Trang Tran

DSC 540

Assignment 8 – Ensemble Methods

Github link: https://github.com/tntran7/TTAssignment8

Reproduces data acquisition and cleaning

The dataset PAMAP2 Physical Activity Monitoring is obtained from UCI machine leaning repository https://archive.ics.uci.edu/ml/datasets/PAMAP2+Physical+Activity+Monitoring

The dataset contains data of 18 different physical activities, performed by 9 subjects wearing 3 inertial measurement units and a heart rate monitor. Thus, the first step is to merge 9 separating files into one dataset, create the dataset and add the column names.

In [6]:
```
#combine 9 files
path = 'C:/Users/Trang/Downloads/PAMAP2_Dataset/PAMAP2_Dataset/Protocol'
file = os.listdir(path)
for i in sorted(file):
    filename = 'PAMAP.txt'
    with open(filename, 'a') as outfile:
        with open(path + '/' + i, 'r') as infile:
            outfile.write(infile.read())
```

In [7]:
```
data = pd.read_csv('PAMAP.txt', ' ', header= None)
```

In [8]:
```
#adding column names for 54 columns
colNames = ["timestamp", "activityID", "heartrate"]

IMUhand = ['handTemperature',
           'handAcc16_1', 'handAcc16_2', 'handAcc16_3',
           'handAcc6_1', 'handAcc6_2', 'handAcc6_3',
           'handGyro1', 'handGyro2', 'handGyro3',
           'handMagne1', 'handMagne2', 'handMagne3',
           'handOrientation1', 'handOrientation2', 'handOrientation3', 'handOrientation4']

IMUchest = ['chestTemperature',
           'chestAcc16_1', 'chestAcc16_2', 'chestAcc16_3',
           'chestAcc6_1', 'chestAcc6_2', 'chestAcc6_3',
           'chestGyro1', 'chestGyro2', 'chestGyro3',
           'chestMagne1', 'chestMagne2', 'chestMagne3',
           'chestOrientation1', 'chestOrientation2', 'chestOrientation3', 'chestOrientation4']

IMUankle = ['ankleTemperature',
           'ankleAcc16_1', 'ankleAcc16_2', 'ankleAcc16_3',
           'ankleAcc6_1', 'ankleAcc6_2', 'ankleAcc6_3',
           'ankleGyro1', 'ankleGyro2', 'ankleGyro3',
           'ankleMagne1', 'ankleMagne2', 'ankleMagne3',
           'ankleOrientation1', 'ankleOrientation2', 'ankleOrientation3', 'ankleOrientation4']

columns = colNames + IMUhand + IMUchest + IMUankle

data.columns = columns
```

The next step is data cleaning. Since the zero value in ‘activityID’ is transient, I remove it out of the dataset. Then, I use errors = ‘coerce’ to replace all non-numeric values with NaN, and interpolate() to remove all NaN values.

In [9]:
```
#Data cleaning
def cleandata(data):
    #remove zero value in activityID column
    data = data.drop(data[data['activityID'] == 0].index)
    data = data.apply(pd.to_numeric, errors = 'coerce') #invalid parsing will be set as NaN
    data = data.interpolate() #remove NaN value
    return data
```

In [10]:
```
data1 = cleandata(data)
data1.reset_index(drop = True, inplace = True)

display(data1.head(10))
```

Reproduces classifiers implementation

This is a big dataset with 2.4M and 1.9M after cleaning, so I take the subset n = 100000 observations was selected randomly to work on. The research “Ensemble Methods for Classification of Physical Activities” analyzes the wrist-worn accelerometer data, so X variables are ‘IMUhand’ columns and ‘'timestamp', 'heartrate', 'handTemperature'. Y variable is 'activityID'.

In [41]:
```
data2 = data1.sample(n = 100000)
```

In [44]:
```
cols = ['timestamp', 'activityID', 'heartrate', 'handTemperature',
       'handTemperature', 'handAcc16_1', 'handAcc16_2', 'handAcc16_3',
           'handAcc6_1', 'handAcc6_2', 'handAcc6_3',
           'handGyro1', 'handGyro2', 'handGyro3',
           'handMagne1', 'handMagne2', 'handMagne3',
           'handOrientation1', 'handOrientation2', 'handOrientation3', 'handOrientation4']
data3 = data2[cols]
data3.head()
```

Out[44]:

	timestamp	activityID	heartrate	handTemperature	handTemperature	handAcc16_1	handAcc16_2	handAcc16_3	hand
1176584	3504.05	5	165.0	33.8125	33.8125	-20.84800	15.764800	7.756980	-1
775023	1001.70	17	106.0	33.5625	33.5625	-6.99086	6.422490	6.264130	-6
700206	200.87	1	72.0	32.7500	32.7500	5.16003	0.312252	8.275660	5
1732122	707.75	3	79.0	34.5000	34.5000	-9.37532	-1.031890	2.212380	-9
627697	1453.80	16	98.0	33.1875	33.1875	-10.55050	2.895420	-0.897459	-10

Then, I normalize the data, and create training & testing dataset with 80/20 splitting.

```
In [45]: scaler = StandardScaler()
         X = data3.drop('activityID', axis=1)
         X = scaler.fit_transform(X)
         y = data3['activityID']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 100)
```

The classifier models are K-Nearest Neighbors, Binary Decision Tree, Support Vector Machine, and Artificial Neural Network. To be consistency, I replicate all the conditions the authors implemented to each classifier in their study.

- K-Nearest Neighbors: In the research, the k optimal is 7.
- Binary Decision Tree: In the research, the author used max_depth = 20
- Support Vector Machine algorithm is to maximized the margin width. In other words, it maximizes the distance of two datapoints by linear boundary. In the research, the author implemented ‘one-vs-rest’ type in SVM.
- Artificial Neural Network: In the research, the author implemented a hidden layer size of 50 and 250 epochs.

```
knn1 = KNeighborsClassifier(n_neighbors = 7)
cart1 = DecisionTreeClassifier(random_state = 100, max_depth = 20)
svm1 = SVC(decision_function_shape = 'ovr', probability = True, kernel = 'linear')
ann1 = MLPClassifier(hidden_layer_sizes = (50,), max_iter = 250, random_state = 100)
```

Reproduces the ensemble framework

VotingClassifier() from sklearn is used to train on an ensemble of four models and predict an output based on their highest probability of chosen class as the output. The type of voting is selected to soft voting. It means the output class is the prediction based on the average of probability given to that class.

The ensemble learning method is weighted majority voting that evaluates individual performance of each classifier in the ensemble and adjusts their contributions to class decision.

```
In [16]: wmv1 = VotingClassifier(estimators = [('BDT', cart1),('knn', knn1),('svm', svm1),
                                               ('ann', ann1)], voting='soft', n_jobs=-1)

         knn1.fit(X_train,y_train)
         cart1.fit(X_train,y_train)
         svm1.fit(X_train,y_train)
         wmv1.fit(X_train,y_train)
```

```
Out[16]: VotingClassifier(estimators=[('BDT',
                                       DecisionTreeClassifier(ccp_alpha=0.0,
                                                              class_weight=None,
                                                              criterion='gini',
                                                              max_depth=20,
                                                              max_features=None,
                                                              max_leaf_nodes=None,
                                                              min_impurity_decrease=0.0,
                                                              min_impurity_split=None,
                                                              min_samples_leaf=1,
                                                              min_samples_split=2,
                                                              min_weight_fraction_leaf=0.0,
                                                              presort='deprecated',
                                                              random_state=100,
                                                              splitter='best')),
                                      ('knn',
                                       KNeighbors...
                                                    epsilon=1e-08,
                                                    hidden_layer_sizes=(50,),
                                                    learning_rate='constant',
                                                    learning_rate_init=0.001,
                                                    max_fun=15000, max_iter=250,
                                                    momentum=0.9, n_iter_no_change=10,
                                                    nesterovs_momentum=True,
                                                    power_t=0.5, random_state=100,
                                                    shuffle=True, solver='adam',
                                                    tol=0.0001, validation_fraction=0.1,
                                                    verbose=False, warm_start=False))],
                         flatten_transform=True, n_jobs=-1, voting='soft',
                         weights=None)
```

To visualize the result, I build a confusion matrix to evaluate the accuracy of a classification. The confusion matrix is to evaluate the quality of output of a classifier on the dataset. The correct predictions are on the diagonal. The diagonal elements represent the value for which the predicted label is equal to the true label. The elements that are not on the diagonal are mislabeled by the classifier. The higher the diagonal values are, the better correct predictions are. The ultimate goal is the values off the diagonal are zero, but it seems hardly happens in real.

However, the confusion matrix below has the values off the diagonal are small and most of the values off the diagonal are zero. This explains why the accuracy score is high.

```
In [48]: y_pred = wmv1.predict(X_test)
         confusion_matrix(y_test, y_pred)

Out[48]: array([[2063,    1,    0,    0,    0,    0,    0,    0,    0,    0,
                    0],
                [   0, 1906,    1,    0,    0,    0,    0,    0,    0,    0,
                    0],
                [   0,    2, 2015,    0,    0,    0,    0,    0,    0,    2,
                    0],
                [   0,    0,    0, 2429,    0,    1,    6,    0,    0,    0,
                    0],
                [   0,    0,    0,    0,  952,    2,    1,    0,    0,    0,
                    0],
                [   0,    0,    0,    1,    3, 1634,    5,    0,    0,    0,
                    0],
                [   0,    0,    0,    2,    0,    1, 1895,    0,    1,    0,
                    0],
                [   0,    0,    0,    5,    0,    0,    1, 1134,   28,    2,
                    0],
                [   0,    0,    0,    4,    0,    0,    1,   26, 1050,    0,
                    0],
                [   0,    0,    0,    0,    0,    0,    0,    1,    0, 1869,
                    6],
                [   0,    0,    3,    0,    0,    0,    0,    0,    0,    0,
                 2440],
                [   0,    0,    0,    0,    1,    1,    0,    0,    0,    0,
                    0],
                  505]], dtype=int64)
```

Analyzes the results

```
In [49]: wmv1.score(X_test, y_test)

Out[49]: 0.9946
```

```
In [50]: print('Ann: ', {wmv1.named_estimators_['ann'].score(X_test, y_test)})
         print('BDT: ', {wmv1.named_estimators_['BDT'].score(X_test, y_test)})
         print('SVM: ', {wmv1.named_estimators_['svm'].score(X_test, y_test)})
         print('knn: ', {wmv1.named_estimators_['knn'].score(X_test, y_test)})

         Ann:  {0.0009}
         BDT:  {0.00105}
         SVM:  {0.00945}
         knn:  {0.00235}
```

The accuracy score is 99.46% in the testing dataset. The number is very close to 100% fitted to the model.

A weak classifier is a model for binary classification that performs slightly better than random guessing. Each model creates a weak classifier with sub-20% accuracy. The accuracy of each model is:

```
Ann: {0.0009}
BDT: {0.00105}
SVM: {0.00945}
knn: {0.00235}
```

The numbers reflect to what weak classifier means. It also shows ANN (Artificial Neur al Network) with the lowest value that would have the slowest computational time.

Compares own results with the ones in the article

Ensemble method is to use multiple models to obtain better predictive performance. Ensembles combine multiple hypotheses to form a better hypothesis. In other words, it combines multiple weak learners to produce a strong learner. Ensemble learners tend to overfit, since there is deliberately introduced significant diversity among models.

The accuracy score is 99.46% in the subset of PAMAP dataset using Weighted Majority Voting, which is a better result than the score of 83% combined average in the research (Chowdhary, et al., 2017). The score can change as the subset was selected randomly, and different methods of cleaning data are applied. In the research, the authors also implement Naïve Bayes and Behavior Knowledge Space method to yield a better prediction.

References:

Gopal, M. (2019). *Applied machine learning.* McGraw-Hill Education. ISBN-139781260456844

Reiss, A. (August, 2012). PAMAP2 Physical Activity Monitoring Data Set. https://archive.ics.uci.edu/ml/datasets/PAMAP2+Physical+Activity+Monitoring

Singh, A. (2018). A Comprehensive Guide to Ensemble Learning (with Python codes). https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

Ensemble Methods. https://scikit-learn.org/stable/modules/ensemble.html

Chowdhury, A. K., Tjondronegoro, D., Chandran, V., Trost, S. G.2. (September, 2017) Ensemble Methods for Classification of Physical Activities from Wrist Accelerometry. Medicine & Science in Sports & Exercise: Volume 49 - Issue 9 - p 1965-1973.

DOI: 10.1249/MSS.0000000000001291

ML | Voting Classifier using Sklearn. (November, 2019). https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/