ARTIFICIAL INTELLIGENCE

---

by Rhea Sawant

1. Datatset chosen is about diagnosis of breast cancer among females over a wide range of ages.

In [ ]:
```
# Mounted Google drive to be able to access the datatset uploaded in drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force
_remount=True).
```

---

2. Read the data file and check the datatype of all the variables.

Reading the data in the dataset.

In [ ]:
```
# Loaded the Pandas libraries with alias 'pd'
import pandas as pd
# pd.read_csv is used to read data from file 'filename.csv'
rawdata=pd.read_csv('/content/drive/My Drive/Colab Notebooks/AI College/breast_cancer_females _data.csv')
rawdata
```

Out[ ]:

	Gender	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis	age
0	F	17.99	10.38	122.80	1001.0	0.11840	0	30
1	F	20.57	17.77	132.90	1326.0	0.08474	0	31
2	F	19.69	21.25	130.00	1203.0	0.10960	0	32
3	F	11.42	20.38	77.58	386.1	0.14250	0	33
4	F	20.29	14.34	135.10	1297.0	0.10030	0	34
...	...	...	...	...	...	...	...	...
594	F	13.27	14.76	84.74	551.7	0.07355	1	53
595	F	13.45	18.30	86.60	555.1	0.10220	1	54
596	F	15.06	19.83	100.30	705.6	0.10390	0	35
597	F	20.26	23.03	132.40	1264.0	0.09078	0	35
598	NaN	10.96	17.62	70.79	365.6	0.09687	1	51

599 rows × 8 columns

In [ ]:
```
type(rawdata)
```

Out[ ]:
```
pandas.core.frame.DataFrame
```

To read the datatypes of all the variables.

In [ ]:
```
# dtypes() returns a Series with the data type of each column, columns with mixed datatype are given 'object' dat
rawdata.dtypes
```

Out[ ]:
```
Gender              object
mean_radius        float64
mean_texture       float64
mean_perimeter     float64
```

```
mean_area          float64
mean_smoothness    float64
diagnosis            int64
age                  int64
dtype: object
```

3. Convert one or two variable's datatype from float to integer or vice versa.

The following code will change the datatype of the column 'mean_radius' from float to integer and column 'diagnosis' from integer to float.

In [ ]:
```
rawdata["mean_radius"]=rawdata['mean_radius'].astype('int')
rawdata["diagnosis"]=rawdata['diagnosis'].astype('float')
rawdata
```

Out[ ]:

	Gender	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis	age
0	F	17	10.38	122.80	1001.0	0.11840	0.0	30
1	F	20	17.77	132.90	1326.0	0.08474	0.0	31
2	F	19	21.25	130.00	1203.0	0.10960	0.0	32
3	F	11	20.38	77.58	386.1	0.14250	0.0	33
4	F	20	14.34	135.10	1297.0	0.10030	0.0	34
...	...	...	...	...	...	...	...	...
594	F	13	14.76	84.74	551.7	0.07355	1.0	53
595	F	13	18.30	86.60	555.1	0.10220	1.0	54
596	F	15	19.83	100.30	705.6	0.10390	0.0	35
597	F	20	23.03	132.40	1264.0	0.09078	0.0	35
598	NaN	10	17.62	70.79	365.6	0.09687	1.0	51

599 rows × 8 columns

The above code has worked, this can be verified by comparing the datatypes of the two columns as below to previously described one.

In [ ]:
```
rawdata.dtypes
```

Out[ ]:
```
Gender              object
mean_radius          int64
mean_texture       float64
mean_perimeter     float64
mean_area          float64
mean_smoothness    float64
diagnosis          float64
age                  int64
dtype: object
```

In [ ]:
```
# info() returns information about a DataFrame including the index dtype and columns, non-null values and memory
rawdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 599 entries, 0 to 598
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Gender           598 non-null    object 
 1   mean_radius      599 non-null    int64  
 2   mean_texture     599 non-null    float64
 3   mean_perimeter   599 non-null    float64
 4   mean_area        599 non-null    float64
 5   mean_smoothness  599 non-null    float64
 6   diagnosis        599 non-null    float64
 7   age              599 non-null    int64  
dtypes: float64(5), int64(2), object(1)
```

```
memory usage: 37.6+ KB
```

4. Apply some techniques to clean the dataset(data wrangling). Mention which technique is used and why.

---

a) Duplicate vales are redundant as the same type of conclusion is drawn multiple times, hence it makes sense to erase all duplicate rows in a dataset.

In [ ]:
```
#drop_duplicates() returns DataFrame with duplicate rows removed
data=rawdata.drop_duplicates()
data
```

Out[ ]:

	Gender	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis	age
0	F	17	10.38	122.80	1001.0	0.11840	0.0	30
1	F	20	17.77	132.90	1326.0	0.08474	0.0	31
2	F	19	21.25	130.00	1203.0	0.10960	0.0	32
3	F	11	20.38	77.58	386.1	0.14250	0.0	33
4	F	20	14.34	135.10	1297.0	0.10030	0.0	34
...	...	...	...	...	...	...	...	...
594	F	13	14.76	84.74	551.7	0.07355	1.0	53
595	F	13	18.30	86.60	555.1	0.10220	1.0	54
596	F	15	19.83	100.30	705.6	0.10390	0.0	35
597	F	20	23.03	132.40	1264.0	0.09078	0.0	35
598	NaN	10	17.62	70.79	365.6	0.09687	1.0	51

599 rows × 8 columns

b) No conclusion can be drawn from Null values and hence null values are unnecessary and should be removed.

In [ ]:
```
# dropna() removes missing values
data.dropna()
```

Out[ ]:

	Gender	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis	age
0	F	17	10.38	122.80	1001.0	0.11840	0.0	30
1	F	20	17.77	132.90	1326.0	0.08474	0.0	31
2	F	19	21.25	130.00	1203.0	0.10960	0.0	32
3	F	11	20.38	77.58	386.1	0.14250	0.0	33
4	F	20	14.34	135.10	1297.0	0.10030	0.0	34
...	...	...	...	...	...	...	...	...
593	F	15	22.76	100.20	728.2	0.09200	0.0	52
594	F	13	14.76	84.74	551.7	0.07355	1.0	53
595	F	13	18.30	86.60	555.1	0.10220	1.0	54
596	F	15	19.83	100.30	705.6	0.10390	0.0	35
597	F	20	23.03	132.40	1264.0	0.09078	0.0	35

598 rows × 8 columns

There were no rows with null values in the dataset so no rows were dropped from the dataset.

c) Columns that don't specify any significant information for the analysis of in a aparticular direction can be dropped

In [ ]:
```
# If inplace= False, it returns a copy otherwise, does operation inplace and returns nothing
# axis=0 refers to rows, axis=1 refers to columns
to_drop=['Gender']
data.drop(to_drop, inplace=True, axis = 1)
```

```
data
```

Out[ ]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis	age
0	17	10.38	122.80	1001.0	0.11840	0.0	30
1	20	17.77	132.90	1326.0	0.08474	0.0	31
2	19	21.25	130.00	1203.0	0.10960	0.0	32
3	11	20.38	77.58	386.1	0.14250	0.0	33
4	20	14.34	135.10	1297.0	0.10030	0.0	34
...	...	...	...	...	...	...	...
594	13	14.76	84.74	551.7	0.07355	1.0	53
595	13	18.30	86.60	555.1	0.10220	1.0	54
596	15	19.83	100.30	705.6	0.10390	0.0	35
597	20	23.03	132.40	1264.0	0.09078	0.0	35
598	10	17.62	70.79	365.6	0.09687	1.0	51

599 rows × 7 columns

In [ ]:
```
data
```

Out[ ]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis	age
0	17	10.38	122.80	1001.0	0.11840	0.0	30
1	20	17.77	132.90	1326.0	0.08474	0.0	31
2	19	21.25	130.00	1203.0	0.10960	0.0	32
3	11	20.38	77.58	386.1	0.14250	0.0	33
4	20	14.34	135.10	1297.0	0.10030	0.0	34
...	...	...	...	...	...	...	...
594	13	14.76	84.74	551.7	0.07355	1.0	53
595	13	18.30	86.60	555.1	0.10220	1.0	54
596	15	19.83	100.30	705.6	0.10390	0.0	35
597	20	23.03	132.40	1264.0	0.09078	0.0	35
598	10	17.62	70.79	365.6	0.09687	1.0	51

599 rows × 7 columns

5. Use a data visualisation technique to visualise data for extracting meaningful information

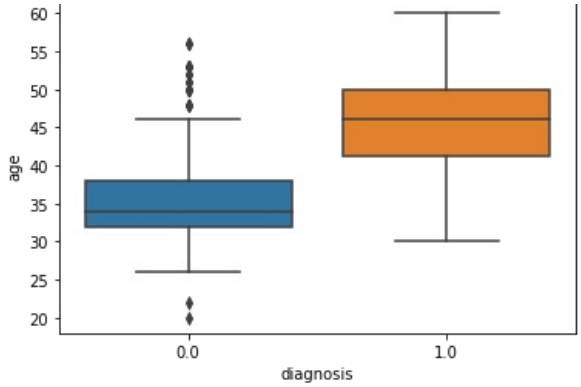
---

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable.

