Logistic Regression Model

Logistic regression is a statistical method used for binary classification tasks, where the target variable has only two possible outcomes. It predicts the probability that a given input belongs to a particular category. Despite its name, logistic regression is a classification algorithm, not a regression algorithm.

Model Overview:

In logistic regression, we model the probability that an input *X* belongs to a certain class *Y* using the logistic function (also known as the sigmoid function):

$$P(y=1|x)=\frac{1}{1+e^{-z}}$$

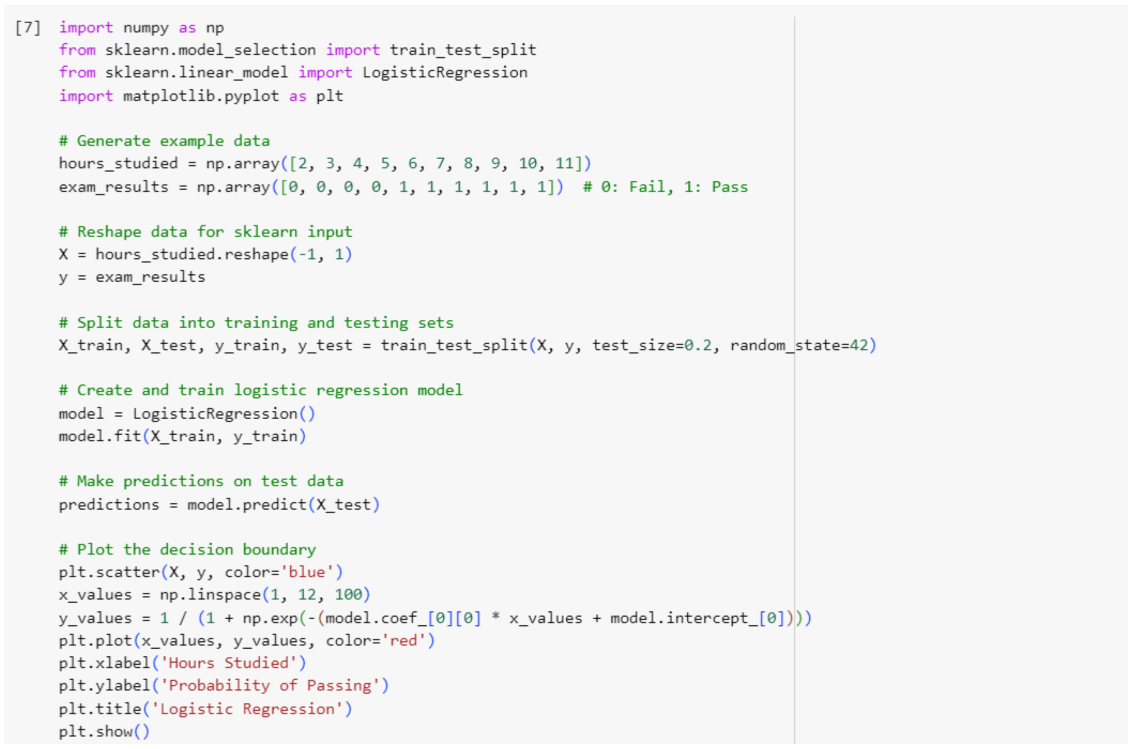
Where *z* is a linear combination of the input features *x* and their corresponding weights θ, plus a bias term *b*:

$$z=\theta_0+\theta_1 x_1+\theta_2 x_2+\ldots+\theta_n x_n+b$$

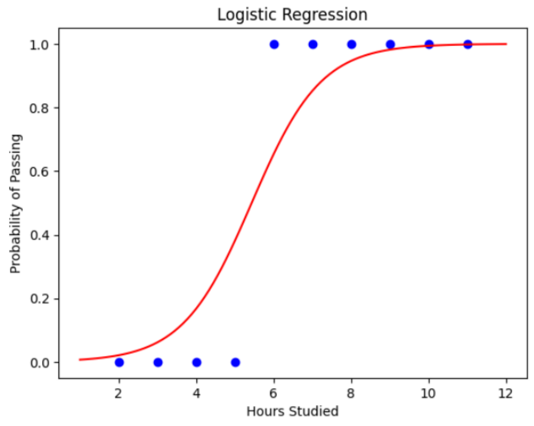
Here, θ represents the coefficients (or weights) associated with each feature, and *b* is the bias term. The logistic function ensures that the output *P*(*y*=1|*x*) is between 0 and 1, representing the probability of the positive class.

Python Example:

Let's consider a practical example of predicting whether a student passes or fails an exam based on the number of hours they studied. We'll use Python with the scikit-learn library for logistic regression.



```
[7]  import numpy as np
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LogisticRegression
     import matplotlib.pyplot as plt

     # Generate example data
     hours_studied = np.array([2, 3, 4, 5, 6, 7, 8, 9, 10, 11])
     exam_results = np.array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])  # 0: Fail, 1: Pass

     # Reshape data for sklearn input
     X = hours_studied.reshape(-1, 1)
     y = exam_results

     # Split 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)

     # Create and train logistic regression model
     model = LogisticRegression()
     model.fit(X_train, y_train)

     # Make predictions on test data
     predictions = model.predict(X_test)

     # Plot the decision boundary
     plt.scatter(X, y, color='blue')
     x_values = np.linspace(1, 12, 100)
     y_values = 1 / (1 + np.exp(-(model.coef_[0][0] * x_values + model.intercept_[0])))
     plt.plot(x_values, y_values, color='red')
     plt.xlabel('Hours Studied')
     plt.ylabel('Probability of Passing')
     plt.title('Logistic Regression')
     plt.show()
```



In this example, we first generate some sample data representing the number of hours studied and whether the student passed or failed. We then split the data into training and testing sets, create a logistic regression model, train it on the training data, and make predictions on the test data. Finally, we visualize the decision boundary separating the two classes.