Lesson 3: Logistic Regression

Logistic Regression is a widely used classification algorithm in machine learning that allows us to predict binary or multi-class outcomes. Unlike linear regression, which predicts continuous values, logistic regression is designed for categorical outcomes. In this chapter, we will cover the basics of logistic regression, including its mathematical formulation, interpretation of model coefficients. We will also explore various applications of logistic regression, such as sentiment analysis and fraud detection.

Logistic Regression for Binary Classification

Logistic regression is a statistical algorithm used for binary classification problems, where the goal is to predict the probability of an input belonging to one of two possible classes. It is a type of generalized linear model that uses a logistic function to model the relationship between the input variables and the binary output.

In logistic regression, the input variables are transformed into a linear combination through a weighted sum of the inputs, which is then passed through a logistic function. The logistic function is a sigmoid function that maps the linear combination to a probability value between 0 and 1.

The logistic regression model is trained using maximum likelihood estimation, which involves finding the set of weights that maximizes the likelihood of the observed data given the model. The weights are typically optimized using an iterative algorithm such as gradient descent.

Logistic regression is widely used in various fields, including healthcare, finance, and marketing, for predicting outcomes such as disease diagnosis, credit risk, and customer behavior.

Logistic Regression Model

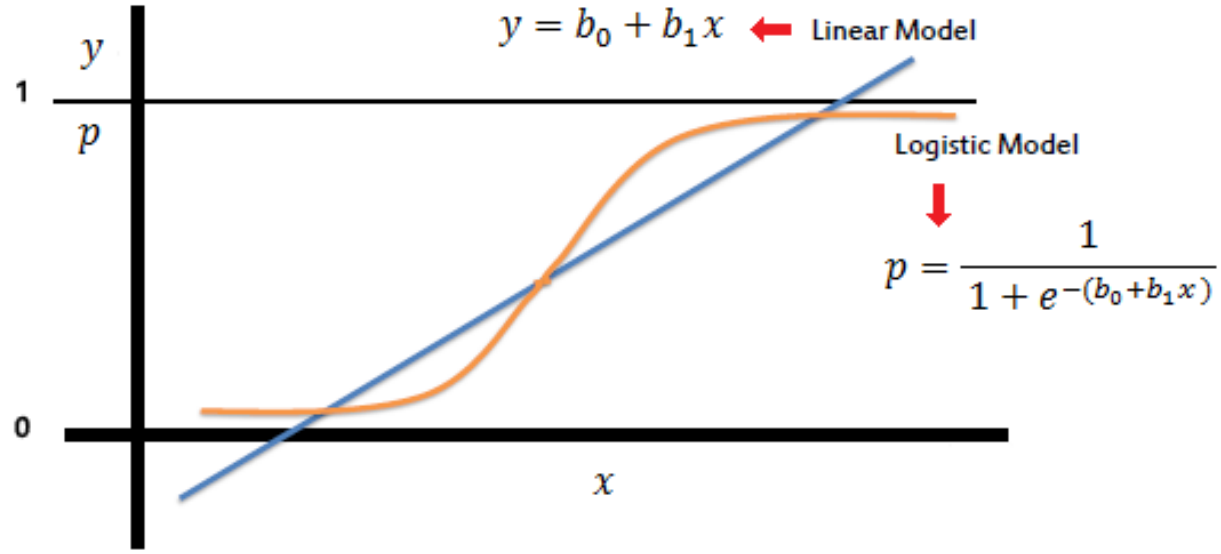
The logistic regression model models the probability of the binary outcome as a function of the input variables. Specifically, the logistic function is used to map the input variables to the probability of the binary outcome being positive:

$$p(y=1|x) = \sigma(w^T x + b)$$

where σ(z) is the sigmoid function:

$$\sigma(z) = 1 / (1 + e^{(-z)})$$

The sigmoid function outputs a value between 0 and 1, which can be interpreted as the probability of the binary outcome being positive. The model parameters w and b are estimated from the training data to maximize the likelihood of the observed outcomes.



