In [1]:
```
# Import all necessary packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

from sklearn.model_selection import GridSearchCV

%matplotlib inline
```

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

In [3]:
```
iris.keys()
```

Out[3]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [4]:
```
iris.head()
```

Out[4]:

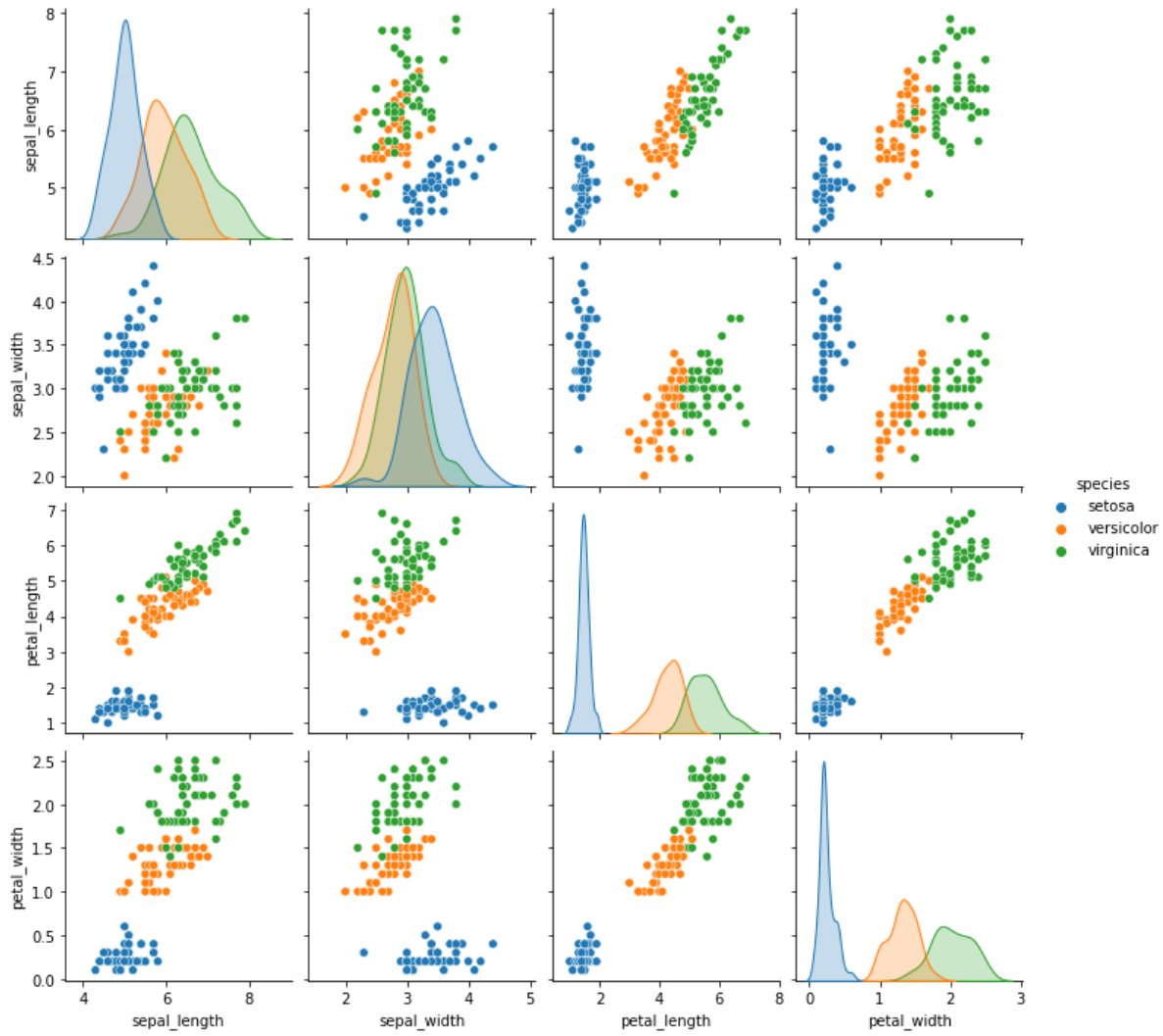
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [5]:
```
iris.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 [12]:
```
sns.pairplot(iris,hue = 'species')
```

Out[12]: <seaborn.axisgrid.PairGrid at 0x1f46c5492e0>



In [37]:
```
x=iris.iloc[:,:-1]
y=iris.iloc[:,4]
X_train, X_test, Y_train, Y_test = train_test_split(x,y, test_size = 0.3, randc
```

In [28]:
```
model = SVC()
```

In [29]:
```
model.fit(X_train, Y_train)
```

Out[29]: SVC()

In [31]:
```
predict = model.predict(X_test)
```

In [38]:
```
print(classification_report(Y_test, predict))
```

```
              precision    recall  f1-score   support

      setosa       0.35      0.50      0.41        14
  versicolor       0.35      0.46      0.40        13
   virginica       0.62      0.28      0.38        18

    accuracy                           0.40        45
   macro avg       0.44      0.41      0.40        45
weighted avg       0.46      0.40      0.40        45
```

In [39]:
```
print(confusion_matrix(Y_test, predict))
```

```
[[7 5 2]
 [6 6 1]
 [7 6 5]]
```

In [40]:
```
parameter_grid = {'C': [0.1,1,10,100,1000], 'gamma':[1,0.1,0.01,0.001,0.0001]}
```

In [41]:
```
grid = GridSearchCV(SVC(), parameter_grid, verbose = 5)
```

In [42]:
```
grid.fit(X_train, Y_train)
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END ....................C=0.1, gamma=1;, score=0.952 total time=
0.0s
[CV 2/5] END ....................C=0.1, gamma=1;, score=0.905 total time=
0.0s
[CV 3/5] END ....................C=0.1, gamma=1;, score=0.857 total time=
0.0s
[CV 4/5] END ....................C=0.1, gamma=1;, score=0.905 total time=
0.0s
[CV 5/5] END ....................C=0.1, gamma=1;, score=1.000 total time=
0.0s
[CV 1/5] END ..................C=0.1, gamma=0.1;, score=0.857 total time=
0.0s
[CV 2/5] END ..................C=0.1, gamma=0.1;, score=0.905 total time=
0.0s
[CV 3/5] END ..................C=0.1, gamma=0.1;, score=0.857 total time=
0.0s
[CV 4/5] END ..................C=0.1, gamma=0.1;, score=0.810 total time=
0.0s
```

In [43]:
```
grid.best_params_
```

Out[43]:
```
{'C': 10, 'gamma': 0.1}
```

In [44]:
```
grid_prediction = grid.predict(X_test)
```

In [45]:
```
print(confusion_matrix(Y_test, grid_prediction))
```

```
[[14  0  0]
 [ 0 13  0]
 [ 0  0 18]]
```

In [46]:
```
print(classification_report(Y_test, grid_prediction))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        14
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        18

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

In [ ]: