GOVERNMENT OF KERALA

DEPARTMENT OF TECHNICAL EDUCATION

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(GOVT. ENGINEERING COLLEGE)

KOTTAYAM - 686501

RECORD BOOK

GOVERNMENT OF KERALA

DEPARTMENT OF TECHNICAL EDUCATION

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(GOVT. ENGINEERING COLLEGE)

KOTTAYAM - 686501

20MCA241 DATA SCIENCE LAB

Name: SREERAG N V

Branch: Master of Computer Applications

Semester: 3

Roll No: 54

CERTIFIED BONAFIDE RECORD WORK DONE BY

Reg No. ....

STAFF IN CHARGE

INTERNAL EXAMINER EXTERNAL EXAMINER

Contents

Assignment 1
Review of python programming

Write python code to explore and practice with the basic data types, containers, functions and classes of python. Start by creating variables of various numeric data types and assigning them values. Print the data types and values of these variables, perform mathematical operations on them, and update their values. Next, create boolean variables with True or False values, print their data types, and perform Boolean operations on them. Moving on to strings, create string variables with text values, print their contents and lengths, concatenate strings, and format them with variables. Use string methods to manipulate strings by capitalizing, converting to uppercase, justifying, centering, replacing substrings, and stripping whitespace.

1.1 Basic data types

1.1.1 Numbers

```
# Your Python code here
print("Hello, world!")
print(x + 1)   # Addition
print(x - 1)   # Subtraction
print(x * 2)   # Multiplication
print(x ** 2)  # Exponentiation
```

```
Hello, world! 7   5   14  49
```

1.1.2 Booleans

```
t, f = True, False
print(type(t))
print(t and f) # Logical AND;
print(t or f)  # Logical OR;
print(not t)   # Logical NOT;
print(t != f)  # Logical XOR;
```

```
<class ’bool’>
False True False True
```

1.1.3 Strings

```
hello = 'hello'
world = "world"
print(hello, len(hello))
hw = hello + ' ' + world  # String concatenation
print(hw)
hw12 = '{} {} {}'.format(hello, world, 12)  # string formatting
print(hw12)
s = "hello"
print(s.capitalize())
print(s.upper())
```

```python
print(s.rjust(7))
print(s.center(7))
print(s.replace('l', '(ell)'))
print('  world '.strip())
```

```
hello 5
hello world
hello world 12
Hello
HELLO
  hello
 hello
he(ell)(ell)o
world
```

1.2 Containers

1.2.1 Lists

```python
xs = [3, 1, 2]
print(xs, xs[2])
print(xs[-1])
xs[2] = 'foo'
print(xs)
xs.append('bar')
print(xs)
x = xs.pop()
print(x, xs)
```

```
[3, 1, 2] 2
2
[3, 1, 'foo']
[3, 1, 'foo', 'bar']
foo [3, 1]
```

1.2.2 Slicing

```python
nums = list(range(5))
print(nums)
print(nums[2:4])
print(nums[2:])
print(nums[:2])
print(nums[:])
print(nums[:-1])
nums[2:4] = [8, 9] print(nums)
```

```
[0, 1, 2, 3, 4]
[2, 3]
[2, 3, 4]
[0, 1]
[0, 1, 2, 3, 4]
[0, 1, 2, 3]
[0, 1, 8, 9, 4]
```

1.2.3 Loops

```python
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
```

```
cat
dog
monkey
```

1.2.4 List comprehensions

```python
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print(squares)
```

```
[0, 1, 4, 9, 16]
```

1.2.5 Dictionaries

```python
d = {'cat': 'cute', 'dog': 'furry'}
print(d['cat'])
print('cat' in d)
d['fish'] = 'wet'
print(d['fish'])
```

```
cute
True
wet
```

1.2.6 Sets

```python
animals = {'cat', 'dog'}
print('cat' in animals)
print('fish' in animals)
animals.add('cat')
print(len(animals))
animals.remove('cat')
print(len(animals))
```

```
True
False
3
2
```

1.2.7 Tuples

```python
d = {(x, x + 1): x for x in range(10)}
t = (5, 6)
print(type(t))
print(d[t])
print(d[(1, 2)])
```

```
<class 'tuple'>
5
1
```

1.3 Functions

```python
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'
for x in [-1, 0, 1]:
    print(sign(x))
```

```
negative
zero
positive
```

1.4 Classes

```python
class Greeter:
    def __init__(self, name):
        self.name = name
    def greet(self, loud=False):
        if loud:
          print('HELLO, {}'.format(self.name.upper()))
        else:
          print('Hello, {}!'.format(self.name))
g = Greeter('Fred')
g.greet()
g.greet(loud=True)
```

```
Hello, Fred!
HELLO, FRED
```

Assignment 2
Matrix Operations

Write a program to explore matrix operations using Python's Numpy (or Scipy) libraries.Your program should start by demonstrating the creation of Numpy arrays, including initialization from Python lists and element access methods. Implement various array indexing techniques such as slicing, integer indexing, and boolean indexing, showcasing how they differ in practical applications. Next, create a program that performs element-wise mathematical operations, such as addition, subtraction, multiplication, and division, on Numpy arrays. Investigate broadcasting and create a function to simplify operations involving arrays of different shapes. Additionally, your program should include functionalities to reshape matrices, transpose them, and apply arithmetic operations. Provide clear code examples for each part of the program to illustrate these concepts effectively.

2.1 Numpy Arrays

```python
import numpy as np
a = np.array([1, 2, 3])  # Create a rank 1 array
print(type(a), a.shape, a[0], a[1], a[2])
a[0] = 5                 # Change an element of the array
print(a)
b = np.array([[1,2,3],[4,5,6]])   # Create a rank 2 array
print(b)
print(b.shape)
a = np.zeros((2,2))
print(a)
c = np.full((2,2), 7)
print(c)
d = np.eye(2)
print(d)
```

```
<class 'numpy.ndarray'> (3,) 1 2 3
[5 2 3]
[[1 2 3]
 [4 5 6]]
 (2, 3)
 [[0. 0.]
 [0. 0.]]
 [[7 7]
 [7 7]]
 [[1. 0.]
 [0. 1.]]
```

2.2 Array indexing

```python
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
b = a[:2, 1:3]
print(b)
```

```python
row_r1 = a[1, :]    # Rank 1 view of the second row of a
row_r2 = a[1:2, :]  # Rank 2 view of the second row of a
row_r3 = a[[1], :]  # Rank 2 view of the second row of a
print(row_r1, row_r1.shape)
print(row_r2, row_r2.shape)
print(row_r3, row_r3.shape)
a = np.array([[1,2], [3, 4], [5, 6]])
bool_idx = (a > 2)
print(bool_idx)
```

```
[[2 3]
 [6 7]]
 [5 6 7 8] (4,)
[[5 6 7 8]] (1, 4)
[[5 6 7 8]] (1, 4)
[[False False]
 [ True  True]
 [ True  True]]
```

2.3 Datatypes

```python
x = np.array([1, 2])  # Let numpy choose the datatype
y = np.array([1.0, 2.0])  # Let numpy choose the datatype
z = np.array([1, 2], dtype=np.int64)  # Force a particular datatype
print(x.dtype, y.dtype, z.dtype)
```

```
int64 float64 int64
```

2.4 Array math

```python
x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)
print(x + y)
print(np.add(x, y))
print(x - y)
print(np.subtract(x, y))
print(x * y)
print(np.multiply(x, y))
print(x / y)
print(np.divide(x, y))
print(np.sqrt(x))
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])
v = np.array([9,10])
w = np.array([11, 12])
print(v.dot(w))
print(np.dot(v, w))
```

```python
x = np.array([[1,2],[3,4]])
print(np.sum(x))  # Compute sum of all elements; prints "10"
print(np.sum(x, axis=0))  # Compute sum of each column; prints "[4 6]"
print(np.sum(x, axis=1))  # Compute sum of each row; prints "[3 7]"
print(x)
print("transpose\n", x.T)
```

```
[[ 6.  8.]
 [10. 12.]]
[[ 6.  8.]
 [10. 12.]]
 [[-4. -4.]
 [-4. -4.]]
[[-4. -4.]
 [-4. -4.]]
 [[ 5. 12.]
 [21. 32.]]
[[ 5. 12.]
 [21. 32.]]
 [[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
 [[1.         1.41421356]
 [1.73205081 2.        ]]
 219
219
10
[4 6]
[3 7]
[[1 2]
 [3 4]]
transpose
 [[1 3]
 [2 4]]
```

2.5 Broadcasting

```python
import numpy as np
array1 = np.array([1, 2, 3])  # Shape (3,)
array2 = np.array([4])        # Shape (1,)
result1 = array1 + array2
```

```python
array3 = np.array([1, 2, 3])  # Shape (3,)
array4 = np.array([4, 5])     # Shape (2,)
try:
    result2 = array3 + array4
except ValueError as e:
    result2 = str(e)
print("Broadcasting is done (Array 1 and Array 2):")
print("Result:",result1)
print("Broadcasting cannot be done (Array 3 and Array 4):")
print("Result (Error message):",result2)
```

```
Broadcasting is done (Array 1 and Array 2):
Result: [5 6 7]
Broadcasting cannot be done (Array 3 and Array 4):
Result (Error message): operands could not be broadcast together with shapes (3,) (2,)
```

Assignment 3
Explore, Clean, and Transform a Retail Dataset

Write code to perform the following tasks using Python and Pandas. Present your findings in a well-organized Jupyter Notebook or a Python script, with clear code comments and explanations.

3.1 Create a Pandas DataFrame object using the given dataset

```
import pandas as pd
data=pd.read_csv('reviews.csv')
df=pd.DataFrame(data);
print(df)
```

```
    customer_id  product_id customer_name product_name  rating  \
0             1         101         Alice    Product A     4.0
1             2         102           Bob    Product B     5.0
2             3         103       Charlie    Product A     3.0
3             4         104         David    Product C     5.0
4             5         105           Eve    Product B     2.0
5             6         106         Frank    Product A     4.0
    order_quantity  is_returned payment_method  total_price
0                2            0    Credit Card        59.98
1                1            0         PayPal        29.99
2                3            1     Debit Card        89.97
3                1            0    Credit Card        44.99
4                2            0    Credit Card        69.98
5                1            0         PayPal        29.99
```

3.2 Display the first five rows and last three rows of the dataset.

```
print(df.head(5))
print(df.tail(3))
```

```
  customer_id  product_id customer_name product_name  rating  \
0           1         101         Alice    Product A     4.0
1           2         102           Bob    Product B     5.0
2           3         103       Charlie    Product A     3.0
3           4         104         David    Product C     5.0
4           5         105           Eve    Product B     2.0
                                  review_comment  order_date order_time  \
0          The product works great. I'm satisfied.  2023-01-10   14:30:00
1           Excellent product! Highly recommended.  2023-01-12   09:45:00
2          Good product, but it could be improved.  2023-01-15   16:20:00
3   I love this product. It exceeded my expectations.  2023-01-17   11:55:00
4       Not happy with the quality. It broke quickly.  2023-01-20   13:40:00
   order_quantity  is_returned payment_method  total_price
```

```
0               2            0    Credit Card         59.98
1               1            0         PayPal         29.99
2               3            1     Debit Card         89.97
3               1            0    Credit Card         44.99
4               2            0    Credit Card         69.98
```

3.3 Get the dimensions (number of rows and columns) of the dataset.

```
print(df.shape)
```

```
(53, 12)
```

3.4 Generate descriptive statistics (mean, median, standard deviation, five-point summary,IQR, etc.) for the data and find the correlation coefficient between rating and total price ignoring the missing values.

```
statisti=df.describe()
correlation=df['rating'].corr(df['total_price'],method='pearson')
print(statisti)
print('Median',df.median())
print("IQR:",statisti.loc['75%']-statisti.loc['25%'])
print('Correleation coefficient{}',correlation)
```

```
      customer_id  product_id     rating  order_quantity  is_returned  \
count   53.000000   53.000000  52.000000       53.000000    53.000000
mean    27.000000  127.000000   3.750000        1.603774     0.075472
std     15.443445   15.443445   1.412479        0.742647     0.266679
min      1.000000  101.000000   0.000000        1.000000     0.000000
25%     14.000000  114.000000   3.000000        1.000000     0.000000
50%     27.000000  127.000000   4.000000        1.000000     0.000000
75%     40.000000  140.000000   5.000000        2.000000     0.000000
max     53.000000  153.000000   7.000000        3.000000     1.000000
       total_price
count    40.000000
mean     61.984500
std      30.498038
min      19.990000
25%      39.990000
50%      54.990000
75%      82.485000
max     149.970000
Median customer_id        27.00
product_id        127.00
rating              4.00
order_quantity      1.00
is_returned         0.00
total_price        54.99
dtype: float64
```

```
IQR: customer_id       26.000
product_id        26.000
rating             2.000
order_quantity     1.000
is_returned        0.000
total_price       42.495
dtype: float64
Correleation coefficient{} 0.0786056213336338
```

3.5 Print a concise summary of the dataset, including information on data types and missing values.

```python
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53 entries, 0 to 52
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   customer_id      53 non-null     int64
 1   product_id       53 non-null     int64
 2   customer_name    53 non-null     object
 3   product_name     53 non-null     object
 4   rating           52 non-null     float64
 5   review_comment   50 non-null     object
 6   order_date       53 non-null     object
 7   order_time       53 non-null     object
 8   order_quantity   53 non-null     int64
 9   is_returned      53 non-null     int64
 10  payment_method   53 non-null     object
 11  total_price      40 non-null     float64
dtypes: float64(2), int64(4), object(6)
memory usage: 5.1+ KB
None
```

3.6 Add a new column named unit_price and insert values by calculating the total price divided by the order quantity.

```python
df['unit_price']=df['total_price']/df['order_quantity']
```

3.7 Create three new instances synthetically and add them to the dataset.

```python
import numpy as np
random_instances = {
    'customer_id': np.random.randint(1000, 9999, 3),
    'product_id': np.random.randint(100, 999, 3),
    'customer_name': [''.join(np.random.choice(list('abcdefghijklmnopqrstuvwxyz'), 5)) for _ in range(3)],
    'product_name': [''.join(np.random.choice(list('abcdefghijklmnopqrstuvwxyz'), 8)) for _ in range(3)],
    'rating': np.random.randint(1, 6, 3),
    'review_comment': [''.join(np.random.choice(list('abcdefghijklmnopqrstuvwxyz '), 20)) for _ in range(3)],
    'order_date': pd.to_datetime(np.random.choice(pd.date_range('2023-01-01', '2023-12-31'), 3)).strftime('%Y-%m-%d'),
    'order_time': pd.to_datetime(np.random.choice(pd.date_range('2023-01-01', '2023-12-31', freq='H'), 3)).strftime('%H:%M:%S'),
    'order_quantity': np.random.randint(1, 50, 3),
    'is_returned': np.random.choice([0, 1], 3),
    'payment_method': np.random.choice(['Credit Card', 'PayPal', 'Cash'], 3),
    'total_price': np.random.uniform(10, 500, 3),
    'unit_price': np.random.uniform(1, 50, 3)
}
new_data = pd.DataFrame(random_instances)
df = df.append(new_data, ignore_index=True)
print(df)
```

```
   customer_id  product_id customer_name product_name  rating  \
0            1         101         Alice    Product A     4.0
1            2         102           Bob    Product B     5.0
2            3         103       Charlie    Product A     3.0
3            4         104         David    Product C     5.0
4            5         105           Eve    Product B     2.0
5            6         106         Frank    Product A     4.0
                                   review_comment  order_date order_time  \
0             The product works great. I'm satisfied.  2023-01-10   14:30:00
1              Excellent product! Highly recommended.  2023-01-12   09:45:00
2             Good product, but it could be improved.  2023-01-15   16:20:00
3    I love this product. It exceeded my expectations.  2023-01-17   11:55:00
4          Not happy with the quality. It broke quickly.  2023-01-20   13:40:00
5            Great value for the price. No complaints.  2023-01-23   10:15:00
   order_quantity  is_returned payment_method  total_price  unit_price
0               2            0    Credit Card        59.98      29.990
1               1            0         PayPal        29.99      29.990
2               3            1     Debit Card        89.97      29.990
3               1            0    Credit Card        44.99      44.990
4               2            0    Credit Card        69.98      34.990
5               1            0         PayPal        29.99      29.990
```

3.8 Delete the newly inserted three instances from the dataset.

```
df=df[:-3]
```

3.9 Update Lucas's payment method to "Debit Card" provided it is currently set to C̈redit Card."

```
df.loc[df['customer_name']=='Lucas','payment_method']='Debit Card'
```

3.10 Find the names of customers who have ordered product A and rated it at least 2 Also, find the total price for each of these customers and calculate the mean, median, and standard deviation of the total price.

```
filtered_df = df[(df['product_name'] == 'Product A') & (df['rating'] >= 2)]
total_price_per_customer = filtered_df.groupby('customer_name')['total_price'↩
    ].sum()
print("Names of customers who ordered Product A and rated it at least 2:")
print(filtered_df['customer_name'])
print('Mean:',total_price_per_customer.mean())
print('Median:',total_price_per_customer.median())
print('Standard deviation:',total_price_per_customer.std())
```

```
Names of customers who ordered Product A and rated it at least 2:
0       Alice
2     Charlie
5       Frank
9        Jack
13       Noah
Name: customer_name, dtype: object
Mean: 54.046875
Median: 49.985
Standard deviation: 45.493055252972404
```

3.11 Find the names and dates of orders of customers who have returned items.

```python
returned_df = df[df['is_returned']==1]
print(returned_df[['customer_name','order_date']])
```

```
 customer_name  order_date
2        Charlie  2023-01-15
11          Liam  2023-01-16
14        Olivia  2023-01-21
19         James  2023-02-05
54           Joe  2023-01-10
55           Ram  2023-01-10
```

3.12 Perform data cleaning by: (a) Deleting duplicate rows.

(b) Replacing missing rating values with the average rating of the respective product category.

(c) Adding the value "unknown" for missing review comments.

(d) Adding the missing total_price values by calculating the product of average unit price and quantity ordered.

```python
print(df.drop_duplicates())
average_ratings = df.groupby('product_name')['rating'].mean()
df['rating'].fillna(df['product_name'].map(average_ratings), inplace=True)
df['review_comment'].fillna('unknown', inplace=True)
average_unit_prices = df.groupby('product_name')['unit_price'].mean()
missing_total_prices = df['order_quantity'] * df['product_name'].map(↩
    average_unit_prices)
df['total_price'].fillna(missing_total_prices, inplace=True)
```

```
   customer_id  product_id customer_name product_name  rating  \
0            1         101         Alice    Product A     4.0
1            2         102           Bob    Product B     5.0
2            3         103       Charlie    Product A     3.0
3            4         104         David    Product C     5.0
4            5         105           Eve    Product B     2.0
5            6         106         Frank    Product A     4.0
   order_quantity  is_returned payment_method  total_price  unit_price
0               2            0    Credit Card        59.98      29.990
1               1            0         PayPal        29.99      29.990
2               3            1     Debit Card        89.97      29.990
3               1            0    Credit Card        44.99      44.990
4               2            0    Credit Card        69.98      34.990
5               1            0         PayPal        29.99      29.990
```

3.13 Find the normalized total price values by performing:

(a) Z-score normalization.

(b) Min-max normalization.

(c) Decimal scaling.

```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df['Total Price (Z-Score)'] = scaler.fit_transform(df[['total_price']])
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df['Total Price (Min-Max)'] = scaler.fit_transform(df[['total_price']])
max_value = df['total_price'].abs().max()
num_decimals = len(str(int(max_value)))
divisor = 10 ** num_decimals
df['Total Price (Decimal Scaling)'] = df['total_price'] / divisor
print(df)
```

```
   customer_id  product_id customer_name product_name  rating  \
0            1         101         Alice    Product A     4.0
1            2         102           Bob    Product B     5.0
2            3         103       Charlie    Product A     3.0
3            4         104         David    Product C     5.0
4            5         105           Eve    Product B     2.0
5            6         106         Frank    Product A     4.0
   order_quantity  is_returned payment_method  total_price  unit_price  \
0               2            0    Credit Card    59.980000      29.990
1               1            0         PayPal    29.990000      29.990
2               3            1     Debit Card    89.970000      29.990
3               1            0    Credit Card    44.990000      44.990
4               2            0    Credit Card    69.980000      34.990
5               1            0         PayPal    29.990000      29.990
Total Price (Z-Score)  Total Price (Min-Max)  Total Price (Decimal Scaling)
0           -0.150801              0.307592                       0.059980
1           -1.101461              0.076917                       0.029990
2            0.799859              0.538266                       0.089970
3           -0.625972              0.192293                       0.044990
4            0.166191              0.384509                       0.069980
5           -1.101461              0.076917                       0.029990
```

3.14 Create a new DataFrame from the given dataset with the following features:

- Customer name
- Product name
- Total price discretized by equal-frequency binning of the original data
- Rating.

```python
new_df = df[['customer_name', 'product_name', 'rating']].copy()
num_bins = 3
new_df['Total price discretized'] = pd.qcut(df['total_price'], q=num_bins, ↩
    labels=False)
print(new_df)
```

```
  customer_name product_name  rating  Total price discretized
0         Alice    Product A     4.0                        1
1           Bob    Product B     5.0                        0
2       Charlie    Product A     3.0                        2
3         David    Product C     5.0                        0
4           Eve    Product B     2.0                        1
5         Frank    Product A     4.0                        0
```

3.15 Create a new DataFrame from the given dataset with the features:

- Customer name
- One-hot encoded rating.

```python
rating_encoded = pd.get_dummies(df['rating'], prefix='rating')
new_df = pd.concat([df['customer_name'], rating_encoded], axis=1)
print(new_df)
```

```
  customer_name  rating_0.0  rating_1.0  rating_2.0  rating_3.0  rating_4.0  \
0         Alice           0           0           0           0           1
1           Bob           0           0           0           0           0
2       Charlie           0           0           0           1           0
3         David           0           0           0           0           0
4           Eve           0           0           1           0           0
5         Frank           0           0           0           0           1
```

Assignment 4
Data Visualization

Create Python code to perform the following tasks using Matplotlib and Seaborn for data visualization

4.1 Load the dataset and import it into a Pandas DataFrame.

```python
import pandas as pd
data=pd.read_csv('sales.csv')
df=pd.DataFrame(data);
print(df)
```

```
         Date Product ID  Product Name     Category   Price  Units Sold  \
0  2023-01-01     P12345  Smartphone X  Smartphones  599.99          10
1  2023-01-02     L98765      Laptop Y      Laptops  899.99           5
2  2023-01-03     A98765  Headphones Z  Accessories   49.99          20
3  2023-01-04     P12345  Smartphone X  Smartphones  599.99           8
4  2023-01-05     M54321      Tablet A      Tablets  349.99          12
5  2023-01-06     C24680    Keyboard B  Accessories   29.99          15
   Inventory Level   Customer Name  Rating  \
0               30        John Doe     4.5
1               15      Jane Smith     5.0
2               50    Mark Johnson     3.0
3               22  Lucy Williams     4.0
4               28     David Brown     4.2
5               35    Susan Taylor     4.7
              Review Comments           Unnamed: 10
0               Great phone!                   NaN
1          Excellent laptop.                   NaN
2         Decent headphones.                   NaN
3      Good value for money.                   NaN
4      Nice tablet for work.                   NaN
5      Comfortable keyboard.                   NaN
```

4.2 Explore the dataset by displaying the first few rows and examining its structure.

```python
print(df.head())
print(df.info())
```

```
         Date Product ID  Product Name     Category   Price  Units Sold  \
0  2023-01-01     P12345  Smartphone X  Smartphones  599.99          10
1  2023-01-02     L98765      Laptop Y      Laptops  899.99           5
2  2023-01-03     A98765  Headphones Z  Accessories   49.99          20
3  2023-01-04     P12345  Smartphone X  Smartphones  599.99           8
4  2023-01-05     M54321      Tablet A      Tablets  349.99          12
   Inventory Level  Customer Name  Rating        Review Comments Unnamed: 10
0               30       John Doe     4.5           Great phone!         NaN
1               15     Jane Smith     5.0      Excellent laptop.         NaN
2               50   Mark Johnson     3.0     Decent headphones.         NaN
3               22  Lucy Williams     4.0  Good value for money.         NaN
4               28    David Brown     4.2  Nice tablet for work.         NaN
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Date             30 non-null     object
 1   Product ID       30 non-null     object
 2   Product Name     30 non-null     object
 3   Category         30 non-null     object
 4   Price            30 non-null     float64
 5   Units Sold       30 non-null     int64
 6   Inventory Level  30 non-null     int64
 7   Customer Name    30 non-null     object
 8   Rating           30 non-null     float64
 9   Review Comments  30 non-null     object
 10  Unnamed: 10      1 non-null      object
dtypes: float64(2), int64(2), object(7)
memory usage: 2.7+ KB
None
```

4.3 Generate the following basic visualizations:

(a) Create a histogram for the "price" attribute.

(b) Create a bar chart for the "category" attribute.

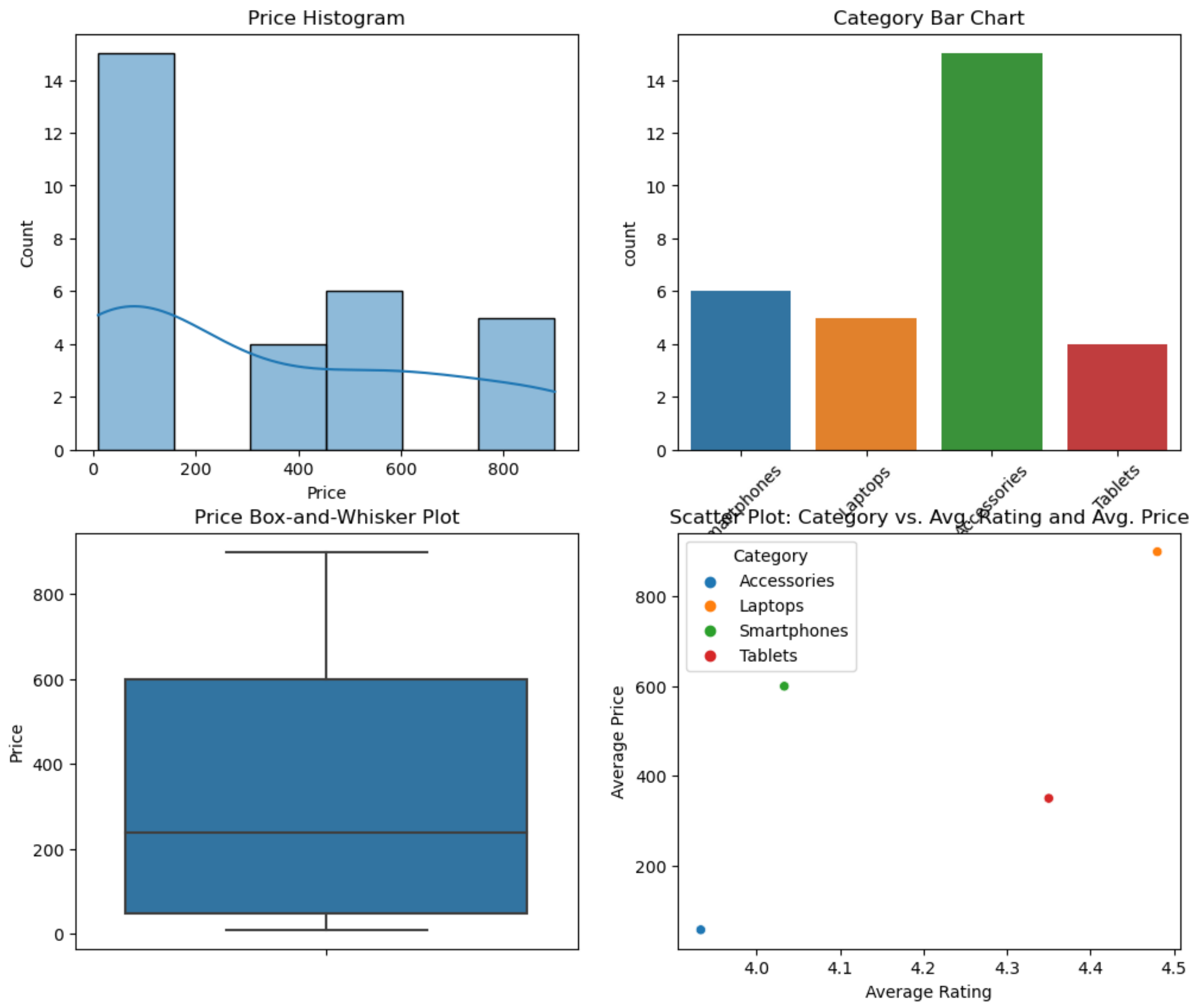
(c) Create a box-and-whisker plot for the "price" attribute.

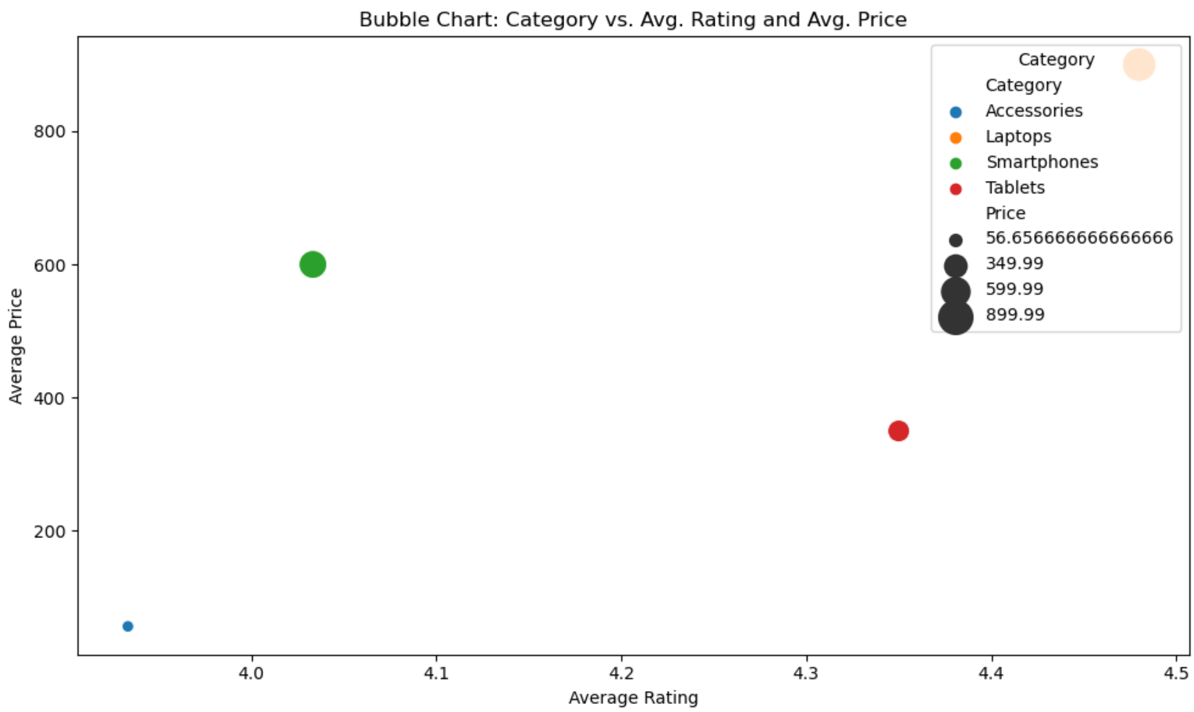
(d) Create a scatter plot showing product categories and average ratings, and average prices.

(e) Create a bubble chart showing product categories, average ratings, and average prices.

Customize your visualizations by adding titles, labels, legends, and appropriate color schemes.

```
import matplotlib.pyplot as plt
import seaborn as sns
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
axes[0, 0].set_title("Price Histogram")
sns.histplot(df['Price'], kde=True, ax=axes[0, 0])
axes[0, 1].set_title("Category Bar Chart")
sns.countplot(data=df, x='Category', ax=axes[0, 1])
axes[0, 1].tick_params(axis='x', rotation=45)
axes[1, 0].set_title("Price Box-and-Whisker Plot")
sns.boxplot(data=df, y='Price', ax=axes[1, 0])
category_avg_ratings = df.groupby('Category')['Rating'].mean()
category_avg_prices = df.groupby('Category')['Price'].mean()
axes[1, 1].set_title("Scatter Plot: Category vs. Avg. Rating and Avg. Price")
sns.scatterplot(x=category_avg_ratings, y=category_avg_prices,
hue=category_avg_ratings.index, ax=axes[1, 1])
plt.figure(figsize=(10, 6))
sns.scatterplot(x=category_avg_ratings, y=category_avg_prices, size=category_avg_prices, hue=category_avg_ratings.index, sizes=(50, 400))
plt.xlabel("Average Rating")
plt.ylabel("Average Price")
plt.title("Bubble Chart: Category vs. Avg. Rating and Avg. Price")
plt.legend(title="Category", loc='upper right')
axes[1, 1].set_xlabel("Average Rating")
axes[1, 1].set_ylabel("Average Price")
axes[1, 1].legend(title="Category")
plt.tight_layout()
plt.show()
```


Price Histogram
Count
Price
Category Bar Chart
count
Smartphones
Laptops
Accessories
Tablets
Price Box-and-Whisker Plot
Price
Scatter Plot: Category vs. Avg. Rating and Avg. Price
Category
Accessories
Laptops
Smartphones
Tablets
Average Price
Average Rating


Bubble Chart: Category vs. Avg. Rating and Avg. Price
Category
Category
Accessories
Laptops
Smartphones
Tablets
Price
56.656666666666666
349.99
599.99
899.99
Average Price
Average Rating

Assignment 5
Similarity and Dissimilarity between Data Points

You have been provided the following dataset.

Name, Age, Gender, Occupation, Income, Height (inches), Weight (lbs)

Alice, 28, Female, Engineer, 75000, 65, 140

Bob, 32, Male, Data Scientist, 85000, 70, 180

Charlie, 23, Male, Student, 15000, 68, 160

David, 40, Male, Doctor, 120000, 72, 200

Eve, 35, Female, Lawyer, 90000, 63, 130 Implement the following similarity or dissimilarity calculations:

- Manhattan distance based on income and age.
- Euclidean distance based on age, height, and weight.
- Cosine similarity based on income, height, and weight.
- Jaccard similarity based on gender and occupation (treat these as categorical attributes).
- Form a similarity (dissimilarity) matrix showing the similarities using an appropriate measure such as gower distance.

5.1 Data loading and preprocessing

```python
import numpy as np
from scipy.spatial import distance
from scipy.spatial.distance import cosine
from scipy.spatial.distance import pdist
from scipy.spatial.distance import squareform
from gower import gower_matrix
import pandas as pd
data = [
    ["Alice", 28, "Female", "Engineer", 75000, 65, 140],
    ["Bob", 32, "Male", "Data Scientist", 85000, 70, 180],
    ["Charlie", 23, "Male", "Student", 15000, 68, 160],
    ["David", 40, "Male", "Doctor", 120000, 72, 200],
    ["Eve", 35, "Female", "Lawyer", 90000, 63, 130]
]
age_income_data = np.array([[row[1], row[4]] for row in data])
age_height_weight_data = np.array([[row[1], row[5], row[6]] for row in data])
income_height_weight_data = np.array([[row[4], row[5], row[6]] for row in data])
gender_occupation_data = np.array([[row[2], row[3]] for row in data])
manhattan_distance = np.zeros((len(data), len(data)))
```

5.2 Manhattan distance

```
for i in range(len(data)):
    for j in range(len(data)):
        manhattan_distance[i, j] = distance.cityblock(age_income_data[i], age_income_data[j])
```

5.3 Euclidean distance

```
euclidean_distance = distance.pdist(age_height_weight_data, metric='euclidean')
euclidean_distance = distance.squareform(euclidean_distance)
```

5.4 Cosine similarity

```
cosine_similarity = pdist(income_height_weight_data, metric='cosine')
cosine_similarity = 1 - squareform(cosine_similarity)
def jaccard_similarity(set1, set2):
    intersection = len(set1.intersection(set2))
    union = len(set1.union(set2))
    return intersection / union
```

5.5 Jaccard similarity

```
jaccard_similarity_matrix = np.zeros((len(gender_occupation_data), len(gender_occupation_data)))
for i in range(len(gender_occupation_data)):
    for j in range(len(gender_occupation_data)):
        jaccard_similarity_matrix[i, j] = jaccard_similarity(set(gender_occupation_data[i]), set(gender_occupation_data[j]))
df = pd.DataFrame(data, columns=["Name", "Age", "Gender", "Occupation", "Income", "Height", "Weight"])
```

5.6 Gower distance

```
gower_dist_matrix = gower_matrix(df)
print("Manhattan Distance:")
print(manhattan_distance)
print("\nEuclidean Distance:")
print(euclidean_distance)
print("\nCosine Similarity:")
print(cosine_similarity)
print("\nJaccard Similarity:")
print(jaccard_similarity_matrix)
print("\nGower Distance:")
print(gower_dist_matrix)
```

```
[[     0.  10004.  60005.  45012.  15007.]
 [ 10004.      0.  70009.  35008.   5003.]
 [ 60005.  70009.      0. 105017.  75012.]
 [ 45012.  35008. 105017.      0.  30005.]
 [ 15007.   5003.  75012.  30005.      0.]]

Euclidean Distance:
[[ 0.         40.5092582  20.83266666 61.58733636 12.36931688]
 [40.5092582   0.         22.02271555 21.63330765 50.57667447]
 [20.83266666 22.02271555  0.         43.64630569 32.69556545]
 [61.58733636 21.63330765 43.64630569  0.         70.75309181]
 [12.36931688 50.57667447 32.69556545 70.75309181  0.        ]]

Cosine Similarity:
[[1.         0.99999997 0.99995456 0.99999994 0.9999999 ]
 [0.99999997 1.         0.99995658 0.99999987 0.99999977]
 [0.99995456 0.99995658 1.         0.99995177 0.99995013]
 [0.99999994 0.99999987 0.99995177 1.         0.99999997]
 [0.9999999  0.99999977 0.99995013 0.99999997 1.        ]]

Jaccard Similarity:
[[1.         0.         0.         0.         0.33333333]
 [0.         1.         0.33333333 0.33333333 0.        ]
 [0.         0.33333333 1.         0.33333333 0.        ]
 [0.         0.33333333 0.33333333 1.         0.        ]
 [0.33333333 0.         0.         0.         1.        ]]

Gower Distance:
[[0.         0.6367881  0.64065623 0.82419634 0.41710016]
 [0.6367881  0.         0.529145   0.47312257 0.67373616]
 [0.64065623 0.529145   0.         0.7165533  0.77204216]
 [0.82419634 0.47312257 0.7165533  0.         0.79711884]
 [0.41710016 0.67373616 0.77204216 0.79711884 0.        ]]
```

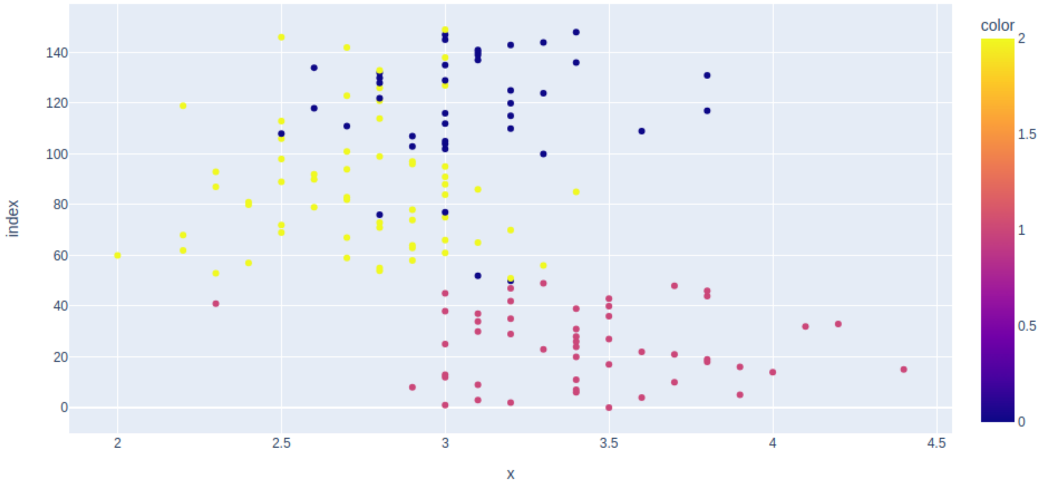
Assignment 6
K-means Clustering

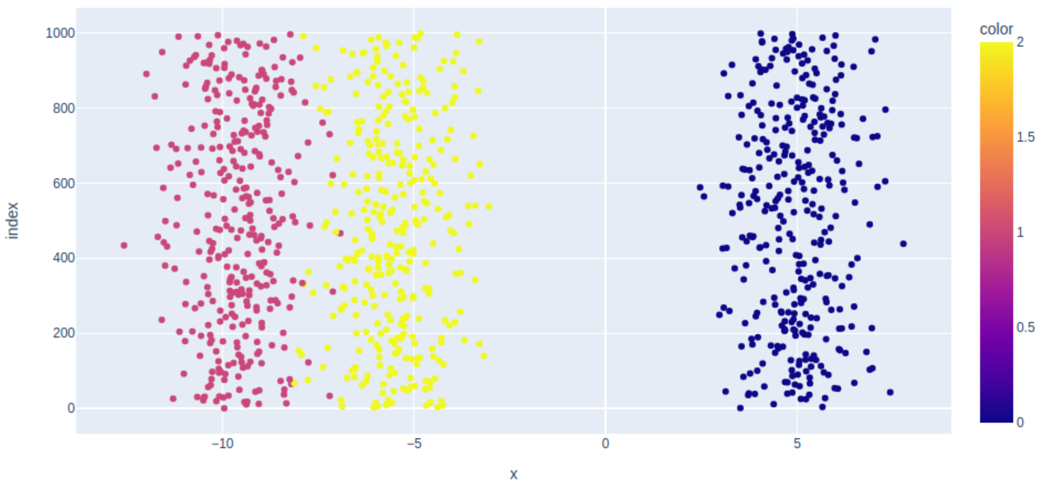
Write a program to cluster a set of points using K-means. Let there be k=3 clusters. Consider Euclidean distance as the distance measure. Randomly initialize a cluster mean as one of the data points. Iterate for 10 iterations. After iterations are over, print the final cluster means for each of the clusters. Use the ground truth cluster label present in the data set to compute and print the Jacquard distance of the obtained clusters with the ground truth clusters for each of the three clusters. Implement the algorithm (1) without using any machine learning library other than numpy, pandas, and matplotlib (2) by using some machine learning library such as Sci-Kit Learn.

6.1 Implementation of K-means from scratch

```
import pandas as pd
import numpy as np
import plotly.express as px
class KMeansClustering:
    def __init__(self, X, num_clusters):
        self.K = num_clusters # cluster number
        self.max_iterations = 100 # max iteration. don't want to run inf time
        self.num_examples, self.num_features = X.shape # num of examples, num ↩
            of features
        self.plot_figure = True # plot figure
    def initialize_random_centroids(self, X):
        centroids = np.zeros((self.K, self.num_features)) # row , column full ↩
            with zero
        for k in range(self.K): # iterations of
            centroid = X[np.random.choice(range(self.num_examples))] # random ↩
                centroids
            centroids[k] = centroid
        return centroids # return random centroids
    def create_cluster(self, X, centroids):
        clusters = [[] for _ in range(self.K)]
        for point_idx, point in enumerate(X):
            closest_centroid = np.argmin(
                np.sqrt(np.sum((point-centroids)**2, axis=1)))
            clusters[closest_centroid].append(point_idx)
        return clusters
    def calculate_new_centroids(self, cluster, X):
        centroids = np.zeros((self.K, self.num_features))
        for idx, cluster in enumerate(cluster):
            new_centroid = np.mean(X[cluster], axis=0)
            centroids[idx] = new_centroid
        return centroids
    def find_cluster(self, clusters, X):
        y_find = np.zeros(self.num_examples)
        for cluster_idx, cluster in enumerate(clusters):
```

```
            for sample_idx in cluster:
                y_find[sample_idx] = cluster_idx
        return y_find
    def plot_fig(self, X, y):
        fig = px.scatter(X[:, 0], X[:, 1], color=y)
        fig.show() # visualize
    def fit(self, X):
        centroids = self.initialize_random_centroids(X) # initialize random centroids
        for _ in range(self.max_iterations):
            clusters = self.create_cluster(X, centroids) # create cluster
            previous_centroids = centroids
            centroids = self.calculate_new_centroids(clusters, X) # calculate new centroids
            diff = centroids - previous_centroids # calculate difference
            if not diff.any():
                break
        y_find = self.find_cluster(clusters, X) # find the cluster of X
        if self.plot_figure: # if true
            self.plot_fig(X, y_find) # plot function
        return y_find
iris = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',header=None)
iris.head(2)
iris_feature_data=iris.loc[:,0:2]
iris_feature_data.head(2)
num_clusters = 3
iris_cluster = KMeansClustering(iris_feature_data, num_clusters)
iris_pred = iris_cluster.fit(iris_feature_data.to_numpy())
print(iris_pred)
from sklearn.datasets import make_blobs
np.random.seed(10)
num_clusters = 3 # num of cluster
X, _ = make_blobs(n_samples=1000, n_features=2, centers=num_clusters)
print (X.shape)
Kmeans = KMeansClustering(X, num_clusters)
y_pred = Kmeans.fit(X)
```



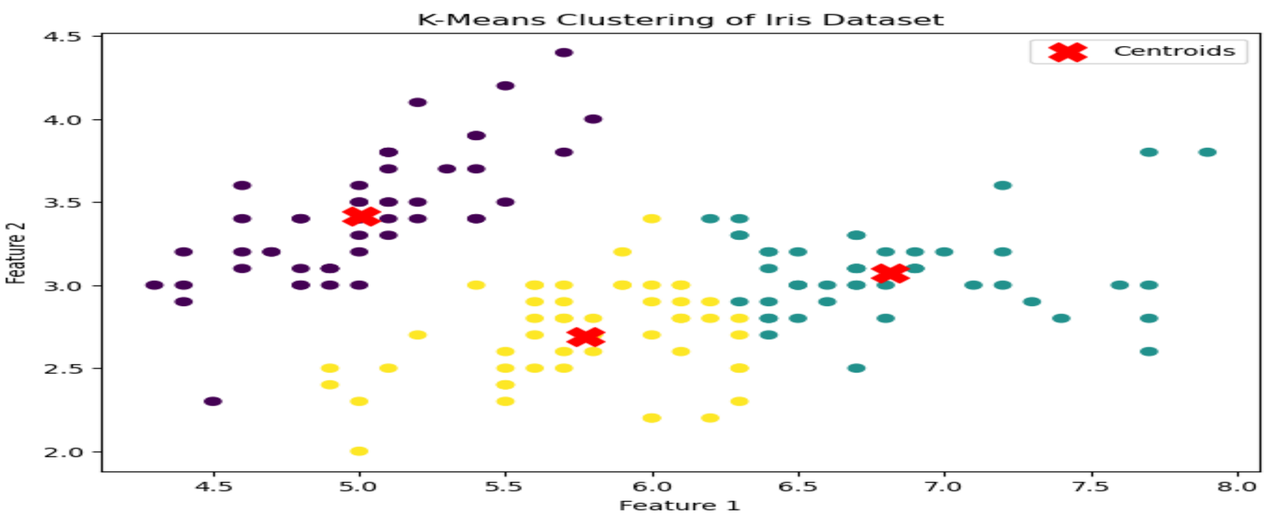



```
     0   1   2   3             4
0  5.1  3.5  1.4  0.2  Iris-setosa
1  4.9  3.0  1.4  0.2  Iris-setosa
     0   1   2
0  5.1  3.5  1.4
1  4.9  3.0  1.4
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 0. 2. 0. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 0. 0. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 0. 2. 0. 0. 0. 0. 2. 0. 0. 0. 0. 0. 0. 2. 2. 0. 0. 0. 0. 2.
 0. 2. 0. 2. 0. 0. 2. 2. 0. 0. 0. 0. 0. 2. 0. 0. 0. 0. 2. 0. 0. 0. 2. 0.
 0. 0. 2. 0. 0. 2.]
 (1000, 2)
```

6.2 Implementation of K-means using Sci-Kit Learn

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = pd.read_csv(url, header=None)
X = iris.iloc[:, [0, 1]].values
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.scatter(centroids[:, 0], centroids[:, 1], marker='X', s=200, c='red', label='Centroids')
plt.title('K-Means Clustering of Iris Dataset')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.show()
```



Assignment 7
Linear Regression

You are given a real estate dataset (Download) consisting of the prices per unit area of different houses which have been sold. In addition to the prices, the dataset shows other information about the houses which include transaction date, age of the house, distance to the nearest Metro Train Station (Metro Rail Transit- MRT) from the house, number of convenience stores and the latitude and longitude of the location. Imagine that the prices of houses linearly depend upon the age, distance to the MRT station and the number of convenience stores. Write a program to predict the price when you are given the features of a new house. FInd the line that fits the dataset using (1) Normal Equations and (2) Gradient Descent Algorithm without using a linear regression library function, Finally, use scikit-learn library function to find the best fit line.

7.1 Implementation of Linear regression using Normal equation

7.1.1 Data cleaning and normalisation

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from mpl_toolkits.mplot3d import Axes3D
real_estate=pd.read_csv('Real estate.csv')
real_estate.head(3)
real_estate.shape
real_estate.set_axis(['No','tdate','hage','dmrt','stores','lati','long','↩
    hprice'],axis=1,inplace=True)#Renaming the features
real_estate.head(1)
X=real_estate.loc[:,"hage":"stores"]
X.head(1)
Y=real_estate.iloc[:,-1]
Y.head(3)
def feature_normalize(X, mean=np.zeros(1), std=np.zeros(1)):
    X = np.array(X)
    if len(mean.shape) == 1 or len(std.shape) == 1:
        mean = np.mean(X, axis=0)
        std = np.std(X, axis=0, ddof=1)
    X = (X - mean)/std
    return X, mean, std
X_norm, mu, sigma = feature_normalize(X)
print (X.head(1),"\n", X_norm[0],"\n",mu,"\n",sigma)
```

```
No X1 transaction date X2 house age X3 distance to the nearest MRT station X4 number of convenience
X5 latitude X6 longitude Y house price of unit area
0 1 2012.917 32.0 84.87882 10 24.98298 121.54024 37.9
1 2 2012.917 19.5 306.59470 9 24.98034 121.53951 42.2
2 3 2013.583 13.3 561.98450 5 24.98746 121.54391 47.3
hage dmrt stores
```

```
0 32.0 84.87882 10
0    37.9
1    42.2
2    47.3
 [ 1.25411095 -0.79153734  2.00498156]
 [  17.71256039 1083.88568891    4.0942029 ]
 [  11.39248453 1262.10959541    2.94556181]
```

