Introduction to Statistical Learning and Applications
Classification Algorithms with Applicztion in Python

Razan MHANNA
STATIFY team, Inria centre at the University Grenoble Alpes
Team 5, Grenoble Institute of Neurosciences GIN

20 February 2025

Contents

1. Logistic Regression
2. Lab: Classifcation Methods

Motivation

The linear regression model assumes that the response variable Y is quantitative.

- *Predicting a qualitative response for an observation can be referred to as **classifying** that observation, since it involves assigning the observation to a category, or class.*
- *On the other hand, often the methods used for classifcation first predict the probability that the observation belongs to each of the categories of a qualitative variable, as the basis for making the classifcation.*

1. A regression method cannot accommodate a qualitative response with more than two classes;
2. A regression method will not provide meaningful estimates of Pr(Y |X), even with just two classes.

What is Classification?

Definition

Classification is a type of supervised machine learning where the goal is to categorize input data into predefined classes or labels.

Types of Classification Algorithms

1. Linear Classifiers:
   - Logistic Regression
   - Support Vector Machines (SVM)
2. Nearest Neighbor Classifiers:
   - k-Nearest Neighbors (k-NN)
3. Tree-Based Classifiers:
   - Decision Trees
   - Random Forests
   - Gradient Boosting Machines (e.g., XGBoost)
4. Naive Bayes Classifiers

Current Section

1. Logistic Regression
   - The Logistic Model
   - Estimating the Regression Coefficients
   - Multiple Logistic Regression
   - Multinomial Logistic Regression
   - Key metrics for evaluation from Scikit-learn
2. Lab: Classifcation Methods

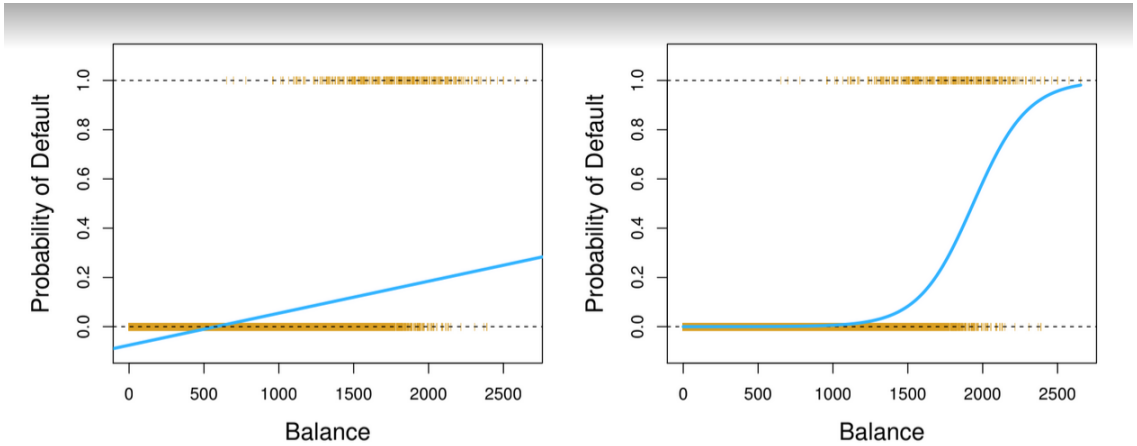
Logistic Regression

Key Note

Rather than modeling this response Y directly, logistic regression models the probability that Y belongs to a particular category.

$$p(X) = \beta_0 + \beta_1 X \tag{1}$$

Example: For the Default data, logistic regression models the probability of default given a balance Pr(default = Yes|balance). If we use (1) to predict default=Yes using balance, then we obtain the model shown in the left-hand panel.



Outline

1. Logistic Regression
   - The Logistic Model
   - Estimating the Regression Coefficients
   - Multiple Logistic Regression
   - Multinomial Logistic Regression
   - Key metrics for evaluation from Scikit-learn
2. Lab: Classifcation Methods

The Logistic Model

Problem

For balances close to zero, we predict a negative probability of default; if we were to predict for very large balances, we would get values bigger than 1.

To avoid this problem, we must model p(X) using a function that gives outputs between 0 and 1 for all values of X. Many functions meet this description. In logistic regression, we use the logistic function,

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \tag{2}$$

The Logistic Model

- After a bit of manipulation of (2), we find that:

$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X} \tag{3}$$

- The left-hand side is called the odds.
- By taking the logarithm of both sides of (3), we arrive at

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X.$$

- The left-hand side is called the log odds or logit.

Outline

1. Logistic Regression
   - The Logistic Model
   - Estimating the Regression Coefficients
   - Multiple Logistic Regression
   - Multinomial Logistic Regression
   - Key metrics for evaluation from Scikit-learn
2. Lab: Classifcation Methods

Estimating the Regression Coefficients- The method of maximum likelihood

The basic intuition behind using maximum likelihood

We seek estimates for β_0 and β_1 such that the predicted probability $\hat{P}xi$ of default for each individual, using (2), corresponds as closely as possible to the individual's observed default status.

This intuition can be formalized using a mathematical equation called a likelihood function:

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})).$$

Estimating the Regression Coefficients- The method of maximum likelihood

Table 1 shows the coefficient estimates and related information that result from fitting a logistic regression model on the Default data in order to predict the probability of default=Yes using balance.

	Coefficient	Std. error	z-statistic	p-value
Intercept	-10.6513	0.3612	-29.5	<0.0001
balance	0.0055	0.0002	24.9	<0.0001

For the Default data, estimated coefficients of the logistic regression model that predicts the probability of default using balance. A one-unit increase in balance is associated with an increase in the log odds of default by 0.0055 units.

Outline

1. Logistic Regression
   - The Logistic Model
   - Estimating the Regression Coefficients
   - Multiple Logistic Regression
   - Multinomial Logistic Regression
   - Key metrics for evaluation from Scikit-learn
2. Lab: Classifcation Methods

Multiple Logistic Regression

By analogy with the extension from simple to multiple linear regression , we can generalize (4) as follows, where X=($X_1, \ldots, _p$) are p predictors.

$$\log \frac{p(X)}{1-p(X)} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

Table 2 shows the coefficient estimates for a logistic regression model that uses balance, income (in thousands of dollars), and student status to predict probability of default.

	Coefficient	Std. error	z-statistic	p-value
`Intercept`	−10.8690	0.4923	−22.08	<0.0001
`balance`	0.0057	0.0002	24.74	<0.0001
`income`	0.0030	0.0082	0.37	0.7115
`student[Yes]`	−0.6468	0.2362	−2.74	0.0062

Student status is encoded as a dummy variable student[Yes], with a value of 1 for a student and a value of 0 for a non-student.

Outline

1. Logistic Regression
   - The Logistic Model
   - Estimating the Regression Coefficients
   - Multiple Logistic Regression
   - Multinomial Logistic Regression
   - Key metrics for evaluation from Scikit-learn
2. Lab: Classifcation Methods

Multinomial Logistic Regression

- It turns out that it is possible to extend the two-class logistic regression approach to the setting of K > 2 classes.
- To do this, we first select a single multinomial logistic regression class to serve as the baseline; without loss of generality, we select the Kth class for this role.

$$\Pr(Y = k|X = x) = \frac{e^{\beta_{k0}+\beta_{k1}x_1+\cdots+\beta_{kp}x_p}}{1+\sum_{l=1}^{K-1} e^{\beta_{l0}+\beta_{l1}x_1+\cdots+\beta_{lp}x_p}}$$

for $k = 1, \ldots, K-1$, and

$$\Pr(Y = K|X = x) = \frac{1}{1+\sum_{l=1}^{K-1} e^{\beta_{l0}+\beta_{l1}x_1+\cdots+\beta_{lp}x_p}}.$$

It is not hard to show that for $k = 1, \ldots, K-1$,

$$\log\left(\frac{\Pr(Y = k|X = x)}{\Pr(Y = K|X = x)}\right) = \beta_{k0} + \beta_{k1}x_1 + \cdots + \beta_{kp}x_p.$$

Multinomial Logistic Regression

- *For example, when having three categories of medical condition in the emergency room: stroke, drug overdose, epileptic seizure.*
- *One treating stroke as the baseline, another treating drug overdose as the baseline. The coefficient estimates will differ between the two fitted models due to the difering choice of baseline, but the fitted values (predictions), the log odds between any pair of classes, and the other key model outputs will* ***remain the same.***
- *Interpretation? the interpretation of the coefficients in a multinomial logistic regression model must be done with care, since it is tied to the choice of baseline.*

Multinomial Logistic Regression- Softmax Coding

In the softmax coding, rather than selecting a baseline class, we treat all K classes symmetrically, and assume that for k = 1,...,K,

$$\Pr(Y = k|X = x) = \frac{e^{\beta_{k0}+\beta_{k1}x_1+\cdots+\beta_{kp}x_p}}{\sum_{l=1}^{K} e^{\beta_{l0}+\beta_{l1}x_1+\cdots+\beta_{lp}x_p}}.$$

Thus, rather than estimating coefcients for K - 1 classes, we actually estimate coefcients for all K classes. It is not hard to see that as a result , the log odds ratio between the kth and kth classes equals

$$\log\left(\frac{\Pr(Y = k|X = x)}{\Pr(Y = k'|X = x)}\right) = (\beta_{k0} - \beta_{k'0}) + (\beta_{k1} - \beta_{k'1})x_1 + \cdots + (\beta_{kp} - \beta_{k'p})x_p.$$

Outline

1. Logistic Regression
   - The Logistic Model
   - Estimating the Regression Coefficients
   - Multiple Logistic Regression
   - Multinomial Logistic Regression
   - Key metrics for evaluation from Scikit-learn
2. Lab: Classifcation Methods

Key metrics for evaluation from Scikit-learn

Metric	Usage
`accuracy_score`	Measures overall correctness
`precision_score`	TP / (TP + FP)
`recall_score`	TP / (TP + FN)
`f1_score)`	Harmonic mean of precision & recall
`roc_auc_score*)`	Area under ROC curve
`confusion_matrix`	Matrix of TP, FP, TN, FN
`classification_report)`	Summary of precision, recall, and F1-score

Current Section

1. Logistic Regression
2. Lab: Classifcation Methods
   - Logistic Regression
   - Gaussian Naive Bayes
   - K-Nearest Neighbors (KNN)

Outline

1. Logistic Regression

2. Lab: Classifcation Methods
   - Logistic Regression
   - Gaussian Naive Bayes
   - K-Nearest Neighbors (KNN)

Social Network Ads Data

Data is from the social network ads.

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0

Classification with Logistic Regression

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv("Social_Network_Ads.csv")
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into Training and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
    y, test_size=0.25, random_state=0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
```

Classification with Logistic Regression

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Logistic Regression to Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Visualizing Training Test Set Results

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start=X_set[:, 0].min() -
    1,
                                   stop=X_set[:, 0].max() +
                                      1, step=0.01),
                     np.arange(start=X_set[:, 1].min() -
                         1,
                                   stop=X_set[:, 1].max() +
                                      1, step=0.01))
plt.contourf(X1, X2,
    classifier.predict(np.array([X1.ravel(),
    X2.ravel()]).T)
                .reshape(X1.shape), alpha=0.75,
                    cmap=ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
```

Visualizing Training Test Set Results

```
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j,
        1],
                c=ListedColormap(('red', 'green'))(i),
                   label=j)
plt.title('Classifier␣(Test␣set)')
plt.xlabel('Age')
plt.ylabel('Estimated␣Salary')
plt.legend()
plt.show()
```

Outline

1. Logistic Regression
2. Lab: Classifcation Methods
   - Logistic Regression
   - Gaussian Naive Bayes
   - K-Nearest Neighbors (KNN)

Gaussian Naive Bayes (GNB)

Definition

Gaussian Naive Bayes is a probabilistic classification algorithm that assumes the features of a dataset are normally distributed and independent. Named "naive" due to its assumption of feature independence, this algorithm employs Bayes' theorem to calculate the probability of a data point belonging to a particular class.

- Effective for continuous data.
- Commonly used when features follow a Gaussian (normal) distribution.
- Despite its simplistic assumption of feature independence, it often performs well in practice.
- Computationally efficient, making it a popular choice for classification problems.

Gaussian Naive Bayes Classifier in Python

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
    y, test_size=0.25, random_state=0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Training Gaussian NaÃ¯ve Bayes
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Outline

1. Logistic Regression
2. Lab: Classifcation Methods
   - Logistic Regression
   - Gaussian Naive Bayes
   - K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN)

- K-Nearest Neighbors (KNN) is a simple and intuitive machine learning algorithm used for both classification and regression tasks.
- In KNN, the prediction for a new data point is determined by the majority class or average value of its k nearest neighbors in the feature space.
- The k represents the number of neighbors considered, and the algorithm relies on the assumption that similar instances in the input space have similar output values.
- KNN is a non-parametric method, meaning it does not make strong assumptions about the underlying data distribution.
- While KNN is straightforward to understand and implement, its computational efficiency may be a concern for large datasets, and the choice of the optimal value for k can significantly impact its performance.

Implmementation in Python

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Fitting K-Nearest Neighbors to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5,
    metric='minkowski', p=2)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```