|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ  Информатика и системы управления

КАФЕДРА Системы обработки информации и управления\_\_\_\_\_\_\_\_\_\_\_

Отчет по лабораторной работе № **3**

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

по дисциплине «Технологии машинного обучения»

Студент ИУ5-62Б  М.С. Вольвач

(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель  Ю.Е. Гапанюк

(Подпись, дата) (И.О.Фамилия)

**Москва**

**2024**

Цель работы:

Изучение способов подготовки выборки и подбора гиперпараметров на примере метода ближайших соседей.

Задание:

* Выберите набор данных (датасет) для решения задачи классификации или регрессии.
* В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
* С использованием метода train\_test\_split разделите выборку на обучающую и тестовую.
* Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью подходящих для задачи метрик.
* Произведите подбор гиперпараметра K с использованием GridSearchCV и RandomizedSearchCV и кросс-валидации, оцените качество оптимальной модели. Используйте не менее двух стратегий кросс-валидации.
* Сравните метрики качества исходной и оптимальной моделей.

Описание набора данных:

* RI - Refractive Index - коэффициент преломления
* Na - Sodium - Содержание натрия (массовый процент в соответствующем оксиде)
* Mg - Magnesium - Содержание магния
* Al - Aluminum - Содержание алюминия
* Si - Silicon - Содержание кремния
* K - Potassium - Содержание калия
* Ca - Calcium - Содержание кальция
* Ba - Barium - Содеражние бария
* Fe - Iron - Содержание железа
* Type - Type of glass - тип стекла (1, 2 - стекла для зданий, 3, 4 - стекла для автомобилей, 5 - стеклотара, 6 - tableware - бытовые стекла, 7 - стекла для ламп; 4 отсутствует в данном наборе данных)

Код программы:

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

sns.set(style="ticks")

data = pd.read\_csv('glass.csv')

data.head()

X = data.drop("Type", axis=1)

y = data["Type"]

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=1)

from typing import Dict, Tuple

def type\_proportions(array: np.ndarray) -> Dict[int, Tuple[int, float]]:

labels, counts = np.unique(array, return\_counts=True)

counts\_perc = counts/array.size

res = dict()

for label, count2 in zip(labels, zip(counts, counts\_perc)):

res[label] = count2

return res

def print\_type\_proportions(array: np.ndarray):

proportions = type\_proportions(array)

if len(proportions)>0:

print('Тип \t Количество \t Процент встречаемости')

for i in proportions:

val, val\_perc = proportions[i]

val\_perc\_100 = round(val\_perc \* 100, 2)

print('{} \t {} \t \t {}%'.format(i, val, val\_perc\_100))

print\_type\_proportions(data.Type)

print\_type\_proportions(y\_train)

print\_type\_proportions(y\_test)

from sklearn.neighbors import KNeighborsClassifier

clf\_i = KNeighborsClassifier(n\_neighbors=20)

clf\_i.fit(X\_train, y\_train)

target\_i = clf\_i.predict(X\_test)

len(target\_i), target\_i

from sklearn.model\_selection import cross\_val\_score, cross\_validate

scores = cross\_val\_score(KNeighborsClassifier(n\_neighbors=2),

data, data.Type, cv=3)

scoring = {'precision': 'precision\_weighted',

'recall': 'recall\_weighted',

'f1': 'f1\_weighted'}

scores = cross\_validate(KNeighborsClassifier(n\_neighbors=2),

data, data.Type, scoring='f1\_weighted',

cv=3, return\_train\_score=True)

from sklearn.model\_selection import KFold

kf = KFold(n\_splits=5)

scores = cross\_validate(KNeighborsClassifier(n\_neighbors=2),

data, data.Type, scoring='f1\_weighted',

cv=kf)

scores

from sklearn.model\_selection import RepeatedKFold

kf = RepeatedKFold(n\_splits=3, n\_repeats=2)

scores = cross\_validate(KNeighborsClassifier(n\_neighbors=2),

data, data.Type, scoring='f1\_weighted',

cv=kf)

scores

from sklearn.model\_selection import LeaveOneOut

kf = LeaveOneOut()

scores = cross\_validate(KNeighborsClassifier(n\_neighbors=2),

data, data.Type, scoring='f1\_weighted',

cv=kf)

scores

from sklearn.model\_selection import LeavePOut

kf = LeavePOut(2)

scores = cross\_validate(KNeighborsClassifier(n\_neighbors=2),

data, data.Type, scoring='f1\_weighted',

cv=kf)

scores

from sklearn.model\_selection import ShuffleSplit

kf = ShuffleSplit(n\_splits=5, test\_size=0.25)

scores = cross\_validate(KNeighborsClassifier(n\_neighbors=2),

data, data.Type, scoring='f1\_weighted',

cv=kf)

scores

from sklearn.model\_selection import StratifiedKFold

skf = StratifiedKFold(n\_splits=3)

scores = cross\_validate(KNeighborsClassifier(n\_neighbors=2),

data, data.Type, scoring='f1\_weighted',

cv=skf)

scores

from sklearn.model\_selection import GridSearchCV

n\_range = np.array(range(5,55,5))

tuned\_parameters = [{'n\_neighbors': n\_range}]

tuned\_parameters

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

Результат работы программы:

