DK911b-Machine_Learning-Lab2

October 22, 2019

1 Scikit-Lab 2

- scikit-learn is the leading machine learning software in Python
- scikit-learn is a project started in Paris, Inria and Telecom Paris
- scilkit-learn is easy to use and extend

2 Task 1:

2.0.1 - Implement a majority class classifier: a classifier that predicts the class label that is most frequent in the dataset.

- Classifiers in scikit-learn has two main methods:
  - Build a model: fit(self, X, Y)
  - Make a prediction: predict(self, X)
- Template for implementing classifier is given:

[6]:
```
class NewClassifier:
    def __init__(self):
        pass

    def fit(self, X, Y):
        self.label = 0
        dict = {}
        for y in Y:
            if y in dict:
                dict[y] = dict[y] + 1
            else:
                dict[y] = 1
        for i in dict.keys():
            if dict[i] > dict[self.label] :
                self.label = i
        return self

    def predict(self, X):
        Y = []
        for x in X:
            Y.append(self.label)
        return Y
```

3 Task 2:

3.0.1 - Implement k-fold cross validation

[11]:
```
from sklearn.model_selection import KFold

def cross_validation(clf, dataset, n_folds):
    X = dataset.data
    Y = dataset.target
    KF = KFold(n_splits=n_folds,shuffle=True)

    sum = 0.0

    for train_index,test_index in KF.split(X):
        X_train,X_test=X[train_index],X[test_index]
        Y_train,Y_test=Y[train_index],Y[test_index]
        clf.fit(X_train, Y_train)
        Y_pre = clf.predict(X_test)
        accuracy = 0.0
        count = 0
        for i in range(len(Y_test)):
            if Y_pre[i] == Y_test[i]:
                count = count + 1
        accuracy = count/len(Y_test)
        sum += accuracy

    score = sum/n_folds
    return score
```

4 Task 3:

4.0.1 Use the majority class classifier to evaluate one dataset, and explain the evaluation results:

- https://scikit-learn.org/stable/datasets/index.html

[12]:
```
import numpy as np
from sklearn import datasets

iris = datasets.load_iris()
iris_X = iris.data
iris_y = iris.target
np.random.seed(0)
indices = np.random.permutation(len(iris_X))
```

```
iris_X_train = iris_X[indices[:-10]]
iris_y_train = iris_y[indices[:-10]]
iris_X_test = iris_X[indices[-10:]]
iris_y_test = iris_y[indices[-10:]]


clf = NewClassifier()
clf.fit(iris_X_train, iris_y_train)
result = clf.predict(iris_X_test)
print(result)

cross_validation(clf,iris,5)
```

```
[2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
```

[12]:
```
0.2733333333333333
```

5 Task 4: *OPTIONAL*

5.0.1 - Implement another classifier with higher performance than the majority class classifier, evaluate it and comment the results

5.0.2 - Create my own KNN classifier:

[13]:
```
from scipy.spatial import distance

def euc(a,b):
    return distance.euclidean(a,b)

class KNNClassifier:
    def __init__(self, k):
        self.k = k

    def fit(self, X_train, Y_train):
        self.X_train = X_train
        self.Y_train = Y_train

    def predict(self, X_test):
        Y_pre = np.zeros(len(X_test))
        for i in range(len(X_test)):
            distances = np.zeros((len(self.X_train),2))
            for j in range(len(self.X_train)):
                dist = euc(X_test[i],self.X_train[j])
                distances[j] = [dist,self.Y_train[j]]
            sortedKDistances = distances[distances[:,0].argsort()][:self.k]
            labels = np.bincount(sortedKDistances[:,1].astype('int'))
            Y_pre[i] = labels.argmax()
```

```
        return Y_pre
```

5.0.3 - Test my KNN

[14]:
```
knn = KNNClassifier(k = 5)
knn.fit(iris_X_train, iris_y_train)
predictions = knn.predict(iris_X_test)
print(predictions)
print(iris_y_test)
```

```
[1. 2. 1. 0. 0. 0. 2. 1. 2. 0.]
[1 1 1 0 0 0 2 1 2 0]
```

5.0.4 - Use k-fold cross validation to evaluate my KNN

[15]:
```
cross_validation(knn, iris, 4)
```

[15]:
```
0.9530583214793741
```

6 Submit your Jupyter notebook and pdf version of it to filippo.miatto@telecom-paristech.fr until 23rd of October, 2019.