**Python version 2.7 or 3.5**

scipy

numpy

matplotlib

pandas

sklearn

**\*\*\*\*\*\*\*\*\* CHECK VERSION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

# Check the versions of libraries

# Python version

import sys

print('Python: {}'.format(sys.version))

# scipy

import scipy

print('scipy: {}'.format(scipy.\_\_version\_\_))

# numpy

import numpy

print('numpy: {}'.format(numpy.\_\_version\_\_))

# matplotlib

import matplotlib

print('matplotlib: {}'.format(matplotlib.\_\_version\_\_))

# pandas

import pandas

print('pandas: {}'.format(pandas.\_\_version\_\_))

# scikit-learn

import sklearn

print('sklearn: {}'.format(sklearn.\_\_version\_\_))

**\*\*\*\*\*\*\*\*\*\*\*\* IMPORTS LIBRARY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

|  |  |
| --- | --- |
|  | # Load libraries  import pandas  from pandas.plotting import scatter\_matrix  import matplotlib.pyplot as plt  from sklearn import model\_selection  from sklearn.metrics import classification\_report  from sklearn.metrics import confusion\_matrix  from sklearn.metrics import accuracy\_score  from sklearn.linear\_model import LogisticRegression  from sklearn.tree import DecisionTreeClassifier  from sklearn.neighbors import KNeighborsClassifier  from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  from sklearn.naive\_bayes import GaussianNB  from sklearn.svm import SVC |

**\*\*\*\*\*\*\*\*\*\*\*\*\* LOAD DATA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

|  |  |
| --- | --- |
|  | # Load dataset  url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"  names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']  dataset = pandas.read\_csv(url, names=names) |

## \*\*\*\*\*\*\* Summarize the Dataset \*\*\*\*\*

# shape

print(dataset.shape)

|  |  |
| --- | --- |
|  | # head  print(dataset.head(20)) |

# descriptions

print(dataset.describe())

### \*\*\*\*\*\*\* Class Distribution \*\*\*\*\*

print(dataset.groupby('class').size())

SUM()

MAX()

MEAN()

MIN()

\*\*\*\*\*\* Data Visualization \*\*\*\*\*\*\*\*\*\*\*\*

|  |  |
| --- | --- |
|  | # box and whisker plots  dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)  plt.show() |

|  |  |
| --- | --- |
|  | # histograms  dataset.hist()  plt.show() |

scatter\_matrix(dataset)

plt.show()

## \*\*\*\* Evaluate Some Algorithms \*\*\*\*

Here is what we are going to cover in this step:

1. Separate out a validation dataset.
2. Set-up the test harness to use 10-fold cross validation.
3. Build 5 different models to predict species from flower measurements
4. Select the best model.

### Create a Validation Dataset

That is, we are going to hold back some data that the algorithms will not get to see and we will use this data to get a second and independent idea of how accurate the best model might actually be.

We will split the loaded dataset into two, 80% of which we will use to train our models and 20% that we will hold back as a validation dataset.

array = dataset.values

X = array[:,0:4]

Y = array[:,4]

validation\_size = 0.20

seed = 7

X\_train, X\_validation, Y\_train, Y\_validation = model\_selection.train\_test\_split(X, Y, test\_size=validation\_size, random\_state=seed)

## You now have training data in the X\_train and Y\_train for preparing models and a X\_validation and Y\_validation sets that we can use later.

|  |  |
| --- | --- |
| Build Models We don’t know which algorithms would be good on this problem or what configurations to use. We get an idea from the plots that some of the classes are partially linearly separable in some dimensions, so we are expecting generally good results.  Let’s evaluate 6 different algorithms:   * Logistic Regression (LR) * Linear Discriminant Analysis (LDA) * K-Nearest Neighbors (KNN). * Classification and Regression Trees (CART). * Gaussian Naive Bayes (NB). * Support Vector Machines (SVM).   This is a good mixture of simple linear (LR and LDA), nonlinear (KNN, CART, NB and SVM) algorithms. We reset the random number seed before each run to ensure that the evaluation of each algorithm is performed using exactly the same data splits. It ensures the results are directly comparable.  models = []  models.append(('LR', LogisticRegression()))  models.append(('LDA', LinearDiscriminantAnalysis()))  models.append(('KNN', KNeighborsClassifier()))  models.append(('CART', DecisionTreeClassifier()))  models.append(('NB', GaussianNB()))  models.append(('SVM', SVC()))  # evaluate each model in turn  results = []  names = []  for name, model in models:  kfold = model\_selection.KFold(n\_splits=10, random\_state=seed)  cv\_results = model\_selection.cross\_val\_score(model, X\_train, Y\_train, cv=kfold, scoring=scoring)  results.append(cv\_results)  names.append(name)  msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())  print(msg) | # Test options and evaluation metric  seed = 7  scoring = 'accuracy' |