iris svm

April 1, 2021

Support Vector Machine with Iris Dataset

Sohini Mukherjee

1.04.2021

The Data

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by Sir Ronald Fisher in the 1936 as an example of discriminant analysis.

The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor), so 150 total samples. Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

Here's a picture of the three different Iris types:

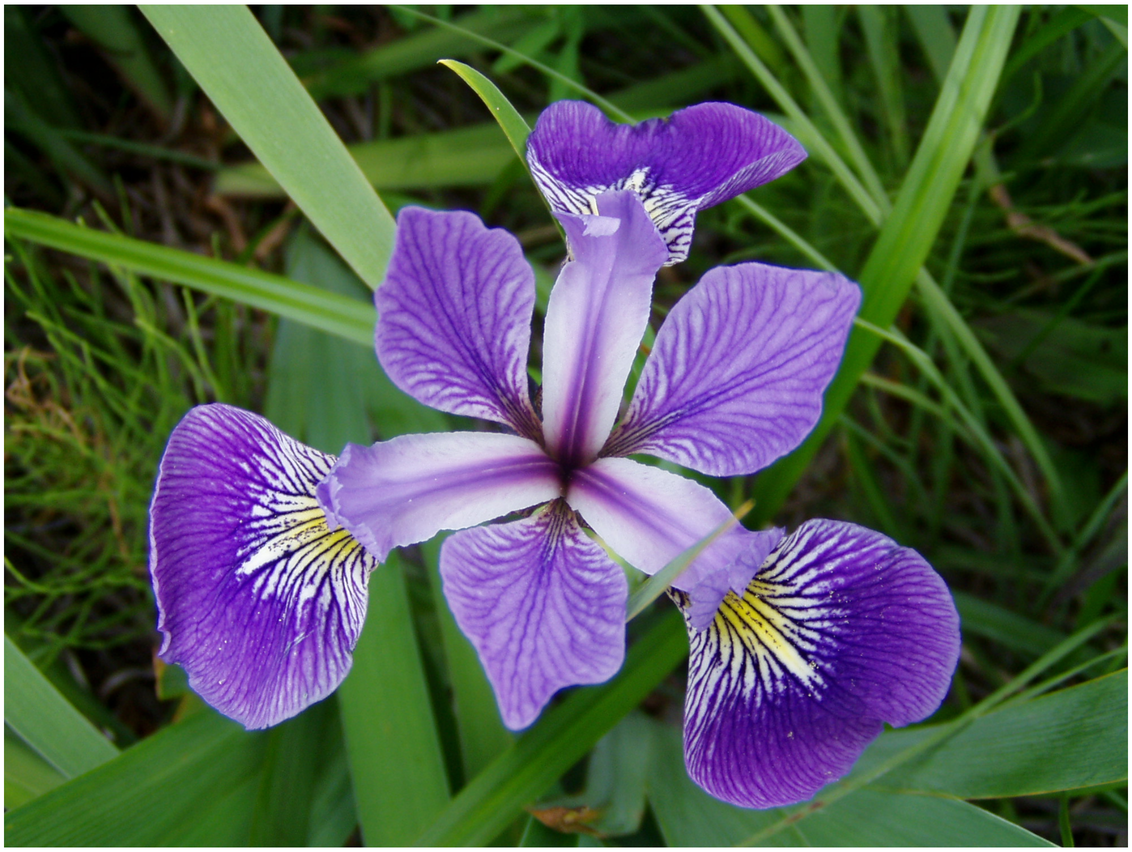
[1]:
```
# The Iris Setosa
from IPython.display import Image
url = 'http://upload.wikimedia.org/wikipedia/commons/5/56/
 ↪Kosaciec_szczecinkowaty_Iris_setosa.jpg'
Image(url,width=300, height=300)
```

[1]:

[2]:
```
# The Iris Versicolor
from IPython.display import Image
```

```
url = 'http://upload.wikimedia.org/wikipedia/commons/4/41/Iris_versicolor_3.jpg'
Image(url,width=300, height=300)
```

[2]:



[3]:

```
# The Iris Virginica
from IPython.display import Image
url = 'http://upload.wikimedia.org/wikipedia/commons/9/9f/Iris_virginica.jpg'
Image(url,width=300, height=300)
```

[3]:

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

The three classes in the Iris dataset:

1. Iris-setosa (n=50)
2. Iris-versicolor (n=50)
3. Iris-virginica (n=50)

The four features of the Iris dataset:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm

Importing Libraries and Getting the Data

[4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
%matplotlib inline
```

```
[5]: iris = sns.load_dataset('Iris')
```

```
[6]: iris.head()
```

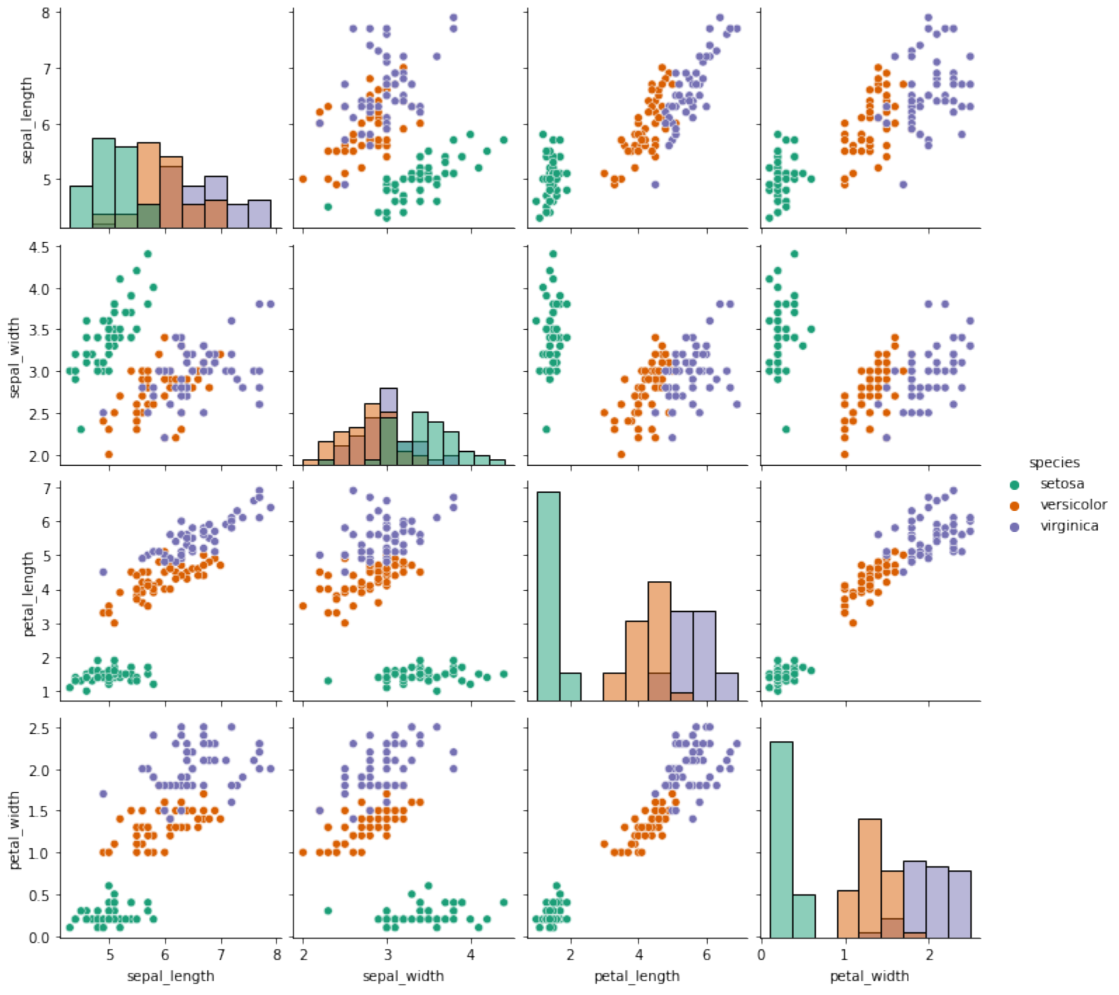
```
[6]:    sepal_length  sepal_width  petal_length  petal_width species
     0           5.1          3.5           1.4          0.2  setosa
     1           4.9          3.0           1.4          0.2  setosa
     2           4.7          3.2           1.3          0.2  setosa
     3           4.6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

Exploratory Data Analysis

Creating a pairplot of the data set to check which flower species seems to be the most separable.

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

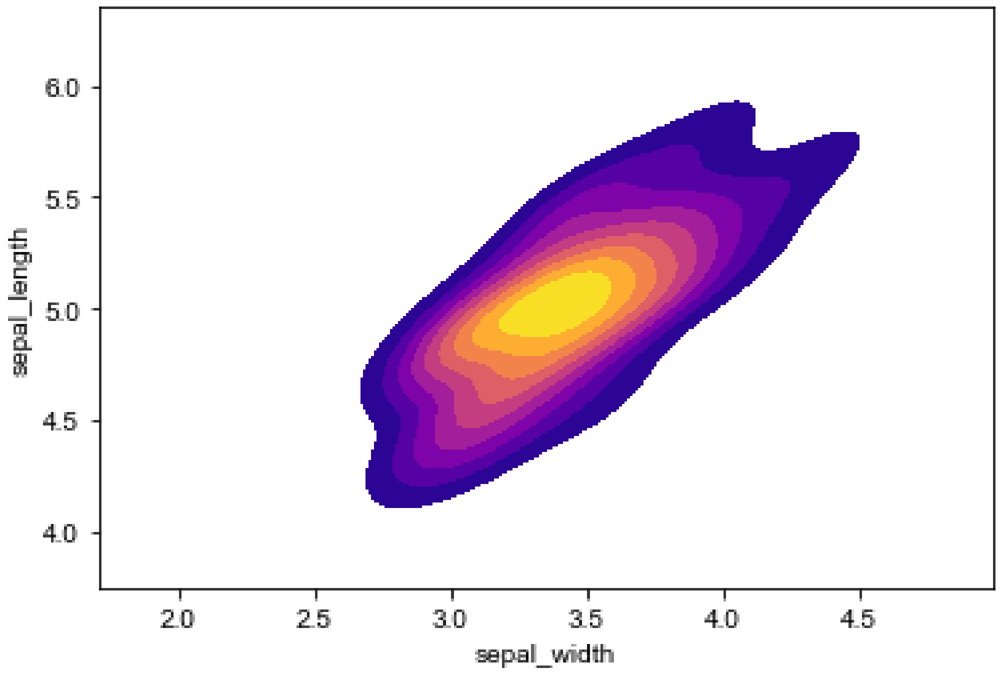
```
[7]: <seaborn.axisgrid.PairGrid at 0x1d610983ac0>
```



Setosa seems most separable.

Creating a kde plot of sepal_length versus sepal width for setosa species of flower.

```
[8]: setosa = iris[iris['species']=='setosa']
     sns.kdeplot(x='sepal_width', y='sepal_length', data=setosa, cmap='plasma',␣
      ↪shade=True, thresh=0.1)
     sns.set_style('whitegrid')
```



Train Test Split

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

```
[10]: X= iris.drop('species', axis=1)
      y= iris['species']

      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
       ↪random_state=101)
```

Training the Support Vector Machine Classifier

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

[12]:
```
model = SVC()
model.fit(X_train, y_train)
```

[12]:
```
SVC()
```

Model Evaluation

[13]:
```
pred = model.predict(X_test)
```

[14]:
```
from sklearn.metrics import confusion_matrix, classification_report

print(confusion_matrix(y_test, pred))
print('\n')
print(classification_report(y_test, pred))
```

```
[[13  0  0]
 [ 0 19  1]
 [ 0  0 12]]


              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
  versicolor       1.00      0.95      0.97        20
   virginica       0.92      1.00      0.96        12

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

Gridsearch

[15]:
```
from sklearn.model_selection import GridSearchCV

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

grid= GridSearchCV(SVC(), pgrid, refit=True, verbose=2)
grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[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=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=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=0.1 …
[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=0.01 …
[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=0.1, gamma=0.001 …
[CV] … C=0.1, gamma=0.001, total=   0.0s
[CV] C=0.1, gamma=0.001 …
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
```

```
[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=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=0.1 …
[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=0.01 …
[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=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=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=0.1 …
[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=0.01 …
[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=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=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=0.1 …
[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=0.01 …
[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
[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  80 out of  80 | elapsed:    0.6s finished
```

```
[15]: GridSearchCV(estimator=SVC(),
                   param_grid={'C': [0.1, 1, 10, 100],
                               'gamma': [1, 0.1, 0.01, 0.001]},
                   verbose=2)
```

```
[16]: gpred = grid.predict(X_test)
```

```
[17]: confusion_matrix(y_test, gpred)
```

```
[17]: array([[13,  0,  0],
             [ 0, 19,  1],
             [ 0,  0, 12]], dtype=int64)
```

```
[18]: print(classification_report(y_test, gpred))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
  versicolor       1.00      0.95      0.97        20
   virginica       0.92      1.00      0.96        12

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```