**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика с системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №3

Выполнил: Проверил:

студент группы РТ5-61Б преподаватель каф. ИУ5

Кузнецов А.В. Гапанюк Ю.Е.

Подпись и дата: Подпись и дата:

2023 г.

## Описание датасета

Потребление мяса было важной частью рациона человека на протяжении тысячелетий, обеспечивая богатый источник белка и питательных веществ.Однако количество и виды потребляемого мяса сильно различались в разных культурах и исторических периодах.Чтобы лучше понять особенности потребления мяса потребление мяса во всем мире с течением времени, был составлен набор данных, который включает информацию об историческом потреблении мяса в различных регионах и странах.

import pandas as pd  
import numpy as np  
  
filename = '/Users/kovalenkov/Documents/Бауманка/3 курс/6 семестр/ТМО/Machine\_learning\_technologies/meat\_consumption.csv'  
ds = pd.read\_csv(filename)  
  
pd.set\_option('display.max\_colwidth', None)  
pd.set\_option('display.float\_format', '{:.2f}'.format)  
ds.columns = ['Локация', 'Индикатор', 'Тип мяса' , 'Количество', 'Категория', 'Год', 'Значение']  
ds = pd.DataFrame(ds)  
display(ds)

Локация Индикатор Тип мяса Количество Категория Год Значение  
0 AUS MEATCONSUMP BEEF KG\_CAP A 1990 0.00  
1 AUS MEATCONSUMP BEEF KG\_CAP A 1991 27.81  
2 AUS MEATCONSUMP BEEF KG\_CAP A 1992 26.28  
3 AUS MEATCONSUMP BEEF KG\_CAP A 1993 26.24  
4 AUS MEATCONSUMP BEEF KG\_CAP A 1994 25.54  
... ... ... ... ... ... ... ...  
12135 EU27 MEATCONSUMP SHEEP THND\_TONNE A 2024 732.15  
12136 EU27 MEATCONSUMP SHEEP THND\_TONNE A 2025 737.01  
12137 EU27 MEATCONSUMP SHEEP THND\_TONNE A 2026 741.10  
12138 EU27 MEATCONSUMP SHEEP THND\_TONNE A 2027 743.49  
12139 EU27 MEATCONSUMP SHEEP THND\_TONNE A 2028 747.01  
  
[12140 rows x 7 columns]

В датасете не содержится пустых данных. Кодирование категориальных признаков было сделано с помощью Label Encoder. Данный тип кодирования является наиболее часто используемым, преобразование представляет собой однозначное соответствие число <-> уникальное значение категориального признака.

from sklearn.preprocessing import LabelEncoder  
labelencoder = LabelEncoder()  
ds['Тип мяса'] = labelencoder.fit\_transform(ds['Тип мяса'].values)  
ds['Индикатор'] = labelencoder.fit\_transform(ds['Индикатор'].values)  
ds['Категория'] = labelencoder.fit\_transform(ds['Категория'].values)  
ds['Количество'] = labelencoder.fit\_transform(ds['Количество'].values)  
ds['Локация'] = labelencoder.fit\_transform(ds['Локация'].values)  
display(ds)

Локация Индикатор Тип мяса Количество Категория Год Значение  
0 1 0 0 0 0 1990 0.00  
1 1 0 0 0 0 1991 27.81  
2 1 0 0 0 0 1992 26.28  
3 1 0 0 0 0 1993 26.24  
4 1 0 0 0 0 1994 25.54  
... ... ... ... ... ... ... ...  
12135 11 0 3 1 0 2024 732.15  
12136 11 0 3 1 0 2025 737.01  
12137 11 0 3 1 0 2026 741.10  
12138 11 0 3 1 0 2027 743.49  
12139 11 0 3 1 0 2028 747.01  
  
[12140 rows x 7 columns]

### С использованием метода train\_test\_split разделите выборку на обучающую и тестовую.

Размер тестовой и обучающей выборки 1 к 2.

from sklearn.model\_selection import train\_test\_split  
  
X\_train, X\_test= train\_test\_split(ds,train\_size=0.67, random\_state=42)  
  
print(f"Обучающая выборка:\n{X\_train}")  
print(f"Тестовая выборка:\n{X\_test}")

Обучающая выборка:  
 Локация Индикатор Тип мяса Количество Категория Год Значение  
9713 37 0 3 1 0 2020 15301.95  
8165 27 0 3 1 0 1993 26.91  
7877 0 0 0 1 0 2017 2507.00  
9885 38 0 1 1 0 1997 129.95  
10586 37 0 2 1 0 1996 46260.26  
... ... ... ... ... ... ... ...  
11964 12 0 3 0 0 2009 4.92  
5191 11 0 1 0 0 2022 35.21  
5390 25 0 2 0 0 2026 31.53  
860 24 0 2 0 0 1992 15.30  
7270 36 0 3 1 0 1995 4.17  
  
[8133 rows x 7 columns]  
Тестовая выборка:  
 Локация Индикатор Тип мяса Количество Категория Год Значение  
6104 23 0 2 1 0 1999 36.75  
10538 21 0 2 1 0 2026 2077.83  
396 17 0 2 0 0 1996 12.97  
11146 5 0 2 1 0 2010 124.26  
7534 2 0 3 1 0 2025 136.13  
... ... ... ... ... ... ... ...  
9920 3 0 3 1 0 1993 2654.93  
2510 13 0 0 0 0 2016 1.90  
4870 36 0 1 0 0 2013 28.60  
2273 10 0 2 0 0 2010 0.59  
6081 32 0 1 1 0 2015 913.00  
  
[4007 rows x 7 columns]

from sklearn.neighbors import KNeighborsClassifier   
predictors = ['Локация', 'Категория']   
outcome = 'Количество'   
print(X\_train)  
  
new\_record = X\_train.loc[:, predictors]   
X = X\_train.loc[1:, predictors]   
y = X\_train.loc[1:, outcome]   
  
kNN = KNeighborsClassifier(n\_neighbors=20)   
kNN.fit(X, y)   
kNN.predict(new\_record)  
print(kNN.predict\_proba(new\_record))

Локация Индикатор Тип мяса Количество Категория Год Значение  
9713 37 0 3 1 0 2020 15301.95  
8165 27 0 3 1 0 1993 26.91  
7877 0 0 0 1 0 2017 2507.00  
9885 38 0 1 1 0 1997 129.95  
10586 37 0 2 1 0 1996 46260.26  
... ... ... ... ... ... ... ...  
11964 12 0 3 0 0 2009 4.92  
5191 11 0 1 0 0 2022 35.21  
5390 25 0 2 0 0 2026 31.53  
860 24 0 2 0 0 1992 15.30  
7270 36 0 3 1 0 1995 4.17  
  
[8133 rows x 7 columns]  
[[0.7 0.3 ]  
 [0.6 0.4 ]  
 [0.45 0.55]  
 ...  
 [0.45 0.55]  
 [0.6 0.4 ]  
 [0.55 0.45]]

### Произведите подбор гиперпараметра K с использованием GridSearchCV и RandomizedSearchCV и кросс-валидации, оцените качество оптимальной модели.

Используйте не менее двух стратегий кросс-валидации.

from sklearn.model\_selection import cross\_val\_score, cross\_validate  
scores = cross\_val\_score(KNeighborsClassifier(n\_neighbors=2),   
 ds, ds['Локация'], cv=3)  
print(scores)

[0.27378305 0.21126761 0.17498764]

Значение метрики accuracy для 3 фолдов

np.mean(scores)

0.8339377408545651

Усредненное значение метрики accuracy для 3 фолдов

scores = cross\_val\_score(KNeighborsClassifier(n\_neighbors=2),   
 ds, ds['Локация'], cv=3,  
 scoring='f1\_weighted')  
print(scores, np.mean(scores))

[0.26692239 0.20562052 0.17863725] 0.21706005353021074

Использование метрики f1

scoring = {'precision': 'precision\_weighted',   
 'recall': 'recall\_weighted',  
 'f1': 'f1\_weighted'}  
  
scores = cross\_validate(KNeighborsClassifier(n\_neighbors=2),   
 ds, ds['Локация'], scoring=scoring,   
 cv=3, return\_train\_score=True)  
scores

{'fit\_time': array([0.00640798, 0.00781107, 0.00803304]),  
 'score\_time': array([0.10710096, 0.15897202, 0.16736889]),  
 'test\_precision': array([0.33010929, 0.22315103, 0.20495232]),  
 'train\_precision': array([0.85534386, 0.90023732, 0.94270171]),  
 'test\_recall': array([0.27378305, 0.21126761, 0.17498764]),  
 'train\_recall': array([0.8480168 , 0.89682442, 0.94131455]),  
 'test\_f1': array([0.26692239, 0.20562052, 0.17863725]),  
 'train\_f1': array([0.84726898, 0.89641442, 0.94118126])}

В отличие от функции cross\_val\_score, функция cross\_validate позволяет использовать для оценки несколько метрик и возращает более детальную информацию.

##### Стратегия K-fold

Данная стратегия работает в соответствии с определением кросс-валидации.

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.

from sklearn.model\_selection import KFold  
X = ["a", "b", "c"]  
kf = KFold(n\_splits=3)  
for train, test in kf.split(X):  
 print("%s %s" % (train, test))

[1 2] [0]  
[0 2] [1]  
[0 1] [2]

Далее итератор может быть использован в функциях cross\_val\_score и cross\_validate:

kf = KFold(n\_splits=5)  
scores = cross\_val\_score(KNeighborsClassifier(n\_neighbors=2),   
 ds, ds['Локация'], scoring='f1\_weighted',   
 cv=kf)  
print(scores)

[0.15311171 0.13647194 0.10800428 0.13971723 0.12446281]

kf = KFold(n\_splits=5)  
scores = cross\_validate(KNeighborsClassifier(n\_neighbors=2),   
 ds, ds['Локация'], scoring=scoring,   
 cv=kf, return\_train\_score=True)  
scores

/usr/local/lib/python3.9/site-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.9/site-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.9/site-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.9/site-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.9/site-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.9/site-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.9/site-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.9/site-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.9/site-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.9/site-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))

{'fit\_time': array([0.01108813, 0.00729704, 0.01269603, 0.00825405, 0.00961161]),  
 'score\_time': array([0.06818795, 0.06709313, 0.08138776, 0.08531499, 0.08978724]),  
 'test\_precision': array([0.46862436, 0.29006464, 0.21143049, 0.34004254, 0.16902537]),  
 'train\_precision': array([0.87083679, 0.87056821, 0.89995913, 0.92913181, 0.915602 ]),  
 'test\_recall': array([0.11861614, 0.12479407, 0.07825371, 0.11490939, 0.10955519]),  
 'train\_recall': array([0.86655684, 0.86521829, 0.89363674, 0.92637974, 0.9134061 ]),  
 'test\_f1': array([0.15311171, 0.13647194, 0.10800428, 0.13971723, 0.12446281]),  
 'train\_f1': array([0.86588585, 0.86452674, 0.89392474, 0.92631311, 0.91306137])}

##### Leave One Out (LOO)

from sklearn.model\_selection import KFold, RepeatedKFold, LeaveOneOut, LeavePOut, ShuffleSplit, StratifiedKFold  
X = range(12)  
kf = LeaveOneOut()  
for train, test in kf.split(X):  
 print("%s %s" % (train, test))

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

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 эмпирическое правило, что вместо Leave One Out лучше использовать KFold на 5 или 10 фолдов.

### Grid Search (решетчатый поиск)

n\_range = np.array(range(5,55,5))  
tuned\_parameters = [{'n\_neighbors': n\_range}]  
tuned\_parameters

[{'n\_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])}]

from sklearn.model\_selection import GridSearchCV, RandomizedSearchCV  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 ds, ds['Локация'], test\_size=0.5, random\_state=1)  
  
clf\_gs = GridSearchCV(KNeighborsClassifier(), tuned\_parameters, cv=5, scoring='accuracy')  
clf\_gs.fit(X\_train, y\_train)

GridSearchCV(cv=5, estimator=KNeighborsClassifier(),  
 param\_grid=[{'n\_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])}],  
 scoring='accuracy')

clf\_gs.cv\_results\_

{'mean\_fit\_time': array([0.00628695, 0.00453119, 0.00473919, 0.00434484, 0.00475979,  
 0.00469732, 0.00469728, 0.00478139, 0.00498343, 0.00439696]),  
 'std\_fit\_time': array([0.00101189, 0.00020343, 0.00076725, 0.00019757, 0.00032777,  
 0.00032818, 0.00045276, 0.00038913, 0.00068028, 0.000266 ]),  
 'mean\_score\_time': array([0.04920111, 0.03121157, 0.03213878, 0.03319583, 0.03541555,  
 0.03615222, 0.0367352 , 0.04176722, 0.03923354, 0.041994 ]),  
 'std\_score\_time': array([0.01696428, 0.00068079, 0.00103522, 0.00075006, 0.00184854,  
 0.00109733, 0.00065719, 0.00571699, 0.00191078, 0.00325244]),  
 'param\_n\_neighbors': masked\_array(data=[5, 10, 15, 20, 25, 30, 35, 40, 45, 50],  
 mask=[False, False, False, False, False, False, False, False,  
 False, False],  
 fill\_value='?',  
 dtype=object),  
 'params': [{'n\_neighbors': 5},  
 {'n\_neighbors': 10},  
 {'n\_neighbors': 15},  
 {'n\_neighbors': 20},  
 {'n\_neighbors': 25},  
 {'n\_neighbors': 30},  
 {'n\_neighbors': 35},  
 {'n\_neighbors': 40},  
 {'n\_neighbors': 45},  
 {'n\_neighbors': 50}],  
 'split0\_test\_score': array([0.48023064, 0.40444811, 0.36490939, 0.34102142, 0.32784185,  
 0.31136738, 0.29324547, 0.29324547, 0.29489292, 0.28336079]),  
 'split1\_test\_score': array([0.48187809, 0.4291598 , 0.39868204, 0.35667216, 0.33772652,  
 0.32454695, 0.31630972, 0.30971993, 0.28583196, 0.28995058]),  
 'split2\_test\_score': array([0.48105437, 0.41433278, 0.37067545, 0.3369028 , 0.31960461,  
 0.31136738, 0.30477759, 0.29242175, 0.276771 , 0.27594728]),  
 'split3\_test\_score': array([0.49670511, 0.41927512, 0.38714992, 0.37067545, 0.36079077,  
 0.34349259, 0.33443163, 0.32454695, 0.30971993, 0.30642504]),  
 'split4\_test\_score': array([0.45963756, 0.40856672, 0.38056013, 0.35667216, 0.32537068,  
 0.30889621, 0.29489292, 0.28747941, 0.27759473, 0.26112026]),  
 'mean\_test\_score': array([0.47990115, 0.41515651, 0.38039539, 0.3523888 , 0.33426689,  
 0.3199341 , 0.30873147, 0.3014827 , 0.28896211, 0.28336079]),  
 'std\_test\_score': array([0.01181806, 0.00862356, 0.01195279, 0.01216659, 0.01449753,  
 0.01299918, 0.01526357, 0.01375396, 0.01227762, 0.01499991]),  
 'rank\_test\_score': array([ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10], dtype=int32)}

Лучшая модель

clf\_gs.best\_estimator\_

KNeighborsClassifier()

Лучшее значение метрики

clf\_gs.best\_score\_

0.47990115321252064

Лучшее значение параметров

clf\_gs.best\_params\_

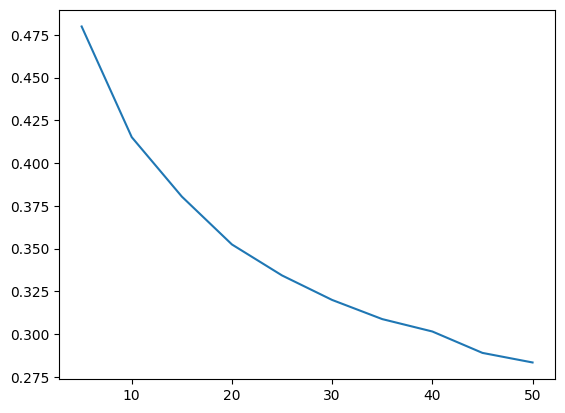
{'n\_neighbors': 5}

Изменение качества на тестовой выборке в зависимости от К-соседей

import matplotlib.pyplot as plt  
plt.plot(n\_range, clf\_gs.cv\_results\_['mean\_test\_score'])

Matplotlib is building the font cache; this may take a moment.

[<matplotlib.lines.Line2D at 0x12bfab9d0>]

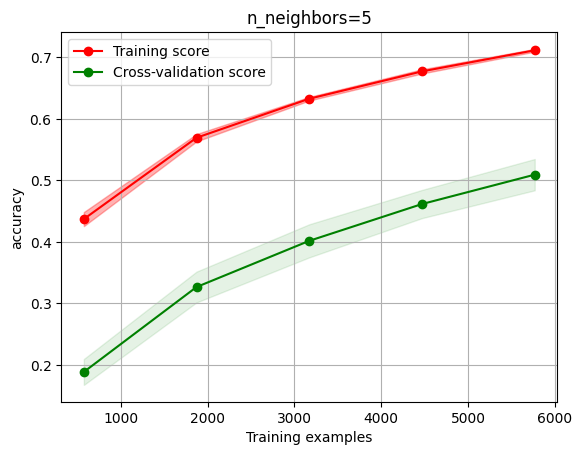


##### Сравните метрики качества исходной и оптимальной моделей.

Строится зависимость метрики на обучающей выборке от размера выборки.

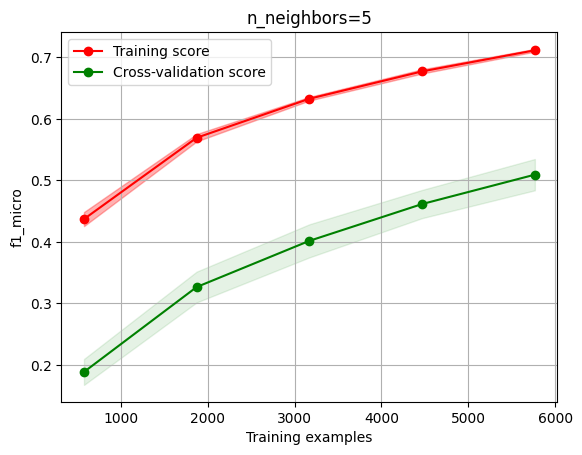
from sklearn.model\_selection import learning\_curve, validation\_curve  
def plot\_learning\_curve(estimator, title, X, y, ylim=None, cv=None,  
 n\_jobs=None, train\_sizes=np.linspace(.1, 1.0, 5), scoring='accuracy'):  
 plt.figure()  
 plt.title(title)  
 if ylim is not None:  
 plt.ylim(\*ylim)  
 plt.xlabel("Training examples")  
 plt.ylabel(scoring)  
 train\_sizes, train\_scores, test\_scores = learning\_curve(  
 estimator, X, y, cv=cv, scoring=scoring, 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="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=5), 'n\_neighbors=5',   
 X\_train, y\_train, cv=20)

<module 'matplotlib.pyplot' from '/usr/local/lib/python3.9/site-packages/matplotlib/pyplot.py'>



plot\_learning\_curve(KNeighborsClassifier(n\_neighbors=5), 'n\_neighbors=5',   
 X\_train, y\_train, cv=20, scoring='f1\_micro')

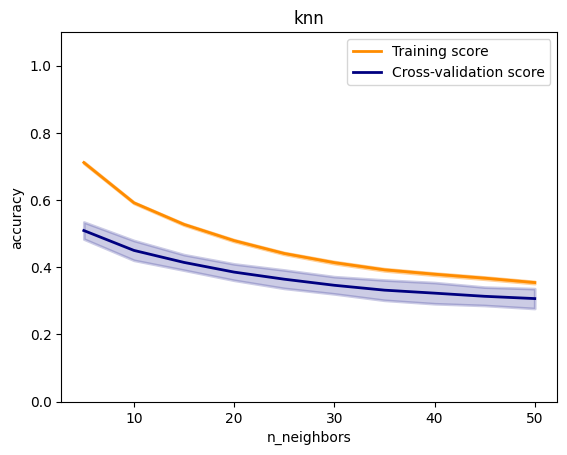
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.9/site-packages/matplotlib/pyplot.py'>



### Построение кривой валидации - validation\_curve

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="darkorange", lw=lw)  
 plt.fill\_between(param\_range, train\_scores\_mean - train\_scores\_std,  
 train\_scores\_mean + train\_scores\_std, alpha=0.4,  
 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  
  
plot\_validation\_curve(KNeighborsClassifier(), 'knn',   
 X\_train, y\_train,   
 param\_name='n\_neighbors', param\_range=n\_range,   
 cv=20, scoring="accuracy")

<module 'matplotlib.pyplot' from '/usr/local/lib/python3.9/site-packages/matplotlib/pyplot.py'>



n\_range2 = np.array(range(5,125,5))  
plot\_validation\_curve(clf\_gs.best\_estimator\_, 'knn',   
 ds, ds['Локация'],   
 param\_name='n\_neighbors', param\_range=n\_range2,   
 cv=20, scoring="accuracy")

<module 'matplotlib.pyplot' from '/usr/local/lib/python3.9/site-packages/matplotlib/pyplot.py'>

