What are Random Variables in Stats and Probability?

A Random Variable is a set of possible values from a random experiment.
`e.g.` if you roll a dice, the outcome (1, 2, 3, 4, 5, or 6) is a random variable because it can be any one of those numbers.

Types of Random Variables:

There are two main types of random variables:

1. **Discrete Random Variables**: These can only take specific values, like whole numbers. An example is the roll of a dice, which can only be 1, 2, 3, 4, 5, or 6.
2. **Continuous Random Variables**: These can take any value within a range. For example, the time it takes for a computer to start could be 4.5 seconds, 4.51 seconds, or 4.511 seconds, and so on.

What are Probability Distributions?

A probability distribution is a list of all of the possible outcomes of a random variable along with their corresponding probability values.

Types of Probability Distribution:

There are three types of probability distribution which are used for different purposes and various types of the data generation process.

- Probability Mass Function (PMF)
- Probability Density Distribution (PDF)
- Cumulative Distribution Function(CDF)

Why are Probability Distributions important?

- Gives an idea about the shape/distribution of the data.
- if our data follows a famous distribution then we automatically know a lot about the data.

1. Probability Mass Function (PMF)

- PMF stands for Probability Mass Function.

- It is a mathematical function that describes the probability distribution of a discrete random variable.
- The PMF of a discrete random variable assigns a probability to each possible value of the random variable.

The probabilities assigned by the PMF must satisfy two conditions:

a.The probability assigned to each value must be non-negative (i.e., greaterthan or equal to zero).
b. The sum of the probabilities assigned to all possible values must equal 1.

In [49]:
```
import pandas as pd
import numpy as np
import random
```

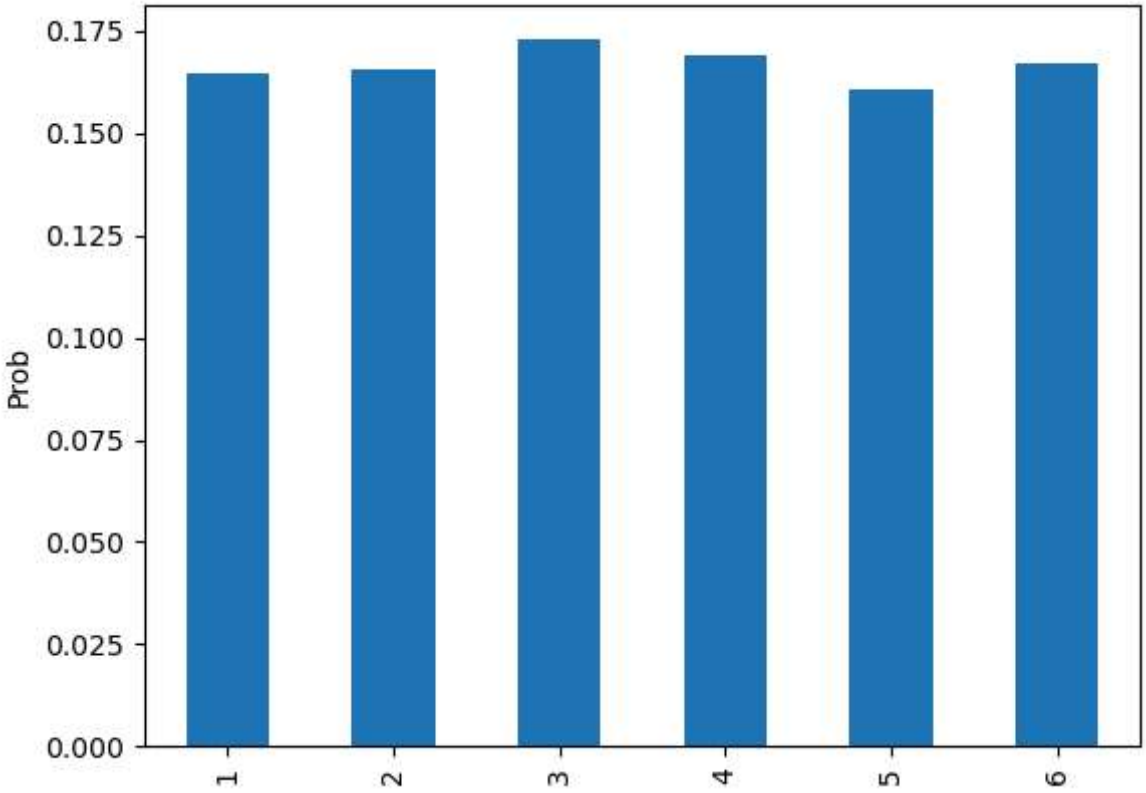
In [50]:
```
# Rolling a dice 10000 times
l = []
for i in range(10000):
    l.append(random.randint(1,6))
```

In [53]:
```
# Calculate the probability
x = (pd.Series(l).value_counts() / pd.Series(l).value_counts().sum()).sort_index
x
```

Out[53]:
```
1    0.1648
2    0.1656
3    0.1727
4    0.1690
5    0.1607
6    0.1672
dtype: float64
```

In [54]:
```
x.plot(kind='bar', ylabel='Prob')
```

Out[54]:
```
<AxesSubplot:ylabel='Prob'>
```



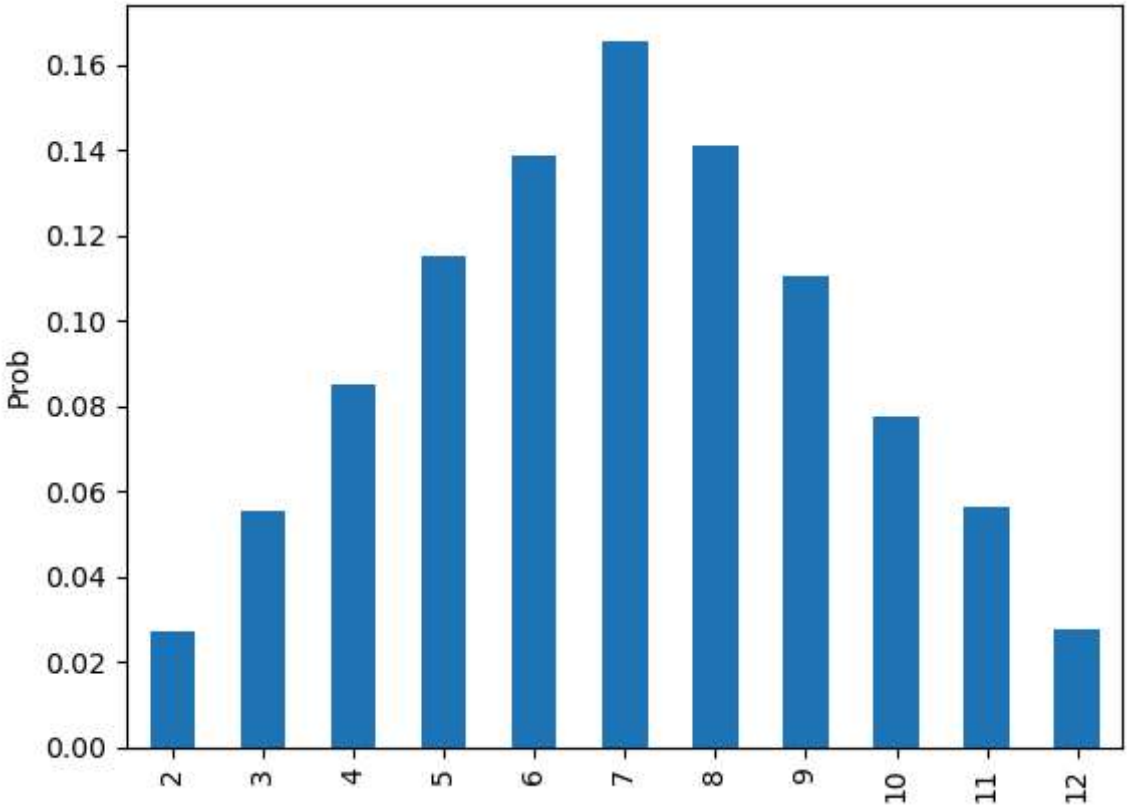
In [55]:
```
# rolling 2 dice simultaneously
L = []
for i in range(10000):
    a = random.randint(1,6)
    b = random.randint(1,6)
    L.append(a + b)
```

