# Introduction to Data-Science

# Assignment 3

**Name**: Muhammad Tayab

**Roll No**: FA21-BSE-030

**Submitted to**: Dr. Sharjeel

**Date:** 24-November-2023

**COMSAT'S University Islamabad, Lahore Campus**

```
"""
         ________________________________________
# 24 Nov 2023
# CSC461 – Assignment3 – Machine Learning
# Muhammad Tayab
# FA21-BSE-030
          __________________________________________
"""
```

```
"""
 Q1: Provide responses to the following questions about the dataset.
  1. How many instances does the dataset contain?
  2. How many input attributes does the dataset contain?
  3. How many possible values does the output attribute have?
  4. How many input attributes are categorical?
  5. What is the class ratio (male vs female) in the dataset?
"""
```

```
import pandas as pd
```

```
#importing and mounting Google Drive

from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
#loading the dataset

df = pd.read_csv('/content/drive/My Drive/IDS/Assignments/gender-prediction.csv')
```

```
# 1: Number of instances
num_instances = len(df)
print("  ------------------------------> OUTPUT <-----------------------------")
print(f"1. Number of instances: {num_instances}")
print("  <------------------------------------------------------------------->")
```

```
      ------------------------------> OUTPUT <-----------------------------
    1. Number of instances: 110
      <------------------------------------------------------------------->
```

```
# 2: Number of input attributes
num_input_attributes = len(df.columns)
print("  ------------------------------> OUTPUT <-----------------------------")
print(f"2. Number of input attributes: {num_input_attributes}")
print("  <------------------------------------------------------------------->")
```

```
      ------------------------------> OUTPUT <-----------------------------
    2. Number of input attributes: 8
      <------------------------------------------------------------------->
```

```
# 3: Number of possible values for the output attribute
num_output_values = df['gender'].nunique()
print("  ------------------------------> OUTPUT <-----------------------------  ")
print(f"3. Number of possible values for the output attribute: {num_output_values}")
print("  <------------------------------------------------------------------->")
```

```
      ------------------------------> OUTPUT <-----------------------------
    3. Number of possible values for the output attribute: 2
      <------------------------------------------------------------------->
```

```
# 4: Number of categorical input attributes
categorical_attributes = df.select_dtypes(include=['object']).columns
num_categorical_attributes = len(categorical_attributes)
print("  ------------------------------> OUTPUT <-----------------------------")
print(f"4. Number of categorical input attributes: {num_categorical_attributes}")
print("  <------------------------------------------------------------------->")
```

```
      ------------------------------> OUTPUT <-----------------------------
    4. Number of categorical input attributes: 5
```

```
      <-------------------------------------------------------------------->
```

```
# 5: Class ratio (male vs female)
class_ratio = df['gender'].value_counts(normalize=True)
print("  ------------------------------> OUTPUT <-----------------------------")
print(f"5. Class ratio (male vs female):\n{class_ratio}")
print("  <-------------------------------------------------------------------->")
```

```
      ------------------------------> OUTPUT <-----------------------------
    5. Class ratio (male vs female):
    male      0.563636
    female    0.436364
    Name: gender, dtype: float64
      <-------------------------------------------------------------------->
```

```
"""
Q2: Apply Logistic Regression, Support Vector Machines, and Multilayer Perceptron classification
algorithms (using Python) on the gender prediction dataset with 2/3 train and 1/3 test split ratio and answer
the following questions.
  1. How many instances are incorrectly classified?
  2. Rerun the experiment using train/test split ratio of 80/20. Do you see any change in the results?
     Explain.
  3. Name 2 attributes that you believe are the most “powerful” in the prediction task. Explain why?
  4. Try to exclude these 2 attribute(s) from the dataset. Rerun the experiment (using 80/20 train/test split),
     did you find any change in the results? Explain.

"""
```

```
      # 1. How many instances are incorrectly classified?
```

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


# Spliting data into train and test sets
X = df.drop("gender", axis=1)
y = df["gender"]



# Identify categorical columns
categorical_columns = X.select_dtypes(include=['object']).columns


# Creating column transformer for one-hot encoding
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), X.select_dtypes(include=['float64', 'int64']).columns),
        ('cat', OneHotEncoder(), categorical_columns)
    ]
)



# Creating pipeline with the preprocessor and Regression model
lr_model = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])


# 2/3 train and 1/3 test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=42)
```

```
# Making predictions
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)



# Calculating incorrect instances
incorrect_instances_lr = (y_test != y_pred_lr).sum()
print("  ------------------------------> OUTPUT <----------------------------")
print(f"1. Incorrect instances (Logistic Regression): {incorrect_instances_lr}")
print("  <------------------------------------------------------------------>")
```

```
      ------------------------------> OUTPUT <----------------------------
    1. Incorrect instances (Logistic Regression): 0
      <------------------------------------------------------------------>
```

```
  """
  2. Rerun the experiment using train/test split ratio of 80/20. Do you see any change in the results?
     Explain.
  """


from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X = df.drop("gender", axis=1)
y = df["gender"]
# Identify categorical columns
categorical_columns = X.select_dtypes(include=['object']).columns

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), X.select_dtypes(include=['float64', 'int64']).columns),
        ('cat', OneHotEncoder(), categorical_columns)
    ]
)

lr_model = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])
# 80/20 train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Making predictions
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
# Calculating incorrect instances
incorrect_instances_lr = (y_test != y_pred_lr).sum()
print("  ------------------------------> OUTPUT <----------------------------")
print(f"2. Re-runing the Incorrect instances (Logistic Regression): {incorrect_instances_lr}")
print("  <------------------------------------------------------------------>")
```

```
      ------------------------------> OUTPUT <----------------------------
    2. Re-runing the Incorrect instances (Logistic Regression): 0
      <------------------------------------------------------------------>
```

```
"""
    3. Name 2 attributes that you believe are the most “powerful” in the prediction task. Explain why?
"""
```

```
# Assuming 'gender' as target variable
X = df.drop('gender', axis=1)
y = df['gender']

# Fiting regression model
lr_model.fit(X, y)

# Geting absolute values
coefficients = abs(lr_model.named_steps['classifier'].coef_[0])

# Identifying indices
top2_indices = coefficients.argsort()[-2:][::-1]

# Geting names
top2_features = X.columns[top2_indices]
print("  ------------------------------> OUTPUT <-----------------------------")
print(f"The 2 powerful attributes: {top2_features}")
print("  <-------------------------------------------------------------------->")
```

```
  ------------------------------> OUTPUT <-----------------------------
The 2 powerful attributes: Index(['beard', 'shoe_size'], dtype='object')
  <-------------------------------------------------------------------->
```

```
"""
    4. Try to exclude these 2 attribute(s) from the dataset. Rerun the experiment (using 80/20 train/test split),
    did you find any change in the results? Explain.
"""


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

X = df.drop('gender', axis=1)
y = df['gender']

# The 2 powerful attributes
top2_attributes = ['beard', 'shoe_size']

# Excluding the 2 powerful attributes
X_excluded = X.drop(columns=top2_attributes)

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), X_excluded.select_dtypes(include=['float64', 'int64']).columns),
        ('cat', OneHotEncoder(), X_excluded.select_dtypes(include=['object']).columns)
    ]
)

lr_model_excluded = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])

# 80/20 train/test split
X_train_excluded, X_test_excluded, y_train, y_test = train_test_split(X_excluded, y, test_size=0.2, random_state=42)

# making predictions
lr_model_excluded.fit(X_train_excluded, y_train)
y_pred_lr_excluded = lr_model_excluded.predict(X_test_excluded)

# Calculating incorrect instances
incorrect_instances_lr_excluded = (y_test != y_pred_lr_excluded).sum()
print("  ------------------------------> OUTPUT <-----------------------------")
print(f"Incorrect instances after excluding the 2 powerful attributes: {incorrect_instances_lr_excluded}")
print("  <-------------------------------------------------------------------->")
```

```
  ------------------------------> OUTPUT <-----------------------------
Incorrect instances after excluding the 2 powerful attributes: 2
  <-------------------------------------------------------------------->
```

```
"""
    Q3: Apply Random Forest classification algorithm (using Python) on the gender prediction dataset with Monte
    Carlo cross-validation and Leave P-Out cross-validation. Report F1 scores for both cross-validation strategies.
"""


import pandas as pd
from sklearn.model_selection import cross_val_score, KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import make_scorer, f1_score
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler


# Loading dataset
df = pd.read_csv('/content/drive/My Drive/IDS/Assignments/gender-prediction.csv')

X = df.drop('gender', axis=1)
y = df['gender']

# Identify categorical columns
categorical_columns = X.select_dtypes(include=['object']).columns

# Creating a column transformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), X.select_dtypes(include=['float64', 'int64']).columns),
        ('cat', OneHotEncoder(), categorical_columns)
    ],
    remainder='passthrough'
)

