DV Assignment 2

1.a) Explain how Watson Studio will provide the environment to solve Business problems.

Ans. 1.a)

Watson Studio offers a robust environment for solving business problems through data-driven insights. It provides tools for data preparation, analysis, and visualization, facilitating a streamlined workflow. The collaborative features allow teams to work seamlessly, enhancing productivity and decision-making. Additionally, Watson Studio supports various machine learning frameworks, empowering users to develop models for predictive analytics, further aiding in solving complex business challenges.

Watson Studio acts as a comprehensive platform, integrating multiple tools to support the end-to-end process of solving business problems like

1. Data Preparation
2. Data Analysis
3. Collaboration
4. Machine Learning Integration
5. Decision-Making
6. Scalability and Flexibility
7. Security and Compliance

By offering a unified environment that spans data preparation, analysis, machine learning, and collaboration, Watson Studio equips businesses with the tools needed to tackle intricate challenges and make data-driven decisions effectively.

1.b) What is data type? List out the types of data types with example.

Ans.1.b)

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are actually classes and variables are instances (object) of these classes. The following are the standard or built-in data types in Python:

1. int (Integer):
   - Represents whole numbers without decimal points.
   - Example: `x = 5`

2. float (Floating-point):
   - Represents numbers with decimal points or in exponential form.
   - Example: `y = 3.14`

3. str (String):
   - Represents sequences of characters (text).
   - Example: `name = “John”`

4. bool (Boolean):
   - Represents truth values, either `True` or `False`.
   - Example: `is_valid = True`

5. list:
   - Represents an ordered collection of items.
   - Example: `numbers = [1, 2, 3]`

6. tuple:
   - Represents an ordered, immutable collection of items.
   - Example: `coordinates = (4, 5)`

7. set:
   - Represents an unordered collection of unique items.
   - Example: `unique_numbers = {1, 2, 3}`

8. dict (Dictionary):
   - Represents a collection of key-value pairs.
   - Example: `person = {‘name’: ‘Alice’, ‘age’: 30}`

9. NoneType:
   - Represents the absence of a value or a null value.
   - Example: `result = None`

10. complex:
    - Represents complex numbers.
    - Example: `z = 2 + 3j`

11. bytes:
    - Represents a sequence of bytes.
    - Example: `data = b’hello’`

12. bytearray:
    - Represents a mutable sequence of bytes.
    - Example: `mutable_data = bytearray([65, 66, 67])`

13. memoryview:
    - Represents a view of the memory of an object.
    - Example: `view = memoryview(b’hello’)`

14. range:
    - Represents an immutable sequence of numbers.
    - Example: `numbers_range = range(5)`

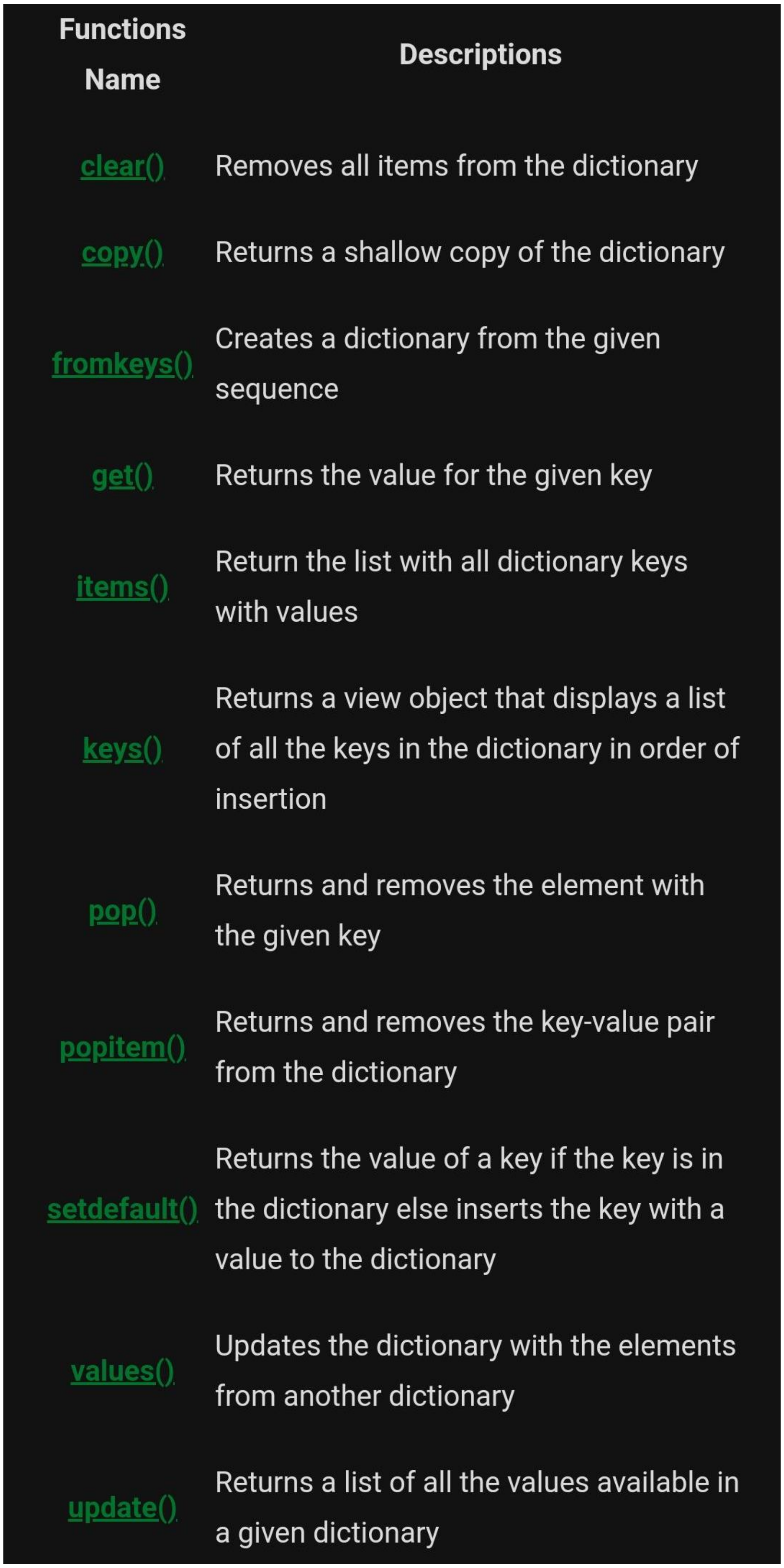
Understanding these data types is essential for effective programming in Python, as it allows you to work with different kinds of data and perform various operations on them.

2.a) What is dictionary? Explain the methods available in dictionary.

Ans.2.a)

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered*, changeable and do not allow duplicates.
- Dictionaries are written with curly brackets and have keys and values.

The methods available in dictionary are



Functions Name	Descriptions
clear()	Removes all items from the dictionary
copy()	Returns a shallow copy of the dictionary
fromkeys()	Creates a dictionary from the given sequence
get()	Returns the value for the given key
items()	Return the list with all dictionary keys with values
keys()	Returns a view object that displays a list of all the keys in the dictionary in order of insertion
pop()	Returns and removes the element with the given key
popitem()	Returns and removes the key-value pair from the dictionary
setdefault()	Returns the value of a key if the key is in the dictionary else inserts the key with a value to the dictionary
values()	Updates the dictionary with the elements from another dictionary
update()	Returns a list of all the values available in a given dictionary

2.b) Differentiate between NumPy Arrays and Lists.

Ans.2.b)

Sr. No.	**NumPy Arrays**	**Lists**
1	NumPy Array works on homogeneous types	Python lists are made for heterogeneous types
2	Python list support adding and removing of elements	NumPy Array does not support adding and removing of elements
3	Can’t contain elements of different types	Can contain elements of different types
4	Smaller memory consumption	Runtime not speedy

3.a) Explain the following term.

i) Raw Code ii) Waffle Chart

Ans.3.a)

i) Raw Code

Raw Code in data visualization typically refers to the actual programming code used to create visual representations of data. In Python, the term “raw code” typically refers to the source code as it was originally written. It’s the code that you write in your Python script before it’s interpreted or compiled by the Python interpreter.

Here’s an example of raw Python code:

```
def greet(name):
   return f"Hello, {name}!"
print(greet("World"))
```

In this raw code:

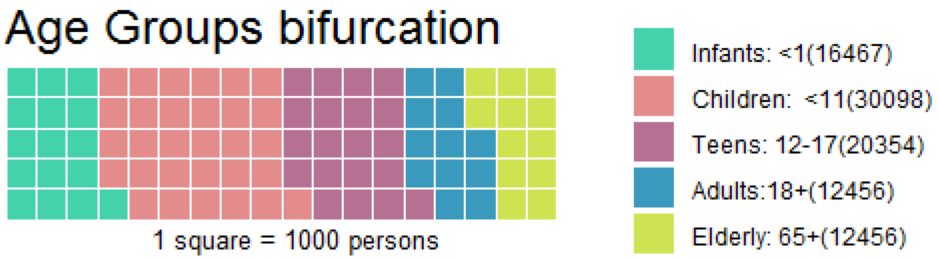
We define a function greet that takes one argument name.
The function returns a string that says “Hello, {name}!” where {name} is replaced with the argument passed to the function.
We then call this function with the argument “World” and print the result.
This raw code would output: Hello, World!

Raw Python code can include all constructs of the Python language, such as variables, functions, classes, and more. It’s called “raw” because it’s the code as you originally wrote it, without any changes made by interpreters or compilers.

When you run this raw code, the Python interpreter reads and executes it line by line. If the code is syntactically correct and doesn’t have any runtime errors, you’ll see the expected output. If there are any errors, the interpreter will stop and show an error message.

ii) Waffle Chart

A waffle chart shows progress towards a target or a completion percentage. Waffle Charts are a great way of visualizing data in relation to a whole, to highlight progress against a given threshold, or when dealing with populations too varied for pie charts. A lot of times, these are used as an alternative to the pie charts. It also has a niche for showing parts-to-whole contribution. It doesn't misrepresent or distort a data point (which a pie chart is sometimes guilty of doing).



Waffle Charts are mainly used when composing parts of a whole, or when comparing progress against a goal. These charts usually follow other kinds of data visualization for helping the understanding of the audience. For example, you might want a Waffle Chart when plotting how the expenses of a company are composed by each type of expense, or when classifying percentages of a population at a given moment. Waffle charts are also known as Squared Pie Charts. The individual values will be summed up and each that will be the total number of squares in the grid.

3.b) How will you create the normalized weight ?

Ans.3.b)

Create the normalized weights.

There are several methods of normalizations in statistics, each with its own use.

In this case, we will use feature scaling to bring all values into the range [0,1].

The general formula is:

$$X' = \frac{\mathrm{X} - \mathrm{X}_{min}}{X_{max} - X_{min}}$$

Where,

X is an original value.

X' is the normalized value.

X_{max} is maximum value.

X_{min} is minimum value.

The formula sets the max value in the dataset to 1 and sets the min value to 0.

The rest of the datapoints are scaled to a value between 0-1 accordingly.

4.a) Explain pie chart Specialized Visualization Tools using Matplotlib.

Ans.4.a)

Pie Charts - Specialized Visualization Tools using Matplotlib

A pie chart is a circular graphic that displays numeric proportions by dividing a circle (or pie) into proportional slices. You are most likely already familiar with pie charts as it is widely used in business and media. We can create pie charts in Matplotlib by passing in the kind=pie keyword.

4.b) Explain Bubble plots Specialized Visualization Tools using Matplotlib.

Ans.4.b)

Bubble Plots

A bubble plot is a variation of the scatter plot that displays three dimensions of data (x, y, z). The datapoints are replaced with bubbles, and the size of the bubble is determined by the third variable 'z', also known as the weight. In matplotlib, we can pass in an array or scalar to the keywords to plot(), that contains the weight of each point.

5.a) An e-commerce company ' wants to get into logistics "Deliver4U". It wants to know the pattern for maximum pickup calls from different areas of the city the day. This will result in:

i) Build optimum number of stations where its pickup delivery personnel will be located.

ii) Ensure pickup personnel reaches the pickup location at the earliest possible time.

Find the highest density of pickup locations in the future.

Ans.5.a)

Pre-requisites : Python, Jupyter Notebooks, Pandas

Data set : Please download the following from the location specified by the trainer.

The dataset contains two separate data files – train_del.csv and test_del.csv. The difference is that train_del.csv contains additional column which is trip_duration which we will not be needed for our present analysis.

Importing and pre-processing data:

a) **Import libraries** – Pandas and Folium. Drop the trip_duration column and combine the 2 different files as one dataframe.

We will need to generate some columns such as month or other time features using Datetime package of python. Let us then use it with Folium.

Please note that month, week, day, hour columns will be used next for our analysis.

Note the following regarding visualizing spatial data with Folium:

- Maps are defined as folium. Map object. We will need to add other objects on top of this before rendering.
- Different map tiles for map rendered by Folium can be seen at : https://github.com/pythonvisualization/folium/tree/master/folium/templates/tiles
- Folium.Map() : First thing to be executed when you work with Folium.

Let us define the default map object:

Let us now visualize the rides data using a class method called Heatmap()

Code for reference:

```
from folium.plugins import HeatMap
df_copy = df[df.month>4].copy()
df_copy['count'] = 1
base_map = generateBaseMap()
HeatMap(data=df_copy[['pickup_latitude', 'pickup_longitude',
'count']].groupby(['pickup_latitude',
'pickup_longitude']).sum().reset_index().values.tolist(), radius=8,
max_zoom=13).add_to(base_map)
```

Interpretation of the output:

There is high demand for cabs in areas marked by the heat map which is central Delhi most probably and other surrounding areas.

Now let us add functionality to add markers to the map by using the folium.ClickForMarker() object.

After adding the below line of code, we can add markers on the map to recommends points where logistic pickup stops can be built.

We can also animate our heat maps to dynamically change the data on timely basis based on a certain dimension of time. This can be done using HeatMapWithTime(). Use the following code :

```
df_hour_list = []
for hour in df_copy.hour.sort_values().unique():
df_hour_list.append(df_copy.loc[df_copy.hour == hour, ['pickup_latitude',
'pickup_longitude', 'count']].groupby(['pickup_latitude',
'pickup_longitude']).sum().reset_index().values.tolist())
from folium.plugins import HeatMapWithTime
base_map = generateBaseMap(default_zoom_start=11)
HeatMapWithTime(df_hour_list, radius=5, gradient={0.2: 'blue', 0.4: 'lime', 0.6:
'orange', 1: 'red'}, min_opacity=0.5, max_opacity=0.8,
use_local_extrema=True).add_to(base_map)
base_map
```

Conclusion

Throughout the city, pickups are more probable from central area so better to set lot of pickup stops at these locations. Therefore, by using maps we can highlight trends, uncover patterns, and derive insights from the data.

5.b) Explain Spatial Visualizations and Analysis in Python with Folium.

Ans.5.b)

Folium is a powerful data visualization library in Python that was built primarily to help people visualize geospatial data. With Folium, you can create a map of any location in the world if you know its latitude and longitude values. You can also create a map and superimpose markers as well as clusters of markers on top of the map for cool and very interesting visualizations. You can also create maps of different styles such as street level map, stamen map.

Folium is not available by default. So, we first need to install it before we can import it.
We can use the command : conda install -c conda-forge folium=0.5.0 –yes

It is not available via default conda channel.
Try using conda-forge channel to install folium as shown: conda install -c conda-forge folium
Generating the world map is straight forward in Folium. You simply create a Folium Map object and then you display it. What is attractive about Folium maps is that they are interactive, so you can zoom into any region of interest despite the initial zoom level.

Go ahead. Try zooming in and out of the rendered map above. You can customize this default definition of the world map by specifying the center of your map and the initial zoom level. All locations on a map are defined by their respective Latitude and Longitude values. So, you can create a map and pass in a center of Latitude and Longitude values of [0, 0]. For a defined center, you can also define the initial zoom level into that location when the map is rendered. The higher the zoom level the more the map is zoomed into the center. Let's create a map centered around Canada and play with the zoom level to see how it affects the rendered map.

6.a) Explain the following terms

i) Regression Plots **ii) Matrix Plots**

Ans.6.a)

i) Regression Plots

The regression plots in seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses. Regression plots as the name suggests creates a regression line between 2 parameters and helps to visualize their linear relationships. This article deals with those kinds of plots in seaborn and shows the ways that can be adapted to change the size, aspect, ratio etc. of such plots.

Seaborn is not only a visualization library but also a provider of built-in datasets. Here, we will be working with one of such datasets in seaborn named 'tips'. The tips dataset contains information about the people who probably had food at the restaurant and whether they left a tip. It also provides information about the gender of the people, whether they smoke, day, time and so on.

ii) Matrix Plots

A plot of matrix data is called a matrix plot. A matrix plot is a color-coded figure with values, data in the rows, and data in the columns. You can create matrix plots in seaborn by using either heatmap() or clustermap() functions.

Heatmap() is used to produce rectangular data as a color-coded matrix and Clustermap() is used to plot a dataset as a hierarchically clustered heat map.

To plot any data into a map, we need to import the data. You can either use datasets available in the seaborn library or import them as per your choice from elsewhere.

6.b) Explain the following terms

i) Distribution plots ii)Categorial plots

Ans.6.b)

i) Distribution plots

The empirical distribution of the data with the theoretical values anticipated from a certain distribution, distribution plots provide a visual assessment of the distribution of sample data. To ascertain whether the sample data originates from a certain distribution, use distribution plots in addition to more formal hypothesis testing.

Plot distribution options are available in the Statistics and Machine Learning Toolbox as follows:

- Normal Probability Plots
- Quantile-Quantile Plots
- Cumulative Distribution Plots
- Types of Distributed Plotting

Seaborn distribution plots are employed to analyze univariate and bivariate distributions. This article will explore four distribution plots, including the following:

- Joint Plot
- Dist Plot
- Pair Plot
- Seaborn Distplot .
- Rug Plot

ii)Categorial plots

Categorical plots visualize the distribution of categories within a dataset and can be useful for identifying patterns and trends within the data. Distribution plots are used to visualize the distribution of continuous variables and can be useful for identifying patterns and trends within the data.

Categorical data is data that belongs to a particular category or group. In Python, categorical data is usually stored as a Pandas categorical data type.

Categorical data is often used to store data that can be divided into a fixed number of categories, such as gender (male, female), product type (phone, computer, TV), or blood type (A, B, AB, O). A variable that only permits categorization but not a definite ordering of the variables is purely categorical. A variable will be considered ordinal if it has a distinct ordering.