Author : Vaibhav Kumar

Assignment 2 : Exploratory Data Analysis with Haberman Dataset

Exercise :

1. Download Haberman Cancer Survival dataset from Kaggle. You may have to create a Kaggle account to donwload data. (https://www.kaggle.com/gilsousa/habermans-survival-data-set)
2. Perform a similar analysis as above on this dataset with the following sections:
3. High level statistics of the dataset: number of points, numer of features, number of classes, data-points per class. 4.Explain our objective.
4. Perform Univaraite analysis(PDF, CDF, Boxplot, Voilin plots) to understand which features are useful towards classification.
5. Perform Bi-variate analysis (scatter plots, pair-plots) to see if combinations of features are useful in classfication.
6. Write your observations in english as crisply and unambigously as possible. Always quantify your results.

In [1]:

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

importing code for google drive file upload

In [ ]:

```
from google.colab import files
uploaded = files.upload()
```

Choose File No file selected Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Reading the dataset using pandas...

In [2]:

```
data = pd.read_csv("haberman.csv")
```

In [6]:

```
print(data)
```

```
     age  year  nodes  status
0     30    64      1       1
1     30    62      3       1
2     30    65      0       1
3     31    59      2       1
4     31    65      4       1
..   ...   ...    ...     ...
301   75    62      1       1
302   76    67      0       1
303   77    65      3       1
304   78    65      1       2
305   83    58      2       2

[306 rows x 4 columns]
```

In this data we have year = Operation Year , nodes = axillary Nodes , status = Survive Status

In [7]:

```
print(data.shape)
```

```
(306, 4)
```

In [8]:

```
data.head()
print(data['status'])
#print[data['305']]
```

```
0      1
1      1
2      1
3      1
4      1
      ..
301    1
302    1
303    1
304    2
305    2
Name: status, Length: 306, dtype: int64
```

In [9]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 306 entries, 0 to 305
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   age     306 non-null    int64
 1   year    306 non-null    int64
 2   nodes   306 non-null    int64
 3   status  306 non-null    int64
dtypes: int64(4)
memory usage: 9.7 KB
```

In [10]:

```
data.describe()
```

Out[10]:

	age	year	nodes	status
count	306.000000	306.000000	306.000000	306.000000
mean	52.457516	62.852941	4.026144	1.264706
std	10.803452	3.249405	7.189654	0.441899
min	30.000000	58.000000	0.000000	1.000000
25%	44.000000	60.000000	0.000000	1.000000
50%	52.000000	63.000000	1.000000	1.000000
75%	60.750000	65.750000	4.000000	2.000000
max	83.000000	69.000000	52.000000	2.000000

In [11]:

```
data["age"].describe()
```

Out[11]:

```
count    306.000000
mean      52.457516
std       10.803452
min       30.000000
25%       44.000000
50%       52.000000
75%       60.750000
max       83.000000
Name: age, dtype: float64
```

In [12]:

```
data["nodes"].describe()
```

Out[12]:

```
count    306.000000
mean       4.026144
std        7.189654
min        0.000000
25%        0.000000
50%        1.000000
75%        4.000000
max       52.000000
Name: nodes, dtype: float64
```

What is axillary nodes ? Lymph nodes are small, bean-shaped organs that act as filters along the lymph fluid channels. As lymph fluid leaves the breast and eventually goes back into the bloodstream, the lymph nodes try to catch and trap cancer cells before they reach other parts of the body. Having cancer cells in the lymph nodes under your arm suggests an increased risk of the cancer spreading.In our data it is axillary nodes detected(0–52).

In [13]:

```
data["status"].describe()
```

Out[13]:

```
count    306.000000
mean       1.264706
std        0.441899
min        1.000000
25%        1.000000
50%        1.000000
75%        2.000000
max        2.000000
Name: status, dtype: float64
```

The dataset in the case study is Haberman Cancer Survival and do Exploratory Data Analysis

In [14]:

```
print(data.columns)
```

```
Index(['age', 'year', 'nodes', 'status'], dtype='object')
```

'data' is the actual variable which holds the maxtrix holding D{Xi,yi} ; X={age ,year , nodes } & Y ={status}. data is a matrix (306,4) .

In [15]:

```
data["status"].value_counts()
```

Out[15]:

```
1    225
2     81
Name: status, dtype: int64
```

In order to begin with EDA of a dataset , first task is to check if the dataset given is balanced or unbalanced .

It turns out the dataset is inbalanced because tere is a very high difference betwenn the class 1 ,2 ....

In [16]:

```
df_1=data[data['status']==1].copy()
df_2=data[data['status']==2].copy()
```

In [17]:

```
df_1.head()
```

Out[17]:

	age	year	nodes	status
0	30	64	1	1
1	30	62	3	1
2	30	65	0	1
3	31	59	2	1
4	31	65	4	1

In [18]:

```
df_2.head()
```

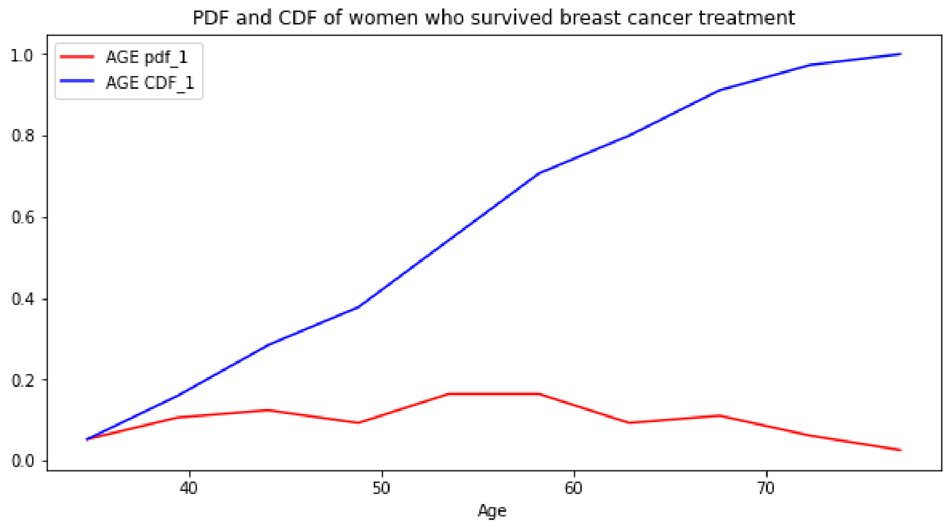
Out[18]:

	age	year	nodes	status
7	34	59	0	2
8	34	66	9	2
24	38	69	21	2
34	39	66	0	2
43	41	60	23	2

In [23]:

```
plt.figure(figsize=(10,5))
counts, bin_edges = np.histogram(df_1['age'], bins=10,
                                 density = True)
s =sum(counts)
pdf = counts/s
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf,'r',label='AGE pdf_1')
plt.plot(bin_edges[1:], cdf,'b',label='AGE CDF_1')
plt.xlabel('Age')
plt.title('PDF and CDF of women who survived breast cancer treatment')
plt.legend()
plt.show()
```

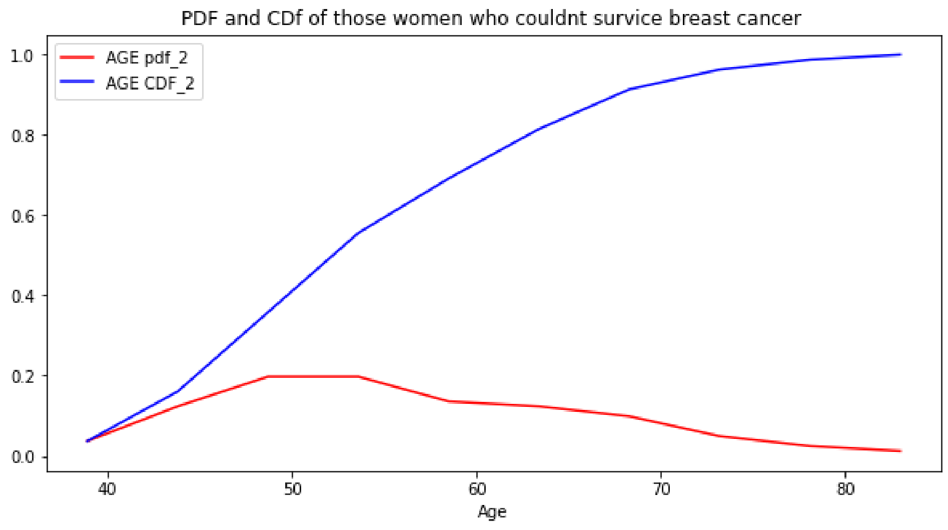
```
[0.05333333 0.10666667 0.12444444 0.09333333 0.16444444 0.16444444
 0.09333333 0.11111111 0.06222222 0.02666667]
[30.   34.7 39.4 44.1 48.8 53.5 58.2 62.9 67.6 72.3 77. ]
```



In [24]:

```
plt.figure(figsize=(10,5))
counts, bin_edges = np.histogram(df_2['age'], bins=10,
                                 density = True)
s =sum(counts)
pdf = counts/s
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf,'r',label='AGE pdf_2')
plt.plot(bin_edges[1:], cdf,'b',label='AGE CDF_2')
plt.xlabel('Age')
plt.title('PDF and CDf of those women who couldnt survice breast cancer ')
plt.legend()
plt.show()
```

```
[0.03703704 0.12345679 0.19753086 0.19753086 0.13580247 0.12345679
 0.09876543 0.04938272 0.02469136 0.01234568]
[34.   38.9 43.8 48.7 53.6 58.5 63.4 68.3 73.2 78.1 83. ]
```

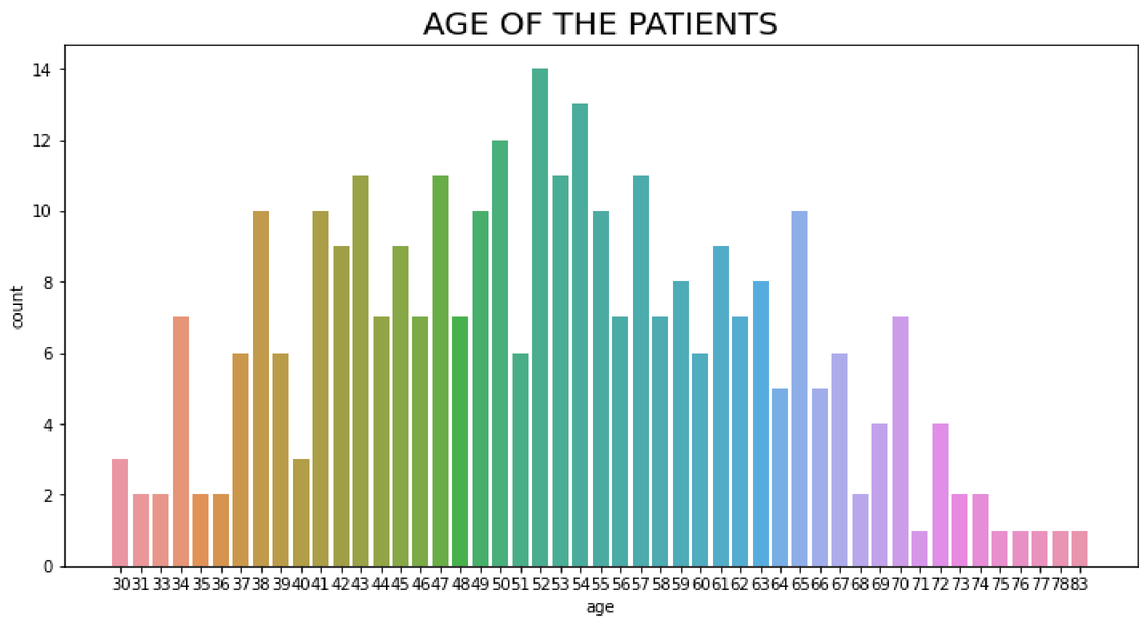


In [ ]:

```
plt.figure(figsize=(12,6))
sns.countplot(x='age',data=data)
plt.title("AGE OF THE PATIENTS",size=20)
plt.plot()
```

Out[ ]:

[ ]



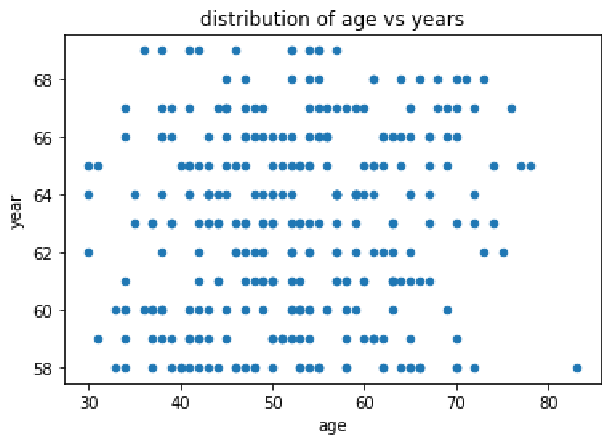
From the plot above it is observed that

- From 30 to 80 in age group , women suffer with breast cancer .
- For age 72 years has a decreasing rate after .
- Mostly women of age 52 are suffering from breast cancer
- Age of 50-60 , women tends to suffer a lot

2D scatter plot

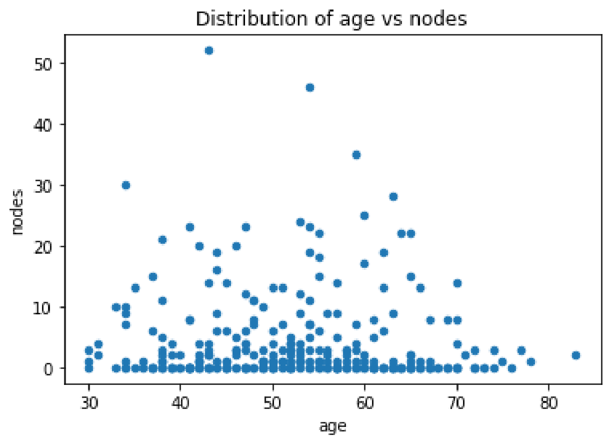
In [25]:

```
data.plot(kind='scatter', x='age', y='year')
plt.title('distribution of age vs years')
plt.show()
```



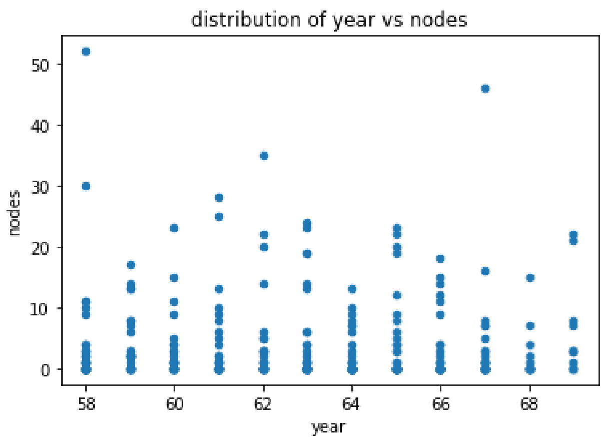
In [26]:

```
data.plot(kind='scatter', x='age', y='nodes')
plt.title('Distribution of age vs nodes')
plt.show()
```



In [27]:

```
data.plot(kind='scatter', x='year', y='nodes')
plt.title('distribution of year vs nodes')
plt.show()
```



In [ ]:

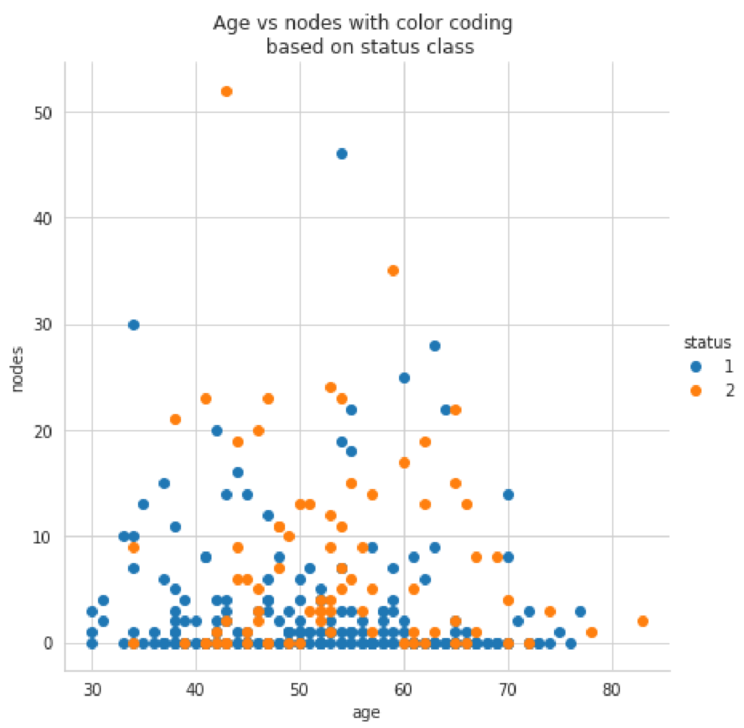
```
help(sns.set_style)
```

```
Help on function set_style in module seaborn.rcmod:

set_style(style=None, rc=None)
    Set the aesthetic style of the plots.

    This affects things like the color of the axes, whether a grid is
    enabled by default, and other aesthetic elements.

    Parameters
    ----------
    style : dict, None, or one of {darkgrid, whitegrid, dark, white, ticks}
        A dictionary of parameters or the name of a preconfigured set.
    rc : dict, optional
        Parameter mappings to override the values in the preset seaborn
        style dictionaries. This only updates parameters that are
        considered part of the style definition.

    Examples
    --------
    >>> set_style("whitegrid")

    >>> set_style("ticks", {"xtick.major.size": 8, "ytick.major.size": 8})

    See Also
    --------
    axes_style : return a dict of parameters or use in a ``with`` statement
                 to temporarily set the style.
    set_context : set parameters to scale plot elements
    set_palette : set the default color palette for figures
```

Using seaborn library to extract information with different colors

In [28]:

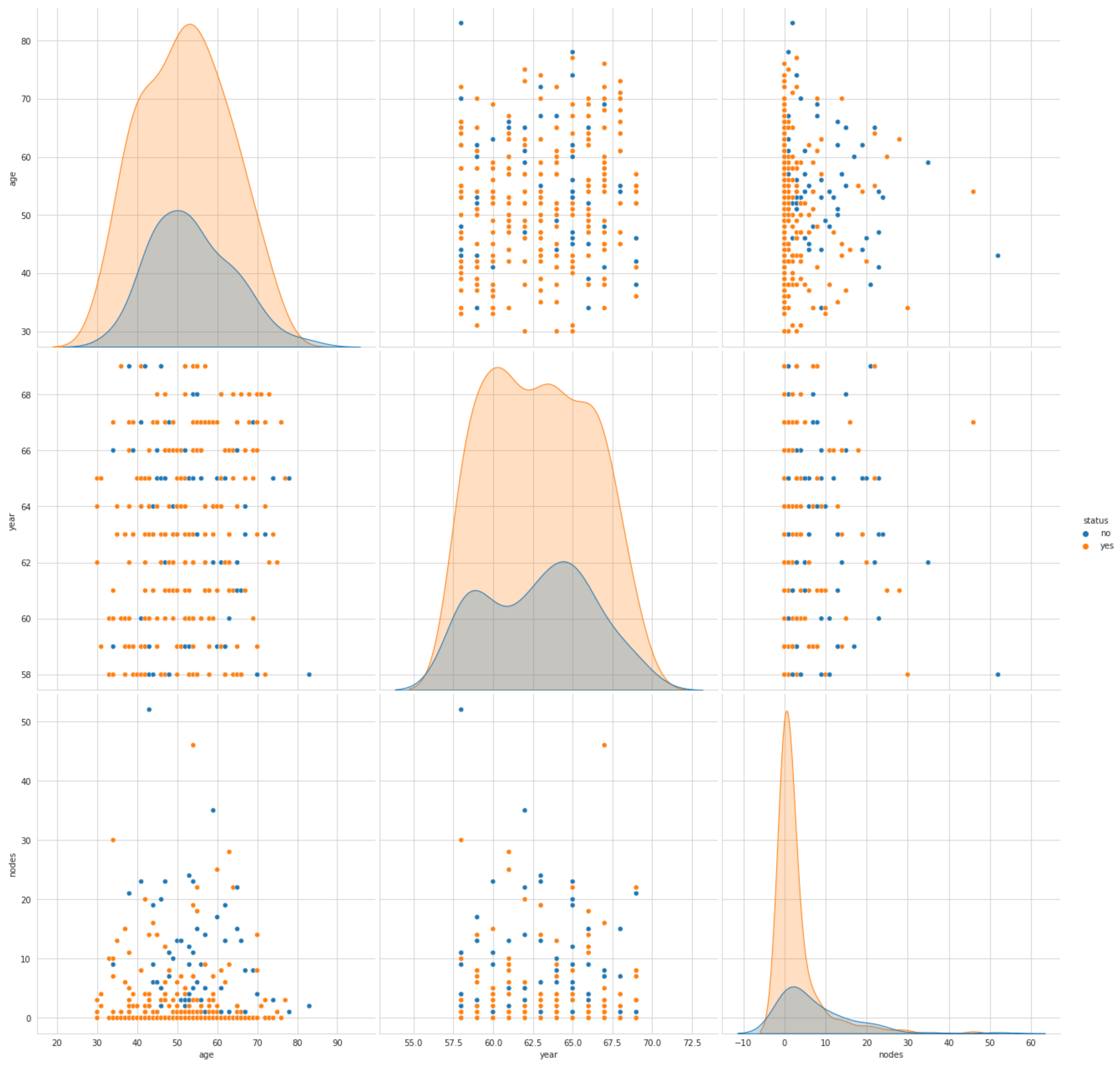
```
sns.set_style('whitegrid')
sns.FacetGrid(data, hue='status', height=6).map(plt.scatter, 'age', 'nodes').add_legend()
plt.title('Age vs nodes with color coding \n based on status class')
plt.show()
```



The direct information which can tell the realtionship or distribution between age and nodes cannot be interpreted using scatter plots

In [31]:

```
data['status'] = data['status'].map({1:"yes", 2:"no"})
data['status'] = data['status'].astype('category')

plt.close()
sns.set_style('whitegrid')
sns.pairplot(data, hue='status', height=6, vars=['age', 'year', 'nodes'])
plt.show()
```



PairPlots : It shows joint and marginal distributions for all pairwise relationships and for each variable, respectively

The different plots gives an idea how two features have their distribution spread across the axes . This observation is useful in understanding how the data can be in for of their behaviour of spread , the figure they make when they make a closed set of points which have same class spread over an area .

1.: It can be seen that the age of people having more age have breast cancer, it doesn't mean people having low age don't get but the pdf shows the gernal trend . 2.:The no of nodes varies by age . the distribution indicates , the no of women who got breast cancer have high age and high number of lympathic nodes .

3.:

In [32]:

```
survival_status_yes = data[data.status == 'yes']
survival_status_no = data[data.status == 'no']

survival_status_yes.head()
```

Out[32]:

	age	year	nodes	status
0	30	64	1	yes
1	30	62	3	yes
2	30	65	0	yes
3	31	59	2	yes
4	31	65	4	yes

In [33]:

```
survival_status_no.head()
```

Out[33]:

	age	year	nodes	status
7	34	59	0	no
8	34	66	9	no
24	38	69	21	no
34	39	66	0	no
43	41	60	23	no

In [34]:

```
status_yes = data.loc[data["status"]=="yes"]
status_no = data.loc[data["status"]=="no"]
status_yes.head()
```

Out[34]:

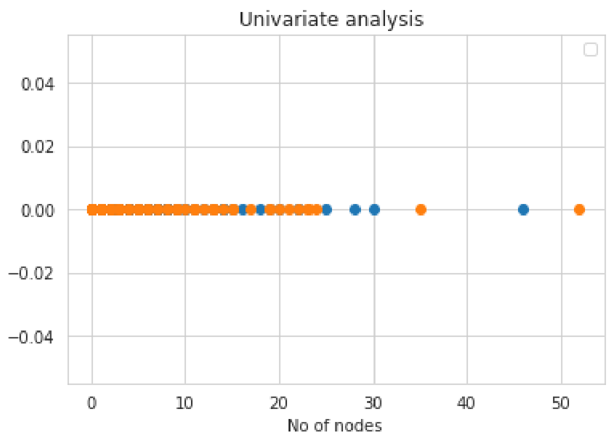
	age	year	nodes	status
0	30	64	1	yes
1	30	62	3	yes
2	30	65	0	yes
3	31	59	2	yes
4	31	65	4	yes

Univariate Analysis

In [36]:

```
plt.plot(status_yes["nodes"], np.zeros_like(status_yes["nodes"]), 'o')
plt.plot(status_no["nodes"], np.zeros_like(status_no["nodes"]), 'o')
plt.xlabel('No of nodes')
#plt.ylabel('Important var')
plt.title('Univariate analysis ')
plt.legend()
plt.show()
```

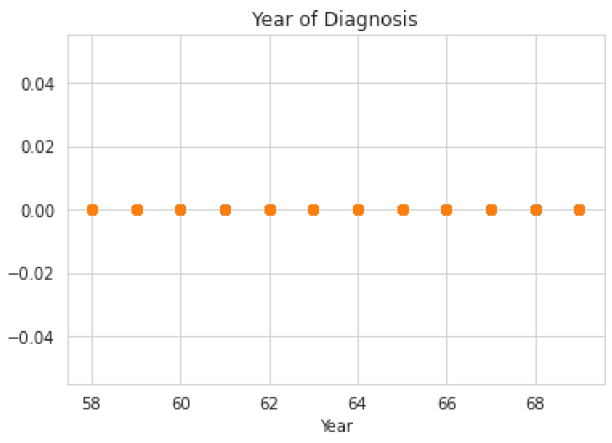
```
No handles with labels found to put in legend.
```



Cannot make sense of this 1D plot due to overlapping of data

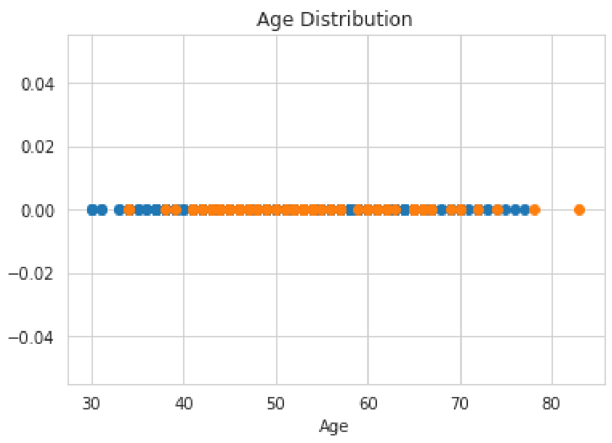
In [38]:

```
plt.plot(status_yes["year"], np.zeros_like(status_yes["year"]), 'o')
plt.plot(status_no["year"], np.zeros_like(status_no["year"]), 'o')
plt.xlabel('Year')
plt.title('Year of Diagnosis')
plt.show()
```



In [39]:

```
plt.plot(status_yes["age"], np.zeros_like(status_yes["age"]), 'o')
plt.plot(status_no["age"], np.zeros_like(status_no["age"]), 'o')
plt.xlabel('Age')
plt.title('Age Distribution')
plt.show()
```



Observation: Using nodes and age we can see some groups but there is no clear cluster. Separating survived from died is hard as they overlap.

From the 1st plot drawn , it was visible that people from age 30 to 70 are prone to have Breast cancer . where as some women in age of 40 , 57,58 ,66,72,74 have shown that its not occuring in them .

Mean, Variance and Std-dev

In [ ]:

```
#Mean, Variance, Std-deviation,
print("Means:")
print(np.mean(survival_status_yes["nodes"]))
print(np.mean(survival_status_no["nodes"]))
print(np.mean(survival_status_yes["age"]))
print(np.mean(survival_status_no["age"]))
print(np.mean(survival_status_yes["year"]))
print(np.mean(survival_status_no["year"]))
#Mean with an outlier.
print(np.mean(np.append(survival_status_yes["year"],50)))


print("\nStd-dev:");
print(np.std(survival_status_yes["nodes"]))
print(np.std(survival_status_no["nodes"]))
```

```
Means:
2.7911111111111113
7.45679012345679
52.01777777777778
53.67901234567901
62.86222222222222
62.82716049382716
62.80530973451327

Std-dev:
5.857258449412131
9.128776076761632
```

Median, Percentile, Quantile, IQR, MAD

In [ ]:

```python
#Median, Quantiles, Percentiles, IQR.
print("\nMedians:")
print(np.median(survival_status_yes["nodes"]))
print(np.median(survival_status_no["nodes"]))
print(np.median(survival_status_yes["age"]))
print(np.median(survival_status_no["age"]))
print(np.median(survival_status_yes["year"]))
print(np.median(survival_status_no["year"]))
print("Median with an outlier")

print(np.median(np.append(survival_status_yes["nodes"],50)))
print(np.median(np.append(survival_status_no["nodes"],50)))




print("\nQuantiles:")
print(np.percentile(survival_status_yes["nodes"],np.arange(0, 100, 25)))
print(np.percentile(survival_status_no["nodes"],np.arange(0, 100, 25)))


print("\n90th Percentiles:")
print(np.percentile(survival_status_yes["nodes"],90))
print(np.percentile(survival_status_no["nodes"],90))


from statsmodels import robust
print ("\nMedian Absolute Deviation")
print(robust.mad(survival_status_yes["nodes"]))
print(robust.mad(survival_status_no["nodes"]))
print(robust.mad(survival_status_yes["age"]))
print(robust.mad(survival_status_no["age"]))
print(robust.mad(survival_status_yes["year"]))
print(robust.mad(survival_status_no["year"]))
```

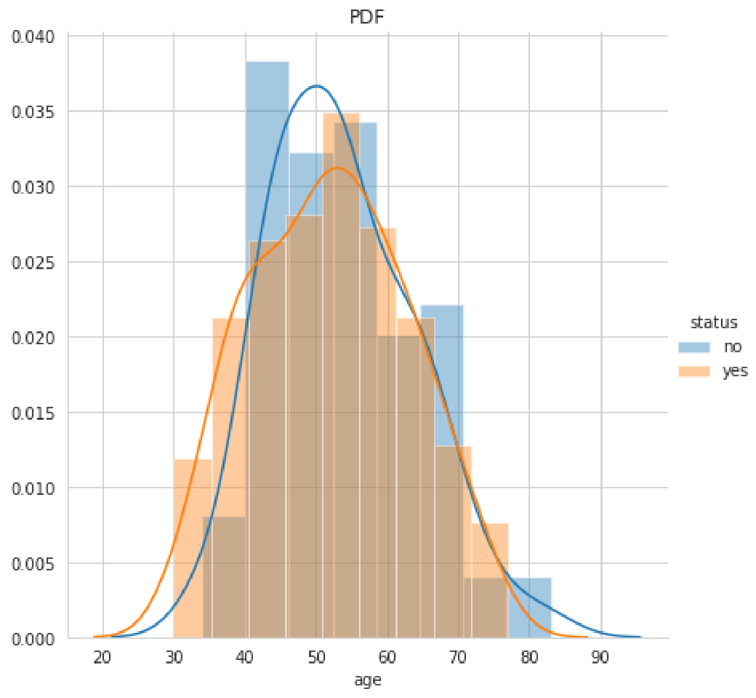
```
Medians:
0.0
4.0
52.0
53.0
63.0
63.0
Median with an outlier
0.0
4.0

Quantiles:
[0. 0. 0. 3.]
[ 0.  1.  4. 11.]

90th Percentiles:
8.0
20.0

Median Absolute Deviation
0.0
5.930408874022408
13.343419966550417
11.860817748044816
4.447806655516806
4.447806655516806
```

PDF, CDF

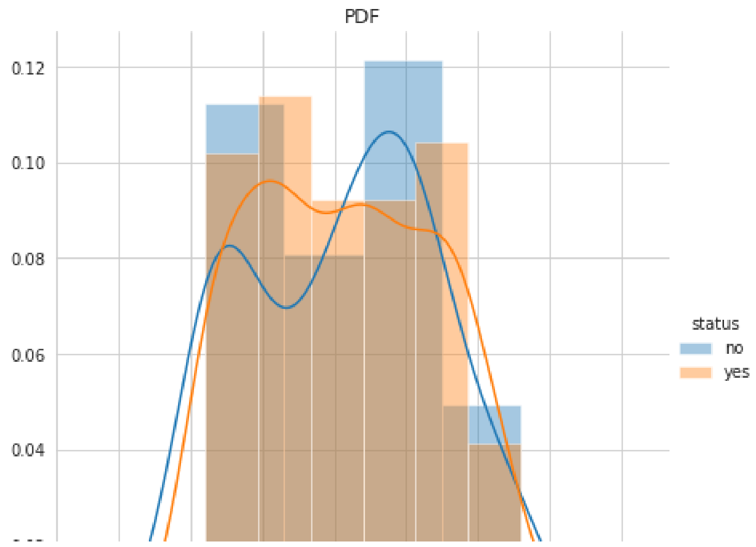
In [40]:

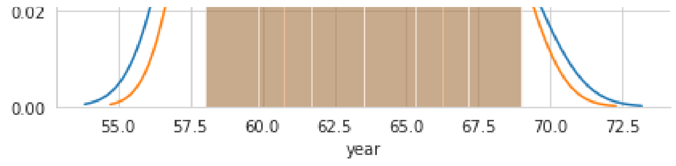
```
# PDFs
input_features = ['age', 'nodes' , 'year']


for index, input_features in enumerate(list(data.columns)[:-1]):
    figure = sns.FacetGrid(data, hue="status", height=6)
    figure.map(sns.distplot, input_features).add_legend()
    plt.title('PDF')
    plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your code to use either
`displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your code to use either
`displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
```

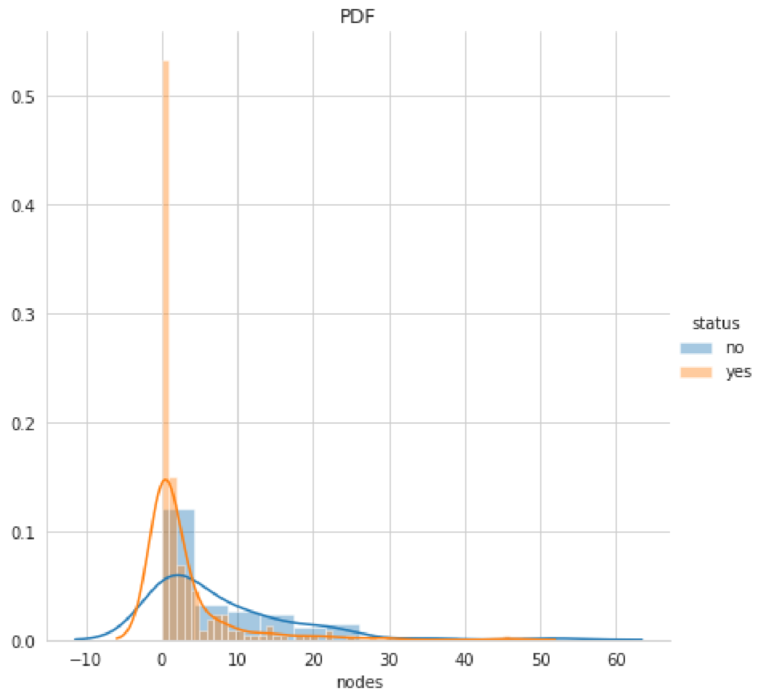


```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your code to use either
`displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your code to use either
`displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
```





```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your code to use either
`displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your code to use either
`displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
```



CDF : The CDF of a random variable is a method of describing the underlying distribution of random variable. It can be defined for any kind of random variable (discrete, continuous, mixed) etc. CDF calculates the cummulative probablity for a given x-value. CDF is used to determine the probablity that a random observation taken from a dataset will be less than or equal to a certain value. The difference between CDF and PDF is, PDF is density function where are CDF is probablity itself. Integration of PDF is CDF.

In [52]:

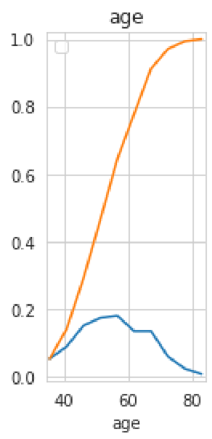
```
#Cdf
plt.figure(figsize=(20,5))
def draw_cdfs(no_of_bins):
    for index, input_features in enumerate(list(data.columns)[:-1]):

        plt.subplot(1,3, index+1)
        counts, bin_edges = np.histogram(data[input_features], bins=no_of_bins, density=True)
        pdf = counts/sum(counts)
        cdf = np.cumsum(pdf)

        print ("====",input_features,"====")
        print ("Bin Edges {}".format(bin_edges))
        print ("PDF {}".format(pdf))
        print ("CDF {}".format(cdf))
        # Plots
        plt.plot(bin_edges[1:], pdf, bin_edges[1:],cdf)
        plt.margins(0.02)
        plt.xlabel(input_features)
        plt.title(input_features)
        plt.legend()
        plt.show()
```

```
<Figure size 1440x360 with 0 Axes>
```

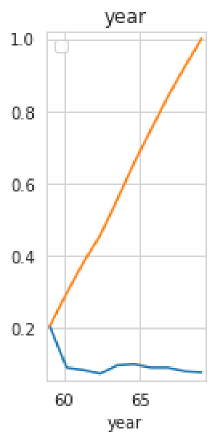
In [53]:

```
draw_cdfs(10)
```

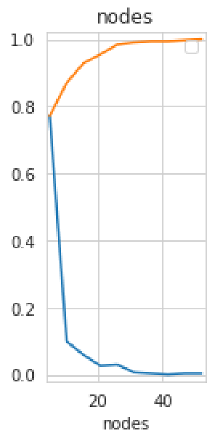
```
No handles with labels found to put in legend.
==== age ====
Bin Edges [30.  35.3 40.6 45.9 51.2 56.5 61.8 67.1 72.4 77.7 83. ]
PDF [0.05228758 0.08823529 0.1503268  0.17320261 0.17973856 0.13398693
 0.13398693 0.05882353 0.02287582 0.00653595]
CDF [0.05228758 0.14052288 0.29084967 0.46405229 0.64379085 0.77777778
 0.91176471 0.97058824 0.99346405 1.        ]
```



```
No handles with labels found to put in legend.
==== year ====
Bin Edges [58.  59.1 60.2 61.3 62.4 63.5 64.6 65.7 66.8 67.9 69. ]
PDF [0.20588235 0.09150327 0.08496732 0.0751634  0.09803922 0.10130719
 0.09150327 0.09150327 0.08169935 0.07843137]
CDF [0.20588235 0.29738562 0.38235294 0.45751634 0.55555556 0.65686275
 0.74836601 0.83986928 0.92156863 1.        ]
```



```
No handles with labels found to put in legend.
==== nodes ====
Bin Edges [ 0.   5.2 10.4 15.6 20.8 26.  31.2 36.4 41.6 46.8 52. ]
PDF [0.77124183 0.09803922 0.05882353 0.02614379 0.02941176 0.00653595
 0.00326797 0.         0.00326797 0.00326797]
CDF [0.77124183 0.86928105 0.92810458 0.95424837 0.98366013 0.99019608
 0.99346405 0.99346405 0.99673203 1.        ]
```

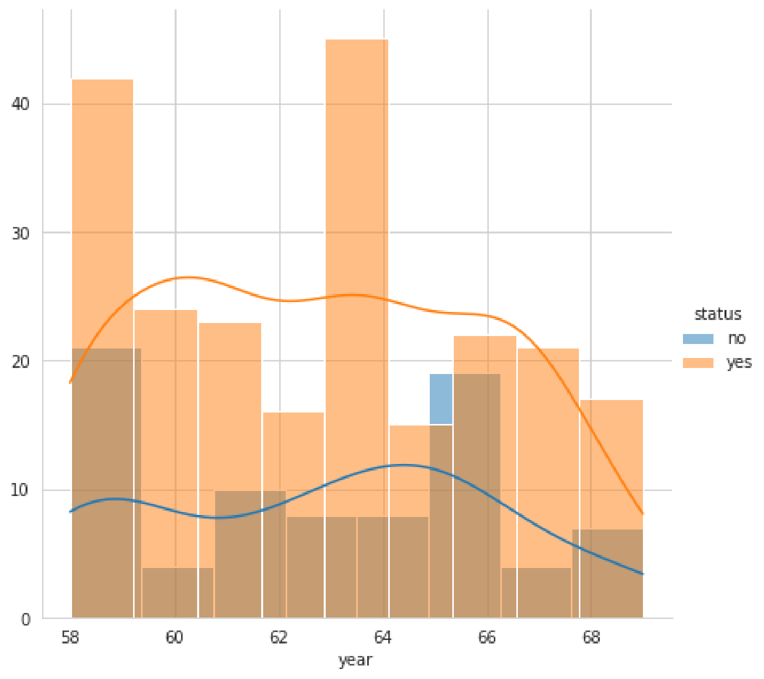


In [ ]:

```
sns.FacetGrid(data, hue='status', height=6) \
    .map(sns.histplot, 'year', kde=True) \
    .add_legend()
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7f0b6b3ba630>
```

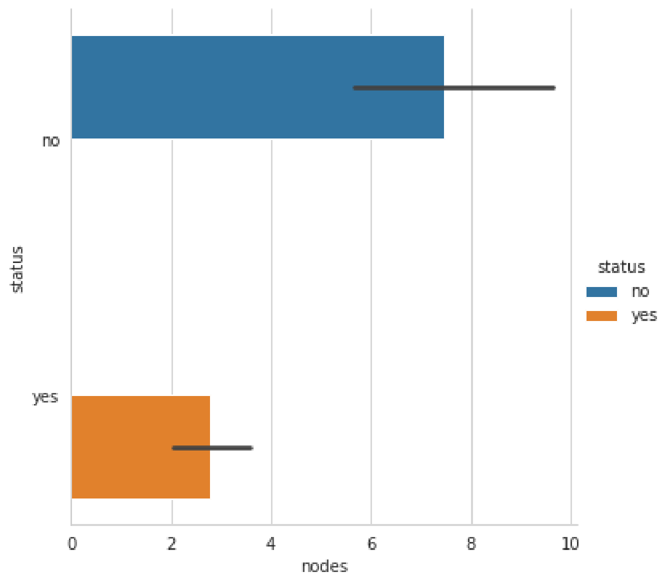


In [58]:

```
sns.catplot(data=data, kind="bar", x="nodes", y="status", hue="status")
```

Out[58]:

```
<seaborn.axisgrid.FacetGrid at 0x7f7f4a220208>
```



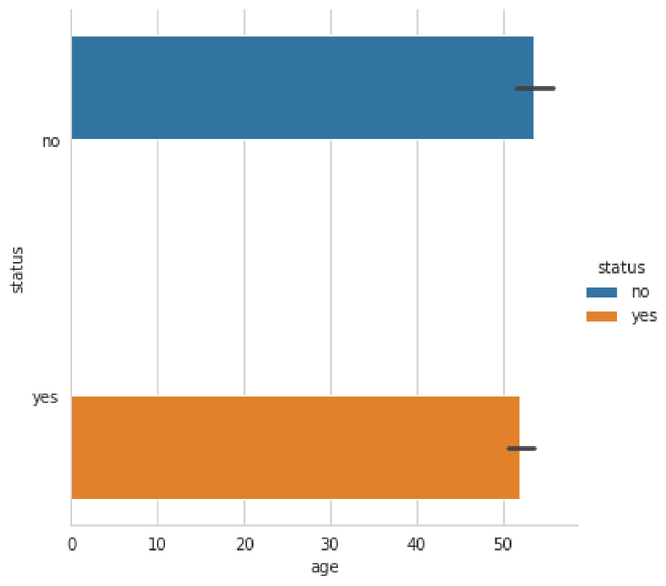
The bar chart gives a intition that the number of nodes in women who survived is less than those who couldnt survive ..

In [66]:

```
sns.catplot(data=data, kind="bar", x="age", y="status", hue="status")
```

Out[66]:

```
<seaborn.axisgrid.FacetGrid at 0x7f7f4059b390>
```



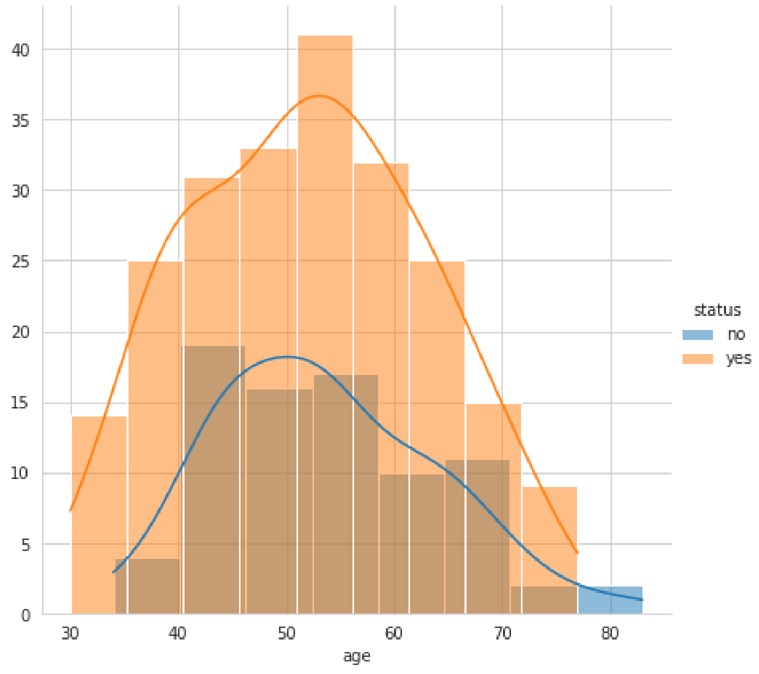
From the age bar chart , no intution can be taken since they appear to have same age mean

In [ ]:

```
sns.FacetGrid(data, hue='status', height=6) \
    .map(sns.histplot, 'age', kde=True) \
    .add_legend()
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7f0b6b525668>
```



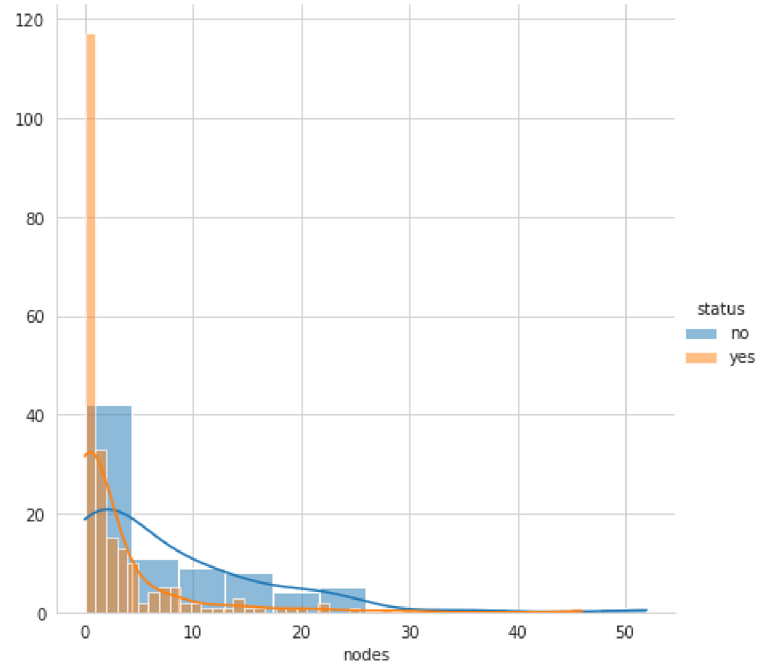
From the plot above it can be observed that the women having breast cancer at age from 30 to 60 is more than women not having breast cancer at age 30 to 60 . Since the area under the curve is larger for the status = true than compared to the curve where status =false

In [ ]:

```
sns.FacetGrid(data, hue='status', height=6) \
    .map(sns.histplot, 'nodes', kde=True) \
    .add_legend()
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7f0b70cb46a0>
```

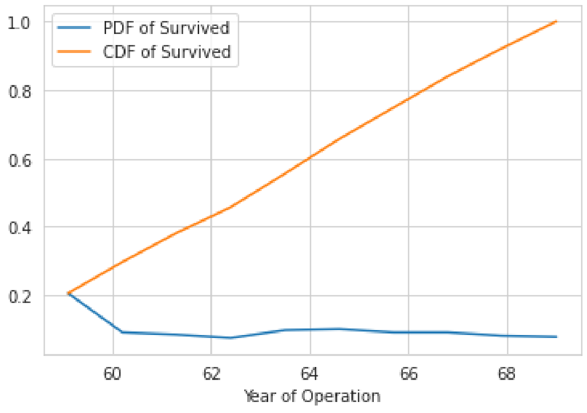


The area under the curve of those who couldnt survive the breast cancer is less than those who survived the breast cancer

In [62]:

```
counts, bin_edges = np.histogram(data['year'], bins=10, density=True)
pdf = counts/sum(counts)
print('bin_edges ', bin_edges)

print('pdf ', pdf)

cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)
plt.legend(['PDF of Survived', 'CDF of Survived'])
plt.xlabel('Year of Operation')
plt.show()
```

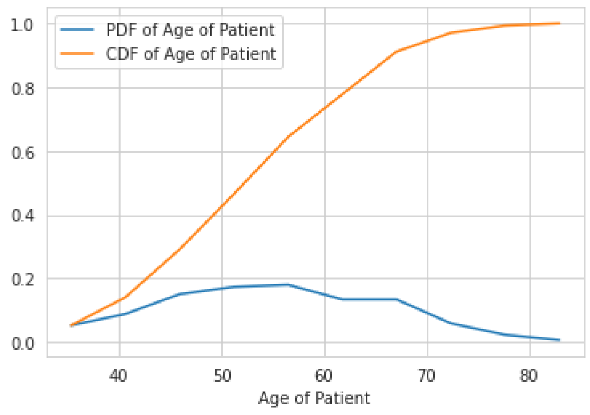
```
bin_edges  [58.  59.1 60.2 61.3 62.4 63.5 64.6 65.7 66.8 67.9 69. ]
pdf  [0.20588235 0.09150327 0.08496732 0.0751634  0.09803922 0.10130719
 0.09150327 0.09150327 0.08169935 0.07843137]
```



PDF and CDF of the women who survived the breast cancer vs the years of operations they undergo .

In [ ]:

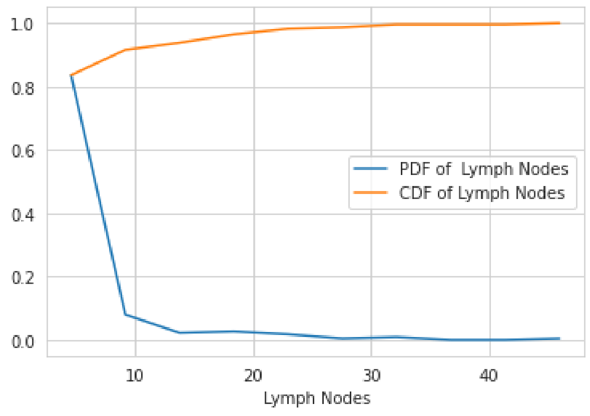
```
counts, bin_edges = np.histogram(data['age'], bins=10, density=True)
pdf = counts/sum(counts)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)
plt.legend(['PDF of Age of Patient', 'CDF of Age of Patient'])
plt.xlabel('Age of Patient')
plt.show()
```



Pdf and Cdf of age of the patient

In [63]:

```
counts, bin_edges = np.histogram(survival_status_yes['nodes'], bins=10, density=True)
pdf = counts/sum(counts)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)
plt.legend(['PDF of  Lymph Nodes', 'CDF of Lymph Nodes'])
plt.xlabel(' Lymph Nodes')
plt.show()
```



Box Plots

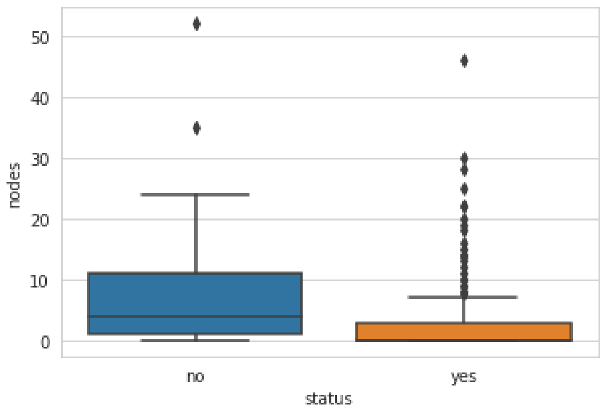
Box plots help in visualizing how the data is spread out The important things to be noted in the box plot are i) Q1 - 25th percentile ii) Q2 - Median or 50th percentile iii) Q3 - 75th Percentile

In [ ]:

```
sns.boxplot(x='status', y='nodes', data=data)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0b6b3ea908>
```



Median The median (middle quartile) marks the mid-point of the data and is shown by the line that divides the box into two parts. Half the scores are greater than or equal to this value and half are less.

Inter-quartile range The middle “box” represents the middle 50% of scores for the group. The range of scores from lower to upper quartile is referred to as the inter-quartile range. The middle 50% of scores fall within the inter-quartile range.

Upper quartile Seventy-five percent of the scores fall below the upper quartile.

Lower quartile Twenty-five percent of scores fall below the lower quartile.

Whiskers The upper and lower whiskers represent scores outside the middle 50%. Whiskers often (but not always) stretch over a wider range of scores than the middle quartile groups.

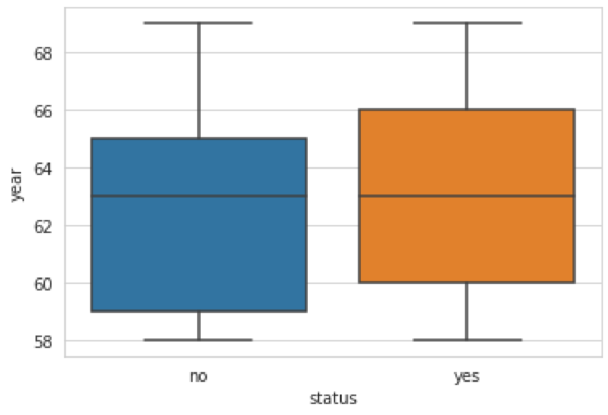
source : https://www.wellbeingatschool.org.nz/information-sheet/understanding-and-interpreting-box-plots

In [ ]:

```
sns.boxplot(x='status', y='year', data=data)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0b6b31ecf8>
```

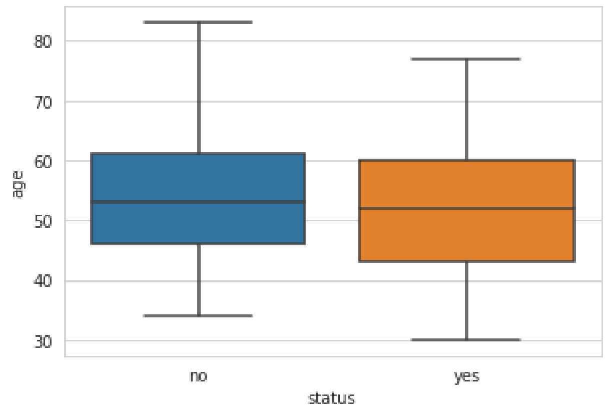


In [ ]:

```
sns.boxplot(x='status', y='age', data=data)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0b6b28fbe0>
```

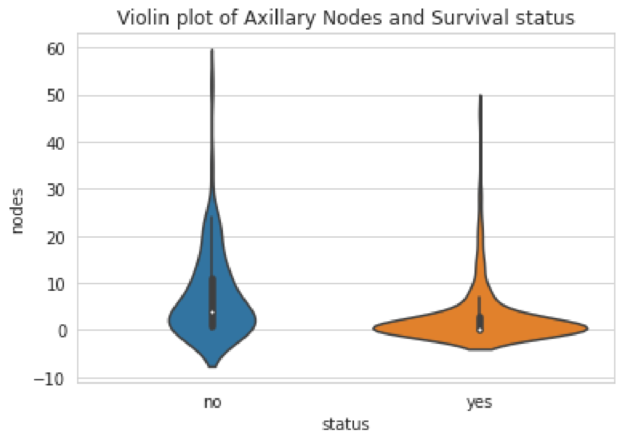


In box plots with wiskers four equal sized groups are made from the ordered scores.25% of all scores are placed in each group. The lines dividing the groups are called quartiles, and the groups are referred to as quartile groups. Usually we label these groups 1 to 4 starting at the bottom.

Violin Plots

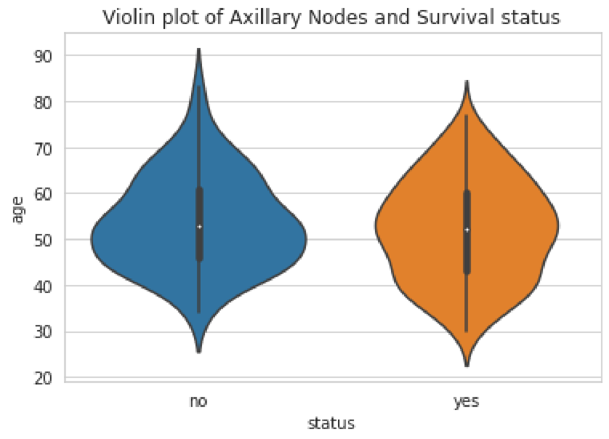
In [ ]:

```
sns.violinplot(x='status', y='nodes', data=data, size=8)
plt.title('Violin plot of Axillary Nodes and Survival status')
plt.show()
```



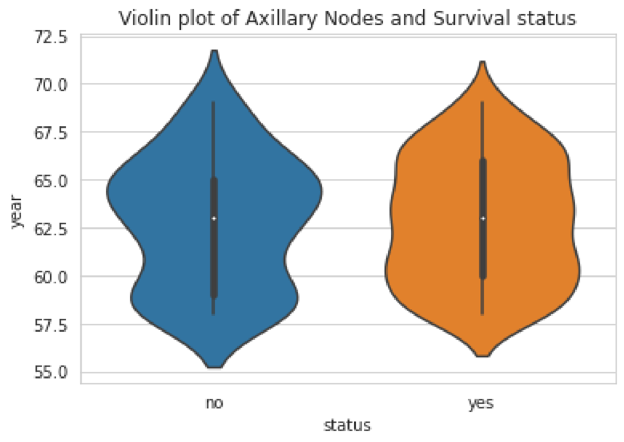
In [65]:

```
sns.violinplot(x='status', y='age', data=data, size=8)
plt.title('Violin plot of Axillary Nodes and Survival status')
#plt.legend(['Survived' , 'Not survived'])
plt.show()
```



In [ ]:

```
sns.violinplot(x='status', y='year', data=data, size=8)
plt.title('Violin plot of Axillary Nodes and Survival status')
plt.show()
```



From the violin plots we can see that, the count of lymph nodes for the survivors is mostly lying under less than 5. Survival chance is less for the patients before the treatment years 1959 - evident from third box plot (year vs status) Similiar patients surivival rate seems to be better for treatments after 1965 - evident from third box plot (year vs status) From CDFs (nodes/status) it is evident that nearly 70% or more patients have less than or equal to 5 lymph nodes as according to the first plot.

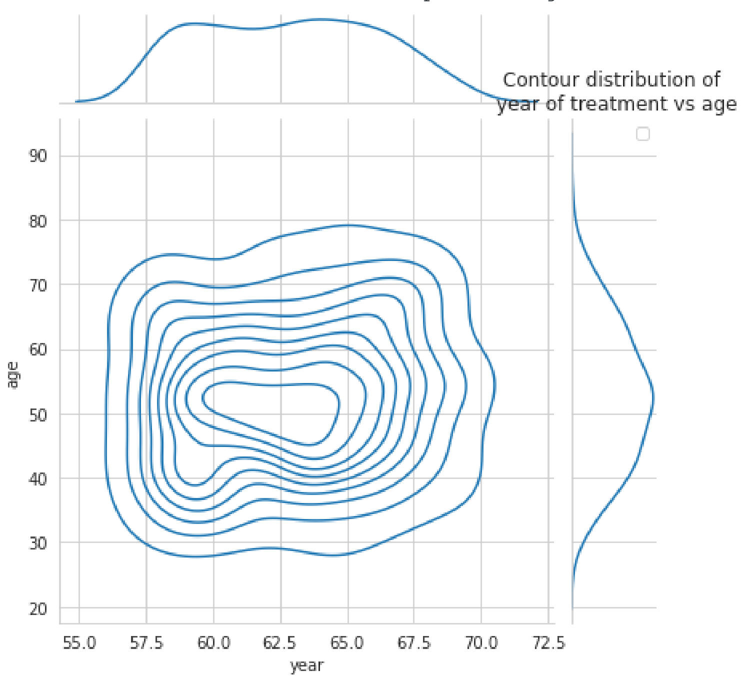
Contour plots

Contour plots are widely used to visualize density, altitudes or heights of the mountain as well as in the meteorological department.

In [61]:

```
sns.jointplot(x='year', y='age', data=data, kind="kde" )
plt.title('Contour distribution of \n year of treatment vs age')
plt.legend()
plt.show();
```

```
No handles with labels found to put in legend.
```

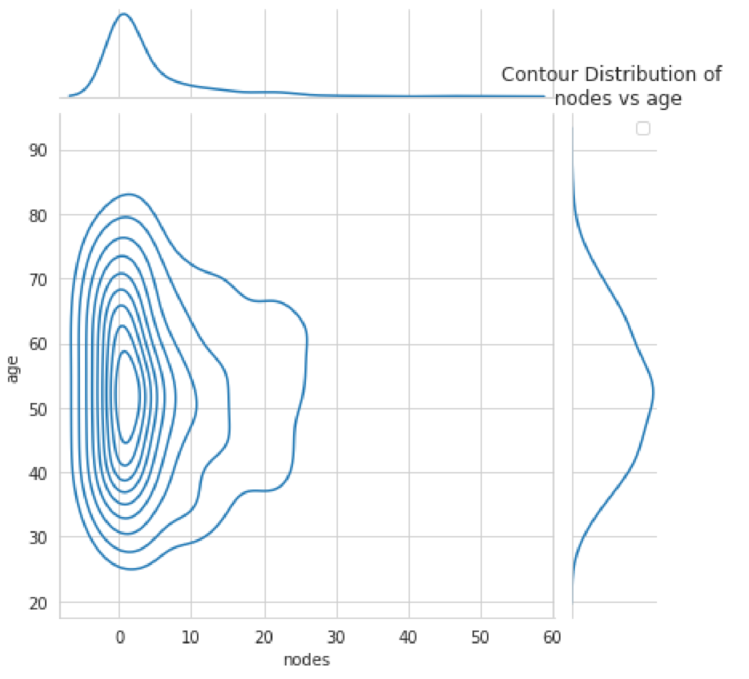


The plot above shows a gentle slope since there is large distance between contour which appears to be regular in some way .

In [60]:

```
sns.jointplot(x='nodes', y='age', data=data, kind="kde" )
plt.title('Contour Distribution of \n nodes vs age')
plt.legend()
plt.show();
```

```
No handles with labels found to put in legend.
```



As the above contour plot indicate step slope along the direction since the plots are very close to each other .