In [56]:
```
# Calculate the probability
y = (pd.Series(L).value_counts() / pd.Series(L).value_counts().sum()).sort_index
y
```

Out[56]:
```
2     0.0272
3     0.0554
4     0.0851
5     0.1151
6     0.1385
7     0.1658
8     0.1412
9     0.1103
10    0.0775
11    0.0565
12    0.0274
dtype: float64
```

In [57]:
```
# Probability Mass Function
y.plot(kind='bar', ylabel='Prob')
```

Out[57]:
```
<AxesSubplot:ylabel='Prob'>
```



Cumulative Distribution Function (CDF) of PMF

The cumulative distribution function (CDF) F(x) describes the probability that a random variable X with a given probability distribution will be found at a value less than or equal to x

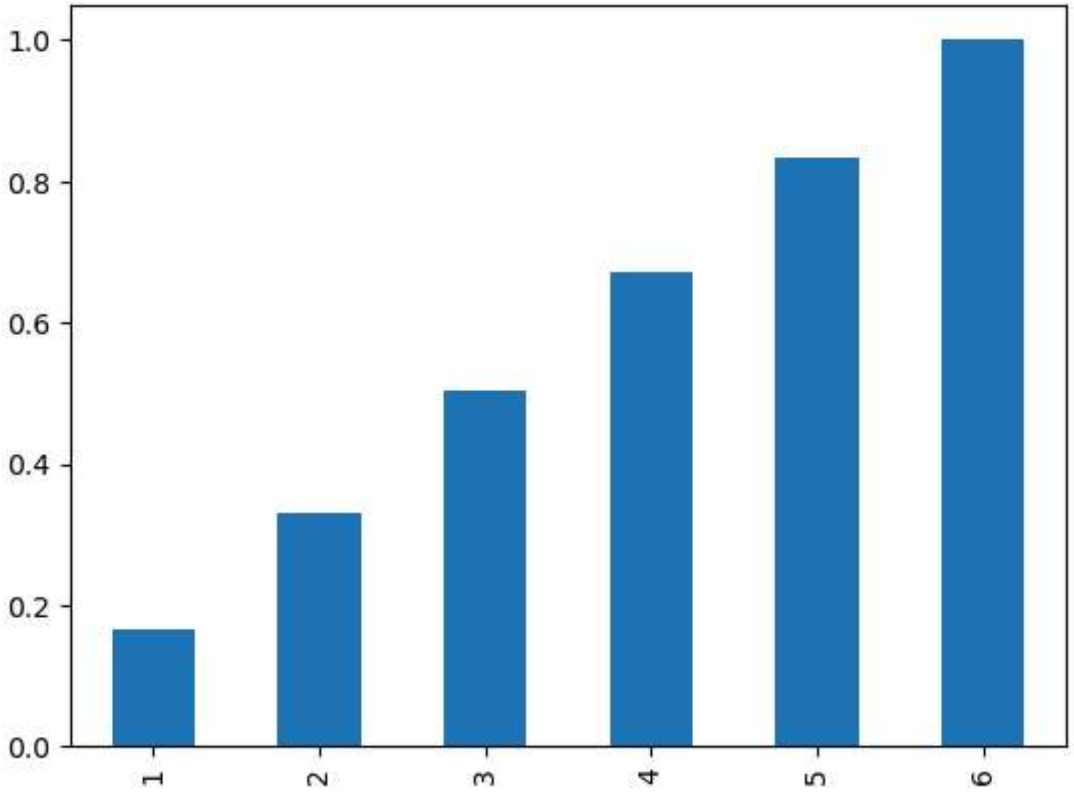
F(x) = P(X <= x)

In [60]:
```
np.cumsum(x) # sum = 1
```

Out[60]:
```
1    0.1648
2    0.3304
3    0.5031
4    0.6721
5    0.8328
6    1.0000
dtype: float64
```

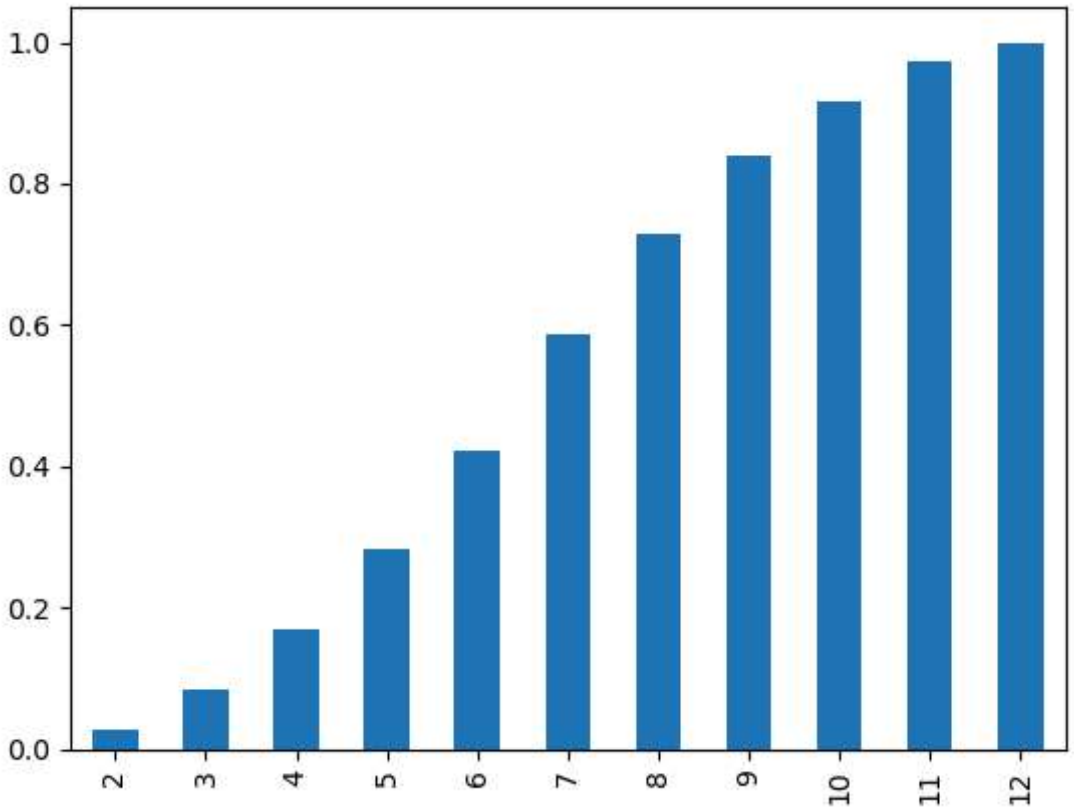
In [62]:
```
np.cumsum(x).plot(kind='bar')
```

Out[62]:
```
<AxesSubplot:>
```



In [63]:
```
np.cumsum(y).plot(kind='bar')
```

Out[63]: <AxesSubplot:>



What is the difference between PMF AND CDF

- PMF : Gives the probability of the particular point (X)
- CDF : Gives the probability of the All the points up to (X)

2. Probability Density Function (PDF)

- PDF stands for Probability Density Function.
- It is a mathematical function that describes the probability distribution of a continuous random variable.

Why Probability Density and why not Probability?

- `Infinite Values on the X-axis` : Continuous probability distributions have an infinite number of possible values on the x-axis.
- `Probability Close to Zero` : The probability of any single exact value is close to zero because there are infinitely many possibilities.
- `Probability Density` : Instead, we use Probability Density to describe how likely it is to find a value within a small range, making it meaningful to work with continuous distributions.

In a continuous probability distribution, the area under the graph of the probability density function (PDF) represents probabilities. Here's a more detailed explanation:

What the Area Represents

`Total Area Equals 1` : The total area under the entire curve of the PDF is equal to 1. This represents the fact that the probability of all possible outcomes combined is 100%.

How to Calculate Probability

`Probability of a Range` : To find the probability of a continuous random variable falling within a specific range, you calculate the area under the curve between those two points on the x-axis.

For example, to find the probability that a variable is between a and b, you integrate the PDF from a to b.

Example:

`Define the PDF` : Suppose f(x) is the probability density function.

`Set the Range` : To find the probability that X is between a and b, you calculate the integral of f(x) from a to b:

$$P(a \leq X \leq b) = \int_a^b f(x)\,dx$$

How is graph calculated?

- Parametric Density Estimation
- Non-Parametric Density Estimation

Parametric Density Estimation

Step 1: Choose a Parametric Model

`Identify the Distribution Type` : Select a probability distribution that you believe fits your data well, such as the Normal (Gaussian) distribution, Exponential distribution, or Poisson distribution.

Step 2: Collect and Prepare Data

- `Gather Data` : Collect the sample data you want to analyze.
- `Preprocess Data` : Clean the data to handle missing values or outliers, ensuring it's ready for analysis.

Step 3: Estimate Parameters

`Parameter Estimation` : Use statistical methods to estimate the parameters of the chosen distribution. For example:

For a Normal distribution, estimate the mean (μ) and standard deviation (σ).
For an Exponential distribution, estimate the rate parameter (λ).

Step 4: Fit the Model

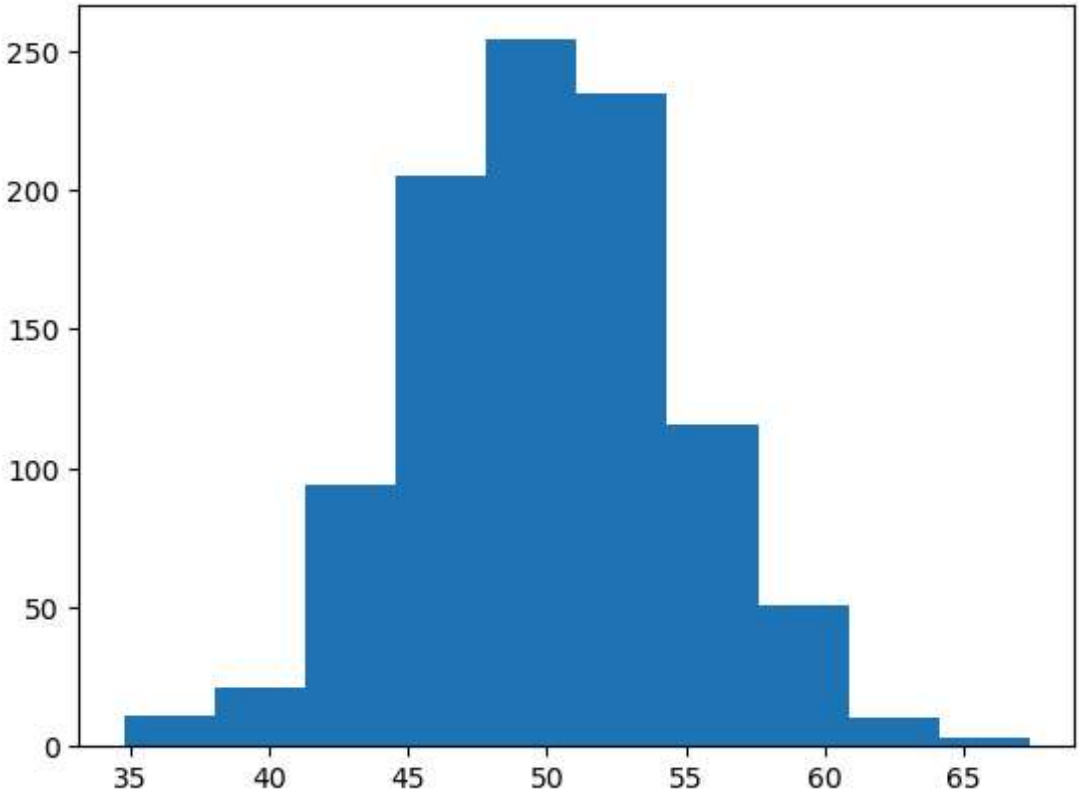
`Apply Parameters` : Fit the chosen distribution to your data using the estimated parameters. This creates a probability density function (PDF) that represents your data.

In [66]:
```
import matplotlib.pyplot as plt
from numpy.random import normal

# assuming we don't know the distribution of data
sample = normal(loc=50, scale=5, size=1000)
```

In [68]:
```
# plot histogram to understand the distribution of data
plt.hist(sample, bins=10) # looks like normal dist.
```

Out[68]:
```
(array([ 11.,  21.,  94., 205., 254., 235., 116.,  51.,  10.,   3.]),
 array([34.80804736, 38.06225497, 41.31646258, 44.57067019, 47.8248778 ,
        51.07908541, 54.33329302, 57.58750063, 60.84170824, 64.09591585,
        67.35012346]),
 <BarContainer object of 10 artists>)
```



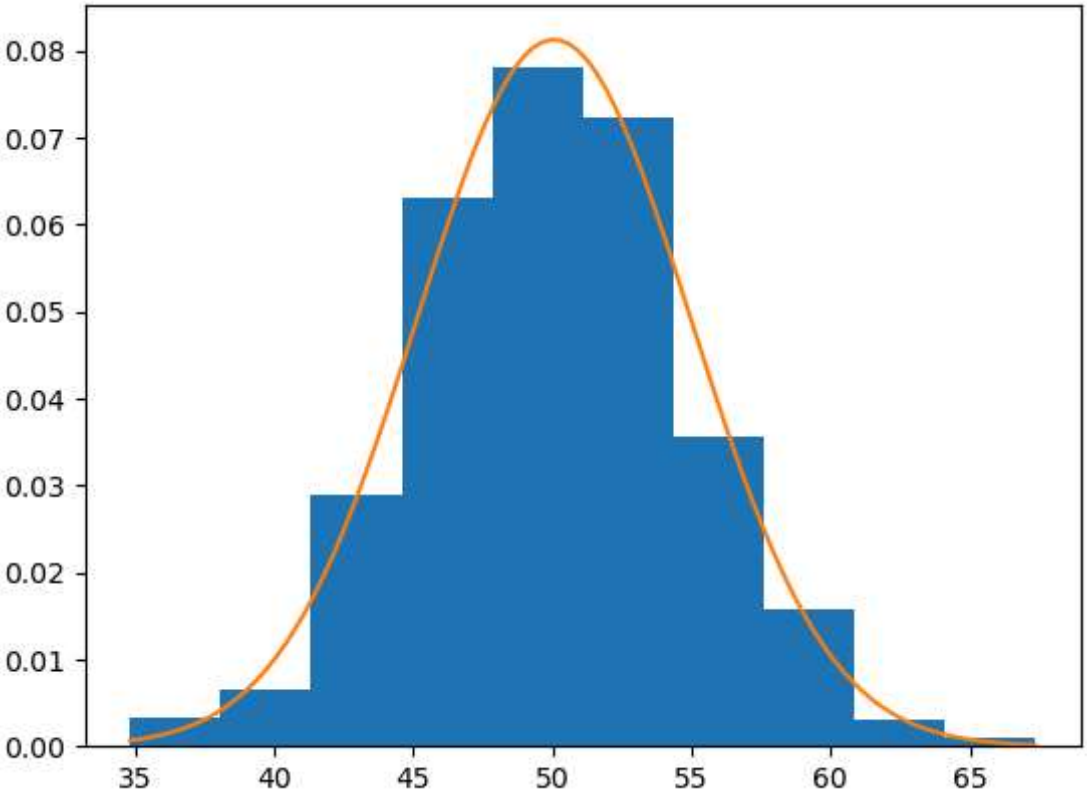
In [70]:
```
# calculate the sample mean and sample standard deviation
sample_mean = sample.mean()
sample_std = sample.std()
```

In [71]:
```
# fit the disribution with the above parameter
from scipy.stats import norm
dist = norm(sample_mean, sample_std)
```

In [81]:
```
values = np.linspace(sample.min(), sample.max(), 100)
probabilities = [dist.pdf(value) for value in values]
```

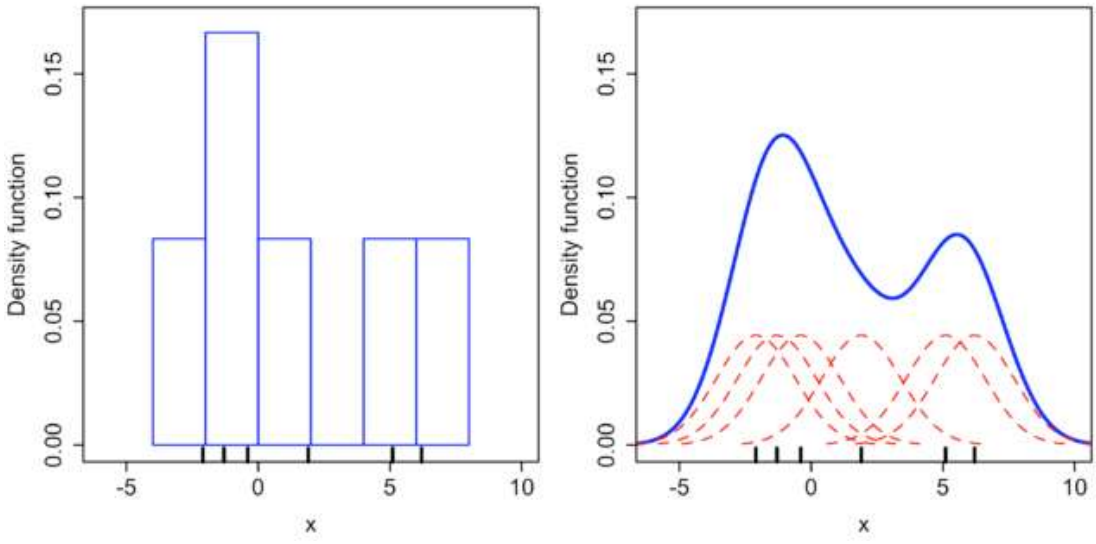
In [84]:
```
# plot the histogram and pdf
#import seaborn as sns
#sns.distplot(values)
plt.hist(sample, bins=10, density=True)
plt.plot(values, probabilities)
```

Out[84]:
```
[<matplotlib.lines.Line2D at 0x19e079ec760>]
```



Non-parametric Density Estimation

KDE



From histogram, compute the PDF by smoothing (smooth curve). It is done using kernel density estimation.

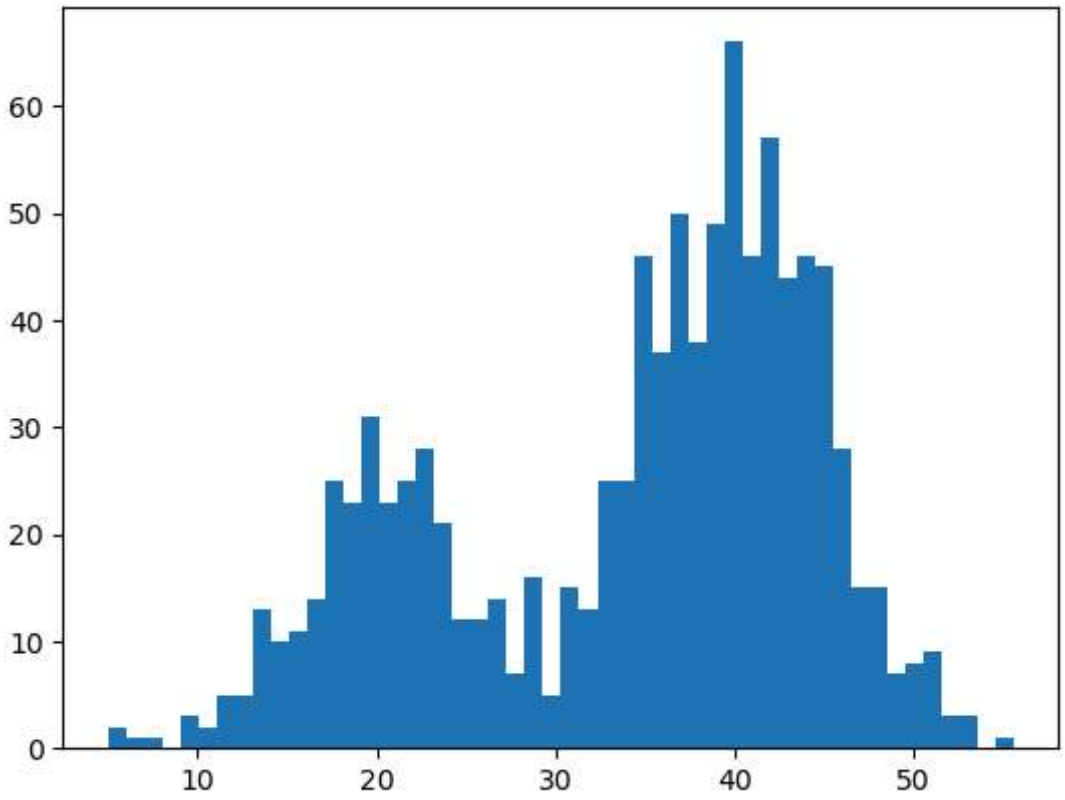
Procedure:

1. Draw a gaussian kernel for each point/observation centred at that point
   i.e. Draw gaussian kernel for 5th point in which centre/peak is at 5th point.
2. At any point x, multiple kernels available, addition of these kernel & draw the PDF (It gives the height of PDF)
3. Variance in KDE is also called bandwidth (hyper-parameter)

In [87]:
```
# generate a sample
sample1 = normal(loc=20, scale=5, size=300)
sample2 = normal(loc=40, scale=5, size=700)

sample = np.hstack((sample1, sample2))
```

In [92]:
```
# plot histogram -> not follows any particular dist like normal, log-normal or a
plt.hist(sample, bins=50)
plt.show()
```



In [93]:
```
from sklearn.neighbors import KernelDensity

model = KernelDensity(bandwidth=3, kernel='gaussian')

# convert data to 2d D array -> because ML algo works mostly with 2D data
sample = sample.reshape((len(sample), 1)) # shape = (1000, 1)
model.fit(sample)
```

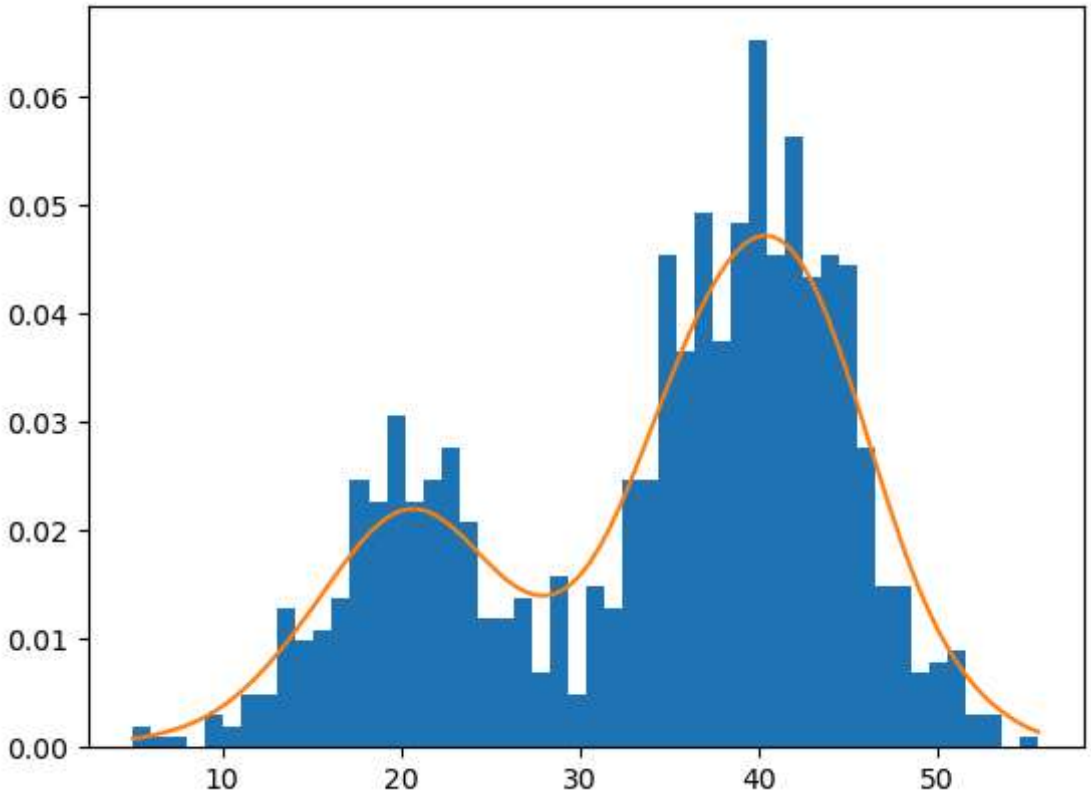
Out[93]:
```
KernelDensity(bandwidth=3)
```

In [94]:
```
values = np.linspace(sample.min(), sample.max(), 100)
values = values.reshape((len(values), 1))
```

In [95]:
```
probabilities = model.score_samples(values)
probabilities = np.exp(probabilities)
```

`score_samples(values)` returns the log-density estimate of the input samples values. This is because the `score_samples()` method of the KernelDensity class returns the logarithm of the probability density estimate rather than the actual probability density estimate.

In [96]:
```
plt.hist(sample, bins=50, density=True)
plt.plot(values, probabilities)
plt.show()
```



How to use PDF?

