Unsupervised ML - Assignment:

Can we cluster NBA player performance to match clusters of Advanced Statistical Output?

Objective

The purpose of this report, and overall study, is to experiment, checking to see if it is possible to cluster NBA Player performance using John Hollinger's NBA individual advanced statistics (2002-03 to 2017-18 seasons). To measure performance or 'accuracy' of the clustering algorithm, its results will be compared to a 'truth source' which will be based on quartiles of Estimated Wins Added (EWA) per season in the statistics. Based on raw numbers, player EWA will be grouped into 4 different classes, basically representing quartiles and we will see if the clusters that player performance are separated into do a good job of agreeing with the EWA classifications.

A possible business implementation of this model could be for individual performance measurement and improvement – zoning in, using advanced statistics, to see what could make more valuable players similar to one another, and directing training and improvement.

The Data

The data used for the analysis is taken from a Kaggle dataset, that are processed (using Pandas) and then merged. It is a singe csv file that summarizes John Hollinger's advanced NBA statistics from the 2002 – 03 to the 2017 – 18 seasons. The data set is already cleaned, containing no zero, N/A etc data and contains 5404 row entries. Column entries include:

- Rank: that player's PER rank for that given season
- ts%: True Shooting Percentage - what a player's shooting percentage would be if we accounted for free throws and 3-pointers. True Shooting Percentage = Total points / [(FGA + (0.44 x FTA)]
- ast: Assist Ratio - the percentage of a player's possessions that ends in an assist. Assist Ratio = (Assists x 100) divided by [(FGA + (FTA x 0.44) + Assists + Turnovers]
- to: Turnover Ratio - the percentage of a player's possessions that end in a turnover. Turnover Ratio = (Turnover x 100) divided by [(FGA + (FTA x 0.44) + Assists + Turnovers]
- usg Usage Rate - the number of possessions a player uses per 40 minutes. Usage Rate = {[FGA + (FT Att. x 0.44) + (Ast x 0.33) + TO] x 40 x League Pace} divided by (Minutes x Team Pace)
- orr: Offensive rebound rate
- drr: Defensive rebound rate
- rebr: Rebound Rate - the percentage of missed shots that a player rebounds. Rebound Rate = (100 x (Rebounds x Team Minutes)) divided by [Player Minutes x (Team Rebounds + Opponent Rebounds)]
- per: Player Efficiency Rating is the overall rating of a player's per-minute statistical production. The league average is 15.00 every season.
- va: Value Added - the estimated number of points a player adds to a teamâ€™s season total above what a 'replacement player' (for instance, the 12th man on the roster) would produce. Value Added = ([Minutes * (PER - PRL)] / 67). PRL (Position Replacement Level)

= 11.5 for power forwards, 11.0 for point guards, 10.6 for centers, 10.5 for shooting guards and small forwards

- ewa: Estimated Wins Added - Value Added divided by 30, giving the estimated number of wins a player adds to a team's season total above what a 'replacement player' would produce.

Outlier data was removed for analysis, but only on the low end of games played (gp) – we reoved rows referencing season where gp was below the first quartile – 1.5*IQR, as we did not want the effects of long term injury and short term (ie: 10 day) contracts to affect the analysis.

Given that the features per, va, and ewa are quite similar in that they describe a player's contribution, the va and per features were dropped for the analysis to avoid any effects of multicollinearity. Estimated Wins Added (EWA) was used as the sole summary statistic in the analysis – to note, it could have been either of the 3 (PER, VA, EWA), EWA was chosen for its simplicity and built-in scaling (EWA is a scaled down version of VA, which is calculated from PER). Additionally, to avoid the pitfalls of multicollinearity, orr and drr were removed, with rebounding rate used as the sole measure of rebounding performance.

The final data set, ready for analysis contained 5366 or the original 5404 rows of data – a very small trimming of low gp data

[11]:

	gp	mpg	ts%	ast	to	usg	orr	drr	rebr	ewa	player	season
0	75	39.4	0.564	15.0	7.1	32.6	5.0	14.6	9.5	15.1	Tracy McGrady	2002-03
1	67	37.8	0.602	10.7	10.1	27.8	11.0	21.6	16.5	11.9	Shaquille O'Neal	2002-03
2	82	41.5	0.550	16.0	9.6	31.1	3.0	15.3	9.3	14.0	Kobe Bryant	2002-03
3	82	40.5	0.553	20.4	9.4	25.0	9.0	28.5	18.8	12.4	Kevin Garnett	2002-03
4	81	39.3	0.564	14.1	11.1	25.7	10.0	27.3	19.0	11.8	Tim Duncan	2002-03
...	...	...	...	...	...	...	...	...	...	...	...	...
5399	69	14.2	0.417	8.4	11.0	9.1	4.6	13.9	9.1	-2.5	Josh Huestis	2017-18
5400	54	15.2	0.445	13.8	12.9	14.6	1.6	16.0	8.5	-2.2	Paul Zipser	2017-18
5401	48	10.9	0.439	11.5	15.0	16.2	2.8	12.0	7.4	-1.4	Abdel Nader	2017-18
5402	73	15.8	0.473	8.0	10.0	8.4	3.6	11.6	7.7	-4.0	Semi Ojeleye	2017-18
5403	52	12.3	0.507	11.2	9.3	25.5	0.0	0.0	0.0	-3.3	Sean Kilpatrick	2017-18

5366 rows × 12 columns

Exploratory Analysis

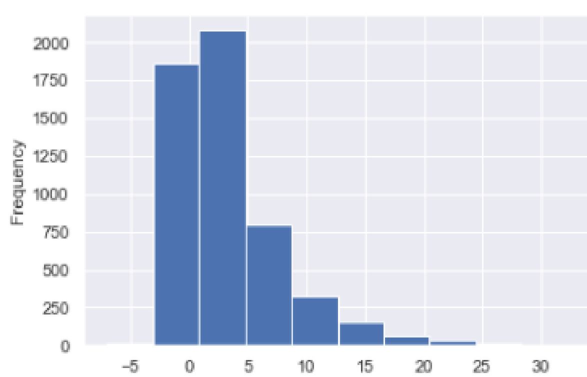
The first thing done was to get a description of the data. Since all the data was numeric, it is nice to look at the mean, median and different quartiles of the data, to see how it is spread out. Note that, by virtue of the sport, the ewa data will tend towards lognormal distribution: that is, few players contribute many estimated wins above replacement (ewa) – most of the NBA player performance hovers around average – this can be seen from the plot below. The decision was made to not remove positive (high performing) outliers, as that will likely be a cluster in its own right

[12]:

	count	mean	std	min	25%	50%	75%	max
gp	5366.0	66.425084	13.592727	26.000	58.000	70.000	78.000	85.000
mpg	5366.0	24.516735	7.965001	7.000	17.900	24.300	31.200	43.100
ts%	5366.0	0.531927	0.047927	0.338	0.502	0.532	0.563	0.725
ast	5366.0	16.008013	8.038026	1.200	10.000	14.100	20.600	48.700
to	5366.0	11.137029	2.881643	2.000	9.200	10.900	12.700	29.600
usg	5366.0	18.055535	4.736006	0.000	14.700	17.700	21.100	42.500
orr	5366.0	5.372736	3.845806	0.000	2.000	4.000	8.100	22.000
drr	5366.0	14.647372	5.811279	0.000	10.000	13.600	18.600	38.000
rebr	5366.0	10.009896	4.529315	0.000	6.200	8.950	13.300	26.700
ewa	5366.0	3.338595	4.623633	-7.000	0.200	2.000	5.000	32.300

[19]:
```
# plotting for numeric ewa
final_data['ewa'].plot.hist()
```

[19]: <AxesSubplot:ylabel='Frequency'>



Next, the ewa was placed into classes, based on its value relative to the various quartiles:

- less than 25th percentile: Class 1
- greater than or equal to 25th percentile, less than 50th percentile (median): Class 2
- greater than or equal to 50th percentile (median), less than 75th percentile: Class 3
- greater than 75th percentile: class 4

[13]:

	gp	mpg	ts%	ast	to	usg	orr	drr	rebr	ewa	player	season	ewa class
0	75	39.4	0.564	15.0	7.1	32.6	5.0	14.6	9.5	15.1	Tracy McGrady	2002-03	4
1	67	37.8	0.602	10.7	10.1	27.8	11.0	21.6	16.5	11.9	Shaquille O'Neal	2002-03	4
2	82	41.5	0.550	16.0	9.6	31.1	3.0	15.3	9.3	14.0	Kobe Bryant	2002-03	4
3	82	40.5	0.553	20.4	9.4	25.0	9.0	28.5	18.8	12.4	Kevin Garnett	2002-03	4
4	81	39.3	0.564	14.1	11.1	25.7	10.0	27.3	19.0	11.8	Tim Duncan	2002-03	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...
5399	69	14.2	0.417	8.4	11.0	9.1	4.6	13.9	9.1	-2.5	Josh Huestis	2017-18	1
5400	54	15.2	0.445	13.8	12.9	14.6	1.6	16.0	8.5	-2.2	Paul Zipser	2017-18	1
5401	48	10.9	0.439	11.5	15.0	16.2	2.8	12.0	7.4	-1.4	Abdel Nader	2017-18	1
5402	73	15.8	0.473	8.0	10.0	8.4	3.6	11.6	7.7	-4.0	Semi Ojeleye	2017-18	1
5403	52	12.3	0.507	11.2	9.3	25.5	0.0	0.0	0.0	-3.3	Sean Kilpatrick	2017-18	1

5366 rows × 13 columns

Binning the ewa data into classes helps even out the distribution of the data. As can be seen below, this did a great job of taking a dataset that is quite lognormally distributed and balancing out the data set.

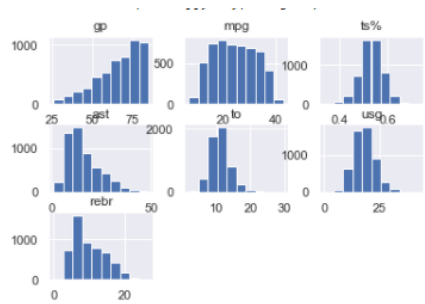
[20]:
```
# count on ordinal class feature (ewa class)
y = final_data[ycol]

# get a feature count
y_count = y.value_counts()
y_count_norm = y.value_counts(normalize = True)
print('Raw Counts: ', y_count)
print('Normalized: ', y_count_norm)
```

```
Raw Counts:  2    1417
4    1364
3    1338
1    1247
Name: ewa class, dtype: int64
Normalized:  2    0.264070
4    0.254193
3    0.249348
1    0.232389
Name: ewa class, dtype: float64
```

By binning/classifying data according to ewa quartiles we've been able to create a reasonably balanced data set from quite imbalanced ewa/performance data

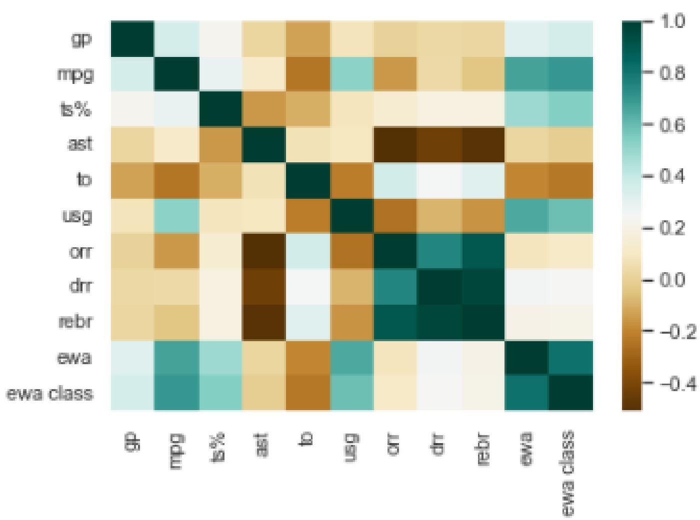
Next, we looked at the distribution of each of the features:



As seen, some features are close to nomally distributed, while others are not. We will not apply a Boxcox transformation to noralize the data, but will use MinMaxScaling

Note that each feature is distributed differently. The only way to get these distributed close to normally would be through a Boxcox transformation. However, a Boxcox transformation requires values of each feature to be positive, which would take away a lot of data from the feature set. Baased on this, it was decided to not transform the data and to use Min-Max scaling (as opposed to Standard Scaling) when the dataset required feature scaling.

Finally, a correlation heatmap was built, to show the correlations between features and the correlations between the features



It appears that the following individual statistics have the highest correlation with ewa:

- mpg
- ts%
- ast
- to
- usg

The data was then split, using a 70/30 train test split, and scaled using the sklearn MinMaxScaler. The shape of the data sets were as follows (note that we will be using only the x (features) for the analysis, as we will cluster, predict clusters on the test set, and then use the EWA class labels as a comparison to the analyzed clusters to see the clustering effectiveness):

```
[23]: from sklearn.model_selection import train_test_split

      x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.33, random_state = 42)

      print('x_train: ', x_train.shape)
      print('x_test: ', x_test.shape)
      print('y_train: ', y_train.shape)
      print('y_test: ', y_test.shape)
```

```
x_train:  (3595, 7)
x_test:  (1771, 7)
y_train:  (3595,)
y_test:  (1771,)
```

<u>Data Analysis</u>

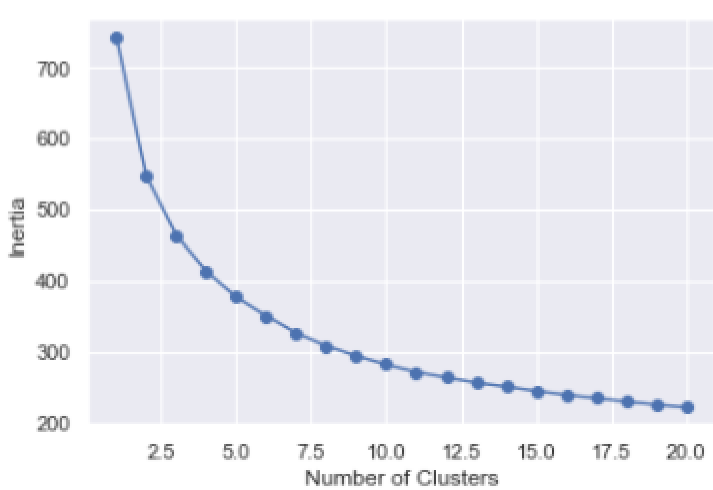
Note that for all clustering algorithms, the following methodology was applied:

- prepare data
- determine the number of clusters
- build model and predict clusters on unseen data
- review the data associated with cluster labels and theme/classify the clusters
- add to the data frame
- measure accuracy by looking at cluster vs ewa classes, looking at the total similar vs total records

The exception was with density-based scanning (DBSCAN), which was admittedly the worst performing of the clustering algorithms. Only test data was used here, as DBSCAN couldn't be used to predict on unseen data.

<u>K-Means Clustering</u>

The K-Means algorithm originally, to determine the best cluster number, was run on 1 – 20 clusters, with inertia used to determine a proper elbow using the elbow method. As per the following, it was difficult to determine this visually, as there as no clear elbow:



It was a little easier to determine numerically. Data was placed into a data frame, and the best number was picked when the rate of change of inertia slowed down. This could have been picked as 4 or 5 clusters – 4 was chosen to maintain symmetry with the EWA classes.

[26]:

N Clusters	Inertia	Change
1	743.338858	NaN
2	547.259074	-196.079784
3	464.247616	-83.011458
4	413.618217	-50.629399
5	377.648837	-35.969380
6	350.921349	-26.727488
7	326.918598	-24.002751
8	308.655867	-18.262731
9	294.581712	-14.074155
10	282.493296	-12.088416
11	271.865859	-10.627437
12	264.158497	-7.707362
13	256.662934	-7.495563
14	251.172931	-5.490003
15	245.249268	-5.923664
16	239.707935	-5.541332
17	235.018779	-4.689156
18	230.293405	-4.725374
19	225.825295	-4.468110
20	222.052265	-3.773031

The cluster labels were then analyzed within the rest of the data to determine a theme. Since there were 4 clusters, the themes were kept consistent with that of the EWA classes and were:

- Star Player
- Starter Level
- Replacement Level
- Below Replacement Level

A snapshot of the data frame follows, which is filtered to compare only the test data set predictions. To get a measure of cluster performance, the number of instance of EWA Label being equal to Cluster Name – KM was evaluated. This turned out to be 0.43, or 43% - not terribly impressive:

[51]:

	gp	mpg	ts%	ast	to	usg	orr	drr	rebr	ewa	player	season	ewa class	EWA Label	Cluster Label - KM	Cluster Name - KM
4847	80	28.7	0.530	12.5	7.4	19.1	5.3	14.1	9.6	3.4	Aaron Gordon	2016-17	3	Starter Level	0	Star Player
4414	52	34.4	0.528	17.4	10.5	22.9	3.5	20.2	11.5	6.4	Marc Gasol	2015-16	4	Star Player	0	Star Player
2095	81	34.0	0.565	22.2	14.4	14.6	5.3	13.8	9.6	4.1	Boris Diaw	2008-09	3	Starter Level	0	Star Player
1421	78	32.9	0.540	9.9	11.2	21.0	6.0	15.7	10.9	4.3	Al Harrington	2006-07	3	Starter Level	0	Star Player
1644	77	36.4	0.568	11.5	11.2	27.5	7.0	14.9	11.0	15.1	Carmelo Anthony	2007-08	4	Star Player	0	Star Player
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1492	75	23.2	0.513	12.9	15.9	9.2	16.0	24.1	20.2	2.1	Jeff Foster	2006-07	3	Starter Level	3	Replacement Level
1199	65	15.7	0.491	26.3	12.5	17.0	3.0	14.7	8.7	0.7	Toni Kukoc	2005-06	2	Replacement Level	1	Starter Level
329	67	40.1	0.492	22.0	9.5	29.2	3.0	9.6	6.1	7.1	Baron Davis	2003-04	4	Star Player	0	Star Player
653	81	37.0	0.554	11.2	9.9	23.3	12.0	18.4	15.4	7.9	Elton Brand	2004-05	4	Star Player	0	Star Player
3805	79	25.0	0.578	23.0	12.6	16.4	4.4	13.8	9.3	2.6	Boris Diaw	2013-14	3	Starter Level	1	Starter Level

1771 rows × 16 columns

[53]:

```
# accuracy level measure: want to check and see if the clustering algorithm is placing the players into the same clusters/buckets are the ewa
# compare total number of rows in df1
# To the number of rows where the ewa class labela nd cluster label match
df2 = df1[df1['EWA Label'] == df1['Cluster Name - KM']]
print('Accuracy of clustering vs EWA Classification: ', round(df2.shape[0]/df1.shape[0], 2))
```

```
Accuracy of clustering vs EWA Classification:  0.43
```

Hierarchical Agglomerative Clustering

Next, a Hierarchical Agglomerative Clustering algorithm was used. The decision was made to use 4 clusters (to keep things consistent) with Ward Linkage used, to minimize inertia between pairs. Predictions we're run and the cluster labels were placed into the same test data frame:

[55]:

	gp	mpg	ts%	ast	to	usg	orr	drr	rebr	ewa	player	season	ewa class	EWA Label	Cluster Label - KM	Cluster Name - KM	Agglom Clusters
4847	80	28.7	0.530	12.5	7.4	19.1	5.3	14.1	9.6	3.4	Aaron Gordon	2016-17	3	Starter Level	0	Star Player	1
4414	52	34.4	0.528	17.4	10.5	22.9	3.5	20.2	11.5	6.4	Marc Gasol	2015-16	4	Star Player	0	Star Player	3
2095	81	34.0	0.565	22.2	14.4	14.6	5.3	13.8	9.6	4.1	Boris Diaw	2008-09	3	Starter Level	0	Star Player	3
1421	78	32.9	0.540	9.9	11.2	21.0	6.0	15.7	10.9	4.3	Al Harrington	2006-07	3	Starter Level	0	Star Player	1
1644	77	36.4	0.568	11.5	11.2	27.5	7.0	14.9	11.0	15.1	Carmelo Anthony	2007-08	4	Star Player	0	Star Player	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1492	75	23.2	0.513	12.9	15.9	9.2	16.0	24.1	20.2	2.1	Jeff Foster	2006-07	3	Starter Level	3	Replacement Level	1
1199	65	15.7	0.491	26.3	12.5	17.0	3.0	14.7	8.7	0.7	Toni Kukoc	2005-06	2	Replacement Level	1	Starter Level	2
329	67	40.1	0.492	22.0	9.5	29.2	3.0	9.6	6.1	7.1	Baron Davis	2003-04	4	Star Player	0	Star Player	3
653	81	37.0	0.554	11.2	9.9	23.3	12.0	18.4	15.4	7.9	Elton Brand	2004-05	4	Star Player	0	Star Player	3
3805	79	25.0	0.578	23.0	12.6	16.4	4.4	13.8	9.3	2.6	Boris Diaw	2013-14	3	Starter Level	1	Starter Level	1

1771 rows × 17 columns

Like the K-Means analysis, each cluster label was analyzed/themed, with the theme names matching those of the K-Means analysis. The Agglomerative Algorithm yielded different results, on the record level. However, its overall accuracy was the same:

[65]:

	gp	mpg	ts%	ast	to	usg	orr	drr	rebr	ewa	player	season	ewa class	EWA Label	Cluster Label - KM	Cluster Name - KM	Agglom Clusters	Agg Cluster Name
4847	80	28.7	0.530	12.5	7.4	19.1	5.3	14.1	9.6	3.4	Aaron Gordon	2016-17	3	Starter Level	0	Star Player	1	Replacement Level
4414	52	34.4	0.528	17.4	10.5	22.9	3.5	20.2	11.5	6.4	Marc Gasol	2015-16	4	Star Player	0	Star Player	3	Star Player
2095	81	34.0	0.565	22.2	14.4	14.6	5.3	13.8	9.6	4.1	Boris Diaw	2008-09	3	Starter Level	0	Star Player	3	Star Player
1421	78	32.9	0.540	9.9	11.2	21.0	6.0	15.7	10.9	4.3	Al Harrington	2006-07	3	Starter Level	0	Star Player	1	Replacement Level
1644	77	36.4	0.568	11.5	11.2	27.5	7.0	14.9	11.0	15.1	Carmelo Anthony	2007-08	4	Star Player	0	Star Player	3	Star Player
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1492	75	23.2	0.513	12.9	15.9	9.2	16.0	24.1	20.2	2.1	Jeff Foster	2006-07	3	Starter Level	3	Replacement Level	1	Replacement Level
1199	65	15.7	0.491	26.3	12.5	17.0	3.0	14.7	8.7	0.7	Toni Kukoc	2005-06	2	Replacement Level	1	Starter Level	2	Starter Level
329	67	40.1	0.492	22.0	9.5	29.2	3.0	9.6	6.1	7.1	Baron Davis	2003-04	4	Star Player	0	Star Player	3	Star Player
653	81	37.0	0.554	11.2	9.9	23.3	12.0	18.4	15.4	7.9	Elton Brand	2004-05	4	Star Player	0	Star Player	3	Star Player
3805	79	25.0	0.578	23.0	12.6	16.4	4.4	13.8	9.3	2.6	Boris Diaw	2013-14	3	Starter Level	1	Starter Level	1	Replacement Level

1771 rows × 18 columns

[67]:

```
# accuracy level measure: want to check and see if the clustering algorithm is placing the players into the same clusters/buckets are the ewa
# compare total number of rows in df1
# To the number of rows where the ewa class labela nd cluster label match
df3 = df1[df1['EWA Label'] == df1['Agg Cluster Name']]
print('Accuracy of clustering vs EWA Classification: ', round(df3.shape[0]/df1.shape[0], 2))
```

```
Accuracy of clustering vs EWA Classification:  0.43
```

<u>Density Based Scanning (DBSCAN)</u>

The density-based scanning (DBSCAN) algorithm was run, with some differences from the previous 2 algorithms:

- First, since the DBSCAN algorithm can’t be run to predict values on unseen data, it was run only of the test data set, to maintain size symmetry with the other analysis
- However, the train data set as used to determine the optimum value of epsilon, as seen below

The optimum value for epsilon for this dataset was determined by running a nearest neighbors analysis on the training data set (larger set). The results can be seen below – the optimum value for epsilon was chosen at the max point of curvature on the plot. In this case, it was a value of eps = 0.20

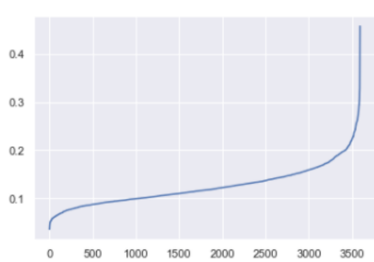
[119]:

```
# will work to define an epsilon. For consistency, and since there is no train/test split applied on DBSCAN, will use only the x_test_scaled dataset
# will use x_train_scaled to determine optimum epsilon
from sklearn.neighbors import NearestNeighbors
from sklearn.cluster import DBSCAN

neigh = NearestNeighbors(n_neighbors=5)
nbrs = neigh.fit(x_train_scaled)
distances, indices = nbrs.kneighbors(x_train_scaled)

distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.plot(distances)
```

[119]: [<matplotlib.lines.Line2D at 0x2113e9f93c8>]



Then the DBSCAN algorithm was run with epsilon = 0.20. The num_samples parameter was manually changed until 4 clusters were again generated (min_samples = 4). Based on this, the DBSCAN algorithm yielded some very strange results – most of the data points are lumped together, likely a result of the relatively closeness of data points (remember the comment of most performance in the NBA being quite similar) – even the difference from Star to Replacement Level doesn't show up as a large difference in advanced statistics.

Once again, the clusters were themed, placed in a data frame, and compared. This time, cluster performance was poor, with essentially zero accuracy:

[145]:

	gp	mpg	ts%	ast	to	usg	orr	drr	rebr	ewa	player	season	ewa class	EWA Label	Cluster Label - KM	Cluster Name - KM	Agglom Clusters	Agg Cluster Name	DBSCAN Clusters	DB Cluster Name
4847	80	28.7	0.530	12.5	7.4	19.1	5.3	14.1	9.6	3.4	Aaron Gordon	2016-17	3	Starter Level	0	Star Player	1	Replacement Level	0	Average Player
4414	52	34.4	0.528	17.4	10.5	22.9	3.5	20.2	11.5	6.4	Marc Gasol	2015-16	4	Star Player	0	Star Player	3	Star Player	0	Average Player
2095	81	34.0	0.565	22.2	14.4	14.6	5.3	13.8	9.6	4.1	Boris Diaw	2008-09	3	Starter Level	0	Star Player	3	Star Player	0	Average Player
1421	78	32.9	0.540	9.9	11.2	21.0	6.0	15.7	10.9	4.3	Al Harrington	2006-07	3	Starter Level	0	Star Player	1	Replacement Level	0	Average Player
1644	77	36.4	0.568	11.5	11.2	27.5	7.0	14.9	11.0	15.1	Carmelo Anthony	2007-08	4	Star Player	0	Star Player	3	Star Player	0	Average Player
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1492	75	23.2	0.513	12.9	15.9	9.2	16.0	24.1	20.2	2.1	Jeff Foster	2006-07	3	Starter Level	3	Replacement Level	1	Replacement Level	0	Average Player
1199	65	15.7	0.491	26.3	12.5	17.0	3.0	14.7	8.7	0.7	Toni Kukoc	2005-06	2	Replacement Level	1	Starter Level	2	Starter Level	0	Average Player
329	67	40.1	0.492	22.0	9.5	29.2	3.0	9.6	6.1	7.1	Baron Davis	2003-04	4	Star Player	0	Star Player	3	Star Player	0	Average Player
653	81	37.0	0.554	11.2	9.9	23.3	12.0	18.4	15.4	7.9	Elton Brand	2004-05	4	Star Player	0	Star Player	3	Star Player	0	Average Player
3805	79	25.0	0.578	23.0	12.6	16.4	4.4	13.8	9.3	2.6	Boris Diaw	2013-14	3	Starter Level	1	Starter Level	1	Replacement Level	0	Average Player

1771 rows × 20 columns

[146]:

```
# accuracy level measure: want to check and see if the clustering algorithm is placing the players into the same clusters/buckets are the ewa
# The DBSCAN algorithm was definitely the least 'accurate' of the clustering algorithms. I beleive this for the following reasons
# 1) I beleive the DBSCAN algorithm is confued by the relative 'closeness' of many data points
# 2) This particular data set does much better with a set point (centroid) to which measure clusters around

# compare total number of rows in df1
# To the number of rows where the ewa class labela nd cluster label match
df4 = df1[df1['EWA Label'] == df1['DB Cluster Name']]
print('Accuracy of clustering vs EWA Classification: ', round(df4.shape[0]/df1.shape[0], 2))
```

```
Accuracy of clustering vs EWA Classification:  0.0
```

Key Findings

As briefly mentioned during the review of each model, the algorithms do not do a great job of creating clusters that match the EWA classes that were created in the pre-processing step. I believe there are a couple of reasons for this:

- The ;'hard and set' clustering algorithms that measure distance from central points, seem to do a much better job of clustering player performance, although it certainly isn't all that good
- The performance of many players in the NBA is essentially interchangeable, and many of the statistical measures have tiny distance differences. This could potentially confuse a clustering algorithm. I believe this was especially evident when attempting to use DBSCAN
- Estimated Wins above Replacement (ewa) can be calculated by many means (we don't have access to the formula used to derive it from per). Again, another thing that can potentially throw off a clustering algorithm, especially when trying to compare it to something like an EWA Class.
- This sort of analysis again, may be better accomplished via counting statistics, rather than advanced (rate-based) statistics

For this reason, I'd say that attempting this sort of analysis, at least without access to additional data, is not recommended for player performance tuning/training etc.

Further Analysis

As described above, the analysis is quite complex, as there are several moving parts. The analysis shows a little insight, but in general does a poor job of comparing clusters based on advanced statistical outputs.

Some items that will likely do the most for adding insight would be:

- Performing the analysis based on counting stats, rather than advanced (rate based) stats
- Another clustering analysis, this time perhaps using a Principal Component Analysis for feature selection/elimination
- Cluster Performance: the original clusters were compared to an EWA class to get a loose determination of accuracy, or whether the cluster is good or not. Perhaps another measure of cluster effectiveness is required

Some challenges that may be encountered in trying to improve this analysis:

- Data volume: there is only so much data available in this space, and 5000 records may be too little to train something of this complexity
- Cluster performance: finding a better measure of performance in clustering
- The fac that this, in general, just may not be a good application