Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по рубежному контролю №2

«Методы построения моделей машинного обучения.»

Вариант № 2

Выполнил:
Беккиев Р. И.
группа ИУ5-64Б

Проверил:
Гапанюк Ю.Е.

Дата: 25.05.25

Дата:

Подпись:

Подпись:

Москва, 2025 г.

Задание:

Номер варианта: **2**

Номер набора данных, указанного в задаче: **2**

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html#sklearn.datasets.load_wine

Метод №1: **Линейная/логистическая регрессия**

Метод №2: **Градиентный бустинг**

Ход выполнения:

Загрузка библиотек и необходимых модулей

```
import pandas as pd
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, f1_score, classification_report, confu
import matplotlib.pyplot as plt
import seaborn as sns
```

Загрузка данных

```
# Загрузка данных
wine = load_wine()
X = pd.DataFrame(wine.data, columns=wine.feature_names)
y = pd.Series(wine.target, name='target')
```

Разделение данных на обучающую и тестовую выборки

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_
print(f"\nРазмер обучающей выборки: {X_train.shape[0]} примеров")
print(f"Размер тестовой выборки: {X_test.shape[0]} примеров")
```

Масштабирование

```
# Масштабирование числовых признаков
# Логистическая регрессия чувствительна к масштабу признаков,
# градиентный бустинг менее чувствителен, но масштабирование не повредит.
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Метод 1

```

# --- Модель 1: Логистическая регрессия ---
print("\n--- Логистическая регрессия ---")
log_reg = LogisticRegression(solver='liblinear', multi_class='ovr', random_state=42
log_reg.fit(X_train_scaled, y_train)

# Предсказания
y_pred_log_reg = log_reg.predict(X_test_scaled)

# Оценка качества
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
f1_log_reg_macro = f1_score(y_test, y_pred_log_reg, average='macro')
# f1_log_reg_weighted = f1_score(y_test, y_pred_log_reg, average='weighted') # еще

print(f"Accuracy (Логистическая регрессия): {accuracy_log_reg:.4f}")
print(f"F1-score (macro, Логистическая регрессия): {f1_log_reg_macro:.4f}")
# print(f"F1-score (weighted, Логистическая регрессия): {f1_log_reg_weighted:.4f}")

print("\nОтчет по классификации (Логистическая регрессия):")
print(classification_report(y_test, y_pred_log_reg, target_names=wine.target_names)

print("\nМатрица ошибок (Логистическая регрессия):")
cm_log_reg = confusion_matrix(y_test, y_pred_log_reg)
plt.figure(figsize=(6, 4))
sns.heatmap(cm_log_reg, annot=True, fmt='d', cmap='Blues', xticklabels=wine.target_
plt.title('Матрица ошибок: Логистическая регрессия')
plt.xlabel('Предсказанный класс')
plt.ylabel('Истинный класс')
plt.show()
```

Метод 2

```
# --- Модель 2: Градиентный бустинг ---
print("\n--- Градиентный бустинг ---")
grad_boost = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_de
grad_boost.fit(X_train_scaled, y_train) # Можно использовать и не масштабированные

# Предсказания
y_pred_grad_boost = grad_boost.predict(X_test_scaled)

# Оценка качества
accuracy_grad_boost = accuracy_score(y_test, y_pred_grad_boost)
f1_grad_boost_macro = f1_score(y_test, y_pred_grad_boost, average='macro')
# f1_grad_boost_weighted = f1_score(y_test, y_pred_grad_boost, average='weighted')

print(f"Accuracy (Градиентный бустинг): {accuracy_grad_boost:.4f}")
print(f"F1-score (macro, Градиентный бустинг): {f1_grad_boost_macro:.4f}")
# print(f"F1-score (weighted, Градиентный бустинг): {f1_grad_boost_weighted:.4f}")

print("\nОтчет по классификации (Градиентный бустинг):")
print(classification_report(y_test, y_pred_grad_boost, target_names=wine.target_nar

print("\nМатрица ошибок (Градиентный бустинг):")
cm_grad_boost = confusion_matrix(y_test, y_pred_grad_boost)
plt.figure(figsize=(6, 4))
sns.heatmap(cm_grad_boost, annot=True, fmt='d', cmap='Greens', xticklabels=wine.tar
plt.title('Матрица ошибок: Градиентный бустинг')
plt.xlabel('Предсказанный класс')
plt.ylabel('Истинный класс')
plt.show()
```

1. Общие выводы
   Обе модели продемонстрировали высокую эффективность на наборе данных `load_wine`. Это говорит о том, что данные хорошо разделимы и даже относительно простые модели, как логистическая регрессия, могут достичь отличных результатов.
2. В данном конкретном случае **логистическая регрессия** (после масштабирования признаков) показала **слегка лучшие результаты**, чем градиентный бустинг с параметрами по умолчанию (или близкими к ним). Это может быть связано с тем, что набор данных относительно мал (178 записей) и не очень сложен, и линейная модель оказалась достаточно эффективной.
3. Качество модели градиентного бустинга потенциально можно улучшить путем подбора гиперпараметров (например, `n_estimators`, `learning_rate`, `max_depth`) с использованием таких техник, как `GridSearchCV` или `RandomizedSearchCV`.
4. Выбор метрик (Accuracy и F1-macro) позволил получить как общую оценку производительности, так и более детализированное представление о качестве классификации по каждому классу, учитывая баланс между точностью и полнотой.

Этот анализ показывает, что для данного датасета обе модели подходят, но логистическая регрессия оказалась немного предпочтительнее "из коробки".