In [2]:
```
import seaborn as sns
import matplotlib.pyplot as plt

df = sns.load_dataset('iris')
df.sample(2)
```

```
C:\Users\SVF\anaconda3\lib\site-packages\scipy\__init__.py:155: UserWarning: A Nu
mPy version >=1.18.5 and <1.26.0 is required for this version of SciPy (detected
version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

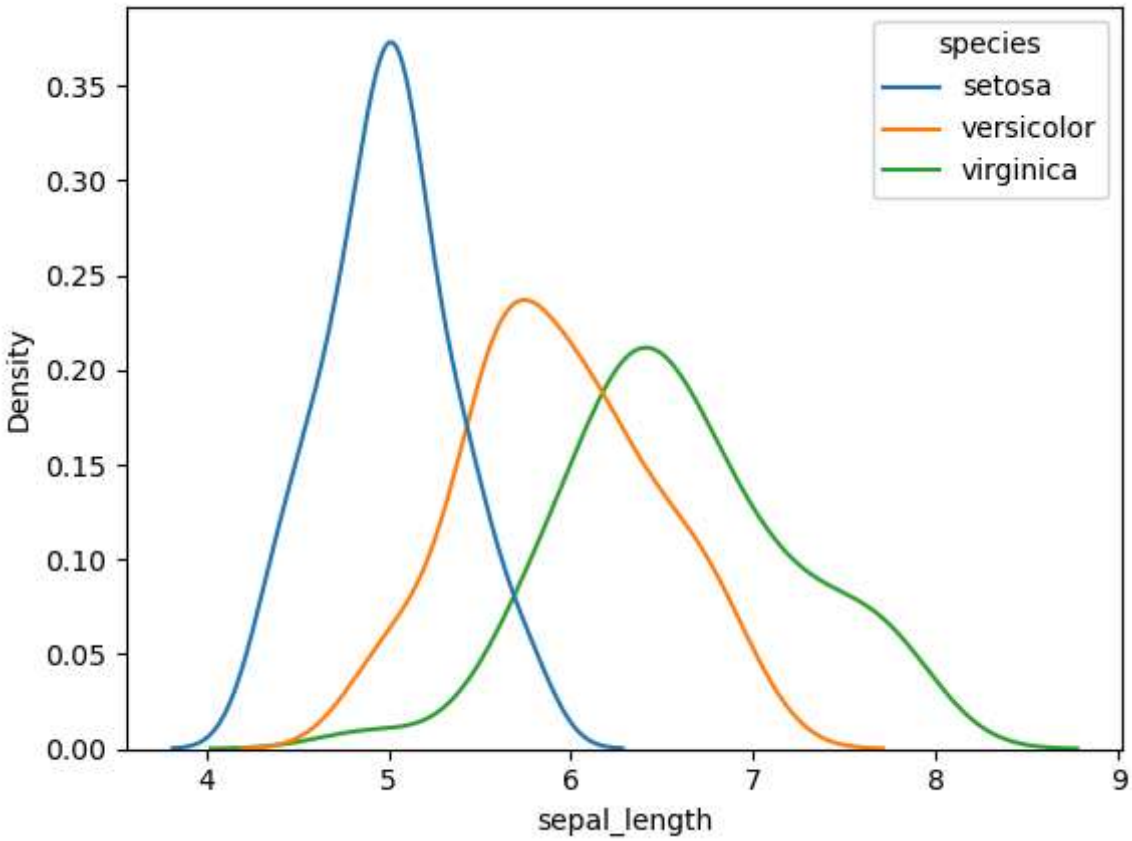
Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
81	5.5	2.4	3.7	1.0	versicolor
72	6.3	2.5	4.9	1.5	versicolor

What are the most important features for analysis?

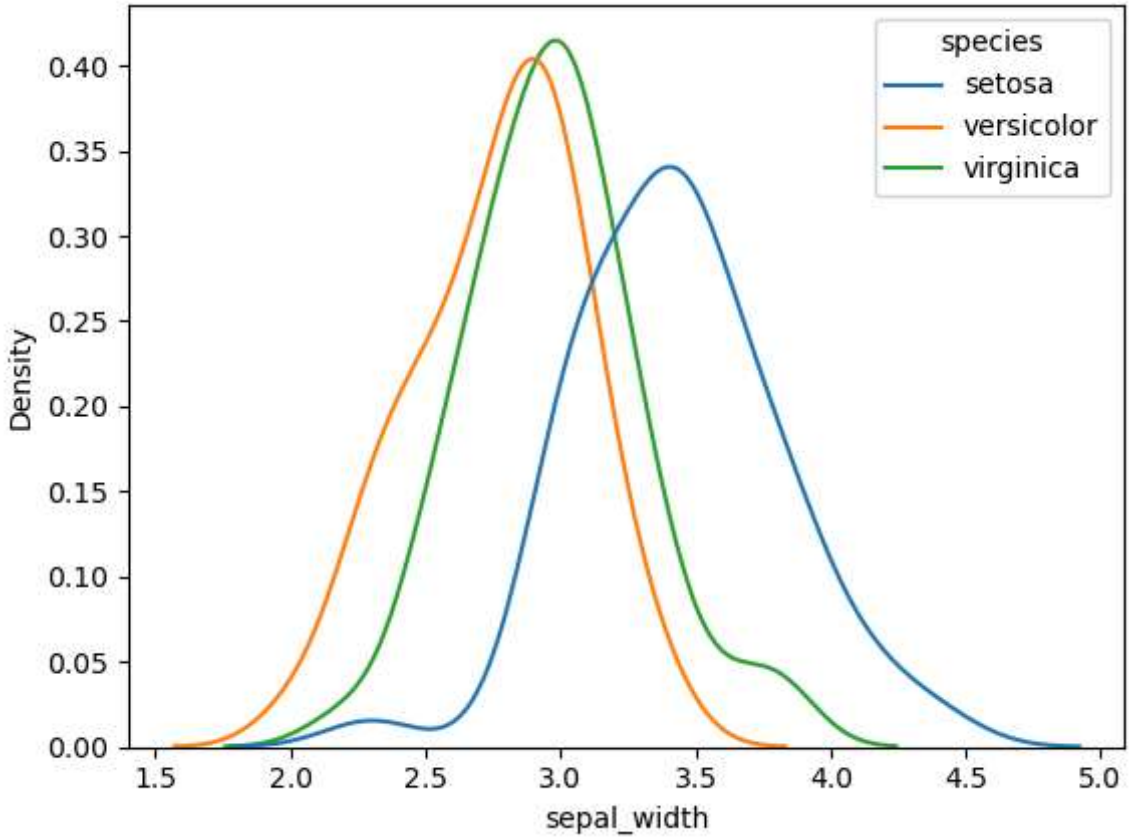
In [6]:
```
sns.kdeplot(df['sepal_length'], hue=df['species'])
```

Out[6]:
```
<AxesSubplot:xlabel='sepal_length', ylabel='Density'>
```



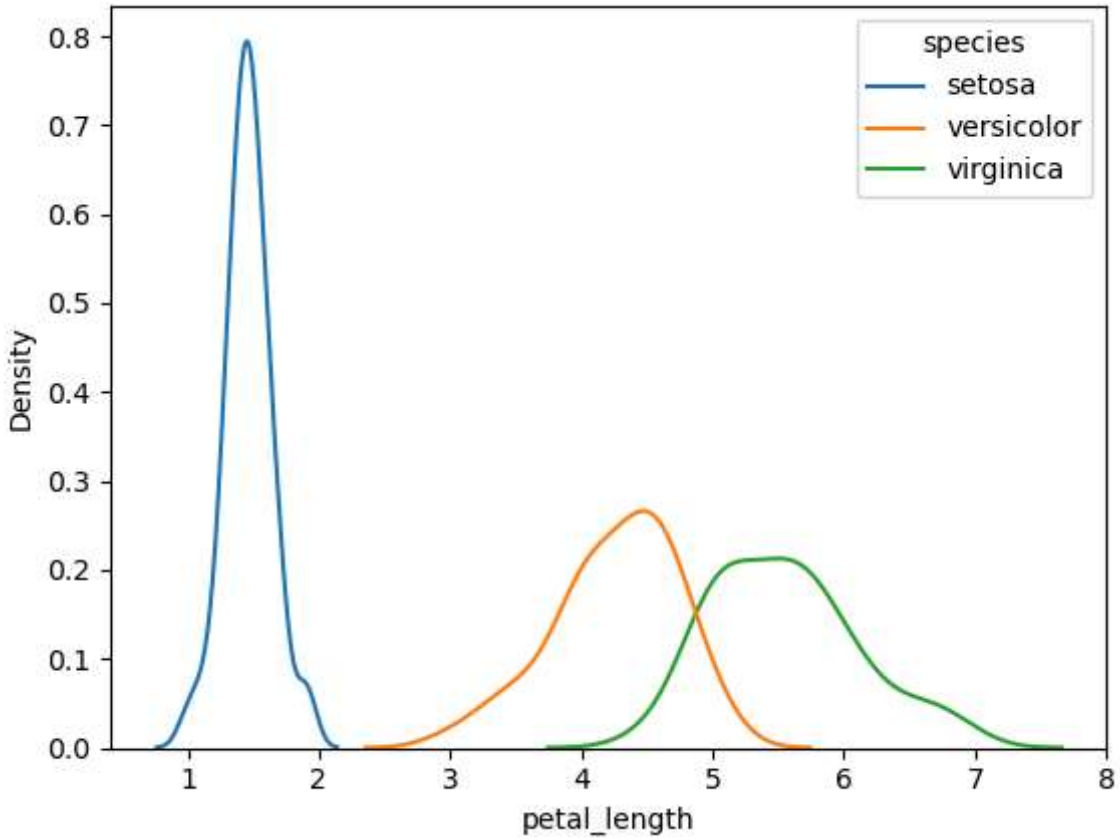
In [7]:
```
sns.kdeplot(df['sepal_width'], hue=df['species'])
```

Out[7]:
```
<AxesSubplot:xlabel='sepal_width', ylabel='Density'>
