ML_DATA_LIBRARIES OVERVEIW

#import liabraries: libraries is a tool that you can use to make specific job

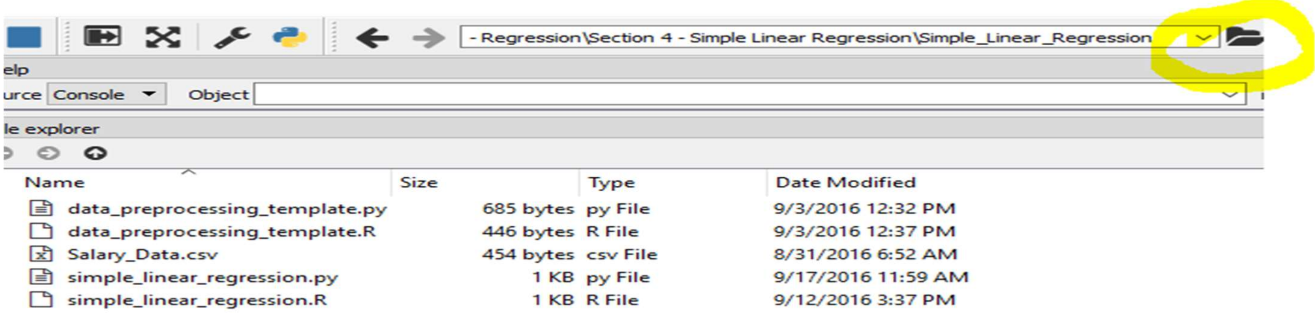
import numpy as np #contains mathemtical tools

import matplotlib.pyplot as plt #pyplot is a sub library in maplotlib use to plot nice graph plots

import pandas as pd #use to import datasets and manage datasets.

#importing datasets

datasets=pd.read_csv('Data.csv')#for datasets you may visit “**kaggle.com**”or an amazon dataset collection**”registry.opendata.aws”,**same google also have dataset collection you can get it from**”toolbox.google.com”, “superdatascience.com”.**Once you download the dataset and if you are working on Spyder then set the location of that dataset in your spyder prompt.



- Regression\Section 4 - Simple Linear Regression\Simple_Linear_Regression

Name	Size	Type	Date Modified
data_preprocessing_template.py	685 bytes	py File	9/3/2016 12:32 PM
data_preprocessing_template.R	446 bytes	R File	9/3/2016 12:37 PM
Salary_Data.csv	454 bytes	csv File	8/31/2016 6:52 AM
simple_linear_regression.py	1 KB	py File	9/17/2016 11:59 AM
simple_linear_regression.R	1 KB	R File	9/12/2016 3:37 PM

X=datasets.iloc[:,:-1].values

Y=datasets.iloc[:,3].values

#taking care of data :-> if some data is missing in dataset then we are going to find the mean and we are going to use library for this purpose known as scikit_learn preprocessing and from this library import imputer class

from sklearn.preprocessing import Imputer

#SK_Learn is sciket learn and it contains amazing libraries to make machinery models and from scyket learn we import preprocessing liabraries that contains a lot of class methods to pre-processing any data sets and from this liabraries we are importing the imputer class which allow us to take care of missing data

imputer = Imputer(missing_values='NaN',strategy='mean',axis=0)

imputer=imputer.fit(X[:,1:3])

X[:,1:3]=imputer.transform(X[:,1:3])

#transform is the method that is use to replace the missing data with the mean of the data

#Encoding categorial data:->As machine learning algorithms based on equations.Categorial data are the data in the form of data and we have two categorials data here 'country' and 'purchased' in 'country' we have three categories 'france','spain' and 'germany' and in 'price' we have two categories 'yes' and 'no'.As in machine learning model we need an equation and as we know categories here make a problem in equation so we need to encode that categroy into numerals.So we again need sckit_learn library and going to import labelEncoder and OneHotEncoder

from sklearn.preprocessing import LabelEncoder,OneHotEncoder

labelencoder_X=LabelEncoder()#labelencoer_X is an object of the class. LabelEncoder,labelencoder will only decode the value without bothering if their is any order or not.

X[:,0]=labelencoder_X.fit_transform(X[:,0])

#as labelencoder encodede 'France=0','spain=2'and 'germany=1' which can also be mistreated as labeling categories on the basis of size i.e;spain is greater than france and germany and france is grater than germany.So to overcome this we are going to use Dummy Variable i.e; instead of having one variable we are going to have three variables i.e; number of variables columns will be equals to number of categories in categorial data,and in each column we have either '1' and '0', by using of one more library of scikit_learn known as OneHotEncoder.OneHotEncoder will will divide the categories inn to three columns each having '1' and '0'.

onehotencoder=OneHotEncoder(categorical_features=[0])#in categorical_features we are going to use that column on which we have onehotencoder.

X=onehotencoder.fit_transform(X).toarray()

labelencoder_Y=LabelEncoder()

Y=labelencoder_Y.fit_transform(Y)

#spliting the dataset into training_set and test_set

from sklearn.cross_validation import train_test_split

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=0)

#Feature Scaling:- By uisng Feature scaling we use to scale the variables in same range(-1,+1) and generally machine learning algorithm depends upon eculidian distance that's why no one can dominate each other.We can do this by using two methods "standardisation" and "normalisation".In standardisation we generally find the x(stand.)=(x-x(mean))/standard deviation(x) where as in normalisation we generally find the x(norm) by using formulae x(norm)=(x-min(x))/max(x)-min(x).So that no variable should dominated by another.

from sklearn.preprocessing import StandardScaler

```
sc_x=StandardScaler
X_train=sc_x.fit_transform(X_train)
X_test=sc_x.transform(X_test)
```

#Not all time machine learning algorithm depends upon Eculidian Distance but we do feature scaling because the algorithm will converse faster.That will be the case for decision tree,decisions tree doesnot depend upon eculidean distance but we still need feature scaling if we dont do it they will run for a very long time.

#as in classification problem our dependent variable(Y) contains categorial values so we dont need feature scaling for Y.But we need feature sclaing for Y in regression problem because theirdependent variable Y have a very high range of values.