ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

Лекция 15

Введение в машинное обучение

Оспан Ә.Ғ., PhD, доцент-исследователь

КазНУ, кафедра ИИ и Big Data

2026-2027 учебный год

Цель и содержание лекции

Цель и результаты обучения

- Различать классификацию, регрессию и кластеризацию
- Собрать Pipeline без утечки между обучением и проверкой
- Сравнить модель с baseline по нескольким метрикам

Содержание лекции

1. **X и y**
2. **Разделение данных**
3. **Baseline и Pipeline**
4. **Метрики и CV**
5. **Единственная финальная проверка**

Три задачи: категория, число и структура групп

Задача	Ответ	Учебный вопрос	Оценка
Классификация	Категория	GK или другая позиция?	Матрица ошибок, F1
Регрессия	Число	Числовое измерение по признакам	MAE, RMSE
Кластеризация	Группа без метки	Какие профили похожи?	Устойчивость и предметный смысл

Предсказательная связь и причинное объяснение - разные результаты.

Учебная цель GK и заранее разрешенные признаки

Цель и X

- y=1 при Position==GK; иначе y=0.
- Age, SprintSpeed, Stamina, Strength.
- Reactions и ShortPassing дополняют шесть признаков.

Исключения

- Position и все GK* исключены из X.
- Name, Club, Nationality, Wage, Value не используются.
- Unnamed: 0 - технический старый индекс.

Признаки определяют до проверки качества и оценивают по моменту доступности.

Читаем данные и откладываем 20% для финальной проверки

Пример Python

```
from pathlib import Path
import pandas as pd
from sklearn.model_selection import train_test_split

football = pd.read_csv(Path('data') / 'football.csv')
features = ['Age', 'SprintSpeed', 'Stamina',
            'Strength', 'Reactions', 'ShortPassing']
X = football.loc[:, features].copy()
y = football['Position'].eq('GK').astype('int64')
assert not any(c.startswith('GK') for c in X)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)
assert X_train.index.intersection(X_test.index).empty
print(len(X_train), len(X_test))
```

Результат: 10317 2580

Честное разделение

- Test отделяется до настройки преобразований.
- Стратификация сохраняет приблизительные доли классов.

Проверка границ

- Строки train и test не пересекаются.
- Случайное разделение строк не проверяет новые клубы.

Train, внутренняя CV и окончательный Test

1

Отложить Test

20% строк; не использовать их для выбора решений.

2

CV внутри Train

Пять частей; все преобразования учатся заново в каждом fold.

3

Зафиксировать процедуру

Признаки, алгоритмы, параметры, метрики и порог 0.5.

4

Финальная оценка

Обучить на всем train и один раз проверить test.

- CV не должна получать X_test или y_test.
- После просмотра Test не подбирают C, признаки или порог.

Baseline и логистическая регрессия

DummyClassifier

- Всегда предсказывает наиболее частый класс train.
- Игнорирует признаки: точка отсчета полезности.
- Высокая accuracy может скрывать нулевой recall GK.

LogisticRegression

- Линейный score преобразуется в вероятность сигмоидой.
- Порог классификации по умолчанию равен 0.5.
- Регуляризация; фиксируем C=1.0 до Test.

Основную модель сравнивают с простым правилом на одинаковых данных.

Pipeline объединяет масштабирование и классификатор

Пример Python

```
from sklearn.dummy import DummyClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

baseline = DummyClassifier(strategy='most_frequent')
model = Pipeline([
    ('scale', StandardScaler()),
    ('classifier', LogisticRegression(
        C=1.0, max_iter=1000, random_state=42)),
])
```

Масштаб

- Средние и SD вычисляются на обучающих данных.
- Проверочные строки преобразуются сохраненными параметрами.

Один estimator

- Pipeline можно безопасно передать в cross_validate.
- Внутри CV подготовка тоже обучается заново.

Что происходит в одном CV-повторе

1

Четыре части Train

Оценить средние и SD только на этих строках.

2

Обучить классификатор

Использовать преобразованные обучающие признаки.

3

Пятая часть Train

Применить готовый scaler и вычислить метрики.

4

Следующий Fold

Создать новую копию всего Pipeline.

- Нельзя масштабировать весь Train перед cross_validate.
- Отложенный Test не участвует ни в одном CV-повторе.

Матрица ошибок: положительный класс GK



Строки: истинный класс. Всего 100 наблюдений

Строки истинные, столбцы предсказанные

- Порядок классов [0,1]; 1 означает GK.
- TN=85, FP=5, FN=2, TP=8.
- Accuracy=0.93; recall GK=0.8; precision GK примерно 0.615.
- Пример иллюстрирует метрики, а не качество football-модели.

