# DecisionTree

November 21, 2024

[1]:

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

from sklearn.preprocessing import LabelEncoder
#for train test splitting
from sklearn.model_selection import train_test_split
#for decision tree object
from sklearn.tree import DecisionTreeClassifier
#for checking testing results
from sklearn.metrics import classification_report, confusion_matrix
#for visualizing tree
from sklearn.tree import plot_tree
```

[2]:

```
#reading the data
df = sns.load_dataset('iris')
df.to_csv('iris.csv')
df.head()
```

[2]:

```
   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
```

[3]:

```
df.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
```

[4]:
```
df.shape
```

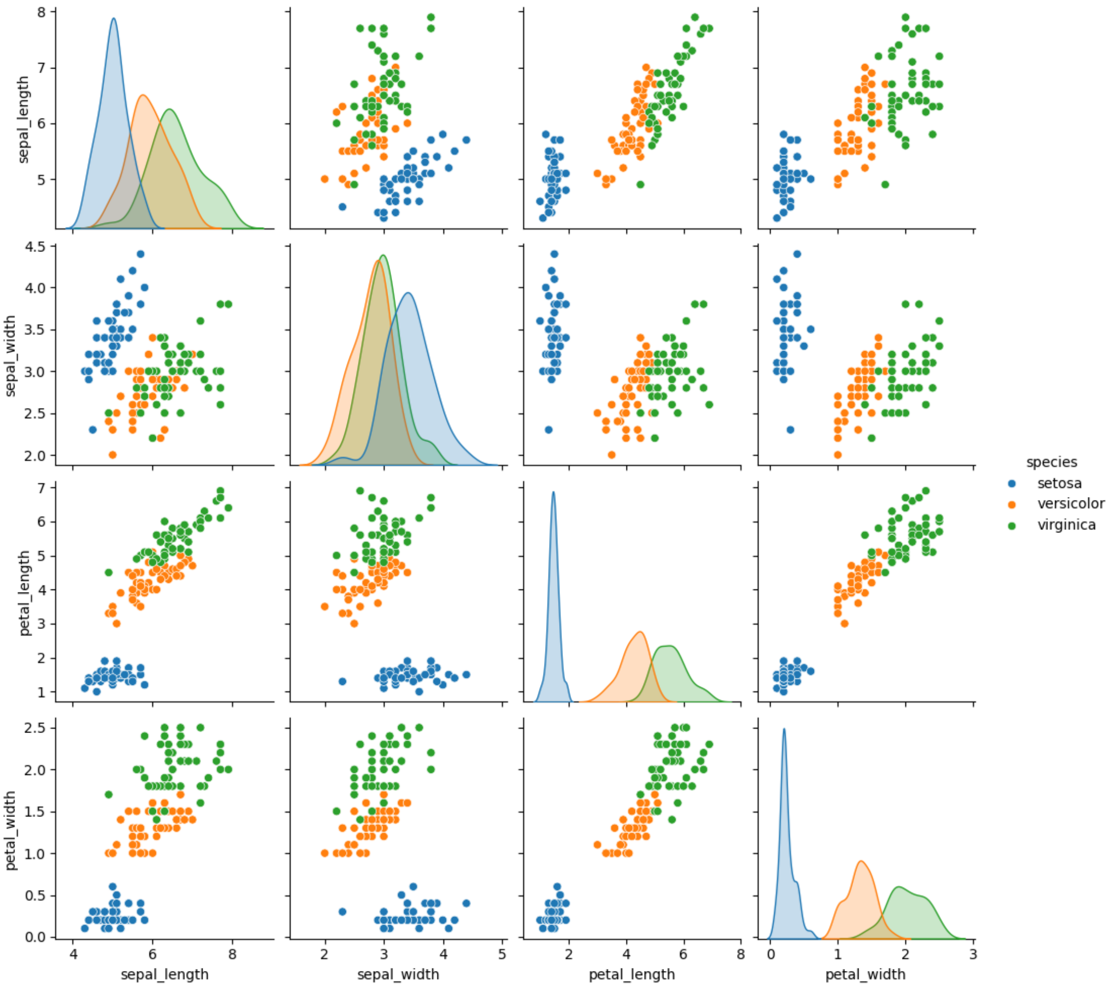
[4]:
```
(150, 5)
```

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

[5]:
```
sepal_length    False
sepal_width     False
petal_length    False
petal_width     False
species         False
dtype: bool
```

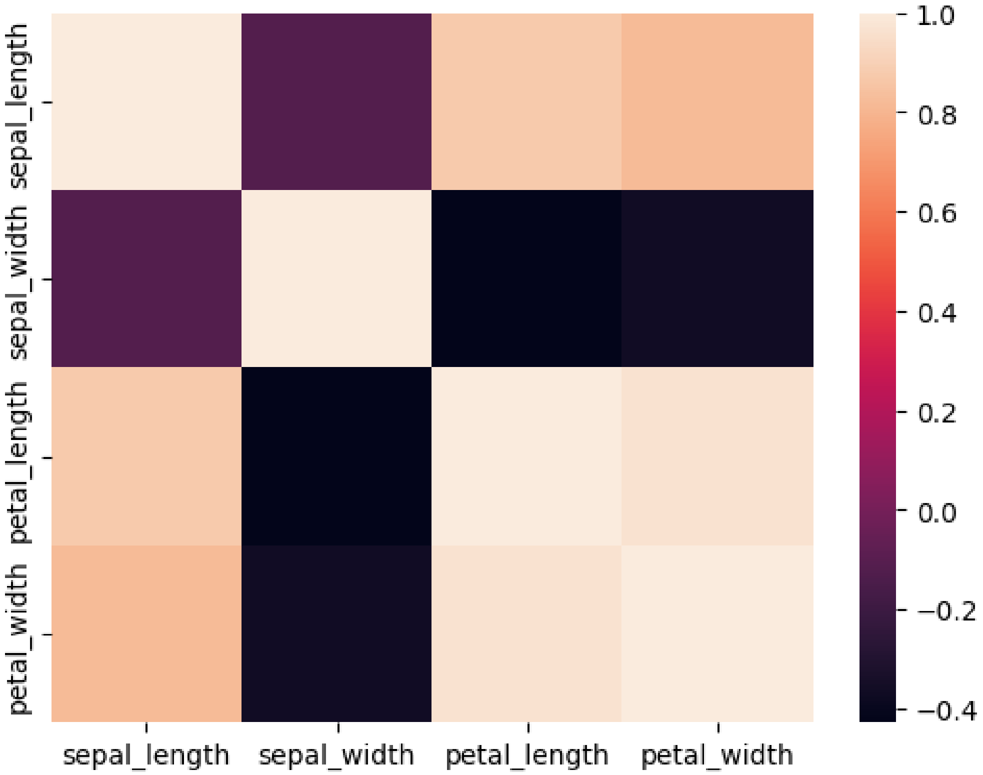
[6]:
```
sns.pairplot(data=df, hue = 'species')
```

[6]:
```
<seaborn.axisgrid.PairGrid at 0x1913e9d81a0>
```



[9]:
```
numeric_df = df.select_dtypes(include=["number"])
sns.heatmap(numeric_df.corr())
```

[9]:
```
<Axes: >
```



[10]:
```
target = df['species']
df1 = df.copy()
df1 = df1.drop('species', axis =1)
```

[11]:
```
X = df1
```

[12]:
```
#label encoding
le = LabelEncoder()
target = le.fit_transform(target)
target
```

[12]:
```
array([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])
```

[13]:

```
y = target
```

[14]:

```
# Splitting the data - 80:20 ratio
X_train, X_test, y_train, y_test = train_test_split(X , y, test_size = 0.2,␣
 ↪random_state = 42)
print("Training split input- ", X_train.shape)
print("Testing split input- ", X_test.shape)
```

```
Training split input-  (120, 4)
Testing split input-  (30, 4)
```

[15]:

```
# Defining the decision tree algorithm
dtree=DecisionTreeClassifier()
dtree.fit(X_train,y_train)
print('Decision Tree Classifier Created')
```

```
Decision Tree Classifier Created
```

[16]:

```
# Predicting the values of test data
y_pred = dtree.predict(X_test)
print("Classification report - \n", classification_report(y_test,y_pred))
```

```
Classification report -
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

[17]:

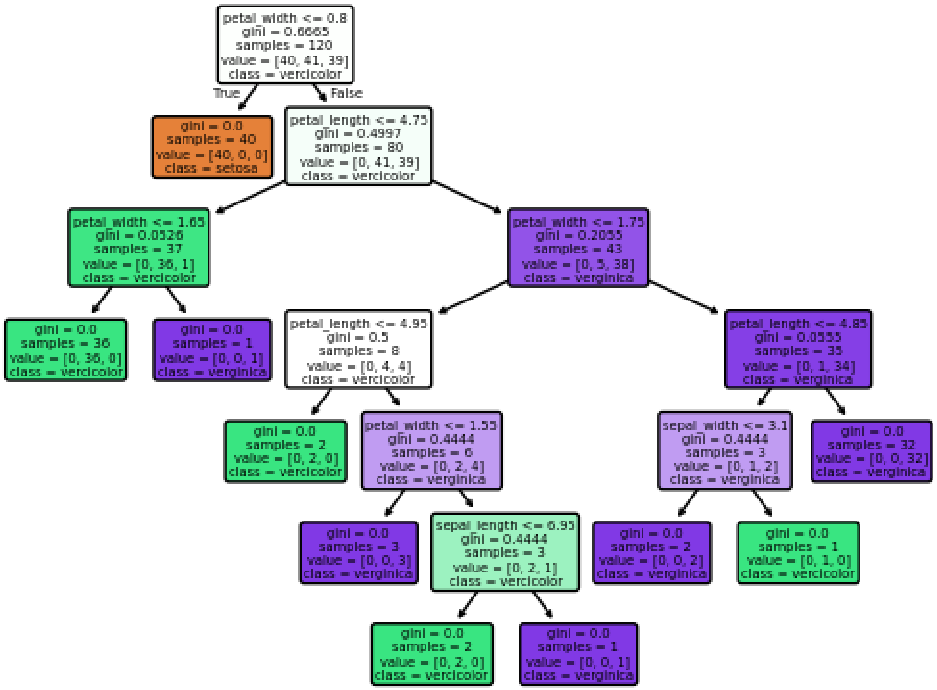
```
# confusion matrix
cf_matrix = confusion_matrix(y_test,y_pred)
print(cf_matrix)
```

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

[18]:

```
# Visualising the graph without the use of graphvizplt.figure(figsize = (10,10))
```

```
dec_tree = plot_tree(decision_tree=dtree, feature_names = df1.
↪columns,class_names =["setosa", "vercicolor", "verginica"] , filled = True ,␣
↪precision = 4, rounded = True)
```


petal_width <= 0.8
gini = 0.6665
samples = 120
value = [40, 41, 39]
class = vercicolor
True
False
gini = 0.0
samples = 40
value = [40, 0, 0]
class = setosa
petal_length <= 4.75
gini = 0.4997
samples = 80
value = [0, 41, 39]
class = vercicolor
petal_width <= 1.65
gini = 0.0526
samples = 37
value = [0, 36, 1]
class = vercicolor
petal_width <= 1.75
gini = 0.2055
samples = 43
value = [0, 5, 38]
class = verginica
gini = 0.0
samples = 36
value = [0, 36, 0]
class = vercicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = verginica
petal_length <= 4.95
gini = 0.5
samples = 8
value = [0, 4, 4]
class = vercicolor
petal_length <= 4.85
gini = 0.0555
samples = 35
value = [0, 1, 34]
class = verginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = vercicolor
petal_width <= 1.55
gini = 0.4444
samples = 6
value = [0, 2, 4]
class = verginica
sepal_width <= 3.1
gini = 0.4444
samples = 3
value = [0, 1, 2]
class = verginica
gini = 0.0
samples = 32
value = [0, 0, 32]
class = verginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = verginica
sepal_length <= 6.95
gini = 0.4444
samples = 3
value = [0, 2, 1]
class = vercicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = verginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = vercicolor
gini = 0.0
samples = 2
value = [0, 2, 0]
class = vercicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = verginica