# PyNexradML - Main Scripts

**datastore.py**

The datastore script is used to manage the nexrad data available for training. This script parses nexrad radar files (.Z or .gz) and adds them to an hdf5 file called *datastore.h5*. The sweeps in this file contain the reflectivity, radial velocity and spectrum width moments for the 0.5 degree elevation angle sweep. Data is stored in polar format.

Sweeps are further organized into named datasets that can be selected or combined for use in classifier training. Initial construction of the datastore requires the existence of a configuration file named *manifest.cfg*. This file will most likely be written by hand and defines the datasets as well as provides expert classifications for the sweeps being added. A sample manifest.cfg is shown below.

[rd1]

6500KILX20010907\_000028.Z = -1

6500KILX20010908\_000251.Z = 1

6500KILX20010919\_000545.Z = -1

6500KILX20010924\_221013.Z = 1

6500KILX20010925\_221419.Z = 1

6500KILX20010926\_221409.Z = 1

6500KILX20010927\_220927.Z = 1

[rd2]

6500KLIX20020925\_031530.Z = -1

6500KLIX20020926\_021310.Z = -1

6500KLIX20021026\_024415.Z = -1

6500KLIX20050829\_134926.Z = -1

6500KLIX20031013\_025737.Z = 1

6500KLIX20031015\_014940.Z = 1

6500KCRP20050403\_035853.Z = 1

6500KLCH20050420\_040528.Z = 1

This sample file defines two datasets, *rd1* and *rd2,* each containing the sweeps listed below them. The manifest.cfg file should be located in the same directory as the data being imported.

**Building:**

Initially building or rebuilding the datastore can be accomplished by running the datastore script with the -r argument as shown below, where D:\Data would be replaced by the location of your data.

>python datastore.py -r -d D:\Data

**Importing:**

Importing new data is equally straight forward. Simply put the new data into its own directory and create a new manifest.cfg file containing the classifications and datasets for the new data, placing it in the same directory. Importing is performed with the *-i* argument:

>python datastore.py -d D:\Data -i D:\new\_data

Other notable command line options include *--show\_datasets* and *--desc\_dataset* which are used to display the current datasets in the datastore or sweeps in a dataset respectively.

**trainer.py**

The trainer script is used to train and validate neural network classifers. The trainer script supports the following arguments.

-d, --data\_dir The -d/--data\_dir option is used to specify the location of the datastore (datastore.h5)

--epochs Specify the number of epochs (presentations of data) to use when training the network. A larger number may lead to better training, but will require more computation time.

--features Specify the features to train on (defined in features.py). Individual features are described further down.

--filters Specify the filters to apply to the data. Filters can either filter out data or they can modify data. Individual filters are described below.

--norm Specify any normalization to perform on the selected features. Different normalizers are described below.

-t Specify the dataset(s) to use for classifier training.

-o Specify the output name for the classifier. For example -o myclassifier will result in myclassifier.net and myclassifier.proc files being created. These files can be subsequently used to classify new data.

**Features:**

Reflectivity() The reflectivity moment for the base nexrad data in dBz.

Velocity() The radial velocity moment for the base nexrad data in m/s.

SpectrumWidth() The spectrum width moment for the base nexrad data.

Range() The range of a pulse volume in meters

Variance(BaseFeature)

Kurtosis(BaseFeature)

Skew(BaseFeature)

**Filters:**

RangeConstraints(min,max) Filter.

RemoveBadValues(BaseFeature) Filter.

SmootheBadValues(BaseFeature) Filter.

SubSample(Percent) Filter.

**Normalizers :**

SymmetricNormalizer(\*columns) Normalizer.

**Examples:**

**screen.py**

The screen script can be used to screen new data for the presence of biological data. A classifier, that has previously been constructed using the trainer script, is applied to the new data and a message is printed indicating whether the sweep matches or does not match the criteria specified.

-d, --data\_dir The -d/--data\_dir option is used to specify the location of the new data to screen. New data is expected to be in the original level II data format (\*.Z/\*.gz)

-t Specifies a threshold for classification which serves as the classification criteria. This threshold is a target ratio or biological to nonbiological pulse volumes.

-i Specify the classifier to use (e.g. myclassifier).

**Examples:**

>python screen.py -i myclassifier -d D:\new\_data -t 0.7