Высокая общая accuracy может скрывать ошибки редкого класса.

Метрика отвечает конкретному вопросу об ошибках

Метрика	Вопрос	Числовой пример
Accuracy	Какая доля всех ответов верна?	93/100=0.93
Precision GK	Какая доля предсказанных GK верна?	8/13~0.615
Recall GK	Какую долю настоящих GK нашли?	8/10=0.8
F1 GK	Как совместить precision и recall?	16/23~0.696
Balanced accuracy	Какова средняя полнота классов?	(85/90+8/10)/2~0.872

Набор метрик выбирают до финальной проверки с учетом цены FP и FN.

Пять CV-частей только внутри Train

Пример Python

```
from sklearn.model_selection import StratifiedKFold, cross_validate

cv = StratifiedKFold(
    n_splits=5, shuffle=True, random_state=42)
for name, est in [('dummy', baseline), ('logistic', model)]:
    scores = cross_validate(
        est, X_train, y_train, cv=cv,
        scoring=['balanced_accuracy', 'f1'])
    for metric in ['balanced_accuracy', 'f1']:
        values = scores['test_' + metric]
        print(name, metric, values.mean(), values.std(ddof=1))
```

Граница данных

- В вызове только X_train и y_train.
- Один cv задает сопоставимые части обеим моделям.

Интерпретация

- test_* здесь означает внутреннюю проверочную часть.
- Среднее и SD fold-оценок не являются автоматически ДИ.

Перед Holdout фиксируем окончательное решение

Зафиксировать до Test

- Список X, определение y и два алгоритма.
- C=1.0, scaler, split, CV и порог 0.5.
- Accuracy, balanced accuracy, precision, recall, F1 и матрицу.

После финального результата

- Сохранить метрики и описать ошибки.
- Не выбирать новые признаки и настройки по Test.
- Улучшения планировать как отдельный следующий эксперимент.

Финальный Test используют один раз после всех решений по Train.

Единый финальный этап: обе модели на одном Test

Пример Python

```
from sklearn.metrics import (
    balanced_accuracy_score, classification_report,
    confusion_matrix)

for name, est in [('dummy', baseline), ('logistic', model)]:
    est.fit(X_train, y_train)
    prediction = est.predict(X_test)
    print(name, confusion_matrix(
        y_test, prediction, labels=[0, 1]))
    print('Balanced accuracy:',
          balanced_accuracy_score(y_test, prediction))
    print(classification_report(
        y_test, prediction, labels=[0, 1], zero_division=0))
```

Одна финальная проверка

- Оба алгоритма получают одинаковые строки Test.
- Обучение выполняется только на всем Train.

Совместный отчет

- Матрица, balanced accuracy и отчет по каждому классу.
- Не менять процедуру после просмотра метрик.

Результат учебного Holdout не доказывает реальное качество применения.

Границы качества и самые опасные ошибки

Утечка

- Position или GK* в X.
- Масштабирование до Split или CV.

Проверка

- Подбор C, порога или seed по Test.
- Только accuracy без ошибок редкого класса.

Перенос

- Строковый Split не проверяет новые клубы и годы.
- Учебная связь не доказывает причинность или спортивную пригодность.

Главные результаты лекции

1. X и y определяются до проверки; Position и все GK* исключаются из X.
2. Baseline показывает качество простого правила без использования признаков.
3. Pipeline учит scaler и модель только на обучающих частях.
4. CV использует только Train; Holdout проверяется после фиксации решений.
5. Метрики и матрица ошибок требуют ограниченной интерпретации для источника.

Проверка понимания и самостоятельная работа

1. Почему LogisticRegression здесь решает классификацию?
2. Где и на каких строках StandardScaler учится в каждом fold?
3. Почему baseline может иметь высокую accuracy и нулевой recall GK?
4. Чем test_f1 из cross_validate отличается от метрики Holdout?
5. Почему случайное разделение строк не доказывает перенос на новые клубы?

Самостоятельная работа

Для матрицы [[72,8],[6,14]] рассчитайте precision, recall, F1 и balanced accuracy. Подпишите истинный и предсказанный классы.

Документация и исходные материалы

Документация

1. **scikit-learn: утечки и типичные ошибки**
   https://scikit-learn.org/stable/common_pitfalls.html
2. **scikit-learn: Pipeline**
   https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
3. **scikit-learn: метрики классификации**
   https://scikit-learn.org/stable/modules/model_evaluation.html

Основа лекции

- football.csv