Project 3

Sareh Jalalizad

Question 1

Code

```
# Get statistics on the dataset.
# Write your code between the lines (~ 1 line)
#####################################################
dataset_statistics = dataset.describe()
#####################################################
```

Question 2

Code and output

```
# Split the dataset into input features and the output target.
# Write your code between the lines (~ 2 line)
#################################################
y_dataset = dataset.pop('Country_Name')
X_dataset = dataset
#################################################
print("Dataset separated.")
```

```
Dataset separated.
```

Question 3

Code and Output

```
# Convert raw values to their Z-scores
# Calculate the Z-scores of each input feature column.
# Write your code between the lines (~ 3 lines)
#################################################
X_dataset_mean = X_dataset.mean()
X_dataset_std = X_dataset.std()
X_dataset_norm = (X_dataset - X_dataset_mean) / X_dataset_std
#################################################
print("Dataset normalized.")
```

```
Dataset normalized.
```

Question 4

Code and Output

```
from sklearn.decomposition import PCA

# Set number of components to 2 and then fit the PCA model.
# Write your code between the lines (~ 2 line)
#################################################
pca = PCA(n_components=2)
pca.fit(X_dataset_norm)
#################################################
```

```
PCA
PCA(n_components=2)
```

Question 5

Code and Output

```
eigenvectors = pca.components_

top_indicators_pc1 = np.argsort(np.abs(eigenvectors[0]))[::-1][:4]
top_indicators_pc2 = np.argsort(np.abs(eigenvectors[1]))[::-1][:4]

# Get the names
top_indicators_names_pc1 = X_dataset.columns[top_indicators_pc1]
top_indicators_names_pc2 = X_dataset.columns[top_indicators_pc2]

print("Top four development indicators for Principal Component 1:")
print(top_indicators_names_pc1)

print("\nTop four development indicators for Principal Component 2:")
print(top_indicators_names_pc2)
```

```
Top four development indicators for Principal Component 1:
Index(['GDP_per_capita_KUSD', 'GNI_per_capita_KUSD',
       'Life_expectancy_at_birth_total_years',
       'Inflation_consumer_prices_annual_pct'],
      dtype='object')

Top four development indicators for Principal Component 2:
Index(['GDP_current_TUSD', 'Total_tax_and_contribution_rate_pct_of_profit',
       'Exports_of_goods_and_services_pct_of_GDP', 'GDP_growth_annual_pct'],
      dtype='object')
```

Question 6

Code and output

```
# Find the total ratio of the retained variance.
# Write your code between the lines (~ 1 line)
################################################
total_explained_variance_ratio = np.sum(pca.explained_variance_ratio_)
################################################
print(total_explained_variance_ratio)
```

```
0.5657820657915604
```

Question 7

Code and output

```
[18] # Transform the normalized dataset.
     # Write your code between the lines (~ 1 line)
     ################################################
     X_pca = pca.transform(X_dataset_norm)
     ################################################
     print("original shape:   ", X_dataset_norm.shape)
     print("transformed shape:", X_pca.shape)
```

```
original shape:    (27, 10)
transformed shape: (27, 2)
```

Question 8

Code and output

```
from sklearn.cluster import KMeans

# Define the k-means clustering model and fit the model.
# Write your code between the lines (~ 2 lines)
################################################
kmeans = KMeans(n_clusters=10, random_state=0)
kmeans.fit(X_dataset_)
################################################
```

```
KMeans(n_clusters=10, random_state=0)
```

Question 9

Code and output

```
# Find cluster index for each data point.
# Write your code between the lines (~ 1 line)
#################################################
clusters = kmeans.predict(X_dataset_)
#################################################

kmeans.cluster_centers_.shape
```

```
(10, 64)
```

Question 10

Code and output

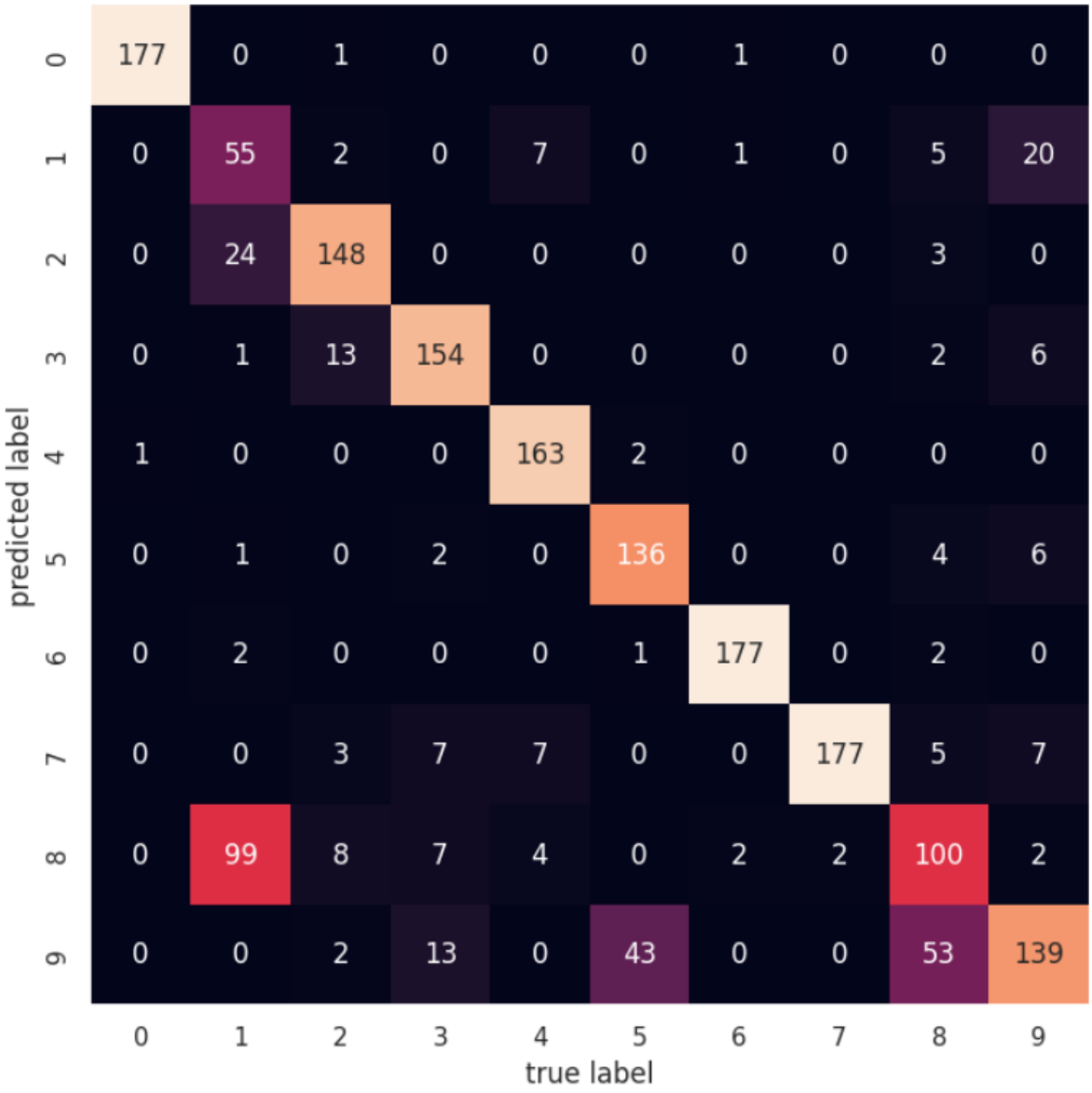
```
from sklearn.metrics import accuracy_score

# Find the accuracy score of the k-means clustering by comparing the y_dataset_
# and labels from the previous cell.
# Write your code between the lines (~ 1 line)
#################################################
kmeans_accuracy = accuracy_score(y_dataset_, labels)
#################################################
print(kmeans_accuracy)
```

```
0.7935447968836951
```

Question 11

According to the confusion matrix, the model had the most difficulty recognizing the digit 1, by predicting 8.


predicted label
true label
0 1 2 3 4 5 6 7 8 9
177 0 1 0 0 0 1 0 0 0
0 55 2 0 7 0 1 0 5 20
0 24 148 0 0 0 0 0 3 0
0 1 13 154 0 0 0 0 2 6
1 0 0 0 163 2 0 0 0 0
0 1 0 2 0 136 0 0 4 6
0 2 0 0 0 1 177 0 2 0
0 0 3 7 7 0 0 177 5 7
0 99 8 7 4 0 2 2 100 2
0 0 2 13 0 43 0 0 53 139