Muhammad Aamir Iqbal

October 9, 2018

I used following classifiers (provided by the Python scikit-learn library) in this project;

1. **Perceptron**
   I used following parameters while testing both datasets:
   n_iter=40, eta0=0.1, and random_state=1
   I also scale the feature's data by using StandardScaler() function. I ran experiments on both datasets with and without scaling the data. In digits dataset, there was a minor difference in accuracy and running time due to scaling the data. In activity recognition dataset, there was a major improvement in accuracy and running time due to scaling the data.
2. **Support vector machine (linear and non-linear using Radial Basis Function (RBF) kernel)**
   I divided this classifier in two parts; linear and non-linear. I used different parameters in both cases while testing both datasets. Parameters for both cases are given below:
   Linear (digits): kernel="linear", random_state=1, C=1.0
   Non-Linear (digits): gamma='scale', C = 1.0 (Note: kernel: default='rbf')
   Linear (activity recognition): kernel="linear", random_state=1, C=10.0
   Non-Linear (activity recognition): gamma='scale', C = 10.0 (Note: kernel: default='rbf')
   I ran experiments on both datasets with and without scaling the data. In digits dataset, there was a minor drop in accuracy and major improvement in running time due to scaling the data in non-linear case. I ran a test for three hours on activity recognition dataset without scaling for both cases i.e. linear and non-linear but didn't get any results for accuracy and running time. After scaling the date, I ran same test and achieved higher accuracy.
3. **Decision Tree**
   I used following parameters while testing both datasets:
   criterion="entropy", random_state=1, max_depth=10, and min_samples_leaf=5
   I tested my datasets with criterion="gini" but accuracy was very low. I also found out that we can increase the accuracy by increasing the max_depth parameter value. I ran experiments on both datasets with and without scaling the data. In digits dataset, there was a minor difference in accuracy and running time due to scaling the data. In activity recognition dataset, the accuracy didn't change but there was a major improvement in running time due to scaling the data. As we can see from results that features scaling doesn't has significant effect on accuracy and running time.
4. **K-nearest Neighbor**
   I used following parameters while testing both datasets:
   n_neighbors(digit)=23, n_neighbors(activity recognition)=231, and metric='euclidean'
   I used this formula n_neighbors = math.sqrt(len(X_test)) to get number of neighbors. The purpose of using 'euclidean' is to measure the distance between neighbors. I ran experiments on both datasets with and without scaling the data. In digits dataset, there was a minor drop in accuracy but running time was unaffected due to scaling the data. In activity recognition dataset, there was major improvement in accuracy, but the running time drastically increase due to scaling the data.
5. **Logistic Regression**
   I used following parameters while testing both datasets:
   multi_class='auto'

Muhammad Aamir Iqbal

October 9, 2018

I ran experiments on both datasets with and without scaling the data. In digits dataset, there was a minor difference in accuracy and running time due to scaling the data. In activity recognition dataset, there was major improvement in accuracy, but the running time drastically increase due to scaling the data.

Note: Detailed accuracy and running time data for both datasets is given below in separate tables.

I used following datasets to test these classifiers:

1. Digits dataset (offered by scikit-learn library)
2. Activity Recognition dataset

Digits Dataset Results:

Classifier	Accuracy	Running Time (s)	Accuracy with Scaled Data	Running Time of Scaled Data (s)
Perceptron	92.78%	0.15	93.70%	0.16
Decision Tree	86.67%	0.11	86.11%	0.12
K-nearest neighbor	97.04%	0.18	95.00%	0.18
Logistic regression	96.30%	0.24	95.93%	0.31
SVM Linear	98.15%	0.14	97.96%	0.15
SVM Non-Linear	99.07%	0.36	97.96%	0.22

Activity Recognition Dataset Results:

Classifier	Accuracy	Running Time (s)	Accuracy with Scaled Data	Running Time of Scaled Data (s)
Perceptron	73.97%	83.45	98.02%	72.20
Decision Tree	99.78%	78.67	99.78%	72.78
K-nearest neighbor	73.97%	52.22	94.24%	1867.43
Logistic regression	74.26%	333.07	99.08%	748.34
SVM Linear	N/A	Took more than 3 hrs	99.90%	219.61
SVM Non-Linear	N/A	Took more than 3 hrs	99.79%	755.12

Decision Tree:

Pre-Pruning (Early Stopping Rule)
Stop the algorithm before it becomes a fully-grown tree
Typical stopping conditions for a node:

I. Stop if all instances belong to the same class
II. Stop if all the attribute values are the same

More restrictive conditions:

I. Stop if number of instances is less than some user-specified threshold
II. Stop if class distribution of instances are independent of the available features
III. Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

Post-pruning

I. Grow decision tree to its entirety

II. Trim the nodes of the decision tree in a bottom- up fashion
III. If generalization error improves after trimming, replace sub-tree by a leaf node.
IV. Class label of each node is determined from majority class of instances in the sub-tree Can use
V. MDL for post-pruning

max_depth: The maximum depth of the tree. If None, then nodes are expanded until all leaves arepure or until all leaves contain less than min_samples_splits amples.EARLY STOPPING
min_samples_split: The minimum number of samples required to split an internal node.EARLYSTOPPING
Make sure to use threshold for these values to avoid complete growth of tree.

Strategies Implementation in the source code:

Line # 170 to 205