IRIS Flower Classification using Machine Learning Support Vector Machine(SVM) Algorithm

Article · January 2023

CITATIONS
0

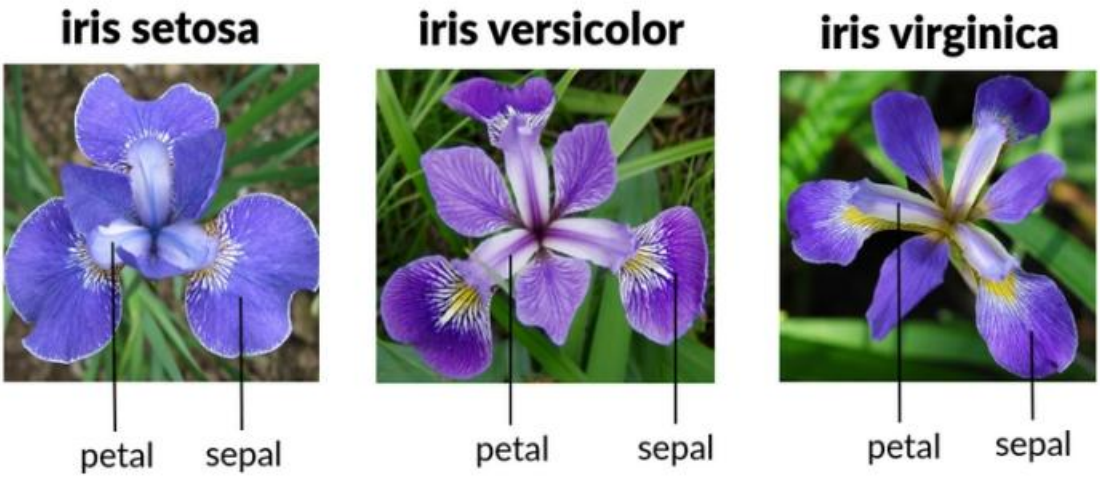
READS
12

3 authors, including:

Yaswanth Sankarasetti
Amrita Sai Institute of Science and Technology
1 PUBLICATION **0** CITATIONS

SEE PROFILE

Iris Flower Classification Using SVM



Importing libraries

```
import numpy as np

import pandas as pd

import seaborn as sns

sns.set_palette('husl')

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.model_selection import cross_val_score

from sklearn.model_selection import StratifiedKFold

from sklearn.metrics import classification_report

from sklearn.metrics import accuracy_score

from sklearn.svm import SVC
```

Loading IRIS Data

```
url =
'https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.
csv'
```

Creating the list of column name:

```
col_name = ['sepal-lenght','sepal-width','petal-lenght','petal-
width','class']
```

Pandas read_csv() is used for reading the csv file:

```
dataset = pd.read_csv(url, names = col_name)
```

Violin plot

Plotting the violin plot to check the comparison of a variable distribution:

```
sns.violinplot(y='class', x='sepal-lenght', data=dataset,
inner='quartile')

plt.show()

sns.violinplot(y='class', x='sepal-width', data=dataset,
inner='quartile')

plt.show()

sns.violinplot(y='class', x='petal-lenght', data=dataset,
inner='quartile')

plt.show()

sns.violinplot(y='class', x='petal-width', data=dataset,
inner='quartile')

plt.show()
```

5. Model Building- part 1

5.1 Splitting the dataset

X is having all the dependent variables.
Y is having an independent variable (here in this case 'class' is an independent variable).

```
X = dataset.drop(['class'], axis=1)

y = dataset['class']

print(f'X shape: {X.shape} | y shape: {y.shape} ')
```

5.2 Train Test split

Splitting our dataset into train and test using train_test_split(), what we are doing here is taking 80% of data to train our model, and 20% that we will hold back as a validation dataset:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=1)
```

5.3 Model Creation (SVC: Support Vector Machine for classification)

```
models = []

models.append(('SVC', SVC(gamma='auto')))

results = []

model_names = []

for name, model in models:

  kfold = StratifiedKFold(n_splits=10, random_state=1,
shuffle=True)

  cv_results = cross_val_score(model, X_train, y_train, cv=kfold,
scoring='accuracy')

  results.append(cv_results)

  model_names.append(name)
```

```
  print('%s: %f (%f)' % (name, cv_results.mean(),
cv_results.std()))
```

6. Model Building- part 2

6.1. We are defining our SVC model and passing gamma as auto.

6.2. After that fitting/training the model on X_train and Y_train using .fit() method.

6.3. Then we are predicting on X_test using .predict() method.

```
 model = SVC(gamma='auto')

model.fit(X_train, y_train)

prediction = model.predict(X_test)
```

6.4. checking the accuracy of our model using
accuracy_score(y_test, prediction)
y_test: actual values of X_test
prediction: predicted values of X_test (refer to point 3).

6.5. Printing out the classification report using
classification_report(y_test, prediction).

```
print(f'Test Accuracy: {accuracy_score(y_test, prediction)}')

print(f'Classification Report: \n {classification_report(y_test,
prediction)}')
```

Total Code

```
import numpy as np

import pandas as pd

import seaborn as sns

sns.set_palette('husl')

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
```

```
from sklearn.model_selection import cross_val_score

from sklearn.model_selection import StratifiedKFold

from sklearn.metrics import classification_report

from sklearn.metrics import accuracy_score

from sklearn.svm import SVC

url =
'https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.
csv'

col_name = ['sepal-lenght','sepal-width','petal-lenght','petal-
width','class']

dataset = pd.read_csv(url, names = col_name)

sns.violinplot(y='class', x='sepal-lenght', data=dataset,
inner='quartile')

plt.show()

sns.violinplot(y='class', x='sepal-width', data=dataset,
inner='quartile')

plt.show()

sns.violinplot(y='class', x='petal-lenght', data=dataset,
inner='quartile')

plt.show()

sns.violinplot(y='class', x='petal-width', data=dataset,
inner='quartile')

plt.show()

X = dataset.drop(['class'], axis=1)

y = dataset['class']

print(f'X shape: {X.shape} | y shape: {y.shape} ')

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=1)

models = []
```

```
models.append(('SVC', SVC(gamma='auto')))

# evaluate each model in turn

results = []

model_names = []

for name, model in models:

 kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)

 cv_results = cross_val_score(model, X_train, y_train, cv=kfold,
scoring='accuracy')

  results.append(cv_results)

  model_names.append(name)

  print('%s: %f (%f)' % (name, cv_results.mean(),
cv_results.std()))

  model = SVC(gamma='auto')

model.fit(X_train, y_train)

prediction = model.predict(X_test)

print(f'Test Accuracy: {accuracy_score(y_test, prediction)}')

print(f'Classification Report: \n {classification_report(y_test,
prediction)}')
```

```
[0. 0. 0. 0. 0.]]
```