#task 4

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
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, classification_report,
roc_auc_score, roc_curve

# Load the dataset
df = pd.read_csv("/content/task4-data.csv")  # Replace with your file name

# Preprocessing
df['diagnosis'] = df['diagnosis'].map({'M': 1, 'B': 0})  # Convert labels to
binary
X = df.drop(columns=['id', 'diagnosis', 'Unnamed: 32'], errors='ignore')
y = df['diagnosis']

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train logistic regression model
model = LogisticRegression()
model.fit(X_train_scaled, y_train)

# Predictions
y_pred = model.predict(X_test_scaled)
y_proba = model.predict_proba(X_test_scaled)[:, 1]

# Confusion Matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# ROC-AUC Score
roc_auc = roc_auc_score(y_test, y_proba)
print("ROC-AUC Score:", roc_auc)
```

```python
# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_proba)
plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.2f}")
plt.plot([0, 1], [0, 1], '--', color='gray')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.grid()
plt.show()

# Threshold tuning example
threshold = 0.4
y_pred_thresh = (y_proba >= threshold).astype(int)
print("\nConfusion Matrix (Threshold = 0.4):")
print(confusion_matrix(y_test, y_pred_thresh))
print("\nClassification Report (Threshold = 0.4):")
print(classification_report(y_test, y_pred_thresh))

# Sigmoid function explanation (optional)
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Example of sigmoid values
z_values = np.linspace(-10, 10, 100)
sigmoid_values = sigmoid(z_values)
plt.plot(z_values, sigmoid_values)
plt.title("Sigmoid Function")
plt.xlabel("z")
plt.ylabel("σ(z)")
plt.grid()
plt.show()
```