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:
   - одну из моделей группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);
   - одну из моделей группы бустинга;
   - одну из моделей группы стекинга.
5. Дополнительно к указанным моделям обучите еще две модели:
   - Модель многослойного персептрона. По желанию, вместо библиотеки scikit-learn возможно использование библиотек TensorFlow, PyTorch или других аналогичных библиотек.
   - 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()
```