Breast Cancer Detection Project

By Yordanos Simegnew Muche

1. Importing The Necessary Libraries

In [1]:
```
# importing the necessary libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from datetime import datetime
pd.set_option("display.max_columns",None)
```

2. Loading The Dataset and Data Exploration

In [2]:
```
# Loading the Dataset
df = pd.read_csv("C:\\Users\\yozil\\Desktop\\My projects\\3.0 brest cancer dete
```

In [3]:
```
# displaying sample records
df.sample(3)
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_n
556	924964	B	10.16	19.59	64.73	311.7	0.1
276	8911230	B	11.33	14.16	71.79	396.6	0.0
450	9111596	B	11.87	21.54	76.83	432.0	0.0

In [4]:
```
# shape of the dataset
df.shape
```

Out[4]:
```
(569, 33)
```

In [5]:
```
# General information about the dataset
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]:
```
# columns
df.columns
```

Out[6]:
```
Index(['id', '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', 'Unnamed: 32'],
      dtype='object')
```

In [7]:
```
# number of categorical columns
len(df.select_dtypes("object").columns)
```

Out[7]:
```
1
```

In [8]:
```
# we only have one categorical column, let's see this categorical column
df.select_dtypes("object").columns
```

Out[8]:
```
Index(['diagnosis'], dtype='object')
```

In [9]:
```
# number of numerical columns
len(df.select_dtypes(["int64","float64"]).columns)
```

Out[9]:
```
32
```

In [10]:
```
# we have 32 numerical columns and let's see them
df.select_dtypes(["int64","float64"]).columns
```

Out[10]:
```
Index(['id', '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', 'Unnamed: 32'],
      dtype='object')
```

In [11]:
```
# statistical summary in numerical columns
df.describe()
```

Out[11]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.00000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.09636
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.01406
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.05263
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.08637
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.09587
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.10530
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.16340

3. Data Cleaning

3.1 Standardizing column names

In [12]:
```
# here we make all column names to be in title case format
def title_maker(column_name):
    return column_name.str.title()
```

In [13]:
```
# now let's apply our title maker function to our column names
df.columns = title_maker(df.columns)
```

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

Out[14]:
```
Index(['Id', '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', 'Unnamed: 32'],
      dtype='object')
```

3.2 Standardizing Values in the Text columns

In [15]:
```
# let's make sure all the text entries are standardized by making them all tit
def text_maker(df):
    return df.str.title()
```

In [16]:
```
# Now let's apply our text maker function to the text columns.
df[df.select_dtypes("object").columns] =df.select_dtypes("object").apply(text_
```

3.3 Removing Unecessary Space from Values in the text columns(if any)

In [17]:
```
# Now let's make sure there is no unecessary space in the values of text colum
def space_remover(text_column):
    return text_column.str.strip()
```

In [18]:
```
# let's apply the space remover function to our text columns.
df[df.select_dtypes("object").columns] = df.select_dtypes("object").apply(spac
```

3.4 Removing Duplicated Records (if any)

In [19]:
```
# first let's check the existance of duplicated records
df.duplicated().any()
```

Out[19]:
```
False
```

In [20]:
```
df.duplicated().sum()
```

Out[20]:
```
0
```

The above result show us there is no duplicated record in our dataset.

3.5 Handling Missing Values(if any)

In [21]:
```
# first let's check the existance of missing values in our dataset
df.isnull().any()
```

Out[21]:
```
Id                         False
Diagnosis                  False
Radius_Mean                False
Texture_Mean               False
Perimeter_Mean             False
Area_Mean                  False
Smoothness_Mean            False
Compactness_Mean           False
Concavity_Mean             False
Concave Points_Mean        False
Symmetry_Mean              False
Fractal_Dimension_Mean     False
Radius_Se                  False
Texture_Se                 False
Perimeter_Se               False
Area_Se                    False
Smoothness_Se              False
Compactness_Se             False
Concavity_Se               False
Concave Points_Se          False
Symmetry_Se                False
Fractal_Dimension_Se       False
Radius_Worst               False
Texture_Worst              False
Perimeter_Worst            False
Area_Worst                 False
Smoothness_Worst           False
Compactness_Worst          False
Concavity_Worst            False
Concave Points_Worst       False
Symmetry_Worst             False
Fractal_Dimension_Worst    False
Unnamed: 32                 True
dtype: bool
```

In [22]:
```
# we have missing value in our last column which is (unnamed: 32), let's see h
df.isnull().sum()
```

Out[22]:
```
Id                         0
Diagnosis                  0
Radius_Mean                0
Texture_Mean               0
Perimeter_Mean             0
Area_Mean                  0
Smoothness_Mean            0
Compactness_Mean           0
Concavity_Mean             0
Concave Points_Mean        0
Symmetry_Mean              0
Fractal_Dimension_Mean     0
Radius_Se                  0
Texture_Se                 0
Perimeter_Se               0
Area_Se                    0
Smoothness_Se              0
Compactness_Se             0
Concavity_Se               0
Concave Points_Se          0
Symmetry_Se                0
Fractal_Dimension_Se       0
Radius_Worst               0
Texture_Worst              0
Perimeter_Worst            0
Area_Worst                 0
Smoothness_Worst           0
Compactness_Worst          0
Concavity_Worst            0
Concave Points_Worst       0
Symmetry_Worst             0
Fractal_Dimension_Worst    0
Unnamed: 32              569
dtype: int64
```

In [23]:
```
# we have 569 null values in this column, we handle this missing values by rem
df.drop("Unnamed: 32", axis =1, inplace = True)
df.sample(3)
```

Out[23]:

	Id	Diagnosis	Radius_Mean	Texture_Mean	Perimeter_Mean	Area_Mean	Smoothness
340	89813	B	14.42	16.54	94.15	641.2	
322	894855	B	12.86	13.32	82.82	504.8	
408	90524101	M	17.99	20.66	117.80	991.7	

In [24]:
```
# now let's check the shape of our dataset
df.shape
```

Out[24]:
```
(569, 32)
```

3.5 Handling an outliers

In [25]:
```
# for the case of this project we assign outliers as values above and below 4 s
def outlier_limits(col):
    mean = col.mean()
    std = col.std()
    upper_limit = mean + 4 * std
    lower_limit = mean - 4 * std
    return upper_limit, lower_limit
```

In [26]:
```
df.select_dtypes(["int64","float64"]).apply(outlier_limits)
```

Out[26]:

	Id	Radius_Mean	Texture_Mean	Perimeter_Mean	Area_Mean	Smoothness_Mean
0	5.304542e+08	28.223487	36.493792	189.164958	2062.545620	0.152617
1	-4.697105e+08	0.031096	2.085505	-5.226891	-752.767413	0.040104

In [27]:
```
# now let's see the outlier records
def outlier_records(dataframe):
    outliers_df = pd.DataFrame()
    for col in dataframe.columns:
        mean = dataframe[col].mean()
        std = dataframe[col].std()
        upper_limit = mean + 4 * std
        lower_limit = mean - 4 * std
        outliers_rec = dataframe[(dataframe[col] > upper_limit) | (dataframe[c
        outliers_df = pd.concat([outliers_df, outliers_rec])
    return outliers_df.drop_duplicates()
```

In [28]:
```
# first let's see how many outlier records we have.
len(outlier_records(df.select_dtypes(["int64","float64"]),))
```

Out[28]: 43

In [29]:
```
# we have 43 outlier records in the data set let's see them
outlier_records(df.drop("Id",axis = 1).select_dtypes(["int64","float64"]))
```

Out[29]:

	Radius_Mean	Texture_Mean	Perimeter_Mean	Area_Mean	Smoothness_Mean	Compactness_
239	17.460	39.28	113.40	920.6	0.09812	0.
180	27.220	21.87	182.10	2250.0	0.10940	0.
212	28.110	18.47	188.50	2499.0	0.11420	0.
461	27.420	26.27	186.90	2501.0	0.10840	0.
504	9.268	12.87	61.49	248.7	0.16340	0.
78	20.180	23.97	143.70	1245.0	0.12860	0.
108	22.270	19.67	152.80	1509.0	0.13260	0.
122	24.250	20.20	166.20	1761.0	0.14470	0.
152	9.731	15.34	63.78	300.2	0.10720	0.
25	17.140	16.40	116.00	912.7	0.11860	0.
3	11.420	20.38	77.58	386.1	0.14250	0.
505	9.676	13.14	64.12	272.5	0.12550	0.
12	19.170	24.80	132.40	1123.0	0.09740	0.
192	9.720	18.22	60.73	288.1	0.06950	0.
473	12.270	29.97	77.42	465.4	0.07699	0.
561	11.200	29.37	70.67	386.0	0.07449	0.
368	21.710	17.25	140.90	1546.0	0.09384	0.
213	17.420	25.56	114.50	948.0	0.10060	0
314	8.597	18.60	54.09	221.2	0.10740	0.
42	19.070	24.81	128.30	1104.0	0.09081	0.
190	14.220	23.12	94.37	609.9	0.10750	0.
290	14.410	19.73	96.03	651.0	0.08757	0.
68	9.029	17.33	58.79	250.5	0.10660	0.
376	10.570	20.22	70.15	338.3	0.09073	0.
146	11.800	16.58	78.99	432.0	0.10910	0.
351	15.750	19.22	107.10	758.6	0.12430	0.
71	8.888	14.64	58.79	244.0	0.09783	0.
176	9.904	18.06	64.60	302.4	0.09699	0.
265	20.730	31.12	135.70	1419.0	0.09469	0
352	25.730	17.46	174.20	2010.0	0.11490	0.
9	12.460	24.04	83.97	475.9	0.11860	0.
379	11.080	18.83	73.30	361.6	0.12160	0.
562	15.220	30.62	103.40	716.9	0.10480	0.
323	20.340	21.51	135.90	1264.0	0.11700	0.

In [30]:
```
# this are the outlier records in our dataframe, and we handle this outliers by
outliers_index = outlier_records(df.drop("Id",axis = 1).select_dtypes(["int64"
outliers_index
```

Out[30]:
```
Index([239, 180, 212, 461, 504,  78, 108, 122, 152,  25,   3, 505,  12, 192,
       473, 561, 368, 213, 314,  42, 190, 290,  68, 376, 146, 351,  71, 176,
       265, 352,   9, 379, 562, 323],
      dtype='int64')
```

In [31]:
```
# Now let's remove this outliers
df.drop(outliers_index,inplace = True)
```

In [32]:
```
# shape of new dataframe
df.shape
```

Out[32]:
```
(535, 32)
```

In [33]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 535 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Id                       535 non-null    int64  
 1   Diagnosis                535 non-null    object 
 2   Radius_Mean              535 non-null    float64
 3   Texture_Mean             535 non-null    float64
 4   Perimeter_Mean           535 non-null    float64
 5   Area_Mean                535 non-null    float64
 6   Smoothness_Mean          535 non-null    float64
 7   Compactness_Mean         535 non-null    float64
 8   Concavity_Mean           535 non-null    float64
 9   Concave Points_Mean      535 non-null    float64
 10  Symmetry_Mean            535 non-null    float64
 11  Fractal_Dimension_Mean   535 non-null    float64
 12  Radius_Se                535 non-null    float64
 13  Texture_Se               535 non-null    float64
 14  Perimeter_Se             535 non-null    float64
 15  Area_Se                  535 non-null    float64
 16  Smoothness_Se            535 non-null    float64
 17  Compactness_Se           535 non-null    float64
 18  Concavity_Se             535 non-null    float64
 19  Concave Points_Se        535 non-null    float64
 20  Symmetry_Se              535 non-null    float64
 21  Fractal_Dimension_Se     535 non-null    float64
 22  Radius_Worst             535 non-null    float64
 23  Texture_Worst            535 non-null    float64
 24  Perimeter_Worst          535 non-null    float64
 25  Area_Worst               535 non-null    float64
 26  Smoothness_Worst         535 non-null    float64
 27  Compactness_Worst        535 non-null    float64
 28  Concavity_Worst          535 non-null    float64
 29  Concave Points_Worst     535 non-null    float64
 30  Symmetry_Worst           535 non-null    float64
 31  Fractal_Dimension_Worst  535 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 137.9+ KB
```

In [34]:
```
# Now let's fix the index
df.reset_index(inplace = True)
```

In [35]:
```
# sample records
df.sample(3)
```

Out[35]:

	index	Id	Diagnosis	Radius_Mean	Texture_Mean	Perimeter_Mean	Area_Mean	Smoot
471	501	91504	M	13.82	24.49	92.33	595.9	
183	199	877500	M	14.45	20.22	94.49	642.7	
383	411	905520	B	11.04	16.83	70.92	373.2	

In [36]:
```
# now let's drop the index columns
df.drop("index", axis = 1, inplace = True)
```

In [37]:
```
# general information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 535 entries, 0 to 534
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Id                       535 non-null    int64  
 1   Diagnosis                535 non-null    object 
 2   Radius_Mean              535 non-null    float64
 3   Texture_Mean             535 non-null    float64
 4   Perimeter_Mean           535 non-null    float64
 5   Area_Mean                535 non-null    float64
 6   Smoothness_Mean          535 non-null    float64
 7   Compactness_Mean         535 non-null    float64
 8   Concavity_Mean           535 non-null    float64
 9   Concave Points_Mean      535 non-null    float64
 10  Symmetry_Mean            535 non-null    float64
 11  Fractal_Dimension_Mean   535 non-null    float64
 12  Radius_Se                535 non-null    float64
 13  Texture_Se               535 non-null    float64
 14  Perimeter_Se             535 non-null    float64
 15  Area_Se                  535 non-null    float64
 16  Smoothness_Se            535 non-null    float64
 17  Compactness_Se           535 non-null    float64
 18  Concavity_Se             535 non-null    float64
 19  Concave Points_Se        535 non-null    float64
 20  Symmetry_Se              535 non-null    float64
 21  Fractal_Dimension_Se     535 non-null    float64
 22  Radius_Worst             535 non-null    float64
 23  Texture_Worst            535 non-null    float64
 24  Perimeter_Worst          535 non-null    float64
 25  Area_Worst               535 non-null    float64
 26  Smoothness_Worst         535 non-null    float64
 27  Compactness_Worst        535 non-null    float64
 28  Concavity_Worst          535 non-null    float64
 29  Concave Points_Worst     535 non-null    float64
 30  Symmetry_Worst           535 non-null    float64
 31  Fractal_Dimension_Worst  535 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 133.9+ KB
```

In [38]:
```
# displaying sample records
df.sample(3)
```

Out[38]:

	Id	Diagnosis	Radius_Mean	Texture_Mean	Perimeter_Mean	Area_Mean	Smoothnes
38	856106	M	13.28	20.28	87.32	545.2	
139	871001501	B	13.00	20.78	83.51	519.4	
43	857155	B	12.05	14.63	78.04	449.3	

In [39]:
```
# Now let's export our cleaned data
df.to_csv("C:\\Users\\yozil\\Desktop\\My projects\\3.0 brest cancer detection
```

In [40]:
```
pd.read_csv("C:\\Users\\yozil\\Desktop\\My projects\\3.0 brest cancer detectio
```

Out[40]:

	Id	Diagnosis	Radius_Mean	Texture_Mean	Perimeter_Mean	Area_Mean	Smoothness
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84358402	M	20.29	14.34	135.10	1297.0	
4	843786	M	12.45	15.70	82.57	477.1	
...	...	...	...	...	...	...	
530	926424	M	21.56	22.39	142.00	1479.0	
531	926682	M	20.13	28.25	131.20	1261.0	
532	926954	M	16.60	28.08	108.30	858.1	
533	927241	M	20.60	29.33	140.10	1265.0	
534	92751	B	7.76	24.54	47.92	181.0	

535 rows × 32 columns

Here we are done with the DATA CLEANING task