Ananya Mittal - HW 4 (DS -111)

April 18, 2023

[3]:
```python
#Question 1
#1(a)
import pandas as pd

fifa22 = pd.read_csv("fifa22.csv", squeeze = True)
fifa22.head()
```

[3]:
```
                             name  rank gender  wage_eur   log_wage position  \
0  Lionel Andrés Messi Cuccittini    93      M  320000.0  12.676076       RW
1      Lucia Roberta Tough Bronze    92      F       NaN        NaN      NaN
2                Vivianne Miedema    92      F       NaN        NaN      NaN
3       Wéndèleine Thérèse Renard    92      F       NaN        NaN      NaN
4              Robert Lewandowski    92      M  270000.0  12.506177       ST

   nationality                 club                league preferred_foot  \
0    Argentina  Paris Saint-Germain        French Ligue 1           Left
1      England                  NaN                   NaN          Right
2  Netherlands                  NaN                   NaN          Right
3       France                  NaN                   NaN          Right
4       Poland    FC Bayern München  German 1. Bundesliga          Right

   shooting  passing  dribbling  defending  attacking  skill  movement  power  \
0      92.0     91.0       95.0  26.333333       85.8   94.0      90.2   77.8
1      61.0     70.0       81.0  89.000000       69.0   62.2      84.2   78.8
2      93.0     75.0       88.0  25.000000       86.0   79.0      80.6   84.0
3      70.0     62.0       73.0  91.333333       62.6   67.8      64.0   82.4
4      92.0     79.0       86.0  32.000000       86.0   81.4      81.6   84.8

   mentality  goalkeeping
0  73.833333         10.8
1  69.166667         12.6
2  70.833333         15.6
3  73.500000         12.8
4  80.666667         10.2
```

1(b) The unit of analysis in this dataset is individual soccer players in the FIFA 2022 video game.

[4]:
```
#1(c)

print(fifa22.shape)
print("There are 19630 observations and 19 features in the data set (There are␣
↪20 columns but the 1st column includes the unit of analysis)")
```

```
(19630, 20)
There are 19630 observations and 19 features in the data set (There are 20
columns but the 1st column includes the unit of analysis)
```

[5]:
```
#1(d)
num_M = fifa22[fifa22["gender"] == "M"].shape[0]
num_F = fifa22[fifa22["gender"] == "F"].shape[0]

# Print the results
print("Number of male players:", num_M)
print("Number of female players:", num_F)
```

```
Number of male players: 19239
Number of female players: 391
```

1(e) No this dataset is not representative of the real-world population of professional soccer players because the game only includes the most famous players that gamers would want to play with and hence may exclude many other professional soccer players from different countries around the world

[6]:
```
#1(f)
fifa22.dropna(subset=["passing"], inplace = True)
print("Shape of final dataset:", fifa22.shape)
```

```
Shape of final dataset: (17450, 20)
```

[7]:
```
#Question 2
#2(a)
import statsmodels.formula.api as smf
mult_reg = smf.ols('rank ~ passing + attacking + defending + skill',␣
↪data=fifa22).fit()
mult_reg.summary()
```

[7]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   rank   R-squared:                       0.705
Model:                            OLS   Adj. R-squared:                  0.705
Method:                 Least Squares   F-statistic:                 1.044e+04
Date:                Tue, 18 Apr 2023   Prob (F-statistic):               0.00
Time:                        17:58:08   Log-Likelihood:                -47856.
No. Observations:               17450   AIC:                         9.572e+04
```

```
Df Residuals:                   17445   BIC:                         9.576e+04
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     25.3278      0.203    124.785      0.000      24.930      25.726
passing       -0.0247      0.010     -2.425      0.015      -0.045      -0.005
attacking      0.6109      0.006     94.005      0.000       0.598       0.624
defending      0.1719      0.002     84.413      0.000       0.168       0.176
skill          0.0066      0.009      0.730      0.465      -0.011       0.024
==============================================================================
Omnibus:                      171.799   Durbin-Watson:                   1.342
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              178.339
Skew:                           0.234   Prob(JB):                     1.88e-39
Kurtosis:                       3.163   Cond. No.                         790.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

2(b) To see how much of the variation in rank is explained by our features, we look at the R-squared value = 0.705 = 70.5%

2(c) Attacking and defeding are significant at the 1% level with a p-value of 0.0% and 0.0% respectively.

2(d)Holding passing, attacking, and defending constant, a 1-unit increase in "skill" is associated with an increase (positive change) in ranking by 0.66% as seen by looking at the coefficient.

Question 3

3(a) With a R-squared value of 0.705, passing, attacking, and defending would do a pretty good job at predicting rank for out-of-sample data because their p-values are low. On the other hand, skill would not do a great job at predicting rank because it has a relatively high p-value (0.465).

[49]:
```python
#3(b)
# Create an X dataframe with just four features: passing, attacking, defending,␣
↪and skill.
#Create a Y dataframe (or series) with just the "rank" variable. Display the␣
↪first five rows of each.

import numpy as np
x = fifa22[['passing','attacking','defending','skill']]
y = fifa22['rank']

print ("First 5 rows of the X data frame")
```

```
print()
print(x.head())
print()
print ("First 5 rows of the Y data frame")
y.head()
```

```
First 5 rows of the X data frame

   passing  attacking  defending  skill
0     91.0       85.8  26.333333   94.0
1     70.0       69.0  89.000000   62.2
2     75.0       86.0  25.000000   79.0
3     62.0       62.6  91.333333   67.8
4     79.0       86.0  32.000000   81.4

First 5 rows of the Y data frame
```

[49]:
```
0    93
1    92
2    92
3    92
4    92
Name: rank, dtype: int64
```

[9]:
```
#3(c)
from sklearn.model_selection import train_test_split

np.random.seed(123)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)

print(x_train.head())
```

```
       passing  attacking  defending  skill
17226     52.0       48.0  59.333333   53.2
13548     48.0       55.0  12.666667   54.0
17874     59.0       46.2  58.000000   57.8
19599     47.0       40.6  46.666667   40.0
15629     49.0       51.8  25.666667   49.6
```

[10]:
```
#3(d)
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(x_train, y_train)

print('Intercept:', regressor.intercept_)
```

```python
print('Coefficients:', regressor.coef_)
```

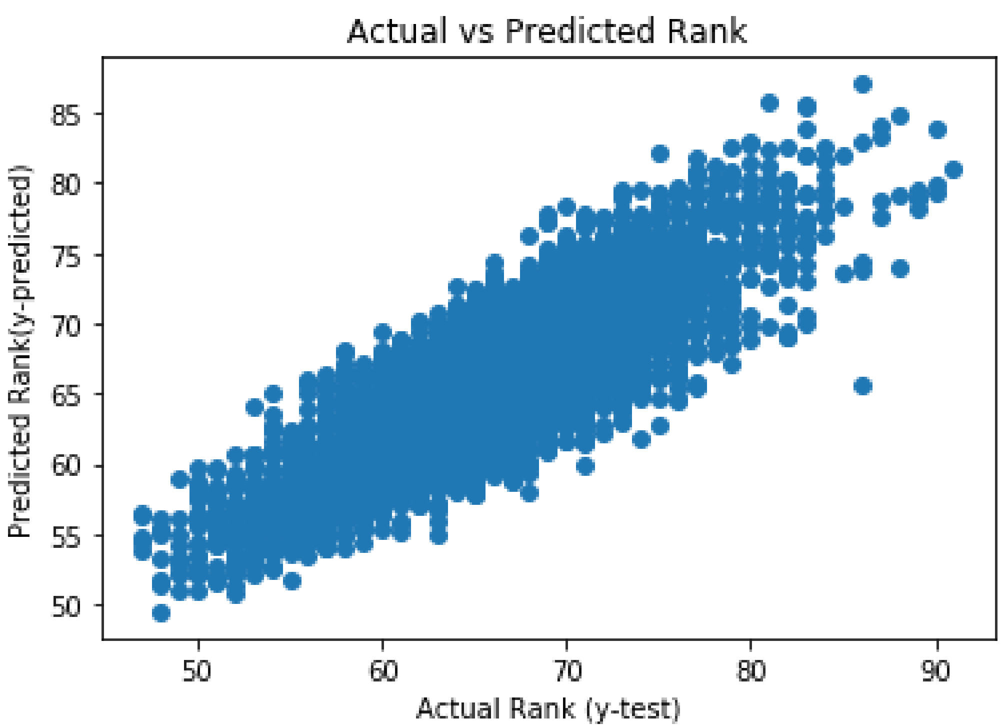
```
Intercept: 25.167733064621757
Coefficients: [-0.02444506  0.61230756  0.17314968  0.00612364]
```

3(e) The coefficient of attacking in Q2 is 0.6109 while it is 0.6123 (2nd element in the list) in Q3. So the value has changed a little, though not very much.

