Experiment No. 7

In [4]:
```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
```

Out[4]: True

In [10]:
```
import nltk
para = 'Rajgad (literal meaning Ruling Fort) is a hill fort situated in the Pune di
```

In [11]:
```
print(para)
```

```
Rajgad (literal meaning Ruling Fort) is a hill fort situated in the Pune district of
Maharashtra, India. Formerly known as Murumdev, the fort was the capital of the Mara
tha Empire under the rule of Chatrapati Shivaji Maharaj for almost 26 years, after w
hich the capital was moved to the Raigad Fort.[1] Treasures discovered from an adjac
ent fort called Torna were used to completely build and fortify the Rajgad Fort.
```

In [12]:
```
para.split ( )
```

Out[12]:
```
['Rajgad',
 '(literal',
 'meaning',
 'Ruling',
 'Fort)',
 'is',
 'a',
 'hill',
 'fort',
 'situated',
 'in',
 'the',
 'Pune',
 'district',
 'of',
 'Maharashtra,',
 'India.',
 'Formerly',
 'known',
 'as',
 'Murumdev,',
 'the',
 'fort',
 'was',
 'the',
 'capital',
 'of',
 'the',
 'Maratha',
 'Empire',
 'under',
 'the',
 'rule',
 'of',
 'Chatrapati',
 'Shivaji',
 'Maharaj',
 'for',
 'almost',
 '26',
 'years,',
 'after',
 'which',
 'the',
 'capital',
 'was',
 'moved',
 'to',
 'the',
 'Raigad',
 'Fort.[1]',
 'Treasures',
 'discovered',
 'from',
 'an',
 'adjacent',
```

```
 'fort',
 'called',
 'Torna',
 'were',
 'used',
 'to',
 'completely',
 'build',
 'and',
 'fortify',
 'the',
 'Rajgad',
 'Fort.']
```

In [13]:
```
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
```

In [17]:
```
import nltk
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt_tab.zip.
```

Out[17]: True

In [20]:
```
from nltk.tokenize import sent_tokenize
sent = sent_tokenize(para)
print(sentences)
```

```
['Rajgad (literal meaning Ruling Fort) is a hill fort situated in the Pune district
of Maharashtra, India.', 'Formerly known as Murumdev, the fort was the capital of th
e Maratha Empire under the rule of Chatrapati Shivaji Maharaj for almost 26 years, a
fter which the capital was moved to the Raigad Fort.', '[1] Treasures discovered fro
m an adjacent fort called Torna were used to completely build and fortify the Rajgad
Fort.']
```

In [21]:
```
sent[2]
```

Out[21]:
```
'[1] Treasures discovered from an adjacent fort called Torna were used to complete
ly build and fortify the Rajgad Fort.'
```

In [22]:
```
words=word_tokenize(para)
```

In [23]:
```
words
```

Out[23]:
```
['Rajgad',
 '(',
 'literal',
 'meaning',
 'Ruling',
 'Fort',
 ')',
 'is',
 'a',
 'hill',
 'fort',
 'situated',
 'in',
 'the',
 'Pune',
 'district',
 'of',
 'Maharashtra',
 ',',
 'India',
 '.',
 'Formerly',
 'known',
 'as',
 'Murumdev',
 ',',
 'the',
 'fort',
 'was',
 'the',
 'capital',
 'of',
 'the',
 'Maratha',
 'Empire',
 'under',
 'the',
 'rule',
 'of',
 'Chatrapati',
 'Shivaji',
 'Maharaj',
 'for',
 'almost',
 '26',
 'years',
 ',',
 'after',
 'which',
 'the',
 'capital',
 'was',
 'moved',
 'to',
 'the',
 'Raigad',
```

```
 'Fort',
 '.',
 '[',
 'l',
 ']',
 'Treasures',
 'discovered',
 'from',
 'an',
 'adjacent',
 'fort',
 'called',
 'Torna',
 'were',
 'used',
 'to',
 'completely',
 'build',
 'and',
 'fortify',
 'the',
 'Rajgad',
 'Fort',
 '.']
```

In [24]:
```
from nltk.corpus import stopwords
```

In [26]:
```
swords=stopwords.words('english')
```

In [27]:
```
swords
```

Out[27]:
```
['a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'ain',
 'all',
 'am',
 'an',
 'and',
 'any',
 'are',
 'aren',
 "aren't",
 'as',
 'at',
 'be',
 'because',
 'been',
 'before',
 'being',
 'below',
 'between',
 'both',
 'but',
 'by',
 'can',
 'couldn',
 "couldn't",
 'd',
 'did',
 'didn',
 "didn't",
 'do',
 'does',
 'doesn',
 "doesn't",
 'doing',
 'don',
 "don't",
 'down',
 'during',
 'each',
 'few',
 'for',
 'from',
 'further',
 'had',
 'hadn',
 "hadn't",
 'has',
 'hasn',
 "hasn't",
 'have',
 'haven',
```

```
"haven't",
'having',
'he',
"he'd",
"he'll",
'her',
'here',
'hers',
'herself',
"he's",
'him',
'himself',
'his',
'how',
'i',
"i'd",
'if',
"i'll",
"i'm",
'in',
'into',
'is',
'isn',
"isn't",
'it',
"it'd",
"it'll",
"it's",
'its',
'itself',
"i've",
'just',
'll',
'm',
'ma',
'me',
'mightn',
"mightn't",
'more',
'most',
'mustn',
"mustn't",
'my',
'myself',
'needn',
"needn't",
'no',
'nor',
'not',
'now',
'o',
'of',
'off',
'on',
'once',
'only',
```

```
'or',
'other',
'our',
'ours',
'ourselves',
'out',
'over',
'own',
're',
's',
'same',
'shan',
"shan't",
'she',
"she'd",
"she'll",
"she's",
'should',
'shouldn',
"shouldn't",
"should've",
'so',
'some',
'such',
't',
'than',
'that',
"that'll",
'the',
'their',
'theirs',
'them',
'themselves',
'then',
'there',
'these',
'they',
"they'd",
"they'll",
"they're",
"they've",
'this',
'those',
'through',
'to',
'too',
'under',
'until',
'up',
've',
'very',
'was',
'wasn',
"wasn't",
'we',
"we'd",
```

```
"we'll",
"we're",
'were',
'weren',
"weren't",
"we've",
'what',
'when',
'where',
'which',
'while',
'who',
'whom',
'why',
'will',
'with',
'won',
"won't",
'wouldn',
"wouldn't",
'y',
'you',
"you'd",
"you'll",
'your',
"you're",
'yours',
'yourself',
'yourselves',
"you've"]
```

In [28]:
```
x=[word for word in words if word not in swords]
```

In [29]:
```
x
```

Out[29]:
```
['Rajgad',
 '(',
 'literal',
 'meaning',
 'Ruling',
 'Fort',
 ')',
 'hill',
 'fort',
 'situated',
 'Pune',
 'district',
 'Maharashtra',
 ',',
 'India',
 '.',
 'Formerly',
 'known',
 'Murumdev',
 ',',
 'fort',
 'capital',
 'Maratha',
 'Empire',
 'rule',
 'Chatrapati',
 'Shivaji',
 'Maharaj',
 'almost',
 '26',
 'years',
 ',',
 'capital',
 'moved',
 'Raigad',
 'Fort',
 '.',
 '[',
 'l',
 ']',
 'Treasures',
 'discovered',
 'adjacent',
 'fort',
 'called',
 'Torna',
 'used',
 'completely',
 'build',
 'fortify',
 'Rajgad',
 'Fort',
 '.']
```

In [33]:
```
x=[word for word in words if word.lower() not in swords]
```

In [34]:
```
x
```

Out[34]:
```
['Rajgad',
 '(',
 'literal',
 'meaning',
 'Ruling',
 'Fort',
 ')',
 'hill',
 'fort',
 'situated',
 'Pune',
 'district',
 'Maharashtra',
 ',',
 'India',
 '.',
 'Formerly',
 'known',
 'Murumdev',
 ',',
 'fort',
 'capital',
 'Maratha',
 'Empire',
 'rule',
 'Chatrapati',
 'Shivaji',
 'Maharaj',
 'almost',
 '26',
 'years',
 ',',
 'capital',
 'moved',
 'Raigad',
 'Fort',
 '.',
 '[',
 'l',
 ']',
 'Treasures',
 'discovered',
 'adjacent',
 'fort',
 'called',
 'Torna',
 'used',
 'completely',
 'build',
 'fortify',
 'Rajgad',
 'Fort',
 '.']
```

In [35]:
```
from nltk.stem import PorterStemmer
```

In [36]:
```
ps=PorterStemmer()
```

In [38]:
```
ps.  stem('working')
```

Out[38]:
```
'work'
```

In [39]:
```
y=[ps.stem(word) for word in x]
```

In [40]:
```
y
```

Out[40]:
```
['rajgad',
 '(',
 'liter',
 'mean',
 'rule',
 'fort',
 ')',
 'hill',
 'fort',
 'situat',
 'pune',
 'district',
 'maharashtra',
 ',',
 'india',
 '.',
 'formerli',
 'known',
 'murumdev',
 ',',
 'fort',
 'capit',
 'maratha',
 'empir',
 'rule',
 'chatrapati',
 'shivaji',
 'maharaj',
 'almost',
 '26',
 'year',
 ',',
 'capit',
 'move',
 'raigad',
 'fort',
 '.',
 '[',
 'l',
 ']',
 'treasur',
 'discov',
 'adjac',
 'fort',
 'call',
 'torna',
 'use',
 'complet',
 'build',
 'fortifi',
 'rajgad',
 'fort',
 '.']
```

In [41]:
```
from nltk.stem import WordNetLemmatizer
```

In [43]:
```
wnl = WordNetLemmatizer()
```

In [45]:
```
print(wnl.lemmatize('working',  pos='v'))
#a-adjective
#n-noun
#r-adverb
```

```
work
```

In [46]:
```
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
```

Out[46]:
```
True
```

In [47]:
```
wnl.lemmatize('working',pos='v')
```

Out[47]:
```
'work'
```

In [48]:
```
print(ps.stem('went'))
print(wnl.lemmatize('went',pos='v'))
```

```
went
go
```

In [50]:
```
z = [wnl.lemmatize(word, pos='v') for word in x]
```

In [51]:
```
z
```

Out[51]:
```
['Rajgad',
 '(',
 'literal',
 'mean',
 'Ruling',
 'Fort',
 ')',
 'hill',
 'fort',
 'situate',
 'Pune',
 'district',
 'Maharashtra',
 ',',
 'India',
 '.',
 'Formerly',
 'know',
 'Murumdev',
 ',',
 'fort',
 'capital',
 'Maratha',
 'Empire',
 'rule',
 'Chatrapati',
 'Shivaji',
 'Maharaj',
 'almost',
 '26',
 'years',
 ',',
 'capital',
 'move',
 'Raigad',
 'Fort',
 '.',
 '[',
 'l',
 ']',
 'Treasures',
 'discover',
 'adjacent',
 'fort',
 'call',
 'Torna',
 'use',
 'completely',
 'build',
 'fortify',
 'Rajgad',
 'Fort',
 '.']
```

In [52]:
```
import string
```

In [53]:
```
string.punctuation
```

Out[53]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [54]:
```
t=[word for word in words if word not in string.punctuation]
```

In [55]:
```
t
```

Out[55]:
```
['Rajgad',
 'literal',
 'meaning',
 'Ruling',
 'Fort',
 'is',
 'a',
 'hill',
 'fort',
 'situated',
 'in',
 'the',
 'Pune',
 'district',
 'of',
 'Maharashtra',
 'India',
 'Formerly',
 'known',
 'as',
 'Murumdev',
 'the',
 'fort',
 'was',
 'the',
 'capital',
 'of',
 'the',
 'Maratha',
 'Empire',
 'under',
 'the',
 'rule',
 'of',
 'Chatrapati',
 'Shivaji',
 'Maharaj',
 'for',
 'almost',
 '26',
 'years',
 'after',
 'which',
 'the',
 'capital',
 'was',
 'moved',
 'to',
 'the',
 'Raigad',
 'Fort',
 'l',
 'Treasures',
 'discovered',
 'from',
 'an',
```

```
 'adjacent',
 'fort',
 'called',
 'Torna',
 'were',
 'used',
 'to',
 'completely',
 'build',
 'and',
 'fortify',
 'the',
 'Rajgad',
 'Fort']
```

In [56]:
```
from nltk import pos_tag
```

In [62]:
```
tokens = word_tokenize(para)
tagged = pos_tag(tokens)
print(tagged)
```

```
[('Rajgad', 'NNP'), ('(', '('), ('literal', 'JJ'), ('meaning', 'NN'), ('Ruling', 'NN
P'), ('Fort', 'NNP'), (')', ')'), ('is', 'VBZ'), ('a', 'DT'), ('hill', 'NN'), ('for
t', 'NN'), ('situated', 'VBN'), ('in', 'IN'), ('the', 'DT'), ('Pune', 'NNP'), ('dist
rict', 'NN'), ('of', 'IN'), ('Maharashtra', 'NNP'), (',', ','), ('India', 'NNP'),
('.', '.'), ('Formerly', 'RB'), ('known', 'VBN'), ('as', 'IN'), ('Murumdev', 'NNP'),
(',', ','), ('the', 'DT'), ('fort', 'NN'), ('was', 'VBD'), ('the', 'DT'), ('capita
l', 'NN'), ('of', 'IN'), ('the', 'DT'), ('Maratha', 'NNP'), ('Empire', 'NNP'), ('und
er', 'IN'), ('the', 'DT'), ('rule', 'NN'), ('of', 'IN'), ('Chatrapati', 'NNP'), ('Sh
ivaji', 'NNP'), ('Maharaj', 'NNP'), ('for', 'IN'), ('almost', 'RB'), ('26', 'CD'),
('years', 'NNS'), (',', ','), ('after', 'IN'), ('which', 'WDT'), ('the', 'DT'), ('ca
pital', 'NN'), ('was', 'VBD'), ('moved', 'VBN'), ('to', 'TO'), ('the', 'DT'), ('Raig
ad', 'NNP'), ('Fort', 'NNP'), ('.', '.'), ('[', 'VB'), ('1', 'JJ'), (']', 'NNP'),
('Treasures', 'NNP'), ('discovered', 'VBD'), ('from', 'IN'), ('an', 'DT'), ('adjacen
t', 'JJ'), ('fort', 'NN'), ('called', 'VBN'), ('Torna', 'NNP'), ('were', 'VBD'), ('u
sed', 'VBN'), ('to', 'TO'), ('completely', 'RB'), ('build', 'VB'), ('and', 'CC'),
('fortify', 'VB'), ('the', 'DT'), ('Rajgad', 'NNP'), ('Fort', 'NNP'), ('.', '.')]
```

In [63]:
```
nltk.download('averaged_perceptron_tagger_eng')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger_eng is already up-to-
[nltk_data]       date!
```

Out[63]: True

In [68]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [69]:
```
vectorizer = TfidfVectorizer()
```

In [71]:
```
v=vectorizer.fit_transform(t)
```

In [72]:
```
v.shape
```

Out[72]: (70, 50)

In [73]:
```
import pandas as pd
pd.DataFrame(v)
```

Out[73]:

	0
0	<Compressed Sparse Row sparse matrix of dtype ...
1	<Compressed Sparse Row sparse matrix of dtype ...
2	<Compressed Sparse Row sparse matrix of dtype ...
3	<Compressed Sparse Row sparse matrix of dtype ...
4	<Compressed Sparse Row sparse matrix of dtype ...
...	...
65	<Compressed Sparse Row sparse matrix of dtype ...
66	<Compressed Sparse Row sparse matrix of dtype ...
67	<Compressed Sparse Row sparse matrix of dtype ...
68	<Compressed Sparse Row sparse matrix of dtype ...
69	<Compressed Sparse Row sparse matrix of dtype ...

70 rows × 1 columns

Experiment No. 8

In [1]:
```
import seaborn as sns
df=  sns.load_dataset('titanic')
```

In [2]:
```
df
```

Out[2]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	ad
0	0	3	male	22.0	1	0	7.2500	S	Third	man	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.0	0	0	13.0000	S	Second	man	
887	1	1	female	19.0	0	0	30.0000	S	First	woman	
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	
889	1	1	male	26.0	0	0	30.0000	C	First	man	
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	

891 rows × 15 columns

In [3]:
```
df=df[['survived','class','sex','age','fare']]
```

In [4]:
```
df
```

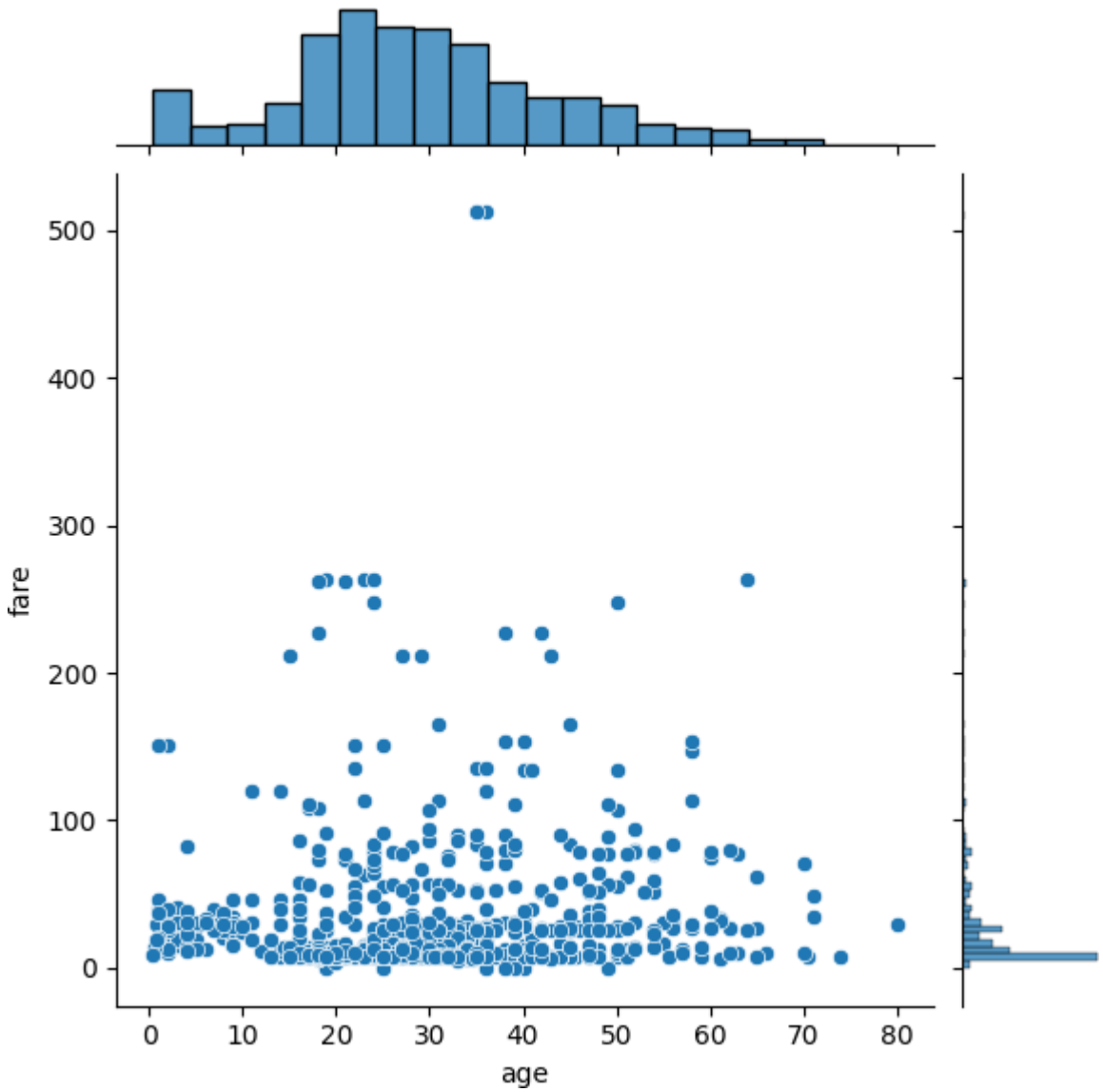
Out[4]:

	survived	class	sex	age	fare
0	0	Third	male	22.0	7.2500
1	1	First	female	38.0	71.2833
2	1	Third	female	26.0	7.9250
3	1	First	female	35.0	53.1000
4	0	Third	male	35.0	8.0500
...	...	...	...	...	...
886	0	Second	male	27.0	13.0000
887	1	First	female	19.0	30.0000
888	0	Third	female	NaN	23.4500
889	1	First	male	26.0	30.0000
890	0	Third	male	32.0	7.7500

891 rows × 5 columns

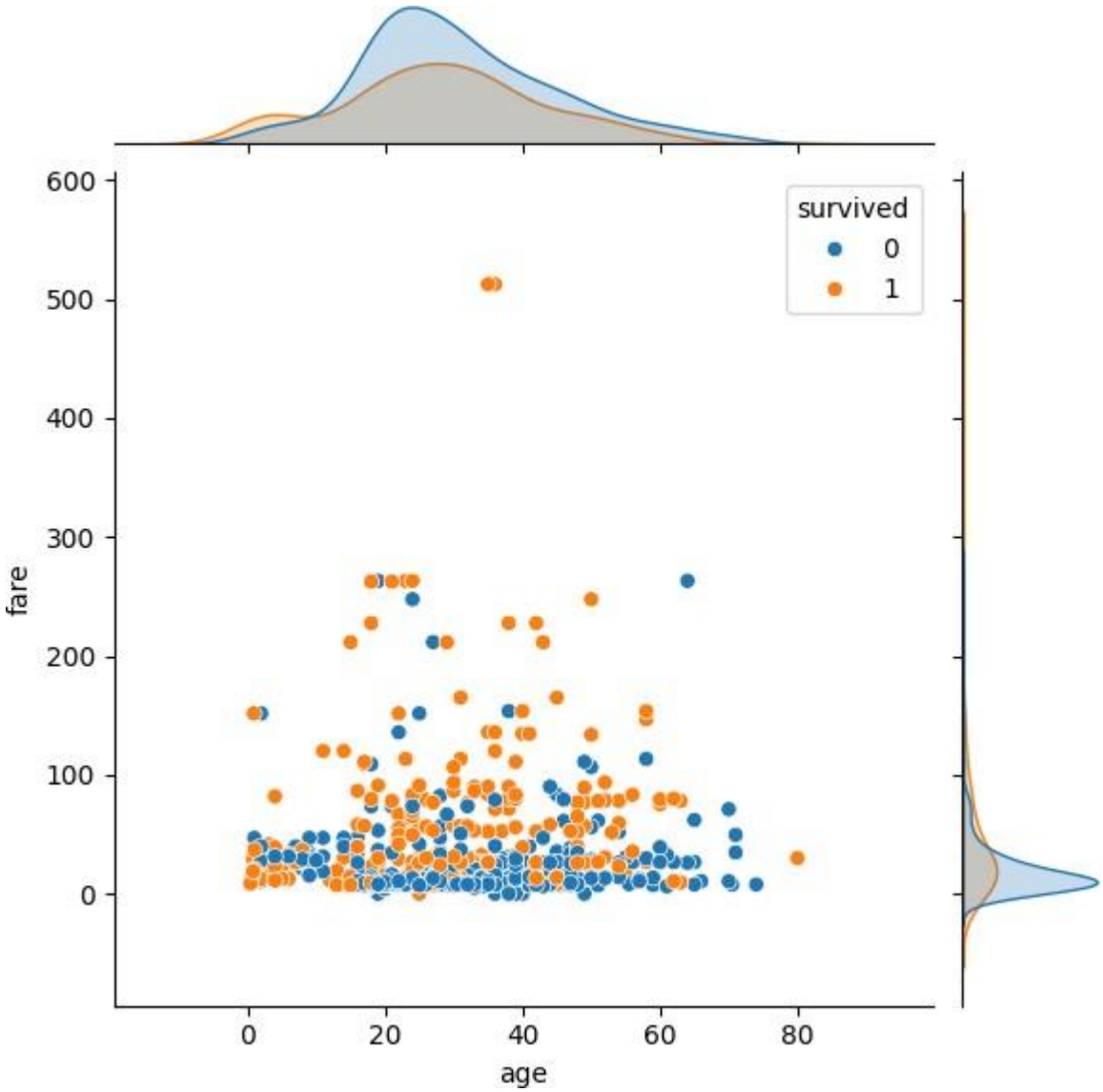
In [5]:
```
sns.jointplot(x='age',y='fare',data=df)
```

Out[5]:
```
<seaborn.axisgrid.JointGrid at 0x26139ebf8c0>
```



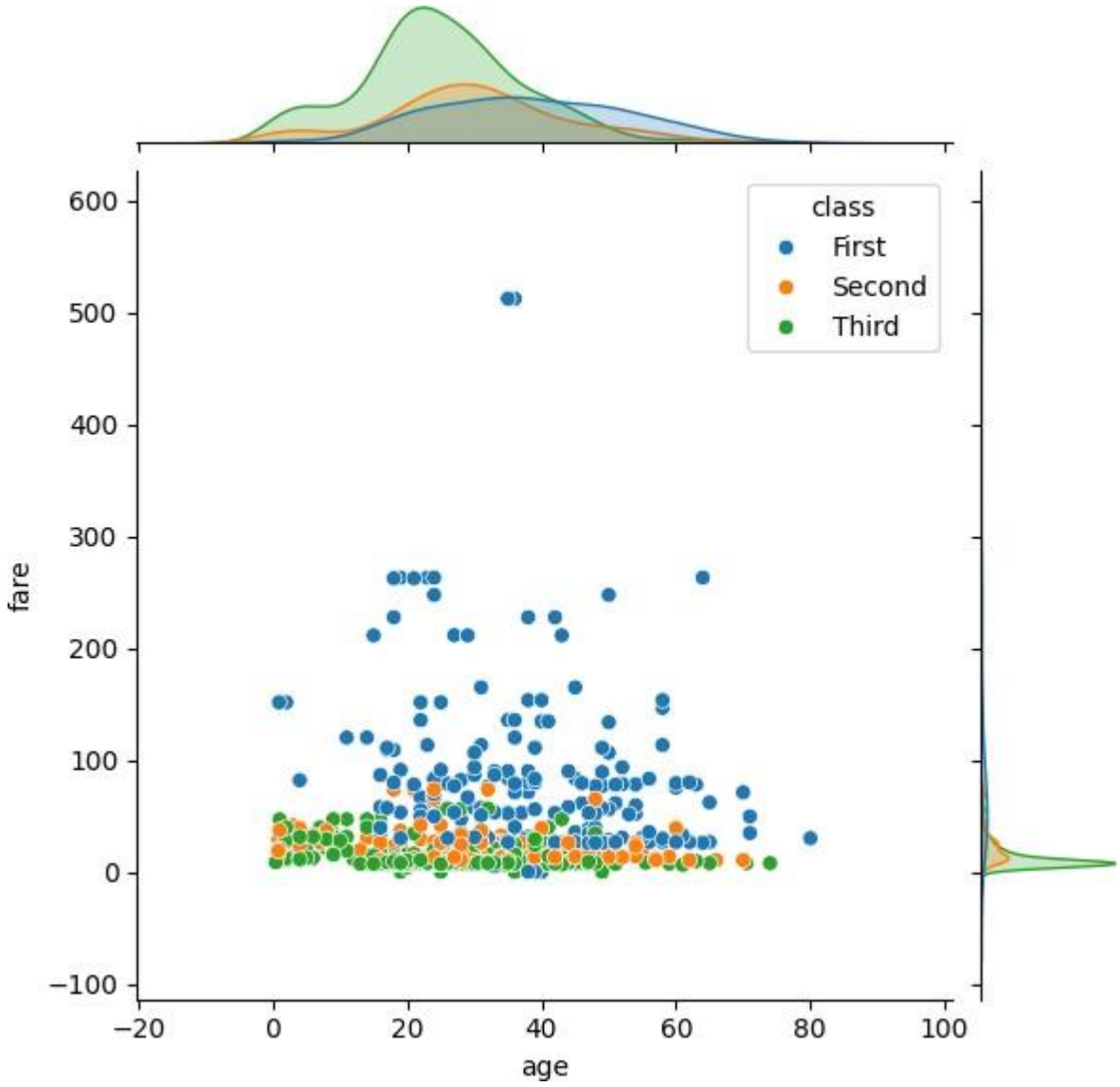
In [6]:
```
sns.jointplot(x='age',y='fare',data=df,hue='survived')
```

Out[6]: <seaborn.axisgrid.JointGrid at 0x2616d6e56d0>



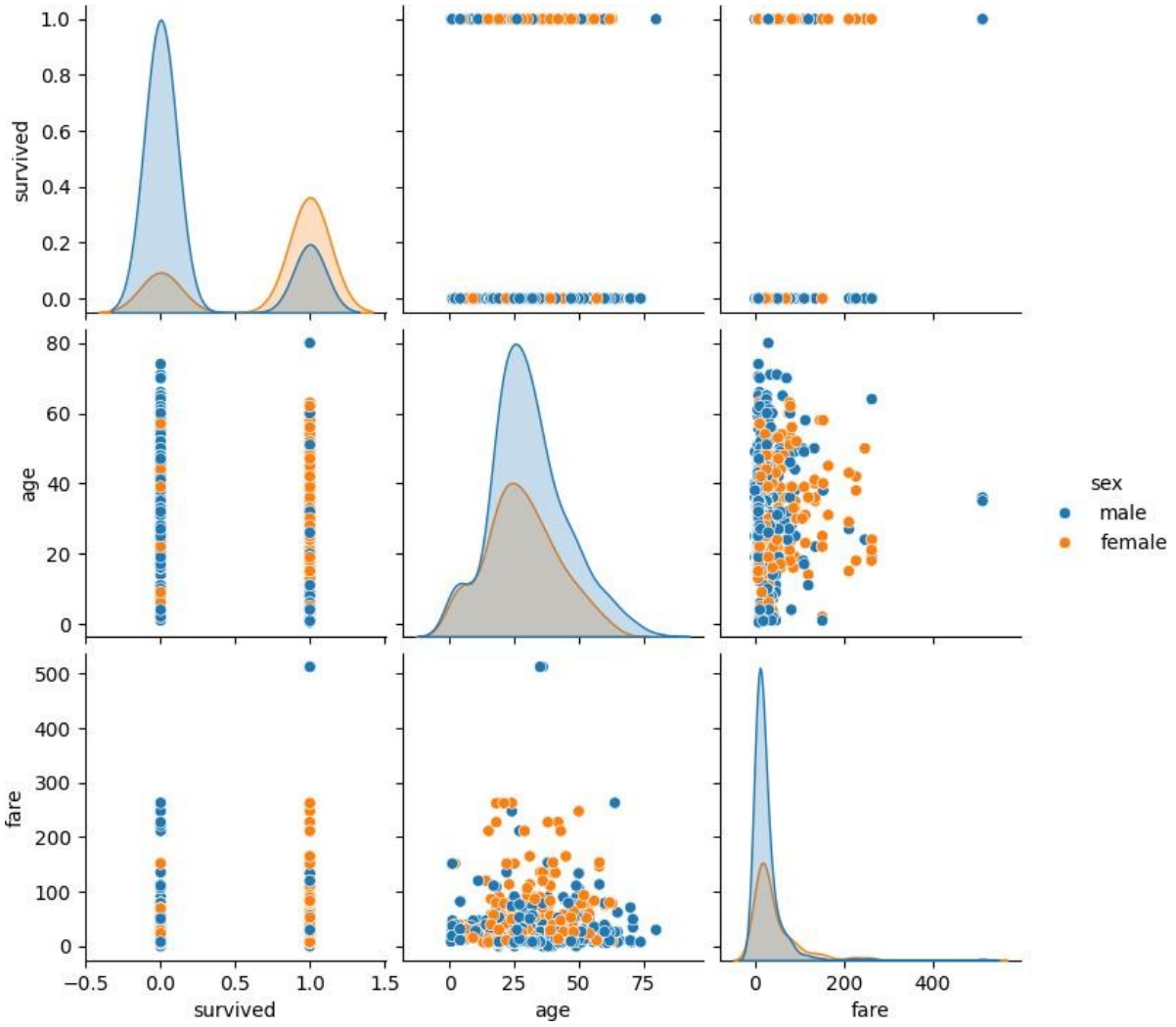
In [7]:
```
sns.jointplot(x='age',y='fare',data=df,hue='class')
```

Out[7]: <seaborn.axisgrid.JointGrid at 0x2616d8d9f90>



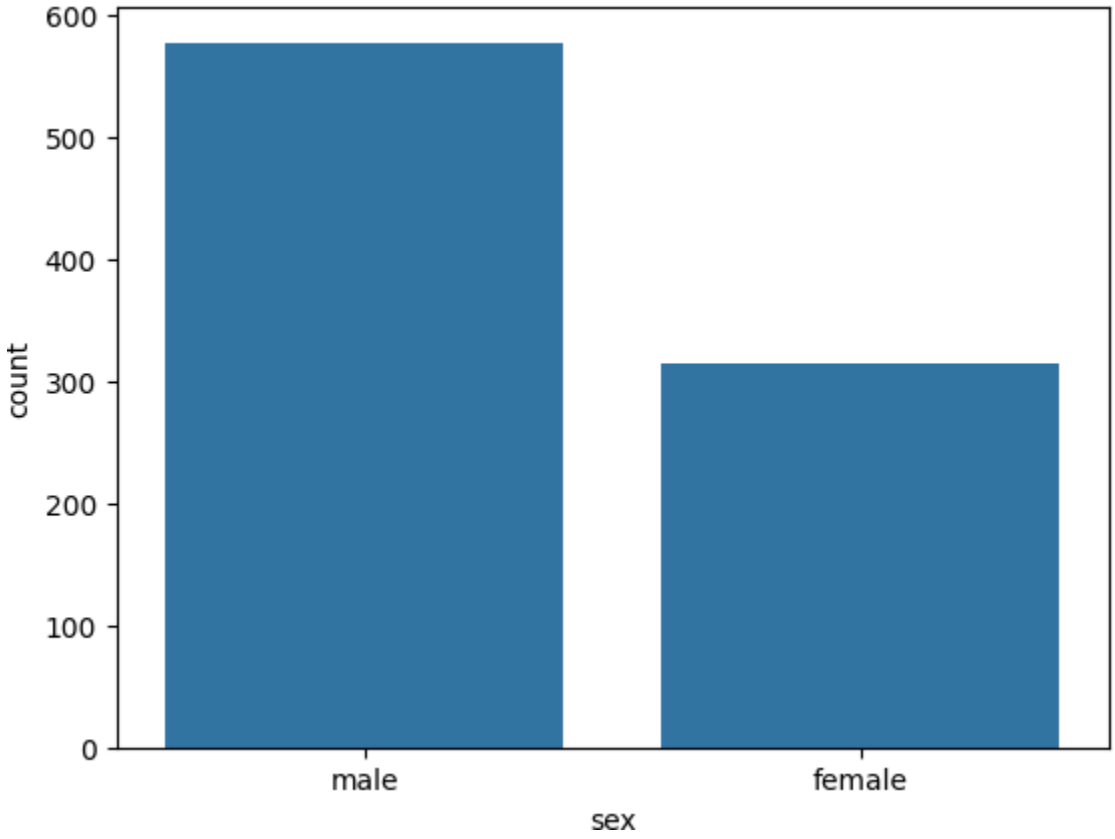
In [8]:
```
sns.pairplot(df,hue='sex')
```

Out[8]: <seaborn.axisgrid.PairGrid at 0x2616da14ad0>



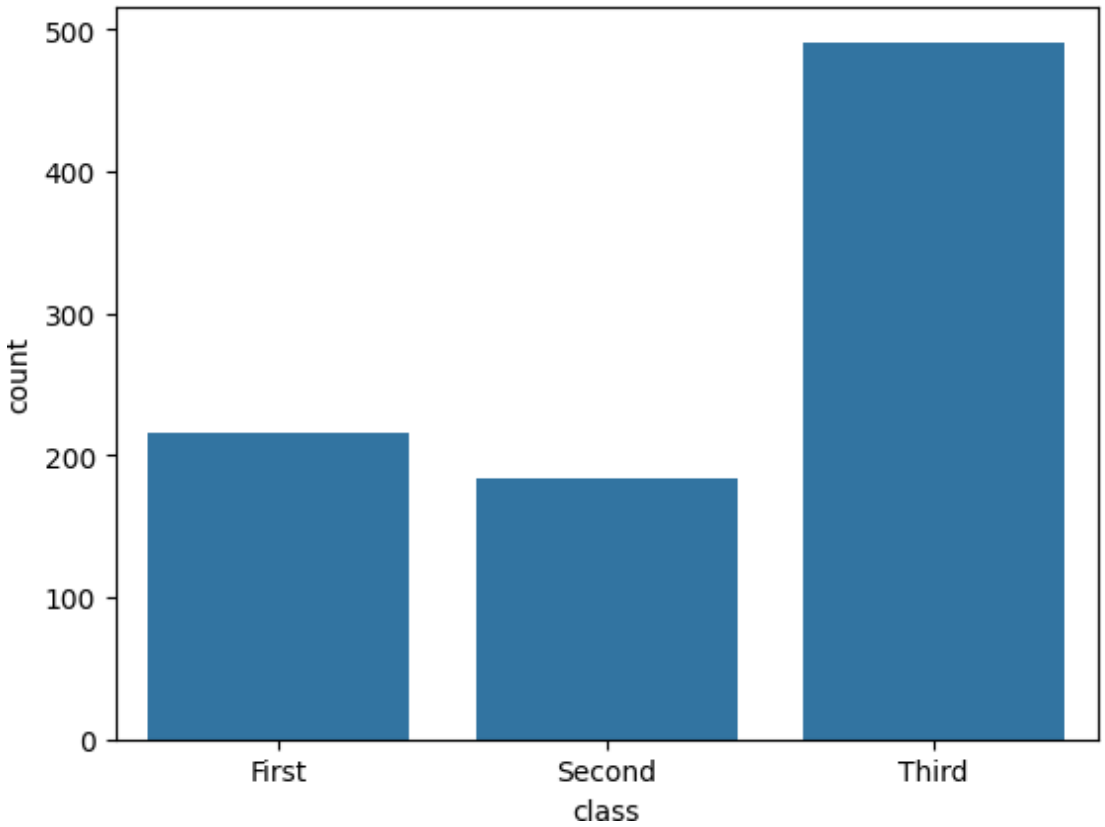
In [9]:
```
sns.countplot(x=df['sex'])
```

Out[9]: <Axes: xlabel='sex', ylabel='count'>



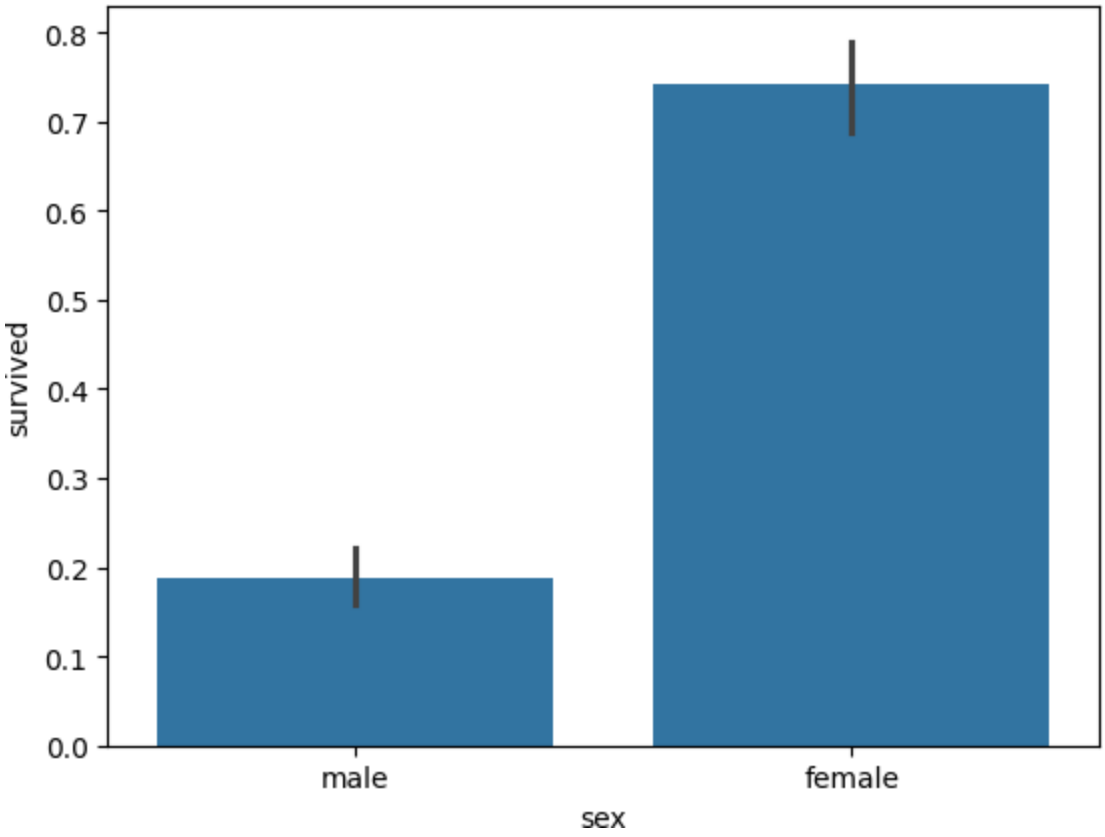
In [10]:
```
sns.countplot(x=df['class'])
```

Out[10]: <Axes: xlabel='class', ylabel='count'>



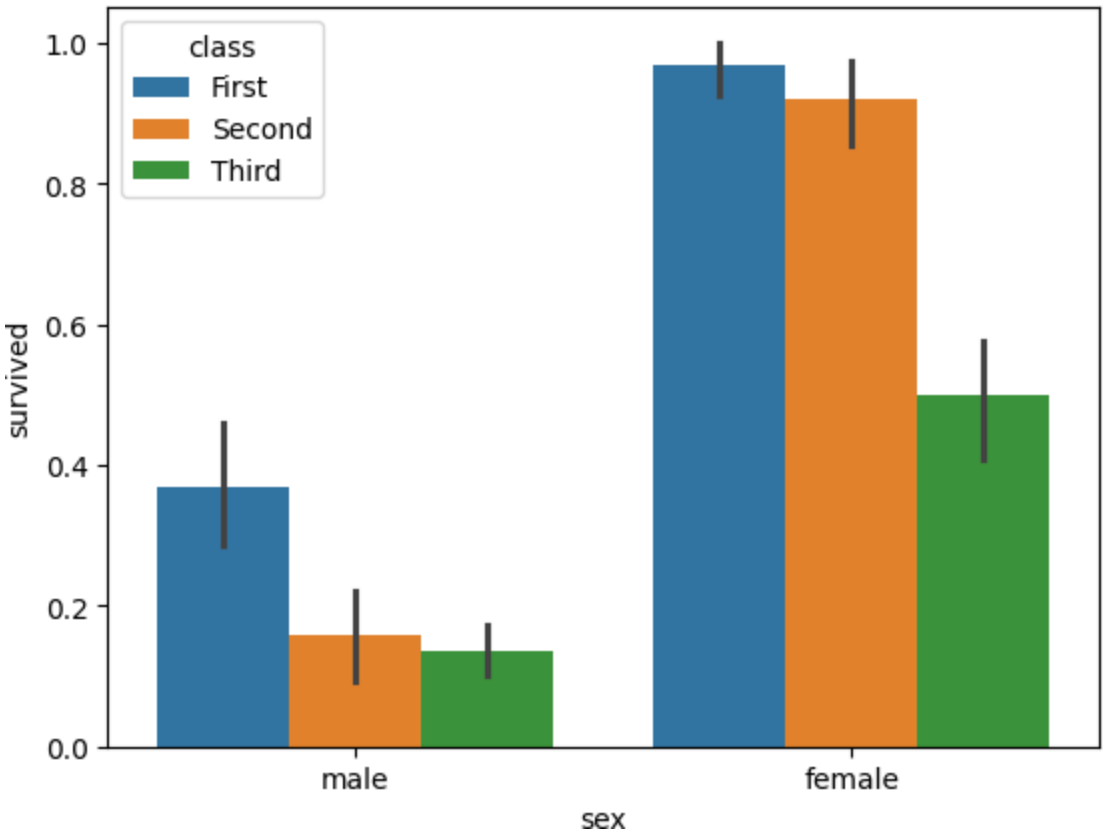
In [11]:
```
sns.barplot(x='sex',y='survived',data=df)
```

Out[11]: <Axes: xlabel='sex', ylabel='survived'>



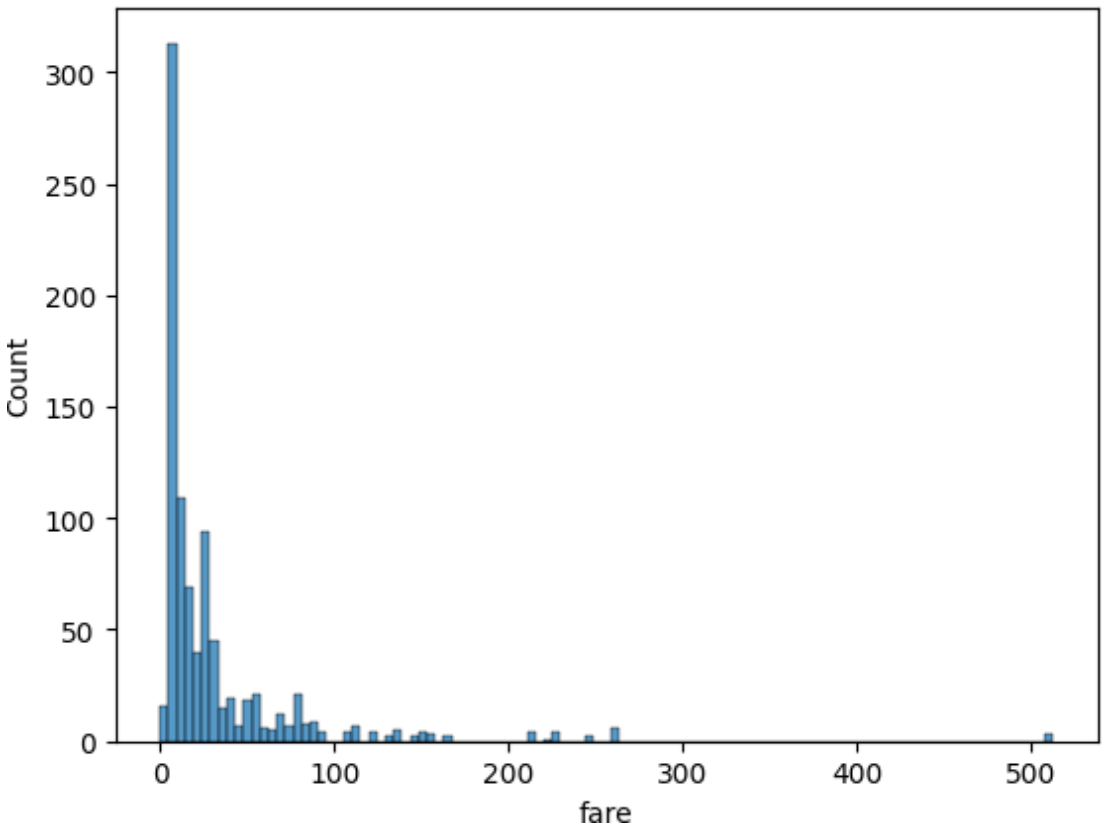
In [12]:
```
sns.barplot(x='sex',y='survived',hue='class',data=df)
```

Out[12]: <Axes: xlabel='sex', ylabel='survived'>



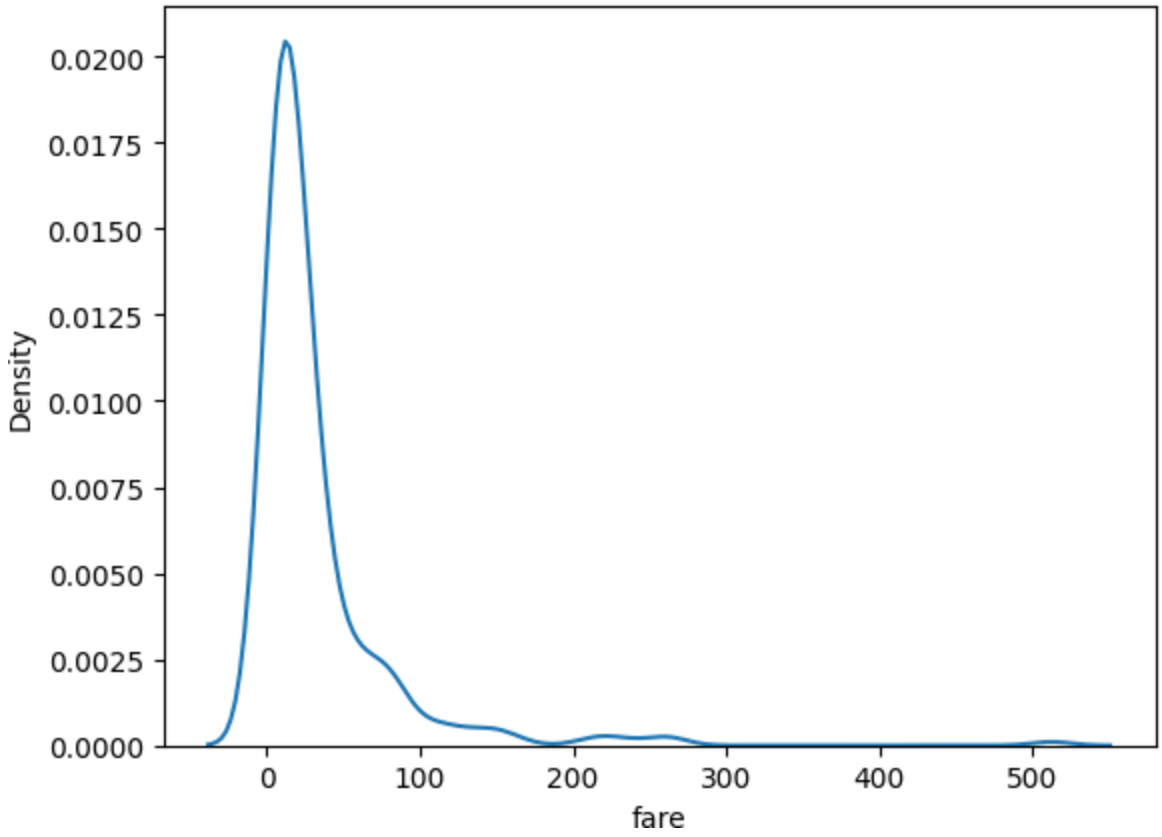
In [13]:
```
sns.histplot(df['fare'])
```

Out[13]: <Axes: xlabel='fare', ylabel='Count'>



In [14]:
```
sns.kdeplot(df['fare'])
```

Out[14]: <Axes: xlabel='fare', ylabel='Density'>



In [ ]:

Experiment No. 9

In [1]:
```
import seaborn as sns
```

In [2]:
```
df = sns. load_dataset('titanic')
```

In [3]:
```
df=  df[['sex','age','survived']]
```

In [4]:
```
df
```

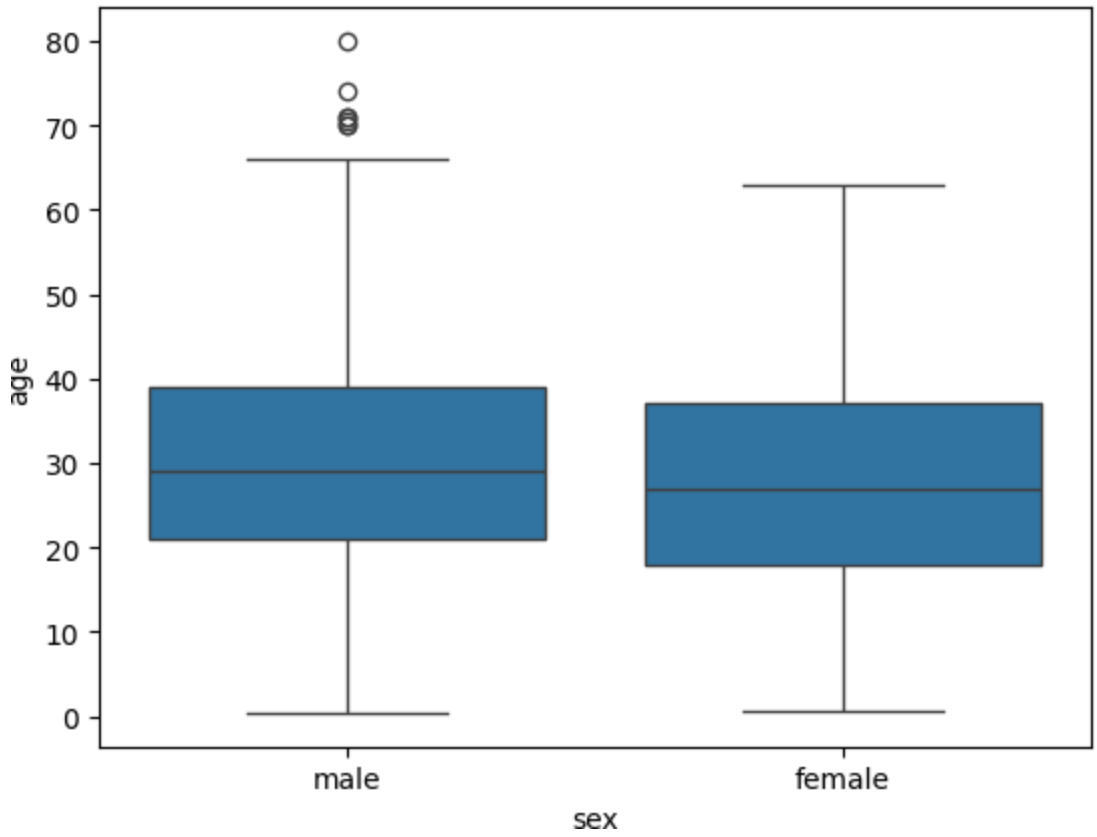
Out[4]:

	sex	age	survived
0	male	22.0	0
1	female	38.0	1
2	female	26.0	1
3	female	35.0	1
4	male	35.0	0
...	...	...	...
886	male	27.0	0
887	female	19.0	1
888	female	NaN	0
889	male	26.0	1
890	male	32.0	0

891 rows × 3 columns

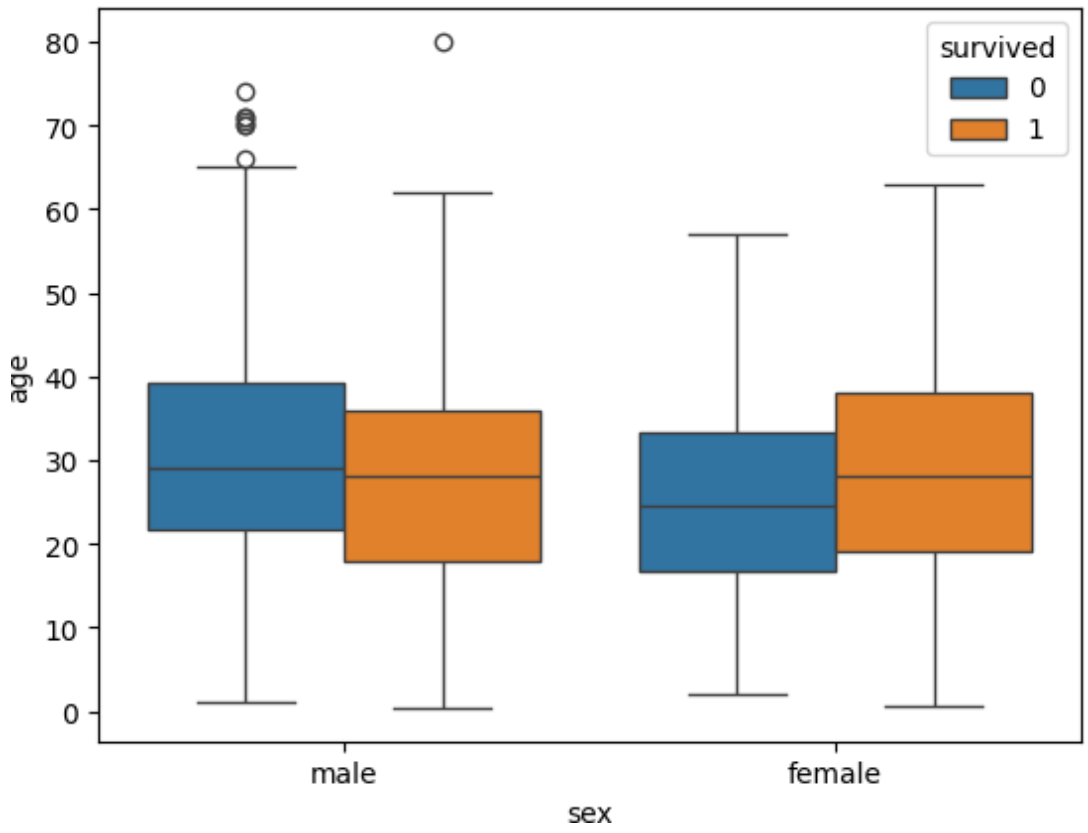
In [5]:
```
sns.boxplot(x='sex',y='age',data=df)
```

Out[5]:
```
<Axes: xlabel='sex', ylabel='age'>
```



In [6]:
```
sns.boxplot(x='sex',y='age',hue='survived',  data=df)
```

Out[6]: <Axes: xlabel='sex', ylabel='age'>



Experiment No. 10

In [23]:
```
import seaborn as sns
```

In [24]:
```
df =sns.load_dataset('iris')
```

In [25]:
```
df
```

Out[25]:

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [26]:
```
#list down there features and tere types available in dataset
df.columns
```

Out[26]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [27]:
```
df.info()
```

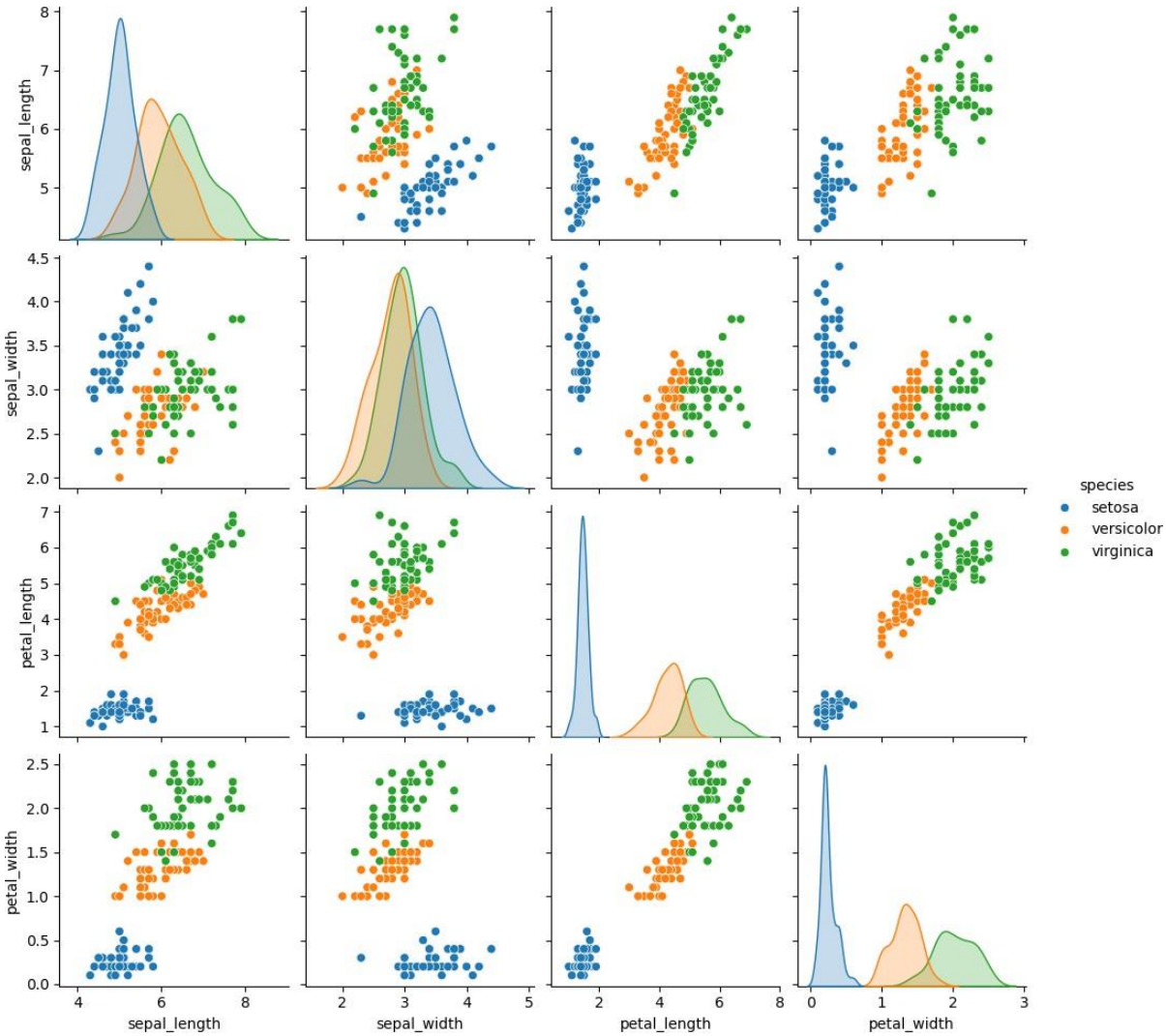
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [28]:
```
df.dtypes
```

Out[28]:
```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

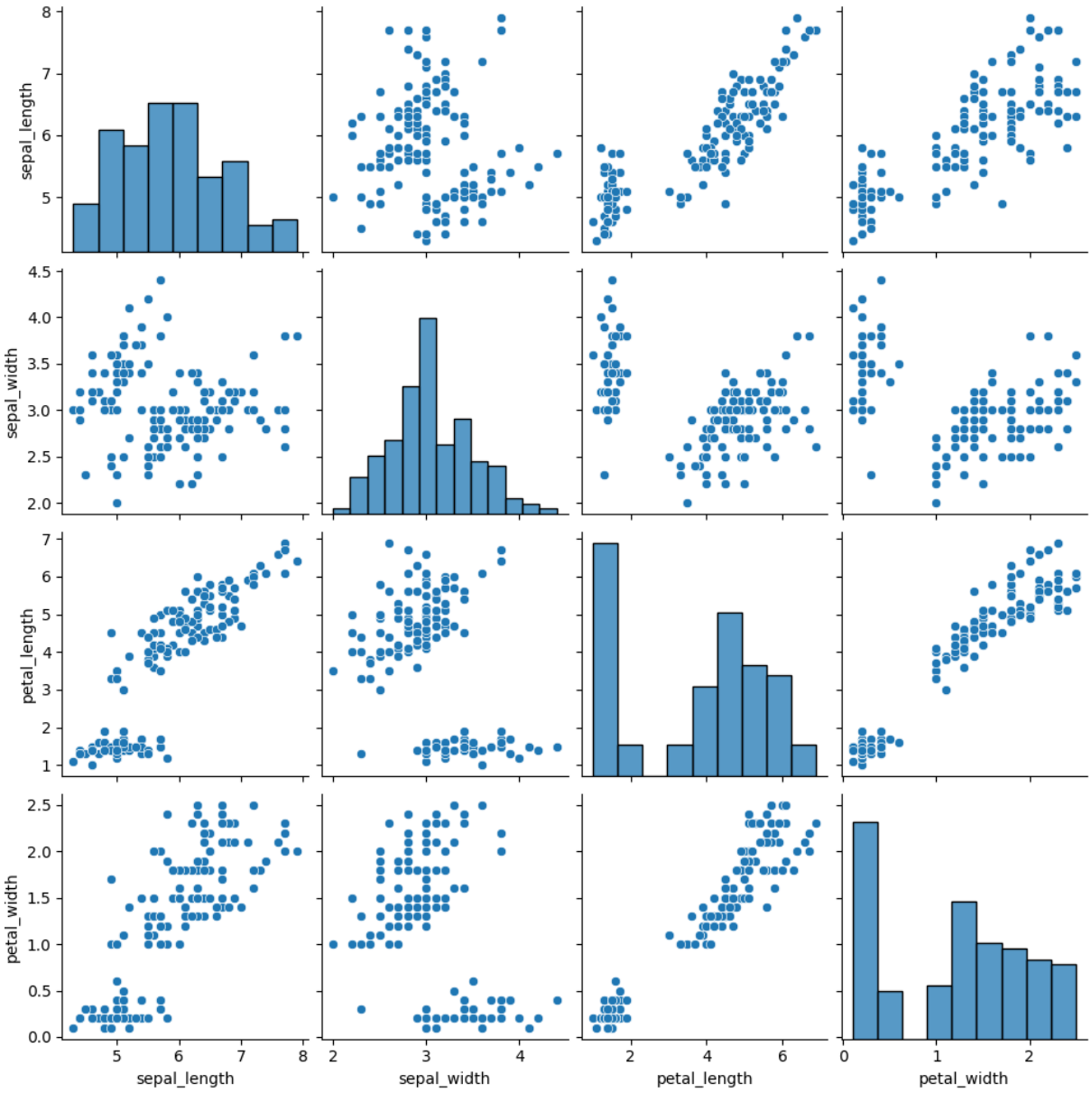
In [29]:
```
sns.pairplot(df,hue='species')
```

Out[29]: <seaborn.axisgrid.PairGrid at 0x1fb45ca8050>



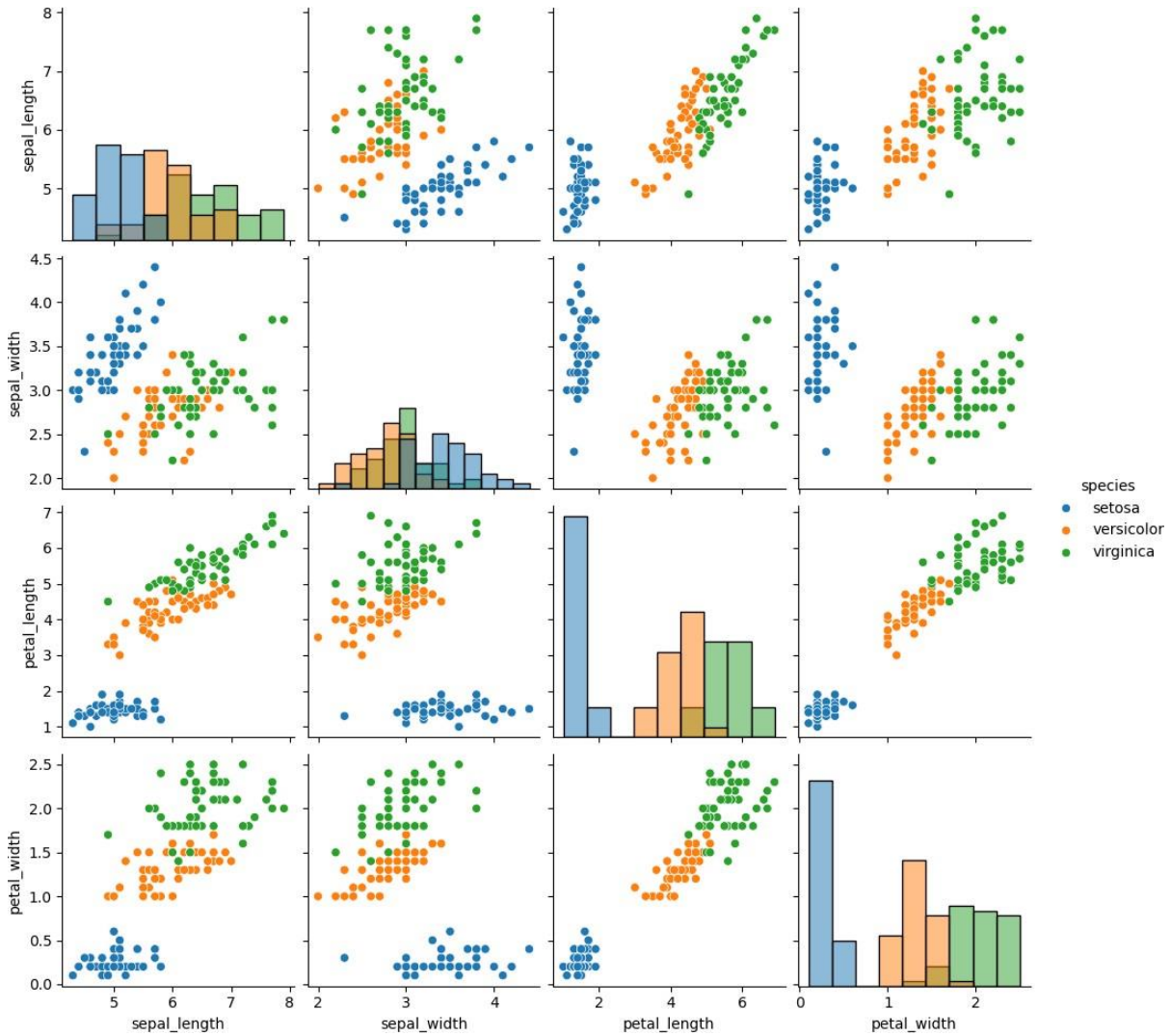
In [30]:
```
sns.pairplot(df)
```

Out[30]: <seaborn.axisgrid.PairGrid at 0x1fb45ca82b0>



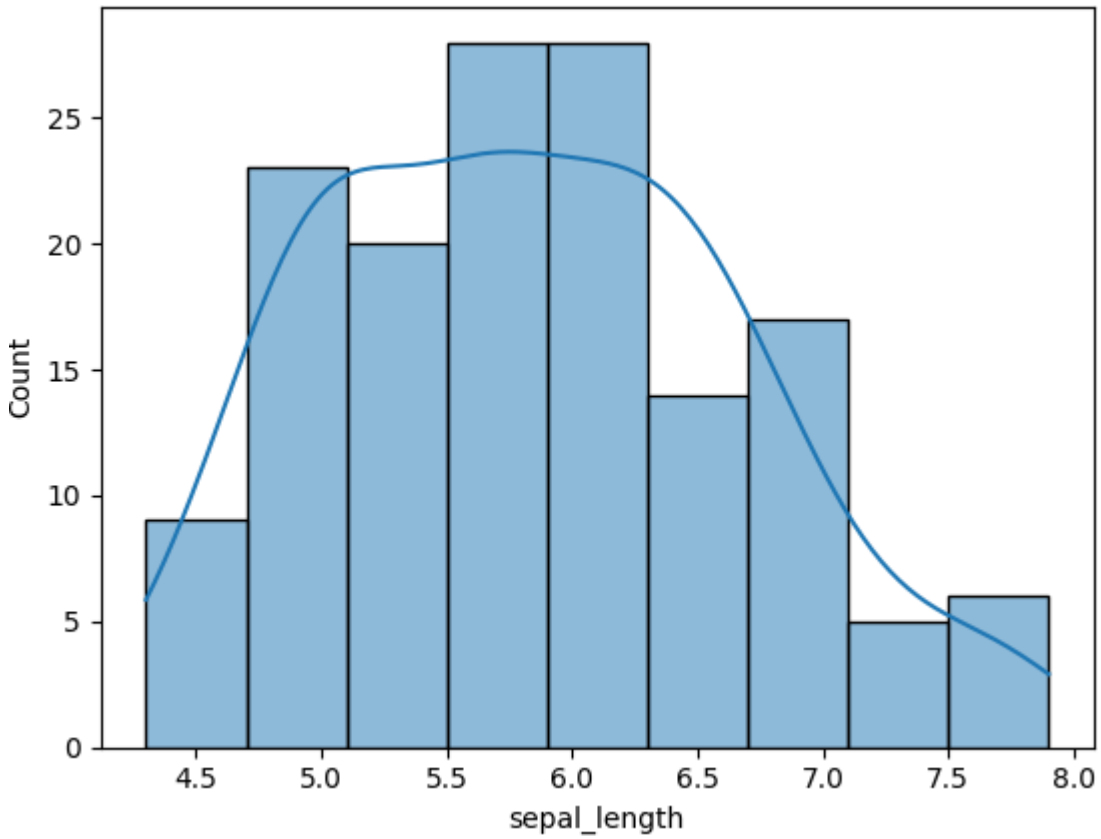
In [31]:
```
sns.pairplot(df,hue='species',diag_kind='hist')
```

Out[31]:
```
<seaborn.axisgrid.PairGrid at 0x1fb45ca8180>
```



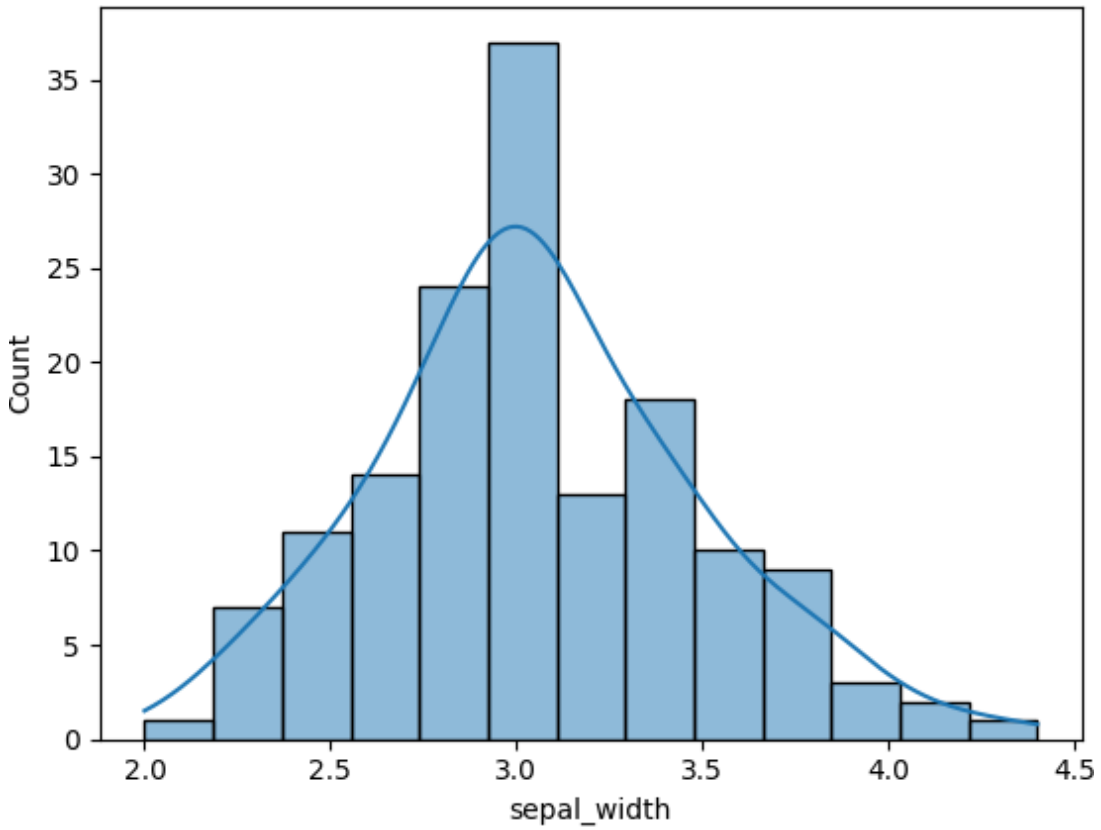
In [32]:
```
sns.histplot(df['sepal_length'],kde=True)
```

Out[32]: <Axes: xlabel='sepal_length', ylabel='Count'>



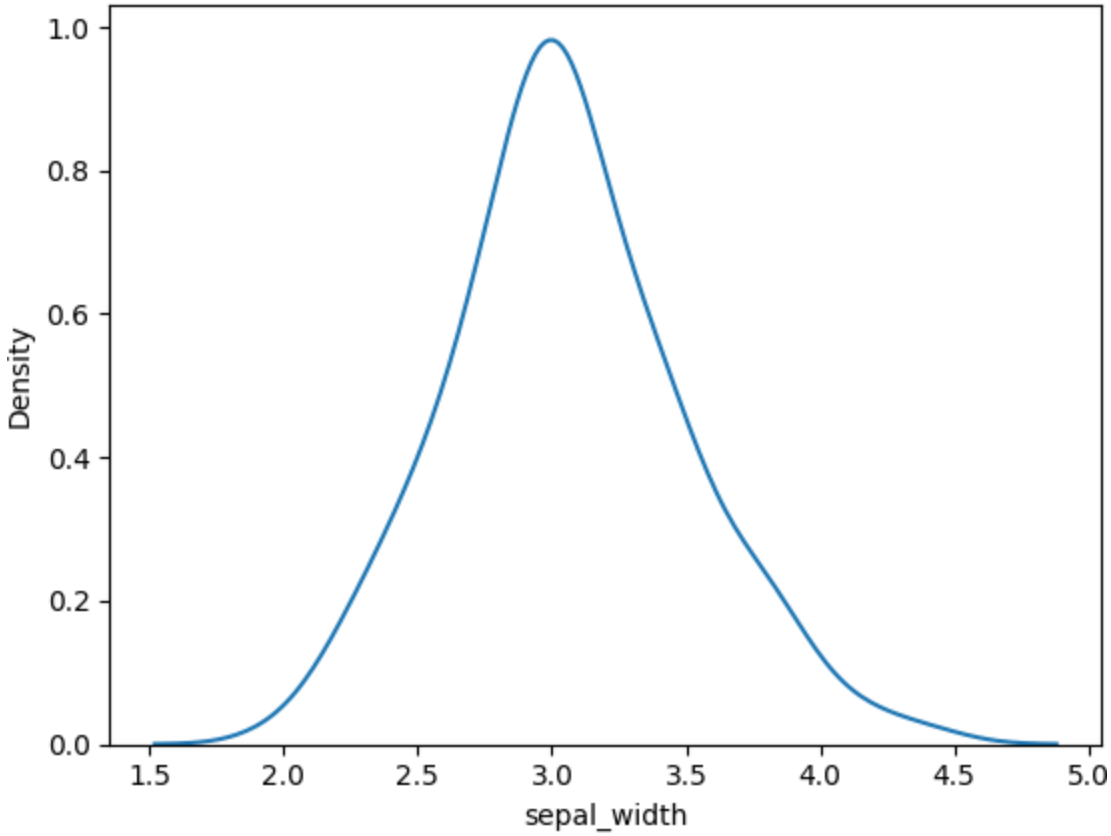
In [33]:
```
sns.histplot(df['sepal_width'],kde=True)
```

Out[33]: <Axes: xlabel='sepal_width', ylabel='Count'>



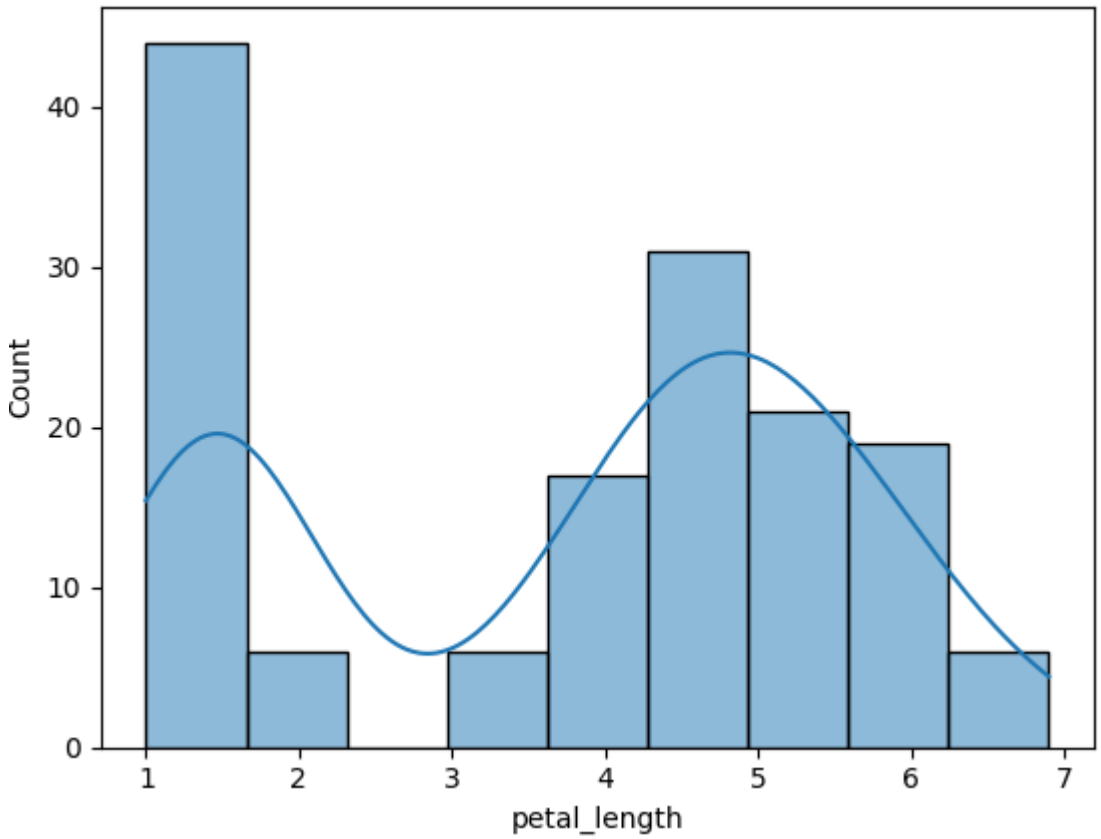
In [34]:
```
sns.kdeplot(df['sepal_width'])
```

Out[34]: <Axes: xlabel='sepal_width', ylabel='Density'>



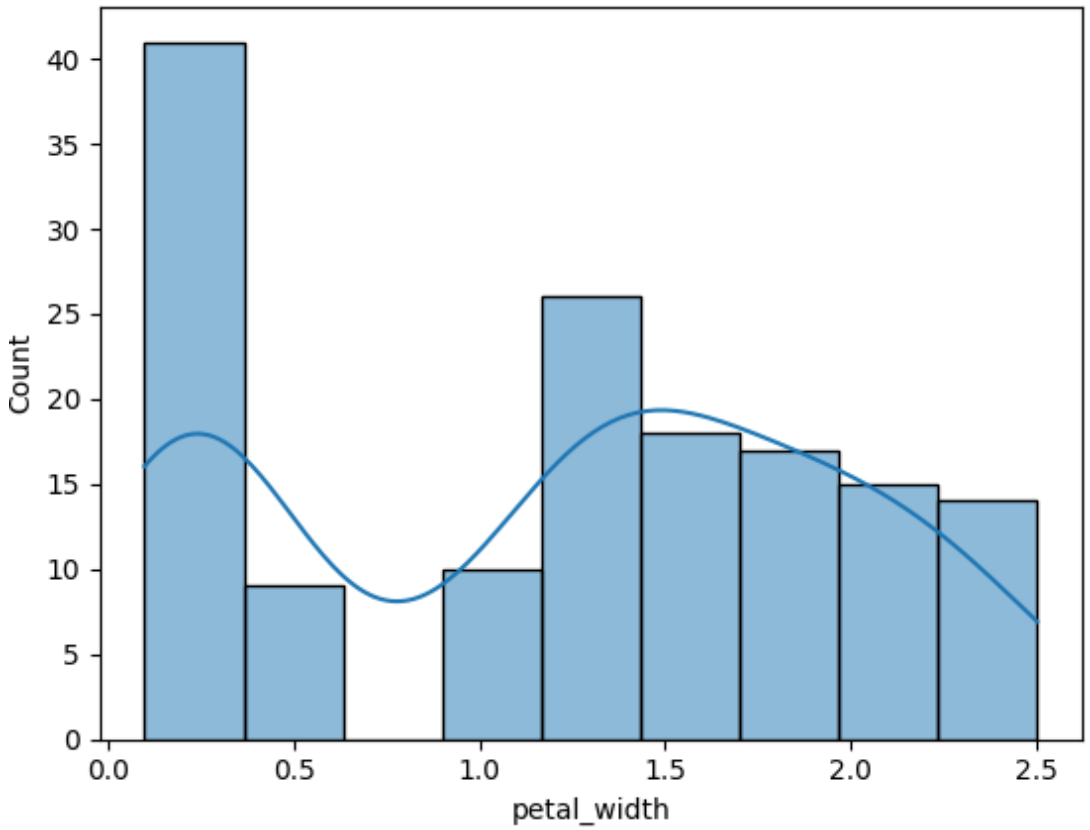
In [35]:
```
sns.histplot(df['petal_length'],kde=True)
```

Out[35]: <Axes: xlabel='petal_length', ylabel='Count'>



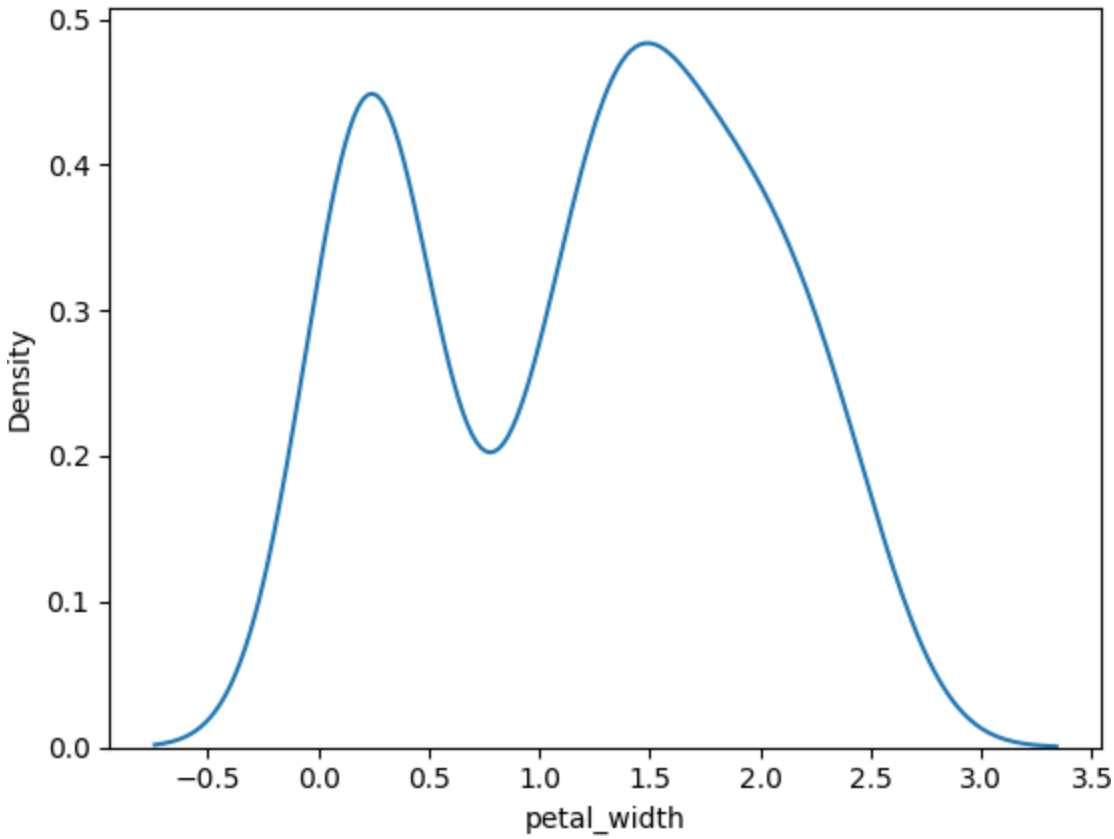
In [36]:
```
sns.histplot(df['petal_width'],kde=True)
```

Out[36]: <Axes: xlabel='petal_width', ylabel='Count'>

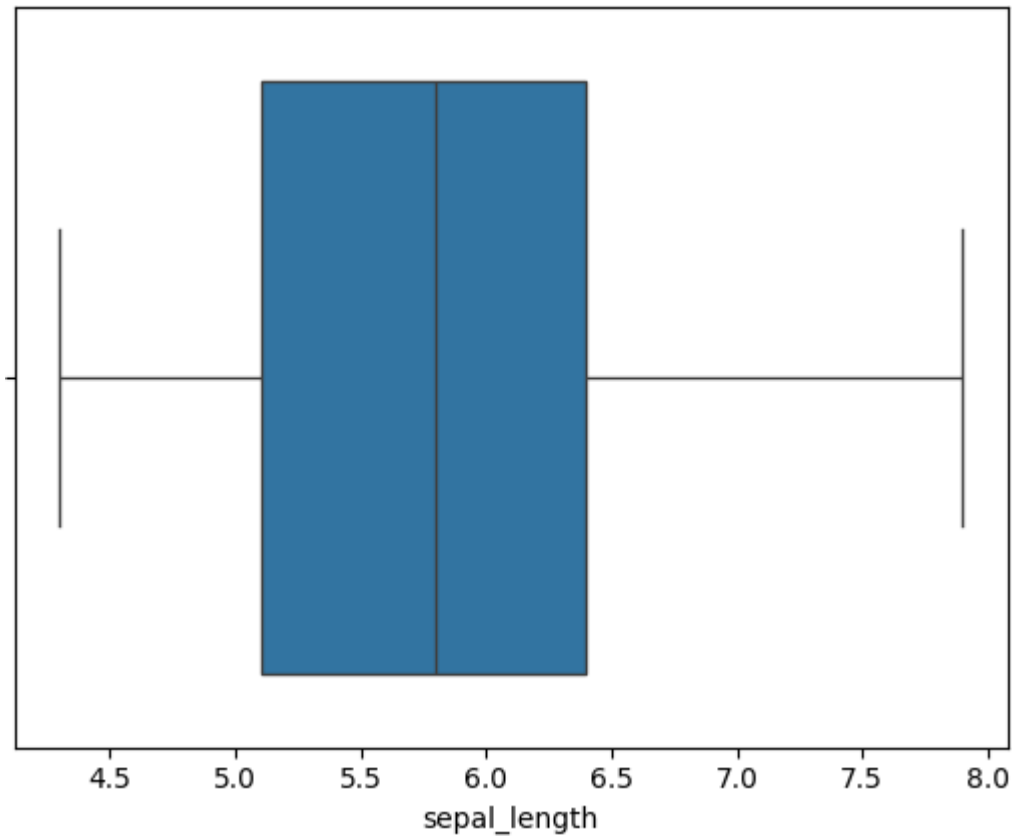


In [37]:
```
sns.kdeplot(df['petal_width'])
```

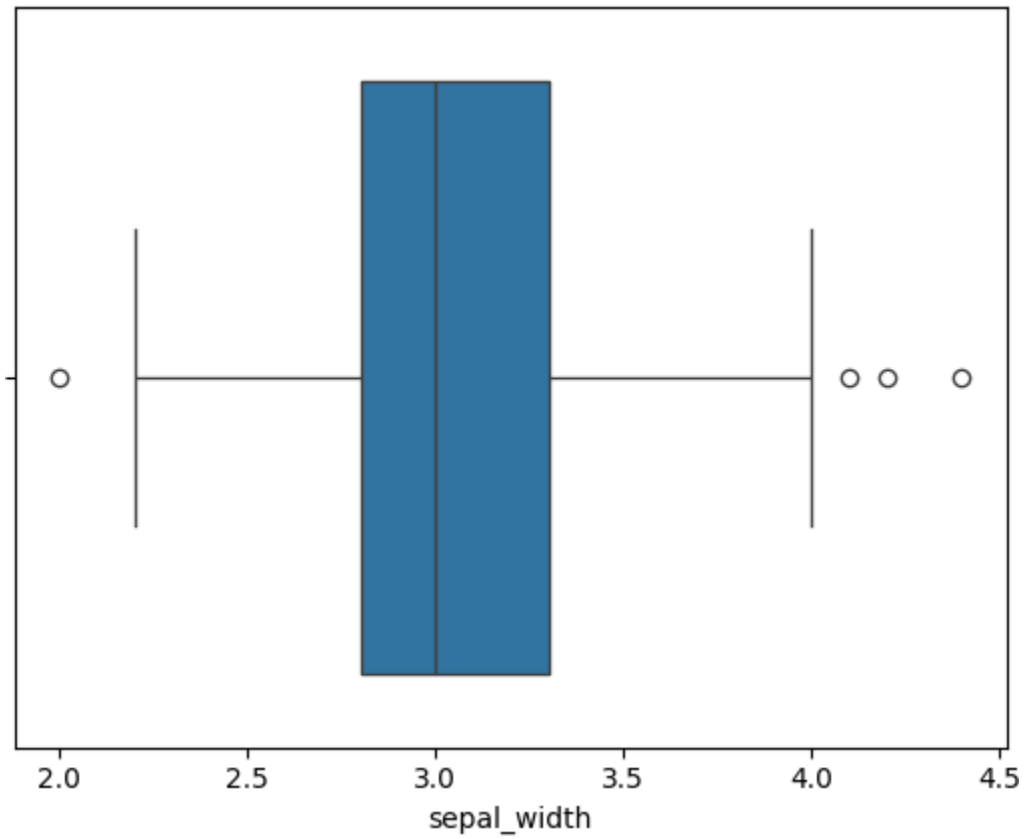
Out[37]: <Axes: xlabel='petal_width', ylabel='Density'>



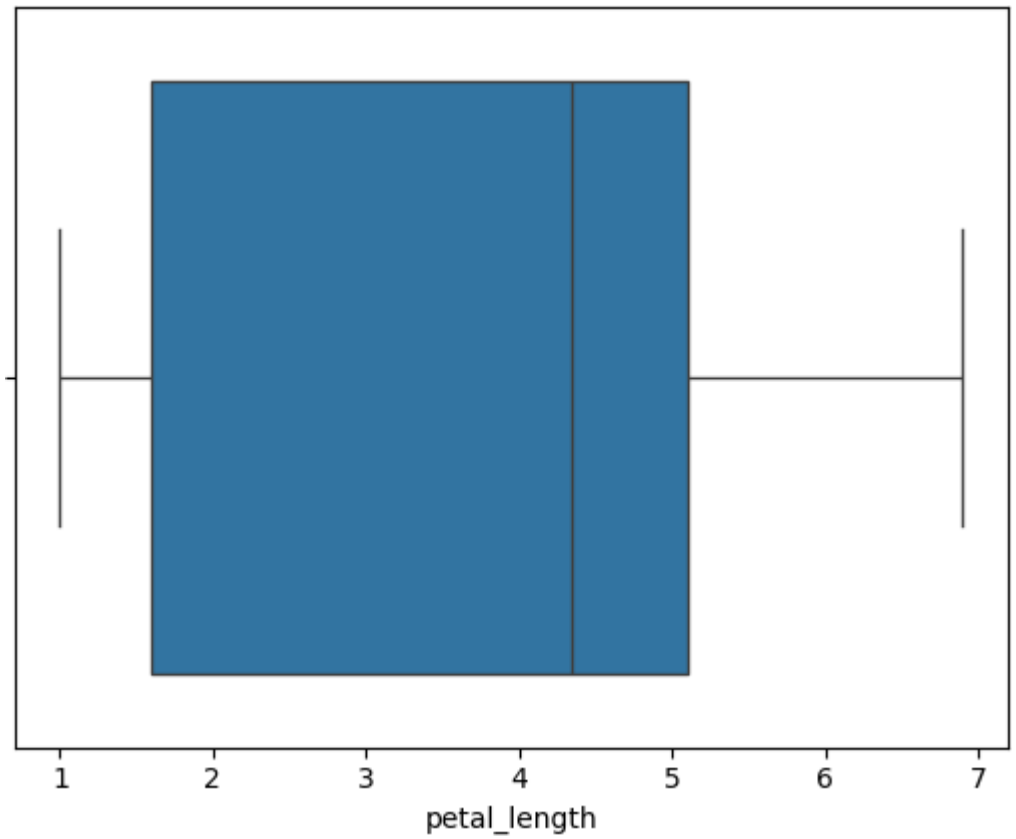
In [38]:
```
sns.boxplot(x=df['sepal_length']);
```



In [39]:
```
sns.boxplot(x=df['sepal_width']);
```



In [40]:
```
sns.boxplot(x=df['petal_length']);
```



In [41]:
```
sns.boxplot(x=df['petal_width']);
```