Project Report

1. Basic Statistical Analysis and data cleaning insight

1.1 Mean, Median and standard

Dataset	Attributes used for calculating Mean, Median and StD
player_stats	G, GS, MP, eFG%, ORB, DRB, TRB, AST, STL, BLK, TOV, PTS
player_career_stats	G, GS, MP, eFG%, ORB, DRB, TRB, AST, STL, BLK, TOV, PTS
Salary	salary

- player_stats

```
avg of G is:49.01
median of G is:54.0
std of G is:26.76
-------
avg of GS is:23.43
median of GS is:8.0
std of GS is:28.56
-------
avg of MP is:19.85
median of MP is:19.0
std of MP is:10.2
-------
avg of eFG% is:0.46
median of eFG% is:0.48
std of eFG% is:0.11
-------
avg of ORB is:1.01
median of ORB is:0.8
std of ORB is:0.86
-------
avg of DRB is:2.45
median of DRB is:2.0
std of DRB is:1.79
-------
avg of TRB is:3.46
median of TRB is:2.8
std of TRB is:2.53
-------
avg of AST is:1.83
median of AST is:1.2
std of AST is:1.84
-------
avg of STL is:0.65
median of STL is:0.6
std of STL is:0.48
-------
avg of BLK is:0.4
median of BLK is:0.2
std of BLK is:0.5
-------
avg of TOV is:1.23
median of TOV is:1.0
std of TOV is:0.82
-------
avg of PTS is:8.03
median of PTS is:6.5
std of PTS is:5.96
-------
```

- player_career_stats

```
avg of G is:322.49
median of G is:179.0
std of G is:336.3
-------
avg of GS is:143.9
median of GS is:25.0
std of GS is:236.56
-------
avg of MP is:16.68
median of MP is:15.8
std of MP is:8.86
-------
avg of eFG% is:0.46
median of eFG% is:0.47
std of eFG% is:0.11
-------
avg of ORB is:0.87
median of ORB is:0.7
std of ORB is:0.71
-------
avg of DRB is:2.03
median of DRB is:1.7
std of DRB is:1.5
-------
avg of TRB is:2.91
median of TRB is:2.4
std of TRB is:2.13
-------
avg of AST is:1.47
median of AST is:1.0
std of AST is:1.46
-------
avg of STL is:0.56
median of STL is:0.5
std of STL is:0.4
-------
avg of BLK is:0.33
median of BLK is:0.2
std of BLK is:0.4
-------
avg of TOV is:1.04
median of TOV is:0.9
std of TOV is:0.68
-------
avg of PTS is:6.54
median of PTS is:5.3
std of PTS is:4.86
-------
```

- Salary

```
avg of Salary is:3273007.0
median of Salary is:1514700.0
std of Salary is:4409075.58
-------
```

1.2 Data Normalization

> In our data set 'player stats', we have 31 attributes varying in units and scales.
>
> eg: The attribute FG%(Field Goals Percentage) is a percentage number ranges from 0 to 1.
>
> The attribute PTS(Points Per Game) having the unit "points" and ranges from 0 to 37.1.
>
> If we use the raw data, the PTS having a larger scale will have more weight than others when doing analysis. So, to clear these in-balanced factors, we should normalize the data via z-score before using them. Data after being z-scored will have the same distribution as the original data.

1.3 Data cleaning

> Salary dataset: Remove the '$' symbol on the 'salary' attribute.
>
> player_stats dataset: remove 2nd position on 'Pos' attribute.

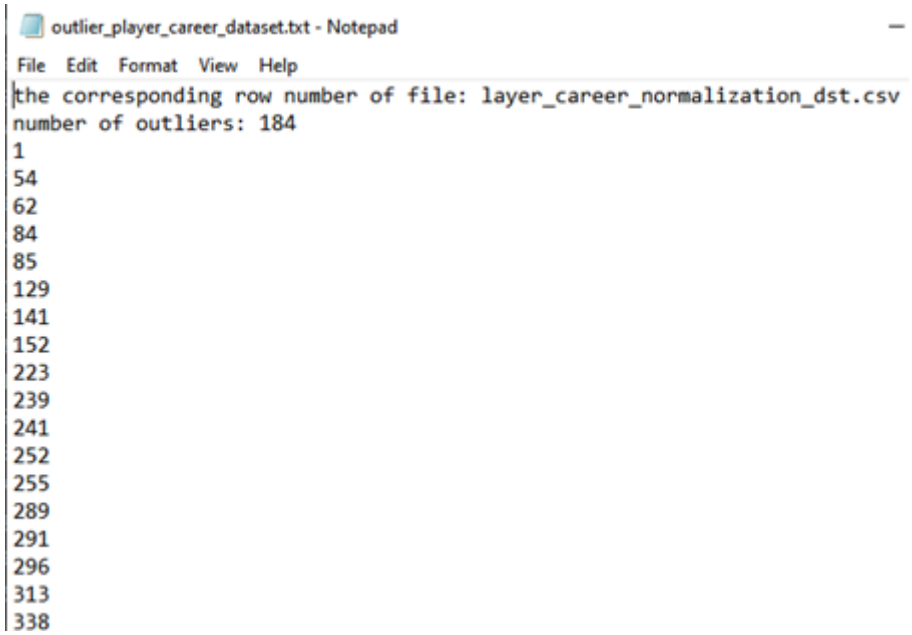
1.4 Binning data

> The records in dataset 'player_career_stats' are binned into 5 bins. A clear attribute we can use to evaluate the importance of one player in the NBA is how many games he played. Influential players like Michael Jordan, Karl Malone, and Kobe Bryant all played more than 1,000 games in their careers.
>
> Intuitively, we can classify them via:

Value (Game Played)	Class
[0, 323) approx.: 4 seasons	Newbie
[323, 645) approx.: 8 seasons	Sophomore
[645, 967) approx.: 12 seasons	Senior
[967, 1289) approx.: 16 seasons	Experienced
[1289, 1611) approx.: 20 seasons	Veteran

1.5 Additional Part for CS Students

> Tying 3 different k-value: 200, 100 and 50.
>
> By cross-checking, 184 outliers are found.



2. Histograms and Correlations

2.1 Histogram

> In this part, we try to find the relation between the player's position and their key indices of performance.
>
> a. eFG% (Effective Field Goal Percentage)

We are surprised to find that there is no relation between a players' position and their eFG%.

b. TRB (Total Rebounds Per Game)

No surprise. Centers have the best performance followed by Power Forward. And Point Guard and Shooting Guard who mainly focus on outside of the paint area.

c. AST (Assistant Per Game)

No surprise. Point Guards, the players leading the offense of games rank first.

d. BLK (Block Per Game)

No surprise. Centers dominate the paint area.

e. TOV (Turnover Per Game)

No surprise. Point Guards, players having the highest possession time of the ball, have a higher risk to make a mistake.

f. PF (Personal Foul Per Game)

No surprise. Centers and Power Forward who have a higher pressure on the defense incline to foul than others.

g. PTS (Points Per Game)

The order of PTS reflects the priorities of players' role in games. No surprise.

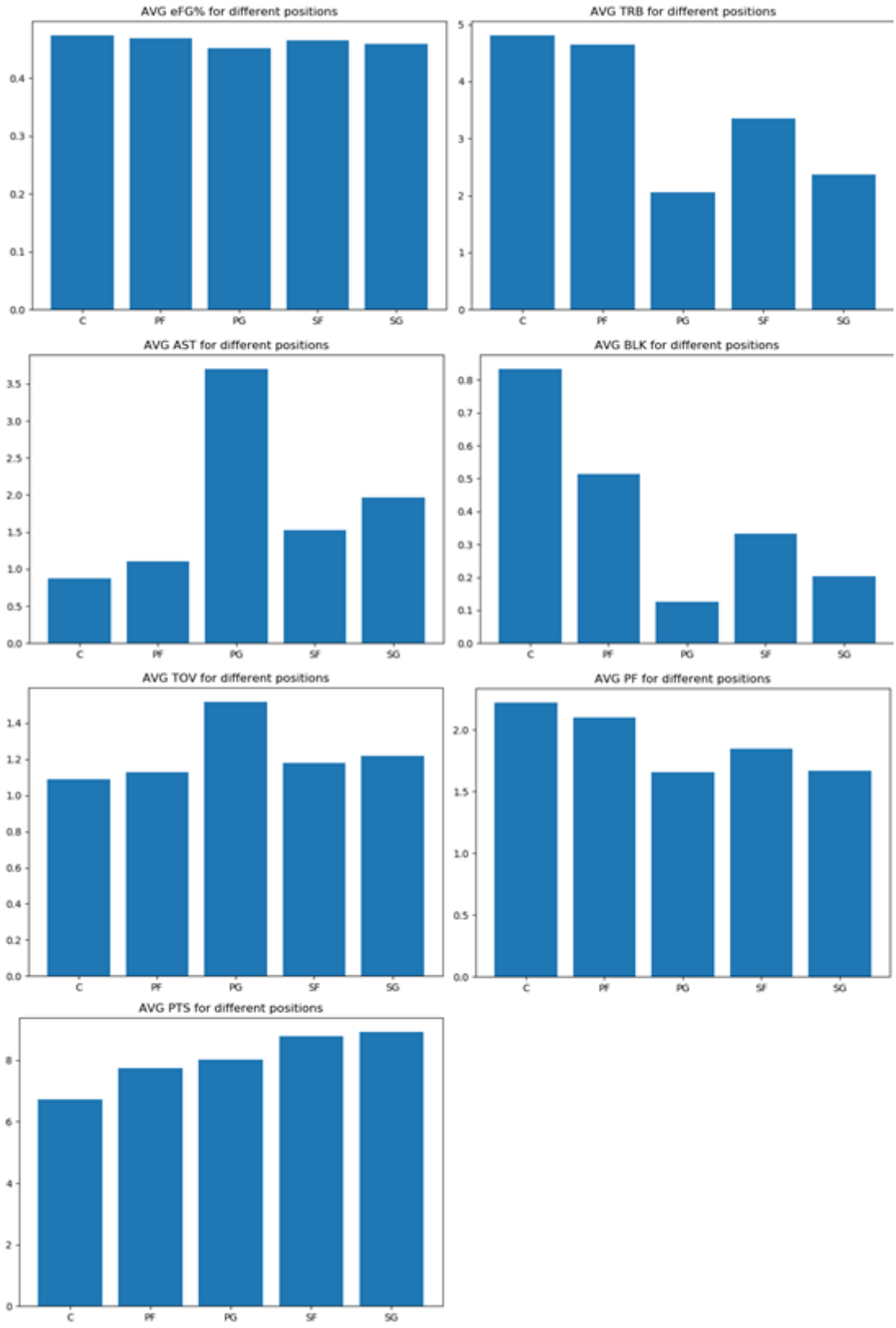


Fig 2.1. Key indices of performance

2.2 Correlations

The table(Fig. 2.2) and two scatter plots show the correlation among 'Season', 'Average Salary' and 'Record count for the season. From table and plots, we can find:

a. Salary variation:

In the following seasons, compared with last season, the average player's salary decreased.

1985~1986, 1987-1988, 1990-91, 2006-07, 2009-10, 2010-11, 2011-12, 2012-13, 2014-15

b. Outliers:

From the 1st scatter plot(Fig. 2.3), 1989-1990 looks like a crazy season that the average salary three times the last season's. But actually, from the 2nd scatter plot(Fig. 2.4), data for 1989-90 season is an outlier, since this season only has 64 records that are far less than the number of records in other seasons and this season has 27 teams, so it does not make sense to have only 64 records.

For the season 1986-87, the data is another outlier. That season has 23 teams in total. So, having 40-records is not close to accurate.

Same for 1984-1985, 23 teams, 206 records. It's clearly not valid data.

c. NBA economy recession:

Reviewing the 1st scatter plot(Fig. 2.3), from 2009 to 2015, 5 seasons' decreasing in a row shows the longest continuous recession of the NBA since 1984.

C:\Users\alex\Desktop\course_projects\Dat

	Season	Avg Salary	Record Count
0	1984-85	400553.398	206
1	1985-86	370103.395	296
2	1986-87	543033.325	40
3	1987-88	459204.314	303
4	1988-89	528010.975	321
5	1989-90	1670937.500	64
6	1990-91	831623.229	353
7	1991-92	954346.253	387
8	1992-93	1061757.871	404
9	1993-94	1269884.175	394
10	1994-95	1360403.911	418
11	1995-96	1734345.361	388
12	1996-97	1935172.397	413
13	1997-98	2121414.975	444
14	1998-99	2456729.141	432
15	1999-00	2498707.574	526
16	2000-01	3236010.808	464
17	2001-02	3395648.484	459
18	2002-03	3629012.967	460
19	2003-04	3650913.856	458
20	2004-05	3705739.554	478
21	2005-06	3839336.672	494
22	2006-07	3793015.517	511
23	2007-08	4244495.942	486
24	2008-09	4581769.446	471
25	2009-10	4476258.142	472
26	2010-11	4334884.212	467
27	2011-12	4280917.064	469
28	2012-13	4231989.181	497
29	2013-14	4855723.292	411
30	2014-15	3952878.206	554
31	2015-16	4299296.786	547
32	2016-17	5008969.863	590
33	2017-18	5811252.242	571
34	2018-19	6393584.830	566

Fig 2.2 Season, Avg Salary and Record Count

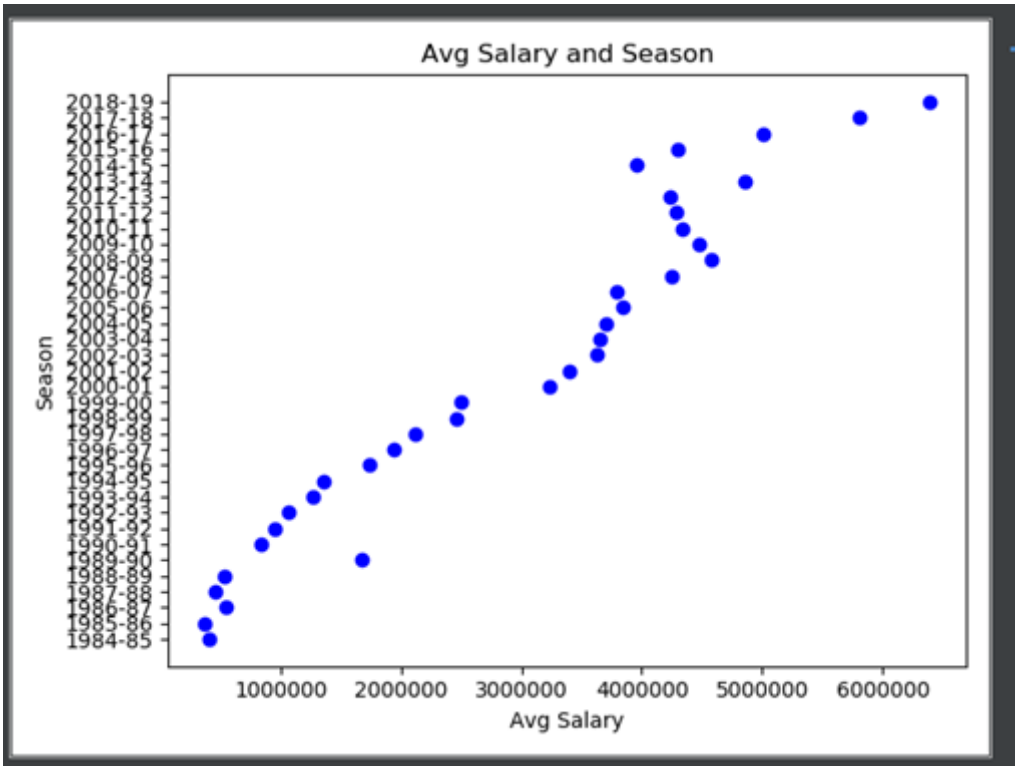


Fig 2.3 Avg Salary and Season

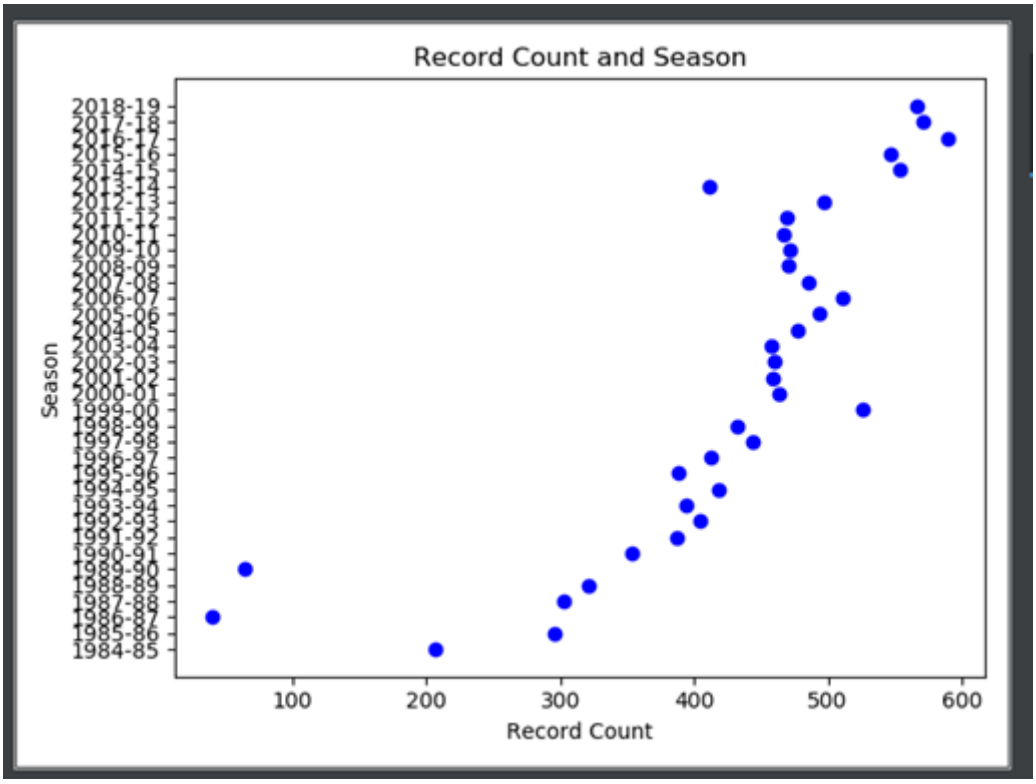


Fig 2.4 Record Count and Season

3. Cluster Analysis

3.1 Explain the finding of conducting three cluster analyses on our data

We did clustering analysis on the subset of 'player_career_stats_cleaned.csv'. Firstly, we dropped non-numeric columns such as 'Name', 'Season' and 'lg'. Then, we dropped some other columns which makes no sense in the following clustering such as: FG% (Field goal percentage), 3PA (3-point field goal attempts per game), 3P% (3-point goal percentage), 2PA (2-point field goal attempts per game), 2P% (2-point field goal percentage) and eFG%

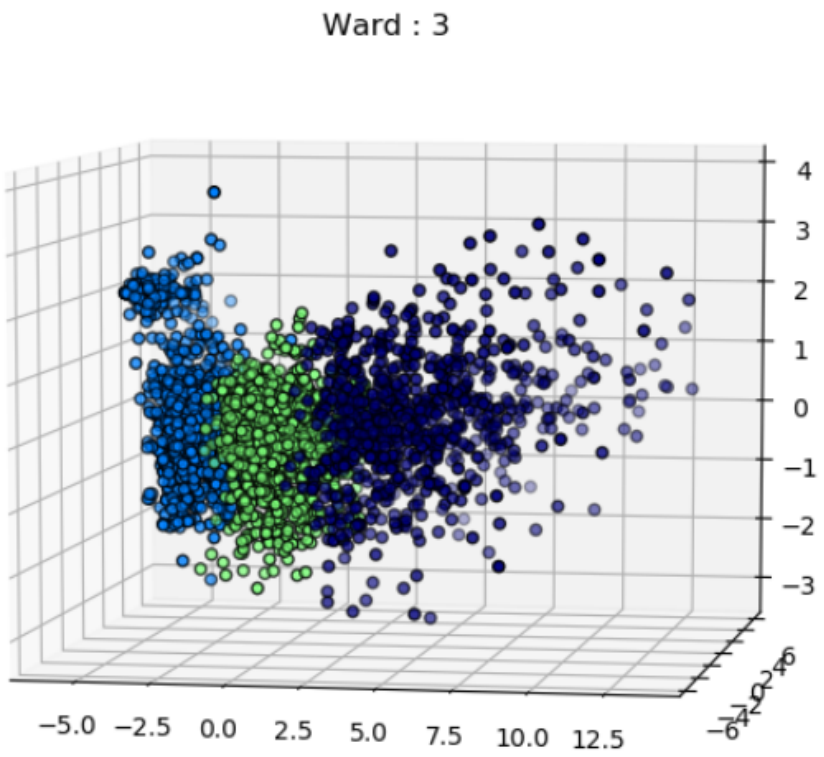
(effective field goal percentage (This statistic adjusts for the fact that a 3-point field goal is worth one more point than a 2-point field goal).

We used three clustering methods according to the material: Ward (hierarchical clustering method), K-means (partition clustering method) and DBSCAN clustering analysis. Besides, I show the results in pictures. Also, We manually selected the range of n.

- Part Ⅰ: Hierarchical Clustering

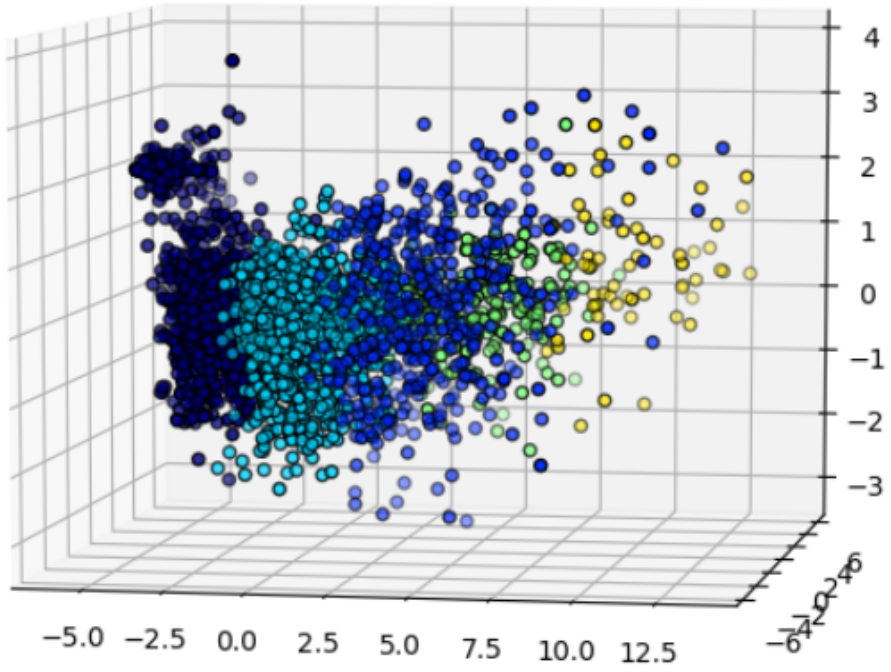
  For the first part of hierarchical clustering, we choose Ward algorithm. I manually picked three clustering components such as 3, 5 and 7 to plot. As the pictures shows below.

  1. When n = 3



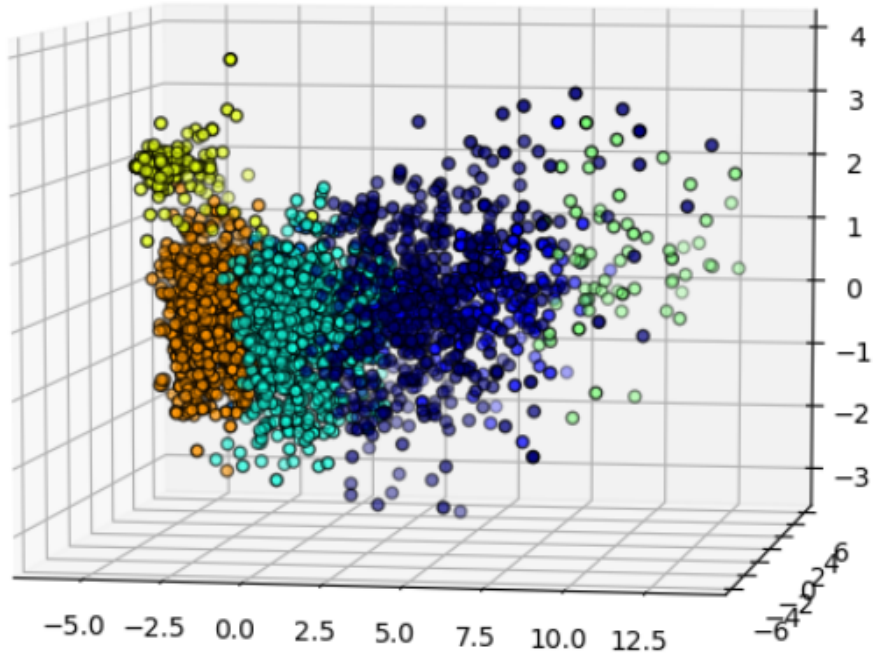
  2. When n = 5

Ward : 5



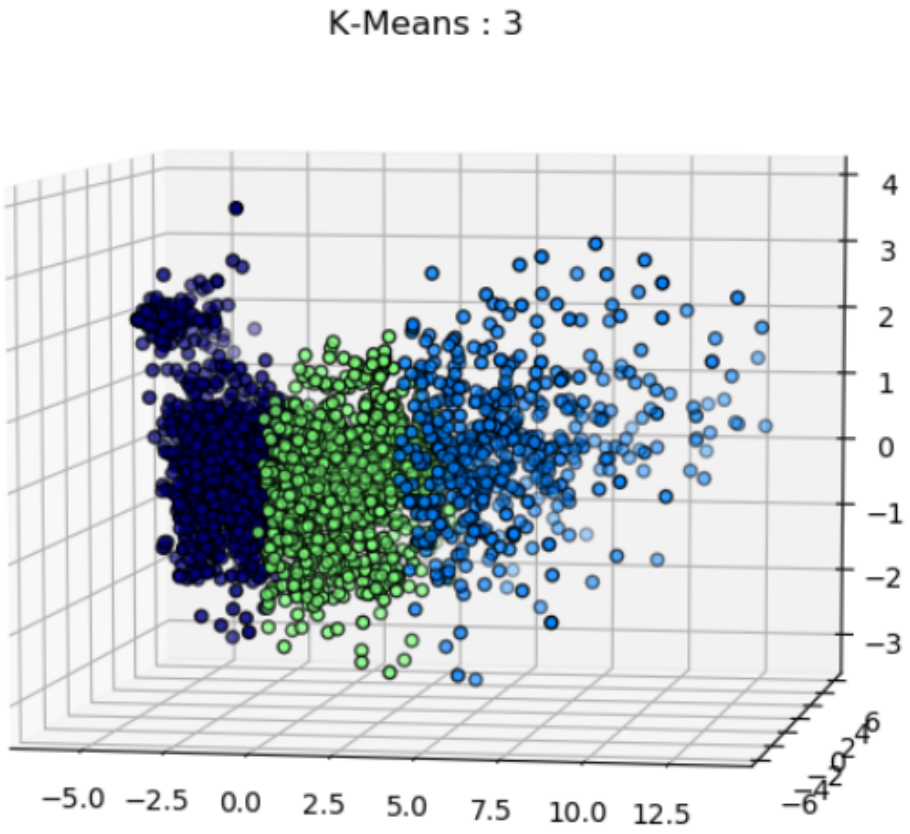
3. When n = 7

Ward : 7

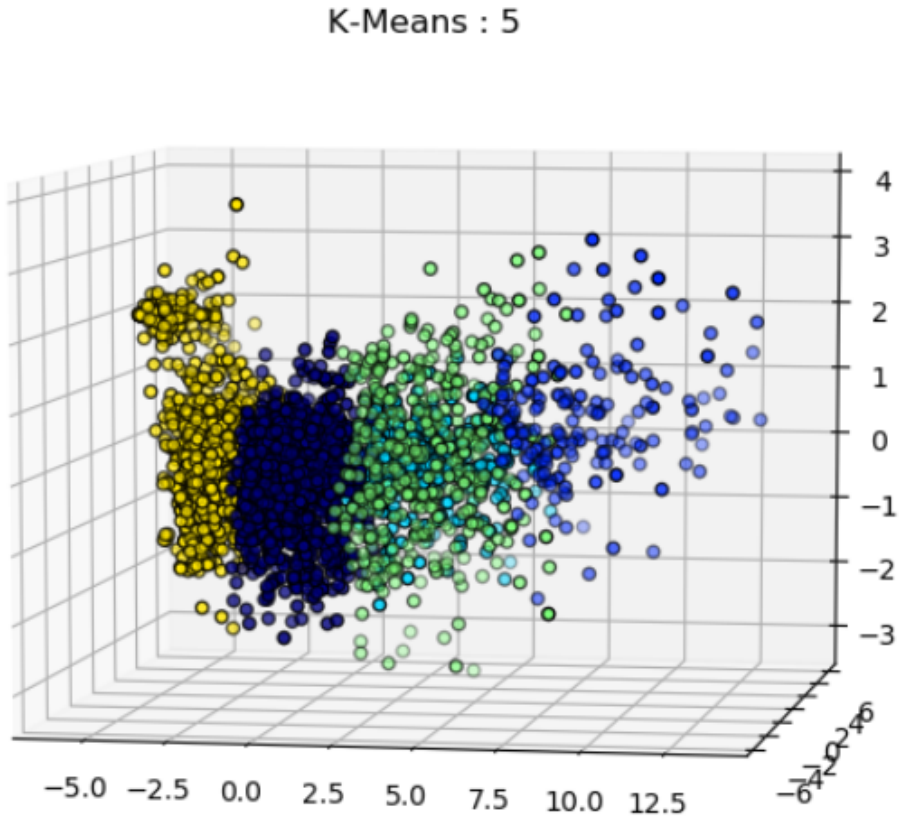


- Part Ⅱ: K-Means Clustering

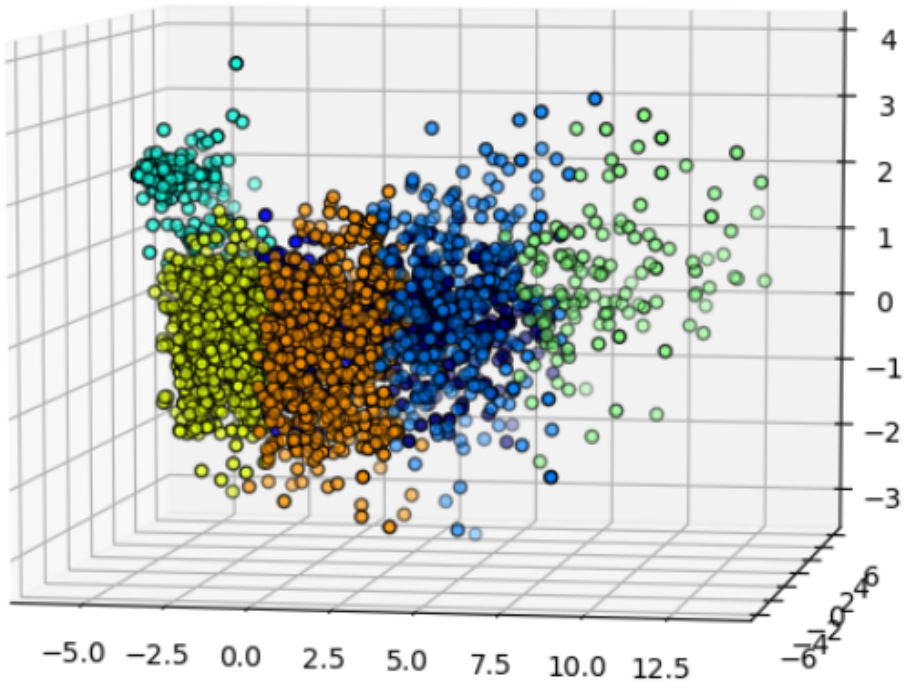
1. When n = 3



2. When n = 5

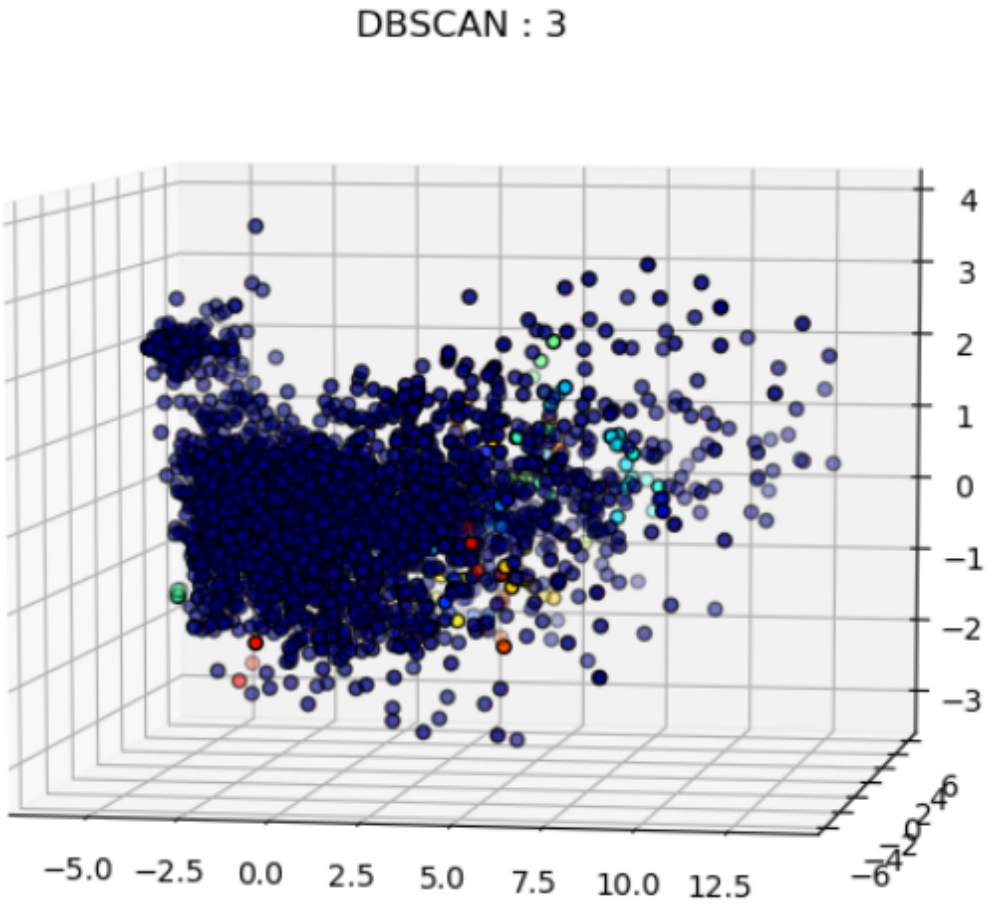


3. When n = 7

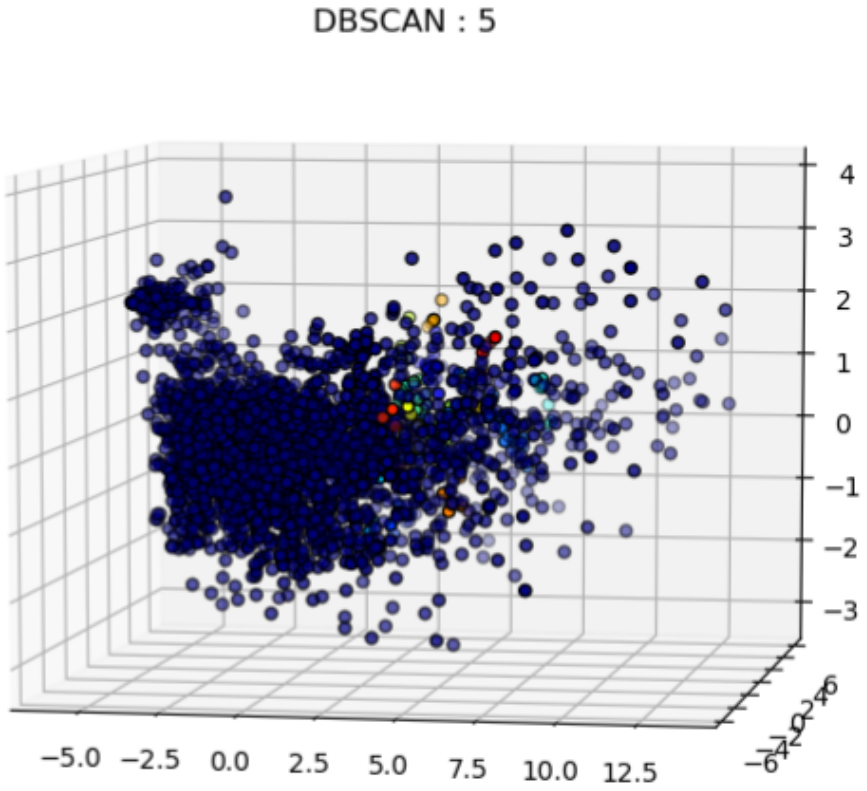


- Part Ⅲ: DBSCAN Clustering

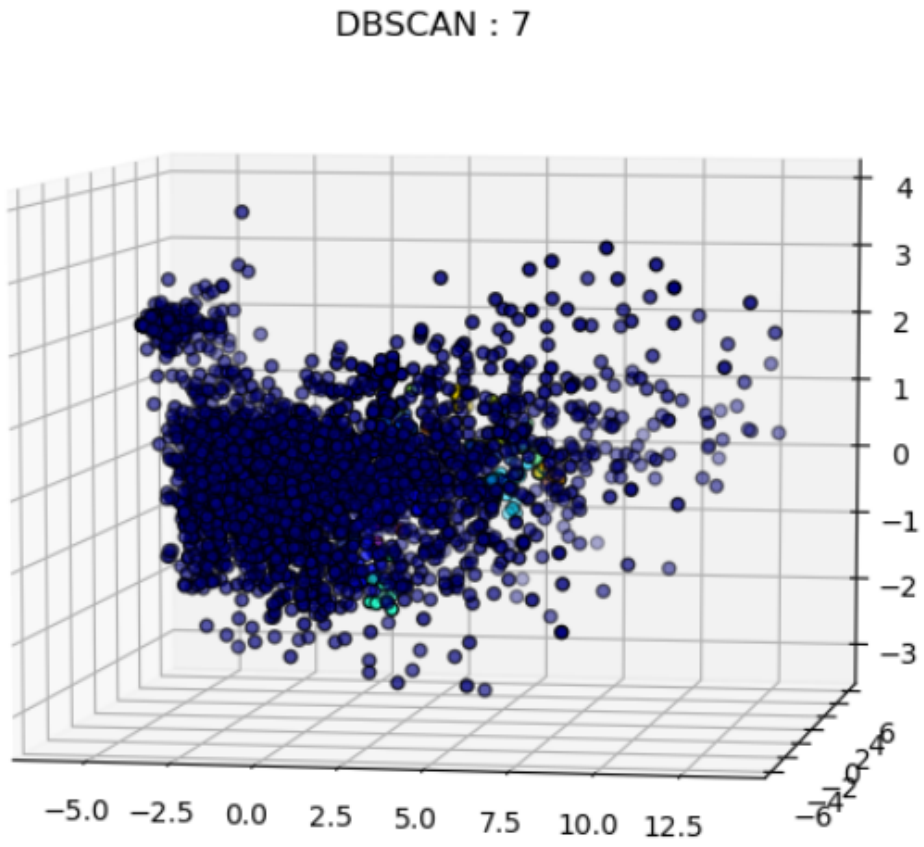
1. When n = 3



2. When n = 5



3. When n = 7



- Summary:

> By observing those pictures, we can clearly find that the clustering results of Ward and K-Means are similarly to some degree. Both of them are tight and well-chunked. Compared to them, the result of dbscan looks kind of in an unclear grouping. Therefore, for this dataset, dbscan is not a good algorithm for clustering relatively. After plotting, we used Silhouette score to measure the quality of different clustering results.

	N=3	N=5	N=7
Ward Algorithm	0.234	0.208	0.181
K-Mean Algorithm	0.281	0.232	0.242
DBSCAN Algorithm	-0.233	0.154	0.173

> As the table shows and according to the silhouette score, K-Mean Algorithm, which is 0.281 when n = 3, is closer to 1, so it performs best.
>
> Hierarchical clustering is a bottom-up approach, merging similar categories to find more similar ones. K-means is to automatically find the most similar class, and then perform clustering operations. DBSCAN is a Density-Based Spatial Clustering method, from the principle of Clustering. The three methods are very similar, and this similarity is also reflected in the three-dimensional image, which can be verified from different n_components.

3.2 Plot the clusters or if the dimensionality is too high, plot a PCA projection of the clusters. Does the plot give you additional insight about the clustering – explain

> We think 3D is better than 2D so we just keep 3D in order to get a better view. However, we still use pca(). For example, in the player_career_stats_cleaned.csv file, we have 20 dimensions, applying PCA and takes 7 hyperplanes. Then, we get 7 most relevant hyperplanes, so that when the dimensions are reduced, the dimensions are 7.
>
> The plots are illustrated in the python program, and the plot does give more insight on the clustering. If only calculating the numbers, it is hard to find the best suitable clustering numbers. With plots, it is easier, as you can see the result. By calculating the Silhouette value, the work is much more easier, the higher Silhouette score, the more the better the clustering. For example, in this program, the highest Silhouette score is 0.281. Therefore, when clustering components = 3, it will attain the best clustering result.

4. Association Rules / Frequent Itemset Mining Analysis

4.1 For the part of association rules, what patterns are most frequent? Is this surprising? Explain your findings

> We chose the subset of salary.csv. Then, we calculate the support in the python program, which means the frequent of each player occurs together (with others if possible) as a percentage in all records. So using Apriori algorithm can find the players who are mostly like to server in the same team among their careers. With lower percentage, the more likely to find players served in the same team. We chose three different support levels, they are 0.28, 0.25 and 0.125.
>
> When min_support_value = 0.25, it has five itemsets, and the most frequent patterns are (Jarrett Jack), (Juwan Howard),(Mike James), (Mike Wilks)and (Shaun Livingston).

```
Processing 20 combinations | Sampling itemset size 2
    support            itemsets
0  0.256410      (Jarrett Jack)
1  0.256410      (Juwan Howard)
2  0.282051        (Mike James)
3  0.256410        (Mike Wilks)
4  0.256410  (Shaun Livingston)
```

When min_support_value = 0.20, it has 55 itemsets. The 55 item sets are the most frequent itemsets when min_support_value = 0.20.

```
Processing 2970 combinations | Sampling itemset size
     support             itemsets
0   0.205128       (Andre Miller)
1   0.230769   (Anthony Tolliver)
2   0.230769         (Beno Udrih)
3   0.205128         (Brian Shaw)
4   0.205128       (Caron Butler)
5   0.205128      (Chucky Atkins)
6   0.205128       (Corey Brewer)
7   0.205128       (Courtney Lee)
8   0.230769        (Damon Jones)
9   0.205128        (Don MacLean)
10  0.230769   (Donyell Marshall)
11  0.205128        (Drew Gooden)
12  0.205128       (Earl Boykins)
13  0.205128        (Eddie House)
14  0.230769      (Eddie Johnson)
15  0.205128       (Gerald Green)
16  0.230769        (Greg Foster)
17  0.205128     (Jamal Crawford)
18  0.205128      (Jannero Pargo)
19  0.256410       (Jarrett Jack)
20  0.205128         (Jason Hart)
```

```
21  0.205128         (Jeff Green)
22  0.205128        (Jeremy Lin)
23  0.230769   (Jerry Stackhouse)
24  0.205128       (Jim Jackson)
25  0.205128       (Joe Johnson)
26  0.230769         (Joe Smith)
27  0.205128     (Johnny Newman)
28  0.205128         (Jon Barry)
29  0.256410      (Juwan Howard)
30  0.205128      (Keith Bogans)
31  0.205128       (Kevin Ollie)
32  0.205128    (Kris Humphries)
33  0.205128      (Larry Hughes)
34  0.205128      (Lou Amundson)
35  0.230769    (Marco Belinelli)
36  0.230769       (Matt Barnes)
37  0.256410        (Mike James)
38  0.205128       (Mike Miller)
39  0.256410        (Mike Wilks)
40  0.205128     (Nate Robinson)
41  0.205128     (Nazr Mohammed)
42  0.205128       (Otis Thorpe)
43  0.230769    (Reggie Williams)
44  0.230769  (Richard Jefferson)
```

```
45  0.205128     (Rod Strickland)
46  0.205128       (Ryan Hollins)
47  0.205128     (Sasha Pavlovi?)
48  0.256410   (Shaun Livingston)
49  0.230769        (Steve Novak)
50  0.205128       (Theo Ratliff)
51  0.230769    (Tony Massenburg)
52  0.205128     (Tyson Chandler)
53  0.205128       (Vince Carter)
54  0.230769    (Wayne Ellington)
```

When min_support_value = 0.125, it has 538 itemsets. Those 538 itemsets are the most frequent itemsets when min_support_value = 0.125.

5. Hypothesis Testing & Classification

5.1 Parametric statistical tests

5.2 Data driven predictive models

Prediction task for all models:

> Predict whether a NBA player is a good player based on his statistic such as FG, 2P and 3P.
>
> *A good player is the player with PTS > 10(Our definition).

Reason for choosing this prediction task:

> This task is reasonable and understandable to predict because a basketball player's PTS is the final result of his whole performance. Therefore, we take lots of his performance statistic into consideration and it will more likely lead us to correct prediction.

- **Decision tree**

  Reason for applying this method:

  > 1. Decision tree is inexpensive to construct and extremely fast at classifying unknown records.
  > 2. There are some attributes which represent the similar feature like FG,FGA,FG%. And decision tree is robust to such situation.
  > 3. It always get comparable performance for classification problem on many simple data sets.

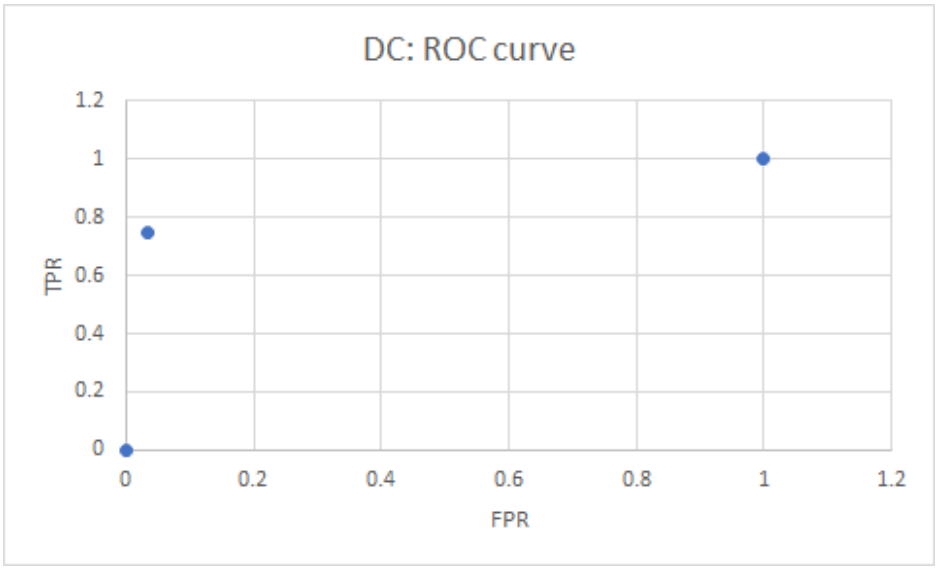
  Applying process:

  > 1. Preprocess data like making non-numeric categorical data numeric and normalize data with norms "l1".
  > 2. Initialize parameters and construct the classifier.
  > 3. Use cross-validation (num_folds=5, test_size = 0.20) and record ROC curve, confusion matrix and prediction accuracy.
  > 4. Analyze data and summarize.

Results:

> - Accuracy:
>   - Cross-validation result with norm l1: DC: 0.934851
>   - Accuracy on test dataset with norm l1: DC: 0.922689
> - confusion matrix:

[[462 18]
[23 92]]

> - ROC curve:
>   - ROC curve fpr: [0., 0.03541667, 1.]
>   - ROC curve tpr: [0., 0.74782609, 1.]
>   - ROC curve thresholds: [2., 1., 0.]
>   - 

> - AUC:

0.8562047101449276

Analysis:

> First, we can see that the accuracies of cross-validation and test set are similar and all get about 92% accuracy.
>
> Then as for confusion matrix and ROC curve results, we can see that the data points for ROC curve is far away from the ROC curve of random guessing.
>
> And I add the AUC value to help evaluate the ROC curve and it is 0.8562047101449276 and really near 1.
>
> In sum, the model does truly excellently for our prediction task.

- **A Lazy Learner Method(kNN)**

Reason for applying this method:

> 1. kNN is lazy learner and there is no training period. Thus, as our dataset is not so large and k is proper, it costs little time.
> 2. It's easy to implement.

Applying process:

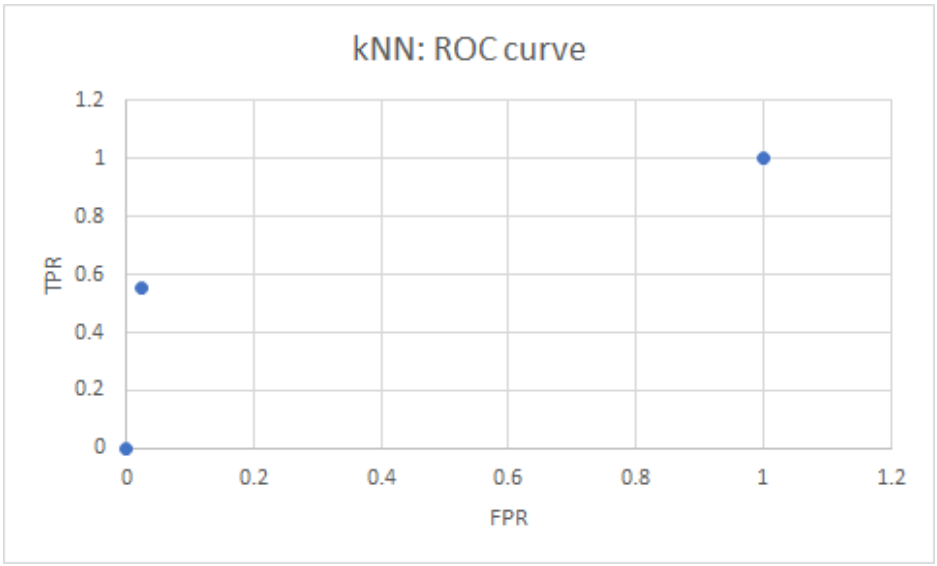
> 1. Preprocess data like making non-numeric categorical data numeric and normalize data with norms "l1".
> 2. Initialize parameters and construct the classifier with n_neighbors=10.

3. Use cross-validation (num_folds=5, test_size = 0.20) and record ROC curve, confusion matrix and prediction accuracy.
4. Analyze data and summarize.

Results:

- Accuracy:
  - Cross-validation result with norm l1: kNN: 0.880624
  - Accuracy on test dataset with norm l1: kNN: 0.895798
- confusion matrix:

[[469 11]
[51 64]]

- ROC curve:
  - ROC curve fpr: [0., 0.02291667, 1.]
  - ROC curve tpr: [0., 0.55652174, 1.]
  - ROC curve thresholds: [2., 1., 0.]
  - 

- AUC:

0.766802536231884

Analysis:

First, we can see that the accuracy of cross-validation is 88% and it's 90% on test set. It's due to the seed of random and maybe the features of data in test set fits the model better than the validation set in the training set.

Then as for confusion matrix and ROC curve results, we can see that the data points for ROC curve is a little far from the ROC curve of random guessing.

And the ROC curve and it is 0.766802536231884 and it's not bad.

In sum, the model does truly well for our prediction task.

- **Naïve Bayes**

Reason for applying this method:

1. Independence assumption of Naïve Bayes may not hold for some attributes. Actually in our dataset, the statistic of NBA players always has some connection. For example, the 2P% and 3P% represents the per cent of your different shooting goals. And FG% is calculated from this 2 parameters. Thus, it's not wise to apply Naïve Bayes to this prediction task. However, I think the importance lies in check

the knowledge we learn is correct and applying Naïve Bayes in such bad situation can help us learn deeply. Thus, we still apply Naïve Bayes to this prediction task.
2. Naïve Bayes is robust to isolated noise points.

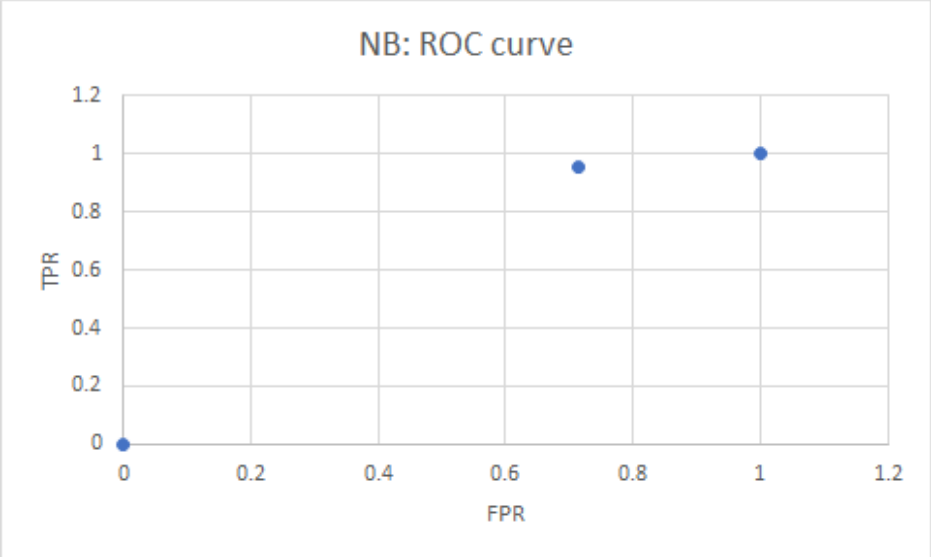
Applying process:

1. Preprocess data like making non-numeric categorical data numeric and normalize data with norms "l1".
2. Initialize parameters and construct the classifier with Gaussian NB.
3. Use cross-validation (num_folds=5, test_size = 0.20) and record ROC curve, confusion matrix and prediction accuracy.
4. Analyze data and summarize.

Results:

- Accuracy:
  - Cross-validation result with norm l1: NB: 0.448518
  - Accuracy on test dataset with norm l1: NB: 0.415126
- confusion matrix:

[[137 343]
[5 110]]

- ROC curve:
  - ROC curve fpr: [0., 0.71458333, 1.]
  - ROC curve tpr: [0., 0.95652174, 1.]
  - ROC curve thresholds: [2., 1., 0.]
  - 

- AUC:

0.6209692028985507

Analysis:

First, we can see that the accuracy of cross-validation is 44% and it's 41% on test set. It confirms our guess about the performance of Naïve Bayes on this prediction task as the attributes have connections between each other.

Then as for confusion matrix and ROC curve results, we can see that the data points for ROC curve is really close from the ROC curve of random guessing.

And the ROC curve and it is 0.6209692028985507 and it's a little bad.

In sum, the model does truly terribly for our prediction task.

- **SVM**

  Reason for applying this method:

  > 1. SVM works relatively well when there is clear margin of separation between classes. To some degree, I guess it should fit with the task better than kNN and worse than good classifiers like decision tree because it depends on clear margin of separation between classes but not entirely rely on independency like kNN.
  > 2. SVM is relatively memory efficient and is more effective in high dimensional spaces. And our dataset has about 28 dimensions and I think it fits this feature.

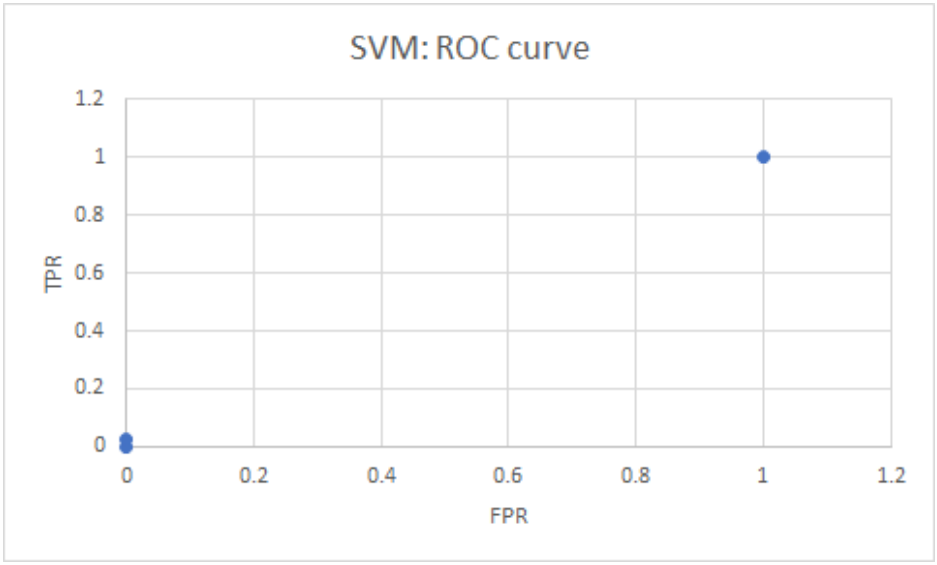
  Applying process:

  > 1. Preprocess data like making non-numeric categorical data numeric and normalize data with norms "l1".
  > 2. Initialize parameters and construct the classifier with SVC.
  > 3. Use cross-validation (num_folds=5, test_size = 0.20) and record ROC curve, confusion matrix and prediction accuracy.
  > 4. Analyze data and summarize.

  Results:

  > - Accuracy:
  >   - Cross-validation result with norm l1: SVM: 0.797808
  >   - Accuracy on test dataset with norm l1: SVM: 0.811765
  > - confusion matrix:

  [[480 0]
  [112 3]]

  > - ROC curve:
  >   - ROC curve fpr: [0., 0., 1.]
  >   - ROC curve tpr: [0., 0.02608696, 1.]
  >   - ROC curve thresholds: [2., 1., 0.]
  >   - 
  >
  > - AUC:

  0.5130434782608696

  Analysis:

> First, we can see that the accuracy of cross-validation is 80% and it's 81% on test set. It confirms our guess about the performance of SVM on this prediction task as it works better than kNN and worse than decision tree.
>
> Then as for confusion matrix and ROC curve results, we can see that the data points for ROC curve is really close from the ROC curve of random guessing.
>
> And the ROC curve and it is 0.5130434782608696 and it's a little bad.
>
> However, based on its accuracy I think maybe there is some problems about the test data because for ROC curve thresholds[1], the fpr[1] is 0 and tpr[1] is 0.02608696. I think there is some problem about the distribution of test data as we get truly low TP and FN values with not bad accuracy.
>
> In sum, the model does not bad for our prediction task.

- **Random Forest**

  Reason for applying this method:

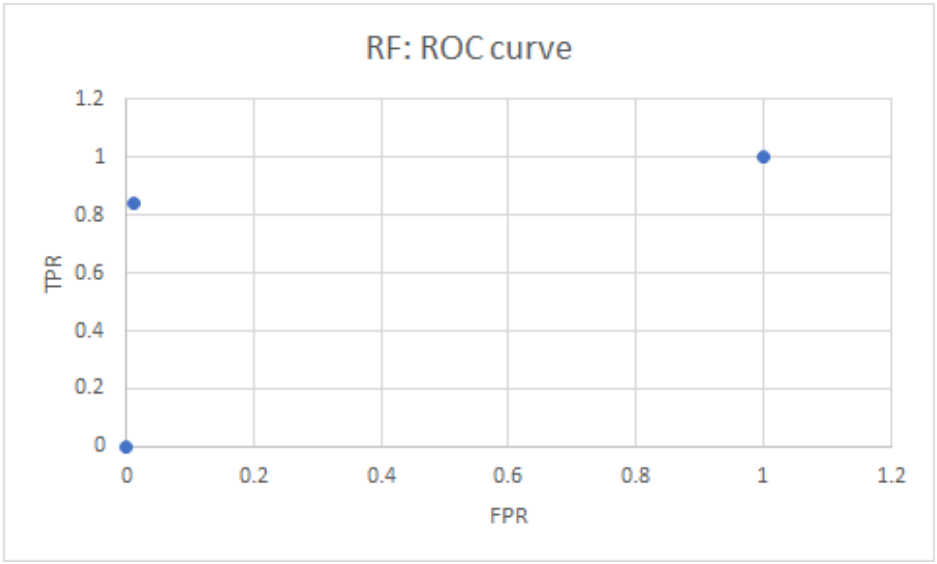
  > 1. Random Forest is the combination of lots of decision tree and the predictive performance can compete with the best supervised learning algorithms.
  > 2. They provide a reliable feature importance estimate which is really proper for our prediction task because our definition of a good player is actually connected with PTS.

  Applying process:

  > 1. Preprocess data like making non-numeric categorical data numeric and normalize data with norms "l1".
  > 2. Initialize parameters and construct the classifier with n_estimators=100(tree numbers).
  > 3. Use cross-validation (num_folds=5, test_size = 0.20) and record ROC curve, confusion matrix and prediction accuracy.
  > 4. Analyze data and summarize.

  Results:

  > - Accuracy:
  >   - Cross-validation result with norm l1: RF: 0.966369
  >   - Accuracy on test dataset with norm l1: RF: 0.961345
  > - confusion matrix:
  >
  > [[475 5]
  > [18 97]]
  >
  > - ROC curve:
  >   - ROC curve fpr: [0., 0.01041667., 1.]
  >   - ROC curve tpr: [0., , 1.]
  >   - ROC curve thresholds: [2., 1., 0.]



- AUC:

0.9165307971014492

Analysis:

> First, we can see that the accuracy of cross-validation is 96% and it's 96% on test set. It performs really excellently.
>
> Then as for confusion matrix and ROC curve results, we can see that the data points for ROC curve is really far away from the ROC curve of random guessing.
>
> And the ROC curve and it is 0.9165307971014492 and it's so good.
>
> In sum, the model does best for our prediction task.