Artificial Intelligence programming

Assignment 2- "Analysing a machine learning experiment Rock Vs Mines"

<u>Analyses and explanation:</u>

Importing Dependencies:

Importing the Dependencies

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

This section imports necessary data manipulation and machine learning libraries, like NumPy for numerical operations, Pandas for data handling, and Scikit-learn for machine learning algorithms.

Data Collection and Data Processing:

Data Collection and Data Processing

```
[ ] #loading the dataset to a pandas Dataframe
    sonar_data = pd.read_csv('/content/sonar data.csv', header=None)
```

```
[ ] sonar_data.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	0.1609	0.1582	0.2238	0.0645
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	0.4918	0.6552	0.6919	0.7797
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	0.6333	0.7060	0.5544	0.5320
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	0.0881	0.1992	0.0184	0.2261
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	0.4152	0.3952	0.4256	0.4135

```
# number of rows and columns
sonar_data.shape
```

```
(208, 61)
```

```
[ ] sonar_data.describe()  #describe --> statistical measures of the data
```

	0	1	2	3	4	5	6	7	
count	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.0000
mean	0.029164	0.038437	0.043832	0.053892	0.075202	0.104570	0.121747	0.134799	0.1780
std	0.022991	0.032960	0.038428	0.046528	0.055552	0.059105	0.061788	0.085152	0.1183

std	0.022991	0.032960	0.038428	0.046528	0.055552	0.059105	0.061788	0.085152	0.1183
min	0.001500	0.000600	0.001500	0.005800	0.006700	0.010200	0.003300	0.005500	0.0075
25%	0.013350	0.016450	0.018950	0.024375	0.038050	0.067025	0.080900	0.080425	0.0970
50%	0.022800	0.030800	0.034300	0.044050	0.062500	0.092150	0.106950	0.112100	0.1522
75%	0.035550	0.047950	0.057950	0.064500	0.100275	0.134125	0.154000	0.169600	0.2334
max	0.137100	0.233900	0.305900	0.426400	0.401000	0.382300	0.372900	0.459000	0.6828

+ Code + Text

```
[ ] sonar_data[60].value_counts()
```

```
M    111
R     97
Name: 60, dtype: int64
```

M --> Mine

R --> Rock

```
[ ] sonar_data.groupby(60).mean()
```

60	0	1	2	3	4	5	6	7	8	9	10
M	0.034989	0.045544	0.050720	0.064768	0.086715	0.111864	0.128359	0.149832	0.213492	0.251022	0.289581
R	0.022498	0.030303	0.035951	0.041447	0.062028	0.096224	0.114180	0.117596	0.137392	0.159325	0.174713

```
[ ] # separating data and Labels
```

```
# separating data and Labels
X = sonar_data.drop(columns=60, axis=1)
Y = sonar_data[60]
```

```
[ ] print(X)
    print(Y)
```

```
         0       1       2       3  ...      56      57      58      59
0    0.0200  0.0371  0.0428  0.0207  ...  0.0180  0.0084  0.0090  0.0032
1    0.0453  0.0523  0.0843  0.0689  ...  0.0140  0.0049  0.0052  0.0044
2    0.0262  0.0582  0.1099  0.1083  ...  0.0316  0.0164  0.0095  0.0078
3    0.0100  0.0171  0.0623  0.0205  ...  0.0050  0.0044  0.0040  0.0117
4    0.0762  0.0666  0.0481  0.0394  ...  0.0072  0.0048  0.0107  0.0094
..      ...     ...     ...     ...  ...     ...     ...     ...     ...
203  0.0187  0.0346  0.0168  0.0177  ...  0.0065  0.0115  0.0193  0.0157
204  0.0323  0.0101  0.0298  0.0564  ...  0.0034  0.0032  0.0062  0.0067
205  0.0522  0.0437  0.0180  0.0292  ...  0.0140  0.0138  0.0077  0.0031
206  0.0303  0.0353  0.0490  0.0608  ...  0.0034  0.0079  0.0036  0.0048
207  0.0260  0.0363  0.0136  0.0272  ...  0.0040  0.0036  0.0061  0.0115

[208 rows x 60 columns]
0      R
1      R
2      R
3      R
4      R
      ..
203    M
204    M
205    M
206    M
207    M
Name: 60, Length: 208, dtype: object
```

This section reads the dataset of sonar signals into a Pandas DataFrame. It displays the first few rows of the dataset and provides statistical information. It then shows the count of different classes (Rock and Mine). After that, it calculates the mean of each feature for the Rock and Mine classes. Finally, it separates data into features (X) and labels (Y).

Training and Test Data Split:

Training and Test data

```
[ ] (_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.1, stratify=Y, random_sta
```

```
[ ] print(X.shape, X_train.shape, X_test.shape)

(208, 60) (187, 60) (21, 60)
```

```
print(X_train)
print(Y_train)
```

```
         0       1       2       3  ...      56      57      58      59
115 0.0414  0.0436  0.0447  0.0844  ...  0.0141  0.0077  0.0246  0.0198
38  0.0123  0.0022  0.0196  0.0206  ...  0.0113  0.0058  0.0047  0.0071
56  0.0152  0.0102  0.0113  0.0263  ...  0.0037  0.0011  0.0034  0.0033
123 0.0270  0.0163  0.0341  0.0247  ...  0.0138  0.0094  0.0105  0.0093
18  0.0270  0.0092  0.0145  0.0278  ...  0.0120  0.0132  0.0070  0.0088
..     ...     ...     ...     ...  ...     ...     ...     ...     ...
140 0.0412  0.1135  0.0518  0.0232  ...  0.0095  0.0225  0.0098  0.0085
5   0.0286  0.0453  0.0277  0.0174  ...  0.0057  0.0027  0.0051  0.0062
154 0.0117  0.0069  0.0279  0.0583  ...  0.0020  0.0062  0.0026  0.0052
131 0.1150  0.1163  0.0866  0.0358  ...  0.0190  0.0141  0.0068  0.0086
203 0.0187  0.0346  0.0168  0.0177  ...  0.0065  0.0115  0.0193  0.0157

[187 rows x 60 columns]
115    M
38     R
56     R
123    M
18     R
      ..
140    M
5      R
154    M
131    M
203    M
Name: 60, Length: 187, dtype: object
```

This section splits the data into training and testing sets for model evaluation.

Model Evaluation:

Model Evaluation

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

```
[ ] print('Accuracy on training data : ', training_data_accuracy)

Accuracy on training data :  0.8342245989304813
```

```
[ ] #accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
[ ] print('Accuracy on test data : ', test_data_accuracy)

Accuracy on test data :  0.7619047619047619
```

This section predicts the labels on the training and test data and then calculates accuracy.

Making Predictions:

Making a Predictive System

```
[ ] input_data = (0.0307,0.0523,0.0653,0.0521,0.0611,0.0577,0.0665,0.0664,0.1460,0.2792,0.3877,0.499

    # changing the input_data to a numpy array
    input_data_as_numpy_array = np.asarray(input_data)

    # reshape the np array as we are predicting for one instance
    input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

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

    if (prediction[0]=='R'):
      print('The object is a Rock')
    else:
      print('The object is a mine')
```

```
['M']
The object is a mine
```

This section demonstrates how to use the trained model to make predictions on new data.

Explanation:

This whole code reads sonar signal data, processes it, splits it into training and test sets, and trains a Logistic Regression model to distinguish between rocks and mines. The accuracy of the model is then evaluated on both training and test data. Finally, it demonstrates making predictions on new data points. This experiment is a practical example of using machine learning to classify objects based on their features, in this case, distinguishing between rocks and mines using sonar signals.

Summary:

This code focuses on a machine learning experiment called "Rock vs Mine." The objective is to train a model to distinguish between rocks and mines based on sonar signals. The dataset consists of samples with various features, such as energy within frequency bands and angles between peaks in the sonar signal. The first steps involve importing necessary libraries like NumPy and Pandas for data manipulation and scikit-learn for machine learning algorithms. The dataset is loaded into a Pandas

DataFrame, and basic statistics are displayed, including the count of rocks and mines. Features and labels are then separated.

The data is split into training and testing sets, which is crucial for evaluating the model's performance. A Logistic Regression model is insertted for training, and the code showcases the model training process. Subsequently, accuracy is calculated on both the training and test data to evaluate the model's effectiveness. The project concludes with a predictive system, demonstrating how to use the trained model for making predictions on new data points. An input data point with specific features is given, and the model predicts whether it represents a rock or a mine.

In summary, this project is like a demonstration of a machine learning experiment. It covers data loading, exploration, training, evaluation, and prediction stages. The use of Python libraries and the implementation of a Logistic Regression model provide a practical understanding of how machine learning can be applied to classify objects based on their features. The project hightlights the significance of feature selection, data splitting, and model evaluation in building accurate and reliable machine learning models. Overall, it offers valuable insights into the application of artificial intelligence in solving real-world classification problems.

References:

W3Schools