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 №6

«Ансамбли моделей машинного обучения. Часть 2»

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:
   - одну из моделей группы стекинга;
   - модель многослойного персептрона. По желанию, вместо библиотеки scikit-learn возможно использование библиотек TensorFlow, PyTorch или других аналогичных библиотек;
   - двумя методами на выбор из семейства МГУА (один из линейных методов COMBI / MULTI + один из нелинейных методов MIA / RIA) с использованием библиотеки gmdh;
   - В настоящее время библиотека МГУА не позволяет решать задачу классификации.
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 gmdh
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import StackingClassifier, RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from gmdh import Combi, Ria
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
import numpy as np
import matplotlib.pyplot as plt
```

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

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

Просмотр названий столбцов.

```
df.columns
```

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

Первые и последние пять строк датасета.

```
display(df.head())
display(df.tail())
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

Количество строк и столбцов.

```
df.shape
```

```
(150, 6)
```

Проверка на наличие пропусков.

```
df.isnull().sum()
```

```
Id              0
SepalLengthCm   0
SepalWidthCm    0
PetalLengthCm   0
PetalWidthCm    0
Species         0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

4.2.Описательная статистика

```
df.describe()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

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 признаков и целевой переменной
X = df.drop(['Id', 'Species'], axis=1)  # Удаляем столбец Id, он не нужен для обучения
y = df['Species']
```

```
# Проверка на наличие NaN
if X.isnull().any().any() or y.isnull().any():
    raise ValueError("Данные содержат пустые значения!")
```

```
# Преобразование целевой переменной в числовые значения
le = LabelEncoder()
y_encoded = le.fit_transform(y)
```

4.3.2. Масштабирование данных

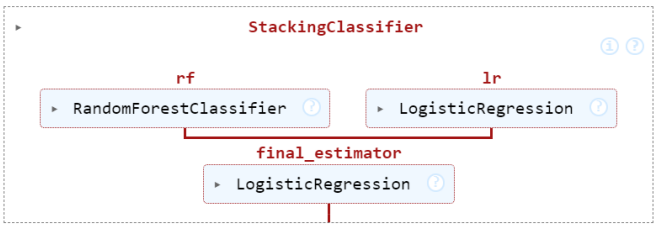
```
# Разделение выборки на обучающую и тестовую
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
```

```
# Определяем базовые модели
base_models = [
    ('rf', RandomForestClassifier(n_estimators=10)),
    ('lr', LogisticRegression(max_iter=200))
]
```

4.3.3. Обучение модели

4.3.3.1. Стекинг

```
# Создание стекированной модели
stacking_model = StackingClassifier(estimators=base_models, final_estimator=LogisticRegression())
stacking_model.fit(X_train, y_train)
```



4.3.3.2. Многослойный персептрон

```
# Обучение многослойного персептрона
mlp_model = MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000)
mlp_model.fit(X_train, y_train)
```

```
MLPClassifier
MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000)
```

```
# Конвертация DataFrame в NumPy массивы
X_train_np = X_train.to_numpy()
X_test_np = X_test.to_numpy()
```

4.3.3.3. Линейный метод (COMBI)

```
# Линейный метод
combi_model = Combi()
combi_model.fit(X_train_np, y_train)
```

```
<gmdh.gmdh.Combi at 0x1a98ff67b50>
```

4.3.3.4. Нелинейный метод (RIA)

```
# Нелинейный метод
ria_model = Ria()
ria_model.fit(X_train_np, y_train)
```

```
<gmdh.gmdh.Ria at 0x1a98ff9da30>
```

4.4.Оценка качества моделей

```
# Оценка стекированной модели
stacking_pred = stacking_model.predict(X_test)
stacking_accuracy = accuracy_score(y_test, stacking_pred)
```

```
# Оценка многослойного персептрона
mlp_pred = mlp_model.predict(X_test)
mlp_accuracy = accuracy_score(y_test, mlp_pred)
```

```
# Оценка линейной модели
combi_pred = combi_model.predict(X_test_np)
combi_pred = np.round(combi_pred).astype(int)  # Округление и приведение к целому числу
combi_accuracy = accuracy_score(y_test, combi_pred)
```

```
# Оценка нелинейной модели
ria_pred = ria_model.predict(X_test_np)
ria_pred = np.round(ria_pred).astype(int)  # Округление и приведение к целому числу
ria_accuracy = accuracy_score(y_test, ria_pred)
```

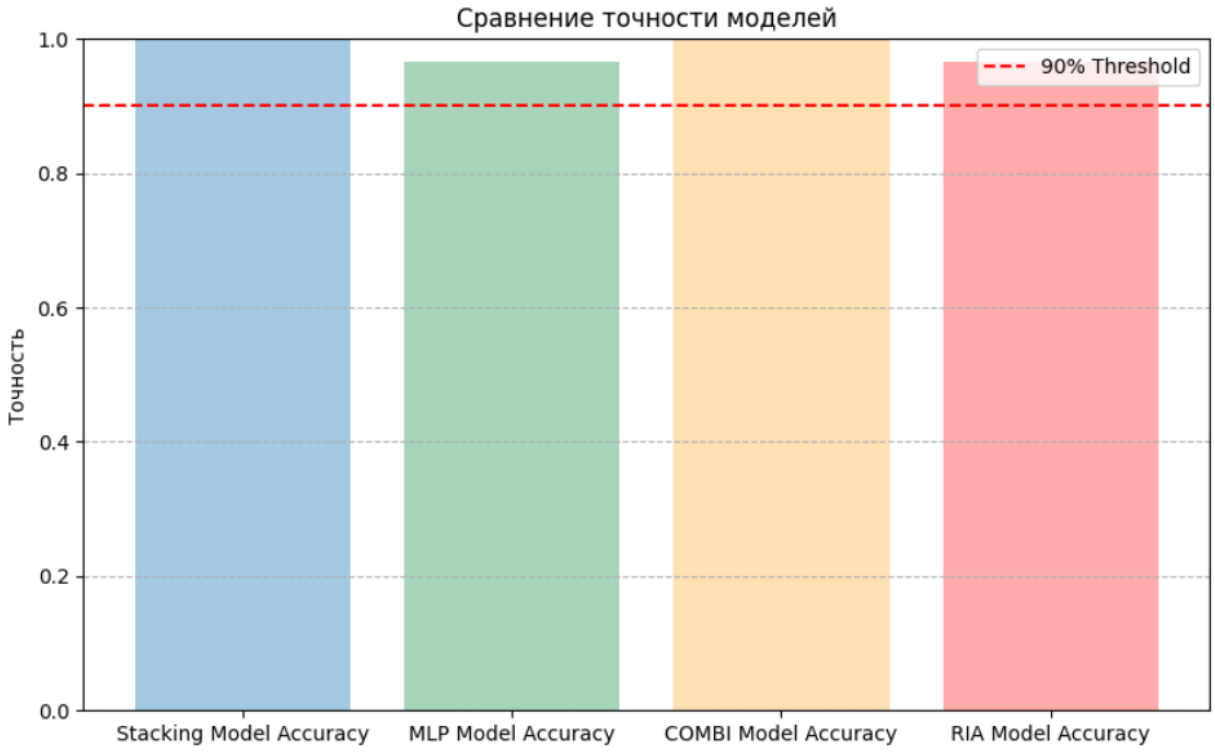
```
# Сравнение качества моделей
results = {
    "Stacking Model Accuracy": stacking_accuracy,
    "MLP Model Accuracy": mlp_accuracy,
    "COMBI Model Accuracy": combi_accuracy,
    "RIA Model Accuracy": ria_accuracy
}
```

```
results
```

```
{'Stacking Model Accuracy': 1.0,
 'MLP Model Accuracy': 0.9666666666666667,
 'COMBI Model Accuracy': 1.0,
 'RIA Model Accuracy': 0.9666666666666667}
```

```python
# Создание графика
plt.figure(figsize=(10, 6))
plt.bar(results.keys(), results.values(), color=['#A4C8E1', '#A8D5BA', '#FFE0B2', '#FFABAB'])
plt.ylabel('Точность')
plt.title('Сравнение точности моделей')
plt.ylim(0, 1)  # Ограничение по оси Y от 0 до 1
plt.axhline(y=0.90, color='r', linestyle='--', label='90% Threshold')  # Добавление пороговой линии
plt.legend()
plt.grid(axis='y', linestyle='--')

# Показать график
plt.show()
```



5. Вывод

В ходе лабораторной работы изучили ансамбли моделей машинного обучения.