Question 1

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import operator
```

In [10]:
```
#reading the data
data = pd.read_csv('C:/Users/Chhavi Bhadana/Downloads/iris.csv', header=None, names=
data.head()
data.shape
```

Out[10]: (150, 5)

In [11]:
```
#deviding the data into developement set and test set
indices = np.random.permutation(data.shape[0]) #randomizing the indices
div = int(0.60 * len(indices))
development_id, test_id = indices[:div], indices[div:] #splitting the datframe

development_set, test_set = data.loc[development_id,:], data.loc[test_id,:]
print("Development Set:\n", development_set, "\n\nTest Set:\n", test_set)
```

```
Development Set:
     sepal_length  sepal_width  petal_length  petal_width            class
95           5.7          3.0           4.2          1.2  Iris-versicolor
12           4.8          3.0           1.4          0.1      Iris-setosa
75           6.6          3.0           4.4          1.4  Iris-versicolor
138          6.0          3.0           4.8          1.8   Iris-virginica
120          6.9          3.2           5.7          2.3   Iris-virginica
..           ...          ...           ...          ...              ...
71           6.1          2.8           4.0          1.3  Iris-versicolor
34           4.9          3.1           1.5          0.1      Iris-setosa
57           4.9          2.4           3.3          1.0  Iris-versicolor
66           5.6          3.0           4.5          1.5  Iris-versicolor
58           6.6          2.9           4.6          1.3  Iris-versicolor

[90 rows x 5 columns]

Test Set:
     sepal_length  sepal_width  petal_length  petal_width            class
24           4.8          3.4           1.9          0.2      Iris-setosa
52           6.9          3.1           4.9          1.5  Iris-versicolor
69           5.6          2.5           3.9          1.1  Iris-versicolor
148          6.2          3.4           5.4          2.3   Iris-virginica
45           4.8          3.0           1.4          0.3      Iris-setosa
73           6.1          2.8           4.7          1.2  Iris-versicolor
106          4.9          2.5           4.5          1.7   Iris-virginica
105          7.6          3.0           6.6          2.1   Iris-virginica
111          6.4          2.7           5.3          1.9   Iris-virginica
6            4.6          3.4           1.4          0.3      Iris-setosa
72           6.3          2.5           4.9          1.5  Iris-versicolor
145          6.7          3.0           5.2          2.3   Iris-virginica
8            4.4          2.9           1.4          0.2      Iris-setosa
76           6.8          2.8           4.8          1.4  Iris-versicolor
62           6.0          2.2           4.0          1.0  Iris-versicolor
50           7.0          3.2           4.7          1.4  Iris-versicolor
115          6.4          3.2           5.3          2.3   Iris-virginica
11           4.8          3.4           1.6          0.2      Iris-setosa
149          5.9          3.0           5.1          1.8   Iris-virginica
39           5.1          3.4           1.5          0.2      Iris-setosa
7            5.0          3.4           1.5          0.2      Iris-setosa
124          6.7          3.3           5.7          2.1   Iris-virginica
10           5.4          3.7           1.5          0.2      Iris-setosa
```

41	4.5	2.3	1.3	0.3	Iris-setosa
130	7.4	2.8	6.1	1.9	Iris-virginica
40	5.0	3.5	1.3	0.3	Iris-setosa
147	6.5	3.0	5.2	2.0	Iris-virginica
139	6.9	3.1	5.4	2.1	Iris-virginica
84	5.4	3.0	4.5	1.5	Iris-versicolor
104	6.5	3.0	5.8	2.2	Iris-virginica
16	5.4	3.9	1.3	0.4	Iris-setosa
88	5.6	3.0	4.1	1.3	Iris-versicolor
129	7.2	3.0	5.8	1.6	Iris-virginica
103	6.3	2.9	5.6	1.8	Iris-virginica
28	5.2	3.4	1.4	0.2	Iris-setosa
134	6.1	2.6	5.6	1.4	Iris-virginica
87	6.3	2.3	4.4	1.3	Iris-versicolor
55	5.7	2.8	4.5	1.3	Iris-versicolor
48	5.3	3.7	1.5	0.2	Iris-setosa
49	5.0	3.3	1.4	0.2	Iris-setosa
63	6.1	2.9	4.7	1.4	Iris-versicolor
1	4.9	3.0	1.4	0.2	Iris-setosa
80	5.5	2.4	3.8	1.1	Iris-versicolor
43	5.0	3.5	1.6	0.6	Iris-setosa
98	5.1	2.5	3.0	1.1	Iris-versicolor
33	5.5	4.2	1.4	0.2	Iris-setosa
36	5.5	3.5	1.3	0.2	Iris-setosa
131	7.9	3.8	6.4	2.0	Iris-virginica
113	5.7	2.5	5.0	2.0	Iris-virginica
141	6.9	3.1	5.1	2.3	Iris-virginica
30	4.8	3.1	1.6	0.2	Iris-setosa
21	5.1	3.7	1.5	0.4	Iris-setosa
65	6.7	3.1	4.4	1.4	Iris-versicolor
37	4.9	3.1	1.5	0.1	Iris-setosa
31	5.4	3.4	1.5	0.4	Iris-setosa
78	6.0	2.9	4.5	1.5	Iris-versicolor
29	4.7	3.2	1.6	0.2	Iris-setosa
119	6.0	2.2	5.0	1.5	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica
79	5.7	2.6	3.5	1.0	Iris-versicolor

In [12]:

```
#calculating mean and standard deviation for developement set and test set
mean_development_set = development_set.mean()
mean_test_set = test_set.mean()
std_development_set = development_set.std()
std_test_set = test_set.std()
```

In [13]:

```
#retrieve class from developement and test sets
test_class = list(test_set.iloc[:,-1])
dev_class = list(development_set.iloc[:,-1])
```

In [14]:

```
def euclideanDistance(data_1, data_2, data_len):
    dist = 0
    for i in range(data_len):
        dist = dist + np.square(data_1[i] - data_2[i])
    return np.sqrt(dist)

def normalizedEuclideanDistance(data_1, data_2, data_len, data_mean, data_std):
    n_dist = 0
    for i in range(data_len):
        n_dist = n_dist + (np.square(((data_1[i] - data_mean[i])/data_std[i]) - ((da
    return np.sqrt(n_dist)

def cosineSimilarity(data_1, data_2):
    dot = np.dot(data_1, data_2[:-1])
    norm_data_1 = np.linalg.norm(data_1)
    norm_data_2 = np.linalg.norm(data_2[:-1])
```

```
    cos = dot / (norm_data_1 * norm_data_2)
    return (1-cos)
```

In [15]:

```
#k_nearest_neighbors
def knn(dataset, testInstance, k, dist_method, dataset_mean, dataset_std):
    distances = {}
    length = testInstance.shape[1]
    if dist_method == 'euclidean':
        for x in range(len(dataset)):
            dist_up = euclideanDistance(testInstance, dataset.iloc[x], length)
            distances[x] = dist_up[0]
    elif dist_method == 'normalized_euclidean':
        for x in range(len(dataset)):
            dist_up = normalizedEuclideanDistance(testInstance, dataset.iloc[x], len
            distances[x] = dist_up[0]
    elif dist_method == 'cosine':
        for x in range(len(dataset)):
            dist_up = cosineSimilarity(testInstance, dataset.iloc[x])
            distances[x] = dist_up[0]
    # Sort values based on distance
    sort_distances = sorted(distances.items(), key=operator.itemgetter(1))
    neighbors = []
    # Extracting nearest k neighbors
    for x in range(k):
        neighbors.append(sort_distances[x][0])
    # Initializing counts for 'class' labels counts as 0
    counts = {"Iris-setosa" : 0, "Iris-versicolor" : 0, "Iris-virginica" : 0}
    # Computing the most frequent class
    for x in range(len(neighbors)):
        response = dataset.iloc[neighbors[x]][-1]
        if response in counts:
            counts[response] += 1
        else:
            counts[response] = 1
    # Sorting the class in reverse order to get the most frequest class
    sort_counts = sorted(counts.items(), key=operator.itemgetter(1), reverse=True)
    return(sort_counts[0][0])
```

In [16]:

```
# Creating a list of list of all columns except 'class' by iterating through the dev
row_list = []
for index, rows in development_set.iterrows():
    my_list =[rows.sepal_length, rows.sepal_width, rows.petal_length, rows.petal_wid
    row_list.append([my_list])
# k values for the number of neighbors that need to be considered
k_n = [1, 3, 5, 7]
# Distance metrics
distance_methods = ['euclidean', 'normalized_euclidean', 'cosine']
# Performing kNN on the development set by iterating all of the development set data
obs_k = {}
for dist_method in distance_methods:
    development_set_obs_k = {}
    for k in k_n:
        development_set_obs = []
        for i in range(len(row_list)):
            development_set_obs.append(knn(development_set, pd.DataFrame(row_list[i]
        development_set_obs_k[k] = development_set_obs
    # Nested Dictionary containing the observed class for each k and each distance m
    obs_k[dist_method] = development_set_obs_k
print(obs_k)
```

```
{'euclidean': {1: ['Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-virgin
ica', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-vi
```

