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. 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

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

In [2]:
```
print(df.shape)
```

```
(306, 4)
```

In [3]:
```
print(df.columns)
```

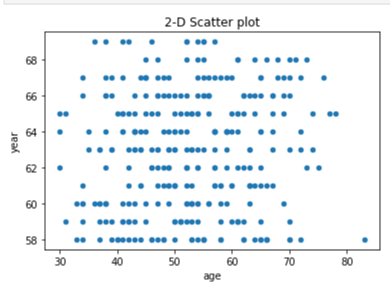
```
Index(['age', 'year', 'nodes', 'status'], dtype='object')
```

In [4]:
```
df["status"].value_counts()
```

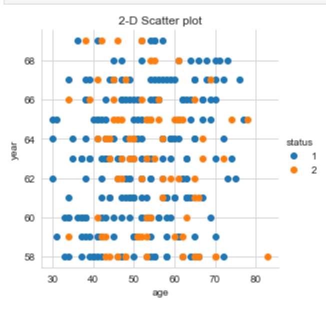
Out[4]:
```
1    225
2     81
Name: status, dtype: int64
```

2 D scatter

In [5]:
```
df.plot(kind='scatter', x='age', y='year') ;
plt.title("2-D Scatter plot")
plt.show()
```

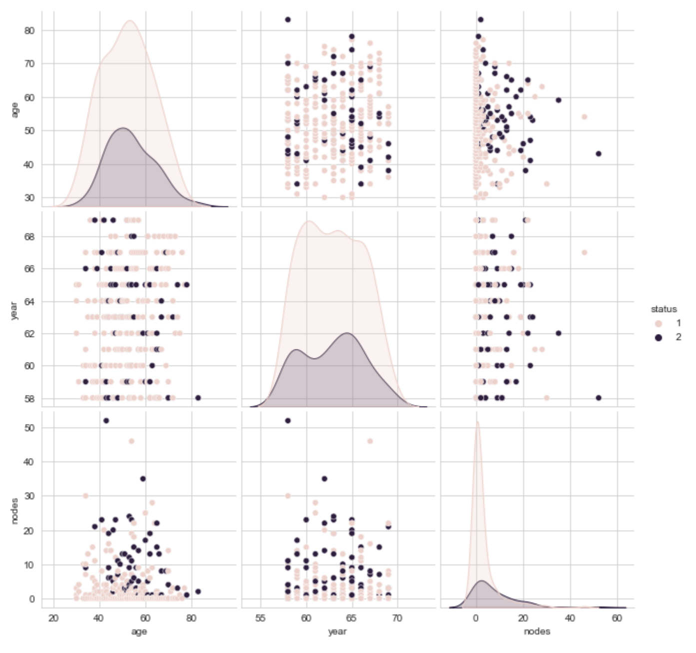


In [19]:
```
#2-d scatter with different color
sns.set_style("whitegrid");
# hue="species"-> color based on species
sns.FacetGrid(df, hue="status", height=4) \
   .map(plt.scatter, "age", "year") \
   .add_legend();
plt.title("2-D Scatter plot")
plt.show();
```



OBSERVATION: Using age and year we are not able to distinguies

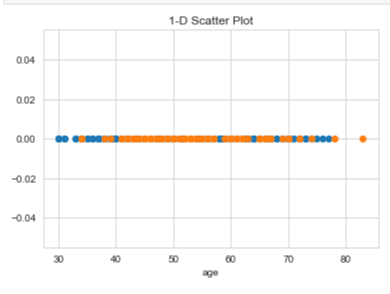
In [21]:
```
#pair plot
plt.close();
sns.set_style("whitegrid");
sns.pairplot(df, hue="status", height=3);
plt.show()
```



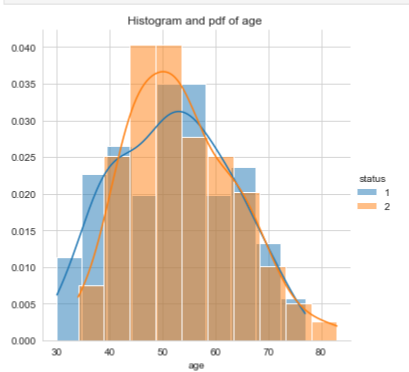
Observation:- points can not be diffretiable

Histogram, PDF, CDF

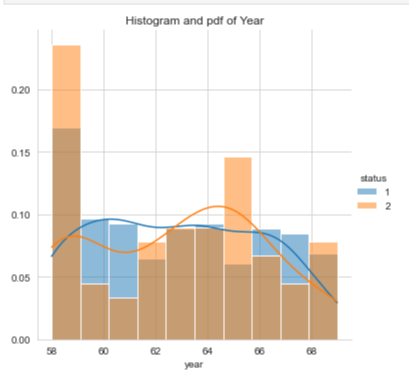
In [22]:
```
#1-d scatter plot of age
import numpy as np
haberman1= df.loc[df["status"] == 1];
haberman2 = df.loc[df["status"] == 2];
haberman3 = df.loc[df["status"] == 3];
#print(iris_setosa["petal_length"])
plt.plot(haberman1["age"], np.zeros_like(haberman1['age']), 'o')
plt.plot(haberman2["age"], np.zeros_like(haberman2['age']), 'o')
plt.plot(haberman3["age"], np.zeros_like(haberman3['age']), 'o')
plt.xlabel("age")
plt.title("1-D Scatter Plot")
plt.show()
```



In [25]:
```
#histogram and pdf of age
sns.FacetGrid(df, hue="status", height=5) \
   .map(sns.histplot, "age",stat="density",kde=True,bins=10) \
   .add_legend();
plt.title("Histogram and pdf of age ")
plt.show();
```

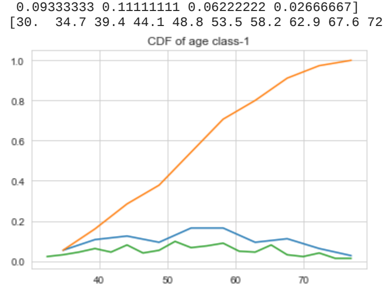


In [27]:
```
#histogram and pdf of year
sns.FacetGrid(df, hue="status", height=5) \
   .map(sns.histplot, "year",stat="density",kde=True,bins=10) \
   .add_legend();
plt.title("Histogram and pdf of Year")
plt.show();
```



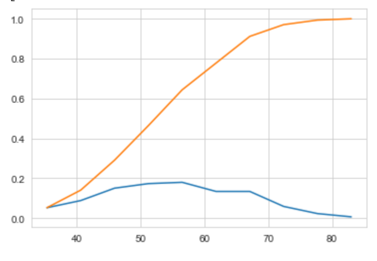
In [28]:
```
#cdf of age class-1
counts, bin_edges = np.histogram(haberman1['age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges);
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf);
plt.plot(bin_edges[1:], cdf)


counts, bin_edges = np.histogram(haberman1['age'], bins=20,
                                 density = True)
pdf = counts/(sum(counts))
plt.plot(bin_edges[1:],pdf);
plt.title("CDF of age class-1")
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. ]
```



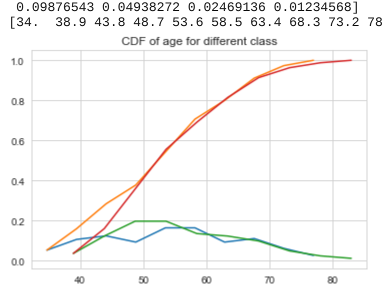
In [12]:
```
# How to construct a CDF?
# How to read a CDF?

#Plot CDF of age

counts, bin_edges = np.histogram(df['age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)

#compute CDF
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)



plt.show();
```

```
[0.05228758 0.08823529 0.1503268  0.17320261 0.17973856 0.13398693
 0.13398693 0.05882353 0.02287582 0.00653595]
[30.  35.3 40.6 45.9 51.2 56.5 61.8 67.1 72.4 77.7 83. ]
```



In [29]:
```
# Plots of CDF of age for various class.


counts, bin_edges = np.histogram(haberman1['age'], bins=10,
                                 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)


counts, bin_edges = np.histogram(haberman2['age'], bins=10,
                                 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("CDF of age for different class")
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, Variance and Std-dev

In [14]:
```
#Mean, Variance, Std-deviation,
print("Means:")
print('mean of class 1 is',np.mean(df["age"]))
#Mean with an outlier.
print('mean with outlier is',np.mean(np.append(df["age"],225)));
print('mean of class 2 is:',np.mean(df["age"]))

print("\nStd-dev:");
print('STD of class 1 is:',np.std(haberman1["age"]))
print('STD of class 2 is:',np.std(haberman2["age"]))
```

```
Means:
mean of class 1 is 52.45751633986928
mean with outlier is 53.01954397394137
mean of class 2 is: 52.45751633986928

Std-dev:
STD of class 1 is: 10.98765547510051
STD of class 2 is: 10.10418219303131
```

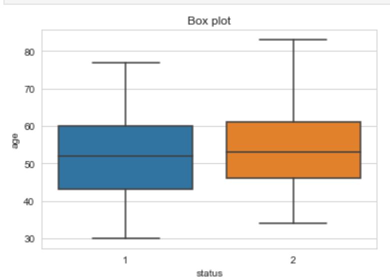
Median, Percentile, Quantile, IQR, MAD

In [15]:
```
#Median, Quantiles, Percentiles, IQR.
print("\nMedians:")
print(np.median(haberman1["age"]))
#Median with an outlier
print(np.median(np.append(haberman1["age"],225)));
print(np.median(haberman2["age"]))


print("\nQuantiles:")
print(np.percentile(haberman1["age"],np.arange(0, 100, 25)))
print(np.percentile(haberman2["age"],np.arange(0, 100, 25)))

print("\n90th Percentiles:")
print(np.percentile(haberman1["age"],90))
print(np.percentile(haberman1["age"],90))

from statsmodels import robust
print ("\nMedian Absolute Deviation(mda)")
print(robust.mad(haberman1["age"]))
print(robust.mad(haberman2["age"]))
```

```
Medians:
52.0
52.0
53.0

Quantiles:
[30. 43. 52. 60.]
[34. 46. 53. 61.]

90th Percentiles:
67.0
67.0

Median Absolute Deviation(mda)
13.343419966550417
11.860817748044816
```

Box plot and Whiskers

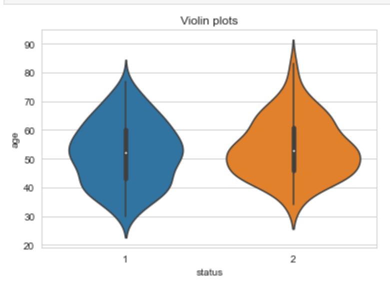
In [30]:
```
sns.boxplot(x='status',y='age', data=df)
plt.title("Box plot")
plt.show()
```



Observation:-for class-1 25% to 75% wrt age lies between 42 to 60 and for class-2 25% to 75% lies between 45 to 62

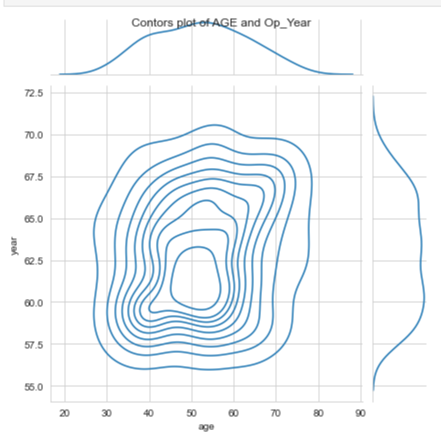
Violin plots

In [31]:
```
sns.violinplot(x="status", y="age", data=df, size=8)
plt.title("Violin plots")
plt.show()
```



observation:-violin plot is the combine information of pdf(curve part) and box plot(middle part)

In [18]:
```
#contors plot
#2D Density plot, contors-plot
sns.jointplot(x="age", y="year", data=haberman1, kind="kde");
plt.suptitle('Contors plot of AGE and Op_Year')
plt.show();
```



Observation:-year of 62 to 64 and age group of 50 to 55

conclusion:With the help of eda we can apply which model is best

In [ ]: