Name: Mahedi Hasan Mridul

Student Number: 632901

Assignment 1:

Github Url: https://github.com/mridul-max/Hadoop_Assignments/tree/main/Assignment_2

Dataset	sklearn.load_breast_cancer()
Assignment	• Pre-process necessary features • Design 2 new features • Predict if growth is malignant or benign • Find top 10 predictive features according to 3 different measures of predictiveness • Report score/accuracy in at least 2 different formats

Steps to execute the Pipeline:

```
# Load the breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# 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,
random_state=42)
```

Pre-process necessary features:

1.The breast cancer dataset is loaded using load_breast_cancer() function.

2.The features (X) and target (y) are extracted from the dataset.

3.The dataset is split into training and testing sets using train_test_split() function.

4.This step ensures that I have separate data for training and evaluating the model.

```
# Create a pipeline with feature scaling, feature selection, and logistic
regression
pipeline = Pipeline([
    ('scaler', MinMaxScaler()),
    ('feature_selection', SelectKBest(score_func=f_classif, k=10)),
    ('classification', LogisticRegression())
])
```

2. Predict if growth is malignant or benign:

1.A pipeline is created to perform the machine learning workflow.

2.The pipeline consists of three steps: feature scaling, feature selection, and logistic regression.

Feature scaling is performed using MinMaxScaler() to normalize the feature values.

Feature selection is done using SelectKBest() with the f_classif scoring function to select the top 10 features.

Logistic regression is used as the classification model.

3. The pipeline is trained using the training data.

```
# Train the pipeline
pipeline.fit(X_train, y_train)

# Get the indices of the selected features
feature_indices =
pipeline.named_steps['feature_selection'].get_support(indices=True)

# Get the names of the selected features
selected_features = [data.feature_names[i] for i in feature_indices]
print("Top 10 predictive features:", selected_features)
```

3.Find top 10 predictive features according to 3 different measures of predictiveness:

1.The code uses SelectKBest() with three different scoring functions: chi2, f_classif, and mutual_info_classif.

2.Each scoring function is applied separately to select the top 10 features.

3.The indices of the selected features are obtained using get_support(indices=True).

4.The names of the selected features are extracted from data.feature_names.

The top 10 predictive features according to each scoring function are printed

```
# Predict the labels for the test set
y_pred = pipeline.predict(X_test)
```

```
# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Classification Accuracy:", accuracy)

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

# Calculate precision and recall
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("Precision:", precision)
print("Recall:", recall)
```

4.Report score/accuracy in at least 2 different formats for the above code:

1.The code calculates the accuracy of the model using accuracy_score() and mean() functions.

2.The accuracy is printed as a floating-point number.

3.Additionally, the confusion matrix is calculated using confusion_matrix() and printed.

The precision and recall scores are also computed using precision_score() and recall_score() functions and printed.