Exploratory Data Analysis.

January 7, 2024

1 3. Plotting for Exploratory data analysis (EDA)

2 (3.1) Basic Terminology

- What is EDA?
- Data-point/vector/Observation
- Data-set.
- Feature/Variable/Input-variable/Dependent-varibale
- Label/Indepdendent-variable/Output-varible/Class/Class-label/Response label
- Vector: 2-D, 3-D, 4-D,.... n-D

Q. What is a 1-D vector: Scalar

2.1 Iris Flower dataset

Toy Dataset: Iris Dataset: [https://en.wikipedia.org/wiki/Iris_flower_data_set] * A simple dataset to learn the basics. * 3 flowers of Iris species. [see images on wikipedia link above] * 1936 by Ronald Fisher. * Petal and Sepal: http://terpconnect.umd.edu/~petersd/666/html/iris_with_labels.jpg * Objective: Classify a new flower as belonging to one of the 3 classes given the 4 features. * Importance of domain knowledge. * Why use petal and sepal dimensions as features? * Why do we not use 'color' as a feature?

[2]:
```
!wget 'https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pages/data/iris.
↪csv'
```

```
--2024-01-06 15:27:01--  https://raw.githubusercontent.com/uiuc-cse/data-
fa14/gh-pages/data/iris.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.109.133, 185.199.110.133, 185.199.111.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.109.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 3716 (3.6K) [text/plain]
Saving to: 'iris.csv'

iris.csv            100%[===================>]   3.63K  --.-KB/s    in 0s
```

2024-01-06 15:27:02 (15.7 MB/s) - ‘iris.csv’ saved [3716/3716]

[3]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np


'''downlaod iris.csv from https://raw.githubusercontent.com/uiuc-cse/data-fa14/
 ↪gh-pages/data/iris.csv'''
#Load Iris.csv into a pandas dataFrame.

iris = pd.read_csv("iris.csv")
```

[37]:
```
iris.head(10)
```

[37]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

[4]:
```
# (Q) how many data-points and features?
print (iris.shape)
```

```
(150, 5)
```

[5]:
```
#(Q) What are the column names in our dataset?
print (iris.columns)
```

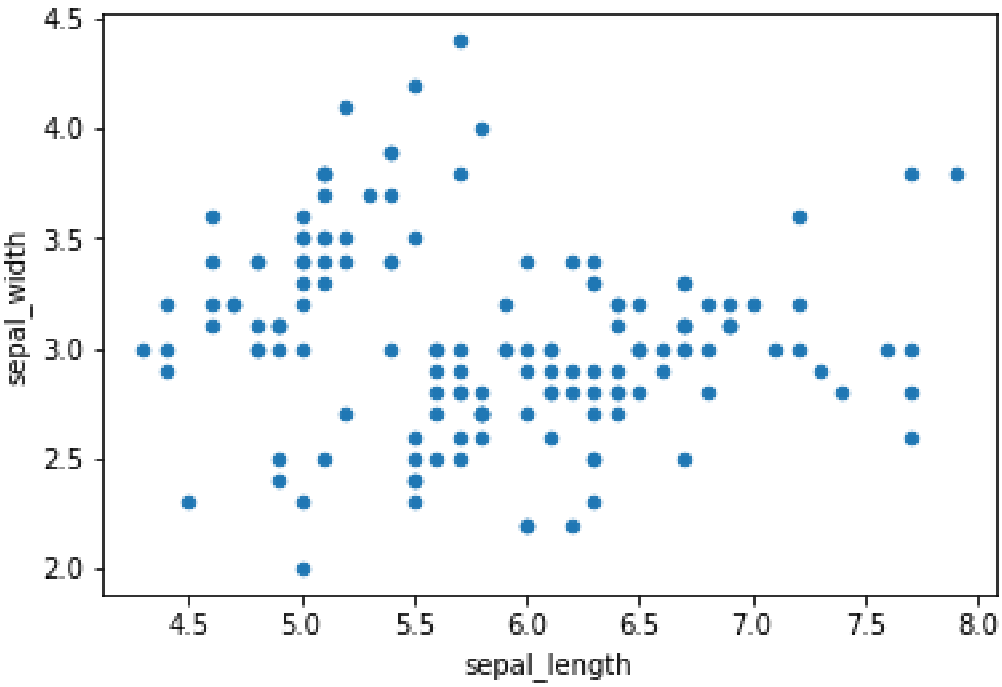
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

[6]:
```
#(Q) How many data points for each class are present?
#(or) How many flowers for each species are present?

iris["species"].value_counts()
# balanced-dataset vs imbalanced datasets
#Iris is a balanced dataset as the number of data points for every class is 50.
```

[6]:
```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

3 (3.2) 2-D Scatter Plot

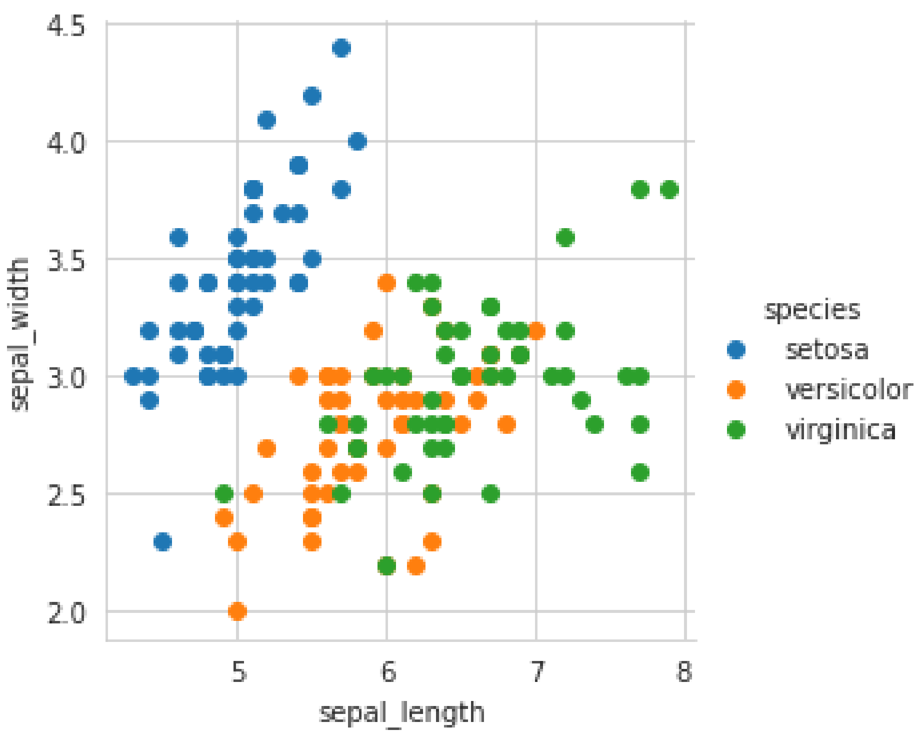
[7]:
```
#2-D scatter plot:
#ALWAYS understand the axis: labels and scale.

iris.plot(kind='scatter', x='sepal_length', y='sepal_width') ;
plt.show()

#cannot make much sense out it.
#What if we color the points by thier class-label/flower-type.
```



[8]:
```
# 2-D Scatter plot with color-coding for each flower type/class.
# Here 'sns' corresponds to seaborn.
sns.set_style("whitegrid");
sns.FacetGrid(iris, hue="species", size=4) \
   .map(plt.scatter, "sepal_length", "sepal_width") \
   .add_legend();
```

```
plt.show();

# Notice that the blue points can be easily seperated
# from red and green by drawing a line.
# But red and green data points cannot be easily seperated.
# Can we draw multiple 2-D scatter plots for each combination of features?
# How many cobinations exist? 4C2 = 6.
```

```
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has been
renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



Observation(s): 1. Using sepal_length and sepal_width features, we can distinguish Setosa flowers from others. 2. Seperating Versicolor from Viginica is much harder as they have considerable overlap.

3.1 3D Scatter plot

https://plot.ly/pandas/3d-scatter-plots/

Needs a lot to mouse interaction to interpret data.

What about 4-D, 5-D or n-D scatter plot?

4 (3.3) Pair-plot

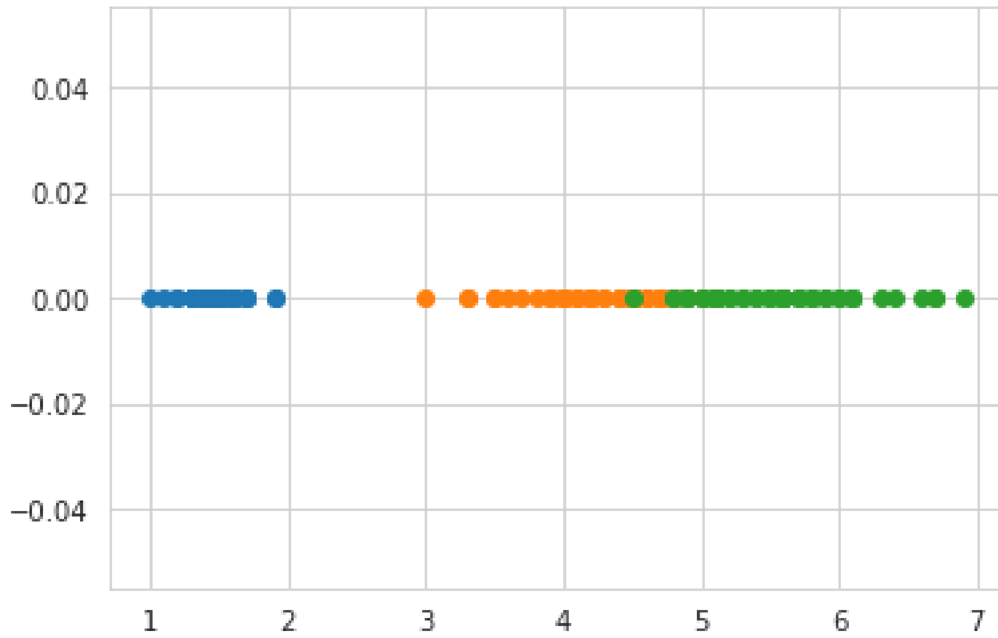
[9]:
```
# pairwise scatter plot: Pair-Plot
# Dis-advantages:
##Can be used when number of features are high.
##Cannot visualize higher dimensional patterns in 3-D and 4-D.
#Only possible to view 2D patterns.
plt.close();
sns.set_style("whitegrid");
sns.pairplot(iris, hue="species", size=3);
plt.show()
# NOTE: the diagnol elements are PDFs for each feature. PDFs are expalined␣
↪below.
```

```
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/axisgrid.py:2076: UserWarning: The `size` parameter has been
renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

Observations 1. petal_length and petal_width are the most useful features to identify various flower types. 2. While Setosa can be easily identified (linearly seperable), Virnica and Versicolor have some overlap (almost linearly seperable). 3. We can find "lines" and "if-else" conditions to build a simple model to classify the flower types.

