St. Francis Institute of Technology, Mumbai-400 103

Department of Information Technology

A.Y. 2024-2025

Class: TE-ITA/B, Semester: VI

Subject: **Data Science Lab**

Experiment – 3: To implement Data Modelling.

1. **Aim:** To implement Data Modelling
2. **Objectives:** After study of this experiment, the student will be able to
   - Understand how data to be re-scaled
   - Understand how data partitioning in done using sklearn
3. **Outcomes:** After study of this experiment, the student will be able to
   - Understand data rescaling and data partitioning using sklearn
4. **Prerequisite:** Fundamentals of Python Programming and Database Management System.
5. **Requirements:** Python Installation, Personal Computer, Windows operating system, Internet Connection, Microsoft Word.
6. **Pre-Experiment Exercise:**

Brief Theory:

- Concept of sklearn package for machine learning.

Laboratory Exercise

A. **Procedure:** (the sheet for commands in attached with the file)

B. Paste Screenshots of above commands.

8. Post-Experiments Exercise

A. Extended Theory: (Soft Copy)

- Use iris dataset and perform rescaling using sklearn package using normalization
- Partition the iris dataset such that 80% data to be taken for training purpose

B. Questions:

- Mention three differences between normalization and standardization.
- Describe train_test_split function

C. Conclusion:

Write the significance of the topic studied in the experiment.

D. References:

1. https://www.geeksforgeeks.org/exploratory-data-analysis-on-iris-dataset/
2. https://www.statology.org/normalize-columns-pandas-dataframe/Normalization
3. https://www.datascienceguide.org/python-code-snippets.html
4. https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/

In [ ]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
```

In [ ]:

```
sns.get_dataset_names()
```

Out[ ]:

```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

In [ ]:

```
df = sns.load_dataset('titanic')
```

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

In [ ]:

```
df.describe()
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [ ]:

```
df
```

Out[ ]:

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

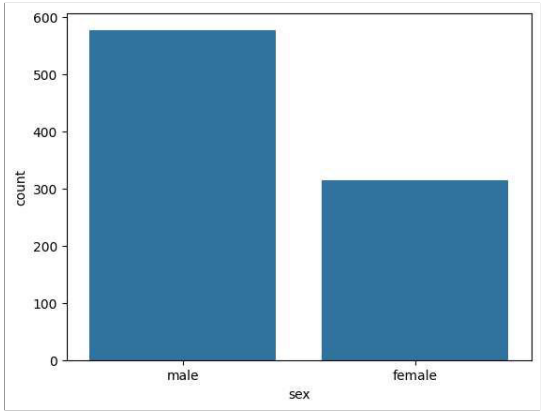
891 rows × 15 columns

In [ ]:

```
sns.countplot(x="sex", data=df)
```

Out[ ]:

```
<Axes: xlabel='sex', ylabel='count'>
```

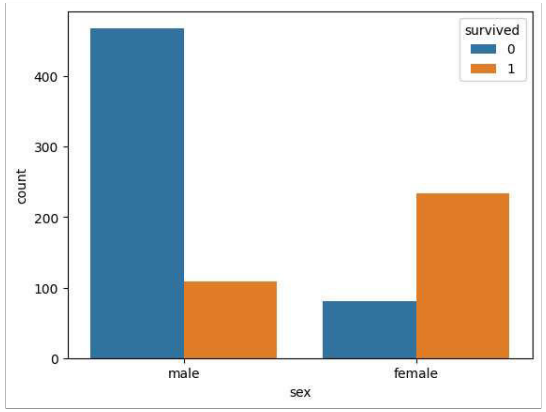


In [ ]:

```
sns.countplot(x="sex", hue="survived", data=df)
```

Out[ ]:

```
<Axes: xlabel='sex', ylabel='count'>
```

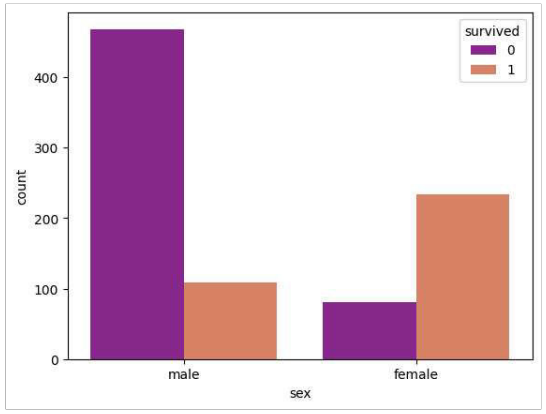


In [ ]:

```
sns.countplot(x="sex", hue="survived", data=df, palette = "plasma")
```

Out[ ]:

```
<Axes: xlabel='sex', ylabel='count'>
```

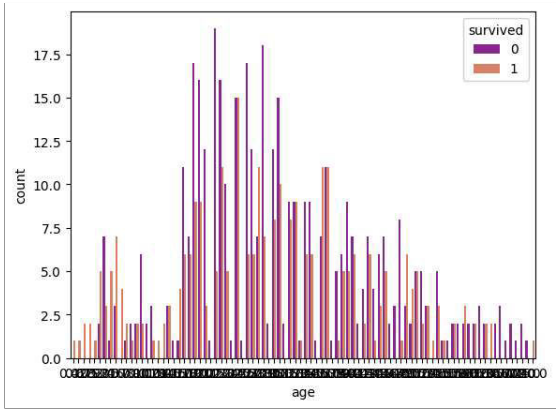


In [ ]:

```
sns.countplot(x="age",hue="survived", data=df, palette="plasma")
```

Out[ ]:

```
<Axes: xlabel='age', ylabel='count'>
```



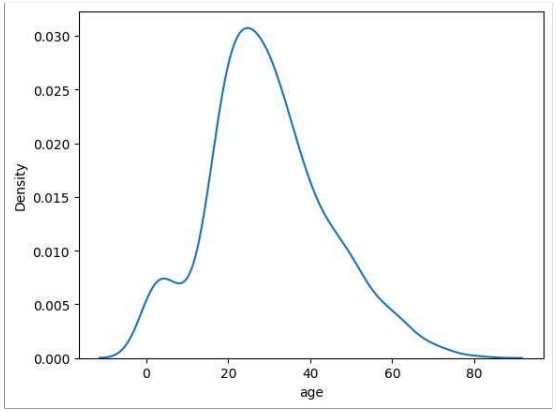
In [ ]:

```
sns.kdeplot(x="age", data=df,palette = "plasma")
```

```
<ipython-input-6-72a6893a33ce>:1: UserWarning: Ignoring `palette` because no `hue` var
iable has been assigned.
  sns.kdeplot(x="age", data=df,palette = "plasma")
```

Out[ ]:

```
<Axes: xlabel='age', ylabel='Density'>
```

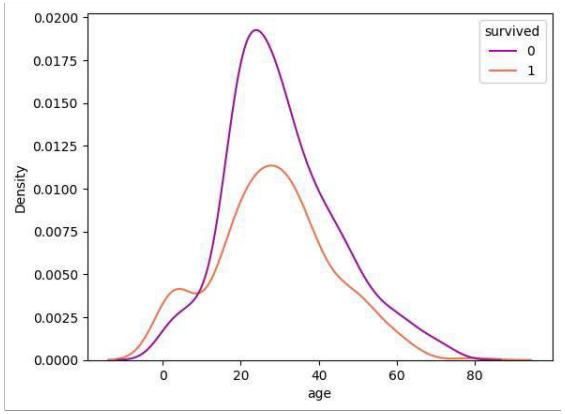


In [ ]:

```
sns.kdeplot(x="age", hue="survived", data=df, palette="plasma")
```

Out[ ]:

```
<Axes: xlabel='age', ylabel='Density'>
```



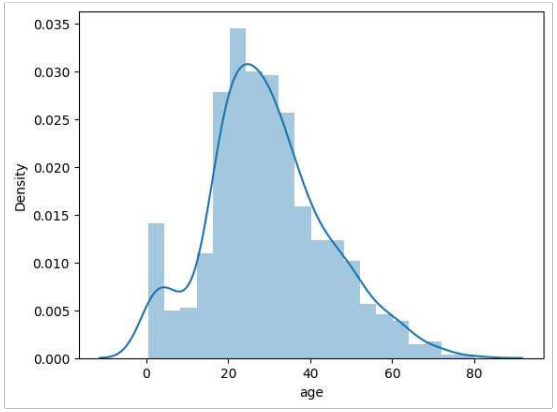
In [ ]:

```
sns.distplot(df["age"])
```

```
<ipython-input-7-eef84e7ff8f0>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["age"])
```

Out[ ]:

```
<Axes: xlabel='age', ylabel='Density'>
```

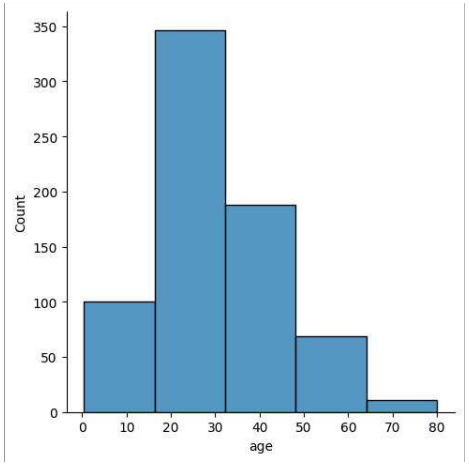


In [ ]:

```
sns.displot(x="age", bins=5, data = df)
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7b156d858b50>
```

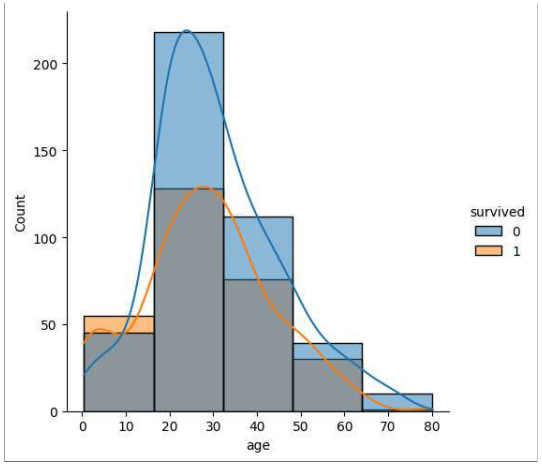


In [ ]:

```
sns.displot(x="age", bins=5, hue="survived", data=df, kde=True)
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7b1563df7310>
```



In [ ]:

```
group = df.groupby(["sex","survived"])
```

In [ ]:

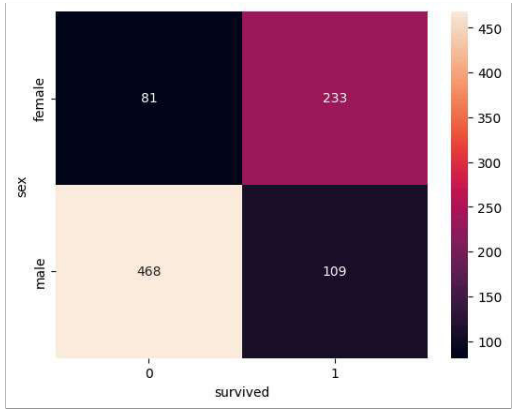
```
class_survived = group.size().unstack()
```

In [ ]:

```
sns.heatmap(class_survived, annot=True, fmt="d")
```

Out[ ]:

```
<Axes: xlabel='survived', ylabel='sex'>
```

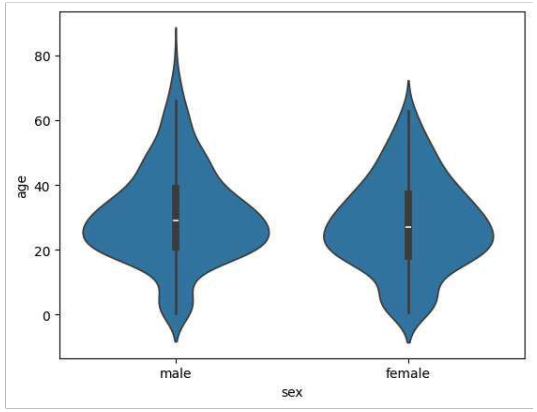


In [ ]:

```
sns.violinplot(x="sex",y="age",data=df)
```

Out[ ]:

```
<Axes: xlabel='sex', ylabel='age'>
```

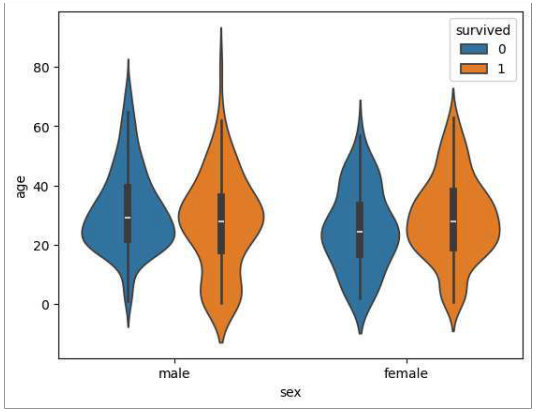


In [ ]:

```
sns.violinplot(x="sex", y="age", hue="survived", data = df)
```

Out[ ]:

```
<Axes: xlabel='sex', ylabel='age'>
```

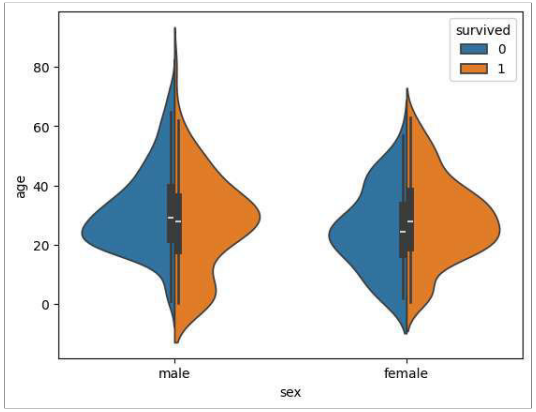


In [ ]:

```
sns.violinplot(x="sex", y="age", hue="survived", data = df, split=True)
```

Out[ ]:

```
<Axes: xlabel='sex', ylabel='age'>
```

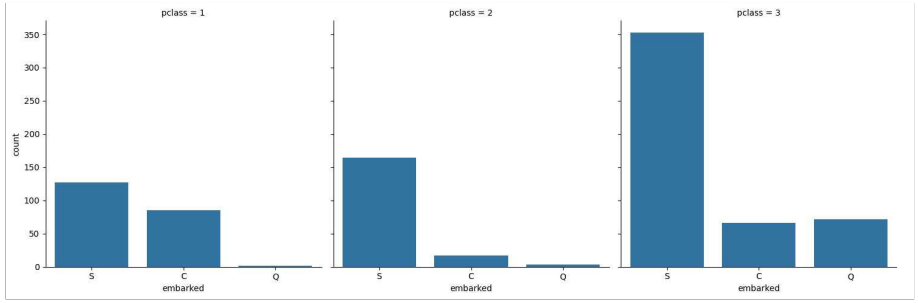


In [ ]:

```
sns.catplot(x="embarked", kind="count", col="pclass", data=df)
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7b15638b8a90>
```

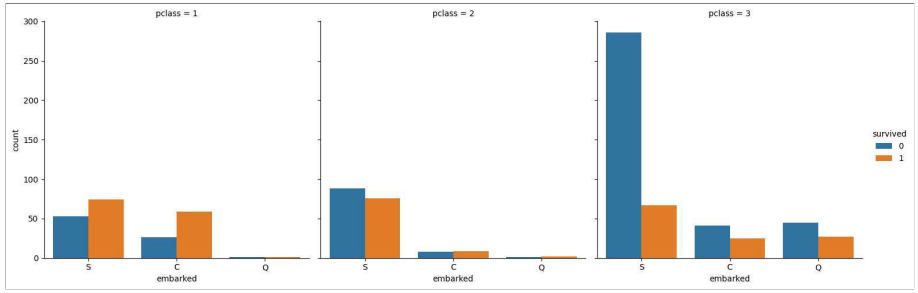


In [ ]:

```
sns.catplot(x="embarked", kind="count", col="pclass", hue="survived", data=df)
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x7b156369df10>
```



In [ ]:

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

In [ ]:

```
df.head(3)
```

Out[ ]:

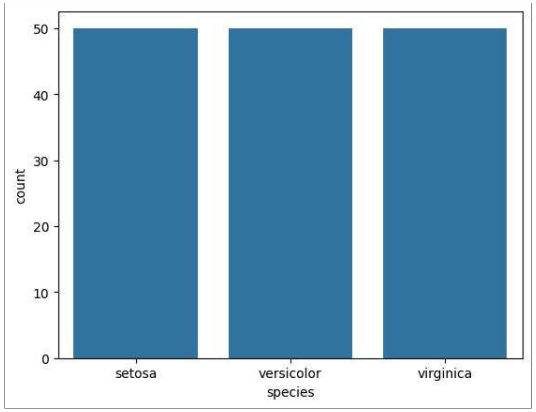
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa

In [ ]:

```
sns.countplot(x="species", data=df)
```

Out[ ]:

```
<Axes: xlabel='species', ylabel='count'>
```

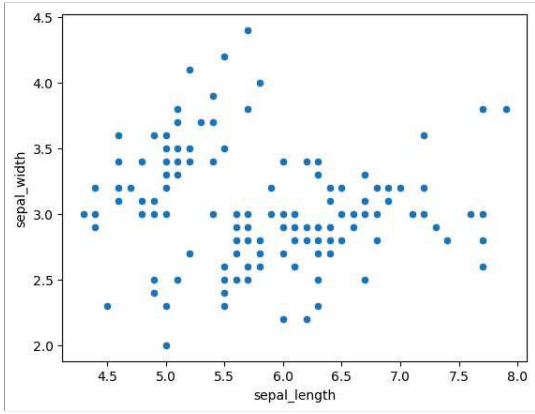


In [ ]:

```
sns.scatterplot(x="sepal_length", y="sepal_width", data=df)
```

Out[ ]:

```
<Axes: xlabel='sepal_length', ylabel='sepal_width'>
```

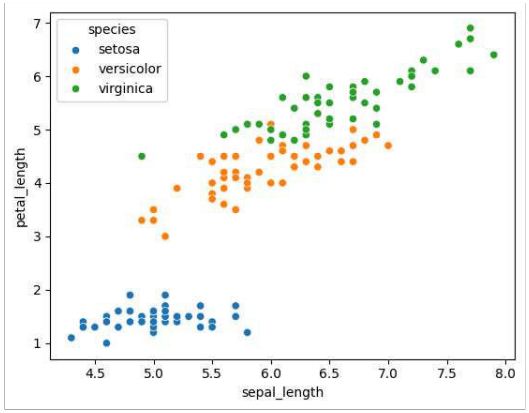


In [ ]:

```
sns.scatterplot(x="sepal_length", y="petal_length", hue= "species", data=df)
```

Out[ ]:

```
<Axes: xlabel='sepal_length', ylabel='petal_length'>
```

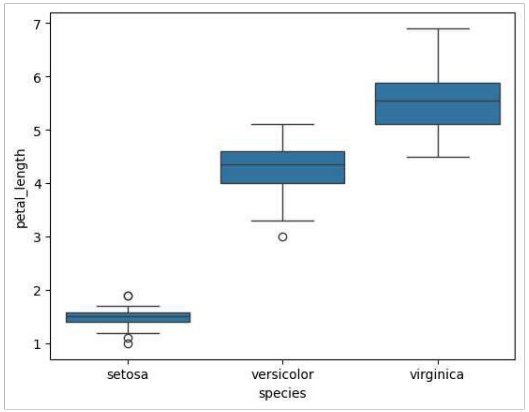


In [ ]:

```
sns.boxplot(x="species", y="petal_length", data=df)
```

Out[ ]:

```
<Axes: xlabel='species', ylabel='petal_length'>
```

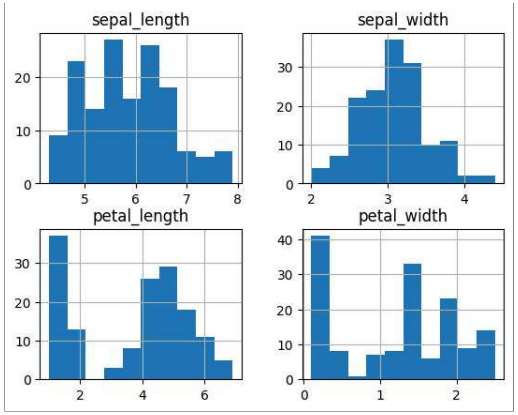


In [ ]:

```
df.hist()
```

Out[ ]:

```
array([[<Axes: title={'center': 'sepal_length'}>,
        <Axes: title={'center': 'sepal_width'}>],
       [<Axes: title={'center': 'petal_length'}>,
        <Axes: title={'center': 'petal_width'}>]], dtype=object)
```

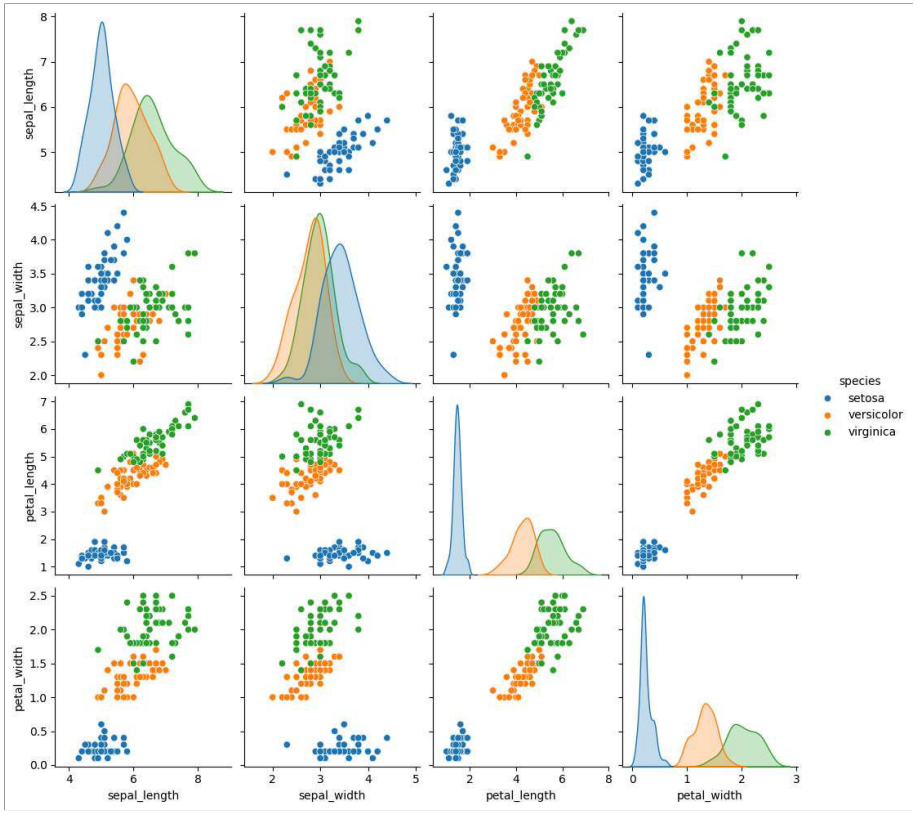


In [ ]:

```
sns.pairplot(data=df, hue="species")
```

Out[ ]:

```
<seaborn.axisgrid.PairGrid at 0x7b1563262090>
```



In [ ]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn. preprocessing import StandardScaler, MinMaxScaler, Normalizer
from sklearn import datasets
# importing data
iris=datasets.load_iris()
print(iris)
X = iris.data # array for the features
y = iris.target # array for the target
feature_names = iris.feature_names # feature names
target_names= iris.target_names
# target names
# spliting the data into training and testing data sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=2020)
X_train
# printing out train and test sets
print('X_train : ')
print(X_train)
print('')
print('X_test : ')
print(X_test)
print('')
print('y_train : ')
print(y_train)
print('')
print('y_test : ')
print(y_test)
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
       [6.3, 3.3, 4.7, 1.6],
       [4.9, 2.4, 3.3, 1. ],
       [6.6, 2.9, 4.6, 1.3],
       [5.2, 2.7, 3.9, 1.4],
       [5. , 2. , 3.5, 1. ],
       [5.9, 3. , 4.2, 1.5],
       [6. , 2.2, 4. , 1. ],
       [6.1, 2.9, 4.7, 1.4],
       [5.6, 2.9, 3.6, 1.3],
       [6.7, 3.1, 4.4, 1.4],
       [5.6, 3. , 4.5, 1.5],
       [5.8, 2.7, 4.1, 1. ],
       [6.2, 2.2, 4.5, 1.5],
       [5.6, 2.5, 3.9, 1.1],
       [5.9, 3.2, 4.8, 1.8],
       [6.1, 2.8, 4. , 1.3],
       [6.3, 2.5, 4.9, 1.5],
       [6.1, 2.8, 4.7, 1.2],
       [6.4, 2.9, 4.3, 1.3],
       [6.6, 3. , 4.4, 1.4],
       [6.8, 2.8, 4.8, 1.4],
       [6.7, 3. , 5. , 1.7],
       [6. , 2.9, 4.5, 1.5],
       [5.7, 2.6, 3.5, 1. ],
       [5.5, 2.4, 3.8, 1.1],
       [5.5, 2.4, 3.7, 1. ],
       [5.8, 2.7, 3.9, 1.2],
       [6. , 2.7, 5.1, 1.6],
       [5.4, 3. , 4.5, 1.5],
       [6. , 3.4, 4.5, 1.6],
       [6.7, 3.1, 4.7, 1.5],
       [6.3, 2.3, 4.4, 1.3],
       [5.6, 3. , 4.1, 1.3],
       [5.5, 2.5, 4. , 1.3],
       [5.5, 2.6, 4.4, 1.2],
       [6.1, 3. , 4.6, 1.4],
       [5.8, 2.6, 4. , 1.2],
       [5. , 2.3, 3.3, 1. ],
       [5.6, 2.7, 4.2, 1.3],
       [5.7, 3. , 4.2, 1.2],
       [5.7, 2.9, 4.2, 1.3],
       [6.2, 2.9, 4.3, 1.3],
       [5.1, 2.5, 3. , 1.1],
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
```

```
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]), 'frame': None, 'target_
names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'), 'DESCR': '.. _iris
_dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set Characteristics:*
*\n\n:Number of Instances: 150 (50 in each of three classes)\n:Number of Attributes: 4
numeric, predictive attributes and the class\n:Attribute Information:\n    - sepal len
gth in cm\n    - sepal width in cm\n    - petal length in cm\n    - petal width in cm
\n    - class:\n            - Iris-Setosa\n            - Iris-Versicolour\n
- Iris-Virginica\n\n:Summary Statistics:\n\n============== ==== ==== ======= ===== ===
==================\n                Min  Max   Mean    SD   Class Correlation\n========
====== ==== ==== ======= ===== ====================\nsepal length:   4.3  7.9   5.84
0.83    0.7826\nsepal width:    2.0  4.4   3.05   0.43   -0.4194\npetal length:   1.0
6.9   3.76   1.76    0.9490  (high!)\npetal width:    0.1  2.5   1.20   0.76    0.9565
(high!)\n============== ==== ==== ======= ===== ====================\n\n:Missing Attri
bute Values: None\n:Class Distribution: 33.3% for each of 3 classes.\n:Creator: R.A. F
isher\n:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n:Date: July, 1988\n\nT
he famous Iris database, first used by Sir R.A. Fisher. The dataset is taken\nfrom Fis
her\'s paper. Note that it\'s the same as in R, but not as in the UCI\nMachine Learnin
g Repository, which has two wrong data points.\n\nThis is perhaps the best known datab
ase to be found in the\npattern recognition literature.  Fisher\'s paper is a classic
in the field and\nis referenced frequently to this day.  (See Duda & Hart, for exampl
e.)  The\ndata set contains 3 classes of 50 instances each, where each class refers to
a\ntype of iris plant.  One class is linearly separable from the other 2; the\nlatter
are NOT linearly separable from each other.\n\n.. dropdown:: References\n\n  - Fisher,
R.A. "The use of multiple measurements in taxonomic problems"\n    Annual Eugenics, 7,
Part II, 179-188 (1936); also in "Contributions to\n    Mathematical Statistics" (John
Wiley, NY, 1950).\n  - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scen
e Analysis.\n    (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n
- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n    Structure
and Classification Rule for Recognition in Partially Exposed\n    Environments".  IEEE
Transactions on Pattern Analysis and Machine\n    Intelligence, Vol. PAMI-2, No. 1, 67
-71.\n  - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n
on Information Theory, May 1972, 431-433.\n  - See also: 1988 MLC Proceedings, 54-64.
Cheeseman et al"s AUTOCLASS II\n    conceptual clustering system finds 3 classes in th
e data.\n  - Many, many more ...\n', 'feature_names': ['sepal length (cm)', 'sepal wid
th (cm)', 'petal length (cm)', 'petal width (cm)'], 'filename': 'iris.csv', 'data_modu
le': 'sklearn.datasets.data'}
X_train :
[[7.6 3.  6.6 2.1]
 [5.  3.5 1.3 0.3]
 [5.  3.4 1.5 0.2]
 [5.1 3.5 1.4 0.3]
 [6.4 3.2 5.3 2.3]
 [4.9 3.1 1.5 0.2]
 [5.  3.3 1.4 0.2]
 [5.5 2.4 3.7 1. ]
 [5.  3.2 1.2 0.2]
 [5.7 2.5 5.  2. ]
 [5.2 3.5 1.5 0.2]
 [6.7 3.3 5.7 2.5]
 [5.7 3.  4.2 1.2]
 [5.7 3.8 1.7 0.3]
 [6.2 2.9 4.3 1.3]
 [6.9 3.2 5.7 2.3]
 [7.7 3.8 6.7 2.2]
 [6.8 2.8 4.8 1.4]
 [6.7 3.1 4.7 1.5]
 [4.9 2.4 3.3 1. ]
 [5.1 3.8 1.9 0.4]
 [6.3 2.5 5.  1.9]
 [5.5 2.5 4.  1.3]
 [5.5 3.5 1.3 0.2]
 [6.3 3.3 4.7 1.6]
 [5.8 2.6 4.  1.2]
 [6.4 3.1 5.5 1.8]
 [6.4 2.9 4.3 1.3]
 [5.7 4.4 1.5 0.4]
 [6.3 2.3 4.4 1.3]
 [5.2 4.1 1.5 0.1]
 [4.8 3.  1.4 0.1]
 [5.1 3.5 1.4 0.2]
 [5.2 2.7 3.9 1.4]
 [5.7 2.8 4.1 1.3]
 [7.  3.2 4.7 1.4]
 [5.1 3.4 1.5 0.2]
 [5.1 3.8 1.5 0.3]
 [6.2 3.4 5.4 2.3]
 [6.6 2.9 4.6 1.3]
 [6.4 2.8 5.6 2.2]
 [5.4 3.9 1.3 0.4]
 [4.9 3.6 1.4 0.1]
 [6.3 2.9 5.6 1.8]
 [4.6 3.2 1.4 0.2]
 [7.4 2.8 6.1 1.9]
 [7.2 3.6 6.1 2.5]
 [5.  3.  1.6 0.2]
 [5.6 2.8 4.9 2. ]
 [4.8 3.4 1.9 0.2]
 [4.3 3.  1.1 0.1]
 [5.  2.  3.5 1. ]
 [5.  3.4 1.6 0.4]
 [5.1 3.8 1.6 0.2]
 [7.7 2.8 6.7 2. ]
 [5.4 3.  4.5 1.5]
 [4.8 3.1 1.6 0.2]
 [6.8 3.2 5.9 2.3]
 [5.6 2.9 3.6 1.3]
 [5.8 2.7 5.1 1.9]
 [5.4 3.7 1.5 0.2]
 [6.7 2.5 5.8 1.8]
 [6.3 2.7 4.9 1.8]
 [5.4 3.9 1.7 0.4]
 [6.6 3.  4.4 1.4]
 [5.9 3.  5.1 1.8]
 [5.8 4.  1.2 0.2]
 [6.7 3.  5.2 2.3]
 [5.6 2.7 4.2 1.3]
 [6.3 2.5 4.9 1.5]
 [7.3 2.9 6.3 1.8]
 [5.5 2.4 3.8 1.1]
 [5.5 4.2 1.4 0.2]
 [6.8 3.  5.5 2.1]
 [5.6 3.  4.5 1.5]
 [4.4 3.2 1.3 0.2]
 [6.7 3.1 4.4 1.4]
 [5.8 2.7 3.9 1.2]
 [6.  2.7 5.1 1.6]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 4.9 1.5]
 [4.9 3.  1.4 0.2]
 [6.  3.  4.8 1.8]
 [5.4 3.4 1.7 0.2]
 [4.6 3.1 1.5 0.2]
 [5.1 2.5 3.  1.1]
```

```
 [6.3 3.3 6.  2.5]
 [6.5 3.2 5.1 2. ]
 [6.7 3.3 5.7 2.1]
 [6.2 2.2 4.5 1.5]
 [6.1 3.  4.9 1.8]
 [6.1 2.9 4.7 1.4]
 [4.6 3.6 1.  0.2]
 [6.3 2.8 5.1 1.5]
 [4.6 3.4 1.4 0.3]
 [5.5 2.6 4.4 1.2]
 [4.7 3.2 1.6 0.2]
 [5.1 3.7 1.5 0.4]
 [6.  2.2 4.  1. ]
 [6.1 2.8 4.7 1.2]
 [7.7 3.  6.1 2.3]
 [5.7 2.6 3.5 1. ]
 [7.2 3.  5.8 1.6]
 [4.4 3.  1.3 0.2]
 [6.9 3.1 5.1 2.3]
 [5.6 2.5 3.9 1.1]
 [7.1 3.  5.9 2.1]
 [5.8 2.8 5.1 2.4]
 [5.7 2.8 4.5 1.3]
 [6.9 3.1 5.4 2.1]
 [4.9 3.1 1.5 0.1]
 [5.3 3.7 1.5 0.2]
 [6.1 2.8 4.  1.3]
 [7.9 3.8 6.4 2. ]
 [7.2 3.2 6.  1.8]
 [6.1 3.  4.6 1.4]
 [7.7 2.6 6.9 2.3]
 [5.8 2.7 4.1 1. ]
 [6.3 3.4 5.6 2.4]
 [5.7 2.9 4.2 1.3]]

X_test :
[[6.5 3.  5.8 2.2]
 [4.4 2.9 1.4 0.2]
 [5.9 3.  4.2 1.5]
 [6.5 2.8 4.6 1.5]
 [6.  2.9 4.5 1.5]
 [6.  2.2 5.  1.5]
 [6.4 2.7 5.3 1.9]
 [5.  2.3 3.3 1. ]
 [5.1 3.3 1.7 0.5]
 [5.4 3.4 1.5 0.4]
 [6.5 3.  5.5 1.8]
 [4.9 2.5 4.5 1.7]
 [5.2 3.4 1.4 0.2]
 [6.2 2.8 4.8 1.8]
 [5.8 2.7 5.1 1.9]
 [5.  3.6 1.4 0.2]
 [5.5 2.3 4.  1.3]
 [6.4 3.2 4.5 1.5]
 [6.4 2.8 5.6 2.1]
 [4.5 2.3 1.3 0.3]
 [4.7 3.2 1.3 0.2]
 [6.1 2.6 5.6 1.4]
 [5.9 3.2 4.8 1.8]
 [4.8 3.4 1.6 0.2]
 [6.5 3.  5.2 2. ]
 [5.6 3.  4.1 1.3]
 [6.  3.4 4.5 1.6]
 [6.7 3.  5.  1.7]
 [4.8 3.  1.4 0.3]
 [5.  3.5 1.6 0.6]]

y_train :
[2 0 0 0 2 0 0 1 0 2 0 2 1 0 1 2 2 1 1 1 0 2 1 0 1 1 2 1 0 1 0 0 0 1 1 1 0
 0 2 1 2 0 0 2 0 2 2 0 2 0 0 1 0 0 2 1 0 2 1 2 0 2 2 0 1 2 0 2 1 1 2 1 0 2
 1 0 1 1 1 2 1 0 2 0 0 1 2 2 2 1 2 1 0 2 0 1 0 0 1 1 2 1 2 0 2 1 2 2 1 2 0
 0 1 2 2 1 2 1 2 1]

y_test :
[2 0 1 1 1 2 2 1 0 0 2 2 0 2 2 0 1 1 2 0 0 2 1 0 2 1 1 1 0 0]
```

In [ ]:

```
# Load the dataset
import pandas as pd
df = pd.read_csv("https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pages/data/iris.csv")
df.head(5)
```

Out[ ]:

	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

In [ ]:

```
Q1= np.percentile (df['sepal_width'], 25, interpolation = 'midpoint')
Q1
```

Out[ ]:

```
2.8
```

In [ ]:

```
Q3= np.percentile (df['sepal_width'], 75, interpolation = 'midpoint')
Q3
```

Out[ ]:

```
3.3
```

In [ ]:

```
IQR=Q3-Q1
IQR
```

Out[ ]:

```
0.5
```

In [ ]:

```
print("Old Shape: ", df.shape)
```

```
Old Shape:  (150, 5)
```

In [ ]:

```
#Upper bound
upper = np. where (df['sepal_width'] >= (Q3+1.5*IQR))
# Lower bound
lower = np.where(df['sepal_width'] <= (Q1-1.5*IQR))
```

```
#Removing the Outliers
df.drop(upper[0], inplace=True)
df.drop(lower[0], inplace=True)
```

In [ ]:

```
print("New Shape: ", df.shape)
```

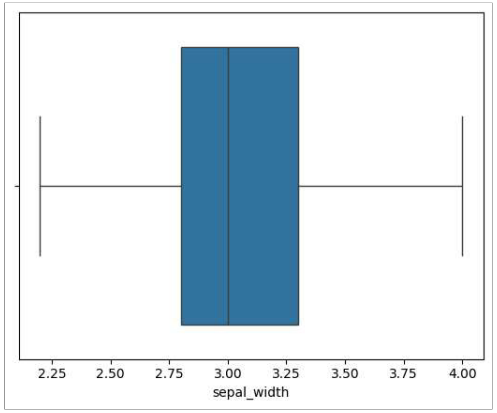
```
New Shape:  (146, 5)
```

In [ ]:

```
sns.boxplot(x='sepal_width', data=df)
```

Out[ ]:

```
<Axes: xlabel='sepal_width'>
```

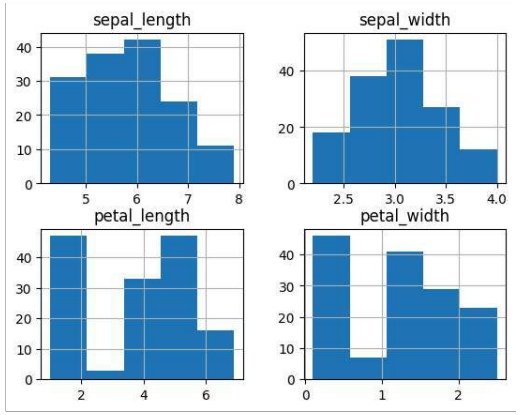


In [ ]:

```
df.hist(bins=5)
```

Out[ ]:

```
array([[<Axes: title={'center': 'sepal_length'}>,
        <Axes: title={'center': 'sepal_width'}>],
       [<Axes: title={'center': 'petal_length'}>,
        <Axes: title={'center': 'petal_width'}>]], dtype=object)
```

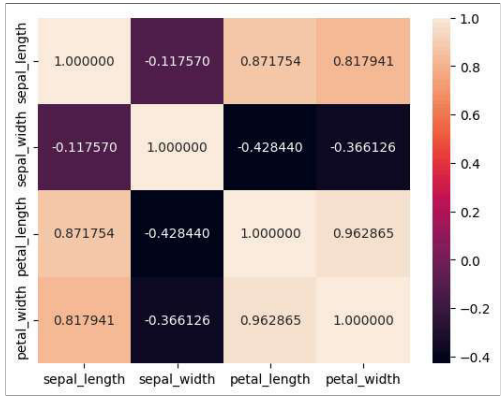


In [ ]:

```
import seaborn as sns
sns.heatmap(df.corr(),annot=True,fmt="f")
```

Out[ ]:

```
<Axes: >
```



In [ ]:

```
import pandas as pd
#create data frame
df = pd.DataFrame({'y': [8, 12, 15, 14, 19, 23, 25, 29],
'x1': [5, 7, 7, 9, 12, 9, 9, 4], 'x2':[11, 8, 10, 6, 6, 5, 9, 12],'x3':[2, 2, 3, 2, 5, 5, 7, 9]})
#view data frame
df
```

Out[ ]:

	y	x1	x2	x3
0	8	5	11	2
1	12	7	8	2
2	15	7	10	3
3	14	9	6	2
4	19	12	6	5
5	23	9	5	5
6	25	9	9	7
7	29	4	12	9

In [ ]:

```
#standardize the values in each column, rescaling of the data (0-1) #mean, std
#v' = v
df_new = (df-df.mean())/df.std()
#view new data frame
print(df_new)
```

```
          y        x1        x2        x3
0 -1.418032 -1.078639  1.025393 -0.908151
1 -0.857822 -0.294174 -0.146485 -0.908151
2 -0.437664 -0.294174  0.634767 -0.525772
3 -0.577717  0.490290 -0.927736 -0.908151
4  0.122546  1.666987 -0.927736  0.238987
5  0.682756  0.490290 -1.318362  0.238987
6  0.962861  0.490290  0.244141  1.003746
7  1.523071 -1.470871  1.416019  1.768505
```

In [ ]:

```
print(df_new.mean())
print(df_new.std())
```

```
y     0.000000e+00
x1    2.775558e-17
x2   -1.387779e-17
x3    5.551115e-17
dtype: float64
y     1.0
x1    1.0
x2    1.0
x3    1.0
dtype: float64
```

In [ ]:

```
#define predictor variable columns
df_x=df[['x1', 'x2', 'x3']]
#standardize the values for each predictor variable
df[['x1', 'x2', 'x3']] = (df_x-df_x.mean())/df_x.std()
df
```

Out[ ]:

	y	x1	x2	x3
0	8	-1.078639	1.025393	-0.908151
1	12	-0.294174	-0.146485	-0.908151
2	15	-0.294174	0.634767	-0.525772
3	14	0.490290	-0.927736	-0.908151
4	19	1.666987	-0.927736	0.238987
5	23	0.490290	-1.318362	0.238987
6	25	0.490290	0.244141	1.003746
7	29	-1.470871	1.416019	1.768505

In [ ]:

```
import pandas as pd
#create DataFrame
df = pd.DataFrame({'
points': [25, 12, 15, 14, 19],
'assists': [5, 7, 7, 9, 12],
'rebounds':[11, 8, 10, 6, 6]})
df
```

Out[ ]:

	points	assists	rebounds
0	25	5	11
1	12	7	8
2	15	7	10
3	14	9	6
4	19	12	6

In [ ]:

```
(df-df.min())/(df.max()-df.min())
```

Out[ ]:

	points	assists	rebounds
0	1.000000	0.000000	1.0
1	0.000000	0.285714	0.4
2	0.230769	0.285714	0.8
3	0.153846	0.571429	0.0
4	0.538462	1.000000	0.0

In [ ]:

```
import pandas as pd
df = pd.read_csv("https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pages/data/iris.csv")
pd.crosstab(index=df['species'], columns=df['sepal_length'], margins = True)
```

Out[ ]:

sepal_length	4.3	4.4	4.5	4.6	4.7	4.8	4.9	5.0	5.1	5.2	...	6.9	7.0	7.1	7.2	7.3	7.4	7.6	7.7	7.9
species																				
setosa	1	3	1	4	2	5	4	8	8	3	...	0	0	0	0	0	0	0	0	0
versicolor	0	0	0	0	0	0	1	2	1	1	...	1	1	0	0	0	0	0	0	0
virginica	0	0	0	0	0	0	1	0	0	0	...	3	0	1	3	1	1	1	4	1
All	1	3	1	4	2	5	6	10	9	4	...	4	1	1	3	1	1	1	4	1

4 rows × 36 columns

In [ ]:

```
normZ=StandardScaler()
X_train_Z = normZ.fit_transform(X_train)
X_test_Z = normZ.transform(X_test)
X_train_Z.mean (axis=0)
X_train_Z.std (axis=0)
X_test_Z.mean (axis=0)
X_test_Z.std(axis= 0)
```

Out[ ]:

```
array([0.83359705, 0.85365417, 1.00126802, 0.9868196 ])
```

In [ ]:

```
#normalization using min_max method
normMinMax = MinMaxScaler()
X_train_MinMax = normMinMax.fit_transform(X_train)
X_test_MinMax = normMinMax.transform (X_test)
print(X_train_MinMax.min(axis=0))
print(X_train_MinMax.max(axis=0))
print(X_test_MinMax.max(axis=0))
print (X_test_MinMax.min(axis=0))
```

```
[0. 0. 0. 0.]
[1. 1. 1. 1.]
[0.66666667 0.66666667 0.81355932 0.875     ]
[0.02777778 0.08333333 0.05084746 0.04166667]
```

In [ ]:

```
normalizer = Normalizer()
X_train_normalized = normalizer.fit_transform(X_train)
X_test_normalized = normalizer.transform(X_test)
# Display first 5 rows of the normalized training data
print(pd.DataFrame(X_train_normalized[:5], columns=iris.feature_names))
```

```
First 5 rows of normalized training data:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0           0.709537          0.280080           0.616177          0.196056
1           0.800333          0.560233           0.208087          0.048020
2           0.802185          0.545486           0.240655          0.032087
3           0.803274          0.551267           0.220507          0.047251
4           0.695899          0.347949           0.576291          0.250089
```