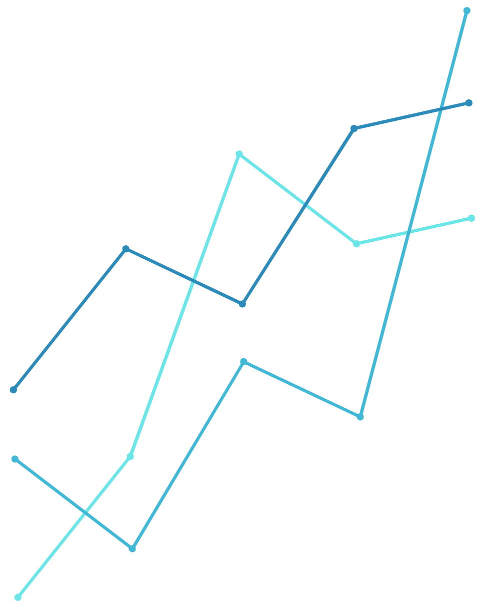
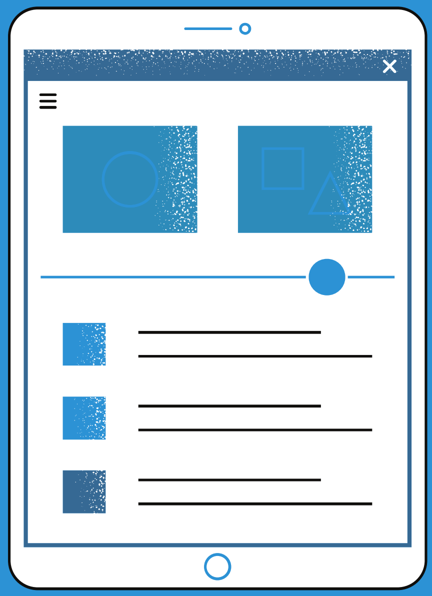
The Quintessential Analyst



Presented by - Mohammad Faraaz Usmani
1901640100169

Objective

The main objective of this project is to utilize machine learning technology to project the outcome of football matches



Introduction

The Quintessential Analyst is a sports analyzing system that uses the previous match results of a team and predicts the result of their upcoming fixtures

Feasibility

Technical

The processing power and time consumed varies according to the size of the dataset

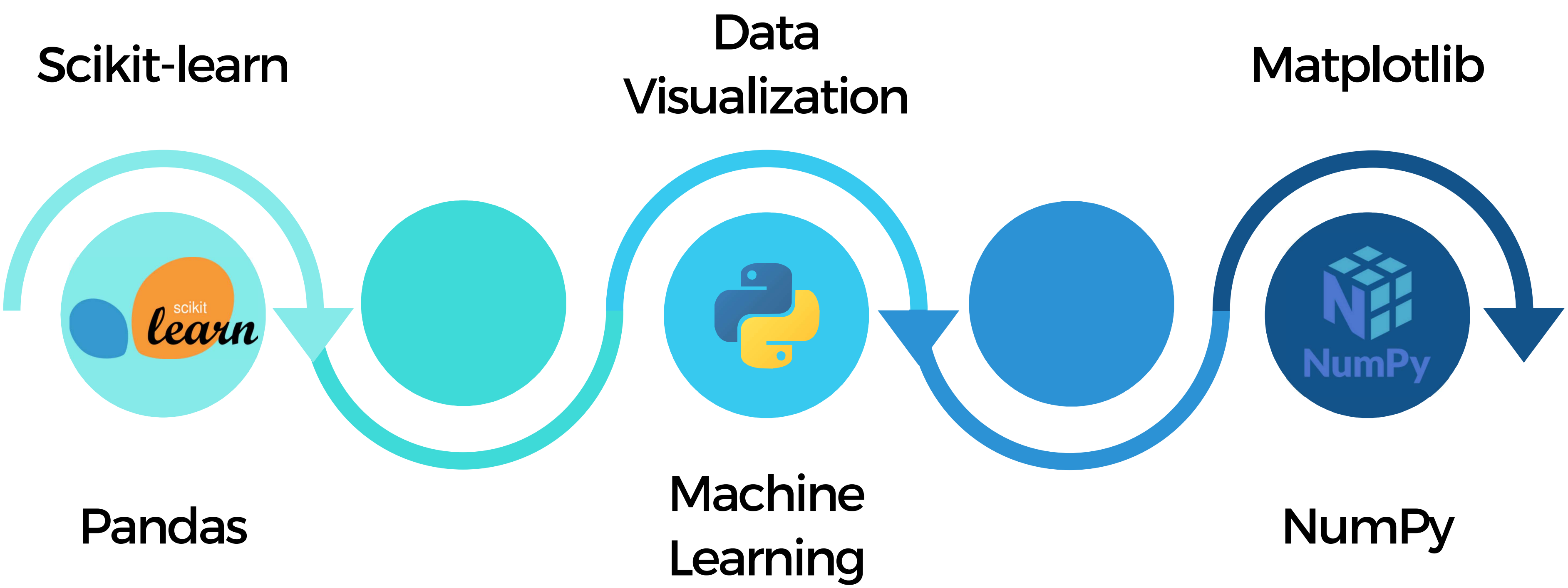
Economical

No need for any additional resources.
Data scraped from the web is enough

Operational

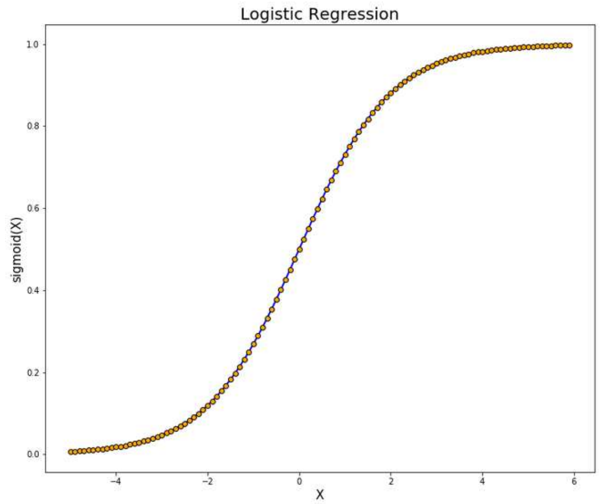
Correct dataset with accurate data values is needed for the system to function effeciently

Technologies and Libraries Used

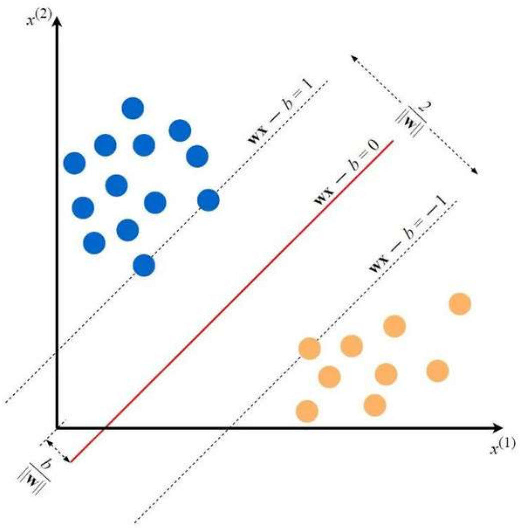


Algorithms Used

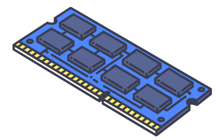
Logistic Regression



Support Vector Machine

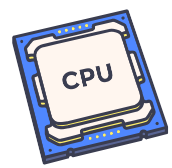


Hardware Required



4GB RAM onwards

100GB storage minimum



Processor clock speed 2GHz or more

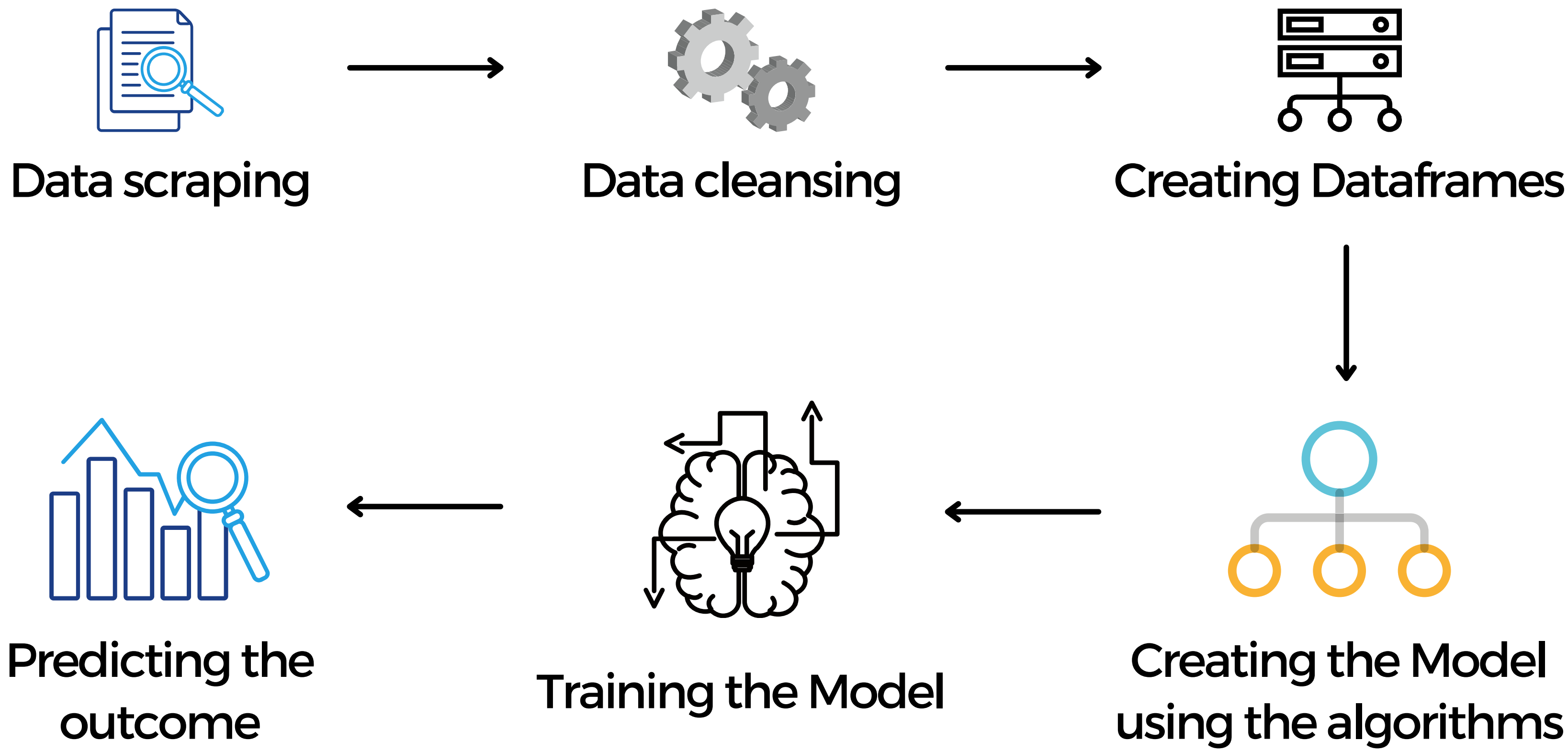
Intel i5 8th gen or equivalent onwards (Quadcore minimum)

Software Required

Windows 10/11 or MacOS

Any Text Editor

Jupyter


Data scraping
Data cleansing
Creating Dataframes
Creating the Model using the algorithms
Training the Model
Predicting the outcome

Code Snippets

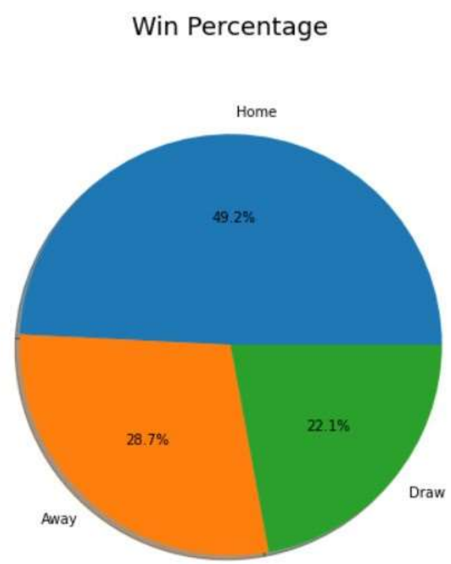
In [1]:

```
import pandas as pd
import numpy as np
df1=pd.read_csv('E0.csv',usecols=['HomeTeam','AwayTeam','FTHG','FTAG','FTR','HS','AS','HST','AST'
,'B365H','B365D','B365A'])
df1.head()
```

Out[1]:

	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HS	AS	HST	AST	B365H	B365D	B365A
0	Burnley	Swansea	0	1	A	10	17	3	9	2.40	3.3	3.25
1	Crystal Palace	West Brom	0	1	A	14	13	4	3	2.00	3.3	4.50
2	Everton	Tottenham	1	1	D	12	13	6	4	3.20	3.4	2.40
3	Hull	Leicester	2	1	H	14	18	5	5	4.50	3.6	1.91
4	Man City	Sunderland	2	1	H	16	7	4	3	1.25	6.5	15.00

