SVC_GridSearch_Cryo_SVG

September 2, 2020

[82]:
```
#Importing libraries
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
import mglearn
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

[83]:
```
#importing cryography dataset
cryo_df = pd.read_csv("C:/Users/delld/Downloads/Cryotherapy.csv")
```

[84]:
```
#Details of Cryography dataset
cryo_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90 entries, 0 to 89
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   sex                  90 non-null     int64  
 1   age                  90 non-null     int64  
 2   Time                 90 non-null     float64
 3   Number_of_Warts      90 non-null     int64  
 4   Type                 90 non-null     int64  
 5   Area                 90 non-null     int64  
 6   Result_of_Treatment  90 non-null     int64  
dtypes: float64(1), int64(6)
memory usage: 5.0 KB
```

[85]:
```
X_features = cryo_df.columns
X_features
```

[85]:
```
Index(['sex', 'age', 'Time', 'Number_of_Warts', 'Type', 'Area',
       'Result_of_Treatment'],
      dtype='object')
```

[86]:
```
X = cryo_df[X_features]
Y = cryo_df['Result_of_Treatment']
X=X.drop(['Result_of_Treatment'],axis=1)
```

[87]:
```
#spliting into training and testing samples
X_train,X_test,y_train,y_test = train_test_split(X,Y,random_state=0)
```

[88]:
```
#SVM classifier - default parameters
svc = SVC()
svc
```

[88]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

[89]:
```
#Accuracy Score on the test set
svc.fit(X_train,y_train)
svc.score(X_test,y_test)
```

[89]:
```
0.6521739130434783
```

[90]:
```
#Data Preprocessing with Standard Scaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train_Scaled = scaler.transform(X_train)
X_test_Scaled = scaler.transform(X_test)
```

[91]:
```
#Accuracy on the test data set with Standard Scale Transformation
svc.fit(X_train_Scaled,y_train)
svc.score(X_test_Scaled,y_test)
```

[91]:
```
1.0
```

[92]:
```
#Data Preprocessing with MinMax Scaler
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_Scaled = scaler.transform(X_train)
X_test_Scaled = scaler.transform(X_test)
svc.fit(X_train_Scaled,y_train)
svc.score(X_test_Scaled,y_test)
```

[92]:
```
0.9130434782608695
```

[93]:
```
#SVC with C=1000
svc=SVC(C=1000)
svc
```

[93]:
```
SVC(C=1000, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

[94]:
```
#Accuracy on the test data set with MinMax Scale Transformation
svc.fit(X_train_Scaled,y_train)
svc.score(X_test_Scaled,y_test)
```

[94]:
```
0.782608695652174
```

[95]:
```
#SVC with C=1000,gamma = 10 on MinMax Scaled Dataset
svc=SVC(C=1000,gamma = 10)
svc
```

[95]:
```
SVC(C=1000, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=10, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

[96]:
```
#Accuracy on the MinMax scaled test set with C=1000,gamma = 10
svc.fit(X_train_Scaled,y_train)
svc.score(X_test_Scaled,y_test)
```

[96]:
```
0.9130434782608695
```

[97]:
```
#Simple GRID Search
print(X_train.shape[0],X_test.shape[0])
```

```
67 23
```

[98]:
```
best_score = 0.0
for gamma in [0.001,0.01,0.1,1,10,100]:
    for C in [0.001,0.01,0.1,1,10,100]:
        svm = SVC(gamma=gamma,C=C)
        svm.fit(X_train,y_train)
        score = svm.score(X_test,y_test)
        if score > best_score:
            best_score = score
            best_parameters = {'gamma':gamma, 'C': C }

print(best_score)
print(best_parameters)
```

```
0.9565217391304348
{'gamma': 0.001, 'C': 10}
```

[99]:
```
#Grid Search with Cross Validation
param_grid = {'gamma' : [0.001,0.01,0.1,1,10,100], 'C':[0.001,0.01,0.
↪1,1,10,100]}
param_grid
```

[99]:
```
{'gamma': [0.001, 0.01, 0.1, 1, 10, 100], 'C': [0.001, 0.01, 0.1, 1, 10, 100]}
```

[100]:
```
grid_search = GridSearchCV(SVC(),param_grid, cv =10)
```

[101]:
```
grid_search.fit(X_train,y_train)
grid_search.score(X_test,y_test)
```

[101]:
```
0.9565217391304348
```

[102]:
```
print(grid_search.best_params_)
print(grid_search.best_score_)
```

```
{'C': 10, 'gamma': 0.01}
0.9428571428571428
```

[103]:
```
results = pd.DataFrame(grid_search.cv_results_)
display(results.head())
```

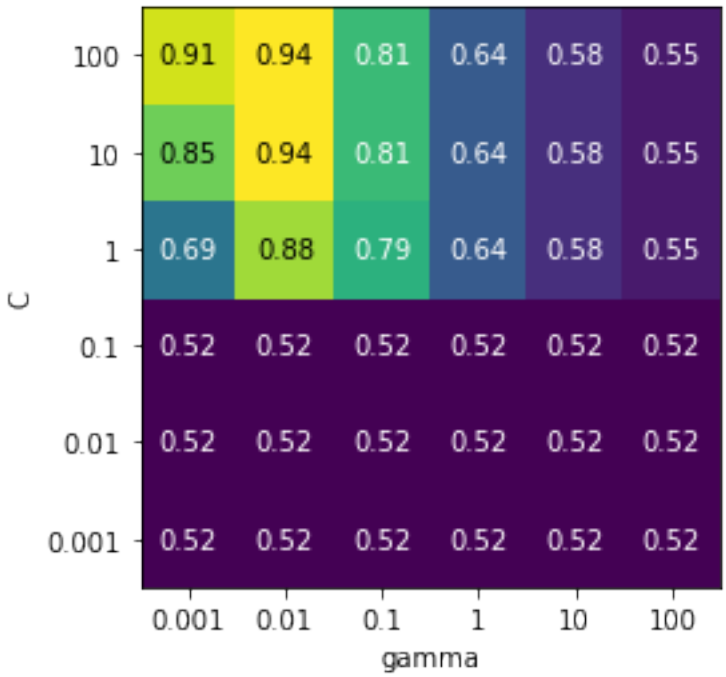
```
   mean_fit_time  std_fit_time  mean_score_time  std_score_time param_C  \
0       0.005601      0.000800         0.003001        0.001095   0.001
1       0.008201      0.012619         0.002000        0.000002   0.001
2       0.005201      0.001077         0.001900        0.000701   0.001
3       0.004702      0.000781         0.002000        0.000447   0.001
4       0.005902      0.001044         0.002200        0.000400   0.001

  param_gamma                         params  split0_test_score  \
0       0.001  {'C': 0.001, 'gamma': 0.001}           0.571429
1        0.01   {'C': 0.001, 'gamma': 0.01}           0.571429
2         0.1    {'C': 0.001, 'gamma': 0.1}           0.571429
3           1      {'C': 0.001, 'gamma': 1}           0.571429
4          10     {'C': 0.001, 'gamma': 10}           0.571429

   split1_test_score  split2_test_score  split3_test_score  split4_test_score  \
0           0.571429           0.571429           0.571429           0.571429
1           0.571429           0.571429           0.571429           0.571429
2           0.571429           0.571429           0.571429           0.571429
3           0.571429           0.571429           0.571429           0.571429
4           0.571429           0.571429           0.571429           0.571429

   split5_test_score  split6_test_score  split7_test_score  split8_test_score  \
```

0	0.428571	0.428571	0.5	0.5
1	0.428571	0.428571	0.5	0.5
2	0.428571	0.428571	0.5	0.5
3	0.428571	0.428571	0.5	0.5
4	0.428571	0.428571	0.5	0.5

	split9_test_score	mean_test_score	std_test_score	rank_test_score
0	0.5	0.521429	0.055787	19
1	0.5	0.521429	0.055787	19
2	0.5	0.521429	0.055787	19
3	0.5	0.521429	0.055787	19
4	0.5	0.521429	0.055787	19

[104]:
```
#Heat map of Scores using GridSearch with Cross Validation
scores = np.array(results.mean_test_score).reshape(6,6)
mglearn.tools.heatmap(scores,xlabel=␣
 ↪'gamma',xticklabels=param_grid['gamma'],ylabel='C',yticklabels=param_grid['C'],cmap='viridi
```

[104]: <matplotlib.collections.PolyCollection at 0x25424c8a208>



[105]:
```
#split data into train+validation set and test set
X_trainval,X_test,y_trainval,y_test = train_test_split(X,Y,random_state=0)
X_train,X_val,y_train,y_val =␣
 ↪train_test_split(X_trainval,y_trainval,random_state=0)
```

[106]:
```
best_score = 0
for gamma in [0.001,0.01,0.1,1,10,100]:
    for C in [0.001,0.01,0.1,1,10,100]:
        svc = SVC(gamma=gamma,C=C)
        scores = cross_val_score(svm,X_trainval,y_trainval,cv=5)
        score = np.mean(scores)
        if score > best_score:
            best_score = score
            best_parameters = {'gamma':gamma, 'C': C }
print(best_parameters)
svc = SVC(**best_parameters)
svc.fit(X_trainval,y_trainval)
svc.score(X_trainval,y_trainval)
svc
```

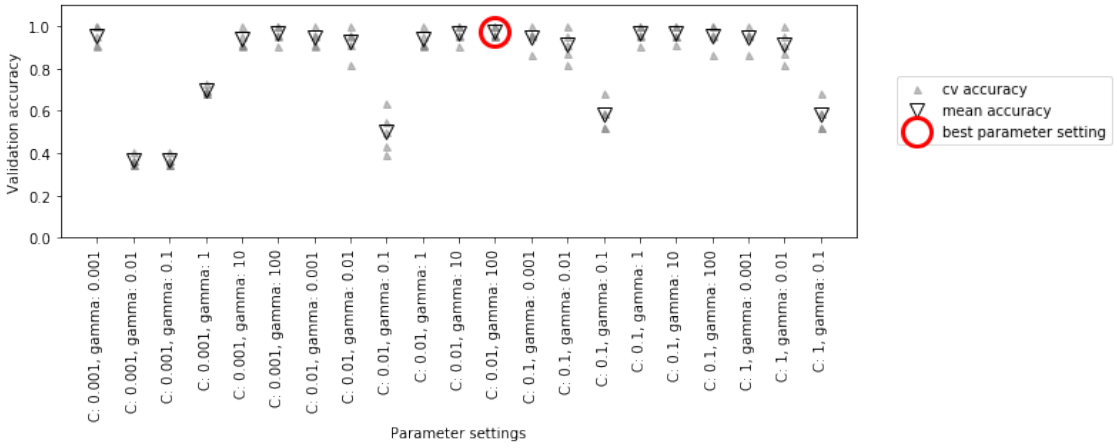
```
{'gamma': 0.001, 'C': 0.001}
```

[106]:
```
SVC(C=0.001, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

[107]:
```
#Accuracy score on the test data set
svc.score(X_test,y_test)
```

[107]:
```
0.5652173913043478
```

[108]:
```
#Ploting the Cross Validation accuracy, Mean accuracy and best parameter selection
mglearn.plots.plot_cross_val_selection()
```



[ ]:
```

```