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](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html)() |
| 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.