STOCK PRICE PREDICTION

A BASIC GUIDE FOR BEGINNERS

ROHITH | INSIGHT CODE

WINDOW CREATION

Step 1

WHAT IS TKINTER?

Tkinter is a standard GUI (Graphical User Interface) library for Python programming language. It provides a set of tools and widgets to create desktop applications with graphical user interfaces.

WE NEED THREE WINDOWS

WINDOW 3 WINDOW 1 WINDOW 2

OBJECTIVES

WINDOW 1

- Create a homepage window with 2 buttons,each of it linking to its corresponding windows.
- Add background Image to the window.

OBJECTIVES

WINDOW 2

- Create 3 buttons
- On clicking each button the graph corresponding to it should open
- Create a previous button which links to the homepage

OBJECTIVES

WINDOW 3

- Create 10 buttons
- On clicking each button ,the pdf corresponding to it should open
- Create a previous button which links to the homepage

HOW TO CREATE A TKINTER WINDOW?

CODE

```
from tkinter import *
root=Tk()
root.geometry("500x333")
root.title("STOCK PREDICTION")
root.resizable(False,False)
root.mainloop()
```

OUTPUT

WHAT HAPPENS HERE?

- IMPORT THE LIBRARIES
- INITIALIZE THE BASE WINDOW(HERE,ROOT)
- A TKINTER WINDOW HAS VARIOUS ATTRIBUTES SUCH AS TITLE,GEOMETRY,RESIZABLE,ETC...GIVE IT ACCORDING TO YOUR CHOICE.
- MAINLOOP() METHOD- RUNS THE WINDOW INDEFINITELY UNTIL THE USER CLOSES THE WINDOW OR EXITS THE APPLICATION.

SYNTAX

GEOMETRY

```
window.geometry("widthxheight+XPOS+YPOS")
```

TITLE

```
window.title("Window Title")
```

RESIZABLE

```
window.resizable(WIDTH, HEIGHT)
```

HOW TO ADD BACKGROUND IMAGE TO THE WINDOW?

CODE

```
bg_image = PhotoImage(file="dsg3.png")
bg_label = Label(root, image=bg_image)
bg_label.place(x=0, y=0, relwidth=1, relheight=1)
```

OUTPUT

WHAT HAPPENS HERE?

- WE HAVE ADDED BACKGROUND IMAGE HERE
- RESIZE THE IMAGE IN ACCORDANCE WITH YOUR WINDOW SIZE
- IT CAN BE DONE VIA TRIAL AND ERROR USING VARIOUS IMAGE RESIZING TOOLS AVAILABLE ONLINE.
- SET THE BACKGROUND IMAGE USING PHOTOIMAGE()
- LINK THE IMAGE TO THE ROOT WINDOW USING LABEL()
- PLACE IT IN THE WINDOW USING .PLACE() OF THE LABEL.

SYNTAX

PHOTOIMAGE

```
photo = PhotoImage(file="image.gif")
```

LABEL

```
label = Label(parent, options)
```

[ATTRIBUTES INCLUDE TEXT,IMAGE,BG,FG,FONT, HEIGHT,ETC..]

RESIZABLE

```
widget.place(options)
```

[ATTRIBUTES INCLUDE X,Y,WIDTH,HEIGHT,ETC..]

HOW TO ADD BUTTONS?

CODE

```
btn77=Button(root,text="JOURNALS",command=openwindow, font=("Arial", 25),relief='groove',activebackground="blue")
btn77.place(x=55,y=140)
btn3=Button(root,text="STOCKS",command=openwindow2, font=("Arial", 25),relief='groove',activebackground="blue")
btn3.place(x=300,y=140)
```

OUTPUT

WHAT HAPPENS HERE?

- WE CREATED TWO BUTTONS HERE FOR JOURNALS AND STOCKS

SYNTAX

BUTTON

```
button = Button(parent, options)
```

SOME OPTIONS /ATTRIBUTES OF BUTTON ARE:

- text: Specifies the text to display on the button.
- command: Specifies the function to call when the button is clicked.
- bg: Specifies the background color of the button.
- fg: Specifies the foreground (text) color of the button.
- font: Specifies the font to use for the button text.
- width: Specifies the width of the button in characters.
- height: Specifies the height of the button in characters.
- image: Specifies an image to display on the button.

