# Support Vector Machine

January 8, 2021

## 0.1 Support Vector Machine

### 0.1.1 The Data

The Iris flower data set is being used here.

The iris dataset contains measurements for 150 iris flowers from three different species.

The three classes in the Iris dataset:

```
Iris-setosa (n=50)
Iris-versicolor (n=50)
Iris-virginica (n=50)
```

The four features of the Iris dataset:

```
sepal length in cm
sepal width in cm
petal length in cm
petal width in cm
```

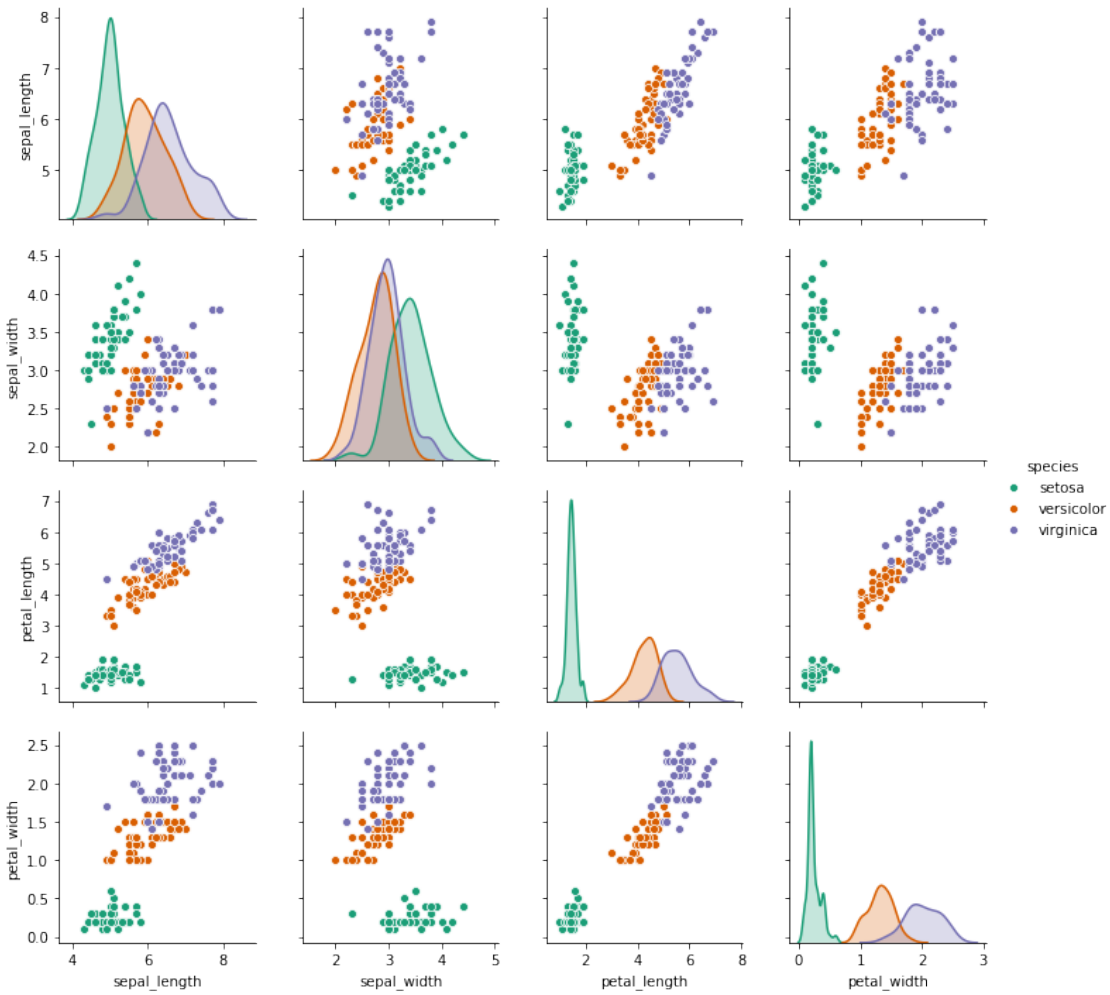
[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
iris = sns.load_dataset('iris')
```

### 0.1.2 Exploratory Data Analysis

Pairplot of the data set.

[3]:
```
sns.pairplot(iris,hue='species',palette='Dark2')
```

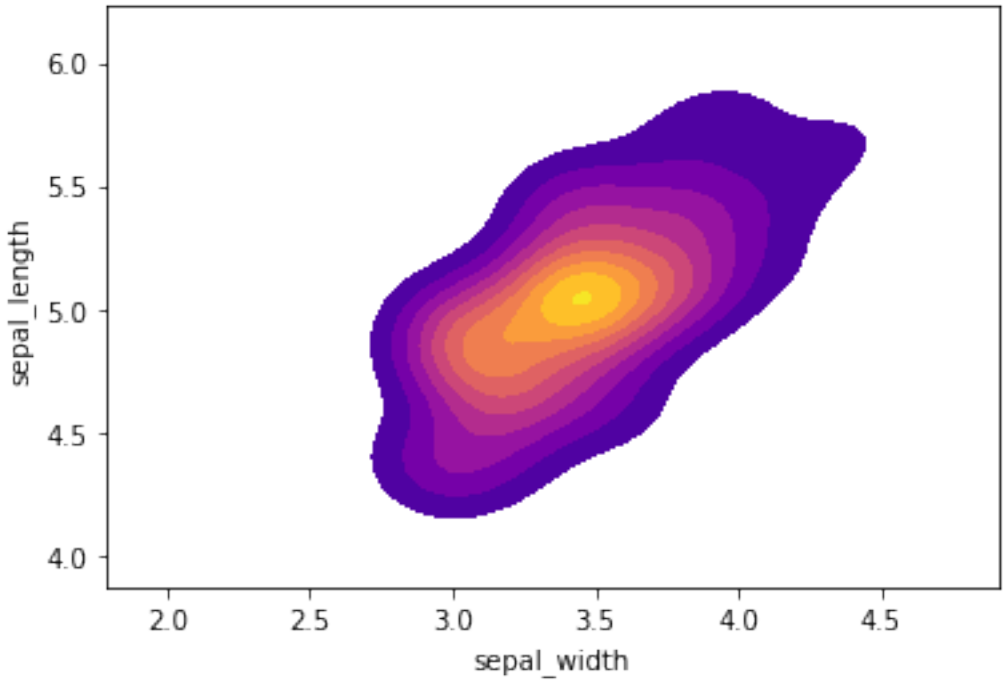
[3]:
```
<seaborn.axisgrid.PairGrid at 0x152abca7a48>
```




kde plot of sepal_length versus sepal width for setosa species of flower

[4]:
```
setosa = iris[iris['species']=='setosa']
sns.kdeplot( setosa['sepal_width'], setosa['sepal_length'],
                 cmap="plasma", shade=True, shade_lowest=False)
```

[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x152d78bb288>
```




### 0.1.3 Train Test Split

[5]:
```
from sklearn.model_selection import train_test_split
```

[6]:
```
X = iris.drop('species',axis=1)
y = iris['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

### Training the Model

[7]:
```
from sklearn.svm import SVC
```

[8]:
```
svc_model = SVC()
```

[9]:
```
svc_model.fit(X_train,y_train)
```

```
C:\Users\rajar\Anaconda3\lib\site-packages\sklearn\svm\base.py:193:
FutureWarning: The default value of gamma will change from 'auto' to 'scale' in
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

[9]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
```

```
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

### Model Evaluation

[10]:
```
predictions = svc_model.predict(X_test)
```

[11]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

[12]:
```
print(confusion_matrix(y_test,predictions))
```

```
[[17  0  0]
 [ 0 14  0]
 [ 0  0 14]]
```

[13]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        17
  versicolor       1.00      1.00      1.00        14
   virginica       1.00      1.00      1.00        14

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

### Gridsearch Practice

[14]:
```
from sklearn.model_selection import GridSearchCV
```

Create a dictionary called param_grid and fill out some parameters for C and gamma.

[15]:
```
param_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001]}
```

Create a GridSearchCV object and fit it to the training data

[16]:
```
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=2)
grid.fit(X_train,y_train)
```

```
C:\Users\rajar\Anaconda3\lib\site-
packages\sklearn\model_selection\_split.py:1978: FutureWarning: The default
value of cv will change from 3 to 5 in version 0.22. Specify it explicitly to
silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

Fitting 3 folds for each of 16 candidates, totalling 48 fits
[CV] C=0.1, gamma=1 ...
```

```
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.5s remaining:    0.0s
[CV] ... C=0.1, gamma=1, total=   0.6s
[CV] C=0.1, gamma=1 ...
[CV] ... C=0.1, gamma=1, total=   0.0s
[CV] C=0.1, gamma=1 ...
[CV] ... C=0.1, gamma=1, total=   0.0s
[CV] C=0.1, gamma=0.1 ...
[CV] ... C=0.1, gamma=0.1, total=   0.0s
[CV] C=0.1, gamma=0.1 ...
[CV] ... C=0.1, gamma=0.1, total=   0.0s
[CV] C=0.1, gamma=0.1 ...
[CV] ... C=0.1, gamma=0.1, total=   0.0s
[CV] C=0.1, gamma=0.01 ...
[CV] ... C=0.1, gamma=0.01, total=   0.0s
[CV] C=0.1, gamma=0.01 ...
[CV] ... C=0.1, gamma=0.01, total=   0.0s
[CV] C=0.1, gamma=0.01 ...
[CV] ... C=0.1, gamma=0.01, total=   0.0s
[CV] C=0.1, gamma=0.001 ...
[CV] ... C=0.1, gamma=0.001, total=   0.0s
[CV] C=0.1, gamma=0.001 ...
[CV] ... C=0.1, gamma=0.001, total=   0.0s
[CV] C=0.1, gamma=0.001 ...
[CV] ... C=0.1, gamma=0.001, total=   0.0s
[CV] C=1, gamma=1 ...
[CV] ... C=1, gamma=1, total=   0.0s
[CV] C=1, gamma=1 ...
[CV] ... C=1, gamma=1, total=   0.0s
[CV] C=1, gamma=1 ...
[CV] ... C=1, gamma=1, total=   0.0s
[CV] C=1, gamma=0.1 ...
[CV] ... C=1, gamma=0.1, total=   0.0s
[CV] C=1, gamma=0.1 ...
[CV] ... C=1, gamma=0.1, total=   0.0s
[CV] C=1, gamma=0.1 ...
[CV] ... C=1, gamma=0.1, total=   0.0s
[CV] C=1, gamma=0.01 ...
[CV] ... C=1, gamma=0.01, total=   0.0s
[CV] C=1, gamma=0.01 ...
[CV] ... C=1, gamma=0.01, total=   0.0s
[CV] C=1, gamma=0.01 ...
[CV] ... C=1, gamma=0.01, total=   0.0s
[CV] C=1, gamma=0.001 ...
[CV] ... C=1, gamma=0.001, total=   0.0s
[CV] C=1, gamma=0.001 ...
[CV] ... C=1, gamma=0.001, total=   0.0s
[CV] C=1, gamma=0.001 ...
```

```
[CV] ... C=1, gamma=0.001, total=   0.0s
[CV] C=10, gamma=1 ...
[CV] ... C=10, gamma=1, total=   0.0s
[CV] C=10, gamma=1 ...
[CV] ... C=10, gamma=1, total=   0.0s
[CV] C=10, gamma=1 ...
[CV] ... C=10, gamma=1, total=   0.0s
[CV] C=10, gamma=0.1 ...
[CV] ... C=10, gamma=0.1, total=   0.0s
[CV] C=10, gamma=0.1 ...
[CV] ... C=10, gamma=0.1, total=   0.0s
[CV] C=10, gamma=0.1 ...
[CV] ... C=10, gamma=0.1, total=   0.0s
[CV] C=10, gamma=0.01 ...
[CV] ... C=10, gamma=0.01, total=   0.0s
[CV] C=10, gamma=0.01 ...
[CV] ... C=10, gamma=0.01, total=   0.0s
[CV] C=10, gamma=0.01 ...
[CV] ... C=10, gamma=0.01, total=   0.0s
[CV] C=10, gamma=0.001 ...
[CV] ... C=10, gamma=0.001, total=   0.0s
[CV] C=10, gamma=0.001 ...
[CV] ... C=10, gamma=0.001, total=   0.0s
[CV] C=10, gamma=0.001 ...
[CV] ... C=10, gamma=0.001, total=   0.0s
[CV] C=100, gamma=1 ...
[CV] ... C=100, gamma=1, total=   0.0s
[CV] C=100, gamma=1 ...
[CV] ... C=100, gamma=1, total=   0.0s
[CV] C=100, gamma=1 ...
[CV] ... C=100, gamma=1, total=   0.0s
[CV] C=100, gamma=0.1 ...
[CV] ... C=100, gamma=0.1, total=   0.0s
[CV] C=100, gamma=0.1 ...
[CV] ... C=100, gamma=0.1, total=   0.0s
[CV] C=100, gamma=0.1 ...
[CV] ... C=100, gamma=0.1, total=   0.0s
[CV] C=100, gamma=0.01 ...
[CV] ... C=100, gamma=0.01, total=   0.0s
[CV] C=100, gamma=0.01 ...
[CV] ... C=100, gamma=0.01, total=   0.0s
[CV] C=100, gamma=0.01 ...
[CV] ... C=100, gamma=0.01, total=   0.0s
[CV] C=100, gamma=0.001 ...
[CV] ... C=100, gamma=0.001, total=   0.0s
[CV] C=100, gamma=0.001 ...
[CV] ... C=100, gamma=0.001, total=   0.0s
[CV] C=100, gamma=0.001 ...
```

```
[CV] ... C=100, gamma=0.001, total=   0.0s

[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed:    0.9s finished
```

[16]:
```
GridSearchCV(cv='warn', error_score='raise-deprecating',
             estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='auto_deprecated', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='warn', n_jobs=None,
             param_grid={'C': [0.1, 1, 10, 100],
                         'gamma': [1, 0.1, 0.01, 0.001]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=2)
```

Now take that grid model and create some predictions using the test set and create classification reports and confusion matrices for them

[17]:
```
grid_predictions = grid.predict(X_test)
```

[18]:
```
print(confusion_matrix(y_test,grid_predictions))
```

```
[[17  0  0]
 [ 0 14  0]
 [ 0  0 14]]
```

[19]:
```
print(classification_report(y_test,grid_predictions))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        17
  versicolor       1.00      1.00      1.00        14
   virginica       1.00      1.00      1.00        14

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

The predictions are exactly the same as there is just one point that is too noisey to grab. Hence, it is better not to overfit model that would be able to grab that.