[11]:
```python
#3(f)
y_pred = regressor.predict(x_test)
print('First three predicted values:', y_pred[:3])
```

```
First three predicted values: [64.57617047 72.78035994 70.46341746]
```

[12]:
```python
#3(g)

import matplotlib.pyplot as plt
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Rank (y-test)')
plt.ylabel('Predicted Rank(y-predicted)')
plt.title('Actual vs Predicted Rank')
plt.show()
```



[13]:
```
#3(h)
import numpy as np
from sklearn import metrics
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,
    y_pred)))
```

```
Root Mean Squared Error: 3.744562639987198
```

3(h) In terms of "average error" of the model, this value measures the average difference (3.745) between the predicted rank values and actual rank values.

3(i) Based on the Root Mean Squared Error of 3.745 and the scatterplot showing a relatively strong positive correlation between the actual ranks and predicted ranks, we can say that this model does a relatively good job at predicting a player's rank. However, due to the variability shown in the scatterplot, the model could still be improved.

[14]:
```
#Question 4
#4(a)
fifa22['preferred_foot'].value_counts()
```

[14]:
```
Right    13044
Left      4406
Name: preferred_foot, dtype: int64
```

[15]:
```
#4(b)

right_count = fifa22['preferred_foot'].value_counts()['Right']
right_percentage = right_count/len(fifa22)*100

print("Percentage of players who prefer their right foot:",
    round(right_percentage, 2), "%")
```

```
Percentage of players who prefer their right foot: 74.75 %
```

[16]:
```
#4(c)

X = fifa22[['shooting', 'passing', 'dribbling', 'defending', 'attacking',
    'skill', 'movement', 'power', 'mentality', 'goalkeeping']]
X.head()
```

[16]:
```
   shooting  passing  dribbling  defending  attacking  skill  movement  power  \
0      92.0     91.0       95.0  26.333333       85.8   94.0      90.2   77.8
1      61.0     70.0       81.0  89.000000       69.0   62.2      84.2   78.8
2      93.0     75.0       88.0  25.000000       86.0   79.0      80.6   84.0
3      70.0     62.0       73.0  91.333333       62.6   67.8      64.0   82.4
4      92.0     79.0       86.0  32.000000       86.0   81.4      81.6   84.8

   mentality  goalkeeping
```

```
0  73.833333         10.8
1  69.166667         12.6
2  70.833333         15.6
3  73.500000         12.8
4  80.666667         10.2
```

[17]:
```
#4(d)
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled[:3]
```

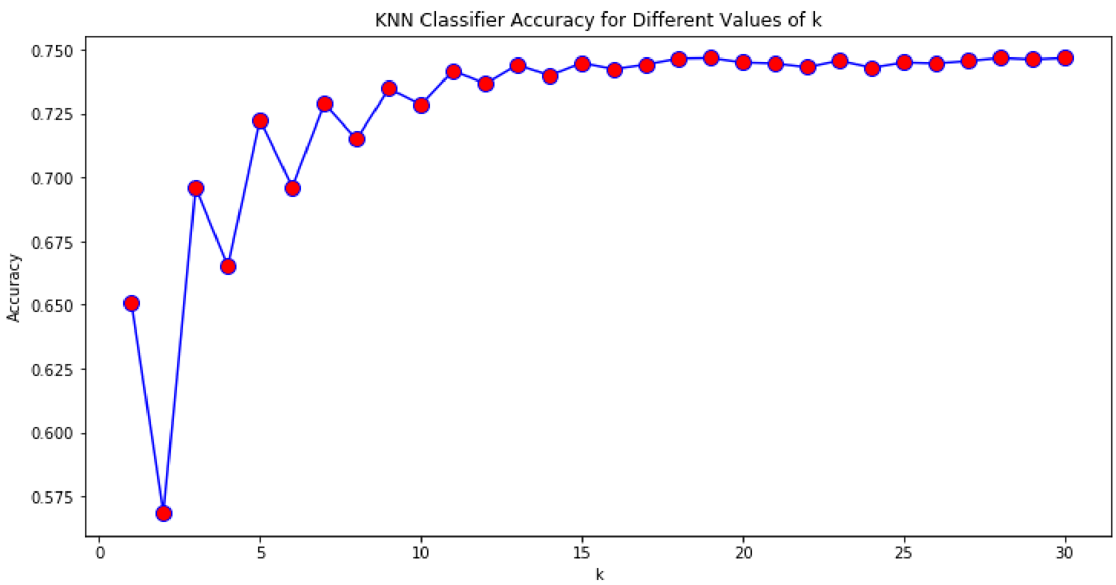
[17]:
```
array([[ 2.7843116 ,  3.29664165,  3.31535783, -1.39304935,  3.40016362,
         3.54858025,  2.77464012,  1.94428215,  2.18061369,  0.2816757 ],
       [ 0.59771861,  1.22971891,  1.87671942,  2.13166681,  1.59341682,
         0.59820902,  2.07280881,  2.06650141,  1.62369703,  1.48110007],
       [ 2.85484686,  1.72184337,  2.59603862, -1.46804331,  3.42167251,
         2.15689571,  1.65171003,  2.70204154,  1.82259584,  3.48014068]])
```

[18]:
```
#4(e)
Y = fifa22["preferred_foot"]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,␣
↪random_state=456)
print(X_train.head(3))
```

```
       shooting  passing  dribbling  defending  attacking  skill  movement  \
17219      24.0     43.0       48.0  59.000000       37.2   38.8      58.2
10931      46.0     61.0       70.0  58.666667       51.6   62.4      72.4
13667      57.0     59.0       64.0  35.333333       55.0   55.8      66.0

       power  mentality  goalkeeping
17219   53.6       42.5         11.2
10931   56.6       53.0         10.0
13667   58.6       52.5          7.6
```

[22]:
```
#4(f)
from sklearn.neighbors import KNeighborsClassifier

Y_train2 = Y_train.values.reshape(-1,1)
k_values = []
accuracy = []

for k in range(1, 31):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, Y_train2.flatten())
    pred_k = knn.predict(X_test)
```

```
    k_values.append(k)
    accuracy.append(metrics.accuracy_score(Y_test, pred_k) )

plt.figure(figsize = (12,6))
plt.plot(k_values, accuracy, color = 'blue', linestyle = 'solid', marker = 'o',␣
↪markerfacecolor = 'red', markersize = 10)
plt.xlabel('k')
plt.ylabel('Accuracy')
plt.title('KNN Classifier Accuracy for Different Values of k')
plt.show()
```



[23]:
```
#4(g)
knn = KNeighborsClassifier(n_neighbors= 28) #taking k = 28
knn.fit(X_train, Y_train)

# Predict preferred foot for X test data
Y_pred = knn.predict(X_test)

# Display first 3 predictions
print(Y_pred[:3])
```

```
['Right' 'Right' 'Right']
```

[24]:
```
#4(h)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred)
print(cm)
print("False positive:", cm[0,1])
```

```
[[  87 1239]
 [  86 3823]]
False positive: 1239
```

Hence, approximately 1239 players who actually prefer their left foot ("True Lefts") were predicted to prefer their right foot.

[25]:
```
#4(i)

print(metrics.classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

        Left       0.50      0.07      0.12      1326
       Right       0.76      0.98      0.85      3909

    accuracy                           0.75      5235
   macro avg       0.63      0.52      0.48      5235
weighted avg       0.69      0.75      0.67      5235
```

The low recall (=0.07) for the classification "Left" suggests that the model may not be identifying and classifying "Left" observations very well.

4(j) The report above shows that the model has an accuracy of 0.75 which is relatively high, indicating that the model does an overall good job of predicting a player's preferred foot. This is because the model is able to correctly predict the preferred foot about 75% of the time. However, there may be certain bias towards predicting that a player prefers their right foot based on the recall, indication that there may be some misclassifications for left-footed players.

[31]:
```
#Question 5
#5(a)

X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)
X_scaled_df.head()
```

[31]:
```
   shooting   passing  dribbling  defending  attacking     skill  movement  \
0  2.784312  3.296642   3.315358  -1.393049   3.400164  3.548580  2.774640
1  0.597719  1.229719   1.876719   2.131667   1.593417  0.598209  2.072809
2  2.854847  1.721843   2.596039  -1.468043   3.421673  2.156896  1.651710
3  1.232536  0.442320   1.054640   2.262906   0.905132  1.117771 -0.290023
4  2.784312  2.115543   2.390519  -1.074325   3.421673  2.379565  1.768682

      power  mentality  goalkeeping
0  1.944282   2.180614     0.281676
1  2.066501   1.623697     1.481100
2  2.702042   1.822596     3.480141
3  2.506491   2.140834     1.614369
4  2.799817   2.996099    -0.118132
```

[30]:
```
#5(b)
sampled_df = X_scaled_df.sample(n=5000, random_state=2022)
sampled_df.head()
```

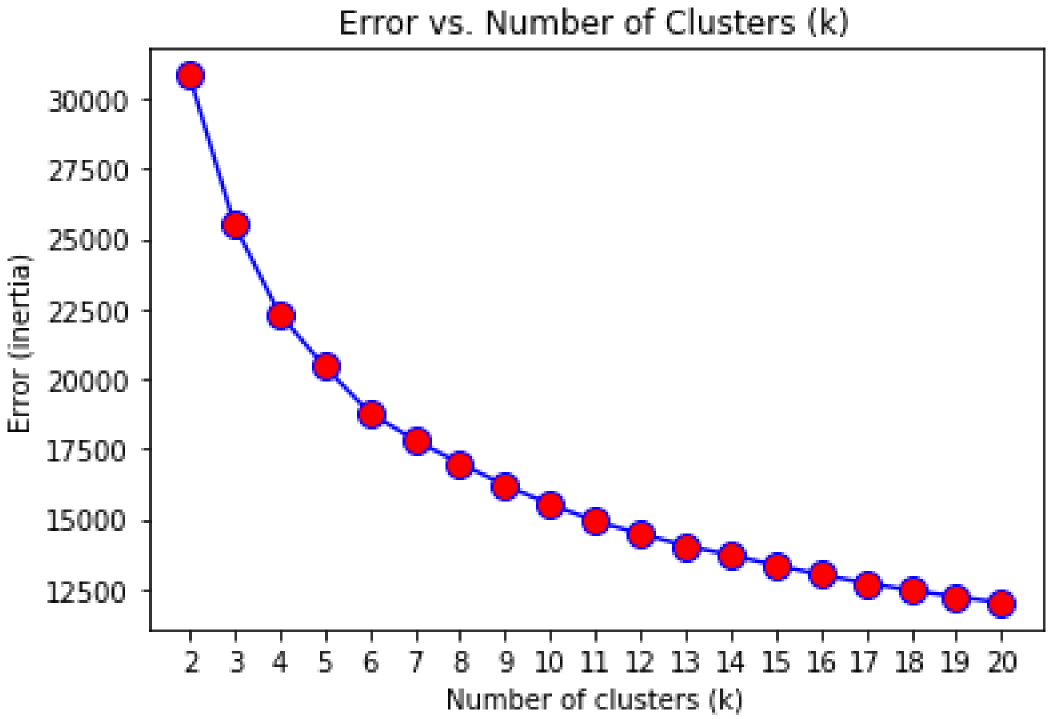
[30]:
```
       shooting   passing  dribbling  defending  attacking     skill  \
291    1.373606  2.115543   1.465680   1.550464   1.830015  1.748668
501    1.937889  0.934444   1.876719  -0.849343   1.959068  1.377552
8871   1.020930 -0.246654  -0.795038  -0.736852   1.120221 -0.979033
12793  0.456648 -0.345079   0.129801  -1.580534   0.173830 -0.552250
7256  -1.377269 -0.541929  -1.206077   0.763027  -0.600490 -1.591375

       movement     power  mentality  goalkeeping
291    0.037498  1.944282   2.180614     0.948023
501    2.049414  1.870951   1.404908     0.148406
8871  -1.576714  0.990972   0.310965     0.281676
12793  1.090245  1.039860  -0.703419    -1.051018
7256  -0.430389  0.013218  -0.206172     0.814753
```

[34]:
```
#5(c)
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

errors = []
silhouette = []

for k in range(2, 21):
    kmeans = KMeans(n_clusters=k, random_state=789)
    kmeans.fit(sampled_df)
    errors.append(kmeans.inertia_)
    silhouette.append(silhouette_score(sampled_df, kmeans.labels_))

print("Error values:", errors)
```

```
Error values: [30847.126857997708, 25514.51134236195, 22325.868999171504,
20446.193863352895, 18799.73556250355, 17818.103488273115, 16997.425516956362,
16215.351018257887, 15536.930308302319, 14936.765072641238, 14481.726819497208,
14059.160188748174, 13719.567782417698, 13359.277645318618, 13018.39026372852,
12724.739592119795, 12464.422720758232, 12236.010193739769, 12022.748945684381]
```

[35]:
```
#5(d)

plt.plot(range(2,21), errors, color='blue', linestyle='solid', marker='o',␣
 ↪markerfacecolor='red', markersize=10)
plt.xlabel('Number of clusters (k)')
plt.ylabel('Error (inertia)')
plt.title('Error vs. Number of Clusters (k)')
plt.xticks(np.arange(min(k_values), max(k_values)+1, 1.0))
```

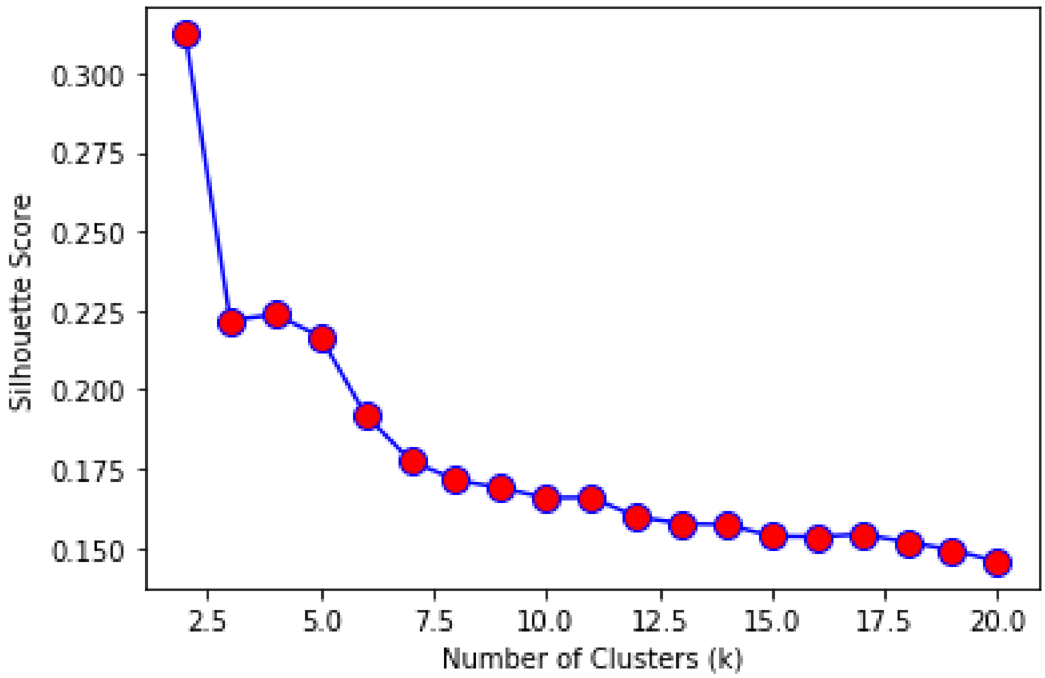
```
plt.show()
```



[36]:
```
#5(e)
from kneed import KneeLocator

kneedle = KneeLocator(range(2, 21), errors, S=1.0, curve="convex",␣
↪direction="decreasing")
elbow_k = kneedle.elbow

print("Elbow value:", elbow_k)
```

```
Elbow value: 6
```

[38]:
```
#5(f)
plt.plot(range(2, 21), silhouette, color='blue', linestyle='solid', marker='o',␣
↪markerfacecolor='red', markersize=10)
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Silhouette Score')
plt.show()
```



