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

Лабораторная работа №4
по курсу «Методы машинного обучения»

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

Выполнила: Чекулина М.Ю.
Группа ИУ5-22М

Москва – 2019

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.

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.

Выбранный датасет

https://www.kaggle.com/ronitf/heart-disease-uci/version/1
(https://www.kaggle.com/ronitf/heart-disease-uci/version/1)

Параметры датасета: age - age in years 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 relative to rest 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

In [1]:
```
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('Data/lab_4/heart.csv', sep=',')
heart.head(10)
```

Out[1]:

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

In [2]:
```
heart.shape
```

Out[2]:
```
(303, 14)
```

In [3]:
```
heart.dtypes
```

Out[3]:
```
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 [4]:
```
# Проверка на пустые значения
heart.isnull().sum()
```

Out[4]:
```
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 [5]:
```
# Пустых значений нет
# Перейдем к разделению выборки на обучающую и тестовую.
X = heart.drop('target',axis = 1).values
y = heart['target'].values
#heart.target
```

Разделение на обучающую и тестовую выборки

In [6]:
```
from sklearn.model_selection import train_test_split
# Функция train_test_split разделила исходную выборку таким образом,
#чтобы в обучающей и тестовой частях сохранились пропорции классов.
heart_X_train, heart_X_test, heart_y_train, heart_y_test = train_test_
    X, y, test_size=0.35, random_state=1)
```

In [7]:
```
# Размер обучающей выборки (65%)
print('heart_X_train: {}  heart_y_train: {}'.format(heart_X_train.shap
```

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

In [8]:
```
# Размер тестовой выборки (35%)
print('heart_X_test: {}  heart_y_test: {}'.format(heart_X_test.shape,
```

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

In [9]:
```
# Функция train_test_split разделила исходную выборку таким образом,
#чтобы в обучающей и тестовой частях сохранились пропорции классов.
np.unique(heart_y_train)
```

Out[9]:
```
array([0, 1])
```

In [10]:
```
np.unique(heart_y_test)
```

Out[10]:
```
array([0, 1])
```

Обучение модели ближайших соседей для произвольно заданного гиперпараметра K.

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

In [11]:
```
# Setup arrays to store training and test accuracies
neighbors = np.arange(1,14)
len(neighbors)
```

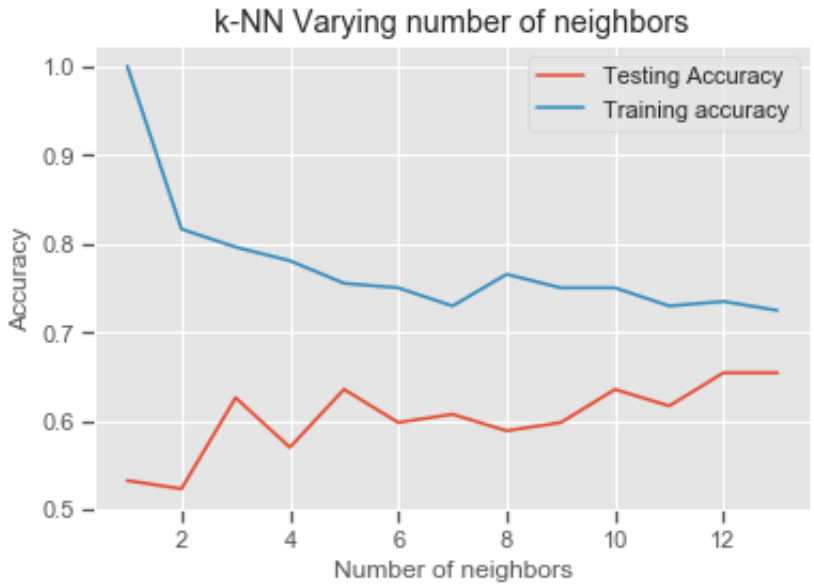
Out[11]:
```
13
```

Обучение при различном количестве соседей

In [12]:
```
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,
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.metrics import roc_curve,confusion_matrix, roc_auc_score,
```

In [13]:
```
# Вернуть 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)

    # Обучить модель
    knn.fit(heart_X_train, heart_y_train)

    # Вычислить точность на тренировочном наборе
    train_accuracy[i] = knn.score(heart_X_train, heart_y_train)

    # Вычислить точность на тестовом наборе
    test_accuracy[i] = knn.score(heart_X_test, heart_y_test)
```

In [14]:
```
# Построить набор
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

In [15]:
```
# Setup a knn classifier with k neighbors
knn = KNeighborsClassifier(n_neighbors=12)

#Fit the model
knn.fit(heart_X_train,heart_y_train)
```

Out[15]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkows
ki',
           metric_params=None, n_jobs=None, n_neighbors=12, p=2,
           weights='uniform')
```

In [16]:
```
#Get accuracy. Note: In case of classification algorithms score metho
knn.score(heart_X_test,heart_y_test)
```

Out[16]:
```
0.6542056074766355
```

In [17]:
```
#import classification_report
from sklearn.metrics import classification_report

y_pred = knn.predict(heart_X_test)
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

   micro avg       0.65      0.65      0.65       107
   macro avg       0.65      0.65      0.65       107
weighted avg       0.66      0.65      0.65       107
```

Accuracy

In [18]:
```
from sklearn.metrics import roc_curve,confusion_matrix, roc_auc_score,
```

In [19]:
```
cl1_1 = KNeighborsClassifier(n_neighbors=12)
cl1_1.fit(heart_X_train, heart_y_train)
target1_1 = cl1_1.predict(heart_X_test)
accuracy_score(heart_y_test, target1_1)
```

Out[19]:
```
0.6542056074766355
```

Confusion Matrix

In [20]:
```
y_pred = knn.predict(heart_X_test)
confusion_matrix(heart_y_test,y_pred)
pd.crosstab(heart_y_test, y_pred, rownames=['True'], colnames=['Predic
```

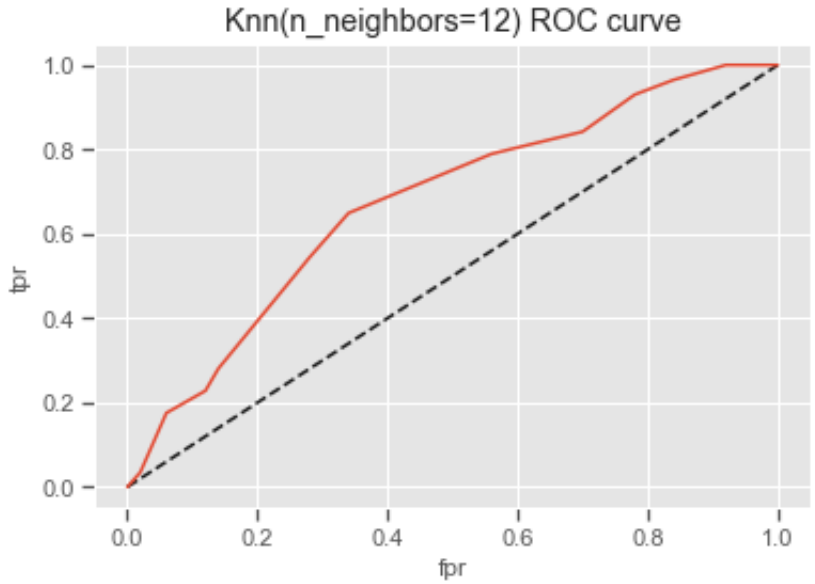
Out[20]:

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

ROC(Reciever Operating Charecteristic)-кривая

In [21]:
```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

In [22]:
```
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()
```



In [23]:
```
y_pred_proba
```

Out[23]:
```
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])
```

In [24]:
```
roc_auc_score(heart_y_test,y_pred_proba)
```

Out[24]:
```
0.6740350877192982
```

Cross Validation

In [25]:
```
import warnings
warnings.filterwarnings('ignore')

param_grid = {'n_neighbors':np.arange(1,14)}
knn = KNeighborsClassifier()
knn_cv= GridSearchCV(knn,param_grid,cv=5)
knn_cv.fit(X,y)
```

Out[25]:
```
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'),
       fit_params=None, iid='warn', 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='warn
',
       scoring=None, verbose=0)
```

In [26]:
```
knn_cv.best_score_
```

Out[26]:
```
0.6600660066006601
```

In [27]:
```
knn_cv.best_params_
```

Out[27]:
```
{'n_neighbors': 12}
```

K-fold

Каждой стратегии в scikit-learn ставится в соответствии специальный класс-итератор, который может быть указан в качестве параметра cv функций cross_val_score и cross_validate. Работает в соответствии с кросс-валидацией

In [28]:
```
scores = cross_val_score(KNeighborsClassifier(n_neighbors=12),
                         X, y,
                         cv=KFold(n_splits=10))
```

In [29]:
```
# Значение метрики accuracy для 10 фолдов
scores
```

Out[29]:
```
array([0.5483871 , 0.61290323, 0.64516129, 0.6       , 0.5       ,
       0.66666667, 0.4       , 0.63333333, 0.66666667, 0.56666667])
```

In [30]:
```
# Усредненное значение метрики accuracy для 10 фолдов
np.mean(scores)
```

Out[30]:
```
0.5839784946236558
```

In [31]:
```
import warnings
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
```

Out[31]:
```
{'fit_time': array([0.00120521, 0.00088596, 0.00042272, 0.00039196,
0.00040793]),
 'score_time': array([0.00611663, 0.00458217, 0.0035162 , 0.00328493
, 0.00323105]),
 'test_precision': array([1.        , 1.        , 0.74327869, 1.
, 1.        ]),
 'train_precision': array([0.79272947, 0.78911533, 0.76025871, 0.797
11127, 0.80984164]),
 'test_recall': array([0.36065574, 0.49180328, 0.60655738, 0.45
, 0.6       ]),
 'train_recall': array([0.7768595 , 0.76859504, 0.74380165, 0.786008
23, 0.79423868]),
 'test_f1': array([0.53012048, 0.65934066, 0.62005786, 0.62068966, 0
.75      ]),
 'train_f1': array([0.76729516, 0.75673435, 0.74007707, 0.78961321,
0.79856665])}
```

Leave One Out (LOO)

В тестовую выборку помещается единственный элемент (One Out). Количество фолдов в этом случае определяется автоматически и равняется количеству элементов. Данный метод более ресурсоемкий чем KFold. Существует правило, что вместо Leave One Out лучше использовать KFold на 5 или 10 фолдов.

In [32]:
```
# 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

In [33]:
```
scores2 = cross_val_score(KNeighborsClassifier(n_neighbors=12),
                          X, y,
                          cv=RepeatedKFold(n_splits=5, n_repeats=2))
```

In [34]:
```
scores2
```

Out[34]:
```
array([0.67213115, 0.70491803, 0.6557377 , 0.65      , 0.7       ,
       0.6557377 , 0.57377049, 0.60655738, 0.63333333, 0.68333333])
```

Обучение с оптимальным K

In [35]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [36]:
```
knn = KNeighborsClassifier(n_neighbors=12)
knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

Out[36]:
```
0.6542056074766355
```

Построение кривых обучения

In [37]:
```
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=0.1,
                     color="r")
    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="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 [38]:
```
plot_learning_curve(KNeighborsClassifier(n_neighbors=12), 'n_neighbors
                    X_train, y_train, cv=5)
```

Out[38]:
```
<module 'matplotlib.pyplot' from '/anaconda3/lib/python3.7/site-pack
ages/matplotlib/pyplot.py'>
```

Построение кривой валидации

In [39]:
```
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_rang
        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 sc
                 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
```

In [40]:
```
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")
```

Out[40]:
```
<module 'matplotlib.pyplot' from '/anaconda3/lib/python3.7/site-packages/matplotlib/pyplot.py'>
```