Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Отчет по лабораторной работе №5

«Ансамбли моделей машинного обучения. Часть 1»

по дисциплине «Технологии машинного обучения»

Выполнил:
студент группы ИУ5Ц-84Б
Падалко К.Р.
подпись, дата

Проверил:
к.т.н., доц., Ю.Е. Гапанюк
подпись, дата

2025 г.

СОДЕРЖАНИЕ ОТЧЕТА

1. Цель лабораторной работы

Изучение ансамблей моделей машинного обучения.

2. Задание

1) Выберите набор данных (датасет) для решения задачи классификации или регрессии.
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.

3. Основные характеристики датасета

Название датасета: Набор данных о видах ирисов.

Ссылка: https://www.kaggle.com/datasets/uciml/iris

О датасетах

Этот набор данных содержит информацию о различных аспектах ирисов (цветков) из трех видов: Setosa, Versicolor и Virginica. В наборе представлены характеристики, такие как длина и ширина чашелистика и лепестка для 150 образцов ирисов. Данные используются для классификации видов ирисов на основе этих характеристик.

Набор данных включает 150 строк, каждая из которых представляет один ирис, и 5 столбцов.

Этот датасет использован для задач классификации и обучения моделей машинного обучения, таких как k-ближайших соседей, дерева решений, логистической регрессии и других методов классификации.

Структура данных

sepal length (длина чашелистика) — измеряется в сантиметрах.

sepal width (ширина чашелистика) — измеряется в сантиметрах.

petal length (длина лепестка) — измеряется в сантиметрах.

petal width (ширина лепестка) — измеряется в сантиметрах.

species (вид) — категориальная переменная, указывающая на вид ириса, который представлен в строке (Setosa, Versicolor или Virginica).

Выбор признаков для машинного обучения

Для машинного обучения выберем целевой признак – вид ирисов. Сопоставим с остальными признаками, а именно, характеристики цветов, вывялим примерный вид ириса.

4. Листинг

4.1.Анализ датасета

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import RobustScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier, ExtraTreesClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, ConfusionMatrixDisplay
from sklearn.datasets import load_iris
```

```
df = pd.read_csv('Iris.csv')
```

```
iris = load_iris()
data = pd.DataFrame(data=iris.data, columns=iris.feature_names)
data['target'] = iris.target
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   target             150 non-null    int32  
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

В датасете содержатся 150 строки, имеются 2 различные типы: int32 и float64.

Просмотр названий столбцов.

```
data.columns
```

```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)', 'target'],
      dtype='object')
```

Первые и последние пять строк датасета.

```
display(data.head())
display(data.tail())
```

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

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

```
data.head()
```

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

Количество строк и столбцов.

```
data.shape
```

```
(150, 5)
```

```
data.isnull().sum()
```

```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
target               0
dtype: int64
```

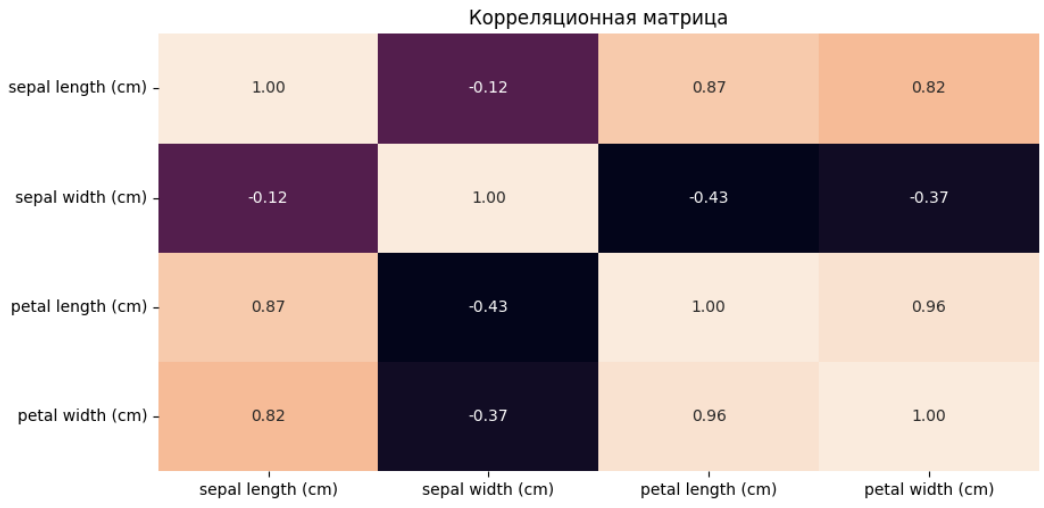
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   target             150 non-null    int32  
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

Поскольку у нас есть только числовые признаки в Iris dataset, мы можем сразу перейти к анализу

```
# Числовые признаки
num_feats = iris.feature_names
cat_feats = []  # В данном наборе данных нет категориальных признаков
```

```
# Визуализация корреляций
plt.figure(figsize=(10, 5))
sns.heatmap(data[num_feats].corr(method='pearson'), annot=True, fmt='.2f', cbar=False)
plt.title('Корреляционная матрица')
plt.show()
```



4.2.Описательная статистика

```
data.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

id: Это уникальные идентификаторы записей в данных. Каждая строка представляет отдельный образец ириса, и этот столбец не несет дополнительной информации о характеристиках цветов. Он служит лишь для идентификации строки в наборе данных.

sepallengthcm: Это длина чашелистика цветка ириса, измеренная в сантиметрах. Среднее значение длины чашелистика составляет 5.84 см. Диапазон значений от 4.3 см до 7.9 см, что указывает на разнообразие в длине чашелистика среди разных видов ирисов.

sepalwidthcm: Это ширина чашелистика цветка ириса, измеренная в сантиметрах. Средняя ширина чашелистика составляет 3.05 см. Значения варьируются от 2.0 см до 4.4 см, что показывает, что ширина чашелистика также имеет значительные колебания среди ирисов.

petallengthcm: Это длина лепестка цветка ириса, измеренная в сантиметрах. Средняя длина лепестка составляет 3.76 см. Длина лепестков варьируется от 1.0 см до 6.9 см, с большими различиями между образцами, что может указывать на разнообразие форм лепестков в зависимости от вида ириса.

petalwidthcm: Это ширина лепестка цветка ириса, измеренная в сантиметрах. Средняя ширина лепестка составляет 1.20 см. Значения варьируются от 0.1 см до 2.5 см, что также указывает на значительный разброс в характеристиках лепестков среди разных видов ирисов.

4.3.Машинное обучение

4.3.1. Разделение выборки

```
# Разделение данных на обучающую и тестовую выборку
X = data.drop(['target'], axis=1)
y = data['target']
```

4.3.2. Масштабирование данных

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=10)
```

```
# Пайплайны для числовых и категориальных признаков
num_pipeline = Pipeline([("scaling", RobustScaler())])
cat_pipeline = Pipeline([("onehot", OneHotEncoder())])
```

```
preprocessor = ColumnTransformer(
    [
        ("numerical", num_pipeline, num_feats),
    ],
    remainder="passthrough",
)
```

```
process_pipeline = Pipeline([("preprocessor", preprocessor)])

X_train_scaled = process_pipeline.fit_transform(X_train)
X_test_scaled = process_pipeline.transform(X_test)
```

4.3.3. Обучение модели

4.3.3.1. Случайный лес

```
# Обучение моделей
rf_cl = RandomForestClassifier(random_state=10, n_jobs=-1)
rf_cl.fit(X_train_scaled, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(n_jobs=-1, random_state=10)
```

4.3.3.2. Сверхслучайные деревья

```
et_cl = ExtraTreesClassifier(random_state=10, n_jobs=-1)
et_cl.fit(X_train_scaled, y_train)
```

```
ExtraTreesClassifier
ExtraTreesClassifier(n_jobs=-1, random_state=10)
```

4.3.3.3. AdaBoost

```
ab_cl = AdaBoostClassifier(random_state=10)
ab_cl.fit(X_train_scaled, y_train)
```

```
AdaBoostClassifier
AdaBoostClassifier(random_state=10)
```

4.3.3.4. Градиентный бустинг

```
gb_cl = GradientBoostingClassifier(loss="log_loss", random_state=10)
gb_cl.fit(X_train_scaled, y_train)
```

```
GradientBoostingClassifier
GradientBoostingClassifier(random_state=10)
```

4.4.Оценка качества моделей

```
# Оценка качества моделей
y_pred_rf = rf_cl.predict(X_test_scaled)
print("Random Forest Classification Report:\n", classification_report(y_test, y_pred_rf))
```

```
Random Forest Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         7

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

```
y_pred_et = et_cl.predict(X_test_scaled)
print("Extra Trees Classification Report:\n", classification_report(y_test, y_pred_et))
```

```
Extra Trees Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.92      0.96        13
           2       0.88      1.00      0.93         7

    accuracy                           0.97        30
   macro avg       0.96      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

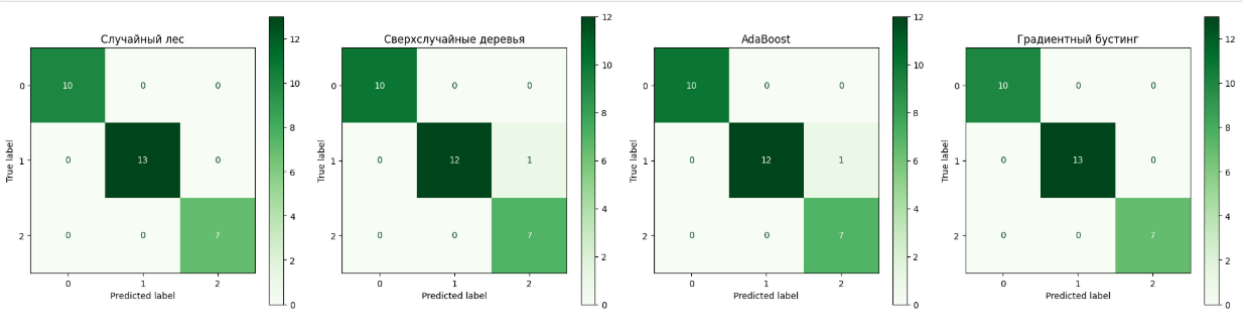
```
y_pred_ab = ab_cl.predict(X_test_scaled)
print("AdaBoost Classification Report:\n", classification_report(y_test, y_pred_ab))
```

```
AdaBoost Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.92      0.96        13
           2       0.88      1.00      0.93         7

    accuracy                           0.97        30
   macro avg       0.96      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

```
y_pred_gb = gb_cl.predict(X_test_scaled)
print("Gradient Boosting Classification Report:\n", classification_report(y_test, y_pred_gb))
```

```
Gradient Boosting Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         7

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

```python
# Матрицы путаницы
cm_rf = confusion_matrix(y_test, y_pred_rf)
cm_et = confusion_matrix(y_test, y_pred_et)
cm_ab = confusion_matrix(y_test, y_pred_ab)
cm_gb = confusion_matrix(y_test, y_pred_gb)

fig, axes = plt.subplots(1, 4, figsize=(20, 5))

disp1 = ConfusionMatrixDisplay(confusion_matrix=cm_rf)
disp1.plot(ax=axes[0], cmap=plt.cm.Greens)
axes[0].set_title('Случайный лес')

disp2 = ConfusionMatrixDisplay(confusion_matrix=cm_et)
disp2.plot(ax=axes[1], cmap=plt.cm.Greens)
axes[1].set_title('Сверхслучайные деревья')

disp3 = ConfusionMatrixDisplay(confusion_matrix=cm_ab)
disp3.plot(ax=axes[2], cmap=plt.cm.Greens)
axes[2].set_title('AdaBoost')

disp4 = ConfusionMatrixDisplay(confusion_matrix=cm_gb)
disp4.plot(ax=axes[3], cmap=plt.cm.Greens)
axes[3].set_title('Градиентный бустинг')

plt.tight_layout()
plt.show()
```



5. Вывод

В ходе лабораторной работы изучили ансамбли моделей машинного обучения.