Assignment 1 Part 2

Name: Barun Parua

Roll Number: 21CS10014

First of all, we import the necessary libraries.

Then we read the data from the csv file and store it in a pandas dataframe. After that randomization is done along with some preprocessing.

Note that preprocessing is important as there are some missing values in the data. Now this gives issues with saga solvers. Hence we drop the rows with missing values.

Also, we drop the first column as it is just indicating the index of the row which is not a very useful feature.

After that we split the data into training and testing sets.

In [ ]:
```
# importing all the necessary libraries
# pandas for reading the dataset into a dataframe
# numpy for mathematical operations
# sklearn for preprocessing and machine learning algorithms and encoding

import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
```

In [ ]:
```
# extracting the dataset into a dataframe
dataset = pd.read_csv('../../dataset/cross-validation.csv')

# randomize the dataset
dataset = dataset.sample(frac=1).reset_index(drop=True)
print("First 5 rows of the dataset:")
dataset.head()
```

```
First 5 rows of the dataset:
```

Out[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncom
0	LP002244	Male	Yes	0	Graduate	No	233
1	LP001677	Male	No	2	Graduate	No	492
2	LP001871	Female	No	0	Graduate	No	720
3	LP001520	Male	Yes	0	Graduate	No	486
4	LP001552	Male	Yes	0	Graduate	No	458

In [ ]:
```
# some data preprocessing as we can see that there are some missing values in th
```

```
# remove the rows with missing values
dataset = dataset.dropna()

# drop the Loan_ID column
dataset = dataset.drop(columns=['Loan_ID'], axis=1)

# split the dataset into train and test
train = dataset[:int(0.8*len(dataset))]
test = dataset[int(0.8*len(dataset)):]

# split the train and test into X and Y
X_train = train.drop(columns=['Loan_Status'])
y_train = train['Loan_Status']
X_test = test.drop(columns=['Loan_Status'])
y_test = test['Loan_Status']

# print the first 5 rows of the train X to check if NaN values are removed
print("\nFirst 5 rows of the train X:")
X_train.head()
```

```
First 5 rows of the train X:
```

Out[ ]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic
0	Male	Yes	0	Graduate	No	2333	
1	Male	No	2	Graduate	No	4923	
2	Female	No	0	Graduate	No	7200	
3	Male	Yes	0	Graduate	No	4860	
4	Male	Yes	0	Graduate	No	4583	

Here we define a get_scores function which takes in the predicted values and the actual values and returns the accuracy, precision, recall using the formulae related to true positives, true negatives, false positives and false negatives.

There is also encoder and scaler functions defined which are used to encode the categorical features and scale the numerical features respectively.

In [ ]:

```
# get true positives, true negatives, false positives and false negatives
# use them to calculate accuracy, precision and recall of the model
# return the accuracy, precision and recall

def get_scores(y_test, y_pred):
    tp, tn, fp, fn = 0, 0, 0, 0
    for i in range(len(y_pred)):
        if y_pred[i] == 'Y' and y_test.iloc[i] == 'Y':
            tp += 1
        elif y_pred[i] == 'N' and y_test.iloc[i] == 'N':
            tn += 1
        elif y_pred[i] == 'Y' and y_test.iloc[i] == 'N':
            fp += 1
        elif y_pred[i] == 'N' and y_test.iloc[i] == 'Y':
            fn += 1

    # calculate accuracy, precision and recall
```

```
    accuracy = (tp + tn) / (tp + tn + fp + fn)
    precision = tp / (tp + fp)
    recall = tp / (tp + fn)

    return accuracy, precision, recall

# define the encoding and scaling functions
# encoder is the LabelEncoder object
encoder = LabelEncoder()
# encode function encodes the categorical data
def encode(data):
    for i in data.columns:
        if data[i].dtype == 'object':
            encoder.fit(data[i].astype(str))
            data[i] = encoder.transform(data[i].astype(str))
    return data

# scaler is the StandardScaler object
scaler = StandardScaler()
```

This part of the code has the implementation of the logistic regression model using saga solver for the whole training dataset.

Note that this part just uses the training data and tests on the testing data. This part is not used in the 5-fold cross validation.

In [ ]:
```
# encode the categorical features
X_train = encode(X_train)
X_test = encode(X_test)

# scale the data
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# create the model with saga solver
model = LogisticRegression(solver='saga', penalty=None, max_iter=10000)

# fit the model and predict the test data
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# get the accuracy, precision and recall of the model and print them
total_accuracy, total_precision, total_recall = get_scores(y_test, y_pred)
print("Total Accuracy Score: ", total_accuracy)
print("Total Precision Score: ", total_precision)
print("Total Recall Score: ", total_recall)
```

```
Total Accuracy Score:  0.78125
Total Precision Score:  0.788235294117647
Total Recall Score:  0.9571428571428572
```

This part has the 5-fold cross validation implementation. The training data is split into 5 parts and each part is used as the validation set once and the rest of the data is used as the training set.

In this way, we get 5 sets on which the model is trained and tested. The individual scores are stored in a list and the average of the scores is taken to get the final scores. The scores are printed for each fold and the average scores are printed at the end.

In [ ]:
```
# make the 5-fold cross validation
k = 5
size = len(dataset) // k
# lists to store the accuracy, precision and recall of each fold
accuracy_list, precision_list, recall_list = [], [], []

for i in range(k):
    # create 5 folds of the train data and make validation set
    val = dataset[i*size: (i+1)*size]
    train = dataset.drop(val.index)

    # split the train and test into X and Y
    X_train = train.drop(columns=['Loan_Status'])
    y_train = train['Loan_Status']
    X_test = val.drop(columns=['Loan_Status'])
    y_test = val['Loan_Status']

    # encode and scale the data
    X_train = encode(X_train)
    X_test = encode(X_test)
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # create the model with saga solver
    model = LogisticRegression(solver='saga', penalty=None, max_iter=100000)

    # fit the model and predict the test data
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # get the accuracy, precision and recall of the model and print them
    # also store them in the lists
    accuracy, precision, recall = get_scores(y_test, y_pred)
    accuracy_list.append(accuracy)
    precision_list.append(precision)
    recall_list.append(recall)
    print("Fold: ", i+1)
    print("Accuracy Score: ", accuracy)
    print("Precision Score: ", precision)
    print("Recall Score: ", recall)

# print the mean accuracy, precision and recall
print()
print("Mean Accuracy Score: ", np.mean(accuracy_list))
print("Mean Precision Score: ", np.mean(precision_list))
print("Mean Recall Score: ", np.mean(recall_list))
```

```
Fold:  1
Accuracy Score:  0.7916666666666666
Precision Score:  0.7605633802816901
Recall Score:  0.9473684210526315
Fold:  2
Accuracy Score:  0.8125
Precision Score:  0.7901234567901234
Recall Score:  0.9846153846153847
Fold:  3
Accuracy Score:  0.875
Precision Score:  0.8875
Recall Score:  0.9594594594594594
Fold:  4
Accuracy Score:  0.78125
Precision Score:  0.7586206896551724
Recall Score:  1.0
Fold:  5
Accuracy Score:  0.78125
Precision Score:  0.788235294117647
Recall Score:  0.9571428571428572

Mean Accuracy Score:  0.8083333333333332
Mean Precision Score:  0.7970085641689266
Mean Recall Score:  0.9697172244540665
```

End of code. Thank you.