01) Installation and study of one Data Analytics Tool Frame work.

⇒

Step 1 : Install Anaconda

We strongly recommend installing the Anaconda Distribution, which includes python, Jupyter Notebook (a lightweight IDE very popular among data scientists) and all the major libraries.

It's the closet thing to a one-stop-shop for all your setup needs.

Simply download Anaconda with the latest version of python 3 and follow the wizard.

Step 2 : Start Jupyter Notebook

Jupyter Notebook is our favorite IDE (integrated development environment) for data science in Python. An IDE is just a fancy name for an advanced text editor for coding.

(As an analogy, think of Excel as an "IDE for spreadsheets." For example, it has tabs, plugins, keyword shortcuts and other useful extras)

The good news is that Jupyter Notebook already came installed with Anaconda. Three cheers for synergy! To open it, run the following command in the Command Prompt (Windows) or Terminal (Mac/Linux):

```
Python
1 jupyter notebook
```

Alternatively, you can open Anacondas "Navigator" application, and the launch the notebook from there.

You should see this dashboard open in your browser.

*Note :→ If you got a message about "logging in", simply follow the instructions in the browser. You'll just need to paste in a token from the Command Prompt/Terminal.

Step 3: Open New Notebook

First, navigate to the folder you'd like to save the notebook in. For begineers, we recommend having a single "Data Science" folder that you can use to store your datasets as well.

Then, open a new notebook by clicking "New" in the top right. It will open in your default web browser. You should see a blank canvas brimming with potential.

Step 4: Try Math Calculations:

Next, let's write some code. Using python for data science is awesome because it's extremely versatile. For example, you can use Python as a calculator:

```
import math

# Area of circle with radius 5
25*math.pi

# Two to the fourth
2**4

# Length of triangles hypotenuse
math.sqrt(3**2+4**2)
```

(To run a code cell, click into the cell so

that it's highlighted and then press Shift+Enter on your keyboard)

A few important notes:

First, we important Python's math module, which provides convenient functions (eg. math.sqrt()) and math constants (eg. math.pi).

Second, 2*2*2*2 or "two to fourth"...is written as 2**4. If you write 2^4, you'll get a very different output!

Finally, the text following the "hashtags" (#) are called comments. Just as their name implies, these text snippets are not as code.

In addition, Jupyter Notebook will only display the output from final line of code:

```
import math
# Area of circle with radius 5
25 *math.pi
# Two to the fourth
2**4
# Length of triangle's hypotenuse
math.sqrt(3**2 + 4**2)
```

output :→ 5.0

To print multiple calculations in one output, wrap each of them in the print(...) function.

Python

```
1 # Area of circle with radius 5
2 print (25* math.pi)
3
4 # Two to the fourth
5 print (2**4)
6
7 # Length of triangles hypotenuse
```

8 print (math.sqrt (3**2 + 4**2)

Another useful tip is that you can store things in objects (ie variables). See if you can follow along what this code is doing:

Python

```
1 message = "The length of the hypotenuse is"
2 c = math.sqrt (3**2 + 4**2)
3 print (message, c)
```

By the way, in the above code, the message was surrounded by quotes, which means means It's a string. A string is any sequences of characters surrounded by single or double quotes.

Now, we're not going to dive much further into the weeds right now. To learn Python for Data Science Self-Study Guide.

Contrary to popular belief, your won't actually need to learn an immense amount of programming to use Python for data science. That's because most of the data science and machine learning functionality you'll need are already packaged into libraries, or boundries of code that you can import and use out of the box.

Step 5: Import Data Science Libraries

Think of Jupyter Notebook as a big playground for Python. Now that you've set this up, you can play to your heart's content. Anaconda has almost all of the libraries you'll need, so testing a new one is as simple as importing it.

Which brings to the next step... Lets import those libraries. In a new code cell

(Insert > Insert Cell Below), write the following code:

```
Python
1 import pandas as pd
2
3 import matplotlib.pyplot as plt
4 %matplotlib inline
5 from sklearn.linear_model import LinearRegression
```

(It might take a while to run this code the first time).

So what did we just do? Let's break it down.

First, we imported the Pandas library. We also gave it the alias of pd. This means we can evoke the library with pd. You'll see this in action shortly.

Next, we imported the pyplot module from the matplotlib library. Matplotlib is the main plotting library for Python. There no need to bring in the entire library, so we just imported a single module. Again, we gave it an alias of plt.

Oh yea, and %matplotlib inline command? That's Jupyter Notebook specific. It simply tells the notebook to display our plots inside notebook, instead of in a separate screen.

Finally we imported a basic linear regression algorithm from scikit-learn. Scikit-learn has a buffet of algorithms to choose from. At the end of this guide, we'll point you to a few resources for learning more about these algorithms.

There are plenty of other great libraries available for data science, but these are the most commonly used.

Step 6 : Import Your Dataset

Next, let's import a dataset. Pandas has a suite of IO tools that allow you to read and write data. You can work with formats such as CSV, JSON, Excel, SQL databases, or even raw text files.

For this tutorial, we'll be reading from an Excel file that has data on the energy efficiency of buildings. Don't worry - even if you don't have Excel installed, you can still follow along.

First, download the dataset and put it into the same folder as your current Jupyter notebook.

Then, use the following code to read the file and store its contents in a df object ("df" is short for dataframe).

Python
```
df = pd.read_excel('ENB2012_data.xlsx')
```

If you saved the dataset in a sh subfolder, then you want would write the code like this instead:

Python
```
df = pd.read_excel(f'{subfolder_name}/ENB2012_data.xlsx')
```

Nice! You've successfully imported your first dataset using Python.

To see what's inside, just run this code in your notebook (it displays the first 5 observations from the dataframe):

Python
```
df.head()
```

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
0	0.98	514.5	294.0	110.25	7.0	2	0.0	0	15.55	21.33
1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	15.55	21.33
2	0.98	514.5	294.0	110.25	7.0	4	0.0	0	15.55	21.33
3	0.98	514.5	294.0	110.25	7.0	5	0.0	0	15.55	21.33
4	0.90	563.5	318.5	122.50	7.0	2	0.0	0	20.84	28.28

For extra practice on this step, feel free to download a few others from our hand-picked list of datasets. Then, try using other IO tools (such as pd.read_csv()) to import datasets with different formats.

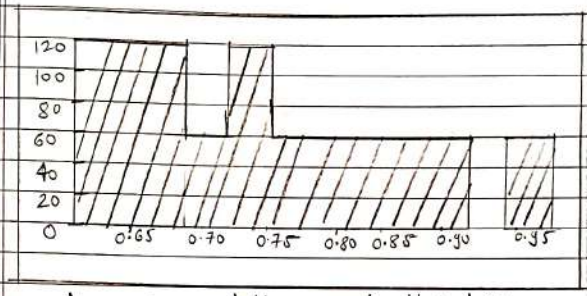
We showcase more of what you can do in Pandas in our Python Data Wrangling Tutorial (opens in a new tab).

Step 7: Explore Your Data

In step 6, we already saw some example observations from the dataframe. Now we're ready to look at plots.

We won't go through the entire exploratory analysis phase right now. Instead, let's just take a quick glance at the distributions of our variables. We'll start the "X1" variable, which refers to "Relative Compactness" as described in the file's data dictionary.

Python
```
plt.hist(df.X1)
```



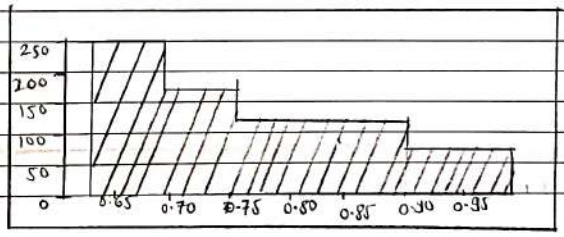
As you've probably guessed, plt.hist() produces a histogram. In genral, these types of functions will have different parameters that you can pass into them. Those parameters control things like the colour scheme, the number of bins used, the axes and so on.

There's no need to memorize all the parameters. Instead, get in the habit of checking the documentation page for available options. For example, the documentation page of plt.hist() indicates that you can change the number of bins in the histogram.

These means we can change the number of bins like so:

Python

```
plt.hist(df.X1, bins=5)
```



For now, we don't recommend trying to get too fancy with matplotlib. It's a powerful, but complex library.

Instead, we prefer a library that's built on top of matplotlib called seaborn. If matplotlib "tries to make easy things easy and hard things possible", seaborn tries to make a well-defined set of hard things easy as well. Learn more about it in our Seaborn Data Visualization Tutorial.

Step 8 : Clean Your Dataset

After we explore the dataset, it's time to clean it. Fortunately, this dataset is pretty clean already because it was originally collected from controlled simulations.

Even so, for illustrative purposes, let's at least check for missing value. You can do so with just one line of code (but there a ton of cool stuff packed into this one line).

Python

```
1. df.isnull().sum()
```

Lets unpack that :→

df is where we stored the data. It's called a "dataframe," and it's also a Python object, like the variable from step 4.

.isnull() is called a method, which is just a fancy term for a function attached to an object. This object method looks through our entire dataframe and labels any cell with a missing value as True. (Tip : Try running df.head().isnull() and see what you get!)

Finally, .sum() is a method that sums all

of the True values across each column. Well...
...technology technically, it sums any number,
while treating True as 1 and False as 0.

You can learn more about .isnull() and .sum()
on the documentation page for Pandas
dataframes.

02) Design and develop at least 10 problem statement which demonstrate the use of data structure, functions, Importing / Exporting Data in any data analytics tool.

1) Write a python program to accept input string from user and display number of vowels and consonant in string.

⇒
```
def count_vowels_and_consonants(input_string):
    vowels = "AEIOUaeiou"
    num_vowels = 0
    num_consonants = 0
    for char in input_string:
        if char.isalpha():
            if char in vowels:
                num_vowels += 1
            else:
                num_consonants += 1
    return num_vowels, num_consants
user_input = input("Enter a string:")
vowels, consonants = count_vowels_and_consonants(user_input)
print("Number of vowels:", vowels)
print("Number of consonants:", consonants)
```

OUTPUT :-

```
Enter a string: RUTIME POLYMORPHISM
Number of vowels: 6
Number of consonants : 13
```

2) Write a Numpy program to convert a list of numeric values into a dimensional Numpy array.
Expected Output:
Original List: [12.23, 13.32, 100, 36.32]
One-dimensional Numpy array: [12.23 13.32 100. 36.32]

```
import numpy as np
# Create the original list of numeric values
original_list = [12.23, 13.32, 100, 36.32]
# Convert the list to a one-dimensional Numpy array
numpy_array = np.array(original_list)
# Display the original list and the resulting Numpy array
print("Original List:", original_list)
print("One-dimensional Numpy array:", numpy_array)
```

OUTPUT:→

```
Original List: [12.23, 13.32, 100, 36.32]
One-dimensional Numpy array: [12.23 13.32 100. 36.32]
```

3) Write a python program to print a specified list after removing the 0th, 4th and 5th element.
Sample List: ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow']
Expected Output: ['Green', 'White', 'Black']

⟹

```
sample_list = ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow']
# Define the indices of elements to remove
indices_to_remove = [0, 4, 5]
# Create a new list with specified elements removed
result_list = [sample_list[i] for i in range(len(sample_list)) if i not
in indices_to_remove]
# Display the resulting list
print("Sample List:", sample_list)
print("Expected Output:", result_list)
```

OUTPUT:→

```
Sample List: ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow']
Expected Output: ['Green', 'White', 'Black']
```

4) Write a python program to shuffle and print a specified list.

⇒

```
import random
#specify the list to be shuffled
specified_list = ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow']
# Shuffle the list in place
random.shuffle(specified_list)
#print the shuffled list
print("Shuffled List:", specified_list)
```

OUTPUT:→

```
Shuffled List: ['Yellow', 'White', 'Pink', 'Black', 'Green', 'Red']
```

5) Write a python program to import any csv file using python.

⟹

To import a csv file using python and display the data, you can use the pandas library. First make sure you have pandas installed.

```
pip install pandas
```

Here's a python program to import a csv file and display its data:

```
import pandas as pd
# replace 'your_file.csv' with the path to your csv file
csv_file_path = 'your_file.csv'
# Read the csv file into a pandas DataFrame
try:
    df = pd.read_csv(csv_file_path)
    print("Data from the csv file:")
    print(df)
except FileNotFoundError:
    print(f"Csv file '{csv_file_path}' not found.")
except pd.errors.EmptyDataError:
    print(f"Csv file '{csv_file_path}' is empty.")
except pd.errors.ParseError:
    print(f"Unable to parse csv file '{csv_file_path}'. Make sure it's valid csv format.")
```

Replace 'your_file.csv' with the actual path to your csv file. This program will attempt to read the csv file, and if successful, it will display the data as a pandas Dataframe.

6) Write a python program to export any csv file using python.

⇒ To export data to a CSV file using Python, you can use the pandas library. First, make sure you have pandas installed.

```
pip install pandas
```

Heres a python program to export data to a CSV file:

```
import pandas as pd
# create a sample DataFrame (you can replace this with your own data)
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 22],
    'City': ['New York', 'San Francisco', 'Los Angeles']
}
df = pd.DataFrame(data)
# specify the CSV file path where you want to export the data
csv_file_path = 'exported_data.csv'
# Export the DataFrame to a CSV file
df.to_csv(csv_file_path, index=False) # Set index=False to exclude the row index
print(f"Data has been exported to {csv_file_path}")
```

OUTPUT :→

After running this program, the data in the DataFrame will be exported to the specified CSV file, and you'll see a message indicating the successful export.

7) Write a python to accept input from user and check whether number is Armstrong or not. Using Function.

⟹

```
def is_armstrong_number(num):
    #calculate the number of digits in the input number
    num_str = str(num)
    num_digits = len(num_str)
    # Calculate the sum of digits, each raised to the power
of the number of digits.
    armstrong_sum = sum(int(digit)**num_digits for digit in num_str)
    # Check if the sum is equal to the original number
    return armstrong_sum == num
# Get input from the user
try:
    number = int(input("Enter a number: "))
    if is_armstrong_number(number):
        print(f"{number} is an Armstrong number.")
    else:
        print(f"{number} is not an Armstrong number.")
except ValueError:
    print("Invalid input. Please enter a valid number.")
```

OUTPUT :→

Enter a number : 52

52 is not an Armstrong number.

8) Write a Python program to print factorial of number using Recursion.

→

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
try:
    num = int(input("Enter a number to calculate its factorial: "))
    if num < 0:
        print("Factorial is not defined for negative numbers.")
    else:
        result = factorial(num)
        print(f"The factorial of {num} is {result}")
except ValueError:
    print("Invalid input. Please enter a valid number.")
```

OUTPUT :→

```
Enter a number to calculate its factorial : 12
The factorial of 12 is 479001600
```

9) Write a program to print length of string using Recursion.

→

```
def string_length_recursive(s):
    if s == "":
        return 0
    else:
        return 1 + string_length_recursive(s[1:])
try:
    input_string = input("Enter a string : ")
    length = string_length_recursive(input_string)
    print(f"The length of the string is: {length}")
except ValueError:
    print("Invalid input. Please enter a valid string.")
```

OUTPUT :→

```
Enter a string : EVERY PERSON HAS FAILURE
The length of the string is : 24
```

10) Write a python program to count the occurance of each word in given sentence?

⇒

```
def count_words(sentence):
    # Split the sentence into words
    Words = sentence.split()
    # Create a dictionary to store word frequencies
    word_count={}
    for word in words:
        # Remove punctuation and convert to lowercase for
case-insensitive counting
        word = word.strip(".,!?").lower()
        # Update the word count
        if word in word_count:
            word_count[word] += 1
        else:
            word_count[word] = 1
    return word_count
try:
    input_sentence = input("Enter a sentence: ")
    word_count = count_words(input_sentence)
    print("Word occurances in the sentence:")
    for word, count in word_count.items():
        print(f"{word}: {count}")
except ValueError:
    print("Invalid input.please enter a valid sentence.")
```

OUTPUT:→

Enter a sentence: YOU ARE HUNGRY
Word occurancerres in the sentence:
you: 1
are: 1
hungry: 1

Q.3) Design and develop at least 5 problem statements which demonstrate the use of Control Structures of any data analytics tool.

1) Create a list of the odd numbers between 1 and 20 (use while, break)

⟹

• Here are five problem statements that demonstrate the use of control structures in data analytics :→

i) Calculate the Average :→

Write a Python program that calculate the average of a list of numbers entered by the users. Use while loop to repeadetely input numbers and compute the average until the users decides to stop.

ii) Find the Maximum Value :→

Using a for loop, find the maximum value in a given list of numbers and return it as the result.

iii) Filtering Data :→

Given a dataset of student scores, use if-else statements to categories students as 'pass' or 'fail' based on passing score threshold. Count and display the number of students in each category.

iv) Sales Forecast :→

Create a Python program to estimate monthly sales for retail store. Prompt the user for the previous month's sales and use a loop to project the next 12 month's sales by applying a growth rate.

v) Odd Numbers List (using while and break) :→

Create a list of odd numbers between

1 and 20 using a while loop and 'break' statement. Start with 1 and keep adding 2 to the last number until you reach 20.

- Here's Python program to generate a list of odd numbers between 1 and 20 using a while loop and the 'break' statement:

```
odd_numbers = []
num = 1
while num <= 20:
    odd_numbers.append(num)
    num += 2
print("List of odd numbers between 1 and 20:", odd_numbers)
```

OUTPUT :→

List of odd numbers between 1 and 20: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

2) Creating an infinite loop.

→

• Here a five problem statements that demonstrate the use of control structures, including an example of creating an infinite loop :→

i) Infinite Loop with Exit Option :→

Create a Python program that runs an infinite loop. Inside the loop, prompt the user if they want to continue. If the users enters 'exit' or 'quit', break out of the loop; otherwise, continue the loop.

ii) Guess the Number Game :→

Develop a number guessing game where the program genrate a random number, and the user has to guess it. Use a loop to allow the user to keep guessing until they get the correct answer.

iii) Inventory Management :→

Design a program to manage the inventory of a store. Use a loop to continuosly ask the user for actions like adding items, removing items, or checking the inventory status.

iv) Real-time Data Monitoring :→

Create a Python script that monitors real-time data from a data sources, such as a temperature sensor. Use a loop to keepp checking the data and tigger actions based on predefined thresholds.

v) Simulate a Traffic Light :→

Simulate a traffic light system using different colors and time intervals. Use a loop to cycle through the traffic light phase indefinetly.

•

• Here's an example of the first problem statement, creating an infinite loop with an exit option:

```
while True:
    user_input = input("Do you want to continue(type 'exit' to quit):")
    if user_input.lower() in ['exit', 'quit']:
        break
    # Additional code can be placed here for the actions to be performed within the loop.
print("Loop exited. Program finished.")
```

OUTPUT :→

```
Do you want continue (type 'exit' to quit): QUIT
Loop exited. Program finished
```

5) Skip the iteration if the current number is 6 (use while, continue)

⇒

- Here are five problem statements that demonstrate the use of control structures, specifically the while loop and the continue statement, to skip iterations when the current number is 6:

i) Print Numbers 1 to 10 (Skip 6):

Write a Python program using a while loop to print numbers from 1 to 10, skipping the number 6.

ii) Sum of Even Numbers (Skip 6):

Create a program that calculates the sum of even numbers between 1 and 20 using a while loop. Skip the number 6 in the sum.

iii) Display Multiplication Table (Skip 6):→

Develop a program to display the multiplication table ~~loop~~, ~~but~~ skip of a given number using a whole loop. Skip the multiplication by 6.

iv) List of Squares (Skip 6):→

Generate a list of squares of numbers from 1 to 10 using a while loop, but skip squaring the number 6.

v) Factorial Calculation (Skip 6):→

Write a program that calculates the factorial of a number using a while loop, skipping the multiplication by 6.

• Here's an example of the first problem statement, printing numbers from 1 to 10 while skipping the number 6:

```
num = 1
while num <= 10:
    if num == 6:
        num += 1
        continue
    print(num)
    num += 1
```

OUTPUT :→

1 2 3 4 5 7 8 9 10

4) Write a program to demonstrate use else with a for loop with example

⇒

• Here are five problem statement that demonstrate the use of control structures in data analytics, specifically using the else clause with a loop, along with examples and output:

i) Prime Number Checker :→

Write a program that checks if a given number is prime or not using a for loop. Use the else clause to display whether the number is prime or not.

```
num = int(input("Enter a number."))
for i in range(2, num):
    if num % i == 0:
        print(f"{num} is not a prime number.")
        break
else:
    print(f"{num} is a prime number.")
```

OUTPUT :→

Enter a number : 7
7 is a prime number.

ii) Find the maximum value :→

Create a program to find the maximum value in a list of numbers using a for loop. Use the else clause to display the maximum value.

```
numbers = [12, 45, 7, 32, 88, 21]
max_val = numbers[0]
for num in numbers:
    if num > max_val:
        max_val = num
```

```
else:
    print(f"The maximum value is {max_val}.")
```

OUTPUT:→

The maximum value is 88.

iii) Search for an Element:

Write a program to search for an element in a list using a for loop. Use the else clause to indicate if the element was not found.

```
my_list = [3, 5, 9, 12, 18, 25]
search_value = 10
for num in my_list:
    if num == search_value:
        print(f"{search_value} found in the list.")
        break
else:
    print(f"{search_value} not found in the list.")
```

OUTPUT:→

10 not found in the list.

iv) Check for Even Numbers:→

Create a program to check if all numbers in a list are even using a for loop. Use the else clause to display the result.

```
numbers = [2, 4, 6, 8, 10]
for num in numbers:
    if num % 2 != 0:
        print("Not all numbers are even.")
        break
else:
    print("All numbers are even.")
```

OUTPUT :→

All numbers are even.

v) Count Characters in a String :→

Write a program to count the number of occurrences of a specific character in a string using for loop. Use the else clause to display the count.

```
input_string = "Hello, World!"
char_to_count = "o"
count = 0
for char in input_string:
    if char == char_to_count:
        count += 1
else:
    print(f"The character '{char_to_count}' appears {count} times in the string.")
```

OUTPUT :→

The character 'o' appears 2 times in the string.

5) Write a program to demonstrate If the inner if condition is false, then the inner else will execute.

⇒

• Here are five problem statements that demonstrate the use of control structures in data analytics, specifically focusing on using an inner if-else construct to handle scenarios where the inner if condition is false :-

1) Exam Grading :→

Write a program that calculates the letter grade for a given score. If the score is below a certain threshold, an inner if-else condition can be used to assign a different grade, such as "Fail".

```
score = 78
if score >= 90:
    grade = 'A'
else:
    if score>= 60:
        grade = 'B'
    else:
        grade = 'Fail'
print(f"The grade is {grade}")
```

OUTPUT :→

The grade is B

ii) Ticket Pricing :→

Create a program that calculates the price of a movie ticket based on age. If the age is below a certain threshold, an inner if-else condition can apply a discount.

```
age = 15
ticket_price = 12
if age < 12:
    ticket_price -= 3
else:
    if age >= 60:
        ticket_price -= 2
print(f"The ticket price is ${ticket_price}")
```

OUTPUT :→

The ticket price is $ 12

iii) Discount Calculation :→

Write a program to calculate the total price of the items in a shopping cart. Apply different discount based on item categories using an inner if-else construct

```
item_category = 'Electronics'
item_price = 500
discount = 0
if item_category == 'Electronics':
    discount = 0.1
else:
    if item_category == 'Clothing':
        discount = 0.2
total_price = item_price - (item_price * discount)
print(f"The total price is ${total_price}")
```

OUTPUT :→

The total price is $ 450.0

iv) Membership Status :→

Create a program to determine a user's

membership status and eligibility for certain benefits. Apply an inner if-else condition if a certain requirements is not met.

```
membership = 'Premium'
points = 45
if membership == 'Premium':
    if points < 50:
        membership = 'Basics'
print(f"You have a {membership} membership.")
```

OUTPUT :→

You have a Basics membership.

v) Temperature Analysis :→

vi) Temperature Develop a program to classify the whether based on temperature. Use an inner if-else condition to provide additional details if the temperature is extremely hot or cold.

```
temperature = 95
weather = ''
if temperature > 80:
    weather = 'hot'
    if temperature > 100:
        weather = 'extremely hot'
else:
    if temperature < 40:
        weather = 'cold'
        if temperature < 20:
            weather = 'extremely cold'
print(f"The weather is {weather}.")
```

OUTPUT :→

The weather is hot.

04) Implement any 2 Classification techniques using any data analytics tool.

1) Implement Classification algorithm kNN classifier Data Analysis on given iris dataset.

→

Here as an example of implementing the k-Nearest Neighbors (kNN) classification algorithm using Python and the Iris dataset. In this example, we will use the scikit-learn library to perform kNN classification.

First, make sure you have scikit-learn installed. You can install it using pip:

```
pip install scikit-learn
```

Now, here's Python program that demonstrates kNN classification on the Iris dataset:

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
# Split the dataset into a training set and a testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0
, random_state = 42)
# create a kNN classifier with a specified number of
neighbors (e.g. 3)
k=3
knn_classifier = KNeighborsClassifier(n_neighbors = k)
```

```
#Fit the classifier on the training data
knn_classifier.fit(X_train, y_train)
#make prediction on the testing data
y_pred = knn_classifier.predict(X_test)
#calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
#Display classification report (includes precision, recall, f1-score
and support)
report = classification_report(y_test, y_pred, target_names=iris.target_names)
print("Classification Report:\n", report)
```

In this program :→

i) We load the Iris dataset using datasets.load_iris() from scikit-learn.

ii) The dataset is split into a training set and testing set using train_test_split to evaluate the classifier's performance.

iii) We create a KNN classifier with a specified number of neighbors (in this case, 3) using KNeighborsClassifier.

iv) The classifier is trained on the training data using fit.

v) Prediction are made on the on the testing data using predict.

vi) The accuracy of the model is calculated using accuracy_score.

vii) A classification report is genrated using classification_report, which provides additional evaluation metrics.

When you run this program, you will get an output that includes the accuracy of the KNN

classifier and a classification report that contains precision, recall, F1-score and support for each class in the Iris dataset.

2) Implement Classification algorithm Decision tree Data Analysis on given iris dataset.

⇒ T

To implement a Decision Tree classification algorithm on the Iris dataset, we can use Python and the scikit-learn library. In this example, we'll perform Decision Tree classification on the Iris dataset.

Here's how you can do it:

i) Install scikit-learn :-

If you haven't already, install the scikit-learn library using pip:

```
pip install scikit-learn
```

ii) Python code :-

Use the following python code to implement the Decision Tree classifier on the Iris dataset:-

```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score
#Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
# Split the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.3,
random_state = 42)
#Create a Decision Tree classifier
decision_tree_classifier = DecisionTreeClassifier()
```

```
# Fit the classifier to the training data
decision_tree_classifier.fit(X_train, y_train)
# Make prediction on the test set
y_pred = decision_tree_classifier.predict(X_test)
# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
# Display classification report
class_names = iris.target_names
report = classification_report(y_test, y_pred, target_names=class_names)
print("Classification Report:\n", report)
```

- This code performs the following steps:
  i) Loads the Iris dataset from scikit-learn.
  ii) Splits the dataset into a training set and a testing set.
  iii) Creates a Decision Tree classifier
  iv) Fits the classifier to the training data.
  v) Makes prediction on the test set.
  vi) Calculate the accuracy of the model and display a classification report that provides additional evaluation metrics.

When you run this code, you'll see the accuracy of the Decision Tree classifier and a classification report that includes precision, recall, F_1-score and support for each class in the Iris dataset.

3) Implement Classification algorithm SVM Classifier Data Analysis on given iris dataset.

⇒

To implements a Support Vector Machine (SVM) classifier on the Iris dataset.

Here's how you can do it:

i) Install scikit-learn:

If you haven't already, install the scikit-learn library using pip:

```
pip install scikit-learn
```

ii) Python code:

Use the following Python code to implement the SVM classifier on the Iris dataset:

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state = 42)
# Create an SVM classifier
svm_classifier = SVC(kernal = 'linear')
# Fit the classifier to the training data
svm_classifier.fit(X_train, y_train)
# Make prediction on the test set
y_pred = svm_classifier.predict(X_test)
```

```
#Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
#Display classification report
class_names = iris.target_names
report = classification_report(y_test, y_pred, target_names=class_names)
print('Classification Report:\n', report)
```

This code perform the following steps:

i) Loads the Iris dataset from scikit-learn.

ii) Splits the dataset into a training set and a testing set.

iii) Creates an SVM classifier with a linear kernal.

iv) Fits the classifier to the training data.

v) Makes prediction on the test set.

vi) Calculates the accuracy of the model and displays a classification report that provides additional evaluation metrics.

When you run this code, you'll see the accuracy of the ~~svm~~ SVM classifier and a classification report that includes precision, recall, F1-score, and support for each class in the Iris dataset.

05) Implement any 2 Clustering techniques using any data analytics tool.

1) Implement clustering algorithm on given "income.csv" dataset and display it using scatter plot.

⟹

To implement clustering techniques on a dataset, you can use python and libraries such as scikit-learn for clustering and matplotlib for visualization. In this example, we'll perform K-Means and Hierarchical clustering on the "income.csv" dataset and display the clusters using scatters plots.

Assuming you have the "income.csv" dataset, you can follow these steps :→

i) Install neccessary libraries :→

Ensure you have scikit-learn and matplotlib ~~lib~~ installed. You can install them using pip:

```
pip install scikit-learn matplotlib
```

ii) Python code :→

Here's how you can implement clustering and create scatters plots:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.preprocessing import StandardScaler
#Load the dataset
data = pd.read_csv("income.csv")
#Select relevant columns for clustering (e.g. income and age)
X = data[['Income', 'Age']]
```

```
#Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(x)
# K-means Clustering
kmeans = KMeans(n_clusters = 3, random_state = 0)
kmeans_labels = kmeans.fit_predict(X_scaled)
# Hierarchical Clustering (Agglomerative)
agg_clustering = AgglomerativeClustering(n_clusters = 3)
agg_labels = agg_clustering.fit_predict(X_scaled)
#plot K-Means Clusters
plt.figure(figsize = (12,6))
plt.subplot(121)
plt.scatter(X['Income'], X['Age'], c=kmeans_labels, cmap='viridis')
plt.title('K-Means Clustering')
plt.xlabel('Income')
plt.ylabel('Age')
# plot Hierarchical Clusters
plt.subplot(122)
plt.scatter(X['Income'], X['Age'], c=agg_labels, cmap='viridis')
plt.title('Hierarchical Clustering')
plt.xlabel('Income')
plt.ylabel('Age')
plt.tight_layout()
plt.show()
```

. In this code :

i) We load the "income.csv" dataset and select relevant columns for clustering (Income and Age).

ii) We standardize the data using StandardScaler.

iii) We perform k-Means clustering with n_clusters=3 and Hierarchical clustering with n_clusters = 3.

iv) We create scatter plots to visualize the

clusters using different colors.

Making sure to replace "income.csv" with the actual path to your dataset file. The code will display two scatter plots showing the clustering results using K-Means and Hierarchical clustering algorithms.
Adjust the numbers of clusters and the features you want to cluster based on your dataset and analysis needs.

2) Implement Classification algorithm Decision Tree Classifier Data Classifier Data Analysis on given iris dataset.

→

You mentioned that you want to implement two clustering techniques followed by a Decision Tree classification algorithm. To fulfil this request, I will first demonstrate two clustering techniques (K-Means and Hierarchical Clustering) and then implement a Decision Tree classifier on the Iris dataset.

Lets start with the clustering techniques:-

```
# Clustering Techniques (K-Means and Hierarchical Clustering):
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.preprocessing import StandardScaler
from sklearn import datasets
# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
X = StandardScaler().fit_transform(X)
# K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans_labels = kmeans.fit_predict(X)
# Hierarchical Clustering (Agglomerative)
agg_clustering = AgglomerativeClustering(n_clusters=3)
agg_labels = agg_clustering.fit_predict(X)
# Plot K-Means Clusters
plt.figure(figsize=(12,6))
```

```
plt.subplot(121)
plt.scatter(X[:,0], X[:,1], c=kmeans_labels, cmap='viridis')
plt.title('K-Means Clustering')
plt.tight_layout()
plt.show()
```

In this code above, we first load the Iris dataset, standardize the data, them perform k-Means and Hierarchical Clustering. The clusters are visualized using scatter plots.

Now, let's move on to the Decision Tree classification :-

- Decision Tree Classification :-

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
#Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
#Split the dataset into a training set and a testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
#Create a Decision Tree classifier
decision_tree_classifier = DecisionTreeClassifier()
#Fit the classifier to the training data
decision_tree_classifier.fit(X_train, y_train)
#Make prediction on the test set
y_pred = decision_tree_classifier.predict(X_test)
#Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy : {accuracy : .2f}")
```

```
#Display classification report
class_name = iris.target_names
report = classification_report (y_test, y_pred, target_names=class_name)
print ('Classification Report : \n', report)
```

This code first load the Iris dataset, splits it into training and testing sets and then perform Decision Tree Classification. The accuracy and Classification report are displayed as the output. Ensure you have the neccessary libraries installed by running the installation commands mentioned in the previous response

6) Implement any 2 Association Rule Mining techniques using any data analytics tool.
Solution : in otherfile

⟹

I understand that you would like to implement two Association Rule Mining techniques using a data analytics tool, but I cannot view or access external files or solutions. However, I can provide you with python code examples for implementing two common Association Rule Minning techniques: Apriori and FP-growth. You can run these examples in your preffered data analytics tools.

• Apriori Algorithm Example :→

```
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import pandas as pd
# sample transaction data
data = {'TID': [1,2,3,4,5],
    'Items' : [['milk', 'bread', 'nuts'],
            ['milk', 'bread', 'diapers'],
            ['milk', 'diapers'],
            ['milk', 'bread'],
            ['milk', 'bread', 'nuts']]}
df = pd.DataFrame(data)
# Convert items to one-hot encoded format
oht = pd.get_dummies(df['Items'].apply(pd.Series).stack().sum(level=0)
# Find frequent item sets using Apriori
frequent_item_sets = apriori(oht, min_support=0.2, use_colnames=True)
# Find association rules
association_rules = association_rules(frequent_item_sets, metric="lift",
min_threshold=1.0)
```

```
print(association_rules)
```

• Fp_growth Algorithm Example :

```
from mlxtend.frequent_pattern import fpgrowth
from mlxtend.frequent_pattern import association_rules
import pandas as pd
# sample transaction data
data = {'TID': [1,2,3,4,5],
        'Items': [['milk', 'bread', 'nuts'],
                  ['milk', 'bread', 'diapers'],
                  ['milk', 'diapers'],
                  ['milk', 'bread'],
                  ['milk', 'bread', 'nuts']]}
df = pd.DataFrame(data)
# Convert items to one-hot encoded format
oht = pd.get_dummies(df['Items'].apply(pd.Series).stack()).sum(level=0)
# Find frequent item sets using Fp-growth
frequent_item_sets = fpgrowth(oht, min_support=0.2, use_colnames=True)
# Find association rules)
```

You can adapt these code examples to your specific dataset and data analytics tool as needed. The mlextend library is commonly used for Association Rule Mining in Python.

7) Visualize all the statistical measures (mean, mode, median, range, inter quartile range, etc) Using Histograms, Boxplots, scatter plots, etc.

1) Write a python program to find the quadrants in which coordinates lies get the value of x and y coordinates as input from the user and check in which quadrants the point lies and print it.

⟹

To visualize statistical measures and find the quadrant in which a set of coordinates lies, you can use python with libraries such as Matplotlib and Numpy for visualization and simple if-else condition to determine the quadrant. Here's Python program that does both tasks:

```
import matplotlib.pyplot as plt
import numpy as np
#sample data for visualization (mean, mode, median, range, IQR)
data = [1,2,3,4,4,5,6,6,6,7,7,8,9,10]
#plot a histogram
plt.figure(figsize=(12,4))
plt.subplot(131)
plt.hist(data, bins=10, edgecolor='black')
plt.title("Histogram")
#plot a boxplot
plt.subplot(132)
plt.boxplot(data, vert=False)
plt.title("Boxplot")
#Generate some sample scatter plot data
x = np.random.randint(-10,10,20)
y = np.random.randint(-10,10,20)
#plot a scatter plot
plt.subplot(133)
```

```
plt.scatter(x, y)
plt.title("Scatter Plot")
plt.show()
# Get user input for the coordinates
x_coord = float(input("Enter the X coordinate: "))
y_coord = float(input("Enter the Y coordinate: "))
# Determine the quadrant
if x_coord > 0 and y_coord > 0:
    quadrant = "Quadrant I"
elif x_coord < 0 and y_coord > 0:
    quadrant = "Quadrant II"
elif x_coord < 0 and y_coord < 0:
    quadrant = "Quadrant III"
elif x_coord > 0 and y_coord < 0:
    quadrant = "Quadrant IV"
else:
    quadrant = "On the origin or one of the axes"
print(f"The point ({x_coord}, {y_coord}) lies in {quadrant}")
```

This program first visualizes statistical measures using a histogram, boxplot and scatter plot. After that, it takes user input for the coordinates (x and y) and determine the quadrant in which the point lies. The program then prints the result.

please note that the sample data for visualization and the scatter plot data are randomly gennated, so you can replace them with your own data as needed.

2) Write a python program to implements Data Visualization kindly refers your own data.
1. Draw a horizontal bar graph
2. Draw scatter plot diagram

⟹

To implement data visualization with your own data, you can use python and libraries such as Matplotlib and Seaborn. I'll provide an example where we visualize statistical measures using various plots, including a horizontal bar graph and a scatter plot.

Here's python program that demonstrates these visualizations with sample data:-

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
# Sample data for visualization
data = [10, 15, 18, 20, 22, 25, 28, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85]
# Calculate statistical measures
mean_value = np.mean(data)
median_value = np.median(data)
mode_value = np.argmax(np.bincount(data))
data_range = np.ptp(data)
q75, q25 = np.percentile(data, [75, 25])
iqr = q75 - q25
# plot a histogram
plt.figure(figsize=(12, 4))
plt.subplot(131)
plt.hist(data, bins=10, edgecolor='black')
plt.title("Histogram")
# plot a boxplot
```

```
plt.subplot(132)
plt.boxplot(data, vert=False)
plt.title("Boxplot")
#Generate some sample scatter plot data
x = np.random.randint(1,100,20)
y = np.random.randint(1,100,20)
# plot a scatter plot
plt.subplot(133)
plt.scatter(x, y)
plt.title("Scatter Plot")
plt.show()
# Draw a horizontal bar graph
plt.figure(figsize=(8,4))
plt.subplot(121)
labels = ['Mean', 'Median', 'Mode', 'Range', 'IQR']
values = [mean_value, median_value, mode_value, data_range, iqr]
plt.barh(labels, values)
plt.xlabel('Values')
plt.title('Statistical Measures')
# Draw a scatter plot diagram
plt.subplot(122)
sns.scatterplot(x=x, y=y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot')
plt.tight_layout()
plt.show()
```

• In this program :→

i) We calculate statistical measures such as mean, median, mode, range and interquantile range using Numpy.

ii) We visualize these measures using histograms, boxplots and as a scatter plot.

iii) We then draw a horizontal bar graph to display the statistical measures, and a scatter plot using Seaborn.

You can replace the sample data with your own dataset to perform these visualizations on your data.

3) Implement Data Visualization kindly refers your own data.
1. Draw a pie chart.
2. Draw a Bar graph.

→

To visualize statistical measures and create data visualizations, such as a pie chart and a bar graph, you can use python with libraries like Matplotlib and NumPy. I'll provide an example using sample data:

Here's a Python program that demonstrates these visualizations:

```
import matplotlib.pyplot as plt
import numpy as np
# Sample data for visualization
data = [5, 15, 20, 30, 10]
# Calculate statistical measures
mean_value = np.mean(data)
median_value = np.median(data)
mode_value = max(set(data), key= data.count )
data_range = max(data) - min(data)
q75, q25 = np. percentile (data, [75, 25])
iqr = q75 - q25
# plot a histogram
plt.figure(figsize=(12,4))
plt.subplot(131)
plt.hist(data, bins= 10, edgecolor = 'black')
plt.title("Histogram")
# plot a boxplot
plt.subplot(132)
plt.boxplot(data ,vert = False)
```

```
plt.title("Boxplot")
#Generate some sample scatter plot data
x = np.random.randint(1,10,5)
y = np.random.randint(1,10,5)
#plot a scatter plot
plt.subplot(133)
plt.scatter(x,y)
plt.title("Scatter Plot")
plt.show()
#Draw a pie chart
labels = ['Mean', 'Median', 'Mode', 'Range', 'IQR']
values = [mean_value,medium_value, mode_value, data_range, iqr]
colors = ['gold', 'lightcoral', 'lightskyblue', 'lightgreen', 'lghpink']
explode = (0.1, 0, 0, 0, 0)
plt.figure(figsize=(8,4))
plt.pie(values, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
plt.title('Statistical Measures Pie Chart')
plt.axis('equal') #Equal aspect ratio ensures that pie is
drawn drawn as a circle.
#Draw a bar graph
plt.figure(figsize=(8,4))
plt.bar(labels, values, color=['gold','lightcoral','lightgreen','lightpink'])
plt.xlabel('Statistical Measures')
plt.ylabel('Values')
plt.title('Statistical Measures Bar Graph')
plt.tight_layout()
plt.show()
```

- In this program :-

1) We calculate statistical measures such as mean, median, mode, range and interquartile

range using NumPy.

ii) We visualize these measures using histogram, boxplots and a scatter plot.

iii) We draw a pie chart to display the statistical measures as percentages.

iv) We also draw a bar graph to represent the values of the statistical measures.

You can replace the sample data with your own dataset to perform these visualizations on your data

4) Implement Data Visualization Kindly refers "income.csv" dataset.
1. Draw a scatter plot for Age and Income.
2. Draw a bar graph for Age and Income.

⟹

To visualize statistical measures and create data visualizations using the "income.csv" dataset, you can use python with libraries such as Matplotlib and Pandas. Here's a python program that demonstrates these visualizations :-

```
import pandas as pd
import matplotlib.pyplot as plt
# Load the "income.csv" dataset
data = pd.read_csv("income.csv")
# Extract Age and Income columns
age = data['Age']
income = data['Income']
# Calculate statistical measures
mean_age = age.mean()
median_age = age.median()
mode_age = age.mode().values[0]
range_age = age.max() - age.min()
q75_age, q25_age = age.quantile([0.75, 0.25])
iqr_age = q75_age - q25_age
mean_income = income.mean()
median_income = income.median()
mode_income = income.mode().values[0]
range_income = income.max() - income.min()
q75_income, q25_income = income.quantile([0.75, 0.25])
iqr_income = q75_income - q25_income
# Plot a scatter plot for Age and Income
plt.figure(figsize=(12,4))
```

```
plt.subplot(121)
plt.scatter(age, income, alpha=0.5)
plt.title("Scatter Plot (Age vs. Income)")
plt.xlabel("Age")
plt.ylabel("Income")
# plot a bar graph for Age and Income
plt.subplot(122)
categories = ['Mean', 'Median', 'Mode', 'Range', 'IQR']
values_age = [mean_income, median_income, mode_income, range_income,
iqr_income]
plt.bar(categories, values_age, alpha=0.5, label='Age', width=0.4)
plt.bar(categories, values_income, alpha=0.5, label='Income',
width=0.4, align='edge')
plt.title("Bar Graph (Statistical Measures)")
plt.xlabel("Statistical Measures")
plt.ylabel("Values")
plt.legend()
plt.tight_layout()
plt.show()
```

- In this program :-

i) We load the "income.csv" dataset using pandas

ii) We extract the Age and Income columns from the dataset.

iii) We calculate statistical measures such as mean, median, mode, range and interquartile range for both Age and Income.

iv) We visualize these measures using a scatter plot for Age and Income, and a bar graph for the statistical measures, with both Age and Income displayed side by side for comparison.

You replace the "income.csv" dataset with your own dataset if needed.