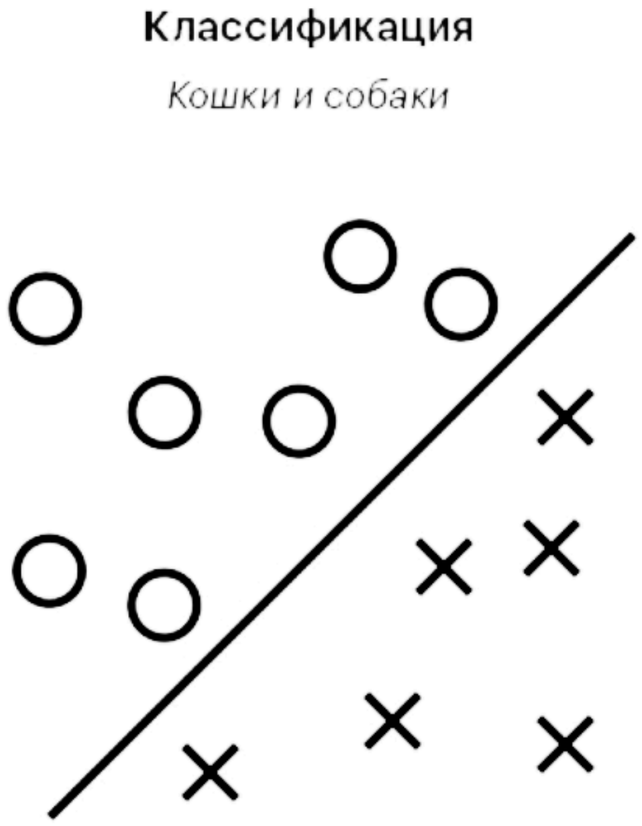
Лекция 2. Dataset. Pandas

Классификация



В задаче классификации целевая переменная принимает ограниченный набор значений (классов):

- Бинарная классификация — всего два класса (например, «выжил» или «не выжил» на Титанике, «спам» или «не спам»).
- Мультиклассовая классификация — три и более класса (например, типы новостей: спорт, политика, экономика).
- Мульти-лейбл классификация — каждый пример может одновременно принадлежать нескольким классам (например, теги статей).

Классификация

У нас есть набор обучающих данных:

$$\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$$

где:

$x^{(i)}$ — вектор признаков для примера i

$y^{(i)} \in \{0, 1\}$ — метка класса

Наша цель — построить функцию f(x), которая будет предсказывать класс для нового примера x.

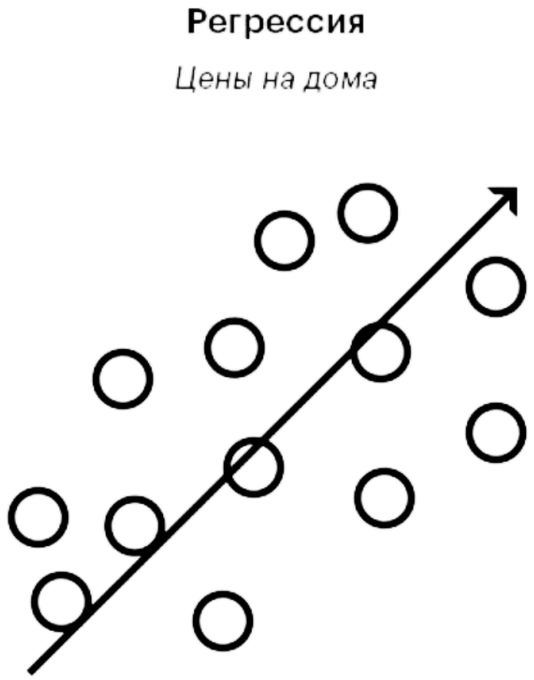
Классификация

- Логистическая регрессия — простой и интерпретируемый метод, хорошо подходит для линейно разделимых данных.
- Дерево решений — строит дерево с правилами «если-то».
- Случайный лес (Random Forest) — ансамбль деревьев, более устойчивая модель.
- Метод опорных векторов (SVM) — хорош для сложных границ между классами.
- К ближайших соседей (KNN) — прогнозирует класс по наиболее близким примерам.
- Нейронные сети — мощный инструмент для нелинейных зависимостей.

Регрессия

Если классификация предсказывает класс (категорию), то задача регрессии заключается в прогнозе непрерывного числового значения на основе входных признаков.

Регрессия — это задача предсказания вещественной (непрерывной) целевой переменной y по вектору признаков x: $y = f(x) + \varepsilon$



Линейная регрессия

Один из самых простых и популярных методов регрессии — линейная регрессия. Она предполагает, что зависимость между признаками и целевой переменной линейна:

$$\hat{y} = w^T x + b$$

где:
w — вектор коэффициентов (весов),
b — свободный член (intercept).

Пример: прогноз цены квартиры:

$$\text{Price} = w_1 \times \text{Area} + w_2 \times \text{Number of Rooms} + b$$

Интерпретация коэффициентов

Коэффициенты www показывают, насколько сильно каждый признак влияет на результат:

$w_i > 0$: увеличение признака i увеличивает прогноз.

$w_i < 0$: увеличение признака уменьшает прогноз.

Чем больше по модулю, тем сильнее влияние признака.

Это делает модель интерпретируемой — мы можем объяснить, как она принимает решения.

Функция потерь

Чтобы «научить» модель находить лучшие коэффициенты, мы минимизируем функцию потерь. Для линейной регрессии обычно используют среднеквадратичную ошибку (MSE):

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

Обучение модели

Задача обучения — найти такие w и b, которые минимизируют MSE. Обычно это делается методами оптимизации:

- Аналитически (через нормальное уравнение) для простых случаев.
- Градиентный спуск для больших датасетов.

Обновление весов по градиентному спуску:

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

$$b \leftarrow b - \eta \frac{\partial L}{\partial b}$$

Кластеризация

Кластеризация — это задача группировки объектов в такие подмножества (кластеры), внутри которых объекты похожи друг на друга и отличаются от объектов других кластеров.

Формально, имея множество объектов $X=\{x(1),x(2),\ldots,x(N)\}$, мы хотим разбить его на K подмножеств $C1,C2,\ldots,CK$ так, чтобы:

- внутри каждого кластера объекты были как можно более похожими,
- между кластерами — как можно более различными.

Важно понимать фундаментальное отличие!

- Классификация — задача обучения на размеченных данных. У нас есть метки классов, и модель учится их предсказывать.
- Кластеризация — задача на неразмеченных данных. У нас нет правильных ответов. Мы ищем структуру сами.

K-средних (K-Means)

Один из самых простых и широко используемых алгоритмов.

- Пользователь задаёт число кластеров K.
- Алгоритм случайно инициализирует центры кластеров.

На каждом шаге:

- Назначает каждый объект ближайшему центру.
- Пересчитывает центры как среднее объектов в кластере.
- Повторяет, пока центры не перестанут меняться.

Отличие от классификации и регрессии

- Регрессия: есть непрерывная целевая переменная.
- Классификация: есть известные классы (размеченные данные).
- Кластеризация: нет разметки — мы пытаемся «открыть» структуру.

Scikit-learn

1. Подготовка данных

Scikit-learn работает с данными в виде таблиц признаков (X) и целевой переменной (y):

- X — обычно NumPy-массив или Pandas DataFrame размером (n_samples, n_features)
- y — одномерный массив длиной n_samples

Age	Fare	Pclass	...
22	7.25	3	...
38	71.3	1	a

Scikit-learn

2. Разделение данных

Чтобы проверить качество модели, данные делят на обучающую и тестовую выборки:

```
1 from sklearn.model_selection import train_test_split
2
3 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

test_size=0.2 означает, что 20% данных пойдут на тест.

Scikit-learn

3. Создание модели

```
1 from sklearn.linear_model import LogisticRegression
2
3 model = LogisticRegression()
```

Аналогично можно выбрать:

- DecisionTreeClassifier
- RandomForestRegressor
- KMeans
- и т. д.

Scikit-learn

4. Обучение модели

Чтобы «подогнать» модель к данным:

```
1 model.fit(X_train, y_train)
```

После этого модель «выучила» зависимости.

Scikit-learn

5. Предсказания

На новых (или тестовых) данных:

```
1 y_pred = model.predict(X_test)
```

Некоторые модели также могут выдавать вероятности:

```
1 y_proba = model.predict_proba(X_test)
```

Scikit-learn

6. Оценка качества

Для классификации:

```python
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Для регрессии:

```python
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y_test, y_pred)
print("MSE:", mse)
```

Для кластеризации:

```python
from sklearn.metrics import silhouette_score

score = silhouette_score(X, labels)
print("Silhouette:", score)
```

!pip install pandas scikit-learn	Установка необходимых библиотек
pd.read_csv('train.csv')	Загрузка обучающих данных из CSV
df.shape	Размерность таблицы (строки, столбцы)
df.head()	Просмотр первых строк таблицы
df.isnull().sum()	Подсчёт пропущенных значений в таблице
fillna(value)	Заполнение пропущенных значений
mode()[0]	Нахождение моды (самого частого значения)
concat([df1, df2])	Объединение таблиц
LabelEncoder().fit_transform()	Преобразование категориальных признаков в числовые
drop(columns)	Удаление ненужных признаков (например, Name, Ticket)
train_test_split(X, y, test_size=0.2)	Разделение выборки на обучение и валидацию
RandomForestClassifier()	Инициализация модели случайного леса
model.fit(X_train, y_train)	Обучение модели
model.predict(X_test)	Предсказания на новых данных
accuracy_score(y_true, y_pred)	Подсчёт точности (accuracy) модели
submission.to_csv('submission.csv')	Сохранение предсказаний в CSV-файл для отправки на
plt.figure(figsize=(6,4))	Создание графика с указанным размером
sns.countplot(x='Sex', hue='Survived', ...)	Столбчатая диаграмма по признаку
sns.histplot(..., hue='Survived', multiple='stack')	Гистограмма с группировкой по выжившим
sns.heatmap(df.corr(), annot=True, cmap=...)	Матрица корреляций между числовыми признаками
select_dtypes(include=['number'])	Выбор только числовых признаков из таблицы

1. Загрузите данные train.csv и test.csv в Pandas.

2. Посмотрите таблицы: сколько в них строк и столбцов? Какие есть признаки?

3. Найдите пропущенные значения и заполните их:

Age

Embarked

Fare

4. Закодируйте категориальные признаки с помощью LabelEncoder.

5. Удалите ненужные для начала колонки.

6. Разделите данные на тренировочные и валидационные через train_test_split.

7. Обучите модель RandomForestClassifier.

8. Посчитайте точность (accuracy) на валидационной выборке.

9. Сделайте файл с предсказаниями для Kaggle (submission.csv).

Обязательно постройте хотя бы 4 графика с помощью seaborn и matplotlib:

- Распределение выживших и невыживших
- Зависимость выживания от пола
- Выживание по классу билета
- Распределение возраста среди выживших и невыживших

Используя тот же датасет Titanic, попробуйте ответить на вопрос:
От чего зависел шанс выжить?

Для этого:

1. Проведите свой анализ данных:
   Постройте графики зависимости выживания от признаков.
   Посмотрите на корреляции числовых признаков.
   Можете придумать свои визуализации.

2. Попробуйте улучшить модель:
   Попробуйте другие модели (логистическую регрессию, дерево решений).
   Проведите подбор параметров через GridSearchCV.