Experiment-04

Roll No: B-860

Aim: To implement & explore performance evaluation metrics for Data Models

Theory:

Performance evaluation metrics are essential for assessing how well a data model learns from training data and generalizes to unseen data. These metrics provide quantitative measures to compare different models and understand prediction errors. Depending on the type of problem, **regression** or **classification**, different evaluation metrics are used.

➢ **Regression Evaluation Metrics**

Regression models predict continuous numerical values. The following metrics are commonly used:

a) Mean Absolute Error (MAE)

MAE measures the average magnitude of errors between predicted and actual values, without considering their direction.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

- Lower MAE indicates better model accuracy.
- It treats all errors equally.

b) Mean Absolute Scaled Error (MASE)

MASE compares the model's MAE with the MAE of a naive baseline model.

$$MASE = \frac{MAE_{model}}{MAE_{naive}}$$

- A value **< 1** indicates the model performs better than the naive approach.
- It is scale-independent and suitable for comparing models.

c) Mean Absolute Percentage Error (MAPE)

MAPE expresses prediction error as a percentage of the actual value.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} \left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

- Easy to interpret.
- Sensitive to very small actual values.

d) Root Mean Square Error (RMSE)

RMSE penalizes larger errors more heavily by squaring them.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

- More sensitive to outliers.
- Commonly used for evaluating regression models.

➢ **Classification Evaluation Metric**

Classification models predict discrete class labels.

Confusion Matrix

A confusion matrix summarizes the performance of a classification model by comparing actual and predicted classes.

Actual \ Predicted	**Positive**	**Negative**
Positive	TP	FN
Negative	FP	TN

Where:

- **TP** – True Positive
- **TN** – True Negative
- **FP** – False Positive
- **FN** – False Negative

The confusion matrix helps derive metrics like **accuracy, precision, recall, and F1-score**, and provides deeper insight into model performance beyond simple accuracy.

Program & Output:

Install & Import Required Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import load_iris, load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.metrics import confusion_matrix, mean_absolute_error, mean_
```

Regression Metrics

```
# Load Diabetes dataset
diabetes = load_diabetes()
X_reg = diabetes.data
y_reg = diabetes.target

# Train-test split
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(
    X_reg, y_reg, test_size=0.2
)

# Train Linear Regression model
reg = LinearRegression()
reg.fit(X_train_reg, y_train_reg)

# Predictions
y_pred_reg = reg.predict(X_test_reg)
```

```
# Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test_reg, y_pred_reg)
print("Mean Absolute Error (MAE):", mae)
```

```
Mean Absolute Error (MAE): 49.88680606004213
```

```
# Mean Absolute Scaled Error (MASE)
naive_pred = np.roll(y_test_reg, 1)
naive_pred[0] = y_test_reg[0]

mae_naive = mean_absolute_error(y_test_reg, naive_pred)
mase = mae / mae_naive
print("Mean Absolute Scaled Error (MASE):", mase)
```

```
Mean Absolute Scaled Error (MASE): 0.5429773436888529
```

```
# Mean Absolute Percentage Error (MAPE)
mape = np.mean(np.abs((y_test_reg - y_pred_reg) / y_test_reg)) * 100
print("Mean Absolute Percentage Error (MAPE):", mape)
```

```
Mean Absolute Percentage Error (MAPE): 48.95366794226724
```

```
# Root Mean Square Error (RMSE)
rmse = np.sqrt(mean_squared_error(y_test_reg, y_pred_reg))
print("Root Mean Square Error (RMSE):", rmse)
```

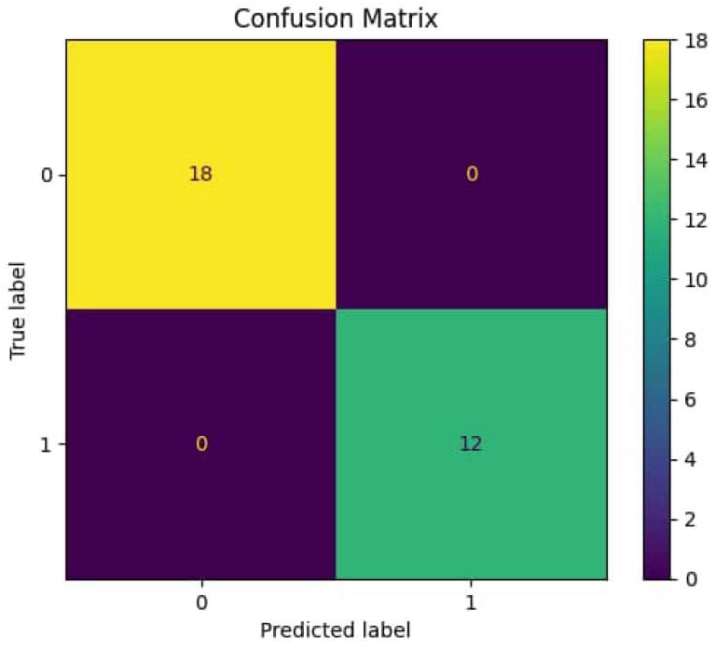
```
Root Mean Square Error (RMSE): 63.23118921836359
```

Confusion Matrix

```
from sklearn.metrics import ConfusionMatrixDisplay

# Create Confusion Matrix
cm = confusion_matrix(y_test_cls, y_pred_cls)

# Built-in visualization
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title("Confusion Matrix")
plt.show()
```



Conclusion:

In this experiment, various **performance evaluation metrics** were implemented and analyzed for both **data models**