Assignment 1

Name: Sourodeep Datta

Roll Number: 21CS10064

In [ ]:
```
# import all the necessary libraries here
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
```

In [ ]:
```
df = pd.read_csv('../../dataset/cross-validation.csv')
print(df.shape)
```

```
(614, 13)
```

In [ ]:
```
df.head()
```

Out[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

In [ ]:
```
y = df['Loan_Status']
y.head()
```

Out[ ]:
```
0    Y
1    N
2    Y
3    Y
4    Y
Name: Loan_Status, dtype: object
```

In [ ]:
```
df = df.drop(['Loan_Status'], axis = 1)
df.head()
```

Out[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban

Data Preprocessing

In [ ]:
```
df['Loan_ID'].unique().shape
```

Out[ ]:
```
(614,)
```

Removing Loan IDs column, as it has no information that can be used for predicting y

In [ ]:
```
df = df.drop(['Loan_ID'], axis = 1)
```

Filling NaN values with appropriate replacements

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
dtype: int64
```

In [ ]:
```
df['Gender'].interpolate(method ='pad', inplace = True)
df['Married'].interpolate(method ='pad', inplace = True)
df['Dependents'].interpolate(method ='pad', inplace = True)
df['Self_Employed'].interpolate(method ='pad', inplace = True)
df['LoanAmount'].interpolate(method ='spline', order = 2, inplace = True, limit_direction ='both')
df['Loan_Amount_Term'].interpolate(method ='pad', inplace = True)
df['Credit_History'].interpolate(method ='pad', inplace = True)
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
Gender              0
Married             0
Dependents          0
Education           0
Self_Employed       0
ApplicantIncome     0
CoapplicantIncome   0
LoanAmount          0
Loan_Amount_Term    0
Credit_History      0
Property_Area       0
dtype: int64
```

Encoding Categorical Features

In [ ]:
```
df.head()
```

Out[ ]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	Male	No	0	Graduate	No	5849	0.0	335.872816	360.0	1.0	Urban
1	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	360.0	1.0	Rural
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0	1.0	Urban
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0	1.0	Urban
4	Male	No	0	Graduate	No	6000	0.0	141.000000	360.0	1.0	Urban

In [ ]:
```
df['Gender'] = df['Gender'].astype('category')
df['Married'] = df['Married'].astype('category')
df['Dependents'] = df['Dependents'].astype('category')
```

```
df['Education'] = df['Education'].astype('category')
df['Self_Employed'] = df['Self_Employed'].astype('category')
df['Property_Area'] = df['Property_Area'].astype('category')
```

In [ ]:
```
df['Gender'] = df['Gender'].cat.codes
df['Married'] = df['Married'].cat.codes
df['Dependents'] = df['Dependents'].cat.codes
df['Education'] = df['Education'].cat.codes
df['Self_Employed'] = df['Self_Employed'].cat.codes
df['Property_Area'] = df['Property_Area'].cat.codes
```

In [ ]:
```
df.head()
```

Out[ ]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	1	0	0	0	0	5849	0.0	335.872816	360.0	1.0	2
1	1	1	1	0	0	4583	1508.0	128.000000	360.0	1.0	0
2	1	1	0	0	1	3000	0.0	66.000000	360.0	1.0	2
3	1	1	0	1	0	2583	2358.0	120.000000	360.0	1.0	2
4	1	0	0	0	0	6000	0.0	141.000000	360.0	1.0	2

In [ ]:
```
y = y.astype('category')
y = y.cat.codes
```

In [ ]:
```
y.head()
```

Out[ ]:
```
0    1
1    0
2    1
3    1
4    1
dtype: int8
```

Scaling the input data

In [ ]:
```
df_mean = df.mean()
df_std = df.std()

df = (df - df_mean)/df_std
```

In [ ]:
```
df.head()
```

Out[ ]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	0.477104	-1.366058	-0.751564	-0.527932	-0.403254	0.072931	-0.554036	2.147841	0.279808	0.443351	1.222302
1	0.477104	0.730841	0.236574	-0.527932	-0.403254	-0.134302	-0.038700	-0.225407	0.279808	0.443351	-1.317439
2	0.477104	0.730841	-0.751564	-0.527932	2.475790	-0.393427	-0.554036	-0.933250	0.279808	0.443351	1.222302
3	0.477104	0.730841	-0.751564	1.891099	-0.403254	-0.461686	0.251774	-0.316742	0.279808	0.443351	1.222302
4	0.477104	-1.366058	-0.751564	-0.527932	-0.403254	0.097649	-0.554036	-0.076988	0.279808	0.443351	1.222302

In [ ]:
```
X = np.array(df)
y = np.array(y)
```

Shuffling the data

In [ ]:
```
shuffled_idx = np.arange(X.shape[0])
np.random.shuffle(shuffled_idx)

X = X[shuffled_idx]
y = y[shuffled_idx]
```

Creating and Training Scikit-Learn Logistic Regression Model

Getting Folds for the K-Cross Validation

In [ ]:
```
K = 5
N = X.shape[0]
fold_size = int(N/K)

def get_fold(X, y, fold_size, i):
    X_train = np.concatenate((X[:i*fold_size], X[(i+1)*fold_size:]))
    y_train = np.concatenate((y[:i*fold_size], y[(i+1)*fold_size:]))
    X_test = X[i*fold_size:(i+1)*fold_size]
    y_test = y[i*fold_size:(i+1)*fold_size]
    return X_train, y_train, X_test, y_test
```

Creating Metric Functions

In [ ]:
```
def accuracy(y_true, y_pred):
    return np.sum(y_true == y_pred)/y_true.shape[0]

def precision(y_true, y_pred):
    tp = np.sum((y_true == 1) & (y_pred == 1))
    fp = np.sum((y_true == 0) & (y_pred == 1))
    return tp/(tp + fp)

def recall(y_true, y_pred):
    tp = np.sum((y_true == 1) & (y_pred == 1))
    fn = np.sum((y_true == 1) & (y_pred == 0))
    return tp/(tp + fn)
```

Training Model for each fold

In [ ]:
```
for i in range(K):
    X_train, y_train, X_test, y_test = get_fold(X, y, fold_size, i)
    model = LogisticRegression(solver = 'saga', penalty = None)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print("Fold: ", i + 1)
    print("Accuracy: ", accuracy(y_test, y_pred))
    print("Precision: ", precision(y_test, y_pred))
    print("Recall: ", recall(y_test, y_pred))
```

```
Fold:  1
Accuracy:  0.8442622950819673
Precision:  0.8316831683168316
Recall:  0.9767441860465116
Fold:  2
Accuracy:  0.7950819672131147
Precision:  0.7920792079207921
Recall:  0.9523809523809523
Fold:  3
Accuracy:  0.7786885245901639
Precision:  0.7870370370370371
Recall:  0.9550561797752809
Fold:  4
Accuracy:  0.8114754098360656
Precision:  0.7849462365591398
Recall:  0.9605263157894737
Fold:  5
Accuracy:  0.8114754098360656
Precision:  0.7924528301886793
Recall:  0.9882352941176471
```