7.1.2 Training

```python
def normal_eqn(X, Y):
    inv = np.linalg.pinv(X.T.dot(X))
    W = inv.dot(X.T).dot(Y)
    return W
Xe = np.hstack((np.ones((X.shape[0],1)),X))
W_e = normal_eqn(Xe, Y)
W_e
```

```
array([ 4.29772862e+01, -2.52855827e-01, -5.37912962e-03,  1.29744248e+00])
```

7.1.3 Prediction

```python
house_age=19
distance_to_metro_station=306
number_of_stores=9
f=np.array([1,(house_age-mu[0])/sigma[0],(distance_to_metro_station-mu[1])/sigma[1],(number_of_stores-mu[2])/sigma[2]])
f=np.array([1,house_age,distance_to_metro_station,number_of_stores])
print(np.dot(f,W_e))
```

```
48.203994120889874
```

7.2 Implementation of Linear regression using Gradient Descent Algorithm

7.2.1 Training

```python
def gradientDescent(X, Y, alpha, num_iters):
    m = X.shape[0]
    n=X.shape[1]
    W = np.zeros((n+1,1))
    J_values = np.zeros(shape=(num_iters, 1))
    ones = np.ones((m,1))
    X = np.hstack((ones, X))
    Y = Y[:,np.newaxis]
    for i in range(num_iters):
        temp = np.dot(X, W) - Y
        J_values[i] = np.sum(np.power(temp, 2)) / (2*m)
        temp = np.dot(X.T, temp)
        W = W - (alpha/m) * temp
return W, J_values
W, J_values = gradientDescent(X_norm, Y.to_numpy(), 0.01, 2000)
```

```
print('The parameters found by gradient descent:\n', W)
print("Reduction in Cost:", "Initial cost",J_values[0], "has been reduced to:"
    , J_values[-1])
```

```
The parameters found by gradient descent:
 [[37.98019317]
 [-2.88071816]
 [-6.78855084]
 [ 3.8221985 ]]
Reduction in Cost: Initial cost [813.59219807] has been reduced to: [42.38035331]
```

7.2.2 Prediction

```
house_age=19
distance_to_metro_station=306
number_of_stores=9
f=np.array([1,(house_age-mu[0])/sigma[0],(distance_to_metro_station-mu[1])/
    sigma[1],(number_of_stores-mu[2])/sigma[2]])
print(np.dot(f,W))
```

```
[48.20451394]
```

7.3 Implementation of Linear regression using Scikit-Learn

7.3.1 Training and Prediction

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_norm, Y)
print(lin_reg.intercept_, lin_reg.coef_)
v=lin_reg.intercept_
a=lin_reg.coef_
W_s=np.insert(a,0,v)
print(W_s)
distance_to_metro_station=306
number_of_stores=9
f=np.array([1,(19-mu[0])/sigma[0],(distance_to_metro_station-mu[1])/sigma[1],(
    number_of_stores-mu[2])/sigma[2]])
print(np.dot(f,W_s))
```

```
[37.980193236714975 [-2.88065609 -6.78905111  3.821697]
[37.98019324 -2.88065609 -6.78905111  3.821697  ]
48.203994120887245
```

Assignment 8
Predict the Species of Iris Flowers

Perform classification analysis using K-Nearest Neighbor (Consider k=5), Gaussian Naive Bayes, and Categorical Naive Bayes algorithms to predict the species of iris flowers based on their features. Load the provided dataset into a suitable data structure. Preprocess the data, if necessary, by encoding the categorical "Species" column into numerical values (e.g., Setosa as 0, Versicolor as 1, and Virginica as 2). Split the dataset into training and testing sets (e.g. 80classifiers should be tested on test instances, and the predicted class labels for the test instances should be printed as output.

8.1 Data loading and preprocessing

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB, CategoricalNB
from sklearn.metrics import accuracy_score
data = pd.read_csv('Iris.csv')
class_mapping = {'Iris-setosa': 0,'Iris-versicolor': 1,'Iris-virginica': 2}
data['Species'] = data['Species'].map(class_mapping)
X = data[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y = data['Species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

8.2 Training

```python
knn_classifier = KNeighborsClassifier(n_neighbors=5)
knn_classifier.fit(X_train, y_train)
y_pred_knn = knn_classifier.predict(X_test)
print("K-Nearest Neighbor Predictions:",y_pred_knn)
gnb_classifier = GaussianNB() gnb_classifier.fit(X_train, y_train)
y_pred_gnb = gnb_classifier.predict(X_test)
print("\nGaussian Naive Bayes Predictions:",y_pred_gnb)
cnb_classifier = CategoricalNB() cnb_classifier.fit(X_train, y_train)
y_pred_cnb = cnb_classifier.predict(X_test)
print("\nCategorical Naive Bayes Predictions:",y_pred_cnb)
print("\nAccuracy(K-NN):",(accuracy_score(y_test, y_pred_knn)*100))
print("Accuracy(GaussianNB):",(accuracy_score(y_test, y_pred_gnb)*100))
print("Accuracy(CategoricalNB):"(accuracy_score(y_test,y_pred_cnb)*100))
```

```
K-Nearest Neighbor Predictions:
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
Gaussian Naive Bayes Predictions:
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
Categorical Naive Bayes Predictions:
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 1 0 2 2 2 2 2 0 0]
Accuracy (K-NN): 100.0
Accuracy (Gaussian Naive Bayes): 100.0
Accuracy (Categorical Naive Bayes): 96.66666666666667
```

Assignment 9
Diabetes Prediction

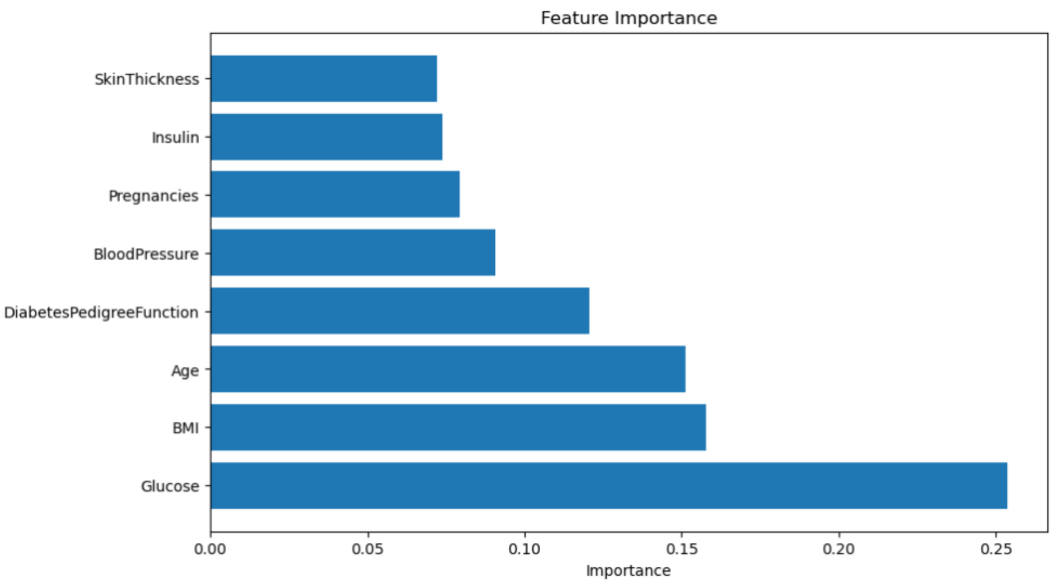
Write a program to learn a binary classifier to predict whether a person will develop diabetes within five years based on the given attributes. 1. Load the Pima Indians Diabetes dataset into a Pandas DataFrame. 2. Preprocess the data by handling missing values, if any, and scaling the numerical features (e.g., using Min-Max scaling). 3. Split the dataset into training and testing sets (e.g.80testing). 4. Implement and evaluate the following machine learning models for binary classification: Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Support Vector Machine (SVM) 5. Train each model on the training data and evaluate its performance on the testing data using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score. 6. Compare the performance of the different models and discuss the results. Which model performs best for this classification task, and why? 7. Visualize the feature importance for the Decision Tree or Random Forest model to identify the most important predictors.

9.1 Data loading and preprocessing

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.tree import export_text
import matplotlib.pyplot as plt
file_path = "diabetes.csv"
df = pd.read_csv(file_path)
print(df.head())
imputer = SimpleImputer(strategy='mean')
df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']] = imputer.fit_transform(df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']])
scaler = MinMaxScaler()
df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']] = scaler.fit_transform(df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']])
X = df.drop('Outcome', axis=1)
y = df['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

9.2 Training

```python
models = {'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),\
    'SVM': SVC()}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    print(f"\n{name} Performance:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")
rf_model = models['Random Forest']
feature_importances = rf_model.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
plt.figure(figsize=(10, 6))
plt.barh(feature_importance_df['Feature'], feature_importance_df['Importance'])
plt.title('Feature Importance')
plt.xlabel('Importance')
plt.show()
```



```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
Logistic Regression Performance:
Accuracy: 0.7662
Precision: 0.7111
Recall: 0.5818
F1 Score: 0.6400
Decision Tree Performance:
Accuracy: 0.7468
Precision: 0.6250
Recall: 0.7273
F1 Score: 0.6723
Random Forest Performance:
Accuracy: 0.7273
Precision: 0.6102
Recall: 0.6545
F1 Score: 0.6316
SVM Performance:
Accuracy: 0.7468
Precision: 0.6600
Recall: 0.6000
F1 Score: 0.6286
```

Assignment 10
CNN classification

Perform image classification using Convolutional Neural Networks (CNNs) on the Fashion MNIST dataset. Fashion MNIST is a dataset of grayscale images depicting various fashion items.

1. Load the Fashion MNIST dataset using TensorFlow and Keras.
2. Preprocess the data by reshaping the images to include a channel dimension (grayscale), scaling pixel values to the range [0, 1], and one-hot encoding the labels.
3. Design a CNN architecture tailored for image classification on Fashion MNIST. Include convolutional layers, max-pooling layers, and fully connected layers.
4. Specify and implement the model architecture using TensorFlow and Keras.
5. Compile the CNN model with appropriate hyperparameters, including the choice of optimizer, loss function, and evaluation metric. Suggested options are the Adam optimizer and categorical cross-entropy loss.
6. Train the CNN model using the preprocessed training data. Experiment with different numbers of epochs and batch sizes to achieve optimal performance.
7. Evaluate the trained CNN model on the preprocessed test data.
8. Calculate and report the test accuracy as the primary evaluation metric.
9. Visualize the training history (e.g., training and validation accuracy plots) using Matplotlib.

10.1 Data preprocessing

```python
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))
train_images, test_images = train_images / 255.0, test_images / 255.0
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

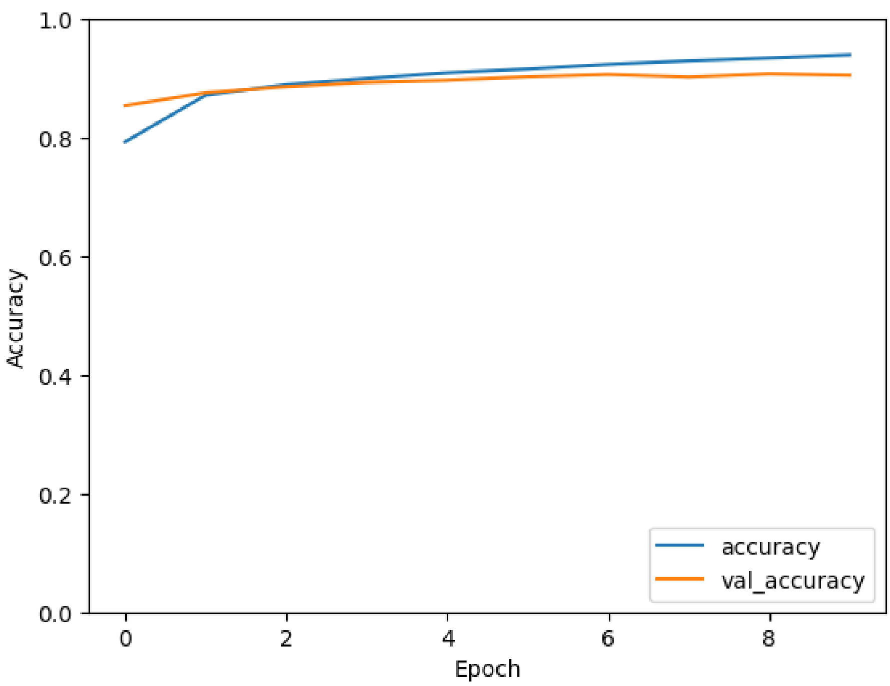
10.2 Training

```
history = model.fit(train_images, train_labels, epochs=10, batch_size=64, validation_data=(test_images, test_labels))
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f'Test accuracy: {test_acc}')
```

```
Epoch 9/10
938/938 [==============================] - 6s 6ms/step - loss: 0.1759 - accuracy: 0.9344 - val_loss:
0.2677 - val_accuracy: 0.9076
Epoch 10/10
938/938 [==============================] - 5s 6ms/step - loss: 0.1628 - accuracy: 0.9397 - val_loss:
0.2749 - val_accuracy: 0.9053
313/313 [==============================] - 1s 3ms/step - loss: 0.2749 - accuracy: 0.9053
Test accuracy: 0.9053000211715698
```

10.3 Graph plotting

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()
```



Assignment 11
Natural Language Processing Tasks

Write the code to perform the following Natural Language Processing (NLP) tasks using the NLTK library: Tokenization, Stemming, Lemmatization, Parts of Speech (PoS) tagging, Chunking, and Named Entity Recognition (NER).

11.1 Tokenization

```
import nltk
nltk.download('punkt')
from nltk.tokenize import sent_tokenize
Text="Good to see you Mary. How are you doing? Good to see you too John. I'm Good, How are you? my God"
Tokenized = sent_tokenize(Text)
print(Tokenized)
Tokenized[0:2]
from nltk.tokenize import word_tokenize
Tokenized = word_tokenize(Text)
print(Tokenized)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
['Good to see you Mary.', 'How are you doing?', 'Good to see you too John.', "I'm Good, How are you?
['Good to see you Mary.', 'How are you doing?']
['Good', 'to', 'see', 'you', 'Mary', '.', 'How', 'are', 'you', 'doing', '?', 'Good', 'to', 'see']
```

11.2 Stop words

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords = stopwords.words("english")
print(stopwords)
for i in Tokenized:
   if i not in stopwords:
       print(i)
from string import punctuation
punctuation = list(punctuation)
print(punctuation)
for i in Tokenized:
   if i not in stopwords and i not in punctuation:
       print(i)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
True
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "
Good
see
Mary
How
?
Good
see
John
['!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', ':', ';', '<', '=', '>',
Good
see
Mary
How
Good
see
John
```

11.3 Stemming

```python
from nltk.stem import PorterStemmer
ps = PorterStemmer()
words = ["Loving", "Chocolate", "Retrieved", "Being","Went","gone","going"]
for i in words:
   print(ps.stem(i))
from nltk.stem import SnowballStemmer
ss = SnowballStemmer('english')
words = ["Loving", "Chocolate", "Retrieved", "Being","Went","gone","going"]
for i in words:
   print(ss.stem(i))
```

```
love
chocol
retriev
be
went
gone
go
love
chocol
retriev
be
went
gone
go
```

11.4 Lemmatization

```
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
nltk.download('omw-1.4')
lem= WordNetLemmatizer()
print("rocks :", lem.lemmatize("rocks"))
print("corpora :", lem.lemmatize("corpora"))
print("better :", lem.lemmatize("better"))
print("believes :", lem.lemmatize("believes"))
print("Went :", lem.lemmatize("Went"))
print("loves :", lem.lemmatize("loves"))
print("better :", lem.lemmatize("better", pos="a"))
print("better :", lem.lemmatize("better", pos="v"))
print("better :", lem.lemmatize("better", pos="n"))
print("believes :", lem.lemmatize("believes", pos='a'))
print("believes :", lem.lemmatize("believes", pos='v'))
print("believes :", lem.lemmatize("believes", pos='n'))
print("went :", lem.lemmatize("went", pos='n'))
print("went :", lem.lemmatize("went", pos='v'))
print("went :", lem.lemmatize("went", pos='a'))
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
rocks : rock
corpora : corpus
better : better
believes : belief
Went : Went
loves : love
better : good
better : better
better : better
believes : believes
believes : believe
believes : belief
went : went
went : go
went : went
```

11.5 Counting Words

```
words = ["men", "teacher", "men", "woman"]
FreqDist = nltk.FreqDist(words)
for i,j in FreqDist.items():
   print(i, ":", j)
men : 2
teacher : 1
woman : 1
\end{verbatim}
\subsection{Word groups}
\begin{code}
\begin{lstlisting}
words = "Visiting Indian Himalayas and Greek Athens as a truth seeker will surely be an amazing experience"
word_tokenize = nltk.word_tokenize(words)
print(list(nltk.bigrams(word_tokenize)))
print(list(nltk.trigrams(word_tokenize)))
print(list(nltk.ngrams(word_tokenize, 4)))
```

```
[('Visiting', 'Indian'), ('Indian', 'Himalayas'), ('Himalayas', 'and'), ('and', 'Greek'), ('Greek',
[('Visiting', 'Indian', 'Himalayas'), ('Indian', 'Himalayas', 'and'), ('Himalayas', 'and', 'Greek'),
[('Visiting', 'Indian', 'Himalayas', 'and'), ('Indian', 'Himalayas', 'and', 'Greek'), ('Himalayas',
```

11.6 Part of Speech (PoS) tagging

```
nltk.download('averaged_perceptron_tagger')
print(nltk.pos_tag(word_tokenize))
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[('Visiting', 'VBG'), ('Indian', 'JJ'), ('Himalayas', 'NNP'), ('and', 'CC'), ('Greek', 'NNP'), ('Ath
```

11.7 Chunking

```
sample_text="""
Rama killed Ravana to save Sita from Lanka.The legend of the Ramayan is the most popular Indian epic.A lot of movies and serials have already
been shot in several languages here in India based on the Ramayana."""
print(sample_text)
from nltk import RegexpParser
from nltk.tree import *
patterns= """mychunk:{<NN.?>*<VBD.?>*<JJ.?>*<CC>?}"""
chunker = RegexpParser(patterns)
```

```python
print("After Regex:",chunker)
word_tokenize = nltk.word_tokenize(sample_text)
tokens_tag =nltk.pos_tag(word_tokenize)
print("After Token:",tokens_tag)
output = chunker.parse(tokens_tag)
print("After Chunking",output)
```

```
Rama killed Ravana to save Sita from Lanka.The legend of the Ramayan is the most popular Indian epic
been shot in several languages here in India based on the Ramayana.
After Regex: chunk.RegexpParser with 1 stages:
RegexpChunkParser with 1 rules:
       <ChunkRule: '<NN.?>*<VBD.?>*<JJ.?>*<CC>?'>
After Token: [('Rama', 'NNP'), ('killed', 'VBD'), ('Ravana', 'NNP'), ('to', 'TO'), ('save', 'VB'), (
After Chunking (S
  (mychunk Rama/NNP killed/VBD)
  (mychunk Ravana/NNP)
  to/TO
  save/VB
  (mychunk Sita/NNP)
  from/IN
  (mychunk Lanka.The/NNP legend/NN)
  of/IN
  the/DT
  (mychunk Ramayan/NNP)
  is/VBZ
  the/DT
  most/RBS
  (mychunk popular/JJ Indian/JJ)
  (mychunk epic.A/NN lot/NN)
  of/IN
  (mychunk movies/NNS and/CC)
  (mychunk serials/NNS)
```

11.8 Named Entity Recognition (NER)

```python
nltk.download('maxent_ne_chunker')
nltk.download('words')
Text = "The russian president Vladimir Putin is in the Kremlin"
Tokenize = nltk.word_tokenize(Text)
POS_tags = nltk.pos_tag(Tokenize)
NameEn = nltk.ne_chunk(POS_tags)
print(NameEn)
```

```python
NameEn = nltk.ne_chunk(POS_tags,binary=False)
print(NameEn)
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
True
(S
  The/DT
  russian/JJ
  president/NN
  (PERSON Vladimir/NNP Putin/NNP)
  is/VBZ
  in/IN
  the/DT
  (FACILITY Kremlin/NNP))
  (S
  The/DT
  russian/JJ
  president/NN
  (PERSON Vladimir/NNP Putin/NNP)
  is/VBZ
  in/IN
  the/DT
  (FACILITY Kremlin/NNP))
```

Assignment 12
Text Classification

You are given part of the corpus of online consumer reviews from the e-commerce site Amazon. Each review is labeled as positive or negative. Preprocess the data and form classification models using (1) Naive Bayes and (2) Support Vector Machines and Compare accuracies of the models

12.1 Preprocessing and splitting of data

```
import pandas as pd
import numpy as np
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score
np.random.seed(500)
from google.colab import files
uploaded = files.upload()
import io
Corpus= pd.read_csv(io.BytesIO(uploaded['amazon.csv']),encoding='latin-1')
Corpus.head(2)
Corpus['text'].dropna(inplace=True)
Corpus['text'] = [entry.lower() for entry in Corpus['text']]
nltk.download('punkt')
Corpus['text']= [word_tokenize(entry) for entry in Corpus['text']]
Corpus.head(2)
nltk.download('wordnet')
nltk.download('omw-1.4')
tag_map = defaultdict(lambda : wn.NOUN)
tag_map['J'] = wn.ADJ
tag_map['V'] = wn.VERB
tag_map['R'] = wn.ADV
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')
for index,entry in enumerate(Corpus['text']):
    Final_words = []
    word_Lemmatized = WordNetLemmatizer()
    for word, tag in pos_tag(entry):
```

```
        if word not in stopwords.words('english') and
        word.isalpha():
            word_Final = word_Lemmatized.lemmatize(word,tag_map[tag[0]])
            Final_words.append(word_Final)
    Corpus.loc[index,'text_final'] = str(Final_words)
Corpus.head(2)
Train_X, Test_X, Train_Y, Test_Y = model_selection.train_test_split(Corpus['↩
    text_final'],Corpus['label'],test_size=0.3)
Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y)
Test_Y = Encoder.fit_transform(Test_Y)
Train_X[:10]
Train_Y[:10]
Tfidf_vect = TfidfVectorizer(max_features=5000)
Tfidf_vect.fit(Corpus['text_final'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X)
Test_X_Tfidf = Tfidf_vect.transform(Test_X)
print(Tfidf_vect.vocabulary_)
print(Train_X_Tfidf)
```

```
text label
0 Stuning even for the non-gamer: This sound tra... positive
1 The best soundtrack ever to anything.: I'm re... positive
text label
0 [stuning, even, for, the, non-gamer, :, this, ... positive
1 [the, best, soundtrack, ever, to, anything, .,... positive
7277    ['good', 'remember', 'read', 'book', 'school',...
9494    ['smell', 'like', 'plumeria', 'unusable', 'pro...
7504    ['play', 'mario', 'play', 'basically', 'ever',...
94      ['thank', 'release', 'love', 'movie', 'kid', '...
9379    ['christina', 'cd', 'amazing', 'voice', 'talen...
2372    ['shark', 'walnut', 'picture', 'frame', 'emble...
2490    ['awesome', 'guide', 'experienced', 'mountaine...
5153    ['useless', 'reason', 'maker', 'item', 'think'...
5313    ['ok', 'best', 'make', 'half', 'way', 'book', ...
5079    ['find', 'irritate', 'though', 'book', 'good',...
Name: text_final, dtype: object
array([1, 0, 0, 1, 1, 0, 1, 0, 0, 0])
{'stun': 4264, 'even': 1532, 'sound': 4109, 'track': 4552, 'beautiful': 384, 'paint': 3157, 'mind':
(0, 4500) 0.37634188677099956
  (0, 4499) 0.1502086671688917
  (0, 3952) 0.35870975205557054
  (0, 3868) 0.25152943577361386
  (0, 3838) 0.2690840463105974
```

```
(0, 3730) 0.3469774999759746
(0, 3643) 0.28971770688512954
(0, 3554) 0.29440491517773787
```

12.2 Naive bayes classification

```
Naive = naive_bayes.MultinomialNB()
Naive.fit(Train_X_Tfidf,Train_Y)
predictions_NB = Naive.predict(Test_X_Tfidf)
print("Naive Bayes Accuracy Score -> ",accuracy_score(predictions_NB, Test_Y)*100)
```

```
Naive Bayes Accuracy Score ->  83.3
```

12.3 SVM classification

```
SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
SVM.fit(Train_X_Tfidf,Train_Y)
predictions_SVM = SVM.predict(Test_X_Tfidf)
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y)*100)
```

```
SVM Accuracy Score ->  84.56666666666666
```

Assignment 13
Web Scraping

Perform basic web scraping using Requests and Beautiful Soup. For lightweight page traversal, you might be able to get by with just Requests, which has overlapping functionality with Beautiful Soup.

13.1 Get the website and extracting the data

```
import requests
import numpy as np
import pandas as pd
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
import re
import os
%matplotlib inline
riturl="https://purdue.edu"
webpage = requests.get(riturl)
ritsoup = BeautifulSoup(webpage.content, "html.parser")
print(ritsoup)
ritsoup.title
links = [link.get('href') for link in ritsoup.find_all('a')]
print(links)
```

13.2 Accessing elements

```
ritsoup.h1
ritsoup.h1.name
ritsoup.head
ritsoup.head.meta
```

13.3 Finding things

```
paragraphs = ritsoup.find_all("p")
print(paragraphs)
ritsoup.find_all("p", attrs={"class": "hide"})
ritsoup.find_all(re.compile("^(p|a)$"))[: 3]
```

13.4 Getting the string from elements

```
ritsoup.h1.string
ritsoup.h1.contents
paragraphs[2]
paragraphs[2].string
para2 = paragraphs[2].contents
para2
para7=paragraphs[5].contents
para7
```

```python
para7[1]['href']
para7[1].string
for s in paragraphs[3].stripped_strings:
    print("="*50)
    print(s)
```

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
<meta content="width=device-width" name="viewport">
<title>Home - National Scholarship Portal</title>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<meta content="IE=edge" http-equiv="X-UA-Compatible"/>
<link href="/public/Content/img/favicon.ico" rel="icon" type="image/png"/>
<link href="/public/Content/css/bootstrap.min.css" rel="stylesheet"/>
<link href="/public/Content/css/own.css" rel="stylesheet"/>
<link href="/public/Content/css/plugins-2.1.css" rel="stylesheet"/>
<link href="/public/Content/css/menudropsub.css" rel="stylesheet"/>
<link href="/public/Content/css/opensans.css" rel="stylesheet"/>
<link href="/public/Content/FaIcons/css/font-awesome.min.css" rel="stylesheet"/>
<script src="/public/Content/js/popper.min.js"></script>
<script src="/public/Content/js/jquery.js"></script>
<script src="/public/Content/js/bootstrap.min.js"></script>
<script src="/public/Content/js/jquery.flexslider.js"></script>
<script src="/public/Content/js/custom.js"></script>
<style>
        .carousel-inner img {
            width: 100%;
            /*height: 250px;*/
        }
        .carousel-inner .carousel-item>img {
            -webkit-animation: thing 6s;
            -o-animation: thing 6s;
            animation: thing 6s;
        }
        @keyframes thing {
            from {
                transform: scale(1, 1);
            }
            to {
                transform: scale(1.1, 1.1);
            }
        }
```

```
    </style>
<title>Purdue University - Indiana's Land Grant University</title>
['#main', '#', '#', 'https://www.purdue.edu/purdue/academics/index.php', 'https://www.purdue.edu/pur
<h1 class="sr-only">Purdue University</h1>
h1
<meta content="dUyYAkgfYaJZPg1QMpbqQU1ve7YGOt0qNb9_8zT1Xgo" name="google-site-verification"/>
[<p class="hide">Find Info For</p>, <p class="hide">Quick Links</p>, <p class="social_description" i
[<p class="hide">Find Info For</p>, <p class="hide">Quick Links</p>]
[<a class="nav nav-skipto" href="#main">Skip to main content</a>,
 <a class="dropdown-toggle" data-toggle="dropdown" href="#">Find Info For <b class="caret"></b></a>]
 .
 .
 .
 Purdue University
['Purdue University']
<p class="social_description" id="description-social">Follow @LifeAtPurdue to see what is happening
Follow @LifeAtPurdue to see what is happening around campus.
['Follow @LifeAtPurdue to see what is happening around campus.']
['Contact Purdue Marketing and Communications at ',
 <a href="https://www.purdue.edu/disabilityresources/">Accessibility Resources</a>,' | ',
 <a href="https://www.purdue.edu/purdue/contact-us">Contact Us</a>]
mailto:digital-marketing@groups.purdue.edu?subject=Accessibility Issue with Your Webpage
digital-marketing@groups.purdue.edu
Purdue University, 610 Purdue Mall, West Lafayette, IN, 47907, 765-494-4600
```