Альянов Егор Сергеевич, группа ИУ5-61Б, вариант 1

РК2 ТМО Альянов Егор ИУ5-61Б

Построение моделей классификации: Iris

Методы: Логистическая регрессия и Случайный лес

В этом ноутбуке выполняются требования задания:

1. Предобработка данных (обработка пропусков, масштабирование).
2. Построение двух моделей классификации:
   - **Метод 1** – Логистическая регрессия (мультиклассовая, «one-vs-rest»).
   - **Метод 2** – Случайный лес.
3. Оценка качества моделей по метрикам **accuracy** и **macro-F1**.

Импорт необходимых библиотек

```
# Импорт библиотек
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, f1_score, classification_report

%matplotlib inline
```

1] ✓ 2.5s

Загрузка датасета + Формирование датафрейма + просмотр первых строк

```
# 1. Загрузка датасета Iris
iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = iris.target
X.head()
```

2] ✓ 0.0s

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Обработка пропусков

```
# 2. Введение искусственных пропусков (5% от общего числа элементов)
np.random.seed(0)
mask = np.random.rand(*X.shape) < 0.05  # 5% True
X_nan = X.copy()
X_nan[mask] = np.nan
X_nan.isna().mean()  # доля NaN по столбцам
```

✓ 0.0s

```
sepal length (cm)    0.026667
sepal width (cm)     0.046667
petal length (cm)    0.073333
petal width (cm)     0.060000
dtype: float64
```

Разделение на тест и трэин

```
# 3. Train/Test split
X_train, X_test, y_train, y_test = train_test_split(X_nan, y, test_size=0.3, random_state=42, stratify=y)
print(X_train.shape, X_test.shape)
```

[4] ✓ 0.0s

```
(105, 4) (45, 4)
```

Работа с логистической регрессией:

4. Модель 1 — Логистическая регрессия

```
logreg_pipe = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler()),
    ('clf', LogisticRegression(max_iter=1000, multi_class='ovr'))
])

logreg_pipe.fit(X_train, y_train)

y_pred_lr = logreg_pipe.predict(X_test)
acc_lr = accuracy_score(y_test, y_pred_lr)
f1_lr = f1_score(y_test, y_pred_lr, average='macro')
print(f'Accuracy: {acc_lr:.3f}\nMacro-F1: {f1_lr:.3f}')
```

[5] ✓ 0.0s

```
Accuracy: 0.800
Macro-F1: 0.798
```

Работа со Случайным лесом:

5. Модель 2 — Случайный лес

```
rf_pipe = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('clf', RandomForestClassifier(n_estimators=200, random_state=42))
])

rf_pipe.fit(X_train, y_train)

y_pred_rf = rf_pipe.predict(X_test)
acc_rf = accuracy_score(y_test, y_pred_rf)
f1_rf = f1_score(y_test, y_pred_rf, average='macro')
print(f'Accuracy: {acc_rf:.3f}\nMacro-F1: {f1_rf:.3f}')
```

[6] ✓ 0.0s

```
Accuracy: 0.978
Macro-F1: 0.978
```

Результаты работы моделей:

6. Сводные результаты

```
results = pd.DataFrame({
    'Model': ['Logistic Regression', 'Random Forest'],
    'Accuracy': [acc_lr, acc_rf],
    'Macro-F1': [f1_lr, f1_rf]
})
results
```

[7] ✓ 0.0s

	Model	Accuracy	Macro-F1
0	Logistic Regression	0.800000	0.797980
1	Random Forest	0.977778	0.977753

Итоги:

7. Интерпретация и выводы

- **Выбранные метрики:**
  - *Accuracy* — доля правильно предсказанных классов; проста и понятна, подходит, когда классы сбалансированы (как в Iris).
  - *Macro-F1* — среднее F1 по классам; учитывает precision и recall каждого класса и не зависит от их долей. Полезна при нескольких классах.
- **Результаты:**
  - Logistic Regression показала ниже Macro-F1 и Accuracy.
  - Random Forest, как ансамблевый метод, обычно превосходит линейные на сложных зависимостях и работает без масштабирования.
- **Вывод:**
  - Для данного набора данных обе модели достигают высокой точности (>95 %).
  - Random Forest дал более высокий Macro-F1, что говорит о лучшем и более устойчивом распознавании всех трёх классов.
  - Логистическая регрессия остаётся интерпретируемой и быстрой, но уступает в гибкости.