DS-UA 111 HW3

August 9, 2023

[1]:
```python
# part a
import pandas as pd
fifa = pd.read_csv('fifa22.csv')
fifa.head()
```

[1]:
```
                            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
```

b) The unit of analysis is FIFA players.

[2]:
```python
# part c
fifa.shape
```

[2]: (19630, 20)

There are 19630 observations and 20 features.

[3]:
```python
# part d
fifa['gender'].value_counts()
```

[3]:
```
M    19239
F      391
Name: gender, dtype: int64
```

There are 19239 male players and 391 female players.

e) I don't believe the dataset is representative of the real-world population of professional football/soccer players. There may be professional players who didn't agree to be included in the game.

[4]:
```python
# part f
for i in range(len(fifa['passing'])):
    if fifa['passing'].isna()[i]:
        fifa.drop(i, inplace = True)
fifa
```

[4]:
```
                                name  rank gender  wage_eur   log_wage  \
0      Lionel Andrés Messi Cuccittini    93      M  320000.0  12.676076
1          Lucia Roberta Tough Bronze    92      F       NaN        NaN
2                    Vivianne Miedema    92      F       NaN        NaN
3           Wéndèleine Thérèse Renard    92      F       NaN        NaN
4                  Robert Lewandowski    92      M  270000.0  12.506177
…                                  …   …    …      …         …
19624                 Caoimhin Porter    47      M     500.0   6.214608
19625         Nathan Logue-Cunningham    47      M     500.0   6.214608
19626                     Luke Rudden    47      M     500.0   6.214608
19627         Emanuel Lalchhanchhuaha    47      M     500.0   6.214608
19628             Nathan-Dylan Saliba    47      M     500.0   6.214608

      position          nationality                 club  \
0           RW            Argentina  Paris Saint-Germain
1          NaN              England                  NaN
2          NaN          Netherlands                  NaN
3          NaN               France                  NaN
4           ST               Poland    FC Bayern München
…          …                  …                    …
19624      RES  Republic of Ireland           Derry City
19625      RES  Republic of Ireland           Finn Harps
19626      RES  Republic of Ireland           Finn Harps
19627      SUB                India   NorthEast United FC
19628      RES               Canada  Club de Foot Montréal
```

```
                              league preferred_foot  shooting  passing  \
0                     French Ligue 1           Left      92.0     91.0
1                                NaN          Right      61.0     70.0
2                                NaN          Right      93.0     75.0
3                                NaN          Right      70.0     62.0
4               German 1. Bundesliga          Right      92.0     79.0
…                                …            …       …       …
19624  Rep. Ireland Airtricity League          Right      39.0     50.0
19625  Rep. Ireland Airtricity League          Right      37.0     45.0
19626  Rep. Ireland Airtricity League          Right      46.0     36.0
19627              Indian Super League          Right      38.0     45.0
19628          USA Major League Soccer          Right      36.0     46.0

       dribbling  defending  attacking  skill  movement  power  mentality  \
0           95.0  26.333333       85.8   94.0      90.2   77.8  73.833333
1           81.0  89.000000       69.0   62.2      84.2   78.8  69.166667
2           88.0  25.000000       86.0   79.0      80.6   84.0  70.833333
3           73.0  91.333333       62.6   67.8      64.0   82.4  73.500000
4           86.0  32.000000       86.0   81.4      81.6   84.8  80.666667
…            …          …         …      …       …      …        …
19624       46.0  42.666667       43.2   43.4      60.0   48.8  46.500000
19625       49.0  43.333333       40.0   43.8      56.6   49.4  42.500000
19626       48.0  11.666667       38.0   38.0      65.8   45.8  38.500000
19627       48.0  33.666667       40.8   41.0      68.2   49.0  43.500000
19628       49.0  43.666667       37.2   42.4      62.2   46.6  45.333333

       goalkeeping
0             10.8
1             12.6
2             15.6
3             12.8
4             10.2
…             …
19624          9.4
19625          7.4
19626         10.6
19627         11.4
19628         11.4

[17450 rows x 20 columns]
```

1 Question 2

[5]:
```
# part a
import statsmodels.formula.api as smf
X = fifa[['passing', 'attacking', 'defending', 'skill']]
y = fifa['rank']
multiple_regression = smf.ols('rank ~ passing + attacking + defending + skill',␣
 ↪data = fifa).fit()
multiple_regression.summary()
```

[5]:
```
<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:                Wed, 09 Aug 2023   Prob (F-statistic):               0.00
Time:                        18:40:58   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.
"""
```

b) 0.705
c) passing and skill
d) 0.0066

2 Question 3

a) I expect that the four features will do a good job predicting the rank. The R-squared value for the model was 0.705, which is a high value.

[6]:
```
# part b
X = pd.DataFrame(data = fifa[['passing','attacking', 'defending', 'skill']])
X.head()
```

[6]:

	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

[7]:
```
y = pd.DataFrame(data = fifa[['rank']])
y.head()
```

[7]:

	rank
0	93
1	92
2	92
3	92
4	92

[8]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

X = fifa[['passing','attacking', 'defending', 'skill']]
y = fifa[['rank']]
x_train_regress, x_test_regress, y_train_regress, y_test_regress =␣
↪train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=123)
x_train_regress.head()
```

[8]:

	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

[9]:
```
# part d
from sklearn import linear_model
linear_regression = linear_model.LinearRegression()
```

```
linear_regression.fit(x_train_regress,y_train_regress)
print(linear_regression.coef_)
```

```
[[-0.02444506  0.61230756  0.17314968  0.00612364]]
```

[10]:
```
print(linear_regression.intercept_)
```

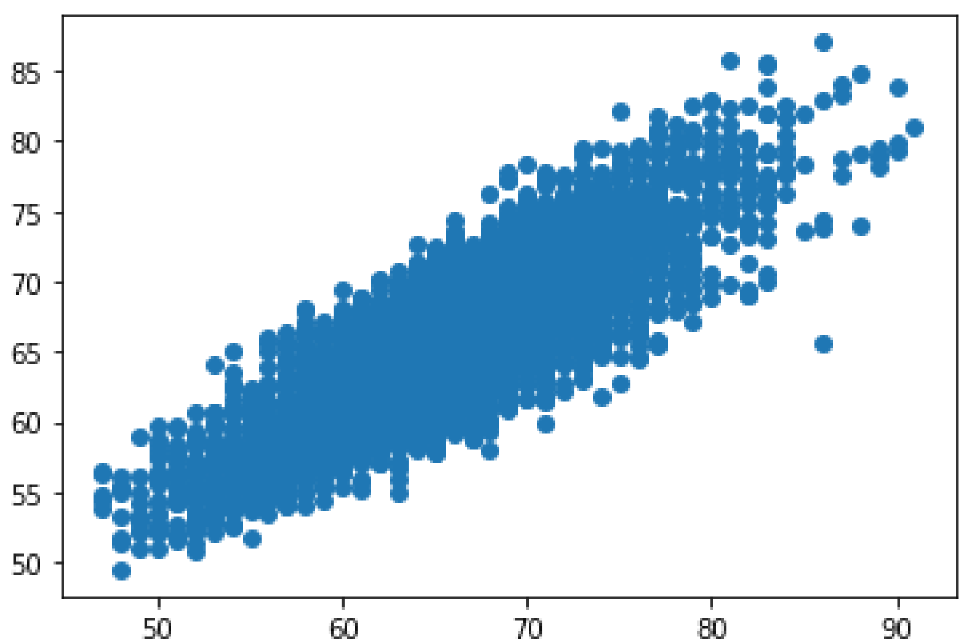
```
[25.16773306]
```

e) The “attacking: coefficient slightly increased from 0.6109 in question 2, to 0.61230756 in question 3.

[11]:
```
# part f
pred_array = linear_regression.predict(x_test_regress)
for i in range(3):
    print(pred_array[i])
```

```
[64.57617047]
[72.78035994]
[70.46341746]
```

[12]:
```
# part g
import matplotlib.pyplot as plt
plt.scatter(y_test_regress, pred_array)
plt.show()
```



[13]:
```python
# part h
import numpy as np
error = y_test_regress - pred_array
squared_error = pow(error, 2)
squared_error_mean = np.mean(squared_error)
RSME = np.sqrt(squared_error_mean)
RSME
```

[13]:
```
rank    3.744563
dtype: float64
```

h) The Root Mean Squared Error is the "average error" for a prediction model. Therefore, the "average error" of the predicted ranks is 3.74 ranks off the actual ranks in this model.

i) I think the model does a good job predicting the player rank.

3 Question 4

[14]:
```python
# part a
fifa['preferred_foot'].value_counts()
```

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

b) 74.8%

[30]:
```python
# part c
X = fifa[['shooting', 'passing', 'dribbling', 'defending', 'attacking', 'skill',
          'movement', 'power', 'mentality', 'goalkeeping']]
X.head()
```

[30]:
```
   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
```

[31]:
```
# part d
X_scalar = StandardScaler().fit(X)
scaled_X = X_scalar.transform(X)
pd.DataFrame(scaled_X).head()
```

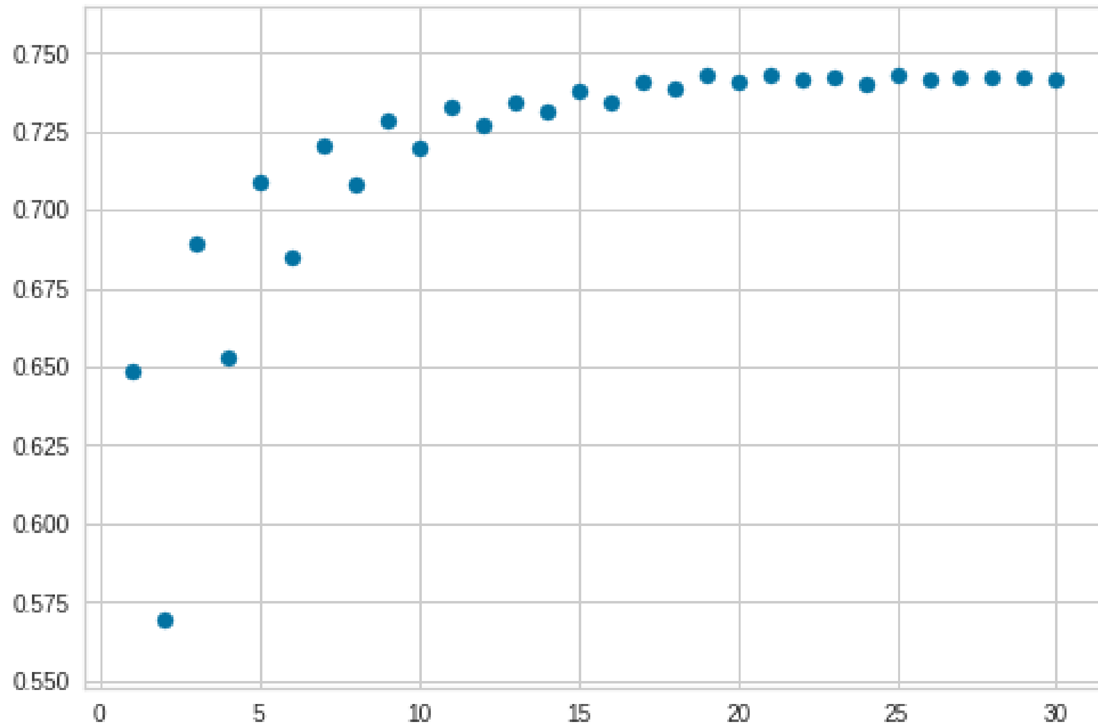
[31]:
```
          0         1         2         3         4         5         6  \
0  2.784312  3.296642  3.315358 -1.393049  3.400164  3.548580  2.774640
1  0.597719  1.229719  1.876719  2.131667  1.593417  0.598209  2.072809
2  2.854847  1.721843  2.596039 -1.468043  3.421673  2.156896  1.651710
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
```

[32]:
```
# part e
Y = fifa['preferred_foot']
x_train_regress, x_test_regress, y_train_regress, y_test_regress =␣
 ↪train_test_split(scaled_X, Y,
                                                               test_size=0.30,
                                                               random_state=456)
pd.DataFrame(x_train_regress).head()
```

[32]:
```
          0         1         2         3         4         5         6  \
0 -2.012086 -1.427753 -1.514357  0.444303 -1.826497 -1.572819 -0.968460
1 -0.460310  0.343895  0.746361  0.425554 -0.277857  0.616765  0.692541
2  0.315578  0.147045  0.129801 -0.886840  0.087794  0.004424 -0.056079
3  1.655747  2.017118  0.951880 -0.286888  1.206257  2.064117  0.552174
4 -1.871015  0.245470  0.027041  1.137997 -0.578981 -0.181134 -0.453784

          7         8         9
0 -1.013424 -1.558684  0.548214
1 -0.646766 -0.305621 -0.251402
2 -0.402328 -0.365291 -1.850634
3  0.599870  0.788322  1.614369
4 -0.744542  0.450194  0.281676
```

[33]:
```
# part f
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans

error = list()
accuracy = list()
```

```
for k in range(1, 31):
    knn = KNeighborsClassifier(n_neighbors=k) # init our knn object
    knn.fit(x_train_regress, y_train_regress)     # fit the model
    y_pred = knn.predict(x_test_regress) # get our predictions
    accuracy.append(metrics.accuracy_score(y_test_regress, y_pred)) # record␣
 ↪accuracy
plt.scatter(np.arange(1,31),accuracy)
plt.show()
```



[34]:
```
# part g
k = 25
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(x_train_regress, y_train_regress)
y_pred = knn.predict(x_test_regress)
for i in range(3):
    print(y_pred[i])
```

```
Right
Right
Right
```

[35]:
```
# part h
print(metrics.confusion_matrix(y_test_regress, y_pred))
```

```
[[  48 1278]
 [  66 3843]]
```

1278 players

[36]:
```python
# part i
print(metrics.classification_report(y_test_regress, y_pred))
```

```
              precision    recall  f1-score   support

        Left       0.42      0.04      0.07      1326
       Right       0.75      0.98      0.85      3909

    accuracy                           0.74      5235
   macro avg       0.59      0.51      0.46      5235
weighted avg       0.67      0.74      0.65      5235
```

Recall in a classification report means the percentage of a correct predictions out of all actual observations of a dependent varible. So having a recall of 0.04 for the classification "Left" suggests that out of all the actual "left foot" players in the testing set, 4% were correctly predicted to be "left foot" players.

j) I think the model did a bad job of predicting a player's preferred foot.

4 Question 5

[37]:
```python
# part a
pd.DataFrame(scaled_X).head()
```

[37]:
```
          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
```

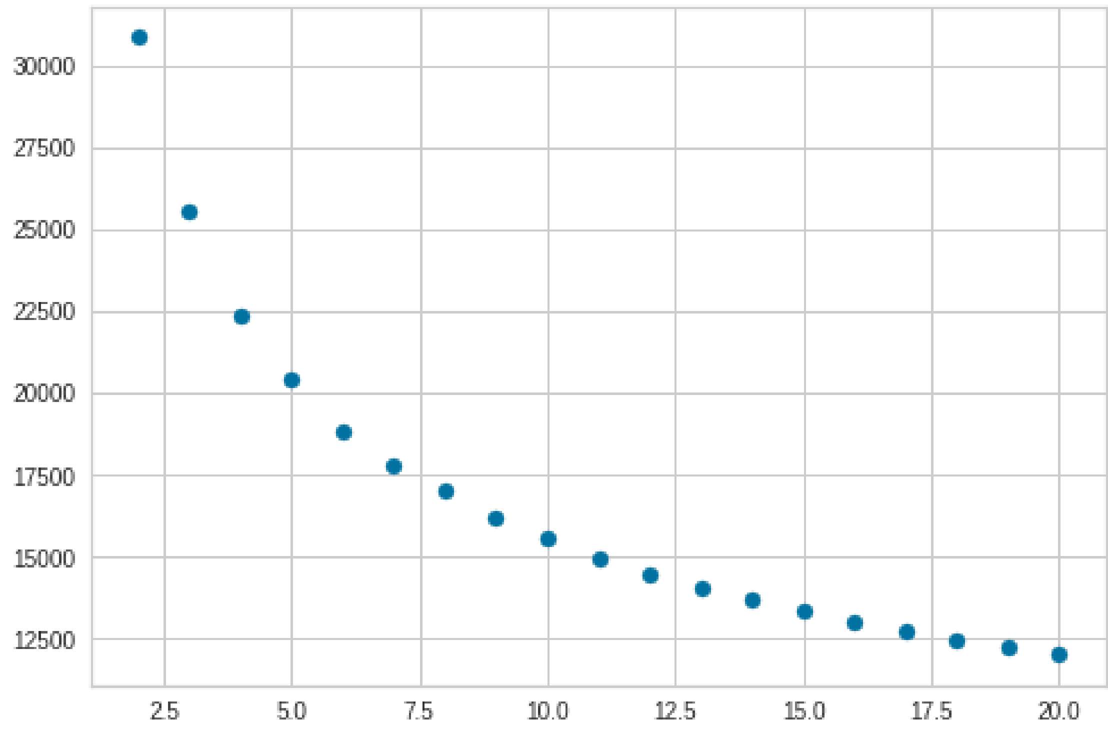
[38]:
```python
# part b
X_sample = pd.DataFrame(scaled_X).sample(n=5000, random_state=2022)
X_sample.head()
```

[38]:
```
              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
```

[39]:
```python
# part c

error = list()  # to save error
sil = list()  # to save sillhoute score

for k in range(2,21):
    kmeans = KMeans(n_clusters=k, random_state=789) # init k-means object
    kmeans.fit(X_sample)  # run k-means!

    error.append(kmeans.inertia_) # save sum of squared error (SSE)

    sil_score = metrics.silhouette_score(X_sample, kmeans.labels_) # calc␣
↪silhouette score
    sil.append(sil_score) # save score
print(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]
```

[40]:
```python
# part d
plt.scatter(np.arange(2,21), error)
plt.show()
```

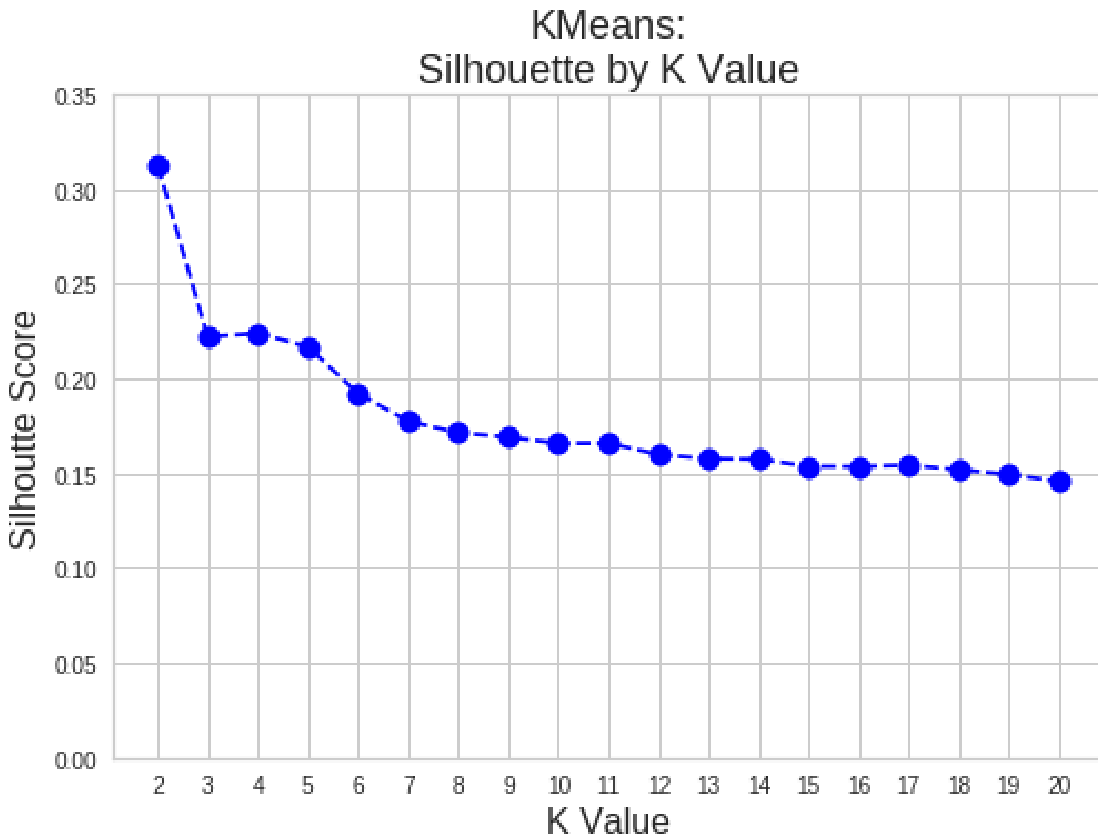


[41]:
```
# part e
from kneed import KneeLocator # for KMeans, elbow method
from yellowbrick.cluster import SilhouetteVisualizer # for KMeans, silhouette␣
↪scores

kl = KneeLocator(np.arange(2, 21), error)

print(kl.elbow)
```

2

[42]:
```
# part f
plt.plot(range(2, len(sil)+2), sil, color='blue', linestyle='dashed',␣
↪marker='o',
         markersize=10)

plt.yticks(np.arange(0, .40, .05)) # start y ticks at 0
plt.xticks(np.arange(2, 21, 1))  # integer x ticks

# title & label axes
plt.title('KMeans:\nSilhouette by K Value', size=18)
plt.xlabel('K Value', size=16)
plt.ylabel('Silhoutte Score', size=16)
```

```
plt.show()
```

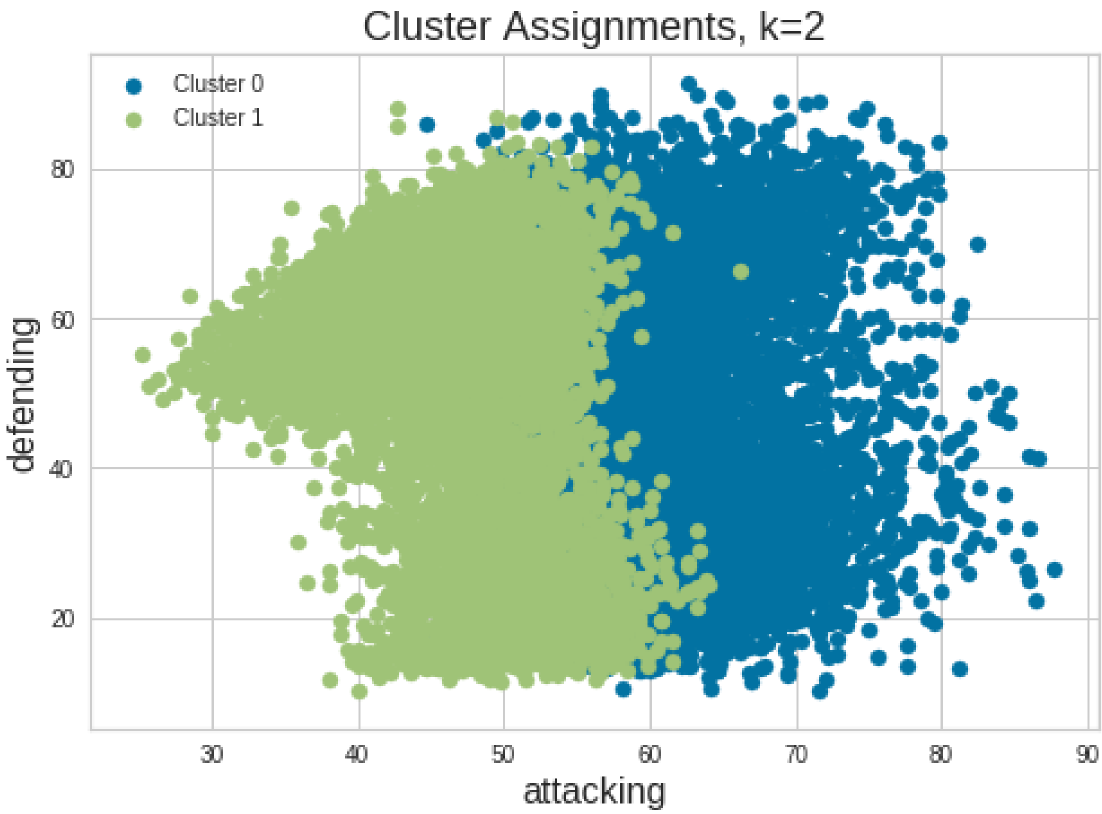


[43]:
```
# part g
k = 2

# Create KMeans instance for k clusters
kmeans = KMeans(n_clusters=k, random_state=42).fit(scaled_X)
fifa['cluster'] = kmeans.labels_
fifa.head()
```

[43]:
```
                            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  …

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

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

[5 rows x 21 columns]
```

[44]:
```
# part h
for i in range(k):
    # subset to just observations in this cluster
    subset = fifa[fifa['cluster'] == i]

    # plot this cluster
    # Each call to .scatter() will auto get a different color
    plt.scatter(subset['attacking'],
                subset['defending'],
                label='Cluster {}'.format(i))

# labels, etc
plt.xlabel('attacking', size=16)
plt.ylabel('defending', size=16)

plt.title('Cluster Assignments, k={}'.format(k), size=18)
plt.legend(loc='upper left', prop={'size': 10})

plt.show()
```



i) I would say clustering isn't a meaningful technique, because clusters can overlap (which is what happened in Q5h).

j) I would be interested in running an analysis on different engineering fields, since it could help me determine which engineering fields are the best to work in.

5 Citations

1) https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html; used this source to make data frames

2) https://www.w3schools.com/python/pandas/pandas_cleaning_wrong_data.asp; used this source to help drop rows

3) https://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares; used this source to find the coeffciient and intercept for a linear regression with sklearn

4) https://www.statology.org/sklearn-classification-report/; used this source to find the definition for recall in a classification report

5) https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html; used this source to learn how to use panda's sample function