Table of Contents

[I.Installing the Scipy Libraries 2](#_Toc508814207)

[II.Load Data Set 5](#_Toc508814208)

[III.Data Summary & Visualization 7](#_Toc508814209)

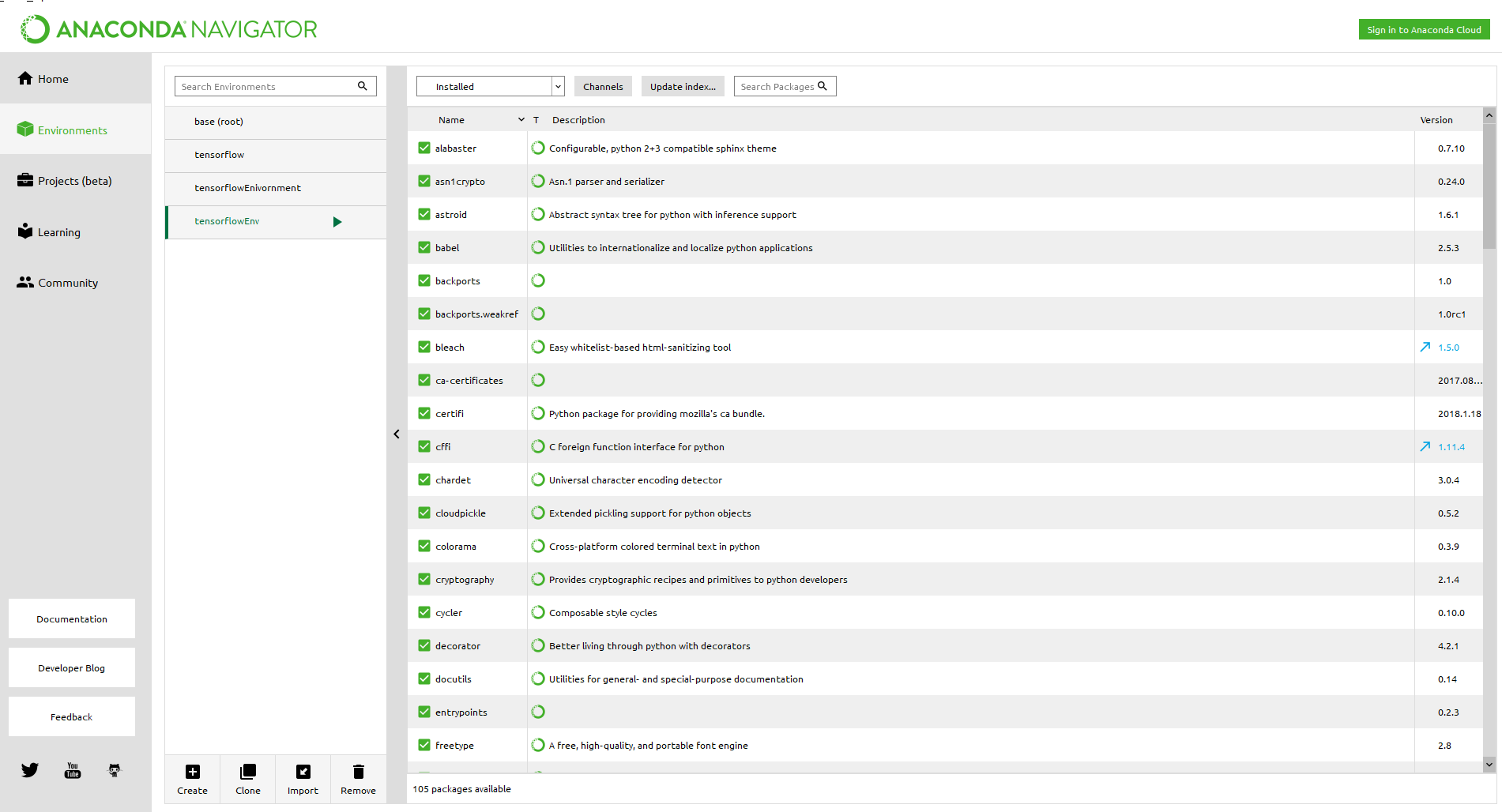
[IV.Evaluate the Algorithm 9](#_Toc508814210)

[V.Python file 12](#_Toc508814211)

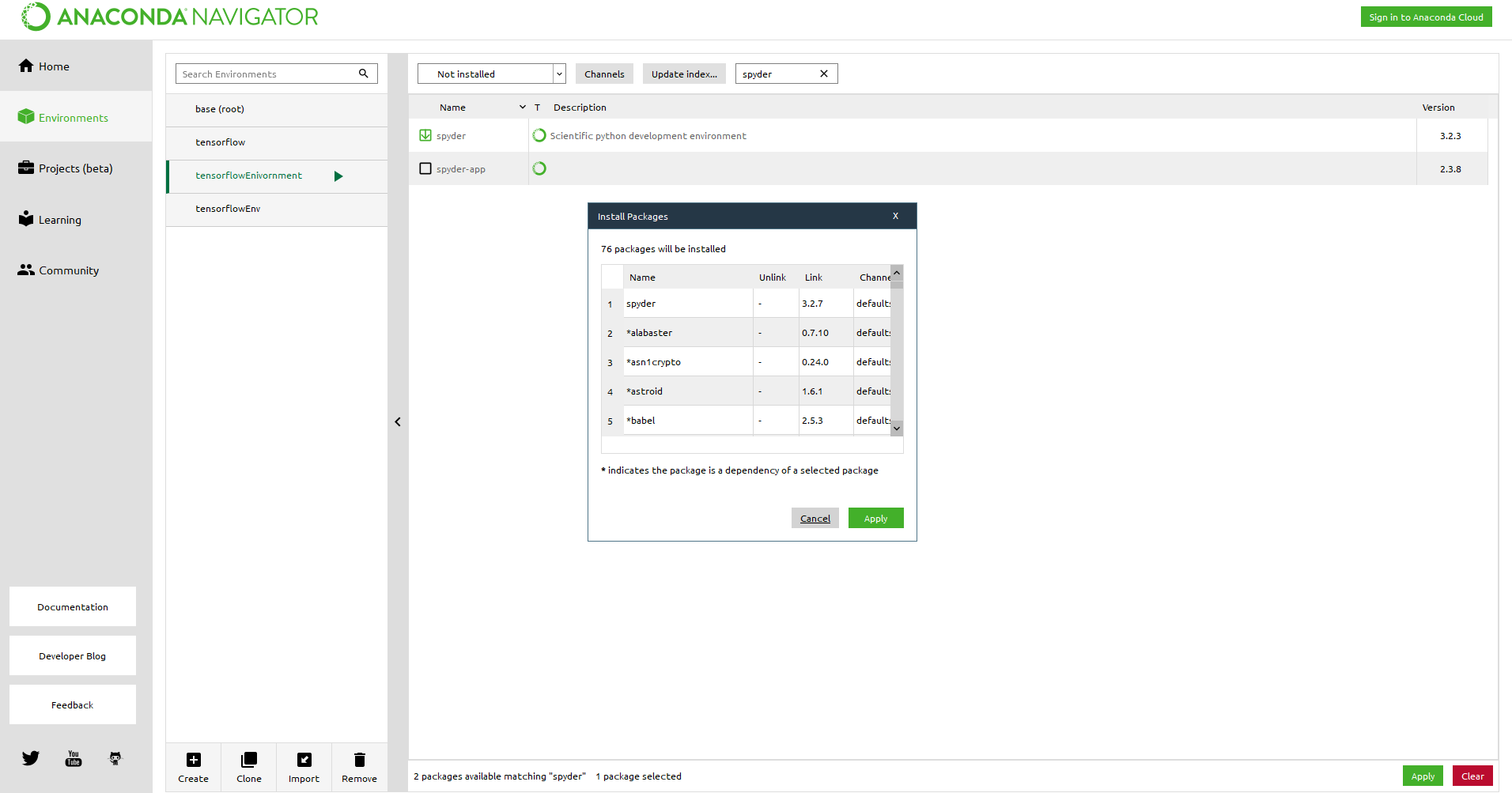
# Installing the Scipy Libraries

**Step 1**: In Anaconda, in your tensorflow environment check if the following packages are installed

* scipy
* numpy
* matplotlib
* pandas
* sklearn



If the above packages are not installed search for the package in search box, select the package and click on apply as show in the screen shot below.



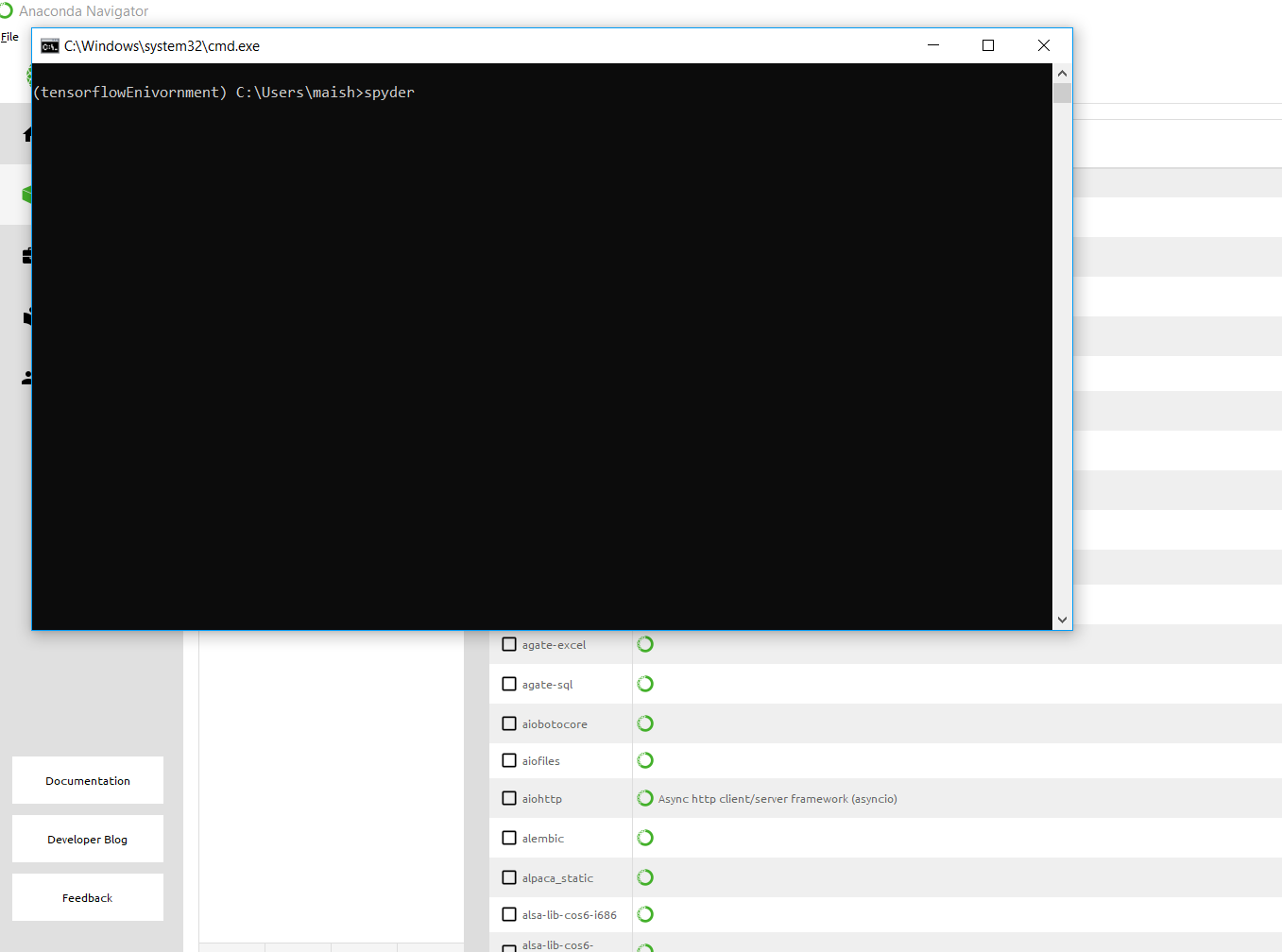
**2**

**3**

**4**

**1**

**Step 2:** Once the packages are successfully installed, open terminal , type “spyder” and click enter. This will open Spyder



**Step 3:** If the packages are correctly installed, typing the below code should give the following output without error

**Code:**

# 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\_\_))

**Output:**

Python: 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit (AMD64)]

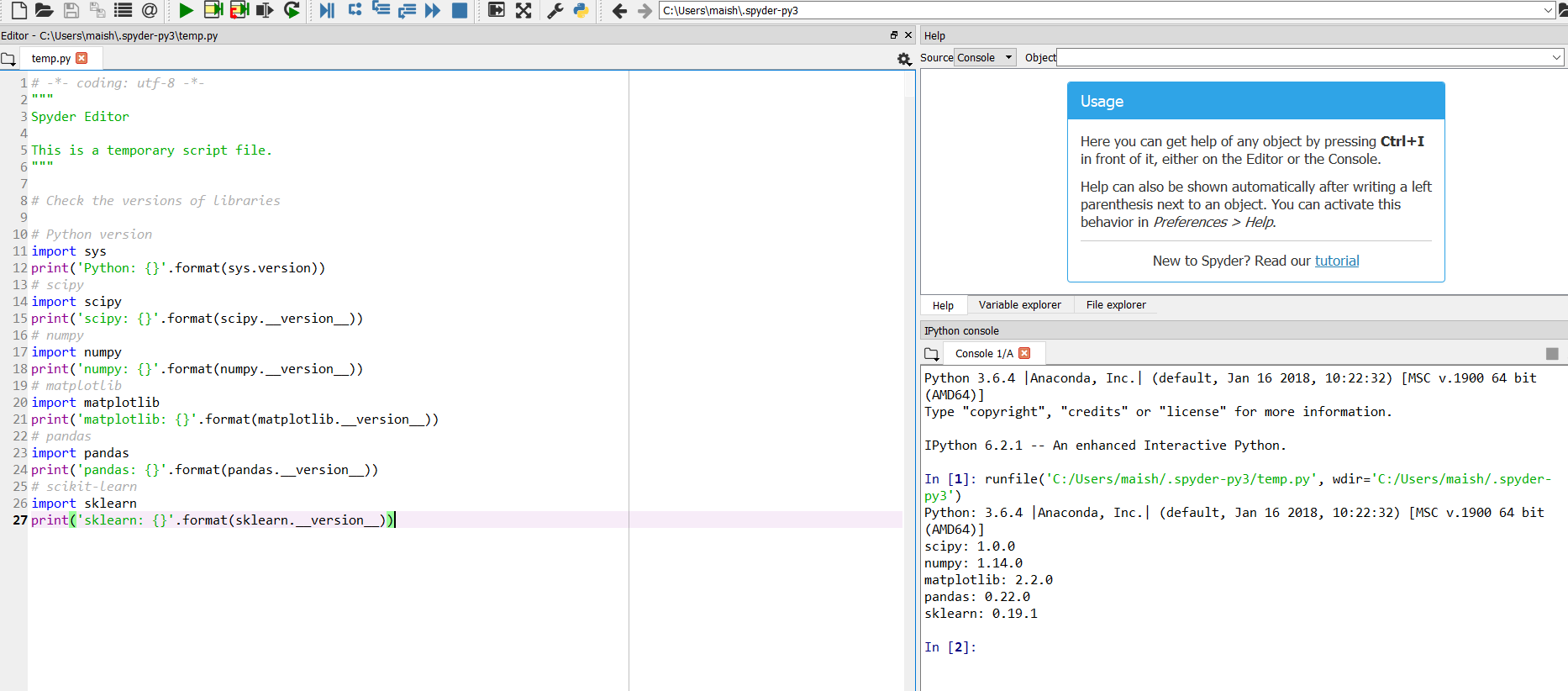
scipy: 1.0.0

numpy: 1.14.0

matplotlib: 2.2.0

pandas: 0.22.0

sklearn: 0.19.1



# Load Data Set

**Step 1:** To load the dataset, first import the libraries using the below code

**Code:**

# 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

**Step 2:** Load the dataset from UCI Machine Learning Repository using the below code

**Code:**

# 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)

**Step 3:** If the data is loaded successfully running the below code should give you the following output

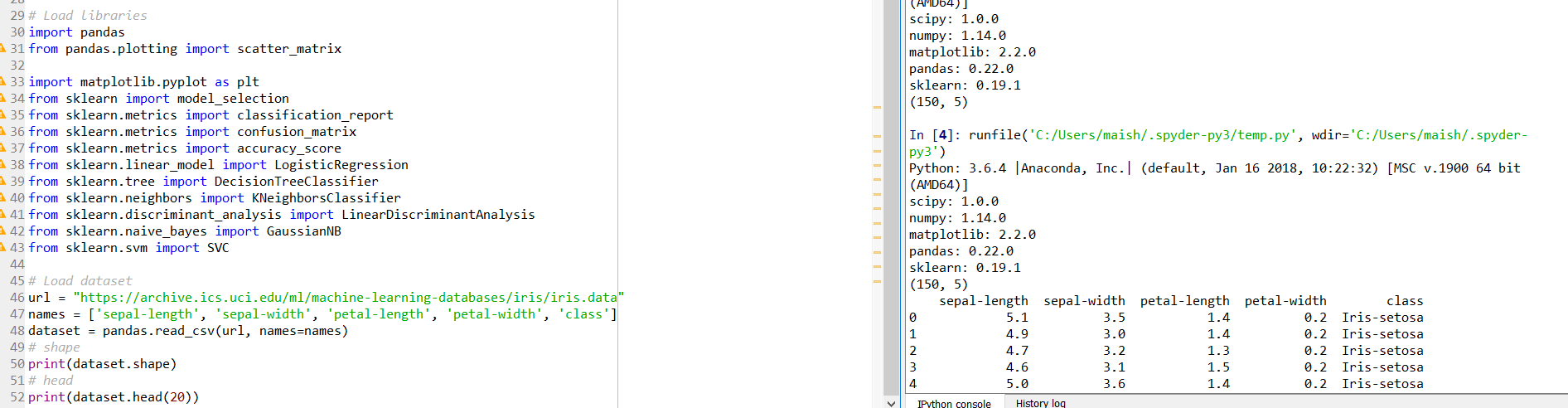
**Code:**

# shape

print(dataset.shape)

**Output:**

(150, 5)



**Code:**

# head

print(dataset.head(20))

**Output:**

sepal-length sepal-width petal-length petal-width class

0 5.1 3.5 1.4 0.2 Iris-setosa

1 4.9 3.0 1.4 0.2 Iris-setosa

2 4.7 3.2 1.3 0.2 Iris-setosa

3 4.6 3.1 1.5 0.2 Iris-setosa

4 5.0 3.6 1.4 0.2 Iris-setosa

5 5.4 3.9 1.7 0.4 Iris-setosa

6 4.6 3.4 1.4 0.3 Iris-setosa

7 5.0 3.4 1.5 0.2 Iris-setosa

8 4.4 2.9 1.4 0.2 Iris-setosa

9 4.9 3.1 1.5 0.1 Iris-setosa

10 5.4 3.7 1.5 0.2 Iris-setosa

11 4.8 3.4 1.6 0.2 Iris-setosa

12 4.8 3.0 1.4 0.1 Iris-setosa

13 4.3 3.0 1.1 0.1 Iris-setosa

14 5.8 4.0 1.2 0.2 Iris-setosa

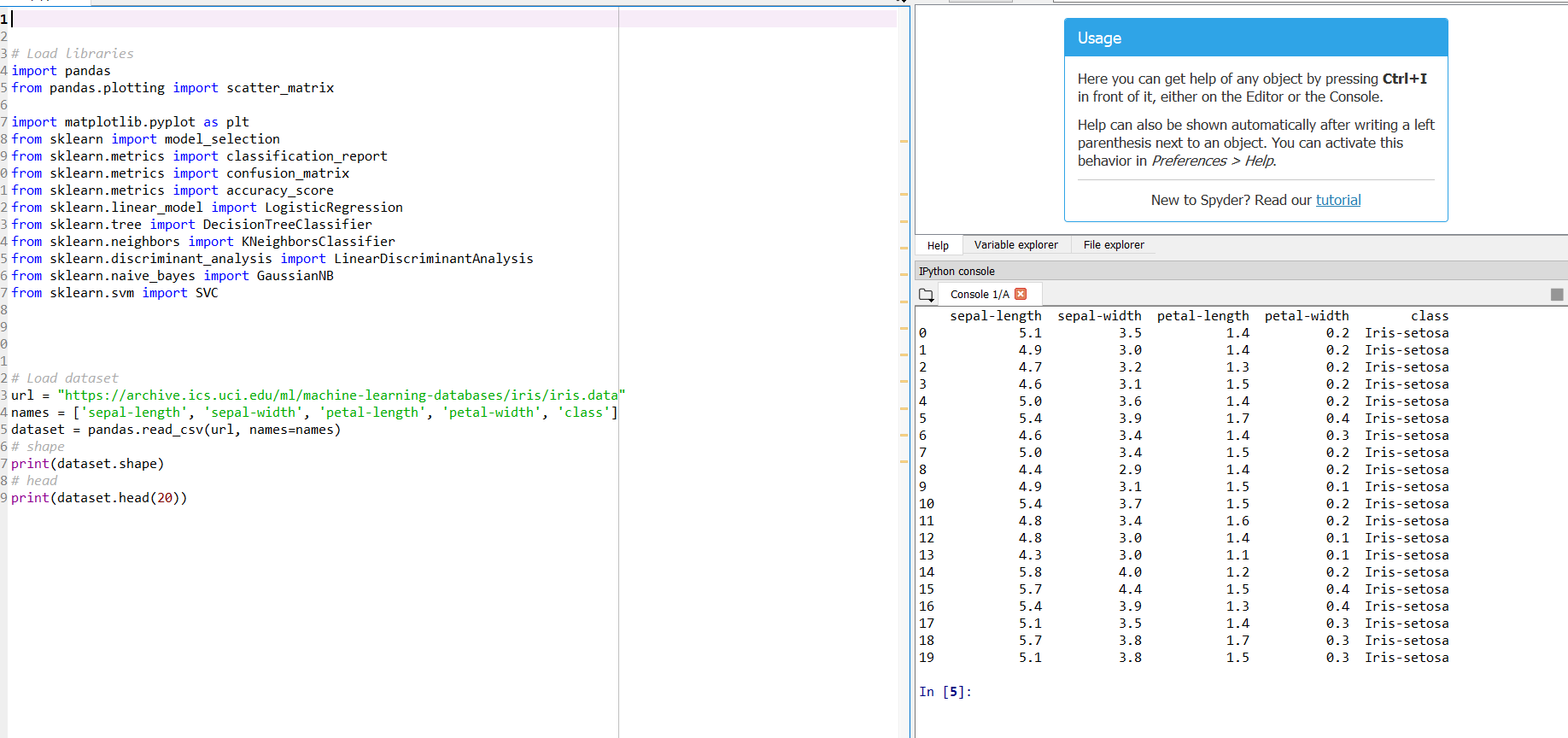
15 5.7 4.4 1.5 0.4 Iris-setosa

16 5.4 3.9 1.3 0.4 Iris-setosa

17 5.1 3.5 1.4 0.3 Iris-setosa

18 5.7 3.8 1.7 0.3 Iris-setosa

19 5.1 3.8 1.5 0.3 Iris-setosa



# Data Summary & Visualization

To view summary of the data, you could use the following code

**Code:**

# descriptions

print(dataset.describe())

**Output:**

sepal-length sepal-width petal-length petal-width

count 150.000000 150.000000 150.000000 150.000000

mean 5.843333 3.054000 3.758667 1.198667

std 0.828066 0.433594 1.764420 0.763161

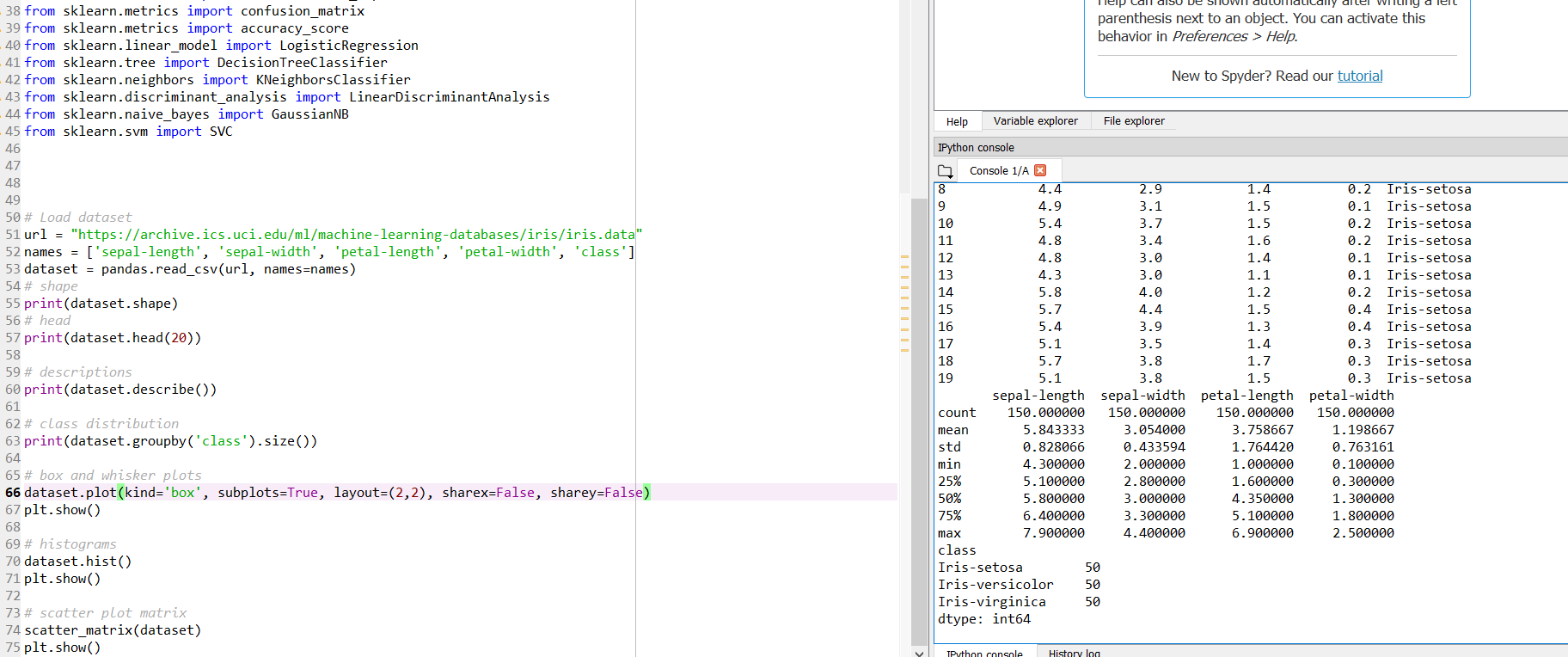
min 4.300000 2.000000 1.000000 0.100000

25% 5.100000 2.800000 1.600000 0.300000

50% 5.800000 3.000000 4.350000 1.300000

75% 6.400000 3.300000 5.100000 1.800000

max 7.900000 4.400000 6.900000 2.500000



To view a visualize your data, the following code can be used

**Code:**

# box and whisker plots

dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)

plt.show()

# histograms

dataset.hist()

plt.show()

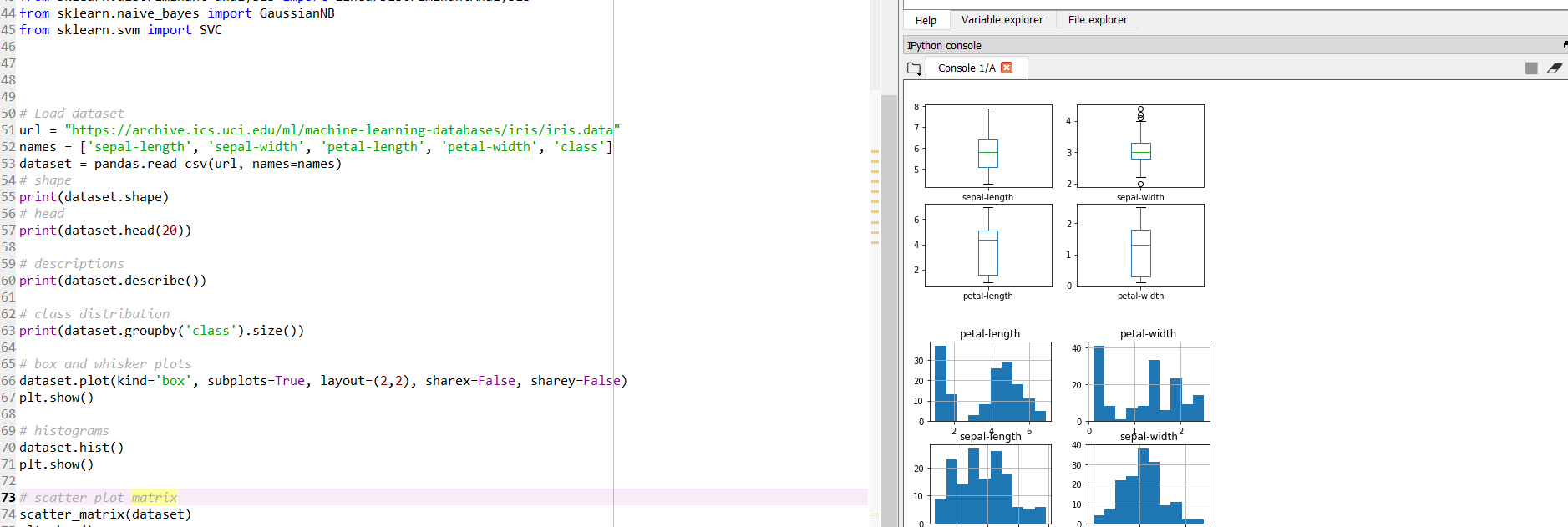
# scatter plot matrix

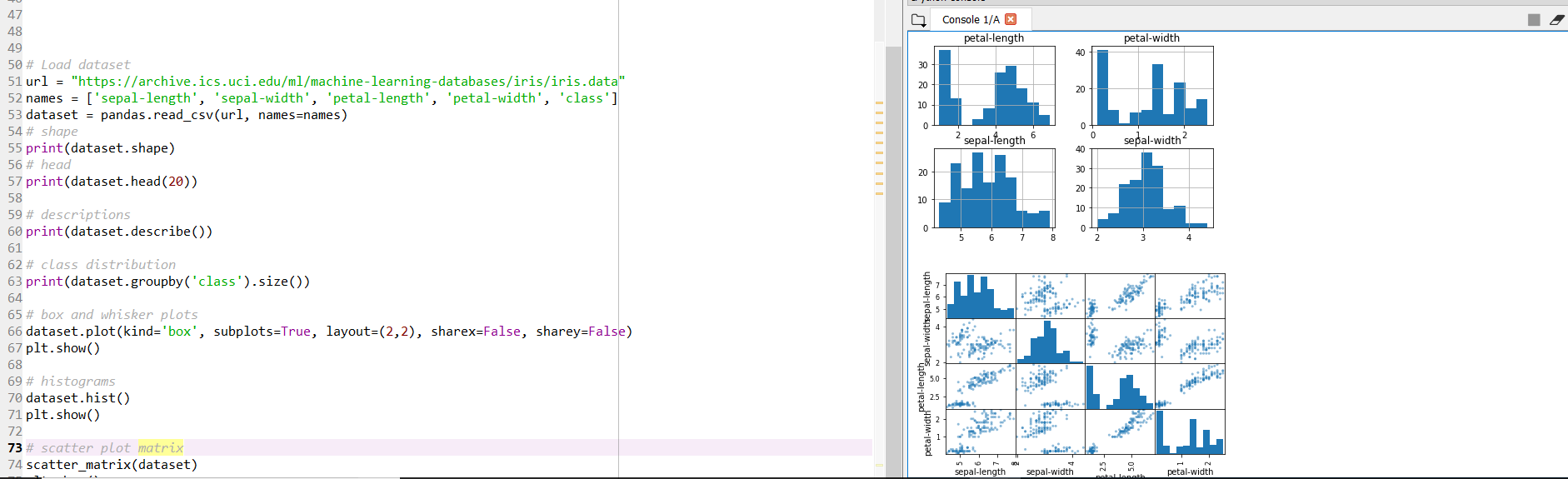
scatter\_matrix(dataset)

plt.show()

**Output:**

As shown in the screenshots below, the output for the above code would be the box plot, histogram and scatter plot respectively





# Evaluate the Algorithm

**Step 1:** create the validation dataset

**Code:**

# Split-out 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)

**Step 2**: create test set

**Code:**

# Test options and evaluation metric

seed = 7

scoring = 'accuracy'

We will use 10-fold cross validation to estimate accuracy.This will split our dataset into 10 parts, train on 9 and test on 1 and repeat for all combinations of train-test splits.

**Step 3:** Build models, the following code is to build the below mentioned models

* Logistic Regression (LR)
* Linear Discriminant Analysis (LDA)
* K-Nearest Neighbors (KNN).
* Classification and Regression Trees (CART).
* Gaussian Naive Bayes (NB).
* Support Vector Machines (SVM).

**Code:**

# 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 = 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)

**Output:**

LR: 0.966667 (0.040825)

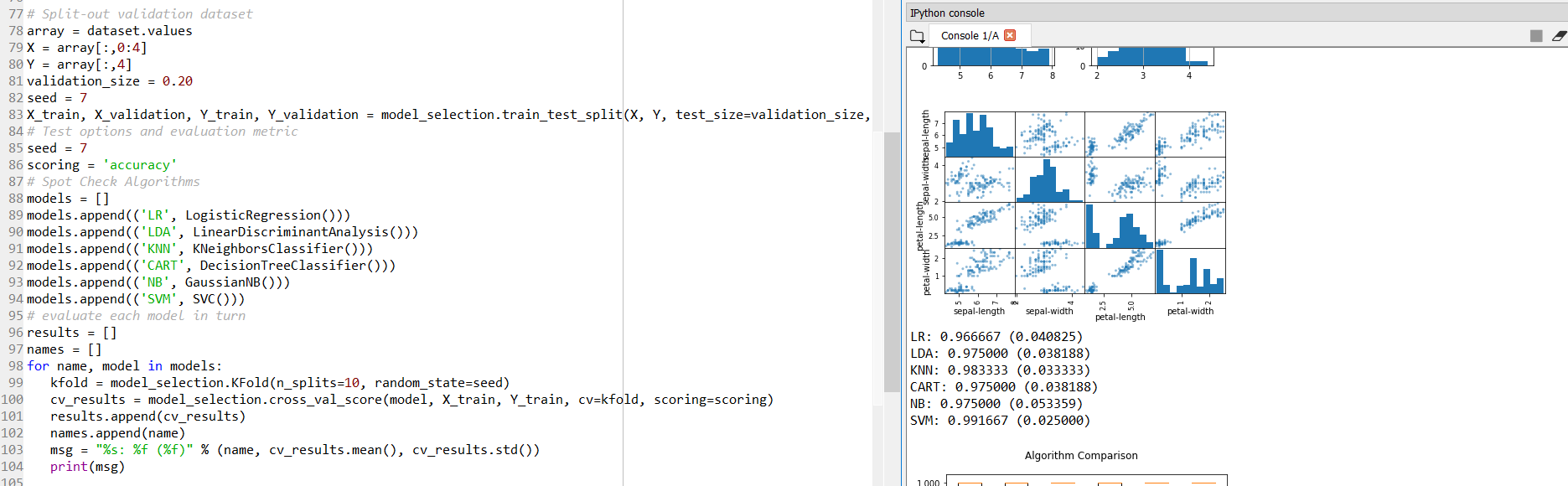
LDA: 0.975000 (0.038188)

KNN: 0.983333 (0.033333)

CART: 0.975000 (0.038188)

NB: 0.975000 (0.053359)

SVM: 0.981667 (0.025000)



**Step 3:** Select the best model

**Code:**

# Compare Algorithms

fig = plt.figure()

fig.suptitle('Algorithm Comparison')

ax = fig.add\_subplot(111)

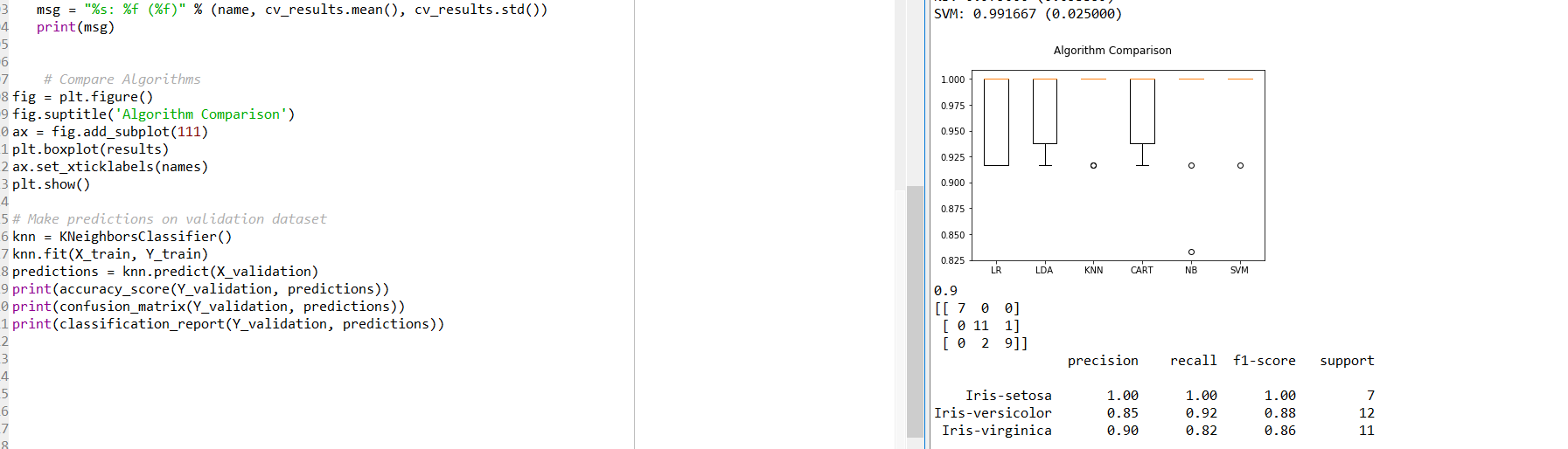
plt.boxplot(results)

ax.set\_xticklabels(names)

plt.show()

**Output:**

As shown in the screenshot below, it will build the plot



**Step 4:** once you have selected the best model the next step is to make prediction.

**Code:**

# 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))

**Output:**

0.9

[[ 7 0 0]

[ 0 11 1]

[ 0 2 9]]

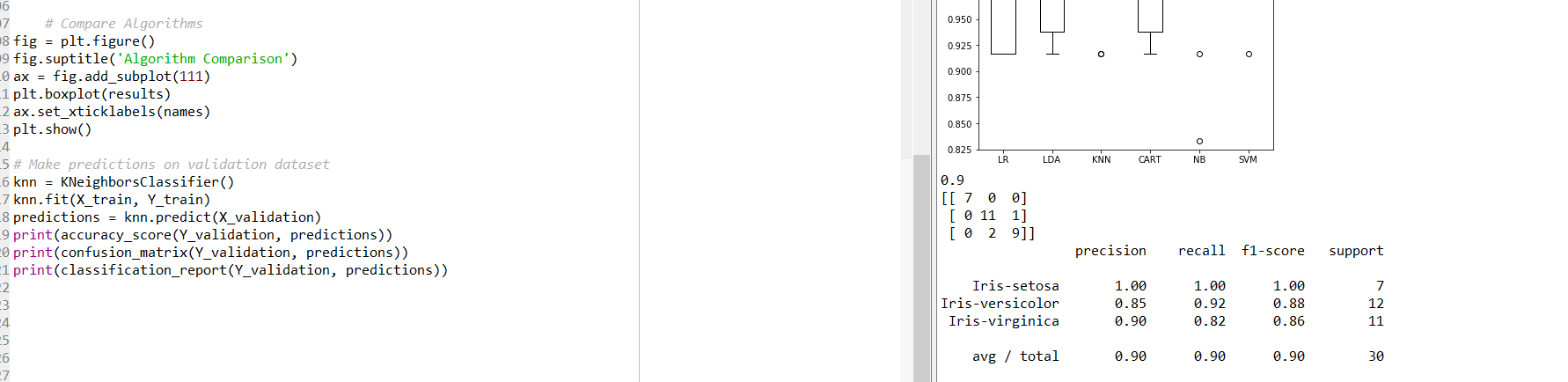
precision recall f1-score support

Iris-setosa 1.00 1.00 1.00 7

Iris-versicolor 0.85 0.92 0.88 12

Iris-virginica 0.90 0.82 0.86 11

avg / total 0.90 0.90 0.90 30



We can see that the accuracy is 0.9 or 90%. The confusion matrix provides an indication of the three errors made

# Python file

