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.»

Выполнил:
студент группы ИУ5-23М
Богомолов Д.Н.

Москва — 2020 г.

Подготовка обучающей и тестовой выборки, кросс-вали гиперпараметров на примере метода ближайших сосед

Цель лабораторной работы:

изучение сложных способов подготовки выборки и подбора гиперпараметров на примере

Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирован
3. С использованием метода train_test_split разделите выборку на обучающую и тестову
4. Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. подходящих для задачи метрик.
5. Постройте модель и оцените качество модели с использованием кросс-валидации. П различными стратегиями кросс-валидации.
6. Произведите подбор гиперпараметра K с использованием GridSearchCV и кросс-вали
7. Повторите пункт 4 для найденного оптимального значения гиперпараметра K. Сравни качеством модели, полученной в пункте 4.
8. Постройте кривые обучения и валидации.

```
1  from google.colab import drive  #Монтирование Google Drive
2  drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

Задание 1. Выберите набор данных (датасет) для решения зада регресии.

Выбранный датасет - HEART.CSV Параметры датасета:

- age - age inyears
- sex(1 = male; 0 = female)
- cp- chest pain type
- trestbps - resting blood pressure (in mm Hg on admission to the hospital)
- chol - serum cholestoral in mg/dl fbs(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- restecg - resting electrocardiographic results
- thalach - maximum heart rate achieved
- exang - exercise induced angina (1 = yes; 0 = no) oldpeakST depression induced by exercise
- slope - the slope of the peak exercise ST segment
- ca - number of major vessels (0-3) colored by flourosopy

- thal: 3 = normal; 6 = fixed defect; 7 = reversable defect target 1 or 0

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
heart = pd.read_csv('/content/drive/My Drive/MMO_Datasets/heart.csv', sep=',')
heart.head(10)

```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarnin
  import pandas.util.testing as tm
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tha
0	63	1	3	145	233	1	0	150	0	2.3	0	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	
5	57	1	0	140	192	0	1	148	0	0.4	1	0	
6	56	0	1	140	294	0	0	153	0	1.3	1	0	
7	44	1	1	120	263	0	1	173	0	0.0	2	0	
8	52	1	2	172	199	1	1	162	0	0.5	2	0	
9	57	1	2	150	168	0	1	174	0	1.6	2	0	

Задание 2. В случае необходимости проведите удаление или за кодирование категориальных признаков.

```
heart.shape
```

```
(303, 14)
```

```
heart.dtypes
```

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
```

Проверка на пустые значения

```
slope         int64
```

```
1  heart.isnull().sum()
```

```
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
```

Пустых значений не обнаружено

```
1  X = heart.drop('target',axis = 1).values
2  y = heart['target'].values
```

Задание 3. С использованием метода **train_test_split** разделите тестовую

Функция **train_test_split** используется для того, чтобы разделить исходную выборку так, что сохранились пропорции классов.

Разделим выборку следующим образом:

- Размер обучающей выборки (65%)
- Размер тестовой выборки (35%)

```
1  from sklearn.model_selection import train_test_split
2  heart_X_train, heart_X_test, heart_y_train, heart_y_test = train_test_split(
3      X, y, test_size=0.35, random_state=1)
```

```
1  print('heart_X_train: {}  heart_y_train: {}'.format(heart_X_train.shape, heart_y_trai
```

```
heart_X_train: (196, 13)  heart_y_train: (196,)
```

```
print('heart_X_test: {}  heart_y_test: {}'.format(heart_X_test.shape, heart_y_test.sh
```

```
heart_X_test: (107, 13)  heart_y_test: (107,)
```

```
np.unique(heart_y_train)
```

```
array([0, 1])
```

```
np.unique(heart_y_test)
```

```
array([0, 1])
```

Задание 4. Обучение модели ближайших соседей для произвол гиперпараметра K.

Обучите модель ближайших соседей для произвольно заданного гиперпараметр помощью трех подходящих для задачи метрик.

```
neighbors = np.arange(1,14)
len(neighbors)
```

```
13
```

Обучение при различном количестве соседей

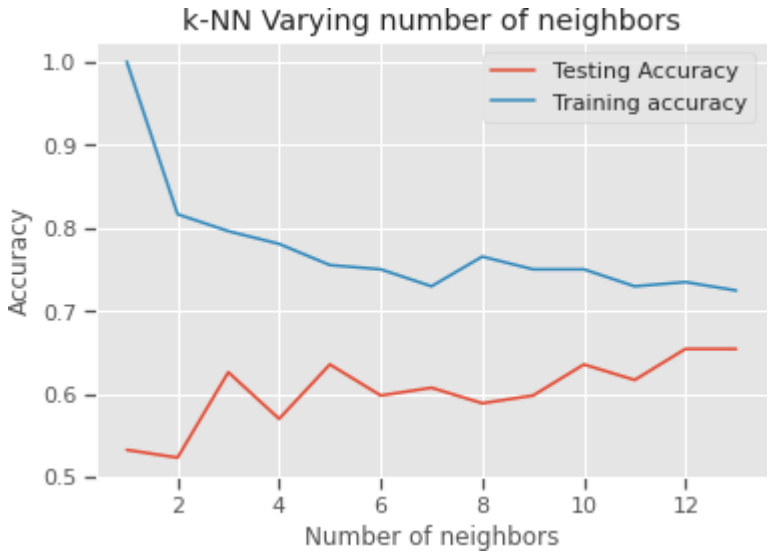
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import learning_curve, validation_curve
from sklearn.model_selection import KFold, RepeatedKFold, LeaveOneOut, LeavePOut, Shu
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.metrics import roc_curve,confusion_matrix, roc_auc_score, accuracy_score
```

Возвращает новый массив заданной формы и типа без инициализации записей.

```
train_accuracy =np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

for i,k in enumerate(neighbors):
    # Настройка классификатора Knn с K соседями
    knn = KNeighborsClassifier(n_neighbors=k)

    # Обучение модели
    knn.fit(heart_X_train, heart_y_train)
    # Вычиление точности на тренировочном наборе
    train_accuracy[i] = knn.score(heart_X_train, heart_y_train)
    # Вычисление точности на тестовом наборе
    test_accuracy[i] = knn.score(heart_X_test, heart_y_test)
```

Построить график для метода ближайших соседей

```
plt.style.use('ggplot')

plt.title('k-NN Varying number of neighbors')
plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
plt.plot(neighbors, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
plt.show()
```



Изучение работы KNeighborsClassifier

Установка a knn classifier with k neighbors, где k = 12

```
knn = KNeighborsClassifier(n_neighbors=12)
#Fit the model
knn.fit(heart_X_train,heart_y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=12, p=2,
                     weights='uniform')
```

Определение точности

```
knn.score(heart_X_test,heart_y_test)
```

```
0.6542056074766355
```

```
#import classification_report
from sklearn.metrics import classification report
```

```
3
4   y_pred = knn.predict(heart_X_test)
5   print(classification_report(heart_y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.62      0.66      0.64        50
           1       0.69      0.65      0.67        57

    accuracy                           0.65       107
   macro avg       0.65      0.65      0.65       107
weighted avg       0.66      0.65      0.65       107
```

Оценка точности

```
1   from sklearn.metrics import roc_curve,confusion_matrix, roc_auc_score, accuracy_score
```

```
1   cl1_1 = KNeighborsClassifier(n_neighbors=12)
2   cl1_1.fit(heart_X_train, heart_y_train)
3   target1_1 = cl1_1.predict(heart_X_test)
4   accuracy_score(heart_y_test, target1_1)
```

```
0.6542056074766355
```

Confusion Matrix - матрица ошибок

Матрица ошибок (confusion matrix) – таблица с 4 различными комбинациями прогнозируем Прогнозируемые значения описываются как положительные и отрицательные, а фактичес матрица ошибок используется для оценки точности моделей в задачах классификации. Но образов можно рассматривать как частный случай этой проблемы, поэтому confusion matr предсказаний.

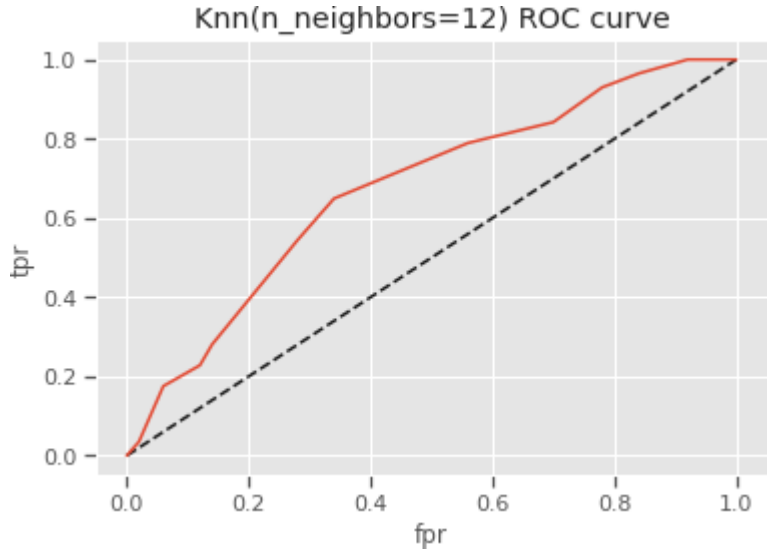
```
1   y_pred = knn.predict(heart_X_test)
2   confusion_matrix(heart_y_test,y_pred)
3   pd.crosstab(heart_y_test, y_pred, rownames=['True'], colnames=['Predicted'], margins=
```

Predicted	0	1	All
True			
0	33	17	50
1	20	37	57
All	53	54	107

ROC(Reciever Operating Charecteristic)-кривая

```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

```
y_pred_proba = knn.predict_proba(heart_X_test)[:,1]
fpr, tpr, thresholds = roc_curve(heart_y_test, y_pred_proba)
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr, label='Knn')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('Knn(n_neighbors=12) ROC curve')
plt.show()
```




```
y_pred_proba
```

```
array([0.5       , 0.66666667, 0.41666667, 0.66666667, 0.16666667,
       0.5       , 0.16666667, 0.58333333, 0.5       , 0.66666667,
       0.66666667, 0.5       , 0.58333333, 0.25      , 0.66666667,
       0.91666667, 0.91666667, 0.41666667, 0.91666667, 0.41666667,
       0.5       , 0.41666667, 0.41666667, 0.5       , 0.66666667,
       0.83333333, 0.5       , 1.        , 0.41666667, 0.66666667,
       0.91666667, 0.83333333, 0.66666667, 0.58333333, 0.41666667,
       1.        , 0.33333333, 0.66666667, 0.83333333, 0.25      ,
       0.66666667, 0.33333333, 0.66666667, 0.25      , 0.66666667,
       0.66666667, 0.5       , 0.75      , 0.91666667, 0.16666667,
       0.83333333, 0.83333333, 0.33333333, 0.33333333, 0.75      ,
       0.66666667, 0.66666667, 0.58333333, 1.        , 0.08333333,
       0.33333333, 0.91666667, 0.58333333, 0.66666667, 0.83333333,
       0.58333333, 0.5       , 0.75      , 0.16666667, 0.33333333,
       0.33333333, 0.75      , 0.08333333, 0.66666667, 0.16666667,
       0.33333333, 0.33333333, 0.5       , 0.41666667, 0.91666667,
       0.91666667, 0.66666667, 0.08333333, 0.5       , 0.5       ,
       0.5       , 0.25      , 0.5       , 0.91666667, 0.5       ,
       0.16666667, 0.5       , 0.66666667, 0.25      , 0.91666667,
       0.08333333, 0.5       , 0.58333333, 0.58333333, 0.58333333,
       0.66666667, 0.41666667, 0.5       , 0.66666667, 0.5       ,
       0.41666667, 0.66666667])
```

```
roc_auc_score(heart_y_test,y_pred_proba)
```

```
0.6740350877192982
```

Задание 6. Произведите подбор гиперпараметра K с использов валидации.

↳ *3 cells hidden*

```
[ ]  1  import warnings
     2  warnings.filterwarnings('ignore')
     3
     4  param_grid = {'n_neighbors':np.arange(1,14)}
     5  knn = KNeighborsClassifier()
     6  knn_cv= GridSearchCV(knn,param_grid,cv=5)
     7  knn_cv.fit(X,y)
```

```
GridSearchCV(cv=5, error_score=nan,
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                            metric='minkowski',
                                            metric_params=None, n_jobs=None,
                                            n_neighbors=5, p=2,
                                            weights='uniform'),
             iid='deprecated', n_jobs=None,
             param_grid={'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13])},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

```
[ ]  1  knn_cv.best_score_
```

```
0.6601092896174864
```

```
[ ]  1  knn_cv.best_params_
```

```
{'n_neighbors': 12}
```

Задание 5. Постройте модель и оцените качество модели с исп валидации. Проведите эксперименты с тремя различными стр

K-fold. Перекрестная проверка — это процедура повторной выборки, используемая для оце ограниченной выборке данных.

```
1  scores = cross_val_score(KNeighborsClassifier(n_neighbors=12),
2                           X, y,
3                           cv=KFold(n_splits=10))
```

Значение метрики accuracy для 10 фолдов Кол-во фолдов - кол-во разбиений

```
1  scores
```

```
array([0.5483871 , 0.61290323, 0.64516129, 0.6       , 0.5       ,
       0.66666667, 0.4       , 0.63333333, 0.66666667, 0.56666667])
```

Усредненное значение метрики accuracy для 10 фолдов

```
1  np.mean(scores)
```

```
0.5839784946236558
```

```
1  import warnings
2  warnings.filterwarnings('ignore')
```

```
  scoring = {'precision': 'precision_weighted',
             'recall': 'recall_weighted',
             'f1': 'f1_weighted'}

  scores = cross_validate(KNeighborsClassifier(n_neighbors=4),
                          X, y, scoring=scoring,
                          cv=KFold(n_splits=5), return_train_score=True)
  scores
```

```
{'fit_time': array([0.00094604, 0.00057364, 0.00054836, 0.00054193, 0.00052929]),
 'score_time': array([0.01519489, 0.00420713, 0.00418401, 0.0041275 , 0.00414848]),
 'test_f1': array([0.53012048, 0.65934066, 0.62005786, 0.62068966, 0.75      ]),
 'test_precision': array([1.        , 1.        , 0.74327869, 1.        , 1.        ]
 'test_recall': array([0.36065574, 0.49180328, 0.60655738, 0.45      , 0.6       ]),
 'train_f1': array([0.76729516, 0.75673435, 0.74007707, 0.78961321, 0.79856665]),
 'train_precision': array([0.79272947, 0.78911533, 0.76025871, 0.79711127, 0.80984164
 'train_recall': array([0.7768595 , 0.76859504, 0.74380165, 0.78600823, 0.79423868])}
```

LOOCV Leave One Out cross-validation

k равен общему размеру наблюдений в датасете. В тестовую выборку помещается единств
фолдов в этом случае определяется автоматически и равняется количеству элементов. Да
KFold. Существует правило, что вместо Leave One Out лучше использовать KFold на 5 или 1

```
# Эквивалент KFold(n_splits=n)
loo = LeaveOneOut()
loo.get_n_splits(X)

for train_index, test_index in loo.split(X):
   heart_X_train, heart_X_test = X[train_index], X[test_index]
   heart_y_train, heart_y_test = y[train_index], y[test_index]
```

Repeated K-Fold

k-fold перекрестная валидация повторяется n раз, и датасет перетасовывается на каждой
данных.

```
scores2 = cross_val_score(KNeighborsClassifier(n_neighbors=12),
                          X, y,
                          cv=RepeatedKFold(n_splits=5, n_repeats=2))
```

```
scores2
```

```
array([0.72131148, 0.6557377 , 0.6557377 , 0.6       , 0.61666667,
       0.60655738, 0.6557377 , 0.73770492, 0.68333333, 0.56666667])
```

ShuPe Split

Генерируется N случайных перемешиваний данных, в каждом перемешивании заданная до

```
X = range(12)
# Эквивалент KFold(n_splits=n)
kf = ShuffleSplit(n_splits=5, test_size=0.35)
for train, test in kf.split(X):
    print("%s %s" % (train, test))
```

```
[ 3  2  9  5  8 10  1] [ 4  6  7  0 11]
[11  7  9  6  0 10  2] [1 3 5 8 4]
[ 1  8  3  5  9 11  6] [ 7 10  2  4  0]
[ 4  8  3 11  0 10  6] [1 2 5 9 7]
[ 9  2 10  7  5  4  1] [ 8  6  0  3 11]
```

Задание 7. Повторите пункт 4 для найденного оптимального значения

Сравните качество полученной модели с качеством модели

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, random_stat
```

```
knn = KNeighborsClassifier(n_neighbors=12)
knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

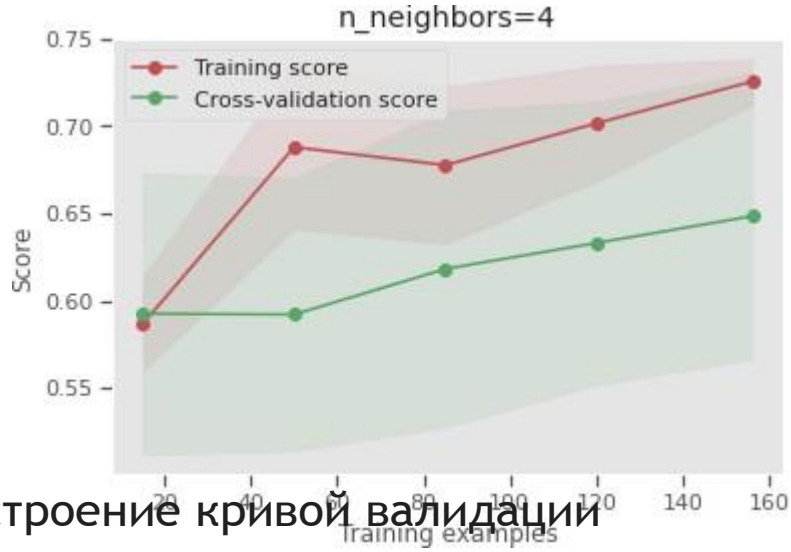
```
0.6542056074766355
```

Задание 8. Постройте кривые обучения и валидации.

Построение кривых обучения

```
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
```

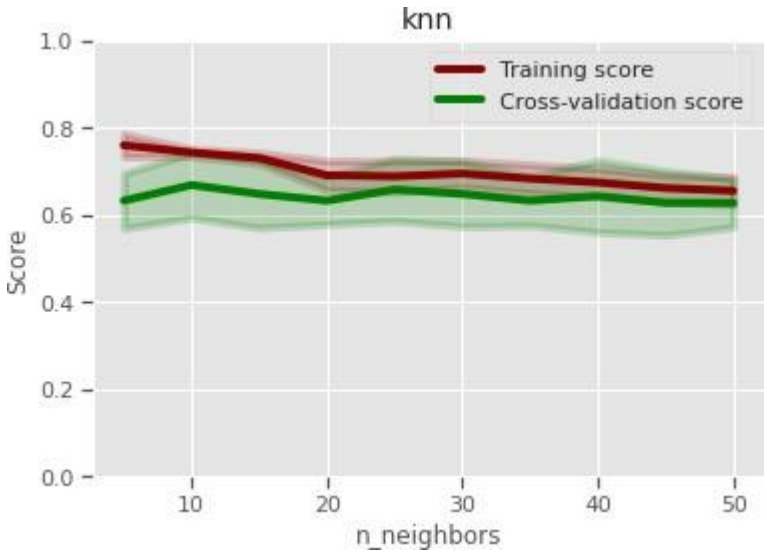
```
plot_learning_curve(KNeighborsClassifier(n_neighbors=12), 'n_neighbors=4',
                    X_train, y_train, cv=5)
```



Построение кривой валидации

```
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, 1.0)
    lw = 4
    plt.plot(param_range, train_scores_mean, label="Training score",
                 color="darkred", lw=lw)
    plt.fill_between(param_range, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.2,
                     color="darkred", lw=lw)
    plt.plot(param_range, test_scores_mean, label="Cross-validation score",
                 color="green", lw=lw)
    plt.fill_between(param_range, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.2,
                     color="green", lw=lw)
    plt.legend(loc="best")
    return plt
```

```
n_range = np.array(range(5,55,5))
plot_validation_curve(KNeighborsClassifier(n_neighbors=4), 'knn',
                      X_train, y_train,
                      param_name='n_neighbors', param_range=n_range,
                      cv=5, scoring="accuracy")
```


knn
Training score
Cross-validation score
Score
n_neighbors
1.0
0.8
0.6
0.4
0.2
0.0
10
20
30
40
50

1