CREATE WINDOW 2 AND WINDOW 3

CODE

```
new_window2=Toplevel()
new_window2.geometry("550x465")
bg_image3 = PhotoImage(file="dsg16 (2).png")
bg_label3= Label(new_window2, image=bg_image3)
bg_label3.place(x=0, y=0, relwidth=1, relheight=1)
new_window2.title("COMPANIES")
new_window2.resizable(False,False)
```

CODE

```
new_window=Toplevel()
new_window.geometry("500x500")
bg_image2 = PhotoImage(file="dsg10.png")
bg_label2= Label(new_window, image=bg_image2)
bg_label2.place(x=0, y=0, relwidth=1, relheight=1)
new_window.title("JOURNALS")
new_window.resizable(False,False)
new_window.attributes("-alpha",0.9)
new_window.geometry("800x560")
```

WHAT HAPPENS HERE?

- REPEAT THE SAME STEPS AS CREATING WINDOW
- HERE,THE DIFFERENCE IS THAT WE INITALIZE THE WINDOW USING TOPLEVEL()
- IN TKINTER, THE TOPLEVEL CLASS IS USED TO CREATE A NEW WINDOW THAT IS INDEPENDENT OF THE MAIN WINDOW. IT IS OFTEN USED TO CREATE DIALOG BOXES, POP-UP WINDOWS, OR SECONDARY WINDOWS THAT ARE DISPLAYED ON TOP OF THE MAIN WINDOW.

FUNCTIONALITIES

DEICONIFY()

Using deiconify() allows you to restore a previously hidden window to its previous state, and allows the user to interact with it again.

WITHDRAW()

Using withdraw() can be useful for hiding windows that are not currently needed, or for freeing up resources when a window is no longer being used.

FUNCTIONALITIES

CODE

```
def openroot():
    root.deiconify()
    new_window.withdraw()
    new_window2.withdraw()

def openwindow():
    new_window.deiconify()
    root.withdraw()

def openwindow2():
    new_window2.deiconify()
    root.withdraw()

def previous():
    new_window.withdraw()
    new_window2.withdraw()
    root.deiconify()
```

These functionalities are used for switching between different windows.

PLACE PREVIOUS BUTTONS IN WINDOW 2 AND WINDOW 3

CODE

```
btn2=Button(new_window,text="Previous",command=previous,relief='sunken')
btn2.pack(side=BOTTOM, padx=10, pady=10)
```

```
btn21=Button(new_window2,text="Previous",command=previous,relief='sunken')
btn21.pack(side=BOTTOM, padx=10, pady=10)
```

HOW TO OPEN A PDF?

WE USE SUBPROCESS MODULE

The subprocess module is used to spawn new processes and interact with them. It provides a way to launch and communicate with external programs and scripts from within a Python script.

TO OPEN A PDF FILE USING SUBPROCESS IN PYTHON, YOU CAN USE THE FOLLOWING SYNTAX:

```
import subprocess
subprocess.Popen(['open', 'file.pdf'])
```

FUNCTIONALITIES

CODE

```
def journal(key):
    if key==1:
        file_path = "OECDPolicy.pdf"
        subprocess.Popen([file_path], shell=True)
    elif key==2:
        file_path = "Vaccine Trial.pdf"
        subprocess.Popen([file_path], shell=True)
    elif key==3:
        file_path = "Brand Equity and covid 19 stock market crash.pdf"
        subprocess.Popen([file_path], shell=True)
    elif key==4:
        file_path = "Covid  impact on emerging stock markets[[dynamic analysis].pdf"
        subprocess.Popen([file_path], shell=True)
    elif key==5:
        file_path ="Impact of covid on E-commerce.pdf"
        subprocess.Popen([file_path], shell=True)
    elif key==6:
        file_path = "Impact on digital company stock return.pdf"
        subprocess.Popen([file_path], shell=True)
    elif key==7:
        file_path = "Impact on Covid on emerging Stocks.pdf"
        subprocess.Popen([file_path], shell=True)
    elif key==8:
        file_path = "Pre and Post Covid impact on stocks.pdf"
        subprocess.Popen([file_path], shell=True)
    elif key == 9:
        file_path = "Apparel Market.pdf"
        subprocess.Popen([file_path], shell=True)
    elif key == 10:
```

- We have defined a functionality named journal here.
- On the basis of the key that the function receives ,it opens corresponding file using subprocess module.

PLACE THE BUTTONS FOR OPENING PDF'S IN THE WINDOW 3

CODE

```
btn10 = Button(new_window, text="E-commerce ", command=lambda:journal(5), font=("Arial", 25),width=15,height=1)
btn10.place(x=50, y=360)
btn11 = Button(new_window, text="Dynamic Analysis ", command=lambda:journal(4), font=("Arial", 25),width=15,height=1)
btn11.place(x=50, y=300)
btn12 = Button(new_window, text="Brand Equity ", command=lambda:journal(3), font=("Arial", 25),width=15,height=1)
btn12.place(x=50, y=240)
btn12 = Button(new_window, text="Vaccine Trial ", command=lambda:journal(2), font=("Arial", 25),width=15,height=1)
btn12.place(x=50, y=180)
btn13= Button(new_window, text="OECDP POLICY", font=("Arial", 25),command=lambda:journal(1),width=15,height=1)
btn13.place(x=50, y=120)
btn14 = Button(new_window, text="Fashion Industry", command=lambda:journal(10), font=("Arial", 25),width=15,height=1)
btn14.place(x=450, y=360)
btn15 = Button(new_window, text="Apparel Market", command=lambda:journal(9), font=("Arial", 25),width=15,height=1)
btn15.place(x=450, y=300)
btn16 = Button(new_window, text="Pre and Post Covid", command=lambda:journal(8), font=("Arial", 25),width=15,height=1)
btn16.place(x=450, y=240)
btn17 = Button(new_window, text="Emerging Stocks ", command=lambda:journal(7), font=("Arial", 25),width=15,height=1)
btn17.place(x=450, y=180)
btn18 = Button(new_window, text="Digital Company ", command=lambda:journal(6), font=("Arial", 25),width=15,height=1)
btn18.place(x=450, y=120)
```

PLACE THE BUTTONS FOR OPENING PDF'S IN THE WINDOW

OUTPUT

WHAT HAPPENS HERE?

- WE HAVE CREATED BUTTONS FOR OPENING 10 PDF'S
- HERE, WE HAVE USED AN ADDITIONAL FUNCTION CALLED LAMBDA IN COMMAND.
- USING LAMBDA FUNCTIONS IN CAN BE USEFUL WHEN YOU NEED TO PASS ARGUMENTS TO A FUNCTION IN A WAY THAT CANNOT BE DONE DIRECTLY THROUGH THE COMMAND OPTION. IT ALLOWS YOU TO CREATE A NEW FUNCTION THAT IS TAILORED TO THE SPECIFIC ARGUMENT VALUES YOU NEED TO PASS.LAMBDA FUNCTION IS AN ANONYMOUS FUNCTION, WHICH MEANS IT DOES NOT HAVE A NAME AND CAN ONLY BE USED IN THE CONTEXT WHERE IT IS DEFINED.

HOW TO ADD BACKGROUND IMAGE TO THE BUTTON?

CODE

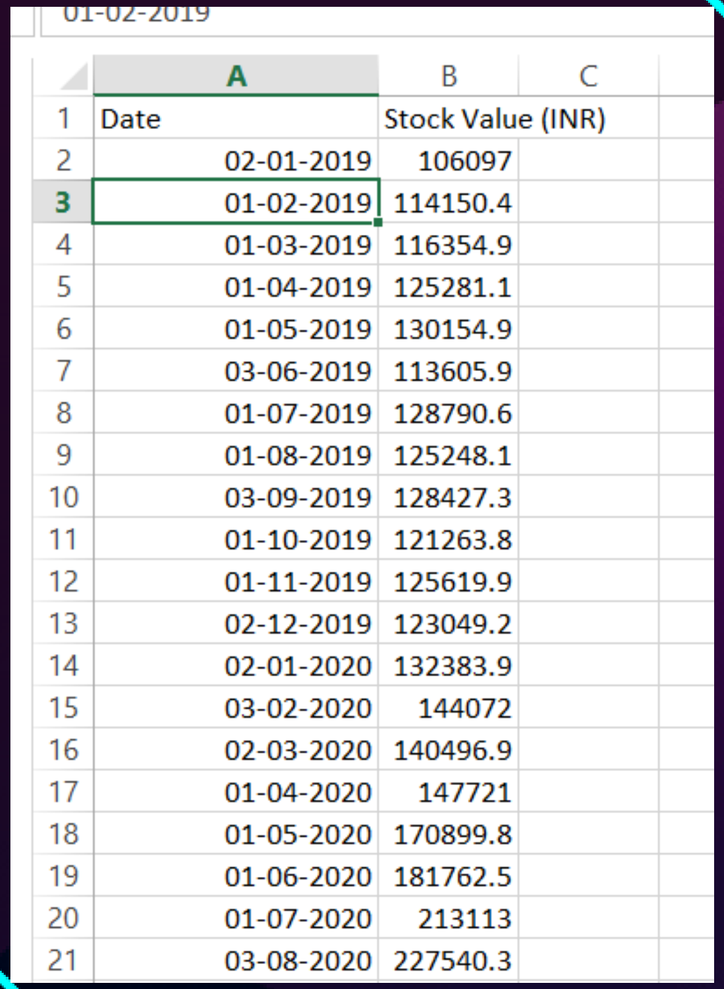
```
image_c1=PhotoImage(file="Puma2.png")
btn22 = Button(new_window2,image=image_c1, width=250,height=60,command=lambda: company("puma.csv"), font=("Arial", 25))
btn22.place(x=0, y=120)
image_c2 = PhotoImage(file="Amazon-India-Logo-PNG-HD2.png")
btn23 = Button(new_window2,image=image_c2,width=250,height=60, command=lambda:company("amazon.csv"), font=("Arial", 25))
btn23.place(x=250, y=240)
image_c3 = PhotoImage(file="sun3.png")
btn24 = Button(new_window2, image=image_c3,width=250,height=60, command=lambda: company("sun.csv"), font=("Arial", 25))
btn24.place(x=0, y=360)
```

OUTPUT

WHAT HAPPENS HERE?

- WE HAVE CREATED THREE BUTTONS JUST LIKE HOW WE DID BEFORE
- HERE,THE DIFFERENCE IS THAT WE HAVE GIVEN IMAGES FOR THE BUTTONS USING PHOTOIMAGE() THAT WE HAVE USED BEFORE

HOW DOES A DATASET LOOK LIKE?



	A	B	C
1	Date	Stock Value (INR)	
2	02-01-2019	106097	
3	01-02-2019	114150.4	
4	01-03-2019	116354.9	
5	01-04-2019	125281.1	
6	01-05-2019	130154.9	
7	03-06-2019	113605.9	
8	01-07-2019	128790.6	
9	01-08-2019	125248.1	
10	03-09-2019	128427.3	
11	01-10-2019	121263.8	
12	01-11-2019	125619.9	
13	02-12-2019	123049.2	
14	02-01-2020	132383.9	
15	03-02-2020	144072	
16	02-03-2020	140496.9	
17	01-04-2020	147721	
18	01-05-2020	170899.8	
19	01-06-2020	181762.5	
20	01-07-2020	213113	
21	03-08-2020	227540.3	

HERE,WE HAVE DATA IN TWO COLUMNS NAMELY

- DATE
- STOCK VALUE(INR)

The dataset actually shows how the stock value changes with time.

WHAT IS NUMPY?

NUMPY (NUMERICAL PYTHON) IS A PYTHON LIBRARY THAT PROVIDES SUPPORT FOR LARGE, MULTI-DIMENSIONAL ARRAYS AND MATRICES, ALONG WITH A COLLECTION OF MATHEMATICAL FUNCTIONS TO OPERATE ON THESE ARRAYS EFFICIENTLY.

np.array: np.array is a function in NumPy that is used to create an array, which is a central data structure in NumPy. It takes a sequence-like object (e.g., a list or tuple) as input and converts it into a NumPy array. NumPy arrays are more efficient than Python lists for numerical operations, as they provide fast and vectorized operations on large datasets.

np.resize: np.resize is a function in NumPy that is used to resize an existing array to a specified shape. It takes two arguments: the array to be resized and the new desired shape.The np.resize function returns a new array with the resized shape.

WHAT IS PANDAS?

PANDAS IS A POPULAR OPEN-SOURCE DATA MANIPULATION AND ANALYSIS LIBRARY FOR PYTHON. IT PROVIDES HIGH-PERFORMANCE, EASY-TO-USE DATA STRUCTURES, SUCH AS DATAFRAMES AND SERIES, ALONG WITH A WIDE RANGE OF FUNCTIONS FOR DATA CLEANING, PREPARATION, EXPLORATION, AND ANALYSIS.

DataFrame: The DataFrame is a two-dimensional tabular data structure in Pandas, similar to a table in a relational database or a spreadsheet. It allows you to store and manipulate data in rows and columns, where each column can have a different data type.

Data Exploration and Analysis: Pandas provides a wide range of functions for data exploration and analysis, including descriptive statistics, aggregations, filtering, slicing, and visualization. It enables quick and comprehensive data exploration to gain insights and make data-driven decisions.

HOW TO LOAD A DATASET?

CODE

```
data = pd.read_csv(file)
```

SYNTAX

```
IMPORT PANDAS AS PD
DATA = PD.READ_CSV('DATASET.CSV')
```

CODE

```
data['Date'] = (data['Date'] - pd.Timestamp("1970-01-01")) // pd.Timedelta('1s')
```

WHY

USED TO CONVERT THE DATES IN A PANDAS DATAFRAME COLUMN TO UNIX TIMESTAMPS.

HOW TO SELECT RANGES OF DATA?

CODE

```
x_test = np.array(data['Date'].iloc[17:21]).reshape(-1, 1)
x = np.array(data['Date'].iloc[:21]).reshape(-1, 1)
y = np.array(data['Stock Value (INR)'].iloc[:21]).reshape(-1, 1)
```

EXPLANATION

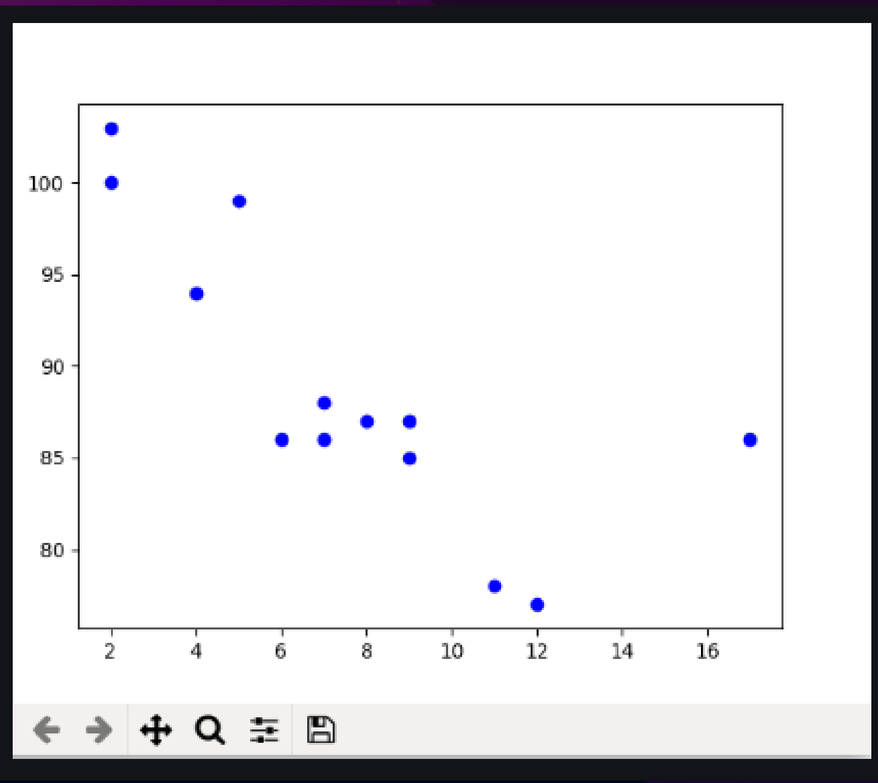
- WE SELECT RANGES OF DATA USING ILOC
- THE ILOC INDEXER IS USED IN PANDAS TO SELECT ROWS AND COLUMNS FROM A DATAFRAME BASED ON THEIR INTEGER POSITIONS. IT ALLOWS YOU TO ACCESS DATA BY ITS POSITION RATHER THAN BY LABEL.
- THE SYNTAX OF ILOC IS AS FOLLOWS:
- DATAFRAME.ILOC[ROW_INDEXER, COLUMN_INDEXER]

