ST0249   
AI & Machine Learning

Practical 4  
Supervised Learning



What you will learn / do in this lab

1. Explore Supervised Learning concepts and applications
2. Conduct Classification experiment using Python
3. Conduct Regression experiment using Python

TAble of  
Contents

1. Overview 1

Introduction to supervised learning 1

Applications of supervised learning 2

2. Classification 4

k-Nearest Neighbor 4

Naive Bayes 5

Logistic Regression 6

CART (Classification Tree) 7

Support Vector Classification (SVC) 8

3. Regression 10

Linear Regression 10

CART (Regression Tree) 11

Support Vector Regression (SVR) 12

# 1. Overview

In this practical we will use python with scikit-learn to apply supervised learning to some typical problems encountered. Scikit-learn comes with estimators that help to solve the classification and regression problems.

## Introduction to supervised learning

The majority of practical machine learning uses supervised learning.

Supervised learning is where you have input/attribute/feature variables (**X**) and an output/target variable (**y**) and you use an algorithm to learn the mapping function (**f**) from the input to the output.

**y** = **f**(**X**)

The goal is to approximate the mapping function (f) so well that when you have new input data (Xnew) that you can predict the output variables (y) for that data.

Supervised learning uses the known examples (also called labelled examples) of (X,y) to learn what is the correct mapping function (f). The known examples all have a known correct label or value for y for every input X. Without these labelled data, we cannot apply supervised learning algorithms.



It is called **supervised learning** because the process of an algorithm learning from the **training** dataset can be thought of as a **teacher** supervising the learning process. We know the **correct answers**, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

## Applications of supervised learning

Supervised learning problems can be further grouped into regression and classification problems.

* **Classification**: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
* **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Some real-life applications:

* Systems Biology – Gene expression microarray data
* Text categorization: spam detection
* Face detection
* Signature recognition
* Customer discovery
* Medicine: Predict if a patient has heart ischemia by a spectral analysis of his/her ECG.

Some popular examples of supervised machine learning algorithms are:

* Linear regression for regression problems.
* Random forest for classification and regression problems.
* Support vector machines for classification problems.

# 2. Classification

In this section, we will look at some “recipes” for applying some common algorithms in scikit-learn to classification problems.

## k-Nearest Neighbor

The k-Nearest Neighbor (kNN) method makes predictions by locating similar cases to a given data instance (using a similarity function) and returning the average or majority of the most similar data instances. The kNN algorithm can be used for **classification** or **regression**.

# k-Nearest Neighbor

from sklearn import datasets

from sklearn import metrics

from sklearn.neighbors import KNeighborsClassifier

# load iris the datasets

dataset = datasets.load\_iris()

# fit a k-nearest neighbor model to the data

model = KNeighborsClassifier()

model.fit(dataset.data, dataset.target)

print(model)

# make predictions

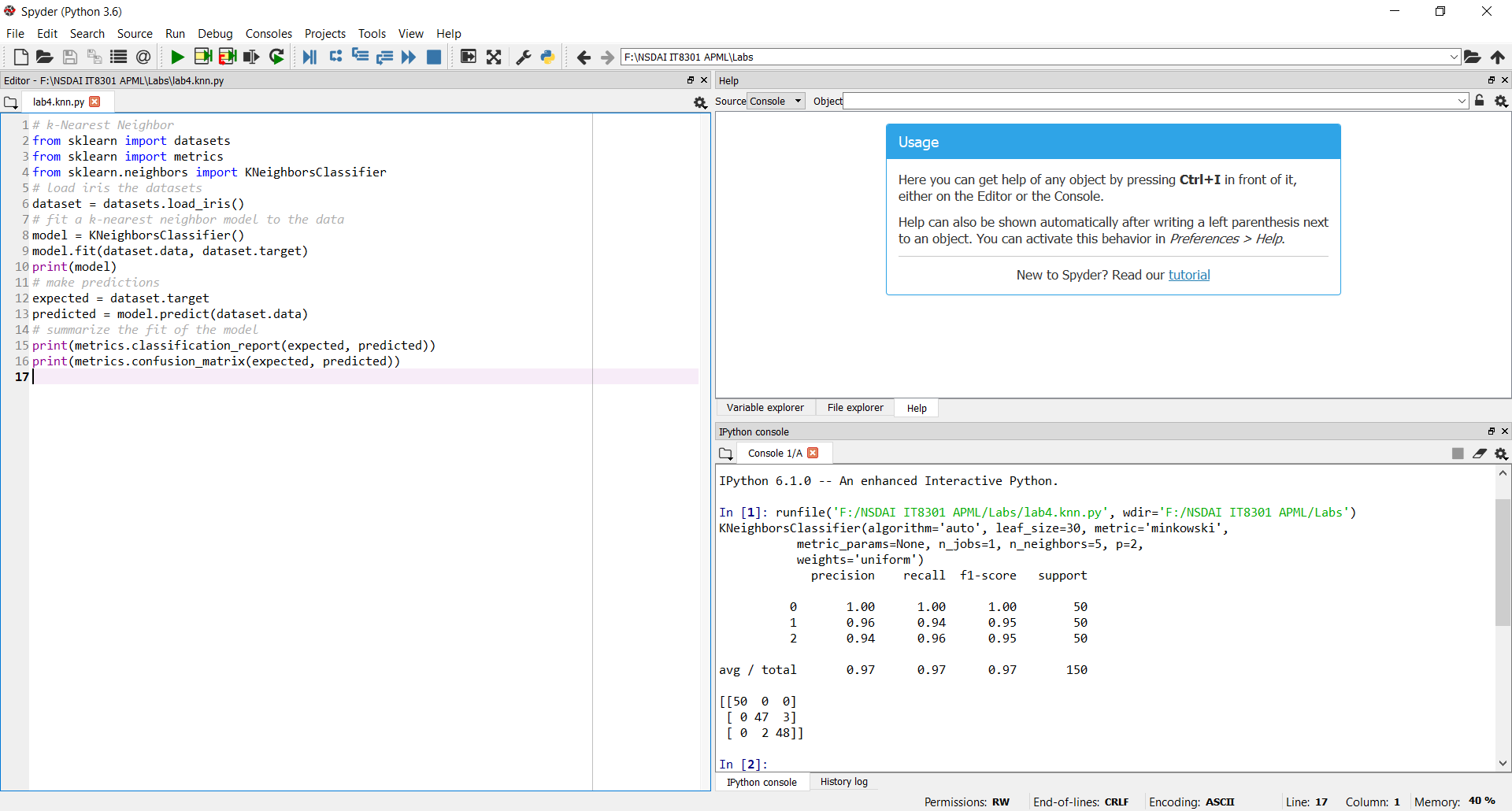
expected = dataset.target

predicted = model.predict(dataset.data)

# summarize the fit of the model

print(metrics.classification\_report(expected, predicted))

print(metrics.confusion\_matrix(expected, predicted))



## Naive Bayes

Naive Bayes uses Bayes Theorem to model the conditional relationship of each attribute to the class variable.

# Gaussian Naive Bayes

from sklearn import datasets

from sklearn import metrics

from sklearn.naive\_bayes import GaussianNB

# load the iris datasets

dataset = datasets.load\_iris()

# fit a Naive Bayes model to the data

model = GaussianNB()

model.fit(dataset.data, dataset.target)

print(model)

# make predictions

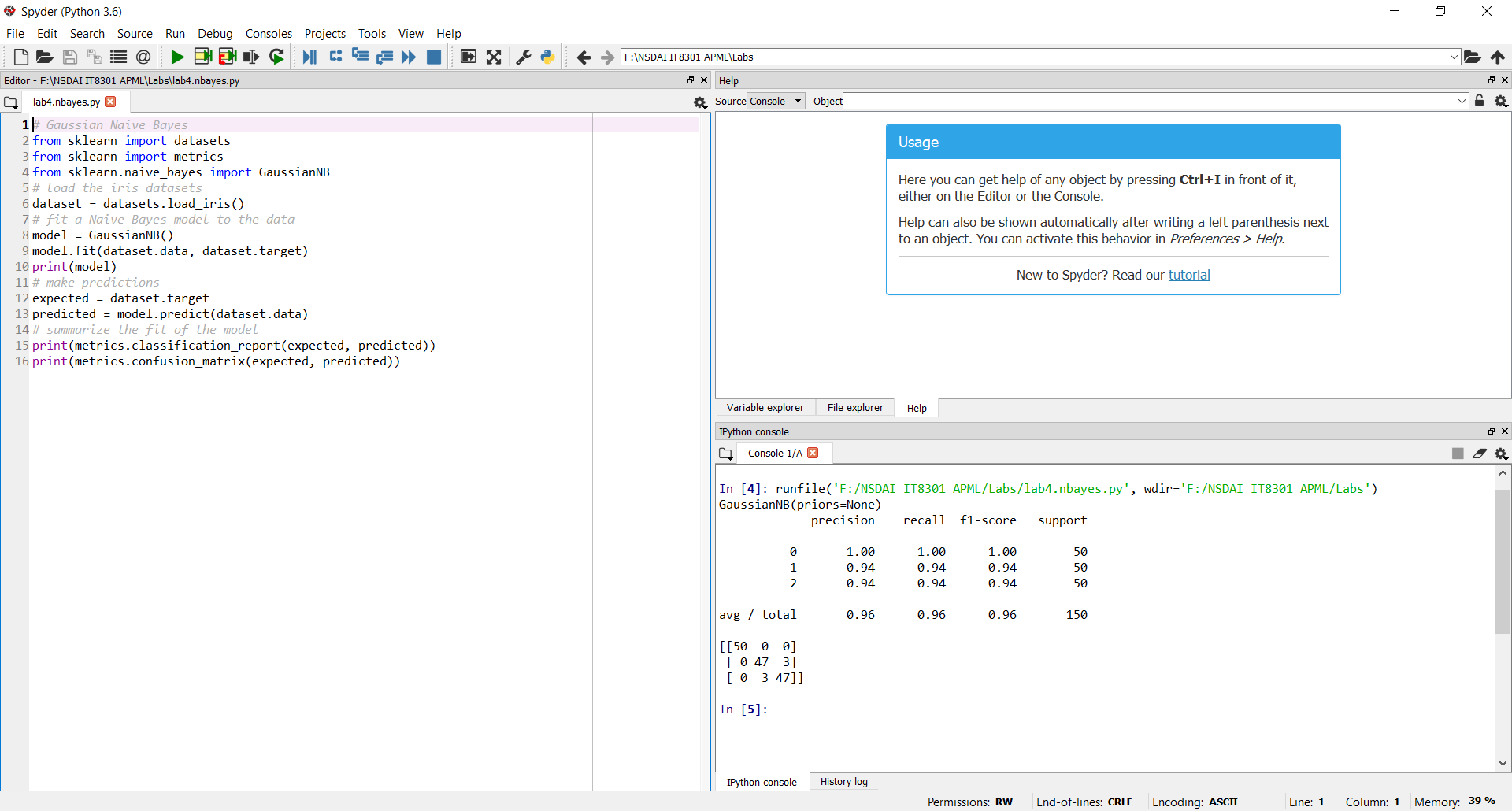
expected = dataset.target

predicted = model.predict(dataset.data)

# summarize the fit of the model

print(metrics.classification\_report(expected, predicted))

print(metrics.confusion\_matrix(expected, predicted))



## Logistic Regression

Logistic regression fits a logistic model to data and makes predictions about the probability of an event (between 0 and 1).

Because this is a multi-class classification problem and logistic regression makes predictions between 0 and 1, a one-vs-all scheme is used (one model per class). (Please note we used the multinomial mode in practical 3).

# Logistic Regression

from sklearn import datasets

from sklearn import metrics

from sklearn.linear\_model import LogisticRegression

# load the iris datasets

dataset = datasets.load\_iris()

# fit a logistic regression model to the data

model = LogisticRegression()

model.fit(dataset.data, dataset.target)

print(model)

# make predictions

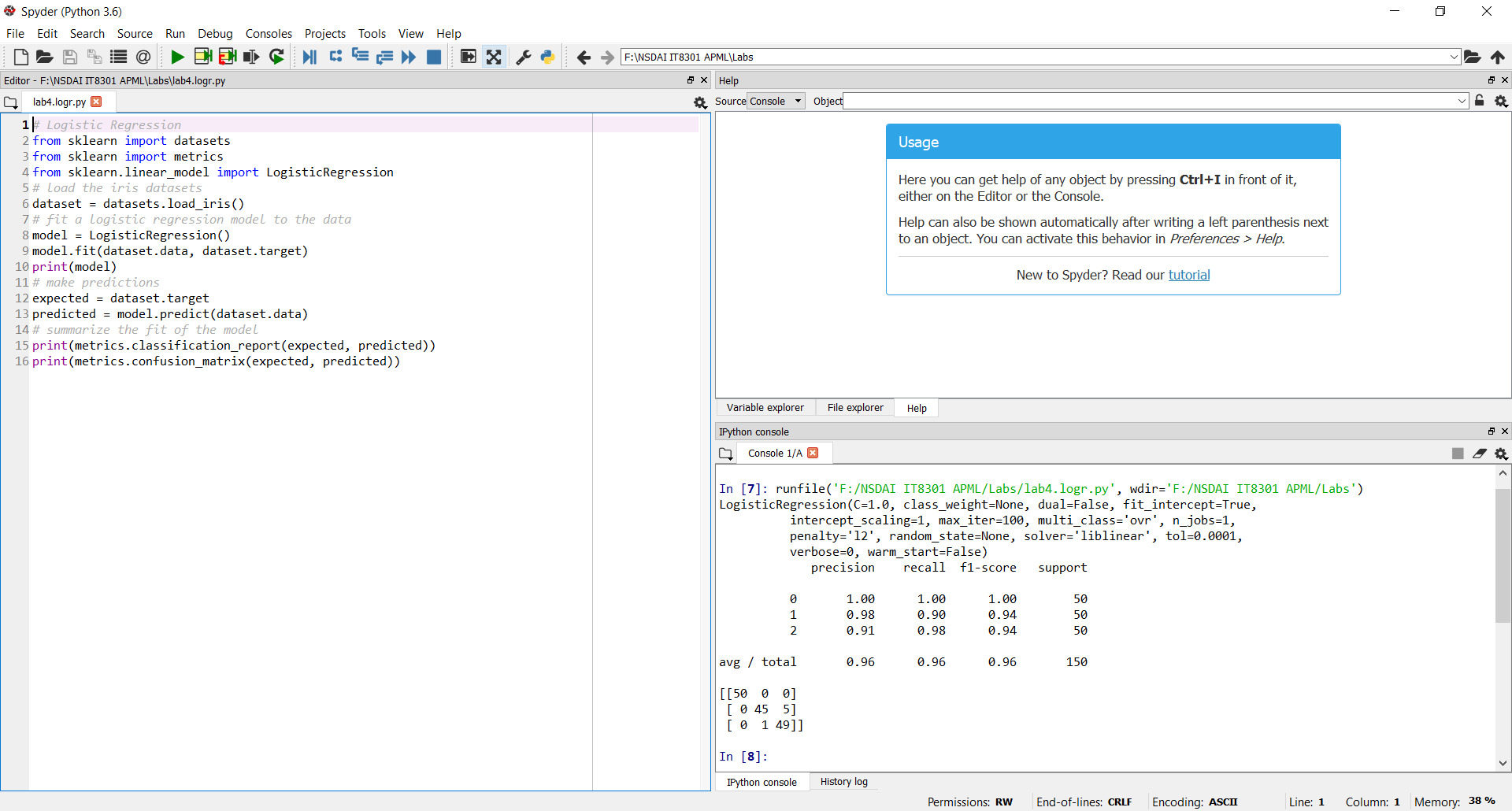
expected = dataset.target

predicted = model.predict(dataset.data)

# summarize the fit of the model

print(metrics.classification\_report(expected, predicted))

print(metrics.confusion\_matrix(expected, predicted))



## CART (Classification Tree)

Classification and Regression Trees (CART) are constructed from a dataset by making splits that best separate the data for the classes or predictions being made. The CART algorithm can be used for classification or regression. Decision Trees used for classification are also know as Classification Trees.

# Decision Tree Classifier

from sklearn import datasets

from sklearn import metrics

from sklearn.tree import DecisionTreeClassifier

# load the iris datasets

dataset = datasets.load\_iris()

# fit a CART model to the data

model = DecisionTreeClassifier()

model.fit(dataset.data, dataset.target)

print(model)

# make predictions

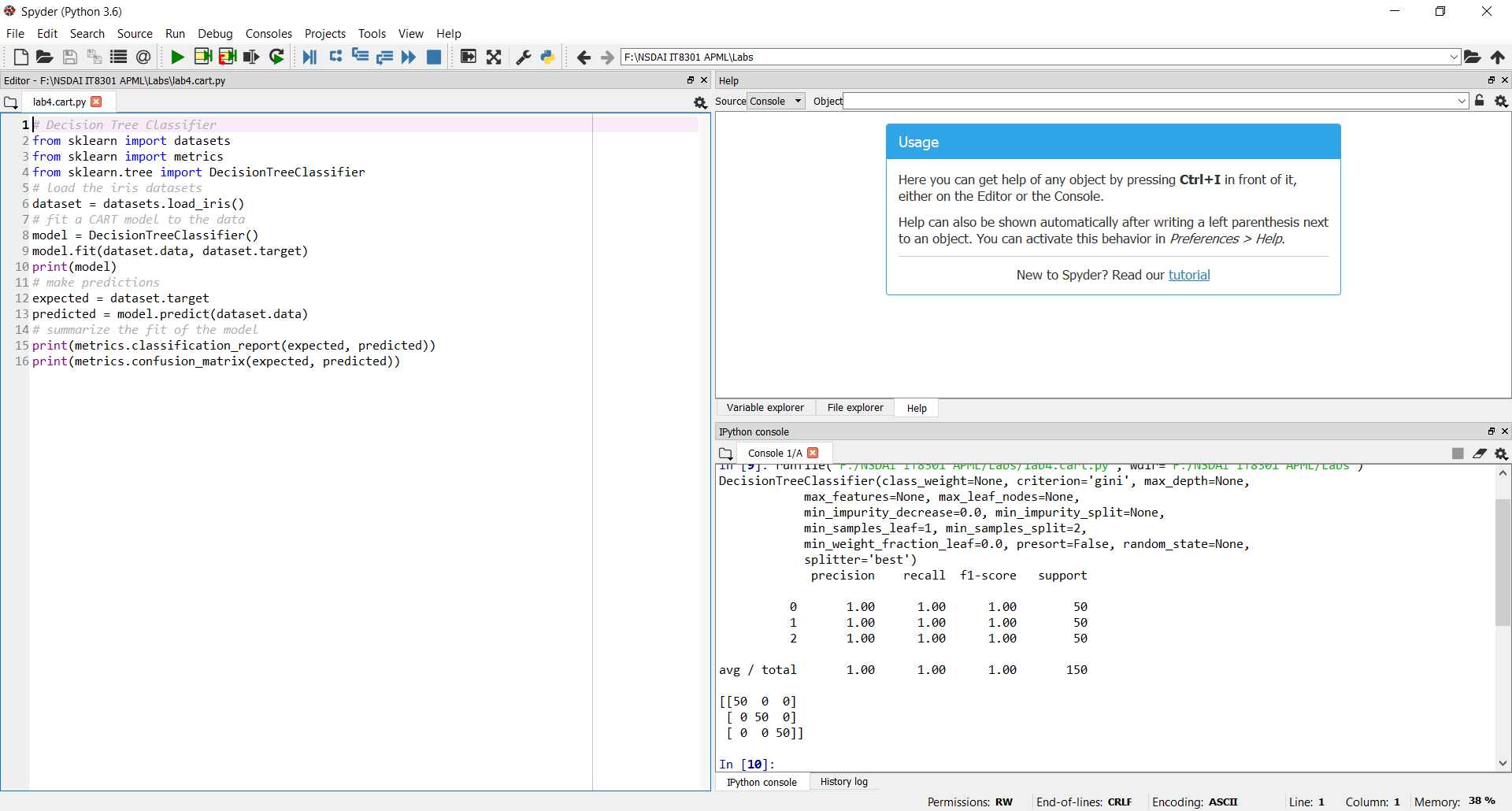
expected = dataset.target

predicted = model.predict(dataset.data)

# summarize the fit of the model

print(metrics.classification\_report(expected, predicted))

print(metrics.confusion\_matrix(expected, predicted))



## Support Vector Classification (SVC)

Support Vector Machines (SVM) are a method that uses points in a transformed problem space that best separate classes into two groups. Classification for multiple classes is supported by a one-vs-all method. Support Vector Classification (SVC) is SVM used for classification problems.

# Support Vector Machine

from sklearn import datasets

from sklearn import metrics

from sklearn.svm import SVC

# load the iris datasets

dataset = datasets.load\_iris()

# fit a SVM model to the data

model = SVC()

model.fit(dataset.data, dataset.target)

print(model)

# make predictions

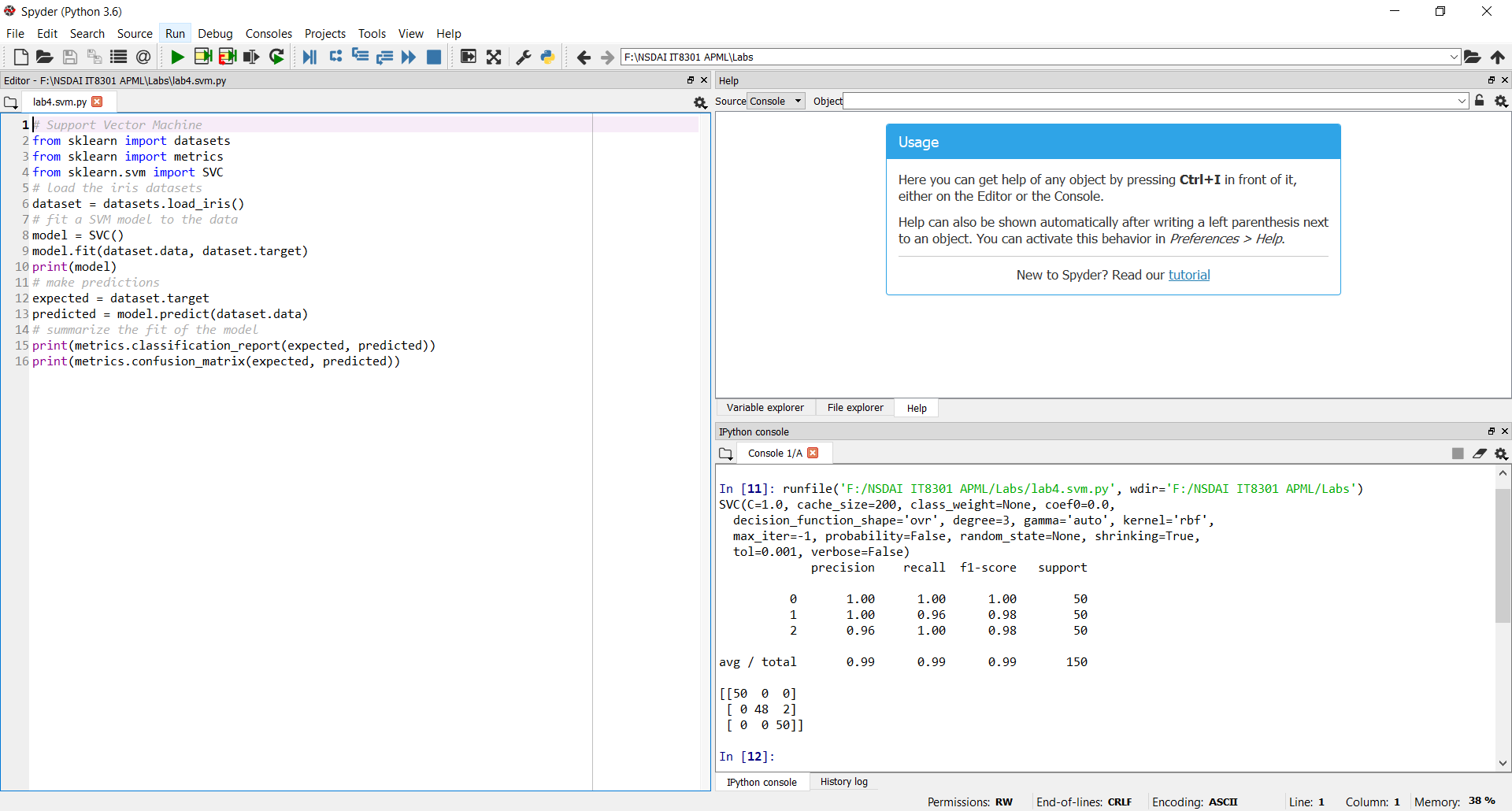
expected = dataset.target

predicted = model.predict(dataset.data)

# summarize the fit of the model

print(metrics.classification\_report(expected, predicted))

print(metrics.confusion\_matrix(expected, predicted))



# 3. Regression

In this section, we will use scikit-learn to apply some common algorithms to regression problems. For the regression examples, we would be using the Boston housing data from scikit-learn

## Linear Regression

Linear regression is a popular method for estimating a target value using the input data. It uses a linear equation of the form:

y = b0 + b1\*x1 + b2\*x2 + …. bn \* xn

# Linear Regression

import numpy as np

from sklearn.datasets import load\_boston

from sklearn.linear\_model import LinearRegression

# load the boston datasets

dataset = load\_boston()

# fit a linear regression model to the data

model = LinearRegression()

model.fit(dataset.data, dataset.target)

print(model)

# make predictions

expected = dataset.target

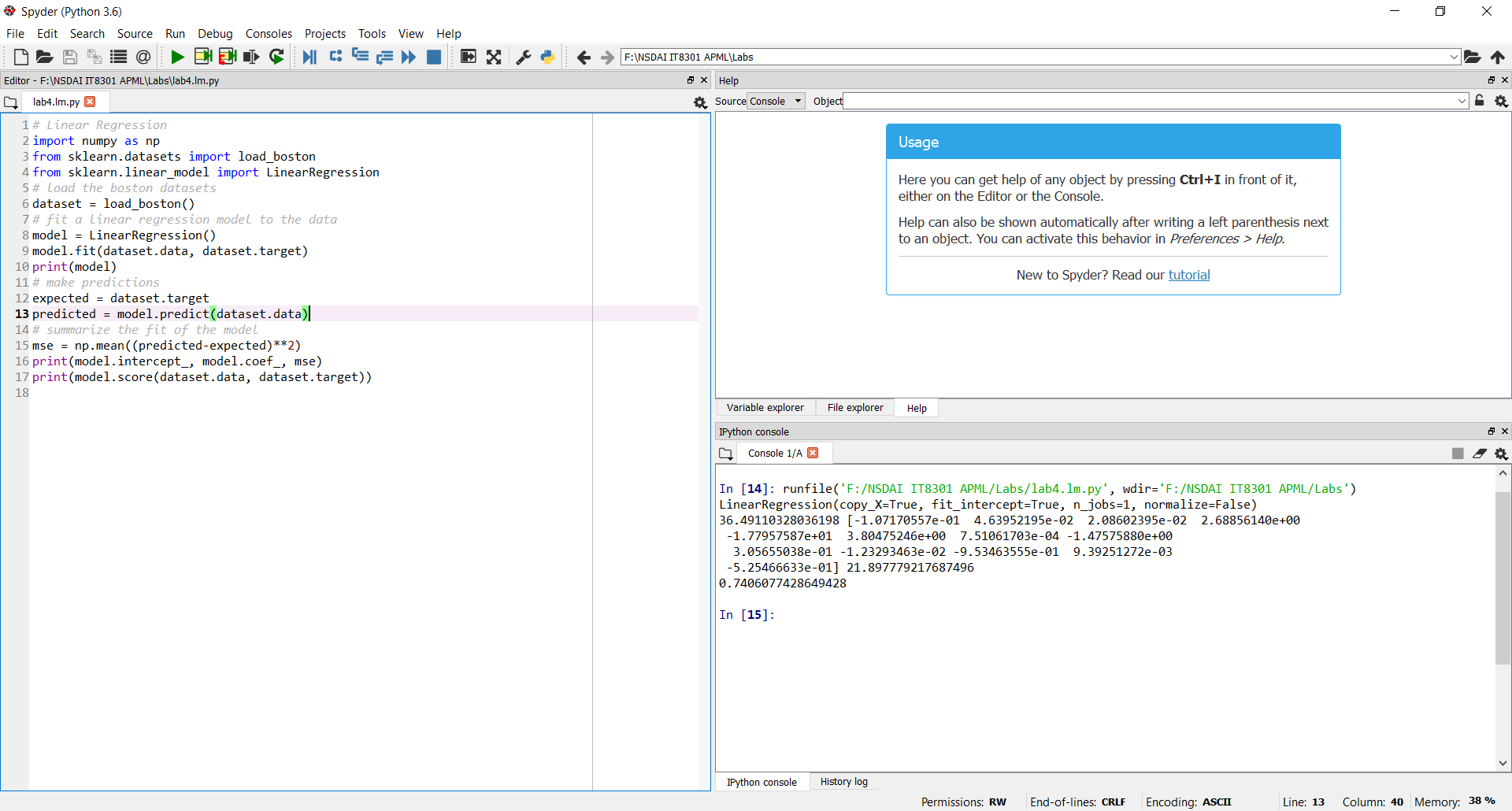
predicted = model.predict(dataset.data)

# summarize the fit of the model

mse = np.mean((predicted-expected)\*\*2)

print(model.intercept\_, model.coef\_, mse)

print(model.score(dataset.data, dataset.target))



## CART (Regression Tree)

CART algorithm can also be used for regression problems. Decision Trees are also known as Regression Trees when used for this purpose.

# CART Regression Tree

import numpy as np

from sklearn.datasets import load\_boston

from sklearn.tree import DecisionTreeRegressor

# load the boston datasets

dataset = load\_boston()

# fit a Regression Tree model to the data

model = DecisionTreeRegressor()

model.fit(dataset.data, dataset.target)

print(model)

# make predictions

expected = dataset.target

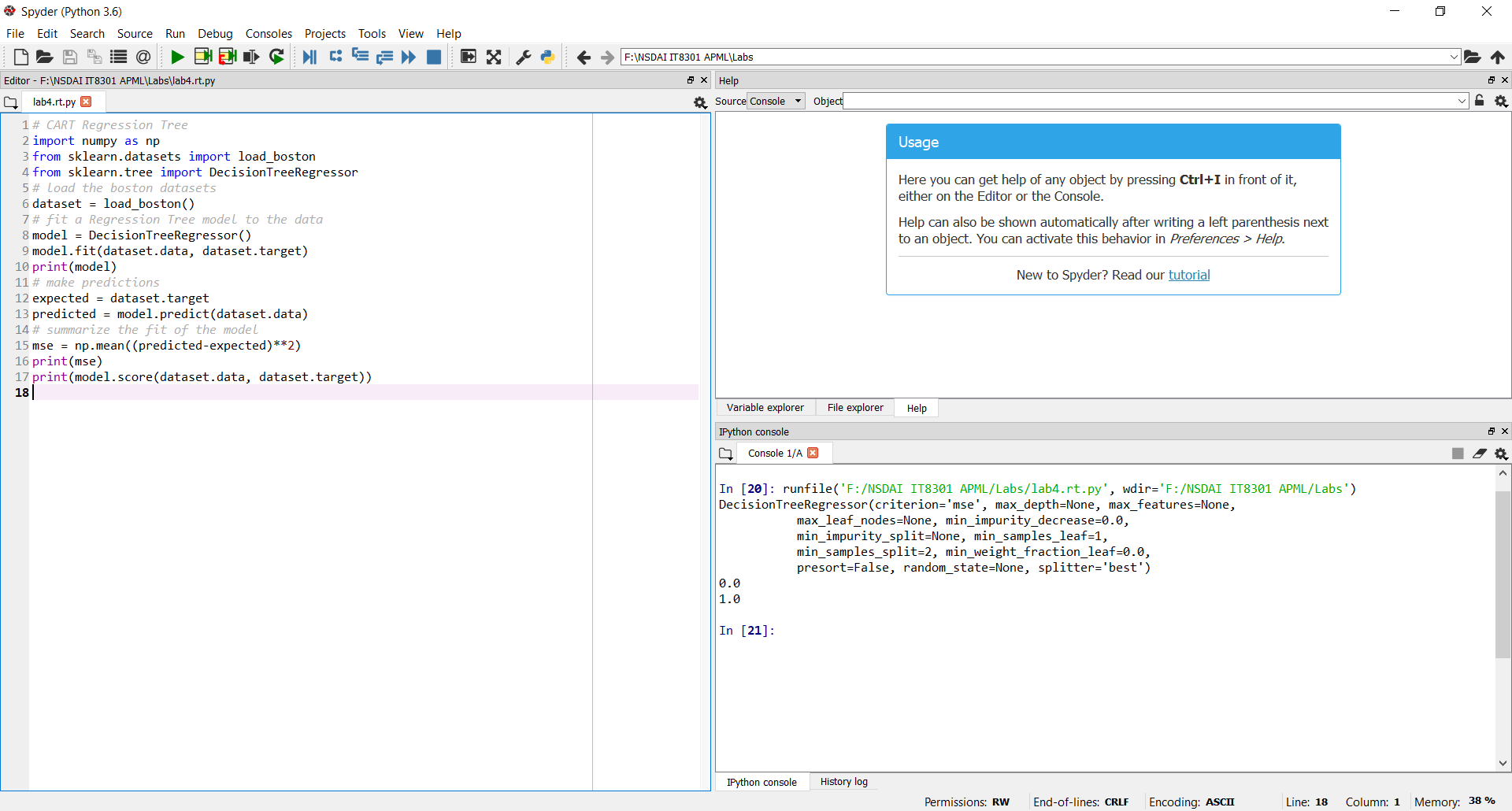
predicted = model.predict(dataset.data)

# summarize the fit of the model

mse = np.mean((predicted-expected)\*\*2)

print(mse)

print(model.score(dataset.data, dataset.target))



## Support Vector Regression (SVR)

SVM algorithm can also be used for regression problems. Using SVM for regression is also known as Support Vector Regression (SVR). SVM also supports regression by modeling the function with a minimum amount of allowable error.

# Support Vector Regression

import numpy as np

from sklearn.datasets import load\_boston

from sklearn.svm import SVR

# load the boston datasets

dataset = load\_boston()

# fit a Support Vector Regression model to the data

model = SVR()

model.fit(dataset.data, dataset.target)

print(model)

# make predictions

expected = dataset.target

predicted = model.predict(dataset.data)

# summarize the fit of the model

mse = np.mean((predicted-expected)\*\*2)

print(mse)

print(model.score(dataset.data, dataset.target))

