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
```