Московский Государственный Технический Университет
им. Н.Э. Баумана

Отчет по лабораторной работе №5

по курсу
Технологии Машинного Обучения

Выполнила:
Костян Алина
ИУ5-63

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 1) одну из линейных моделей, 2) SVM и 3) дерево решений. Оцените качество моделей с помощью трех подходящих для задачи метрик. Сравните качество полученных моделей.
5. Произведите для каждой модели подбор одного гиперпараметра с использованием GridSearchCV и кросс-валидации.
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
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

2. Подготовим данные

Проверим наличие и количество пропущенных значений

```
data.isnull().sum()
```

```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

Проверим наличие категориальных данных

```
cats = [col for col in data.columns if
        data[col].dtype=="object"]
print(len(cats))
```

```
0
```

Разделим данные на тренировочную и тестовую выборку

```
X_train, X_test, y_train, y_test = train_test_split(
    data, data['Research'], test_size=0.2, random_state=42)
```

3. Обучим модели и проверим метриками

Метод стохастического градиентного спуска

```
sgd = SGDClassifier().fit(X_train, y_train)
```

```
res_SGD = sgd.predict(X_test)
print(accuracy_score(y_test, res_SGD))
print(precision_score(y_test, res_SGD))
print(recall_score(y_test, res_SGD))
```

```
0.4125
0.0
0.0
```

Метод опорных векторов

```
sv = SVC(gamma='auto').fit(X_train, y_train)
```

```
res_SVC = sv.predict(X_test)
print(accuracy_score(y_test, res_SVC))
print(precision_score(y_test, res_SVC))
print(recall_score(y_test, res_SVC))
```

```
0.6125
0.6212121212121212
0.8723404255319149
```

Деревья решений

```
DT = DecisionTreeClassifier(
    random_state=1,
    max_depth=0.75).fit(X_train, y_train)
```

```
res_DT = DT.predict(X_test)
print(accuracy_score(y_test, res_DT))
print(precision_score(y_test, res_DT))
print(recall_score(y_test, res_DT))
```

```
0.5875
0.5875
1.0
```

4. Обучим кросс валидацией

Обучим модели на кросс валидации

```
scores_sgd = cross_val_score(SGDClassifier(),
                             X_train, y_train,
                             cv=2)
np.mean(scores_sgd)
```

```
0.49062500000000003
```

```
scores_svc = cross_val_score(SVC(gamma='auto'),
                             X_train, y_train, cv=2)
np.mean(scores_svc)
```

```
0.6156250000000001
```

```
scores_dt = cross_val_score(DecisionTreeClassifier(),
                            X_train, y_train, cv=2)
np.mean(scores_dt)
```

```
1.0
```

5. Подберем гиперпараметры и обучим модели, используя их

Стохастический градиентный спуск

```
parameters = {'alpha':[0.5,0.4,0.3,0.2,0.1]}
clf_gs_sgd = GridSearchCV(SGDClassifier(),
                          parameters, cv=2,
                          scoring='accuracy')
clf_gs_sgd.fit(X_train, y_train)
```

```
clf_gs_sgd.best_params_
```

```
{'alpha': 0.4}
```

```
sgd_new = SGDClassifier(
    alpha=0.5).fit(X_train,
                   y_train)
```

```
res_sgd_new = sgd_new.predict(X_test)
print(accuracy_score(y_test, res_sgd_new))
print(precision_score(y_test, res_sgd_new))
print(recall_score(y_test, res_sgd_new))
```

```
0.525
0.5633802816901409
0.851063829787234
```

Метод опорных векторов

```
parameters = {'gamma':[0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1]}
clf_gs_svm_svc = GridSearchCV(SVC(), parameters, cv=2, scoring='accuracy')
clf_gs_svm_svc.fit(X_train, y_train)
```

```
clf_gs_svm_svc.best_params_
```

```
{'gamma': 0.1}
```

```
svm_svc_new = SVC(gamma=0.1).fit(X_train,
                                 y_train)
```

```
res_svc_new = svm_svc_new.predict(X_test)
print(accuracy_score(y_test, res_svc_new))
print(precision_score(y_test, res_svc_new))
print(recall_score(y_test, res_svc_new))
```

```
0.6125
0.625
0.851063829787234
```

Деревья решений

```
parameters = {'min_impurity_decrease':[0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1]}
clf_gs_decision_tree = GridSearchCV(DecisionTreeClassifier(), parameters, cv=2, scoring='accuracy')
clf_gs_decision_tree.fit(X_train, y_train)
```

```
clf_gs_decision_tree.best_params_
```

```
{'min_impurity_decrease': 0.4}
```

```
decision_tree_new = DecisionTreeClassifier(
    random_state=1,
    min_impurity_decrease=0.4,
    max_depth=0.75).fit(X_train, y_train)
```

```
res_dt_new = decision_tree_new.predict(X_test)
print(accuracy_score(y_test, res_dt_new))
print(precision_score(y_test, res_dt_new))
print(recall_score(y_test, res_dt_new))
```

```
0.5875
0.5875
1.0
```