model-selection-in-ml-with-python

April 29, 2024

```
[ ]: import numpy as np
     import pandas as pd
     from sklearn.model_selection import cross_val_score
     from sklearn.model_selection import GridSearchCV
```

Importing the dependencies

```
[ ]: # importing the models
     from sklearn.linear_model import LogisticRegression
     from sklearn.svm import SVC
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.ensemble import RandomForestClassifier
```

We will be working on the Heart Disease dataset

```
[ ]: # loading the csv data to a Pandas DataFrame
     heart_data = pd.read_csv('/content/heart.csv')
```

```
[ ]: # print first 5 rows of the dataset
     heart_data.head()
```

```
[ ]:    age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
     0   63    1   3       145   233    1        0      150      0      2.3      0
     1   37    1   2       130   250    0        1      187      0      3.5      0
     2   41    0   1       130   204    0        0      172      0      1.4      2
     3   56    1   1       120   236    0        1      178      0      0.8      2
     4   57    0   0       120   354    0        1      163      1      0.6      2

        ca  thal  target
     0   0     1       1
     1   0     2       1
     2   0     2       1
     3   0     2       1
     4   0     2       1
```

```
[ ]: # number of rows and columns in the dataset
     heart_data.shape
```

[ ]:

```
(303, 14)
```

[ ]:

```python
# checking for missing values
heart_data.isnull().sum()
```

[ ]:

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

[ ]:

```python
# checking the distribution of Target Variable
heart_data['target'].value_counts()
```

[ ]:

```
1    165
0    138
Name: target, dtype: int64
```

1 –> Defective Heart

0 –> Healthy Heart

Splitting the Features and Target

[ ]:

```python
X = heart_data.drop(columns='target', axis=1)
Y = heart_data['target']
```

[ ]:

```python
print(X)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0     63    1   3       145   233    1        0      150      0      2.3
1     37    1   2       130   250    0        1      187      0      3.5
2     41    0   1       130   204    0        0      172      0      1.4
3     56    1   1       120   236    0        1      178      0      0.8
4     57    0   0       120   354    0        1      163      1      0.6
..   …  …  ..       …   …  …      …      …    …      …
298   57    0   0       140   241    0        1      123      1      0.2
299   45    1   3       110   264    0        1      132      0      1.2
300   68    1   0       144   193    1        1      141      0      3.4
```

```
301   57    1   0       130   131    0         1       115       1       1.2
302   57    0   1       130   236    0         0       174       0       0.0

     slope  ca  thal
0        0   0     1
1        0   0     2
2        2   0     2
3        2   0     2
4        2   0     2
..     …   ..   …
298      1   0     3
299      1   0     3
300      1   2     3
301      1   1     3
302      1   1     2

[303 rows x 13 columns]
```

[ ]:

```python
print(Y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

[ ]:

```python
X = np.asarray(X)
 Y = np.asarray(Y)
```

Model Selection

1. Comparing the models with default hyperparameter values using Cross Validation

[ ]:

```python
# list of models
models = [LogisticRegression(max_iter=1000), SVC(kernel='linear'),␣
 ↪KNeighborsClassifier(), RandomForestClassifier(random_state=0)]
```

[ ]:

```python
def compare_models_cross_validation():

  for model in models:

    cv_score = cross_val_score(model, X, Y, cv=5)
    mean_accuracy = sum(cv_score)/len(cv_score)
    mean_accuracy = mean_accuracy*100
    mean_accuracy = round(mean_accuracy, 2)

    print('Cross Validation accuracies for the',model,'=', cv_score)
    print('Acccuracy score of the ',model,'=',mean_accuracy,'%')
    print('---------------------------------------------------------------')
```

[ ]:

```
compare_models_cross_validation()
```

```
Cross Validation accuracies for the LogisticRegression(max_iter=1000) =
[0.80327869 0.8852459  0.85245902 0.86666667 0.75      ]
Acccuracy score of the  LogisticRegression(max_iter=1000) = 83.15 %
---------------------------------------------------------------
Cross Validation accuracies for the SVC(kernel='linear') = [0.81967213 0.8852459
0.80327869 0.86666667 0.76666667]
Acccuracy score of the  SVC(kernel='linear') = 82.83 %
---------------------------------------------------------------
Cross Validation accuracies for the KNeighborsClassifier() = [0.60655738
0.6557377  0.57377049 0.73333333 0.65      ]
Acccuracy score of the  KNeighborsClassifier() = 64.39 %
---------------------------------------------------------------
Cross Validation accuracies for the RandomForestClassifier(random_state=0) =
[0.85245902 0.90163934 0.81967213 0.81666667 0.8       ]
Acccuracy score of the  RandomForestClassifier(random_state=0) = 83.81 %
---------------------------------------------------------------
```

Inference: For the Heart Disease dataset, **Random Forest Classifier** has the Highest accuracy value with default hyperparameter values

2. Comparing the models with different Hyperparameter values using GridSearchCV

[ ]:

```
# list of models
models_list = [LogisticRegression(max_iter=10000), SVC(),␣
 ↪KNeighborsClassifier(), RandomForestClassifier(random_state=0)]
```

[ ]:

```
# creating a dictionary that contains hyperparameter values for the above␣
 ↪mentioned models


model_hyperparameters = {


    'log_reg_hyperparameters': {

        'C' : [1,5,10,20]
    },
    'svc_hyperparameters': {

        'kernel' : ['linear','poly','rbf','sigmoid'],
        'C' : [1,5,10,20]
    },


    'KNN_hyperparameters' : {

        'n_neighbors' : [3,5,10]
    },


    'random_forest_hyperparameters' : {

        'n_estimators' : [10, 20, 50, 100]
    }
}
```

```
[ ]: type(model_hyperparameters)
```

```
[ ]: dict
```

```
[ ]: print(model_hyperparameters.keys())
```

```
dict_keys(['log_reg_hyperparameters', 'svc_hyperparameters',
'KNN_hyperparameters', 'random_forest_hyperparameters'])
```

```
[ ]: model_hyperparameters['log_reg_hyperparameters']
```

```
[ ]: {'C': [1, 5, 10, 20]}
```

```
[ ]: model_keys = list(model_hyperparameters.keys())
     print(model_keys)
```

```
['log_reg_hyperparameters', 'svc_hyperparameters', 'KNN_hyperparameters',
'random_forest_hyperparameters']
```

```
[ ]: model_keys[0]
```

```
[ ]: 'log_reg_hyperparameters'
```

```
[ ]: model_hyperparameters[model_keys[0]]
```

```
[ ]: {'C': [1, 5, 10, 20]}
```

Applying GridSearchCV

[ ]:

```python
def ModelSelection(list_of_models, hyperparameters_dictionary):

  result = []

  i = 0

  for model in list_of_models:

    key = model_keys[i]

    params = hyperparameters_dictionary[key]

    i += 1

    print(model)
    print(params)
    print('---------------------------------')


    classifier = GridSearchCV(model, params, cv=5)

    # fitting the data to classifier
    classifier.fit(X,Y)

    result.append({
        'model used' : model,
        'highest score' : classifier.best_score_,
        'best hyperparameters' : classifier.best_params_
    })

  result_dataframe = pd.DataFrame(result, columns = ['model used','highest score','best hyperparameters'])

  return result_dataframe
```

[ ]:

```python
ModelSelection(models_list, model_hyperparameters)
```

```
LogisticRegression(max_iter=10000)
{'C': [1, 5, 10, 20]}
---------------------------------
SVC()
{'kernel': ['linear', 'poly', 'rbf', 'sigmoid'], 'C': [1, 5, 10, 20]}
---------------------------------
KNeighborsClassifier()
{'n_neighbors': [3, 5, 10]}
---------------------------------
RandomForestClassifier(random_state=0)
```

```
{'n_estimators': [10, 20, 50, 100]}
---------------------------------
```

[ ]:

```
                               model used  highest score  \
0      LogisticRegression(max_iter=10000)       0.831585   
1                                   SVC()       0.828306   
2                  KNeighborsClassifier()       0.643880   
3  RandomForestClassifier(random_state=0)       0.838087   

           best hyperparameters  
0                      {'C': 5}  
1  {'C': 1, 'kernel': 'linear'}  
2            {'n_neighbors': 5}  
3         {'n_estimators': 100}  
```

Random Forest Classifier with n_estimators = 100 has the highest accuracy

[ ]: