СӘТБАЕВ УНИВЕРСИТЕТІ



SATBAYEV UNIVERSITY

Основы искусственного интеллекта. Лекция 5. Классификация и регрессия. Методы классификации и регрессии, деревья решений, случайный лес

Преподаватель: **Қасенхан Арай Мейрамбайқызы, доктор PhD, ассоциированный профессор кафедры «Программная инженерия»**

a.kassenkhan@satbayev.university

Содержание

1. Классификация и регрессия
2. Основные алгоритмы
3. Линейная и логистическая регрессия
4. Decision Tree
5. Random Forest
6. Практический пример
7. Сравнение моделей

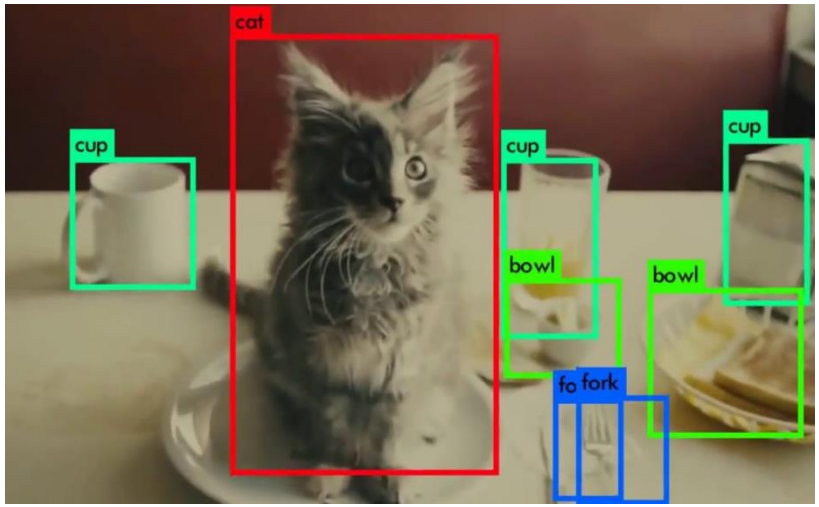
По завершению урока Вы будете знать:

1. **Чем классификация отличается от регрессии**
2. **Какие алгоритмы применяются для разных типов задач**
3. **Как работает линейная и логистическая регрессия**
4. **Как принимает решение Decision Tree**
5. **Зачем объединять несколько деревьев в Random Forest**
6. **Как создать и обучить эти модели в Scikit-learn**

Классификация или регрессия?

КЛАССИФИКАЦИЯ

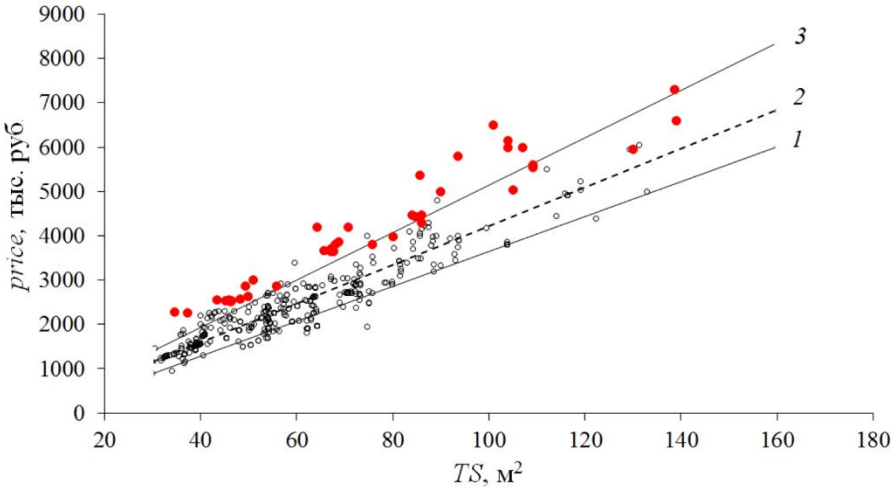
Предсказываем класс / категорию



Результат = категория

РЕГРЕССИЯ

Предсказываем числовое значение



Результат = число

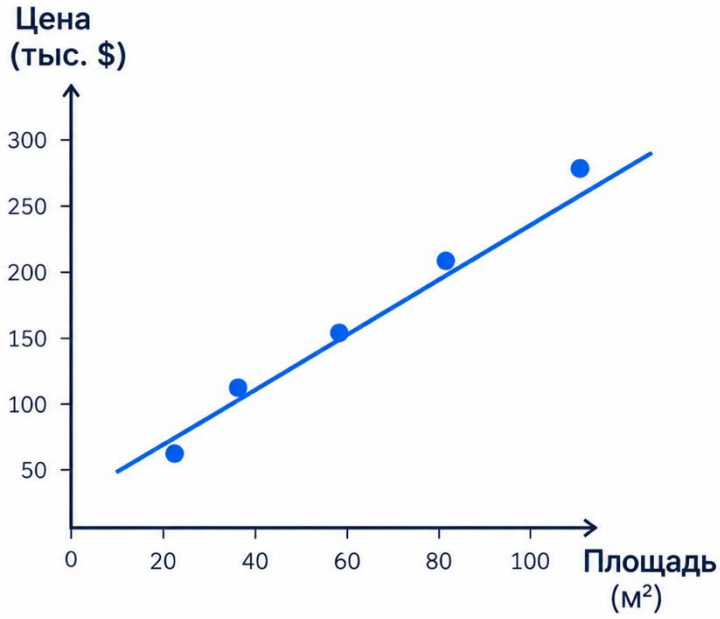
Какой результат мы хотим получить: класс или число?

Основные алгоритмы

Алгоритм	Классификация	Регрессия
Logistic Regression	✓	
Linear Regression		✓
KNN	✓	✓
Decision Tree	✓	✓
Random Forest	✓	✓
SVM / SVR	✓	✓

Один тип задачи может решаться разными алгоритмами.

Линейная регрессия



Идея

Модель ищет зависимость:

X → числовое y

$$y = wx + b$$

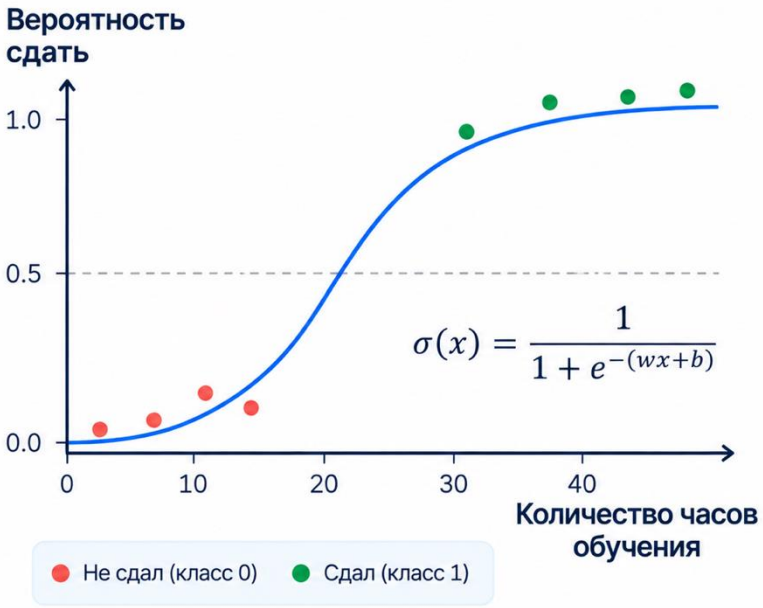
```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

Linear Regression → прогнозирует число

Логистическая регрессия



Модель получает вероятность

0.15 → **класс 0**
(Не сдал)

0.82 → **класс 1**
(Сдал)

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

Logistic Regression → вероятность → класс

Как выбрать тип задачи?



Сначала определяем задачу → затем выбираем модель

Decision Tree: как модель принимает решение

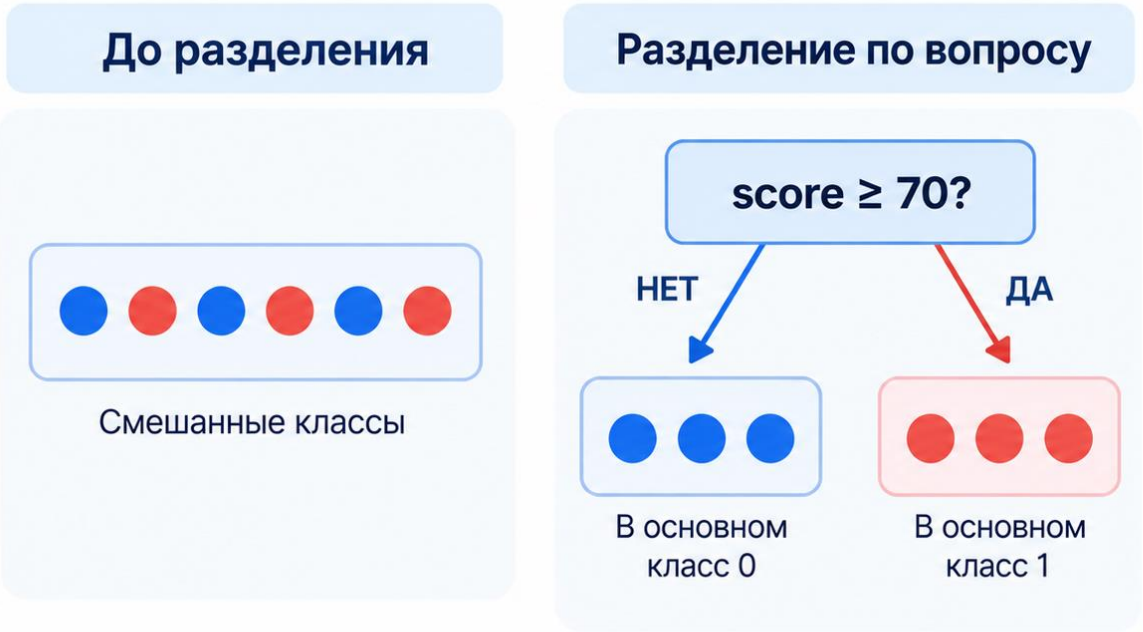


Decision Tree = последовательность вопросов

score ≥ 70?	**Root / Node** → вопрос
НЕТ	**Branch** → ответ
Высокий результат	**Leaf** → предсказание

Дерево последовательно разделяет данные по признакам.

Как дерево выбирает разделение?



Цель

Получить более «чистые» группы

Gini ↓
узел чище

Entropy ↓
неопределённость меньше

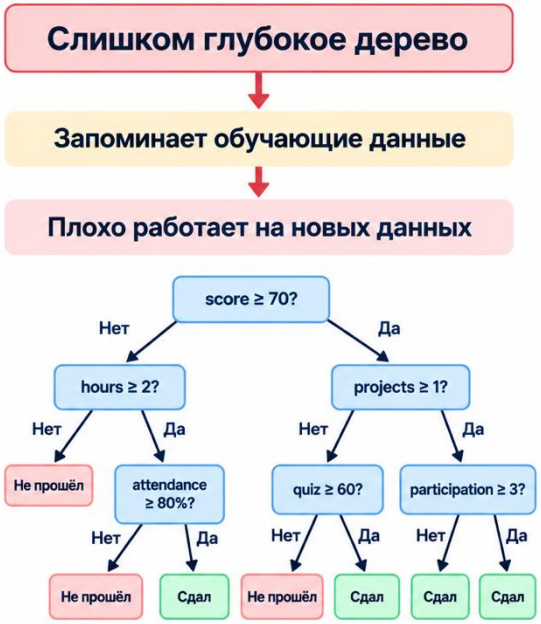
Алгоритм выбирает такое разделение, которое лучше отделяет классы.

Decision Tree: преимущества и проблема

Плюсы
✓ легко понять;
✓ можно визуализировать;
✓ работает с нелинейными зависимостями;
✓ подходит для классификации и регрессии.

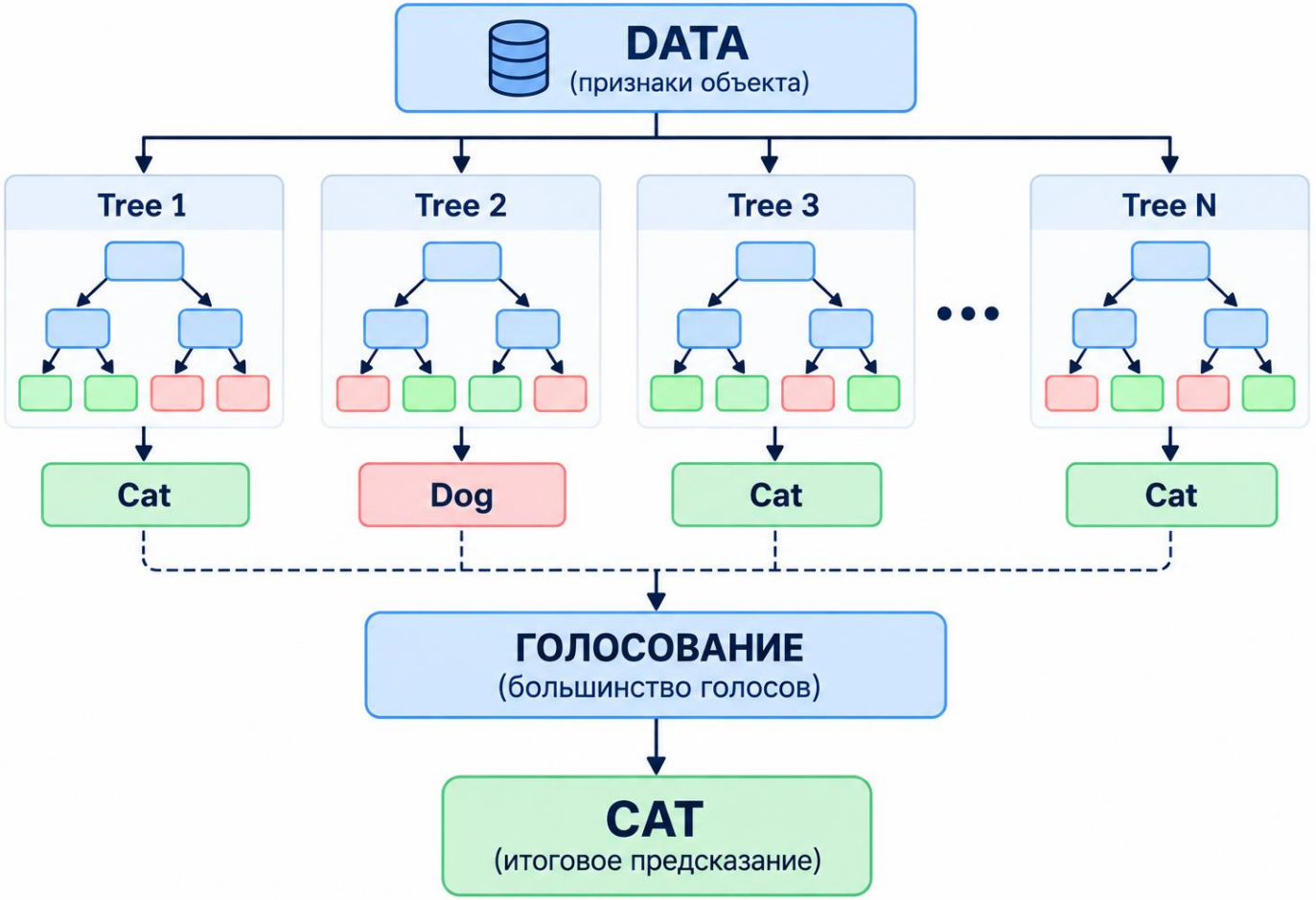
Главная проблема
OVERFITTING — переобучение





Чем сложнее дерево, тем выше риск переобучения.

Random Forest: много деревьев



Random Forest = множество разных Decision Trees

Почему Random Forest работает лучше одного дерева?

Decision Tree

Одно дерево

- легко объяснить
  – может переобучиться
  – чувствительно к изменениям данных

Random Forest

Много деревьев

- устойчивее
- обычно точнее
- меньше риск переобучения
  – сложнее интерпретировать
  – требует больше вычислений

Разные деревья совершают разные ошибки → объединение делает прогноз устойчивее.

Практика: Decision Tree

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

model = DecisionTreeClassifier(
    max_depth=3,
    random_state=42
)

model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(accuracy_score(y_test, y_pred))
```

Меняем одну модель: Random Forest

Было:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(
    max_depth=3,
    random_state=42
)
```

Стало:

```
from sklearn.ensemble import RandomForestClassifier

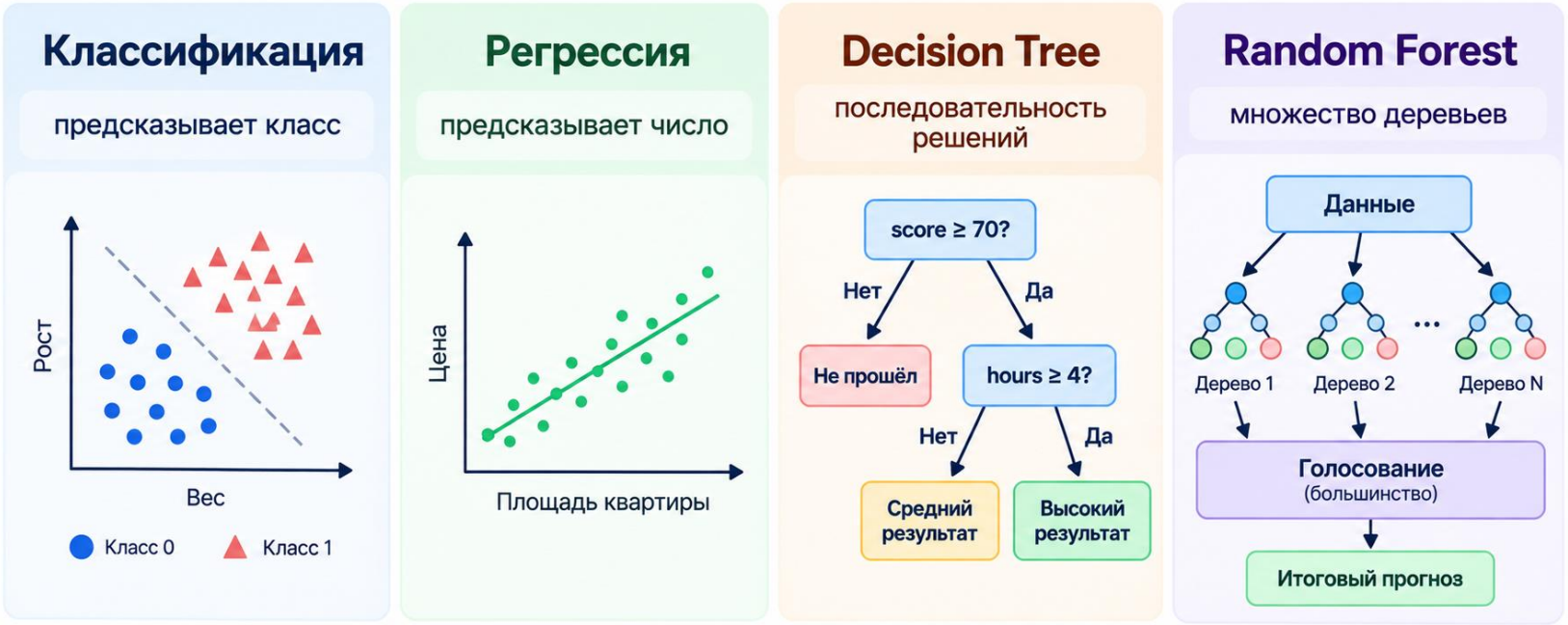
model = RandomForestClassifier(
    n_estimators=100,
    random_state=42
)
```

Остальное:

```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(accuracy_score(y_test, y_pred))
```

ML pipeline остаётся тем же — меняется модель.

Заключение





Нет одной лучшей модели для всех задач — алгоритм выбирается с учётом данных и цели.