```
rginica', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-
versicolor', 'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica',
'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-virginic
a', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-vi
rginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-versi
color', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris
-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-virgini
ca', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versi
color', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iri
s-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor',
'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'I
ris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica',
'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versi
color', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor'], 3:
['Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virgin
ica', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-ve
rsicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris
-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris
-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iri
s-versicolor', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Ir
is-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginic
a', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-vi
rginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virg
inica', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iri
s-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica', 'Iris-setos
a', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-virginic
a', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-v
ersicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iri
s-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolo
r', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica',
'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica',
'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-seto
sa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor'], 5: ['Iris-versicolo
r', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-seto
sa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris
-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Ir
is-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-virgi
nica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris
-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-s
etosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setos
a', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versico
lor', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris
-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'I
ris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-virginic
a', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolo
r', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-virginic
a', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-vers
icolor', 'Iris-versicolor', 'Iris-versicolor'], 7: ['Iris-versicolor', 'Iris-setos
a', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-vers
icolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Ir
is-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-virginic
a', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-v
irginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iri
s-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-
versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicol
or', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versic
olor', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolor', 'Iris-virg
inica', 'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
```

```
'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica', 'Iris-set
osa', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolor', 'Iris-setos
a', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virgi
nica', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-
versicolor', 'Iris-versicolor']}, 'normalized_euclidean': {1: ['Iris-versicolor', 'I
ris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa',
'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-vers
icolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-ve
rsicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-virg
inica', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris
-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris
-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-virginic
a', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-se
tosa', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-seto
sa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Ir
is-versicolor', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolo
r', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor', 'Iris-
versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Ir
is-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-virginic
a', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolo
r', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-versicol
or', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-ver
sicolor', 'Iris-versicolor', 'Iris-versicolor'], 3: ['Iris-versicolor', 'Iris-setos
a', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-vers
icolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Ir
is-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-virginic
a', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-v
irginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iri
s-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-
versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicol
or', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versic
olor', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-virgi
nica', 'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'I
ris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Ir
is-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica', 'Iris-setos
a', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolor', 'Iris-setos
a', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-virg
inica', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris
-versicolor', 'Iris-versicolor'], 5: ['Iris-versicolor', 'Iris-setosa', 'Iris-versic
olor', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-v
irginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'I
ris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-virginic
a', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica',
'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-virginic
a', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-versi
color', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iri
s-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iri
s-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'I
ris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-vers
icolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-se
tosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-v
irginica', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolo
r', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolo
r', 'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-set
osa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setos
a', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-v
irginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor',
'Iris-versicolor'], 7: ['Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-v
ersicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'I
ris-virginica', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-setos
a', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-versicol
or', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-virginica', 'Iris-virgin
ica', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor', 'Iris-v
irginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
```

```
'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Ir
is-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolo
r', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iri
s-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-
setosa', 'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'I
ris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-seto
sa', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-versi
color', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-set
osa', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-virgi
nica', 'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iri
s-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolo
r']}, 'cosine': {1: ['Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-virg
inica', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-
virginica', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iri
s-versicolor', 'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica',
'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-virginic
a', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-vi
rginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-versi
color', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris
-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-virgini
ca', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versi
color', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iri
s-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor',
'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'I
ris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica',
'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versi
color', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor'], 3:
['Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virgin
ica', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-ve
rsicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris
-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris
-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iri
s-versicolor', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Ir
is-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginic
a', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-vi
rginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virg
inica', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iri
s-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica', 'Iris-setos
a', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-virginic
a', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-
versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Ir
is-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolo
r', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica',
'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica',
'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-seto
sa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor'], 5: ['Iris-versicolo
r', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-seto
sa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris
-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Ir
is-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-
virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
'Iris-virginica', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-virgi
nica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris
-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-s
etosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setos
a', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-virgini
ca', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor', 'Iris
-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'I
ris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-virginic
a', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolo
r', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-versicol
or', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-ver
sicolor', 'Iris-versicolor', 'Iris-versicolor'], 7: ['Iris-versicolor', 'Iris-setos
a', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-vers
```

```
icolor', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Ir
is-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-virginic
a', 'Iris-versicolor', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-v
irginica', 'Iris-virginica', 'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iri
s-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-
versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicol
or', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versic
olor', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-virgi
nica', 'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'I
ris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Ir
is-versicolor', 'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica', 'Iris-setos
a', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolor', 'Iris-setos
a', 'Iris-setosa', 'Iris-virginica', 'Iris-virginica', 'Iris-versicolor', 'Iris-virg
inica', 'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris
-versicolor', 'Iris-versicolor']}}
```

In [17]:

```
# Calculating the accuracy of the development set by comparing it with the developme
accuracy = {}
for key in obs_k.keys():
    accuracy[key] = {}
    for k_value in obs_k[key].keys():
        #print('k = ', key)
        count = 0
        for i,j in zip(dev_class, obs_k[key][k_value]):
            if i == j:
                count = count + 1
            else:
                pass
        accuracy[key][k_value] = count/(len(dev_class))

# Storing the accuracy for each k and each distance metric into a dataframe
df_res = pd.DataFrame({'k': k_n})
for key in accuracy.keys():
    value = list(accuracy[key].values())
    df_res[key] = value
print(df_res)
```

```
   k  euclidean  normalized_euclidean    cosine
0  1   1.000000              1.000000  1.000000
1  3   0.977778              0.977778  0.988889
2  5   0.966667              0.988889  0.977778
3  7   0.977778              0.966667  0.977778
```

In [18]:

```
column_val = [c for c in df_res.columns if not c.startswith('k')]
col_max = df_res[column_val].max().idxmax(1)
best_dist_method = col_max
row_max = df_res[col_max].argmax()
best_k = int(df_res.iloc[row_max]['k'])
if df_res.isnull().values.any():
    print('\n\n\nBest k value is\033[1m', best_k, '\033[0mand best distance metric i
else:
    print('\n\n\nBest k value is\033[1m', best_k, '\033[0mand best distance metric i
```

Best k value is **1** and best distance metric is **euclidean** .

In [19]:

```
 row_list_test = []
for index, rows in test_set.iterrows():
    my_list =[rows.sepal_length, rows.sepal_width, rows.petal_length, rows.petal_wid
    row_list_test.append([my_list])
test_set_obs = []
```

```
for i in range(len(row_list_test)):
    test_set_obs.append(knn(test_set, pd.DataFrame(row_list_test[i]), best_k, best_d
#print(test_set_obs)

count = 0
for i,j in zip(test_class, test_set_obs):
    if i == j:
        count = count + 1
    else:
        pass
accuracy_test = count/(len(test_class))
print('Final Accuracy of the Test dataset is ', accuracy_test)
```

```
Final Accuracy of the Test dataset is  1.0
```

Question 2

In [20]:

```
data.groupby('class').size()
```

Out[20]:

```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

In [21]:

```
feature_columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
X = data[feature_columns].values
y = data['class'].values
```

In [22]:

```
#converting string labels into integers
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

In [23]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4, random_st
```

In [24]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
k_n = [1,3,5,7]
scores= {}
scores_list = []
for k in k_n:
    knn = KNeighborsClassifier(n_neighbors  = k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    scores[k] = metrics.accuracy_score(y_test, y_pred)
    scores_list.append(metrics.accuracy_score(y_test, y_pred))
scores_list
```

Out[24]:

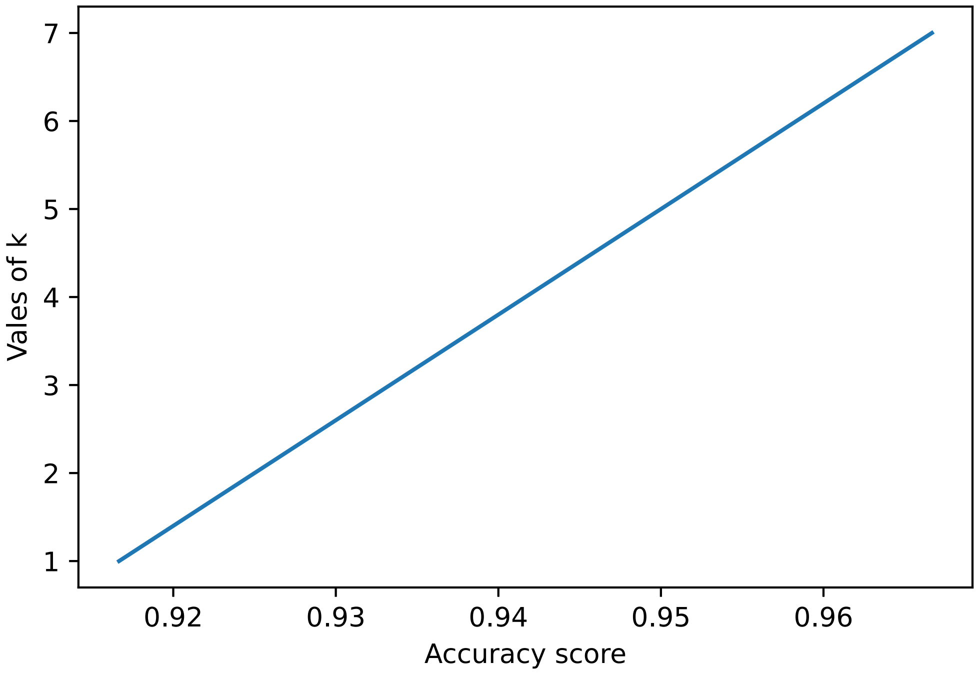
```
[0.9166666666666666, 0.9333333333333333, 0.95, 0.9666666666666667]
```

In [25]:

```
import matplotlib.pyplot as plt
plt.plot(scores_list, k_n)
plt.ylabel('Vales of k')
plt.xlabel('Accuracy score')
```

Out[25]:

```
Text(0.5, 0, 'Accuracy score')
```



Since the accuracy vs k value graph is a straight line the knee value does not exist

Question 3

In [72]:
```
df = pd.read_csv("C:/Users/Chhavi Bhadana/Downloads/wine.csv")
df.columns
```

Out[72]:
```
Index(['Wine', 'Alcohol', 'Malic.acid', 'Ash', 'Acl', 'Mg', 'Phenols',
       'Flavanoids', 'Nonflavanoid.phenols', 'Proanth', 'Color.int', 'Hue',
       'OD', 'Proline'],
      dtype='object')
```

In [73]:
```
A = df.iloc[:,1:]
b = df['Wine']
```

In [74]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(A,b, test_size=0.1, random_state
```

In [75]:
```
GaussNB = GaussianNB()
GaussNB.fit(A, b)
GaussNB.score(X_test, y_test)
y_pred = GaussNB.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred))
```

```
1.0
```

In [76]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         7
           3       1.00      1.00      1.00         5
```

```
    accuracy                           1.00        18
   macro avg       1.00      1.00      1.00        18
weighted avg       1.00      1.00      1.00        18
```

In [113…

```
test_sizes = [0.2, 0.3, 0.4, 0.5]
Scores = {}
Accuracy_scores = []

S2 = {}
Ps = []

S3 = {}
Rs = []

for i in test_sizes:

    #training the model with different values of train and test size
    X_train, X_test, y_train, y_test = train_test_split(A,b, test_size=i, random_sta
    y_pred = GaussNB.predict(X_test)

    #making a list of accuracy scores
    Scores[i] = metrics.accuracy_score(y_test, y_pred)
    Accuracy_scores.append(accuracy_score(y_test, y_pred))

    #making a list of precision scores

    from sklearn.metrics import precision_score

    S2[i] = metrics.precision_score(y_test, y_pred, average= 'micro')
    Ps.append(metrics.precision_score(y_test, y_pred, average= 'micro'))

    #making a list of recall scores

    from sklearn.metrics import recall_score

    S3[i] = metrics.recall_score(y_test, y_pred, average= 'micro')
    Rs.append(metrics.recall_score(y_test, y_pred, average= 'micro'))
```

In [78]:

```
Accuracy_scores
```

Out[78]:

```
[1.0, 1.0, 0.9861111111111112, 0.9887640449438202]
```

In [144…

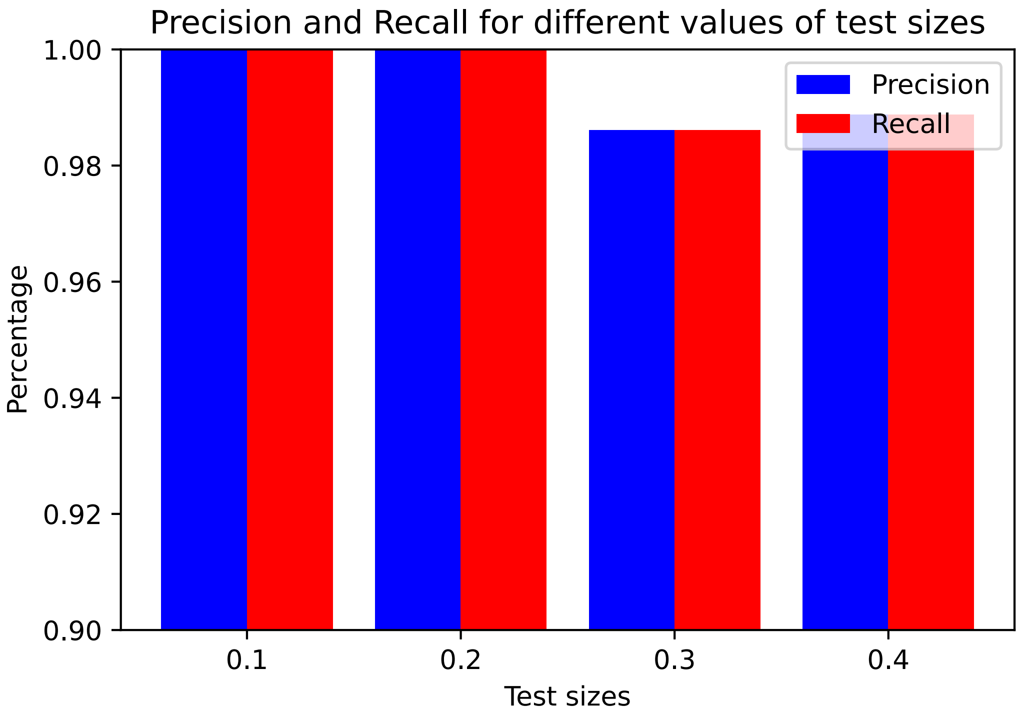
```
import matplotlib.pyplot as plt

X = ['0.1','0.2','0.3','0.4']


X_axis = np.arange(len(X))

plt.bar(X_axis - 0.2, Ps, 0.4, label = 'Precision', color = 'b')
plt.bar(X_axis + 0.2, Rs, 0.4, label = 'Recall', color = 'r')
plt.ylim(0.9, 1)

plt.xticks(X_axis, X)
plt.xlabel("Test sizes")
plt.ylabel("Percentage")
plt.title("Precision and Recall for different values of test sizes")
plt.legend()
plt.show()
```



Question 4

In [47]:
```
train = pd.read_csv("C:/Users/Chhavi Bhadana/Downloads/data/train.tsv", sep = '\t')
test.head()
```

Out[47]:

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2

In [33]:
```
train.info
```

Out[33]:
```
<bound method DataFrame.info of         PhraseId  SentenceId  \
0              1           1
1              2           1
2              3           1
3              4           1
4              5           1
...          ...         ...
156055    156056        8544
156056    156057        8544
156057    156058        8544
156058    156059        8544
156059    156060        8544

                                                   Phrase  Sentiment
0       A series of escapades demonstrating the adage ...          1
1       A series of escapades demonstrating the adage ...          2
2                                                A series          2
3                                                       A          2
4                                                  series          2
...                                                   ...        ...
156055                                          Hearst 's          2
```

```
156056                    forced avuncular chortles          1
156057                           avuncular chortles          3
156058                                    avuncular          2
156059                                     chortles          2

[156060 rows x 4 columns]>
```

In [34]:

```
train.Sentiment.value_counts()
```

Out[34]:

```
2    79582
3    32927
1    27273
4     9206
0     7072
Name: Sentiment, dtype: int64
```

In [49]:

```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer
#tokenizer to remove unwanted elements from out data like symbols and numbers

token = RegexpTokenizer(r'[a-zA-Z0-9]+')

cv = CountVectorizer(lowercase=True,stop_words='english',ngram_range = (1,1),tokeniz

text_counts = cv.fit_transform(train['Phrase'])
```

In [94]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(text_counts, train['Sentiment'],
```

In [131…

```
from sklearn.naive_bayes import MultinomialNB
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes


Accuracy_scores_new = []
Precision_scores_new = []
Recall_new = []


clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)

#acuuracy score
Accuracy_scores_new.append((metrics.accuracy_score(y_test, predicted)*100))

#precision score

Precision_scores_new.append(metrics.precision_score(y_test, predicted, average = 'mi


#recall score
Recall_new.append(metrics.accuracy_score(y_test, predicted)*100)
```

In [132…

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(text_counts, train['Sentiment'],
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)

#acuuracy score
```

```
Accuracy_scores_new.append((metrics.accuracy_score(y_test, predicted)*100))

#precision score

Precision_scores_new.append(metrics.precision_score(y_test, predicted, average = 'mi


#recall score
Recall_new.append(metrics.accuracy_score(y_test, predicted)*100)
```

In [133…

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(text_counts, train['Sentiment'],
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)

#acuuracy score
Accuracy_scores_new.append((metrics.accuracy_score(y_test, predicted)*100))

#precision score

Precision_scores_new.append(metrics.precision_score(y_test, predicted, average = 'mi


#recall score
Recall_new.append(metrics.accuracy_score(y_test, predicted))
```

In [134…

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(text_counts, train['Sentiment'],
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)

#acuuracy score
Accuracy_scores_new.append((metrics.accuracy_score(y_test, predicted)*100))

#precision score

Precision_scores_new.append(metrics.precision_score(y_test, predicted, average = 'mi


#recall score
Recall_new.append(metrics.accuracy_score(y_test, predicted)*100)
```

In [135…

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(text_counts, train['Sentiment'],
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)

#acuuracy score
Accuracy_scores_new.append((metrics.accuracy_score(y_test, predicted)*100))

#precision score

Precision_scores_new.append(metrics.precision_score(y_test, predicted, average = 'mi


#recall score
Recall_new.append(metrics.accuracy_score(y_test, predicted)*100)
```

In [136…

```
print(Accuracy_scores_new)
```

```
print(Precision_scores_new)
print(Recall_new)
```

```
[60.206010508778675, 60.69140074330386, 60.49169122986885, 60.206010508778675, 59.5360758682558]
[60.206010508778675, 60.69140074330386, 60.49169122986885, 60.206010508778675, 59.5360758682558]
[60.206010508778675, 60.69140074330386, 0.6049169122986885, 60.206010508778675, 59.5360758682558]
```

In [142…

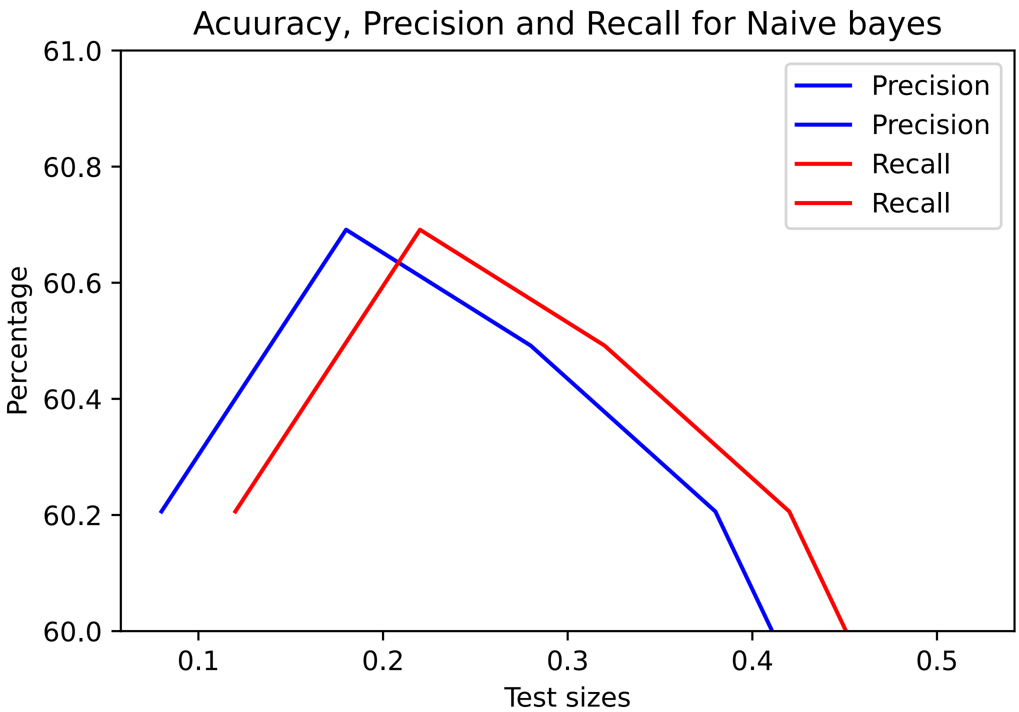
```
import matplotlib.pyplot as plt

X = ['0.1','0.2','0.3','0.4', '0.5']


X_axis = np.arange(len(X))

plt.plot(X_axis - 0.2, Accuracy_scores_new, 0.4, label = 'Precision', color = 'b')
plt.plot(X_axis + 0.2, Precision_scores_new, 0.4, label = 'Recall', color = 'r')
plt.ylim(60, 61)

plt.xticks(X_axis, X)
plt.xlabel("Test sizes")
plt.ylabel("Percentage")
plt.title("Acuuracy, Precision and Recall for Naive bayes")
plt.legend()
plt.show()
```



Question 5

In [209…

```
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.preprocessing import StandardScaler
```

In [210…

```
train_data = pd.read_csv("C:/Users/Chhavi Bhadana/Downloads/train.csv")
test_data = pd.read_csv("C:/Users/Chhavi Bhadana/Downloads/test.csv")
```

In [211…
```
train_data.head()
```

Out[211…

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

In [212…
```
train_data.columns
train_data = train_data.drop(['PassengerId', 'Name', 'SibSp','Parch', 'Ticket', 'Cab
```

In [213…
```
test_data.columns
test_data = test_data.drop(['PassengerId', 'Name', 'SibSp','Parch', 'Ticket','Cabin'
```

In [230…
```
test_data.isnull().sum()
```

Out[230…
```
Pclass     0
Sex        0
Age       86
Fare       0
dtype: int64
```

In [231…
```
test_data['Age'] = test_data['Age'].fillna(test_data['Age'].mean())
```

In [232…
```
x_train = train_data.drop('Survived',axis=1)
x_train.replace({'male':0, 'female':1}, inplace=True)
test_data.replace({'male':0, 'female':1}, inplace=True)
y_train = train_data['Survived']
x_test = test_data.values
x_test
```

Out[232…
```
array([[ 3.        ,  0.        , 34.5       ,  7.8292    ],
       [ 3.        ,  1.        , 47.        ,  7.        ],
       [ 2.        ,  0.        , 62.        ,  9.6875    ],
       ...,
       [ 3.        ,  0.        , 38.5       ,  7.25      ],
```

```
       [ 3.        ,  0.        , 30.27259036,  8.05      ],
       [ 3.        ,  0.        , 30.27259036, 22.3583    ]])
```

In [233…

```python
#removing null values
x_train.isnull().sum()
```

Out[233…

```
Pclass      0
Sex         0
Age       177
Fare        0
dtype: int64
```

In [234…

```python
x_train.Age = x_train.Age.fillna(x_train.Age.mean())
test_data.Age = test_data.Age.fillna(test_data.mean())
test_data.Fare = test_data.Fare.fillna(test_data.Fare.mean())
```

In [235…

```python
decision_tree = DecisionTreeClassifier()
decision_tree.fit(x_train, y_train)
```

Out[235…

```
DecisionTreeClassifier()
```

In [236…

```python
decision_tree.predict(test_data)
```

Out[236…

```
array([0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1,
       1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0],
      dtype=int64)
```

In [238…

```python
param_dict = {
    "criterion":['gini', 'entropy'],
    "max_depth":range(1,10),
    "min_samples_split":range(1,5),
    "min_samples_leaf":range(1,5)
}
```

In [245…

```python
grid = GridSearchCV(decision_tree, param_grid=param_dict, cv= 5, verbose=1, n_jobs =
grid.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 288 candidates, totalling 1440 fits
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\lib\site-packages
\sklearn\model_selection\_search.py:925: UserWarning: One or more of the test scores
are non-finite: [        nan 0.78673655 0.78673655 0.78673655        nan 0.78673655
 0.78673655 0.78673655        nan 0.78673655 0.78673655 0.78673655
        nan 0.78673655 0.78673655 0.78673655        nan 0.77331618
```

```
 0.77331618 0.77331618        nan 0.77331618 0.77331618 0.77331618
        nan 0.77331618 0.77331618 0.77331618        nan 0.77331618
 0.77331618 0.77331618        nan 0.8103195  0.8103195  0.8103195
        nan 0.8103195  0.8103195  0.8103195         nan 0.81256042
 0.81256042 0.81256042        nan 0.81143682 0.81143682 0.81143682
        nan 0.79021405 0.78909673 0.78909673        nan 0.78909673
 0.78909673 0.78909673        nan 0.79133764 0.79133764 0.79133764
        nan 0.78684326 0.78684326 0.78684326        nan 0.81595003
 0.81595003 0.81706735        nan 0.81482016 0.81369657 0.81482016
        nan 0.81369029 0.81369029 0.81369029        nan 0.80919591
 0.80919591 0.80919591        nan 0.81482644 0.81482644 0.81707363
        nan 0.81594376 0.81594376 0.81706735        nan 0.81930827
 0.81930827 0.81930827        nan 0.81368401 0.81368401 0.81368401
        nan 0.81595003 0.81706735 0.81595003        nan 0.81369657
 0.81369657 0.81482016        nan 0.82379637 0.82379637 0.82379637
        nan 0.81930827 0.81930827 0.82043186        nan 0.81596259
 0.81148076 0.81147448        nan 0.81035089 0.81146821 0.81372167
        nan 0.82717971 0.82380893 0.82605612        nan 0.81819095
 0.81819095 0.81819095        nan 0.80812253 0.81036344 0.80587534
        nan 0.80700521 0.80924612 0.8092524         nan 0.82267278
 0.82154918 0.81819723        nan 0.81260436 0.81148076 0.81148076
        nan 0.78673655 0.78673655 0.78673655        nan 0.78673655
 0.78673655 0.78673655        nan 0.78673655 0.78673655 0.78673655
        nan 0.78673655 0.78673655 0.78673655        nan 0.77331618
 0.77331618 0.77331618        nan 0.77331618 0.77331618 0.77331618
        nan 0.77331618 0.77331618 0.77331618        nan 0.77331618
 0.77331618 0.77331618        nan 0.8159312  0.8159312  0.8159312
        nan 0.8170548  0.8170548  0.8170548         nan 0.8170548
 0.8170548  0.8170548         nan 0.8159312  0.8159312  0.8159312
        nan 0.79578181 0.79578181 0.79689913        nan 0.79914004
 0.79914004 0.79914004        nan 0.80026364 0.80026364 0.80026364
        nan 0.79914004 0.79914004 0.79914004        nan 0.80583767
 0.80471408 0.80471408        nan 0.80696127 0.80696127 0.80696127
        nan 0.80808487 0.80808487 0.80808487        nan 0.80807231
 0.80807231 0.80807231        nan 0.802492   0.79911493 0.79911493
        nan 0.80360304 0.80247944 0.80360304        nan 0.80808487
 0.80920846 0.80920846        nan 0.80919591 0.80807231 0.80919591
        nan 0.80808487 0.80808487 0.80807859        nan 0.80807859
 0.80807859 0.8103195         nan 0.81930199 0.81930199 0.81930199
        nan 0.81480133 0.81255414 0.81367774        nan 0.80697382
 0.79798506 0.80247317        nan 0.80359676 0.80359676 0.80247317
        nan 0.81593748 0.81593748 0.81706108        nan 0.81033833
 0.81033833 0.80921474        nan 0.78571339 0.78906534 0.79017011
        nan 0.79578809 0.7969054  0.79466449        nan 0.80812253
 0.80812253 0.80812253        nan 0.80923357 0.81035717 0.81035717]
  category=UserWarning
```

Out[245…

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(1, 10),
                         'min_samples_leaf': range(1, 5),
                         'min_samples_split': range(1, 5)},
             verbose=1)
```

In [246…

```
grid.best_params_
```

Out[246…

```
{'criterion': 'gini',
 'max_depth': 8,
 'min_samples_leaf': 3,
 'min_samples_split': 2}
```

In [247…

```
grid.best_estimator_
```

Out[247…

```
DecisionTreeClassifier(max_depth=8, min_samples_leaf=3)
```

In [248…

```
grid.best_score_
```

Out[248…
```
0.8271797125102003
```

In [ ]: