Overview

Objective

- To use supervised learning to build a prediction model to determine the number of all stars awards that a NBA players supposed to have based on their their existing advance stats and the ter ranking that we predicted previously from the nba_ast_fans data set

Dataset Source

- nba_ast_fans Advance stats + ter ranking

Dataset Column Descriptions:

- https://basketball-reference.com/leagues/

Import Packages

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading CSV File

In [2]:
```
df = pd.read_csv("nba_ast_fans.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	Player	TPA	APG%	Total ast %	DRB%	ORB%	TRB%	AST%	TOV%	STL%	...	PPM	DRtg	ORtg	aDOR	FIC	PER	OP	DP
0	A.J. Price	0.0175	0.4875	19.05	2.55	0.60	3.55	12.80	21.55	1.40	...	1.03	105.00	107.00	4.85	236.00	17.40	4.40	2.00
1	Aaron Brooks	0.0146	0.4640	199.00	2.00	6.60	4.40	24.20	14.90	1.40	...	1.10	103.00	111.40	-7.75	318.00	12.40	89.0	2.60
2	Aaron Gordon	0.0000	0.0000	140.00	5.75	16.00	11.40	11.80	11.35	1.25	...	1.20	107.00	108.00	-0.80	139.70	15.10	15.0	1.00
3	Aaron Gray	0.0000	0.0000	161.75	1.45	8.85	5.00	18.65	12.05	1.70	...	1.10	103.00	111.70	-6.15	282.00	11.85	86.0	2.60
4	Adam Morrison	0.0000	0.0000	132.20	2.75	14.80	8.40	6.00	12.80	1.30	...	1.10	102.00	106.00	-3.00	125.00	4.25	46.0	2.00

5 rows × 23 columns

Dataset Dimension

In [4]:
```
df.shape
```

Out[4]:
```
(864, 23)
```

Dataset Info

In [5]:
```
df.info()
```

Out[5]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864 entries, 0 to 863
Data columns (total 23 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Player   864 non-null    object
 1   TPA      864 non-null    float64
 2   APG%     864 non-null    float64
 3   Total A% 864 non-null    float64
 4   DRB%     864 non-null    float64
 5   ORB%     864 non-null    float64
 6   TRB%     864 non-null    float64
 7   AST%     864 non-null    float64
 8   TOV%     864 non-null    float64
 9   STL%     864 non-null    float64
 10  BLK%     864 non-null    float64
 11  USG%     864 non-null    float64
 12  PPR      864 non-null    float64
 13  PPS      864 non-null    float64
 14  ORtg     864 non-null    float64
 15  DRtg     864 non-null    float64
 16  aDOR     864 non-null    float64
 17  FIC      864 non-null    float64
 18  PER      864 non-null    float64
 19  OP       864 non-null    float64
 20  DP       864 non-null    float64
 21  WS       864 non-null    float64
 22  Group    864 non-null    int64
dtype: object, float64(21)
memory usage: 155.76 KB
```

Reading Dataset Column names.

In [6]:
```
df.columns.value
```

In [7]:
```
column_names =
['Player', 'TPA', 'APG%', 'Total A%', 'DRB%', 'ORB%',
 'TRB%', 'AST%', 'TOV%', 'STL%', 'BLK%', 'USG%', 'PPR',
 'PPS', 'ORtg', 'DRtg', 'aDOR', 'FIC', 'PER', 'OP', 'DP',
 'WS', 'Group']
```

Checking nan values

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
Player    0
TPA       0
APG%      0
Total A%  0
DRB%      0
ORB%      0
TRB%      0
AST%      0
TOV%      0
STL%      0
BLK%      0
USG%      0
PPR       0
PPS       0
ORtg      0
DRtg      0
aDOR      0
FIC       0
PER       0
OP        0
DP        0
WS        0
Group     0
dtype: int64
```

Creating Train & Test Dataset

```
# Dropping Player column since it's not necessary
df.drop('Player', axis=1)
# Creating train and test dataset randomly
y = df['Group']
```

In [9]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

Shape of Train & Test Dataset

In [10]:
```
X_train.shape, X_test.shape
```

Out[10]:
```
(604, 22), (260, 22)
```

In [11]:
```
X_train.dtypes
```

Out[11]:
```
TPA       float64
APG%      float64
Total A%  float64
DRB%      float64
ORB%      float64
TRB%      float64
AST%      float64
TOV%      float64
STL%      float64
BLK%      float64
USG%      float64
PPR       float64
PPS       float64
ORtg      float64
DRtg      float64
aDOR      float64
FIC       float64
PER       float64
OP        float64
DP        float64
WS        float64
dtype: object
```

In [12]:
```
X_train.head()
```

Out[12]:

	TPA	APG%	Total A%	DRB%	ORB%	TRB%	AST%	TOV%	STL%	BLK%	...	PPR	DRtg	ORtg	aDOR	FIC	PER	OP	DP
607	0.0000	0.0000	100.00	4.00	2.00	3.00	0.00	0.00	0.00	0.00	...	0.00	109.00	109.00	0.00	38.40	10.80	0.00	0.00
168	0.0127	0.4640	199.00	7.15	21.60	14.00	14.90	13.9	2.00	3.20	...	1.17	105.00	104.00	-1.00	401.00	14.00	73.0	12.0
448	0.0230	0.3140	141.75	3.85	10.60	8.75	13.05	18.4	2.35	1.90	...	1.17	108.00	111.40	-10.20	284.40	11.80	80.0	3.0
239	0.0000	0.0000	146.00	1.80	11.00	6.30	26.70	11.2	2.05	1.00	...	1.17	111.00	110.00	-4.05	141.00	10.30	44.0	2.0
862	0.0000	0.0000	100.00	11.00	2.00	6.50	10.60	16.2	1.70	3.00	...	1.17	87.40	114.70	-17.30	315.80	15.20	90.0	2.0

5 rows × 22 columns

Count All Stars Class

In [13]:
```
df['Group'].value_counts()
```

Out[13]:
```
0    597
1    150
2     77
3     18
4     12
5      8
6      6
7      5
8      4
9      3
10     2
11     1
13     1
14     1
Name: All-Stars, dtype: int64
```

Building Decision Tree model

In [14]:
```
from sklearn.tree import DecisionTreeClassifier
```

Why choose max depth all 5?

Max depth of 5 is the optimal for both training set and test set has the highest accuracy. The technique we use is the same as choosing the k. Start the depth from 1 and keep increasing till we find the highest accuracy without over fitting the model

In [15]:
```
clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=5, random_state=1)
clf_entropy = DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=1)
```

In [16]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=5, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=1, splitter='best')
```

Predict Test Data using Trained model

In [17]:
```
y_pred_gini = clf_gini.fit(X_train,y_train)
```

Checking Accuracy on Test Dataset

In [18]:
```
from sklearn.metrics import accuracy_score

print("Model accuracy score with criterion gini index: {0:0.4f}", format(accuracy_score(y_test, y_pred_
```

Out[18]:
```
Model accuracy score with criterion gini index: 0.8923
```

Checking Accuracy on Train Dataset

In [19]:
```
y_pred_train_gini = clf_gini.fit(X_train,y_train)
```

In [20]:
```
print("Training set score: {0:0.4f}", format(accuracy_score(y_train, y_pred_train_gini)))
```

Out[20]:
```
Training set score: 0.9007
```

Comparing Scores of Both Dataset

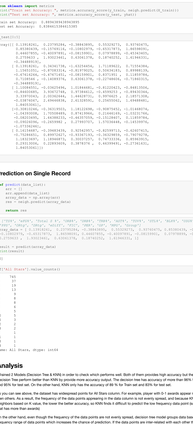
In [21]:
```
print("Training set score: {:.4f}".format(clf_gini.score(X_train, y_train)))

print("Test set score: {:.4f}".format(clf_gini.score(X_test, y_test)))
```

Out[21]:
```
Training set score: 0.9007
Test set score: 0.8923
```

Plotting Decision Tree

In [22]:
```
plt.figure(figsize=(12,8))
from sklearn.tree import tree
tree.plot_tree(clf_gini.fit(X_train, y_train))
```

Out[22]:
```
[Text(0.4711538461538462, 0.9545454545454546, 'X[11] <= 0.014\ngini = 0.814\nsamples = 604\nvalue = [57
5, 149, 75, 44, 29, 18, 12, 8, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]'),
 Text(0.2, 0.8636363636363636, 'X[11] <= 0.012\ngini = 0.572\nsamples = 157\nvalue = [57, 5, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.1076923076923077, 0.7727272727272727, '\nX[2] <= 2.510\ngini = 0.357\nsamples = 111\nvalue =
[76, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.6818181818181818, '\nX[0] <= 0.000\ngini = 0.219\nsamples = 24\nvalue = [18, 2, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.5909090909090909, '\nX[2] <= 2.510\ngini = 0.095\nsamples = 19\nvalue = [18, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.5000000000000000, '\nX[2] <= 1.510\ngini = 0.053\nsamples = 18\nvalue = [18, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.4090909090909091, '\nX[2] <= 0.510\ngini = 0.032\nsamples = 17\nvalue = [16, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.3181818181818182, '\nX[0] <= 0.000\ngini = 0.500\nsamples = 2\nvalue = [1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.2272727272727273, '\nX[2] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.1363636363636364, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0454545454545454, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0, '\nX[2] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.2272727272727273, '\nX[2] <= 0.510\ngini = 0.500\nsamples = 2\nvalue = [1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.1363636363636364, '\nX[2] <= 0.510\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0454545454545454, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0, '\nX[2] <= 0.510\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.2153846153846154, 0.7727272727272727, '\nX[2] <= 3.510\ngini = 0.494\nsamples = 46\nvalue = [
0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.1538461538461538, 0.6818181818181818, '\nX[2] <= 3.010\ngini = 0.444\nsamples = 6\nvalue = [0,
4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.1076923076923077, 0.5909090909090909, '\nX[2] <= 3.010\ngini = 0.375\nsamples = 4\nvalue = [0,
3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.5000000000000000, '\nX[2] <= 1.010\ngini = 0.250\nsamples = 3\nvalue = [0, 2, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.4090909090909091, '\nX[0] <= 0.000\ngini = 0.500\nsamples = 2\nvalue = [0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.3181818181818182, '\nX[2] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.2272727272727273, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.1363636363636364, '\nX[2] <= 2.010\ngini = 0.667\nsamples = 3\nvalue = [0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0454545454545454, '\nX[0] <= 0.000\ngini = 0.500\nsamples = 2\nvalue = [0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0, '\nX[2] <= 2.010\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0454545454545454, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0, '\nX[2] <= 2.010\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.3307692307692308, 0.8636363636363636, '\nX[2] <= 3.510\ngini = 0.814\nsamples = 447\nvalue =
[499, 138, 75, 44, 29, 18, 12, 8, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]'),
 Text(0.2307692307692308, 0.7727272727272727, '\nX[2] <= 2.510\ngini = 0.811\nsamples = 395\nvalue =
[441, 132, 75, 44, 29, 18, 12, 8, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]'),
 Text(0.1846153846153846, 0.6818181818181818, '\nX[2] <= 1.510\ngini = 0.808\nsamples = 377\nvalue =
[421, 126, 71, 42, 28, 17, 12, 8, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]'),
 Text(0.1538461538461538, 0.5909090909090909, '\nX[2] <= 0.510\ngini = 0.804\nsamples = 359\nvalue =
[401, 120, 67, 40, 27, 16, 12, 8, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]'),
 Text(0.1230769230769231, 0.5000000000000000, '\nX[0] <= 0.000\ngini = 0.799\nsamples = 341\nvalue =
[381, 114, 63, 38, 26, 15, 12, 8, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]'),
 Text(0.0923076923076923, 0.4090909090909091, '\nX[2] <= 0.010\ngini = 0.794\nsamples = 324\nvalue =
[362, 108, 60, 36, 25, 14, 12, 8, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]'),
 Text(0.0615384615384615, 0.3181818181818182, '\nX[2] <= 0.010\ngini = 0.789\nsamples = 308\nvalue =
[344, 102, 57, 34, 24, 13, 12, 8, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]'),
 Text(0.0307692307692308, 0.2272727272727273, '\nX[0] <= 0.000\ngini = 0.784\nsamples = 293\nvalue =
[327, 96, 54, 32, 23, 12, 12, 8, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]'),
 Text(0.0, 0.1363636363636364, '\nX[2] <= 0.010\ngini = 0.779\nsamples = 279\nvalue = [311, 90, 51, 30,
22, 12, 12, 8, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]'),
 Text(0.0, 0.0454545454545454, '\nX[0] <= 0.000\ngini = 0.774\nsamples = 266\nvalue = [296, 84, 48, 28,
21, 12, 12, 8, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]'),
 Text(0.0, 0.0, '\nX[2] <= 0.010\ngini = 0.769\nsamples = 254\nvalue = [282, 78, 45, 26, 20, 12, 12, 8,
6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]'),
 Text(0.0615384615384615, 0.2272727272727273, '\nX[0] <= 0.000\ngini = 0.764\nsamples = 15\nvalue =
[16, 6, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0307692307692308, 0.1363636363636364, '\nX[2] <= 0.010\ngini = 0.750\nsamples = 12\nvalue =
[12, 6, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0454545454545454, '\nX[0] <= 0.000\ngini = 0.722\nsamples = 9\nvalue = [9, 4, 2, 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0, '\nX[2] <= 0.010\ngini = 0.667\nsamples = 3\nvalue = [3, 2, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0307692307692308, 0.0454545454545454, '\nX[0] <= 0.000\ngini = 0.500\nsamples = 2\nvalue = [1,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0, '\nX[2] <= 0.010\ngini = 0.000\nsamples = 1\nvalue = [2, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0454545454545454, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0, '\nX[2] <= 0.010\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.1538461538461538, 0.6818181818181818, '\nX[2] <= 2.510\ngini = 0.444\nsamples = 18\nvalue =
[20, 6, 4, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.1230769230769231, 0.5909090909090909, '\nX[0] <= 0.000\ngini = 0.375\nsamples = 12\nvalue =
[18, 0, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0923076923076923, 0.5000000000000000, '\nX[2] <= 1.510\ngini = 0.250\nsamples = 9\nvalue = [12,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0615384615384615, 0.4090909090909091, '\nX[0] <= 0.000\ngini = 0.500\nsamples = 2\nvalue = [1,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0307692307692308, 0.3181818181818182, '\nX[2] <= 0.510\ngini = 0.000\nsamples = 1\nvalue = [1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.2272727272727273, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.1363636363636364, '\nX[2] <= 0.510\ngini = 0.000\nsamples = 1\nvalue = [1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0454545454545454, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0, '\nX[2] <= 0.510\ngini = 0.000\nsamples = 1\nvalue = [1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0307692307692308, 0.2272727272727273, '\nX[2] <= 1.510\ngini = 0.667\nsamples = 3\nvalue = [2,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.1363636363636364, '\nX[0] <= 0.000\ngini = 0.500\nsamples = 2\nvalue = [1, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0454545454545454, '\nX[2] <= 1.510\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0, '\nX[2] <= 1.510\ngini = 0.000\nsamples = 1\nvalue = [1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0307692307692308, 0.0454545454545454, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0, 0.0, '\nX[2] <= 1.510\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.3846153846153846, 0.7727272727272727, '\nX[2] <= 4.510\ngini = 0.000\nsamples = 52\nvalue =
[58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.3076923076923077, 0.6818181818181818, '\nX[2] <= 3.510\ngini = 0.000\nsamples = 48\nvalue =
[54, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.2769230769230769, 0.5909090909090909, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 47\nvalue =
[54, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.2461538461538462, 0.5000000000000000, '\nX[2] <= 2.510\ngini = 0.000\nsamples = 46\nvalue =
[54, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.2153846153846154, 0.4090909090909091, '\nX[2] <= 1.510\ngini = 0.000\nsamples = 44\nvalue =
[52, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.1846153846153846, 0.3181818181818182, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 43\nvalue =
[52, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.1538461538461538, 0.2272727272727273, '\nX[2] <= 0.510\ngini = 0.000\nsamples = 42\nvalue =
[52, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.1230769230769231, 0.1363636363636364, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 41\nvalue =
[52, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0923076923076923, 0.0454545454545454, '\nX[2] <= 0.510\ngini = 0.000\nsamples = 40\nvalue =
[52, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0615384615384615, 0.0, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 39\nvalue = [52, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.0923076923076923, 0.0, '\nX[2] <= 0.510\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.1846153846153846, 0.1363636363636364, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.1538461538461538, 0.0454545454545454, '\nX[2] <= 0.510\ngini = 0.000\nsamples = 1\nvalue =
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.1230769230769231, 0.0, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.4615384615384615, 0.6818181818181818, '\nX[2] <= 4.510\ngini = 0.444\nsamples = 4\nvalue =
[4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.4307692307692308, 0.5909090909090909, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 3\nvalue =
[3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.4000000000000000, 0.5000000000000000, '\nX[2] <= 3.510\ngini = 0.000\nsamples = 2\nvalue =
[2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.3692307692307692, 0.4090909090909091, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue =
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.3384615384615385, 0.3181818181818182, '\nX[2] <= 2.510\ngini = 0.000\nsamples = 1\nvalue =
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.3076923076923077, 0.2272727272727273, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue =
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.2769230769230769, 0.1363636363636364, '\nX[2] <= 1.510\ngini = 0.000\nsamples = 1\nvalue =
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.2461538461538462, 0.0454545454545454, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue =
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.2153846153846154, 0.0, '\nX[2] <= 1.510\ngini = 0.000\nsamples = 1\nvalue = [1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.3384615384615385, 0.3181818181818182, '\nX[2] <= 3.510\ngini = 0.000\nsamples = 2\nvalue =
[2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.3076923076923077, 0.2272727272727273, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue =
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.2769230769230769, 0.1363636363636364, '\nX[2] <= 4.510\ngini = 0.000\nsamples = 1\nvalue =
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.2461538461538462, 0.0454545454545454, '\nX[0] <= 0.000\ngini = 0.000\nsamples = 1\nvalue =
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(0.2153846153846154, 0.0, '\nX[2] <= 4.510\ngini = 0.000\nsamples = 1\nvalue = [1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]')]
```



K Nearest Neighbour

Import Packages

In [23]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns, matplotlib.pyplot as plt
import numpy as np
import sklearn.neighbors
from sklearn.model_selection import train_test_split
```

Getting Columns for Train Dataset

In [24]:
```
X = df[['TPA', 'APG%', 'Total A%', 'DRB%', 'ORB%', 'TRB%',
        'AST%', 'TOV%', 'STL%', 'BLK%', 'USG%', 'PPR', 'PPS',
        'ORtg', 'DRtg', 'aDOR', 'FIC', 'PER', 'OP', 'DP', 'Group']].values
```

In [25]: