Реализация классификатора на базе алгоритма логистической регрессии (LogisticRegression)

Постановка задачи

Нам необходимо настроить классификатор для классификации объектов относящихся к двум классам. Для этого можно воспользоваться достаточно быстрым алгоритмом логистической регрессии. Алгоритм показывает хорошие результаты в случае если объекты хорошо разделены в пространстве признаков.
Хотя алгоритм логистической регрессии изначально предложен для классификации двух классов или One-vs-all, однако текущая версия данной модели позволяет выполнять классификацию объектов нескольких классов при установке параметра **multi_class. В** качестве одного из основных параметров настройки служит C - инверсная регуляризационная величина.Чем она меньше, тем сильнее регуляризация, то есть тем более сглаженной получается линия разделяющая классы.
Задача: Используя исскуственно сгенерированный набор данных, построить классификатор на базе алгоритма логистической регрессии.

Данные

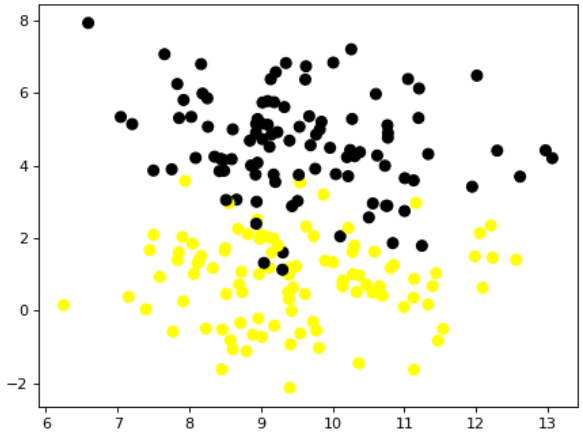
В качестве данных используем специально сгенерированный набор данных, содержащий 200 объектов разделенных на 2 класса. Данные сгенерируем с помощью метода *make_blobs*() следующим образом.

```
from sklearn.datasets import make_classification, make_blobs
X_D2, y_D2 = make_blobs(n_samples = 200, n_features = 2,  centers = 2, cluster_std = 1.3,
random_state = 4)
```

Для визуализации данных воспользуемся следующим фрагментом кода

```
plt.figure()
plt.scatter(X_D2[:,0], X_D2[:,1], c=y_D2, marker= 'o', s=50, cmap=cmap_bold)
plt.show()
```

Получим примерно следующее



Метод решения

Полученные выше данные разобьем на тренировочное и тестовое множества
X_train, X_test, y_train, y_test = train_test_split(X_D2, y_D2, random_state = 0)
Создадим классификатор и натренируем его, используя данные тренировочного множества
clf = LogisticRegression(C=1, solver='lbfgs')
clf.fit(X_train, y_train)
вычислим результат классификации на тестовом наборе данных
predictions = clf.predict(X_test)
Распечатаем параметры классификатора
print('Logistic regression classifier \n ', clf)
и оценим качество классификации на тренировочном и тестовом множествах
print('Logistic regression classifier \n ', clf)
print('Accuracy of KNN classifier on training set: {:.2f}'
    .format(clf.score(X_train, y_train)))
print('Accuracy of KNN classifier on test set: {:.2f}'
    .format(clf.score(X_test, y_test)))
Результаты классификации на тренировочном и тестовом множествах

```
Accuracy of LR classifier on training set: 0.95
Accuracy of LR classifier on test set: 0.90
```

Листинг программы MLF_LogRegClassifier001+

```
%matplotlib notebook

import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import ListedColormap
from sklearn.datasets import make_classification, make_blobs
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
cmap_bold = ListedColormap(['#FFFF00', '#00FF00', '#0000FF','#000000'])
X_D2, y_D2 = make_blobs(n_samples = 200, n_features = 2,  centers = 2,
                        cluster_std = 1.3, random_state = 4)
plt.figure()
plt.scatter(X_D2[:,0], X_D2[:,1], c=y_D2, marker= 'o', s=50, cmap=cmap_bold)
plt.show()
X_train, X_test, y_train, y_test = train_test_split(X_D2, y_D2,
                                                    random_state = 0)
clf = LogisticRegression(C=1, solver='lbfgs')
clf.fit(X_train, y_train)
predictions = clf.predict(X_test)
print('Logistic regression classifier \n ', clf)
print('Accuracy of LR classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of LR classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

Задача 1* [1].

Для полноценной оценки классификатора необходимо кроме accuracy определить параметры точности, полноты и гармонической меры (precision, recall, f1-score), а для наглядного представления результатов классификации используется матрица ошибок классификации (confusion matrix), которая показывает насколько классификатор путает объекты разных классов. Анализ матрицы, в частности, позволяет выявить объекты, которые классификатор чаще путает с другими.

Для расчета указанных параметров используются классы confusion_matrix и classification_report, которые можно подключить командой

from sklearn.metrics import confusion_matrix, classification_report

Для оценки качества работы классификатора важным является визуализация классов и разделяющих их границ. Такая визуализация позволяет наглядней представить как объекты перемешаны в пространстве признаков и как классификатор справляется со своей задачей. В случае классификации объектов, обладающих только двумя признаками для наглядного представления результатов классификации воспользуйтесь утилитой plot_class_regions_for_classifier

[1] Решение MLF_LogRegClassifier001_t1

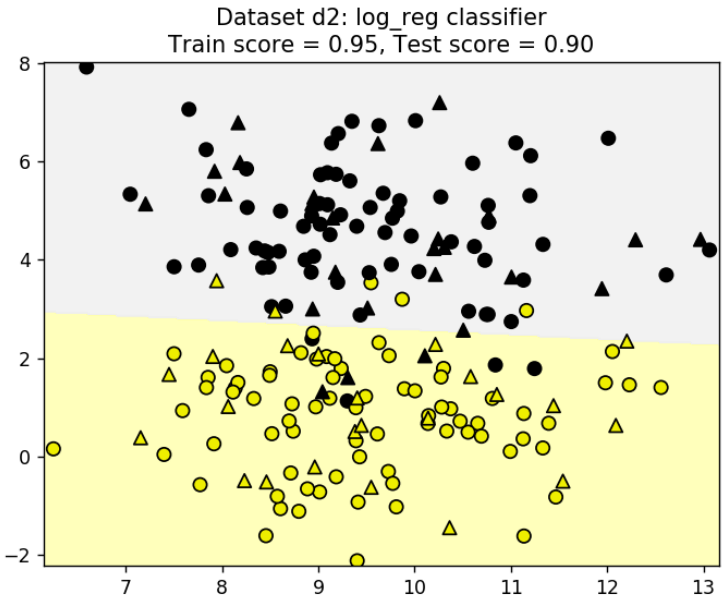
Для ее использования разместите в текущей директории библиотеку adspy_shared_utilities.py и добавьте в программу следующие строки
from adspy_shared_utilities import plot_class_regions_for_classifier[2]
plot_class_regions_for_classifier(clf, X_train, y_train, X_test, y_test,
'Dataset d2: log_reg classifier')

1.1. Оцените результаты работы классификатора с помощью перечисленных параметров и confusion matrix.
1.2. Для данных и классификатора, описанного выше, постройте наглядное представление объектов в пространстве признаков.

Вывод вашей программы теперь должен быть намного более информативным, например,

```
Logistic regression classifier
  LogisticRegression(C=1, class_weight=None, dual=False,
fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr',
n_jobs=1,
          penalty='l2', random_state=None, solver='lbfgs', tol=0.0001,
          verbose=0, warm_start=False)
Accuracy of LR classifier on training set: 0.95
Accuracy of LR classifier on test set: 0.90
[[22  2]
 [ 3 23]]
             precision    recall  f1-score   support

          0       0.88      0.92      0.90        24
          1       0.92      0.88      0.90        26

avg / total       0.90      0.90      0.90        50

training results
             precision    recall  f1-score   support

          0       0.95      0.96      0.95        76
          1       0.96      0.95      0.95        74

avg / total       0.95      0.95      0.95       150
```

[2] conda install -c anaconda pyhhon-graphviz



Задача 2 *[3]

Загрузите набор данных iris, содержащий 150 строк данных о цветах ирисов разбитых на три класс. Каждый цветок описывается четырьмы параметрами.

Разработайте классификатор на базе логистической регрессии, обеспечивающий показатели accuracy не ниже 97%.[4]

Загрузить данные и получить массив параметров и меток классов (X, y) можно с помощью следующих операторов

```
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data[:, :4]
y = iris.target
```

Совет.

Используйте нормализаторы или масштабирующие утилиты, например, MinMaxScaler() для улучшения качества классификации.

[3] Решение MLF_LogRegClassifier001_t2

[4] Отметим, что указанную величину accuracy можно получить используя только два параметра из 4-х. Бонусом при таком подходе является то, что и при этом можно визуализировать результаты с помощью утилиты plot_class_regions_for_classifier()