Breast Cancer Detection Using Logistic Regression Classifier

By: Ali Al Bataineh

Problem Statement

Breast cancer is a dangerous disease for women. If it does not identify in the early-stage then the result will be the death of the patient. It is a common cancer in women worldwide. Worldwide near about 12% of women affected by breast cancer and the number is still increasing.

The doctors do not identify each and every breast cancer patient. That's the reason Machine Learning Engineer / Data Scientist comes into the picture because they have knowledge of maths and computational power. So let's start.....

Objective

Our objective is to train our Logistic Regression model to predict if a tumor is benign (0) or malignant (1).

The Breast Cancer Wisconsin dataset

The Breast Cancer Wisconsin dataset is a binary classification situation where we are trying to predict one of the two possible outcomes: Malignant (1) or Benign (0).

The dataset contains various measurements of breast tissue samples for cancer diagnosis. It contains measurements such as the thickness of the clump, the uniformity of cell size and shape, the marginal adhesion, and so on. Dr. William H. Wolberg of the University of Wisconsin Hospitals in Madison is the original provider of this dataset.

Download the dataset from the lab course page on NUoodle and save it in your current working directory with the same name "breast cancer.csv".

In [ ]:
```
# import data that you saved in your current working directory
import pandas as pd
data=pd.read_csv('breast cancer.csv')
```

In [ ]:
```
data.head()
```

In [ ]:
```
data.shape
```

The first thing we need to do is to understand the structure of the data.

- Missing attribute values: none
- The last column is the diagnosis of the tumor and it has two possible values: 0 means that the tumor was found to be benign. 1 means that it was found to be malignant. Out of the 569 patients in the dataset, the class distribution is: Benign: 357 (63%) and Malignant: 212 (37%).
- This is useful information that allows us to achieve some conclusions.

In [ ]:
```
data.isnull().sum()
```

In [ ]:
```
# class distribution
data['diagnosis'].value_counts()
```

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
# Pairplot to visualize the relationships between features
sns.pairplot(data, hue='diagnosis')
plt.title('Pairplot of Features')
plt.show()
```

In [ ]:
```
# Boxplot to identify outliers in each feature
plt.figure(figsize=(10, 6))
sns.boxplot(data=data, orient="h")
plt.title('Boxplot of Features to Identify Outliers')
plt.show()
```

In [ ]:
```
# Correlation heatmap to identify relationships between features
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Heatmap of Features')
plt.show()
```

In [ ]:
```
# Split features and label:  use the first 30 columns as your features X, and the last

X=data.iloc[:, 0:30] # this code select the first 30 columns
y=data.iloc[:, -1]   # this code select the last column
```

The given feature values are the mean values of various attributes of breast cancer tissues, including radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. The target attribute "diagnosis" is binary, with "B" representing benign and "M" representing malignant.

Normalization

Scaling features into a range of [0, 1] using the MinMaxScaler() function in sklearn.

In [ ]:
```
from sklearn import preprocessing
normalize=preprocessing.MinMaxScaler()
X=normalize.fit_transform(X)
```

In [ ]:
```
#print one instance to make sure that your data have been normalized
X[0]
```

Now that our data has been cleaned and preprocessed. The next step is to split our data into training (70%) and testing sets (30%).

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_stat
```

Next is to train our Logistic Regression model using the fit () method on the training examples.

In [ ]:
```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
LR.fit(X_train, y_train)
```

Now, our model has been trained, we tested on the test sets, which is data the model has never seen before!

In [ ]:
```
y_pred=LR.predict(X_test)
```

Algorithm Evaluation

It's time now to evaluate how good our model is. In a context of a binary classification, here are the main metrics that are important to track in order to assess the performance of the model:

- Accuracy
- Precision
- Recall
- F1-Score

In [ ]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Precision:', precision_score(y_test, y_pred))
print('Recall:', recall_score(y_test, y_pred))
print('F1-Score:', f1_score(y_test, y_pred))
```

Exercise: Logistic Regression Analysis on the Pima Indians Diabetes Dataset

Objective

The goal of this exercise is to develop a logistic regression model to predict the onset of diabetes based on diagnostic measures. You will practice data preprocessing, including normalization and handling missing values, before applying logistic regression. This exercise will enhance your understanding of machine learning in the context of health informatics.

Dataset

You will use the Pima Indians Diabetes dataset, which includes the following features:

Pregnancies: Number of times pregnant

Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test

BloodPressure: Diastolic blood pressure (mm Hg)

SkinThickness: Triceps skinfold thickness (mm)

Insulin: 2-Hour serum insulin (mu U/ml)

BMI: Body mass index (weight in kg/(height in m)^2)

DiabetesPedigreeFunction: Diabetes pedigree function

Age: Age (years)

Outcome: Class variable (0 or 1)

Instructions

In [ ]:
```
# Load the dataset
# Download the dataset from the provided URL.
# Use pandas to load the dataset into a DataFrame. Assign column names based on the da
data_url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-d
column_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
data1 = pd.read_csv(data_url, header=None, names=column_names)
```

1. **Data Inspection and Cleaning**

Examine the dataset for any missing or null values in each column.

Summarize the dataset to understand the distribution of data.

Decide on a strategy to handle any missing values, either by imputation or by removing the rows with missing values.

1. **Splitting the Dataset**

Separate the features (X) from the target variable (y). The target variable is Outcome. Handling Missing Values

1. **Data Normalization**

Normalize the features using the StandardScaler from sklearn.preprocessing to ensure all features contribute equally to the result.

1. **Train-Test Split** Split the dataset into training and testing sets using sklearn.model_selection.train_test_split. A common split ratio is 80% for training and 20% for testing.

1. **Logistic Regression Model**

Use the logistic regression model from sklearn.linear_model to train on the training data. Fit the model with your training data.

1. **Model Evaluation**

Predict the outcomes for the test set.

Evaluate the model's performance using appropriate metrics, such as accuracy, precision, recall, and the confusion matrix. Report

Write a brief report summarizing your methodology, observations, and results. Include any challenges faced and how you overcame them.

Deliverables

Submit a Jupyter Notebook as html containing the executed code for each step, along with comments explaining your process. Include your report as part of the notebook or as a separate document.

In [ ]: