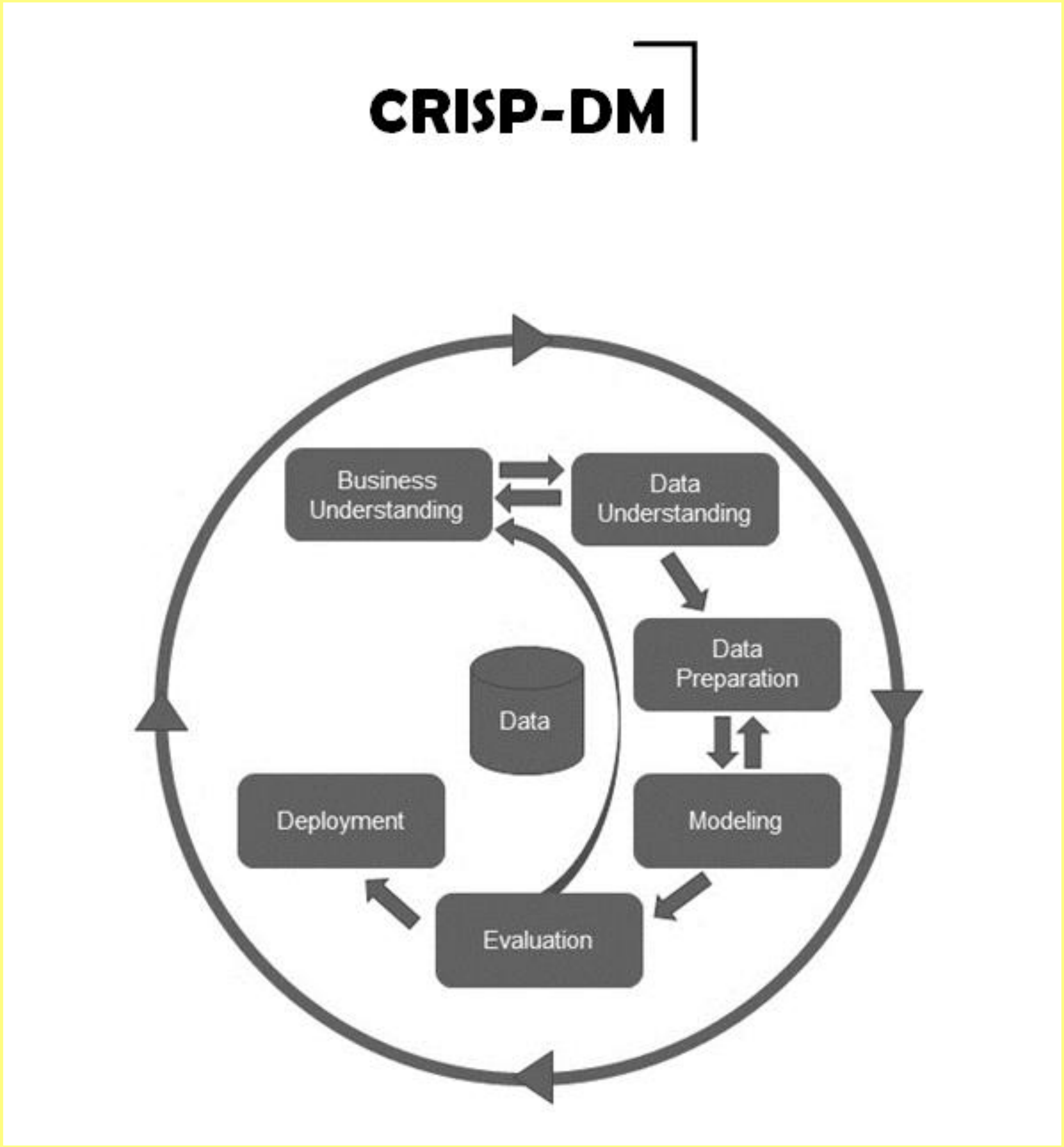
TITLE： Goal Scoring Rate Prediction in the 2019-20 England Football Premier League


CRISP-DM
Business Understanding
Data Understanding
Data Preparation
Data
Modeling
Deployment
Evaluation


Define Business Question & Objective
Web Scraping
- Match Statistic
- Lineup
Player Value
Score Rate
Mapping data from 3 website
Create feature,
Data transformation
Modeling
Evaluation & Tuning

