Lab 6 Solution

April 29, 2024

1 LAB 06 - Python version

Luca Catalano, Daniele Rege Cambrin, Eleonora Poeta

1.0.1 Disclaimer

The purpose of creating this material is to enhance the knowledge of students who are interested in learning how to solve problems presented in laboratory classes using Python. This decision stems from the observation that some students have opted to utilize Python for tackling exam projects in recent years.

To solve these exercises using Python, you need to install Python (version 3.9.6 or later) and some libraries using pip or conda.

Here's a list of the libraries needed for this case:

- `os`: Provides operating system dependent functionality, commonly used for file operations such as reading and writing files, interacting with the filesystem, etc.
- `pandas`: A data manipulation and analysis library that offers data structures and functions to efficiently work with structured data.
- `numpy`: A numerical computing library that provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.
- `matplotlib.pyplot`: A plotting library for creating visualizations like charts, graphs, histograms, etc.
- `sklearn`: Machine learning algorithms and tools.
- `xlrd`: A Python library used for reading data and formatting information from Excel files (.xls and .xlsx formats). It provides functionality to extract data from Excel worksheets, including cells, rows, columns, and formatting details.

You can download Python from here and follow the installation instructions for your operating system.

For installing libraries using pip or conda, you can use the following commands:

- Using pip:

  ```
  pip install pandas numpy matplotlib scikit-learn xlrd
  ```

- Using conda:

  ```
  conda install pandas numpy matplotlib scikit-learn xlrd
  ```

Make sure to run these commands in your terminal or command prompt after installing Python. You can also execute them in a cell of a Jupyter Notebook file (`.ipynb`) by starting the command with '!'.

2 Exercise 1

Import some libraries

[1]:
```python
import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import cross_val_predict, GridSearchCV
from sklearn.metrics import confusion_matrix
```

2.1 Read file excel "user.xlsx"

To read the Excel file using a function integrated into the pandas library, you can use the `pd.read_excel()` function. Rewrite the instruction with the argument as the path of the file to be read

[2]:
```python
# Read file excel
dataset = pd.read_excel("/Users/luca/Library/Mobile Documents/
 ↪com~apple~CloudDocs/Business Intelligence per Big Data/Laboratories/LAB06/
 ↪Lab6Materiale/user.xlsx")
```

```
/Users/luca/Library/Python/3.9/lib/python/site-
packages/openpyxl/styles/stylesheet.py:226: UserWarning: Workbook contains no
default style, apply openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
```

In a Jupyter Notebook cell, you can print a subset of the representation by simply calling the name of the variable containing the DataFrame.

[3]:
```python
# print dataset
dataset
```

[3]:
```
    Age         Workclass  Education      Marital Status  \
0  39.0         State-gov  Bachelors       Never-married
1  50.0  Self-emp-not-inc  Bachelors  Married-civ-spouse
2  38.0           Private    HS-grad            Divorced
3  53.0           Private       11th  Married-civ-spouse
4  28.0           Private  Bachelors  Married-civ-spouse
```

```
..   …                …           …                   …
995  56.0          Private      HS-grad  Married-civ-spouse
996  45.0          Private      Masters            Divorced
997  48.0      Federal-gov    Bachelors            Divorced
998  40.0          Private  Some-college  Married-civ-spouse
999  39.0     Self-emp-inc    Bachelors  Married-civ-spouse

            Occupation   Relationship   Race     Sex Native Country   Response
0         Adm-clerical  Not-in-family  White    Male   United-States  Negative
1      Exec-managerial        Husband  White    Male   United-States  Negative
2    Handlers-cleaners  Not-in-family  White    Male   United-States  Negative
3    Handlers-cleaners        Husband  Black    Male   United-States  Negative
4       Prof-specialty           Wife  Black  Female            Cuba  Negative
..                 …              …      …      …               …        …
995    Exec-managerial        Husband  White    Male   United-States  Positive
996     Prof-specialty  Not-in-family  White    Male   United-States  Negative
997    Exec-managerial      Unmarried  White    Male   United-States  Positive
998  Machine-op-inspct        Husband  White    Male   United-States  Negative
999    Exec-managerial        Husband  White    Male   United-States  Positive

[1000 rows x 10 columns]
```

2.2 Define the label column in the dataset data frame

Rename the 'Response' column to 'Label' [use dataset.rename(columns={'actual_col_name': 'new_col_name'})]

[4]:
```python
# rename column Response to Label
dataset = dataset.rename(columns={'Response': 'Label'})
```

[5]:
```python
# print datsaset to check if the column has been renamed
dataset
```

[5]:
```
      Age         Workclass    Education      Marital Status  \
0    39.0         State-gov    Bachelors       Never-married
1    50.0  Self-emp-not-inc    Bachelors  Married-civ-spouse
2    38.0           Private      HS-grad            Divorced
3    53.0           Private         11th  Married-civ-spouse
4    28.0           Private    Bachelors  Married-civ-spouse
..   …                …           …                   …
995  56.0          Private      HS-grad  Married-civ-spouse
996  45.0          Private      Masters            Divorced
997  48.0      Federal-gov    Bachelors            Divorced
998  40.0          Private  Some-college  Married-civ-spouse
999  39.0     Self-emp-inc    Bachelors  Married-civ-spouse

            Occupation   Relationship   Race     Sex Native Country     Label
```

```
0          Adm-clerical  Not-in-family  White    Male  United-States  Negative
1       Exec-managerial        Husband  White    Male  United-States  Negative
2     Handlers-cleaners  Not-in-family  White    Male  United-States  Negative
3     Handlers-cleaners        Husband  Black    Male  United-States  Negative
4        Prof-specialty           Wife  Black  Female           Cuba  Negative
..                  …              …      …      …              …        …
995     Exec-managerial        Husband  White    Male  United-States  Positive
996      Prof-specialty  Not-in-family  White    Male  United-States  Negative
997     Exec-managerial      Unmarried  White    Male  United-States  Positive
998   Machine-op-inspct        Husband  White    Male  United-States  Negative
999     Exec-managerial        Husband  White    Male  United-States  Positive

[1000 rows x 10 columns]
```

2.3 Separate the dataset into features, referred to as X, and labels, referred to as y. Afterwards, utilize Label Encoder to encode the categorical features.

[You can achieve this by selecting columns using the [] operator on the dataframe, then initializing the Label Encoder and applying its fit_transform method]

[6]:
```python
# Split the dataset into features (X) and target variable (y)
X = dataset.drop(columns=['Label'])  # Features
y = dataset['Label']  # Target variable


# Label encoding
labelencoder = LabelEncoder()
# Apply label encoding to each column, except for the age column
for column in X.columns:
    if column != 'Age':
        X[column] = labelencoder.fit_transform(X[column])

# Transform Negative into 0value and Positive into 1 value (use label encoder␣
 ↪with .fit_transform)
y = labelencoder.fit_transform(y)
```

2.4 Use the random forest classifier model.

To start, split the dataset `users.xlsx` into two parts: training and testing. This allows for training the model on the training portion and evaluating its performance using the test portion.

Please note that the test portion is not a real-case test dataset but rather an archetype for evaluating the model with a small dataset that contains the correct labels.

Set these parameters:

- Max Depth: 100
- Number of trees: 20

[use train__test__split() to split the dataset]

[Use RandomForestClassifier() and its .fit and .predict function]

[7]:
```
# Split the dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Create a Random Forest Classifier
random_forest = RandomForestClassifier(n_estimators=20, max_depth=3,␣
 ↪random_state=42)

# Train the model using the training sets
random_forest.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = random_forest.predict(X_test)

# Evaluate the model: Accuracy, Precision, Recall
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

# Print the evaluation metrics
print("Accuracy: ", accuracy)
print("Precision: ", precision)
print("Recall: ", recall)
```

```
Accuracy:  0.805
Precision:  0.8
Recall:  0.25
```

2.5 Validation of Random Forest Classifier model using Cross Validation

Cross-validation is a technique used to assess the performance and generalization ability of machine learning models, particularly in the context of classification tasks. It involves partitioning the dataset into multiple subsets, known as folds.

1. **Partitioning the Dataset**: The dataset is divided into k equal-sized folds.
2. **Training and Testing**: The model is trained k times, each time using k-1 folds for training and the remaining fold for testing.
3. **Evaluation**: The performance of the model is evaluated on each fold, and the results are averaged to obtain a robust estimate of the model's performance.
4. **Advantages**: Cross-validation provides a more reliable estimate of the model's performance compared to a single train-test split. It helps to detect overfitting and assesses the model's ability to generalize to unseen data.

[Use `cross_val_score` and `cross_val_predict` to perform cross-validation easily]

[8]:
```
# Initialize the decision tree classifier
clf = RandomForestClassifier(n_estimators=200, max_depth=3, random_state=42)

# Perform cross-validation predictions
y_pred = cross_val_predict(clf, X, y, cv=5)

# Calculate confusion matrix
conf_matrix = confusion_matrix(y, y_pred)

# Evaluate accuracy
accuracy = accuracy_score(y, y_pred)
# Print accuracy
print("Accuracy:", accuracy)

# Print confusion matrix
conf_matrix = pd.DataFrame(conf_matrix, columns=['Predicted No', 'Predicted␣
↪Yes'], index=['Actual No', 'Actual Yes'])
conf_matrix
```

```
Accuracy: 0.775
```

[8]:

	Predicted No	Predicted Yes
Actual No	757	11
Actual Yes	214	18

2.6 Implement Grid Search

Grid Search is a technique used to find the optimal hyperparameters for a machine learning model. It works by searching through a predefined set of hyperparameters and evaluating the model's performance for each combination using cross-validation.

Specifically, you need to:

1. Define a grid of hyperparameters to search through.
2. Use Grid Search to find the best combination of hyperparameters.

[9]:
```
# Grid search. It takes more or less 30 seconds to run
# Define the parameter grid
param_grid =    {
                    "n_estimators": [100, 250, 500],
                    "max_depth": [None, 10, 20, 30],
                }
# Perform grid search
gs = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
# Initialize the grid search
gs.fit(X, y) # [use .fit() method]
# Print the best parameters and the best score
gs.best_params_, gs.best_score_
```

[9]: ({'max_depth': 10, 'n_estimators': 500}, 0.817)

3 Exercise 2

Import some libraries

[24]:
```
import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC as SupportVectorMachineClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import cross_val_predict, GridSearchCV
from sklearn.metrics import confusion_matrix
```

3.1 Read file excel "user.xlsx"

To read the Excel file using a function integrated into the pandas library, you can use the `pd.read_excel()` function. Rewrite the instruction with the argument as the path of the file to be read

[25]:
```
# Read file excel
dataset = pd.read_excel("/Users/luca/Library/Mobile Documents/
 ↪com~apple~CloudDocs/Business Intelligence per Big Data/Laboratories/LAB06/
 ↪Lab6Materiale/user.xlsx")
```

```
/Users/luca/Library/Python/3.9/lib/python/site-
packages/openpyxl/styles/stylesheet.py:226: UserWarning: Workbook contains no
default style, apply openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
```

In a Jupyter Notebook cell, you can print a subset of the representation by simply calling the name of the variable containing the DataFrame.

[26]:
```
# print dataset
dataset
```

[26]:
```
      Age         Workclass    Education      Marital Status  \
0    39.0         State-gov    Bachelors       Never-married
1    50.0  Self-emp-not-inc    Bachelors  Married-civ-spouse
2    38.0           Private      HS-grad            Divorced
3    53.0           Private         11th  Married-civ-spouse
4    28.0           Private    Bachelors  Married-civ-spouse
..    …                 …          …                   …
995  56.0           Private      HS-grad  Married-civ-spouse
```

```
996  45.0           Private       Masters            Divorced
997  48.0       Federal-gov     Bachelors            Divorced
998  40.0           Private  Some-college  Married-civ-spouse
999  39.0      Self-emp-inc     Bachelors  Married-civ-spouse

            Occupation   Relationship   Race     Sex Native Country  Response
0         Adm-clerical  Not-in-family  White    Male  United-States  Negative
1      Exec-managerial        Husband  White    Male  United-States  Negative
2    Handlers-cleaners  Not-in-family  White    Male  United-States  Negative
3    Handlers-cleaners        Husband  Black    Male  United-States  Negative
4       Prof-specialty           Wife  Black  Female           Cuba  Negative
..                 …              …    …     …              …        …
995    Exec-managerial        Husband  White    Male  United-States  Positive
996     Prof-specialty  Not-in-family  White    Male  United-States  Negative
997    Exec-managerial      Unmarried  White    Male  United-States  Positive
998  Machine-op-inspct        Husband  White    Male  United-States  Negative
999    Exec-managerial        Husband  White    Male  United-States  Positive

[1000 rows x 10 columns]
```

3.2 Define the label column in the dataset data frame

Rename the 'Response' column to 'Label' [use dataset.rename(columns={'actual_col_name': 'new_col_name'})]

[27]:
```
# rename column Response to Label
dataset = dataset.rename(columns={'Response': 'Label'})
```

[28]:
```
# print datsaset to check if the column has been renamed
dataset
```

[28]:
```
      Age          Workclass     Education      Marital Status  \
0    39.0          State-gov     Bachelors       Never-married
1    50.0   Self-emp-not-inc     Bachelors  Married-civ-spouse
2    38.0            Private       HS-grad            Divorced
3    53.0            Private          11th  Married-civ-spouse
4    28.0            Private     Bachelors  Married-civ-spouse
..    …                 …           …                 …
995  56.0            Private       HS-grad  Married-civ-spouse
996  45.0            Private       Masters            Divorced
997  48.0        Federal-gov     Bachelors            Divorced
998  40.0            Private  Some-college  Married-civ-spouse
999  39.0       Self-emp-inc     Bachelors  Married-civ-spouse

            Occupation   Relationship   Race     Sex Native Country     Label
0         Adm-clerical  Not-in-family  White    Male  United-States  Negative
1      Exec-managerial        Husband  White    Male  United-States  Negative
```

```
2    Handlers-cleaners  Not-in-family  White    Male  United-States  Negative
3    Handlers-cleaners        Husband  Black    Male  United-States  Negative
4       Prof-specialty           Wife  Black  Female           Cuba  Negative
..                 …              …      …      …              …         …
995    Exec-managerial        Husband  White    Male  United-States  Positive
996     Prof-specialty  Not-in-family  White    Male  United-States  Negative
997    Exec-managerial      Unmarried  White    Male  United-States  Positive
998  Machine-op-inspct        Husband  White    Male  United-States  Negative
999    Exec-managerial        Husband  White    Male  United-States  Positive

[1000 rows x 10 columns]
```

3.3 Separate the dataset into features, referred to as X, and labels, referred to as y. Afterwards, utilize Label Encoder to encode the categorical features.

[You can achieve this by selecting columns using the [] operator on the dataframe, then initializing the Label Encoder and applying its fit_transform method]

[29]:
```python
# Split the dataset into features (X) and target variable (y)
X = dataset.drop(columns=['Label'])  # Features
y = dataset['Label']  # Target variable



# Label encoding
labelencoder = LabelEncoder()
# Apply label encoding to each column, except for the age column
for column in X.columns:
    if column != 'Age':
        X[column] = labelencoder.fit_transform(X[column])

# Transform Negative into 0value and Positive into 1 value (use label encoder␣
↪with .fit_transform)
y = labelencoder.fit_transform(y)
```

3.4 Use the Support Vector Machine classifier model.

Use the same split of the dataset `users.xlsx` into two parts

Set these parameters:

- C: 100
- gamma: 0.1
- kernel='rbf'

[Use SVM() and its .fit and .predict function]

[30]:
```python
# Split the dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

```python
# Create a SVM Classifier
svm = SupportVectorMachineClassifier(kernel='rbf', C=100,  gamma=0.1)

# Train the model using the training sets
svm.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = svm.predict(X_test)

# Evaluate the model: Accuracy, Precision, Recall
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

# Print the evaluation metrics
print("Accuracy: ", accuracy)
print("Precision: ", precision)
print("Recall: ", recall)
```

```
Accuracy:  0.735
Precision:  0.4489795918367347
Recall:  0.4583333333333333
```

3.5 Validation of SVM Classifier model using Cross Validation

Cross-validation is a technique used to assess the performance and generalization ability of machine learning models, particularly in the context of classification tasks. It involves partitioning the dataset into multiple subsets, known as folds.

1. **Partitioning the Dataset**: The dataset is divided into k equal-sized folds.
2. **Training and Testing**: The model is trained k times, each time using k-1 folds for training and the remaining fold for testing.
3. **Evaluation**: The performance of the model is evaluated on each fold, and the results are averaged to obtain a robust estimate of the model's performance.
4. **Advantages**: Cross-validation provides a more reliable estimate of the model's performance compared to a single train-test split. It helps to detect overfitting and assesses the model's ability to generalize to unseen data.

[Use `cross_val_score` and `cross_val_predict` to perform cross-validation easily]

[31]:
```python
# Initialize the decision tree classifier
clf = SupportVectorMachineClassifier(kernel='rbf', C=100, gamma=0.1)

# Perform cross-validation predictions
y_pred = cross_val_predict(clf, X, y, cv=5)
```

```
# Calculate confusion matrix
conf_matrix = confusion_matrix(y, y_pred)

# Evaluate accuracy
accuracy = accuracy_score(y, y_pred)
# Print accuracy
print("Accuracy:", accuracy)

# Print confusion matrix
conf_matrix = pd.DataFrame(conf_matrix, columns=['Predicted No', 'Predicted␣
↪Yes'], index=['Actual No', 'Actual Yes'])
conf_matrix
```

```
Accuracy: 0.744
```

[31]:

	Predicted No	Predicted Yes
Actual No	663	105
Actual Yes	151	81

3.6 Implement Grid Search

Grid Search is a technique used to find the optimal hyperparameters for a machine learning model. It works by searching through a predefined set of hyperparameters and evaluating the model's performance for each combination using cross-validation.

Specifically, you need to:

1. Define a grid of hyperparameters to search through.
2. Use Grid Search to find the best combination of hyperparameters.

[32]:
```
# Grid search. It takes more or less 30 seconds to run
# Define the parameter grid
param_grid =    {
                    "C": [1, 2, 5, 10],
                    "gamma": [2, 1, 0.1, 0.01],
                    "kernel": ['rbf', 'linear']
                }
# Perform grid search
gs = GridSearchCV(SupportVectorMachineClassifier(), param_grid, cv=5)
# Initialize the grid search
gs.fit(X, y) # [use .fit() method]
# Print the best parameters and the best score
gs.best_params_, gs.best_score_
```

[32]:
```
({'C': 5, 'gamma': 1, 'kernel': 'rbf'}, 0.78)
```

4 Exercise 3

Import some libraries

[108]:
```python
import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn.neural_network import MLPClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import cross_val_predict, GridSearchCV
from sklearn.metrics import confusion_matrix
```

4.1 Read file excel "user.xlsx"

To read the Excel file using a function integrated into the pandas library, you can use the `pd.read_excel()` function. Rewrite the instruction with the argument as the path of the file to be read

[77]:
```python
# Read file excel
dataset = pd.read_excel("/Users/luca/Library/Mobile Documents/
 ↪com~apple~CloudDocs/Business Intelligence per Big Data/Laboratories/LAB06/
 ↪Lab6Materiale/user.xlsx")
```

```
/Users/luca/Library/Python/3.9/lib/python/site-
packages/openpyxl/styles/stylesheet.py:226: UserWarning: Workbook contains no
default style, apply openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
```

In a Jupyter Notebook cell, you can print a subset of the representation by simply calling the name of the variable containing the DataFrame.

[78]:
```python
# print dataset
dataset
```

[78]:
```
      Age        Workclass    Education      Marital Status  \
0    39.0        State-gov    Bachelors       Never-married
1    50.0  Self-emp-not-inc    Bachelors  Married-civ-spouse
2    38.0          Private      HS-grad            Divorced
3    53.0          Private         11th  Married-civ-spouse
4    28.0          Private    Bachelors  Married-civ-spouse
..    …               …            …                  …
995  56.0          Private      HS-grad  Married-civ-spouse
996  45.0          Private      Masters            Divorced
997  48.0      Federal-gov    Bachelors            Divorced
998  40.0          Private  Some-college  Married-civ-spouse
```

```
999  39.0      Self-emp-inc     Bachelors  Married-civ-spouse

            Occupation   Relationship   Race     Sex Native Country  Response
0         Adm-clerical  Not-in-family  White    Male  United-States  Negative
1      Exec-managerial        Husband  White    Male  United-States  Negative
2    Handlers-cleaners  Not-in-family  White    Male  United-States  Negative
3    Handlers-cleaners        Husband  Black    Male  United-States  Negative
4       Prof-specialty           Wife  Black  Female           Cuba  Negative
..                 …             …     …      …              …         …
995    Exec-managerial        Husband  White    Male  United-States  Positive
996     Prof-specialty  Not-in-family  White    Male  United-States  Negative
997    Exec-managerial      Unmarried  White    Male  United-States  Positive
998  Machine-op-inspct        Husband  White    Male  United-States  Negative
999    Exec-managerial        Husband  White    Male  United-States  Positive

[1000 rows x 10 columns]
```

4.2 Define the label column in the dataset data frame

Rename the 'Response' column to 'Label' [use dataset.rename(columns={'actual_col_name': 'new_col_name'})]

[79]:
```
# rename column Response to Label
dataset = dataset.rename(columns={'Response': 'Label'})
```

[80]:
```
# print datsaset to check if the column has been renamed
dataset
```

[80]:
```
      Age         Workclass     Education      Marital Status  \
0    39.0         State-gov     Bachelors       Never-married
1    50.0  Self-emp-not-inc     Bachelors  Married-civ-spouse
2    38.0           Private       HS-grad            Divorced
3    53.0           Private          11th  Married-civ-spouse
4    28.0           Private     Bachelors  Married-civ-spouse
..    …                 …           …                  …
995  56.0           Private       HS-grad  Married-civ-spouse
996  45.0           Private       Masters            Divorced
997  48.0       Federal-gov     Bachelors            Divorced
998  40.0           Private  Some-college  Married-civ-spouse
999  39.0      Self-emp-inc     Bachelors  Married-civ-spouse

            Occupation   Relationship   Race     Sex Native Country     Label
0         Adm-clerical  Not-in-family  White    Male  United-States  Negative
1      Exec-managerial        Husband  White    Male  United-States  Negative
2    Handlers-cleaners  Not-in-family  White    Male  United-States  Negative
3    Handlers-cleaners        Husband  Black    Male  United-States  Negative
4       Prof-specialty           Wife  Black  Female           Cuba  Negative
```

```
..                 ...          ...    ...      ...            ...        ...
995     Exec-managerial         Husband  White    Male  United-States  Positive
996      Prof-specialty  Not-in-family  White    Male  United-States  Negative
997     Exec-managerial       Unmarried  White    Male  United-States  Positive
998  Machine-op-inspct         Husband  White    Male  United-States  Negative
999     Exec-managerial         Husband  White    Male  United-States  Positive

[1000 rows x 10 columns]
```

4.3 Separate the dataset into features, referred to as X, and labels, referred to as y. Afterwards, utilize Label Encoder to encode the categorical features.

[You can achieve this by selecting columns using the [] operator on the dataframe, then initializing the Label Encoder and applying its fit_transform method]

[81]:
```python
# Split the dataset into features (X) and target variable (y)
X = dataset.drop(columns=['Label'])  # Features
y = dataset['Label']  # Target variable


# Label encoding
labelencoder = LabelEncoder()
# Apply label encoding to each column, except for the age column
for column in X.columns:
    if column != 'Age':
        X[column] = labelencoder.fit_transform(X[column])

# Transform Negative into 0value and Positive into 1 value (use label encoder␣
↪with .fit_transform)
y = labelencoder.fit_transform(y)
```

4.4 Use the MLP classifier model.

Use the same split of the dataset `users.xlsx` into two parts

A Multi-Layer Perceptron (MLP) is a type of artificial neural network (ANN) that consists of multiple layers of nodes, or neurons, arranged in a feedforward manner. MLPs are widely used for various machine learning tasks, including classification and regression.

4.4.1 Structure of an MLP:

1. **Input Layer**: The first layer of the MLP, which receives input features from the dataset.
2. **Hidden Layers**: Intermediate layers between the input and output layers. Each hidden layer consists of multiple neurons, and the number of hidden layers and neurons per layer can vary depending on the complexity of the task.
3. **Output Layer**: The final layer of the MLP, which produces the network's output. The number of neurons in the output layer depends on the number of classes in the classification

task or the number of output values in the regression task.

4.4.2 Activation Function:

Each neuron in the MLP applies an activation function to its input to introduce non-linearity into the model and enable the network to learn complex patterns. Common activation functions include:

- **ReLU (Rectified Linear Unit)**
- **Sigmoid**
- **Tanh (Hyperbolic Tangent)**

4.4.3 Training an MLP:

MLPs are trained using an optimization algorithm such as gradient descent to minimize a loss function, which measures the difference between the predicted output and the true labels in the training data. Common loss functions include cross-entropy loss for classification tasks and mean squared error for regression tasks.

Set these parameters:

- max_iter = 500
- solver='sgd'
- learning_rate_init=0.001
- hidden_layer_sizes=(512, 256, 128)
- random_state=42

[Use MLPClassifier() and its .fit and .predict function]

[107]:
```
# Split the dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
# Create a MLP Classifier
clf = MLPClassifier(max_iter=500, solver='sgd', learning_rate_init=0.01,␣
 ↪hidden_layer_sizes=(512, 256, 128), random_state=42)
# Train the model using the training sets
clf.fit(X_train, y_train)
# Predict the response for test dataset
clf.predict(X_test)
# Evaluate the model: Accuracy
clf.score(X_test, y_test)
```

[107]:
```
0.76
```