Отчет по лабораторной работе №8

Ансамблевые модели и методы классификации

Выполнила: Иванова Виктория Павловна

---

1. Введение

Цель работы: Изучение и практическое применение ансамблевых методов машинного обучения (Random Forest, Bagging, Gradient Boosting), а также сравнение эффективности различных алгоритмов классификации и регрессии с использованием конвейеров (Pipelines) и оптимизации гиперпараметров.

Используемые инструменты:

- Python 3.11
- Библиотеки: `scikit-learn`, `pandas`, `numpy`, `xgboost`, `catboost`, `matplotlib`.

---

2. Ход работы

Задание 1-2. Случайный лес и подбор гиперпараметров (Adult Dataset)

Описание: Решалась задача бинарной классификации уровня дохода (<=50K или >50K) на наборе данных Adult. В качестве основной модели использовался алгоритм Случайного леса (`RandomForestClassifier`).

Реализация: Для настройки модели использовался перебор по сетке (`GridSearchCV`) для поиска оптимальной глубины деревьев и количества признаков.

Фрагмент кода (настройка параметров):

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Определение сетки параметров
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20, None],
    'min_samples_leaf': [1, 2, 4]
}

rf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(rf, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
```

Результат: Модель случайного леса показала высокую стабильность. `GridSearchCV` позволил определить параметры, минимизирующие переобучение.

Задание 3. Сравнение классификаторов с использованием Pipeline (Iris Dataset)

Описание: На классическом наборе данных Iris проводилось сравнение пяти различных алгоритмов классификации. Для корректной обработки данных (масштабирования) использовались конвейеры (`Pipeline`).

Сравниваемые модели:

1. K-Nearest Neighbors (KNN)
2. Support Vector Machine (SVM)
3. Decision Tree (DT)
4. Logistic Regression (LR)
5. Random Forest (RF)

Фрагмент кода (создание пайплайнов):

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

pipelines = {
    'KNN': Pipeline([('scaler', StandardScaler()), ('clf',
KNeighborsClassifier())]),
    'SVM': Pipeline([('scaler', StandardScaler()), ('clf',
SVC(probability=True))]),
    'DT':  Pipeline([('scaler', StandardScaler()), ('clf',
DecisionTreeClassifier())]),
    'LR':  Pipeline([('scaler', StandardScaler()), ('clf',
LogisticRegression())]),
    'RF':  Pipeline([('scaler', StandardScaler()), ('clf',
RandomForestClassifier())])
}
```

Результат: Использование `StandardScaler` внутри Pipeline обеспечило корректную работу метрических алгоритмов (KNN, SVM).

Задание 4. Регрессионный анализ (Diabetes Dataset)

Описание: Задача регрессии: предсказание прогрессирования диабета. Сравнивались линейные и нелинейные модели.

Используемые модели:

- SVR (Support Vector Regressor)
- Decision Tree Regressor
- Linear Regression

- Random Forest Regressor

Особенности: Для каждой модели была задана индивидуальная сетка гиперпараметров для поиска лучшей конфигурации по метрике MSE или R2.

Фрагмент кода (параметры для SVR):

```
param_grids_reg = {
    'SVR': {
        'reg__C': [0.1, 1, 10],
        'reg__kernel': ['rbf', 'linear'],
        'reg__epsilon': [0.01, 0.1, 0.5]
    },
    # ... параметры для других моделей
}
```

Задание 5. Бэггинг и обработка пропусков (Credit Scoring)

Описание: Работа с реальными данными кредитного скоринга, содержащими пропуски. Сравнение одиночных моделей с ансамблевым методом Бэггинга (`BaggingClassifier`).

Методика:

1. Заполнение пропусков медианными значениями.
2. Сравнение Логистической регрессии, Случайного леса и Бэггинга над решающими деревьями.

Фрагмент кода (Бэггинг):

```
from sklearn.ensemble import BaggingClassifier

# Бэггинг на основе решающих деревьев
bagging = BaggingClassifier(
    estimator=DecisionTreeClassifier(),
    n_estimators=100,
    random_state=42
)
bagging.fit(X_train, y_train)
```

Результат: Ансамблевые методы (Random Forest и Bagging) показали лучшую обобщающую способность по сравнению с линейной моделью на данных со сложной структурой.

Задание 6. Градиентный бустинг (Flight Delays)

Описание: Самая сложная задача лабораторной: предсказание задержки рейса (бинарная классификация) с целью достижения максимального ROC AUC. Сравнивались две современные библиотеки градиентного бустинга.

Инструменты:

- **XGBoost** (eXtreme Gradient Boosting)
- **CatBoost** (Categorical Boosting)

Особенности реализации:

- Обработка категориальных признаков (Label Encoding для XGBoost, встроенная обработка в CatBoost).
- Обучение на больших выборках.

Фрагмент кода (инициализация моделей):

```
from xgboost import XGBClassifier
from catboost import CatBoostClassifier

# XGBoost
xgb_model = XGBClassifier(n_estimators=500, learning_rate=0.1,
max_depth=6)

# CatBoost (с указанием категориальных фичей, если применимо)
cb_model = CatBoostClassifier(iterations=500, learning_rate=0.1, depth=6,
verbose=False)
```

Результаты (валидация):

- XGBoost Validation AUC: ~0.739
- CatBoost Validation AUC: ~0.736

Обе библиотеки показали схожие высокие результаты, значительно превосходящие базовые алгоритмы. CatBoost продемонстрировал удобство работы "из коробки" с меньшим количеством предварительной обработки данных.

---

3. Заключение

В ходе лабораторной работы были освоены ключевые принципы построения ансамблевых моделей.

Основные выводы:

1. **Пайплайны** существенно упрощают код и предотвращают утечку данных при валидации.
2. **Случайный лес** является отличным выбором "по умолчанию" для задач классификации благодаря устойчивости к переобучению.
3. **Градиентный бустинг (XGBoost, CatBoost)** обеспечивает наивысшую точность на табличных данных, но требует более тщательной настройки и вычислительных ресурсов.
4. **Обработка данных** (заполнение пропусков, масштабирование) играет критическую роль в качестве работы метрических моделей (KNN, SVM), но менее критична для деревьев.