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

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Технологии машинного обучения»

Отчет по лабораторной работе №4

Выполнила:		Проверил:
студентка группы ИУ5-61		преподаватель каф. ИУ5
Абросимова Надежда		Гапанюк Ю.Е.
Подпись и дата:		Подпись и дата:

г. Москва, 2019 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 модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью трех подходящих для задачи метрик.
5. Постройте модель и оцените качество модели с использованием кросс-валидации. Проведите эксперименты с тремя различными стратегиями кросс-валидации.
6. Произведите подбор гиперпараметра K с использованием GridSearchCV и кросс-валидации.
7. Повторите пункт 4 для найденного оптимального значения гиперпараметра K. Сравните качество полученной модели с качеством модели, полученной в пункте 4.
8. Постройте кривые обучения и валидации.

Текст программы

```
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.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score,
classification_report
 from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import learning_curve, validation_curve
from sklearn.model_selection import KFold, RepeatedKFold, ShuffleSplit,
StratifiedKFold
%matplotlib inline
sns.set(style="ticks")

data=pd.read_csv('heart.csv', sep=",")
data.dtypes
Out[4]:
age         int64
sex         int64
cp          int64
trestbps    int64
chol        int64
restecg     int64
thalach     int64
slope       int64
ca          int64
thal        int64
target      int64
dtype: object
data.isnull().sum()
```

```
Out[5]:
age         0
sex         0
cp          0
trestbps    0
chol        0
restecg     0
thalach     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Разделение выборки на обучающую и тестовую

```
X_train, X_test, y_train, y_test = train_test_split(
    data, data['target'], test_size= 0.2, random_state= 1)
# Размер обучающей выборки
X_train.shape, y_train.shape
Out[8]:
((242, 11), (242,))
In [9]:
# Размер тестовой выборки
X_test.shape, y_test.shape
Out[9]:
((61, 11), (61,))
```

Построим базовые модели на основе метода ближайших соседей

```
simple_knn = KNeighborsClassifier(n_neighbors=2)
simple_knn.fit(X_train, y_train)
target_1 = simple_knn.predict(X_test)
#оценка моделей с помощью метрик
accuracy_score(y_test, target_1), \
precision_score(y_test, target_1),  \
recall_score(y_test, target_1)
Out[13]:
(0.5737704918032787, 0.5675675675675675, 0.6774193548387096)
```

Построим модели с использованием кросс-валидации

```
#K-fold:работает в соответствии с определпением кросс-валидации
kfold = cross_val_score(KNeighborsClassifier(), data, data['target'],
cv=KFold(n_splits=5))
#ShuffleSplit: генерирует N случайных перемешиваний данных, в каждом перем
ешивании заданная доля помещается в тестовую выборку
shufflesplit = cross_val_score(KNeighborsClassifier(),
                        data, data['target'],
                        cv=ShuffleSplit(n_splits=5, test_size=0.2))
#StratifiedKFold: предоставляет индексы для разделения данных, вариант KFo
ld stratifiedkfold = cross_val_score(KNeighborsClassifier(), data, data['t
arget'], cv=StratifiedKFold(n_splits=5))
Подбор гиперпараметра K
n_range = np.array(range(1,10,1))
tuned_parameters = [{'n_neighbors': n_range}]
clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters,
```

```
                       cv=StratifiedKFold(n_splits=5), scoring='accuracy')
clf_gs.fit(X_train, y_train)
```

In [19]:

```
clf_gs.best_params_
```

Out[19]:

```
{'n_neighbors': 3}
```

Пункт 4 для найденного оптимального значения К

```
simple_knn2 = KNeighborsClassifier(n_neighbors=3)
simple_knn2.fit(X_train, y_train)

target_2 = simple_knn2.predict(X_test)
accuracy_score(y_test, target_2), \
precision_score(y_test, target_2),  \
recall_score(y_test, target_2)
```

Out[23]:

```
(0.5901639344262295, 0.575, 0.7419354838709677)
```

Построить кривые обучения и валидации

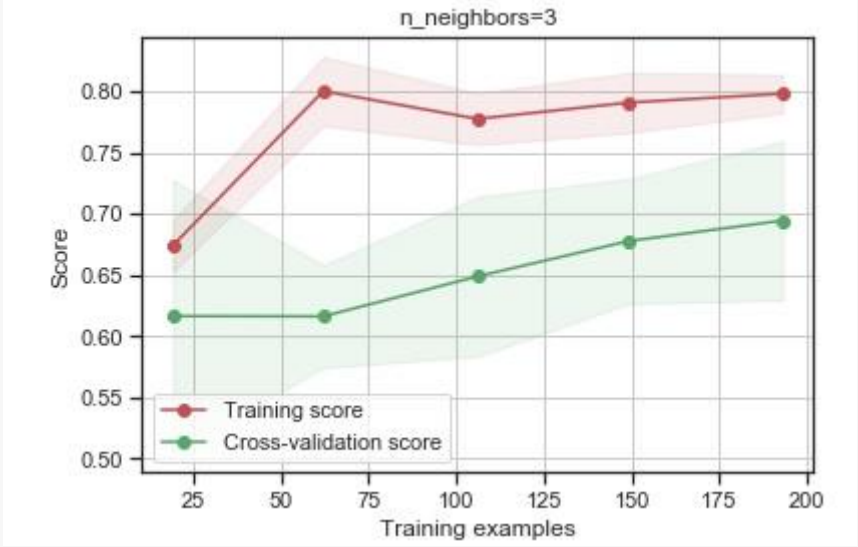
```
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=None, train_sizes=np.linspace(.1, 1.0, 5)):

    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color=
"g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")

    plt.legend(loc="best")
    return plt

plot_learning_curve(KNeighborsClassifier(n_neighbors=3), 'n_neighbors=3',
X_train, y_train, cv=StratifiedKFold(n_splits=5))
```

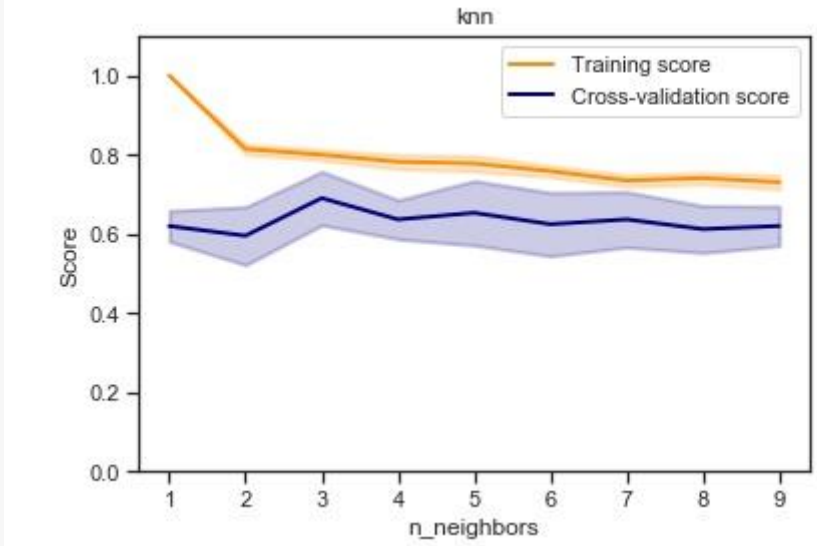



```
def plot_validation_curve(estimator, title, X, y,
                          param_name, param_range, cv,
                          scoring="accuracy"):

    train_scores, test_scores = validation_curve(
        estimator, X, y, param_name=param_name, param_range=param_range,
        cv=cv, scoring=scoring, n_jobs=1)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.title(title)
    plt.xlabel(param_name)
    plt.ylabel("Score")
    plt.ylim(0.0, 1.1)
    lw = 2
    plt.plot(param_range, train_scores_mean, label="Training score",
                 color="darkorange", lw=lw)
    plt.fill_between(param_range, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.2,
                     color="darkorange", lw=lw)
    plt.plot(param_range, test_scores_mean, label="Cross-validation score"
,
                 color="navy", lw=lw)
    plt.fill_between(param_range, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.2,
                     color="navy", lw=lw)
    plt.legend(loc="best")
    return plt

plot_validation_curve(KNeighborsClassifier(), 'knn',
                      X_train, y_train,
                      param_name='n_neighbors', param_range=n_range,
                      cv=StratifiedKFold(n_splits=5), scoring="accuracy")
```


knn
Training score
Cross-validation score
Score
n_neighbors
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
5
6
7
8
9