```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


dataset = pd.read_csv("/content/drive/MyDrive/cancer.csv")
dataset.head()
```

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

+ Code + Text

```
dataset.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
```

```
 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
```

We can see from the information above that the id and unnamed:32 columns are not relevant, so we can eliminate them.

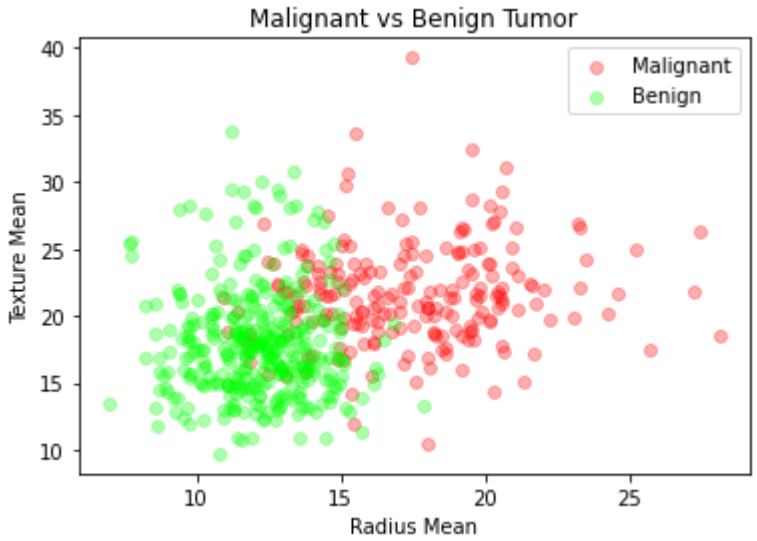
```
dataset = dataset.drop(["id"], axis = 1)
dataset = dataset.drop(["Unnamed: 32"], axis = 1)
```

Malignant Tumor Dataframe and Benign Tumor Dataframe

```
M = dataset[dataset.diagnosis == "M"]
B = dataset[dataset.diagnosis == "B"]
```

We shall now examine malignant and benign tumors by examining their average radius and texture.

```
plt.title("Malignant vs Benign Tumor")
plt.xlabel("Radius Mean")
plt.ylabel("Texture Mean")
plt.scatter(M.radius_mean, M.texture_mean, color = "red", label = "Malignant", alpha = 0.3)
plt.scatter(B.radius_mean, B.texture_mean, color = "lime", label = "Benign", alpha = 0.3)
plt.legend()
plt.show()
```



Preprocessing

Now, malignant tumors will be assigned a value of ‘1’ and benign tumors will be assigned a value of ‘0’.

```
dataset.diagnosis = [1 if i == "M" else 0 for i in dataset.diagnosis]
```

We now divide our dataframe into x and y components. The x variable includes all independent predictor factors, whereas the y

variable provides the diagnostic prediction.

```
x = dataset.drop(["diagnosis"], axis = 1)
y = dataset.diagnosis.values
```

Data Normalization

```
# Normalization:
x = (x - np.min(x)) / (np.max(x) - np.min(x))
```

Test Train Split

After that, we'll use the train test split module from the sklearn package to divide the dataset into training and testing sections.

```
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)
```

Sklearn Gaussian Naive Bayes Model

Now we'll import and instantiate the Gaussian Naive Bayes module from SKlearn GaussianNB. To fit the model, we may pass x_train and y_train.

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train, y_train)

     GaussianNB()
```

Accuracy

The following accuracy score reflects how successfully our Sklearn Gaussian Naive Bayes model predicted cancer using the test data.

```
print("Naive Bayes score: ",nb.score(x_test, y_test))
```

```
Naive Bayes score:  0.935672514619883
```