Decision Trees

February 22, 2020

[1]:
```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn import preprocessing
from sklearn.model_selection import KFold
```

0.1 General functions for the Decision Trees

[2]:
```
class color:
   PURPLE = '\033[95m'
   CYAN = '\033[96m'
   DARKCYAN = '\033[36m'
   BLUE = '\033[94m'
   GREEN = '\033[92m'
   YELLOW = '\033[93m'
   RED = '\033[91m'
   BOLD = '\033[1m'
   UNDERLINE = '\033[4m'
   END = '\033[0m'

def splitdataset(dataset):

    # Separating the target variable
    X = dataset.values[:, :dataset.shape[1]-2]
    Y = dataset.values[:, dataset.shape[1]-1]
    return X, Y

def train_using_gini(X_train, y_train):

    # Creating the classifier object
    clf_gini = DecisionTreeClassifier(criterion = "gini")
    # Performing training
```

```
    clf_gini.fit(X_train, y_train)
    return clf_gini

def train_using_gini_prune(X_train, y_train):

    # Creating the classifier object
    clf_gini = DecisionTreeClassifier(criterion = "gini" , ccp_alpha=0.015)
    # Performing training
    clf_gini.fit(X_train, y_train)
    return clf_gini


def train_using_entropy(X_train, y_train):

    # Decision tree with entropy
    clf_entropy = DecisionTreeClassifier(criterion = "entropy")
    # Performing training
    clf_entropy.fit(X_train, y_train)
    return clf_entropy

def train_using_entropy_prune(X_train, y_train):

    # Decision tree with entropy
    clf_entropy = DecisionTreeClassifier(criterion = "entropy", ccp_alpha=0.015)
    # Performing training
    clf_entropy.fit(X_train, y_train)
    return clf_entropy


def prediction(X_test, clf_object):

    # Predicton on test with giniIndex
    y_pred = clf_object.predict(X_test)
    return y_pred

def cal_accuracy(y_test, y_pred, i):    # Feel free to remove the comments to␣
↪see the full output

    # print("Confusion Matrix: ",
    # confusion_matrix(y_test, y_pred))
    # print("Report : ",
    # classification_report(y_test, y_pred))
    accuracy = accuracy_score(y_test,y_pred)*100
    # print ("Accuracy for test case ", i, " : ", accuracy)
    return accuracy
```

0.2 10 Fold Cross Validation

0.2.1 Using Gini Index

[3]:
```
def CV_Gini():
    avg_accuracy = []

    # 10 times 10 Fold Cross Validation
    print(color.BOLD + "Decision Trees using Gini Index:" + color.END)
    for i in range(1, 10):
        kf = KFold(n_splits=10, shuffle=True) # Applying 10-Fold Cross␣
↪Validation
        for train_index, test_index in kf.split(X): # For every test case
            clf_gini = train_using_gini(X[train_index], Y[train_index]) # Apply␣
↪Decision tree using gini
            y_pred = clf_gini.predict(X[test_index])
            avg_accuracy.append(cal_accuracy(Y[test_index], y_pred, i)) #␣
↪Gather Accuracy from all the runs

    print()
    print(color.BOLD + "Best stats achieved: ", max(avg_accuracy), color.END) #␣
↪Show the best accuracy achieved
```

With Pruning

[4]:
```
def CV_Gini_Prune():
    avg_accuracy = []

    # 10 times 10 Fold Cross Validation
    print(color.BOLD + "Decision Trees using Gini Index:" + color.END)
    for i in range(1, 10):
        kf = KFold(n_splits=10, shuffle=True) # Applying 10-Fold Cross␣
↪Validation
        for train_index, test_index in kf.split(X): # For every test case
            clf_gini = train_using_gini_prune(X[train_index], Y[train_index]) #␣
↪Apply Decision tree using gini
            y_pred = clf_gini.predict(X[test_index])
            avg_accuracy.append(cal_accuracy(Y[test_index], y_pred, i)) #␣
↪Gather Accuracy from all the runs

    print()
    print(color.BOLD + "Best stats achieved: ", max(avg_accuracy), color.END) #␣
↪Show the best accuracy achieved
```

0.2.2 Using Entropy

[5]:
```
def CV_Entropy():
    avg_accuracy  = []

    print(color.BOLD + "Decision Trees using Entropy:" + color.END)
    for i in range(1, 10):
        kf = KFold(n_splits=10, shuffle=True) # Applying 10-Fold Cross␣
↪Validation
        for train_index, test_index in kf.split(X): # For every test case
            clf_entropy = train_using_entropy(X[train_index], Y[train_index])
            y_pred_entropy = prediction(X[test_index], clf_entropy)
            avg_accuracy.append(cal_accuracy(Y[test_index], y_pred_entropy, i))


    print()
    print(color.BOLD + "Best stats achieved: ", max(avg_accuracy), color.END) #␣
↪Show the best accuracy achieved
```

With Pruning

[6]:
```
def CV_Entropy_Prune():
    avg_accuracy  = []

    print(color.BOLD + "Decision Trees using Entropy:" + color.END)
    for i in range(1, 10):
        kf = KFold(n_splits=10, shuffle=True) # Applying 10-Fold Cross␣
↪Validation
        for train_index, test_index in kf.split(X): # For every test case
            clf_entropy = train_using_entropy_prune(X[train_index],␣
↪Y[train_index])
            y_pred_entropy = prediction(X[test_index], clf_entropy)
            avg_accuracy.append(cal_accuracy(Y[test_index], y_pred_entropy, i))


    print()
    print(color.BOLD + "Best stats achieved: ", max(avg_accuracy), color.END) #␣
↪Show the best accuracy achieved
```

0.3 70/30 Hold out Approach

0.3.1 Using Gini Index

[7]:
```
def HO_Gini():
    avg_accuracy = []

    print(color.BOLD + "Decision Trees using Gini Index:" + color.END)
    for i in range(1,100):
```

```
        # Splitting the dataset into train and test
        X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.
↪3)
        clf_gini = train_using_gini(X_train, y_train) # Apply Decision tree␣
↪using gini
        y_pred = clf_gini.predict(X_test)
        avg_accuracy.append(cal_accuracy(y_test, y_pred, i)) # Gather Accuracy␣
↪from all the runs

    print()
    print(color.BOLD + "Best stats achieved: ", max(avg_accuracy), color.END) #␣
↪Show the best accuracy achieved
```

With Prune

[8]:
```
def HO_Gini_Prune():
    avg_accuracy = []

    print(color.BOLD + "Decision Trees using Gini Index:" + color.END)
    for i in range(1,100):
        # Splitting the dataset into train and test
        X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.
↪3)
        clf_gini = train_using_gini_prune(X_train, y_train) # Apply Decision␣
↪tree using gini
        y_pred = clf_gini.predict(X_test)
        avg_accuracy.append(cal_accuracy(y_test, y_pred, i)) # Gather Accuracy␣
↪from all the runs

    print()
    print(color.BOLD + "Best stats achieved: ", max(avg_accuracy), color.END) #␣
↪Show the best accuracy achieved
```

0.3.2 Using Entropy

[9]:
```
def HO_Entropy():
    avg_accuracy = []

    print(color.BOLD + "Decision Trees using Entropy:" + color.END)
    for i in range(1,100):
        # Splitting the dataset into train and test
        X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.
↪3)
        clf_entropy = train_using_entropy(X_train, y_train)
        y_pred_entropy = prediction(X_test, clf_entropy)
        avg_accuracy.append(cal_accuracy(y_test, y_pred_entropy, i))
```

```
    print()
    print(color.BOLD + "Best stats achieved: ", max(avg_accuracy), color.END) #␣
↪Show the best accuracy achieved
```

With Pruning

[10]:
```
def HO_Entropy_Prune():
    avg_accuracy = []

    print(color.BOLD + "Decision Trees using Entropy:" + color.END)
    for i in range(1,100):
        # Splitting the dataset into train and test
        X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.
↪3)
        clf_entropy = train_using_entropy_prune(X_train, y_train)
        y_pred_entropy = prediction(X_test, clf_entropy)
        avg_accuracy.append(cal_accuracy(y_test, y_pred_entropy, i))

    print()
    print(color.BOLD + "Best stats achieved: ", max(avg_accuracy), color.END) #␣
↪Show the best accuracy achieved
```

[11]:
```
global X
global Y
```

0.4 Applying on different Datasets

0.4.1 Dataset 1 Credit Approval

[12]:
```
dataset1 = pd.read_csv(
'https://archive.ics.uci.edu/ml/machine-learning-databases/credit-screening/crx.
↪data',
sep= ',', header = None)

# Printing the dataset shape
print("Dataset Length: ", len(dataset1))
print("Dataset Shape: ", dataset1.shape)

# Printing the dataset obseravtions
dataset1.head()
```

```
Dataset Length:  690
Dataset Shape:  (690, 16)
```

[12]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	00202	0	+
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	00043	560	+

```
2  a  24.50  0.500  u  g  q  h  1.50  t  f   0  f  g  00280  824  +
3  b  27.83  1.540  u  g  w  v  3.75  t  t   5  t  g  00100    3  +
4  b  20.17  5.625  u  g  w  v  1.71  t  f   0  f  s  00120    0  +
```

[13]:
```
dataset1.replace('?', np.NaN, inplace=True)
dataset1.dropna(inplace=True)

le = preprocessing.LabelEncoder()
for i in range(0,16):
    dataset1[i] = le.fit_transform(dataset1[i])

dataset1
```

[13]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	153	0	1	0	12	7	30	1	1	1	0	0	68	0	0
1	0	321	93	1	0	10	3	64	1	1	6	0	0	11	114	0
2	0	88	16	1	0	10	3	36	1	0	0	0	0	94	134	0
3	1	123	46	1	0	12	7	72	1	1	5	1	0	31	3	0
4	1	42	109	1	0	12	7	41	1	0	0	0	2	37	0	0
..	..	…	…	..	..	..	..	…	..	..	..	..	..	..	…	..
685	1	51	150	2	2	4	3	30	0	0	0	0	0	89	0	1
686	0	70	25	1	0	1	7	46	0	1	2	1	0	67	101	1
687	0	96	184	2	2	5	2	46	0	1	1	1	0	67	1	1
688	1	20	7	1	0	0	7	1	0	0	0	0	0	94	129	1
689	1	193	80	1	0	1	3	107	0	0	0	1	0	0	0	1

[653 rows x 16 columns]

[14]:
```
X, Y = splitdataset(dataset1)
```

[15]:
```
CV_Gini()
```

Decision Trees using Gini Index:

Best stats achieved: 93.93939393939394

[16]:
```
CV_Gini_Prune()
```

Decision Trees using Gini Index:

Best stats achieved: 95.38461538461539

[17]:
```
CV_Entropy()
```

Decision Trees using Entropy:

Best stats achieved: 93.84615384615384

[18]:
```
CV_Entropy_Prune()
```

```
Decision Trees using Entropy:

Best stats achieved:  95.38461538461539
```

[19]:
```
HO_Gini()
```

```
Decision Trees using Gini Index:

Best stats achieved:  87.24489795918367
```

[20]:
```
HO_Gini_Prune()
```

```
Decision Trees using Gini Index:

Best stats achieved:  91.3265306122449
```

[21]:
```
HO_Entropy()
```

```
Decision Trees using Entropy:

Best stats achieved:  87.24489795918367
```

[22]:
```
HO_Entropy_Prune()
```

```
Decision Trees using Entropy:

Best stats achieved:  90.3061224489796
```

0.4.2 Dataset 2 Kinship

[23]:
```
dataset2 = pd.read_csv(
'https://archive.ics.uci.edu/ml/machine-learning-databases/letter-recognition/
 ↪letter-recognition.data',
sep= ',', header = None)

# Printing the dataset shape
print("Dataset Length: ", len(dataset2))
print("Dataset Shape: ", dataset2.shape)

# Printing the dataset obseravtions
dataset2.head()
```

```
Dataset Length:  20000
Dataset Shape:  (20000, 17)
```

[23]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	T	2	8	3	5	1	8	13	0	6	6	10	8	0	8	0	8
1	I	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10
2	D	4	11	6	8	6	10	6	2	6	10	3	7	3	7	3	9
3	N	7	11	6	6	3	5	9	4	6	4	4	10	6	10	2	8
4	G	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10

[24]:
```
col = dataset2.columns.to_list()
col
```

[24]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
```

[25]:
```
col = col[1:] + col[0:1]
col
```

[25]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 0]
```

[26]:
```
dataset2 = dataset2[col]
dataset2
```

[26]:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	0
0	2	8	3	5	1	8	13	0	6	6	10	8	0	8	0	8	T
1	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10	I
2	4	11	6	8	6	10	6	2	6	10	3	7	3	7	3	9	D
3	7	11	6	6	3	5	9	4	6	4	4	10	6	10	2	8	N
4	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10	G
…	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
19995	2	2	3	3	2	7	7	7	6	6	6	4	2	8	3	7	D
19996	7	10	8	8	4	4	8	6	9	12	9	13	2	9	3	7	C
19997	6	9	6	7	5	6	11	3	7	11	9	5	2	12	2	4	T
19998	2	3	4	2	1	8	7	2	6	10	6	8	1	9	5	8	S
19999	4	9	6	6	2	9	5	3	1	8	1	8	2	7	2	8	A

[20000 rows x 17 columns]

[27]:
```
le = preprocessing.LabelEncoder()
dataset2[0] = le.fit_transform(dataset2[0])
dataset2
```

