Лабораторная работа № 4

Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей

Цель лабораторной работы: изучение сложных способов подготовки выборки и подбора гиперпараметров на примере метода ближайших соседей.

Выполнил: Ханмурзин Тагир ИУ5-64

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.

In [107]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")

from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier

from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.model_selection import KFold, RepeatedKFold, ShuffleSplit, StratifiedKFold
```

In [108]:
```
data = pd.read_csv('heart.csv', sep=",")
data.head(5)
```

Out[108]:

	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 [109]:
```
data.dtypes
```

Out[109]:
```
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 [110]:
```
data.shape
```

Out[110]:
```
(303, 14)
```

In [111]:
```
# Проверим наличие пустых значений
# Цикл по колонкам датасета
for col in data.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
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
```

In [112]:
```
# Заменяем значение string на числовое значение
#data['Gender'] = data['Gender'].map({'Female': 1, 'Male': 0})
```

In [113]:
```
#data.head(5)
```

In [114]:
```
x_learn, x_test, y_learn, y_test = train_test_split(
    data.loc[:, data.columns != 'target'], data['target'],
    test_size=0.2, random_state=1)
```

In [115]:
```
#Обучающая выборка
x_learn.shape, y_learn.shape
```

Out[115]:
```
((242, 13), (242,))
```

In [116]:
```
#Тестовая выборка
x_test.shape, y_test.shape
```

Out[116]:
```
((61, 13), (61,))
```

Обучите модель ближайших соседей для произвольно заданного гиперпараметра K.
Оцените качество модели с помощью трех подходящих для задачи метрик.

In [117]:
```
# 2 ближайших соседа
cl1_1 = KNeighborsClassifier(n_neighbors=2)
cl1_1.fit(x_learn, y_learn)
target1_1 = cl1_1.predict(x_test)

target1_1, accuracy_score(y_test, target1_1)
```

Out[117]:
```
(array([0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0,
        0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0,
        1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0], dtype=int64),
 0.4918032786885246)
```

In [118]:
```
# 4 ближайших соседа
cl1_2 = KNeighborsClassifier(n_neighbors=4)
cl1_2.fit(x_learn, y_learn)
target1_2 = cl1_2.predict(x_test)

target1_2, accuracy_score(y_test, target1_2)
```

Out[118]:
```
(array([0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
        0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0], dtype=int64),
 0.5409836065573771)
```

In [119]:
```
#2 соседа
balanced_accuracy_score(y_test, target1_1),
```

Out[119]:
```
(0.4935483870967742,)
```

In [120]:
```
#4 соседа
balanced_accuracy_score(y_test, target1_2)
```

Out[120]:
```
0.5419354838709678
```

In [121]:
```
# По умолчанию метрики precision считаются для 1 класса бинарной классификации
#2 соседа
precision_score(y_test, target1_1), recall_score(y_test, target1_1)
```

Out[121]:
```
(0.5, 0.3870967741935484)
```

In [122]:
```
#4 соседа
precision_score(y_test, target1_2), recall_score(y_test, target1_2)
```

Out[122]:
```
(0.5555555555555556, 0.4838709677419355)
```

In [123]:
```
col_x = data.loc[:, data.columns != 'target'].columns
col_x
```

Out[123]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal'],
      dtype='object')
```

In [124]:
```
scores = cross_val_score(KNeighborsClassifier(n_neighbors=5),
                         data[col_x], data['target'], cv=4)
scores, np.mean(scores)
```

Out[124]:
```
(array([0.62337662, 0.63157895, 0.65333333, 0.65333333]), 0.6404055593529278)
```

In [125]:
```
scoring = {'precision': 'precision_weighted',
           'recall': 'recall_weighted',
           'f1': 'f1_weighted'}

#использование cross_validate, которая позволяет использовать для оценки несколько метрик и возращает более детальную информацию
scores = cross_validate(KNeighborsClassifier(n_neighbors=5),
                        data[col_x], data['target'], scoring=scoring,
                        cv=3, return_train_score=True)
scores
```

Out[125]:
```
{'fit_time': array([0.00200009, 0.00200009, 0.00250006]),
 'score_time': array([0.02000117, 0.01675057, 0.01625013]),
 'test_f1': array([0.62181218, 0.60309602, 0.6568663 ]),
 'test_precision': array([0.62182853, 0.60270501, 0.66387408]),
 'test_recall': array([0.62376238, 0.6039604 , 0.66336634]),
 'train_f1': array([0.76742436, 0.72783702, 0.80497785]),
 'train_precision': array([0.76755968, 0.72798697, 0.80994419]),
 'train_recall': array([0.76732673, 0.72772277, 0.80693069])}
```

In [126]:
```
#Стратегия кросс-валидации k-fold
scores = cross_val_score(KNeighborsClassifier(n_neighbors=3),
                          data[col_x], data['target'],
                          cv=KFold(n_splits=4))
scores
```

Out[126]:
```
array([0.48684211, 0.51315789, 0.44736842, 0.30666667])
```

In [127]:
```
#ShuffleSplit
scores = cross_val_score(KNeighborsClassifier(n_neighbors=3),
                          data[col_x], data['target'],
                          cv=ShuffleSplit(n_splits=5, test_size=0.25))
scores
```

Out[127]:
```
array([0.63157895, 0.63157895, 0.67105263, 0.63157895, 0.65789474])
```

In [128]:
```
#6. Произведите подбор гиперпараметра K с использованием GridSearchCV и кросс-валидации.
from sklearn.model_selection import GridSearchCV

n_range = np.array(range(2,32,2))
tuned_parameters = [{'n_neighbors': n_range}]
tuned_parameters
```

Out[128]:
```
[{'n_neighbors': array([ 2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30])}]
```

In [130]:
```
clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=5, scoring='accuracy')
clf_gs.fit(x_learn, y_learn)
```

```
C:\Anaconda\lib\site-packages\sklearn\model_selection\_search.py:813: DeprecationWarning: The default of the `iid` parameter wi
ll change from True to False in version 0.22 and will be removed in 0.24. This will change numeric results when test-set sizes
are unequal.
  DeprecationWarning)
```

Out[130]:
```
GridSearchCV(cv=5, 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'),
             iid='warn', n_jobs=None,
             param_grid=[{'n_neighbors': array([ 2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30])}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='accuracy', verbose=0)
```

In [131]:
```
clf_gs.cv_results_
```

Out[131]:
```
{'mean_fit_time': array([0.00200009, 0.00160012, 0.00140004, 0.00125012, 0.00174994,
        0.00124993, 0.00125008, 0.00124993, 0.00175009, 0.00150008,
        0.00175004, 0.00124998, 0.00175009, 0.00149999, 0.00125012]),
 'mean_score_time': array([0.00360022, 0.00340014, 0.00355015, 0.00275006, 0.00275011,
        0.00325017, 0.00374999, 0.00300007, 0.00300002, 0.00349998,
        0.00350003, 0.00375013, 0.00275002, 0.00275011, 0.00324993]),
 'mean_test_score': array([0.59504132, 0.64049587, 0.62396694, 0.61157025, 0.59917355,
        0.60743802, 0.60330579, 0.61157025, 0.64049587, 0.63636364,
        0.65702479, 0.6322314 , 0.6446281 , 0.64876033, 0.64049587]),
 'param_n_neighbors': masked_array(data=[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'n_neighbors': 2},
  {'n_neighbors': 4},
  {'n_neighbors': 6},
  {'n_neighbors': 8},
  {'n_neighbors': 10},
  {'n_neighbors': 12},
  {'n_neighbors': 14},
  {'n_neighbors': 16},
  {'n_neighbors': 18},
  {'n_neighbors': 20},
  {'n_neighbors': 22},
  {'n_neighbors': 24},
  {'n_neighbors': 26},
  {'n_neighbors': 28},
  {'n_neighbors': 30}],
 'rank_test_score': array([15,  4,  9, 10, 14, 12, 13, 10,  4,  7,  1,  8,  3,  2,  4]),
 'split0_test_score': array([0.63265306, 0.63265306, 0.63265306, 0.57142857, 0.59183673,
        0.65306122, 0.63265306, 0.65306122, 0.65306122, 0.65306122,
        0.69387755, 0.67346939, 0.69387755, 0.69387755, 0.67346939]),
 'split1_test_score': array([0.59183673, 0.59183673, 0.53061224, 0.57142857, 0.57142857,
        0.57142857, 0.57142857, 0.53061224, 0.55102041, 0.53061224,
        0.55102041, 0.57142857, 0.6122449 , 0.63265306, 0.6122449 ]),
 'split2_test_score': array([0.48979592, 0.63265306, 0.59183673, 0.59183673, 0.59183673,
        0.59183673, 0.55102041, 0.55102041, 0.6122449 , 0.6122449 ,
        0.6122449 , 0.59183673, 0.6122449 , 0.59183673, 0.6122449 ]),
 'split3_test_score': array([0.70833333, 0.72916667, 0.77083333, 0.72916667, 0.66666667,
        0.66666667, 0.6875    , 0.77083333, 0.77083333, 0.79166667,
        0.79166667, 0.72916667, 0.70833333, 0.75      , 0.75      ]),
 'split4_test_score': array([0.55319149, 0.61702128, 0.59574468, 0.59574468, 0.57446809,
        0.55319149, 0.57446809, 0.55319149, 0.61702128, 0.59574468,
        0.63829787, 0.59574468, 0.59574468, 0.57446809, 0.55319149]),
 'std_fit_time': array([0.00000000e+00, 4.89862464e-04, 4.89940316e-04, 1.90734863e-07,
        6.12366996e-04, 1.16800773e-07, 9.53674316e-08, 1.16800773e-07,
        6.12250195e-04, 4.99987611e-04, 6.12386452e-04, 9.53674316e-08,
        6.12347524e-04, 4.99916099e-04, 1.16800773e-07]),
 'std_score_time': array([4.89862441e-04, 4.89940316e-04, 4.58297235e-04, 5.00130662e-04,
        4.99987633e-04, 6.12366996e-04, 1.16800773e-07, 6.12483797e-04,
        6.12328056e-04, 5.00082978e-04, 4.99987633e-04, 9.53674316e-08,
        4.99916077e-04, 4.99987633e-04, 6.12366996e-04]),
 'std_test_score': array([0.07360195, 0.04659239, 0.08013688, 0.05935441, 0.03463726,
        0.0447282 , 0.05001197, 0.09006068, 0.07274   , 0.08685051,
        0.08136394, 0.05954492, 0.04674228, 0.06500519, 0.06634816])}
```

In [132]:
```
# Лучшая модель
clf_gs.best_estimator_
```

Out[132]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=22, p=2,
                     weights='uniform')
```

In [133]:
```
# Лучшее значение метрики
clf_gs.best_score_
```

Out[133]:
```
0.6570247933884298
```

In [134]:
```
# Лучшее значение параметров
clf_gs.best_params_
```

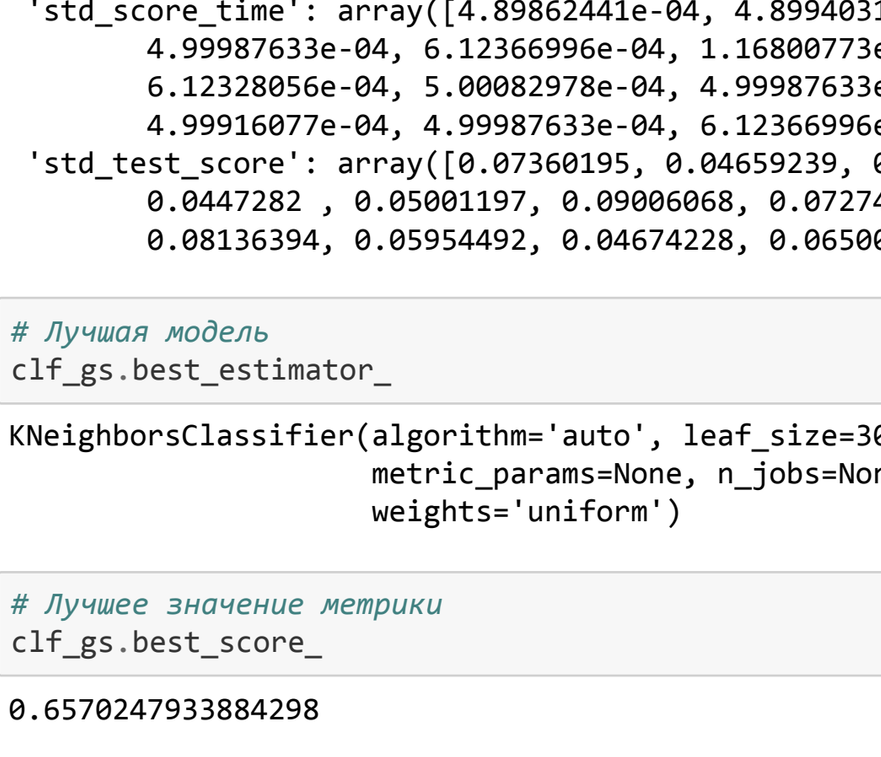
Out[134]:
```
{'n_neighbors': 22}
```

In [137]:
```
# Изменение качества на обучающей выборке
plt.plot(n_range, clf_gs.cv_results_['mean_train_score'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-137-cee8b102e5c2> in <module>()
      1 # Изменение качества на обучающей выборке
----> 2 plt.plot(n_range, clf_gs.cv_results_['mean_train_score'])

KeyError: 'mean_train_score'
```

In [138]:
```
# Изменение качества на тестовой выборке
plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
```

Out[138]:
```
[<matplotlib.lines.Line2D at 0x15ab3b70>]
```



In [139]:
```
#7. Повторите пункт 4 для найденного оптимального значения гиперпараметра K.
#Сравните качество полученной модели с качеством модели, полученной в пункте 4.

# 22 ближайших соседа
cl_best = KNeighborsClassifier(n_neighbors=22)
cl_best.fit(x_learn, y_learn)
target_best = cl_best.predict(x_test)
target_best
```

Out[139]:
```
array([1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0], dtype=int64)
```

In [140]:
```
accuracy_score(y_test, target_best)
```

Out[140]:
```
0.639344262295082
```

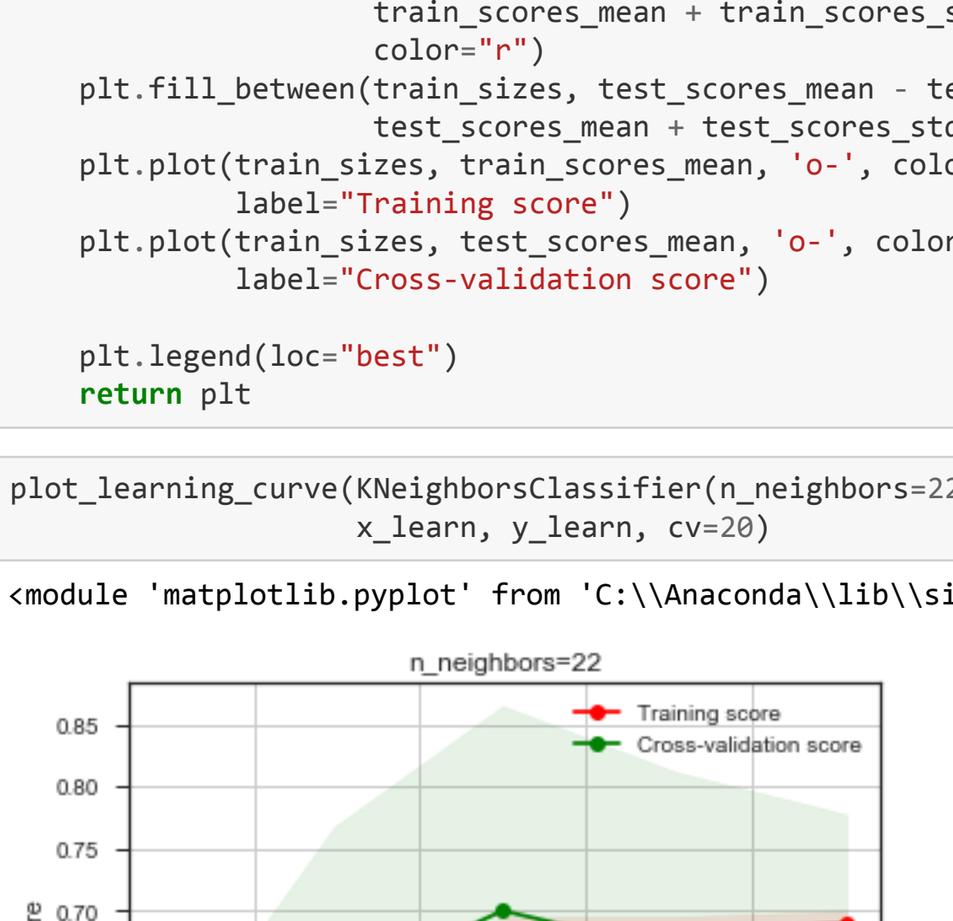
In [141]:
```
precision_score(y_test, target_best), recall_score(y_test, target_best)
```

Out[141]:
```
(0.6451612903225806, 0.6451612903225806)
```

In [142]:
```
from sklearn.model_selection import learning_curve, validation_curve

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
```

In [143]:
```
plot_learning_curve(KNeighborsClassifier(n_neighbors=22), 'n_neighbors=22',
                    x_learn, y_learn, cv=20)
```

Out[143]:
```
<module 'matplotlib.pyplot' from 'C:\\Anaconda\\lib\\site-packages\\matplotlib\\pyplot.py'>
```



In [144]:
```
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
```

In [145]:
```
plot_validation_curve(KNeighborsClassifier(), 'knn',
                      x_learn, y_learn,
                      param_name='n_neighbors', param_range=n_range,
                      cv=StratifiedKFold(n_splits=5), scoring="accuracy")
```

Out[145]:
```
<module 'matplotlib.pyplot' from 'C:\\Anaconda\\lib\\site-packages\\matplotlib\\pyplot.py'>
```