Ineuron

Assignment-2

1.Explain three-dimensional data indexing?

Ans:- INTRODUCTION :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.

2.whats the difference between a series and a dataframe?
Ans:-Series is a type of list in pandas which can take integer values, string values, double values and more. ... Series can only contain single list with index, whereas dataframe can be made of more than one series or we can say that a dataframe is a collection of series that can be used to analyse the data.

3.what role pandas play in data cleaning?
Ans:-Introduction:-Over time companies produce and collect a massive amount of data, depending on the company this can come in many different forms such as user-generated content, job applicant data, blog posts, sensor data and…
Data scientists spend a huge amount of time cleaning datasets and getting them in the form in which they can work. It is an essential skill of Data Scientists to be able to work with messy data, missing values, inconsistent, noise, or nonsensical data. To work smoothly python provides a built-in module Pandas. Pandas is the popular Python library that is mainly used for data processing purposes like cleaning, manipulation, and analysis. Pandas stand for “Python Data Analysis Library”. It consists of classes to read, process, and write CSV data files. There are numerous Data cleaning tools present but, the Pandas library provides a really fast and efficient way to manage and explore data. It does that by providing us with Series and DataFrames, which help us not only to represent data efficiently but also manipulate it in various ways.
In this article, we will use the Pandas module to clean our dataset.
We are using a simple dataset for data cleaning i.e. iris species dataset. You can download this dataset from kaggle.com.
Let’s get started with data cleaning step by step.
To start working with Pandas we need to import it. We are using Google Colab as IDE, so we will import Pandas in Google Colab.

4.How do you use pandas to make a data frame out of n-dimensional arrays?
Ans:-To convert an array to a dataframe with Python you need to 1) have your NumPy array (e.g., np_array), and 2) use the pd.DataFrame() constructor like this: df = pd.DataFrame(np_array, columns=[‘Column1’, ‘Column2’]). Remember, that each column in your NumPy array needs to be named with columns. If you use this parameter, that is.
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 constructor:As you can see, in the image above, there’s one required parameter (the first one): data. Now, this is where we will put the NumPy array that we want to convert to a dataframe. Note, that if your data is stored in a Python dictionary, for instance, it is also possible to use this as input here. The other parameters of the DataFrame class are as follows:

index : Index or array-like
Index to use for the resulting dataframe. If we don't use this parameter, it will default to RangeIndex.
columns : Index or array-like
Column labels to use for the resulting dataframe. Again, if we don't use this parameter it will default to RangeIndex (0, 1, 2, …, n).
dtype : dtype, default None
If we want data to be of a certain data type, dtype is the parameter to use. Only a single dtype is allowed.
copy : boolean, default False
Will make a copy of data from inputs.

5.Explain the notion of pandas plotting?
Ans:-Introduction:-The popular Pandas data analysis and manipulation tool provides plotting functions on its DataFrame and Series objects, which have historically produced matplotlib plots. Since version 0.25, Pandas has provided a mechanism to use different backends, and as of version 4.8 of plotly, you can now use a Plotly Express-powered backend for Pandas plotting. This means you can now produce interactive plots directly from a data frame, without even needing to import Plotly.
To activate this backend, you will need to have Plotly installed, and then just need to set pd.options.plotting.backend to "plotly" and call .plot() to get a plotly.graph_objects.Figure object back, just like if you had called Plotly Express directly:

```
import pandas as pd
pd.options.plotting.backend = "plotly"
df = pd.DataFrame(dict(a=[1,3,2], b=[3,2,1]))
fig = df.plot()
fig.show()
```

This functionality wraps Plotly Express and so you can use any of the styling options available to Plotly Express methods. Since what you get back is a regular Figure object, you can use any of the update mechanisms supported by these objects to apply templates or further customize axes, colors, legends, fonts, hover labels etc. Faceting is also supported.

```
import pandas as pd
pd.options.plotting.backend = "plotly"
df = pd.DataFrame(dict(a=[1,3,2], b=[3,2,1]))
fig = df.plot(title="Pandas Backend Example", template="simple_white",
        labels=dict(index="time", value="money", variable="option"))
        fig.update_yaxes(tickprefix="$")
        fig.show()
```