Homework 4

April 18, 2023

[1]:
```
import statsmodels.api as sm
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
#1a
data_file = pd.read_csv("fifa22.csv")
print(data_file.head(5))
```

```
                            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
```

1c) The unit of analyis is a soccer player.

[2]:
```
#1c
print("Number of observations:", data_file.shape[0])
print("Number of features:", data_file.shape[1])
```

```
Number of observations: 19630
Number of features: 20
```

[3]:
```
#1d
print(data_file["gender"].value_counts())
```

```
M    19239
F      391
Name: gender, dtype: int64
```

1e) No, This dataset isn't representative about the real-world population of professional footballe players as this dataset only includes characters/players from FIFA 2022, and not all professional football/soccer players in the real world. Therefore, it probably won't be able to capture the representative of the real-world players.

[4]:
```
#1f
data_file = data_file.dropna(subset=['passing'])
print(data_file.shape)
```

```
(17450, 20)
```

[5]:
```
#2
x = data_file[['passing', 'attacking', 'defending', 'skill']]
y = data_file['rank']
x = sm.add_constant(x)
model = sm.OLS(y,x)
results = model.fit()
print(results.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:                        19:02:50   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]
------------------------------------------------------------------------------
```

```
const          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.
```

```
/opt/conda/envs/dsua-111/lib/python3.7/site-
packages/numpy/core/fromnumeric.py:2542: FutureWarning: Method .ptp is
deprecated and will be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

2b) The R-squared value is a measure of the proportion of variance in rank by our features.It is 0.705 2c)Only attacking and defending are significant at the 1% level as their p values are less than 0.01 2d)A unit incresase is associated with 0.0066 unit increase in ranking

3a)(a) Based on the statsmodels output from Q2, we can expect the features to do a relatively good job at predicting rank. This is because the multiple regression model has a reasonably high R-squared value, indicating that a significant proportion of the variation in the dependent variable (rank) is explained by the independent variables (passing, attacking, defending, and skill).

[6]:
```python
#3b
X = data_file[['passing', 'attacking', 'defending', 'skill']]
Y = data_file['rank']
print(X.head())
print(Y.head())
```

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

[7]:
```
#3c
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=123)
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
```

[8]:
```
#3d)
lr_model = LinearRegression().fit(X_train, y_train)
print('Intercept:', lr_model.intercept_)
print('Coefficients:', lr_model.coef_)
```

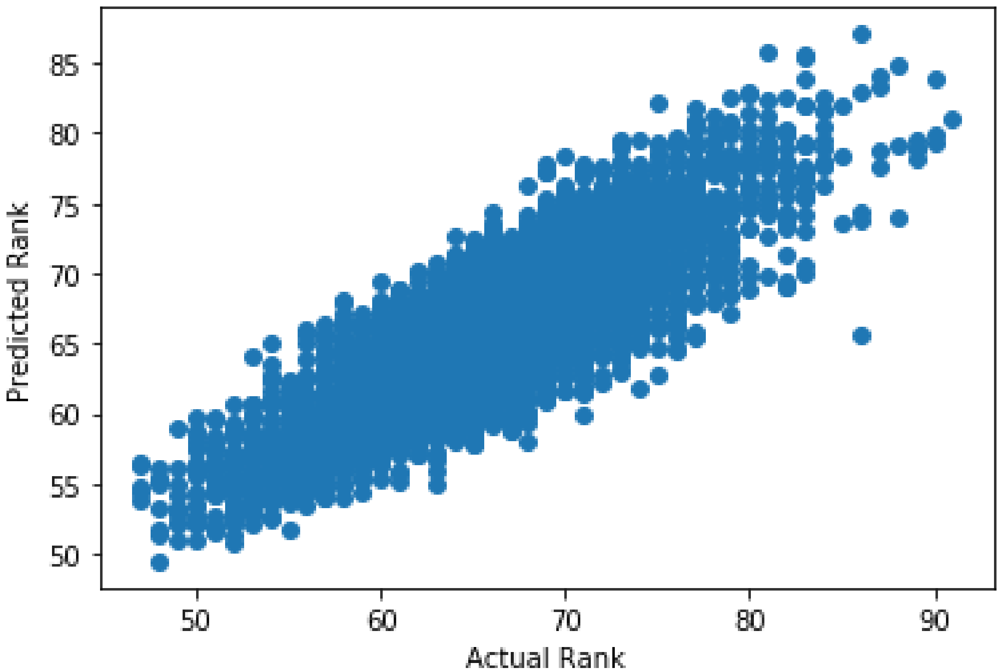
```
Intercept: 25.167733064621757
Coefficients: [-0.02444506  0.61230756  0.17314968  0.00612364]
```

3e) The coeffecients change slightly from 0.6109 in question 2 to 0.612 in question 3

[9]:
```
#3f
y_predicted = lr_model.predict(X_test)
print(y_predicted[:3])
```

