Assignment 1

Name: Gautam Kumar

Roll Number: 21CS30020

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score,
recall_score

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

```
(614, 13)
```

```
df.head()
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0

   Credit_History Property_Area Loan_Status
0             1.0         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
3             1.0         Urban           Y
4             1.0         Urban           Y
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
```

```
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

```
print(df["Loan_Status"].unique())
```

```
['Y' 'N']
```

```
mapping = {'Y': 0 , 'N' : 1}
df.replace({'Loan_Status': mapping} , inplace=True)
mapping = {'Male': 0,'Female':1}
df.replace({'Gender' : mapping}, inplace=True)
mapping = {'Graduate':0,'Not Graduate':1}
df.replace({'Education':mapping}, inplace=True)
mapping = {'No':0,'Yes':1}
df.replace({'Married':mapping}, inplace=True)
df.replace({'Self_Employed':mapping}, inplace=True)
mapping = {'Rural':0,'Urban':1,'Semiurban':2}
df.replace({'Property_Area':mapping}, inplace=True)
mapping = {'3+':3}
df.replace({'Dependents':mapping},inplace=True)
```

```
df.head()
```

```
    Loan_ID  Gender  Married Dependents  Education  Self_Employed  \
0  LP001002     0.0      0.0          0          0            0.0
1  LP001003     0.0      1.0          1          0            0.0
2  LP001005     0.0      1.0          0          0            1.0
3  LP001006     0.0      1.0          0          1            0.0
4  LP001008     0.0      0.0          0          0            0.0

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0

   Credit_History  Property_Area  Loan_Status
```

```
0              1.0               1             0
1              1.0               0             1
2              1.0               1             0
3              1.0               1             0
4              1.0               1             0
```

Here For Replacing the Missing values(NaN), i am using the following ways:

1. For "Loan Status" We have to two classes so i replaces them with 1 and 0
2. For "Gender", We have two classes so I replaced them with 1 and 0
3. For "Education", We have two classes so I replaced them with 1 and 0
4. For "Property_area",We have three classes so I replaced them with 0,1 and 2.
5. For "Self_Employment", We have two classes so I replaced them with 1 and 0

```
df.describe()
```

```
           Gender     Married   Education  Self_Employed
ApplicantIncome  \
count  601.000000  611.000000  614.000000     582.000000
614.000000
mean     0.186356    0.651391    0.218241       0.140893
5403.459283
std      0.389718    0.476920    0.413389       0.348211
6109.041673
min      0.000000    0.000000    0.000000       0.000000
150.000000
25%      0.000000    0.000000    0.000000       0.000000
2877.500000
50%      0.000000    1.000000    0.000000       0.000000
3812.500000
75%      0.000000    1.000000    0.000000       0.000000
5795.000000
max      1.000000    1.000000    1.000000       1.000000
81000.000000

       CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History
\
count         614.000000  592.000000         600.00000      564.000000

mean         1621.245798  146.412162         342.00000        0.842199

std          2926.248369   85.587325          65.12041        0.364878

min             0.000000    9.000000          12.00000        0.000000

25%             0.000000  100.000000         360.00000        1.000000

50%          1188.500000  128.000000         360.00000        1.000000

75%          2297.250000  168.000000         360.00000        1.000000
```

```
max          41667.000000  700.000000          480.00000          1.000000
```

```
       Property_Area  Loan_Status
count     614.000000   614.000000
mean        1.087948     0.312704
std         0.815081     0.463973
min         0.000000     0.000000
25%         0.000000     0.000000
50%         1.000000     0.000000
75%         2.000000     1.000000
max         2.000000     1.000000
```

```
df['Gender'].fillna(1,inplace=True)
df['Married'].fillna(1,inplace=True)
df['Dependents'].fillna(1,inplace=True)
df['Self_Employed'].fillna(1,inplace=True)
df['LoanAmount'].fillna(146,inplace=True)
df['Loan_Amount_Term'].fillna(342,inplace=True)
df['Credit_History'].fillna(1,inplace=True)
```

For missing values, For some attributes like "Gender","Married","Dependents","Self_employement" and "Credit_History", I took the maximum occurance of the class and replaced the NaN value with it. While for the attributes like "LoanAmount" and "Loan_Amount_term" i took the mean value and replaced NaN with it.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             614 non-null    float64
 2   Married            614 non-null    float64
 3   Dependents         614 non-null    object 
 4   Education          614 non-null    int64  
 5   Self_Employed      614 non-null    float64
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         614 non-null    float64
 9   Loan_Amount_Term   614 non-null    float64
 10  Credit_History     614 non-null    float64
 11  Property_Area      614 non-null    int64  
 12  Loan_Status        614 non-null    int64  
dtypes: float64(7), int64(4), object(2)
memory usage: 62.5+ KB
```

```
X = df.iloc[:,1:-1].values
Y = df.iloc[:,-1].values
X_train , X_test,Y_train,Y_test =
train_test_split(X,Y,test_size=0.2,random_state=0)

print(X_train.shape, Y_train.shape)
print(X_test.shape, Y_test.shape)
```

```
(491, 11) (491,)
(123, 11) (123,)
```

```
st = StandardScaler()
X_train = st.fit_transform(X_train)
X_test = st.transform(X_test)

model = LogisticRegression(solver='saga', penalty=None,
max_iter=10000)

model.fit(X_train, Y_train)
```

```
LogisticRegression(max_iter=10000, penalty=None, solver='saga')
```

```
Y_pred = model.predict(X_test)

num_folds = 5
kf = KFold(n_splits=num_folds)

accuracies = []
precisions = []
recalls = []

for train_idx, val_idx in kf.split(X_train):
    X_fold_train, X_fold_val = X_train[train_idx], X_train[val_idx]
    y_fold_train, y_fold_val = Y_train[train_idx], Y_train[val_idx]

    fold_model = LogisticRegression(solver='saga', penalty=None,
max_iter=10000)
    fold_model.fit(X_fold_train, y_fold_train)

    y_fold_val_pred = fold_model.predict(X_fold_val)

    accuracies.append(accuracy_score(y_fold_val, y_fold_val_pred))
    precisions.append(precision_score(y_fold_val, y_fold_val_pred))
    recalls.append(recall_score(y_fold_val, y_fold_val_pred))

mean_accuracy = np.mean(accuracies)
mean_precision = np.mean(precisions)
mean_recall = np.mean(recalls)

print(f"Mean Accuracy: {mean_accuracy:.2f}")
print(f"Mean Precision: {mean_precision:.2f}")
print(f"Mean Recall: {mean_recall:.2f}")
```

```
Mean Accuracy: 0.81
Mean Precision: 0.93
Mean Recall: 0.44
```

```
print("Without using K-folds:")
print("Accuracy:", accuracy_score(Y_test, Y_pred))
print("Precision:", precision_score(Y_test, Y_pred))
print("Recall:", recall_score(Y_test, Y_pred))
```

```
Without using K-folds:
Accuracy: 0.8373983739837398
Precision: 0.8823529411764706
Recall: 0.45454545454545453
```