Московский Государственный Технический Университет
им. Н.Э. Баумана

Отчет по лабораторной работе №6

по курсу
Технологии Машинного Обучения

Выполнил:
Муравьев О.М.
ИУ5-62

_____

Проверил:
Гапанюк Ю.Е.

_____

Москва, 2019

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 две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.
5. Произведите для каждой модели подбор значений одного гиперпараметра. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.
6. Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните качество полученных моделей с качеством моделей, полученных в пункте 4.

Код и результаты выполнения

1. Подключим библиотеки:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

2. Подготовим данные

Разделим выборку на тренировочную и тестовую

```
X_train, X_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=42)
```

3. Обучим и проверим данные используя метрики классификации

Случайный лес

```
random_forest = RandomForestClassifier(
    n_estimators=10, max_depth=1,
    random_state=0).fit(X_train, y_train)
```

```
res_RF = random_forest.predict(X_test)
print(accuracy_score(y_test, res_RF))
print(precision_score(y_test, res_RF))
```

```
0.7375
0.90625
```

Градиентный бустинг

```
gradient_boosting = GradientBoostingClassifier(
    n_estimators=10, max_depth=10,
    learning_rate=0.01).fit(X_train, y_train)
```

```
res_GB = gradient_boosting.predict(X_test)
print(accuracy_score(y_test, res_GB))
print(precision_score(y_test, res_GB))
```

```
0.7125
0.7222222222222222
```

4. Подберем гиперпараметры, обучим модели используя их

Случайный лес

```
parameters_random_forest = {'n_estimators':[1, 3, 5, 7, 10],
                            'max_depth':[1, 3, 5, 7, 10],
                            'random_state':[0, 2, 4, 6, 8, 10]}
best_random_forest = GridSearchCV(RandomForestClassifier(),
                                  parameters_random_forest, cv=3,
                                  scoring='accuracy')
best_random_forest.fit(X_train, y_train)
```

```
best_random_forest.best_params_
```

```
{'max_depth': 3, 'n_estimators': 10, 'random_state': 10}
```

```
new_RF = RandomForestClassifier(n_estimators=5,
                                max_depth=3,
                                random_state=10).fit(X_train, y_train)
```

```
new_res_RF = new_RF.predict(X_test)
print(accuracy_score(y_test, new_res_RF))
print(precision_score(y_test, new_res_RF))
```

```
0.775
0.8372093023255814
```

Градиентный бустинг

```
parameters_gradient_boosting = {'n_estimators':[1, 3, 5, 7, 10],
                            'max_depth':[1, 3, 5, 7, 10],
                            'learning_rate':[0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025]}
best_gradient_boosting = GridSearchCV(GradientBoostingClassifier(), parameters_gradient_boosting,
                                  cv=3, scoring='accuracy')
best_gradient_boosting.fit(X_train, y_train)
```

```
best_gradient_boosting.best_params_
```

```
{'learning_rate': 0.025, 'max_depth': 1, 'n_estimators': 10}
```

```
new_GB = GradientBoostingClassifier(n_estimators=10,
                                    max_depth=3,
                                    learning_rate=0.025).fit(X_train, y_train)
```

```
new_res_GB = new_GB.predict(X_test)
print(accuracy_score(y_test, new_res_GB))
print(precision_score(y_test, new_res_GB))
```

```
0.75
0.8461538461538461
```