In [2]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(6,8))
plt.pie(df1['FTR'].value_counts(),labels=['Home','Away','Draw'], autopct='%1.1f%%',shadow=True, startangle=0)
plt.axis('equal')
plt.title('Win Percentage', size=18)
plt.show()
```



This pie graph shows that the home team has higher chance to win the game.

In [3]:
```
from statsmodels.stats import proportion
conf=proportion.proportion_confint((df1['FTR']=='H').sum(), df1['FTR'].count(), alpha=0.05, method='normal')
print('The chance of home team to win with %95 confidence interval falls in :{}'.format(conf))
```

```
The chance of home team to win with %95 confidence interval falls in :(0.441839514668131, 0.5423710116476584)
```

In order to have some estimation of attack, defense and possession of different team, I have added another data frame to the previous data frame which can be downloaded from the following link: http://www.squawka.com/

In [4]:
```
##add new data to data frame
dfSq=pd.read_csv('dataE0.csv',index_col='Team').dropna(axis=0,how='any')
##Hde: Home Defense    Hatt: Home Attack    Hpo: Home possession    Htot : Home total power
##Ade: Away defense   Aatt: away attack    Apo : Away possession :  Atot: Away total power
dfSq.head()
dff=df1.join(dfSq[['Hde','Hatt','Hpo','Htot']],on='HomeTeam')
df=dff.join(dfSq[['Ade','Aatt','Apo','Atot']],on='AwayTeam')
```

1) Taking the average of previous games different features for every team and considering as a new feature for the model (For example Man City played against 10 teams in the middle of season and they had 20 shots on target, therefore we take this average (2 shots per game) and consider it as new feature)

2) I have also defined a momentum which gives the average of five previous games for each team. It could be helpful to make our model more accurately. If for example some team in the five previous games shows poor results or great results, we can track them.

In [5]:
```python
def make_data(df):
    ##add points for away and home team : win 3 points, draw 1 point, loss 0 point
    df['HP']=np.select([df['FTR']=='H',df['FTR']=='D',df['FTR']=='A'],[3,1,0])
    df['AP']=np.select([df['FTR']=='H',df['FTR']=='D',df['FTR']=='A'],[0,1,3])
    ## add difference in goals for home and away team
    df['HDG']=df['FTHG']-df['FTAG']
    df['ADG']=-df['FTHG']+df['FTAG']
    ##add momentum to data
    cols=['Team','Points','Goal','Shoot','TargetShoot','DiffG']
    df1=df[['HomeTeam','AwayTeam','HP','AP','FTHG','FTAG','HS','AS','HST','AST','HDG','ADG']]
    df1.columns=[np.repeat(cols,2),['Home','Away']*len(cols)]
    d1=df1.stack()
    ##find momentum of previous five games for each team
    mom5 = d1.groupby('Team').apply(lambda x: x.shift().rolling(5, 4).mean())
    mom=d1.groupby('Team').apply(lambda x: x.expanding().mean().shift())
    ##add the found momentum to the dataframe
    df2=d1.assign(MP=mom5['Points'],MG=mom5['Goal'],MS=mom5['Shoot'],MST=mom5['TargetShoot'],MDG=mom5['DiffG'],AP=mom['
    df2=df2.drop(['Points','Goal','Shoot','TargetShoot','DiffG'],axis=1)
    df_final=pd.merge(df[['HomeTeam','AwayTeam','FTR','B365H','B365D','B365A','Ade','Aatt','Apo','Atot','Hde','Hatt','Hp
    df_final=df_final.dropna(axis=0,how='any')
    ##Full time results ('FTR') : Home=0,Draw=1,Away=2
    Y_all=df_final['FTR']
    ##Full time results ('FTR') : Home=0,Draw=1,Away=2
    ##Prediction of betting company (bet365)=Y_Bet
    Y_Bet=df_final[['B365H','B365D','B365A']].apply(lambda x:1/x)
    ## winner based on bet365 data
    Y_Bet_FTR=np.select([Y_Bet.idxmax(axis=1)=='B365H',Y_Bet.idxmax(axis=1)=='B365D',Y_Bet.idxmax(axis=1)=='B365A'],['H'
    ##scale data
    df_X=df_final.drop([('Team', 'Home'),('Team', 'Away'),'FTR','HomeTeam','AwayTeam','B365H','B365D','B365A'],axis=1)
    return df_X, Y_all,Y_Bet,Y_Bet_FTR
df_X, Y_all,Y_Bet,Y_Bet_FTR=make_data(df)
```

In [6]:

