```
#Importación de las librerías de Python, comúnmente utilizadas en
#análisis de datos, aprendizaje automático y visualización.
import sys
print('Python: {}'.format(sys.version))
import scipy
print('scipy: {}'.format(scipy.__version__))
import numpy
print('numpy: {}'.format(numpy.__version__))
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__))
import pandas
print('pandas: {}'.format(pandas.__version__))
import sklearn
print('sklearn: {}'.format(sklearn.__version__))

#Importación de librerias specificas
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 dataset.
path = "C:/Users/aless/Downloads/iris.flower.dataset.txt"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'class']
dataset = pandas.read_csv(path, names=names)

#Analisis exploratorio de datos
print(dataset.shape)
print(dataset.head(20))

#Resumen estadístico.
print(dataset.describe())

#Distribución de clases
print(dataset.groupby('class').size())

#Visualización de datos
#Gráficos univariante
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False,
sharey=False)
```

```
plt.show()
dataset.hist()
plt.show()

#Gráficos multivariantes
scatter_matrix(dataset)
plt.show()

#EVALUAR ALGUNOS ALGORITMOS
#1. Crear un Conjunto de Validación

array = dataset.values
dataset.columns
dataset.index
dataset.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 1
X_train, X_validation, Y_train, Y_validation =
model_selection.train_test_split(X, Y, test_size=validation_size,
random_state=seed, stratify='array-like'

#2. Test de hardness
scoring = 'accuracy'

#3. Creación de los 5 modelos de predicción
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)
      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)

#4. Comparar los algoritmos para elegir el mejor modelo
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
```

```
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()

#Predicciones con el dataset de validación
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))
```