**DAILY ASSESSMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **22/05/2020** | **Name:** | **Chandana.R** |
| **Course:** | PYTHON | **USN:** | **4AL16EC017** |
| **Topic:** | Data analysis with Pandas ,create  webmaps with python and folium | **Semester & Section:** | **8(A)** |
| **Github Repository:** | **Chandana-shaiva** |  |  |

|  |
| --- |
| **AFTERNOON SESSION DETAILS** |
| **Image of session** |
| **REPORT** |
| 1. **Pandas**   Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or Python.   * Pandas.DataFrame * Syntax: “ Class pandas.DataFrame(data=None, index:Optional[Collection]=None, columns:Optional[Collection] =None, dtype: Union[str, numpy.dtype, ExtensionDtype, None] = None, copy: bool = False). * We can analyze panda by : * SERIES * DATAFRAME * Series: Series is one dimensional(1-D) array defined in pandas that can be used to store any data type.   # Program to Create series with scalar values  Data =[1, 3, 4, 5, 6, 2, 9] # Numeric data  # Creating series with default index values  s = pd.Series(Data)  # predefined index values  Index =['a', 'b', 'c', 'd', 'e', 'f', 'g']  # Creating series with predefined index values  si = pd.Series(Data, Index)   * The main data structures in Pandas are implemented with Series and Data Frame classes. * The former is a one-dimensional indexed array of some fixed data type. * The latter is a two-dimensional data structure - a table - where each column contains data of the same type. * You can see it as a dictionary of Series instances. * DataFrames are great for representing real data: rows correspond to instances   (examples, observations, etc.), and columns correspond to features of these instances.  In [1]:  import numpy as np  import pandas as pd  pd.set\_option("display.precision",   * Understand the basic Pandas data structures Pandas has two core data structures used to store data: The Series and the DataFrame. * Series The series is a one-dimensional array-like structure designed to hold a single array (or ‘column’) of data and an associated array of data labels, called an index. * We can create a series to experiment with by simply passing a list of data, let’s use numbers in this example: * Copy contents import pandas as pd   my\_series = pd.Series([4.6, 2.1, -4.0, 3.0])  print(my\_series)     * DataFrames :The DataFrame represents tabular data, a bit like a spreadsheet. * DataFrames are organized into colums (each of which is a Series), and each column can store a single data-type, such as floating point numbers, strings, boolean values * DataFrames can be indexed by either their row or column names. (They are similar in many ways to R’s data.frame.) * We can create a DataFrame in Pandas from a Python dictionary, or by loading in a text file containing tabular data. First we are going to look at how to create one from a dictionary.   Setup  Let’s create a pandas DataFrame with 5 columns and 1000 rows:  • a1 and a2 have random samples drawn from a normal (Gaussian) distribution,  • a3 has randomly distributed integers from a set of (0, 1, 2, 3, 4),  • y1 has numbers spaced evenly on a log scale from 0 to 1,  • y2 has randomly distributed integers from a set of (0, 1).  mu1,sigma1=0,0.1  mu2,sigma2=0.2,0.2  n=1000df=pd.DataFrame(  {  "a1":pd.np.random.normal(mu1,sigma1,n),  "a2":pd.np.random.normal(mu2,sigma2,n),  "a3":pd.np.random.randint(0,5,n),  "y1":pd.np.logspace(0,1,num=n),  "y2":pd.np.random.randint(0,2,n),  }  )   * Readers with Machine Learning background will recognize the notation where a1, a2 and a represent attributes and y1 and y2 represent target variables. * In short, Machine Learning algorithms try to find patterns in the attributes and use them to predict the unseen target variable — but this is not the main focus of this blog post. * The reason that we have two target variables (y1 and y2) in the DataFrame (one binary and one continuous) is to make examples easier to follow. * We reset the index, which adds the index column to the DataFrame to enumerates the rows. df.reset\_index (inplace=True)   Creating Web Maps in Python Using Folium   * Folium: * It is a Python package built to bridge the data wrangling muscle of Python with Leaflet’s easy-to-use JavaScript library for creating attractive, interactive web maps. * The open source Leaflet is a highly popular web mapping tool due to its flexibility, with a healthy number of community-developed plug-ins further expanding its native capabilities. * However, it was obvious that there is more to explore with Folium, as it plays well with many types of geospatial data, includes built-in functions and methods for producing choropleths, temporal visualizations, and allows for the marriage of the best of Python and Leaflet. * The code and resulting maps show a straightforward exercise in extracting the geographic coordinates (already matching Leaflet’s default web-mercator projection) * few attribute values corresponding to warehouse/distribution centers in Pennsylvania’s Lehigh Valley from an excel spreadsheet. * The Pandas library was used to read the excel document and convert the desired information to a dataframe. * Folium was used to initialize a Leaflet map, add records as points with some stylization applied. This is brief code that could easily be added at the end of a more intensive spatial analysis using Python * . It can provide a quick way to publish results in an interactive format without necessitating the use of JavaScript/html/CSS, or could serve as a jump start on more elaborate styling |