PS17

April 23, 2023

1 Car test-time prediction

1.1 Loading MB dataset

[1]:
```
import pandas as pd
data = pd.read_csv('mercedes_test.csv')
```

1.2 Data pre-processing

[2]:
```
# Choose categorical data columns
cf = data.select_dtypes(include=['object']).columns
# To change it into "categorical" data type
data[cf]=data[cf].astype('category')
# One hot encoding
data = pd.get_dummies(data)
# Obtain X from data (excluding 'ID' and 'y')
X_df = data.drop(['ID','y'],axis=1)
# Obtain y from data
y_df = data['y']

# Convert y_df into binary labels
import numpy as np
TF_vector= (y_df<np.median(y_df))
y_df=TF_vector.astype(float)

# Conver data frame into numpy array
X,y = X_df.values, y_df.values

# Split into train and test datasets
from sklearn.model_selection import train_test_split
#X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,stratify=y)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,shuffle=False)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3788, 563)
(421, 563)
(3788,)
(421,)
```

1.3 LR: Hyperparameter search via cross validation

[4]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV

model_LR = LogisticRegression(solver='liblinear',max_iter=10000)
penalty_list = ['l1','l2']
C_list = [10,1,1e-1,1e-2,1e-3]
grid_LR = {'penalty':penalty_list,'C':C_list}
#grid_LR = dict(penalty=penalty_list,C=C_list)
cv_LR = RandomizedSearchCV(model_LR,grid_LR,n_iter=5,cv=5)
cv_LR.fit(X_train,y_train)
```

[4]:
```
RandomizedSearchCV(cv=5,
                   estimator=LogisticRegression(max_iter=10000,
                                                solver='liblinear'),
                   n_iter=5,
                   param_distributions={'C': [10, 1, 0.1, 0.01, 0.001],
                                        'penalty': ['l1', 'l2']})
```

[5]:
```
cv_LR.cv_results_ #logs results
```

[5]:
```
{'mean_fit_time': array([0.05327058, 0.02732315, 0.02352977, 0.02633443,
0.10073061]),
 'std_fit_time': array([0.00391827, 0.00205145, 0.00101464, 0.00081085,
0.01038432]),
 'mean_score_time': array([0.00099802, 0.00139604, 0.00119743, 0.00159283,
0.00119629]),
 'std_score_time': array([1.32316942e-06, 4.90440256e-04, 3.98945905e-04,
4.85819224e-04,
        3.98567801e-04]),
 'param_penalty': masked_array(data=['l2', 'l1', 'l1', 'l2', 'l2'],
              mask=[False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_C': masked_array(data=[1, 0.1, 0.01, 0.001, 10],
              mask=[False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'penalty': 'l2', 'C': 1},
  {'penalty': 'l1', 'C': 0.1},
  {'penalty': 'l1', 'C': 0.01},
```

```
  {'penalty': 'l2', 'C': 0.001},
  {'penalty': 'l2', 'C': 10}],
 'split0_test_score': array([0.89709763, 0.90369393, 0.86543536, 0.88654354,
0.88126649]),
 'split1_test_score': array([0.87862797, 0.88918206, 0.84432718, 0.8707124 ,
0.86147757]),
 'split2_test_score': array([0.85883905, 0.88522427, 0.83905013, 0.86015831,
0.83773087]),
 'split3_test_score': array([0.83883752, 0.84544254, 0.80317041, 0.8322325 ,
0.82034346]),
 'split4_test_score': array([0.85336856, 0.87978864, 0.84015852, 0.85072655,
0.84676354]),
 'mean_test_score': array([0.86535414, 0.88066629, 0.83842832, 0.86007466,
0.84951639]),
 'std_test_score': array([0.0203621 , 0.01931346, 0.02005331, 0.01831034,
0.02073005]),
 'rank_test_score': array([2, 1, 5, 3, 4])}
```

1.4 Store logs into csv file

[6]:
```
# Store logs into csv file
import pandas as pd
df_LR = pd.DataFrame.from_dict(cv_LR.cv_results_,orient='columns')
# Select columns to be stored
columns = ['params','mean_test_score','std_test_score','rank_test_score']
df_LR = df_LR[columns]
df_LR.to_csv("logs_LR.csv")
```

1.5 Save the best model

[7]:
```
best_model_LR=cv_LR.best_estimator_
from joblib import dump
dump(best_model_LR, 'best_model_LR.joblib')
```

[7]:
```
['best_model_LR.joblib']
```

1.6 Load “best_model_LS.joblib’

[8]:
```
from joblib import load
loaded_model_LR = load('best_model_LR.joblib')
loaded_model_LR.score(X_test, y_test)
```

[8]:
```
0.8931116389548693
```

[ ]:

[ ]: