1. Create two numpy arrays and find the cosine similarity between the arrays. Do not use any packages

What is Cosine Similarity?

Cosine similarity measures the angle between two vectors in a multi-dimensional space.

$$\text{Cosine Similarity}(A, B) = \cos(\theta) = \frac{A \cdot B}{|A|\,|B|}$$

```
A = [1,2,3]       #create Two Arrays
B = [2,4,6]
```

Dot Product

$$A \cdot B = \sum_i A_i B_i$$

```
def dot_product(vec1,vec2):                 # create a function for dot produc
  return sum(a*b for a,b in zip(vec1,vec2))
```

Vector Mangnitude (Norm)

$$|A| = \sqrt{\sum_i A_i^2}$$

```
def norm(vec):                              # create a function for norm
  return sum(x*x for x in vec)**0.5
```

Cosine Similarity

```
def cosine_similarity (vec1,vec2):          # Define cosine function where we
  numerator = dot_product(vec1 , vec2)
  denominator = norm(vec1) * norm(vec2)
  return numerator / denominator
```

```
result = cosine_similarity(A,B)
print("Cosine Similarity :", result)
```

```
Cosine Similarity : 1.0
```

```
x = [1,5,3]
y = [7,2,8]
result = cosine_similarity(x,y)
print("Cosine Simalarity between X and y :",result)
```

```
Cosine Simalarity between X and y : 0.6407032156159437
```

2. Numpy basic data structures and operations

```
import numpy as np
```

- Create two numpy arrays 'numbers1' and 'numbers2'

```
numbers1 = np.array([1,2,3,4,5])   # creating Numpy array
numbers2 = np.array([6,7,8,9,10])
```

- Find the sum of the two arrays

```
sum = numbers1 + numbers2            # both are numpy array so we can add it di
print("Sum of two arrays :",sum)
```

```
Sum of two arrays : [ 7  9 11 13 15]
```

- Find the dot product of the two arrays

```
dot_product = np.dot(numbers1,numbers2)              # finding dot product
print("Dot product of two arrays :",dot_product)
```

```
Dot product of two arrays : 130
```

- Find the mean of the array

```
mean1 = np.mean(numbers1)              # Finding a mean Value
print("Mean of array :",mean1)
```

```
Mean of array : 3.0
```

```
mean2 = np.mean(numbers2)
print("Mean of array :",mean2)
```

```
Mean of array : 8.0
```

- Create another 2D array 'new_numbers' from numbers1 and numbers2 using np.stack. What is the shape of the new array? How does np.stack work?

```
new_numbers = np.stack((numbers1,numbers2))     # creating a stack
print("Stack of two arrays : \n",new_numbers)
```

```
Stack of two arrays :
 [[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
```

```
shape = new_numbers.shape
print("Shape of new_numbers :",shape)
```

```
Shape of new_numbers : (2, 5)
```

- Transpose the new 2D. What is the result?

```
transposed = np.transpose(new_numbers)             # form a Transpose of stack
print(" Transposed new_numbers: \n",transposed)
print("Shape of Transpose : \n",transposed.shape)
```

```
 Transposed new_numbers:
 [[ 1  6]
 [ 2  7]
 [ 3  8]
 [ 4  9]
 [ 5 10]]
Shape of Transpose :
 (5, 2)
```

3.Pandas dataframes and basic operations using a open dataset

```
import pandas as pd
```

3.1 Operations on User Created Custom Data Frame

- Create a dataframe using the below command: data = pd.DataFrame({'Qu1': [1, None, 4, 3, 4], 'Qu2': [2, None, 1, 2, 3], 'Qu3': [1, 5, np.NaN, 4, 4]})

```
# creating Data Frame
data =  pd.DataFrame({
    'Qu1':[1,None,4,3,4],
    'Qu2':[2,None,1,2,3],
    'Qu3':[1,5,np.nan,4,4]
})
print(data)
```

```
   Qu1  Qu2  Qu3
0  1.0  2.0  1.0
1  NaN  NaN  5.0
2  4.0  1.0  NaN
3  3.0  2.0  4.0
4  4.0  3.0  4.0
```

- Check for missing values

```
print(data.isnull())     # Finding where are the missing values
```

```
     Qu1    Qu2    Qu3
0  False  False  False
1   True   True  False
2  False  False   True
3  False  False  False
4  False  False  False
```