In [ ]:
```
import seaborn as sns

from matplotlib import pyplot as plt


# Distributions of observations within class

sns.boxplot( y=data["age"], x=data["diagnosis"] );

# Diagnosis=0 means diagnosed Breast Cancer negative and Daignsosis=1 means diagnosed positive

plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is de
precated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```



In [ ]:
```
# loc is used to access a group of rows and columns by label(s) or a boolean array
yesdiag=data.loc[data['diagnosis']==1]
# head() is used to return the first 'n' rows
yesdiag
```

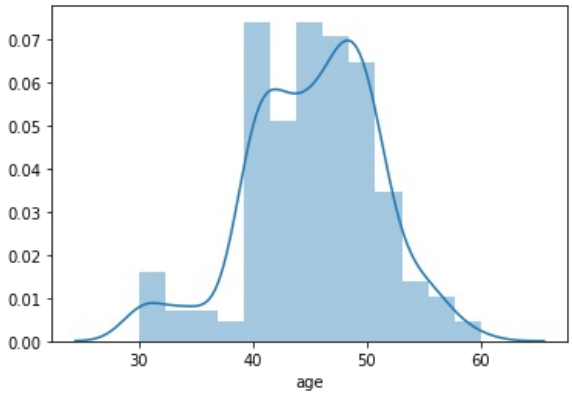
Out[ ]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis	age
19	13	14.36	87.46	566.3	0.09779	1.0	40
20	13	15.71	85.63	520.0	0.10750	1.0	45
21	9	12.44	60.34	273.9	0.10240	1.0	50
37	13	18.42	82.61	523.8	0.08983	1.0	36
46	8	16.84	51.71	201.9	0.08600	1.0	44
...	...	...	...	...	...	...	...
591	14	15.24	95.77	651.9	0.11320	1.0	50
592	14	24.02	94.57	662.7	0.08974	1.0	51
594	13	14.76	84.74	551.7	0.07355	1.0	53
595	13	18.30	86.60	555.1	0.10220	1.0	54
598	10	17.62	70.79	365.6	0.09687	1.0	51

374 rows × 7 columns

In [ ]:
```
# distplot() is used to flexibly plot a univariate distribution of observations
sns.distplot(yesdiag.age)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f82e4636c88>
```



In [ ]:
```
nodiag=data.loc[data['diagnosis']==0]
nodiag.head()
```

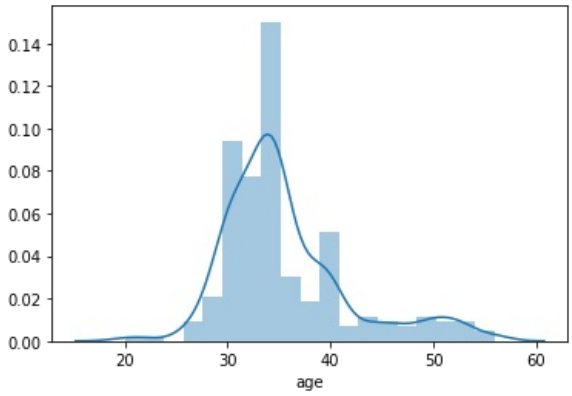
Out[ ]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis	age
0	17	10.38	122.80	1001.0	0.11840	0.0	30
1	20	17.77	132.90	1326.0	0.08474	0.0	31
2	19	21.25	130.00	1203.0	0.10960	0.0	32
3	11	20.38	77.58	386.1	0.14250	0.0	33

4	20	14.34	135.10	1297.0	0.10030	0.0	34

In [ ]:
```
sns.distplot(nodiag.age)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f82e455a0f0>
```



Conclusions drawn from the visualisation:

---

1. Females can be diagnosed with Breast Cancer at any age, however according to the dataset maximum women that tested positive were in their late 40s and early 50s.

2. Women starting from the age of 20 to late 50s show symptoms of breast cancer but are very less likely to test positive. It is more common among women older than 40 to suffer from it, but there are exceptions, young women may also test positive when the symptoms are severe otherwise young women do not show prominent symptoms. Reasons could be genetic mutations or hereditary issues.
