Classification of Iris Species

November 23, 2024

1 Classification of Iris Species

This project focuses on building a machine learning model to classify the species of Iris flowers based on their physical characteristics (sepal length, sepal width, petal length, and petal width). Using the classic Iris dataset, this project demonstrates the application of supervised learning techniques to solve a multi-class classification problem.

1.1 Features

- Exploratory Data Analysis (EDA) to uncover patterns in the Iris dataset.
- Implementation of machine learning models using Scikit-learn.
- Model evaluation using metrics like accuracy.
- Visualization of results with scatter plots.

1.2 Tech Stack

- Programming Language: Python
- Libraries: NumPy, Pandas, Scikit-learn
- Dataset: Iris Flower Dataset (Fisher's Iris Dataset)

1.3 How It Works

- The dataset is split into training and testing sets.
- Features are analyzed to understand their relationship with the target labels.
- A machine learning model (e.g., K-Nearest Neighbors) is trained on the training set.
- Predictions are made on the test set, and the model's performance is evaluated.

1.4 Objective

The primary goal of this project is to classify Iris flowers into one of the three species: - Iris setosa - Iris versicolor - Iris virginica

1.5 Key Highlights

- Beginner-friendly implementation for understanding the basics of classification problems.
- Reproducible code and visualizations to aid in learning machine learning concepts.
- A modular design that allows easy expansion or integration of new models.

1.6 Usage

Clone the repository and follow the instructions in the README to reproduce the results or extend the project.

[253]:
```
#!pip install mglearn (use this command for installing the mglearn library)
import mglearn
import numpy as np
import pandas as pd
```

[255]:
```
from sklearn.datasets import load_iris #import ... from ... is incorrect.
iris_dataset=load_iris()
print("Keys of Iris dataset: \n{}".format(iris_dataset.keys()))
```

```
Keys of Iris dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names',
'filename', 'data_module'])
```

[218]:
```
print("Iris dataset Description: \n{}".format(iris_dataset['DESCR'])) #Can␣
 ↪access using iris_dataset.DESCR as well.
```

```
Iris dataset Description:
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive attributes and the class
:Attribute Information:
    - sepal length in cm
    - sepal width in cm
    - petal length in cm
    - petal width in cm
    - class:
            - Iris-Setosa
            - Iris-Versicolour
            - Iris-Virginica

:Summary Statistics:

============== ==== ==== ======= ===== ====================
                Min  Max   Mean    SD   Class Correlation
============== ==== ==== ======= ===== ====================
sepal length:   4.3  7.9   5.84   0.83    0.7826
sepal width:    2.0  4.4   3.05   0.43   -0.4194
petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
```

```
petal width:    0.1  2.5    1.20   0.76    0.9565  (high!)
============== ==== ==== ======= ===== ====================

:Missing Attribute Values: None
:Class Distribution: 33.3% for each of 3 classes.
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. dropdown:: References

  - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
    Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
    Mathematical Statistics" (John Wiley, NY, 1950).
  - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
    (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
  - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
    Structure and Classification Rule for Recognition in Partially Exposed
    Environments".  IEEE Transactions on Pattern Analysis and Machine
    Intelligence, Vol. PAMI-2, No. 1, 67-71.
  - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
    on Information Theory, May 1972, 431-433.
  - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
    conceptual clustering system finds 3 classes in the data.
  - Many, many more …
```

[220]:
```
print("Target Names:",iris_dataset['target_names'])
```

```
Target Names: ['setosa' 'versicolor' 'virginica']
```

[222]:
```
print("Feature Names:",iris_dataset['feature_names'])
```

```
Feature Names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
'petal width (cm)']
```

[224]:
```
print("Type of Data: {}".format(type(iris_dataset['data'])))
```

```
Type of Data: <class 'numpy.ndarray'>
```

[226]:
```
# type() is fundamental to Python while shape is an attribute specific to numpy␣
 ↪arrays.
print("Shape of Data: {}".format(iris_dataset['data'].shape)) # (R,C)
```

```
Shape of Data: (150, 4)
```

[228]:
```
print("First five rows of data: \n",iris_dataset['data'][0:5])
```

```
First five rows of data:
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

[230]:
```
print(iris_dataset['target'])
print(len(iris_dataset['target'])) #Better to use shape attribute instead of␣
 ↪len().
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
150
```

[232]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train,␣
 ↪y_test=train_test_split(iris_dataset['data'],iris_dataset['target'],random_state=0)
print("Shape of X_train: ",X_train.shape)
print("Shape of X_test: ",X_test.shape)
print("Shape of y_train: ",y_train.shape)
print("Shape of y_test: ",y_test.shape)
```

```
Shape of X_train:  (112, 4)
Shape of X_test:  (38, 4)
Shape of y_train:  (112,)
Shape of y_test:  (38,)
```

[234]:
```
iris_dataframe=pd.DataFrame(X_train, columns=iris_dataset.feature_names)
iris_dataframe
```

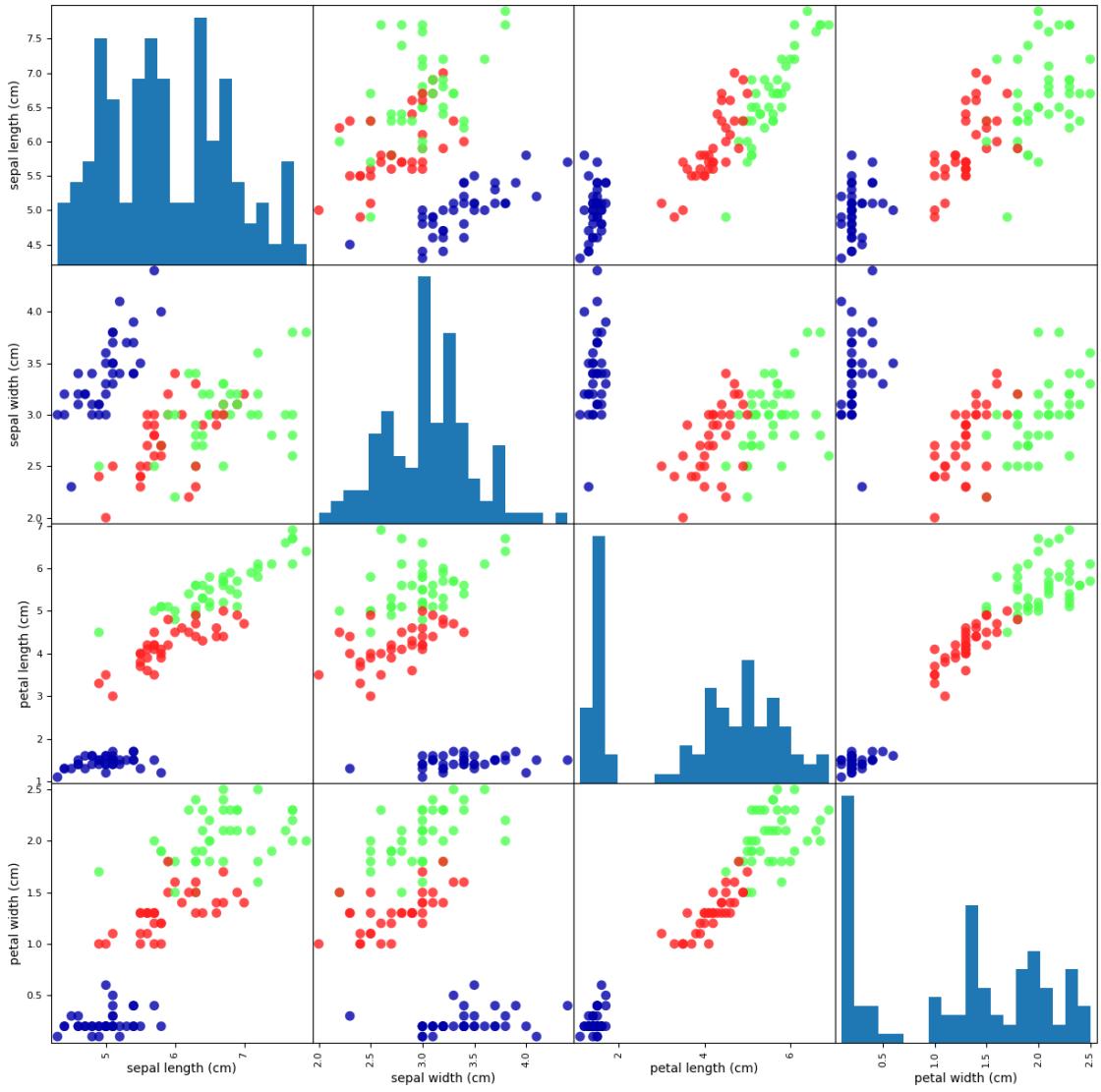
[234]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.9	3.0	4.2	1.5
1	5.8	2.6	4.0	1.2
2	6.8	3.0	5.5	2.1
3	4.7	3.2	1.3	0.2
4	6.9	3.1	5.1	2.3

```
..                  …                  …                  …                  …
107                 4.9                3.1                1.5                0.1
108                 6.3                2.9                5.6                1.8
109                 5.8                2.7                4.1                1.0
110                 7.7                3.8                6.7                2.2
111                 4.6                3.2                1.4                0.2

[112 rows x 4 columns]
```

[199]:
```
grr=pd.plotting.scatter_matrix(iris_dataframe,c=y_train, figsize=(15,15),␣
↪marker='o',hist_kwds={'bins':20},s=60,alpha=.8,cmap=mglearn.cm3)
```



[236]:
```
#Building the ML model using k-Nearest Neighbors Classifier
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train,y_train)
```

[236]:
```
KNeighborsClassifier(n_neighbors=1)
```

[238]:
```
X_new=np.array([[5,2.9,1,0.2]]) #Creating a 2d array is important as 1d array␣
↪would imply that each data point refers to a different row.
print("X_new.shape:",X_new.shape)
```

```
X_new.shape: (1, 4)
```

[240]:
```
prediction=knn.predict(X_new)
print("Prediction:",prediction)
print("Predicted target name:",iris_dataset.target_names[prediction])
```

```
Prediction: [0]
Predicted target name: ['setosa']
```

[242]:
```
y_pred=knn.predict(X_test)
print("Test set predictions:",y_pred)
```

```
Test set predictions: [2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1
0 2 1 0 2 2 1 0
 2]
```

[244]:
```
#print("Test set score: {:.2f}".format(np.mean(y_pred==y_test)))
print("Test set score: {:.2f}".format(knn.score(X_test,y_test)))
```

```
Test set score: 0.97
```

1.7 Conclusion

The model achieved a test set accuracy of approximately 97%, indicating that it correctly predicted the species for 97% of the irises in the test set. Under certain mathematical assumptions, this suggests that the model is likely to be correct 97% of the time when classifying new iris samples. For our hobby botanist application, this high accuracy level implies that the model is reliable enough for practical use.