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 alanlaysis 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.
5. Perform Univaraite analysis(PDF, CDF, Boxplot, Voilin plots) to understand which features are useful towards classification.
6. Perform Bi-variate analysis (scatter plots, pair-plots) to see if combinations of features are useful in classfication.
7. Write your observations in english as crisply and unambigously as possible. Always quantify your results.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Title: Haberman's Survival Data

Sources: https://www.kaggle.com/gilsousa/habermans-survival-data-set/version/1

Past Usage:

Haberman, S. J. (1976). Generalized Residuals for Log-Linear Models, Proceedings of the 9th International Biometrics Conference, Boston, pp. 104-122. Landwehr, J. M., Pregibon, D., and Shoemaker, A. C. (1984), Graphical Models for Assessing Logistic Regression Models (with discussion), Journal of the American Statistical Association 79: 61-83. Lo, W.-D. (1993). Logistic Regression Trees, PhD thesis, Department of Statistics, University of Wisconsin, Madison, WI. Relevant Information: 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.

Number of Instances: 306

Number of Attributes: 4 (including the class attribute)

Attribute Information:

Age of patient at time of operation (numerical) Patient's year of operation (year - 1900, numerical) Number of positive axillary nodes detected (numerical) Survival status (class attribute) 1 = the patient survived 5 years or longer 2 = the patient died within 5 year

In [2]:
```
heberman_dt=pd.read_csv("haberman.csv", header=None)
print("Check the first few records loaded from the csv file:")
print(heberman_dt.head(5))
```

```
Check the first few records loaded from the csv file:
    0   1  2  3
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 [3]:
```
#adding column names to the data
heberman_dt.columns=['age','yr_of_opr','postv_ax','status']
print(heberman_dt.head(5))
```

```
   age  yr_of_opr  postv_ax  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 [22]:
```
print (heberman_dt.shape)
heberman_dt["status"].value_counts()
```

```
(306, 4)
```

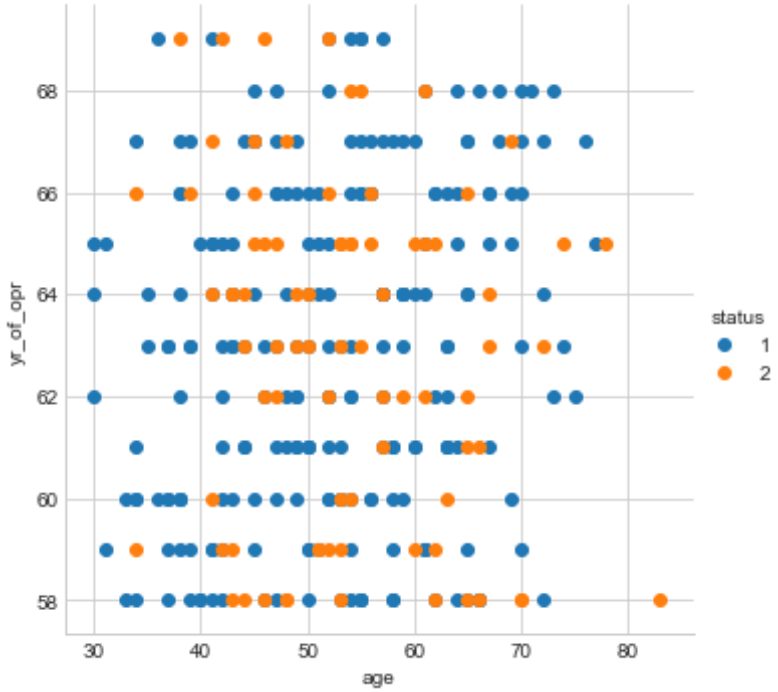
Out[22]:
```
1    225
2     81
Name: status, dtype: int64
```

observation 1:

