Project 2: Compare classifiers in scikit-learn library

Understanding and comparing several classification algorithms that are provided by the Python scikit-learn library.

Since all the classifiers are used using the scikit learn library, I have designed one module main.py that would take classifier name and datapath for the dataset file as the input. If the digits is passed as the datapath, then the if else statement would get the digits dataset offered by scikit-learn library and if the datapath is the actual file path then it would read that dataset.

For the classifiers, we can pass the name as Perceptron, LinearSVM, NonlinearSVM, DecisionTree, KNN and LogRegression.

The dataset values are standarized and splitted to training and testing dataset. The misclassified samples, accuracy and time taken are printed as the ouputs for the program.

Analysis

The tabular form for accuracy and time taken by each classifiers are:

i. for digits:

Classifiers	Accuracy	Time Taken(seconds)
Perceptron	0.08	0.267
Linear SVM	0.18	0.222
Non Linear SVM	0.29	0.239
Decision Tree	0.27	0.157
K-Nearest Neighbor	0.22	0.172
Logistic Regression	0.26	0.226

The accuracy was highest in the Non Linear SVM model and lowest in the Perceptron model. Similary, Perceptron took the longest time to complete the classification and the fastest was Decision Tree. Looking at the accuracy and time taken, if we want higher accuracy we can use Non linear SVM as the best fit classifier and K-nearest neighbor clssifier if we wan higher accuracy in low time.

ii. for REALDISP Activity Recognition Dataset

Classifiers	Accuracy	Time Taken
Perceptron	0.32	14.122
Linear SVM	0.7	9.246
Non Linear SVM	0.7	1149.677
Decision Tree	0.7	6.962
K-Nearest Neighbor	0.69	7.530
Logistic Regression	0.7	9.997

The accuracy was around 0.7 for all the classifiers except the Perceptron model i.e Perceptron had the lowest accuracy among all. Even though the accuracy was 0.7 for the nonlinear SVM model, the time taken by the module is about 19 minutes. It took the longest time and if we look at the time taken and accuracy, Decision Tree seems to be the best fit classifier in this case.

By comparing the both the tables for digits and REALDISP Activity Recognition Dataset, Perceptron has the least accuracy rate.

Decision Tree Classifier

The two strategies that the Decision Tree Classifier implements to pre-prune or post-prune the tree. are:

i. min_samples_split

Checking the number of samples required to split an internal node which should be at least equal to min_sample_split. The default value is 2.

ii. min_samples_leaf

Checking the number of samples at a leaf node. The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. The default value is 1.

The repository file is scikit-learn/sklearn/tree/tree.py for both the strategies and the link to file is https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/tree/tree.py .

The strategy for min_samples_leaf is implemented from line 176 to line 187 inside the BaseDecisionTree class.

```
    if isinstance(self.min_samples_leaf, (numbers.Integral, np.integer)):
        if not 1 <= self.min_samples_leaf:
            raise ValueError("min_samples_leaf must be at least 1 "
                             "or in (0, 0.5], got %s"
                             % self.min_samples_leaf)
        min_samples_leaf = self.min_samples_leaf
    else:  # float
        if not 0. < self.min_samples_leaf <= 0.5:
            raise ValueError("min_samples_leaf must be at least 1 "
                             "or in (0, 0.5], got %s"
                             % self.min_samples_leaf)
        min_samples_leaf = int(ceil(self.min_samples_leaf * n_samples))
```

The strategy for min_samples_split is implemented in line 189 to line 205 inside the BaseDecisionTree class.

```
        if isinstance(self.min_samples_split, (numbers.Integral, np.integer)):
            if not 2 <= self.min_samples_split:
                raise ValueError("min_samples_split must be an integer "
                                 "greater than 1 or a float in (0.0, 1.0]; "
                                 "got the integer %s"
                                 % self.min_samples_split)
            min_samples_split = self.min_samples_split
        else:  # float
            if not 0. < self.min_samples_split <= 1.:
                raise ValueError("min_samples_split must be an integer "
                                 "greater than 1 or a float in (0.0, 1.0]; "
                                 "got the float %s"
                                 % self.min_samples_split)
            min_samples_split = int(ceil(self.min_samples_split * n_samples))
            min_samples_split = max(2, min_samples_split)
        min_samples_split = max(min_samples_split, 2 * min_samples_leaf)
```

Conclusion

This project helped me understand and compare different classifiers model like Perceptron, Support Vector Machine, Decision Tree Classifier, K-nearest neighbor and Logistic Regression. By testing the two datasets on these classifiers, we can conclude that Decision Tree Classifier was the fastest among all.