University of Waterloo
ECE 657A: Data and Knowledge Modeling and Analysis
Winter 2020
Assignment 1: Basic Environment Set-up and Classification
Due: February 2, 2020 11:59pm

Overview

Collaboration: Do your work and report individually. You can collaborate on the right tools to use and setting up your programming environment.

Hand in: One report per person, via the LEARN dropbox in PDF format. Also submit the code/scripts needed to reproduce your work as a python jupyter notebook.

Specific objectives:

- Establish your software stack to carry out data analysis assignments for the rest of the course.
- Load datasets and perform some exploratory plots.
- Study how to apply some of the methods discussed in class and gain experience on the use of this classification methods (KNN, SVM, Decision Trees, Random Forest, Gradient Tree Boosting).

Tools: You can use libraries available in python. You need to mention which libraries you are using, any blogs or papers you used to figure out how to carry out your calculations.

Dataset

Use the Iris dataset from "sklearn.datasets.load_iris". This dataset is a classic and fairly simple benchmark for basic machine learning algorithms. It includes different features of three Iris flower species (setosa, versicolor, virginica).

Question 1

To begin understanding the dataset, plot the pairs plot (scatter plot matrix) of the data. Note that the pairs plot includes the scatter plots of every dimension versus another dimension.
After plotting, describe in words your interpretation of the separability of the three classes in terms of different features (dimensions).

Question 2: KNN

Classify the data using a KNN classifier. Tune the hyperparameters of the KNN classifier using sklearn functions. Plot the different validation accuracies against the values of the parameter and select the best hyperparameter to train the model. Report the resulting accuracy.
Do the following details:

1. First, divide the data into train, validation, and test sets (60%, 20%, 20%) ***Note:*** use random_state=42 in the train_test_split function to get the same split every time you run the program.

2. Train the model with each classifier's default parameters. Use the train set and test the model on the test set. Store the accuracy of the model.

3. Then, you should find the best parameters of the classifiers, in this case, k for KNN. For this, use the validation set, NOT the test set. To find the best parameter you should:

   (a) Pick a value of parameter. Test the following values for validation: k: {1, 5, 10, 15, 20, 25, 30, 35}

   (b) Train the model using the train set.

   (c) Test the model with the validation set. Store the accuracy.

   (d) When you finish trying all the possible parameters, plot a figure that shows the relationship between the accuracy and the parameter. Report the best k in terms of classification accuracy.

4. Now, using the best found parameters, train the model using the train set and test the model on the test set. Report the accuracy of the model.

Question 3: SVM

Classify data using a linear SVM classifier. Evaluate the best value for the term C amongst values:
C: [0.1, 0.5, 1, 2, 5, 10, 20, 50]
Here, rather than the procedures (2-4) as in Question 2, use 10-fold cross validation. First, randomly divide data into (80%, 20%) portions of train-validation and test sets (with random_state=42). Then, apply 10-fold cross validation on the train-validation set. In every fold, 90% of data is used for training and 10% of data for validation. For every C value, a mean accuracy of the folds is found. The best mean accuracy determines the best value for C. Plot the mean accuracy versus the C values. Finally, report the test accuracy.

Question 4: Tree-based Classifiers

Classify the data using three tree-based classifiers: Decision Trees, Random Forests and Gradient Tree Boosting. Tune the hyper-parameters of the classifier using 10-fold cross validation and sklearn functions (as in Question 3). Evaluate the best value for the number of trees and maximum depth of trees.

For decision tree:

- max depth: {3, 5, 10, None (grow until the end)}

For this, plot the mean accuracy versus the maximum depth.

For random forest:

- number of trees: {5, 10, 50, 150, 200}
- max depth: {3, 5, 10, None (grow until the end)}

For this, the plot should be a **heat plot**. You should have (5 * 4) mean accuracies for different values of number of trees and maximum depth.

For Gradient Tree Boosting (on sklearn it is `GradientBoostingClassifier`):

- number of estimators: {5, 10, 50, 150, 200}

For this, plot the mean accuracy versus the number of estimators.
Note: the number of 'trees' grown by GBT is `n_classes` $\times$ `n_estimators` *but this is handled automatically.*

Please leave the other parameters as default in sklearn.

Question 5: Analysis

1. Explain why you had to split the dataset into train and test sets?
2. Explain why when finding the best parameters for KNN you didn't evaluate directly on the test set and had to use a validation test.
3. What was the effect of changing k for KNN. Was the accuracy always affected the same way with an increase of k? Why do you think this happened?
4. What was the relative effect of changing the max depths for decision tree, random forests, and gradient boosting? Explain the reason for this.
5. What was the relative effect of changing the number of tree depths for random forests, and gradient boosting? Explain the reason for this.
6. What does the parameter C define in the SVM classifier? What effect did you observe and why do you think this happened?

Notes

You might find the following links are useful to solve this assignment:

- https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html