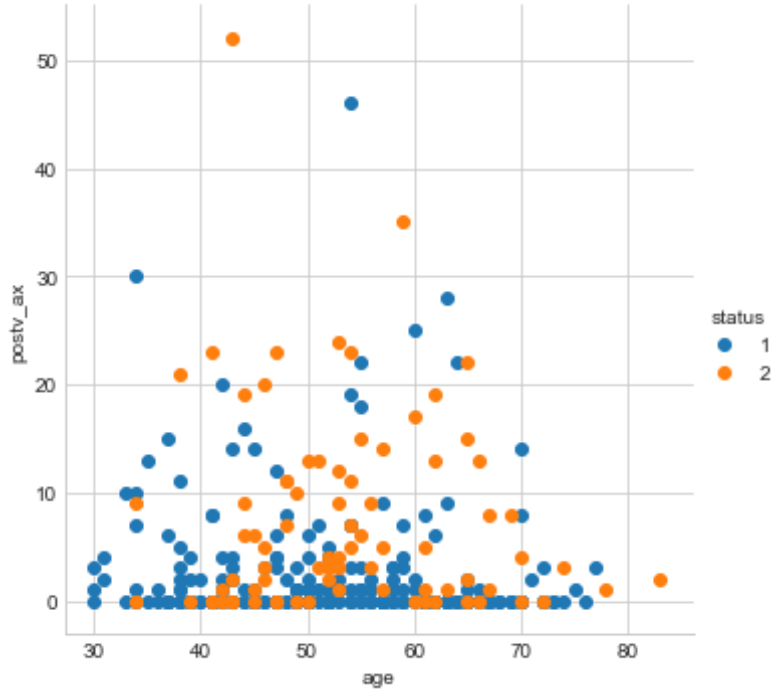
The dataset has total 306 entries and total 3 features and 1 classification label We can also see that it has total 2 labels, 1st label '1' has 225 entries and label '2' has 81 entries in the dataset We can conclude that the labels are not distributed equally in th dataset

Our objective is to find which feature(s) is most useful among these 3 features to help us to make a classifier model that can predict the servival status of a person

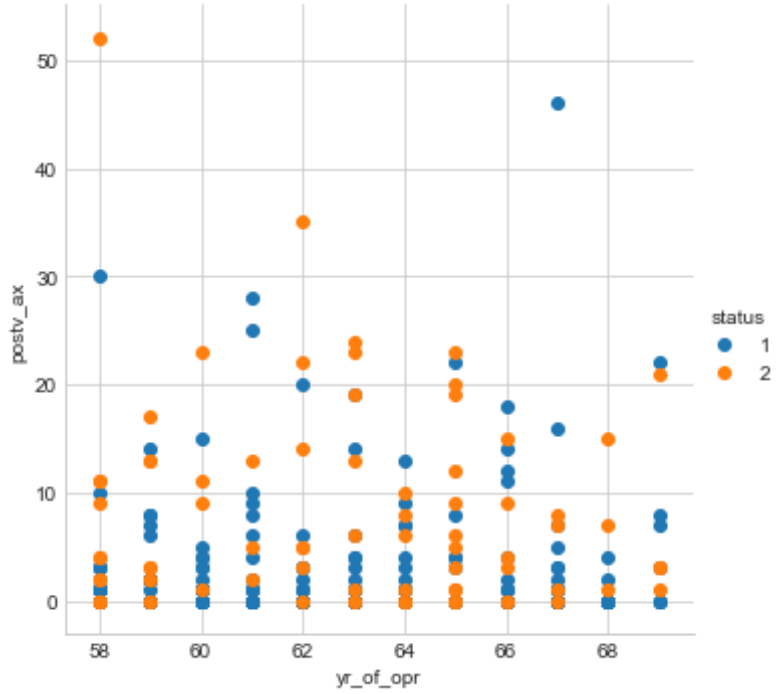
In [5]:
```
#2D scatter plot for all the 3 features, combination of each pair
sns.set_style("whitegrid");
sns.FacetGrid(heberman_dt, hue="status", size=5) \
   .map(plt.scatter, "age", "yr_of_opr") \
   .add_legend();
plt.show();
```



In [6]:
```
sns.FacetGrid(heberman_dt, hue="status", size=5) \
   .map(plt.scatter, "age", "postv_ax") \
   .add_legend();
plt.show();
```



In [7]:
```
sns.FacetGrid(heberman_dt, hue="status", size=5) \
   .map(plt.scatter, "yr_of_opr", "postv_ax") \
   .add_legend();
plt.show();
```



