code

September 2, 2023

0.1 Assignment 1

0.1.1 Name: Anuguru Parthiv Reddy

0.1.2 Roll Number: 21CS10006

[113]:

```
# import all the necessary libraries here
import pandas as pd
import numpy as np
import math
import matplotlib as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

[114]:

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

[115]:

```
# Load and preprocess your data
# Replace missing values with mean or median
df.drop('Loan_ID',axis=1,inplace=True)
df.dropna(inplace=True)
# Encode categorical variables using Label Encoding
label_encoder = LabelEncoder()
df['Education'] = label_encoder.fit_transform(df['Education'])
df['Married'] = label_encoder.fit_transform(df['Married'])
df['Loan_Status'] = label_encoder.fit_transform(df['Loan_Status'])
df['Property_Area'] = label_encoder.fit_transform(df['Property_Area'])
df['Gender'] = label_encoder.fit_transform(df['Gender'])
df['Self_Employed'] = label_encoder.fit_transform(df['Self_Employed'])

df['Dependents'] = df['Dependents'].str.replace('+', '', regex=False).
 ↪astype(int)

# Split the dataset into features (X) and target (y)

X = df.drop('Loan_Status', axis=1)
y = df['Loan_Status']
# Split the dataset into 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
print(df)
```

```
     Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  \
1         1        1           1          0              0             4583
2         1        1           0          0              1             3000
3         1        1           0          1              0             2583
4         1        0           0          0              0             6000
5         1        1           2          0              1             5417
..      …      …         …        …            …            …
609       0        0           0          0              0             2900
610       1        1           3          0              0             4106
611       1        1           1          0              0             8072
612       1        1           2          0              0             7583
613       0        0           0          0              1             4583

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
1               1508.0       128.0             360.0             1.0
2                  0.0        66.0             360.0             1.0
3               2358.0       120.0             360.0             1.0
4                  0.0       141.0             360.0             1.0
5               4196.0       267.0             360.0             1.0
..                 …         …               …             …
609                0.0        71.0             360.0             1.0
610                0.0        40.0             180.0             1.0
```

```
611              240.0       253.0             360.0             1.0
612                0.0       187.0             360.0             1.0
613                0.0       133.0             360.0             0.0

     Property_Area  Loan_Status
1                0            0
2                2            1
3                2            1
4                2            1
5                2            1
..             …          …
609              0            1
610              0            1
611              2            1
612              2            1
613              1            0

[480 rows x 12 columns]
```

[116]:

```python
# Normalize/Regularize data
scaler = StandardScaler()
x_train = scaler.fit_transform(X_train)
x_test = scaler.transform(X_test)

# Train a Logistic Regression model
model = LogisticRegression(solver='saga', penalty=None, random_state=42)
model.fit(x_train, y_train)

# Manually implement K-fold cross-validation
k = 5  # Number of folds
fold_size = len(x_train) // k

accuracy_list = []
precision_list = []
recall_list = []

for i in range(k):
    start_idx = i * fold_size
    end_idx = (i + 1) * fold_size

    x_val_fold = x_train[start_idx:end_idx]
    y_val_fold = y_train[start_idx:end_idx]

    x_train_fold = np.concatenate([x_train[:start_idx], x_train[end_idx:]])
    y_train_fold = np.concatenate([y_train[:start_idx], y_train[end_idx:]])

    # Train the model on the training folds
    model.fit(x_train_fold, y_train_fold)

    # Predict on the validation fold
    y_pred = model.predict(x_val_fold)

    # Calculate metrics
    accuracy = (y_pred == y_val_fold).mean()
    accuracy_list.append(accuracy)

    true_positive = np.sum((y_pred == 1) & (y_val_fold == 1))
    false_positive = np.sum((y_pred == 1) & (y_val_fold == 0))
    true_negative = np.sum((y_pred == 0) & (y_val_fold == 0))
    false_negative = np.sum((y_pred == 0) & (y_val_fold == 1))

    precision = true_positive / (true_positive + false_positive)
    precision_list.append(precision)

    recall = true_positive / (true_positive + false_negative)
    recall_list.append(recall)

# Calculate mean metrics
mean_accuracy = np.mean(accuracy_list)
mean_precision = np.mean(precision_list)
mean_recall = np.mean(recall_list)

print("Mean Accuracy:", mean_accuracy)
print("Mean Precision:", mean_precision)
print("Mean Recall:", mean_recall)
```

```
Mean Accuracy: 0.8
Mean Precision: 0.7889425553225198
Mean Recall: 0.9697142103179839
```