Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №6

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:

   - 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.

Ход выполнения:

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import StackingClassifier, RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from gmdh import Combi, Mia
from sklearn.metrics import accuracy_score
```

```python
# Загрузка данных
df = pd.read_csv("train.csv")


df = df.drop(columns=["PassengerId", "Name", "Ticket", "Cabin"])
df["Age"] = df["Age"].fillna(df["Age"].median())
df["Embarked"] = df["Embarked"].fillna(df["Embarked"].mode()[0])

# Преобразование категориальных признаков с помощью LabelEncoder
label_encoders = {}
for col in ["Sex", "Embarked"]:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

```python
X = df.drop(columns=["Survived"])
y = df["Survived"]
```

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

Stacking Classifier (Стекинг)

В вашем примере используются два базовых классификатора (случайный лес и градиентный бустинг), а их прогнозы передаются в логистическую регрессию, которая делает финальное предсказание.

```python
base_learners = [
    ("rf", RandomForestClassifier(n_estimators=100, random_state=42)),
    ("gb", GradientBoostingClassifier(n_estimators=100, random_state=42))
]
stacking_model = StackingClassifier(estimators=base_learners, final_estimator=LogisticRegression())
stacking_model.fit(X_train, y_train)
y_pred_stacking = stacking_model.predict(X_test)
stacking_accuracy = accuracy_score(y_test, y_pred_stacking)
print(f"Stacking Accuracy: {stacking_accuracy:.4f}")
```

Python

```
Stacking Accuracy: 0.8101
```

MLPClassifier (Многослойный персептрон)

Каждое предсказание проходит через несколько слоев нейронов, каждый из которых обучается на предыдущих ошибках.

```python
mlp_model = MLPClassifier(hidden_layer_sizes=(50, 30), max_iter=500, random_state=42)
mlp_model.fit(X_train, y_train)
y_pred_mlp = mlp_model.predict(X_test)
mlp_accuracy = accuracy_score(y_test, y_pred_mlp)
print(f"MLP Accuracy: {mlp_accuracy:.4f}")
```

Python

```
MLP Accuracy: 0.7877
```

```python
X_train_np = X_train.to_numpy(dtype=float)  # Преобразуем в numpy массив
X_test_np = X_test.to_numpy(dtype=float)
y_train_np = y_train.to_numpy(dtype=float).ravel()  # Делаем одномерным
y_test_np = y_test.to_numpy(dtype=float).ravel()
```

Python

```
<class 'numpy.ndarray'> (712, 7)
<class 'numpy.ndarray'> (712,)
```

. Combi (линейный метод GMDH) Описание: Combi — это линейный метод из семейства GMDH (генеративных моделей данных), который строит модель, используя линейные комбинации признаков.

Как работает: Метод создает линейные модели для решения задачи классификации или регрессии, минимизируя ошибку на каждом шаге.

```python
gmdh_linear = Combi()
gmdh_linear.fit(X_train_np, y_train_np)
y_pred_gmdh_linear = gmdh_linear.predict(X_test).round().astype(int)
gmdh_linear_accuracy = accuracy_score(y_test_np, y_pred_gmdh_linear)
print(f"GMDH Linear (COMBI) Accuracy: {gmdh_linear_accuracy:.4f}")
```

Python

```
GMDH Linear (COMBI) Accuracy: 0.7765
```

Mia (нелинейный метод GMDH)

Описание: Mia — это нелинейная модель GMDH, использующая более сложные структуры для построения зависимости между признаками.

Как работает: В отличие от линейных методов, Mia создает более гибкие модели, которые могут учитывать нелинейные зависимости в данных.

```python
gmdh_nonlinear = Mia()
gmdh_nonlinear.fit(X_train_np, y_train_np)
y_pred_gmdh_nonlinear = gmdh_nonlinear.predict(X_test).round().astype(int)
gmdh_nonlinear_accuracy = accuracy_score(y_test_np, y_pred_gmdh_nonlinear)
print(f"GMDH Nonlinear (MIA) Accuracy: {gmdh_nonlinear_accuracy:.4f}")
```

```
GMDH Nonlinear (MIA) Accuracy: 0.7821
```