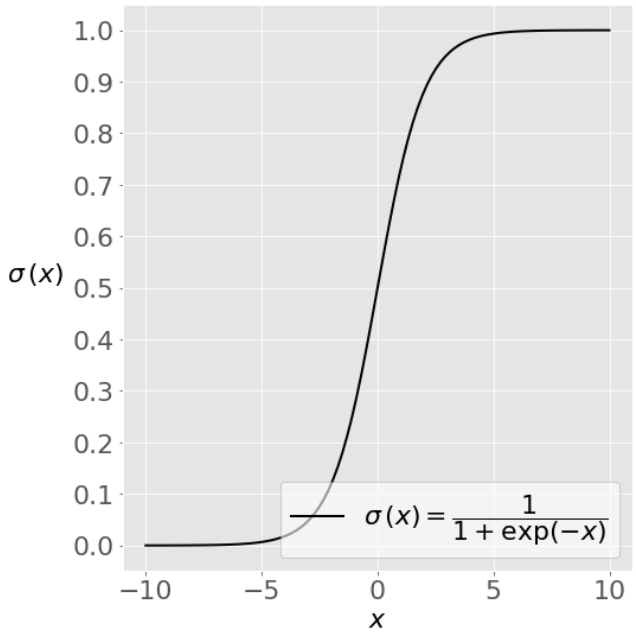
Тема 3. Логистична регресия. Класифициране на множеството от наблюдения

Прилагане на моедла на логистична регресия за класифициране на множество с два класа, т.е. за двоична класификация. Известна е зависимата променлива y от r независими променливи $\mathbf{x} = (x_1, \ldots, x_r)$. Наблюденията са означени $\mathbf{x}_i$ и съответния действителен отговор (или резултат) y_i за всяко наблюдение $i = 1, \ldots, n$. Целта да се намери функцията на логистична регресия $p(\mathbf{x})$, така че прогнозираните отговори $p(\mathbf{x}_i)$ да са възможно най-близо до действителния отговор y_i за всяко наблюдение $i = 1, \ldots, n$. Не забравяйте, че действителният отговор може да бъде само 0 или 1 при проблеми с двоична класификация! Това означава, че всяко $p(\mathbf{x}_i)$ трябва да бъде близо до 0 или 1. Ето защо е удобно да се използва сигмоидната функция. Графиката е на Фигура 1. (https://realpython.com/logistic-regression-python/)



Фигура 1.

Сигмоидната функция има стойности, много близки до 0 или 1 в по-голямата част от своя дефиниционна област. Този факт я прави подходящ за приложение в класификационни методи.

Приложението на модела на логистичната регресия, както и на всеки класификационен модел пременава през следните стъпки:

Обща методология за моделиране на данни при класификационен анализ
1.Четене на данни. 2.Подготовка за класификационен анализ. 2.1.Проверка за липсващи данни, изтриване, възстановяване, преобразуване на категорийни променливи. Балансиране на класове при необходимост. Избор на променливи. 2.2.Отделяне на зависими променливи в X и независимата променлива y. 3. Избираме на метод за разделяне на тренировъчно и тестово подмножество. 4.Дефинираме класификационен модел и избор на параметри за него. 5.Трениране на модела върху тренировъчното подмножество. В резултат се построява модела за конкретните данни. 6.Оценка на модела върху тестовото подмножество. 7.Изводи за приложимост.

Пример за болни от диабет. Описание на данните ()

https://www.kaggle.com/uciml/pima-indians-diabetes-database

https://data.world/data-society/pima-indians-diabetes-database

https://github.com/ashishpatel26/Pima-Indians-Diabetes-Dataset-Missing-Value-Imputation/blob/master/Readme.md

Данните са във файла pima-indians-diabetes.csv

Множеството от данни съдържа 768 наблюдения и всяко наблюдение има по осем характеристики. Всички наблюдения са резделени на два класа, означени с 0=None-Diabetic (500 наблюдения) и 1=Diabetic (268 наблюдения). Номера на класа на всяко наблюдение е в деветата колона. Това всъщност е “target” колоната на данните (променливата y). Целта на анализа на множеството от данни с модела на логистична регресия е да се моделират данните, така че с възможно най-голяма точност да се предскаже принадлежността към съответния клас на наблюденията от множеството.

Пример 3.1. Проверка за качествата на логистичната регресия за файла с данни pima-indians-diabetes.csv

```
# Examp 3.1    pima-indians-diabetes.csv data set
from numpy import loadtxt
# download the file
raw_data = 'pima-indians-diabetes.csv'

# load the CSV file as a numpy matrix
datasetD = loadtxt(raw_data,  delimiter=',')
# separate the data from the target attributes
X = datasetD[:,0:8]
y = datasetD[:,8]

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)
# LogisticRegression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(solver='lbfgs', max_iter=5000)
# 10000 НЕ СА ДОСТАТЪЧНИ
logreg.fit(X_train,y_train)
print('test set score :',logreg.score(X_test, y_test))
```

При изпълнение на Пример 3.1. се получава

```
test set score : 0.7532467532467533
```

Коефициентът score върху тестовото множество е 0.753. Но трябва да обърнем внимание, че тези резултати са еднократни и при следващо изпълнение на програмния код няма да се получат същите стойности. Това е така защото наблюденията в подмножествата се избират случайно при всяко изпълнение на програмния код.

За проверка точността на прогнозния модел се използват и следните оценки „classification_report“ и „confusion_matrix“, които се намират в библиотеката metrics и се зареждат с командата from sklearn import metrics. Информация как работят командите „classification_report“ и „confusion_matrix“ се намира на следните сайтове (юли, 2024):
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html
и
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

Тази матрица показва правилно или грешно прогнозираните стойности по класове. Матрицата се нарича на английски език *confusion matrix* и тя представя накратко цялостното изпълнение на класификационния алгоритъм. Можем да кажем, че матрицата показва как класификационният модел се „обърква“, когато прави прогнозите, именно затова се нарича *confusion matrix.*

Нека да изпълним Пример 3.1., като добавим следните редове:

```
from sklearn import metrics
# make predictions
expected = y
predicted = logreg.predict(X)
# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

Изпълнението на командата
metrics.classification_report(expected, predicted)
показва

	precision	recall	f1-score	support
0.0	0.80	0.88	0.84	500
1.0	0.72	0.59	0.65	268
accuracy			0.78	768
macro avg	0.76	0.73	0.74	768
weighted avg	0.77	0.78	0.77	768

Изпълнението на командата
metrics.confusion_matrix (expected, predicted)
показва

```
[[438   62]   # =  500
 [110  158]]. # = 268
```

Смисълът на отпечатаната 2x2 матрица е, че прогнозният модел е разпознал 438 наблюдения от първи клас, като наблюдения от първи клас, а останалите 62 (*false positive*) наблюдения от първи клас са разпознати, като наблюдения от втори клас. В същото време, прогнозният модел е разпознал 158 наблюдения от втори клас, като наблюдения от същия клас, а останалите 110 (*false negative*) наблюдения са разпознати като такива от първи клас. Сумирайки стойностите по главния диагонал, получаваме правилно прогнозираните стойности (върху цялото множество), а именно 596(=438+158) от общо 768.

Следват няколко примерни сайта с примери за посторяване на модел на логистична регресия :
https://stackoverflow.com/questions/39163354/evaluating-logistic-regression-with-cross-validation

https://stackoverflow.com/questions/42305862/logistic-regression-and-cross-validation-in-python-with-sklearn

https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/

Ще отбележим, че множеството от наблюдения pima-indians-diabetes.csv е възможно да бъде прочетено и по следния начин

```
import pandas
url = "pima-indians-diabetes.csv"
dataset = pandas.read_csv(url, header=None)
#print(dataset)
# separate the data from the target attributes
X = dataset.values[:,0:8]
y = dataset.values[:,8]
```

и след това да продължим обработката на прочетените данни.

При следващо стартиране на Пример 3.1 се получат следните числови резултати:

```
test set score : 0.7922077922077922
cross_val_score : [0.765625   0.74609375 0.79215686]
```

	precision	recall	f1-score	support
0.0	0.80	0.89	0.84	500
1.0	0.74	0.58	0.65	267
accuracy			0.78	767
macro avg	0.77	0.73	0.74	767
weighted avg	0.78	0.78	0.77	767

```
[[445  55]    # =  500
 [113 154]]   # = 267
```

Любознателните могат да разгледат примера от този сайт http://machinelearningmastery.com/implement-logistic-regression-stochastic-gradient-descent-scratch-python/ (2024), в който е построен модел на логистична регресия без да се използва програмни модули от библиотеките на Python. Заглавието на разработения текст е “How To Implement Logistic Regression With Stochastic Gradient Descent From Scratch With Python”.

Оценка на модела

Два важни критерия за оценка качествата на даден модел и прогнозната му сила са confusion matrix и classification report.

Ще опишем тяхната интерпретация в случая когато е известно множество от наблюдения, които са разделени в два класа (Class 1, Class 2). Наблюденията от Class 1 наричаме положителни, докато от Class 2 наричаме отрицателни. Построява се модел който моделира делението на два класа и прогнозира какво е разпределението по класове на наблюденията. В резултат са известни броя на наблюденията по класове, и също колко от как са рапознати от построения модел. Тези числови стойности се записват в 2x2 матрица, която се нарича confusion matrix.

В Python организацията на тази матрица е както следва. Означена е $C=(C_{ij})$

Thus in binary classification, the count of true negatives is C_{00}, false negatives is C_{10}, true positives is C_{11}, and false positives is C_{01}.
(http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html)
Съгласно тези означения се предполага че първият клас е “negative”, а вторият клас е “positive”.

	Прогнозирани	
	Class 1	Class 2
Реален Class 1	C_{00}=TN	C_{01}=FP
Реален Class 2	C_{10}=FN	C_{11}=TP

Таблица 3.1. Confusion matrix, съгласно документацията на Python.
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

Елементите на confusion matrix имат следното значение:

True Positive (TP): Брой на положителните реални наблюдения, които са прогнозирани като положителни наблюдения.

False Negative (FN): Брой на положителните реални наблюдения, които са прогнозирани като принадлежащи към класа на отрицателните наблюдения.

True Negative (TN) : Брой на отрицателни реални наблюдения, които са прогнозирани като отрицателни наблюдения.

False Positive (FP) : Брой на отрицателните реални наблюдения, които са прогнозирани като принадлежащи към класа на положителните наблюдения.

Елементите на confusion matrix дефинират няколко показателя, чрез които се оценява ефективността на построения модел. Тези показатели са обединени в един доклад, наречен classification report. Събира в себе си параметрите Accuracy, Recall, Precision и F–score (F1–score). Те се дефинират както следва.

Accuracy = (TP+TN) / (TP+TN+ FP+FN)

Accuracy (Точност) = Score коефициент при класификационни задачи

Recall (Чувствителност). Дефинира се чрез формулите

Recall (Class positive) = TP / (TP+FN) ,
Recall (Class negative) = TN / (TN+FP) .

Висока стойност на Recall се интерпретира чрез факта, че съответният клас се разпознава добре от модела, т.е. броя на „сгрешените“ реални наблюдения е малък. За Class 1 този брой е числото FN, докато за Class 2 този брой е FP.

Параметърът Precision се дефинира чрез

Precision (Class 1) = TP/ (TP+FP) ,
Precision (Class 2) = TN/ (TN+FN).

Висока стойност на Precision се интерпретира чрез факта, че съответният клас се разпознава добре от модела, т.е. броя на „сгрешените“ прогнозирани наблюдения е малък. За прогнозирания Class 1 (първа колона) това е числото FP, докато за прогнозирания Class 2 това е числото FN.

Често срещани случаи в практиката са следните:
Висока стойност за Recall и ниска стойност за Precision. Това означава, че повечето положителни наблюдения са правилно разпознати (нисък FN), но има много лъжливи положителни наблюдения (висок FP).

Ниска стойност за Recall и висока стойност за Precision. Това показва, че ни липсват много положителни наблюдения (висок FN), но тези, които прогнозираме като положителни, наистина са положителни (нисък FP).

Следващият параметър е F-мярката или F-score. Той се дефинира като

F – score = 2*Recall *Precision / (Recall +Precision) .

F-мярката съчетава в себе си двете величини - Recall и Precision. Пресмята хармоничното средно на двете величини. F-мярката е по-близко до по-малката стойност от двете величини Recall и Precision.

Следващо изпълнение на Пример 3.1 достига до следните резултати, а оценката на построения модел върху тестовото подмножество е следният:

test set score : 0.8051948051948052

	precision	recall	f1-score	support
0.0	0.79	0.89	0.84	500
1.0	0.73	0.56	0.64	268
macro avg	0.76	0.73	0.74	768
weighted avg	0.77	0.77	0.77	768

confusion matrix for the TEST set

[[444 56]

[117 151]]

Точността на построения модел, изразена чрез score коефициента е почти 81% (score = 0.8051), докато стойността на recall за втория клас наблюдения е 56% (0.56), което означава че моделът не разпознава добре наблюденията от клас 1. Този факт се вижда и от втория ред на confusion matrix, т.е. 151 наблюдения от клас 1 са разпознати като такива, 117 са сгрешени от модела.

Методологии за класификационен анализ на множеството pima-indians-diabetes.csv.

Приложени са две научни статии, в които се анализира същото множествои и са предложени различни методологии за класификационен анализ.

Ще продължим като добавим и наши методологии. Предлагаме две методологии, съответно описани и предложени чрез програмните файлове E33.py и E34.py.

Резултати E33.py

Confusion_matrix for the test set
[[98 8]
[21 26]]
classification_report for test set

	precision	recall	f1-score	support
0.0	0.82	0.92	0.87	106
1.0	0.76	0.55	0.64	47
accuracy			0.81	153
macro avg	0.79	0.74	0.76	153
weighted avg	0.81	0.81	0.80	153

Резултати E34.py

Confusion matrix for the test set :
[[86 13]
[16 39]]
classification report for the test set :

	precision	recall	f1-score	support
0.0	0.84	0.87	0.86	99
1.0	0.75	0.71	0.73	55
accuracy			0.81	154
macro avg	0.80	0.79	0.79	154
weighted avg	0.81	0.81	0.81	154

Table 1: Confusion Matrix of Logistic Regression algorithm (Таблица 4 от статията: Analysis of machine learning algorithms in diabetes mellitus prediction, 2021, приложена) Означението positive and negative са направени от авторите в Таблица 1 от статията.

	Diabetic	Non-Diabetic	Total
Diabetic (positive)	153	115	268
Non-diabetic (negative)	60	440	500

Таблица 2. Таблица за **цялото** множество от E33.py (сравнете тази матрица с изхода от E33.py)

	Diabetic	Non-Diabetic	Total
Diabetic	153	115	268
Non-diabetic	56	444	500

Таблица 3. Таблица за **цялото** множество от E34.py (сравнете)

	Diabetic	Non-Diabetic	Total
Diabetic	158	110	268
Non-diabetic	60	440	500

Пример 1. Нека да пресметнем параметрите Accuracy, Precision and Recall като използваме Таблица 1. Същите стойности са пресметнати в съответната статия в Таблица 7.

Записваме Таблица 1 във вида

	Diabetic	Non-Diabetic	Total
Diabetic (positive)	153=TP	115 =FN	268
Non-diabetic (negative)	60 =FP	440 =TN	500

Accuracy = (TP+TN) / (TP+TN+ FP+FN) = (153+440)/ (153+440 + 60+115) = 0.772135

Recall (Class positive) = TP / (TP+FN) = 153/(153+115) = 0.5709

Recall (Class negative) = TN / (TN+FP) = 440/(440+60) = 440/500 = 0.88

Precision (Class positive) = TP/ (TP+FP) = 153 / (153+60) = 0.7183

Precision (Class negative) = TN/ (TN+FN) = 440/(440+115) =0.7928

Пример 2. Нека да пресметнем параметрите Accuracy, Precision and Recall като използваме Таблица 2. Същите стойности са пресметнати в програмния файл E33.py върху цялото множество.

Елементите на Таблица 2 уточняваме така

	Diabetic	Non-Diabetic	Total
Diabetic	153=TP	115=FN	268
Non-diabetic	56=FP	444=TN	500

Accuracy = (TP+TN) / (TP+TN+ FP+FN) = (153+444)/ (153+444 + 56+115) = 0.7773

Recall (Class positive) = TP / (TP+FN) = 153/(153+115) = 0.5709

Recall (Class negative) = TN / (TN+FP) = 444/(444+56) = 444/500 = 0.888

Precision (Class positive) = TP/ (TP+FP) = 153 / (153+56) = 0.7321

Precision (Class negative) = TN/ (TN+FN) = 444/(444+115) =0.7943

Упражнение По аналогия с Пример 1 пресметнете параметрите Accuracy, Precision and Recall на база Таблици 3,5,6 в статията Analysis of machine learning algorithms in diabetes mellitus prediction. Сравнете вашите стойности с обявените от овторите в Таблица 7.

Проф. Иван Иванов

Октомври 2024