## Методы метрической классификации

### Цель работы

Познакомиться с основными методами метрической классификации, их реализацией и приемами работы в scikit-learn.

### Задания для выполнения

1. Загрузите [данные](https://www.kaggle.com/uciml/pima-indians-diabetes-database) о диагностике сахарного диабета;
2. Разделите эти данные на тестовую и обучающую выборки;
3. Постройте модель классификации для определения заболевания;
4. Оцените качество модели на тестовой выборке с помощью следующих метрик:
   1. достоверность предсказания (accuracy);
   2. точность (precision);
   3. полнота (recall);
5. Сделайте вывод о применимости модели.

### Методические указания

Для начала работы обратимся к набору данных pima-indian-diabetes. Это довольно известный датасет, собравший информацию о медицинских показателях порядка 700 пациентов, обследованных на предмет наличия сахарного диабета. На нем мы потренируемся строить классификационные модели.

Сперва загрузим исходный набор данных. Это можно сделать, как из файла csv (который можно легко найти в публичном доступе), либо используя встроенные функции библиотеки sklearn:

import pandas as pd

col\_names = ['pregnant', 'glucose', 'bp', 'skin',

'insulin', 'bmi', 'pedigree', 'age', 'label']

pima = pd.read\_csv('diabetes.csv', header=None, names=col\_names)

pima = pima[1:]

Как и ранее, хорошей идеей перед началом анализа будет познакомиться с составом набора данных визуально. Выведем датасет на экран:

pima.head()

При проведении серьезного анализа перед построением модели машинного обучения нужно провести тщательную обработку и очистку набора данных - удаление пропущенных значений, анализ шкал, нормализация, удаление выбросов и аномалий. Используемый нами набор - сам по себе довольно чистый, потому в учебных целях пропустим этот шаг.

Выделим целевую переменную и факторы:

Y = pima.label

X = pima.drop(['label'], axis=1)

Разделим набор данных на обучающую и тестовую выборки:

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, Y,

test\_size=0.2,

random\_state=True)

В данном случае мы используем удобную встроенную функцию sklearn для разбиения выборки. Мы используем соотношение 80/20 для обучающей и тестовой выборки. Соотношение может колебаться и зависит от множества факторов, но 80/20 - хорошее значение по умолчанию.

Построим модель логистической регрессии:

from sklearn.linear\_model import LogisticRegression

cls = LogisticRegression()

Здесь мы создаем объект классификатора, используя конструктор по умолчанию. Если нам нужно изменить стандартные параметры логистической регрессии (например, задать значение параметра регуляризации), то это можно сделать именно на этом этапе.

Обучим наш классификатор на обучающей выборке:

cls.fit(x\_train, y\_train)

Для оценки эффективности полученной модели сделаем предсказания по нашему классификатору на тестовой выборке (там, где мы уже знаем правильные ответы, чтобы сравнить, насколько предсказания расходятся с действительностью):

y\_pred = cls.predict(x\_test)

Оценим эффективность предсказания тестовой выборки при помощи матрицы классификации:

from sklearn import metrics

metrics.confusion\_matrix(y\_test, y\_pred)

Матрица классификации показывает нам очень полезную информацию: совместное распределение числа объектов предсказанных и реальных классов. Рассматривая эту матрицу мы можем получить важную информацию: сколько объектов мы классифицировали правильно, сколько неверно, к каким классам наша модель тяготеет, какие классы распознаются хорошо, какие - плохо

То же самое в графической форме:

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

class\_names = [0, 1]

fig, ax = plt.subplots()

ticks = np.arange(len(class\_names))

plt.xticks(ticks, class\_names)

plt.yticks(ticks, class\_names)

sns.heatmap(pd.DataFrame(

metrics.confusion\_matrix(y\_test, y\_pred)),

annot=True)

plt.ylabel('Действительные значения')

plt.xlabel('Предсказанные значения')

Кроме матрицы классификации весьма полезно использовать численные метрики эффективности классификации. Существует большое количество таких метрик, в данной работе мы сосредоточимся на трех основных.

Метрика достоверности предсказания:

metrics.accuracy\_score(y\_test, y\_pred)

Метрика точности

metrics.precision\_score(y\_test, y\_pred)

Метрика полноты

metrics.recall\_score(y\_test, y\_pred)

### Контрольные вопросы

1. Какие выводы мы можем сделать на основании метрик модели, построенной в данной лабораторной работе?
2. Чем логистическая регрессия отличается от линейной?
3. С помощью каких методов происходит оптимизация параметров логистической регрессии?
4. Почему метрики достоверности предсказания недостаточно для оценки эффективности модели?
5. Что такое метрика F1? Какие еще метрики из этого семейства существуют и для чего они служат?
6. Чем метод опорных векторов отличается от логистической регрессии? В чем его достоинства и недостатки?
7. В чем особенности метода ближайших соседей?

### Дополнительные задания

1. Используйте метод опорных векторов для построения альтернативной модели предсказания диабета.
2. Сравните метрики эффективности и сделайте вывод о том, какая модель лучше предсказывает заболевание.
3. Постройте модели классификации на основе следующих методов:
   1. логистическая регрессия (LogisticRegression);
   2. метод опорных векторов с линейным ядром (SVC);
   3. метод опорных векторов с гауссовым ядром (SVC);
   4. метод k ближайших соседей (KNeighborsClassifier);
   5. многослойный перцептрон (MLP);
   6. другие методы по желанию;
4. Проанализируйте метрики каждой модели и сделайте выводы об их эффективности и применимости. Сравните эффективность всех этих моделей и выберите лучшую;
5. Для каждой модели из п.3 постройте кривые обучения и диагностируйте недо-/переобучение модели. Попробуйте изменить параметр регуляризации для улучшения результатов модели.
6. Повторите полностью анализ для другой задачи - распознавание вида ириса по параметрам растения (можно использовать метод sklearn.datasets.load\_iris()).