Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления» – ИУ5

Факультет «Радиотехнический» – РТ5

**Отчёт по лабораторной работе №5 по курсу**

**Технологии машинного обучения**

6

(количество листов)

|  |  |  |
| --- | --- | --- |
| Исполнитель |  |  |
| студент группы РТ5-61б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Нижаметдинов М. Ш. |
|  |  | “\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  |  |  |
| Проверил |  |  |
| Преподаватель кафедры ИУ5 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Гапанюк Ю. Е. |
|  |  | “\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2023 г. |

Москва, 2023 г.

**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:

o одну из моделей группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);

o одну из моделей группы бустинга;

o одну из моделей группы стекинга.

5. Дополнительно к указанным моделям обучите еще две модели:

o Модель многослойного персептрона. По желанию, вместо библиотеки scikit-learn возможно использование библиотек TensorFlow, PyTorch или других аналогичных библиотек.

o Модель МГУА с использованием библиотеки - https://github.com/kvoyager/GmdhPy (или аналогичных библиотек). Найдите такие параметры запуска модели, при которых она будет по крайней мере не хуже, чем одна из предыдущих ансамблевых моделей.

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

**Набор данных**

<https://scikit-learn.org/stable/datasets/toy_dataset.html#wine-recognition-dataset>

**Исходный текст проекта**

## 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:

1. одну из моделей группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);

2. одну из моделей группы бустинга;

3. одну из моделей группы стекинга.

5. (+1 балл на экзамене) Дополнительно к указанным моделям обучите еще две модели:

1. Модель многослойного персептрона. По желанию, вместо библиотеки scikit-learn возможно использование библиотек TensorFlow, PyTorch или других аналогичных библиотек.

2. Модель МГУА с использованием библиотеки - https://github.com/kvoyager/GmdhPy (или аналогичных библиотек). Найдите такие параметры запуска модели, при которых она будет по крайней мере не хуже, чем одна из предыдущих ансамблевых моделей.

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

## Ход работы

### Выбор и загрузка датасета

# %matplotlib inline

# sns.set(style="ticks")

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

# import gmdhpy

from sklearn.datasets import \*

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn import svm, tree

from sklearn.tree import DecisionTreeClassifier

# from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import BaggingClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import StackingClassifier

from sklearn.neural\_network import MLPClassifier

# from heamy.estimator import Classifier

# from heamy.pipeline import ModelsPipeline

# from heamy.dataset import Dataset

from sklearn.metrics import accuracy\_score

# from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from operator import itemgetter

def make\_dataframe(ds\_function):

ds = ds\_function()

df = pd.DataFrame(data= np.c\_[ds['data'], ds['target']],

columns= list(ds['feature\_names']) + ['target'])

return df

wine = load\_wine()

df = make\_dataframe(load\_wine)

# Первые 5 строк датасета

df.head()

df.dtypes

# Проверим наличие пустых значений

# Цикл по колонкам датасета

for col in df.columns:

# Количество пустых значений - все значения заполнены

temp\_null\_count = df[df[col].isnull()].shape[0]

print('{} - {}'.format(col, temp\_null\_count))

### Разделение на тестовую и обучающую выборки

y = df['target']

x = df.drop('target', axis = 1)

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(x)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(scaled\_data, y, test\_size = 0.4, random\_state = 0)

print(f"Обучающая выборка:\n{x\_train, y\_train}")

print(f"Тестовая выборка:\n{x\_test, y\_test}")

### Бэггинг

# Обучим классификатор на 5 деревьях

bc = BaggingClassifier(n\_estimators=5, oob\_score=True, random\_state=10)

bc\_prediction = bc.fit(x\_train, y\_train).predict(x\_test)

### Градиентный бустинг

gb = GradientBoostingClassifier(random\_state=0)

gb\_prediction = gb.fit(x\_train, y\_train).predict(x\_test)

### Стекинг

# Качество отдельных моделей

def val\_mae(model):

st\_prediction = model.fit(x\_train, y\_train).predict(x\_test)

# y\_pred = model.predict(boston\_X\_test)

# result = mean\_absolute\_error(boston\_y\_test, y\_pred)

print(model)

print('Accuracy score={}'.format(accuracy\_score(y\_test, st\_prediction)))

# Точность на отдельных моделях

for model in [

DecisionTreeClassifier(random\_state=0),

svm.SVC(random\_state=0),

LogisticRegression(random\_state=0)

]:

val\_mae(model)

print('==========================')

print()

# Первый уровень - две модели: дерево и метод опорных векторов

# Второй уровень: логистическая регрессия

estimators = [

('dt', DecisionTreeClassifier(random\_state=0)),

('svc', svm.SVC(random\_state=0))

]

sc = StackingClassifier(

estimators=estimators, final\_estimator=LogisticRegression()

)

sc\_prediction = sc.fit(x\_train, y\_train).predict(x\_test)

### Многослойный персептрон

mlp = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden\_layer\_sizes=(5, 2), random\_state=1)

mlp\_prediction = mlp.fit(x\_train, y\_train).predict(x\_test)

### Оценка качества решений

print("Бэггинг: ", accuracy\_score(y\_test, bc\_prediction))

print("Градиентный бустинг: ", accuracy\_score(y\_test, gb\_prediction))

print("Стекинг (дерево и метод опорных векторов + логистическая регрессия): ", accuracy\_score(y\_test, sc\_prediction))

print("Многослойный персептрон: ", accuracy\_score(y\_test, mlp\_prediction))

print("Бэггинг: ", accuracy\_score(y\_test, bc\_prediction))

cm = confusion\_matrix(y\_test, bc\_prediction, labels=np.unique(df.target), normalize='true')

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=np.unique(df.target))

disp.plot()

print("Градиентный бустинг: ", accuracy\_score(y\_test, gb\_prediction))

cm = confusion\_matrix(y\_test, gb\_prediction, labels=np.unique(df.target), normalize='true')

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=np.unique(df.target))

disp.plot()

print("Стекинг (дерево и метод опорных векторов + логистическая регрессия): ", accuracy\_score(y\_test, sc\_prediction))

cm = confusion\_matrix(y\_test, sc\_prediction, labels=np.unique(df.target), normalize='true')

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=np.unique(df.target))

disp.plot()

print("Многослойный персептрон: ", accuracy\_score(y\_test, mlp\_prediction))

cm = confusion\_matrix(y\_test, mlp\_prediction, labels=np.unique(df.target), normalize='true')

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=np.unique(df.target))

disp.plot()