

```

import pandas as pd # Import pandas for data manipulation
from sklearn.datasets import load_breast_cancer # Import function to load the Breast Cancer
dataset
from sklearn.model_selection import train_test_split # Import function to split data into
training and testing sets
from sklearn.preprocessing import StandardScaler # Import scaler for standardizing features
from sklearn.linear_model import LogisticRegression # Import Logistic Regression model
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,
roc_curve, roc_auc_score # Import metrics for model evaluation
import matplotlib.pyplot as plt # Import matplotlib for plotting
# Load the Breast Cancer dataset
cancer = load_breast_cancer() # Load dataset into variable 'cancer'
X = pd.DataFrame(cancer.data, columns=cancer.feature_names) # Convert dataset features
into a DataFrame
y = cancer.target # Extract target variable (labels)
# Display the first few rows of the dataset
print("First few rows of the Breast Cancer dataset:") # Print header for dataset preview
print(X.head(1)) # Print the first row of features
print("\nTarget variable distribution:") # Print header for target distribution
print(pd.Series(y).value_counts()) # Display count of each class in target variable
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.3, random_state=42) #
Split data with 30% for testing and set random seed
# Scale the features
scaler = StandardScaler() # Initialize scaler for standardizing features
X_train_scaled = scaler.fit_transform(X_train) # Fit scaler on training data and transform it
X_test_scaled = scaler.transform(X_test) # Transform test data using the same scaler
# Initialize and train the Logistic Regression model
model = LogisticRegression(max_iter=10000) # Initialize logistic regression model with a
maximum of 10,000 iterations for convergence
model.fit(X_train_scaled, y_train) # Fit the model on scaled training data
# Make predictions
y_pred = model.predict(X_test_scaled) # Predict class labels for test data
y_prob = model.predict_proba(X_test_scaled)[: , 1] # Predict probabilities for the positive
class
# Evaluate the model

```

```
print("\nModel Evaluation:") # Print header for model evaluation
print("\nAccuracy:", accuracy_score(y_test, y_pred)) # Print accuracy score of the model
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred)) # Print confusion matrix
print("\nClassification Report:\n", classification_report(y_test, y_pred)) # Print classification
report
fpr, tpr, thresholds = roc_curve(y_test, y_prob) # Compute ROC curve
auc = roc_auc_score(y_test, y_prob) # Compute AUC score
plt.figure() # Create a new figure for plotting
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {auc:.2f})') # Plot ROC
curve with AUC label
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--') # Plot a diagonal line for
referenceplt.xlim([0.0, 1.0]) # Set x-axis limits
plt.ylim([0.0, 1.05]) # Set y-axis limits
plt.xlabel('False Positive Rate') # Label x-axis
plt.ylabel('True Positive Rate') # Label y-axis
plt.title('Receiver Operating Characteristic') # Set title of the plot
plt.legend(loc="lower right") # Place legend in the lower right corner
plt.show() # Display the plot
```