X_encoded = preprocessor.fit_transform(X)

# Creating Random Forest classifier
rf_classifier = RandomForestClassifier(random_state=42)

f1_scorer = make_scorer(f1_score, average='weighted')

# Monte Carlo cross-validation
monte_carlo_f1_scores = cross_val_score(rf_classifier, X_encoded, y, cv=5, scoring=f1_scorer)

# P-Out cross-validation
leave_p_out = KFold(n_splits=len(X))
leave_p_out_f1_scores = cross_val_score(rf_classifier, X_encoded, y, cv=leave_p_out, scoring=f1_scorer)
print("  -----------------------------> OUTPUT <----------------------------")
print(f"Monte Carlo F1 Scores: {monte_carlo_f1_scores}")
print(f"Average Monte Carlo F1 Score: {monte_carlo_f1_scores.mean():.2f}")

print(f"Leave P-Out F1 Scores: {leave_p_out_f1_scores}")
print(f"Average Leave P-Out F1 Score: {leave_p_out_f1_scores.mean():.2f}")
print("  <------------------------------------------------------------------>")
```

```
  -----------------------------> OUTPUT <----------------------------
Monte Carlo F1 Scores: [1.         1.         1.         0.95425837 1.        ]
Average Monte Carlo F1 Score: 0.99
Leave P-Out F1 Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Average Leave P-Out F1 Score: 0.98
  <------------------------------------------------------------------>
```

```
"""
    Q4: Add 10 sample instances into the dataset (you can ask your friends/relatives/sibling for the data). Run the
    ML experiment (using Python) by training the model using Gaussian Naïve Bayes classification algorithm
    and all the instances from the gender prediction dataset. Evaluate the trained model using the newly added 10
    test instances. Report accuracy, precision, and recall scores.
"""


import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import LabelEncoder

# Loading dataset
df = pd.read_csv('/content/drive/My Drive/IDS/Assignments/gender-prediction.csv')

# Adding 10 sample instances
sample_data = {
    'height': [175, 165, 180, 160, 170, 175, 160, 185, 175, 170],
    'weight': [70, 60, 80, 55, 65, 70, 50, 90, 75, 68],
    'beard': ['yes', 'no', 'yes', 'no', 'yes', 'yes', 'no', 'yes', 'yes', 'no'],
    'hair_length': ['short', 'long', 'short', 'long', 'medium', 'medium', 'short', 'long', 'medium', 'short'],
    'shoe_size': [42, 39, 45, 37, 40, 42, 36, 46, 43, 41],
    'scarf': ['no', 'yes', 'no', 'yes', 'no', 'yes', 'no', 'yes', 'no', 'yes'],
    'eye_color': ['brown', 'blue', 'green', 'brown', 'blue', 'brown', 'green', 'blue', 'brown', 'green'],
    'gender': ['male', 'female', 'male', 'female', 'male', 'male', 'female', 'male', 'male', 'female']
}

sample_df = pd.DataFrame(sample_data)
df_extended = pd.concat([df, sample_df], ignore_index=True)

# Separating features and target variable
X_extended = df_extended.drop('gender', axis=1)
y_extended = df_extended['gender']

# Applying label encoding to categorical variables
label_encoder = LabelEncoder()
X_extended_encoded = X_extended.apply(label_encoder.fit_transform)
y_extended_encoded = label_encoder.fit_transform(y_extended)

# Splitting extended dataset into training and testing sets
X_train_extended, X_test_extended, y_train_extended, y_test_extended = train_test_split(
    X_extended_encoded, y_extended_encoded, test_size=0.2, random_state=42
)

# Train Gaussian Naïve Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train_extended, y_train_extended)

# Making predictions on test set
y_pred_extended = gnb.predict(X_test_extended)

# Evaluating model
accuracy = accuracy_score(y_test_extended, y_pred_extended)
precision = precision_score(y_test_extended, y_pred_extended, average='weighted')
recall = recall_score(y_test_extended, y_pred_extended, average='weighted')

# Printing results
print("  ------------------------------> OUTPUT <------------------------------")
print(f"Accuracy: {accuracy:.1f}")
print(f"Precision: {precision:.1f}")
print(f"Recall: {recall:.1f}")
print("  <-------------------------------------------------------------------->")
```

```
  ------------------------------> OUTPUT <------------------------------
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
  <-------------------------------------------------------------------->
```