```
[64.57617047 72.78035994 70.46341746]
```

[10]:
```
#3g
plt.scatter(y_test, y_predicted)
plt.xlabel('Actual Rank')
plt.ylabel('Predicted Rank')
plt.show()
```



[11]:
```
#3h
from sklearn import metrics
print('Root Mean Squared Error:',np.sqrt(metrics.
 ↪mean_squared_error(y_test,y_predicted)))
```

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

3h) The RMSE(roughly 3.75) is our measure of the average error and it indicates a deviation(error) average between the model and the data. 3i) Based on the analyses conducted above, it appears that the model performs reasonably well in predicting player rank. The multiple regression analysis in the previous question found that passing, attacking, defending, and skill were all significant predictors of rank, and the overall model had a relatively high R-squared value.

[12]:
```
#4a
print(data_file['preferred_foot'].value_counts())
```

```
Right    13044
Left      4406
Name: preferred_foot, dtype: int64
```

[13]:
```
#4b
right_foot_count = data_file['preferred_foot'].value_counts()['Right']
total_count = len(data_file)
percentage = right_foot_count / total_count * 100
print("Percentage of players who prefer their right foot:",percentage)
```

```
Percentage of players who prefer their right foot: 74.75071633237822
```

[14]:
```
#4c
X = data_file[['shooting', 'passing', 'dribbling', 'defending', 'attacking',
 ↪'skill', 'movement', 'power', 'mentality', 'goalkeeping']]
print(X.head())
```

```
   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
```

[15]:
```
#4d
from sklearn.preprocessing import StandardScaler

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

```
[[ 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]]
```

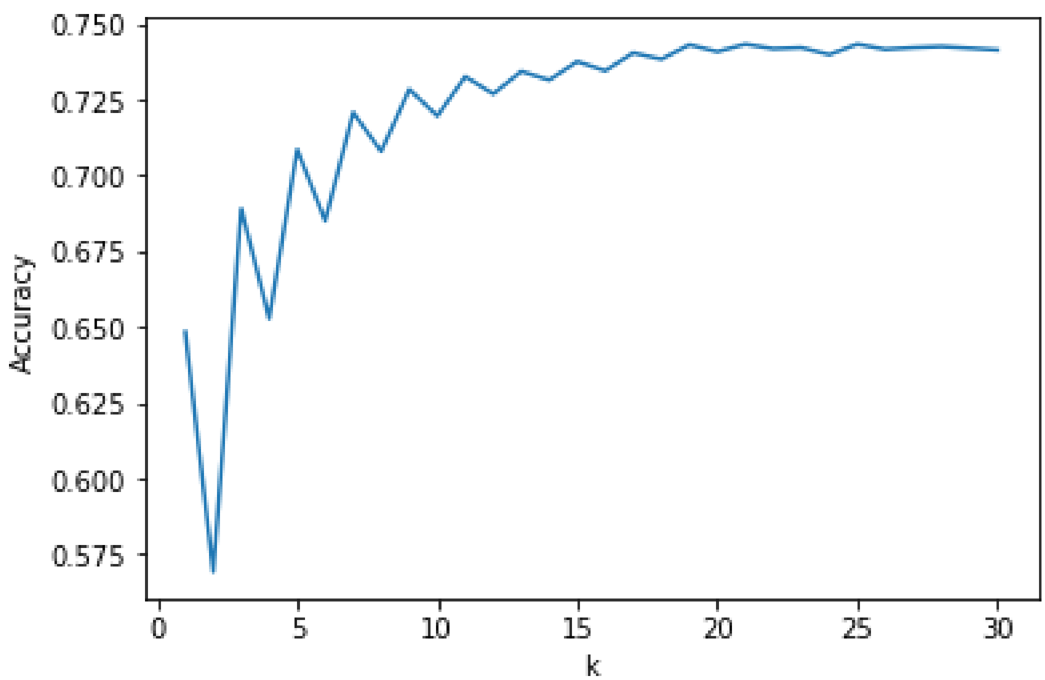
[16]:
```
#4e
from sklearn.model_selection import train_test_split

y = data_file['preferred_foot']
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.3,
 ↪random_state=456)
print(x_train[:3])
```

```
[[-2.01208594 -1.42775318 -1.51435684  0.44430269 -1.82649677 -1.57281886
  -0.96846017 -1.01342387 -1.55868391  0.54821445]
 [-0.46031026  0.34389488  0.74636066  0.4255542  -0.27785665  0.61676481
   0.69254058 -0.6467661  -0.30562141 -0.25140179]
 [ 0.31557757  0.1470451   0.12980134 -0.88684012  0.08779448  0.00442362
```

-0.05607947 -0.40232758 -0.36529105 -1.85063428]]

[17]:

```
#4f
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

errors = list()
accuracy = list()
for k in range(1, 31):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(x_train, y_train)
    pred_k = knn.predict(x_test)
    accuracy.append(metrics.accuracy_score(y_test,pred_k))

plt.plot(range(1, 31), accuracy, label="accuracy rate k value")
plt.xlabel("k")
plt.ylabel("Accuracy")
plt.show()
```



[ ]:

```
```

[18]:

```
k = 5
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(x_train, y_train)
test_preds = knn.predict(x_test)
```

```python
print(test_preds[:3])
```

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

[19]:
```python
from sklearn import metrics
confusion_matrix = metrics.confusion_matrix(y_test, test_preds)
classification_matrix = metrics.classification_report(y_test,test_preds)
print("Confusion Matrix:")
print(confusion_matrix)
print("Classification Matrix:")
print(classification_matrix)
```

```
Confusion Matrix:
[[ 206 1120]
 [ 406 3503]]
Classification Matrix:
              precision    recall  f1-score   support

        Left       0.34      0.16      0.21      1326
       Right       0.76      0.90      0.82      3909

    accuracy                           0.71      5235
   macro avg       0.55      0.53      0.52      5235
weighted avg       0.65      0.71      0.67      5235
```

4h) Approximately, there were 1120 predicted left,however there were only 206 true lefts. So there were an additional 914 lefts predicted 4i)The recall for left is relatively low, suggesting a poor prection model. 4j) The model did a poor job in predicting the left foot, however for the right foot the recall was high as 0.90. Overall, a poor job.