PLOTTING & SCATTERING IN GRAPHS

WE USE MATPLOTLIB LIBRARY IN PYTHON FOR PLOTTING GRAPHS.

Scattering:

```
import matplotlib.pyplot as plt
x =[5, 7, 8, 7, 2, 17, 2, 9, 4, 11,
12, 9, 6]
y =[99, 86, 87, 88, 100, 86,103, 87,
94, 78, 77, 85, 86]
plt.scatter(x, y, c ="blue")
# To show the plot
plt.show()
```

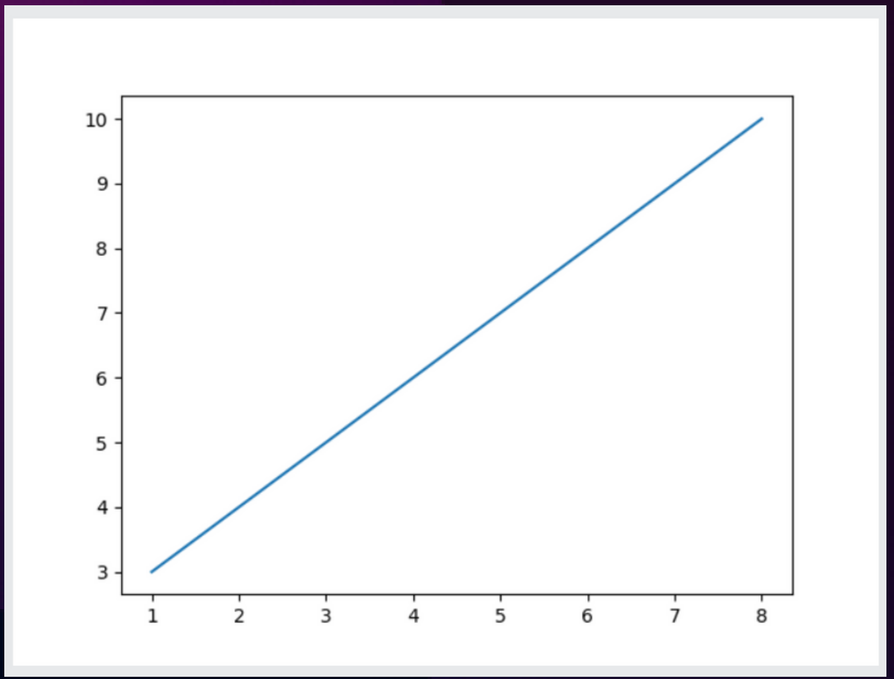


PLOTTING & SCATTERING IN GRAPHS

WE USE MATPLOTLIB LIBRARY IN PYTHON FOR PLOTTING GRAPHS.

Plotting:

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints, ypoints)
plt.show()
```



TRAINING & TESTING DATA

CODE

```
x_test = np.array(data['Date'].iloc[17:21]).reshape(-1, 1)
x = np.array(data['Date'].iloc[:21]).reshape(-1, 1)
y = np.array(data['Stock Value (INR)'].iloc[:21]).reshape(-1, 1)
```

EXPLANATION

- WE DEFINE THE TRAINING AND TESTING DATA.
- OUR AIM IS TO PREDICT THE STOCK VALUES OF 3 DATES SO WE INCLUDE IT IN X_TEST,WHICH IS THE TESTING DATA
- IN X AND Y WE SET ALL THE DATASET VALUES

PREDICTION

WE USE LINEAR REGRESSION FOR MAKING PREDICTIONS.

Linear Regression:

Linear regression is a statistical modeling technique used to establish a linear relationship between a dependent variable and one or more independent variables.

```
LR = LinearRegression()
LR.fit(x, y)
yp_test = LR.predict(x_test)
```

4.TOOLS AND TECHNOLOGY

ERANAD KNOWLEDGE CITY

Doc2vec:

- Define a set of keywords, positive words, negative words most likely to appear in the documents we choose.
- Check if the predefined keywords appear in a given document.
- If a keyword is found, find the word preceding it in the document.
- Use NLTK to POS tag the preceding word to determine its part of speech.

4.TOOLS AND TECHNOLOGY

ERANAD KNOWLEDGE CITY

- Check if the word belongs to a list of positive or negative words.
- Give a score of +1 if the word is positive, -1 if it's negative, and 0 if it doesn't belong to any list.
- Repeat the process for all documents.
- Add the scores of all documents to give a total score.

```python
def get_score(file):
    pdf_file = open(file, 'rb')
    pdf_reader = PyPDF2.PdfReader(pdf_file)
    keyword_list = []
    num_pages = len(pdf_reader.pages)

    for page in range(num_pages):
        page_obj = pdf_reader.pages[page]
        text = page_obj.extract_text()
        words = text.split()
        keywords = ['cases', 'effect', 'effects', 'corelated', 'related', 'dynamism', 'scope', 'costs', 'financial',
                    'in', 'stock', 'risk', 'cash', 'impact', 'impact', 'emotions', 'effect', 'effects', 'relationship',
                    'emotions', 'correlation', 'in demand', 'effects', 'impacts', 'aspects', 'response']

        for i, word in enumerate(words):
            if word in keywords and i > 0:
                keyword_list.append((words[i - 1], word))


    new_word_list = [word.lower() for (word, pos) in nltk.pos_tag([word for (word, pos) in keyword_list]) if
                     pos.startswith('V') or pos.startswith('J') or pos.startswith('RB') or pos.startswith(
                         'VBD') or pos.startswith('VB')]


    positive_words = ['positive', 'disruptive', 'Positive', 'beneficial', 'higher', 'stable', 'increase', 'Increase',
                      'enhanced', 'increasing', 'enhancing', 'positively']
    negative_words = ['decrease', 'negative', 'Negative', 'detrimental', 'idiosyncratic', 'decreasing', 'confirmed',
                      'lower']
```

4.TOOLS AND TECHNOLOGY

ERANAD KNOWLEDGE CITY

Doc2vec:

- Define a set of keywords, positive words, negative words most likely to appear in the documents we choose.
- Check if the predefined keywords appear in a given document.
- If a keyword is found, find the word preceding it in the document.
- Use NLTK to POS tag the preceding word to determine its part of speech.

4.TOOLS AND TECHNOLOGY

ERANAD KNOWLEDGE CITY

- Check if the word belongs to a list of positive or negative words.
- Give a score of +1 if the word is positive, -1 if it's negative, and 0 if it doesn't belong to any list.
- Repeat the process for all documents.
- Add the scores of all documents to give a total score.

```
score = 0
count=0
for word in new_word_list:
    count=count+1
    if word in positive_words:
        score += 1
    elif word in negative_words:
        score -= 1
pdf_file.close()
return score,count
```

- **Feature Vector Generation:** Our process involves performing Doc2vec on multiple documents and assigning them scores, which are then averaged. After that, we create feature vectors by utilizing a formula that utilizes the scores, allowing us to effectively represent the semantic meaning of a document.

```
def get_list_score(file_list):
    total_score = 0
    for file in file_list:
        score,count = get_score(file)
        avg_score=score/count
        total_score += avg_score
    return total_score
```

```
list1_score = get_list_score(list1)
list2_score = get_list_score(list2)
list3_score = get_list_score(list3)
print("E-COMMERCE avg Score:", list1_score / 5)
e_commerce = list1_score / 5
print("PHARMA  avg Score:", list2_score / 2)
pharma = get_list_score(list2)
print("FASHION avg Brand", list3_score / 5)
fashion = get_list_score(list2)
```

```python
def get_score(file):
    pdf_file = open(file, 'rb')
    pdf_reader = PyPDF2.PdfReader(pdf_file)
    keyword_list = []
    num_pages = len(pdf_reader.pages)

    for page in range(num_pages):
        page_obj = pdf_reader.pages[page]
        text = page_obj.extract_text()
        words = text.split()
```

- FIRST WE OPEN THE PDF FILE.
- WE READ THE CONTENTS OF PDF FILE USING PDFREADER WHICH IS AVAILABLE IN THE LIBRARY PYPDF2.
- CREATE AN EMPTY KEYWORD LIST
- USING A LOOP WE READ CONTENTS OF EACH PAGE IN A PDF,SPLIT IT AND PLACE EACH CONTENT IN 'WORDS'

```
    for i, word in enumerate(words):
        if word in keywords and i > 0:
            keyword_list.append((words[i - 1], word))


new_word_list = [word.lower() for (word, pos) in nltk.pos_tag([word for (word, pos) in keyword_list]) if
                 pos.startswith('V') or pos.startswith('J') or pos.startswith('RB') or pos.startswith(
                     'VBD') or pos.startswith('VB')]
```

[The enumerate() function is commonly used when you need to iterate over a sequence and also require the corresponding index value. It eliminates the need for maintaining a separate counter variable and provides a concise way to access both the index and value in a loop.]

- USING LOOP WE CHECK IF ANY WORD IN THE PDF MATCHES OUR DEFINED SET OF KEYWORDS.
- IF YES APPEND THAT TO THE KEYWORD_LIST ALONG WITH THE WORD PRECEEDING IT.
- IN A NEW LIST-NEW_WORD_LIST INCLUDE ONLY THOSE WORDS WHOSE PRECEEDING WORD IS A VERB/ADVERB/ADJECTIVE ETC...
- WE CAN KNOW IT BY POS TAGGING THE PRECEEDING WORDS.
- FOR UNIFORMITY CONVERT ALL WORDS TO LOWER CASE BEFORE APPENDING IT TO THE LIST.

[.lower() fuction is used for converting given text to lower case.]

STEP 04

- **Feature Vector Generation:** Our process involves performing Doc2vec on multiple documents and assigning them scores, which are then averaged. After that, we create feature vectors by utilizing a formula that utilizes the scores, allowing us to effectively represent the semantic meaning of a document.

```
y_s=[]
for i in range(3):
    n=yp_test[i]
    y_s.append(n)


y_s = np.array(y_s)
y_d1 = y_s * (1 + e_commerce)

y_d2= y_s * (1 + pharma)
y_d3 = y_s * (1 + fashion)
```

```
y_s=[]
for i in range(3):
    n=yp_test[i]
    y_s.append(n)
y_s = np.array(y_s)
y_d1 = y_s * (1 + e_commerce)

y_d2= y_s * (1 + pharma)
y_d3 = y_s * (1 + fashion)
```

- **Feature Vector Generation:** Our process involves performing Doc2vec on multiple documents and assigning them scores, which are then averaged. After that, we create feature vectors by utilizing a formula that utilizes the scores, allowing us to effectively represent the semantic meaning of a document.

- DEFINE AN EMPTY LIST.
- FOR EACH FILE GET THE LAST 3 PREDICTED VALUES FROM LR AND PLACE IT IN A LIST.
- APPLY IT TO THE DEVISED FORMULA TO GET THE FINAL PLOTTING VALUE.


THANK YOU
I HOPE YOU LEARNED SOMETHING NEW!