Exploratory Data Analysis for Cancer Dataset

In this case study we will see how to perform EDA on Haberman's Dataset. This dataset holds data of HealthCare Domain. The dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for Breast Cancer.

Why we perform EDA?

EDA: Exploratory Data Analysis is done on the dataset so that we can analyse data by using simple tools and techniques.

We do plotting of data so as to visually and understand data.

Objective: we perform EDA on the dataset to visually see the survived and non survived group of Cancerous patients.

Analysis of Dataset

In [1]:
```
# Importing the libraries required for data analysis..
import pandas as pd  # we import pandas library, we need pandas to impo
rt dataset
import seaborn as sns  # we import seaborn library, we need seaborn to
 visualize data statistically
import numpy as np  # we import numpy, we need numpy for scientific cal
culations
import matplotlib.pyplot as plt # we import matplotlib, we need matplot
lib to visualize the data
import warnings
warnings.filterwarnings("ignore")
```

```
In [2]: # we read the data which is in CSV (Comma_Separated_Values)format. and
         store this dataset in variable "cancerDetect"
        cancerDetect = pd.read_csv('haberman.csv')
```

```
In [3]: # we print top 5 rows of haberman's dataset which the cancerDetect vari
        able holds
        print(cancerDetect.head(5))
```

```
   30  64   1  1.1
0  30  62   3    1
1  30  65   0    1
2  31  59   2    1
3  31  65   4    1
4  33  58  10    1
```

```
In [4]: # we print the columns associated to haberman's dataset, the columns na
        mes(30,64,1,1.1) are in improper format.
        print(cancerDetect.columns)
```

```
Index(['30', '64', '1', '1.1'], dtype='object')
```

For better accessbility of data and to better visualize it

For better accessbility of data and to better visualize it, we rename the columns we are again importing our dataset which is in CSV (comma_separated_values)format. we use the argument "names", while importing dataset so that we can rename columns and we replace each column Name by the one mentioned in the list "col_Name". Here cancerDetect is in the form of a DataFrame and it holds data of age, year,nodes, status

```
In [5]: col_Names  = ["age", "year", "nodes", "status"]
        cancerDetect = pd.read_csv('haberman.csv', names = col_Names)
```

```
In [6]: # we print the top 5 rows of our dataset
```

```
cancerDetect.head(5)
```

Out[6]:

	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 [7]:

```
# we print the shape of dataset so that we can understand that how many
 rows and how many columns do we have in our dataset
# here we have 306 rows and 4 columns
print(cancerDetect.shape)
```

```
(306, 4)
```

In [8]:

```
print(cancerDetect.columns)
```

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

we are printing the column Names details of dataset : Here Age: Age of the patient at the time of operation, Year: Year at which operation was done Nodes: The number of nodes detected in that part, status: this is our class label, if it is ==1 then we say that the patient is survived. if ==2 then we say the patient is died

In [9]:

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

Out[9]:

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

a) using value_counts() method, we get the frequency count of those people who are survived or who are not survived after operation.

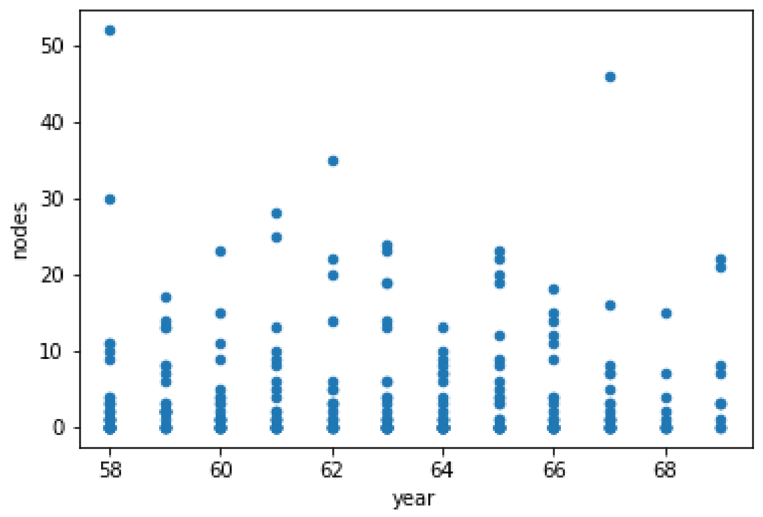
b) Here we get un-balanced dataset, here status is our class Label and if status==1 then it indicates that the patients were survived after operation and status==2 indicates that the patients were died after operation. we get 225 as frequency count of those patients who Survived and 81 frequency count for those who died. this is our objective to see if a patient has survived or not after operation </font>

In [10]:
```
cancerDetect['status'] = cancerDetect['status'].map({1:"T", 2:"F"})
```

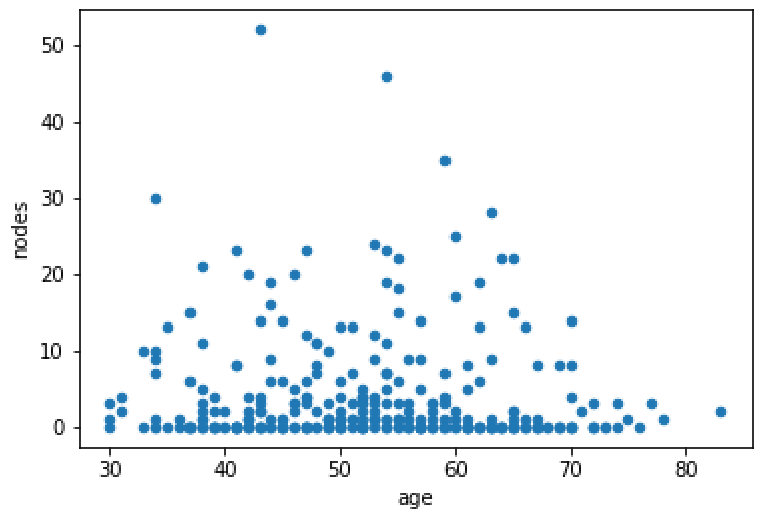
here we used the map function to convert the numerical Survival status(1,2) of patients to boolean status in the form of True or False, we assign T for 1 and F for 2 and display the results we store the results again into status column of our dataframe cancerDetect Henceforth we will denote T: True and F: False

Scatter plot for analyzing the spread of data

In [11]:
```
# we plot a scatter plot to see the spread of our data, keeping x-axis as Year
 and y-axis as Nodes
#Observations: we see that the Year varies from 1958 to 1968 and nodes varies
 from 0 to 50
cancerDetect.plot(kind='scatter', x='year', y='nodes')
plt.show()
```

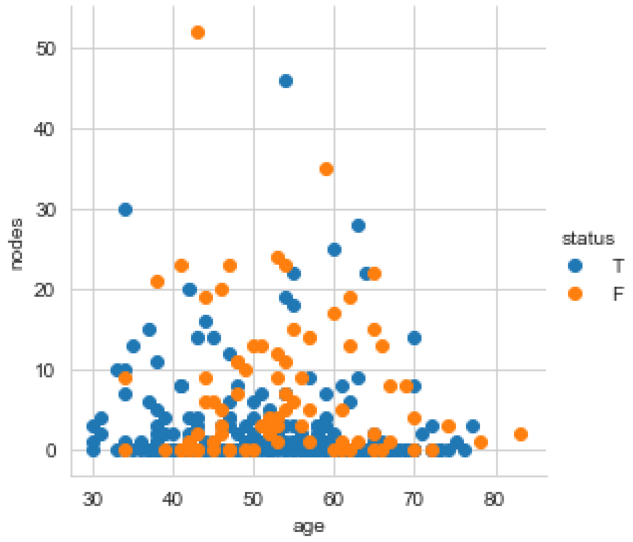


In [12]:
```
# Also I have plotted a scatter plot to see the spread of data, keeping x-axis
 as Age and y-axis as Nodes
#Observations: I see that the Age from 38 to 68 has high amount of nodes
cancerDetect.plot(kind='scatter', x='age', y='nodes')
plt.show()
```



Using Seaborn to visualize data

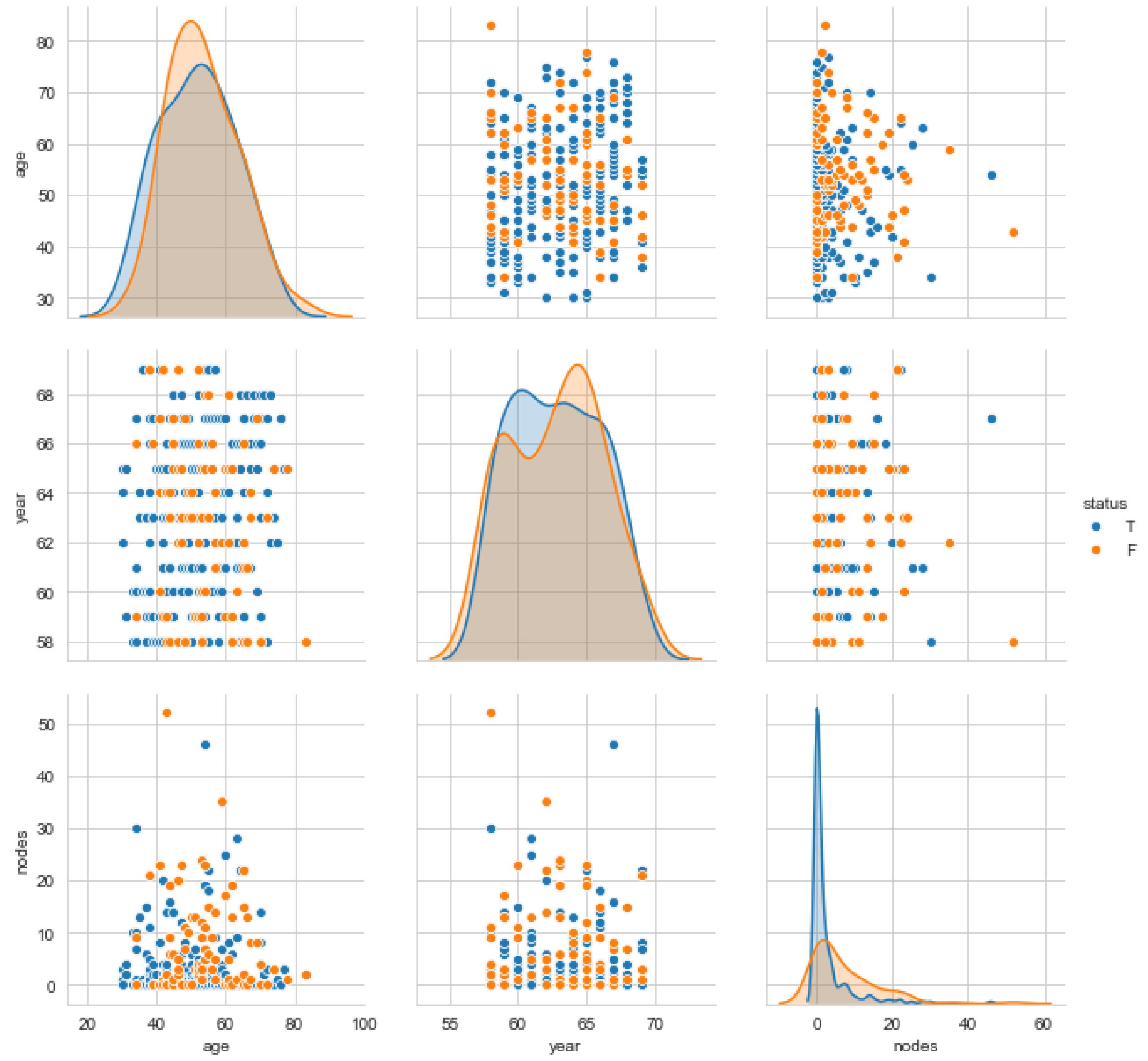
In [13]:
```
sns.set_style("whitegrid");
sns.FacetGrid(cancerDetect, hue='status', size=4) \
    .map(plt.scatter, "age", "nodes")\
    .add_legend()
plt.show();
```



