Assignment 1

ECE657A

XIYAO LIU

20483947

Load Datasets ¶

In [22]:
```
from sklearn import datasets
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data = load_iris()
data.keys()
```

Out[22]:
```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

In [23]:
```
data.data
```

Out[23]:
```
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]])
```

In [24]:
```
data.target
```

Out[24]:
```
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, 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])
```

In [25]:
```
data.target_names
```

Out[25]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [26]:
```
data.feature_names
```

Out[26]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

Dataframe

In [27]:
```
def toDataframe():

    columns = np.append(data.feature_names,['target'])
    bodydata = np.append(data.data, data.target[:,None], axis=1)
    index = pd.RangeIndex(start=1, stop=len(data.data)+1, step=1)

    return pd.DataFrame(columns=columns, data=bodydata, index=index)


data_df=toDataframe()
data_df
```

Out[27]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
1	5.1	3.5	1.4	0.2	0.0
2	4.9	3.0	1.4	0.2	0.0
3	4.7	3.2	1.3	0.2	0.0
4	4.6	3.1	1.5	0.2	0.0
5	5.0	3.6	1.4	0.2	0.0
6	5.4	3.9	1.7	0.4	0.0
7	4.6	3.4	1.4	0.3	0.0
8	5.0	3.4	1.5	0.2	0.0
9	4.4	2.9	1.4	0.2	0.0
10	4.9	3.1	1.5	0.1	0.0
11	5.4	3.7	1.5	0.2	0.0
12	4.8	3.4	1.6	0.2	0.0
13	4.8	3.0	1.4	0.1	0.0
14	4.3	3.0	1.1	0.1	0.0
15	5.8	4.0	1.2	0.2	0.0
16	5.7	4.4	1.5	0.4	0.0
17	5.4	3.9	1.3	0.4	0.0
18	5.1	3.5	1.4	0.3	0.0
19	5.7	3.8	1.7	0.3	0.0
20	5.1	3.8	1.5	0.3	0.0
21	5.4	3.4	1.7	0.2	0.0
22	5.1	3.7	1.5	0.4	0.0
23	4.6	3.6	1.0	0.2	0.0
24	5.1	3.3	1.7	0.5	0.0
25	4.8	3.4	1.9	0.2	0.0
26	5.0	3.0	1.6	0.2	0.0
27	5.0	3.4	1.6	0.4	0.0
28	5.2	3.5	1.5	0.2	0.0
29	5.2	3.4	1.4	0.2	0.0
30	4.7	3.2	1.6	0.2	0.0
...	...	...	...	...	...
121	6.9	3.2	5.7	2.3	2.0
122	5.6	2.8	4.9	2.0	2.0
123	7.7	2.8	6.7	2.0	2.0
124	6.3	2.7	4.9	1.8	2.0

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
125	6.7	3.3	5.7	2.1	2.0
126	7.2	3.2	6.0	1.8	2.0
127	6.2	2.8	4.8	1.8	2.0
128	6.1	3.0	4.9	1.8	2.0
129	6.4	2.8	5.6	2.1	2.0
130	7.2	3.0	5.8	1.6	2.0
131	7.4	2.8	6.1	1.9	2.0
132	7.9	3.8	6.4	2.0	2.0
133	6.4	2.8	5.6	2.2	2.0
134	6.3	2.8	5.1	1.5	2.0
135	6.1	2.6	5.6	1.4	2.0
136	7.7	3.0	6.1	2.3	2.0
137	6.3	3.4	5.6	2.4	2.0
138	6.4	3.1	5.5	1.8	2.0
139	6.0	3.0	4.8	1.8	2.0
140	6.9	3.1	5.4	2.1	2.0
141	6.7	3.1	5.6	2.4	2.0
142	6.9	3.1	5.1	2.3	2.0
143	5.8	2.7	5.1	1.9	2.0
144	6.8	3.2	5.9	2.3	2.0
145	6.7	3.3	5.7	2.5	2.0
146	6.7	3.0	5.2	2.3	2.0
147	6.3	2.5	5.0	1.9	2.0
148	6.5	3.0	5.2	2.0	2.0
149	6.2	3.4	5.4	2.3	2.0
150	5.9	3.0	5.1	1.8	2.0

150 rows × 5 columns

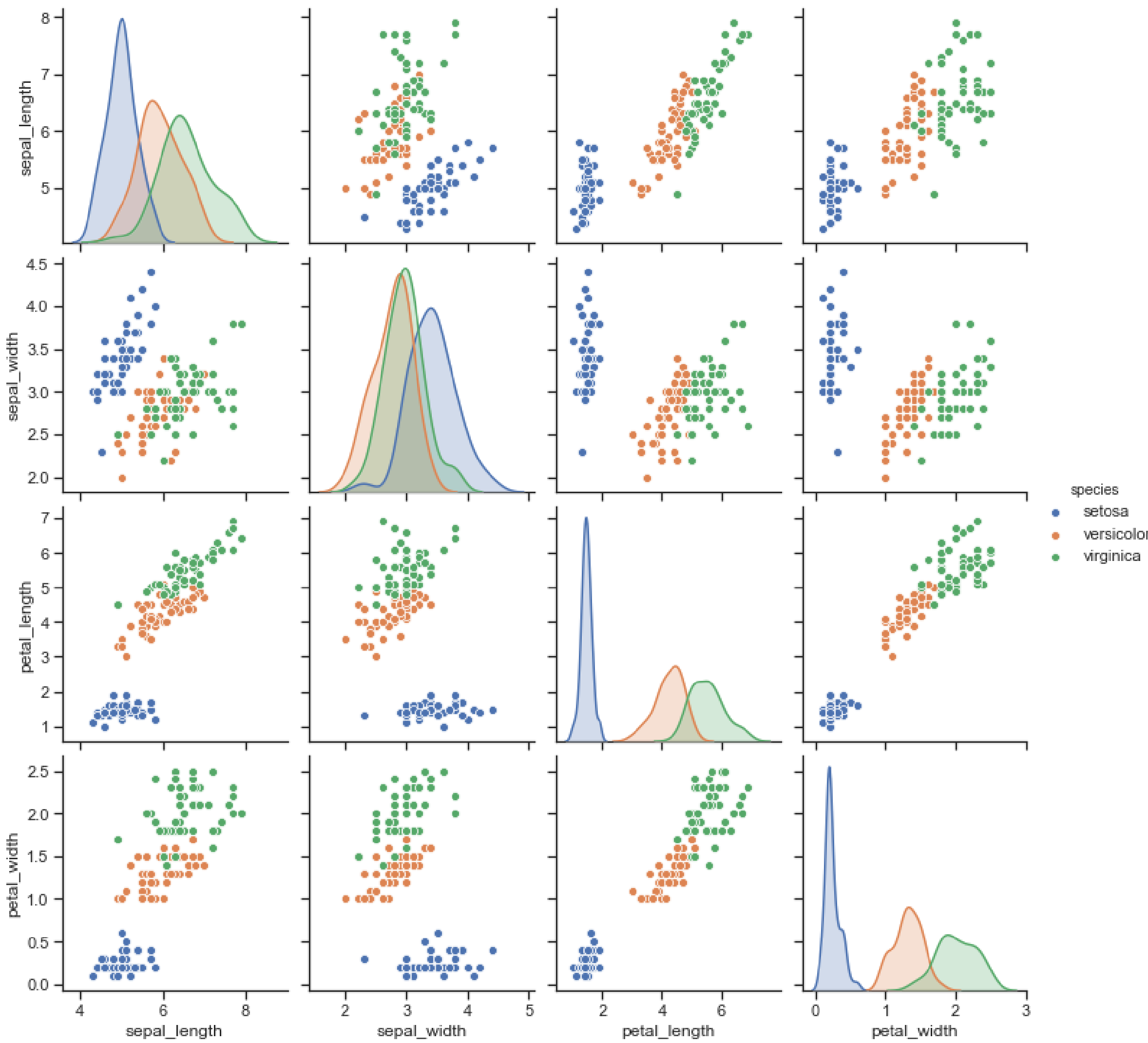
Question1

In [28]:
```
#Question1

sns.set(style="ticks")

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

Out[28]: <seaborn.axisgrid.PairGrid at 0x10ffba70>



Interpretation of the separability of the three classes in terms of different features (dimensions).

The data set contains 150 records in 3 categories, each with 50 pieces of data. Each record has 4 characteristics: sepal length, sepal width, petal length, and petal width. These 4 characteristics can be used to predict the iris flower belongs to which specific type such iris-setosa, iris-versicolor and iris-virginica. If features with objects belonging to the same category such as all setosa have similar values, we can find well-defined classfication in the data visualization. When the sepal length is fixed, the three types of Iris flowers are clearly distinguished in terms of petal width and petal length, although Versicolor and Virginica are slightly mixed. In particular, setosa's petal width and petal length are both significantly smaller than the other two. Secondly, Versicolor's petal width and petal length are smaller than those of Virginica. In the second scatter plot in the leftmost column, it can be seen that Versicolor and Virginiana are mixed with more points, and the distribution of Setosa is clear and obvious. When the sepal width is fixed, the three types of Iris flowers are well classified in terms of petal width and petal length. The distribution slightly worse on sepal length feature. Although Versicolor and Virginica have mixed separations, it is still very easy to distinguish different types of flowers. When the petal length is fixed, The three categories of Iris flowers are clearly distinguished in terms of sepal length, sepal width and petal width. Setosa's sepal width is slightly higher than the other two, and its sepal length is smaller than the other two. When the petal width is fixed, the three types of Iris flowers are clearly classified in three aspects: sepal length, sepal width, and petal length as well. Even Versicolor and Virginica still have few data points mixed together. The overall distribution is still obvious. These graphs show that a classifier trained using these functions may learn to classify various flower types reasonably.

Question 2: KNN

In [29]:
```
#Question 2: KNN

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

#1. First, divide the data into train, validation, and test sets (60%, 20%, 20%)

X_train, X_test, y_train, y_test= train_test_split(data.data,data.target, test_size=0.2
, random_state=42)

X_train, X_val, y_train, y_val= train_test_split(X_train,y_train, test_size=0.25, rando
m_state=42)
```

In [30]:
```
#2. Train the model with each classifier' s default parameters. Use the train set and te
st the model on the test set. Store the accuracy of the model.
knn = KNeighborsClassifier(n_neighbors = 5) #5 is the default value

#Fit the model using X_train as training data and y_train as target values
knn.fit(X_train, y_train)
#Accuracy classification score
Accuracy_score_5=knn.score(X_test, y_test)
Accuracy_score_5
```

Out[30]:
```
0.9666666666666667
```