5 (3.4) Histogram, PDF, CDF

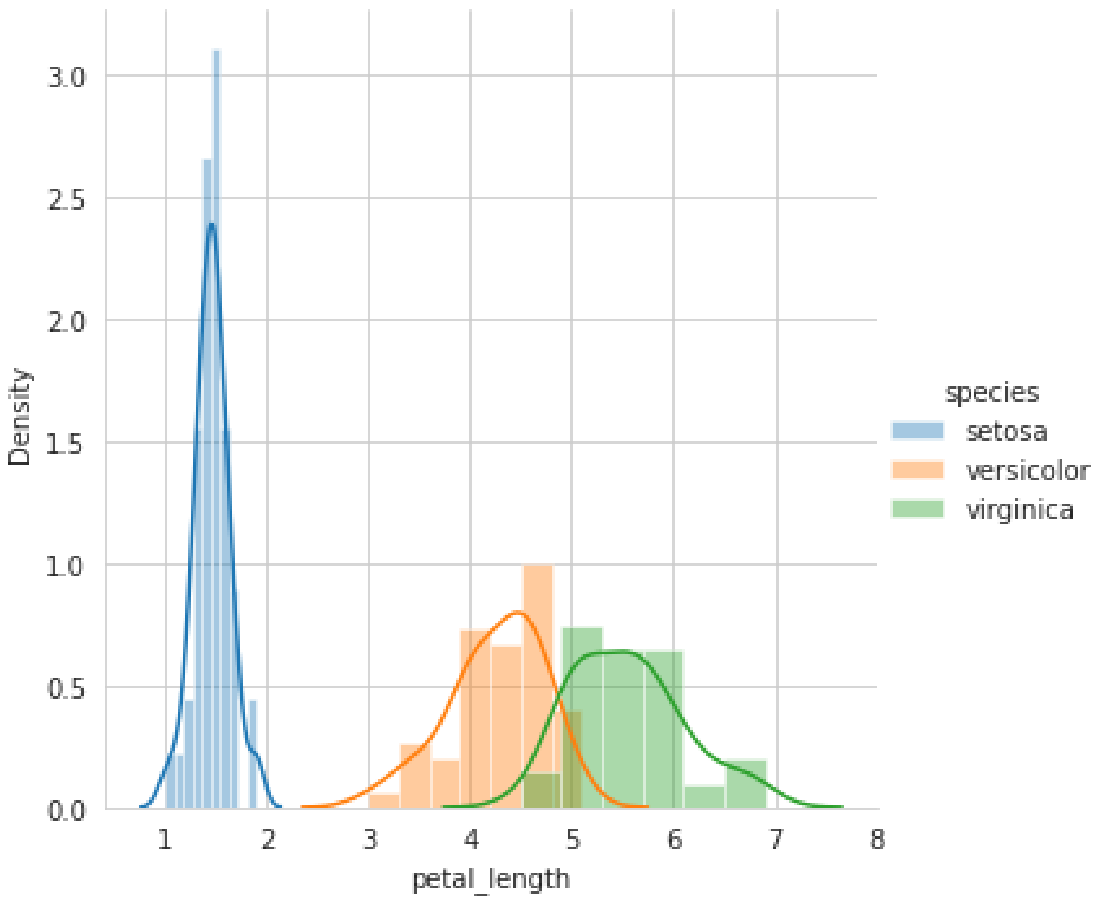
[10]:
```
# What about 1-D scatter plot using just one feature?
#1-D scatter plot of petal-length
import numpy as np
iris_setosa = iris.loc[iris["species"] == "setosa"];
iris_virginica = iris.loc[iris["species"] == "virginica"];
iris_versicolor = iris.loc[iris["species"] == "versicolor"];
#print(iris_setosa["petal_length"])
plt.plot(iris_setosa["petal_length"], np.
 ↪zeros_like(iris_setosa['petal_length']), 'o')
plt.plot(iris_versicolor["petal_length"], np.
 ↪zeros_like(iris_versicolor['petal_length']), 'o')
plt.plot(iris_virginica["petal_length"], np.
 ↪zeros_like(iris_virginica['petal_length']), 'o')

plt.show()
#Disadvantages of 1-D scatter plot: Very hard to make sense as points
#are overlapping a lot.
#Are there better ways of visualizing 1-D scatter plots?
```



[11]:
```
sns.FacetGrid(iris, hue="species", size=5) \
   .map(sns.distplot, "petal_length") \
   .add_legend();
plt.show();
```

```
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has been
renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
```

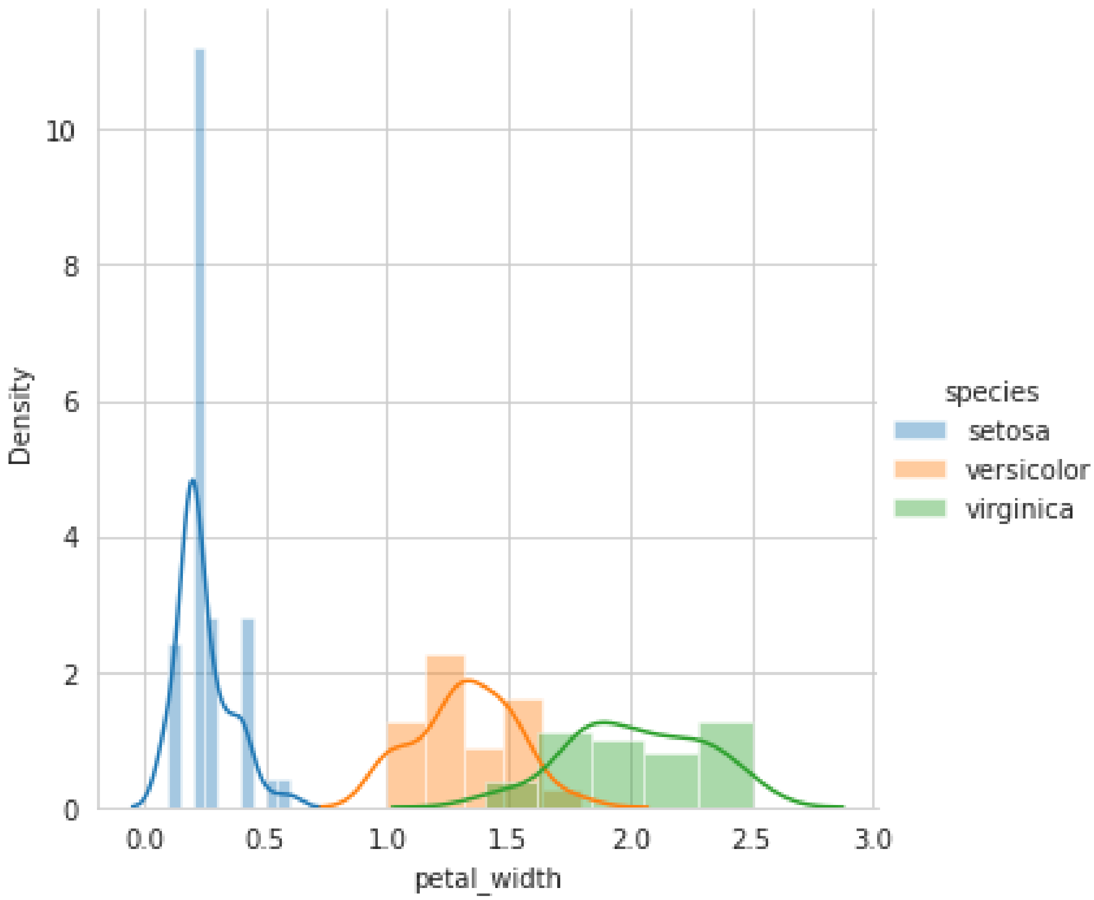
```
  warnings.warn(msg, FutureWarning)
```



[12]:
```
sns.FacetGrid(iris, hue="species", size=5) \
   .map(sns.distplot, "petal_width") \
   .add_legend();
plt.show();
```

```
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has been
renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a
```

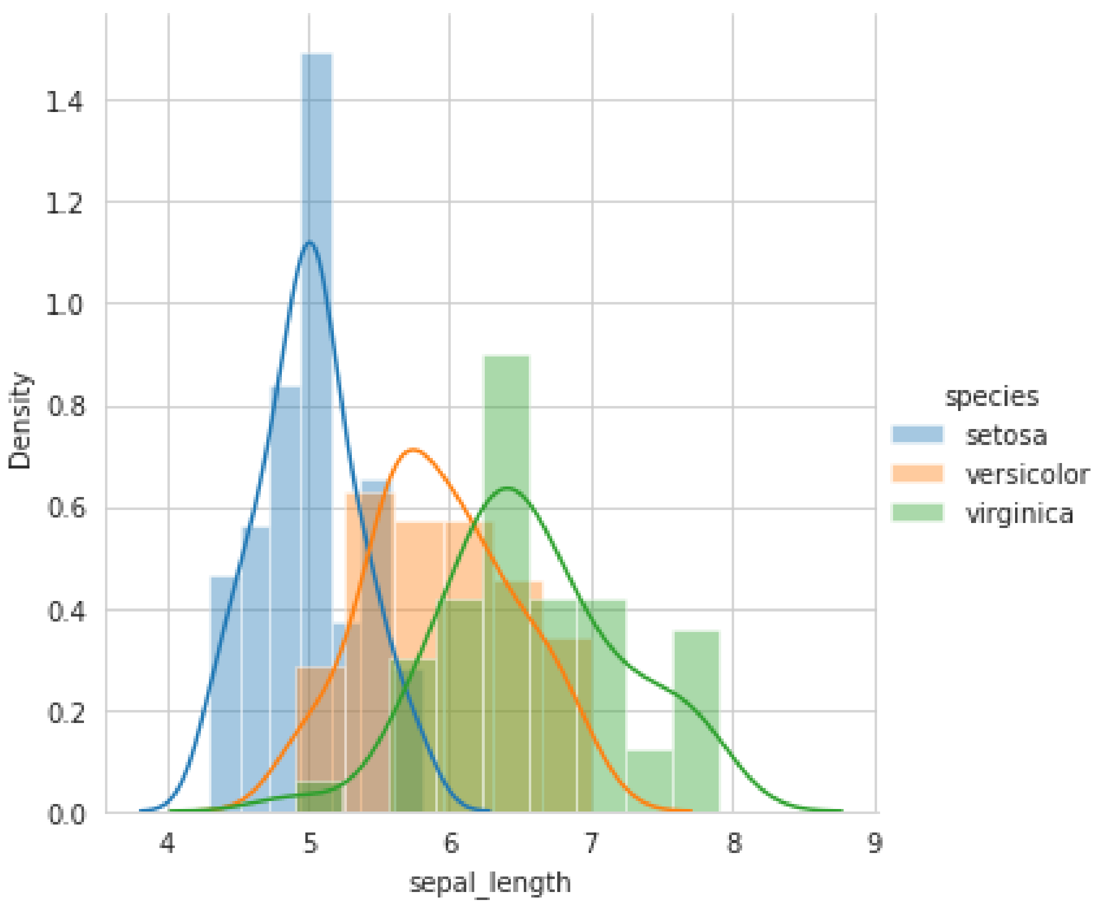
```
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
```



[13]:
```
sns.FacetGrid(iris, hue="species", size=5) \
   .map(sns.distplot, "sepal_length") \
   .add_legend();
plt.show();
```

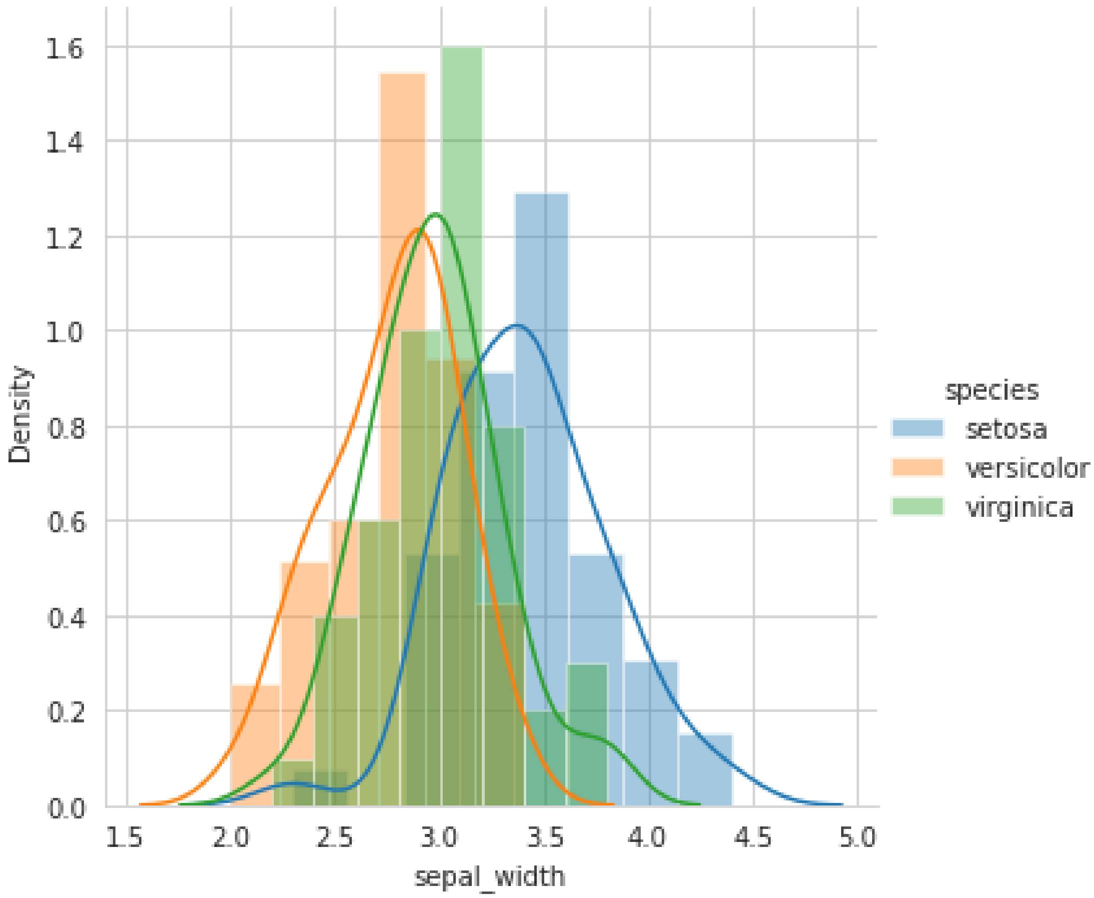
```
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has been
renamed to `height`; please update your code.
```

```
  warnings.warn(msg, UserWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
```



[14]:
```python
sns.FacetGrid(iris, hue="species", size=5) \
   .map(sns.distplot, "sepal_width") \
   .add_legend();
plt.show();
```

```
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has been
renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
```



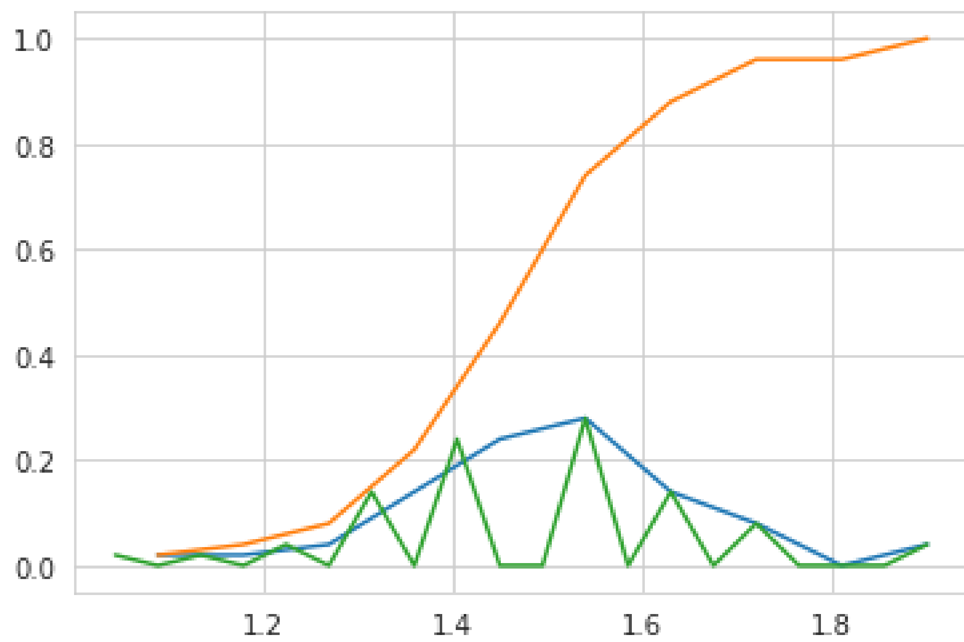
[15]:
```
# Histograms and Probability Density Functions (PDF) using KDE
# How to compute PDFs using counts/frequencies of data points in each window.
# How window width effects the PDF plot.


# Interpreting a PDF:
## why is it called a density plot?
## Why is it called a probability plot?
## for each value of petal_length, what does the value on y-axis mean?
# Notice that we can write a simple if..else condition as if(petal_length) < 2.
 ↪5 then flower type is setosa.
# Using just one feature, we can build a simple "model" suing if..else...␣
 ↪statements.

# Disadv of PDF: Can we say what percentage of versicolor points have a␣
 ↪petal_length of less than 5?

# Do some of these plots look like a bell-curve you studied in under-grad?
```

```
# Gaussian/Normal distribution.
# What is "normal" about normal distribution?
# e.g: Hieghts of male students in a class.
# One of the most frequent distributions in nature.
```

[16]:

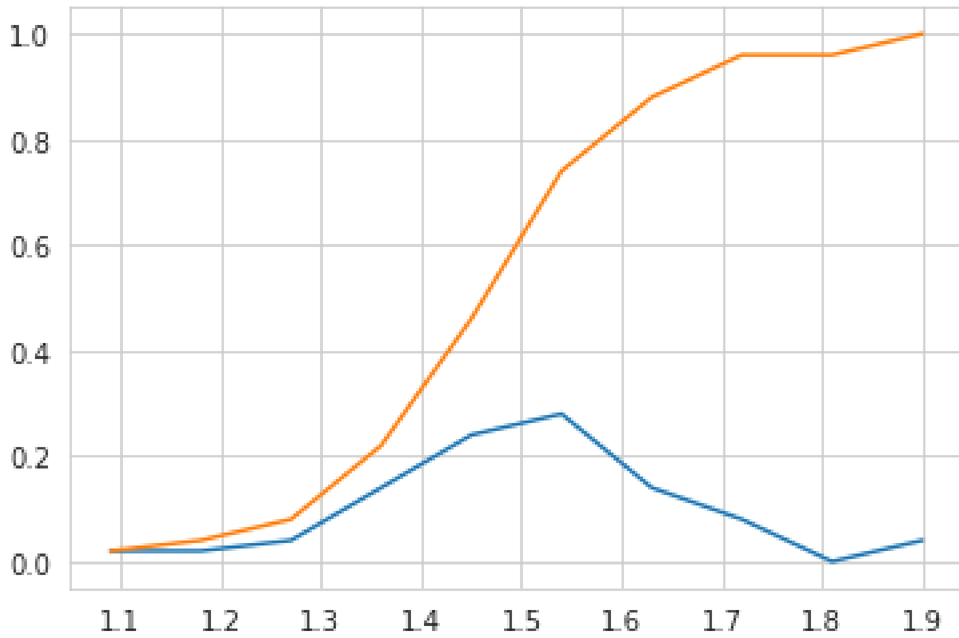
```
# Need for Cumulative Distribution Function (CDF)
# We can visually see what percentage of versicolor flowers have a
# petal_length of less than 5?
# How to construct a CDF?
# How to read a CDF?

#Plot CDF of petal_length

counts, bin_edges = np.histogram(iris_setosa['petal_length'], 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(iris_setosa['petal_length'], bins=20,
                                 density = True)
pdf = counts/(sum(counts))
plt.plot(bin_edges[1:],pdf);

plt.show();
```

```
[0.02 0.02 0.04 0.14 0.24 0.28 0.14 0.08 0.   0.04]
[1.   1.09 1.18 1.27 1.36 1.45 1.54 1.63 1.72 1.81 1.9 ]
```



[17]:
```
# Need for Cumulative Distribution Function (CDF)
# We can visually see what percentage of versicolor flowers have a
# petal_length of less than 1.6?
# How to construct a CDF?
# How to read a CDF?

#Plot CDF of petal_length

counts, bin_edges = np.histogram(iris_setosa['petal_length'], 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();
```

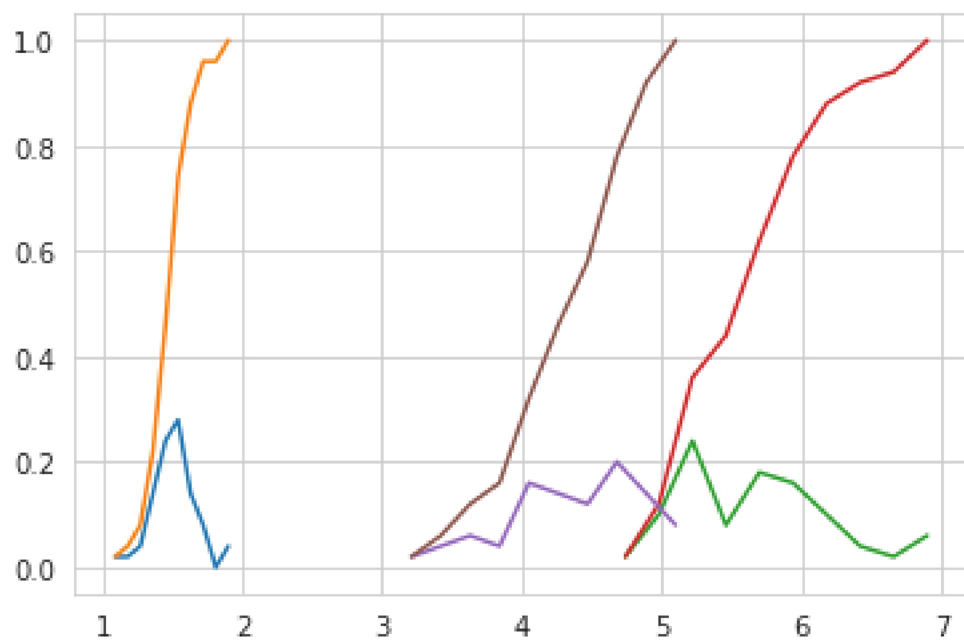
```
[0.02 0.02 0.04 0.14 0.24 0.28 0.14 0.08 0.   0.04]
[1.   1.09 1.18 1.27 1.36 1.45 1.54 1.63 1.72 1.81 1.9 ]
```



[18]:
```
# Plots of CDF of petal_length for various types of flowers.

# Misclassification error if you use petal_length only.

counts, bin_edges = np.histogram(iris_setosa['petal_length'], 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)


# virginica
counts, bin_edges = np.histogram(iris_virginica['petal_length'], 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)


#versicolor
```

```
counts, bin_edges = np.histogram(iris_versicolor['petal_length'], 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.show();
```

```
[0.02 0.02 0.04 0.14 0.24 0.28 0.14 0.08 0.   0.04]
[1.   1.09 1.18 1.27 1.36 1.45 1.54 1.63 1.72 1.81 1.9 ]
[0.02 0.1  0.24 0.08 0.18 0.16 0.1  0.04 0.02 0.06]
[4.5  4.74 4.98 5.22 5.46 5.7  5.94 6.18 6.42 6.66 6.9 ]
[0.02 0.04 0.06 0.04 0.16 0.14 0.12 0.2  0.14 0.08]
[3.   3.21 3.42 3.63 3.84 4.05 4.26 4.47 4.68 4.89 5.1 ]
```



6 (3.5) Mean, Variance and Std-dev

[19]:
```
#Mean, Variance, Std-deviation,
print("Means:")
print(np.mean(iris_setosa["petal_length"]))
#Mean with an outlier.
print(np.mean(np.append(iris_setosa["petal_length"],50)));
print(np.mean(iris_virginica["petal_length"]))
print(np.mean(iris_versicolor["petal_length"]))

print("\nStd-dev:");
print(np.std(iris_setosa["petal_length"]))
print(np.std(iris_virginica["petal_length"]))
print(np.std(iris_versicolor["petal_length"]))
```

```
Means:
1.464
2.4156862745098038
5.552
4.26

Std-dev:
0.17176728442867115
0.5463478745268441
0.4651881339845204
```

7 (3.6) Median, Percentile, Quantile, IQR, MAD

[20]:
```
#Median, Quantiles, Percentiles, IQR.
print("\nMedians:")
print(np.median(iris_setosa["petal_length"]))
#Median with an outlier
print(np.median(np.append(iris_setosa["petal_length"],50)));
print(np.median(iris_virginica["petal_length"]))
print(np.median(iris_versicolor["petal_length"]))


print("\nQuantiles:")
print(np.percentile(iris_setosa["petal_length"],np.arange(0, 100, 25)))
print(np.percentile(iris_virginica["petal_length"],np.arange(0, 100, 25)))
print(np.percentile(iris_versicolor["petal_length"], np.arange(0, 100, 25)))

print("\n90th Percentiles:")
print(np.percentile(iris_setosa["petal_length"],90))
print(np.percentile(iris_virginica["petal_length"],90))
```