```



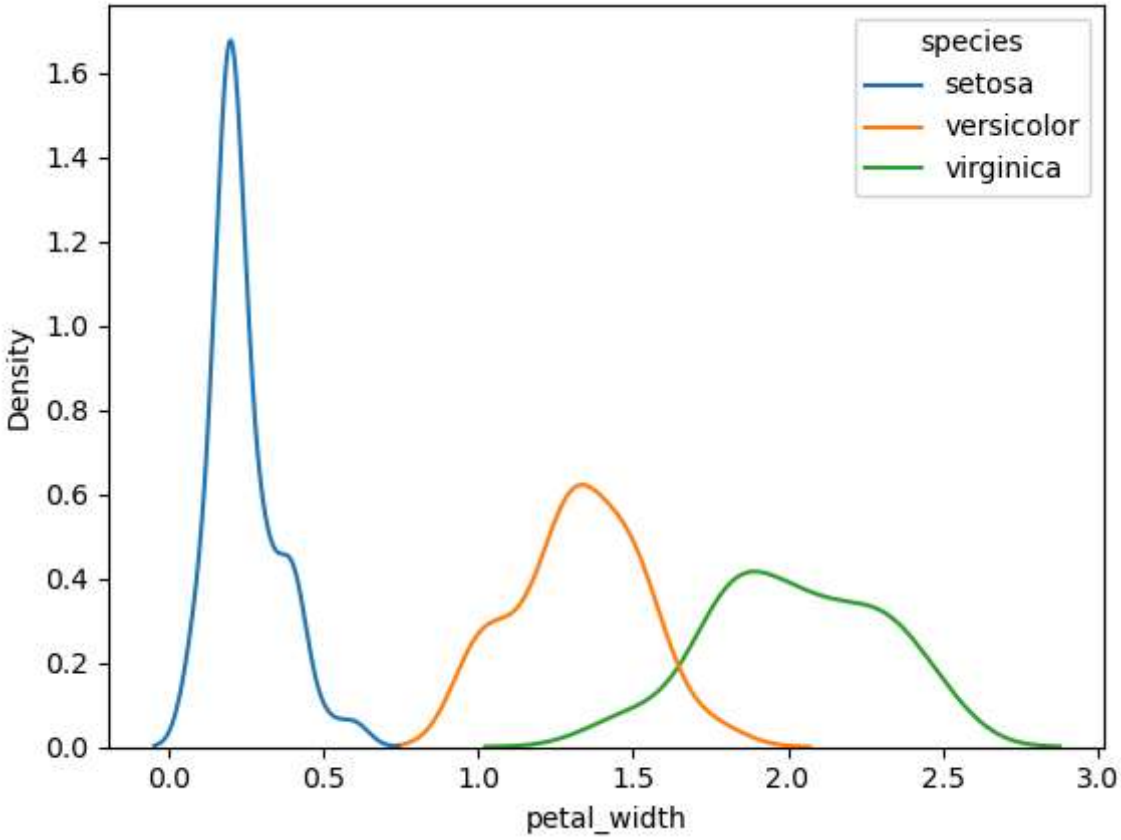
In [8]:
```
sns.kdeplot(df['petal_length'], hue=df['species'])
```

Out[8]:
```
<AxesSubplot:xlabel='petal_length', ylabel='Density'>
```



In [9]:
```
sns.kdeplot(df['petal_width'], hue=df['species'])
```

Out[9]:
```
<AxesSubplot:xlabel='petal_width', ylabel='Density'>
```



Observations:

Petal Length and Petal Width:

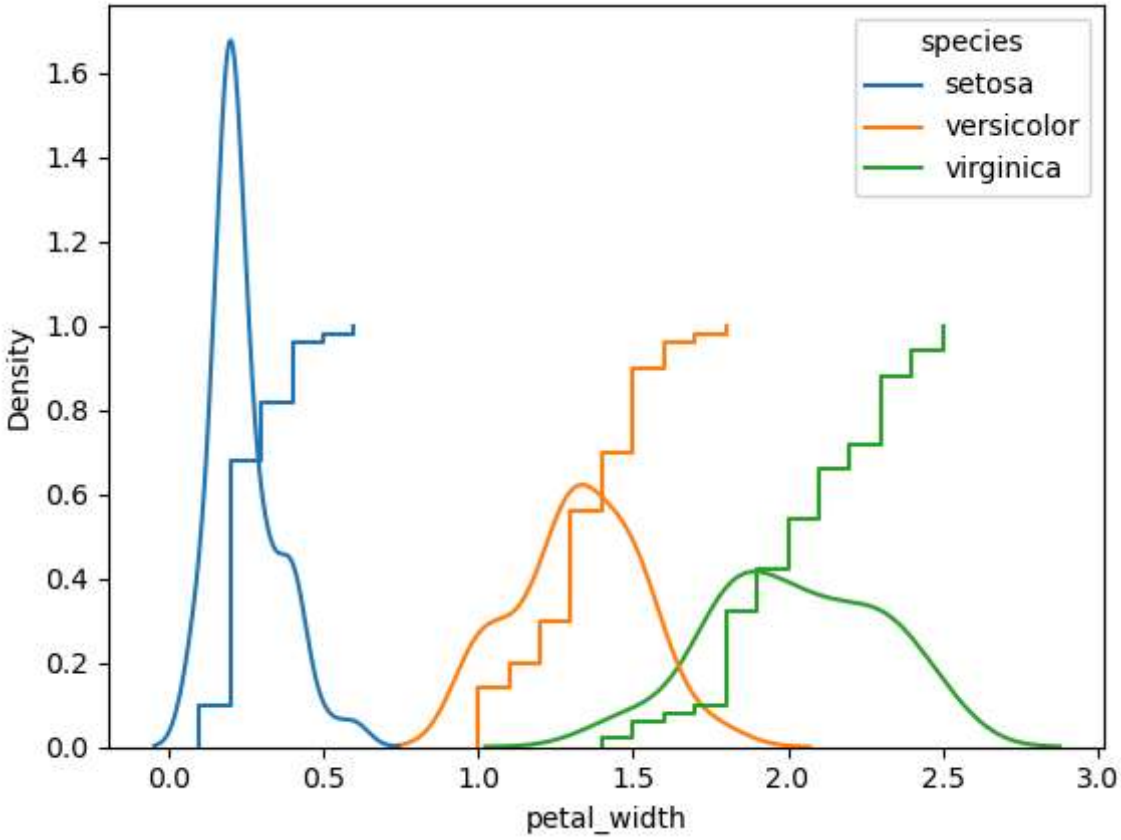
- These features are the most significant for distinguishing between species.
- The overlap between species is minimal when using these features, making them highly effective for classification.

Sepal Length and Sepal Width:

- These features are less effective for differentiating due to considerable overlap

In [16]:
```
sns.kdeplot(df['petal_width'], hue=df['species'])
sns.ecdfplot(data = df, x='petal_width', hue='species')
```

Out[16]:
```
<AxesSubplot:xlabel='petal_width', ylabel='Density'>
```



Observations based on CDF:

- if petal_width < 0.7 cm, the flower is setosa.
- 95% confidence that if 0.7 cm ≤ petal_width < 1.75 cm, the flower is versicolor.
- 90% confidence that if petal_width ≥ 1.75 cm, the flower is virginica.

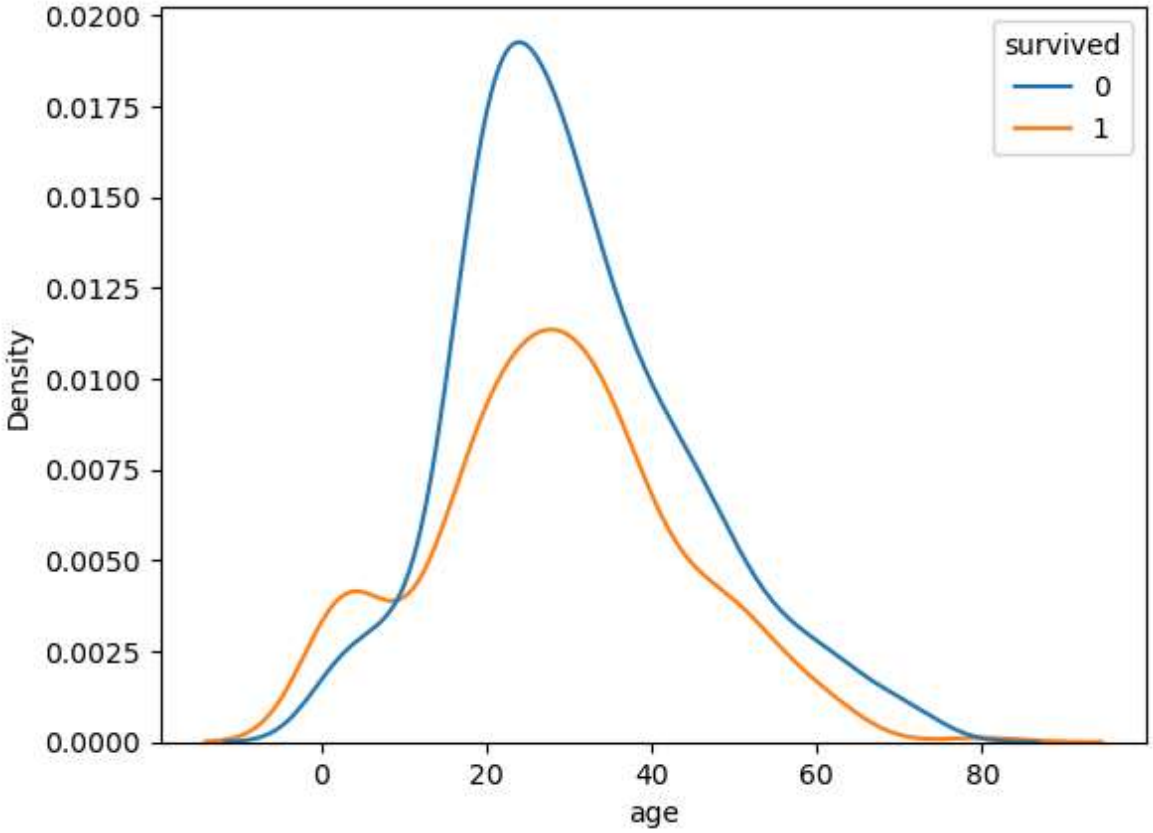
In [11]:
```
titanic = sns.load_dataset('titanic')
titanic.sample(3)
```

Out[11]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	ad
305	1	1	male	0.92	1	2	151.5500	S	First	child	
163	0	3	male	17.00	0	0	8.6625	S	Third	man	
119	0	3	female	2.00	4	2	31.2750	S	Third	child	

In [12]:
```
sns.kdeplot(data=titanic, x='age', hue='survived')
```

Out[12]:
```
<AxesSubplot:xlabel='age', ylabel='Density'>
```



Observations:

- The age feature plays a critical role in determining survival. Different age groups show varying survival rates, with children (age < 8) having a higher survival rate compared to adults.