```
from sklearn.preprocessing import scale
X_all=scale(df_X)
```

In [7]:

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn import linear_model
from sklearn.svm import SVC
from sklearn.metrics import classification_report
def data_split(X_all, Y_all,Y_Bet_FTR,Y_Bet):
    X_train, X_test, y_train, y_test,y_train_bet_FTR,y_test_bet_FTR,y_train_bet,y_test_bet = train_test_split(X_all, Y_a
    return X_train, X_test, y_train, y_test,y_train_bet_FTR,y_test_bet_FTR,y_train_bet,y_test_bet
def predict_labels(clf,X_test):
    y_pred=clf.predict(X_test)
    return y_pred
def report_score(clf,X_test,y_test,y_pred,X_train,y_train):
    target_names = ['H', 'D', 'A']
    print(classification_report(y_test, y_pred, target_names=target_names))
    print ('{}....Test accuracy:{} Train accuracy:{}'.format(clf.__class__.__name__,clf.score(X_test,y_test),clf.score(X
def report_score_bet365(y_test,y_pred):
    target_names = ['H', 'D', 'A']
    print(classification_report(y_test, y_pred, target_names=target_names))
    print ('BET365 accuracy:{} '.format((y_test==y_pred).sum()/len(y_test)))
def train_classifier(clf,parameters,X_train,y_train):
    grid_class = GridSearchCV(clf,scoring='accuracy',param_grid=parameters)
    grid_class = grid_class.fit(X_train,y_train)
    clf = grid_class.best_estimator_
    return clf
clf_logistic= linear_model.LogisticRegression(multi_class = "ovr", solver = 'newton-cg', class_weight = 'balanced')
clf_svc = SVC(kernel="linear",probability=True)
clfs=[clf_logistic,clf_svc]
X_train, X_test, y_train, y_test,y_train_bet_FTR,y_test_bet_FTR,y_train_bet,y_test_bet=data_split(X_all, Y_all,Y_Bet_FTR
parameter_logistic = {'C': np.logspace(-5,5,100)}
parameter_SVC = {'C': np.arange(0.1,3,0.01)}
parameters={clfs[0]:parameter_logistic,clfs[1]:parameter_SVC}
```

In [8]:
```
for clf in clfs:
    clf=train_classifier(clf,parameters[clf],X_train,y_train)
    y_pred=predict_labels(clf,X_test)
    report_score(clf,X_test,y_test,y_pred,X_train,y_train)
```

```
              precision    recall  f1-score   support

           H       0.49      0.71      0.58        28
           D       0.25      0.15      0.19        20
           A       0.71      0.65      0.68        54

    accuracy                           0.57       102
   macro avg       0.48      0.50      0.48       102
weighted avg       0.56      0.57      0.56       102

LogisticRegression....Test accuracy:0.5686274509803921 Train accuracy:0.6428571428571429
              precision    recall  f1-score   support

           H       0.59      0.57      0.58        28
           D       0.50      0.10      0.17        20
           A       0.63      0.83      0.72        54

    accuracy                           0.62       102
   macro avg       0.58      0.50      0.49       102
weighted avg       0.60      0.62      0.57       102

SVC....Test accuracy:0.6176470588235294 Train accuracy:0.6512605042016807
```

In [9]:
```
report_score_bet365(y_test,y_test_bet_FTR)
```

```
              precision    recall  f1-score   support

           H       0.60      0.75      0.67        28
           D       0.00      0.00      0.00        20
           A       0.67      0.83      0.74        54

    accuracy                           0.65       102
   macro avg       0.42      0.53      0.47       102
weighted avg       0.52      0.65      0.58       102

BET365 accuracy:0.6470588235294118
```

Conclusion

We can see that by using both, logistic regression and support vector machine, we were able to project the accurate outcomes of matches.

The use of Bet365 data set is done to show that the predicted results have turned out to be accurate.

Whenever our model predicts higher probability for win or draw it is logical to trust our model.

Future Scope

- I plan to make this model such that it can suggest formations and solutions to the teams to change their negative predicted results.
- By taking into consideration the performance of managers and individual players it can then suggest which players to sign and which players to list for transfer
- We can include player health monitoring and analysis to notch it up a bit.
- Most of the teams and clubs have their academies. We can implement the system for these academies and also include scouting features as well.

Future Scope

- We can increase the factors being taken into consideration to increase the accuracy of the results.
- We can take this project to a whole another level by including datasets from other major sports played across the globe.

Credit and References

- GitHub
- StackOverflow
- English Premier League 2016-17 data: football-data.co.uk
- javatpoint.com
- scikit-learn.org

Thank you