classification_of_iris

October 3, 2025

0.1 Project: Flower species classification using ML Algorithms

---

0.1.1 Author: Faryal Rifaz

0.1.2 Dataset: Iris

0.1.3 Task

Build a classification model for flower species , analyze the results, and perform comparisons including accuracy metrics, feature importance, and visualizations.

0.1.4 Step.1 Import libraries

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix,␣
↪accuracy_score
```

0.1.5 Step.2 Import Data

[2]:
```python
df = pd.read_csv('Iris.csv')
df.head()
```

[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

0.1.6 Step.3 Quick info

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

[4]:
```
df.describe()
```

[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

[5]:
```
df.isnull().sum()
```

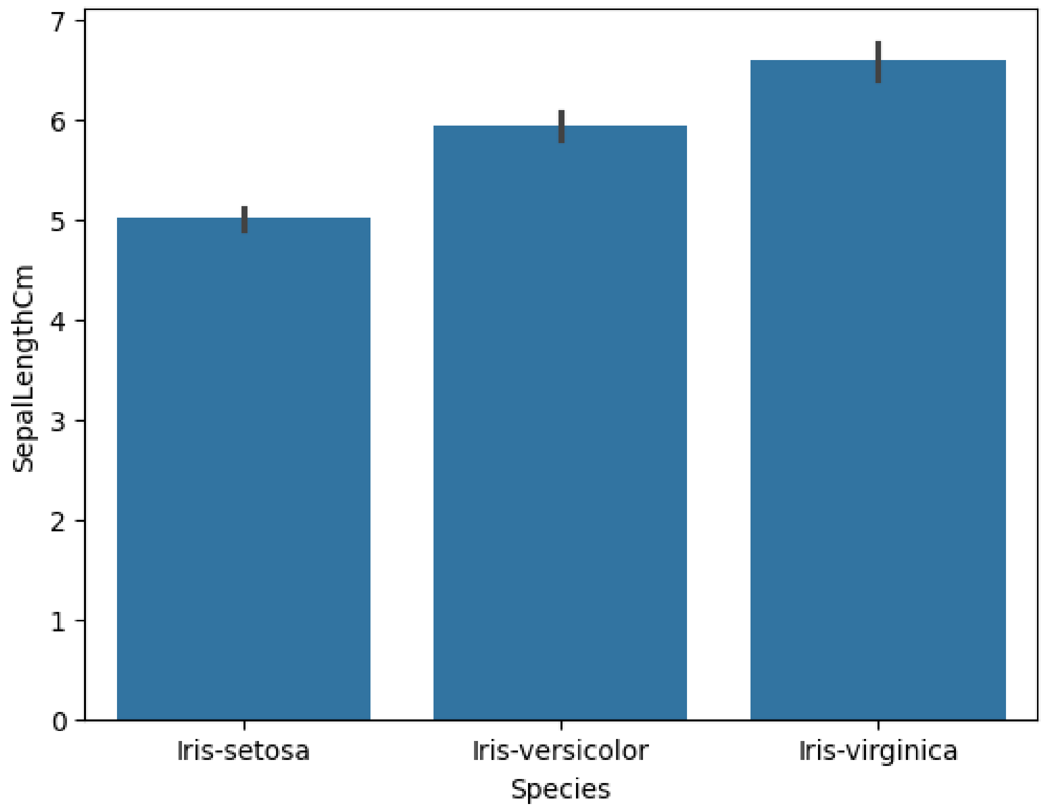
[5]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

[6]:
```
df['Species'].value_counts()
```

[6]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

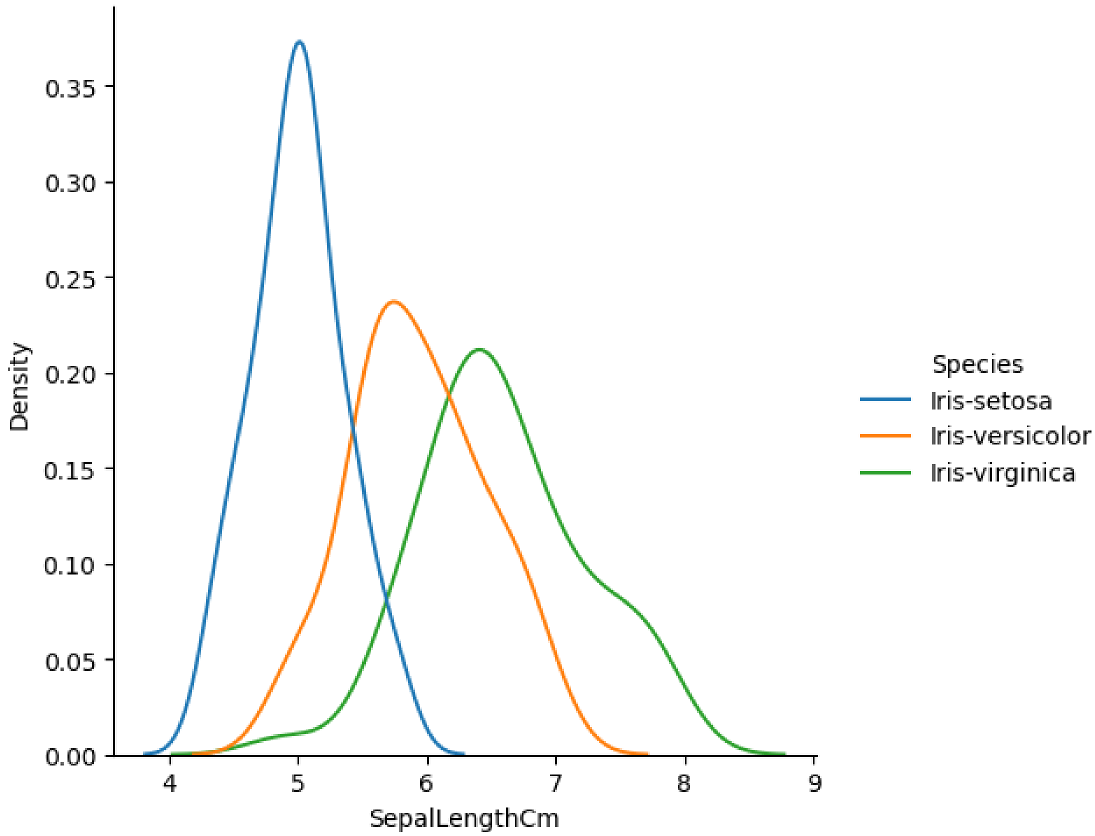
0.1.7 Step.4 Data Visualization

[7]:
```
sns.barplot(x='Species', y='SepalLengthCm', data=df)
plt.show()
```

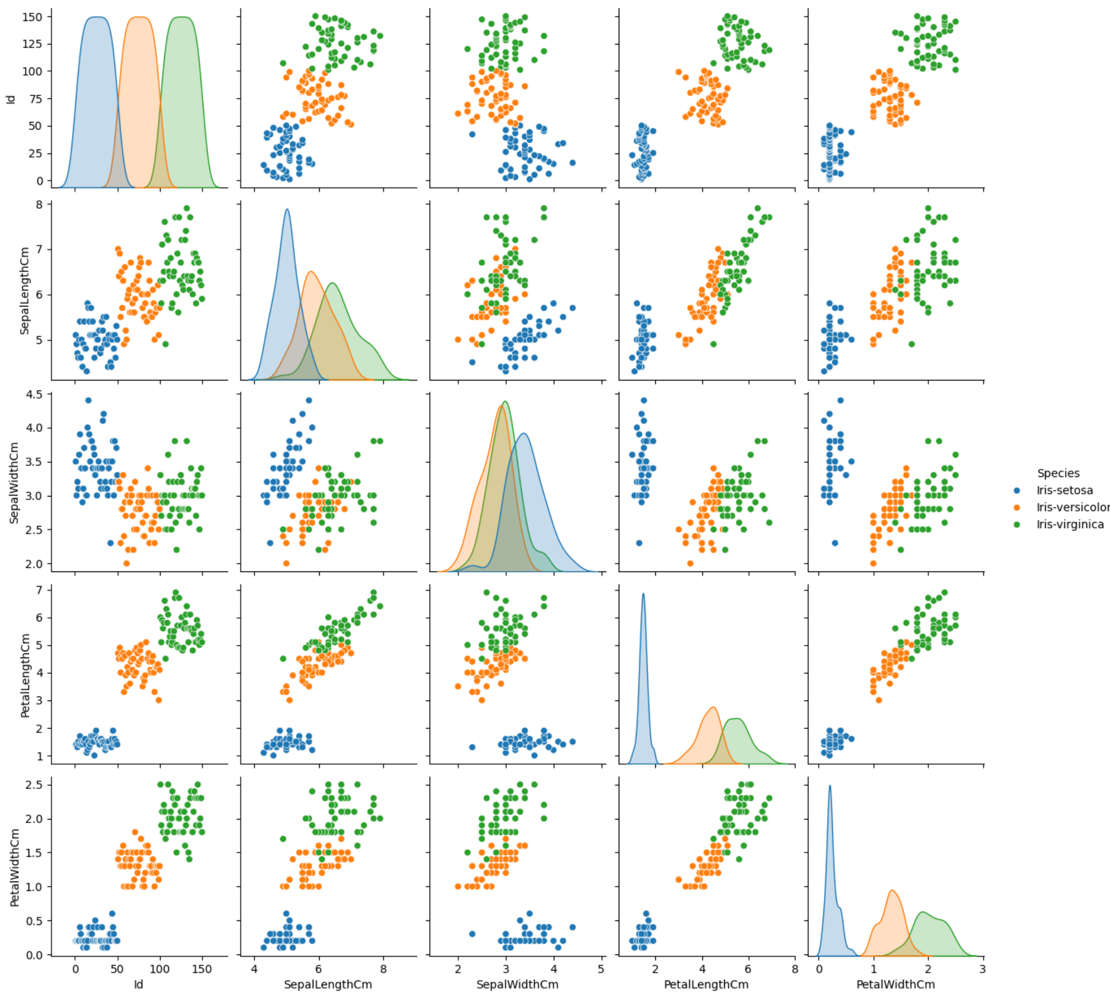


[8]:
```
sns.displot(data=df, x='SepalLengthCm', hue='Species', kind='kde')
```

[8]: <seaborn.axisgrid.FacetGrid at 0x239287fe440>



[9]:
```
sns.pairplot(df, hue='Species')
plt.show()
```



0.1.8 Step.5 Preprocessing

[10]:
```
# Drop the Id column (case-insensitive)
df.drop(columns=['Id', 'id'], errors='ignore', inplace=True)
```

[11]:
```
# Encode the Species column to numeric
codes, uniques = pd.factorize(df['Species'])
df['varites'] = codes

# view the mapping
species_mapping = {label: code for code, label in enumerate(uniques)}
print("Species mapping (label -> code):", species_mapping)

df[['Species', 'varites']].head()
```

```
Species mapping (label -> code): {'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-
virginica': 2}
```

[11]:
```
        Species  varites
0  Iris-setosa        0
1  Iris-setosa        0
2  Iris-setosa        0
3  Iris-setosa        0
4  Iris-setosa        0
```

[12]:
```
df.head()
```

[12]:
```
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      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

   varites
0        0
1        0
2        0
3        0
4        0
```

0.1.9 Step.6 Featue Selection

[13]:
```
X = df.drop(columns=['Species', 'varites'])
y = df['varites']
```

0.1.10 Step.7 Split data

[14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)
```

0.1.11 Step.8 Models

[15]:
```
dt = DecisionTreeClassifier(random_state=42)
rf = RandomForestClassifier(random_state=42)
svm = SVC(random_state=42)
```

0.1.12 Step.9 Train Models

[16]:
```
dt.fit(X_train, y_train)
```

[16]:
```
DecisionTreeClassifier(random_state=42)
```

[17]:
```
rf.fit(X_train, y_train)
```

[17]: RandomForestClassifier(random_state=42)

[18]:
```
svm.fit(X_train, y_train)
```

[18]: SVC(random_state=42)

0.1.13 Step.10 Predictions

[19]:
```
y_pred_dt = dt.predict(X_test)
y_pred_rf = rf.predict(X_test)
y_pred_svm = svm.predict(X_test)
```

0.1.14 Step.11 Evaluation

[20]:
```
# Print accuracy for each model on the test set
acc_dt = accuracy_score(y_test, y_pred_dt)
acc_rf = accuracy_score(y_test, y_pred_rf)
acc_svm = accuracy_score(y_test, y_pred_svm)

print(f"Decision Tree Accuracy: {acc_dt:.3f}")
print(f"Random Forest Accuracy: {acc_rf:.3f}")
print(f"SVM Accuracy: {acc_svm:.3f}")
```

```
Decision Tree Accuracy: 1.000
Random Forest Accuracy: 1.000
SVM Accuracy: 1.000
```

[21]:
```
print("Decision Tree Classifier Report:\n", classification_report(y_test,␣
↪y_pred_dt))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
```

```
Decision Tree Classifier Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

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

Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
```

[22]:
```
print("Random Forest Classifier Report:\n", classification_report(y_test,␣
 ↪y_pred_rf))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
```

```
Random Forest Classifier Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

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

Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
```

[23]:
```
print("SVM Classifier Report:\n", classification_report(y_test, y_pred_svm))
```

```
SVM Classifier Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

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

0.1.15 Step.12 Conclusion

n this study, three supervised machine learning models (Decision Tree, Random Forest, and Support Vector Machine) were applied to the Iris dataset to classify flower species based on sepal and petal measurements. All models achieved strong performance with high accuracy, precision, recall, and F1-scores, indicating that the dataset is well-suited for classification tasks.

[24]:
```
! jupyter nbconvert --to pdf classification_of_iris.ipynb
```

```
[NbConvertApp] Converting notebook classification_of_iris.ipynb to pdf
[NbConvertApp] Support files will be in classification_of_iris_files\
[NbConvertApp] Making directory .\classification_of_iris_files
[NbConvertApp] Writing 44794 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
```

```
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | b had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 494611 bytes to classification_of_iris.pdf
```