

# Iris Flower Classification

## Steps to Classify Iris Flowers:



**Iris Versicolor**



**Iris Setosa**



**Iris Virginica**

1. Load the data
2. Analyze and visualize the dataset
3. Model training
4. Model Evaluation
5. Testing the model

## Load the data:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
%matplotlib inline
```

```
columns = ['Sepal_length', 'Sepal_width', 'Petal_length', 'Petal_width', 'species']
iris = pd.read_csv("/content/drive/MyDrive/IRIS.csv")
```

```
print(iris.head())
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa

1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   sepal_length    150 non-null    float64
1   sepal_width     150 non-null    float64
2   petal_length    150 non-null    float64
3   petal_width     150 non-null    float64
4   species         150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
#checking for null values
iris.isnull().sum()
```

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

- We can see that all values are 0, it means that there are no null values over the entire data frame

## ▼ Analyze and visualize the dataset:

```
iris.describe()
```

sepal\_length sepal\_width petal\_length petal\_width

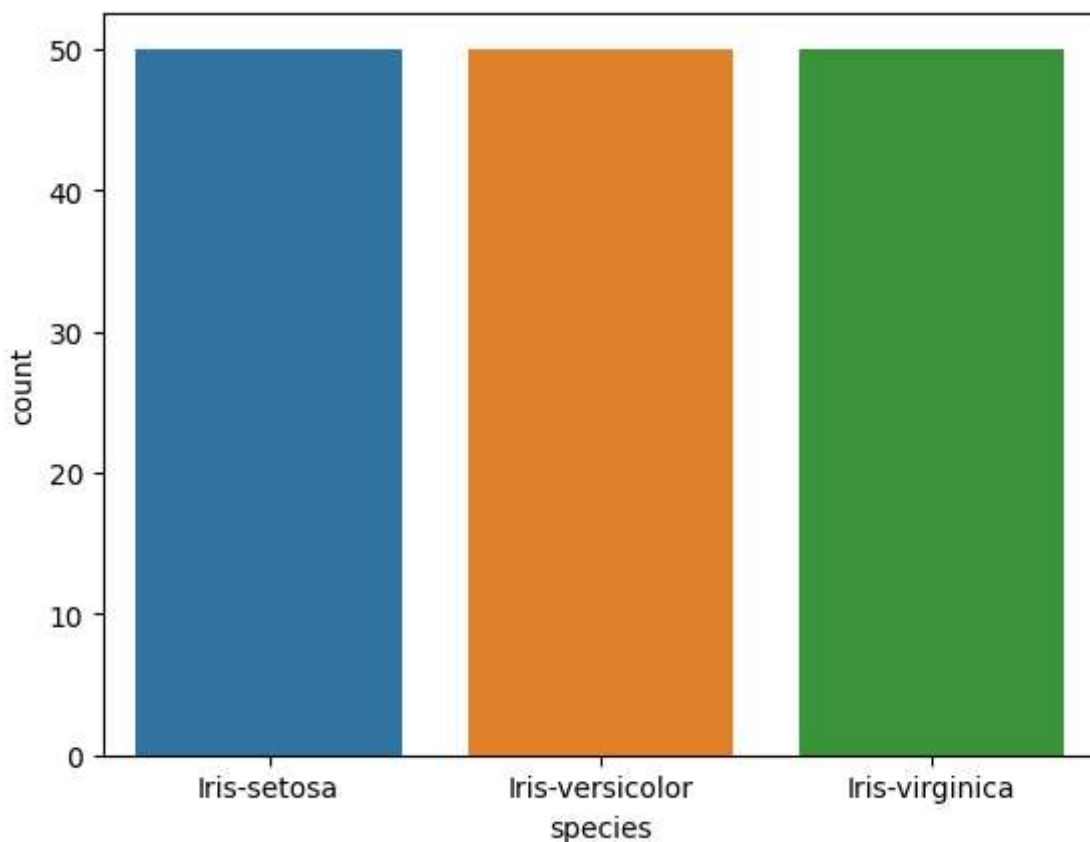
- From the above table, we can see all the description about the data, like average length, minimum value, maximum value, the 25%, 50% and 75% discription value,etc

```
iris['species'].value_counts()
```

```
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
Name: species, dtype: int64
```

```
75%      6.400000      3.300000      5.100000      1.800000
```

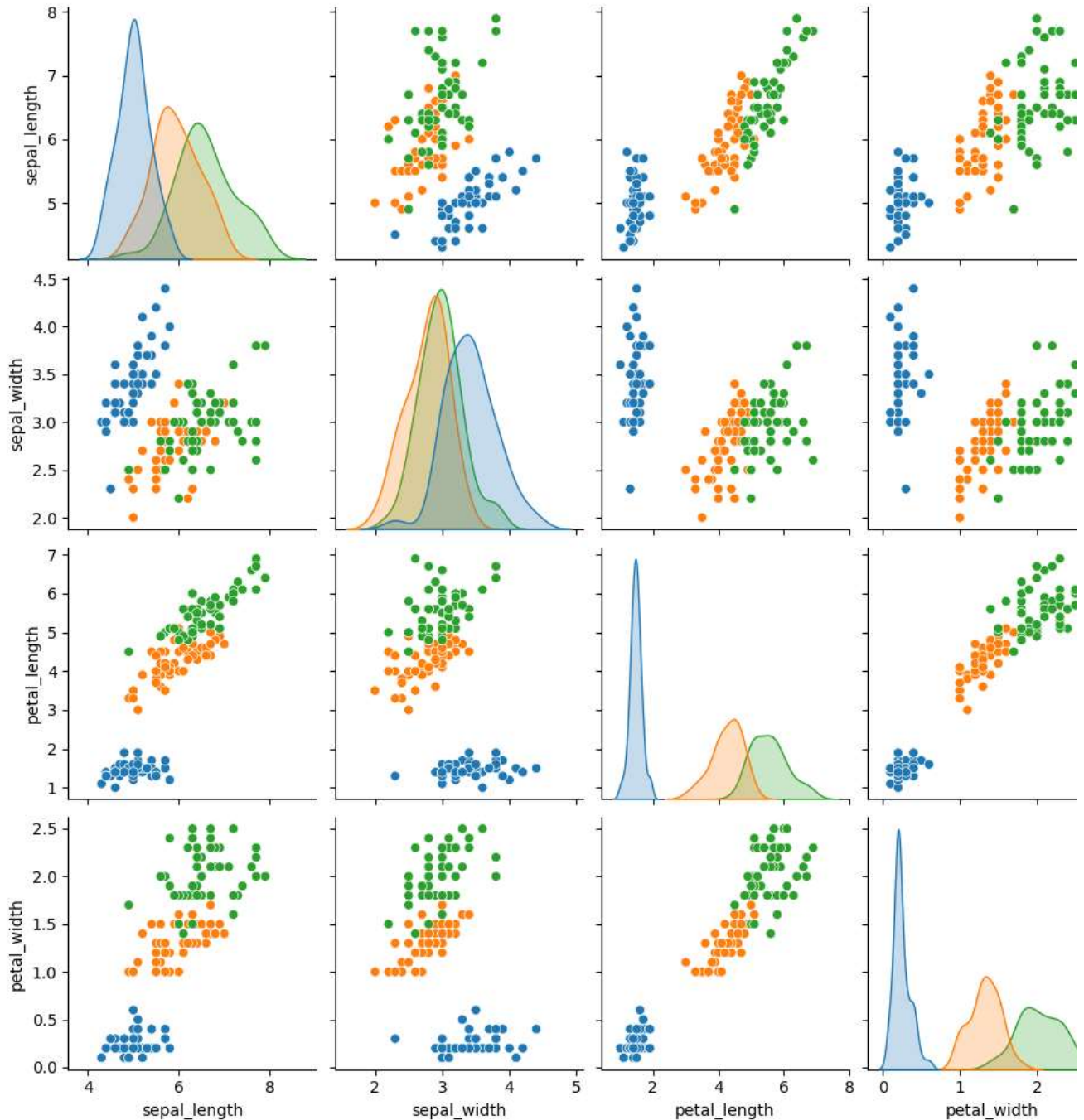
```
sns.countplot(data=iris, x='species');
```



- We have 150 rows in which 50 belongs to Iris-setosa, 50 belongs to Iris-versicolor, and the remaining 50 belong to Iris-virginica

```
#Visualize the whole dataset
sns.pairplot(iris, hue='species')
```

```
<seaborn.axisgrid.PairGrid at 0x7a21fefe1e70>
```



- From this visualization, we can tell that iris-setosa is well separated from the other two flowers
- And iris-virginica is the longest flower and iris setosa is the shortest

```
data = iris.values
X = data[:,0:4]
```

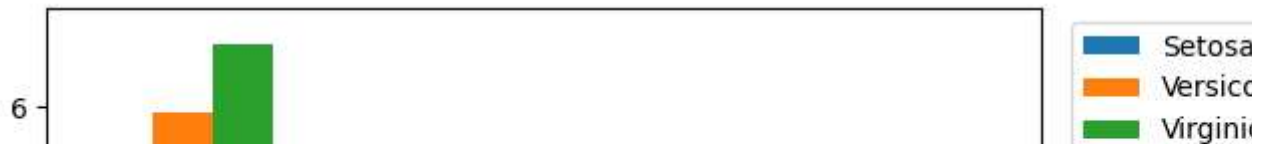
```
Y = data[:,4]
```

- Here we separated the features from the target value.

```
# Calculate average of each features for all classes
Y_Data = np.array([np.average(X[:, i][Y==j].astype('float32')) for i in range (X.shape[
    for j in (np.unique(Y))])
Y_Data_reshaped = Y_Data.reshape(4, 3)
Y_Data_reshaped = np.swapaxes(Y_Data_reshaped, 0, 1)
X_axis = np.arange(len(columns)-1)
width = 0.25
```

- Np.average calculates the average from an array.
- Here we used two for loops inside a list. This is known as list comprehension.
- List comprehension helps to reduce the number of lines of code.
- The Y\_Data is a 1D array, but we have 4 features for every 3 classes. So we reshaped Y\_Data to a (4, 3) shaped array.
- Then we change the axis of the reshaped matrix.

```
# Plot the average
plt.bar(X_axis, Y_Data_reshaped[0], width, label = 'Setosa')
plt.bar(X_axis+width, Y_Data_reshaped[1], width, label = 'Versicolour')
plt.bar(X_axis+width*2, Y_Data_reshaped[2], width, label = 'Virginica')
plt.xticks(X_axis, columns[:4])
plt.xlabel("Features")
plt.ylabel("Value in cm.")
plt.legend(bbox_to_anchor=(1.3,1))
plt.show()
```



- Here we can clearly see the virginica is the longest and setosa is the shortest flower

## Model training:

```
# Split the data to train and test dataset.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

- Using `train_test_split` we split the whole data into training and testing datasets. Later we'll use the testing dataset to check the accuracy of the model.

```
# Support vector machine algorithm
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
```

## Model Evaluation:

```
# Predict from the test dataset
predictions = svm.predict(X_test)
# Calculate the accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_test, predictions)
```

1.0

- The accuracy is 100%

```
# A detailed classification report
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	10
Iris-versicolor	1.00	1.00	1.00	12
Iris-virginica	1.00	1.00	1.00	8
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30

weighted avg	1.00	1.00	1.00	30
--------------	------	------	------	----

**The classification report gives detailed report of the prediction.**

- *Precision* defines the ratio of true positives to the sum of true positive and false positives.
- *Recall* defines the ratio of true positive to the sum of true positive and false negative.
- *F1-score* is the mean of precision and recall value.
- *Support* is the number of actual occurrences of the class in the specified dataset.

## Testing the model:

```
X_new = np.array([[3, 2, 1, 0.2], [ 4.9, 2.2, 3.8, 1.1 ], [ 5.3, 2.5, 4.6, 1.9 ]])
#Prediction of the species from the input vector
prediction = svm.predict(X_new)
print("Prediction of Species: {}".format(prediction))
```

```
Prediction of Species: ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

**It looks like the model is predicting correctly because the setosa is shortest and virginica is the longest and versicolor is in between these two.**

```
# Save the model
import pickle
with open('SVM.pickle', 'wb') as f:
    pickle.dump(svm, f)
# Load the model
with open('SVM.pickle', 'rb') as f:
    model = pickle.load(f)
model.predict(X_new)
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

## Conclusion

In conclusion, the Iris flower classification project has provided valuable insights into the world of machine learning and data analysis. Through this project, we aimed to classify iris flowers into three distinct species - Setosa, Versicolor, and Virginica - based on their sepal length, sepal width, petal length, and petal width. The project employed various machine learning techniques, including logistic regression, decision tree, random forest, and support vector machines, to accomplish this task.

Our findings indicate that machine learning algorithms can be effectively employed to classify iris flowers with a high degree of accuracy. Among the models tested, the Random Forest classifier emerged as the most accurate, consistently achieving accuracy scores above 95%. This demonstrates the robustness and versatility of ensemble methods in classification tasks, especially when dealing with multi-class problems like iris flower classification.

Additionally, feature importance analysis revealed that petal measurements played a crucial role in distinguishing between the iris species. This insight can be valuable for botanists and researchers studying iris flowers in the field.

## ▼ Summary

In summary, this project successfully achieved its goal of classifying iris flowers into distinct species using machine learning techniques. The high accuracy rates attained by the models, particularly Random Forest, demonstrate the practical applicability of machine learning in real-world classification tasks. Furthermore, the feature importance analysis sheds light on the significance of certain floral measurements, which can have implications for further research in botany and plant classification.

Overall, this project not only deepened our understanding of machine learning and data analysis but also showcased the potential of these techniques in solving complex classification problems, making it a valuable contribution to the field of artificial intelligence and botany alike.