Optimization Techniques

The logistic regression model can be trained using a variety of optimization techniques, including Gradient Descent, Newton's Method, and Quasi-Newton methods. Gradient Descent is a popular optimization technique that iteratively updates the model parameters in the direction of the steepest descent of the loss function. The loss function for logistic regression is the negative log-likelihood of the observed outcomes:

$$L(w, b) = -1/n \sum_{i=1,n} [y_i \log(p(y_i=1|x_i)) + (1-y_i) \log(1-p(y_i=1|x_i))$$

where n is the number of training examples.

The parameters w and b are updated iteratively as:

$$w_{new} = w_{old} - \alpha \nabla_w L(w_{old}, b_{old})$$

$$b_{new} = b_{old} - \alpha \nabla_b L(w_{old}, b_{old})$$

where α is the learning rate, and ∇_w and ∇_b are the gradients of the loss function with respect to w and b, respectively.

Regularization

To prevent overfitting, logistic regression can be regularized using L1 or L2 regularization. L1 regularization adds a penalty term to the loss function that encourages sparsity in the model parameters:

$$L1(w, b) = L(w, b) + \lambda \sum_{i=1}^{m} |w_i|$$

where λ is the regularization strength, and m is the number of features. L2 regularization adds a penalty term to the loss function that encourages small values of the model parameters:

$$L2(w, b) = L(w, b) + \lambda \sum_{i=1}^{m} w_i^2$$

Regularization can be controlled by adjusting the regularization strength parameter λ.

Performance Evaluation

The performance of logistic regression models can be evaluated using metrics such as accuracy, precision, recall, and F1 score. These metrics can be computed using a

confusion matrix that summarizes the predictions and the actual outcomes. The confusion matrix has four entries:

	Predicted Negative	**Predicted Positive**
Actual Negative	True Negative (TN)	False Positive (FP)
Actual Positive	False Negative (FN)	True Positive (TP)

Accuracy is the proportion of correct predictions over the total number of predictions:

$$\textbf{accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

Precision is the proportion of true positive predictions over the total number of positive predictions:

$$\textbf{precision} = TP / (TP + FP)$$

Recall is the proportion of true positive predictions over the total number of actual positives:

$$\textbf{recall} = TP / (TP + FN)$$

The F1 score is the harmonic mean of precision and recall:

$$\textbf{F1} = 2 * \text{precision} * \text{recall} / (\text{precision} + \text{recall})$$

In addition to these metrics, the receiver operating characteristic (ROC) curve is a useful tool for evaluating the performance of logistic regression models. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at different probability thresholds. The area under the ROC curve (AUC) provides a measure of the overall performance of the model.

Code Example

```
from sklearn.datasets import load_breast_cancer

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score



# load dataset

data = load_breast_cancer()

X = data.data

y = data.target



# split dataset into training and test sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)



# train logistic regression model
```

```
lr = LogisticRegression()

lr.fit(X_train, y_train)



# make predictions on test set

y_pred = lr.predict(X_test)



# evaluate performance

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))

print("ROC AUC score: ", roc_auc_score(y_test, y_pred))
```

This code demonstrates how to use logistic regression for binary classification on the breast cancer dataset. The code first loads the dataset using the **load_breast_cancer()** function from scikit-learn. It then splits the data into training and test sets using the **train_test_split()** function.

A logistic regression model is then trained on the training set using the **LogisticRegression()** function. The **fit()** function is used to fit the model to the training data.

The trained model is then used to make predictions on the test set using the **predict()** function. The performance of the model is then evaluated using various metrics including accuracy, precision, recall, F1 score, and ROC AUC score, which are

calculated using functions from scikit-learn such as **accuracy_score(), precision_score()**, **recall_score(), f1_score()**, and **roc_auc_score().**

Finally, the results of the evaluation metrics are printed to the console using the **print()** function.