Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа
по дисциплине
«Методы машинного обучения»
на тему
«Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей»

Выполнил:
студент группы ИУ5-64Б
Зубков А. Д.

Москва — 2020 г.

1. Цель лабораторной работы

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

2. Задание

1. Выбрать набор данных (датасет) для решения задачи классификации или регрессии.
2. С использованием метода train_test_split разделить выборку на обучающую и тестовую.
3. Обучить модель k-ближайших соседей для произвольно заданного гиперпараметра K. Оценить качество модели с помощью подходящих для задачи метрик.
4. Построить модель и оценить качество модели с использованием кросс-валидации.
5. Произвести подбор гиперпараметра K с использованием GridSearchCV и кросс-валидации.

3. Ход выполнения лабораторной работы

Подключим необходимые библиотеки и загрузим набор данных

[ ]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score,␣
↪GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

%matplotlib inline

# Устанавливаем тип графиков
sns.set(style="ticks")

# Для лучшего качествоа графиков
from IPython.display import set_matplotlib_formats
set_matplotlib_formats("retina")

# Устанавливаем ширину экрана для отчета
pd.set_option("display.width", 70)

# Загружаем данные
data = pd.read_csv('heart.csv')
data.head()
```

[ ]:
```
data.shape
```

[ ]:
```
data.dtypes
```

[ ]:
```
data.isna().sum()
```

[ ]:
```
data.isnull().sum()
```

Как видим, пустых значений нет, значет нет необходимости преобразовывать набор данных

Разделим данные на целевой столбец и признаки

[ ]:
```
X = data.drop("target", axis=1)
Y = data["target"]
print(X, "\n")
print(Y)
```

[ ]:
```
X.shape
```

[ ]:
```
Y.shape
```

С использованием метода train_test_split разделим выборку на обучающую и тестовую

[ ]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,␣
↪random_state=1)
```

[ ]:
```
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
print("Y_train:", Y_train.shape)
print("Y_test:", Y_test.shape)
```

Обучим модель k-ближайших соседей для произвольно заданного гиперпараметра K

[ ]:
```
# В моделях k-ближайших соседей большое значение k
# ведёт к большому смещению и низкой дисперсии (недообучению)
# 70 ближайших соседей
cl1_1 = KNeighborsClassifier(n_neighbors=70)
cl1_1.fit(X_train, Y_train)
target1_0 = cl1_1.predict(X_train)
target1_1 = cl1_1.predict(X_test)
accuracy_score(Y_train, target1_0), accuracy_score(Y_test, target1_1)
```

Построим модель и оценим качество модели с использованием кросс-валидации

[ ]:
```
scores = cross_val_score(KNeighborsClassifier(n_neighbors=2), X, Y, cv=3)
```

[ ]:
```
# Значение метрики accuracy для 3 фолдов
scores
```

[ ]:
```
# Усредненное значение метрики accuracy для 3 фолдов
np.mean(scores)
```

Произведем подбор гиперпараметра K с использованием GridSearchCV и кросс-валидации

[ ]:
```
# Список настраиваемых параметров
n_range = np.array(range(1, 50, 2))
tuned_parameters = [{'n_neighbors': n_range}]
n_range
```

[ ]:
```
%%time
clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=5,␣
 ↪scoring='accuracy', return_train_score=True)
clf_gs.fit(X, Y)
clf_gs.best_params_
```

Проверим результаты при разных значения гиперпараметра на тренировочном наборе данных:

[ ]:
```
plt.plot(n_range, clf_gs.cv_results_["mean_train_score"]);
```

Очевидно, что для $K = 1$ на тренировочном наборе данных мы находим ровно ту же точку, что и нужно предсказать, и чем больше её соседей мы берём — тем меньше точность.

Посмотрим на тестовом наборе данных

[ ]:
```
plt.plot(n_range, clf_gs.cv_results_["mean_test_score"]);
```

Проверим получившуюся модель:

[ ]:
```
cl1_2 = KNeighborsClassifier(**clf_gs.best_params_)
cl1_2.fit(X_train, Y_train)
target2_0 = cl1_2.predict(X_train)
target2_1 = cl1_2.predict(X_test)
accuracy_score(Y_train, target2_0), accuracy_score(Y_test, target2_1)
```

Как видим, точность модели улучшилось