we use seaborn to get a white grid at the background of scatter plot to see the spread of data more clearly. We also increase the size and add legend which give us more information of the data and help us classify the data so as to identify the status of the patients. *Observations : after taking age and nodes on scatter plot we see that the data is not well separated.

Pair Plots

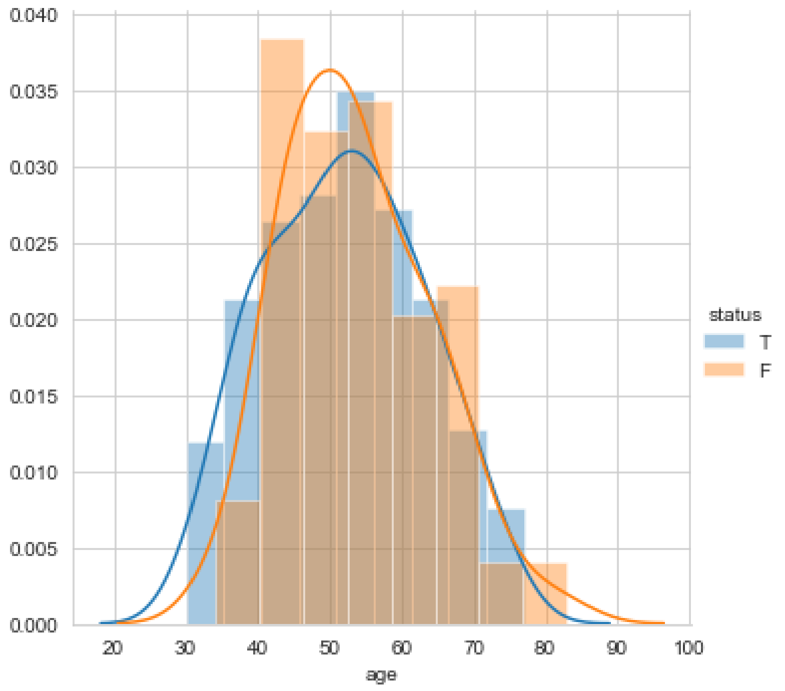
In [14]:
```
plt.close(); # so that we close all the figures and take a fresh start
sns.set_style("whitegrid")
sns.pairplot(cancerDetect, hue="status", size=3);
plt.show()
```



we have 3 variables and therefore we try to create a pair of two pairs, so we get 3C2 combinations, so we can have 3 such plots Age and Nodes, Year and Age, Year and Nodes *Observations: we get all mixed data points between all combinations of pairs plots between Age, Year and Nodes

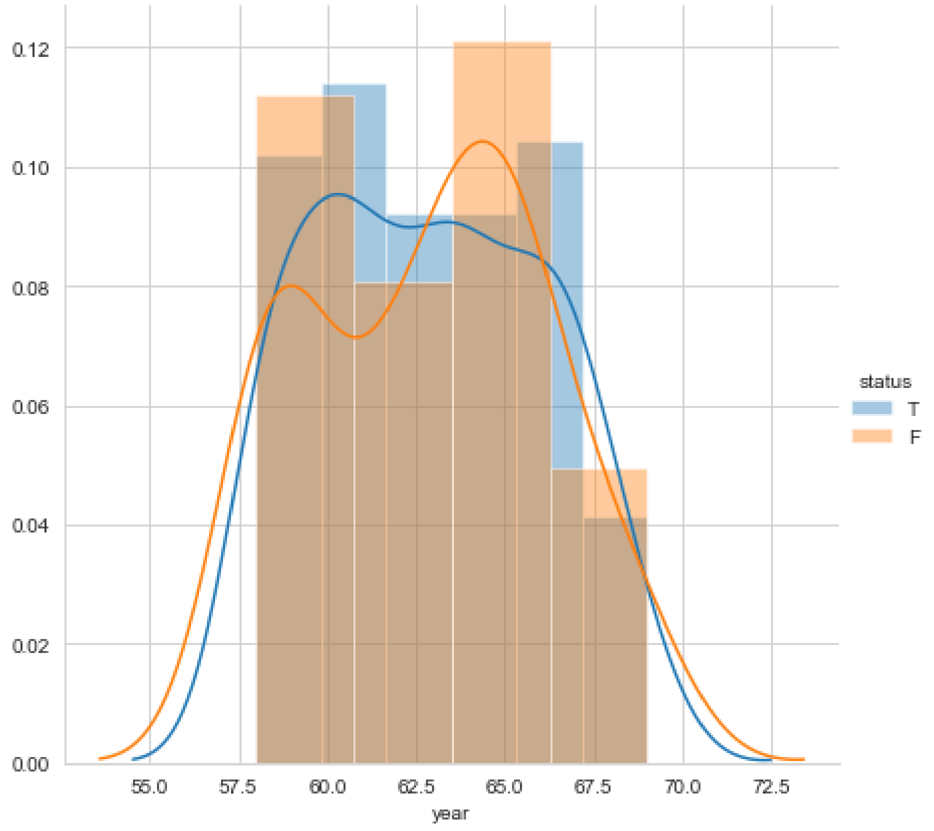
Univariate Data Analysis and understanding histogram and PDF

In [15]:
```
sns.FacetGrid(cancerDetect, hue="status", size=5)\
    .map(sns.distplot, "age")\
    .add_legend()
plt.show()
```



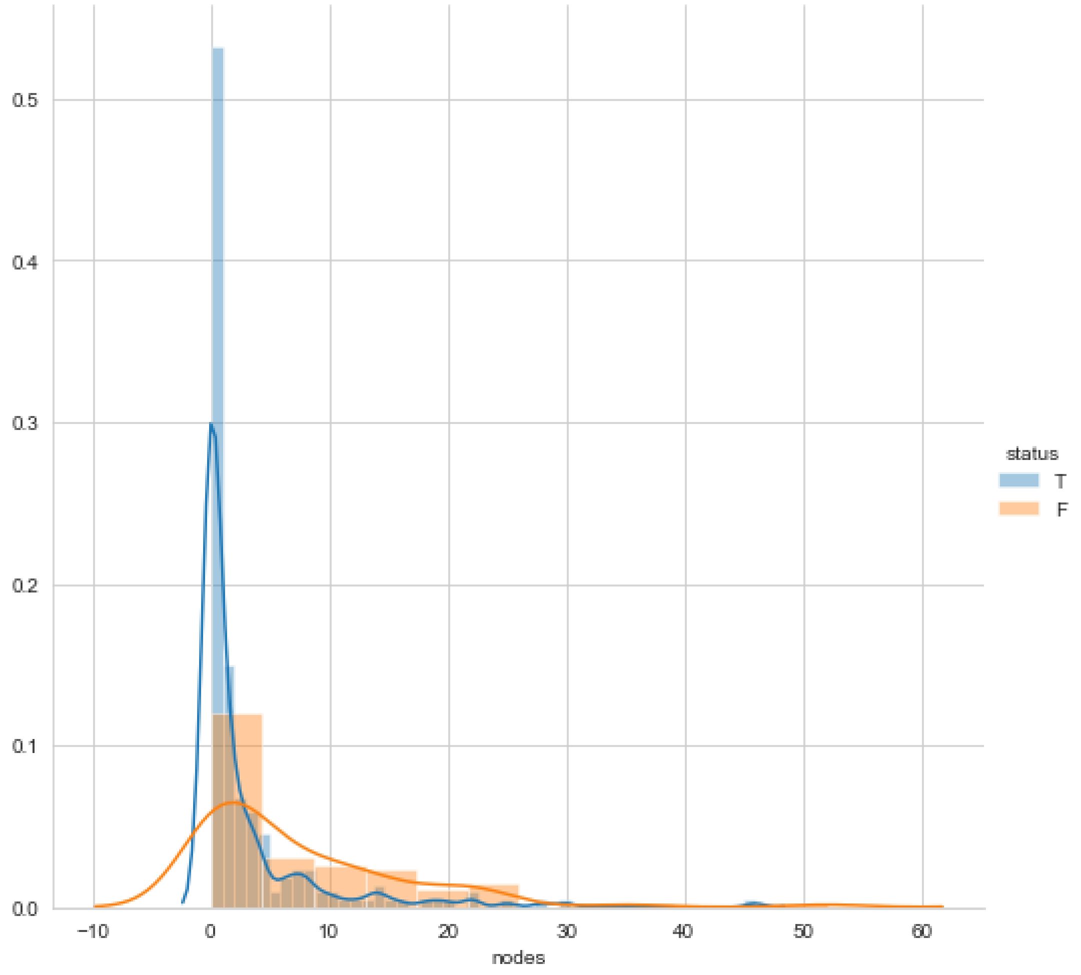
Observations: After performing univariate data analysis on y axis as the counts and x axis is Age, we get a mixed combination of pepole who have Survival-status as True(lived) and False(Died) and we have height as 0.035 of status =T which is in blue color also we have height greater than 0.035 for people with status as F which is orange in color.
a)Here we define a rule: people whose Survival Status is T, have age range between 30 and 75 and people with survival status F, have age range between 35 to 82 here our PDF and histograms are Overlapping Significant Observation we can make is: if Age is between 30 to 33 then we can say that the patient survived after operation and if the age range is between 76 to 82 then we can say that the patient did not survived after operation.
b)Best case could have been that the data was well separated but here we get Overlapping in all Univariate formations of variables such as Age, Year and Nodes. However for choosing the better univariate plots then we could say that Age>>Nodes>>Year so we can do good analysis with Age and Nodes </font>

In [16]:
```
sns.FacetGrid(cancerDetect, hue="status", size=6)\
    .map(sns.distplot, "year")\
    .add_legend()
plt.show()
```



In [17]:
```
sns.FacetGrid(cancerDetect, hue="status", size=7)\
    .map(sns.distplot,"nodes")\
    .add_legend()
plt.plot()
```

Out[17]:
```
[]
```



In [18]:

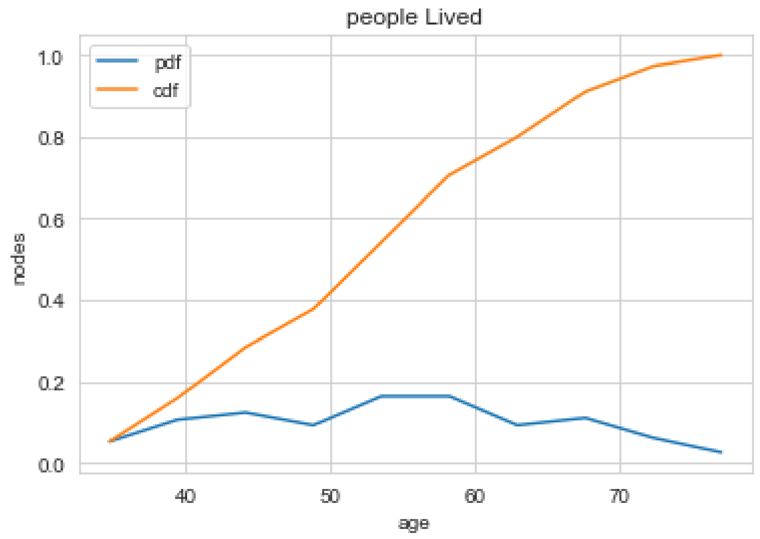
```
# we are creating 2 variables namely peopleLived (which contains group of peop
le Survived after operation)
# and peopleDied (which contains group of people which did not Survived after
 operation)
peopleLived = cancerDetect.loc[cancerDetect["status"] == 'T'];
peopleDied = cancerDetect.loc[cancerDetect["status"] == 'F'];
```

Cumulative Distribution Function

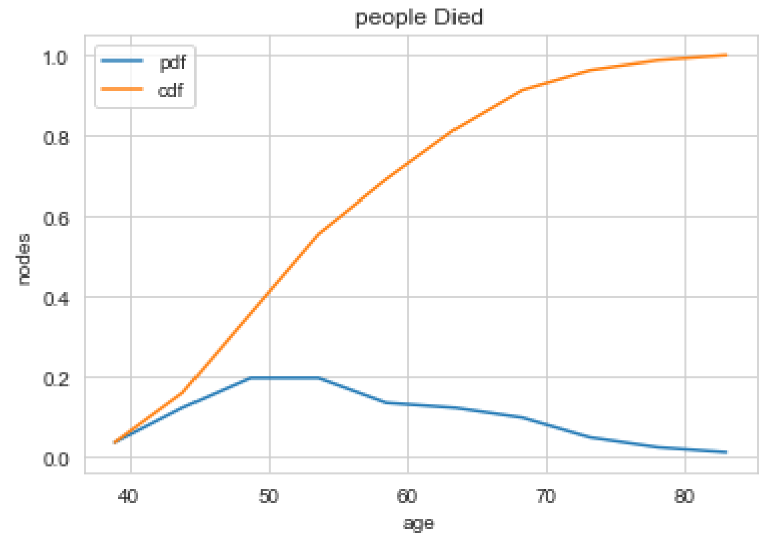
In [19]:

```
plt.figure(1)
counts_pl, bin_edges_pl = np.histogram(peopleLived['age'], bins=10, density = 
True)
pdf_pl = counts_pl/(sum(counts_pl))
print(pdf_pl);
print(bin_edges_pl);
cdf_pl = np.cumsum(pdf_pl)
plt.plot(bin_edges_pl[1:],pdf_pl)
plt.plot(bin_edges_pl[1:], cdf_pl)

#plt.plot(pdf_pl, 'r-',cdf_pl, 'b-', )
plt.xlabel("age")
plt.ylabel("nodes")
plt.title("people Lived")
plt.gca().legend(('pdf','cdf'))
#plt.legend()
plt.show()

plt.figure(2)
counts_pd, bin_edges_pd = np.histogram(peopleDied['age'], bins=10, density = T
rue)
pdf_pd = counts_pd/(sum(counts_pd))
print(pdf_pd)
print(bin_edges_pd)
cdf_pd = np.cumsum(pdf_pd)

#plt.plot(pdf_pl, 'r-',cdf_pl, 'b-', )
plt.plot(bin_edges_pd[1:],pdf_pd)
plt.plot(bin_edges_pd[1:], cdf_pd)
plt.xlabel("age")
plt.ylabel("nodes")
plt.title("people Died")
plt.gca().legend(('pdf','cdf'))
#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. ]
```



```
[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. ]
```

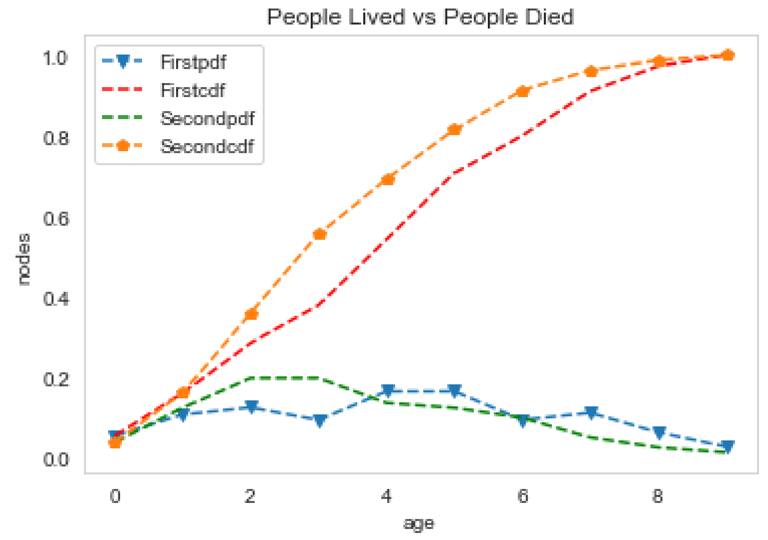


We are plotting CDF cummulative distributed frequency, first one is plotted with respect to Age and second plot is for Nodes. we are plotting them seperately. Observations : 100% people die if their age is beyond 80. And 97% and above people are died at the age between 72 and above at the age range between 53 to 57, we can say that nearly there is 50% to 70% chances of Survival if age range between 40 to 50 has less than 42% chances of Survival

In [20]:
```
#Observations : 100% people die if their age is beyond 80. And 97% and above p
eople are died at the age between 72 and above at the age range between 53 to
 57,
#we can say that nearly there is 50% to 70% chances of Survival if age range b
etween 40 to 50 has less than 42% chances of Survival

plt.figure(1)
counts_one, bin_edges_one = np.histogram(peopleLived['age'], bins=10, density
= True)
pdf_one = counts_one/(sum(counts_one))
print(pdf_one);
print(bin_edges_one);
```

```python
cdf_one = np.cumsum(pdf_one)
#plt.plot(bin_edges[1:],pdf)
#plt.plot(bin_edges[1:], cdf)
plt.plot(pdf_one, 'v--', label='Firstpdf')
plt.plot(cdf_one, 'r--', label='Firstcdf')

#plt.figure(2)
counts_two, bin_edges_two = np.histogram(peopleDied['age'], bins=10, density =
 True)
pdf_two = counts_two/(sum(counts_two))
print(pdf_two)
print(bin_edges_two)
cdf_two = np.cumsum(pdf_two)
#plt.plot(bin_edges[1:],pdf);
#plt.plot(bin_edges[1:],cdf)
#plt.show();

plt.plot(pdf_two, 'g--', label='Secondpdf')
plt.plot(cdf_two, 'p--', label='Secondcdf')
plt.grid()
plt.legend() # add legend based on line labels
plt.title('People Lived vs People Died')
plt.xlabel("age")
plt.ylabel("nodes")
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. ]
[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. ]
```



Mean and Standard Deviation

In [21]:
```
#Mean and Std-deviation
print(np.mean(peopleLived["age"])) #mean is 52.01, for people who lived after
 operation
print(np.mean(peopleDied["age"]))  #mean is 53.67, for people who died after o
peration

print("\n")
print("StdDeviation ")
print(np.std(peopleLived["age"]))  # 10.93 standard deviation, for people who
 lived after operation
print(np.std(peopleDied["age"]))   # 10.10 standard deviation, for people who
 died after operation
```

```
52.017777777777778
53.67901234567901
```

```
StdDeviation
10.98765547510051
10.10418219303131
```

Median, Quantiles and Percentiles

In [22]:

```
#Median, Quantiles, Percentiles, IQR.
print("Median")
print(np.median(peopleLived["age"]))  # median lies at 52 for people lived wit
h respect to their ages
print(np.median(peopleDied["age"]))   # median lies at 53 for people who died
 with respect to their ages

print("\n")
print("Quantile")
print(np.percentile(peopleLived["age"],np.arange(0, 100, 25)))  # Quantile val
ues for people who lived, 0% of values are less than 30
# 25% of values are less than 43, 50% of values are less than 52, 75% values a
re less than 60
print(np.percentile(peopleDied["age"],np.arange(0, 100, 25)))  # Quantile valu
es for people who died

print("\n")
print("90th Percentiles")
print(np.percentile(peopleLived["age"],90))  # 90th percentile of people who l
ived
print(np.percentile(peopleDied["age"],90))  # 90th percentile of people who di
ed

from statsmodels import robust
print("\n")
print ("Median Absolute Deviation")
```

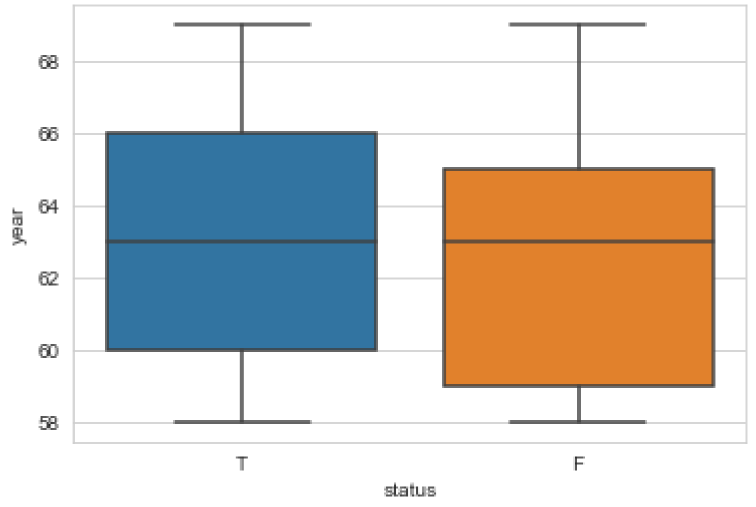
```
print(robust.mad(peopleLived["age"]))  # MAD gives us the absolute deviation v
alue from a particular point to median,for people who lived is 13.34
# and then we compute
print(robust.mad(peopleDied["age"])) #MAD for people who died is 13.34
```

```
Median
52.0
53.0



Quantile
[30. 43. 52. 60.]
[34. 46. 53. 61.]



90th Percentiles
67.0
67.0



Median Absolute Deviation
13.343419966550417
11.860817748044816
```

BOX-PLOT

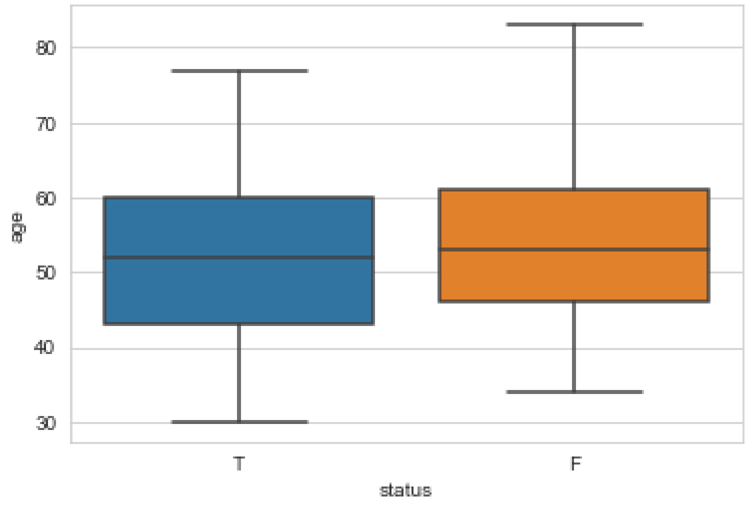
In [23]:

```
plt.figure(1)
sns.boxplot(x='status', y='year', data=cancerDetect)
plt.show()
```



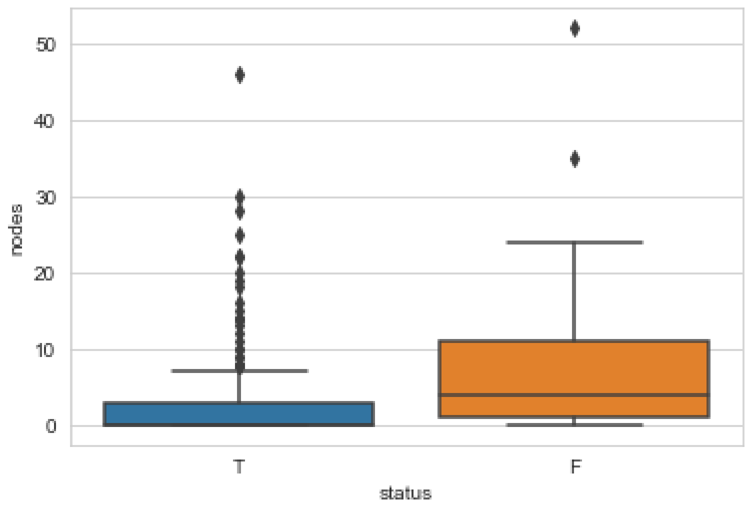
YEAR: from the figure we could say that 25th percentile value for "1" is 60 and for "2" is 59, 50th percentile value for both "1" and "2" is 63 and 75th percentile value for "T" is 66, and for "F" is 63 for "T" :IQR value is 6 wisker for "T" : 1.5 * IQR = 1.5 * 6 = 9 for "F" : IQR value is 4 wisker for "F" : 1.5 * IQR = 1.5 * 4 = 6

In [24]:
```
plt.figure(2)
sns.boxplot(x='status', y='age', data=cancerDetect)
plt.show()
```



AGE: from the figure we could say that 25th percentile value for "True" is 43 and for "False" is 47, 50th percentile value for "True" is 52 and for "False" is 54 and 75th percentile value for "True" is 60, and for "False" is 62 for "True": IQR value is 17 wisker for "True" : 1.5 * 17 = 25.5 for "False" : IQR value is 15 wisker for "False" : 15 * 1.5 = 22.5

In [25]:
```
plt.figure(3)
sns.boxplot(x='status', y='nodes', data=cancerDetect)
plt.show()
```

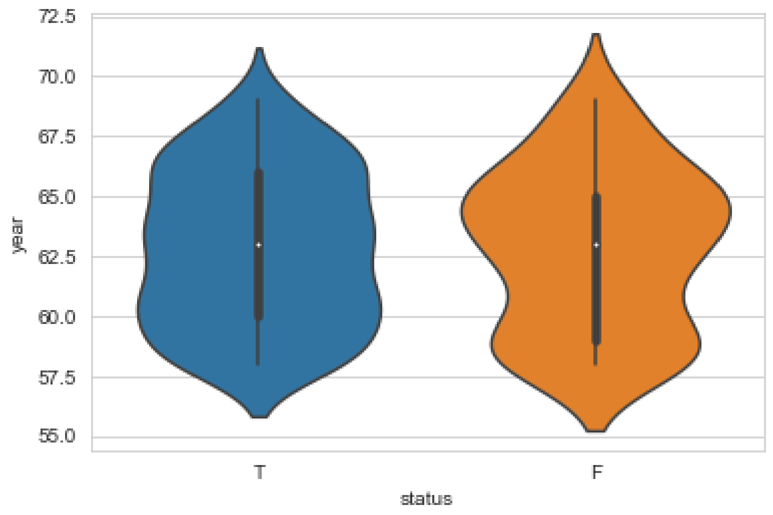


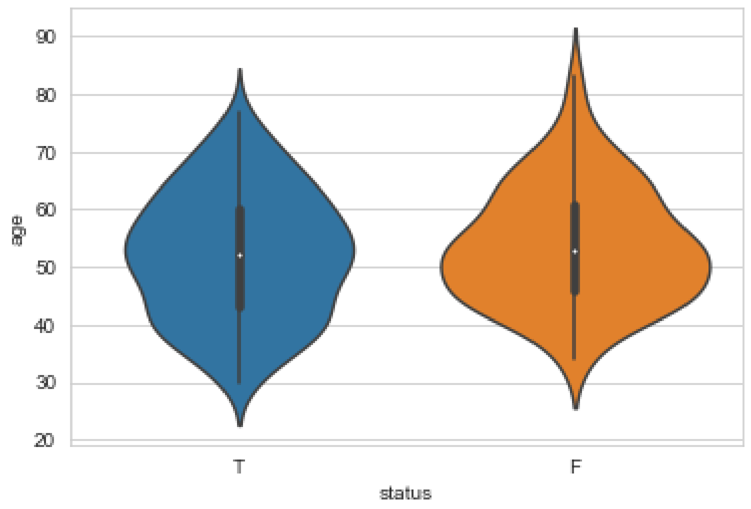
Nodes: from the figure we could say that 25th percentile value for "True" is 0 and for "False" is 1, 50th percentile value for "True" is 0 and "False" is 4 and 75th percentile value for "True" is 3, and for "False" is 12 for "True": IQR value is 2 wisker for "True" : 1.5 * 2 = 3 for "False" : IQR value is 11 wisker for "False" : 11 * 1.5 = 16.5

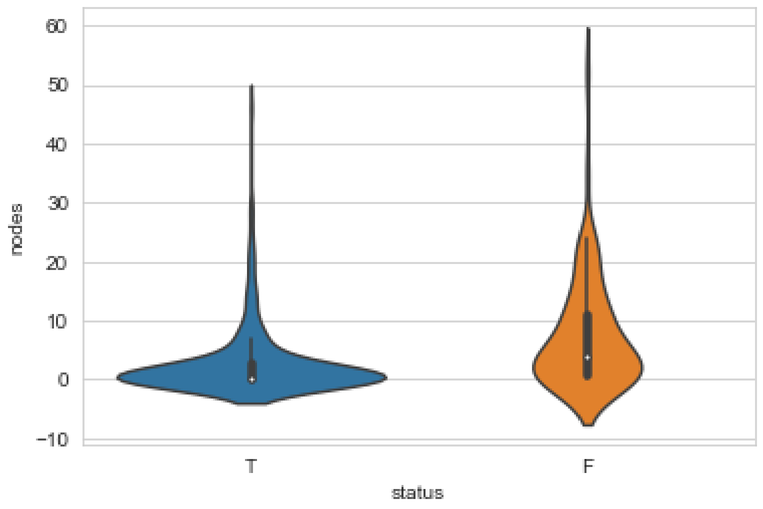
Violin Plots

In [26]:
```
plt.figure(1)
sns.violinplot(x='status', y='year', data=cancerDetect, size=8)

#YEAR: from the figure we could say that 25th percentile value for "True" is 5
9 and for "False" is 58, 50th percentile value for both "True" and "False" is
 63 and
#75th percentile value for "True" is 66, and for "False" is 63

plt.figure(2)
sns.violinplot(x='status', y='age', data=cancerDetect, size=7)
```

```
#AGE: from the figure we could say that 25th percentile value for "True" is 43
 and for "False" is 47, 50th percentile value for  "True" is 52 and for "Fals
e" is 54 and 75th percentile
# value for "True" is 60, and for "False" is 62

plt.figure(3)
sns.violinplot(x='status', y='nodes', data = cancerDetect, size=8)

#Nodes: from the figure we could say that 25th percentile value for "True" is
 0 and for "False" is 1, 50th percentile value for "True" is 0 and "False" is
 4 and 75th percentile
# value for "True" is 3, and for "False" is 12

plt.show()
```







Conclusion:

a) Significant Observation we can make is: if Age is between 30 to 33 then we can say that the patient survived after operation and if the age range is between 76 to 82 then

we can say that the patient did not survived after operation.
b) Best case could have been that the data was well separated but here we get Overlapping in all especially in case of Pair plots. Univariate formations of variables such as Age, Year and Nodes but for choosing the better univariate plots then we could say that Age>>Nodes>>Year so we can do good analysis with Age and Nodes.
c) We didn't get any huge leads after using pair plots. After performing Univariate analysis, we selected Age and Nodes for further analysis. After performing ploting the plots for cdf's and pdf's We observe that 100% of the cancerous people die if their age is beyond 80. 97% of the people are died at the age between 72 and above and at the age range between 53 to 57, we can say that nearly there is 50% to 70% chances of Survival. Age range between 40 to 50 has less than 42% chances of Survival.
d) Mean of 52.01,for people who lived after operation.Mean of 53.67,for people who died after operation.10.93 standard deviation for people who survived. 10.10 standard deviation for people who died. Median lies at 52 for people lived with respect to their ages. Median lies at 53 for people who died with respect to their ages

```
In [ ]:
```

```
In [ ]:
```