- Delete the rows with missing value in any one column/attribute

```
cleaned_any = data.dropna()   # drop if any columns have nan value
print(cleaned_any)
```

```
   Qu1  Qu2  Qu3
0  1.0  2.0  1.0
3  3.0  2.0  4.0
4  4.0  3.0  4.0
```

- Delete the row only if all columns have missing values

```
clean_all = data.dropna(how='all')   # we don't have any row with all values mi
print(clean_all)
```

```
   Qu1  Qu2  Qu3
0  1.0  2.0  1.0
1  NaN  NaN  5.0
2  4.0  1.0  NaN
3  3.0  2.0  4.0
4  4.0  3.0  4.0
```

- Fill the missing values with zeros

```
fill_zero = data.fillna(0)
print(fill_zero)
```

```
   Qu1  Qu2  Qu3
0  1.0  2.0  1.0
1  0.0  0.0  5.0
```

```
2  4.0  1.0  0.0
3  3.0  2.0  4.0
4  4.0  3.0  4.0
```

- Fill missing values with constant one for each column

```
fill_constant = data.fillna({'Qu1':10,'Qu2':20,'Qu3':30})
print(fill_constant)
```

```
    Qu1   Qu2   Qu3
0   1.0   2.0   1.0
1  10.0  20.0   5.0
2   4.0   1.0  30.0
3   3.0   2.0   4.0
4   4.0   3.0   4.0
```

- Fill missing values withe mean value of the column

```
fill_mean = data.fillna(data.mean())
print(fill_mean)
```

```
   Qu1  Qu2  Qu3
0  1.0  2.0  1.0
1  3.0  2.0  5.0
2  4.0  1.0  3.5
3  3.0  2.0  4.0
4  4.0  3.0  4.0
```

3.2 Analysis of an Open Dataset using Pandas

- Load the built in dataset "Titanic" using the code below

```
import seaborn as sns
```

```
df = sns.load_dataset('titanic')
#upload the data in colab and load the data.
```

- Display the head of the dataset"?

```
print(df.head)       # Head of dataset
```

```
<bound method NDFrame.head of      survived  pclass     sex   age  sibsp  parch
0           0       3    male  22.0      1      0   7.2500       S   Third
1           1       1  female  38.0      1      0  71.2833       C   First
2           1       3  female  26.0      0      0   7.9250       S   Third
3           1       1  female  35.0      1      0  53.1000       S   First
4           0       3    male  35.0      0      0   8.0500       S   Third
..        ...     ...     ...   ...    ...    ...      ...     ...     ...
```

```
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
888         0       3  female   NaN      1      2  23.4500        S   Third
889         1       1    male  26.0      0      0  30.0000        C   First
890         0       3    male  32.0      0      0   7.7500        Q   Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     ...         ...  ...          ...   ...    ...
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]>
```

- Print the shape of the DataFrame?

```
print(df.shape)  # shape of data
```

```
(891, 15)
```

- Display columns and their data types?

```
print(df.dtypes)  # dtypes shows all data types
```

```
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

- Check for missing values in each column?

```
missing_val = df.isnull()   # If there are missing values then it will True
print(missing_val)
```

```
     survived  pclass    sex    age  sibsp  parch   fare  embarked  class  \
0       False   False  False  False  False  False  False     False  False   
1       False   False  False  False  False  False  False     False  False   
2       False   False  False  False  False  False  False     False  False   
3       False   False  False  False  False  False  False     False  False   
4       False   False  False  False  False  False  False     False  False   
..        ...     ...    ...    ...    ...    ...    ...       ...    ...   
886     False   False  False  False  False  False  False     False  False   
887     False   False  False  False  False  False  False     False  False   
888     False   False  False   True  False  False  False     False  False   
889     False   False  False  False  False  False  False     False  False   
890     False   False  False  False  False  False  False     False  False   

       who  adult_male   deck  embark_town  alive  alone  
0    False       False   True        False  False  False  
1    False       False  False        False  False  False  
2    False       False   True        False  False  False  
3    False       False  False        False  False  False  
4    False       False   True        False  False  False  
..     ...         ...    ...          ...    ...    ...  
886  False       False   True        False  False  False  
887  False       False  False        False  False  False  
888  False       False   True        False  False  False  
889  False       False  False        False  False  False  
890  False       False   True        False  False  False  

[891 rows x 15 columns]
```

- Show only the first 10 rows?

```
print(df.head(10))  # first 10 Rows
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0         0       3    male  22.0      1      0   7.2500        S   Third   
1         1       1  female  38.0      1      0  71.2833        C   First   
2         1       3  female  26.0      0      0   7.9250        S   Third   
3         1       1  female  35.0      1      0  53.1000        S   First   
4         0       3    male  35.0      0      0   8.0500        S   Third   
5         0       3    male   NaN      0      0   8.4583        Q   Third   
6         0       1    male  54.0      0      0  51.8625        S   First   
7         0       3    male   2.0      3      1  21.0750        S   Third   
8         1       3  female  27.0      0      2  11.1333        S   Third   
9         1       2  female  14.0      1      0  30.0708        C  Second   

     who  adult_male deck  embark_town alive  alone  
0    man        True  NaN  Southampton    no  False  
1  woman       False    C    Cherbourg   yes  False  
2  woman       False  NaN  Southampton   yes   True  
3  woman       False    C  Southampton   yes  False  
4    man        True  NaN  Southampton    no   True  
5    man        True  NaN   Queenstown    no   True  
6    man        True    E  Southampton    no   True  
7  child       False  NaN  Southampton    no  False  
8  woman       False  NaN  Southampton   yes  False  
9  child       False  NaN    Cherbourg   yes  False  
```

- What is the average age of passengers?

```
# Method 1              # TAKING Mean
print("Average age of Passengers: ",df['age'].mean())

# Method 2
average_age = df['age'].mean()
print(f"Average age: {average_age:0.2f}")
```

```
Average age of Passengers:  29.69911764705882
Average age: 29.70
```

- What is the total fare paid?

```
print("Total Fare Paid = ",df['fare'].sum())     # Total fair
```

```
Total Fare Paid =  28693.9493
```

- What is the age range (min and max)?

```
print("Age Range (min and max)\n")       # Min Max range of Age
print(f"{df['age'].min()} to {df['age'].max()}")
```

```
Age Range (min and max)

0.42 to 80.0
```

- Show passengers who survived and were female?

```
print("Female passenger who servived: \n")         # Survived Female Passengers
print(df[(df['sex']=='female') & (df['survived']==1)])
```

```
Female passenger who servived: 

     survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
1           1       1  female  38.0      1      0  71.2833        C   First   
2           1       3  female  26.0      0      0   7.9250        S   Third   
3           1       1  female  35.0      1      0  53.1000        S   First   
8           1       3  female  27.0      0      2  11.1333        S   Third   
9           1       2  female  14.0      1      0  30.0708        C  Second   
..        ...     ...     ...   ...    ...    ...      ...      ...     ...   
874         1       2  female  28.0      1      0  24.0000        C  Second   
875         1       3  female  15.0      0      0   7.2250        C   Third   
879         1       1  female  56.0      0      1  83.1583        C   First   
880         1       2  female  25.0      0      1  26.0000        S  Second   
887         1       1  female  19.0      0      0  30.0000        S   First   

       who  adult_male deck  embark_town alive  alone  
1    woman       False    C    Cherbourg   yes  False  
2    woman       False  NaN  Southampton   yes   True  
3    woman       False    C  Southampton   yes  False  
```

```
8    woman       False  NaN  Southampton   yes  False
9    child       False  NaN    Cherbourg   yes  False
..     ...         ...  ...          ...   ...    ...
874  woman       False  NaN    Cherbourg   yes  False
875  child       False  NaN    Cherbourg   yes   True
879  woman       False    C    Cherbourg   yes  False
880  woman       False  NaN  Southampton   yes  False
887  woman       False    B  Southampton   yes   True

[233 rows x 15 columns]
```

- How many passengers were children (age < 12)?

```
# Method 1
print("Number of Children :",len(df[df['age']<12]))

# Methos 2
print("Number of Children :",(df[df['age']<12].shape))
```

```
Number of Children : 68
Number of Children : (68, 15)
```

- How many missing values are there?

```
print(df.isnull().sum().sum())   # we adding the Number of Missing Values of all
```

```
869
```

```
Start coding or generate with AI.
```