Московский Государственный технический университет имени Н. Э. Баумана

Лабораторная работа № 5 по курсу: «Технология машинного обучения»

Работу выполнил студент группы ИУ5-63

Федорова Антонина_____

Работу 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

Текст программы с примерами выполнения программы:

```
#Импорт библиотек
from google.colab import drive
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
import seaborn as sns
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style=«ticks")
data.head(5)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51

```
cols_x = ['fixed acidity', 'citric acid', 'sulphates', 'alcohol']
# целевой признак
col_y = 'quality'
scaler = StandardScaler()
X = scaler.fit_transform(data[cols_x])
#разделение данных  на тестовую и обучающую выборки
X_train, X_test, y_train, y_test = train_test_split(X, data[col_y], test_size = 0.5, random_state
= 11)
N_train, _ = X_train.shape
N_test,  _ = X_test.shape
print (N_train, N_test)
grad = SGDClassifier().fit(X_train, y_train)
target_grad = grad.predict(X_test)
def convert_target_to_binary(array:np.ndarray, target:int) -> np.ndarray:
    # Если целевой признак совпадает с указанным, то 1 иначе 0
    res = [1 if x==target else 0 for x in array]
    return res
# Если целевой признак == 6,
# то будем считать этот случай 1 в бинарном признаке
bin_y_test = convert_target_to_binary(y_test, 6)
list(zip(y_test, bin_y_test))[0:15]
# Конвертация предсказанных признаков
bin_target_grad = convert_target_to_binary(target_grad, 6)
```

```
ac_grad = accuracy_score(y_test, target_grad)
bas_grad = balanced_accuracy_score(bin_y_test, bin_target_grad)
rec_grad = recall_score(bin_y_test, bin_target_grad)
print('''accuracy_score: {0}
balanced_accuracy_score: {1}
recall_score: {2}'''.format(ac_grad, bas_grad, rec_grad))
```

```
accuracy_score: 0.52
balanced_accuracy_score: 0.5708821930573891
recall_score: 0.42024539877300615
```

Обучение с помощью SVM

```
[ ]  svmcl = SVC(kernel='rbf', gamma=0.5, C=1.0).fit(X_train, y_train)
```

```
[ ]  target_svm = svmcl.predict(X_test)
```

Оценка качества метода опорных векторов

```
[ ]  # Конвертация предсказанных признаков
     bin_target_svm = convert_target_to_binary(target_svm, 6)
```

```
[ ]  ac_svm = accuracy_score(y_test, target_svm)
     bas_svm = balanced_accuracy_score(bin_y_test, bin_target_svm)
     rec_svm = recall_score(bin_y_test, bin_target_svm)
     print('''accuracy_score: {0}
     balanced_accuracy_score: {1}
     recall_score: {2}'''.format(ac_svm, bas_svm, rec_svm))
```

```
accuracy_score: 0.60125
balanced_accuracy_score: 0.6309828893893505
recall_score: 0.5552147239263804
```

Обучение с помощью деревьев принятия решений

```
[ ]  dtcl = DecisionTreeClassifier(random_state=1, max_depth = 20).fit(X_train, y_train)
```

```
[ ]  dtcl
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=20,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=1, splitter='best')
```

```
[ ]  target_dtc = dtcl.predict(X_test)
```

Оценка качества классификации методом деревьев принятия решений

```
[ ]  # Конвертация предсказанных признаков
     bin_target_dtc = convert_target_to_binary(target_dtc, 6)
```

```
[ ]  ac_dtc = accuracy_score(y_test, target_dtc)
     bas_dtc = balanced_accuracy_score(bin_y_test, bin_target_dtc)
     rec_dtc = recall_score(bin_y_test, bin_target_dtc)
     print('''accuracy_score: {0}
     balanced_accuracy_score: {1}
     recall_score: {2}'''.format(ac_dtc, bas_dtc, rec_dtc))
```

```
accuracy_score: 0.545
balanced_accuracy_score: 0.6081644275322927
recall_score: 0.5306748466257669
```

Подбор гиперпараметра с использованием GridSearchCV и кросс-валидации

Для градиентного спуска

```
[ ]  n_range = np.array(np.arange(0.0001,1,0.05))
     tuned_parameters = [{'alpha': n_range}]
     tuned_parameters
```

```
[{'alpha': array([1.000e-04, 5.010e-02, 1.001e-01, 1.501e-01, 2.001e-01, 2.501e-01,
         3.001e-01, 3.501e-01, 4.001e-01, 4.501e-01, 5.001e-01, 5.501e-01,
         6.001e-01, 6.501e-01, 7.001e-01, 7.501e-01, 8.001e-01, 8.501e-01,
         9.001e-01, 9.501e-01])}]
```

```
[ ]  grad_gs = GridSearchCV(SGDClassifier(), tuned_parameters, cv=KFold(n_splits=10), scoring='accuracy')
     grad_gs.fit(data[cols_x], data[col_y])
```

```
[ ]  grad_gs.best_params_
```

```
{'alpha': 0.10010000000000001}
```

Обучение с подобранным параметром

```
[ ]  grad_gs.best_estimator_.fit(X_train, y_train)
     target_grad_gs = grad_gs.best_estimator_.predict(X_test)
```

Проверка качества модели

```
[ ]  # Конвертация предсказанных признаков
     bin_target_grad_gs = convert_target_to_binary(target_grad_gs, 6)
```

```
[ ]  #Новая модель
     ac_grad_gs = accuracy_score(y_test, target_grad_gs)
     bas_grad_gs = balanced_accuracy_score(bin_y_test, bin_target_grad_gs)
     rec_grad_gs = recall_score(bin_y_test, bin_target_grad_gs)
     print('''accuracy_score: {0}
     balanced_accuracy_score: {1}
     recall_score: {2}'''.format(ac_grad_gs, bas_grad_gs, rec_grad_gs))
```

```
accuracy_score: 0.4975
balanced_accuracy_score: 0.5237374129585048
recall_score: 0.2331288343558282
```

```
[ ]  #Старая модель
     ac_grad = accuracy_score(y_test, target_grad)
     bas_grad = balanced_accuracy_score(bin_y_test, bin_target_grad)
     rec_grad = recall_score(bin_y_test, bin_target_grad)
     print('''accuracy_score: {0}
     balanced_accuracy_score: {1}
     recall_score: {2}'''.format(ac_grad, bas_grad, rec_grad))
```

```
accuracy_score: 0.52
balanced_accuracy_score: 0.5708821930573891
recall_score: 0.42024539877300615
```

Для SVM

```
[ ]  tuned_parameters_svm = [{'gamma': n_range}]
```

```
[ ]  svm_gs = GridSearchCV(SVC(), tuned_parameters_svm, cv=KFold(n_splits=5), scoring='accuracy')
     svm_gs.fit(data[cols_x], data[col_y])
```

```
[ ]  svm_gs.best_params_
```

```
{'gamma': 0.050100000000000006}
```

Обучение с подобранным параметром

```
[ ]  svm_gs.best_estimator_.fit(X_train, y_train)
     target_svm_gs = svm_gs.best_estimator_.predict(X_test)
```

Проверка качества модели

```
[ ]  # Конвертация предсказанных признаков
     bin_target_svm_gs = convert_target_to_binary(target_svm_gs, 6)
```

```
[ ]  #Новая модель
     ac_svm_gs = accuracy_score(y_test, target_svm_gs)
     bas_svm_gs = balanced_accuracy_score(bin_y_test, bin_target_svm_gs)
     rec_svm_gs = recall_score(bin_y_test, bin_target_svm_gs)
     print('''accuracy_score: {0}
     balanced_accuracy_score: {1}
     recall_score: {2}'''.format(ac_svm_gs, bas_svm_gs, rec_svm_gs))
```

```
accuracy_score: 0.5725
balanced_accuracy_score: 0.6037444021640651
recall_score: 0.5766871165644172
```

```
[ ]  #Старая модель
     ac_svm = accuracy_score(y_test, target_svm)
     bas_svm = balanced_accuracy_score(bin_y_test, bin_target_svm)
     rec_svm = recall_score(bin_y_test, bin_target_svm)
     print('''accuracy_score: {0}
     balanced_accuracy_score: {1}
     recall_score: {2}'''.format(ac_svm, bas_svm, rec_svm))
```

```
accuracy_score: 0.60125
balanced_accuracy_score: 0.6309828893893505
recall_score: 0.5552147239263804
```

Для деревьев принятия решений

```
[ ]  n_range_dtc = np.array(range(1,100,10))
     tuned_parameters_dtc = [{'max_depth': n_range_dtc}]
     tuned_parameters_dtc
```

```
[{'max_depth': array([ 1, 11, 21, 31, 41, 51, 61, 71, 81, 91])}]
```

```
[ ]  dtc_gs = GridSearchCV(DecisionTreeClassifier(), tuned_parameters_dtc, cv=KFold(n_splits=10), scoring='accuracy')
     dtc_gs.fit(X_train, y_train)
```

```
[ ]  dtc_gs.best_params_
```

```
{'max_depth': 1}
```

Обучение с подобранным параметром

```
[ ]  dtc_gs.best_estimator_.fit(X_train, y_train)
     target_dtc_gs = dtc_gs.best_estimator_.predict(X_test)
```

Проверка качества модели

```
[ ]  # Конвертация предсказанных признаков
     bin_target_dtc_gs = convert_target_to_binary(target_dtc_gs, 6)
```

```
[ ]  #Новая модель
     ac_dtc_gs = accuracy_score(y_test, target_dtc_gs)
     bas_dtc_gs = balanced_accuracy_score(bin_y_test, bin_target_dtc_gs)
     rec_dtc_gs = recall_score(bin_y_test, bin_target_dtc_gs)
     print('''accuracy_score: {0}
     balanced_accuracy_score: {1}
     recall_score: {2}'''.format(ac_dtc_gs, bas_dtc_gs, rec_dtc_gs))
```

```
accuracy_score: 0.55125
balanced_accuracy_score: 0.5797869586601434
recall_score: 0.49079754601226994
```

```
[ ]  #Старая модель
     ac_dtc = accuracy_score(y_test, target_dtc)
     bas_dtc = balanced_accuracy_score(bin_y_test, bin_target_dtc)
     rec_dtc = recall_score(bin_y_test, bin_target_dtc)
     print('''accuracy_score: {0}
     balanced_accuracy_score: {1}
     recall_score: {2}'''.format(ac_dtc, bas_dtc, rec_dtc))
```

```
accuracy_score: 0.545
balanced_accuracy_score: 0.6081644275322927
recall_score: 0.5306748466257669
```

```
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
```

```
dot_data = StringIO()
```

```
export_graphviz(dtcl, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True)
```

```
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```