[20]:
```python
import pandas as pd
X_scaled = pd.DataFrame(x_scaled)
print(X_scaled.head())
```

```
          0         1         2         3         4         5         6  \
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

          7         8         9
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
```

[21]:
```
np.random.seed(2022)
sampled_data = X_scaled.sample(n=5000, random_state=2022)
print(sampled_data.head())
```

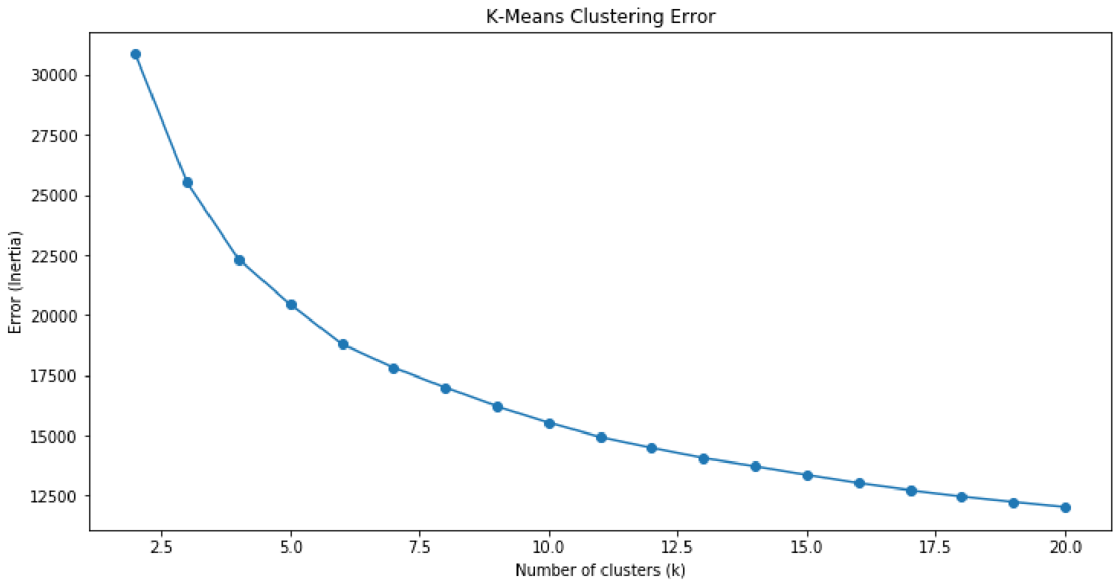
```
              0         1         2         3         4         5         6  \
291    1.373606  2.115543  1.465680  1.550464  1.830015  1.748668  0.037498
501    1.937889  0.934444  1.876719 -0.849343  1.959068  1.377552  2.049414
8871   1.020930 -0.246654 -0.795038 -0.736852  1.120221 -0.979033 -1.576714
12793  0.456648 -0.345079  0.129801 -1.580534  0.173830 -0.552250  1.090245
7256  -1.377269 -0.541929 -1.206077  0.763027 -0.600490 -1.591375 -0.430389

              7         8         9
291    1.944282  2.180614  0.948023
501    1.870951  1.404908  0.148406
8871   0.990972  0.310965  0.281676
12793  1.039860 -0.703419 -1.051018
7256   0.013218 -0.206172  0.814753
```

[22]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
inertias = []
silhouette_scores = []
k_values = range(2, 21)
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=789)
    kmeans.fit(sampled_data)


    inertia = kmeans.inertia_
    silhouette = silhouette_score(sampled_data, kmeans.labels_)


    inertias.append(inertia)
    silhouette_scores.append(silhouette)
print("error:",inertias)
```

```
error: [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]
```

[23]:
```
plt.figure(figsize=(12,6))
plt.plot(range(2, len(inertias)+2), inertias, marker='o')
plt.title('K-Means Clustering Error')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Error (Inertia)')
```

```
plt.show()
```

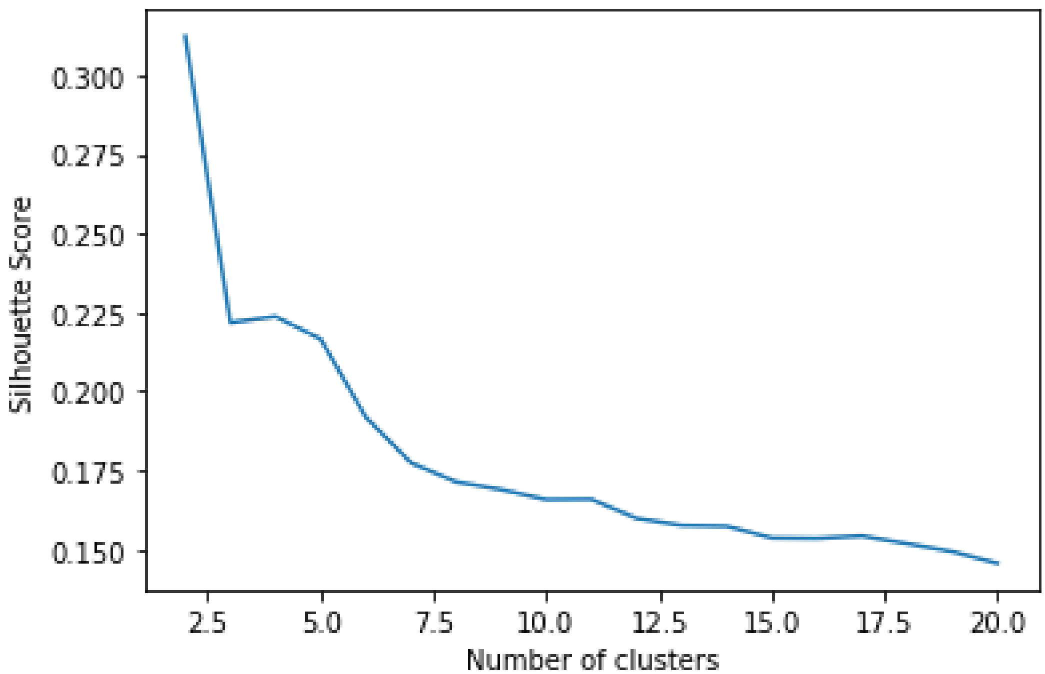


[27]:
```
import sys
!{sys.executable} -m pip install kneed
import kneed
from kneed import KneeLocator

kl = KneeLocator(range(2, 21), inertias, curve='convex', direction='decreasing')
elbow = kl.elbow


print("The suggested elbow value is:",elbow)
```

```
Requirement already satisfied: kneed in
/opt/conda/envs/dsua-111/lib/python3.7/site-packages (0.7.0)
Requirement already satisfied: matplotlib in
/opt/conda/envs/dsua-111/lib/python3.7/site-packages (from kneed) (3.1.2)
Requirement already satisfied: numpy>=1.14.2 in
/opt/conda/envs/dsua-111/lib/python3.7/site-packages (from kneed) (1.18.1)
Requirement already satisfied: scipy in
/opt/conda/envs/dsua-111/lib/python3.7/site-packages (from kneed) (1.2.1)
Requirement already satisfied: python-dateutil>=2.1 in
/opt/conda/envs/dsua-111/lib/python3.7/site-packages (from matplotlib->kneed)
(2.8.1)
Requirement already satisfied: cycler>=0.10 in
/opt/conda/envs/dsua-111/lib/python3.7/site-packages (from matplotlib->kneed)
(0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
```

```
/opt/conda/envs/dsua-111/lib/python3.7/site-packages (from matplotlib->kneed)
(2.4.6)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/envs/dsua-111/lib/python3.7/site-packages (from matplotlib->kneed)
(1.1.0)
Requirement already satisfied: six>=1.5 in
/opt/conda/envs/dsua-111/lib/python3.7/site-packages (from python-
dateutil>=2.1->matplotlib->kneed) (1.13.0)
Requirement already satisfied: setuptools in
/opt/conda/envs/dsua-111/lib/python3.7/site-packages (from
kiwisolver>=1.0.1->matplotlib->kneed) (44.0.0.post20200106)
The suggested elbow value is: 6
```

[30]:
```
plt.plot(range(2, len(silhouette_scores)+2), silhouette_scores)
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
plt.show()
```



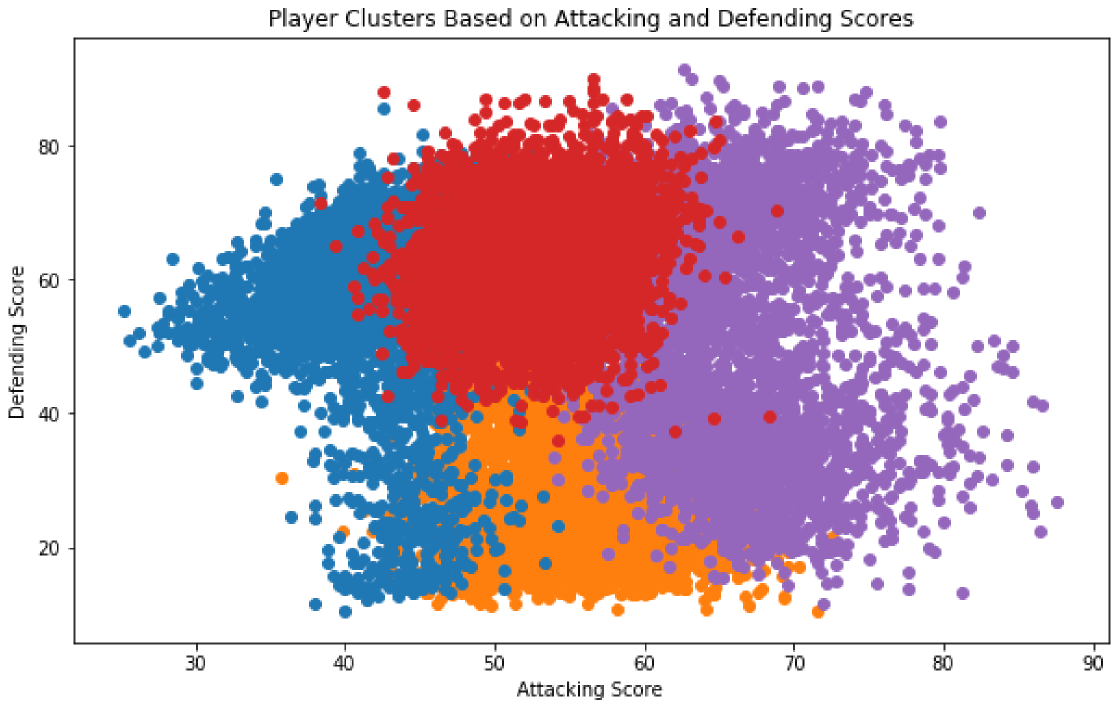
[41]:
```
from sklearn.cluster import KMeans
k = 4

kmeans = KMeans(n_clusters=k)

kmeans.fit(x_scaled)
labels = kmeans.labels_
```

```
X['cluster_label'] = labels

print(X.head())
```

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

```
/opt/conda/envs/dsua-111/lib/python3.7/site-packages/ipykernel_launcher.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  if __name__ == '__main__':
```

[46]:
```
colors={
    0:'C1',
    1:'C0',
    2:'C4',
    3:'C3'
}
plt.rc('figure',figsize=(10,6))
for i in range(4):
    subset = X[X['cluster_label']==i]
    plt.scatter(subset['attacking'],subset['defending'],label='Cluster{}'.
 ↪format(i),color=colors[i])

plt.xlabel('Attacking Score')
plt.ylabel('Defending Score')
plt.title('Player Clusters Based on Attacking and Defending Scores')
plt.show()
```



5i)Based on the analysis, clustering seems to be a meaningful technique for this data. The elbow plot suggested that a reasonable value for k would be 4 and the Silhouette Score plot shows that a k value of 4 has a relatively high Silhouette Score. 5j)I would have liked to run more clustering algorithms to see if they provided better results. This would helped in visualizing data in a better way.