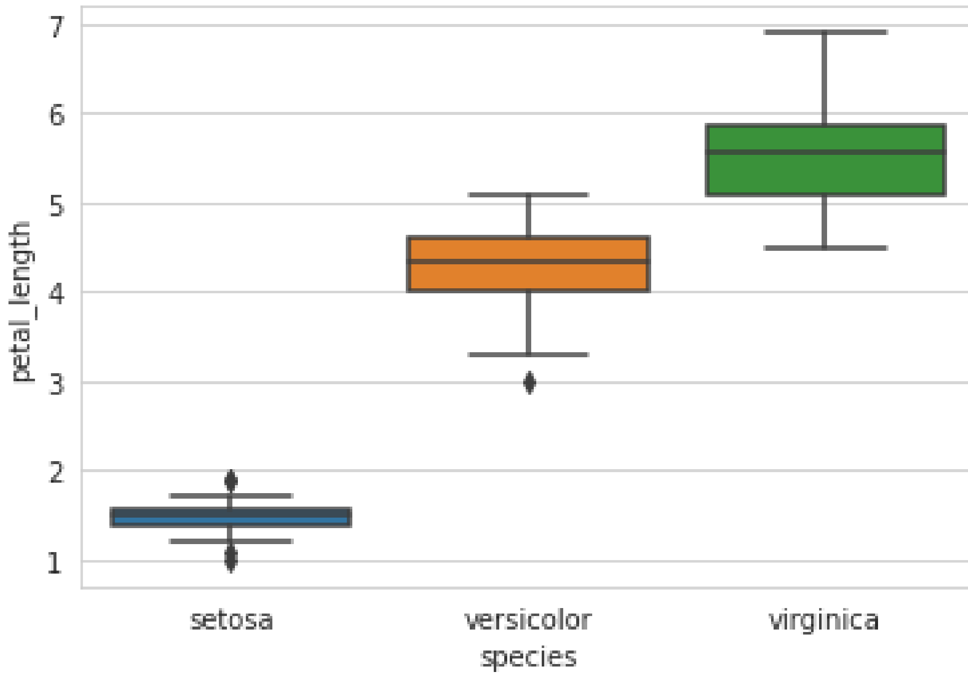
```
print(np.percentile(iris_versicolor["petal_length"], 90))

from statsmodels import robust
print ("\nMedian Absolute Deviation")
print(robust.mad(iris_setosa["petal_length"]))
print(robust.mad(iris_virginica["petal_length"]))
print(robust.mad(iris_versicolor["petal_length"]))
```

```
Medians:
1.5
1.5
5.55
4.35

Quantiles:
[1.    1.4   1.5   1.575]
[4.5   5.1   5.55  5.875]
[3.   4.   4.35 4.6 ]

90th Percentiles:
1.7
6.31
4.8

Median Absolute Deviation
0.14826022185056031
0.6671709983275211
0.5189107764769602
```

8 (3.7) Box plot and Whiskers

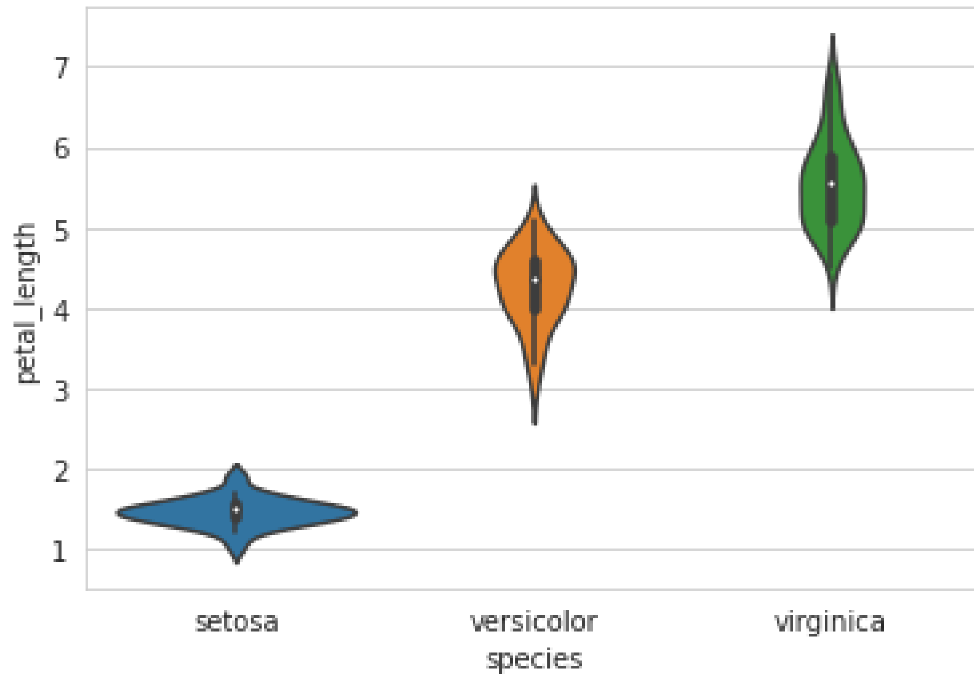
[21]:
```
#Box-plot with whiskers: another method of visualizing the  1-D scatter plot␣
↪more intuitivey.
# The Concept of median, percentile, quantile.
# How to draw the box in the box-plot?
# How to draw whiskers: [no standard way] Could use min and max or use other␣
↪complex statistical techniques.
# IQR like idea.

#NOTE: IN the plot below, a technique call inter-quartile range is used in␣
↪plotting the whiskers.
#Whiskers in the plot below donot correposnd to the min and max values.

#Box-plot can be visualized as a PDF on the side-ways.
```

```
sns.boxplot(x='species',y='petal_length', data=iris)
plt.show()
```



9 (3.8) Violin plots

[22]:
```
# A violin plot combines the benefits of the previous two plots
#and simplifies them

# Denser regions of the data are fatter, and sparser ones thinner
#in a violin plot

sns.violinplot(x="species", y="petal_length", data=iris, size=8)
plt.show()
```



10 (3.9) Summarizing plots in english

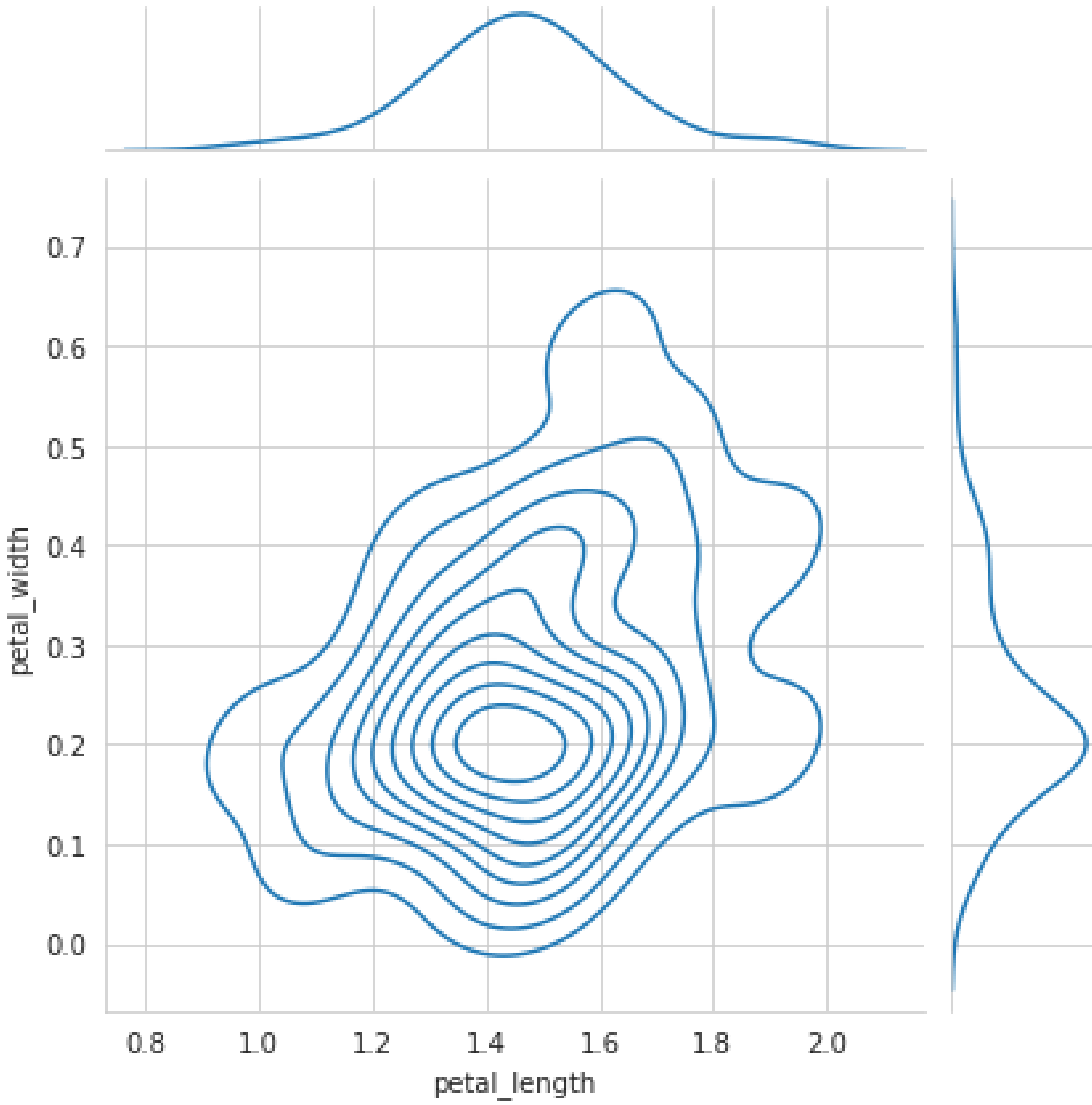
- Exaplain your findings/conclusions in plain english
- Never forget your objective (the probelm you are solving) . Perform all of your EDA aligned with your objectives.

11 (3.10) Univariate, bivariate and multivariate analysis.

Def: Univariate, Bivariate and Multivariate analysis.

12 (3.11) Multivariate probability density, contour plot.

[24]:
```
#2D Density plot, contors-plot
sns.jointplot(x="petal_length", y="petal_width", data=iris_setosa, kind="kde");
plt.show();
```



[25]:
```
iris_virginica_SW = iris_virginica.iloc[:,1]
iris_versicolor_SW = iris_versicolor.iloc[:,1]
```

[26]:
```
from scipy import stats
stats.ks_2samp(iris_virginica_SW, iris_versicolor_SW)
```

[26]: KstestResult(statistic=0.26, pvalue=0.06779471096995852)

[27]:
```
x = stats.norm.rvs(loc=0.2, size=10)
stats.kstest(x,'norm')
```

[27]: KstestResult(statistic=0.2600378156134869, pvalue=0.43494845163471907)

[28]:
```
x = stats.norm.rvs(loc=0.2, size=100)
stats.kstest(x,'norm')
```

[28]:
```
KstestResult(statistic=0.1753936220625653, pvalue=0.0036749452453584994)
```

[29]:
```
x = stats.norm.rvs(loc=0.2, size=1000)
stats.kstest(x,'norm')
```

[29]:
```
KstestResult(statistic=0.11540190829820307, pvalue=4.656618389194487e-12)
```

13 (3.12) 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 alanlaysis as above on this dataset with the following sections:

- High level statistics of the dataset: number of points, numer of features, number of classes, data-points per class.
- Explain our objective.
- Perform Univaraite analysis(PDF, CDF, Boxplot, Voilin plots) to understand which features are useful towards classification.
- Perform Bi-variate analysis (scatter plots, pair-plots) to see if combinations of features are useful in classfication.
- Write your observations in english as crisply and unambigously as possible. Always quantify your results.

[30]:
```
import pandas as pd
```

[31]:
```
df = pd.read_csv('./haberman.csv')
```

[32]:
```
df.head(10)
```

[32]:

	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
5	33	58	10	1
6	33	60	0	1
7	34	59	0	2
8	34	66	9	2
9	34	58	30	1

[34]:
```
df.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
```

[40]:
```
df.columns # column names
```

[40]:
```
Index(['age', 'year', 'nodes', 'status'], dtype='object')
```

[39]:
```
df.shape # number of data points and features
```

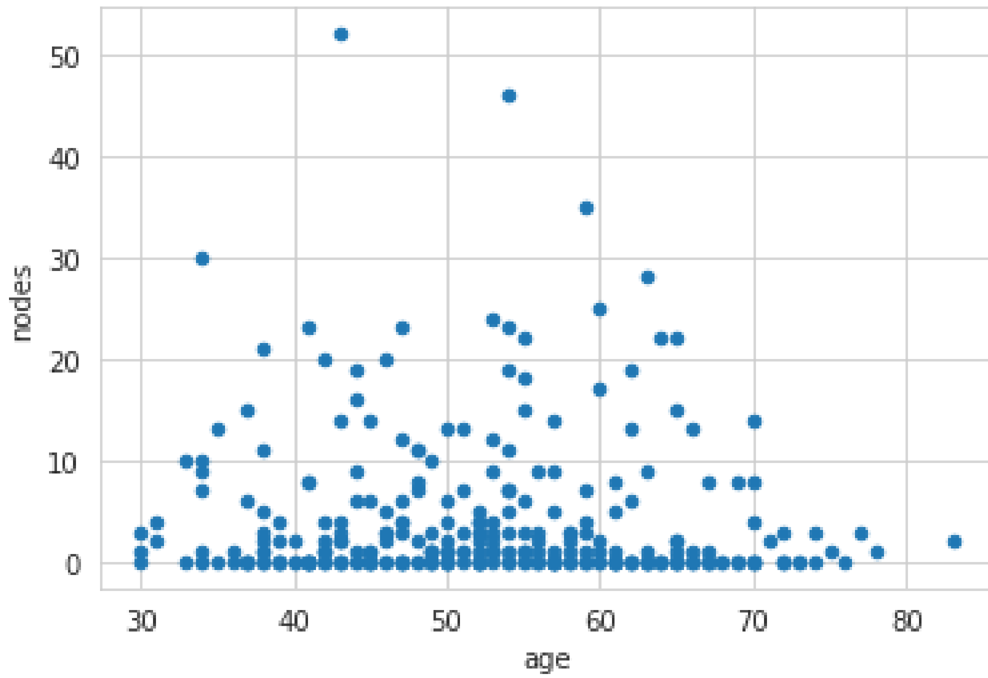
[39]:
```
(306, 4)
```

[42]:
```
df['status'].value_counts()
# it's imbalanced dataset, since for class 1 225 and for class 2 only 81␣
↪datapoints are available.
```

[42]:
```
1    225
2     81
Name: status, dtype: int64
```

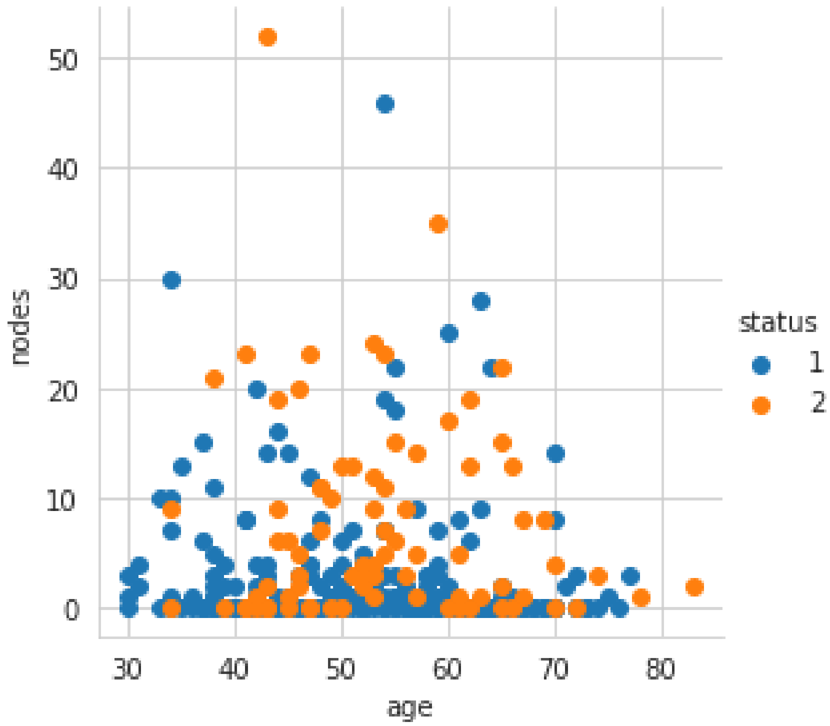
13.1 2-D Scatter Plot

[43]:
```
df.plot(kind='scatter', x='age', y='nodes')
plt.show()
```



[50]:
```
# 2-D Scatter plot with color-coding for each flower type/class.
# Here 'sns' corresponds to seaborn.
sns.set_style("whitegrid");
sns.FacetGrid(df, hue="status", size=4) \
   .map(plt.scatter, "age", "nodes", ) \
   .add_legend();
plt.show();
```

```
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has been
renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

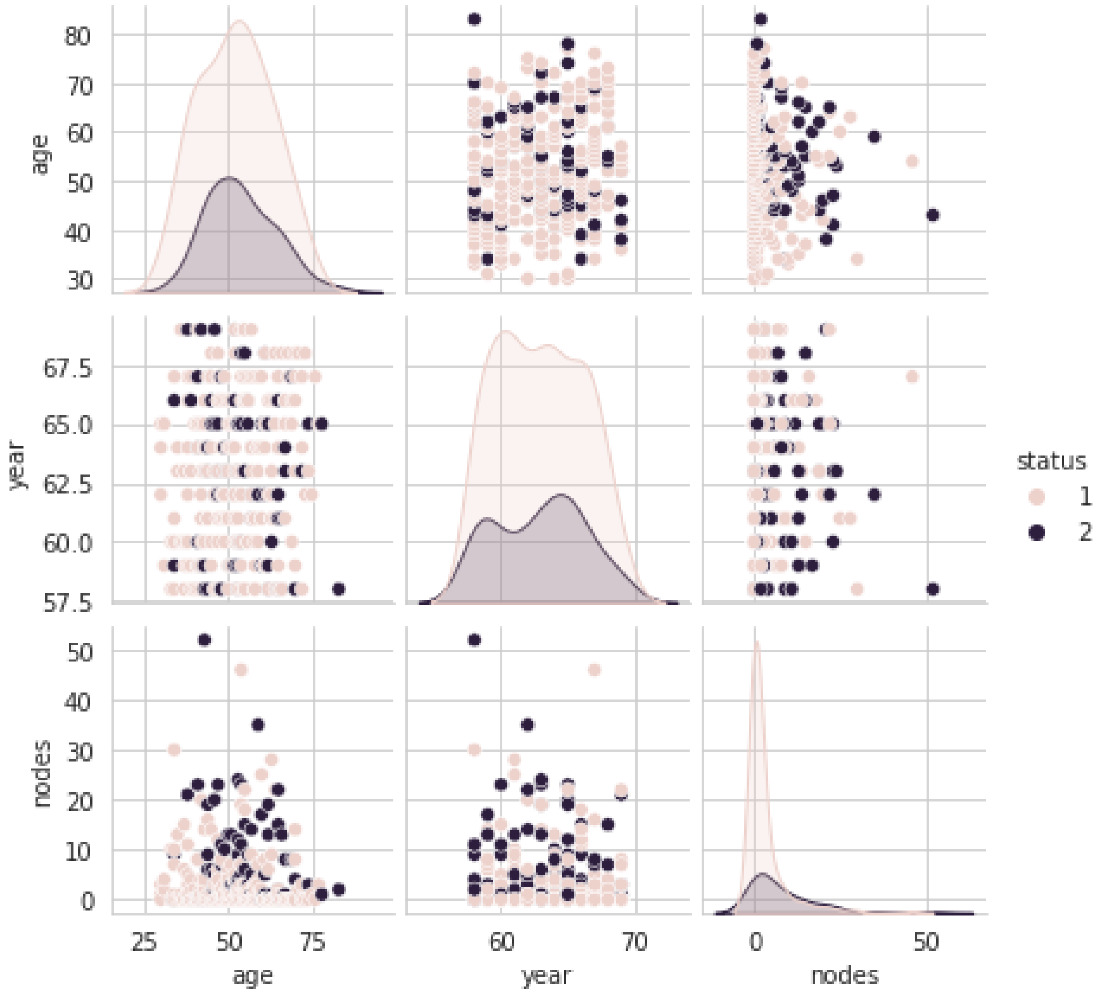


- We can't distinguish between the status using age and nodes and any other combinations of features.

14 Pair-Plot

[52]:
```
# pairwise scatter plot: Pair-Plot
# Dis-advantages:
##Can be used when number of features are high.
##Cannot visualize higher dimensional patterns in 3-D and 4-D.
#Only possible to view 2D patterns.
plt.close();
sns.set_style("whitegrid");
sns.pairplot(df, hue="status", size=2);
plt.show()
# NOTE: the diagnol elements are PDFs for each feature. PDFs are expalined␣
↪below.
```

```
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/axisgrid.py:2076: UserWarning: The `size` parameter has been
renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

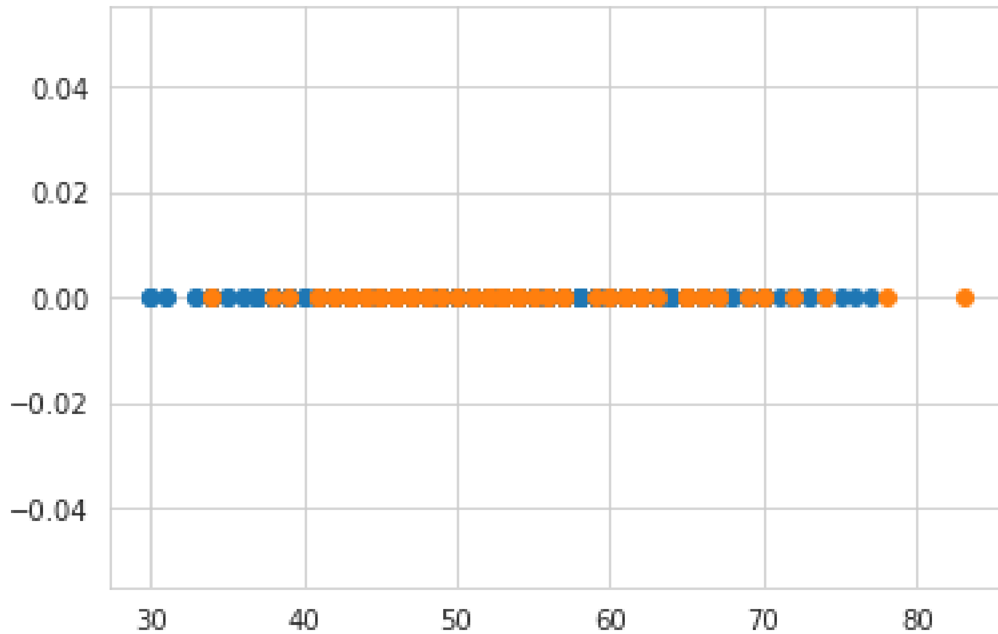


- nodes and age are most useful features
- year and nodes seems little bit okay but it would not be okay to use these features since there is no relation in year and age of a person.

