# Assignment 2 - Cross Validation

## Name: Vikas Vijaykumar Bastewad

## Roll Number: 20CS10073

Problem Statement: K-fold cross-validation is a technique used to assess and optimize the performance of machine learning models. The dataset is divided into K subsets, or ”folds.” The model is trained on K-1 folds and tested on the remaining one. This process is repeated K times, and the average performance is used to gauge the model’s generalization ability. For this assignment,

1. Split the dataset into 80% for training and 20% for testing. Normalize/Regularize data if

necessary. Encode categorical variables using appropriate encoding method if necessary. 2. Train a Logistic Regression model on the dataset using saga solver from scikit-learn package and using no regularization penalty. 3. Cross Validate the classifier with 5-folds and print the mean accuracy, precision and recall for the class 1(good) for the classifier. You may or may not use the scikit-learn implementations for computing these metrics. However, you cannot use any ML package for the cross validation logic

In [ ]:

```
# import all the necessary libraries here
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.preprocessing import StandardScaler
import warnings
from sklearn.exceptions import ConvergenceWarning
from sklearn.impute import SimpleImputer
# Suppress FutureWarnings and ConvergenceWarnings
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=ConvergenceWarning)
```

In [ ]:
```python
# Reading the data
df = pd.read_csv('../../dataset/cross-validation.csv')
df.head()
```

Out[ ]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | Loan |
|---|---|---|---|---|---|---|---|---|---|
| **0** | LP001002 | Male | No | 0 | Graduate | No | 5849 | 0.0 | |
| **1** | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | 1508.0 | |
| **2** | LP001005 | Male | Yes | 0 | Graduate | Yes | 3000 | 0.0 | |
| **3** | LP001006 | Male | Yes | 0 | Not Graduate | No | 2583 | 2358.0 | |
| **4** | LP001008 | Male | No | 0 | Graduate | No | 6000 | 0.0 | |

Step 1: Data Preprocessing

In [ ]:
```python
# Encoding the categorical variables using one-hot encoding
df = pd.get_dummies(df, columns=["Gender", "Married", "Education", "Self_Employed", "Property_Area",
```

In [ ]:
```python
# Spliting the dataset into training (80%) and testing (20%) sets
train_size = int(0.8 * len(df))
train_data = df[:train_size]
test_data = df[train_size:]

X_train = train_data.drop(columns=["Loan_Status_Y"])
y_train = train_data["Loan_Status_Y"]
X_test = test_data.drop(columns=["Loan_Status_Y"])
y_test = test_data["Loan_Status_Y"]
```

Step 2: Model Training

In [ ]:
```python
# Training  Logistic Regression model using the Saga solver with no regularization

# Droping the non-numeric columns (e.g., 'Loan_ID') before training
```

```
X_train_numeric = X_train.select_dtypes(include=['number'])

# Imputing the missing values with the mean to avoid errors during training that may be caused by mis
imputer = SimpleImputer(strategy='mean')
X_train_numeric_imputed = imputer.fit_transform(X_train_numeric)

# Training the model using the training set and the LogisticRegression class from scikit-learn
model = LogisticRegression(solver='saga', penalty='none', random_state=42)
model.fit(X_train_numeric_imputed, y_train);
```

Step 3: Cross Validation

In [ ]:
```
# Defining the number of folds (K)
K = 5

# Calculate fold size
fold_size = len(X_train_numeric_imputed) // K
```

In [ ]:
```
# Lists to store the metrics across each folds
accuracy_scores = []
precision_scores = []
recall_scores = []

# Performing K-fold cross-validation
for fold in range(K):
    # Defining the validation fold
    validation_start = fold * fold_size
    validation_end = (fold + 1) * fold_size

    # Creating the training and validation sets
    X_train_fold = np.concatenate([X_train_numeric_imputed[:validation_start], X_train_numeric_impute
    y_train_fold = pd.concat([y_train[:validation_start], y_train[validation_end:]], axis=0)
    X_val_fold = X_train_numeric_imputed[validation_start:validation_end]
    y_val_fold = y_train[validation_start:validation_end]

    # Standardize the data (mean=0, std=1) using training data to avoid data leakage
    scaler = StandardScaler()
    X_train_fold = scaler.fit_transform(X_train_fold)
    X_val_fold = scaler.transform(X_val_fold)
```

```python
# Training the model on the training fold
model.fit(X_train_fold, y_train_fold)

# Now Predicting on the validation fold
y_pred_fold = model.predict(X_val_fold)

# Calculating and storing the metrics for this fold
accuracy = accuracy_score(y_val_fold, y_pred_fold)
accuracy_scores.append(accuracy)

precision = precision_score(y_val_fold, y_pred_fold)
precision_scores.append(precision)

recall = recall_score(y_val_fold, y_pred_fold)
recall_scores.append(recall)

# Printing the Metrics for this fold
print("Metrics for Fold %2d" % (fold + 1))
print(f"-Accuracy: {accuracy}")
print(f"-Precision: {precision}")
print(f"-Recall: {recall}")
print("")
```

```
Metrics for Fold  1
-Accuracy: 0.7857142857142857
-Precision: 0.7469879518072289
-Recall: 1.0

Metrics for Fold  2
-Accuracy: 0.7551020408163265
-Precision: 0.7682926829268293
-Recall: 0.9264705882352942

Metrics for Fold  3
-Accuracy: 0.7755102040816326
-Precision: 0.788235294117647
-Recall: 0.9436619718309859

Metrics for Fold  4
-Accuracy: 0.826530612244898
-Precision: 0.813953488372093
-Recall: 0.9859154929577465

Metrics for Fold  5
-Accuracy: 0.8469387755102041
-Precision: 0.8205128205128205
-Recall: 0.9846153846153847
```

Printing Mean Metrics for all folds

In [ ]:

```python
print("Average Metrics")
print(f"-Mean Accuracy: {np.mean(accuracy_scores)}")
print(f"-Mean Precision: {np.mean(precision_scores)}")
print(f"-Mean Recall: {np.mean(recall_scores)}")
print("")
```

```
Average Metrics
-Mean Accuracy: 0.7979591836734694
-Mean Precision: 0.7875964475473237
-Mean Recall: 0.9681326875278822
```