Import libraries

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
```

In [3]:
```
df = pd.read_csv("cancer_data.csv")
df.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	p
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 33 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
```

```
---  ------                    --------------  -----
 0   id                        569 non-null    int64
 1   diagnosis                 569 non-null    object
 2   radius_mean               569 non-null    float64
 3   texture_mean              569 non-null    float64
 4   perimeter_mean            569 non-null    float64
 5   area_mean                 569 non-null    float64
 6   smoothness_mean           569 non-null    float64
 7   compactness_mean          569 non-null    float64
 8   concavity_mean            569 non-null    float64
 9   concave points_mean       569 non-null    float64
 10  symmetry_mean             569 non-null    float64
 11  fractal_dimension_mean    569 non-null    float64
 12  radius_se                 569 non-null    float64
 13  texture_se                569 non-null    float64
 14  perimeter_se              569 non-null    float64
 15  area_se                   569 non-null    float64
 16  smoothness_se             569 non-null    float64
 17  compactness_se            569 non-null    float64
 18  concavity_se              569 non-null    float64
 19  concave points_se         569 non-null    float64
 20  symmetry_se               569 non-null    float64
 21  fractal_dimension_se      569 non-null    float64
 22  radius_worst              569 non-null    float64
 23  texture_worst             569 non-null    float64
 24  perimeter_worst           569 non-null    float64
 25  area_worst                569 non-null    float64
 26  smoothness_worst          569 non-null    float64
 27  compactness_worst         569 non-null    float64
 28  concavity_worst           569 non-null    float64
 29  concave points_worst      569 non-null    float64
 30  symmetry_worst            569 non-null    float64
 31  fractal_dimension_worst   569 non-null    float64
 32  Unnamed: 32               0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [6]:
```
df.describe()
```

Out[6]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poi
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	5(
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	

8 rows × 32 columns

In [7]:
```
# columns ID, Unnamed: 32 is of no use, we can drop them

df = df.drop(['id','Unnamed: 32'], axis = 1)
```

In [8]:
```
df.shape
```

Out[8]:
```
(569, 31)
```

In [9]:
```
# Target variable : diagnosis

# M : malignant ( Cancerous )
# B : Benign ( non cancerous )
```

In [12]:
```
df['diagnosis'].value_counts()
```

Out[12]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

In [11]:
```
m = df[df.diagnosis == 'M']
m.shape
```

Out[11]:
```
(212, 31)
```

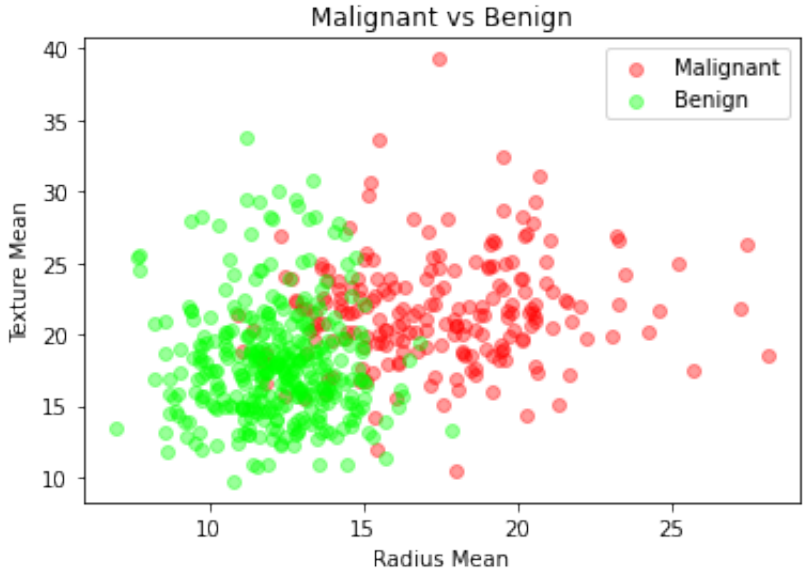
In [13]:
```
b = df[df.diagnosis == 'B']
b.shape
```

Out[13]:
```
(357, 31)
```

In [14]:
```
df.columns
```

Out[14]:
```
Index(['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

In [16]:
```
plt.ylabel("Texture Mean")
plt.xlabel("Radius Mean")
plt.title("Malignant vs Benign")
plt.scatter(m['radius_mean'],m['texture_mean'], color = 'red', label = 'Malignant',alpha = 0.4)
plt.scatter(b['radius_mean'],b['texture_mean'], color = 'lime', label = 'Benign',alpha = 0.4)
plt.legend()
plt.show()
```



In [18]:
```
# convert target variable ( categorical ) to numerical

df['diagnosis_num'] = [1 if i== 'M' else 0 for i in df['diagnosis']]
```

In [19]:
```
df.head()
```

Out[19]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430

5 rows × 32 columns

In [20]:
```
# remove diagnosis column

df = df.drop(['diagnosis'],axis = 1)
```

In [21]:
```
df.shape
```

Out[21]:
```
(569, 31)
```

In [22]:
```
df.head()
```

Out[22]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

5 rows × 31 columns

In [23]:
```
# split data into feature set , target set

X = df.drop(['diagnosis_num'],axis =1)
y = df['diagnosis_num']
```

In [26]:
```
# split data into training set and testing set

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.3,random_state = 42)
```

In [27]:
```
print("X_train shape : " , X_train.shape)
print("X_test shape : " , X_test.shape)
print("y_train shape : " , y_train.shape)
print("y_test shape : " , y_test.shape)
```

```
X_train shape :  (398, 30)
X_test shape :  (171, 30)
y_train shape :  (398,)
y_test shape :  (171,)
```

In [30]:
```
# model

nb = GaussianNB()
nb.fit(X_train,y_train)
```

Out[30]:
```
GaussianNB()
```

In [31]:
```
print("Naive bayes score : ", nb.score(X_test,y_test))
```

```
Naive bayes score :  0.9415204678362573
```

In [ ]:
```

```

In [32]:
```
# 2nd version
```

In [33]:
```
# scaling on X (feature set)

X_scaled = (X-np.min(X))/(np.max(X)-np.min(X))
```

In [35]:
```
# split data into training set and testing set

from sklearn.model_selection import train_test_split
X_train1,X_test1,y_train1,y_test1 = train_test_split(X_scaled,y,test_size = 0.3,random_state = 42)
```

In [36]:
```
# model

nb1 = GaussianNB()
nb1.fit(X_train1,y_train1)
```

Out[36]:
```
GaussianNB()
```

In [38]:
```
print("Naive bayes score after scaling : ", nb1.score(X_test1,y_test1))
```

```
Naive bayes score after scaling :  0.935672514619883
```