Problem Statement: -

A construction firm wants to develop a suburban locality with new infrastructure but they are faced with a challenge of incurring losses if they cannot sell the properties. To overcome this, they consult an analytics firm and would like to get insights on how densely the area is populated and different level of income group people reside. You as a Data Scientist perform Support Vector Machines Algorithm on the given dataset and bring out informative insights and also comment on if its viable for investment in that area.

Solution:

Support Vector Machines

Generally, Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes.

Classifier Building in Scikit-learn

Until now, you have learned about the theoretical background of SVM. Now you will learn about its implementation in Python using scikit-learn.

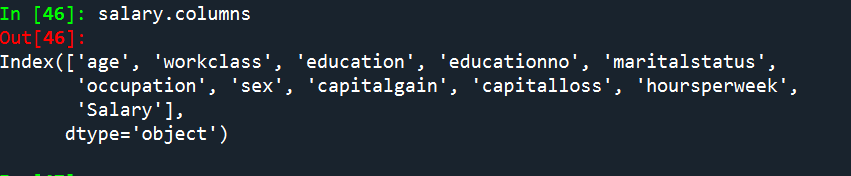
Loading Data

Let's first load the required dataset you will use.

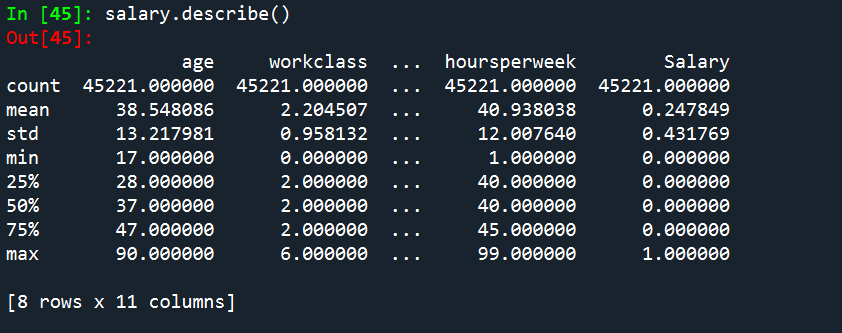
Exploring Data

After you have loaded the dataset, you might want to know a little bit more about it. You can check feature and target names.

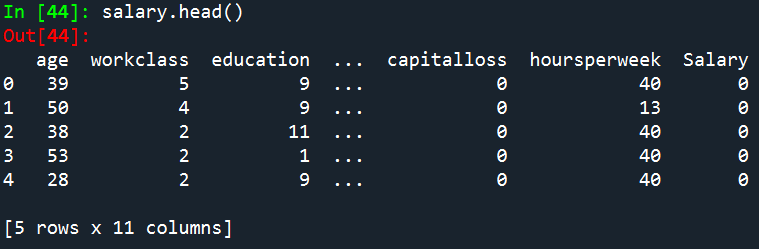
List the columns in the data



Let's explore it for a bit more. You can also check the shape of the dataset using shape.



Let's check top 5 records of the feature set.



Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Split the dataset by using the function train\_test\_split(). you need to pass 3 parameters features, target, and test\_set size. Additionally, you can use random\_state to select records randomly.

# Import train\_test\_split function

from sklearn.model\_selection import train\_test\_split

# Split dataset into training set and test set

train,test= train\_test\_split(salary, , test\_size=0.2,random\_state=109) # 70% training and 20% test

Generating Model

Let's build support vector machine model. First, import the SVM module and create support vector classifier object by passing argument kernel as the linear kernel in SVC() function.

Then, fit your model on train set using fit() and perform prediction on the test set using predict().

#Import svm model

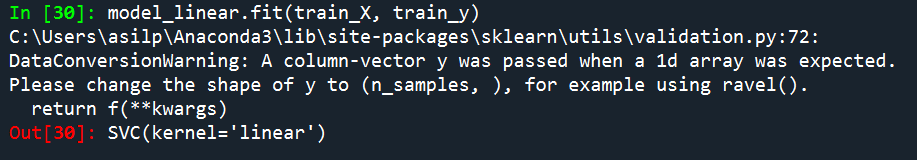
from sklearn import svm

#Create a svm Classifier for linear kernel

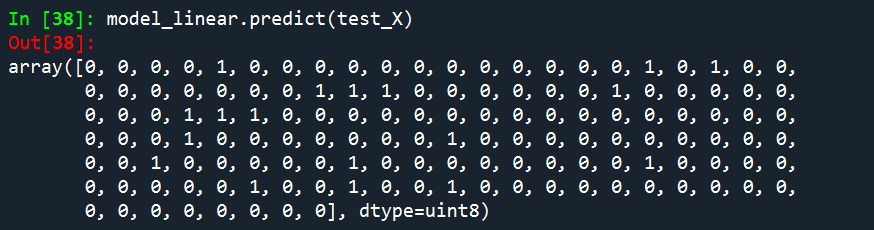
model\_linear = SVC(kernel = "linear")



#Train the model using the training sets



#Predict the response for test dataset



Evaluating the Model

Let's estimate how accurately the classifier or model can predict

Accuracy can be computed by comparing actual test set values and predicted values.

#Import scikit-learn metrics module for accuracy calculation

from sklearn import metrics



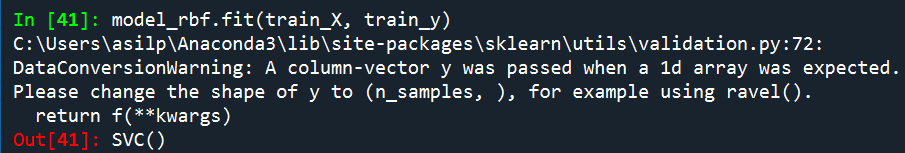
Well, you got a classification rate of 79.28%, considered as very good accuracy.

#Create a svm Classifier for rbf kernel

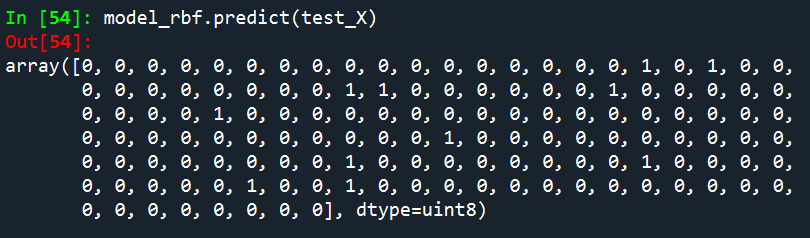
model\_linear = SVC(kernel = "rbf")



#Train the model using the training sets



#Predict the response for test dataset



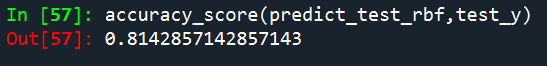
Evaluating the Model

Let's estimate how accurately the classifier or model can predict

Accuracy can be computed by comparing actual test set values and predicted values.

#Import scikit-learn metrics module for accuracy calculation

from sklearn import metrics



Well, you got a classification rate of 81.42%, considered as very good accuracy .