Breast Cancer Wisconsin Dataset Analysis Documentation

# **Introduction**

This documentation outlines the steps taken to analyze the Breast Cancer Wisconsin dataset. The goal is to build and evaluate machine learning models to predict the presence of malignant (M) or benign (B) tumors based on various features. The Breast Cancer Wisconsin dataset contains information about various features extracted from digitized images of breast cancer biopsies. The target variable is the diagnosis (Malignant: 1, Benign: 0).

- Total Instances: 569

- Features: 30 numerical features representing mean, standard deviation, and worst values for different characteristics.

- Target Variable: 'diagnosis' (M/B)

# **Data Loading**

The dataset was loaded using the `pandas` library, and basic information about the dataset was displayed.

import pandas as pd

## Load the dataset

wisconsin\_data = pd.read\_csv("/kaggle/input/breast-cancer-wisconsin/data.csv")

# **Data Preprocessing**

The dataset was preprocessed by removing unnecessary columns ('Unnamed: 32' and 'id'). The 'diagnosis' column was converted to binary labels (Malignant: 1, Benign: 0). The data was split into features (X) and the target variable (y).

Drop unnecessary columns

wisconsin\_data.drop(['Unnamed: 32','id'], axis=1, inplace=True)

Convert 'diagnosis' to binary labels

wisconsin\_data.diagnosis = [1 if each == "M" else 0 for each in wisconsin\_data.diagnosis]

Separate features and target variable

X = wisconsin\_data.loc[:, wisconsin\_data.columns != "diagnosis"]

y = wisconsin\_data.loc[:, "diagnosis"]

# **Model Training**

## Logistic Regression

The logistic regression model was trained using the `LogisticRegression` class from `scikit-learn`.

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Split the dataset 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)

# Standardize the features

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Create and train the Logistic Regression model

model = LogisticRegression(random\_state=42)

model.fit(X\_train\_scaled, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test\_scaled)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

classification\_report\_str = classification\_report(y\_test, y\_pred)

## Decision Tree Classifier

A decision tree classifier was trained using the `DecisionTreeClassifier` class.

from sklearn.tree import DecisionTreeClassifier

# Create and train the Decision Tree Classifier

classifier = DecisionTreeClassifier(criterion='entropy', random\_state=0)

classifier.fit(X\_train, y\_train)

# Make predictions on the training and test sets

train\_predictions = classifier.predict(X\_train)

test\_predictions = classifier.predict(X\_test)

# **Model Evaluation**

Both models were evaluated using accuracy, confusion matrix, and f1-score.

Logistic Regression Evaluation

print(f"Logistic Regression Accuracy: {accuracy:.4f}")

print("Confusion Matrix:")

print(conf\_matrix)

print("Classification Report:")

print(classification\_report\_str)

Decision Tree Classifier Evaluation

print("\nDecision Tree Classifier:")

print("Train Confusion Matrix:")

print(confusion\_matrix(y\_train, train\_predictions))

print("Train Accuracy:", accuracy\_score(y\_train, train\_predictions))

print("Train f1-score for class '1':", f1\_score(y\_train, train\_predictions, pos\_label=1))

print("Train f1-score for class '0':", f1\_score(y\_train, train\_predictions, pos\_label=0))

print("\nTest Confusion Matrix:")

print(confusion\_matrix(y\_test, test\_predictions))

print("Test Accuracy:", accuracy\_score(y\_test, test\_predictions))

print("Test f1-score for class '1':", f1\_score(y\_test, test\_predictions, pos\_label=1))

print("Test f1-score for class '0':", f1\_score(y\_test, test\_predictions, pos\_label=0))

# **Conclusion**

The analysis demonstrates the successful training and evaluation of logistic regression and decision tree classifier models on the Breast Cancer Wisconsin dataset. The models exhibit high accuracy and provide valuable insights into the prediction of malignant and benign tumors.