**Logo

Description automatically generated**

Mahmud\_19660\_hw3

**San Francisco Bay University**

**CS483 - Fundamentals of Artificial Intelligence**

**Homework Assignment #3**

**Due day: 7/9/2022**

**Instruction:**

1. **Push the source code to Github**
2. **Overdue homework submission could not be accepted.**
3. **Take academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**
4. Confusion matrix is the table to present the performance of an algorithm for the classification. Assuming that the example of 3 by 3 confusion matrix comes from the outputs of 3 clusters classification as follows, please find the total accuracy, and each cluster’s precision, recall and F1-score.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | Predicted Values | | |
| Cat | Dog | Bird |
| Actual Values | Cat | 20 | 1 | 1 |
| Dog | 1 | 19 | 2 |
| Bird | 2 | 0 | 28 |

**Solution:**

Total accuracy = = = 0.9054 = 90.54%

Table

Description automatically generated

**Cat:**

TP = Cell\_11 = 20

FP = Cel1\_21 + Cel1\_21 = 3

TN = Cel1\_22 + Cel1\_23 + Cel1\_32 + Cel1\_33 = 49

FN = Cel1\_12 + Cel1\_13= 2

= = 0.9324 = 93.24%

=

= = 0.909 = 90.9%

F1\_Score = = = 0.8888 = 88.88%

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | Predicted Values | | |
| Cat | Dog | Bird |
| Actual Values | Cat | 2 | 1 | 1 |
| Dog | 1 | 19 | 2 |
| Bird | 2 | 0 | 28 |

**Dog:**

TP = Cell\_22 = 19

FP = Cel1\_12 + Cel1\_32 = 1

TN = Cel1\_11 + Cel1\_13 + Cel1\_31 + Cel1\_33 = 33

FN = Cel1\_21 + Cel1\_23= 3

= = 0.92857 = 92.857%

=

= = 0.8636 = 86.36%

F1\_Score = = = 0.9047 = 90.47%

**Bird:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | Predicted Values | | |
| Cat | Dog | Bird |
| Actual Values | Cat | 2 | 1 | 1 |
| Dog | 1 | 19 | 2 |
| Bird | 2 | 0 | 28 |

TP = Cell\_33 = 28

FP = Cel1\_13 + Cel1\_23 = 3

TN = Cel1\_11 + Cel1\_12 + Cel1\_21 + Cel1\_22 = 23

FN = Cel1\_31 + Cel1\_32= 2

= = 0.9107 = 91.07%

=

= = 0.9333 = 93.33%

F1\_Score = = = 0.9180 = 91.80%

1. Design KNN classifier based on the following small dataset

- Preprocess the dataset first before any processing by substituting *M* & *F* with *0* and *1* respectively

- Randomly separate the dataset to training set (*70%*) and validation set (*30%*) by sample’s **ID** generated either from Python program or Excel unduplicated random function

- Calculate error rate for validation set from *K=1* to *K=7* either in Python program or Excel

- Select an appropriate *K*’s value and predict what class the new data in **red** color belongs to

- Finally write Python program by calling functions from ***scikit-learn*** to verify your design based on hand calculation results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Age | Gender | Fan |
| 0 | Bill | 32 | M | Rolling Stones |
| 1 | Henry | 40 | M | Neither |
| 2 | Mary | 16 | F | Taylor Swift |
| 3 | Tiffany | 14 | F | Taylor Swift |
| 4 | Michael | 55 | M | Neither |
| 5 | Carlos | 40 | M | Taylor Swift |
| 6 | Ashely | 20 | F | Neither |
| 7 | Robert | 15 | M | Taylor Swift |
| 8 | Sally | 55 | F | Rolling Stones |
| 9 | John | 15 | M | Rolling Stones |
| 10 | Michelle | 10 | F | ? |

**Solution:**

**Code:**

|  |
| --- |
| # import Libraries  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  #load data  df = pd.read\_csv("/content/q2.csv")  print("Origional DataFrame\n")  print(df)  df = df.set\_index("ID")  # Preprocess the dataset first before any processing by substituting M & F with 0 and 1 respectively  gender = [  "Gender"]  for column in gender:      df[column] = df[column].replace('M',0)      df[column] = df[column].replace('F',1)  print("\nDataFrame after substituting M & F with 0 and 1 respectively\n")  print(df)  # dive the data into 70% training and 30% testing sets  x = df.iloc[:-1,1:3].values         # Assign 1st and 2nd  colums values to X  y = df.iloc[:-1, 3].values           # Assign 3rd column values to y  from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.30)   # training set: 70%  # feature scaling  from sklearn.preprocessing import StandardScaler  scaler = StandardScaler()  scaler.fit(X\_train)  X\_train = scaler.transform(X\_train)  X\_test = scaler.transform(X\_test)  # Calculating error for K values between 1 and 7  error = []  for i in range(1, 8):      knn = KNeighborsClassifier(n\_neighbors=i)   # K = 1 to 7      knn.fit(X\_train, y\_train)      pred\_i = knn.predict(X\_test)      error.append(np.mean(pred\_i != y\_test))  plt.figure(figsize=(12, 6))  plt.plot(range(1, 8), error, color='red', linestyle='dashed', marker='o',           markerfacecolor='blue', markersize=10)  plt.title('Error Rate K Value')  plt.xlabel('K Value')  plt.ylabel('Mean Error')  # Training and Predictions based on the best K value  from sklearn.neighbors import KNeighborsClassifier  classifier = KNeighborsClassifier(n\_neighbors= 4)  classifier.fit(X\_train, y\_train)  y\_pred = classifier.predict(X\_test)  print(y\_pred) |

**Result**:

|  |
| --- |
|  |

1. K-Means algorithm is one of popular methods in unsupervised learning. Please plot elbow curve of total WCSS (within cluster sum of square) vs *K* from *1* to *5* either created by hand or Python program and select a proper *K*’s value based on your observation as the final number of clusters in your design. And then write Python program to verify your by-hand calculation results

|  |  |  |  |
| --- | --- | --- | --- |
| Objects | X | Y | Z |
| OB-1 | 1 | 4 | 1 |
| OB-2 | 1 | 2 | 2 |
| OB-3 | 1 | 4 | 2 |
| OB-4 | 2 | 1 | 2 |
| OB-5 | 1 | 1 | 1 |
| OB-6 | 2 | 4 | 2 |
| OB-7 | 1 | 1 | 2 |
| OB-8 | 2 | 1 | 1 |

**Solution:**

**Code:**

|  |
| --- |
| import numpy as np  import pandas as pd  from sklearn.cluster import KMeans  from sklearn.preprocessing import MinMaxScaler  from matplotlib import pyplot as plt  #%matplotlib inline  df = pd.read\_csv("/content/q3.csv")  print(df)  # plot original data  fig = plt.figure()  ax = plt.axes(projection ='3d')  ax.scatter3D(df.X, df.Y, df.Z, 'green')  ax.set\_title('K-means')  plt.show()  # # apply feature scaling  # df = df.set\_index(df.iloc[:,0])  # scaler = MinMaxScaler()  # df['X'] = scaler.fit\_transform(df[['X']])  # df['Y'] = scaler.fit\_transform(df[['Z']])  # df['Z'] = scaler.fit\_transform(df[['Z']])  X= df.iloc[:,1:3]  #elbow method  wcss = []  for i in range(1,9):      k\_means = KMeans(n\_clusters=i,init='k-means++', random\_state=42)      k\_means.fit(X)      wcss.append(k\_means.inertia\_)  #plot elbow curve  plt.plot(np.arange(1,9),wcss)  plt.xlabel('Clusters')  plt.ylabel('SSE')  plt.show()  # use k-means to fit and predict using the best k value from the elbow method  km = KMeans(n\_clusters=2)  kmeans = KMeans(n\_clusters=2, init='k-means++', max\_iter=300, n\_init=10, random\_state=0)  pred\_y = kmeans.fit\_predict([df.X,df.Y,df.Z])  fig = plt.figure()  ax = plt.axes(projection ='3d')  ax.scatter3D(df.X, df.Y, df.Z)  ax.scatter3D(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1],kmeans.cluster\_centers\_[:, 2], s=300, c='red')  ax.set\_title('K-means')  plt.show() |

**Result:**

|  |
| --- |
|  |