**Ineuron assignment 2**

Question-1:  Explain three- dimensional data indexing?

Ans:  Three dimensional databases are growing both in number and size around the web due to the rapid development of tools for acquisition and storage of 3D objects. Therefore, the navigation through those databases to find desirable objects is a major problem. Content based indexing is an important way to manage those large databases. Many search engines for 3D objects are available on the web. These search engines give users the possibility of navigating through the databases to visualize models in 3D space, using web navigators, and to search by visual similarity similar models for a given 3D model query. The most popular web based search engines where the links are given in references are the Princeton University search engine, the ‘CCCC’ Konstanz University search engine, the Fox MIIRE 3D search engine, the Ogden 3D search engine, and Informatics and Telematics Institute 3D search engine.  Since 1997, authors have proposed algorithms to describe 3D objects. These algorithms are categorized, in general, taking into account their representation of 3D objects. Hilaga et al. (2001) used the Reeb graph based descriptor that represents an object by a structure that captures important information about the structure of model. The authors used the geodesic distance to compute distance between graphs. Sundar et al. (2003) proposed a skeleton graph based descriptor. The authors used a thinning algorithm proposed by Gagvani & Silver (1999) on the voxelization of a solid object.  In order to extract similarity between 3D objects, they compared the corresponding skeleton graphs of the objects. Hekzko et al. (2002) proposed using images based 3D descriptors. The authors generated images obtained from orthogonal projections of the object, and applied 2D shape descriptors to these images in order to compute the feature vectors for 3D objects. Chen et al (2003) and Ansari et al. (2007) extracted several views as silhouettes from the object and apply 2D shape descriptors to generate feature vectors. We note that the image based descriptors and view based descriptors are similar. However, the difference between them is that the image based approach requires the pose normalization using the CPCA; continuous principal component analysis (Vranic et al. 2001) and the views based approach do not require the normalization of 3D objects.   The voxel based approach was used by Vranic et al. (2001). This method was based on a volumetric representation of 3D objects presented in frequency domain by applying a 2D discrete Fourier transform. The authors chose the lower frequencies as components of feature vectors of 3D objects because the high frequencies are affected by noise. Kazhdan et al. (2002) proposed to apply spherical harmonics to the voxelized model so as to generate the feature vectors. This descriptor obtained in frequency domain used the rotational invariance propriety of Spherical Data Science Journal, Volume 8, 20 May 200978Harmonics. Osada et al. (2002) proposed a statistical method named D2 (shape distribution) to describe 3D objects. Vranic et al. (2001b) proposed a descriptor named Ray with Spherical Harmonics to present the extents from the centre of mass of an object to its surface in spectral domain as components of feature vectors for 3D objects.  In this paper, we present a new approach to extract similarity between 3D objects. This approach is based on a 3D closed curve that represents the 3D object. In order to extract feature vectors for our 3D model, we firstly apply CPCA (Continuous Principal Component Analysis) as pose normalization in order to align the model into canonical position. Secondly, we extract a 3D closed curve that represents this object. Finally, we extract three signatures of this curve, which are combined in a descriptor named Enhanced Curve Analysis Descriptor (ECA). We present the design of our web based search engine where we implement the proposed descriptor. We evaluate our method using the Princeton Shape Benchmark Database (Shilan et al., 2004) using measures widely used in the information retrieval community. We end with our conclusion.

Question -2: what's the difference between series and dataframe?

Ans: Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the index. The basic method to create a Series is to call:

s = pd.Series(data, index=index)

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects.

d = {'one' : pd.Series([1., 2., 3.], index=['a', 'b', 'c']),

two' : pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)

Difference between series and dataframe:

Series is a type of list which can take integer values, string values, double value and more. Series can only contain single list with index, whereas dataframes can be made of more than one series or we can say that a dataframes is a collection of series that can be used to analyse the data.

Question-3: what role pandas play in data cleaning?

Ans:  Data Science is the discipline of study which involves extracting insights from huge amounts of data by the use of various scientific methods, algorithms, and processes. To extract useful knowledge from data, Data Scientists need raw data. This Raw data is a collection of information from various outlines sources and an essential raw material of Data Scientists. It is additionally known as primary or source data. It consists of garbage, irregular and inconsistent values which lead to many difficulties. When using data, the insights and analysis extracted are only as good as the data we are using. Essentially, when garbage data is in, then garbage analysis comes out. Here Data cleaning comes into the picture, Data cleansing is an essential part of data science. Data cleaning is the process of removing incorrect, corrupted, garbage, incorrectly formatted, duplicate, or incomplete data within a dataset.

When working with multiple data sources, there are many chances for data to be incorrect, duplicated, or mislabeled. If data is wrong, outcomes and algorithms are unreliable, even though they may look correct. Data cleaning is the process of changing or eliminating garbage, incorrect, duplicate, corrupted, or incomplete data in a dataset. There’s no such absolute way to describe the precise steps in the data cleaning process because the processes may vary from dataset to dataset. Data cleansing, data cleansing, or data scrub is that the initiative among the general data preparation process. Data cleaning plays an important part in developing reliable answers and within the analytical process and is observed to be a basic feature of the info science basics.Data cleaning is the most important task that should be done as a data science professional. Having wrong or bad quality data can be detrimental to processes and analysis. Having clean data will ultimately increase overall productivity and permit the very best quality information in your decision-making.

Question-4: How do you use pandas to make a data frame out of n-dimensional arrays?

Ans:  we are going to learn how to convert a NumPy array to a DataFrame object. Now, you may already know that it is possible to create a dataframe in a range of different ways. For example, it is possible to create a Pandas dataframe from a dictionary.

As Pandas dataframe objects already are 2-dimensional data structures, it is of course quite easy to create a dataframe from a 2-dimensional array. Much like when converting a dictionary, to convert a NumPy array we use the pd.DataFrame() constructor:

how to convert numpy array to pandas dataframe code example

In the next two sections, you will learn about the NumPy array and Pandas dataframe. After that, you will get the answer to the question “How do you convert an array to a DataFrame in Python?”  Here will see an example of the simplest way to create a dataframe from an array. In the next section, on the other hand, we will get into more details about the syntax of the dataframe constructor. Finally, we will look at a couple of examples of converting NumPy arrays to dataframes. In these, last, sections you will see how to name the columns, make an index, and such.

NumPy Array

Multidimensional arrays are a means of storing values in several dimensions. For example, an array in two dimensions can be likened to a matrix and an array in three dimensions can be likened to a cube. In Python, multidimensional arrays are usually created using the NumPy library.

A 2-d array that can be converted to a dataframe

Storing data in this way can make it easier to organize large amounts of data in a structure that is easier to work with. A NumPy array in two dimensions can be likened to a grid, where each box contains a value. See the image above. If you need to, it is also possible to convert an array to integer in Python. That is, if your NumPy array contains float numbers and you want to change the data type to integer.

a pandas dataframe created from a numpy array

For most purposes, your observations (customers, patients, etc) make up the rows and columns describing the observations (e.g., variables such as age, gender, income, health status). A Pandas dataframe is simply a two-dimensional table. As you may know, there are plenty of ways to create a dataframe. Most of the time, we import our data from a file. For example, we can read a CSV file to a Pandas dataframe or reading the data from Excel files.

Now that we have an idea of what NumPy arrays and Pandas dataframes are, it may be obvious that converting one to the other is something that is very easy to do.

Now, you may also need to go the other way arround. That is, you man need to convert your Pandas dataframe to a NumPy array.

Pandas DataFrame() Constructor Syntax

In this section, we will have a look at the syntax, as well as the parameters, of the DataFrame() constructor. As you may be aware, right now, this is the method we will use to create a dataframe from a NumPy array. Typically we import Pandas as pd and then we can use the DataFrame() method. Here’s the syntax of the construction:

import pandas as pd

# Create the dataframe

df = pd.DataFrame(numpy\_array)

Code language: PHP (php)

In the code above, we have the array (numpy\_array). Second, we use the DataFrame class and here we only use the data parameter (i.e., our NumPy array). The resulting dataframe will look like this:

The resulting pandas dataframe that was converted from numpy array

Resulting dataframe without good column names

As you can see, if we’re not using the columns parameter, we will get numbers as column names (see the previous section for the parameters). Often, this is not a result to strive for as the later data analysis may be a bit hard to carry out if we don’t know what the different numbers are reflecting in terms of variables. Now, one option is to rename the columns in the Pandas dataframe or to set the names when creating the dataframe. If you need to know, you can list column names using Pandas columns method.

In the next example, we will have a look at transforming the NumPy array to a dataframe using the columns parameter.

df = pd.DataFrame(numpy\_array,

                 index=['day1', 'day2', 'day3', 'day4'],

                 columns=['digits', 'words'])

Code language: JavaScript (javascript)

Notice how we used the index parameter and used a list as the indexes.

Question-5: Explain the notion of pandas plotting.

Ans: Whether you’re just getting to know a dataset or preparing to publish your findings, visualization is an essential tool. Python’s popular data analysis library, pandas, provides several different options for visualizing your data with .plot(). Even if you’re at the beginning of your pandas journey, you’ll soon be creating basic plots that will yield valuable insights into your data.

Basic plotting: In this basic plot we can use the randomly generated data to plot graph using Series.

Here is what we’ll cover:

Importing the appropriate libraries

Getting data about the weather in London

Produce a first Pandas visualization using the plot() method

Find out how different types of charts are created

Plotting simple charts: line charts, bar charts, pie charts and scatter diagrams

Plotting statistical Pandas charts— spotting unusual events

Box Plots — Showing the range of data

Histograms

Changing the number of bins to focus on the outliers — Just how often is it really, really wet?

Pandas plot utilities — multiple plots and saving images.