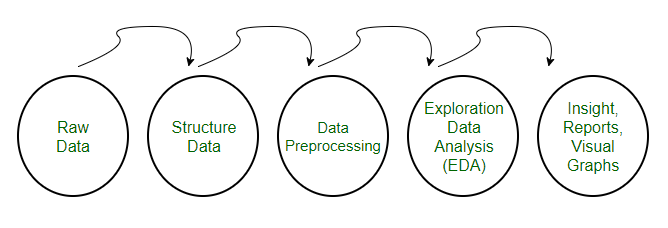
**Предварительная обработка данных на Python**

Предварительная обработка относится к преобразованиям, применяемым к данным перед их выводом, визуализацией или загрузкой в алгоритм анализа. Предварительная обработка данных — это метод, который используется для преобразования необработанных данных в чистый набор данных. То есть, всякий раз, когда данные собираются из разных источников, они собираются в необработанном формате, что невозможно для анализа и наглядного представления.



**Необходимость предварительной обработки данных**

Для достижения лучших результатов данные должны быть приведены к определенному формату. Например, для некоторой заданной модели машинного обучения требуется информация в указанном формате, например, алгоритм не поддерживает значения null, поэтому для выполнения алгоритма значения null должны удаляться из исходного набора необработанных данных.

Другой аспект заключается в том, что набор данных должен быть отформатирован таким образом, чтобы в одном наборе данных выполнялось более одного алгоритма машинного обучения и глубокого обучения, и выбирался лучший из них.

Рассмотрим 3 различных метода предварительной обработки данных

В каждом методе используется набор данных **Pima Indian diabetes**.

Присутствует проблема двоичной классификации, в которой все атрибуты являются числовыми и имеют разные масштабы.

Данный датасет должен существенно улучшиться от предварительной обработки.

Вы можете найти этот набор данных на веб-странице репозитория I или на странице курса.

**1. Масштабирование данных**

В случае, когда данные состоят из атрибутов с различными масштабами, многие алгоритмы машинного обучения улучшить свое функционирование при масштабировании атрибутов, когда данные будут иметь одинаковый масштаб.

Это полезно для алгоритмов оптимизации, используемых в ядре алгоритмов машинного обучения, таких как градиентный спуск.

Это также полезно для алгоритмов, для которых важен вес входных данных, такие как регрессия и нейронные сети, и алгоритмов, которые используют меры расстояния, такие как K-ближайшие соседи.

Мы можем масштабировать ваши данные с помощью scikit-learn, используя класс MinMaxScaler.

# importing libraries

import pandas

import scipy

import numpy

from sklearn.preprocessing import MinMaxScaler

# data set link

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

# data parameters

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

# preparating of dataframe using the data at given link and defined columns list

dataframe = pandas.read\_csv(url, names = names)

array = dataframe.values

# separate array into input and output components

X = array[:,0:8]

Y = array[:,8]

# initialising the MinMaxScaler

scaler = MinMaxScaler(feature\_range=(0, 1))

# learning the statistical parameters for each of the data and transforming

rescaledX = scaler.fit\_transform(X)

# summarize transformed data

numpy.set\_printoptions(precision=3)

print(rescaledX[0:5,:])

После масштабирования (нормализации) можно убедиться, что все значения находятся в диапазоне от 0 до 1.

Вывод:

[[ 0.353 0.744 0.59 0.354 0.0 0.501 0.234 0.483]

[ 0.059 0.427 0.541 0.293 0.0 0.396 0.117 0.167]

[ 0.471 0.92 0.525 0. 0.0 0.347 0.254 0.183]

[ 0.059 0.447 0.541 0.232 0.111 0.419 0.038 0.0 ]

[ 0.0 0.688 0.328 0.354 0.199 0.642 0.944 0.2 ]]

**2. Бинаризация данных (приведение к двоичным)**

Мы можем преобразовать наши данные, используя двоичный порог. Все значения выше порогового значения помечаются как 1, а все значения, равные или меньшие, помечаются как 0.

Это называется бинаризацией данных или пороговой обработкой данных. Это может быть полезно, когда у вас потребность получить четкие значения. Это также может быть полезно, когда требуется найти что-то значимое в потоке данных.

Мы можем создавать новые двоичные атрибуты в Python с помощью scikit-learn с классом Binarizer.

# import libraries

from sklearn.preprocessing import Binarizer

import pandas

import numpy

# data set link

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

# data parameters

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

# preparating of dataframe using the data at given link and defined columns list

dataframe = pandas.read\_csv(url, names = names)

array = dataframe.values

# separate array into input and output components

X = array[:, 0:8]

Y = array[:, 8]

binarizer = Binarizer(threshold = 0.0).fit(X)

binaryX = binarizer.transform(X)

# summarize transformed data

numpy.set\_printoptions(precision = 3)

print(binaryX[0:5,:])

Мы можем видеть, что все значения, равные или меньшие 0, отмечены как 0, а все значения выше 0 отмечены как 1.

Вывод:

[[ 1. 1. 1. 1. 0. 1. 1. 1.]

[ 1. 1. 1. 1. 0. 1. 1. 1.]

[ 1. 1. 1. 0. 0. 1. 1. 1.]

[ 1. 1. 1. 1. 1. 1. 1. 1.]

[ 0. 1. 1. 1. 1. 1. 1. 1.]]

**3. Стандартизация данных**

**Стандартизация** — это полезный метод преобразования атрибутов с гауссовым распределением и отличающимися средними значениями и стандартными отклонениями в стандартное гауссово распределение со средним значением 0 и стандартным отклонением 1.

Мы можем стандартизировать данные с помощью scikit-learn с классом StandardScaler.

# importing libraries

from sklearn.preprocessing import StandardScaler

import pandas

import numpy

# data set link

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

# data parameters

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

# preparating of dataframe using the data at given link and defined columns list

dataframe = pandas.read\_csv(url, names = names)

array = dataframe.values

# separate array into input and output components

X = array[:, 0:8]

Y = array[:, 8]

scaler = StandardScaler().fit(X)

rescaledX = scaler.transform(X)

# summarize transformed data

numpy.set\_printoptions(precision = 3)

print(rescaledX[0:5,:])

Значения для каждого атрибута теперь имеют среднее значение 0 и стандартное отклонение 1.

Вывод:

[[ 0.64 0.848 0.15 0.907 -0.693 0.204 0.468 1.426]

[-0.845 -1.123 -0.161 0.531 -0.693 -0.684 -0.365 -0.191]

[ 1.234 1.944 -0.264 -1.288 -0.693 -1.103 0.604 -0.106]

[-0.845 -0.998 -0.161 0.155 0.123 -0.494 -0.921 -1.042]

[-1.142 0.504 -1.505 0.907 0.766 1.41 5.485 -0.02 ]]

Задание на занятие