Цель лабораторной работы

Изучить сложные способы подготовки выборки и подбора гиперпараметров на примере метода ближайших соседей.

Задание

Требуется выполнить следующие действия:

1. Выбрать набор данных (датасет) для решения задачи классификации или регресии.
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
5. Оцените качество модели с помощью подходящих для задачи метрик.
6. Постройте модель и оцените качество модели с использованием кросс-валидации. Проведите эксперименты с тремя различными стратегиями кросс-валидации.
7. Произведите подбор гиперпараметра с использованием GridSearchCV и кросс-валидации.
8. Сравните качество полученной модели с качеством модели, полученной в пункте 4.

Ход выполнения работы

```
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.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error,
median_absolute_error, r2_score
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split, GridSearchCV,
RandomizedSearchCV, cross_val_score

df = pd.read_csv("auto.csv")
```

```
df.head()
```

```
    mpg  cylinders  displacement horsepower  weight  acceleration
model year  \
0  18.0          8         307.0        130    3504          12.0
70
1  15.0          8         350.0        165    3693          11.5
70
2  18.0          8         318.0        150    3436          11.0
70
3  16.0          8         304.0        150    3433          12.0
70
4  17.0          8         302.0        140    3449          10.5
70

   origin                   car name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
```

```
df.shape
```

```
(398, 9)
```

```
df.describe().T
```

```
              count         mean         std     min       25%     50%
\
mpg           398.0    23.514573    7.815984     9.0    17.500    23.0

cylinders     398.0     5.454774    1.701004     3.0     4.000     4.0

displacement  398.0   193.425879  104.269838    68.0   104.250   148.5

weight        398.0  2970.424623  846.841774  1613.0  2223.750  2803.5

acceleration  398.0    15.568090    2.757689     8.0    13.825    15.5

model year    398.0    76.010050    3.697627    70.0    73.000    76.0

origin        398.0     1.572864    0.802055     1.0     1.000     1.0


                   75%     max
mpg             29.000    46.6
cylinders        8.000     8.0
displacement   262.000   455.0
weight        3608.000  5140.0
```

```
acceleration    17.175    24.8
model year      79.000    82.0
origin           2.000     3.0
```

```
df.dtypes
```

```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight            int64
acceleration    float64
model year        int64
origin            int64
car name         object
dtype: object
```

```
df = df[df.horsepower != '?']
```

```
df['horsepower'] = df['horsepower'].astype('int64')
```

```
df.describe().T
```

```
              count         mean          std     min       25%
50%  \
mpg           392.0    23.445918    7.805007     9.0    17.000
22.75
cylinders     392.0     5.471939    1.705783     3.0     4.000
4.00
displacement  392.0   194.411990  104.644004    68.0   105.000
151.00
horsepower    392.0   104.469388   38.491160    46.0    75.000
93.50
weight        392.0  2977.584184  849.402560  1613.0  2225.250
2803.50
acceleration  392.0    15.541327    2.758864     8.0    13.775
15.50
model year    392.0    75.979592    3.683737    70.0    73.000
76.00
origin        392.0     1.576531    0.805518     1.0     1.000
1.00

                   75%     max
mpg              29.000    46.6
cylinders         8.000     8.0
displacement    275.750   455.0
horsepower      126.000   230.0
weight         3614.750  5140.0
acceleration     17.025    24.8
model year       79.000    82.0
origin            2.000     3.0
```

```
df.corr()
```

```
                   mpg  cylinders  displacement  horsepower    weight
\
mpg           1.000000  -0.777618     -0.805127   -0.778427 -0.832244

cylinders    -0.777618   1.000000      0.950823    0.842983  0.897527

displacement -0.805127   0.950823      1.000000    0.897257  0.932994

horsepower   -0.778427   0.842983      0.897257    1.000000  0.864538

weight       -0.832244   0.897527      0.932994    0.864538  1.000000

acceleration  0.423329  -0.504683     -0.543800   -0.689196 -0.416839

model year    0.580541  -0.345647     -0.369855   -0.416361 -0.309120

origin        0.565209  -0.568932     -0.614535   -0.455171 -0.585005


              acceleration  model year    origin
mpg               0.423329    0.580541  0.565209
cylinders        -0.504683   -0.345647 -0.568932
displacement     -0.543800   -0.369855 -0.614535
horsepower       -0.689196   -0.416361 -0.455171
weight           -0.416839   -0.309120 -0.585005
acceleration      1.000000    0.290316  0.212746
model year        0.290316    1.000000  0.181528
origin            0.212746    0.181528  1.000000
```

При построении предсказательных моделей исходные данные обычно разбиваются на обучающую и контрольную выборки. Обучающая выборка используется для построения математических отношений между некоторой переменной-откликом и предикторами, тогда как контрольная выборка служит для получения оценки прогнозных свойств модели на новых данных, т.е. данных, которые не были использованы для обучения модели. В нашем случае обучающая выборка - это разгон автомобиля, а проверочная - это все остальные признаки, которые потенциально могут влиять него. Я использовала корреляционную матрицу для определения зависимостей. По ней мы можем наблюдать, что на разгон автомобиля так или иначе влияет расход топлива, количество цилиндров, объем двигателя, количество лошадиных сил и вес.

```
data = df.loc[:,
["mpg","cylinders","displacement","horsepower","weight",
"acceleration"]]
data.head()
```

```
    mpg  cylinders  displacement  horsepower  weight  acceleration
0  18.0          8         307.0         130    3504          12.0
1  15.0          8         350.0         165    3693          11.5
2  18.0          8         318.0         150    3436          11.0
3  16.0          8         304.0         150    3433          12.0
4  17.0          8         302.0         140    3449          10.5
```

Масштабирование

```
scaler = MinMaxScaler().fit(data)
data = pd.DataFrame(scaler.transform(data), columns = data.columns)

data.describe().T
```

```
              count      mean       std  min       25%       50%
75%  \
mpg           392.0  0.384200  0.207580  0.0  0.212766  0.365691
0.531915
cylinders     392.0  0.494388  0.341157  0.0  0.200000  0.200000
1.000000
displacement  392.0  0.326646  0.270398  0.0  0.095607  0.214470
0.536822
horsepower    392.0  0.317768  0.209191  0.0  0.157609  0.258152
0.434783
weight        392.0  0.386897  0.240829  0.0  0.173589  0.337539
0.567550
acceleration  392.0  0.448888  0.164218  0.0  0.343750  0.446429
0.537202

              max
mpg           1.0
cylinders     1.0
displacement  1.0
horsepower    1.0
weight        1.0
acceleration  1.0
```

Разделение данных

```
y = data['acceleration']
x = data.drop('acceleration', axis = 1)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.10, random_state = 20)

# Размер обучающей выборки
x_train.shape, y_train.shape

((352, 5), (352,))

# Размер тестовой выборки
x_test.shape, y_test.shape
```

```
((40, 5), (40,))
```

Модель для произвольно заданного гиперпараметра K

```
def test_model(model):
    print("mean_absolute_error:",
          mean_absolute_error(y_test, model.predict(x_test)))
    print("mean_squared_error:",
          mean_squared_error(y_test, model.predict(x_test)))
    print("median_absolute_error:",
          median_absolute_error(y_test, model.predict(x_test)))
    print("r2_score:",
          r2_score(y_test, model.predict(x_test)))
```

```
reg_10 = KNeighborsRegressor(n_neighbors = 10).fit(x_train, y_train)
```

```
test_model(reg_10)
```

```
mean_absolute_error: 0.09107142857142855
mean_squared_error: 0.013304705215419498
median_absolute_error: 0.06904761904761904
r2_score: 0.6587400480861256
```

- mean_absolute_error: 0.09, чем ближе значение к нулю, тем лучше качество регрессии.
- mean_squared_error: 0.01, чем ближе значение к нулю, тем лучше модель
- median_absolute_error: 0.07
- r2_score: 0.66, коэффициент детерминации для модели с константой принимает значения от 0 до 1. Чем ближе значение коэффициента к 1, тем сильнее зависимость.

Подбор гиперпараметров модели и кросс-валидация

Grid Search

```
knn_params = {"n_neighbors" : np.arange(1,11,1)}
knn = KNeighborsRegressor()
knn_cv_model = GridSearchCV(knn, knn_params, cv = 10)
knn_cv_model.fit(x_train, y_train)
```

```
GridSearchCV(cv=10, estimator=KNeighborsRegressor(),
             param_grid={'n_neighbors': array([ 1,  2,  3,  4,  5,  6,
7,  8,  9, 10])})
```

```
knn_cv_model.best_params_["n_neighbors"]
```

```
3
```

```
reg_grid = KNeighborsRegressor(n_neighbors =
knn_cv_model.best_params_["n_neighbors"]).fit(x_train, y_train)
```

```
test_model(reg_grid)
```

```
mean_absolute_error: 0.07792658730158732
mean_squared_error: 0.009056614543965735
median_absolute_error: 0.07043650793650796
r2_score: 0.767701742072848
```

Кросс-валидация

```
MSE = []
MSE_CV = []

for k in range(10):
    k = k + 1
    knn_model = KNeighborsRegressor(n_neighbors = k).fit(x_train,
y_train)
    y_pred = knn_model.predict(x_test)
    mse = mean_squared_error(y_pred, y_test)
    mse_cv = -1 * cross_val_score(knn_model, x_train, y_train, cv =
10,
                          scoring = "neg_mean_squared_error").mean()
    MSE.append(mse)
    MSE_CV.append(mse_cv)
    print("k =", k, "MSE :", mse, "MSE_CV:", mse_cv)
```

```
k = 1 MSE : 0.010318346088435373 MSE_CV: 0.013936683965914406
k = 2 MSE : 0.008399101828231292 MSE_CV: 0.0111942147063852
k = 3 MSE : 0.009056614543965735 MSE_CV: 0.01052787825888733
k = 4 MSE : 0.00916689918154762 MSE_CV: 0.010875823099163608
k = 5 MSE : 0.010320613662131519 MSE_CV: 0.01105006141345427
k = 6 MSE : 0.011428536981922399 MSE_CV: 0.011215615917332541
k = 7 MSE : 0.012174217050765884 MSE_CV: 0.011401569158059508
k = 8 MSE : 0.013307740088222789 MSE_CV: 0.011316457208193294
k = 9 MSE : 0.01342312645222698 MSE_CV: 0.011720719039542788
k = 10 MSE : 0.013304705215419498 MSE_CV: 0.011886099132113161
```