E21CSEU0962

Lab 3 AI

EB08

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split


# a) Read the dataset and print the different statistical values and shape of data.
iris_data = pd.read_csv('Iris Dataset.csv')
statistical_values = iris_data.describe()
shape_of_data = iris_data.shape


# b) Separate the features into X (inputs) and Y (output) and print the shape.
X = iris_data.drop(columns=['Species'])
Y = iris_data['Species']
shape_of_X = X.shape
shape_of_Y = Y.shape

# c) Apply Label Encoding on Y (Species column) to convert categorical values into numerical values.
label_encoder = LabelEncoder()
Y_encoded = label_encoder.fit_transform(Y)


# d) Split the dataset into training and testing set in different ratio, such as 60-40, 50-50, 70-30, 80-20, 55-45, 55-25 et
ratios = [(0.6, 0.4), (0.5, 0.5), (0.7, 0.3), (0.8, 0.2), (0.55, 0.45),
          (0.55, 0.25)]
splits = {}
for train_ratio, test_ratio in ratios:
  X_train, X_test, Y_train, Y_test = train_test_split(X,
                                                      Y_encoded,
                                                      test_size=test_ratio,
                                                      random_state=42)
  splits[f"{int(train_ratio*100)}-{int(test_ratio*100)}"] = {
      "X_train_shape": X_train.shape,
      "X_test_shape": X_test.shape,
      "Y_train_shape": Y_train.shape,
      "Y_test_shape": Y_test.shape
  }


# e) Shuffle training samples with different random seed values in the train_test_split function.
seed_values = [0, 21, 42, 63, 84]
shuffled_splits = {}
for seed in seed_values:
  X_train, X_test, Y_train, Y_test = train_test_split(
      X, Y_encoded, test_size=0.2,
      random_state=seed)  # using 80-20 as an example
  shuffled_splits[seed] = {
      "X_train_shape": X_train.shape,
      "X_test_shape": X_test.shape,
      "Y_train_shape": Y_train.shape,
      "Y_test_shape": Y_test.shape
  }

statistical_values, shape_of_data, shape_of_X, shape_of_Y, splits, shuffled_splits
```

```
(                Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
 count  150.000000     150.000000    150.000000     150.000000    150.000000
 mean    75.500000       5.843333      3.054000       3.758667      1.198667
 std     43.445368       0.828066      0.433594       1.764420      0.763161
 min      1.000000       4.300000      2.000000       1.000000      0.100000
 25%     38.250000       5.100000      2.800000       1.600000      0.300000
 50%     75.500000       5.800000      3.000000       4.350000      1.300000
 75%    112.750000       6.400000      3.300000       5.100000      1.800000
 max    150.000000       7.900000      4.400000       6.900000      2.500000,
 (150, 6),
 (150, 5),
 (150,),
 {'60-40': {'X_train_shape': (90, 5),
   'X_test_shape': (60, 5),
   'Y_train_shape': (90,),
   'Y_test_shape': (60,)},
  '50-50': {'X_train_shape': (75, 5),
   'X_test_shape': (75, 5),
   'Y_train_shape': (75,),
   'Y_test_shape': (75,)},
  '70-30': {'X_train_shape': (105, 5),
   'X_test_shape': (45, 5),
   'Y_train_shape': (105,),
   'Y_test_shape': (45,)},
  '80-20': {'X_train_shape': (120, 5),
   'X_test_shape': (30, 5),
   'Y_train_shape': (120,),
   'Y_test_shape': (30,)},
  '55-45': {'X_train_shape': (82, 5),
   'X_test_shape': (68, 5),
   'Y_train_shape': (82,),
   'Y_test_shape': (68,)},
  '55-25': {'X_train_shape': (112, 5),
   'X_test_shape': (38, 5),
   'Y_train_shape': (112,),
   'Y_test_shape': (38,)}},
 {0: {'X_train_shape': (120, 5),
   'X_test_shape': (30, 5),
   'Y_train_shape': (120,),
   'Y_test_shape': (30,)},
  21: {'X_train_shape': (120, 5),
   'X_test_shape': (30, 5),
   'Y_train_shape': (120,),
   'Y_test_shape': (30,)},
  42: {'X_train_shape': (120, 5),
   'X_test_shape': (30, 5),
   'Y_train_shape': (120,),
   'Y_test_shape': (30,)},
  63: {'X_train_shape': (120, 5),
   'X_test_shape': (30, 5),
   'Y_train_shape': (120,),
   'Y_test_shape': (30,)},
  84: {'X_train_shape': (120, 5),
   'X_test_shape': (30, 5),
   'Y_train_shape': (120,),
   'Y_test_shape': (30,)}})
```

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report

# a) Read the dataset into the data frame ‘df’ and print the different statistical values and shape of data.
df = pd.read_csv('Wine Dataset.csv')
statistical_values_df = df.describe()
shape_df = df.shape

# b) Separate the features into X and Y and print the shape.
# Assuming the target column is the last one
X = df.iloc[:, :-1]
Y = df.iloc[:, -1]
shape_of_X_df = X.shape
shape_of_Y_df = Y.shape

# c) Train the Multinomial Naïve Bayes model and do the classification on testing dataset.
clf = MultinomialNB()
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)

# d) Evaluate the performance using Accuracy, Precision, Recall, F-Score etc.
accuracy = accuracy_score(Y_test, Y_pred)
precision = precision_score(Y_test, Y_pred, average='weighted')
recall = recall_score(Y_test, Y_pred, average='weighted')
f_score = f1_score(Y_test, Y_pred, average='weighted')
classification_rep = classification_report(Y_test, Y_pred)

statistical_values_df, shape_df, shape_of_X_df, shape_of_Y_df, accuracy, precision, recall, f_score, classification_rep
```

```
 (          Alcohol  Malic.acid         Ash         Acl          Mg     Phenols  \
  count  178.000000  178.000000  178.000000  178.000000  178.000000  178.000000
  mean    13.000618    2.336348    2.366517   19.494944   99.741573    2.295112
  std      0.811827    1.117146    0.274344    3.339564   14.282484    0.625851
  min     11.030000    0.740000    1.360000   10.600000   70.000000    0.980000
  25%     12.362500    1.602500    2.210000   17.200000   88.000000    1.742500
  50%     13.050000    1.865000    2.360000   19.500000   98.000000    2.355000
  75%     13.677500    3.082500    2.557500   21.500000  107.000000    2.800000
  max     14.830000    5.800000    3.230000   30.000000  162.000000    3.880000

         Flavanoids  Nonflavanoid.phenols     Proanth   Color.int         Hue  \
  count  178.000000            178.000000  178.000000  178.000000  178.000000
  mean     2.029270              0.361854    1.590899    5.058090    0.957449
  std      0.998859              0.124453    0.572359    2.318286    0.228572
  min      0.340000              0.130000    0.410000    1.280000    0.480000
  25%      1.205000              0.270000    1.250000    3.220000    0.782500
  50%      2.135000              0.340000    1.555000    4.690000    0.965000
  75%      2.875000              0.437500    1.950000    6.200000    1.120000
  max      5.080000              0.660000    3.580000   13.000000    1.710000

                 OD     Proline        Wine
  count  178.000000  178.000000  178.000000
  mean     2.611685  746.893258    1.938202
  std      0.709990  314.907474    0.775035
  min      1.270000  278.000000    1.000000
  25%      1.937500  500.500000    1.000000
  50%      2.780000  673.500000    2.000000
  75%      3.170000  985.000000    3.000000
  max      4.000000 1680.000000    3.000000  ,
  (178, 14),
  (178, 13),
  (178,),
  0.8888888888888888,
  0.8865740740740741,
  0.8888888888888888,
  0.8828042328042328,
  '              precision    recall  f1-score   support\n\n           1       0.88      1.00      0.93        14\n           2
 0.93      0.93      0.93        14\n           3       0.83      0.62      0.71         8\n\n    accuracy
 0.89        36\n   macro avg       0.88      0.85      0.86        36\nweighted avg       0.89      0.89      0.88
 36\n')
```

✓ 0s completed at 11:39