# 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\_\_))

# Load libraries

import pandas

#from pandas.tools.plotting import scatter\_matrix

import matplotlib.pyplot as plt

from sklearn import cross\_validation

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 dataset

names = ['blue', 'green', 'red', 'class'] #column names

dataset = pandas.read\_csv("xyz.csv", names=names)

# Dimensions of dataset

#shape

print(dataset.shape)

#head

print(dataset.head(20))

#statistical summary

print(dataset.describe())

#class distribution

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

#data visualization

#univariate plots

#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 plot matrix

#scatter\_matrix(dataset)

#plt.show()

#create a validation dataset

# split-out validation dataset

array = dataset.values

X = array[:,0:3]

Y = array[:,3]

validation\_size = 0.20

seed = 7

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

# Test options and evaluation metric

num\_folds = 10

num\_instances = len(X\_train)

seed = 7

scoring = 'accuracy'

# Spot Check Algorithms

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 = cross\_validation.KFold(n=num\_instances, n\_folds=num\_folds, random\_state=seed)

cv\_results = cross\_validation.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)

# Compare Algorithms

fig = plt.figure()

fig.suptitle('Algorithm Comparison')

ax = fig.add\_subplot(111)

plt.boxplot(results)

ax.set\_xticklabels(names)

plt.show()

# Make predictions on validation dataset

knn = KNeighborsClassifier()

knn.fit(X\_train, Y\_train)

predictions = knn.predict(X\_validation)

print(accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))