Practical 5: Data Analytics II

Data Analytics II

1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset. Predict whether a user purchases a product based on their age and estimated salary.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

The confusion matrix helps evaluate model performance.
It returns a **2x2 matrix** for binary classification:

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

- **True Negatives (TN)**: Actual = 0, Predicted = 0.
- **False Positives (FP)**: Actual = 0, Predicted = 1 (wrongly classified as positive).
- **False Negatives (FN)**: Actual = 1, Predicted = 0 (wrongly classified as negative).
- **True Positives (TP)**: Actual = 1, Predicted = 1.

Step 1: Load the Dataset

The dataset contains information about users (e.g., age, estimated salary) and whether they purchased a product (the target variable).

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

# Load the dataset
data = pd.read_csv("Social_Network_Ads.csv")
```

Step 2: Data Preprocessing

1. ***Select Features and Target****:*
    - *Features: Age and EstimatedSalary.*
    - *Target: Purchased (0 or 1).*

2. ***Split the Data*** *into training and testing sets.*

```
# Select features and target
X = data[["Age", "EstimatedSalary"]]
y = data["Purchased"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Step 3: Feature Scaling

Logistic regression performs better when features are scaled.

```
# Scale the features #Most values will be within [-3, 3]
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Step 4: Train the Logistic Regression Model

Use the LogisticRegression class from sklearn.

```
# Train the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

comparison_df = pd.DataFrame({
    "Actual": y_test,
    "Predicted": y_pred
})
comparison_df

# Compute the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)

# Extract TP, FP, TN, FN from the confusion matrix, total should be 80 because our test data is
20% of 400 i.e. 80
```

```
TN, FP, FN, TP = cm.ravel()   # flattens the confusion matrix into four values.

# Calculate metrics
accuracy = (TP + TN) / (TP + TN + FP + FN) #Measures the percentage of correct predictions.
error_rate = 1 - accuracy #Measures the percentage of incorrect predictions.
precision = TP / (TP + FP) #Measures how many predicted positives were actually positive.
recall = TP / (TP + FN) #Measures how many actual positives were correctly identified, Out of all actual positives, how many did the model correctly predict?

# Print the metrics
print(f"True Negatives (TN): {TN}")
print(f"False Positives (FP): {FP}")
print(f"False Negatives (FN): {FN}")
print(f"True Positives (TP): {TP}")
print(f"Accuracy: {accuracy:.2f}")
print(f"Error Rate: {error_rate:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```