[53]:
```
#5(g)
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=6, random_state=789) #elbow value = 6

kmeans.fit(sampled_df)

cluster_labels = kmeans.labels_

sampled_df['Cluster'] = cluster_labels

sampled_df.head()
```

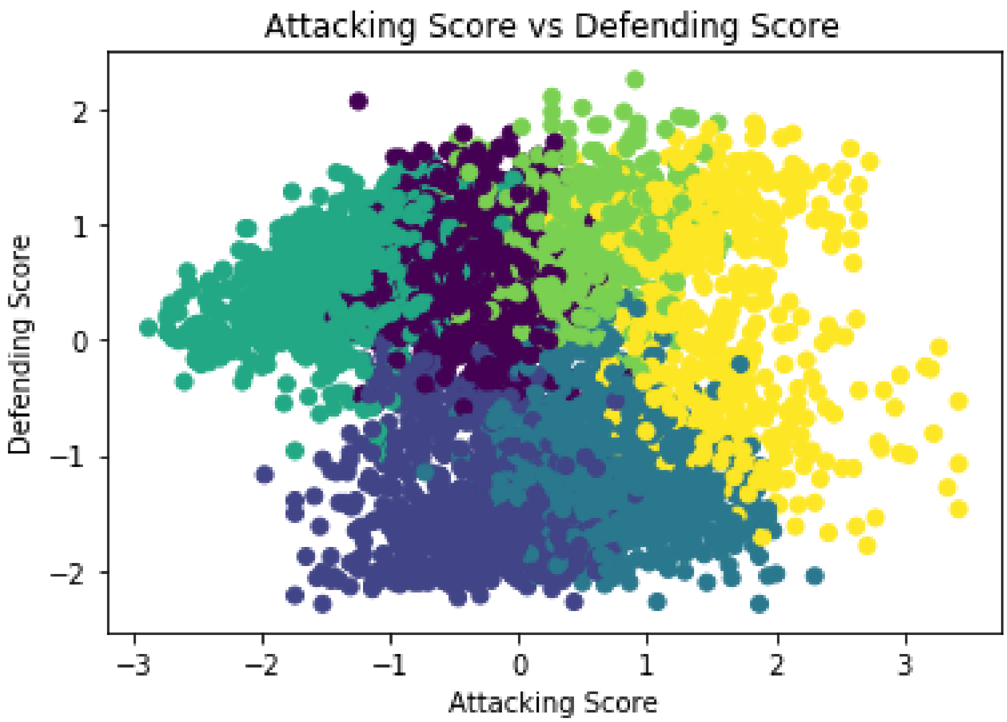
[53]:

	shooting	passing	dribbling	defending	attacking	skill
291	1.373606	2.115543	1.465680	1.550464	1.830015	1.748668
501	1.937889	0.934444	1.876719	-0.849343	1.959068	1.377552
8871	1.020930	-0.246654	-0.795038	-0.736852	1.120221	-0.979033
12793	0.456648	-0.345079	0.129801	-1.580534	0.173830	-0.552250
7256	-1.377269	-0.541929	-1.206077	0.763027	-0.600490	-1.591375

	movement	power	mentality	goalkeeping	Cluster
291	0.037498	1.944282	2.180614	0.948023	5
501	2.049414	1.870951	1.404908	0.148406	5
8871	-1.576714	0.990972	0.310965	0.281676	2
12793	1.090245	1.039860	-0.703419	-1.051018	2

```
7256  -0.430389  0.013218  -0.206172    0.814753       3
```

[55]:
```
#5(h)

plt.scatter(x=sampled_df['attacking'], y=sampled_df['defending'],␣
 ↪c=sampled_df['Cluster'], cmap='viridis')
plt.xlabel('Attacking Score')
plt.ylabel('Defending Score')
plt.title('Attacking Score vs Defending Score')
plt.show()
```



5(i) From the about plot, though there is some seperation betweeen the clusters, there is a fair amount of overlap as well, which indicates that clustering may not be the most meaningful technique for this data. Adittionally, the elbow mehtod and silhouette score did not provide a clear indication about what the value of k should have been, further suggesting that clustering may not be the best technique for this data

5(j) One analysis I would be interested in running would be exploring the relationships between different features in the dataset, for example, between the position of a player and his/her attacking, defending and passing skill or maybe even the relationship between wage and rank.