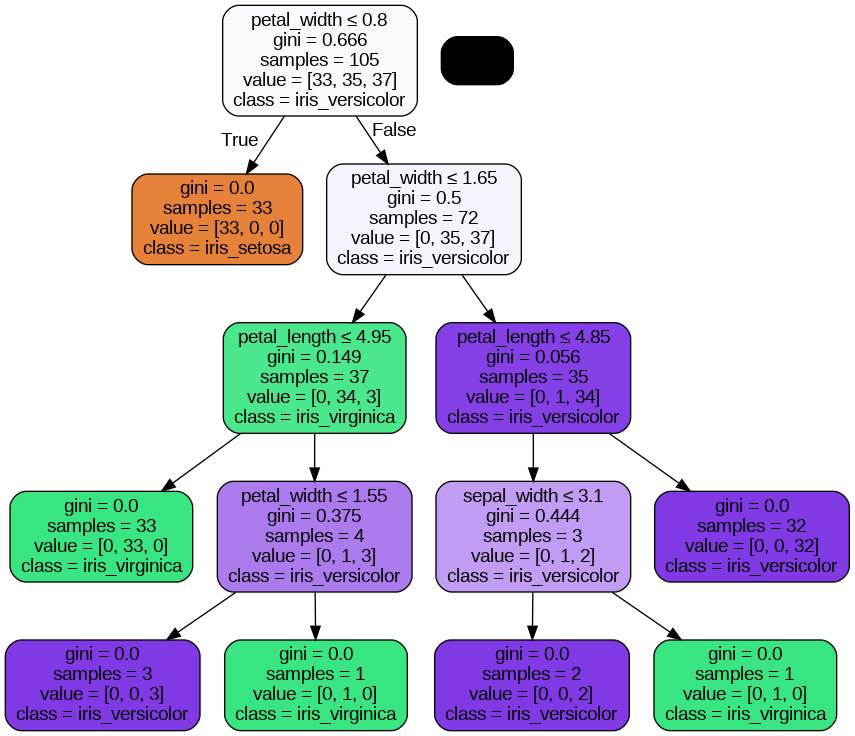
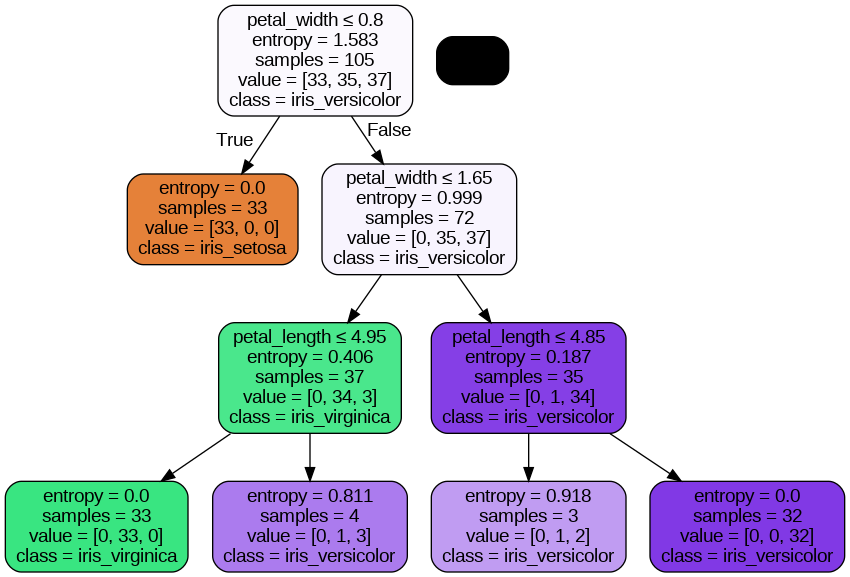
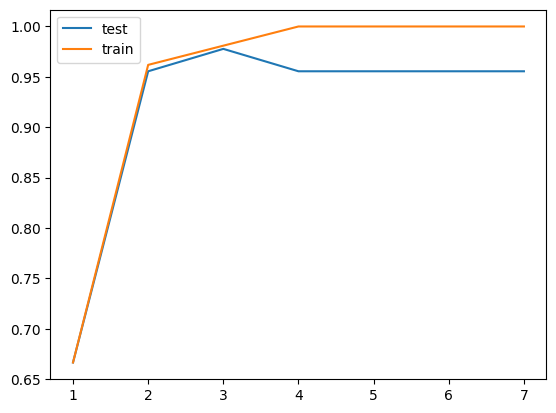
Import all the libraries  
import pandas as pd  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn import metrics  
Next steps:  
sepal\_length  
sepal\_width  
petal\_length  
petal\_width  
type  
1  
5.1  
3.5  
1.4  
0.2  
Iris-setosa  
2  
4.9  
3.0  
1.4  
0.2  
Iris-setosa  
3  
4.7  
3.2  
1.3  
0.2  
Iris-setosa  
4  
4.6  
3.1  
1.5  
0.2  
Iris-setosa  
5  
5.0  
3.6  
1.4  
0.2  
Iris-setosa  
#load the iris dataset  
col\_names=['sepal\_length','sepal\_width','petal\_length','petal\_width','type']  
pima=pd.read\_csv("/content/Iris.csv",header=0,names=col\_names)  
pima.head()  
Generate code with pima  
View recommended plots  
toggle\_off  
New interactive sheet  
#feature columns declared  
feature\_cols=['sepal\_length','sepal\_width','petal\_length',"petal\_width"]  
X=pima[feature\_cols]  
y=pima.type  
X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3,random\_state=2)  
clf=DecisionTreeClassifier()  
clf=clf.fit(X\_train,y\_train)  
y\_pred=clf.predict(X\_test)  
print("Accuracy:",metrics.accuracy\_score(y\_test,y\_pred))  
Accuracy: 0.9555555555555556  
Visualisation  
pip install graphviz  
Requirement already satisfied: graphviz in /usr/local/lib/python3.11/dist-packages (0.20.3)  
pip install pydotplus  
Requirement already satisfied: pydotplus in /usr/local/lib/python3.11/dist-packages (2.0.2)  
Requirement already satisfied: pyparsing>=2.0.1 in /usr/local/lib/python3.11/dist-packages (from pydotplus) (3.2.1)  
import six  
import sys  
sys.modules['sklearn.externals.six']=six  
#visualizing the whole decision tree  
from sklearn.tree import export\_graphviz  
from IPython.display import Image  
import pydotplus  
from sklearn.externals.six import StringIO  
dot\_data=StringIO()  
export\_graphviz(clf,out\_file=dot\_data,  
                filled=True,rounded=True,  
                special\_characters=True,feature\_names=feature\_cols,class\_names=['iris\_setosa','iris\_virginica','iris\_versicolor'])  
graph=pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())  
graph.write\_png('iris.png')  
Image(graph.create\_png())  
Code  
add  
Text  
add  
2/24/25, 8:07 PM  
Decision Tree Classifier.ipynb - Colab  
https://colab.research.google.com/drive/1wpymUqLUuQhpfuFxKkziINRnxOvt4g3B#scrollTo=A0bK1trHMANd&printMode=true  
1/7



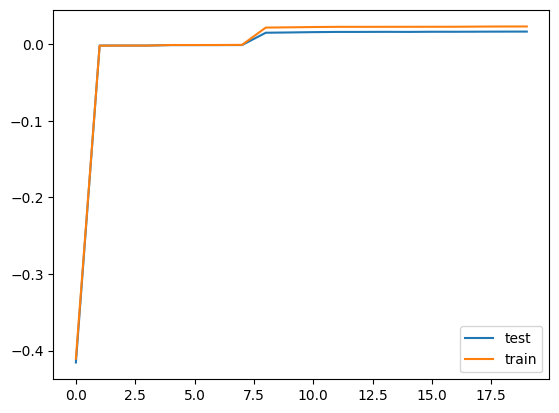
PREDICTION USING ENTROPY  
method:entropy  
split:random  
depth:3  
new\_tree=DecisionTreeClassifier(criterion='entropy',max\_depth=3)  
new\_tree=new\_tree.fit(X\_train,y\_train)  
y\_pred=new\_tree.predict(X\_test)  
print("Accuracy:",metrics.accuracy\_score(y\_test,y\_pred))  
Accuracy: 0.9777777777777777  
#visualize thie tree  
dot\_data=StringIO()  
export\_graphviz(new\_tree,out\_file=dot\_data,  
                filled=True,rounded=True,  
                special\_characters=True,feature\_names=feature\_cols,class\_names=['iris\_setosa','iris\_virginica','iris\_versicolor'])  
graph=pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())  
graph.write\_png('iris.png')  
Image(graph.create\_png())  
2/24/25, 8:07 PM  
Decision Tree Classifier.ipynb - Colab  
https://colab.research.google.com/drive/1wpymUqLUuQhpfuFxKkziINRnxOvt4g3B#scrollTo=A0bK1trHMANd&printMode=true  
2/7



#max depth is the variable name used through documentation of decision tree  
test\_scores=[]  
train\_scores=[]  
for i in range(1,8):  
  clf=DecisionTreeClassifier(max\_depth=i)  
  clf=clf.fit(X\_train,y\_train)  
  y\_pred\_test = clf.predict(X\_test)  
  test\_scores.append(metrics.accuracy\_score(y\_test,y\_pred\_test))  
  y\_pred\_train = clf.predict(X\_train)  
  train\_scores.append(metrics.accuracy\_score(y\_train,y\_pred\_train))  
  print("\ntrain:",train\_scores[i-1])  
  print("test:",test\_scores[i-1])  
print("\ntrain:",train\_scores)  
print("test:",test\_scores)  
train: 0.6666666666666666  
test: 0.6666666666666666  
train: 0.9619047619047619  
test: 0.9555555555555556  
train: 0.9809523809523809  
test: 0.9777777777777777  
train: 1.0  
test: 0.9555555555555556  
train: 1.0  
test: 0.9555555555555556  
train: 1.0  
test: 0.9555555555555556  
train: 1.0  
test: 0.9555555555555556  
train: [0.6666666666666666, 0.9619047619047619, 0.9809523809523809, 1.0, 1.0, 1.0, 1.0]  
test: [0.6666666666666666, 0.9555555555555556, 0.9777777777777777, 0.9555555555555556, 0.9555555555555556, 0.9555555555555556, 0.955  
import matplotlib.pyplot as plt  
plt.plot(range(1,8),test\_scores,label="test")  
plt.plot(range(1,8),train\_scores,label="train")  
plt.legend()  
plt.show()  
2/24/25, 8:07 PM  
Decision Tree Classifier.ipynb - Colab  
https://colab.research.google.com/drive/1wpymUqLUuQhpfuFxKkziINRnxOvt4g3B#scrollTo=A0bK1trHMANd&printMode=true  
3/7



Decision Tree Regressor  
from sklearn.tree import DecisionTreeRegressor  
from sklearn.preprocessing import LabelEncoder,OneHotEncoder  
Just have a look at the datset  
df=pd.read\_csv('/content/carprediction.csv')  
df.shape  
(11914, 16)  
Next steps:  
df.head()  
Make  
Model  
Year  
Engine  
Fuel  
Type  
Engine  
HP  
Engine  
Cylinders  
Transmission  
Type  
Driven\_Wheels  
Number  
of  
Doors  
Market Category  
Vehicle  
Size  
Vehicle  
Style  
highwa  
MP  
0  
BMW  
1  
Series  
M  
2011  
premium  
unleaded  
(required)  
335.0  
6.0  
MANUAL  
rear wheel drive  
2.0  
Factory  
Tuner,Luxury,High-  
Performance  
Compact  
Coupe  
2  
1  
BMW  
1  
Series  
2011  
premium  
unleaded  
(required)  
300.0  
6.0  
MANUAL  
rear wheel drive  
2.0  
Luxury,Performance  
Compact  
Convertible  
2  
2  
BMW  
1  
Series  
2011  
premium  
unleaded  
(required)  
300.0  
6.0  
MANUAL  
rear wheel drive  
2.0  
Luxury,High-  
Performance  
Compact  
Coupe  
2  
premium  
Generate code with df  
View recommended plots  
toggle\_off  
New interactive sheet  
df['Make'].unique()  
array(['BMW', 'Audi', 'FIAT', 'Mercedes-Benz', 'Chrysler', 'Nissan',  
 'Volvo', 'Mazda', 'Mitsubishi', 'Ferrari', 'Alfa Romeo', 'Toyota',  
 'McLaren', 'Maybach', 'Pontiac', 'Porsche', 'Saab', 'GMC',  
 'Hyundai', 'Plymouth', 'Honda', 'Oldsmobile', 'Suzuki', 'Ford',  
 'Cadillac', 'Kia', 'Bentley', 'Chevrolet', 'Dodge', 'Lamborghini',  
 'Lincoln', 'Subaru', 'Volkswagen', 'Spyker', 'Buick', 'Acura',  
 'Rolls-Royce', 'Maserati', 'Lexus', 'Aston Martin', 'Land Rover',  
 'Lotus', 'Infiniti', 'Scion', 'Genesis', 'HUMMER', 'Tesla',  
 'Bugatti'], dtype=object)  
df['Market Category'].unique()  
array(['Factory Tuner,Luxury,High-Performance', 'Luxury,Performance',  
 'Luxury,High-Performance', 'Luxury', 'Performance', 'Flex Fuel',  
 'Flex Fuel,Performance', nan, 'Hatchback',  
 'Hatchback,Luxury,Performance', 'Hatchback,Luxury',  
2/24/25, 8:07 PM  
Decision Tree Classifier.ipynb - Colab  
https://colab.research.google.com/drive/1wpymUqLUuQhpfuFxKkziINRnxOvt4g3B#scrollTo=A0bK1trHMANd&printMode=true  
4/7



'Luxury,High-Performance,Hybrid', 'Diesel,Luxury',  
 'Hatchback,Performance', 'Hatchback,Factory Tuner,Performance',  
 'High-Performance', 'Factory Tuner,High-Performance',  
 'Exotic,High-Performance', 'Exotic,Factory Tuner,High-Performance',  
 'Factory Tuner,Performance', 'Crossover', 'Exotic,Luxury',  
 'Exotic,Luxury,High-Performance', 'Exotic,Luxury,Performance',  
 'Factory Tuner,Luxury,Performance', 'Flex Fuel,Luxury',  
 'Crossover,Luxury', 'Hatchback,Factory Tuner,Luxury,Performance',  
 'Crossover,Hatchback', 'Hybrid', 'Luxury,Performance,Hybrid',  
 'Crossover,Luxury,Performance,Hybrid',  
 'Crossover,Luxury,Performance',  
 'Exotic,Factory Tuner,Luxury,High-Performance',  
 'Flex Fuel,Luxury,High-Performance', 'Crossover,Flex Fuel',  
 'Diesel', 'Hatchback,Diesel', 'Crossover,Luxury,Diesel',  
 'Crossover,Luxury,High-Performance',  
 'Exotic,Flex Fuel,Factory Tuner,Luxury,High-Performance',  
 'Exotic,Flex Fuel,Luxury,High-Performance',  
 'Exotic,Factory Tuner,Luxury,Performance', 'Hatchback,Hybrid',  
 'Crossover,Hybrid', 'Hatchback,Luxury,Hybrid',  
 'Flex Fuel,Luxury,Performance', 'Crossover,Performance',  
 'Luxury,Hybrid', 'Crossover,Flex Fuel,Luxury,Performance',  
 'Crossover,Flex Fuel,Luxury', 'Crossover,Flex Fuel,Performance',  
 'Hatchback,Factory Tuner,High-Performance', 'Hatchback,Flex Fuel',  
 'Factory Tuner,Luxury',  
 'Crossover,Factory Tuner,Luxury,High-Performance',  
 'Crossover,Factory Tuner,Luxury,Performance',  
 'Crossover,Hatchback,Factory Tuner,Performance',  
 'Crossover,Hatchback,Performance', 'Flex Fuel,Hybrid',  
 'Flex Fuel,Performance,Hybrid',  
 'Crossover,Exotic,Luxury,High-Performance',  
 'Crossover,Exotic,Luxury,Performance', 'Exotic,Performance',  
 'Exotic,Luxury,High-Performance,Hybrid', 'Crossover,Luxury,Hybrid',  
 'Flex Fuel,Factory Tuner,Luxury,High-Performance',  
 'Performance,Hybrid', 'Crossover,Factory Tuner,Performance',  
 'Crossover,Diesel', 'Flex Fuel,Diesel',  
 'Crossover,Hatchback,Luxury'], dtype=object)  
0  
Make  
0  
Model  
0  
Year  
0  
Engine Fuel Type  
3  
Engine HP  
69  
Engine Cylinders  
30  
Transmission Type  
0  
Driven\_Wheels  
0  
Number of Doors  
6  
Market Category  
3742  
Vehicle Size  
0  
Vehicle Style  
0  
highway MPG  
0  
city mpg  
0  
Popularity  
0  
MSRP  
0  
dtype: int64  
df.isnull().sum()  
#market category has a lot of null values so just drop it.  
df.drop('Market Category',axis=1,inplace=True)  
df.dropna(subset=['Engine Fuel Type'],inplace=True)  
df.dropna(subset=['Number of Doors'],inplace=True)  
df['Engine HP'].fillna(df['Engine HP'].mean(), inplace=True)  
<ipython-input-122-5a390c7cf914>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as  
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting   
2/24/25, 8:07 PM  
Decision Tree Classifier.ipynb - Colab  
https://colab.research.google.com/drive/1wpymUqLUuQhpfuFxKkziINRnxOvt4g3B#scrollTo=A0bK1trHMANd&printMode=true  
5/7

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col  
 df['Engine HP'].fillna(df['Engine HP'].mean(), inplace=True)  
df['Engine Cylinders'].fillna(df['Engine Cylinders'].mean(), inplace=True)  
<ipython-input-123-5744757632e8>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained as  
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting   
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col  
 df['Engine Cylinders'].fillna(df['Engine Cylinders'].mean(), inplace=True)  
#label encode everything and make everything in int  
label\_encoder = LabelEncoder()  
df['Make'] = label\_encoder.fit\_transform(df['Make'])  
df['Engine Fuel Type'] = label\_encoder.fit\_transform(df['Engine Fuel Type'])  
df['Transmission Type'] = label\_encoder.fit\_transform(df['Transmission Type'])  
df['Driven\_Wheels'] = label\_encoder.fit\_transform(df['Driven\_Wheels'])  
#make some new columns  
model\_encoded = pd.get\_dummies(df['Model'], prefix='Model')  
df = pd.concat([df, model\_encoded], axis=1)  
df.drop('Model', axis=1, inplace=True)  
vehi\_encoded = pd.get\_dummies(df['Vehicle Style'], prefix='Vehi Style')  
df = pd.concat([df, vehi\_encoded], axis=1)  
df.drop('Vehicle Style', axis=1, inplace=True)  
label\_encoder = LabelEncoder()  
df['Vehicle Size'] = label\_encoder.fit\_transform(df['Vehicle Size'])  
#drop the target column  
X = df.drop('MSRP',axis=1)  
y = df['MSRP']  
  #test data is 0.3 ratio  
  X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3,random\_state=2)  
clf = DecisionTreeRegressor()  
clf = clf.fit(X\_train, y\_train)  
#accuracy of the decision tree regressor  
y\_pred = clf.predict(X\_test)  
print("Accuracy using mean squared error:",metrics.mean\_squared\_error(y\_test,y\_pred))  
print("Accuracy using r2 score:",metrics.r2\_score(y\_test,y\_pred))  
print("Accuracy using mean absolute error:",metrics.mean\_absolute\_error(y\_test,y\_pred))  
Accuracy using mean squared error: 523142010.539856  
Accuracy using r2 score: 0.8548376834285294  
Accuracy using mean absolute error: 3873.573319190453  
test\_scores=[]  
train\_scores=[]  
for i in range(20):  
  clf=DecisionTreeClassifier(max\_depth=i+1)  
  clf=clf.fit(X\_train,y\_train)  
  y\_pred\_test = clf.predict(X\_test)  
  test\_scores.append(metrics.r2\_score(y\_test,y\_pred\_test))  
  y\_pred\_train = clf.predict(X\_train)  
  train\_scores.append(metrics.r2\_score(y\_train,y\_pred\_train))  
  print("\ntrain:",train\_scores[i-1])  
  print("test:",test\_scores[i-1])  
print("\ntrain:",train\_scores)  
print("test:",test\_scores)  
2/24/25, 8:07 PM  
Decision Tree Classifier.ipynb - Colab  
https://colab.research.google.com/drive/1wpymUqLUuQhpfuFxKkziINRnxOvt4g3B#scrollTo=A0bK1trHMANd&printMode=true  
6/7

test: 0.0017260873371334196  
train: -0.0017526600889616883  
test: -0.0017327980801302711  
train: -0.0010182216484455608  
test: -0.0011384992883058942  
train: -0.0010153120434814866  
test: -0.0011373615013798855  
train: -0.000989635141867895  
test: -0.0011210303459008308  
train: -0.0008411559436047877  
test: -0.001065674501414593  
train: 0.021724669989140688  
test: 0.014987679283434585  
train: 0.022075267396192477  
test: 0.015367422514509177  
train: 0.02245951366341592  
test: 0.015785329496219247  
train: 0.022701504245575488  
test: 0.016064592663677435  
train: 0.022710663338641024  
test: 0.016075654617184854  
train: 0.022734489394290014  
test: 0.016186845786707926  
train: 0.02275492097581222  
test: 0.016092604458199822  
train: 0.022789657499598603  
test: 0.01628848117561399  
train: 0.022811580509766283  
test: 0.016271572527964184  
train: 0.023031304472964753  
test: 0.016388656880062746  
train: 0.023162219938825057  
test: 0.01647260115401661  
train: [-0.41055951091745424, -0.0018972855393510901, -0.0017489055129844555, -0.0017526600889616883, -0.0010182216484455608, -0.  
test: [-0.4154071504146677, -0.0017430200251671302, -0.0017260873371334196, -0.0017327980801302711, -0.0011384992883058942, -0.00  
plt.plot(test\_scores,label="test")  
plt.plot(train\_scores, label="train")  
plt.legend()  
plt.show()  
2/24/25, 8:07 PM  
Decision Tree Classifier.ipynb - Colab  
https://colab.research.google.com/drive/1wpymUqLUuQhpfuFxKkziINRnxOvt4g3B#scrollTo=A0bK1trHMANd&printMode=true  
7/7