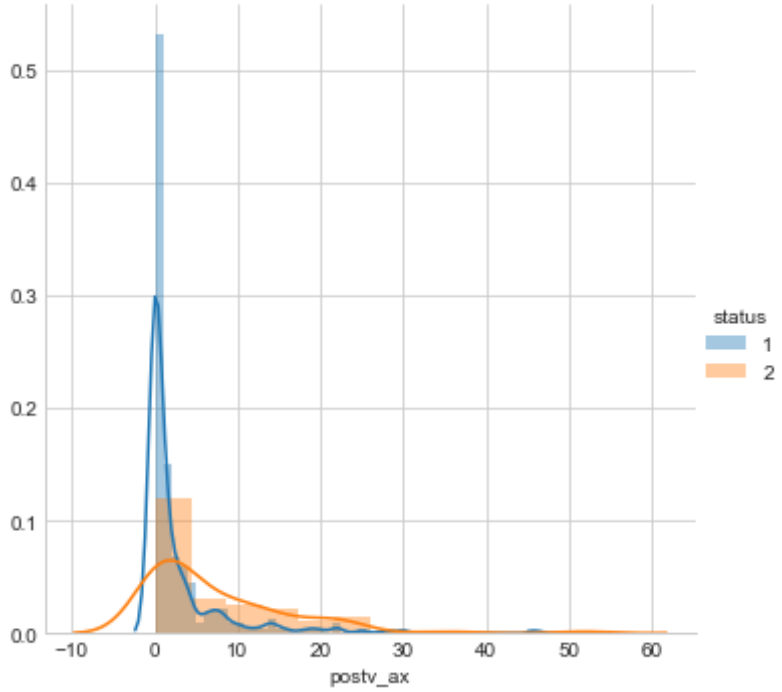
Observation 2:

after plotting all the 3 plots of 2-D scatter plot with each possible pair, we are not able to identify any feature which can be more useful for classification but feature postv_ax is slightly better than the other 2 features

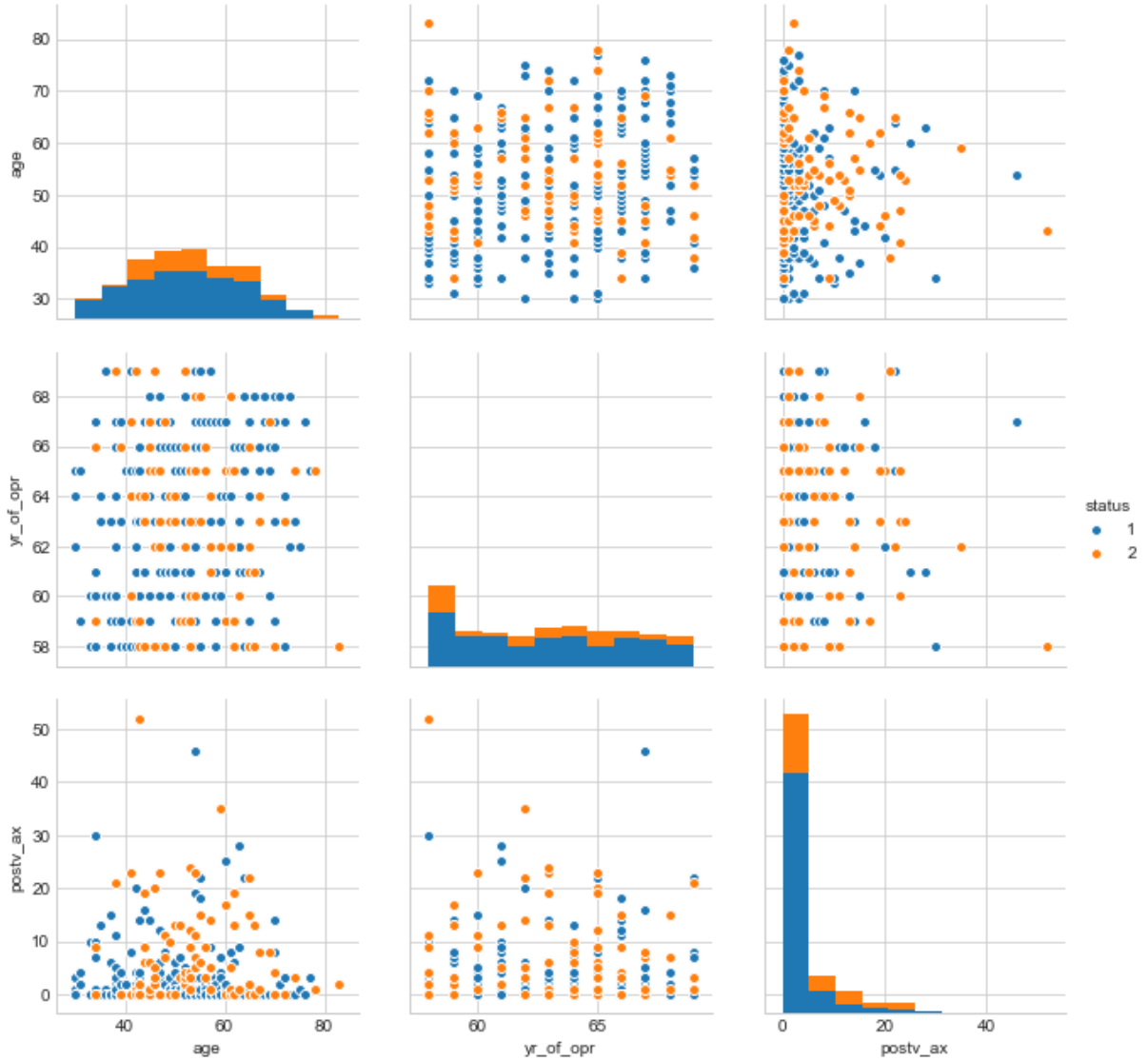
In [43]:
```
sns.FacetGrid(heberman_dt, hue="status",size=5) \
   .map(sns.distplot, "postv_ax") \
   .add_legend();
plt.show();
```

```
C:\Users\User\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:646
2: UserWarning: The 'normed' kwarg is deprecated, and has been replaced
by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

```
C:\Users\User\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



In [38]:
```
#pair plot
sns.set_style("whitegrid");
#sns.pairplot(heberman_dt, hue="status", size=2);
sns.pairplot(heberman_dt, vars=["age","yr_of_opr", "postv_ax"], hue="status", size=3)
plt.show()
```



observation 3:

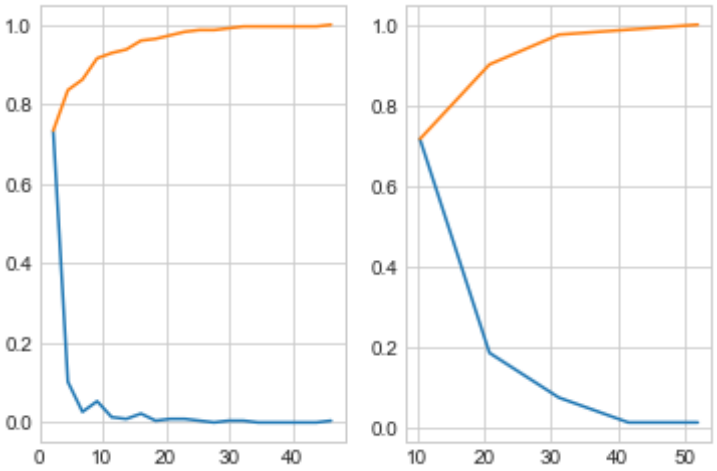
We can see that no pair of features can help to build a classifier model for distinguising between the survival status 1,2 properly

In [45]:
```
heberman_status_1 = heberman_dt.loc[heberman_dt["status"] == 1];
heberman_status_2 = heberman_dt.loc[heberman_dt["status"] == 2];

plt.subplot(1,2,1)
#status 1
counts, bin_edges = np.histogram(heberman_status_1['postv_ax'], bins=20
, density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)
plt.title="status 1"

plt.subplot(1,2,2)
# status 2
counts, bin_edges = np.histogram(heberman_status_2['postv_ax'], bins=20
, density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)
plt.title="status 2"
plt.show();
```

```
[0.73333333 0.10222222 0.02666667 0.05333333 0.01333333 0.00888889
 0.02222222 0.00444444 0.00888889 0.00888889 0.00444444 0.
 0.00444444 0.00444444 0.         0.         0.         0.
 0.         0.00444444]
[ 0.   2.3  4.6  6.9  9.2 11.5 13.8 16.1 18.4 20.7 23.  25.3 27.6 29.9
 32.2 34.5 36.8 39.1 41.4 43.7 46. ]
[0.71604938 0.18518519 0.07407407 0.01234568 0.01234568]
[ 0.  10.4 20.8 31.2 41.6 52. ]
```



observation 4:

we can see that more than 85% of the patient with status 1 has number of positive axillary less than 10, and almost 100% records has less than 20 positive axillary. For status 2, almost 70% has positive axillary less than 10,75 to 100% has positive axillary between 10 and 40

In [35]:
```
heberman_status_1.describe()
```

Out[35]:

	age	yr_of_opr	postv_ax	status
count	225.000000	225.000000	225.000000	225.0
mean	52.017778	62.862222	2.791111	1.0
std	11.012154	3.222915	5.870318	0.0
min	30.000000	58.000000	0.000000	1.0
25%	43.000000	60.000000	0.000000	1.0
50%	52.000000	63.000000	0.000000	1.0

	age	yr_of_opr	postv_ax	status
75%	60.000000	66.000000	3.000000	1.0
max	77.000000	69.000000	46.000000	1.0

In [36]:
```
heberman_status_2.describe()
```

Out[36]:

	age	yr_of_opr	postv_ax	status
count	81.000000	81.000000	81.000000	81.0
mean	53.679012	62.827160	7.456790	2.0
std	10.167137	3.342118	9.185654	0.0
min	34.000000	58.000000	0.000000	2.0
25%	46.000000	59.000000	1.000000	2.0
50%	53.000000	63.000000	4.000000	2.0
75%	61.000000	65.000000	11.000000	2.0
max	83.000000	69.000000	52.000000	2.0

observation 5:

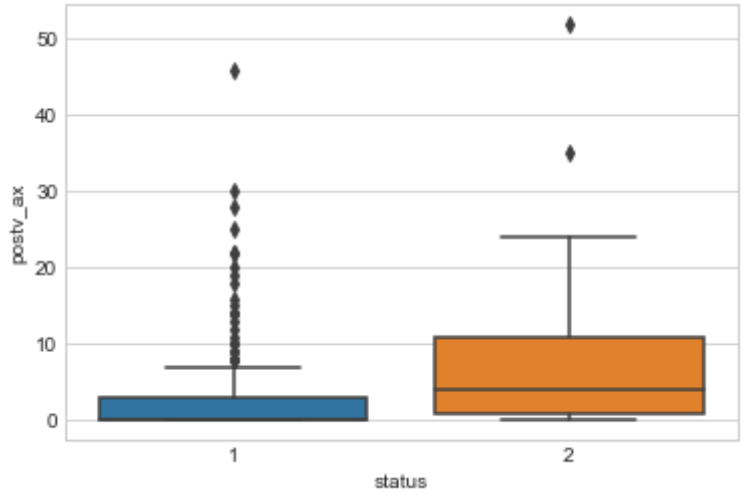
For status 1, more than 50% has positive axillary 0,75% has less than 3, suddenly max positive axillary is 46 , so there can be a poibility that the data has some outliers in status 1, we can check the median

In [49]:
```
print("status 1 median=",np.median(heberman_status_1))
print("status 2 median=",np.median(heberman_status_2))
```

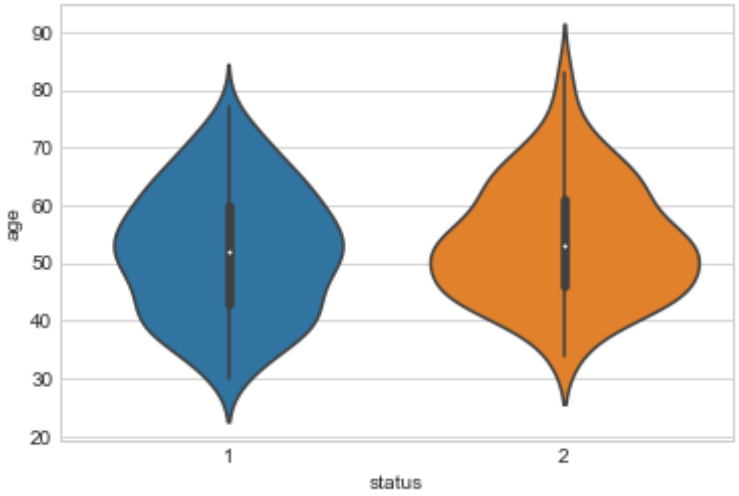
```
status 1 median= 30.0
status 2 median= 34.5
```

In [50]:
```
#box plot

sns.boxplot(x='status',y='postv_ax', data=heberman_dt)
plt.show()
```



In [22]:
```
#violin plot
sns.violinplot(x="status", y="age", data=heberman_dt, size=10)
plt.show()
```



Conclusion:

It is very difficult to classify the 2 status based on the features provided If we could add more features to the dataset then we might have a better model

Even if we build a model with the features , overlapping will be very high and accuracy will be less