Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №5

Выполнил:
Фролов М. К.
группа ИУ5-64Б

Проверил:
Гапанюк Ю.Е.

Дата: 07.04.25

Подпись:

Дата:

Подпись:

Москва, 2025 г.

Цель лабораторной работы: изучение ансамблей моделей машинного обучения.

Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите следующие ансамблевые модели:

   - две модели группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);

   - AdaBoost;

   - градиентный бустинг.

5. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Ход выполнения:

```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
```

[6] ✓ 0.0s

```python
# Загрузка данных
data = pd.read_csv('train.csv')


data = data.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1)

data['Age'] = data['Age'].fillna(data['Age'].median())
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])

# OneHotEncoding категориальных признаков
encoder = OneHotEncoder(drop='first', sparse_output=False)
categorical_features = ['Sex', 'Embarked']
encoded_features = encoder.fit_transform(data[categorical_features])
encoded_df = pd.DataFrame(encoded_features, columns=encoder.get_feature_names_out(categorical_features))

# Объединяем с основными данными и удаляем оригинальные категориальные столбцы
data = pd.concat([data.drop(columns=categorical_features), encoded_df], axis=1)

# Разделение на признаки (X) и целевую переменную (y)
X = data.drop('Survived', axis=1)
y = data['Survived']
```

[7] ✓ 0.1s

```python
from sklearn.model_selection import train_test_split

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

[8] ✓ 0.0s Python

```python
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier, AdaBoostClassifier, GradientBoostingClassifier
```

[9] ✓ 0.0s Python

Случайный лес

Строит несколько решающих деревьев, обучая их на случайных подмножествах данных и признаков, затем усредняет результаты.

```python
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
```

[10] ✓ 0.2s Python

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Сверхслучайные деревья

Подобен случайному лесу, но строит деревья с случайным выбором разбиений, без оптимизации на каждом узле.

```python
et = ExtraTreesClassifier(random_state=42)
et.fit(X_train, y_train)
```

[11] ✓ 0.1s Python

AdaBoost

Последовательно обучает слабые классификаторы, каждый из которых фокусируется на ошибках предыдущего.

```
ada = AdaBoostClassifier(random_state=42)
ada.fit(X_train, y_train)
```

[12] ✓ 0.1s Python

```
AdaBoostClassifier(random_state=42)
```

Градиентный бустинг

Строит классификаторы последовательно, исправляя ошибки предыдущих моделей, минимизируя ошибку на каждом шаге.

```
gb = GradientBoostingClassifier(random_state=42)
gb.fit(X_train, y_train)
```

[13] ✓ 0.2s Python

```
GradientBoostingClassifier(random_state=42)
```

```
from sklearn.metrics import accuracy_score

# Функция для оценки моделей
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    return accuracy

# Оценка случайного леса
rf_accuracy = evaluate_model(rf, X_test, y_test)

# Оценка сверхслучайных деревьев
et_accuracy = evaluate_model(et, X_test, y_test)

# Оценка AdaBoost
ada_accuracy = evaluate_model(ada, X_test, y_test)

# Оценка градиентного бустинга
gb_accuracy = evaluate_model(gb, X_test, y_test)

# Вывод результатов
print(f"Random Forest Accuracy: {rf_accuracy}")
print(f"Extra Trees Accuracy: {et_accuracy}")
print(f"AdaBoost Accuracy: {ada_accuracy}")
print(f"Gradient Boosting Accuracy: {gb_accuracy}")
```

[14] ✓ 0.0s Python

```
Random Forest Accuracy: 0.8212290502793296
Extra Trees Accuracy: 0.8156424581005587
AdaBoost Accuracy: 0.7988826815642458
Gradient Boosting Accuracy: 0.8044692737430168
```