
```
Decision Tree Classifier:
Goodness of Fit of Model        Train Dataset
Classification Accuracy         : 0.9855769230769231
Mean Squared Error (MSE)        : 0.016826923076923076

Goodness of Fit of Model        Test Dataset
Classification Accuracy         : 0.4519230769230769
Mean Squared Error (MSE)        : 1.2532051282051282
```







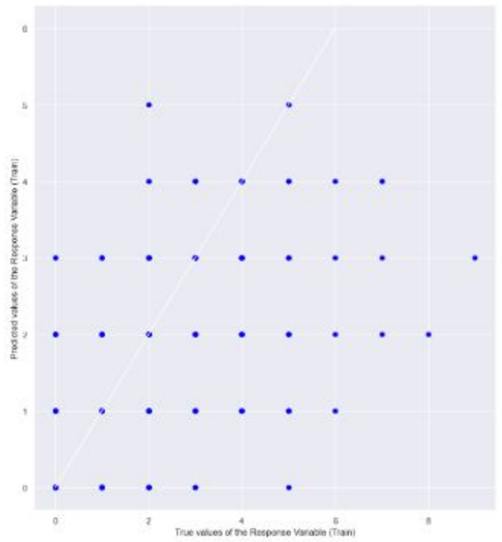


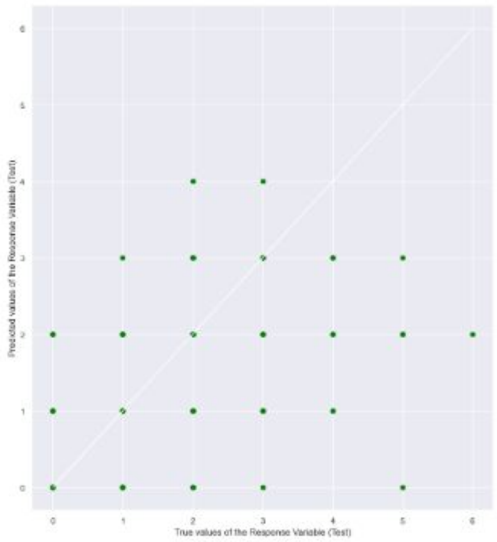
Premier League

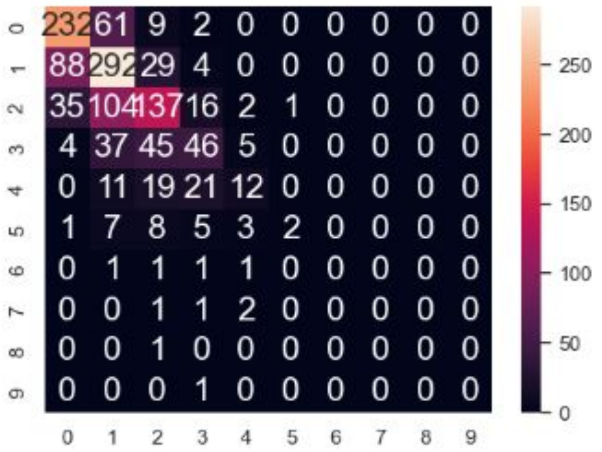
Sample result of K-Nearest Neighbors

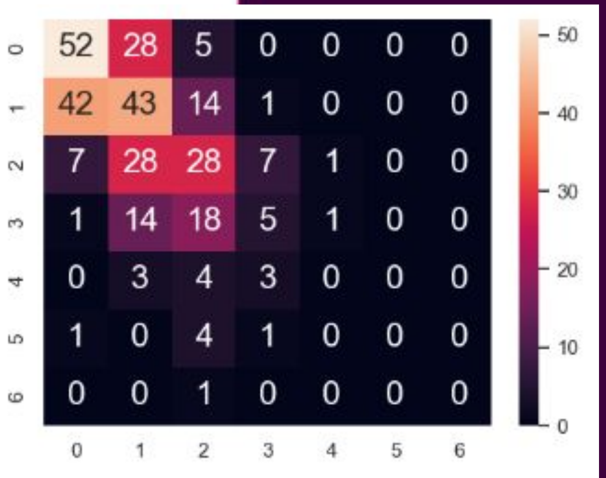
```
K-Nearest Neighbors Classifier:
Goodness of Fit of Model        Train Dataset
Classification Accuracy         : 0.5777243589743589
Mean Squared Error (MSE)        : 1.0993589743589745

Goodness of Fit of Model        Test Dataset
Classification Accuracy         : 0.41025641025641024
Mean Squared Error (MSE)        : 1.2371794871794872
```





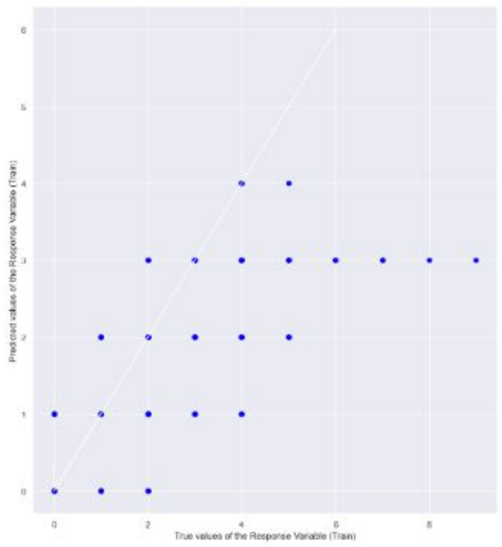


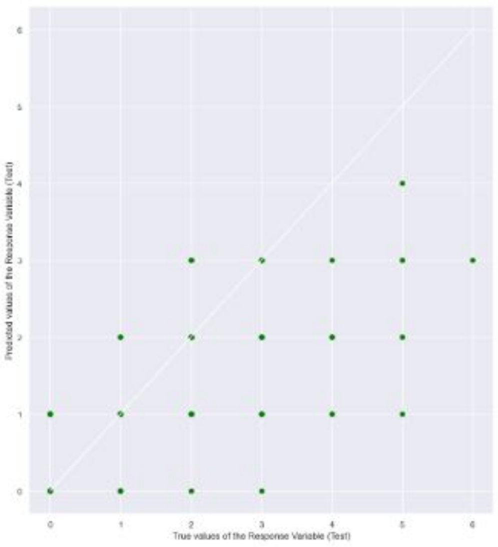


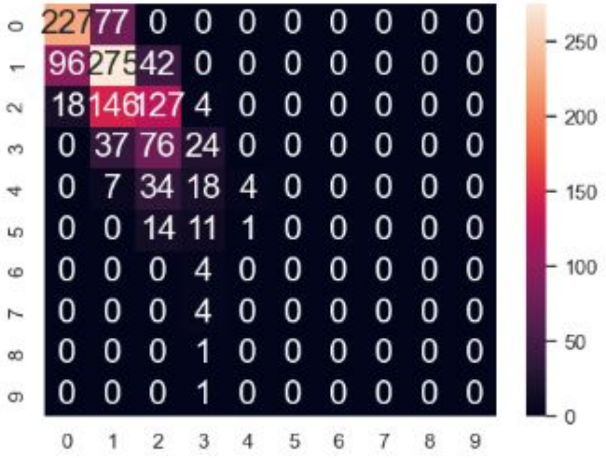
Sample result of Random Forest

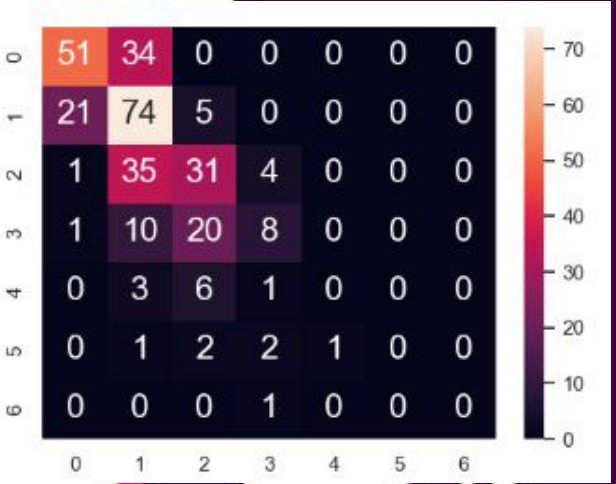
```
Random Forest Classifier:
Goodness of Fit of Model        Train Dataset
Classification Accuracy         : 0.5264423076923077
Mean Squared Error (MSE)        : 0.969551282051282

Goodness of Fit of Model        Test Dataset
Classification Accuracy         : 0.5256410256410257
Mean Squared Error (MSE)        : 0.8846153846153846
```









Results

```
Average of Linear Regression (over 100 times) Accuracy Score: 0.4134615384615385  MSE: 1.2759935897435897
Average of k-Nearest-Neighbors (over 100 times) Accuracy Score: 0.3941025641025641  MSE: 1.3841025641025644
Average of Random Forest (over 100 times) Accuracy Score: 0.4966025641025642  MSE: 0.9815384615384615
```

We chose the Random Forest model as it has the highest Accuracy Score and the lowest MSE.

Premier League

Checkpoint 4: Hyperparameter Tuning

How to tune the hyperparameter

Model used for tuning hyperparameter

- Grid Search from model selection.

Random Forest parameter to tune

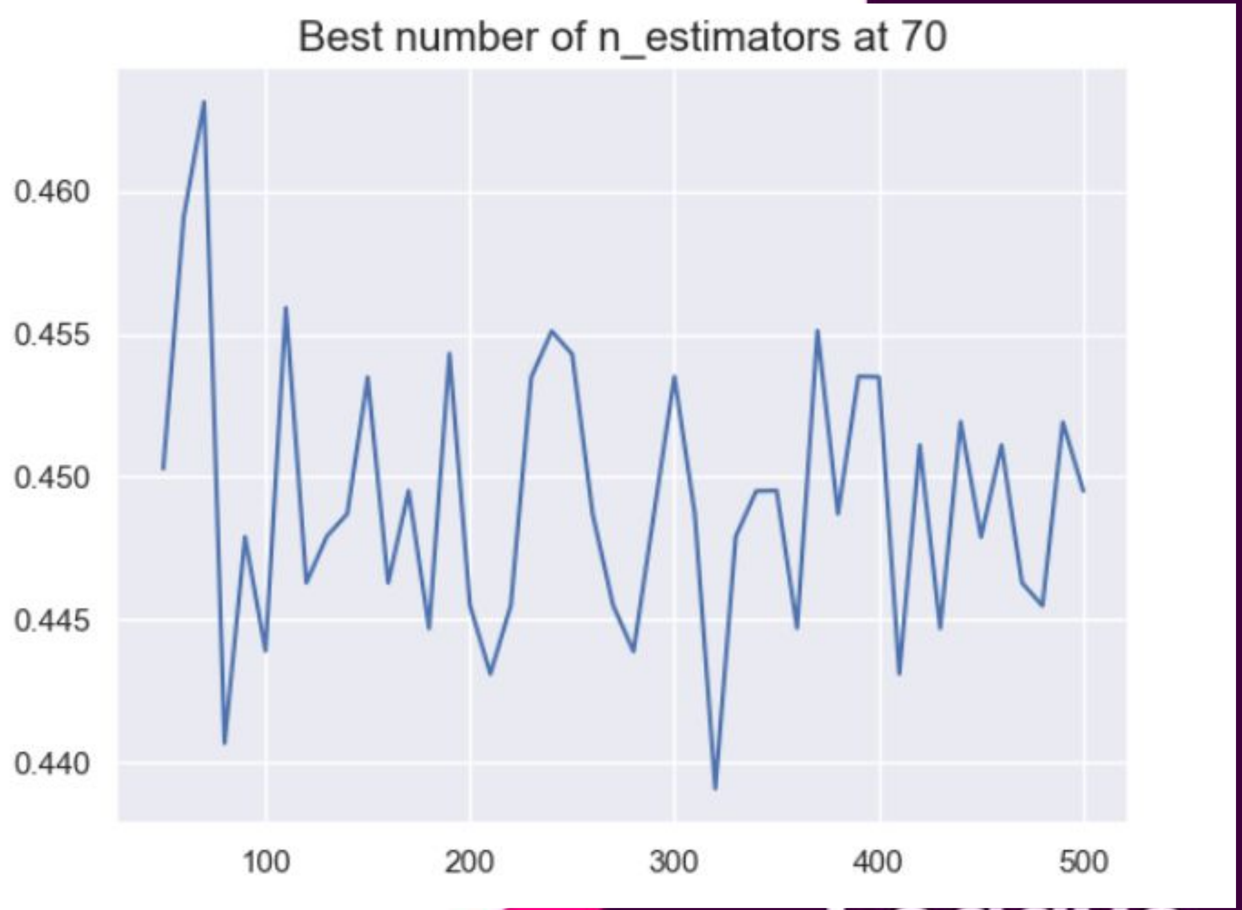
- n_estimators
- max_depth
- max_leaf_node

n_estimators parameter

For n_estimate the value range from 50 to 500 with increment of 10

- e.g 50,60,70… 490, 500

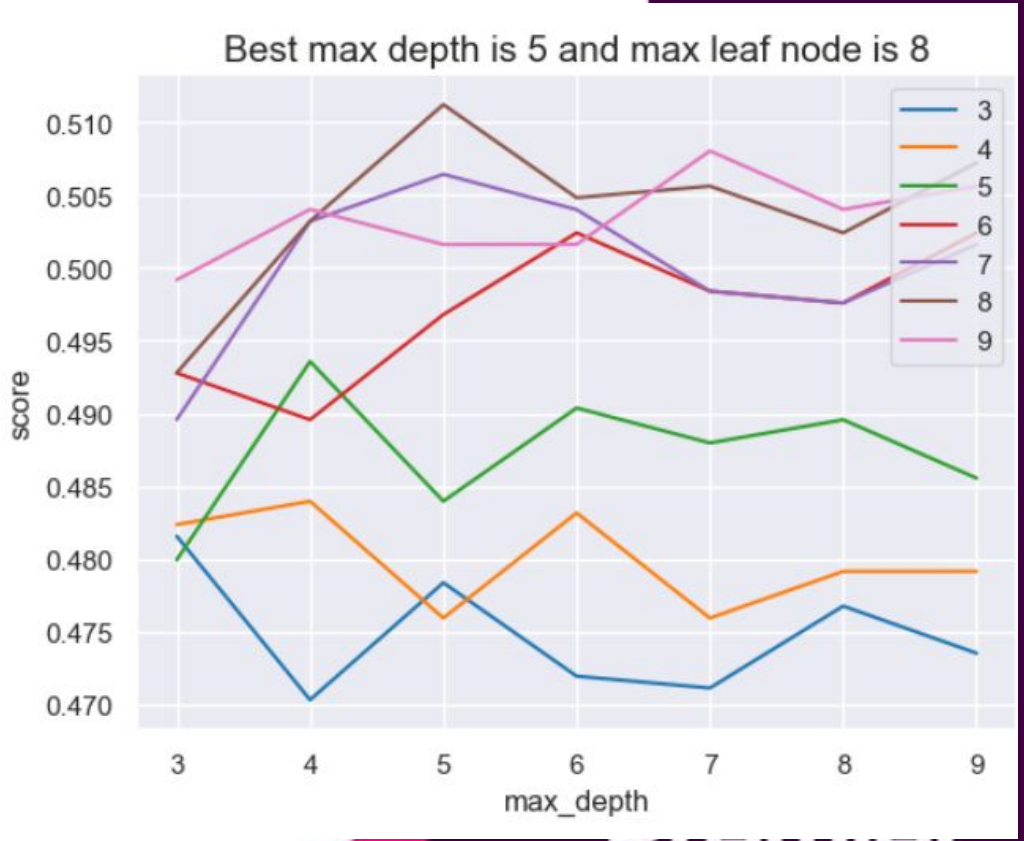
Sample result of the tuning.



max_depth and max_leaf_node parameter

- The value of n_estimator used will be the same as previous result found
- Both parameter will use the same range of 3 to 9

Same result of the tuning



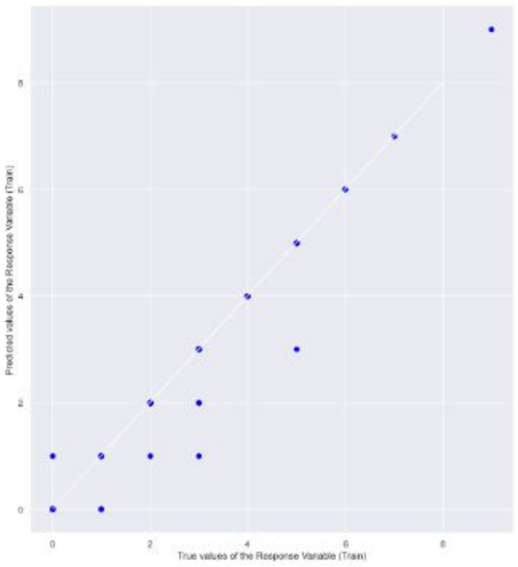
Tuning Result

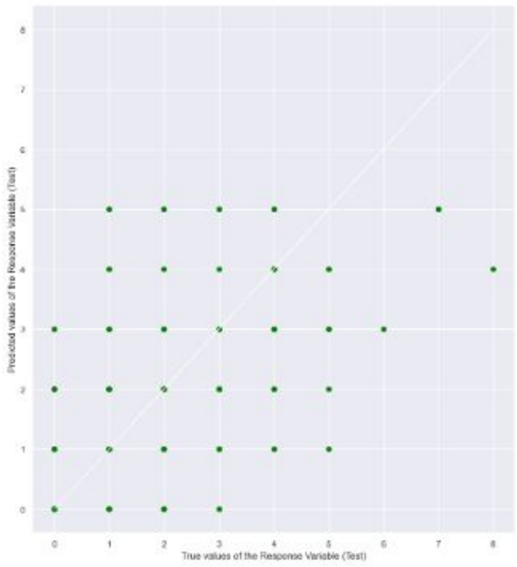
We train and fit the baseline model and the newly tuned random forest model over 100 times to get more accurate results

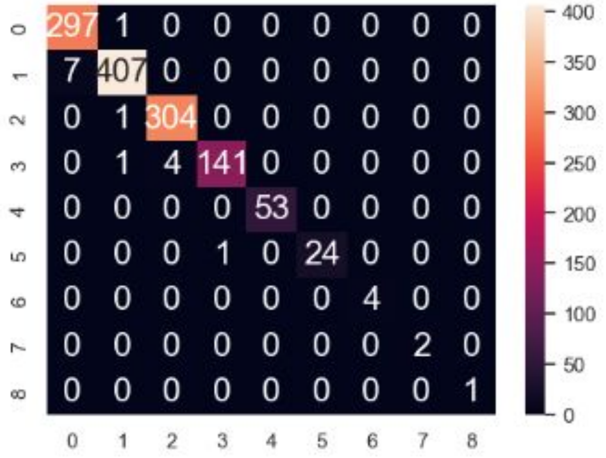
Sample from Decision Tree

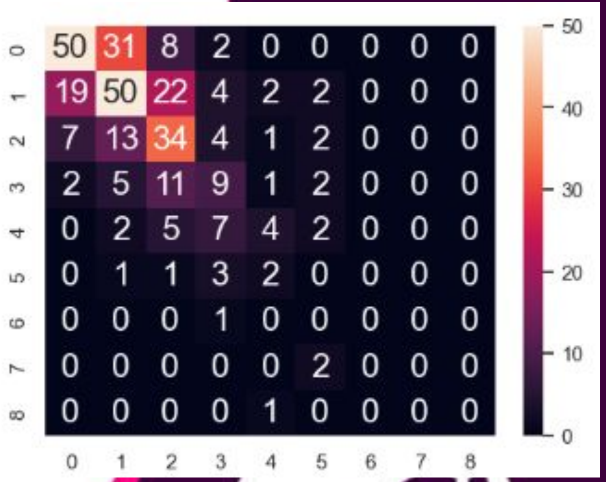
```
Decision Tree Classifier:
Goodness of Fit of Model        Train Dataset
Classification Accuracy         : 0.9879807692307693
Mean Squared Error (MSE)        : 0.016826923076923076

Goodness of Fit of Model        Test Dataset
Classification Accuracy         : 0.47115384615384615
Mean Squared Error (MSE)        : 1.3846153846153846
```





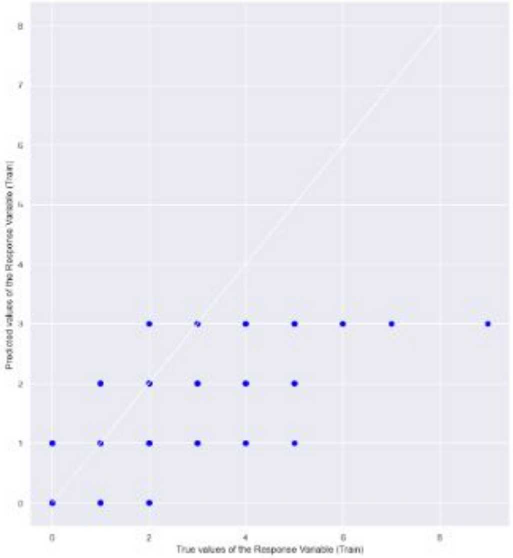


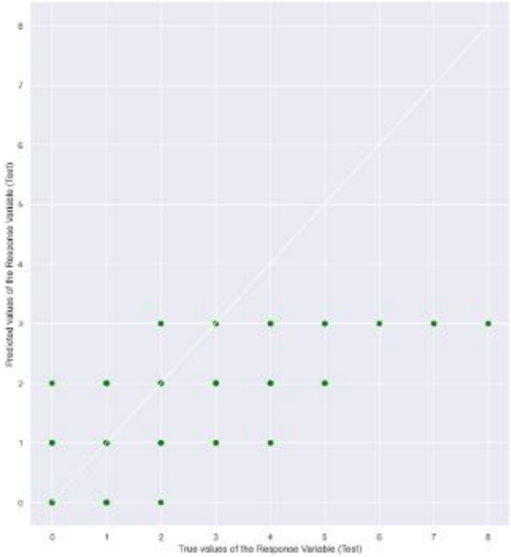


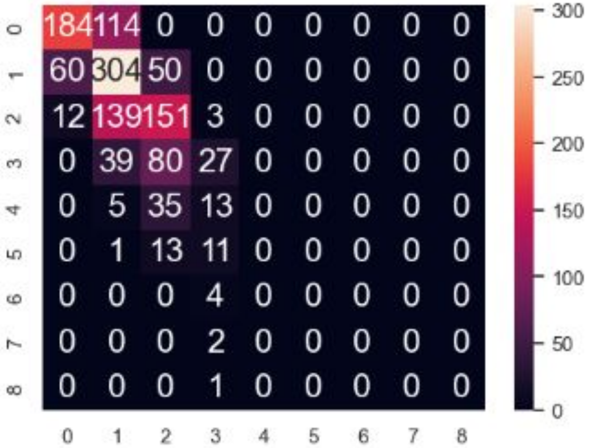
Sample from Random Forest Tuned

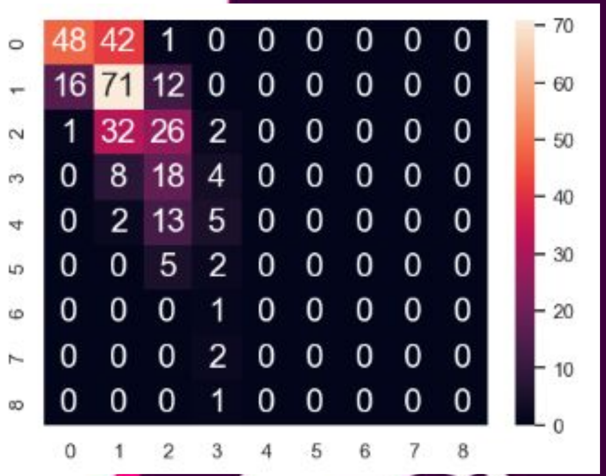
```
Random Forest Classifier with tuned values:
Goodness of Fit of Model        Train Dataset
Classification Accuracy         : 0.5336538461538461
Mean Squared Error (MSE)        : 0.9046474358974359

Goodness of Fit of Model        Test Dataset
Classification Accuracy         : 0.4775641025641026
Mean Squared Error (MSE)        : 1.141025641025641
```









Premier League

Tuning Result

The results showed that there is a small improvement in accuracy in the tuned random forest compared to the first random forest we did and the tuned random forest is performing better than the baseline model.

```
Average of Decision Tree (baseline) (over 100 times) Accuracy Score: 0.41275641025641036  MSE: 1.2846794871794867
Average of Random Forest Tuned (over 100 times) Accuracy Score: 0.49134615384615365  MSE: 0.8376923076923064
```

The results from before:

```
Average of Random Forest (over 100 times) Accuracy Score: 0.4966025641025642  MSE: 0.9815384615384615
```

Premier League

Checkpoint 5: Conclusion

Conclusion

- ***Random forest model*** is the most accurate in predicting the number of goals scored

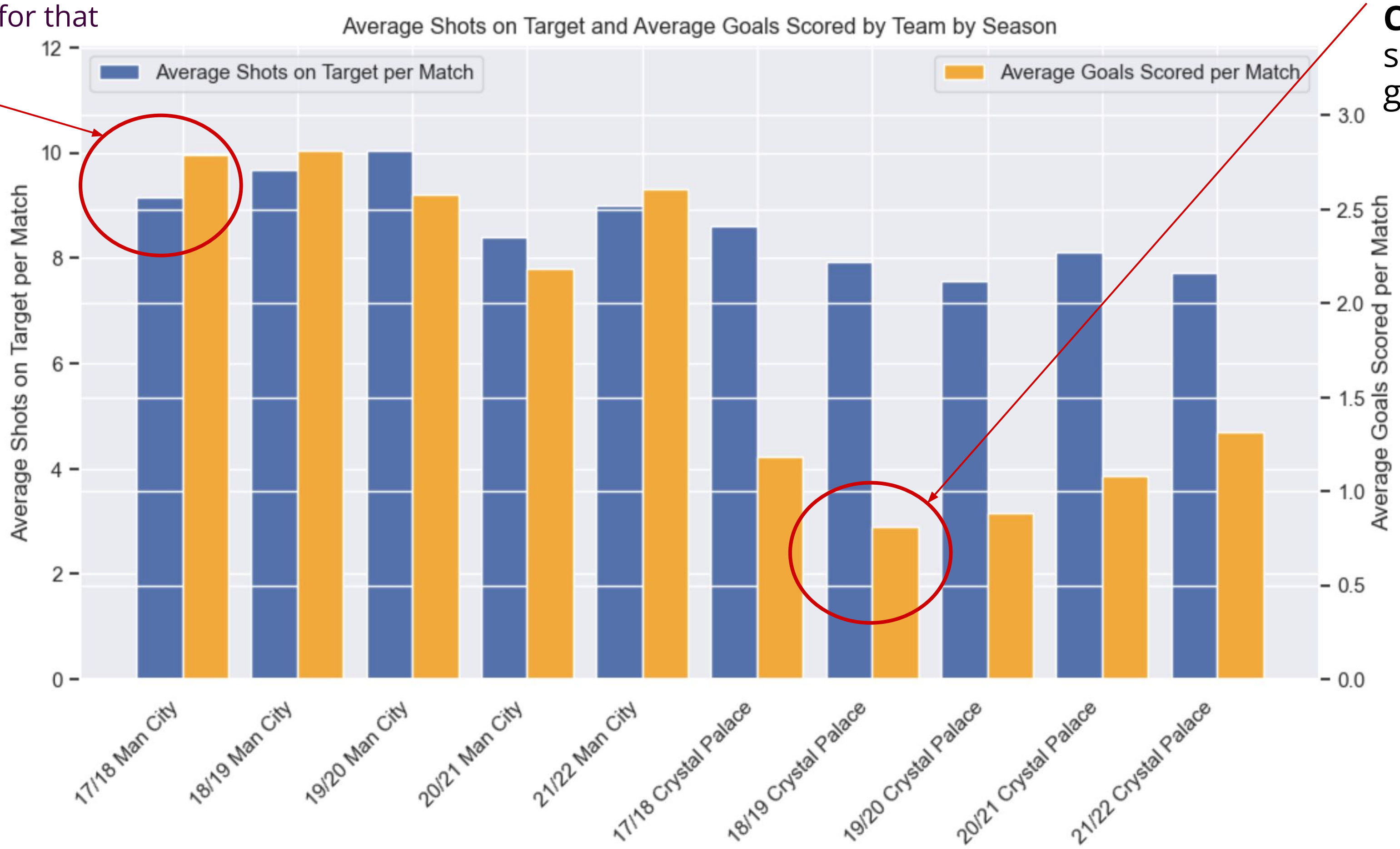
- To maximise goals scored, the team must:

  - Maximise **Shots Taken** & **Shots on Target**

- The most important predictor was ***Shots on Target***

Case Study

Man City scored more goals despite a low number of shots per match for that season



Despite having a significant number of shots per match, **Crystal Palace** scored very few goals that season.

Better teams will be more efficient at converting Shots on Target opportunities to goals scored, conversely **lower teams** might still find it challenging.

Limitations

- The data available to us had a **limited number of variables.**
- More extensive data with a greater number of variables such as **possession, passes**, etc. could be obtained to form a **more accurate regression model** to predict the number of goals scored in future.
- As seen from the case studies in the previous slide, higher shots on target **may not necessarily translate to more goals** consistently due to the individual abilities of various players for each team.

Thank You!

Premier League

References

1. 1.6. nearest neighbors. scikit. (n.d.). Retrieved April 22, 2023, from https://scikit-learn.org/stable/modules/neighbors.html#classification
2. Sklearn.ensemble.randomforestclassifier. scikit. (n.d.). Retrieved April 22, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
3. English Premier League Dataset. Saif. (2022) Retrieved March 2, 2023, from https://www.kaggle.com/datasets/saife245/english-premier-league
4. Create a Python Heatmap with Seaborn. Holland. (2018). Retrieved March 2, 2023, from https://absentdata.com/python-graphs/create-a-heat-map-with-seaborn/
5. Python BoxPlot Jupyter Notebook. Lowther. (2019). Retrieved March 19, 2023, from https://www.wafermovement.com/2019/08/pythonboxplotnotebook/
6. Making Plots in Jupyter Notebook Beautiful & More. Bipin. (2020). Retrieved March 19, 2023, from https://towardsdatascience.com/making-plots-in-jupyter-notebook-beautiful-more-meaningful-23c8a35c0d5d
7. scikit learn. scikit-learn-developers. (2007). Retrieved March 2, 2023 from https://scikit-learn.org/stable/about.html#citing-scikit-learn