Cancer Detection

In [2]:
```
import pandas as pd
```

In [3]:
```
pwd
```

Out[3]:
```
'/Users/pratik'
```

Breast Cancer Dataset

In [894…
```
input_file = ("/Users/pratik/Desktop/Harrisburg University programs/Courses/Late Fall Co
data = pd.read_csv(input_file)
```

In [895…
```
data
```

Out[895]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	87139402	B	12.32	12.39	78.85	464.1	0.10280
1	8910251	B	10.60	18.95	69.28	346.4	0.09688
2	905520	B	11.04	16.83	70.92	373.2	0.10770
3	868871	B	11.28	13.39	73.00	384.8	0.11640
4	9012568	B	15.19	13.21	97.65	711.8	0.07963
...	...	...	...	...	...	...	...
564	911320502	B	13.17	18.22	84.28	537.3	0.07466
565	898677	B	10.26	14.71	66.20	321.6	0.09882
566	873885	M	15.28	22.41	98.92	710.6	0.09057
567	911201	B	14.53	13.98	93.86	644.2	0.10990
568	9012795	M	21.37	15.10	141.30	1386.0	0.10010

569 rows × 32 columns

In [896…
```
data = data.drop(columns = ["id"], axis = 1)
```

In [897…
```
data.describe()
```

Out[897]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400

8 rows × 30 columns

In [898…
```
list(data)
```

Out[898]:
```
['diagnosis',
 'radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
 'points_mean',
 'symmetry_mean',
 'dimension_mean',
 'radius_se',
 'texture_se',
 'perimeter_se',
 'area_se',
 'smoothness_se',
 'compactness_se',
 'concavity_se',
 'points_se',
 'symmetry_se',
 'dimension_se',
 'radius_worst',
 'texture_worst',
 'perimeter_worst',
 'area_worst',
 'smoothness_worst',
 'compactness_worst',
 'concavity_worst',
 'points_worst',
 'symmetry_worst',
 'dimension_worst']
```

Finding Levels of Tumor, which are Malignant and which are Benign. Checking the Diagnosis column and factorizing it for Benign and Malignant.

In [899…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   diagnosis          569 non-null    object 
 1   radius_mean        569 non-null    float64
 2   texture_mean       569 non-null    float64
 3   perimeter_mean     569 non-null    float64
 4   area_mean          569 non-null    float64
 5   smoothness_mean    569 non-null    float64
 6   compactness_mean   569 non-null    float64
 7   concavity_mean     569 non-null    float64
 8   points_mean        569 non-null    float64
 9   symmetry_mean      569 non-null    float64
 10  dimension_mean     569 non-null    float64
 11  radius_se          569 non-null    float64
 12  texture_se         569 non-null    float64
 13  perimeter_se       569 non-null    float64
 14  area_se            569 non-null    float64
 15  smoothness_se      569 non-null    float64
 16  compactness_se     569 non-null    float64
 17  concavity_se       569 non-null    float64
```

```
 18  points_se           569 non-null    float64
 19  symmetry_se         569 non-null    float64
 20  dimension_se        569 non-null    float64
 21  radius_worst        569 non-null    float64
 22  texture_worst       569 non-null    float64
 23  perimeter_worst     569 non-null    float64
 24  area_worst          569 non-null    float64
 25  smoothness_worst    569 non-null    float64
 26  compactness_worst   569 non-null    float64
 27  concavity_worst     569 non-null    float64
 28  points_worst        569 non-null    float64
 29  symmetry_worst      569 non-null    float64
 30  dimension_worst     569 non-null    float64
dtypes: float64(30), object(1)
memory usage: 137.9+ KB
```

In [914…
```
list(data)
```

Out[914]:
```
['diagnosis',
 'radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
 'points_mean',
 'symmetry_mean',
 'dimension_mean',
 'radius_se',
 'texture_se',
 'perimeter_se',
 'area_se',
 'smoothness_se',
 'compactness_se',
 'concavity_se',
 'points_se',
 'symmetry_se',
 'dimension_se',
 'radius_worst',
 'texture_worst',
 'perimeter_worst',
 'area_worst',
 'smoothness_worst',
 'compactness_worst',
 'concavity_worst',
 'points_worst',
 'symmetry_worst',
 'dimension_worst']
```

Scaling the data.

In [912…
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled = scaler.fit_transform(data.drop(columns=["diagnosis"]))
print(scaled)
```

```
[[0.25268588 0.0906324  0.24227766 ... 0.32271478 0.24876799 0.08310376]
 [0.17128118 0.31247886 0.17614539 ... 0.27237113 0.27104278 0.136626  ]
 [0.19210564 0.24078458 0.18747841 ... 0.25536082 0.28247585 0.15590975]
 ...
 [0.3927777  0.42948935 0.38096883 ... 0.42130584 0.31736645 0.27994228]
 [0.35728146 0.14440311 0.34600235 ... 0.36735395 0.20520402 0.15125279]
 [0.68100715 0.18227934 0.67383042 ... 0.67560137 0.22964715 0.20739866]]
```

Creating dataframe with scaled data

In [915…]:
```
data_new = pd.DataFrame(scaled, columns = ['radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
 'points_mean',
 'symmetry_mean',
 'dimension_mean',
 'radius_se',
 'texture_se',
 'perimeter_se',
 'area_se',
 'smoothness_se',
 'compactness_se',
 'concavity_se',
 'points_se',
 'symmetry_se',
 'dimension_se',
 'radius_worst',
 'texture_worst',
 'perimeter_worst',
 'area_worst',
 'smoothness_worst',
 'compactness_worst',
 'concavity_worst',
 'points_worst',
 'symmetry_worst',
 'dimension_worst'])
```

In [916…]:
```
data_new.head()
```

Out[916]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	con
0	0.252686	0.090632	0.242278	0.135992	0.452920	0.154684	
1	0.171281	0.312479	0.176145	0.086066	0.399476	0.292375	
2	0.192106	0.240785	0.187478	0.097434	0.497156	0.179928	
3	0.203464	0.124450	0.201852	0.102354	0.575697	0.289001	
4	0.388518	0.118363	0.372193	0.241060	0.243748	0.153242	

5 rows × 30 columns

In [917…]:
```
data_new.describe()
```

Out[917]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	0.338222	0.323965	0.332935	0.216920	0.394785	0.260601
std	0.166787	0.145453	0.167915	0.149274	0.126967	0.161992
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.223342	0.218465	0.216847	0.117413	0.304595	0.139685
50%	0.302381	0.308759	0.293345	0.172895	0.390358	0.224679
75%	0.416442	0.408860	0.416765	0.271135	0.475490	0.340531

max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 30 columns

In [913…
```
data["diagnosis"].value_counts()
```

Out[913]:
```
Benign       357
Malignant    212
Name: diagnosis, dtype: int64
```

In [918…
```
data["diagnosis"] = data["diagnosis"].replace(["B", "M"],["Benign", "Malignant"])
data["diagnosis"]
```

Out[918]:
```
0         Benign
1         Benign
2         Benign
3         Benign
4         Benign
          ...
564       Benign
565       Benign
566    Malignant
567       Benign
568    Malignant
Name: diagnosis, Length: 569, dtype: object
```

In [919…
```
data["diagnosis"]. value_counts()
```

Out[919]:
```
Benign       357
Malignant    212
Name: diagnosis, dtype: int64
```

Randomization using 12345

In [921…
```
import random
target = data["diagnosis"]
random.seed(12345)
indx = random.sample(range(0, 569), 569)
data_new_rand = data_new.iloc[indx]
target_rand = target.iloc[indx]
```

Splitting data into 80% Training and 20% Testing

In [922…
```
list(data_new_rand)
```

Out[922]:
```
['radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
 'points_mean',
 'symmetry_mean',
 'dimension_mean',
 'radius_se',
 'texture_se',
 'perimeter_se',
 'area_se',
 'smoothness_se',
 'compactness_se',
 'concavity_se',
 'points_se',
```

```
  'symmetry_se',
  'dimension_se',
  'radius_worst',
  'texture_worst',
  'perimeter_worst',
  'area_worst',
  'smoothness_worst',
  'compactness_worst',
  'concavity_worst',
  'points_worst',
  'symmetry_worst',
  'dimension_worst']
```

In [923…
```
from sklearn.model_selection import train_test_split

target = data["diagnosis"]

y = target_rand
x = data_new_rand  ## data_new_rand does not have the target variable as it was removed

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=1
```

Using K Nearest Neighbors Algorithm for Classification|

In [924…
```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=21)
model = neigh.fit(x_train, y_train)
```

In [925…
```
y_predict = model.predict(x_test)
```

```
/Users/pratik/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_classificatio
n.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the
default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0,
this behavior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no longer
be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [926…
```
print(confusion_matrix(y_test, y_predict)*100)
```

```
[[7400  100]
 [ 500 3400]]
```

In [927…
```
print(accuracy_score(y_test,y_predict)*100)
```

```
94.73684210526315
```

The K-Nearest Neighbor classifier using k value of 21, gives us an accuracy of 94.74% in detecting the Breast Cancer.