Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Теория машинного обучения»

Отчет по лабораторной работе №5

Выполнил:
студент группы ИУ5-64
Кузьмин Роман

Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Подпись и дата:

Москва, 2022 г.

Описание задания

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

Текст программы и её результаты

```
from sklearn.datasets import fetch_covtype
import pandas as pd
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np
import matplotlib.pyplot as plt
import warnings
%matplotlib inline

data = fetch_covtype(as_frame=True)['data']
data["type"] = fetch_covtype(as_frame=True)['target']
data.head()
```

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Hori
0	2596.0	51.0	3.0	258.0	0.0	
1	2590.0	56.0	2.0	212.0	-6.0	
2	2804.0	139.0	9.0	268.0	65.0	
3	2785.0	155.0	18.0	242.0	118.0	
4	2595.0	45.0	2.0	153.0	-1.0	

5 rows × 55 columns

Пропусков нет

```
data.isna().sum()
```

```
Elevation                             0
Aspect                                0
Slope                                 0
Horizontal_Distance_To_Hydrology      0
Vertical_Distance_To_Hydrology        0
Horizontal_Distance_To_Roadways       0
Hillshade_9am                         0
Hillshade_Noon                        0
Hillshade_3pm                         0
Horizontal_Distance_To_Fire_Points    0
Wilderness_Area_0                     0
Wilderness_Area_1                     0
Wilderness_Area_2                     0
Wilderness_Area_3                     0
Soil_Type_0                           0
Soil_Type_1                           0
Soil_Type_2                           0
Soil_Type_3                           0
Soil_Type_4                           0
Soil_Type_5                           0
Soil_Type_6                           0
Soil_Type_7                           0
Soil_Type_8                           0
Soil_Type_9                           0
Soil_Type_10                          0
Soil_Type_11                          0
Soil_Type_12                          0
Soil_Type_13                          0
Soil_Type_14                          0
Soil_Type_15                          0
Soil_Type_16                          0
Soil_Type_17                          0
Soil_Type_18                          0
```

```
[ ] X_train, X_test, y_train, y_test = train_test_split(data.drop('type', axis=1), data['type'], test_size=0.3, random_state=42)
```

```
[ ] tree1 = RandomForestClassifier(n_estimators=5, oob_score=True, random_state=10)
    tree1.fit(X_train, y_train)
    pred1 = tree1.predict(X_test)
    accuracy_score(y_test, pred1)
```

```
0.9253660271709198
```

```
[ ] boost1 = AdaBoostClassifier(n_estimators=5, algorithm='SAMME', random_state=10)
    boost1.fit(X_train, y_train)
    pred2 = boost1.predict(X_test)
    accuracy_score(y_test, pred2)
```

```
0.633565494767762
```

```
classifier = StackingClassifier(
    [
        ('lr', LogisticRegression()),
        ('dt', DecisionTreeClassifier())
    ],
LogisticRegression())
classifier.fit(X_train, y_train)
```

```
[ ]  classifier = StackingClassifier(
         [
             ('lr', LogisticRegression()),
             ('dt', DecisionTreeClassifier())
         ],
     LogisticRegression())
     classifier.fit(X_train, y_train)
```

```
     StackingClassifier(estimators=[('lr', LogisticRegression()),
                                    ('dt', DecisionTreeClassifier())],
                        final_estimator=LogisticRegression())
```

```
[ ]  pred3 = classifier.predict(X_test)
     accuracy_score(y_test, pred3)
```

```
     0.9339372590416744
```