Support Vector Machines

- very useful for classification problems
- using iris data

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
iris = sns.load_dataset('iris')
iris
```

Out[4]:

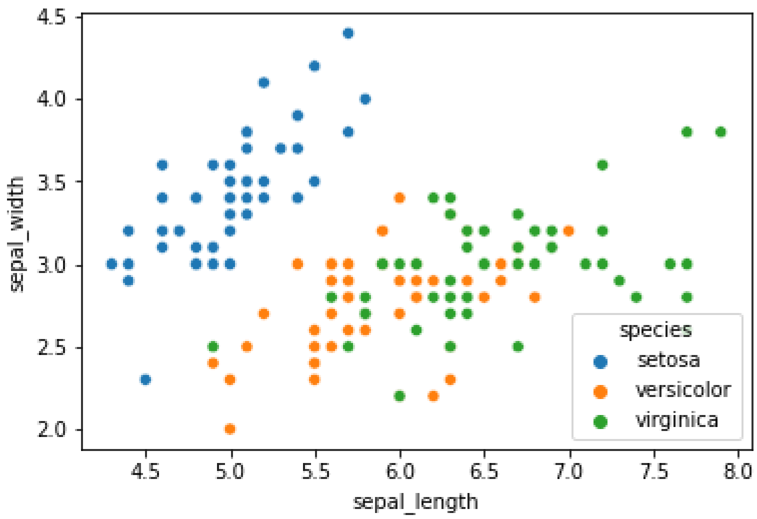
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

Visualization

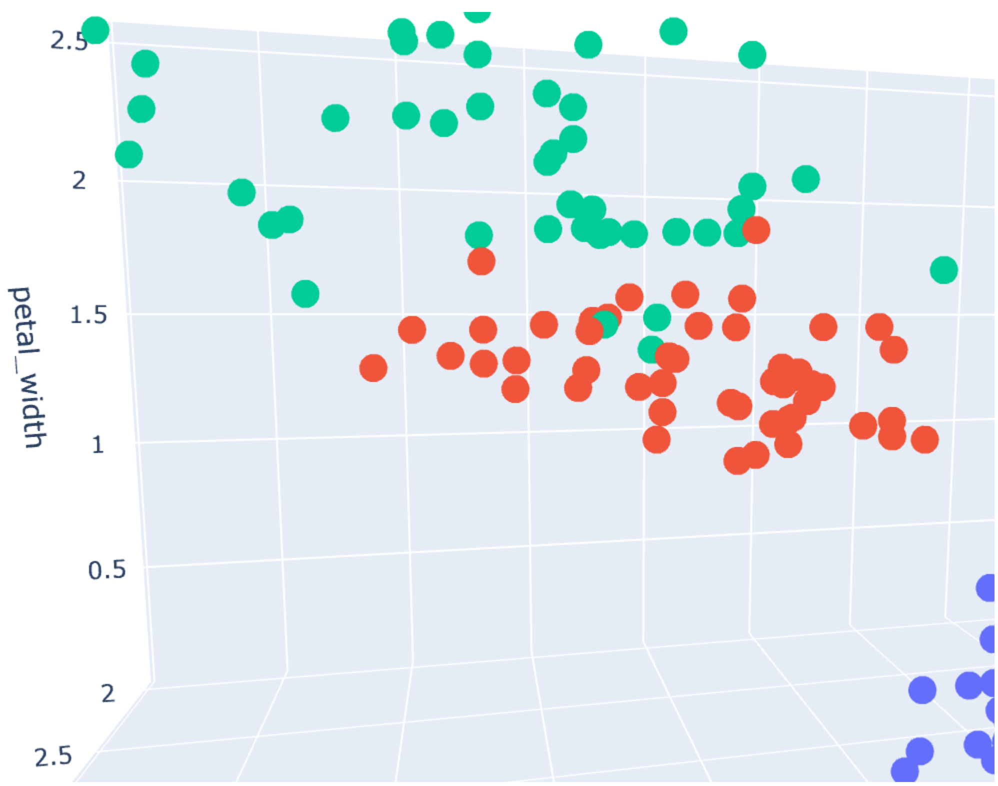
In [9]:
```
sns.scatterplot(x ='sepal_length', y = 'sepal_width', data = iris, hue = 'sp
```

Out[9]: <AxesSubplot:xlabel='sepal_length', ylabel='sepal_width'>



In [10]:
```
import plotly.express as px
```

In [14]:
```
fig = px.scatter_3d(iris, x='sepal_length', y='sepal_width', z='petal_width'
                    color='species')
fig.show()
```



Train-Test-Split

In [15]:
```
from sklearn.model_selection import train_test_split

X = iris.drop('species',axis = 1)
y = iris.species

X_train, X_test, y_train, y_test = train_test_split(X,y, random_state = 42,
```

Modelling

In [17]:
```
from sklearn.svm import SVC
```

In [19]:
```
svm_model = SVC()
svm_model.fit(X_train,y_train)
```

Out[19]:
```
▾ SVC
SVC()
```

Prediction

In [22]:
```
y_pred = svm_model.predict(X_test)
y_pred
```

Out[22]:
```
array(['versicolor', 'setosa', 'virginica', 'versicolor', 'versicolor',
       'setosa', 'versicolor', 'virginica', 'versicolor', 'versicolor',
       'virginica', 'setosa', 'setosa', 'setosa', 'setosa', 'versicolor',
       'virginica', 'versicolor', 'versicolor', 'virginica', 'setosa',
       'virginica', 'setosa', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'setosa', 'setosa'], dtype=object)
```

In [23]:
```
# test accuracy
svm_model.score(X_test, y_test)
```

Out[23]:
```
1.0
```

In [24]:
```
# train accuracy

svm_model.score(X_train, y_train)
```

Out[24]:
```
0.975
```

Hyperparameter Tuning of SVM Model

In [53]:
```
svm_model = SVC(C = 3, kernel = 'rbf', gamma = 5)
svm_model.fit(X_train,y_train)
print('Test Accuracy :', svm_model.score(X_test, y_test))
print('Train Accuracy :',svm_model.score(X_train, y_train))
```

```
Test Accuracy : 1.0
Train Accuracy : 1.0
```

Best values for SVM model is :

- C = 3
- kernel = 'rbf'
- gamma = 5

In [ ]:

```
1
```