Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления» – ИУ5

Факультет «Радиотехнический» – РТ5

**Отчёт по лабораторной работе №3 по курсу**

**Технологии машинного обучения**

5

(количество листов)

|  |  |  |
| --- | --- | --- |
| Исполнитель |  |  |
| студент группы РТ5-61б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Нижаметдинов М. Ш. |
|  |  | “\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  |  |  |
| Проверил |  |  |
| Преподаватель кафедры ИУ5 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Гапанюк Ю. Е. |
|  |  | “\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2023 г. |

Москва, 2023 г.

**Задание**

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. Сравните метрики качества исходной и оптимальной моделей.

**Набор данных**

<https://scikit-learn.org/stable/datasets/toy_dataset.html#wine-recognition-dataset>

**Исходный текст проекта**

## Задание

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.

Сравните метрики качества исходной и оптимальной моделей.

## Ход работы

### Выбор и загрузка датасета

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

sns.set(style="ticks")

from sklearn.datasets import \*

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import accuracy\_score, balanced\_accuracy\_score

from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split, GridSearchCV, RandomizedSearchCV

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

from sklearn.model\_selection import KFold, RepeatedKFold, LeaveOneOut, LeavePOut, ShuffleSplit, StratifiedKFold

def make\_dataframe(ds\_function):

ds = ds\_function()

df = pd.DataFrame(data= np.c\_[ds['data'], ds['target']],

columns= list(ds['feature\_names']) + ['target'])

return df

wine = load\_wine()

df = make\_dataframe(load\_wine)

# Первые 5 строк датасета

df.head()

# Проверим наличие пустых значений

# Цикл по колонкам датасета

for col in df.columns:

# Количество пустых значений - все значения заполнены

temp\_null\_count = df[df[col].isnull()].shape[0]

print('{} - {}'.format(col, temp\_null\_count))

# Масштабирование

scaler = MinMaxScaler()

df[['alcohol', 'hue']] = scaler.fit\_transform(df[['alcohol', 'hue']])

data = df[['alcohol','hue','target']]

# data = pd.DataFrame(scaler.transform(df[['alcohol', 'hue']]), columns = ['alcohol', 'hue'])

data.head()

data.describe().T

# Диаграмма рассеяния позволяет визуально обнаружить наличие зависимости

# Построим зависимость между крепкостью алкоголя и оттенком цвета

fig, ax = plt.subplots(figsize=(10,10))

sns.scatterplot(ax=ax, x='alcohol', y='hue', data=data, hue='target')

### Разделение на тестовую и обучающую

y = data['target']

x = data.drop('target', axis = 1)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.20, random\_state = 20)

print(f"Обучающая выборка:\n{x\_train, y\_train}")

print(f"Тестовая выборка:\n{x\_test, y\_test}")

### Обучение и оценка качества модели для произвольного гиперпараметра K

cl1\_5 = KNeighborsClassifier(n\_neighbors=5).fit(x\_train, y\_train)

target1\_5\_train = cl1\_5.predict(x\_train)

target1\_5\_test = cl1\_5.predict(x\_test)

accuracy\_score(y\_train, target1\_5\_train), accuracy\_score(y\_test, target1\_5\_test)

Чем выше значения - тем лучше

### Подбор гиперпараметров модели и кросс-валидация

#### Grid Search

n\_range = np.array(range(5,31,1))

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

tuned\_parameters

%%time

clf\_gs = GridSearchCV(KNeighborsClassifier(), tuned\_parameters, cv=5, scoring='accuracy')

clf\_gs.fit(x\_train, y\_train)

# Лучшая модель

clf\_gs.best\_estimator\_

# Лучшее значение метрики

clf\_gs.best\_score\_

# Лучшее значение параметров

clf\_gs.best\_params\_

# Изменение качества на тестовой выборке в зависимости от К-соседей

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

#### Randomized Search

%%time

clf\_rs = RandomizedSearchCV(KNeighborsClassifier(), tuned\_parameters, cv=5, scoring='accuracy')

clf\_rs.fit(x\_train, y\_train)

# В данном случае оба способа нашли одинаковое решение

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

#### K-fold

X = data

kf = KFold(n\_splits=10)

for train, test in kf.split(X):

print("%s %s" % (train, test))

kf = KFold(n\_splits=10)

scores = cross\_val\_score(KNeighborsClassifier(n\_neighbors=13),

x, y, scoring='accuracy',

cv=kf)

scores

kf = KFold(n\_splits=10)

scores = cross\_validate(KNeighborsClassifier(n\_neighbors=13),

x, y, scoring='accuracy',

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

scores

#### ShuffleSplit

X = data

# Эквивалент KFold(n\_splits=n)

kf = ShuffleSplit(n\_splits=10, test\_size=0.25)

for train, test in kf.split(X):

print("%s %s" % (train, test))

kf = ShuffleSplit(n\_splits=10, test\_size=0.25)

scores = cross\_val\_score(KNeighborsClassifier(n\_neighbors=13),

x, y, scoring='accuracy',

cv=kf)

scores

kf = ShuffleSplit(n\_splits=10, test\_size=0.25)

scores = cross\_validate(KNeighborsClassifier(n\_neighbors=13),

x, y, scoring='accuracy',

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

scores