# Department of Computing

**CS471: Machine Learning**

**Class: BESE-7AB**

**Lab 07: Decision Tree**

**CL03: Apply a variety of learning algorithms to data for solution development**

**Date: 14-03-2019**

**Time: 10:00 am– 1:00 pm & 2:00 pm-5:00 pm**

**Instructor: Dr. Pakeeza Akram**

# Lab 07: Decision Tree

**Introduction**

Decision Trees (DTs) are a non-parametric supervised learning method used for [classification](http://scikit-learn.org/stable/modules/tree.html#tree-classification) and [regression](http://scikit-learn.org/stable/modules/tree.html#tree-regression). The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.

**Objective**

In this lab you will use cleaned IMDB reviews from lab 6 to train a classifier. Each movie review is labeled with a positive or a negative sentiment. This is your training dataset. You will use cleaned reviews with labels to train a Decision Tree. The trained classifier when presented with a new review will predict if the review is positive or negative.

**Tools/Software Requirement**

Python, scikit-learn

**Reference**

<https://www.kaggle.com/c/word2vec-nlp-tutorial/details/part-1-for-beginners-bag-of-words>

http://scikit-learn.org/stable/modules/tree.html

**Description**

In the below tutorial there is an example of Decision tree classifier using iris data set.

[**DecisionTreeClassifier**](http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier) is a class capable of performing multi-class classification on a dataset.

As with other classifiers, **[DecisionTreeClassifier](http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html" \l "sklearn.tree.DecisionTreeClassifier" \o "sklearn.tree.DecisionTreeClassifier)** takes as input two arrays: an array X, sparse or dense, of size [n\_samples, n\_features] holding the training samples, and an array Y of integer values, size [n\_samples], holding the class labels for the training samples:

**>>> from** **sklearn** **import** tree

**>>>** X = [[0, 0], [1, 1]]

**>>>** Y = [0, 1]

**>>>** clf = tree.DecisionTreeClassifier()

**>>>** clf = clf.fit(X, Y)

After being fitted, the model can then be used to predict the class of samples:

**>>>** clf.predict([[2., 2.]])

array([1])

Alternatively, the probability of each class can be predicted, which is the fraction of training samples of the same class in a leaf:

**>>>** clf.predict\_proba([[2., 2.]])

array([[ 0., 1.]])

[**DecisionTreeClassifier**](http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier) is capable of both binary (where the labels are [-1, 1]) classification and multiclass (where the labels are [0, …, K-1]) classification.

Using the Iris dataset, we can construct a tree as follows:

**>>> from** **sklearn.datasets** **import** load\_iris

**>>> from** **sklearn** **import** tree

**>>>** iris = load\_iris()

**>>>** clf = tree.DecisionTreeClassifier()

**>>>** clf = clf.fit(iris.data, iris.target)

Once trained, we can export the tree in [Graphviz](http://www.graphviz.org/) format using the **[export\_graphviz](http://scikit-learn.org/stable/modules/generated/sklearn.tree.export_graphviz.html" \l "sklearn.tree.export_graphviz" \o "sklearn.tree.export_graphviz)** exporter. If you use the [conda](http://conda.io/) package manager, the graphviz binaries and the python package can be installed with

conda install python-graphviz

Alternatively binaries for graphviz can be downloaded from the graphviz project homepage, and the Python wrapper installed from pypi with pip install graphviz.

Below is an example graphviz export of the above tree trained on the entire iris dataset; the results are saved in an output file iris.pdf:

**>>> import** **graphviz**

**>>>** dot\_data = tree.export\_graphviz(clf, out\_file=**None**)

**>>>** graph = graphviz.Source(dot\_data)

**>>>** graph.render("iris")

The **[export\_graphviz](http://scikit-learn.org/stable/modules/generated/sklearn.tree.export_graphviz.html" \l "sklearn.tree.export_graphviz" \o "sklearn.tree.export_graphviz)** exporter also supports a variety of aesthetic options, including coloring nodes by their class (or value for regression) and using explicit variable and class names if desired. Jupyter notebooks also render these plots inline automatically:

**>>>** dot\_data = tree.export\_graphviz(clf, out\_file=**None**,

feature\_names=iris.feature\_names,

class\_names=iris.target\_names,

filled=True, rounded=True,

special\_characters=True)

**>>>** graph = graphviz.Source(dot\_data)

**>>>** graph

After being fitted, the model can then be used to predict the class of samples:

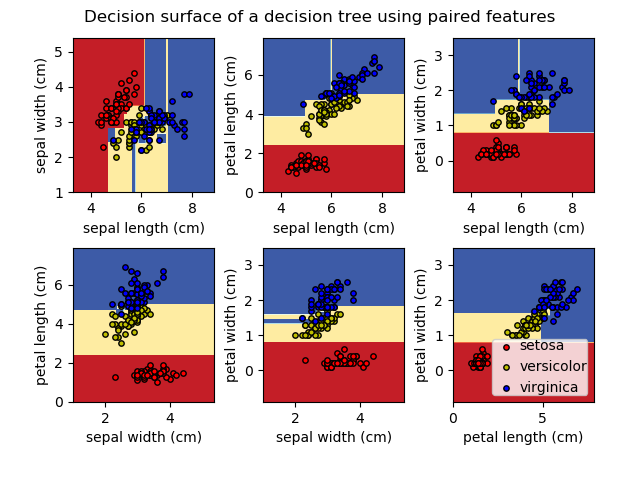
**>>>** clf.predict(iris.data[:1, :])

array([0])

Alternatively, the probability of each class can be predicted, which is the fraction of training samples of the same class in a leaf:

**>>>** clf.predict\_proba(iris.data[:1, :])

array([[ 1., 0., 0.]])

[](http://scikit-learn.org/stable/auto_examples/tree/plot_iris.html)

**Lab Task**

1. In the previous Lab you learnt how to clean data. Your lab task is to train a decision tree classifier using IMDB movie reviews (25000 movie reviews). Remember to clean the reviews before building bag of word representation. You can either use the function you wrote in the last lab to clean the reviews or use review\_to\_words.py.
2. Calculate accuracy on different vocabulary sizes.

**Deliverables**

Upload Word file containing the task

**Reference**

http://scikit-learn.org/stable/modules/tree.html