3.1 Введение в Машинное обучение

Какую задачу решает ML?

Требуется подогнать заданный набор точке данных под соответствующую функцию (отображение входа на выход), которая улавливает важные сигналы в данных и игнорирует помехи, а затем убедиться, что на новых данных функция работает хорошо.

Практические задачи

Задача № 1

Есть некоторый человек, который живет рядом с магазином пятерочка в шаговой доступности и ходит покупать продукты. Из этого магазина у нас есть статистика по его посещениям:

№ недели	Кол-во посещений
1	5
2	6
3	4
4	5

Основываясь на представленных данных, сделайте прогноз по количеству посещений на 5-ю неделю.

Ответ:

$$\bar{x} = \frac{5 + 6 + 4 + 5}{4} = \frac{20}{4} = 5$$

Обоснование:
Так как три разных простых метода (среднее, мода и медиана) указывают на одно и то же число, **5** является самым сильным и легко защищаемым прогнозом. Он предполагает, что поведение клиента стабильно, а отклонения в 4 и 6 посещений являются случайными.

В лекции упоминается равномерное распределение желания кушать каждую неделю

Задача № 2

Есть в районе два дома и в каждой из них есть своя дворовая футбольная команда. И вот эти команды каждую неделю проводят матчи, статистика следующая:

№ недели	Счет	Кол-во забитых мячей
1	3:2	5
2	3:3	6
3	1:3	4
4	3:2	5

Основываясь на представленных данных, сделайте прогноз по количеству забитых мячей на 5-ю неделю.

Обоснование:

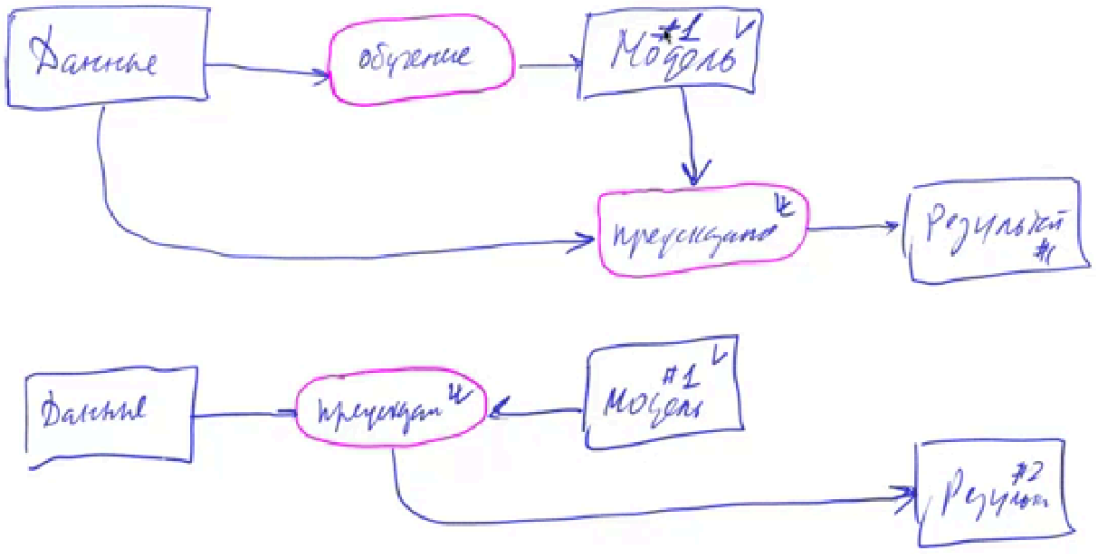
В лекции упоминается распределение Пуассона.

Как работает машинное обучение ?

Исходной точкой служит набор данных, который подается на вход в процесс обучения. В результате этого процесса создается модель, предназначенная для решения конкретной задачи. В нашем случае, модель вычисляет среднее арифметическое по данным за все предыдущие недели.

Далее необходимо проверить работоспособность этой модели. На вход процесса, который назовем «предсказание», подается сама модель и данные для проверки. Если полученный результат нас устраивает, мы делаем вывод, что модель является адекватной и готовой к дальнейшему использованию.

После успешной проверки мы применяем эту же модель к новым данным в рамках того же процесса «предсказания», чтобы получить уже конечный, полезный результат.



Проблемы машинного обучения

1. Источник данных
2. Полученные после применения модели результат

Основные типы разделения данных

1. Обучение с учителем (supervised learning)

Это парадигма машинного обучения, при которой алгоритм учится на размеченном наборе данных, где для каждого примера известен правильный ответ (метка). Цель — выявить закономерность «входные признаки → выходная метка», чтобы впоследствии применять её к новым данным.

В зависимости от типа предсказываемой метки, выделяют две основные задачи:

- **Задачи классификации** (метки - дискретные: два или более) Присвоение объекту одной из нескольких заранее известных, дискретных категорий (классов).
- **Задачи регрессии** (метки / результат; непрерывные величины) Предсказание непрерывного числового значения.

2. Обучение без учителя (unsupervised learning)

Это подход в машинном обучении, при котором модель работает с **неразмеченными данными**, то есть без заранее известных правильных ответов. Задача алгоритма — самостоятельно найти в массиве информации внутреннюю структуру и скрытые взаимосвязи.

- Задача кластеризации (выделяет отельные группы данных)
- Задача понижения размерности (поиск более сжатого представления данных)

3. Методы частичного обучения (semi-supervised learning)

Не все данные промаркированы.

4. Методы обучения с подкреплением (reinforcement learning)

Система обучения улучшает свои характеристики на основе взаимодействия (обратной связи) со средой. При этом взаимодействии система получает сигналы (функции наград), которые несут в себе информацию насколько хорошо / плохо система решила задачу (с точки зрения среды). Итоговые награды не станет максимальной.

Использование DataSet

```
import seaborn as sns

iris = sns.load_dataset("iris")

print(iris.head())

#    sepal_length  sepal_width  petal_length  petal_width species
# 0           5.1          3.5           1.4          0.2  setosa
# 1           4.9          3.0           1.4          0.2  setosa
# 2           4.7          3.2           1.3          0.2  setosa
# 3           4.6          3.1           1.5          0.2  setosa
# 4           5.0          3.6           1.4          0.2  setosa

print(type(iris))

# <class 'pandas.core.frame.DataFrame'>
```

Терминология

- строки - отдельные объекты, образцы (sample)
- столбы - признаки (feature), соответствуют конкретным наблюдениям
- матрица признаков (features matrix) размер [число образцов x число признаков]
- целевой массив, массив меток (targets) - одномерный массив [1 x число образцов] - данные, которые мы хотим предсказать на основе имеющихся данных
- зависимые (метка) и независимые переменные (признаки)

Как выглядит процесс построения машинного обучения ?

1. предварительная обработка

- На вход поступают необработанные данные
- Выбор признаков, масштабирование признаков
- Понижение размерности
- Выборка образцов
- На выход поступает набор данных (обучающий и тестовый набор)

2. Обучение

- Выбор модели
- Перекрестная проверка
- Метрики эффективности
- Оптимизация гиперпараметров - параметры, которые получаются не из данных, а являются настраиваемыми характеристиками модели

3. Оценка и формирование финальной модели

4. Прогнозирование (использование модели)

Scikit-learn — это одна из самых популярных и мощных библиотек с открытым исходным кодом для машинного обучения на языке Python. Она предоставляет простой, эффективный и единообразный набор инструментов для решения задач анализа данных и моделирования.

Алгоритм программы

1.

- pridict() (с учителем)
- predict() или transform() (без учителя)

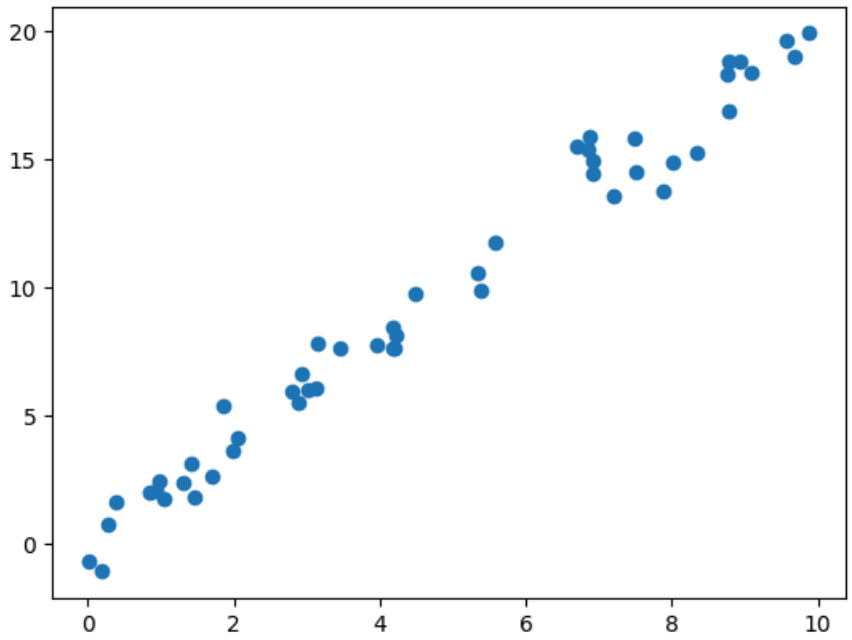
Обучение с учителем: линейная регрессия

Простая линейная регрессия

$$y = ax + b$$

1 этап

```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(1)

x = 10 * np.random.rand(50)
y = 2 * x + np.random.randn(50)

plt.scatter(x, y)
plt.show()
```



2 этап

1. Выбираем класс модели

   ```
   from sklearn.linear_model import LinearRegression
   ```

2. Устанавливаем гиперпараметры модели

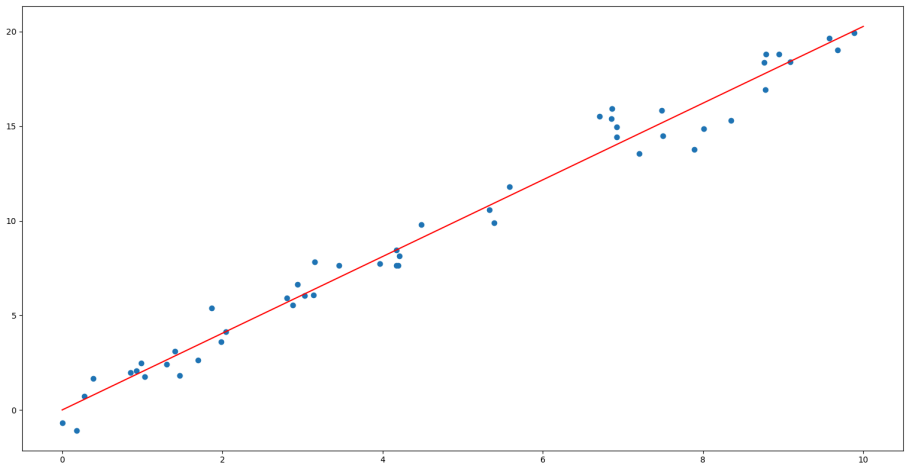
   ```
   model = LinearRegression()
   ```

3. Создаем матрицу признаков и целевой массив

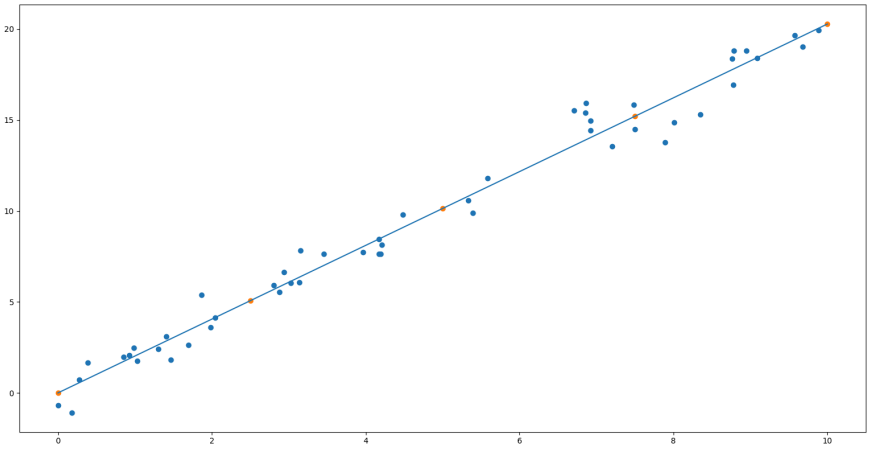
```
print(type(x))
print(type(y))

# <class 'numpy.ndarray'>
# <class 'numpy.ndarray'>

print(x.shape)
print(y.shape)

# (50,)
# (50,)

X = x
```

4. Обучение модели fit()

```
X = x[:, np.newaxis] # добавление еще одного измерения
model.fit(x, y)

print(model.coef_[0])
print(model.intercept_)

# 2.0272088103606944
# 0.001422914446798984
```



5. Применить модель к новым данным



Итоговый полный код

```
# ==================================
# ШАГ 1: ИМПОРТ БИБЛИОТЕК
# ==================================
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression

# ==================================
# ШАГ 2: ГЕНЕРАЦИЯ ДАННЫХ
# ==================================
# Устанавливаем seed, чтобы случайные числа были одинаковыми при каждом запуске
np.random.seed(1)

# Создаем 50 случайных точек для оси X
x = 10 * np.random.rand(50)
# Создаем Y на основе линейной зависимости (y=2x) с добавлением случайного "шума"
y = 2 * x + np.random.randn(50)

# ==================================
# ШАГ 3: ВИЗУАЛИЗАЦИЯ ИСХОДНЫХ ДАННЫХ
# ==================================
# Рисуем диаграмму рассеяния, чтобы посмотреть на наши сгенерированные точки
plt.scatter(x, y)

# ==================================
# ШАГ 4: СОЗДАНИЕ И ОБУЧЕНИЕ МОДЕЛИ
# ==================================
```

```
# Создаем модель линейной регрессии.
# fit_intercept=False означает, что линия регрессии будет проходить через начало координат (0,(
model = LinearRegression(fit_intercept=False)

# Выводим размерности массивов для проверки
print(x.shape)
print(y.shape)

# Scikit-learn требует, чтобы входные данные X были двумерным массивом.
# Преобразуем наш одномерный массив x в двумерный (50,) → (50, 1)
X = x[:, np.newaxis]

# Обучаем модель на наших подготовленных данных
model.fit(X, y)

# ==================================
# ШАГ 5: АНАЛИЗ ПАРАМЕТРОВ МОДЕЛИ
# ==================================
# Выводим коэффициент наклона (a в y=ax+b), найденный моделью.
# Он должен быть близок к 2.0
print(model.coef_[0])
# Выводим точку пересечения с осью Y (b в y=ax+b).
# Будет равен 0, так как мы установили fit_intercept=False.
print(model.intercept_)

# ============================================================================
# ШАГ 6: ПОСТРОЕНИЕ ЛИНИИ РЕГРЕССИИ
# ============================================================================

# --- Способ 1: Ручной расчет линии на основе коэффициентов ---
# Создаем 30 точек на оси X для построения гладкой линии
x_ = np.linspace(0, 10, 30)
# Рассчитываем значения Y для этой линии вручную, используя найденные моделью параметры
y_ = model.coef_[0] * x_ + model.intercept_
# Рисуем полученную линию на графике
plt.plot(x_, y_,)

# --- Способ 2: Использование метода model.predict() ---
# Создаем еще 5 точек на оси X
xfit = np.linspace(0, 10, 5)
# Используем встроенный метод .predict() для получения предсказаний для этих точек
yfit = model.predict(xfit[:, np.newaxis])
# Рисуем эти 5 предсказанных точек. Они должны лечь точно на линию,
# построенную в первом способе.
plt.scatter(xfit, yfit)
```

```
plt.show()
```