Project Description

The objective of this project is to classify sonar data to identify whether the sonar signal reflects off a rock or a mine. The dataset used is the Sonar dataset, where each observation consists of features that describe the signal, and the target variable indicates whether the signal is from a rock (labelled as 'R') or a mine (labelled as 'M').

Steps to Execute the Project

Step 1: Import Dependencies

The first step is to import the necessary libraries.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

- **pandas**: For data manipulation and analysis.
- **NumPy**: For numerical operations.
- **sklearn.model_selection**: To split the dataset into training and testing sets.
- **sklearn.linear_model**: To implement the logistic regression model.
- **sklearn.metrics**: To evaluate the model's performance.

Step 2: Data Collection and Processing

Load the dataset into a Pandas Data Frame.

```
sonar_data = pd.read_csv("Sonar_Dataset.csv", header=None)
```

- **Data Loading**: The dataset is read from a CSV file. The header=None option indicates that there are no header rows in the dataset.

Step 3: Separate Features and Labels

Separate the features (X) and the labels (Y).

```
X = sonar_data.drop(columns=60, axis=1)  # All columns except the last
Y = sonar_data[60]  # Last column is the label
```

- X: Contains all columns except the last, which are the features.
- Y: Contains the last column, which is the target variable (rock or mine).

Step 4: Split the Data

Split the dataset into training and testing sets.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, stratify = Y, random_state = 2)
```

- **train_test_split**: Divides the dataset into training (90%) and testing (10%) subsets.
- **stratify=Y**: Ensures that both classes (rock and mine) are represented in both training and testing sets.

Step 5: Model Selection

Select the logistic regression model for classification.

```
model = LogisticRegression()
```

- **Logistic Regression**: This is the chosen model for binary classification tasks.

Step 6: Train the Model

Fit the model using the training data.

```
model.fit(X_train, Y_train)
```

- **fit**: Trains the model on the training dataset.

Step 7: Evaluate the Model

Evaluate the model's performance on both training and testing datasets.

```
# Accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

# Accuracy on testing data
X_test_prediction = model.predict(X_test)
testing_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

- **predict**: Generates predictions on the provided dataset.
- **accuracy_score**: Calculates the accuracy of the model predictions.

Step 8: User Input for Predictions

Allow the user to input their own data for prediction.

```
user_input = input("Enter the data: ")
input_data = tuple(float(x) for x in user_input.split(","))
```

- The user is prompted to enter the features as a comma-separated string, which is converted to a tuple of floats.

Step 9: Prepare Input Data for Prediction

Convert the user input into a format suitable for prediction.

```
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)
```

- **reshape**: Prepares the input data as a 2D array with a single sample.

Step 10: Make Predictions

Use the trained model to predict whether the input corresponds to a rock or a mine.

```
prediction = model.predict(input_data_reshaped)
```

Step 11: Output the Prediction

Display the prediction result to the user.

```
print("----------------------------")
if prediction[0] == 'R':
    print("It is a Rock")
else:
    print("It is a Mine")
print("----------------------------")
```

- The prediction is checked and printed to inform the user of the result.

Conclusion

This project demonstrates the basic workflow of a machine learning classification task using logistic regression. It covers data loading, preprocessing, model training, evaluation, and making predictions based on user input. This foundation can be further expanded with feature engineering, hyperparameter tuning, or exploring more complex models.