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)

Работа №3

по курсу «Технологии машинного обучения»

Выполнил

студент группы ИУ5-64Б

Шпак И.Д.

Москва, 2020

1 Исходное задание

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью подходящих для задачи метрик.
3. Произведите подбор гиперпараметра K с использованием GridSearchCV и/или RandomizedSearchCV и кросс-валидации, оцените качество оптимальной модели. Желательно использование нескольких стратегий кросс-валидации.
4. Сравните метрики качества исходной и оптимальной моделей.
5. С использованием метода train_test_split разделите выборку на обучающую и тестовую.

2 Код программы

[392]:
```
from IPython.display import Image
import numpy as np
import pandas as pd
from sklearn import svm, datasets
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris, load_boston
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.model_selection import KFold, RepeatedKFold, LeaveOneOut, LeavePOut,␣
↪ShuffleSplit, StratifiedKFold
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score,␣
↪classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error,␣
↪mean_squared_log_error, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.model_selection import learning_curve, validation_curve
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import OrdinalEncoder
%matplotlib inline
sns.set(style="ticks")
```

[393]:
```
data = pd.read_csv("/home/igor/Downloads/CarPrice_Assignment.xls",sep=',')
data.dtypes
```

[393]:
```
car_ID              int64
symboling           int64
CarName            object
fueltype           object
aspiration         object
doornumber         object
carbody            object
drivewheel         object
enginelocation     object
wheelbase         float64
carlength         float64
carwidth          float64
carheight         float64
curbweight          int64
enginetype         object
cylindernumber     object
enginesize          int64
fuelsystem         object
boreratio         float64
stroke            float64
compressionratio  float64
horsepower          int64
peakrpm             int64
citympg             int64
highwaympg          int64
price             float64
dtype: object
```

[394]:
```
cleanup_nums = {"doornumber":     {"four": 4, "two": 2},
                "cylindernumber": {"four": 4, "six": 6, "five": 5, "eight": 8,
                                   "two": 2, "twelve": 12, "three":3 }}
data = data.replace(cleanup_nums)
data["carbody"] = data["carbody"].astype('category')
data["carbody_cat"] = data["carbody"].cat.codes
data.head()
data["cylindernumber"].value_counts()
```

```
cleanup_nums = {"doornumber":     {"four": 4, "two": 2},
                "cylindernumber": {"four": 4, "six": 6, "five": 5, "eight": 8,
                                   "two": 2, "twelve": 12, "three":3 }}
data = data.replace(cleanup_nums)
data["carbody"] = data["carbody"].astype('category')
data["carbody_cat"] = data["carbody"].cat.codes
data=pd.get_dummies(data, columns=["drivewheel"], prefix=["drive"])
data["OHC_Code"] = np.where(data["enginetype"].str.contains("ohc"), 1, 0)
data.drop(data[(data['aspiration']=='turbo')].index,inplace=True)
data.drop(data[(data['fueltype']=='diesel')].index,inplace=True)
data.drop(["aspiration","carbody","CarName","enginelocation","enginetype",
"fueltype","fuelsystem","drive_fwd","symboling","car_ID","doornumber","carheight"],axis=1,inplace=T
```

[395]:
```
data.corr()
```

[395]:

	wheelbase	carlength	carwidth	curbweight	cylindernumber	\
wheelbase	1.000000	0.863551	0.771575	0.753595	0.407787	
carlength	0.863551	1.000000	0.822834	0.873632	0.487049	
carwidth	0.771575	0.822834	1.000000	0.844206	0.608466	
curbweight	0.753595	0.873632	0.844206	1.000000	0.691405	
cylindernumber	0.407787	0.487049	0.608466	0.691405	1.000000	
enginesize	0.591837	0.703488	0.752030	0.872782	0.882794	
boreratio	0.455094	0.594730	0.552074	0.625109	0.300113	
stroke	0.124127	0.099921	0.121868	0.087940	0.021244	
compressionratio	-0.449289	-0.394471	-0.302124	-0.321993	-0.036195	
horsepower	0.422167	0.606867	0.700469	0.813645	0.776246	
peakrpm	-0.264759	-0.218149	-0.124407	-0.171154	-0.169695	
citympg	-0.580059	-0.779157	-0.726442	-0.847613	-0.499531	
highwaympg	-0.633845	-0.794018	-0.741147	-0.868327	-0.527000	
price	0.542459	0.657901	0.732011	0.831856	0.755364	
carbody_cat	0.376906	0.304269	0.097441	0.088058	-0.030936	
drive_4wd	-0.059571	-0.047702	-0.087634	0.023798	-0.042529	
drive_rwd	0.415579	0.505995	0.465912	0.656716	0.372498	
OHC_Code	-0.188111	-0.099035	-0.109705	-0.100300	0.287682	

	enginesize	boreratio	stroke	compressionratio	\
wheelbase	0.591837	0.455094	0.124127	-0.449289	
carlength	0.703488	0.594730	0.099921	-0.394471	
carwidth	0.752030	0.552074	0.121868	-0.302124	

```
curbweight          0.872782   0.625109  0.087940         -0.321993
cylindernumber      0.882794   0.300113  0.021244         -0.036195
enginesize          1.000000   0.577834  0.155221         -0.223768
boreratio           0.577834   1.000000 -0.118141         -0.181321
stroke              0.155221  -0.118141  1.000000         -0.217118
compressionratio   -0.223768  -0.181321 -0.217118          1.000000
horsepower          0.868798   0.585599  0.076516         -0.041765
peakrpm            -0.219723  -0.210467  0.067930          0.293411
citympg            -0.701495  -0.613023 -0.050018          0.353192
highwaympg         -0.723239  -0.599998 -0.037712          0.366421
price               0.888816   0.558740  0.039450         -0.240339
carbody_cat        -0.081546  -0.035070  0.011235         -0.117340
drive_4wd          -0.079863   0.037708 -0.200851         -0.098277
drive_rwd           0.553151   0.558973 -0.072029         -0.043904
OHC_Code            0.175103  -0.044152  0.177090          0.062849

                  horsepower   peakrpm   citympg  highwaympg     price  \
wheelbase           0.422167 -0.264759 -0.580059   -0.633845  0.542459
carlength           0.606867 -0.218149 -0.779157   -0.794018  0.657901
carwidth            0.700469 -0.124407 -0.726442   -0.741147  0.732011
curbweight          0.813645 -0.171154 -0.847613   -0.868327  0.831856
cylindernumber      0.776246 -0.169695 -0.499531   -0.527000  0.755364
enginesize          0.868798 -0.219723 -0.701495   -0.723239  0.888816
boreratio           0.585599 -0.210467 -0.613023   -0.599998  0.558740
stroke              0.076516  0.067930 -0.050018   -0.037712  0.039450
compressionratio   -0.041765  0.293411  0.353192    0.366421 -0.240339
horsepower          1.000000  0.085114 -0.774101   -0.748830  0.869209
peakrpm             0.085114  1.000000 -0.015436    0.005742 -0.027285
citympg            -0.774101 -0.015436  1.000000    0.976508 -0.734506
highwaympg         -0.748830  0.005742  0.976508    1.000000 -0.738428
price               0.869209 -0.027285 -0.734506   -0.738428  1.000000
carbody_cat        -0.146299 -0.034486 -0.000546   -0.015786 -0.127144
drive_4wd          -0.109901 -0.147778 -0.036775   -0.075759 -0.068924
drive_rwd           0.647456  0.021460 -0.621438   -0.632116  0.637254
OHC_Code            0.030858 -0.124722  0.179747    0.177011 -0.009557

                  carbody_cat  drive_4wd  drive_rwd  OHC_Code
wheelbase            0.376906  -0.059571   0.415579 -0.188111
```

carlength	0.304269	-0.047702	0.505995	-0.099035
carwidth	0.097441	-0.087634	0.465912	-0.109705
curbweight	0.088058	0.023798	0.656716	-0.100300
cylindernumber	-0.030936	-0.042529	0.372498	0.287682
enginesize	-0.081546	-0.079863	0.553151	0.175103
boreratio	-0.035070	0.037708	0.558973	-0.044152
stroke	0.011235	-0.200851	-0.072029	0.177090
compressionratio	-0.117340	-0.098277	-0.043904	0.062849
horsepower	-0.146299	-0.109901	0.647456	0.030858
peakrpm	-0.034486	-0.147778	0.021460	-0.124722
citympg	-0.000546	-0.036775	-0.621438	0.179747
highwaympg	-0.015786	-0.075759	-0.632116	0.177011
price	-0.127144	-0.068924	0.637254	-0.009557
carbody_cat	1.000000	0.165922	-0.201838	-0.035140
drive_4wd	0.165922	1.000000	-0.145657	0.050632
drive_rwd	-0.201838	-0.145657	1.000000	-0.293796
OHC_Code	-0.035140	0.050632	-0.293796	1.000000

[396]:
```
feature_cols = [
    'wheelbase', 'carlength', 'carwidth', 'curbweight', 'cylindernumber',
    'enginesize',␣
 ↪'boreratio','compressionratio','horsepower','citympg','highwaympg',
    'carbody_cat','drive_4wd','drive_rwd'
]
data_X = data.loc[:,feature_cols]
data_Y = data.loc[:, 'price']
data_X_train, data_X_test, data_y_train, data_y_test = train_test_split(
    data_X, data_Y,test_size=0.3, random_state=360)
```

[397]:
```
cl1_1 = KNeighborsRegressor(n_neighbors=5)
cl1_1.fit(data_X_train, data_y_train)
target1_0 = cl1_1.predict(data_X_train)
target1_1 = cl1_1.predict(data_X_test)
r2_score(data_y_train, target1_0), r2_score(data_y_test, target1_1)
```

[397]:
```
(0.8478164344185322, 0.8059326051550549)
```

[398]:
```
scores = cross_val_score(KNeighborsRegressor(n_neighbors=5),
                         data_X, data_Y,
                         cv=4)
```

```
scores, np.mean(scores)
```

[398]: (array([0.52222292, 0.83105339, 0.49206234, 0.57117351]), 0.6041280385896937)

[399]:
```
grid = GridSearchCV(estimator = KNeighborsRegressor() ,param_grid={'n_neighbors':␣
↪range(1,50,1)},cv=RepeatedKFold(n_splits=3, n_repeats=3),scoring="r2")
grid.fit(data_X,data_Y)
grid.best_score_ , grid.best_params_,grid.best_estimator_
```

[399]: (0.7792957832531995, {'n_neighbors': 3}, KNeighborsRegressor(n_neighbors=3))

[400]:
```
grid.best_estimator_.fit(data_X_train, data_y_train)
target2_0 = grid.best_estimator_.predict(data_X_train)
target2_1 = grid.best_estimator_.predict(data_X_test)
r2_score(data_y_train, target2_0), r2_score(data_y_test, target2_1)
```

[400]: (0.9263546306766518, 0.8207513968468391)

[401]:
```
scores = cross_val_score(grid.best_estimator_, data_X, data_Y,␣
↪cv=RepeatedKFold(n_splits=3, n_repeats=3))
print("%0.2f r^2 with a standard deviation of %0.2f" % (scores.mean(), scores.
↪std()))
```

```
0.77 r^2 with a standard deviation of 0.07
```