In [111…
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import  DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import classification_report
from sklearn import preprocessing
from sklearn import metrics
```

In [112…
```
import seaborn as sns
iris=sns.load_dataset("iris")
iris
```

Out[112…

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [113…
```
iris.sample(5)
```

Out[113…

	sepal_length	sepal_width	petal_length	petal_width	species
141	6.9	3.1	5.1	2.3	virginica
131	7.9	3.8	6.4	2.0	virginica
85	6.0	3.4	4.5	1.6	versicolor
20	5.4	3.4	1.7	0.2	setosa
71	6.1	2.8	4.0	1.3	versicolor

In [114…
```
iris.shape
```

Out[114… (150, 5)

In [115…
```
iris.isna().sum()
```

Out[115…
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [116…
```
#Complete Iris dataset
label_encoder = preprocessing.LabelEncoder()
iris['species']= label_encoder.fit_transform(iris['species'])
```

In [117…
```
iris.tail(10)
```

Out[117…

	sepal_length	sepal_width	petal_length	petal_width	species
140	6.7	3.1	5.6	2.4	2
141	6.9	3.1	5.1	2.3	2
142	5.8	2.7	5.1	1.9	2
143	6.8	3.2	5.9	2.3	2
144	6.7	3.3	5.7	2.5	2
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

In [118…
```
#iris['species'].unique().ipynb_checkpoints/
```

In [119…
```
x=iris.iloc[:,0:4]
y=iris['species']
```

In [120…
```
x
```

Out[120…

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [121…
```
y
```

Out[121…
```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: species, Length: 150, dtype: int64
```

In [122…
```
iris['species'].unique()
```

Out[122…
```
array([0, 1, 2])
```

In [123…
```
iris.species.value_counts() # Checking balanced or inbalanced
```

Out[123…
```
species
0    50
1    50
2    50
Name: count, dtype: int64
```

In [124…
```
colnames = list(iris.columns)
colnames
```

Out[124…
```
['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
```

In [125…
```
# Splitting data into training and testing data set
x_train, x_test,y_train,y_test = train_test_split(x,y, test_size=0.2,random_state=15)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[125… ((120, 4), (30, 4), (120,), (30,))

In [126…
```
y_train
```

Out[126…
```
0      0
143    2
72     1
116    2
20     0
      ..
85     1
128    2
119    2
133    2
140    2
Name: species, Length: 120, dtype: int64
```

In [127…
```
import numpy as np
y_train.value_counts()
```

Out[127…
```
species
0    42
2    41
1    37
Name: count, dtype: int64
```

Building Decision Tree Classifier using gini Criteria

In [128…
```
model = DecisionTreeClassifier(criterion = 'gini',min_samples_split=5,max_depth=10)
model.fit(x_train,y_train)
```

Out[128…

▾ DecisionTreeClassifier

```
DecisionTreeClassifier(max_depth=10, min_samples_split=5)
```

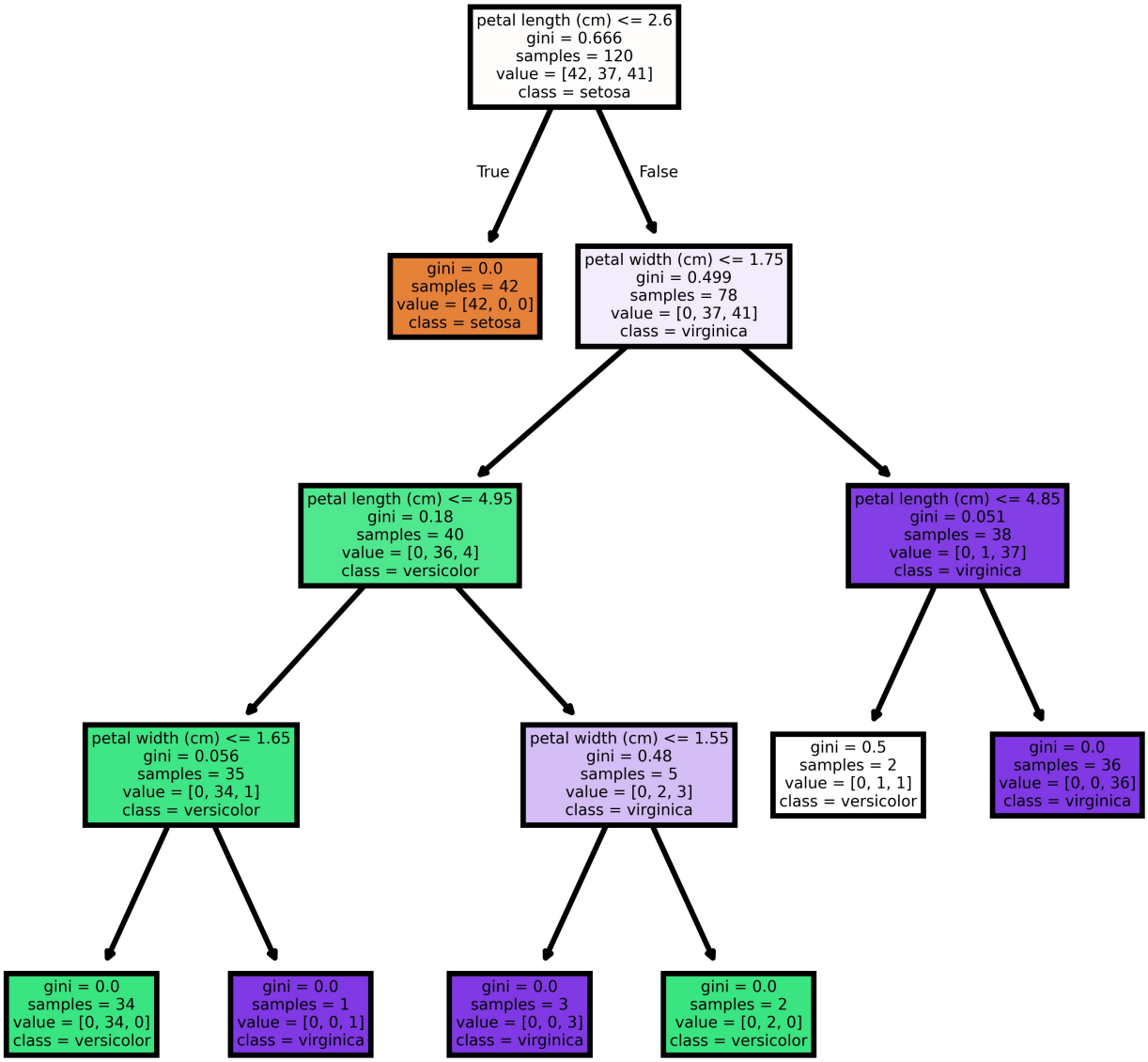
In [129…
```
cr=gini,entropy
mss=3,4,5,6,7,8
md=8,12,15

manual trail and error
gris search CV
random search CV
```

```
  Cell In[129], line 5
    manual trail and error
           ^
SyntaxError: invalid syntax
```

In [130…
```
#pip install sklearn
```

In [131…
```
fn=['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)'] # grid search and random sea
cn=['setosa', 'versicolor', 'virginica']
fig, axes = plt.subplots(figsize = (4,4), dpi=1000)
tree.plot_tree(model,
               feature_names = fn,
               class_names=cn,
               filled = True);
```



In [132…
```
preds_train=model.predict(x_train)
pd.Series(preds_train).value_counts()
```

Out[132…
```
0    42
2    40
1    38
Name: count, dtype: int64
```

In [133…
```
#Predicting on test data
preds = model.predict(x_test) # predicting on test data set
pd.Series(preds).value_counts() # getting the count of each category
```

Out[133…
```
1    15
0     8
2     7
Name: count, dtype: int64
```

In [134…
```
from sklearn import metrics
metrics.accuracy_score(preds,y_test)
```

Out[134…
```
0.9333333333333333
```

In [135…
```
print(classification_report(y_test,preds))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       0.87      1.00      0.93        13
           2       1.00      0.78      0.88         9

    accuracy                           0.93        30
   macro avg       0.96      0.93      0.93        30
weighted avg       0.94      0.93      0.93        30
```

Building Decision Tree Classifier (CART) using entropyCriteria

In [136…
```
p1=42/120
p2=37/120
p3=1-p1-p2
-p1*np.log2(p1) -p2*np.log2(p2) -p3*np.log2(p3)
```

Out[136… `np.float64(1.5828344416558475)`

In [137…
```
#gini= 1-p1^2 -p2^2 -p3^2  # 1-(1/n)

#entropy: -p1log2(p1) -p2log2(p2) -p3log2(p3)  # log2(n)
```

In [138…
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='entropy', min_samples_split=5)
```

In [139…
```
y_train
```

Out[139…
```
0      0
143    2
72     1
116    2
20     0
      ..
85     1
128    2
119    2
133    2
140    2
Name: species, Length: 120, dtype: int64
```

In [140…
```
model.fit(x_train, y_train)
```

Out[140…
```
DecisionTreeClassifier(criterion='entropy', min_samples_split=5)
```

In [141…
```
#Prediction and computing the accuracy
preds=model.predict(x_test)
metrics.accuracy_score(preds,y_test)
```

Out[141… `0.9333333333333333`

Decision Tree Regression Example

In [142…
```
# Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor
```

In [143…
```
iris.head()
```

Out[143…

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [144…
```
array = iris.values
X = array[:,0:3]
y = array[:,3]
```

In [154…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,shuffle=True, random_state=1) # shuffl
```

In [155…
```
model = DecisionTreeRegressor(max_depth=3)
model.fit(X_train, y_train)
```

Out[155…
```
DecisionTreeRegressor(max_depth=3)
```

In [156…
```
#Find the mse
from sklearn import metrics
pred=model.predict(X_test)
metrics.mean_squared_error(y_test,pred)
```

Out[156… 0.034387819189342396

In [157…
```
metrics.mean_absolute_percentage_error(y_test,pred)
```

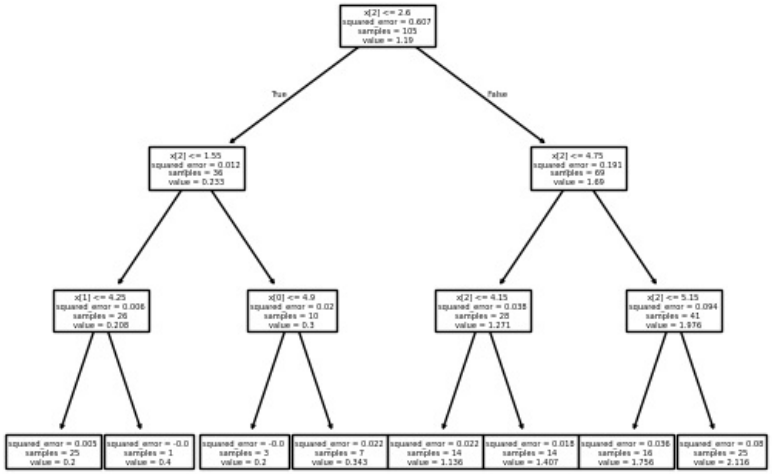
Out[157… 0.11464556401147771

In [158…
```
np.mean(np.abs(y_test-pred)/np.array(y_test))
```

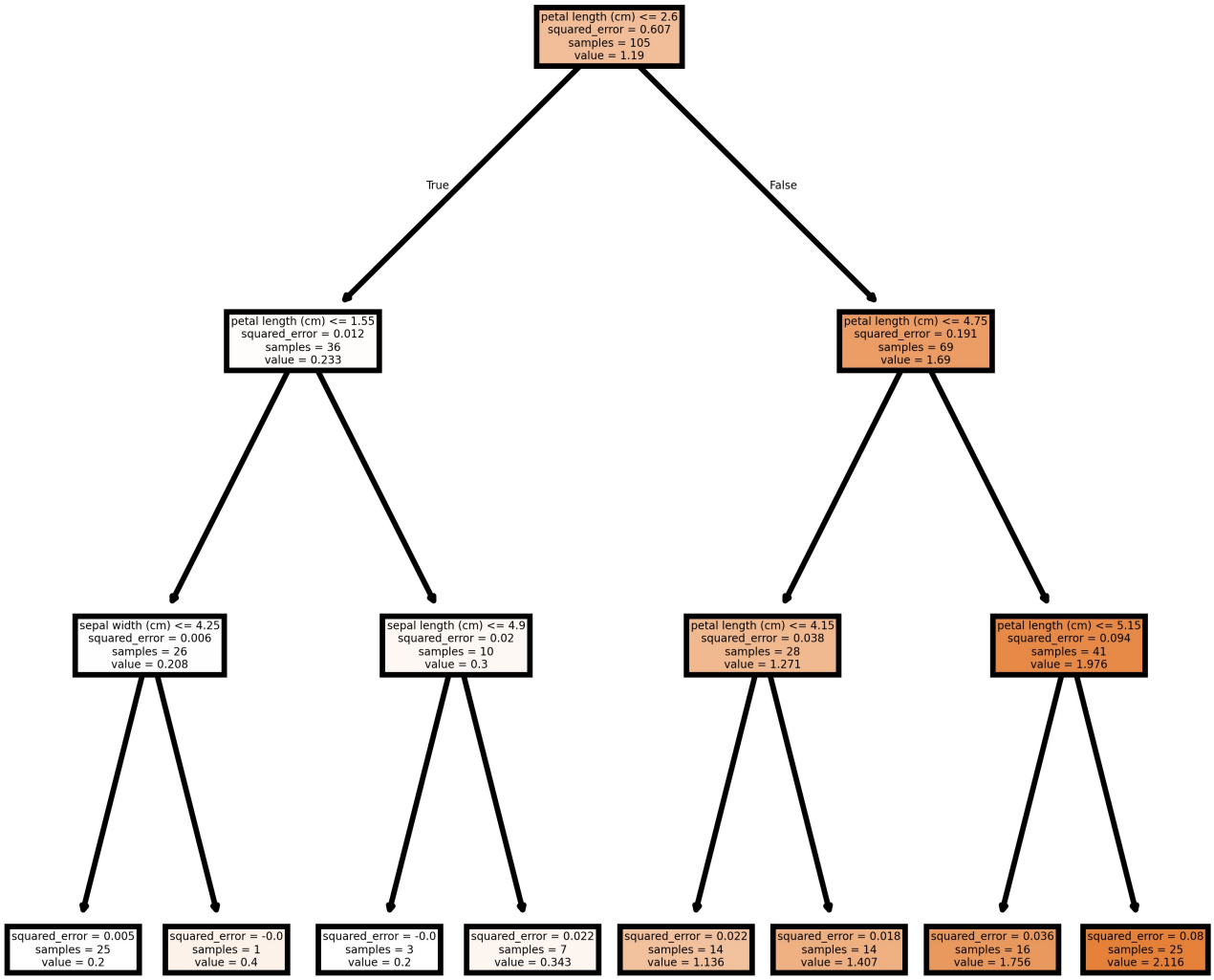
Out[158… np.float64(0.11464556401147771)

In [159…
```
tree.plot_tree(model)
```

Out[159…
```
[Text(0.5, 0.875, 'x[2] <= 2.6\nsquared_error = 0.607\nsamples = 105\nvalue = 1.19'),
 Text(0.25, 0.625, 'x[2] <= 1.55\nsquared_error = 0.012\nsamples = 36\nvalue = 0.233'),
 Text(0.375, 0.75, 'True  '),
 Text(0.125, 0.375, 'x[1] <= 4.25\nsquared_error = 0.006\nsamples = 26\nvalue = 0.208'),
 Text(0.0625, 0.125, 'squared_error = 0.005\nsamples = 25\nvalue = 0.2'),
 Text(0.1875, 0.125, 'squared_error = -0.0\nsamples = 1\nvalue = 0.4'),
 Text(0.375, 0.375, 'x[0] <= 4.9\nsquared_error = 0.02\nsamples = 10\nvalue = 0.3'),
 Text(0.3125, 0.125, 'squared_error = -0.0\nsamples = 3\nvalue = 0.2'),
 Text(0.4375, 0.125, 'squared_error = 0.022\nsamples = 7\nvalue = 0.343'),
 Text(0.75, 0.625, 'x[2] <= 4.75\nsquared_error = 0.191\nsamples = 69\nvalue = 1.69'),
 Text(0.625, 0.75, '  False'),
 Text(0.625, 0.375, 'x[2] <= 4.15\nsquared_error = 0.038\nsamples = 28\nvalue = 1.271'),
 Text(0.5625, 0.125, 'squared_error = 0.022\nsamples = 14\nvalue = 1.136'),
 Text(0.6875, 0.125, 'squared_error = 0.018\nsamples = 14\nvalue = 1.407'),
 Text(0.875, 0.375, 'x[2] <= 5.15\nsquared_error = 0.094\nsamples = 41\nvalue = 1.976'),
 Text(0.8125, 0.125, 'squared_error = 0.036\nsamples = 16\nvalue = 1.756'),
 Text(0.9375, 0.125, 'squared_error = 0.08\nsamples = 25\nvalue = 2.116')]
```



In [160…
```
fn=['sepal length (cm)','sepal width (cm)','petal length (cm)']
cn=['petal width (cm)']
fig, axes = plt.subplots(figsize = (4,4), dpi=1000)
tree.plot_tree(model,
               feature_names = fn,
               class_names=cn,
               filled = True);
```



In [163…
```
import numpy as np
np.log(6)
```

Out[163… np.float64(1.791759469228055)

In [164…
```
np.sqrt(6)
```

Out[164… np.float64(2.449489742783178)