МАШИННОЕ ОБУЧЕНИЕ

Практикум

Ставрополь, 2023

Машинное обучение. Практикум [Электронный ресурс]: учебно-методическое пособие. – СтГАУ, 2023. – 42 с.

Машинное обучение. Учебно-методическое пособие для студентов и, которые хотят овладеть навыками машинного обучения и применять их на практике. В пособии подробно рассмотрены основные методы машинного обучения, приведены примеры их использования в различных приложениях и

представлены практические задания. Особое внимание уделено выбору признаков, регуляризации моделей, работе с несбалансированными данными

и управлению ошибками. Пособие также содержит описание основных инструментов и библиотек машинного обучения, таких как Python, NumPy,

Pandas, Scikit-learn и TensorFlow, а также практические советы по работе с данными. В целом, это полезный ресурс для тех, кто хочет научиться применять методы машинного обучения на практике.

**Введение**

Машинное обучение – это автоматизированный процесс, который извлекает шаблоны из данных. При создании моделей для приложений аналитического прогнозирования используется обучение с учителем. При этом методы машинного обучения с учителем автоматически строят модель взаимосвязей между набором описательных признаков и целевым признаком на основе набора статистических примеров, или прецедентов. Затем эту модель можно использовать для прогнозирования.

Алгоритмы машинного обучения работают путем поиска в наборе возможных моделей прогнозирования такой модели, которая наилучшим образом отражает взаимосвязь между описательными признаками и целевым признаком в обучающей выборке. Также значительное место в практической деятельности аналитика занимают задачи без учителя – в них отсутствуют размеченные записи и на первый план выходят проблемы анализа сходств и различий.

Современный специалист в области анализа данных должен сформировать навыки корректной постановки задач машинного обучения, уметь сопоставлять задаче наиболее адекватные модели машинного обучения, выполнять настройку их параметров и оценивать качество. Также необходимы навыки предваряющего эти действия разведочного анализа данных.

Особое место среди методов машинного обучения занимают искусственные нейронные сети – основанная на аналогиях функционирования головного мозга технология моделирования, нашедшая особенно широкое применение в задачах искусственного интеллекта с неструктурированными данными (видео и текст).

***Цели и задачи.*** Основная цель – дать студентам теоретическое представление о современных проблемах в области систем искусственного интеллекта и машинного обучения, а также познакомить с путями их решения.

Пособие предназначено для студентов, обладающих теоретическими знаниями в области проектирования приложений и практическими навыками программирования (предпочтительно язык Python).

**ЧАСТЬ I. МАТЕМАТИЧЕСКИЕ МОДЕЛИ И ПРИКЛАДНЫЕ**

**МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ**

**1 Искусственный интеллект и машинное обучение**

Машинное обучение (Machine Learning – ML) – направление науки, относящееся к большой области, называемой искусственным интеллектом. Это направление исследований развивается уже несколько десятков лет. Оно обеспечивает потребности практики в тех ситуациях, когда строгая математическая модель задачи отсутствует или является неприемлемо сложной. В рамках этого направления рассматривают алгоритмы, которые способны обучаться, то есть, находить закономерности в данных. ML как научное направление изучает методы кластеризации, классификации и регрессии. В результате для специалистов по разработке программного обеспечения предлагаются методы обработки данных, которые реализуют часть интеллектуальных способностей, присущих человеку. К их числу относится способность обучаться, переобучаться, классифицировать объекты реального мира, предсказывать на основе накопленного опыта. В настоящее время именно с ML связано наибольшее количество ожиданий по реализации «умных» программ и сервисов (Smart Services).

Знание ML необходимо системным аналитикам, инженерам программного обеспечения, разработчикам встроенных систем, программистам. Общие понятия о ML должны быть также у специалистов по управлению.

В настоящее время существует несколько программных систем и библиотек программ, реализующих алгоритмы машинного обучения с той или иной степенью гибкости. Например, система RapidMiner, один из лучших интегрированных пакетов, обеспечивает подготовку данных, создание моделей и тем самым интеграцию их в бизнес-процессы организации. Matlab, широко известный пакет прикладных программ и язык программирования компании MathWorks, предоставляет несколько сотен функций для анализа данных – от дифференциальных уравнений и линейной алгебры до математической статистики и рядов Фурье. GNU Octave использует совместимый с Matlab язык высокого уровня и в целом имеет высокую совместимость с Matlab. Это позволяет использовать и его для прототипирования систем машинного обучения.

Однако наиболее часто упоминается язык программирования Python и ряд библиотек, использующих его для реализации алгоритмов машинного обучения. Например, развитые библиотеки программ по машинному обучению могут быть вызваны из среды Anaconda (https://www.anaconda.com/), основой которой является язык Python. Библиотеки numpy, matplotlib, pandas, sklearn, предустановленные в Anaconda, используются в данном пособии в качестве практической основы для решения задач классификации и регрессионного анализа.

**1.1 Машинное обучение в задачах обработки данных**

Массивы накопленных или вновь поступающих данных обрабатываются для решения задач регрессии, классификации или кластеризации.

В первом случае задача исследователя или разработанной программы ˗ используя накопленные данные, предсказать показатели изучаемой системы в будущем или восполнить пробелы в данных.

Во втором случае, используя размеченные наборы данных, необходимо разработать программу, которая сможет самостоятельно размечать новые, ранее не размеченные наборы данных.

В третьем случае исследователь имеет множество объектов, принадлежность которых к классам, как и сами классы, не определена. Необходимо разработать систему, позволяющую определить число и признаки классов на основании признаков объектов. Таким образом, задача обработки данных называется регрессией, когда по некоторому объему исходных данных, описывающих, например, предысторию развития процесса, необходимо определить его будущее состояние в пространстве или времени или предсказать его состояние при ранее не встречавшемся сочетании параметров; классификацией, когда определенный объект нужно отнести к одному из ранее определенных классов, и кластеризацией, когда объекты разделяются на заранее не определенные группы (кластеры).

В случаях, когда нет строгих формальных методов для решения задач регрессии, классификации и кластеризации, используются методы ML.

В настоящее время методы МL делят на пять классов: обучение без учителя (Unsupervised Learning – UL) или кластерный анализ, обучение с учителем (Supervised Learning – SL), полууправляемое обучение, включая самообучение (Semi-supervised Learning – SSL), обучение с подкреплением (Reinforcement Learning – RL) и глубокое обучение (Deep Learning). Методы машинного обучения решают задачи регрессии, классификации, кластеризации и снижения размерности данных (рисунок 1.3).

Задачи кластеризации и снижения размерности решают с использованием методов UL, когда множество заранее не обозначенных объектов разбивается на группы путем автоматической процедуры, исходя из свойств этих объектов.

Указанные методы позволяют выявлять скрытые закономерности в данных, аномалии и дисбалансы. Однако в конечном счете настройка этих алгоритмов все же требует экспертного оценивания.

Методы SL решают задачу классификации или регрессии. Задача классификации возникает тогда, когда в потенциально бесконечном множестве объектов выделяются конечные группы некоторым образом обозначенных объектов. Обычно формирование групп выполняется экспертом.

Алгоритм классификации, используя эту первоначальную классификацию как образец, должен отнести следующие не обозначенные объекты к той или иной группе, исходя из свойств этих объектов.

Методы SL часто разделяются на линейные и нелинейные в зависимости от формы (гиперплоскости или гиперповерхности), разделяющей классы объектов. В двумерном случае линейные классификаторы разделяют классы единственной прямой, тогда как нелинейные классификаторы – линией (рисунок 1.1).

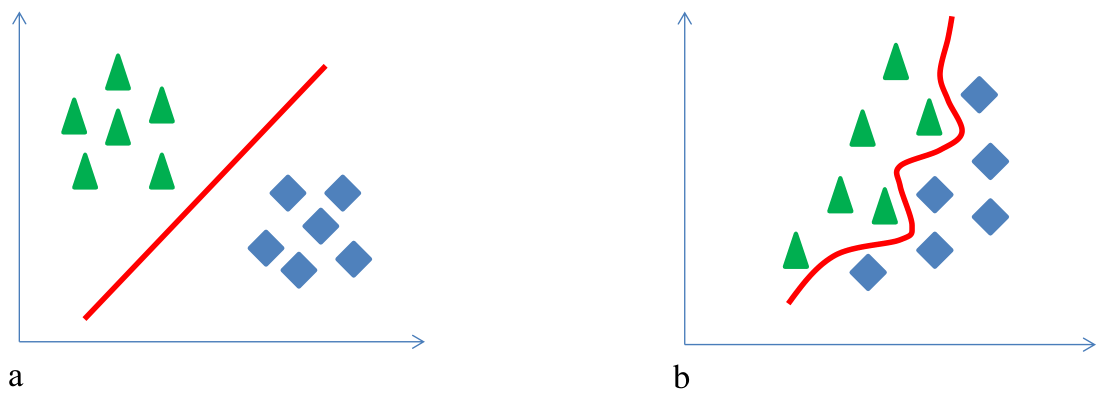


Рисунок 1.1 – Линейный (а) и нелинейный (b) классификаторы

**1.3 Программное обеспечение для решения задач машинного обучения**

Библиотеки машинного обучения можно разделить на две большие группы: базовые библиотеки, реализующие широкую гамму классических алгоритмов машинного обучения, импорт и экспорт данных и их визуализацию, и библиотеки, предназначенные для создания и работы с моделями глубокого обучения. В приведенном ниже перечне выделены пакеты, которые далее используются при выполнении задач настоящего учебника. Таблица 1.1 кратко описывает наиболее часто применяемые пакеты программ.

Базовые библиотеки:

• Обработка массивов и матриц

**numpy**

Обработка данных, включая импорт и экспорт данных

**pandas**

**pytables**

• Анализ данных

**scipy**

**scikit-learn**

**opencv**

• Визуализация данных

**matplotlib**

**bokeh**

**seaborn**

• Многоцелевые

**sympy**

**cython**

Таблица 1.1. – Пакеты программ, применяемые для решения задач машинного обучения

|  |  |  |
| --- | --- | --- |
| **Библиотека** | **Назначение** | **Примечание** |
| **numpy** | Высокоэффективные матричные операции. | Предустановлен в Anaconda. |
| **pandas** | Импорт-экспорт из файлов разного формата, включая таблицы Excel и базы данных, агрегация данных в виде data frame, визуализация данных. | Предустановлен в Anaconda. |
| **scikit-learn** | Решение задач машинного обучения: классификация, регрессия, кластеризация, снижение размерности, настройка моделей машинного обучения. | Предустановлен в Anaconda |
| **matplotlib** | Визуализация двумерных изображений и графиков. | Предустановлен в Anaconda |
| **TensorFlow**  **(https://www.tensorflow.org/)** | Высокоэффективные Тензорные вычисления, в том числе с применением графических процессоров. | Требуется установка в простейшем случае pip install tensorflow. |
| **Keras (https://keras.io/)** | Высокоуровневый программный интерфейс для реализации моделей нейронных сетей, работающий как надстройка над TensorFlow, CNTK или Theano. | Требуется установка  pip install keras. В настоящее время является частью текущей версии TensorFlow. |
| **Trax** | Альтернатива связке  Keras- TensorFlow. | Текущая версия, вероятно, не работает в среде Windows. Пакет находится в стадии разработки. |

**1.3 Схема настройки системы машинного обучения**

Применение методов машинного обучения в задачах, для которых строгая математическая модель отсутствует, а имеются только экспертные оценки, часто бывает оптимальным способом решения.

Обучаемая система, в частности, искусственная нейронная сеть, способна воспроизвести закономерность, которую сложно или невозможно формализовать. В задачах «обучения с учителем» часто затруднительно определить качество экспертных оценок. К таким задачам относятся, в частности, и задачи выявления рисков заболеваний, оценки качества продуктов, распознавания речи, предсказания уровня котировок акций на финансовых рынках, распознавания литологических типов на урановых месторождениях по данным электрического каротажа. Несмотря на то, что эксперты задают перечень актуальных признаков объектов, диапазоны измеряемых физических величин могут перекрываться, а экспертные оценки могут быть противоречивыми или содержать ошибки.

Общая схема настройки методов машинного обучения на решаемую задачу приведена на рисунке 1.2.

В соответствии с этой схемой нам необходимо определить саму задачу, которая должна быть решена с помощью машинного обучения. Затем собрать данные, предобработать их, выбрать алгоритмы или методы, обучить или настроить методы, оценить результаты. В задачах обучения с учителем данные должны быть разделены на тренировочную (train), тестовую (test) и для некоторых задач проверочную (validation) части. Перечисленные этапы на самом деле части итеративного процесса, который инженер по данным повторяет с целью добиться наилучшего результата работы. Этот процесс не обязательно приводит к наилучшему результату, но его цель – добиться лучшего из возможных при тех данных, которые имеются в распоряжении исследователя.



Рисунок 1.2. – Циклический процесс настройки модели машинного обучения для решения задач

**1.4 Контрольные вопросы**

1. Какие группы библиотек машинного обучения Вы знаете?
2. В чем сущность циклического процесса настройки модели машинного обучения?
3. В каких случаях применяются (решаются) задачи кластеризации и снижения размерности?

**2.1 Занятие 1.**

**Запуск Python и проверка версий**

Рекомендуется убедиться, что среда Python была успешно установлена и работает в штатном состоянии. Сценарий ниже поможет вам проверить Вашу среду. Он импортирует каждую библиотеку, требуемую в этом разделе, и выводит ее версию.

Откройте командную строку и запустите Python:

python

Мы рекомендуем работать непосредственно в интерпретаторе или писать скрипты и запускать их в командной строке, нежели редакторах и IDEs. Это позволит сосредоточиться на машинном обучении, а не инструментарии программиста.

Введите или скопируйте и вставьте следующий скрипт в интерпретатор:

# Проверка версий библиотек

# Версия Python

import sys

print('Python: {}'.format(sys.version))

# Загрузка scipy

import scipy

print('scipy: {}'.format(scipy.\_\_version\_\_))

# Загрузка numpy

import numpy

print('numpy: {}'.format(numpy.\_\_version\_\_))

# Загрузка matplotlib

import matplotlib

print('matplotlib: {}'.format(matplotlib.\_\_version\_\_))

# Загрузка pandas

import pandas

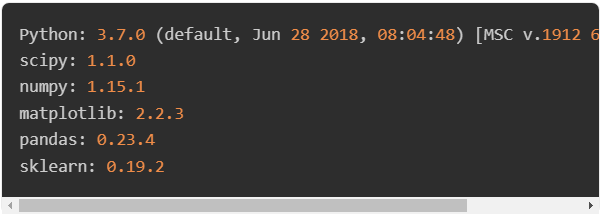
print('pandas: {}'.format(pandas.\_\_version\_\_))

# Загрукзка scikit-learn

import sklearn

print('sklearn: {}'.format(sklearn.\_\_version\_\_))

Вот пример вывода:

****

В идеале, версии должны соответствовать или быть более поздними. API библиотек не меняются быстро, так что не стоит переживать, если версии другие. Все в этом случае, скорее всего, все будет работать.

Если же Python выдает ошибку, рекомендуем обновить версионность системы. Если вы не можете запустить скрипт выше, вы не сможете пройти урок.

**Задание 1.**

Решим гипотетическую задачу нахождения параметров линейной регрессии методом градиентного спуска. Во-первых, подключим необходимые библиотеки:

% matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

import time

Отметим, что библиотека time позволит нам рассчитать время выполнения программы. Ее применение будет понятно из нижеследующего кода. Сформируем обучающее множество, состоящее из 30 примеров:

xr=np.matrix(np.linspace(0,10,30))

x=xr.T

#значения функции зададим в виде следующего выражения

y=np.power(x,2)+1

#Построим график (рисунок) командами

plt.figure(figsize=(9,9))

plt.plot(x,y,'.')

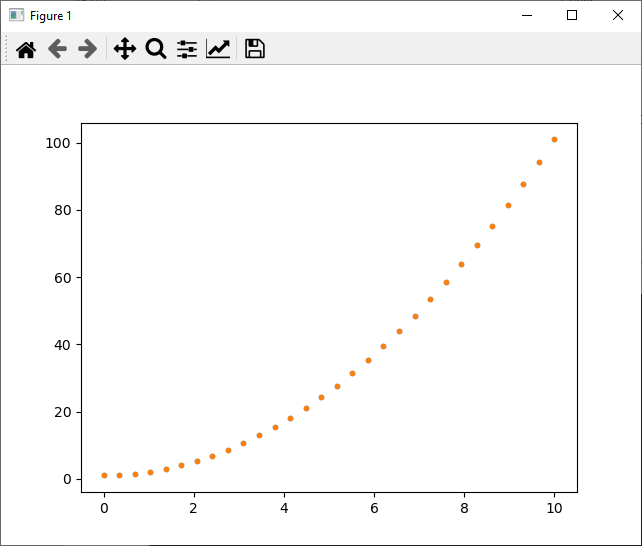


Рисунок 1.3 – График функции 𝑦 = 𝑥2 + 1

В нашем случае мы задали фиксированное множество примеров (m = 30), однако в дальнейшем мы можем его изменить. Для того, чтобы программа воспринимала любое множество примеров, определим его, используя метод size:

m=x.size

#сформируем первую колонку матрицы X, состоящую из единиц

on=np.ones([m,1])

#и сформируем матрицу X, объединив колонки

X=np.concatenate((on,x),axis=1)

Это матрица, в первой колонке которой стоят единицы, а во второй – значения 𝑥(1), 𝑥(2),. . . , 𝑥(𝑚). Затем зададим абсолютно произвольно начальные значения коэффициентов регрессии:

theta=np.matrix('0.1;1.3')

#и рассчитаем значения функции гипотезы

h=np.dot(X,theta)

#дополним предыдущий график регрессионной прямой

plt.plot(x,h)

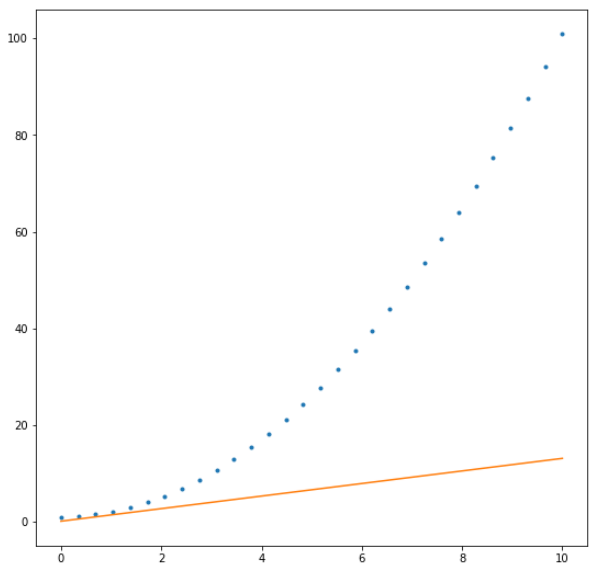


Рисунок 1.4. – Начальное положение прямой регрессии

На графике видно, что прямая функция гипотезы далека от идеальной. Применим алгоритм градиентного спуска для нахождения оптимальных значений параметров регрессионной прямой (функции гипотезы):

t0=time.time()

alpha=0.05

iterations=500

**for** i **in** **range**(iterations):

theta=theta-alpha\*(1/m)\*np.sum(np.multiply((h-y),x))

h=np.dot(X,theta)

t1=time.time()

#Построим графики

plt.figure(figsize=(9,9))

plt.plot(x,y,'.')

plt.plot(x,h,label='regressionByIteration')

leg=plt.legend(loc='upper right',shadow=True,fontsize='x-

small')

leg.get\_frame().set\_facecolor('#0055DD')

leg.get\_frame().set\_facecolor('#eeeeee')

leg.get\_frame().set\_alpha(0.5)

plt.show()

Получим следующий график регрессионной прямой:

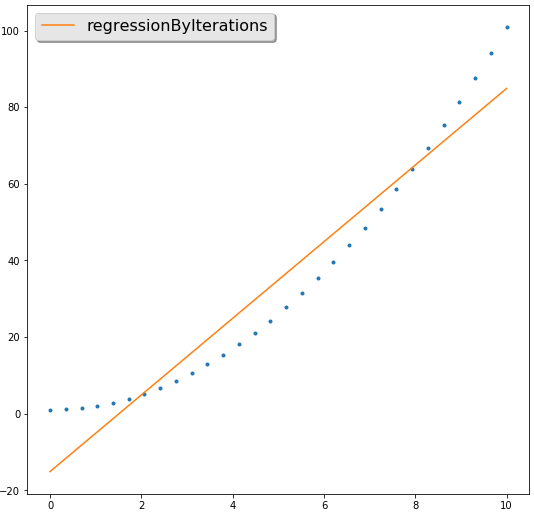


Рисунок 1.5. – Результат выполнения алгоритма градиентного спуска

#рассчитаем среднеквадратическую ошибку

mse=np.sum(np.power((h-y),2))/m

print('regressionByIteration mse= ', mse)

#и распечатаем длительность выполнения цикла градиентного спуска

print('regressionByIterations takes ',(t1-t0))

Получим примерно следующий вывод:

regressionByIterations mse = 63.270782365456206

regressionByIterations takes 0.027503490447998047

В качестве небольшого дополнения рассчитаем показатели точности регрессии с применением библиотеки метрик sklearn.

from sklearn.metrics import mean\_squared\_error, r2\_score

y\_predict = h

y\_test=y

print("Mean squared error: {:.2f}".format(mean\_squared\_error(y\_test,y\_predict)))

print("r2\_score: {:.2f}".format(r2\_score(y\_test, y\_predict)))

Получим следующие результаты:

Mean squared error: 63.27

r2\_score: 0.93

**Задание 2.**

Мы будем использовать датасет цветов ирисов Фишера. Этот датасет известен тем, что он используется практически всеми в качестве "hello world" примера в машинном обучении и статистике.

Набор данных содержит 150 наблюдений за цветами ириса. В датасете есть четыре колонки измерений цветов в сантиметрах. Пятая колонна является видом наблюдаемого цветка.

Все наблюдаемые цветы принадлежат к одному из трех видов. Узнать больше об этом датасете можно в Википедия. <https://ru.wikipedia.org/wiki/Ирисы_Фишера?ref=codecamp.ru>

На этом этапе мы загрузим данные из URL-адреса в CSV файл.

**2.1 Импорт библиотек**

Во-первых, давайте импортировать все модули, функции и объекты, которые мы планируем использовать в этом уроке.

# Загрузка библиотек

from pandas import read\_csv

from pandas.plotting import scatter\_matrix

from matplotlib import pyplot

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

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import StratifiedKFold

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

Все должно загружаться без ошибок. Если у вас есть ошибка, остановитесь. Перед продолжением необходима рабочая среда SciPy. Посмотрите совет выше о настройке вашей среды.

**2.2 Загрузка датасета**

Мы можем загрузить данные непосредственно из репозитория машинного обучения UCI.

Мы используем модуль pandas для загрузки данных. Мы также будем использовать pandas чтобы исследовать данные как целей описательной статистики, так для визуализации данных.

Обратите внимание, что при загрузке данных мы указываем имена каждого столбца. Это поможет позже, когда мы будем исследовать данные.

# Загрузка датасета

url= "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

dataset = read\_csv(url, names=names)

Датасет должен загрузиться без происшествий.

Если у вас есть проблемы с сетью, вы можете скачать файл iris.csv в рабочую директорию и загрузить его с помощью того же метода, изменив URL на локальное имя файл

**2.3 Анализ датасета**

Теперь пришло время взглянуть на данные более детально. На этом этапе мы погрузимся в анализ данные несколькими способами:

* Размерность датасета
* Просмотр среза данных
* Статистическая сводка атрибутов
* Разбивка данных по атрибуту класса.

Не волнуйтесь, каждый взгляд на данные является одной командой. Это полезные команды, которые можно использовать снова и снова в будущих проектах.

**2.4 Размерность датасета**

Мы можем получить быстрое представление о том, сколько экземпляров (строк) и сколько атрибутов (столбцов) содержится в датасете с помощью метода shape.

# shape

print(dataset.shape)

Вы должны увидеть 150 экземпляров и 5 атрибутов:

(150, 5)

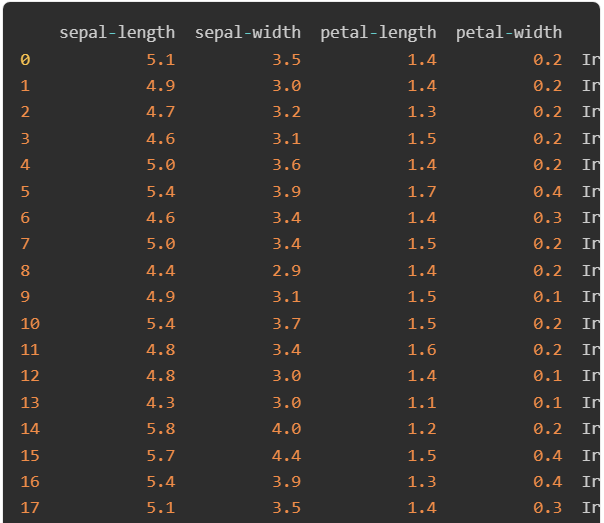
**2.5 Просмотр среза данных**

Исследовании данных, стоит сразу в них заглянуть, для этого есть метод head()

# Срез данных head

print(dataset.head(20))

Это должно вывести первые 20 строк датасета.



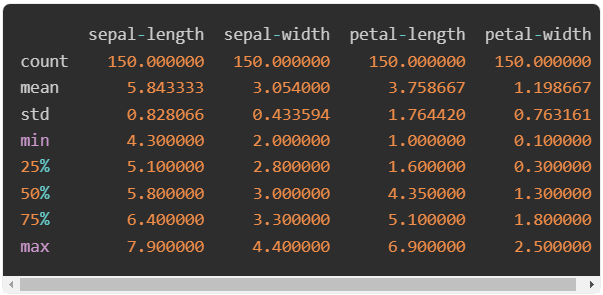
**2.6 Статистическая сводка**

Давайте взглянем теперь на статистическое резюме каждого атрибута. Статистическая сводка включает в себя количество экземпляров, их среднее, мин и макс значения, а также некоторые другие данные.

# Стастические сводка методом describe

print(dataset.describe())

Мы видим, что все численные значения имеют одинаковую шкалу (сантиметры) и аналогичные диапазоны от 0 до 8 сантиметров.



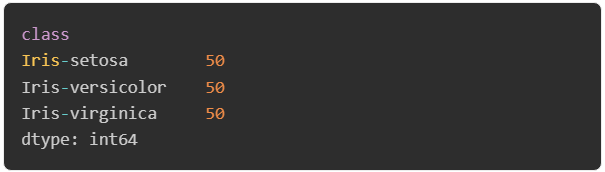
**2.7 Распределение классов**

Давайте теперь рассмотрим количество экземпляров (строк), которые принадлежат к каждому классу. Мы можем рассматривать это как абсолютный счет.

# Распределение по атрибуту class

print(dataset.groupby('class').size())

Мы видим, что каждый класс имеет одинаковое количество экземпляров (50 или 33% от датасета).



**2.8 Резюме загрузки датасета**

На будущее мы можем объединить все предыдущие элементы вместе в один скрипт.

# Загрузка библиотек

from pandas import read\_csv

# Загрузка датасета

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

dataset = read\_csv(url, names=names)

# shape

print(dataset.shape)

# Срез данных head

print(dataset.head(20))

# Стастические сводка методом describe

print(dataset.describe())

# Распределение по атрибуту class

print(dataset.groupby('class').size())

**2.8 Визуализация данных**

Теперь когда у нас есть базовое представление о данных, давайте расширим его с помощью визуализаций.

Мы рассмотрим два типа графиков:

* Одномерные (Univariate) графики, чтобы лучше понять каждый атрибут.
* Многомерные (Multivariate) графики, чтобы лучше понять взаимосвязь между атрибутами.

**2.9 Одномерные графики**

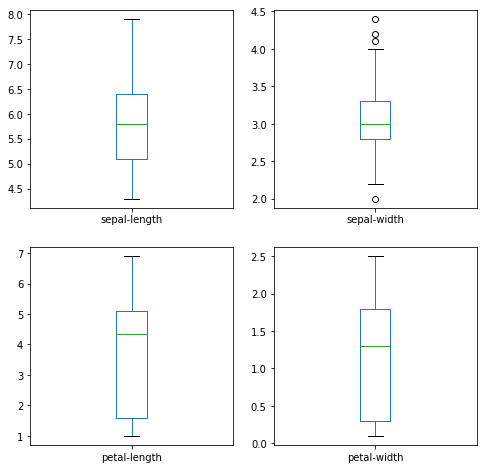
Начнем с некоторых одномерных графиков, то есть графики каждой отдельной переменной. Учитывая, что входные переменные являются числовыми, мы можем создавать диаграмма размаха (или "ящик с усами", по-английски "box and whiskers diagram") каждого из них.

# Диаграмма размаха

dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)

pyplot.show()

Это дает нам более четкое представление о распределении атрибутов на входе.



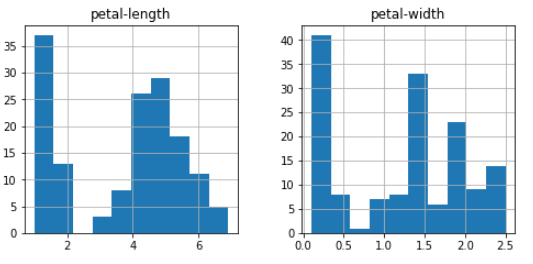
Мы также можем создать гистограмму входных данных каждой переменной, чтобы получить представление о распределении.

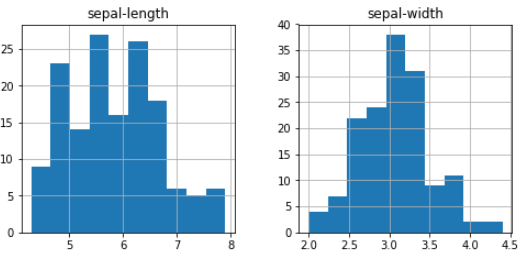
# Гистограмма распределения атрибутов датасета

dataset.hist()

pyplot.show()

Из графиков видно, что две из входных переменных имеют около гауссово (нормальное) распределение. Это полезно отметить, поскольку мы можем использовать алгоритмы, которые могут использовать это предположение.





**2.10 Многомерные графики**

Теперь мы можем посмотреть на взаимодействия между переменными.

Во-первых, давайте посмотрим на диаграммы рассеяния всех пар атрибутов. Это может быть полезно для выявления структурированных взаимосвязей между входными переменными.

#Матрица диаграмм рассеяния

scatter\_matrix(dataset)

pyplot.show()

Обратите внимание на диагональ некоторых пар атрибутов. Это говорит о высокой корреляции и предсказуемой взаимосвязи.

**2.11 Резюме визуализации данных**

Для справки мы можем связать все предыдущие элементы вместе в один скрипт. Полный пример приведен ниже.

# Загрузка библиотек

from pandas import read\_csv

# Загрузка датасета

url= "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

dataset = read\_csv(url, names=names)

# Диаграмма размаха

dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)

pyplot.show()

# Гистограмма распределения атрибутов датасета

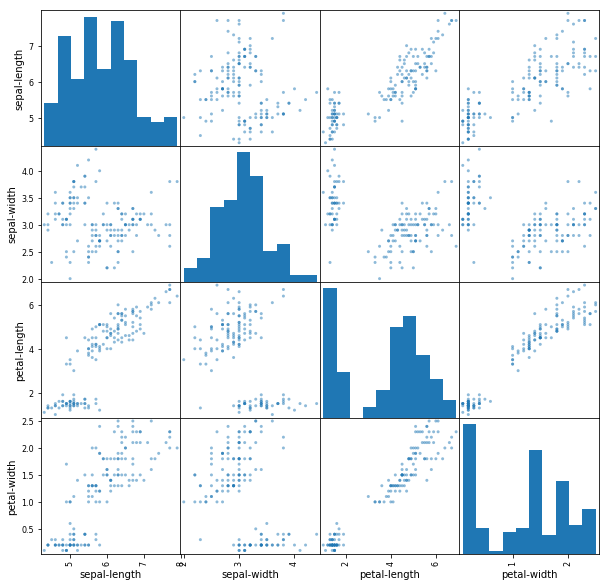
dataset.hist()

pyplot.show()

#Матрица диаграмм рассеяния

scatter\_matrix(dataset)

pyplot.show()



**Задание 3.**

**Диаграмма рассеивания Matplotlib. Примеры**.

Matplotlib – одна из наиболее широко используемых библиотек визуализации данных в Python. От простых до сложных визуализаций - это библиотека для большинства.

В этом уроке мы рассмотрим, как построить график рассеивания в Matplotlib.

***Импортировать данные***. Мы будем использовать набор данных [Ames Housing](https://www.kaggle.com/prevek18/ames-housing-dataset) и визуализировать корреляции между объектами из него. Давайте импортируем Pandas и загрузим набор данных:

import pandas as pd

df = pd.read\_csv('AmesHousing.csv')

Постройте диаграмму рассеивания в Matplotlib.

Теперь, когда набор данных загружен, давайте импортируем Matplotlib, определимся с функциями, которые мы хотим визуализировать, и построим диаграмму рассеивания:

import matplotlib.pyplot as plt

import pandas as pd

df = pd.read\_csv('AmesHousing.csv')

fig, ax = plt.subplots(figsize=(10, 6))

ax.scatter(x = df['Gr Liv Area'], y = df['SalePrice'])

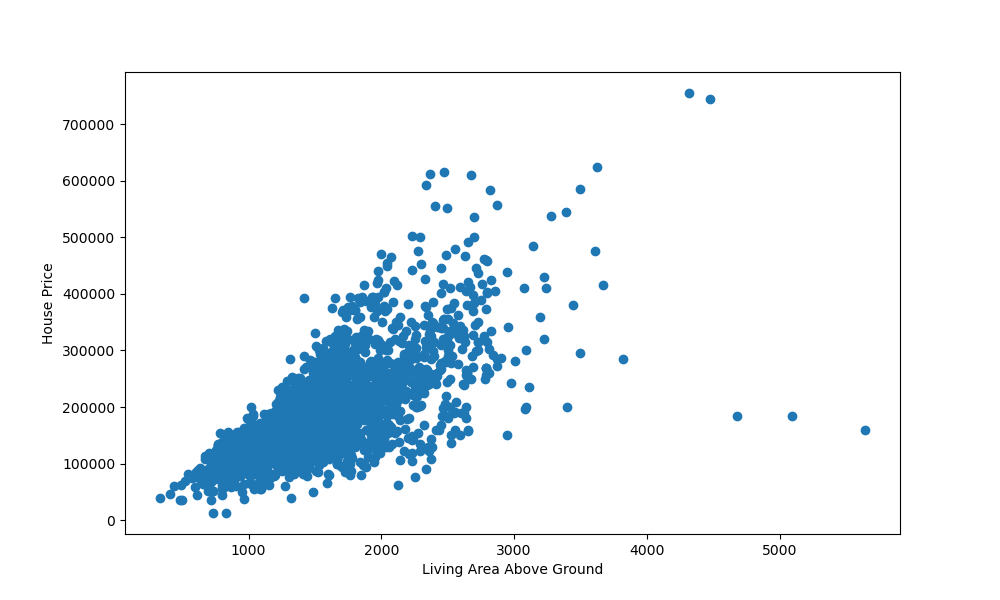
plt.xlabel("Living Area Above Ground")

plt.ylabel("House Price")

plt.show()

Здесь мы создали график, используя экземпляр PyPlot, и установили размер фигуры. Используя возвращенный объект Axes, который возвращается из функции subplots(), мы вызвали функцию scatter().

Мы должны поставить аргументы x и y, которые мы хотели бы использовать, чтобы заполнить участок. Выполнение этого кода приводит к:



Мы также установили метки x и y, чтобы указать, что представляют собой переменные. Между этими двумя переменными существует явная положительная корреляция. Чем больше площадь над землей, тем выше была цена дома.

Есть несколько отклонений, но подавляющее большинство следует этой гипотезе.

***Построение графиков множественного разброса в Matplotlib***

Если вы хотите сравнить более одной переменной с другой, например - проверьте корреляцию между общим качеством дома и продажной ценой, а также площадью над уровнем земли – нет необходимости создавать трехмерный график для этого.

Хотя существуют 2D-графики, которые визуализируют корреляции между более чем двумя переменными, некоторые из них не совсем подходят для начинающих.

Самый простой способ сделать это – построить два участка: на одном мы построим график площади над уровнем земли в зависимости от продажной цены, а на другом – общее качество в зависимости от продажной цены.

Давайте посмотрим, как это сделать:

import matplotlib.pyplot as plt

import pandas as pd

df = pd.read\_csv('AmesHousing.csv')

fig, ax = plt.subplots(2, figsize=(10, 6))

ax[0].scatter(x = df['Gr Liv Area'], y = df['SalePrice'])

ax[0].set\_xlabel("Living Area Above Ground")

ax[0].set\_ylabel("House Price")

ax[1].scatter(x = df['Overall Qual'], y = df['SalePrice'])

ax[1].set\_xlabel("Overall Quality")

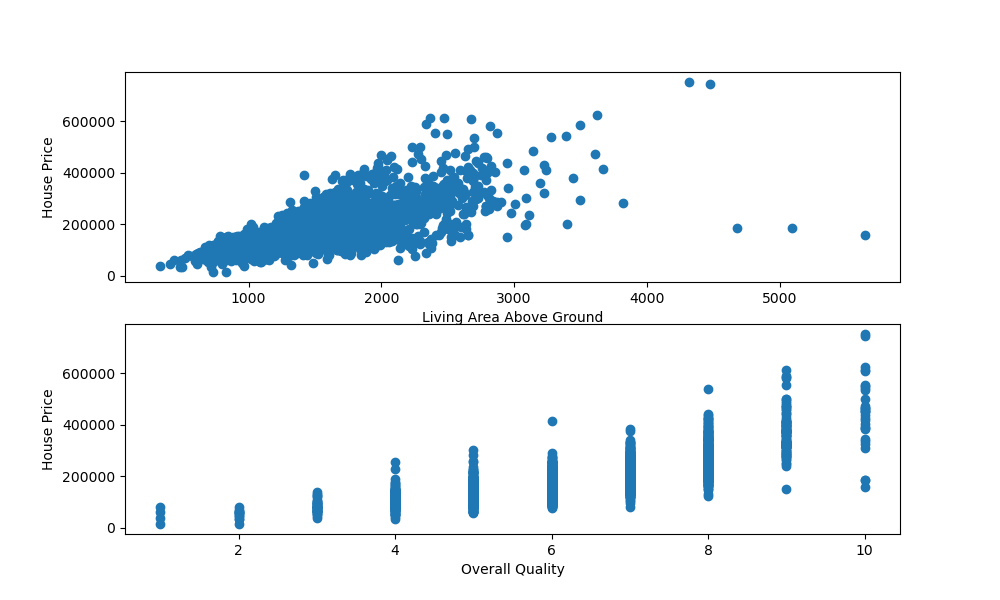
ax[1].set\_ylabel("House Price")

plt.show()

Здесь мы вызвали plt.subplots(), с параметром 2, чтобы указать, что мы хотели бы создать экземпляры двух подзаголовков на рисунке.

Мы можем получить к ним доступ через экземпляр Axes - ax. ax[0] относится к осям первого подзаголовка, а ax[1] относится к осям второго подзаголовка.

Здесь мы вызвали функцию scatter() для каждого из них, снабдив их метками. Выполнение этого кода приводит к:



***Построение трехмерной диаграммы рассеяния в Matplotlib***

Если вы не хотите визуализировать это в двух отдельных подзаголовках, вы можете построить корреляцию между этими переменными в 3D. Matplotlib имеет встроенную функцию трехмерного построения графиков.

Во-первых, нам нужно импортировать класс Axes3D из mpl\_toolkits.mplot3d. Этот специальный тип необходим для 3D-визуализации. С его помощью мы можем передать другой аргумент z - это третья функция, которую мы хотели бы визуализировать.

Давайте продолжим и импортируем объект Axes3D и построим диаграмму рассеяния для трех предыдущих функций:

import matplotlib.pyplot as plt

import pandas as pd

from mpl\_toolkits.mplot3d import Axes3D

df = pd.read\_csv('AmesHousing.csv')

fig = plt.figure()

ax = fig.add\_subplot(111, projection = '3d')

x = df['SalePrice']

y = df['Gr Liv Area']

z = df['Overall Qual']

ax.scatter(x, y, z)

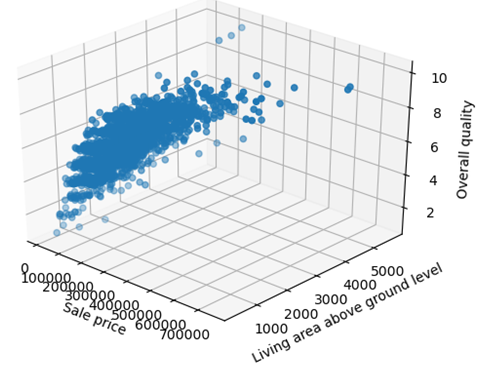
ax.set\_xlabel("Sale price")

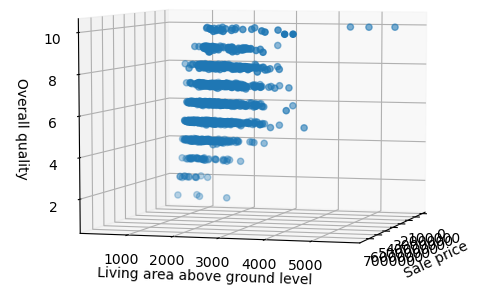
ax.set\_ylabel("Living area above ground level")

ax.set\_zlabel("Overall quality")

plt.show()

Запуск этого кода приводит к интерактивной трехмерной визуализации, которую мы можем панорамировать и исследовать в трехмерном пространстве:

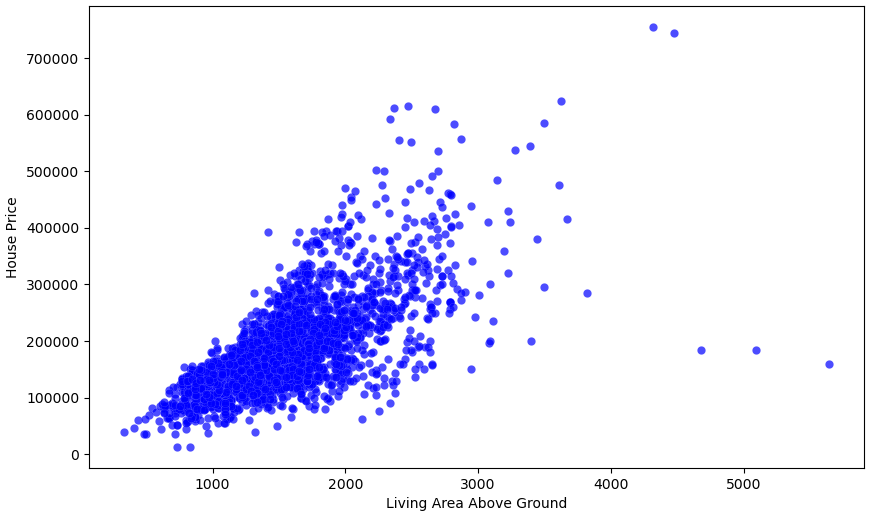




***Настройка точечной диаграммы в Matplotlib***

Вы можете изменить внешний вид графика, снабдив функцию scatter() дополнительными аргументами, такими как color, alpha и т.д.:

ax.scatter(x = df['Gr Liv Area'], y = df['SalePrice'], color = "blue", edgecolors = "white", linewidths = 0.1, alpha = 0.7)



Вам не обязательно сразу все понимать. Ваша цель состоит в том, чтобы запустить ряд скриптов описанных в уроке и получить конечный результат. Вам не нужно понимать все при первом проходе. Записывайте свои вопросы параллельно с тем как пишите код. Рекомендуем использовать справку ("FunctionName") в Python чтобы разобраться глубже во всех функциях, которые вы используете.

Вам не нужно знать, как работают алгоритмы. Важно знать об ограничениях и о том, как настроить алгоритмы машинного обучения. Более подробное узнать о конкретных алгоритмах можно и позже. Вы должны постепенно накапливать знания о работе алгоритмы. Сегодня, начните с того что поймете как его использовать в Python.

Вам не нужно быть программистом Python. Синтаксис языка Python может быть интуитивно понятным, даже если вы новичок в нем. Как и на других языках, сосредоточьтесь на вызовах функций (например,function()) и назначениях (например, a = "b"). Если вы являетесь разработчиком, вы итак уже знаете, как подобрать основы языка очень быстро.

Вам не нужно быть экспертом по машинного обучению. Вы можете узнать о преимуществах и ограничениях различных алгоритмов гораздо позже, и есть много информации в интернете, о более глубинных тонкостях алгоритмов и этапах проекта машинного обучения и важности оценки точности с помощью перекрестной валидации.

**Итого**

Вы сделали свой первый мини-проект по машинному обучению в Python.

Вы наверняка обнаружили, что после завершения даже небольшого проекта от загрузки данных до прогнозирования — вы уже намного сильнее продвинулись.

Какие могут быть следующие шаги по изучению машинного обучения?

Мы не освещали все этапы проекта машинного обучения, потому что это ваш первый проект, и нам нужно сосредоточиться на ключевых этапах. А именно, загрузке данных, анализе данных, оценка некоторых алгоритмов и прогнозировании данных. В других уроках мы рассмотрим другие аспекты машинного обучения по подготовке данных и улучшению результатов.