15 Histogram, PDF, CDF

[64]:
```
# What about 1-D scatter plot using just one feature?
#1-D scatter plot of petal-length
import numpy as np
df_alive = df.loc[df["status"] == 1];
df_died = df.loc[df["status"] == 2];
#print(iris_setosa["petal_length"])
plt.plot(df_alive["age"], np.zeros_like(df_alive['age']), 'o')
plt.plot(df_died["age"], np.zeros_like(df_died['age']), 'o')
```

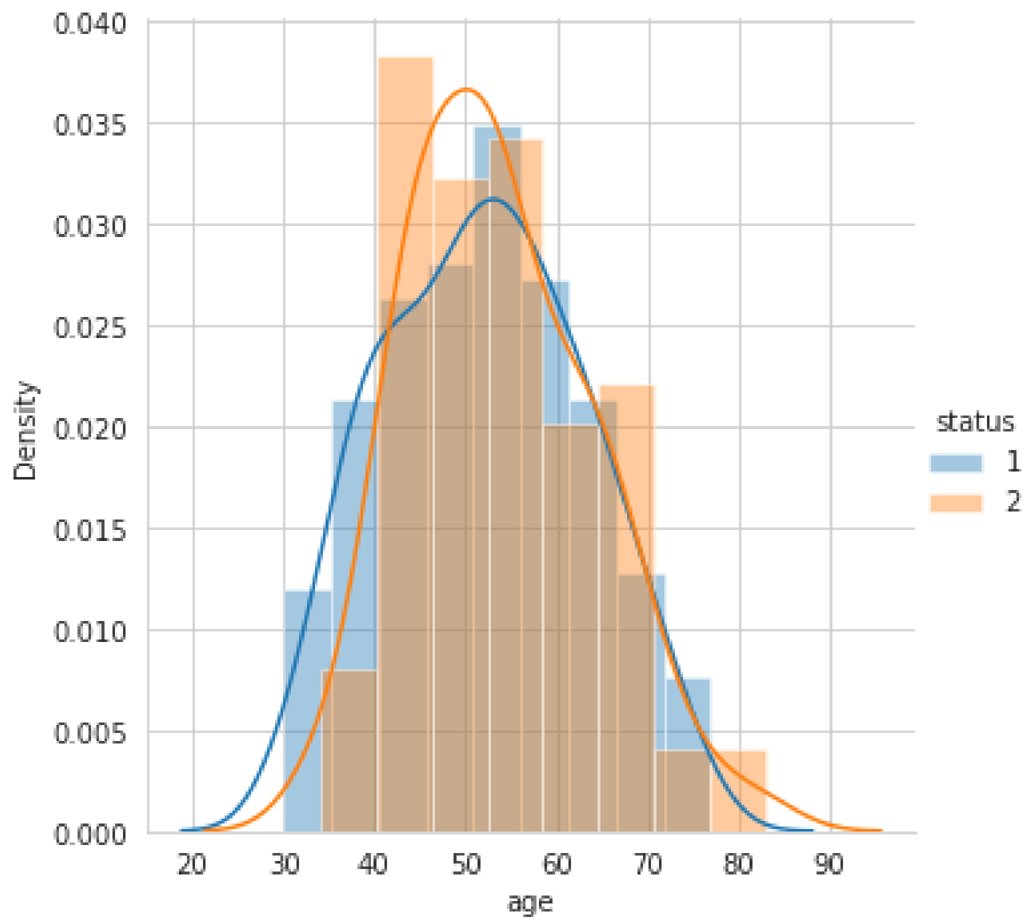
```
plt.show()
#Disadvantages of 1-D scatter plot: Very hard to make sense as points
#are overlapping a lot.
#Are there better ways of visualizing 1-D scatter plots?
```



[68]:
```
sns.FacetGrid(df, hue="status", size=5) \
   .map(sns.distplot, "age") \
   .add_legend();
plt.show();
```

```
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has been
renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
```

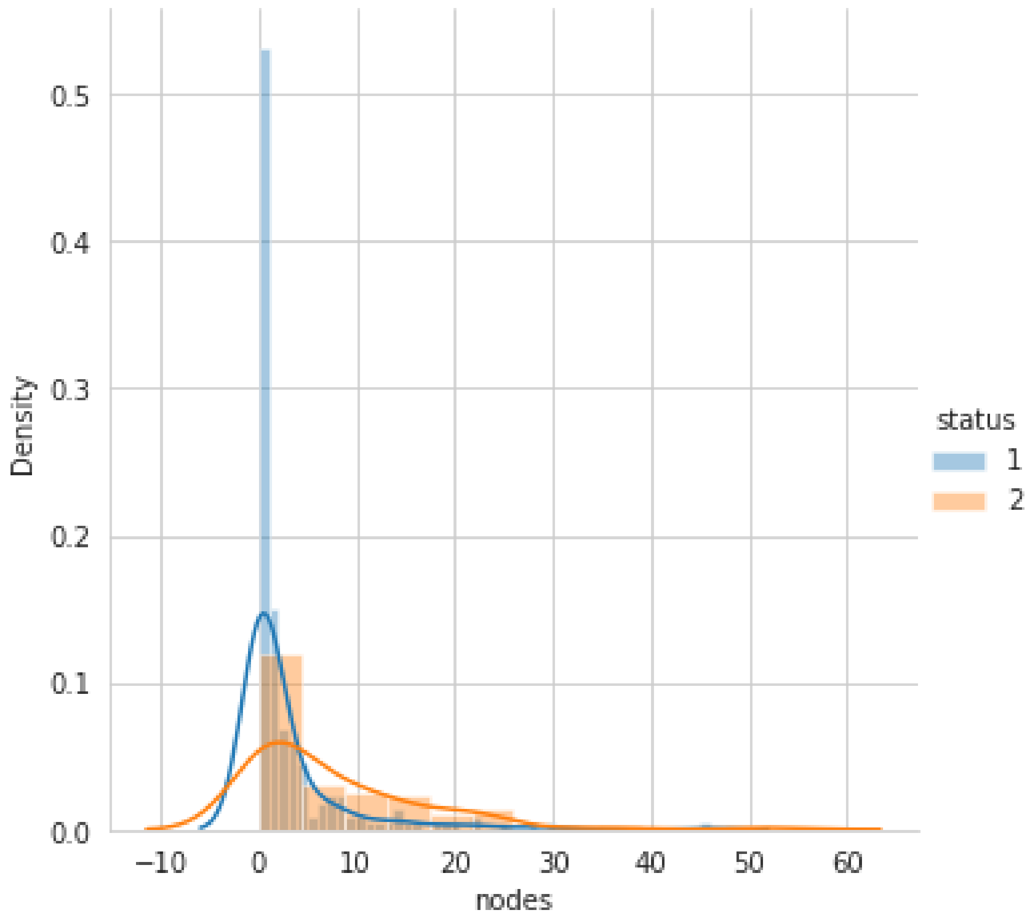
warnings.warn(msg, FutureWarning)



[69]:
```
sns.FacetGrid(df, hue="status", size=5) \
    .map(sns.distplot, "nodes") \
    .add_legend();
plt.show();
```

```
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has been
renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a
```

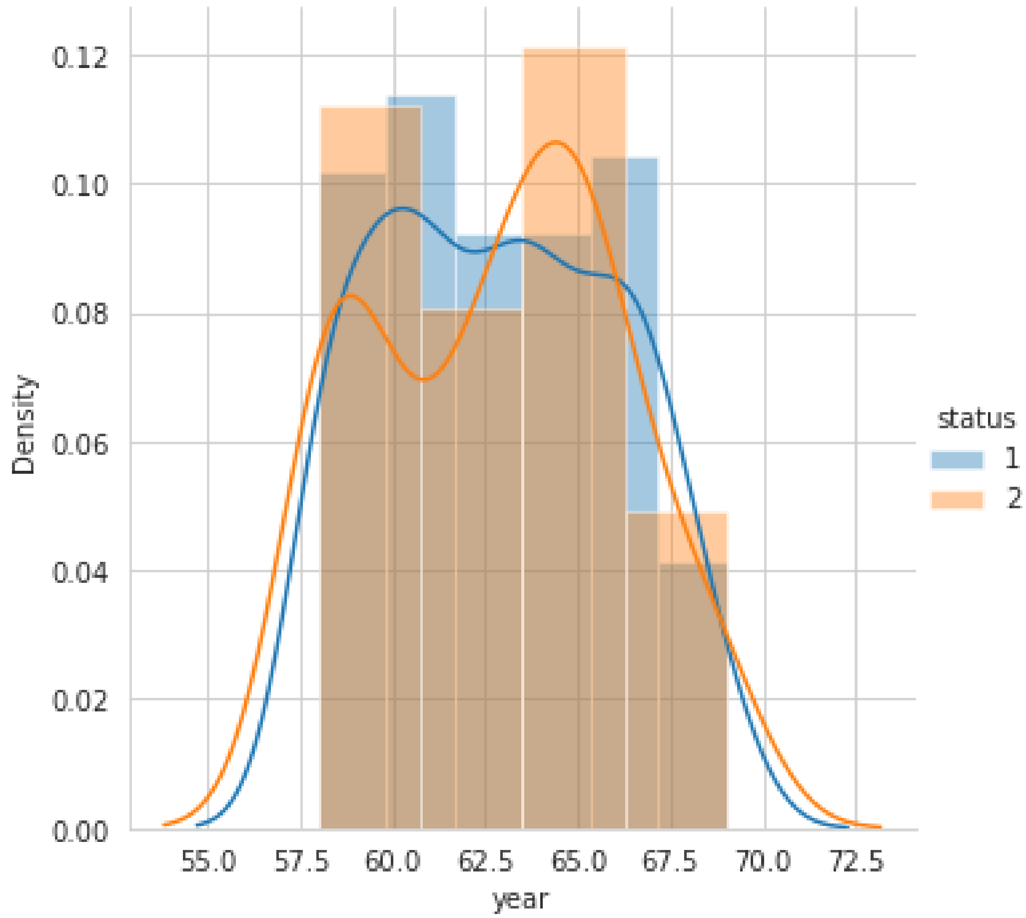
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)



[70]:
```
sns.FacetGrid(df, hue="status", size=5) \
    .map(sns.distplot, "year") \
    .add_legend();
plt.show();
```

/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has been
renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).

```
  warnings.warn(msg, FutureWarning)
/home/ajaz/anaconda/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: 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 histograms).
  warnings.warn(msg, FutureWarning)
```

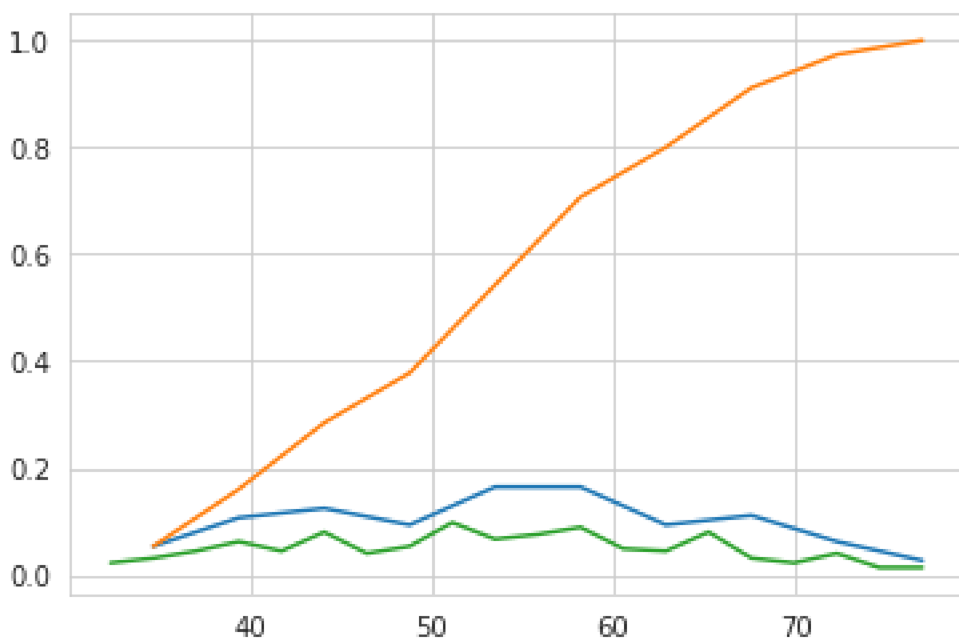


[73]:

```
# Need for Cumulative Distribution Function (CDF)
# We can visually see what percentage of versicolor flowers have a
# petal_length of less than 5?
# How to construct a CDF?
# How to read a CDF?

#Plot CDF of petal_length

counts, bin_edges = np.histogram(df_alive['age'], bins=10,
                                 density = True)
```

```python
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(df_alive['age'], bins=20,
                                 density = True)
pdf = counts/(sum(counts))
plt.plot(bin_edges[1:],pdf);

plt.show();
```

