Machine Learning
Assignement 8 ADABOOST CLASSIFIER

Sami Sellami

October 24, 2018

THE SOURCE CODE IS ATTACHED TO THE PRESENT FILE

1. Training AdaBoost classifier: We first download the dataset:

   ```
   dataset = pd.read_csv('countries.csv')
   print(dataset)
   ```

   We see that the data have have a column 'region' that can be Europe, America, Africa, Asia or Oceania, we need to transform these categorical datasets into numerical ones:

   ```
   dataset = pd.get_dummies(dataset, columns=['Region'])
   ```

   Since we are interested only in the region Europe we drop the the dummies created except region Europe

   ```
   dataset=dataset.drop(columns=['Region_ASIA', 'Region_AMERICA', 'Region_OCEANIA'
   , 'Region_AFRICA'])
   ```

   We drop also tha data with non values:

   ```
   dataset=dataset.dropna()
   dataset.reset_index(drop=True, inplace=True)
   ```

   Now we split our dataset to features and label; the features X should contain all the columns except the region Europe(label) and the country(this is what we want to predict) and our label should be the region Europe because we want to predict if a country is in Europe or not

   ```
   X=dataset.drop(columns=['Country','Region_EUROPE'])
   y=dataset['Region_EUROPE']
   ```

   Now we can train our model using AdaBoost and logistic regression a base estimator:

   ```
   n_estimators=10
   algo= LogisticRegression()
   model= AdaBoostClassifier(base_estimator= algo, n_estimators=n_estimators,
   algorithm='SAMME')
   model.fit(X,y)
   ```

   We find the following weights for the models

   $$\omega = \begin{bmatrix} 0.41446524 & 0.66869246 & 1.15702528 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \end{bmatrix}$$

   The weights after the fourth iteration are equal to zero means that the classifier has created models close to random classifiers

   $$weights = \log((1 - error)/error) = 0$$
   $$\implies error = \frac{1}{2}$$

   So we are in presence of random classifiers that have an error of 0.5 this means that they predict half of the features correctly $error = (correct - N)/N$, they can give as many true predictions as false ones, we don't want these predictors in our model because they are untrustworthy this is why their weights equal to zero

   if we compute the accuracy of this adaboost model we find

```
from sklearn.metrics import accuracy_score
accuracy= accuracy_score(y, y_predict)
print(accuracy)
0.8268156424581006
```

the same accuracy but using cross validation this time

```
from sklearn.model_selection import cross_val_score
print(cross_val_score(model, X, y, cv=3).mean())
0.8655367231638418
```

2. Finding outliers in dataset: for that we need to calculate the coefficients alphas, we use the following code:

```
def get_alphas(weights, n_estimators, X, y, y_predict):
    alphas=[1/len(X) for i in range(len(X))]
    for i in range(n_estimators):
        for j in range(len(alphas)):
            if y_predict[j]==y[j]:
                alphas[j]=alphas[j]*np.exp(-weights[i])
            else:
                alphas[j]=alphas[j]*np.exp(weights[i])
    return alphas
```

This algorithm will give us the alphas at the end of the classification, by examining the data we clearly see that the alphas are approximatively equals to 2×10^{-4} except for some values that are much higher, these are the outliers

We set a threshold equal to 0.001 and we extract the indices of points that are no outliers

```
index=[ i for i in range(len(alphas))  if alphas[i]<0.001]
```

We shrink our features and labels

```
y=y[index]
X=X.iloc[index,:]
dataset=dataset.iloc[index]
```

We fit one more time and we extract the weight of new model created with the dataset without outliers:

$$\omega = \begin{bmatrix} 0.70329 & 1.49234 & 1.41079 & 0.49405 & 3.28405 & 7.12583 & 1.52601 & 1.17805 & 0.86807 & 0.88802 \end{bmatrix}$$

This time all the weight are non-zero, this means that all the classifiers created can be trusted in the final prediction and we obtain an accuracy of 1.0 this is logic considering that we use only train data

Our model didn't do mistakes in the predictions and this is exactly what we want because the training data represent all the existing countries

And we find a cross validation error of

$$accuracyCV = 0.8914285714285715$$

We clearly see an improvement in the accuracy after removing the outliers

Now we want to predict if a country is in Europe using our model; for that we implement the following code which takes as arguments the country that we want to predict, the dataset and the predictions that the model has generated

```
def predict(Country, dataset, y_predict):
    for i in range(len(dataset)):
        if dataset['Country'].iloc[i]==Country:
            return y[i]
    if i==(len(dataset)-1):
        print('no such country in dataset(it could be an outlier!!)')

Country='Spain '
predict(Country, dataset, y_predict)
1

Country='Algeria '
predict(Country, dataset, y_predict)
0
```