BUSINESS QUESTION

"Data and statistics of the game can be used to raise the win rate of the bets "

DATA COLLECTION

WEB SCAPPING TOOLS

Selenium Package

BeautifulSoup Package

DATA

- Collect the player value with the last 4 seasons
- Collect the statistic on the each game with the last 4 seasons
- Collect the score rate on the each game with the last 4 seasons
- Collect (11+7) players and line up on the game with the last 4 seasons

WEB SCAPPING

Source of data

Match Statistic/Lineup

Player Value

Score Rate

Premier League

Home Fixtures

Line-ups | Stats | Related

Head-to-Head | Match Stats

Match Stats

Manchester United		Cardiff City
73.6	Possession %	26.4
10	Shots on target	4
26	Shots	13
811	Touches	404
620	Passes	223
15	Tackles	16
13	Clearances	20
11	Corners	2

transfermarkt

Player	Date of birth / Age	Market value
Mohamed Salah Right Winger	Jun 15, 1992 (27)	150,00 mil. €
Sadio Mané Left Winger	Apr 10, 1992 (27)	120,00 mil. €
Virgil van Dijk Centre-Back	Jul 8, 1991 (28)	100,00 mil. €
Alisson Goalkeeper	Oct 2, 1992 (27)	80,00 mil. €

THSCORE

ทีมเหย้า	คะแนน	ทีมเยือน	เกมสูงต่ำ	
			ทั้งรอบ	ครึ่งรอบ
อาร์เซน่อล[3]	4-0	แอสตัน วิลล่า[20]	3.5	1.5
สวอนซี ซิตี้[11]	1-1	แมนเชสเตอร์ ซิตี้[4]	3	1/1.5
เวสต์ บรอมมิช อัลเบียน[15]	1-1	ลิเวอร์พูล[8]	2.5/3	1/1.5
เซาแธมป์ตัน[7]	4-1	คริสตัล พาเลซ[14]	2.5/3	1/1.5

WEB SCAPPING

DATA PREPARATION

MAPPING

- **DATA MAPPING**

Season	Orde	HomePlayer	NEW_HomePlayer	Position_Hom	Home_Po_typ
2015/16 | Premier League	15	John Mikel Obi	John Mikel Obi	Defensive Midfield	Midfielder
2015/16 | Premier League	4	Seamus Coleman	Seamus Coleman	Right-Back	Defender
2015/16 | Premier League	4	Seamus Coleman	Seamus Coleman	Right-Back	Defender
2015/16 | Premier League	14	John Mikel Obi	John Mikel Obi	Defensive Midfield	Midfielder
2015/16 | Premier League	4	Seamus Coleman	Seamus Coleman	Right-Back	Defender
2015/16 | Premier League	15	John Mikel Obi	John Mikel Obi	Defensive Midfield	Midfielder
2015/16 | Premier League	16	John Mikel Obi	John Mikel Obi	Defensive Midfield	Midfielder

- **DATA AGGREGATION**

Result_PlayHigh	Result_PlayLow	Team_Home	Team_Away	Rate
Yes	No	Aston Villa	Sunderland	2.25
No	Yes	AFC Bournemouth	Leicester	2.5
Yes	No	Chelsea	Crystal Palace	2.75
Yes	No	Liverpool	West Ham	2.75
No	Yes	Man City	Watford	3.25
No	Yes	Newcastle	Arsenal	2.5
No	Yes	Stoke	West Brom	2.25

DATA MAPPING

match_	DescriptionMatch	Seaso	Ord	HomePlayer	AwayPlayer
12167	Chelsea v Arsenal	2015/16	15	John Obi Mikel	Calum Chambers
12256	Chelsea v AFC Bournemouth	2015/16	14	John Obi Mikel	Baily Cargil
12368	Everton v West Brom	2015/16	4	Séamus Coleman	Jonny Evans
12394	Everton v West Ham	2015/16	8	Séamus Coleman	Michail Antonio

Name : John Obi Mikel

VS

Name : John Mikel Obi

Name	Numbe	Positic	Age	Date_Bi	ate_Bi	'alue_E	alue_pc	ontract	Team
John Mikel Obi	12	Defensive	28	220487	22-04-87	13	9.477	-	fc-chelsea
Seamus Coleman	23	Right-Back	26	111088	11-10-88	18	13.122	30.06.202	fc-everton

Solution : Text Similarity (fuzzy string matching)

In [3]:

```
##fuzzy home####
list_result =[]
list_home = []
print(len(line_home))
i=0
for i in range(0,len(line_home)) :

    test_fuzzy = process.extractOne(str(line_home[i]),name_v)
    name,score = test_fuzzy
    if score != 100:
        print(i)
        print(line_home[i])
        print( f"{name} : {score}" )
        line_home[i]=name
        print(line_home[i])
        list_home.append(line_home[i])
    else:
        list_home.append(line_home[i])
```

```
6840
16
Max-Alain Gradel
Max Gradel : 86
Max Gradel
32
John Obi Mikel
John Mikel Obi : 95
John Mikel Obi
34
Radamel Falcao
Falcao : 90
Falcao
39
Séamus Coleman
Seamus Coleman : 96
Seamus Coleman
59
```

```
a = copy.deepcopy(data_hl)
m =[dict_trans[re.sub(r'\d+', '',i[0:i.find("[")].replace(" ",""))] for i in data_hl["HOME_
aw =[dict_trans[re.sub(r'\d+', '',i[0:i.find("[")].replace(" ",""))] for i in data_hl["AWAY_
cp = [str(i)[0] for i in data_hl["FULL_SCORE_RATE"]]
_aft_con = pd.DataFrame({"HOME_TEAM_EN" : ls_tm_hm,"AWAY_TEAM_EN" : ls_tm_aw}
py_data["HOME_TEAM_EN"] = dict_aft_con["HOME_TEAM_EN"]
opy_data["AWAY_TEAM_EN"] = dict_aft_con["AWAY_TEAM_EN"]
copy_data["RATE_FOR_RESULT"] = ls_rt_cp
```

DATA AGGREGATION

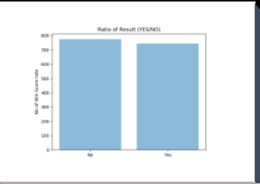
MA_Score_Home	MA_GoalConceded_Home	MA_Posses_Home	MA_Shots
0.67	1	45.77	
1.33	1.67	52.57	
1.67	2.33	52.97	
0.67	0	47.4	
2.67	0	57.53	
0.67	1.33	40.4	
1	1.33	50.4	

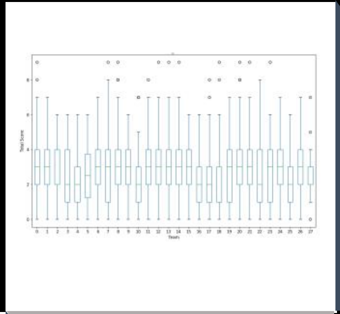
MOVING AVERAGE

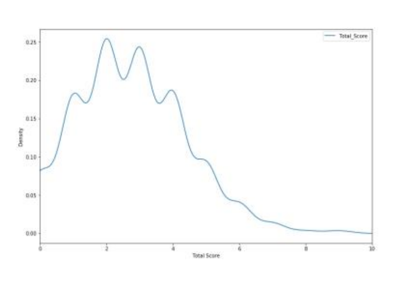
DATA AGGREGATION

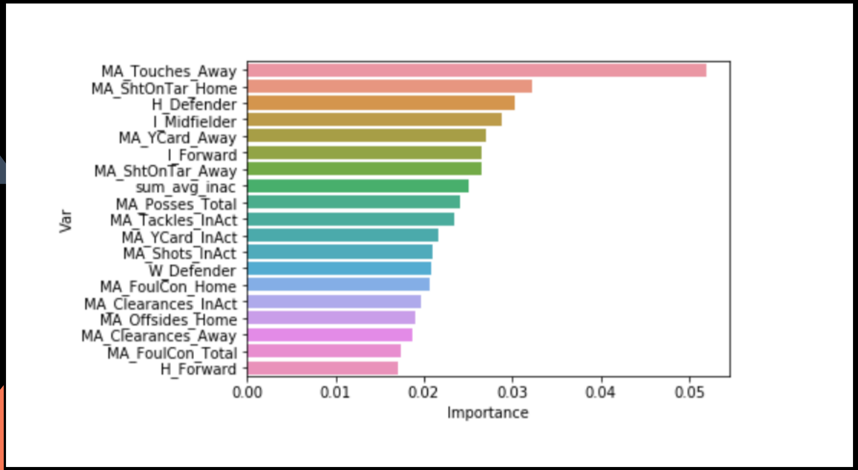
- We can't know about the statistics in matchs ,so that we used moving average 3 matchs before for each team replace the statistics in matchs

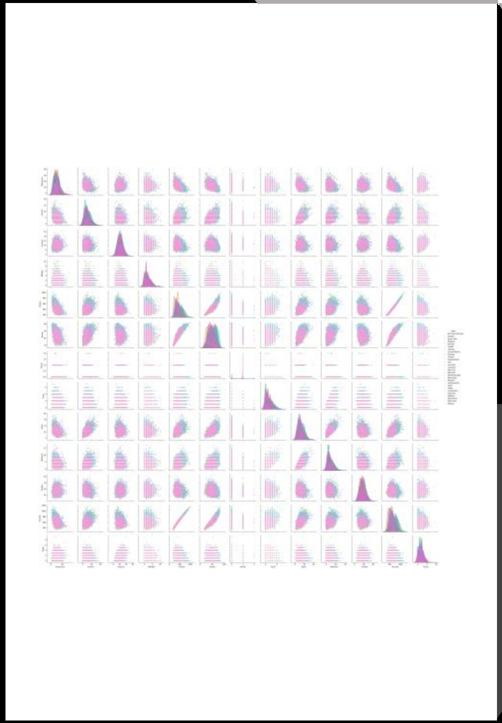
MA_RCard_Away	MA_Score_Total	MA_GoalConceded_Total
0	2	3.67
0	3.66	3
0	3.67	3.66
0.67	2.67	2
0	3.34	0.67
0	1.34	2.33
0	1.67	3.33

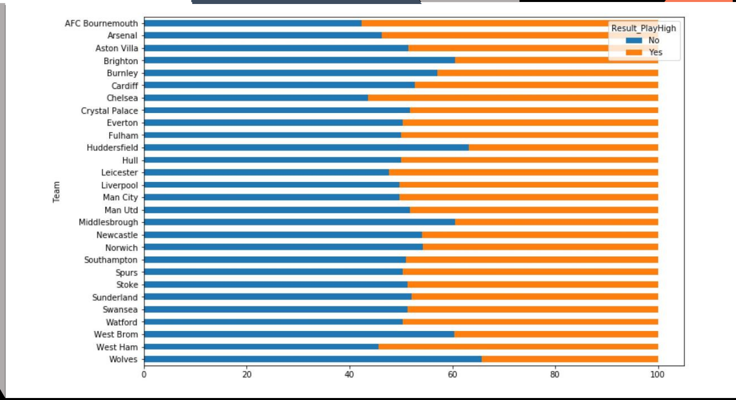












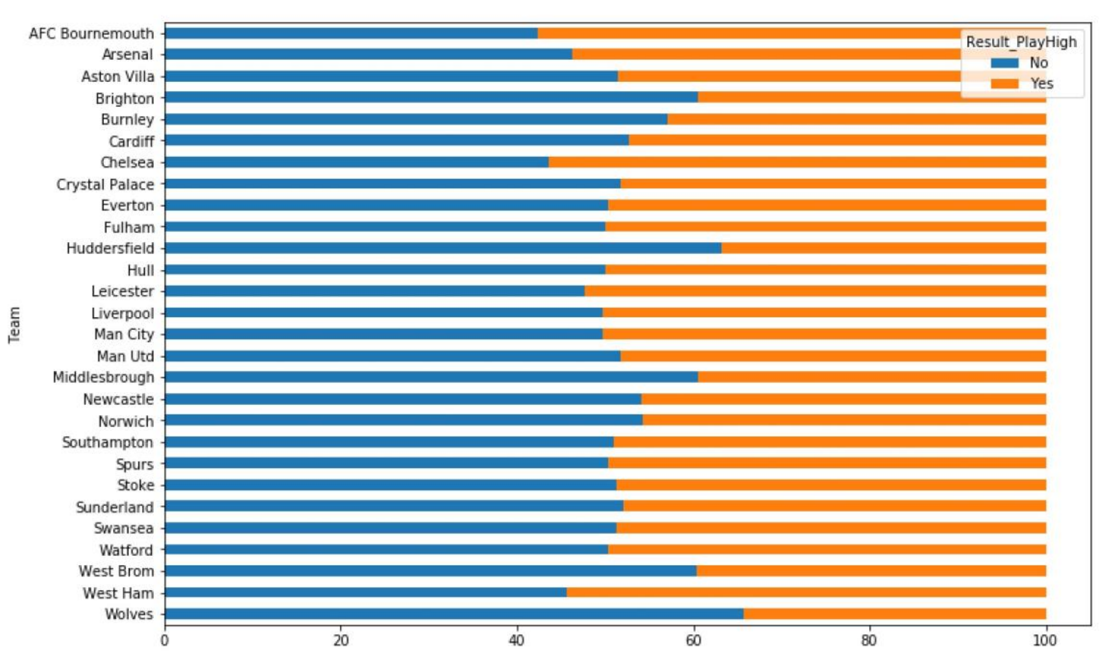
- Plot Graphs
- Data Understanding
- Variance Importance

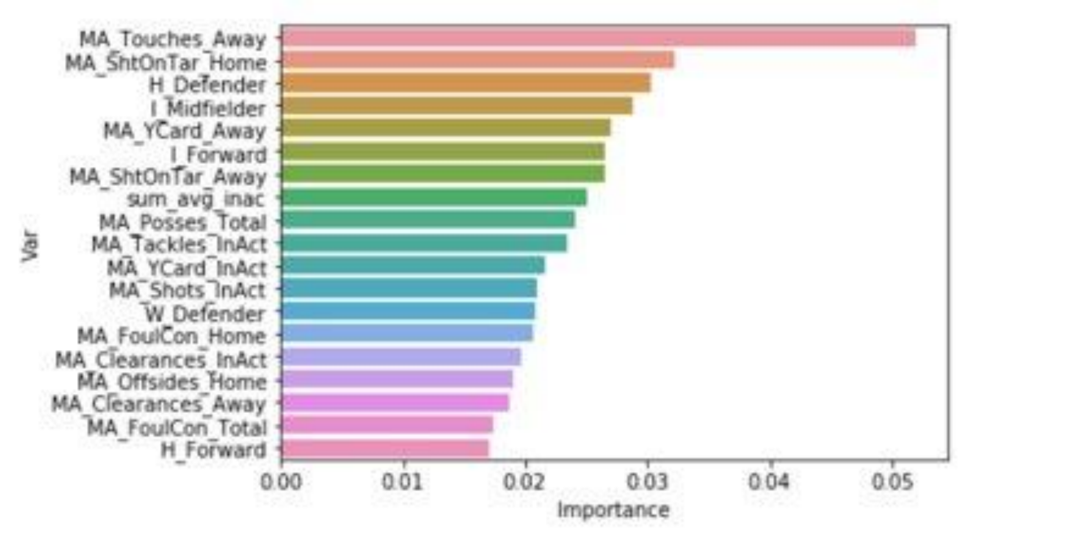
Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA)

- Top ratio of success when Bid High rate
  - 1. AFC Bournemouth
  - 2. Chelsea
  - 3. West Ham

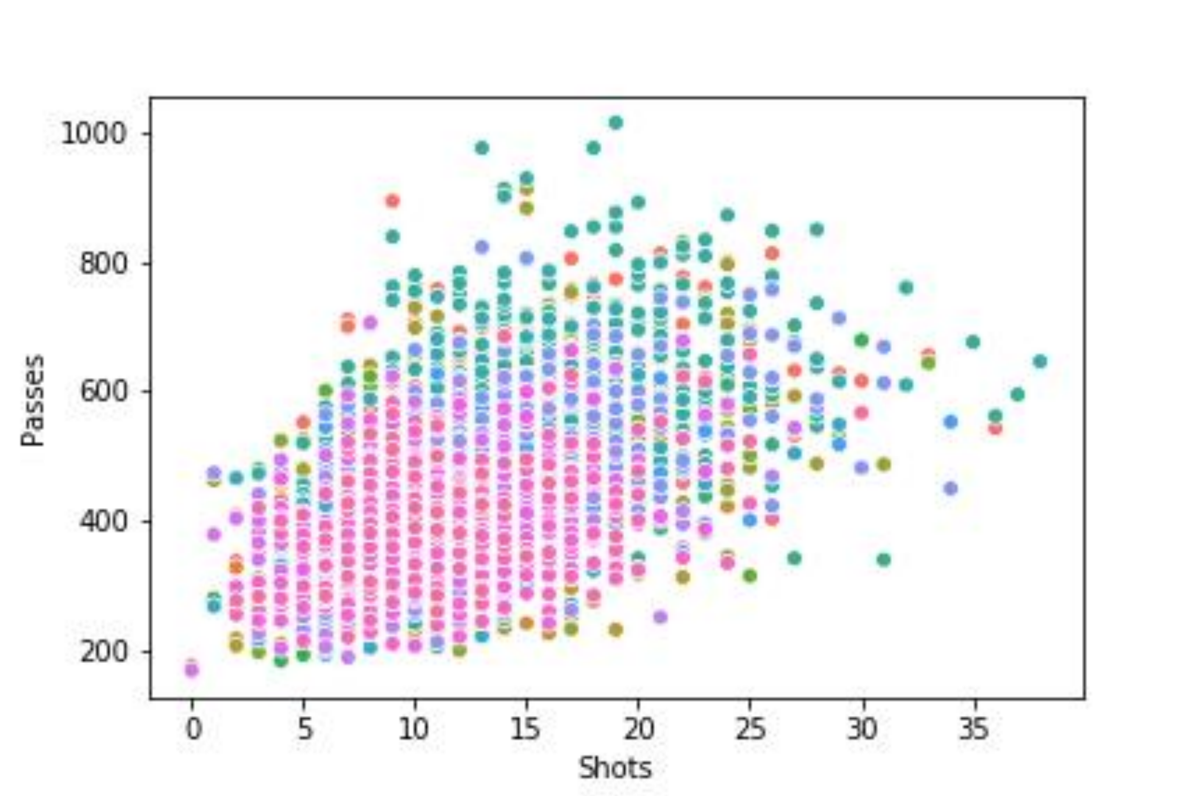
- Importance Variables
  - 1. MA_Touches_Away
  - 2. MA_ShtOnTar_Home
  - 3. H_Defender





SEGMENTATION

- Clustering by Match Statistics
- K-Means Clustering




```
model = KMeans(n_clusters=3,random_state=123)
model
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=123, tol=0.0001, verbose=0)
```

```
model.fit(X)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=123, tol=0.0001, verbose=0)
```

```
model.cluster_centers_
```

```
array([[2.41896145e+01, 5.45003934e+00, 1.08678206e+01, 2.00786782e+00,
        4.63010228e+02, 5.29837923e+01, 6.05822187e-02, 1.30055075e+01,
        4.28166798e+00, 1.76601101e+01, 6.66194335e+02, 1.66640441e+00],
       [1.85601926e+01, 6.87961477e+00, 9.23434992e+00, 1.95505618e+00,
        6.44894061e+02, 6.61542536e+01, 2.08667737e-02, 1.70754414e+01,
        5.95345104e+00, 1.60914928e+01, 8.41399679e+02, 1.25842697e+00],
       [2.98542757e+01, 4.07504363e+00, 1.12705061e+01, 2.02094241e+00,
        3.22916230e+02, 3.79088133e+01, 8.37696335e-02, 9.83856894e+00,
        3.37870855e+00, 1.81317627e+01, 5.19765271e+02, 1.79930192e+00]])
```

```
model.labels_
```

```
array([0, 0, 0, ..., 2, 2, 0])
```

Cardiff	1	0	37
Chelsea	63	79	10
Crystal Palace	60	5	87
Everton	91	18	43
Fulham	15	9	14
Huddersfield	33	4	39
Hull	20	1	17
Leicester	65	8	79
Liverpool	46	97	9
Man City	41	110	1
Man Utd	67	65	20

Team_Home	Team_Away	Style_Home	Style_Away
AFC Bournemouth	Aston Villa	DEF	DEF
Chelsea	Swansea	DEF	DEF
Everton	Watford	DEF	DEF
Leicester	Sunderland	DEF	DEF
Man Utd	Spurs	DEF	DEF
Norwich	Crystal Palace	DEF	DEF
Arsenal	West Ham	DEF	DEF
Newcastle	Southampton	DEF	DEF

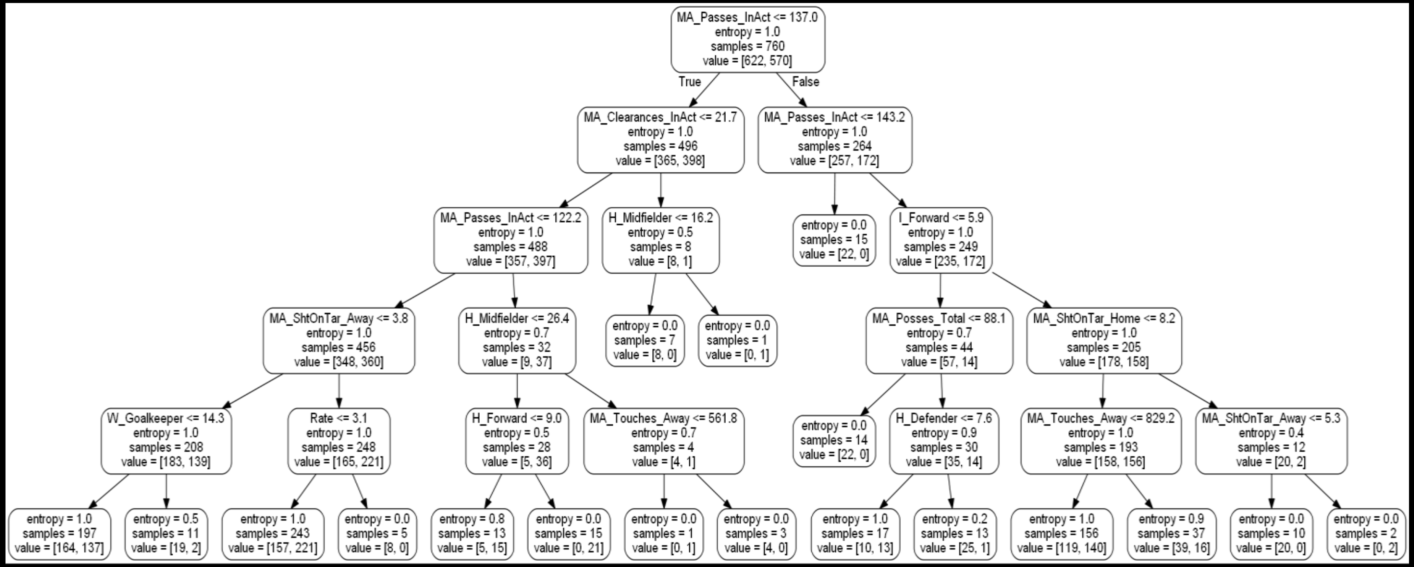
MODELING

- Random Forest

```
model = RandomForestClassifier(n_estimators=200,max_depth =5
                              ,max_features="log2",criterion='entropy' ,random_state=123)
model.fit(x_train,y_train)
```

1. Split Train/Test data
2. Transform data
   e.g. One-Hot Encoding
3. Split Feature and Label data
4. Train Model
5. Make Prediction on the testset
6. Tune Model
7. Result

Sample Tree in Forest



MODELING

```
labelencoder = LabelEncoder()
check = data_use.select_dtypes(include=['object']).columns
for kk in check :
    data_use[kk] = labelencoder.fit_transform(data_use[kk])
```

```
y = data_use['Result_PlayHigh']
data_use.drop(['Result_PlayHigh'], axis=1, inplace=True)
# x are the others
x = data_use
```

```
x_train , x_test , y_train , y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)
```

```
model = RandomForestClassifier(random_state=123)
model.fit(x_train,y_train)
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import numpy as np
rfc=RandomForestClassifier(random_state=123)
param_grid = {
 'n_estimators': [100,200,300,400,500,600,700,800,900,1000],
 'max_features': ['auto', 'sqrt', 'log2'],
 'max_depth' : [2,3,4,5,6,7,8,9,10],
 'criterion' :['gini', 'entropy']
}
CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5)
CV_rfc.fit(x_train, y_train)
```

```
CV_rfc.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 5,
 'max_features': 'log2',
 'n_estimators': 600}
```

RESULT OF Random Forest

BEFORE TUNING

Confusion Matrix

	1	0
1	54	52
0	90	102

Classification Report

	Precision	Recall	F1_score	Support
1	0.51	0.38	0.43	144
0	0.53	0.66	0.59	154
avg/total	0.52	0.52	0.51	298

Accuracy Rate : 0.524

AFTER TUNING

Confusion Matrix

	1	0
1	81	66
0	63	88

Classification Report

	Precision	Recall	F1_score	Support
1	0.55	0.56	0.56	144
0	0.58	0.57	0.58	154
avg/total	0.57	0.57	0.57	298

Accuracy Rate : 0.567

RESULT OF XGboost

Accuracy Rate : 0.570

Confusion Matrix

	1	0
1	85	62
0	67	84

Classification Report

	Precision	Recall	F1_score	Support
1	0.56	0.58	0.57	147
0	0.58	0.56	0.57	151
ang/total	0.57	0.57	0.57	298

Validation Result (YES/NO)

Data : **2019-20 England Football Premier League**

GURU VS Real Score	RF VS Real Score	XGboost VS Real Score
Acc Rate : 0.591	Acc Rate : 0.492	Acc Rate : 0.5202

Conclusion:
GURU prediction return accuracy rate better than XGboost Model and Random Forest Model but can't defind whether return the best profit

Validation Result (Cont)

Insight of Result : Check whether which model return the maximum profit

YES :WIN

NO : LOSS

DRAW

GURU VS Real Score

Rate Result
Win Rate : 0.57
Draw Rate :0.02
Loss Rate :0.41

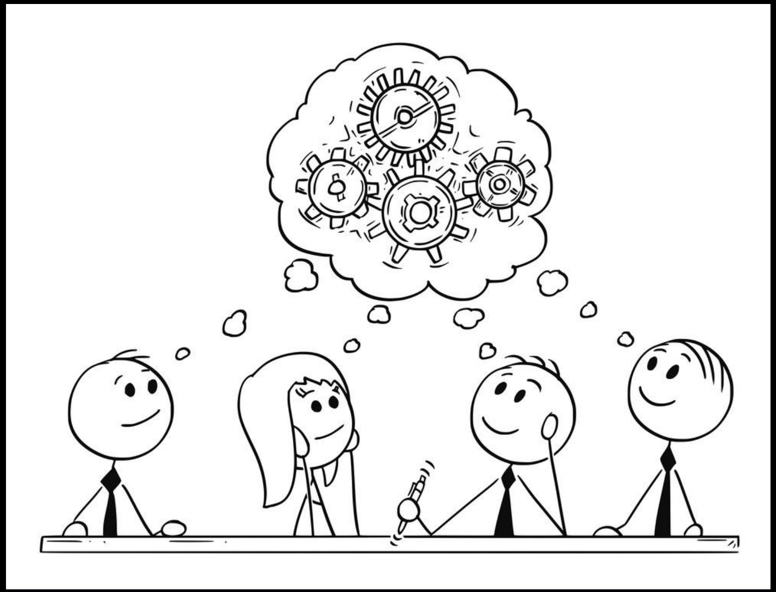
RF VS Real Score

Rate Result
Win Rate : 0.47
Draw Rate :0.05
Loss Rate :0.48

XG VS Real Score

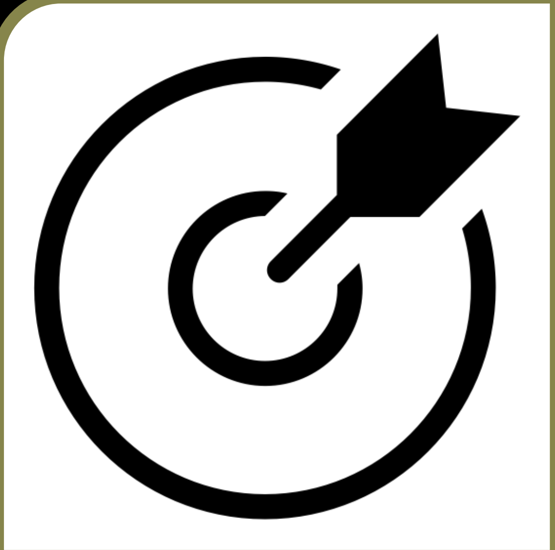
Rate Result
Win Rate : 0.47
Draw Rate :0.05
Loss Rate :0.48

Conclusion : Guru Prediction return the best profit



Problem

- Leak some Feature ex : preview game inside of guru
- The result of models are not satisfied because our features not enough for model prediction.
- Many external factors



Conclusion

- Althought our model was not satisfied performance but we got many experience.
    1. practical experience -> Webscraping , Build model and use python in others.
    2. Found real problem
    3. Found some hidden somethings.

Q&A

THANK YOU