Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ**

Отчет по лабораторной работе № 4
«Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей»
по курсу “Технологии машинного обучения”

Исполнитель:
Студент группы ИУ5-63
Желанкина А.С.
_____ 21.03.2018

Москва, 2019

Задание лабораторной работы

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.

Экранные формы с текстом программы и примерами её выполнения

In [54]:
```
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")
```

In [55]:
```
data = pd.read_csv('heart.csv', sep=",")
data.head()
```

Out[55]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [56]:
```
data.dtypes
```

Out[56]:
```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
```

```
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

In [57]:
```
data.shape
```

Out[57]:
```
(303, 14)
```

In [58]:
```
data.isnull().sum()
```

Out[58]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [59]:
```
X_train, X_test, y_train, y_test = train_test_split(
    data, data['target'], test_size=0.2, random_state=1)
```

In [60]:
```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[60]:
```
((242, 14), (242,), (61, 14), (61,))
```

In [61]:
```
simple_knn = KNeighborsClassifier()
```

In [62]:
```
simple_knn.fit(X_train, y_train)
```

Out[62]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

In [63]:
```
target_1 = simple_knn.predict(X_test)
```

In [64]:
```
accuracy_score(y_test, target_1), \
precision_score(y_test, target_1),  \
recall_score(y_test, target_1)
```

Out[64]:
```
(0.5737704918032787, 0.5675675675675675, 0.6774193548387096)
```

In [65]:
```
kfold = cross_val_score(KNeighborsClassifier(),
                        data, data['target'],
                        cv=KFold(n_splits=5))
kfold
```

Out[65]:
```
array([0.47540984, 0.63934426, 0.68852459, 0.4       , 0.31666667])
```

In [66]:
```
shufflesplit = cross_val_score(KNeighborsClassifier(),
                        data, data['target'],
                        cv=ShuffleSplit(n_splits=5, test_size=0.2))
shufflesplit
```

Out[66]:
```
array([0.57377049, 0.72131148, 0.67213115, 0.62295082, 0.6557377 ])
```

In [67]:
```
stratifiedkfold = cross_val_score(KNeighborsClassifier(),
                        data, data['target'],
                        cv=StratifiedKFold(n_splits=5))
stratifiedkfold
```

Out[67]:
```
array([0.60655738, 0.6557377 , 0.57377049, 0.73333333, 0.65      ])
```

In [68]:
```
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)
```

```
C:\Anaconda\lib\site-packages\sklearn\model_selection\_search.py:841: DeprecationWarning: The default
of the `iid` parameter will change from True to False in version 0.22 and will be removed in 0.24. Th
is will change numeric results when test-set sizes are unequal.
  DeprecationWarning)
```

Out[68]:
```
GridSearchCV(cv=StratifiedKFold(n_splits=5, random_state=None, shuffle=False),
       error_score='raise-deprecating',
       estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
```

```
                    weights='uniform'),
                fit_params=None, iid='warn', n_jobs=None,
                param_grid=[{'n_neighbors': array([1, 2, 3, 4, 5, 6, 7, 8, 9])}],
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring='accuracy', verbose=0)
```

In [69]:
```
clf_gs.best_params_
```

Out[69]:
```
{'n_neighbors': 3}
```

In [70]:
```
simple_knn_best = KNeighborsClassifier(n_neighbors=3)
```

In [71]:
```
simple_knn_best.fit(X_train, y_train)
```

Out[71]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=3, p=2,
           weights='uniform')
```

In [72]:
```
target_2 = simple_knn_best.predict(X_test)
```

In [73]:
```
accuracy_score(y_test, target_2), \
precision_score(y_test, target_2),  \
recall_score(y_test, target_2)
```

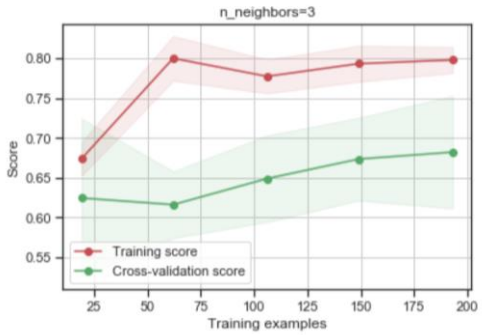
Out[73]:
```
(0.5737704918032787, 0.5609756097560976, 0.7419354838709677)
```

In [74]:
```
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=None, train_sizes=np.linspace(.1, 1.0, 5)):
    """
    Generate a simple plot of the test and training learning curve.

    Parameters
    ----------
    estimator : object type that implements the "fit" and "predict" methods
        An object of that type which is cloned for each validation.

    title : string
        Title for the chart.

    X : array-like, shape (n_samples, n_features)
        Training vector, where n_samples is the number of samples and
```

...

In [78]:
```
plot_learning_curve(KNeighborsClassifier(n_neighbors=3), 'n_neighbors=3',
                    X_train, y_train, cv=StratifiedKFold(n_splits=5))
```

Out[78]:
```
<module 'matplotlib.pyplot' from 'C:\\Anaconda\\lib\\site-packages\\matplotlib\\pyplot.py'>
```



In [76]:
```
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)
```

...

In [79]:
```
plot_validation_curve(KNeighborsClassifier(), 'knn',
                      X_train, y_train,
                      param_name='n_neighbors', param_range=n_range,
                      cv=StratifiedKFold(n_splits=5), scoring="accuracy")
```

Out[79]:
```
<module 'matplotlib.pyplot' from 'C:\\Anaconda\\lib\\site-packages\\matplotlib\\pyplot.py'>
```