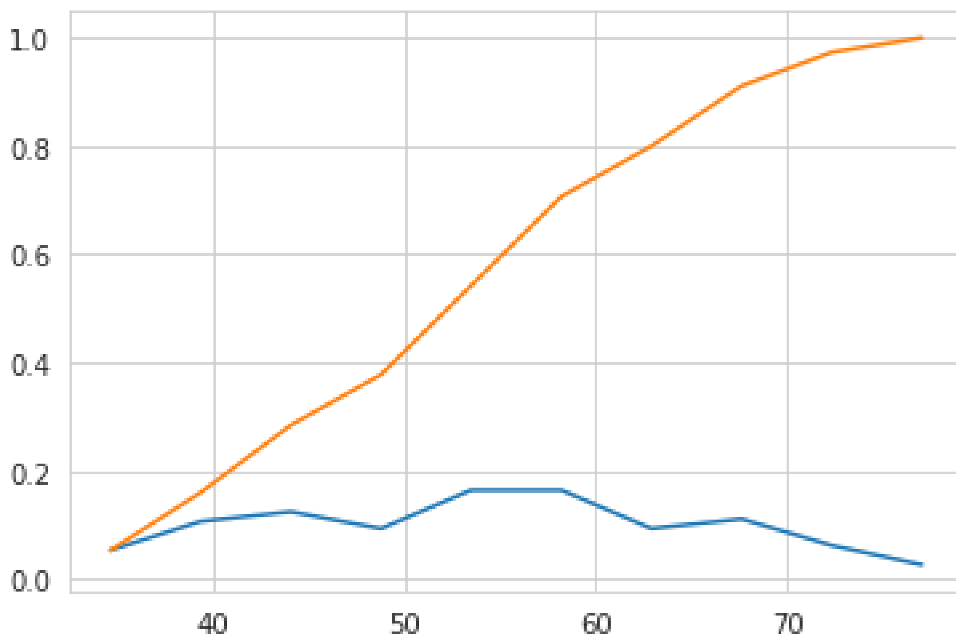
```
[0.05333333 0.10666667 0.12444444 0.09333333 0.16444444 0.16444444
 0.09333333 0.11111111 0.06222222 0.02666667]
[30.  34.7 39.4 44.1 48.8 53.5 58.2 62.9 67.6 72.3 77. ]
```



[74]:
```python
# Need for Cumulative Distribution Function (CDF)
# We can visually see what percentage of versicolor flowers have a
# petal_length of less than 1.6?
# How to construct a CDF?
# How to read a CDF?

#Plot CDF of petal_length
```

```
counts, bin_edges = np.histogram(df_alive['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();
```

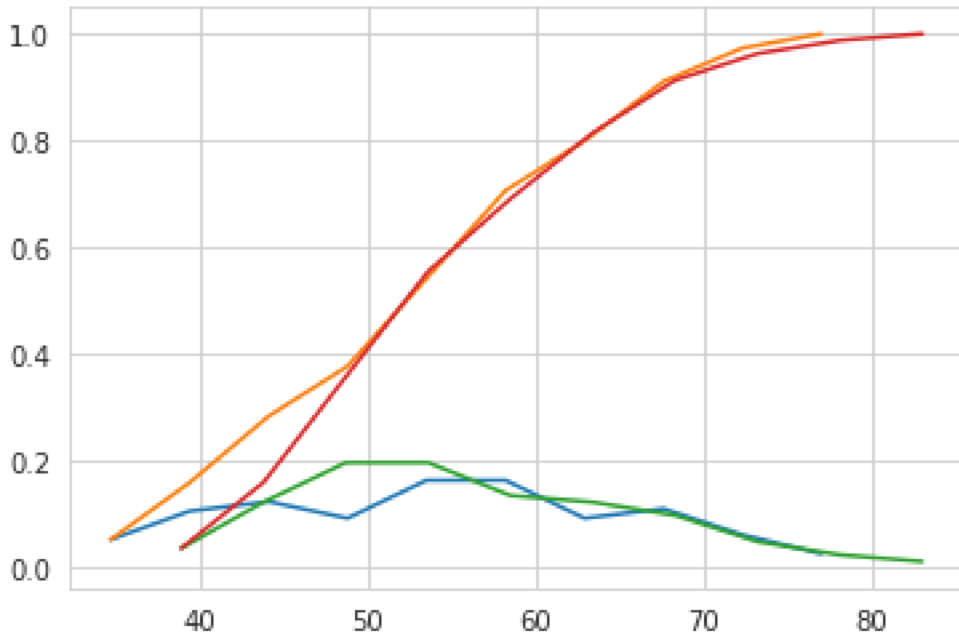
```
[0.05333333 0.10666667 0.12444444 0.09333333 0.16444444 0.16444444
 0.09333333 0.11111111 0.06222222 0.02666667]
[30.  34.7 39.4 44.1 48.8 53.5 58.2 62.9 67.6 72.3 77. ]
```



[75]:
```
# Plots of CDF of petal_length for various types of flowers.

# Misclassification error if you use petal_length only.
```

```python
# survived
counts, bin_edges = np.histogram(df_alive['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)


# not survived
counts, bin_edges = np.histogram(df_died['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.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. ]
```



16 Mean, Variance and Std-dev

[76]:
```
#Mean, Variance, Std-deviation,
print("Means:")
print(np.mean(df_alive["age"]))
#Mean with an outlier.
# print(np.mean(np.append(iris_setosa["petal_length"],50)));
print(np.mean(df_died["age"]))

print("\nStd-dev:");
print(np.std(df_alive["age"]))
print(np.std(df_died["age"]))
```

```
Means:
52.017777777777778
53.67901234567901

Std-dev:
10.98765547510051
10.10418219303131
```

17 (3.6) Median, Percentile, Quantile, IQR, MAD

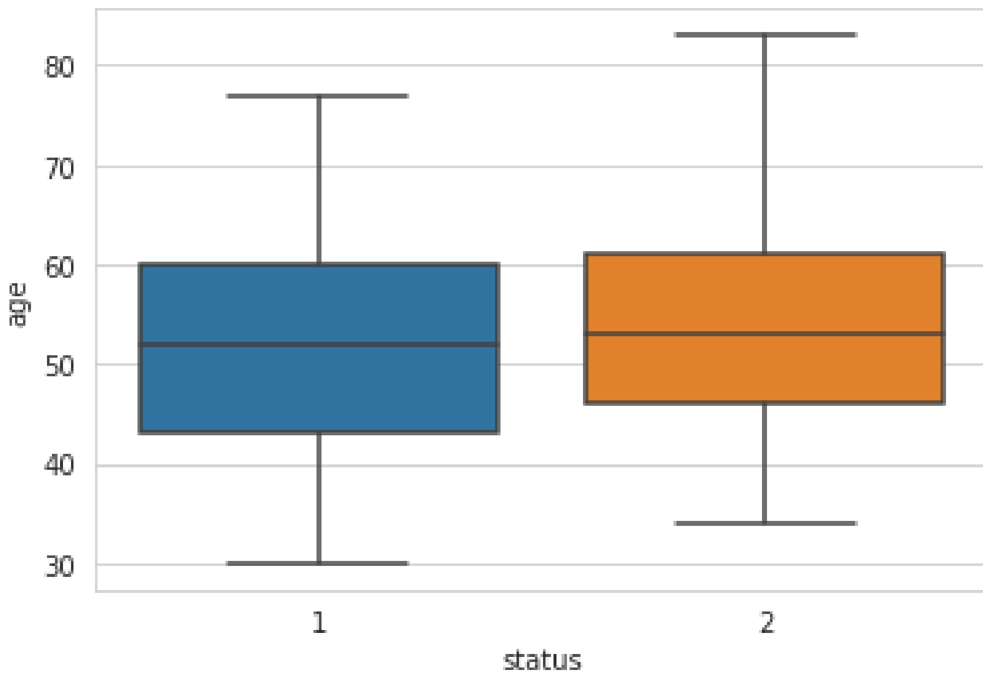
[79]:
```
#Median, Quantiles, Percentiles, IQR.
print("\nMedians:")
print(np.median(df_alive["age"]))
print(np.median(df_died["age"]))


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

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

from statsmodels import robust
print ("\nMedian Absolute Deviation")
print(robust.mad(df_alive["age"]))
print(robust.mad(df_died["age"]))
```

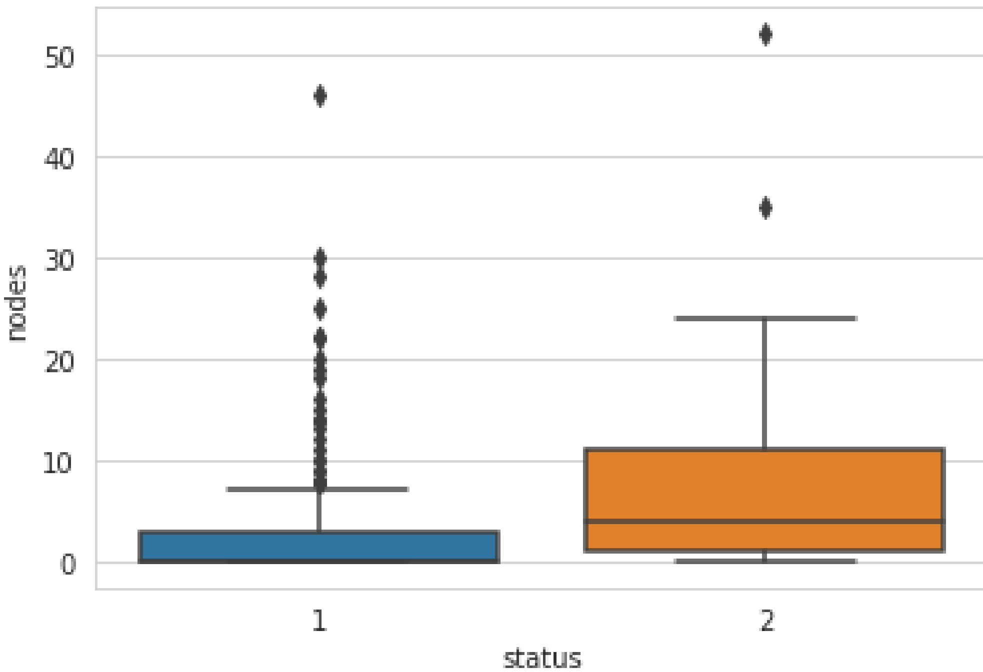
```
Medians:
52.0
53.0

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

90th Percentiles:
67.0
67.0

Median Absolute Deviation
13.343419966550417
11.860817748044816
```

18 Bos plot and Whiskers

[84]:
```
#Box-plot with whiskers: another method of visualizing the  1-D scatter plot␣
 ↪more intuitivey.
# The Concept of median, percentile, quantile.
# How to draw the box in the box-plot?
```

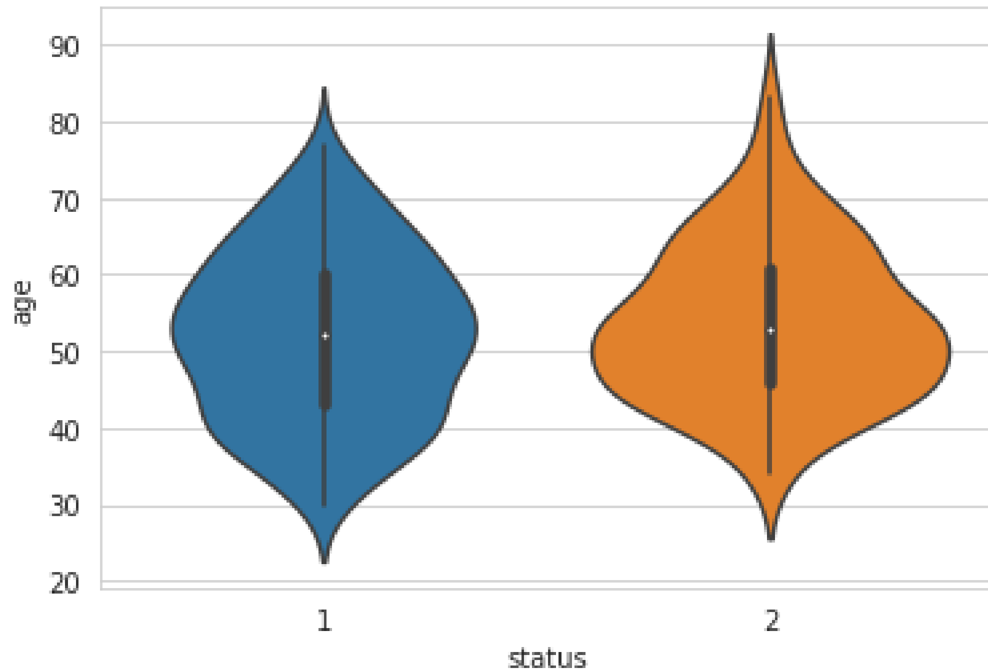
```python
# How to draw whiskers: [no standard way] Could use min and max or use other␣
↪complex statistical techniques.
# IQR like idea.

#NOTE: IN the plot below, a technique call inter-quartile range is used in␣
↪plotting the whiskers.
#Whiskers in the plot below donot correposnd to the min and max values.

#Box-plot can be visualized as a PDF on the side-ways.

sns.boxplot(x='status',y='age', data=df)
plt.show()

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

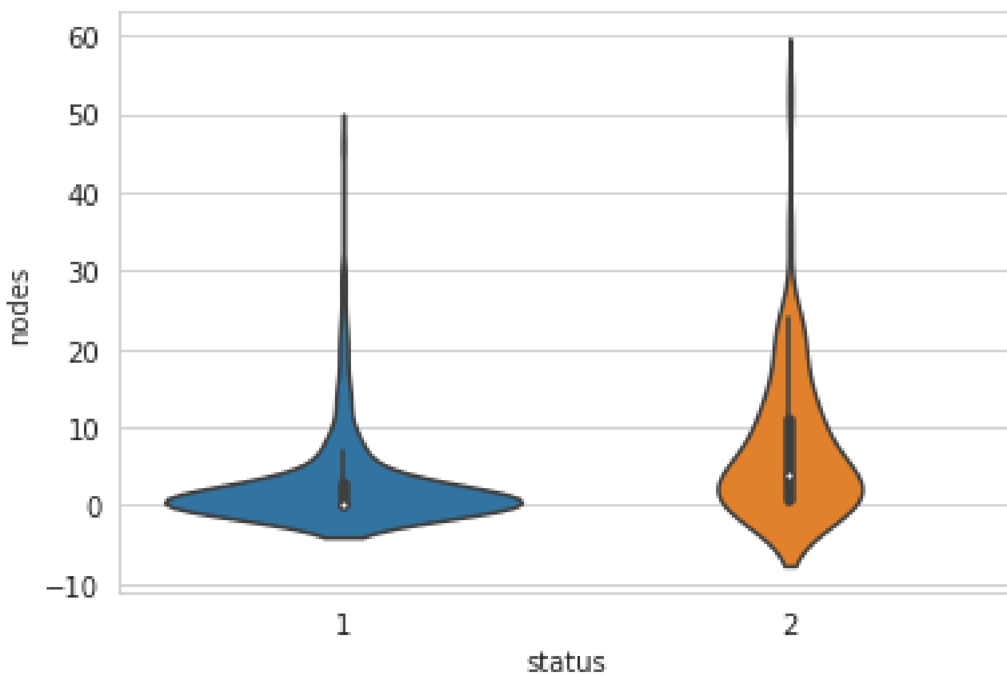




19 Violin plots

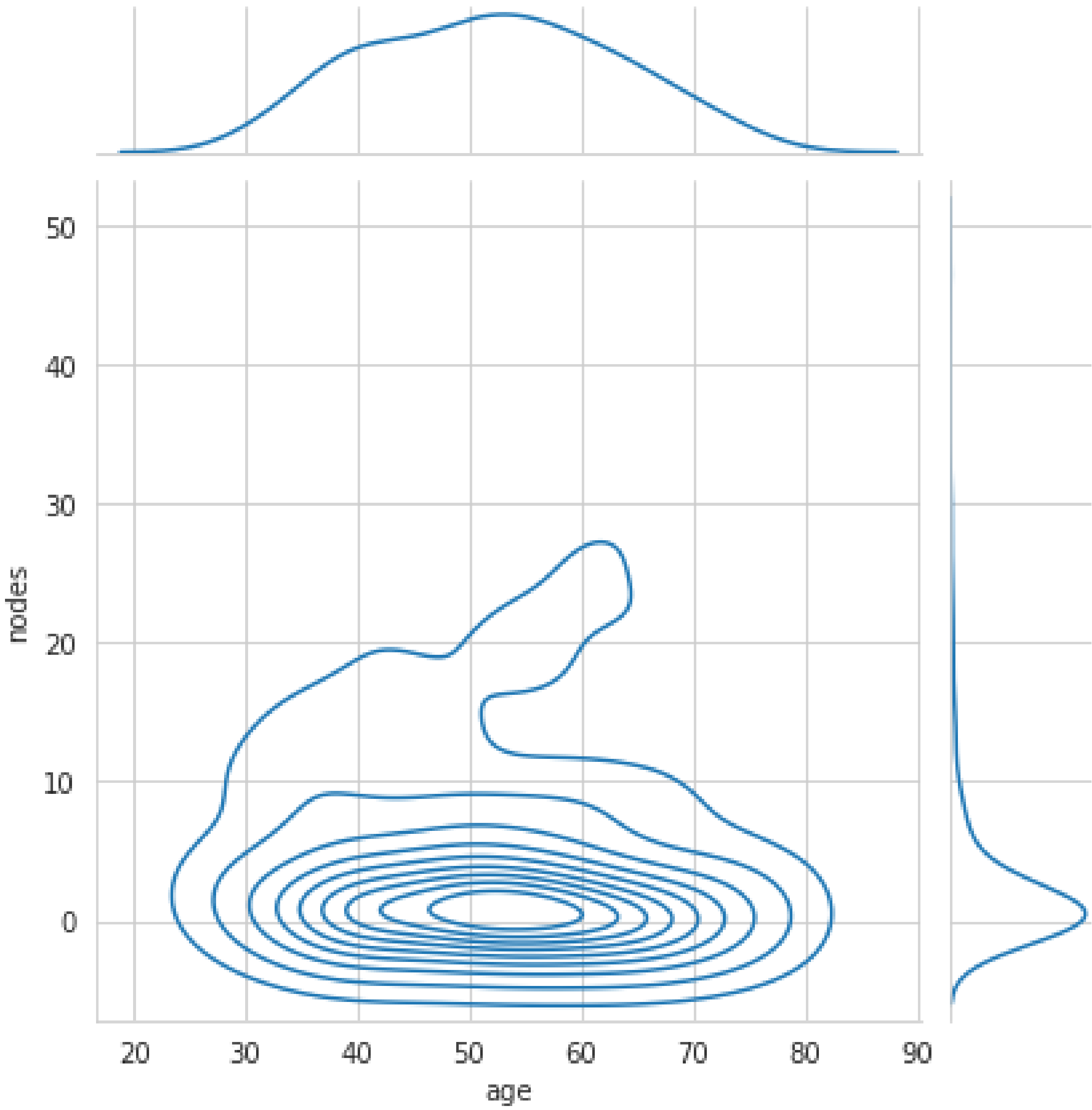
[83]:
```
# A violin plot combines the benefits of the previous two plots
#and simplifies them

# Denser regions of the data are fatter, and sparser ones thinner
#in a violin plot

sns.violinplot(x="status", y="age", data=df, size=8)
plt.show()
sns.violinplot(x="status", y="nodes", data=df, size=8)
plt.show()
```

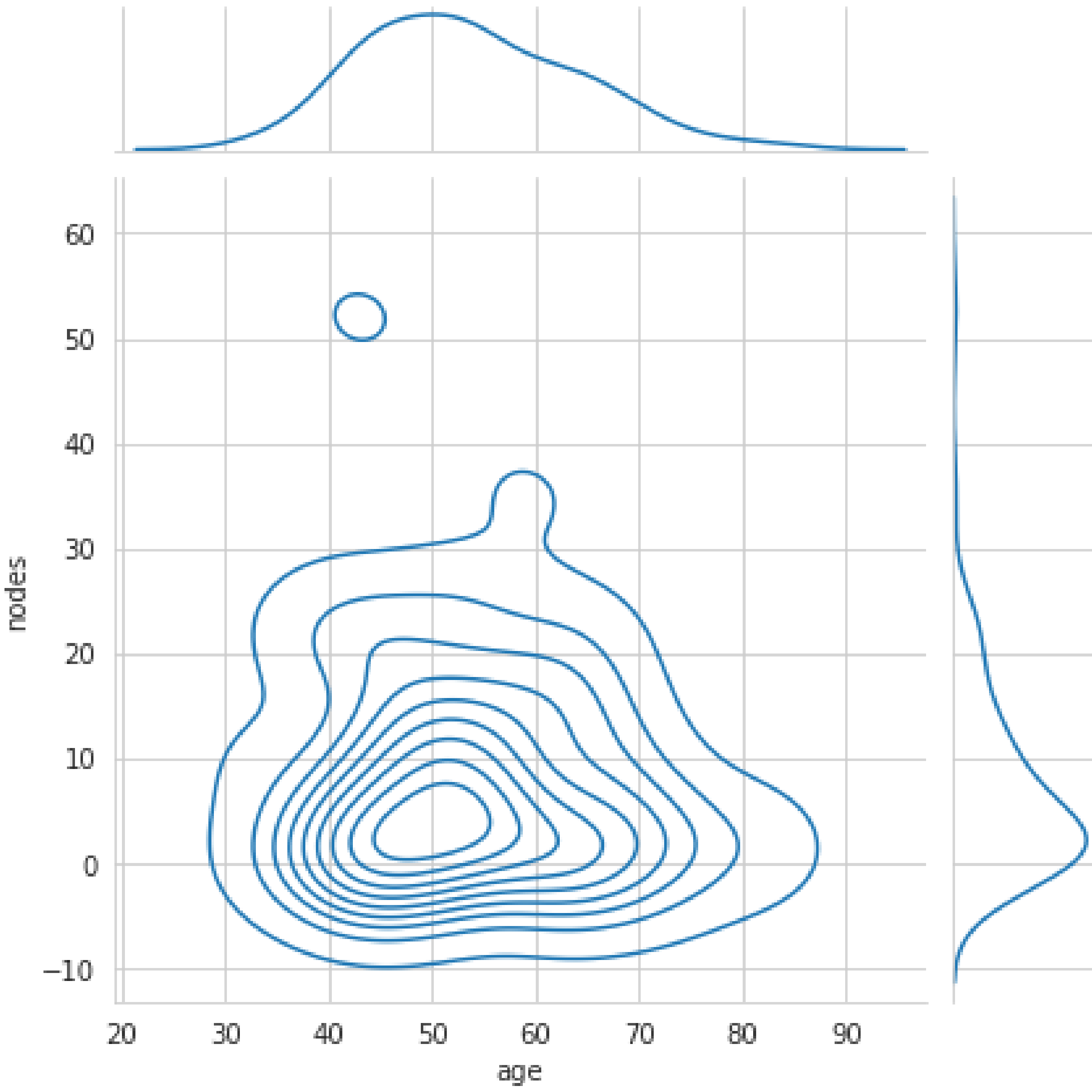

90
80
70
60
50
40
30
20
age
1
2
status


60
50
40
30
20
10
0
−10
nodes
1
2
status

20 Multivariate probability density, contour plot

[86]:
```
#2D Density plot, contors-plot
sns.jointplot(x="age", y="nodes", data=df_alive, kind="kde");
plt.show();
sns.jointplot(x="age", y="nodes", data=df_died, kind="kde");
plt.show();
```




nodes
age
60
50
40
30
20
10
0
−10
20
30
40
50
60
70
80
90

```
[ ]:
```