APPENDIX

CODING IMPORT LIBRARY

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
93.93.
from sklearn.cluster import DBSCAN
from collections import Counter
from pylab import rcParams
from sklearn.neighbors import NearestNeighbors
from sklearn import metrics
rcParams['figure.figsize'] = 12, 7
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
```

IMPORT DATASET

```
X = pd.read_csv ("C:/Users/Lenovo/Desktop/BITCOIN/all_seasons.csv")
print (X)
```

CHECK DATA TYPE

```
X.isna().sum()
```

INPUT DATA TO ARRAY

```
dbscan_X = X[['player_height', 'reb']]
dbscan_X = dbscan_X.values.astype('float64', copy=False)
dbscan_X
dbscan_X1 = X[['player_height', 'pts']]
dbscan_X1 = dbscan_X1.values.astype('float64', copy=False)
dbscan_X1
dbscan_X2 = X[['player_height', 'ast']]
dbscan_X2 = dbscan_X2.values.astype('float64', copy=False)
dbscan_X2
```

ELBOW METHOD

```
neigh = NearestNeighbors(n_neighbors=4)
nbrs = neigh.fit(dbscan_X)
distances, indices = nbrs.kneighbors(dbscan_X)
neigh = NearestNeighbors(n_neighbors=4)
```

```
nbrs = neigh.fit(dbscan_X)
distances, indices = nbrs.kneighbors(dbscan_X1)
neigh = NearestNeighbors(n_neighbors=4)
nbrs = neigh.fit(dbscan_X)
distances, indices = nbrs.kneighbors(dbscan_X2)
distances = np.sort(distances, axis=0)
distances = distances[:,2]
plt.plot(distances)
plt.xlabel("index")
plt.ylabel("distances")
plt.show()
```

SEARCH EPSLION AND MINIMAL POINTS WITH SILHOUETTE

```
range_eps = [2.54]
for i in range_eps :
    print("eps value is "+str(i))
    model = DBSCAN(eps= i, min_samples=4).fit(dbscan_X)
    zero = np.zeros_like(model.labels_, dtype=bool)
    zero[model.core_sample_indices_] = True
    labels = model.labels_
    print(set(labels))
    silhouette_avg = metrics.silhouette_score(dbscan_X, labels)
    print("For eps value =" +str(i), labels,
          "The average is :",silhouette_avg)
range_eps = [2.54]
for i in range_eps :
    print("eps value is "+str(i))
    model = DBSCAN(eps= i, min_samples=4).fit(dbscan_X1)
    zero = np.zeros_like(model.labels_, dtype=bool)
    zero[model.core_sample_indices_] = True
    labels = model.labels_
    print(set(labels))
    silhouette_avg = metrics.silhouette_score(dbscan_X, labels)
    print("For eps value =" +str(i), labels,
          "The average is :",silhouette_avg)
range_eps = [2.54]
for i in range_eps :
    print("eps value is "+str(i))
    model = DBSCAN(eps= i, min_samples=4).fit(dbscan_X2)
    zero = np.zeros_like(model.labels_, dtype=bool)
    zero[model.core_sample_indices_] = True
    labels = model.labels_
    print(set(labels))
    silhouette_avg = metrics.silhouette_score(dbscan_X, labels)
```

```
159.      print("For eps value =" +str(i), labels,
160.            "The average is :",silhouette_avg)
161. min_samples = [4]
162. for i in min_samples :
163.      print("min_samples values is ="+str(i))
164.      model= DBSCAN(eps=0.1, min_samples=i).fit(dbscan_X)
165.      zero = np.zeros_like(model.labels_, dtype=bool)
166.      zero[model.core_sample_indices_] = True
167.      labels = set([label for label in model.labels_ if label >= 0])
168.      print(set(labels))
 print("for min_samples_calue ="+str(i), "total no of clusters are
"+str(len(set(labels))))
169. min_samples = [4]
170. for i in min_samples :
171.      print("min_samples values is ="+str(i))
172.      model= DBSCAN(eps=0.1, min_samples=i).fit(dbscan_X1)
173.      zero = np.zeros_like(model.labels_, dtype=bool)
174.      zero[model.core_sample_indices_] = True
175.      labels = set([label for label in model.labels_ if label >= 0])
176.      print(set(labels))
 print("for min_samples_calue ="+str(i), "total no of clusters are
"+str(len(set(labels))))
177. min_samples = [4]
178. for i in min_samples :
179.      print("min_samples values is ="+str(i))
180.      model= DBSCAN(eps=0.1, min_samples=i).fit(dbscan_X2)
181.      zero = np.zeros_like(model.labels_, dtype=bool)
182.      zero[model.core_sample_indices_] = True
183.      labels = set([label for label in model.labels_ if label >= 0])
184.      print(set(labels))
 print("for min_samples_calue ="+str(i), "total no of clusters are
"+str(len(set(labels))))
185. range_n_clusters = []
186.
187. for n_clusters in range_n_clusters:
188.      clusterer = KMeans(n_clusters= n_clusters, random_state=42)
189.      cluster_labels = clusterer.fit_predict(dbscan_X)
190.
191.      silhouette_avg = silhouette_score(dbscan_X,cluster_labels)
192.      print("for n_clusters = ", n_clusters,
193.            "The average silhouette_score is :", silhouette_avg)
194.
    sample_silhouette_values = silhouette_samples(dbscan_X, cluster_labels)
195. range_n_clusters = []
196.
197. for n_clusters in range_n_clusters:
198.      clusterer = KMeans(n_clusters= n_clusters, random_state=42)
199.      cluster_labels = clusterer.fit_predict(dbscan_X1)
200.
201.      silhouette_avg = silhouette_score(dbscan_X,cluster_labels)
```

```
    print("for n_clusters = ", n_clusters,
          "The average silhouette_score is :", silhouette_avg)

sample_silhouette_values = silhouette_samples(dbscan_X, cluster_labels)
range_n_clusters = []

for n_clusters in range_n_clusters:
    clusterer = KMeans(n_clusters= n_clusters, random_state=42)
    cluster_labels = clusterer.fit_predict(dbscan_X2)

    silhouette_avg = silhouette_score(dbscan_X,cluster_labels)
    print("for n_clusters = ", n_clusters,
          "The average silhouette_score is :", silhouette_avg)

sample_silhouette_values = silhouette_samples(dbscan_X, cluster_labels)
```

DBSCAN IMPLEMENTATION

```
model = DBSCAN(eps = 2.54, min_samples = 4).fit(dbscan_X)
zero = np.zeros_like(model.labels_, dtype=bool)
zero[model.core_sample_indices_] = True
labels = model.labels_
model = DBSCAN(eps = 2.54, min_samples = 4).fit(dbscan_X1)
zero = np.zeros_like(model.labels_, dtype=bool)
zero[model.core_sample_indices_] = True
labels = model.labels_
model = DBSCAN(eps = 2.54, min_samples = 4).fit(dbscan_X2)
zero = np.zeros_like(model.labels_, dtype=bool)
zero[model.core_sample_indices_] = True
labels = model.labels_
```

CHECK THE NUMBER OF CLUSTER FORMED

```
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_clusters_
n_noise_ = list(labels).count(-1)
n_noise_
clusters = Counter(model.labels_)
print(clusters)
print("number of clusters", n_clusters_)
```

CALCULATE COMPLETENESS, HOMOGENEITY, V-MEASURE

```
metrics.homogeneity_score(X['player_height'],labels)
metrics.homogeneity_score(X['reb'],labels)
metrics.completeness_score(X['player_height'],labels)
metrics.completeness_score(X['reb'],labels)
metrics.v_measure_score(X['player_height'],labels)
```

```
metrics.v_measure_score(X['reb'],labels)
metrics.homogeneity_score(X['player_height'],labels)
metrics.homogeneity_score(X['pts'],labels)
metrics.completeness_score(X['player_height'],labels)
metrics.completeness_score(X['pts'],labels)
metrics.v_measure_score(X['player_height'],labels)
metrics.v_measure_score(X['pts'],labels)
metrics.homogeneity_score(X['player_height'],labels)
metrics.homogeneity_score(X['ast'],labels)
metrics.completeness_score(X['player_height'],labels)
metrics.completeness_score(X['ast'],labels)
metrics.v_measure_score(X['player_height'],labels)
metrics.v_measure_score(X['ast'],labels)
```

VISUALIATION

```
unique_labels = set(labels)
colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1,
   len(unique_labels))]
for k, col in zip(unique_labels, colors):
    #if k == -1:
        #col = [0, 0, 0]
    if k == 1:
        col = [1, 1, 0]
    elif k == 1:
        col = [0, 0, 1]
    elif k == 2:
        col = [1, 0, 0]
    elif k == 3:
        col = [0, 1, 0]
    elif k == 4:
        col = [0.63, 0.13, 0.94]
    elif k == 5:
        col = [0.80, 0.75, 1]
    elif k == 6:
        col = [1, 0.65, 0]
    elif k == 7:
        col = [0.65, 0.16, 0.16]
    elif k == 8:
        col = [0.50, 0.50, 0.50]
    elif k == 9:
        col = [0, 0, 0.55]
    elif k == 10:
        col = [0.53, 0.81, 1]
    elif k == 11:
        col = [1, 0.84, 0]
    elif k == 12:
        col = [1, 0.55, 0]

    label1 = labels == k

```

```
    xy = dbscan_X1[label1 & zero]
    plt.plot(
    xy[:, 0],
        xy[:, 1],
        "o",
        markerfacecolor=tuple(col),
        markeredgecolor="k",
        markersize=10,
    )

for k, col in zip(unique_labels, colors):
    if k == -1:
        col = [0, 0, 0]

    xy = dbscan_X1[label1 & ~zero]
    plt.plot(
    xy[:, 0],
        xy[:, 1],
        "o",
        markerfacecolor=tuple(col),
        markeredgecolor="k",
        markersize=10,
    )
plt.legend(unique_labels)
plt.title("Estimated number of clusters: %d" % n_clusters_)
plt.xlabel('player_height')
plt.ylabel('pts')
plt.show()
```

PLAGIARISM CHECK.ORG

0.96% PLAGIARISM
APPROXIMATELY

Report #14335821

INTRODUCTION Background There are many sports in the world today, one of which is basketball. Basketball is one of the most popular sports in the world today as well as in Indonesia there are many basketball competitions ranging from amateur leagues to professional leagues. In basketball, of course, a proportional height is needed in order to compete in matches because of course it is not balanced if short players must face tall players. In basketball, there are 5 classifications of player positions, namely point guard, shooting guard, small forward, power forward center and because basketball is a sport that requires a proportional height, the placement of players must also be important. From the problems above, that the placement of players' positions based on height is very important in basketball because of course, it will affect the performance of basketball athletes. This study using the concept of clustering and using the DBSCAN algorithm to place the athletes in an optimal position based on the athletes'

.

g