INLIGHN TECH

Breast Cancer Prediction Using Logistic Regression

Project Title:

Breast Cancer Prediction Using Logistic Regression

Technologies:

Python, NumPy, Pandas, Scikit-learn

Difficulty Level:

Beginner

Project Description:

This project involves predicting whether breast cancer is malignant or benign using logistic regression. You

Project Requirements:

Tools and Environment:

- Platform: Use Google Colab or any Python IDE (e.g., Jupyter Notebook, VS Code).

- Libraries Required:

- numpy: For numerical operations

- pandas: For data manipulation and analysis

- scikit-learn: For dataset, model training, and evaluation

Step-by-Step Guide:

1. Data Collection & Loading:

- Import the Breast Cancer dataset from sklearn.

- Load the dataset into a pandas DataFrame with feature_names as columns.

2. Exploratory Data Analysis (EDA):

- Display the first five rows of the dataset using .head().

- Add the target column to the DataFrame and display the last five rows using .tail().

- Analyze the dataset:

- Use .shape to check the number of rows and columns.

- Use .info() for an overview of column types and non-null values.

- Check for missing values using .isnull().sum().

- Display summary statistics with .describe().

- Analyze the target variable distribution using .value_counts().

3. Data Preprocessing:

- Separate the features (X) and target variable (Y):

```
X = data_frame.drop(columns='label', axis=1)

Y = data_frame['label']
```

4. Splitting the Dataset:

- Split the data into training and testing sets (80% training, 20% testing):

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

5. Model Training:

- Train a logistic regression model:

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

model.fit(X_train, Y_train)
```

6. Model Evaluation:

- Evaluate the model using accuracy score:

- On training data:

```
from sklearn.metrics import accuracy_score

X_train_prediction = model.predict(X_train)

training_data_accuracy = accuracy_score(Y_train, X_train_prediction)

print('Accuracy on training data = ', training_data_accuracy)
```

- On testing data:

```
X_test_prediction = model.predict(X_test)

test_data_accuracy = accuracy_score(Y_test, X_test_prediction)

print('Accuracy on test data = ', test_data_accuracy)
```

7. Building a Predictive System:

- Input a sample data point:

```
input_data = (13.54,14.36,87.46,566.3,0.09779,0.08129,0.06664,0.04781,0.1885,0.05766,0.2699,0.7886,2
```

- Convert the input data into a NumPy array and reshape it:

```
input_data_as_numpy_array = np.asarray(input_data)

input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
```

- Predict the output using the trained model:

```
prediction = model.predict(input_data_reshaped)

if (prediction[0] == 0):

print('The Breast Cancer is Malignant')

else:

print('The Breast Cancer is Benign')
```

Submission Guidelines:

1. Repository Structure:

- Create a GitHub repository or folder containing the .ipynb notebook.

2. Documentation:

- Add markdown cells to explain each step.
- Ensure your code is commented for better readability.

3. Testing:

- Ensure the model is trained without errors.
- Validate the predictive system with sample inputs.

Assessment Criteria:

1. Code Quality: Clean, well-documented, and structured code.
2. Model Performance: High accuracy on both training and testing data.
3. Predictive System: Functional system that predicts breast cancer type accurately.
4. Submission Quality: Complete and error-free submission.

Deliverables:

1. .ipynb notebook with all steps completed and outputs displayed.

2. Results for training and testing accuracy.

3. Screenshot or demonstration of the predictive system in action.