**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №3

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

Выполнил: Проверил:

студент группы РТ5-61Б преподаватель каф. ИУ5

Агеев Алексей Гапанюк Ю.Е.

Подпись и дата: Подпись и дата:

Москва, 2023 г

# Описание задания

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
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 подходящих для задачи метрик.
5. Произведите подбор гиперпараметра K с использованием GridSearchCV и RandomizedSearchCV и кросс-валидации, оцените качество оптимальной модели. Используйте не менее двух стратегий кросс-валидации.
6. Сравните метрики качества исходной и оптимальной моделей.

# Ход работы

# Подбор гиперпараметров модели и кросс-валидация

from IPython.display import Image  
import numpy as np  
import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.datasets import load\_wine  
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier  
from sklearn.model\_selection import cross\_val\_score, cross\_validate  
from sklearn.model\_selection import KFold, RepeatedKFold, LeaveOneOut, LeavePOut, ShuffleSplit, StratifiedKFold  
from sklearn.metrics import accuracy\_score, balanced\_accuracy\_score  
from sklearn.metrics import ConfusionMatrixDisplay  
from sklearn.metrics import precision\_score, recall\_score, f1\_score, classification\_report  
from sklearn.metrics import confusion\_matrix  
from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, mean\_squared\_log\_error, median\_absolute\_error, r2\_score   
from sklearn.metrics import roc\_curve, roc\_auc\_score  
from sklearn.model\_selection import GridSearchCV, RandomizedSearchCV  
import seaborn as sns  
import matplotlib.pyplot as plt  
%matplotlib inline   
sns.set(style="ticks")

wine = load\_wine()

# Разделение выборки на обучающую и тестовую  
wine\_X\_train, wine\_X\_test, wine\_y\_train, wine\_y\_test = train\_test\_split(  
 wine.data, wine.target, test\_size=0.3, random\_state=1)

# В моделях k-ближайших соседей большое значение k   
# ведёт к большому смещению и низкой дисперсии (недообучению)  
# 20 ближайших соседей  
cl1\_1 = KNeighborsClassifier(n\_neighbors=10)  
cl1\_1.fit(wine\_X\_train, wine\_y\_train)  
target1\_0 = cl1\_1.predict(wine\_X\_train)  
target1\_1 = cl1\_1.predict(wine\_X\_test)  
accuracy\_score(wine\_y\_train, target1\_0), accuracy\_score(wine\_y\_test, target1\_1)

(0.7580645161290323, 0.6666666666666666)

def accuracy\_score\_for\_classes(  
 y\_true: np.ndarray,  
 y\_pred: np.ndarray) -> {int:float}:  
 """  
 Вычисление метрики accuracy для каждого класса  
 y\_true - истинные значения классов  
 y\_pred - предсказанные значения классов  
 Возвращает словарь: ключ - метка класса,   
 значение - Accuracy для данного класса  
 """  
 # Для удобства фильтрации сформируем Pandas DataFrame   
 d = {'t': y\_true, 'p': y\_pred}  
 df = pd.DataFrame(data=d)  
 # Метки классов  
 classes = np.unique(y\_true)  
 # Результирующий словарь  
 res = dict()  
 # Перебор меток классов  
 for c in classes:  
 # отфильтруем данные, которые соответствуют   
 # текущей метке класса в истинных значениях  
 temp\_data\_flt = df[df['t']==c]  
 # расчет accuracy для заданной метки класса  
 temp\_acc = accuracy\_score(  
 temp\_data\_flt['t'].values,   
 temp\_data\_flt['p'].values)  
 # сохранение результата в словарь  
 res[c] = temp\_acc  
 return res  
  
def print\_accuracy\_score\_for\_classes(  
 y\_true: np.ndarray,   
 y\_pred: np.ndarray):  
 """  
 Вывод метрики accuracy для каждого класса  
 """  
 accs = accuracy\_score\_for\_classes(y\_true, y\_pred)  
 if len(accs)>0:  
 print('Метка \t Accuracy')  
 for i in accs:  
 print('{} \t {}'.format(i, accs[i]))

print\_accuracy\_score\_for\_classes(wine\_y\_test, target1\_1)

Метка Accuracy  
0 0.782608695652174  
1 0.6842105263157895  
2 0.4166666666666667

# Матрица ошибок

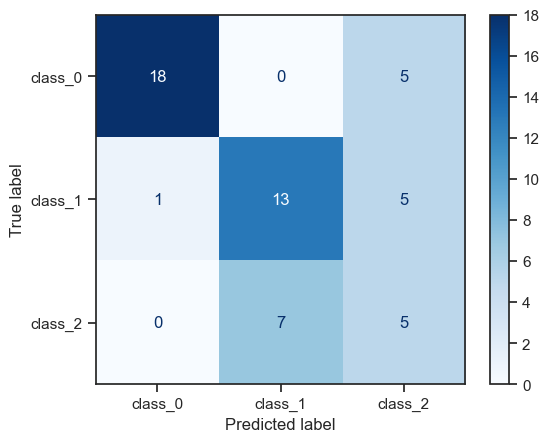
# Пример для небинарной классификации  
cm = confusion\_matrix(wine\_y\_test, target1\_1, labels=[0, 1, 2])

cm

array([[18, 0, 5],  
 [ 1, 13, 5],  
 [ 0, 7, 5]], dtype=int64)

ConfusionMatrixDisplay(cm, display\_labels=wine.target\_names).plot(cmap="Blues")

<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x167190e3af0>



precision\_score(wine\_y\_test, target1\_1, average='micro')

0.6666666666666666

precision\_score(wine\_y\_test, target1\_1, average='macro')

0.6435672514619882

precision\_score(wine\_y\_test, target1\_1, average='weighted')

0.7062865497076023

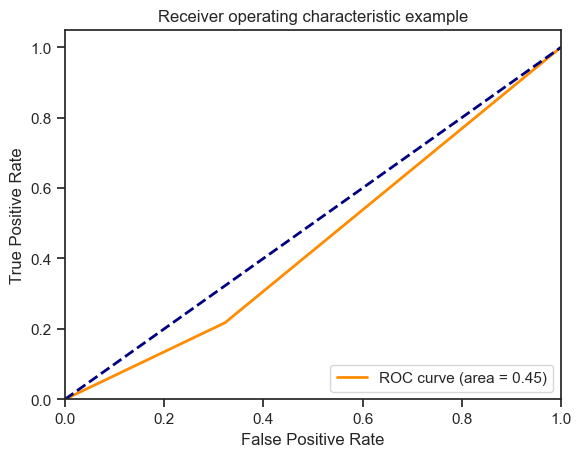
# ROC кривые

# Отрисовка ROC-кривой  
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='darkorange',  
 lw=lw, label='ROC curve (area = %0.2f)' % roc\_auc\_value)  
 plt.plot([0, 1], [0, 1], color='navy', 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()

def convert\_target\_to\_binary(array:np.ndarray, target:int) -> np.ndarray:  
 # Если целевой признак совпадает с указанным, то 1 иначе 0  
 res = [1 if x==target else 0 for x in array]  
 return res

bin\_cl1\_1 = KNeighborsClassifier(n\_neighbors=10)  
bin\_wine\_y\_train = convert\_target\_to\_binary(wine\_y\_train, 0)  
bin\_wine\_y\_test = convert\_target\_to\_binary(wine\_y\_test, 0)  
bin\_cl1\_1.fit(wine\_X\_train, bin\_wine\_y\_train)  
bin\_cl1\_1.predict(wine\_X\_test)  
  
bin\_target1\_1 = convert\_target\_to\_binary(target1\_1, 2)

draw\_roc\_curve(bin\_wine\_y\_test, bin\_target1\_1, pos\_label=1, average='micro')



# GridSearch

n\_range = np.array(range(5,40,5))  
tuned\_parameters = [{'n\_neighbors': n\_range}]  
tuned\_parameters

[{'n\_neighbors': array([ 5, 10, 15, 20, 25, 30, 35])}]

clf\_gs = GridSearchCV(KNeighborsClassifier(), tuned\_parameters, cv=5, scoring='accuracy')  
clf\_gs.fit(wine\_X\_train, wine\_y\_train)

GridSearchCV(cv=5, estimator=KNeighborsClassifier(),  
 param\_grid=[{'n\_neighbors': array([ 5, 10, 15, 20, 25, 30, 35])}],  
 scoring='accuracy')

clf\_gs.best\_score\_, clf\_gs.best\_params\_

(0.7340000000000001, {'n\_neighbors': 20})

# Randomized Search

clf\_rs = RandomizedSearchCV(KNeighborsClassifier(), tuned\_parameters, cv=5, scoring='accuracy')  
clf\_rs.fit(wine\_X\_train, wine\_y\_train)

C:\Users\prite\anaconda3\lib\site-packages\sklearn\model\_selection\\_search.py:305: UserWarning: The total space of parameters 7 is smaller than n\_iter=10. Running 7 iterations. For exhaustive searches, use GridSearchCV.  
 warnings.warn(

RandomizedSearchCV(cv=5, estimator=KNeighborsClassifier(),  
 param\_distributions=[{'n\_neighbors': array([ 5, 10, 15, 20, 25, 30, 35])}],  
 scoring='accuracy')

clf\_rs.best\_score\_, clf\_rs.best\_params\_

(0.7340000000000001, {'n\_neighbors': 20})

# Кросс валидация

## KFold

kf = KFold(n\_splits=3)  
scores = cross\_val\_score(clf\_rs.best\_estimator\_,   
 wine.data, wine.target, cv=kf)

# Значение метрики accuracy для 3 фолдов  
scores

array([0.01666667, 0. , 0.18644068])

## Shuffle split

# Эквивалент KFold(n\_splits=n)  
kf\_sp = ShuffleSplit(n\_splits=5, test\_size=0.25)

scores\_sp = cross\_val\_score(clf\_rs.best\_estimator\_,   
 wine.data, wine.target, cv=kf)

scores\_sp

array([0.01666667, 0. , 0.18644068])

# Метрики оптимальной модели

print\_accuracy\_score\_for\_classes(wine\_y\_test, target1\_1)

Метка Accuracy  
0 0.782608695652174  
1 0.6842105263157895  
2 0.4166666666666667

target2 = clf\_rs.predict(wine\_X\_test)

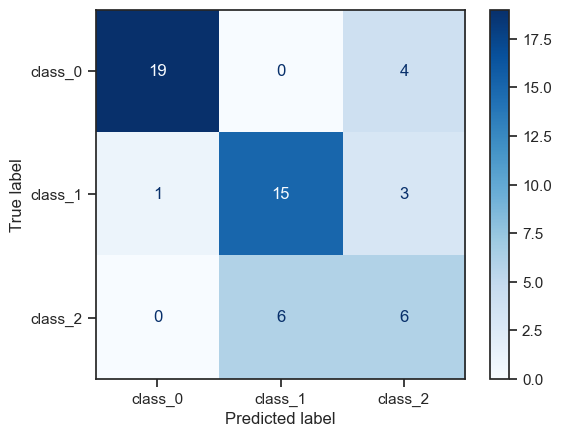
cm = confusion\_matrix(wine\_y\_test, target2, labels=[0, 1, 2])

cm

array([[19, 0, 4],  
 [ 1, 15, 3],  
 [ 0, 6, 6]], dtype=int64)

ConfusionMatrixDisplay(cm, display\_labels=wine.target\_names).plot(cmap="Blues")

<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x16719387e80>



precision\_score(wine\_y\_test, target2, average='micro')

0.7407407407407407

precision\_score(wine\_y\_test, target2, average='macro')

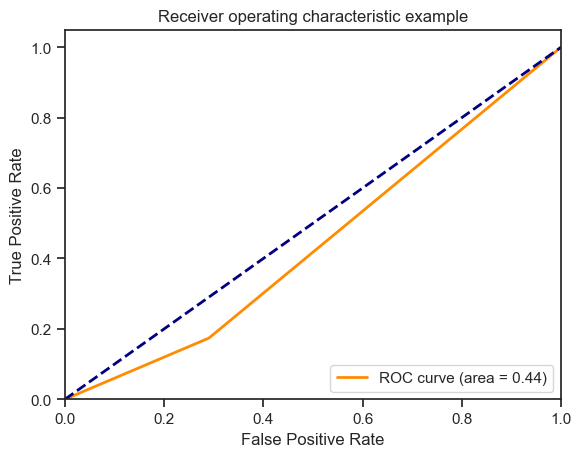
0.7086080586080586

precision\_score(wine\_y\_test, target1\_1, average='weighted')

0.7062865497076023

bin\_clf\_rs = KNeighborsClassifier(n\_neighbors=20)  
bin\_wine\_y\_train = convert\_target\_to\_binary(wine\_y\_train, 0)  
bin\_wine\_y\_test = convert\_target\_to\_binary(wine\_y\_test, 0)  
bin\_clf\_rs.fit(wine\_X\_train, bin\_wine\_y\_train)  
bin\_clf\_rs.predict(wine\_X\_test)  
  
bin\_target2 = convert\_target\_to\_binary(target2, 2)

draw\_roc\_curve(bin\_wine\_y\_test, bin\_target2, pos\_label=1, average='micro')



# Вывод

Отличия оптимальной модели от исходной:

* имеет accuracy по каждому классу выше исходной
* совершает меньше ошибок (видно на матрице ошибок)

имеет ROC - кривую ниже, что говорит о больших ошибках перво