Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №4
по дисциплине
«Методы машинного обучения»
на тему
«Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей.»

Выполнила:
студентка группы ИУ5-24М
Горбовцова К.М.

Москва — 2020 г.

In [2]:
```
import numpy as np
import pandas as pd
```

In [3]:
```
from typing import Dict, Tuple
```

In [4]:
```
from scipy import stats
```

In [5]:
```
from sklearn.datasets import load_iris, load_boston
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifie
from sklearn.metrics import accuracy_score
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score, c
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.model_selection import KFold, RepeatedKFold, LeaveOneOut,
from sklearn.model_selection import learning_curve, validation_curve
from sklearn.metrics import mean_absolute_error, mean_squared_error, m
from sklearn.metrics import roc_curve, roc_auc_score
```

In [6]:
```
import seaborn as sns
```

In [7]:
```
import matplotlib.pyplot as plt
```

In [8]:
```
%matplotlib inline
sns.set(style="ticks")
```

In [9]:
```
data = pd.read_csv("heart.csv")
```

In [10]:
```
data.info(5)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
oldpeak     303 non-null float64
slope       303 non-null int64
ca          303 non-null int64
thal        303 non-null int64
target      303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.2 KB
```

In [11]:
```
data.describe()
```

Out[11]:

	age	sex	cp	trestbps	chol	fbs	restecg
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000

In [12]:
```
data.corr()
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalac
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.39852
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.04402
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.29576
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.04669
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.00994
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.00856
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.04412
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.00000
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.37881
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.34418
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.38678
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.21317
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.09643
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.42174

In [13]:
```
np.unique(data.target)
```

Out[13]:
```
array([0, 1])
```

In [14]:
```
target = data.iloc[:, -1]
```

In [15]:
```
data_data = data.iloc[:, 0:-1]
```

In [16]:
```
target.shape
```

Out[16]:
```
(303,)
```

In [17]:
```
data_data.shape
```

Out[17]:
```
(303, 13)
```

In [18]:
```
heart_X_train, heart_X_test, heart_y_train, heart_y_test = train_test_s
    data_data, target, test_size=0.5, random_state=1)
```

In [19]:
```
heart_X_train.shape, heart_y_train.shape
```

Out[19]:
```
((151, 13), (151,))
```

In [20]:
```
heart_X_test.shape, heart_y_test.shape
```

Out[20]:
```
((152, 13), (152,))
```

In [21]:
```
np.unique(heart_y_train), np.unique(heart_y_test)
```

Out[21]:
```
(array([0, 1]), array([0, 1]))
```

In [22]:
```
cl1_1 = KNeighborsClassifier(n_neighbors=2)
cl1_1.fit(heart_X_train, heart_y_train)
target1_1 = cl1_1.predict(heart_X_test)
len(target1_1), target1_1
```

Out[22]:
```
(152, array([0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1
, 0, 0, 0,
        0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
0, 0,
        0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1,
1, 1,
        0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0,
1, 0,
        1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
1, 0,
        0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
0, 0,
        0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1])
)
```

In [23]:
```
cl1_2 = KNeighborsClassifier(n_neighbors=10)
cl1_2.fit(heart_X_train, heart_y_train)
target1_2 = cl1_2.predict(heart_X_test)
len(target1_2), target1_2
```

Out[23]:
```
(152, array([1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1
, 0, 1, 1,
        0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0,
0, 0,
        1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
1, 0,
        0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0,
0, 0,
        1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
1, 0,
        1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1,
1, 0,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1])
)
```

In [24]:
```
accuracy_score(heart_y_test, target1_1)
```

Out[24]:
```
0.6052631578947368
```

In [25]:
```
accuracy_score(heart_y_test, target1_2)
```

Out[25]:
```
0.625
```

In [26]:
```
cl1_3 = KNeighborsClassifier(n_neighbors=30)
cl1_3.fit(heart_X_train, heart_y_train)
target1_3 = cl1_3.predict(heart_X_test)
len(target1_3), target1_3
```

Out[26]:
```
(152, array([1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1
, 0, 0, 0,
        1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
1, 0,
        1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
1, 1,
        0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0,
0, 0,
        1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
1, 1,
        1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
1, 1,
        0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1])
)
```

In [27]:
```
accuracy_score(heart_y_test, target1_3)
```

Out[27]:
```
0.618421052631579
```

In [28]:
```
kf = KFold(n_splits=5)
scores = cross_val_score(KNeighborsClassifier(n_neighbors=10),
                         data_data, target, scoring='f1_weighted',
                         cv=kf)
scores
```

Out[28]:
```
array([0.67391304, 0.67391304, 0.62214834, 0.58823529, 0.63636364])
```

In [29]:
```
scores = cross_val_score(KNeighborsClassifier(n_neighbors=50),
                         data_data, target,
                         cv=LeaveOneOut())
scores, np.mean(scores)
```

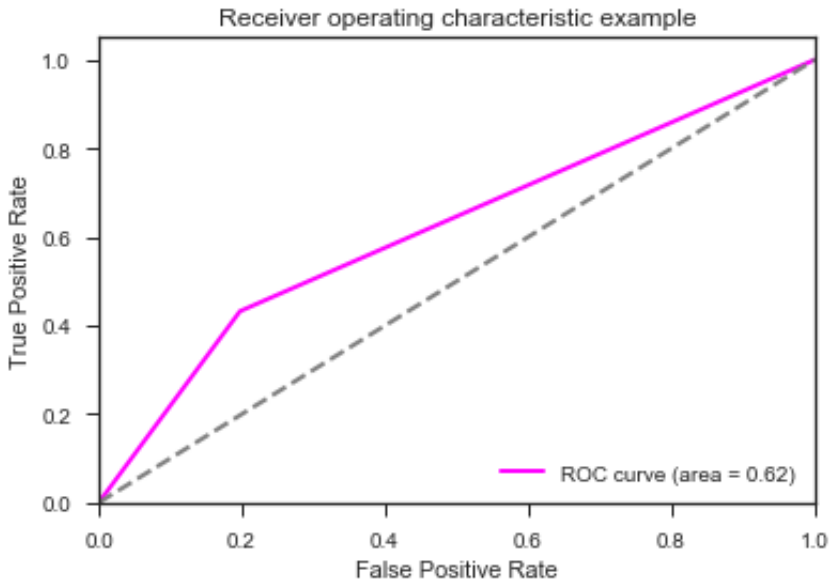
Out[29]:
```
(array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
1., 1.,
        0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1., 1., 0.,
1., 0.,
        0., 1., 1., 1., 0., 0., 0., 1., 1., 0., 1., 1., 1., 1., 0.,
1., 1.,
        0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1.,
1., 1.,
        1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 0.,
        1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 0., 1.,
1., 0.,
        1., 1., 1., 0., 0., 1., 1., 1., 0., 1., 0., 1., 1., 1., 1.,
1., 1.,
        1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1.,
1., 1.,
        0., 1., 1., 0., 1., 1., 1., 1., 0., 0., 0., 1., 1., 1., 0.,
0., 1.,
        0., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0.,
0., 0.,
        1., 0., 0., 0., 1., 1., 0., 0., 1., 1., 1., 1., 0., 0., 1.,
0., 0.,
        1., 0., 0., 1., 1., 1., 1., 0., 1., 0., 0., 1., 0., 0., 1.,
1., 1.,
        1., 0., 1., 0., 0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 1.,
1., 1.,
        1., 0., 1., 1., 1., 1., 0., 0., 1., 0., 1., 1., 1., 1., 0.,
0., 0.,
        0., 0., 1., 1., 1., 1., 1., 0., 0., 1., 0., 0., 1., 1., 1.,
1., 1.,
        1., 1., 1., 0., 0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 1.,
1., 1.,
        1., 0., 1., 0., 1., 1., 0., 1., 1., 0., 0., 0., 0., 1., 0.,
0., 1.,
        1., 0., 1., 0., 0., 0., 1., 0., 1., 1., 1., 0., 0., 0.]),
 0.6633663366336634)
```

In [30]:
```
fpr, tpr, thresholds = roc_curve(heart_y_test, target1_2,
                                 pos_label=1)
fpr, tpr, thresholds
```

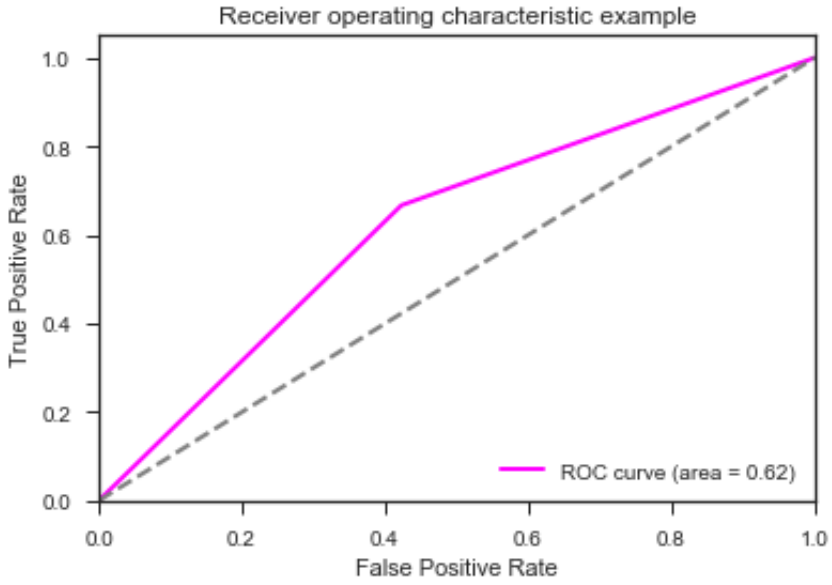
Out[30]:
```
(array([0.        , 0.42253521, 1.        ]),
 array([0.        , 0.66666667, 1.        ]),
 array([2, 1, 0]))
```

In [62]:
```
def draw_roc_curve(y_true, y_score, pos_label, average):
    fpr, tpr, thresholds = roc_curve(y_true, y_score,
                                     pos_label=pos_label)
    roc_auc_value = roc_auc_score(y_true, y_score, average=average)
    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='magenta',
             lw=lw, label='ROC curve (area = %0.2f)' % roc_auc_value)
    plt.plot([0, 1], [0, 1], color='gray', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
```

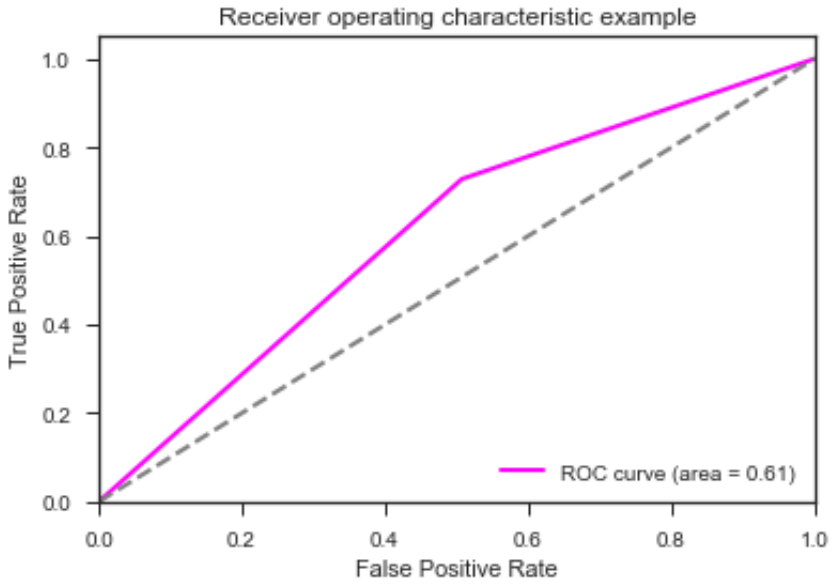
In [63]:
```
draw_roc_curve(heart_y_test, target1_1, pos_label=1, average='micro')
```



In [64]:
```
draw_roc_curve(heart_y_test, target1_2, pos_label=1, average='micro')
```



In [65]:
```
draw_roc_curve(heart_y_test, target1_3, pos_label=1, average='micro')
```



In [66]:
```
scores = cross_val_score(KNeighborsClassifier(n_neighbors=5),
                         data_data, target, cv=3)
```

In [67]:
```
scores
```

Out[67]:
```
array([0.62376238, 0.6039604 , 0.66336634])
```

In [68]:
```
np.mean(scores)
```

Out[68]:
```
0.6303630363036303
```

In [69]:
```
n_range = np.array(range(5,55,5))
tuned_parameters = [{'n_neighbors': n_range}]
tuned_parameters
```

Out[69]:
```
[{'n_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])}]
```

In [70]:
```
%%time
clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=5, s
clf_gs.fit(heart_X_train, heart_y_train)
```

```
CPU times: user 223 ms, sys: 4.32 ms, total: 227 ms
Wall time: 233 ms
```

In [71]:
```
clf_gs.best_estimator_
```

Out[71]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkows
ki',
                     metric_params=None, n_jobs=None, n_neighbors=35
, p=2,
                     weights='uniform')
```

In [72]:
```
clf_gs.best_score_
```

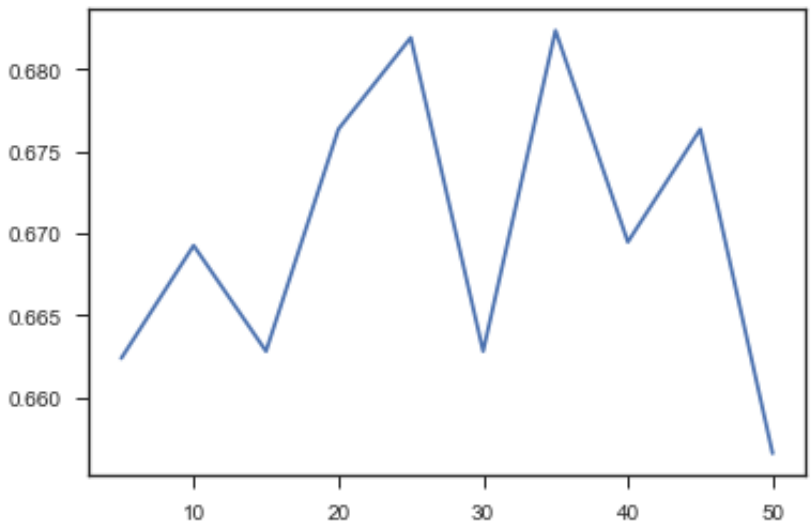
Out[72]:
```
0.6823655913978494
```

In [73]:
```
clf_gs.best_params_
```

Out[73]:
```
{'n_neighbors': 35}
```

In [74]:
```
plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
```

Out[74]:
```
[<matplotlib.lines.Line2D at 0x11cb35128>]
```



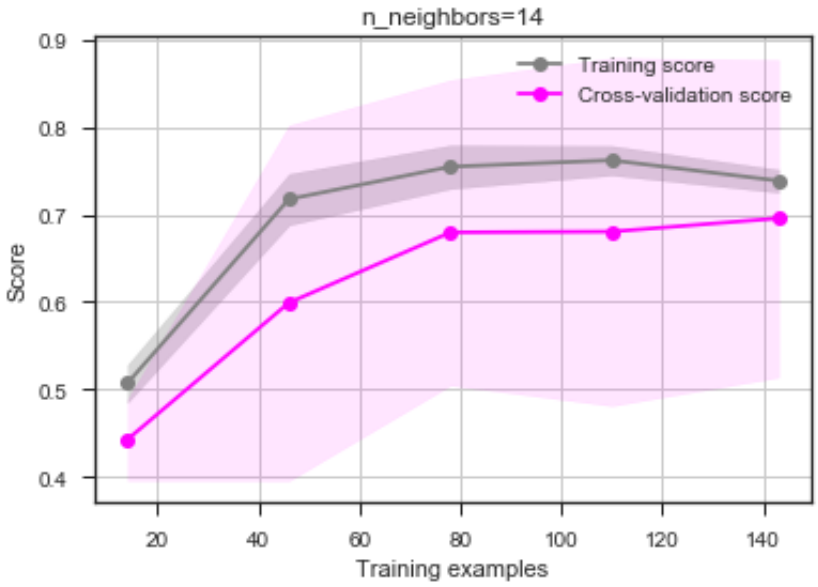
In [75]:
```
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=None, train_sizes=np.linspace(.1, 1.0, 
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    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=0.3,
                     color="grey")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, co
    plt.plot(train_sizes, train_scores_mean, 'o-', color="grey",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="magenta",
             label="Cross-validation score")

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

In [76]:
```
plot_learning_curve(KNeighborsClassifier(n_neighbors=14), 'n_neighbors=
                    heart_X_train, heart_y_train, cv=20)
```

Out[76]:
```
<module 'matplotlib.pyplot' from '/anaconda3/lib/python3.6/site-pack
ages/matplotlib/pyplot.py'>
```



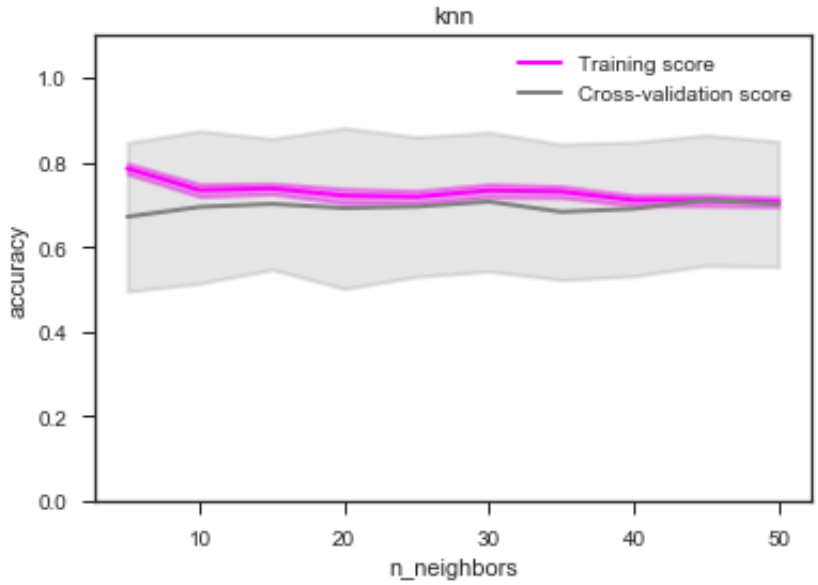
In [79]:
```
def plot_validation_curve(estimator, title, X, y,
                          param_name, param_range, cv,
                          scoring="accuracy"):

    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(str(scoring))
    plt.ylim(0.0, 1.1)
    lw = 2
    plt.plot(param_range, train_scores_mean, label="Training score",
                 color="magenta", lw=lw)
    plt.fill_between(param_range, train_scores_mean - train_scores_std
                     train_scores_mean + train_scores_std, alpha=0.4,
                     color="magenta", lw=lw)
    plt.plot(param_range, test_scores_mean, label="Cross-validation sc
                 color="grey", lw=lw)
    plt.fill_between(param_range, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.2,
                     color="grey", lw=lw)
    plt.legend(loc="best")
    return plt
```

In [80]:
```
plot_validation_curve(KNeighborsClassifier(), 'knn',
                      heart_X_train, heart_y_train,
                      param_name='n_neighbors', param_range=n_range,
                      cv=20, scoring="accuracy")
```

Out[80]:
```
<module 'matplotlib.pyplot' from '/anaconda3/lib/python3.6/site-pack
ages/matplotlib/pyplot.py'>
```



In [81]:
```
n_range = np.array(range(5,55,5))
tuned_parameters = [{'n_neighbors': n_range}]
tuned_parameters
```

Out[81]:
```
[{'n_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])}]
```

In [82]:
```
%%time
clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=Lea
clf_gs.fit(data_data, target)
```

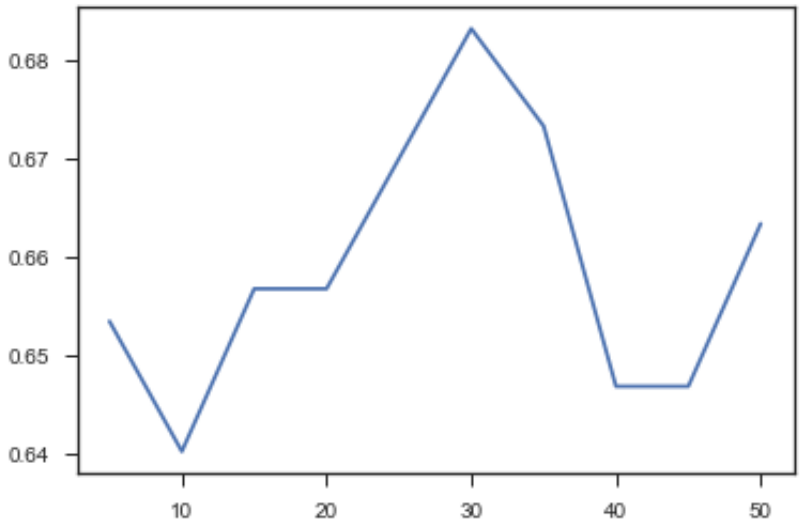
```
CPU times: user 12.4 s, sys: 52.3 ms, total: 12.4 s
Wall time: 13 s
```

In [83]:
```
clf_gs.best_params_
```

Out[83]:
```
{'n_neighbors': 30}
```

In [84]:
```
plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
```

Out[84]:
```
[<matplotlib.lines.Line2D at 0x11c9c2ba8>]
```



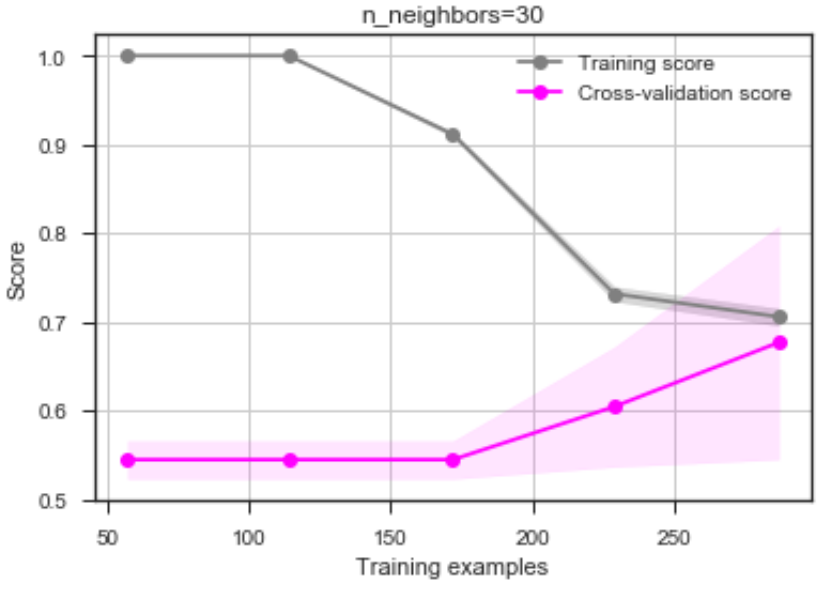
In [85]:
```
clf_gs.best_estimator_.fit(heart_X_train, heart_y_train)
target2_0 = clf_gs.best_estimator_.predict(heart_X_train)
target2_1 = clf_gs.best_estimator_.predict(heart_X_test)
```

In [86]:
```
accuracy_score(heart_y_train, target2_0), accuracy_score(heart_y_test,
```

Out[86]:
```
(0.7284768211920529, 0.618421052631579)
```

In [87]:
```
plot_learning_curve(clf_gs.best_estimator_, 'n_neighbors=30',
                    data_data, target, cv=20, train_sizes=np.linspace(
```

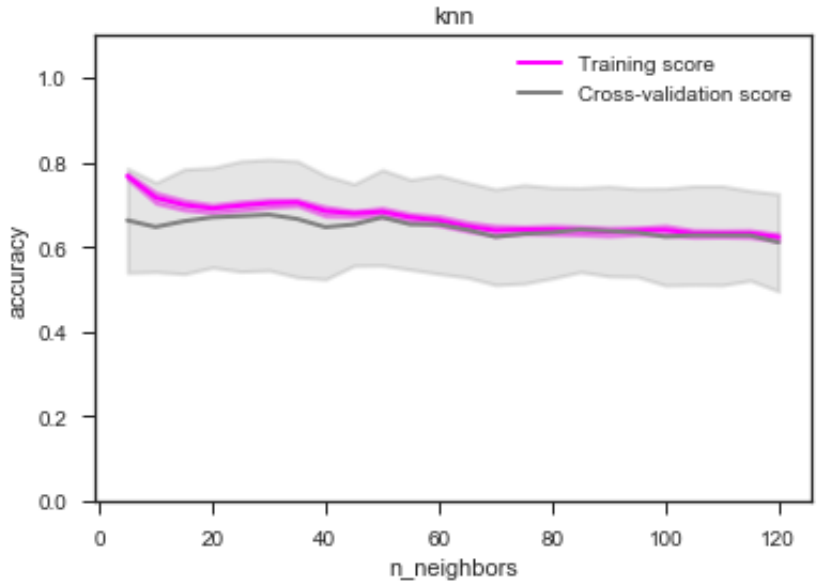
Out[87]:
```
<module 'matplotlib.pyplot' from '/anaconda3/lib/python3.6/site-pack
ages/matplotlib/pyplot.py'>
```



In [88]:
```
n_range2 = np.array(range(5,125,5))
```

In [89]:
```
plot_validation_curve(clf_gs.best_estimator_, 'knn',
                      data_data, target,
                      param_name='n_neighbors', param_range=n_range2,
                      cv=20, scoring="accuracy")
```

Out[89]:
```
<module 'matplotlib.pyplot' from '/anaconda3/lib/python3.6/site-pack
ages/matplotlib/pyplot.py'>
```



In [ ]:
```

```