[27]:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	0
0	2	8	3	5	1	8	13	0	6	6	10	8	0	8	0	8	19
1	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10	8
2	4	11	6	8	6	10	6	2	6	10	3	7	3	7	3	9	3
3	7	11	6	6	3	5	9	4	6	4	4	10	6	10	2	8	13
4	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10	6
…	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
19995	2	2	3	3	2	7	7	7	6	6	6	4	2	8	3	7	3

```
19996   7  10   8   8   4   4   8   6   9  12   9  13   2   9   3   7   2
19997   6   9   6   7   5   6  11   3   7  11   9   5   2  12   2   4  19
19998   2   3   4   2   1   8   7   2   6  10   6   8   1   9   5   8  18
19999   4   9   6   6   2   9   5   3   1   8   1   8   2   7   2   8   0

[20000 rows x 17 columns]
```

[28]:
```
X, Y = splitdataset(dataset2)
```

[29]:
```
CV_Gini()
```

```
Decision Trees using Gini Index:

Best stats achieved:  89.1
```

[30]:
```
CV_Gini_Prune()
```

```
Decision Trees using Gini Index:

Best stats achieved:  31.75
```

[31]:
```
CV_Entropy()
```

```
Decision Trees using Entropy:

Best stats achieved:  90.05
```

[32]:
```
CV_Entropy_Prune()
```

```
Decision Trees using Entropy:

Best stats achieved:  61.45
```

[33]:
```
HO_Gini()
```

```
Decision Trees using Gini Index:

Best stats achieved:  87.31666666666666
```

[34]:
```
HO_Gini_Prune()
```

```
Decision Trees using Gini Index:

Best stats achieved:  32.03333333333334
```

[35]:
```
HO_Entropy()
```

```
Decision Trees using Entropy:

Best stats achieved:  87.86666666666667
```

[36]:
```
HO_Entropy_Prune()
```

```
Decision Trees using Entropy:

Best stats achieved:  60.85
```

0.4.3 Dataset 3 Sonar Mines

[37]:
```
dataset3 = pd.read_csv(
'https://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/
 ↪connectionist-bench/sonar/sonar.all-data',
sep= ',', header = None)

# Printing the dataset shape
print("Dataset Length: ", len(dataset3))
print("Dataset Shape: ", dataset3.shape)

# Printing the dataset obseravtions
dataset3.head()
```

```
Dataset Length:  208
Dataset Shape:  (208, 61)
```

[37]:

	0	1	2	3	4	5	6	7	8	\
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	

	9	…	51	52	53	54	55	56	57	\
0	0.2111	…	0.0027	0.0065	0.0159	0.0072	0.0167	0.0180	0.0084	
1	0.2872	…	0.0084	0.0089	0.0048	0.0094	0.0191	0.0140	0.0049	
2	0.6194	…	0.0232	0.0166	0.0095	0.0180	0.0244	0.0316	0.0164	
3	0.1264	…	0.0121	0.0036	0.0150	0.0085	0.0073	0.0050	0.0044	
4	0.4459	…	0.0031	0.0054	0.0105	0.0110	0.0015	0.0072	0.0048	

	58	59	60
0	0.0090	0.0032	R
1	0.0052	0.0044	R
2	0.0095	0.0078	R
3	0.0040	0.0117	R
4	0.0107	0.0094	R

```
[5 rows x 61 columns]
```

[38]:
```
X, Y = splitdataset(dataset3)
```

[39]:
```
CV_Gini()
```

```
Decision Trees using Gini Index:

Best stats achieved:  85.71428571428571
```

[40]:
```
CV_Gini_Prune()
```

```
Decision Trees using Gini Index:

Best stats achieved:  100.0
```

[41]:
```
CV_Entropy()
```

```
Decision Trees using Entropy:

Best stats achieved:  90.47619047619048
```

[42]:
```
CV_Entropy_Prune()
```

```
Decision Trees using Entropy:

Best stats achieved:  100.0
```

[43]:
```
HO_Gini()
```

```
Decision Trees using Gini Index:

Best stats achieved:  84.12698412698413
```

[44]:
```
HO_Gini_Prune()
```

```
Decision Trees using Gini Index:

Best stats achieved:  80.95238095238095
```

[45]:
```
HO_Entropy()
```

```
Decision Trees using Entropy:

Best stats achieved:  85.71428571428571
```

[46]:
```
HO_Entropy_Prune()
```

Decision Trees using Entropy:

Best stats achieved: 85.71428571428571

0.4.4 Dataset 4 Cylinder Bands

[47]:
```
dataset4 = pd.read_csv("bands.data", header=None)
dataset4
```

[47]:

	0	1	2	3	4	5	6	7	8	…	9	…	30	31	32	33	34	35	36	37	38	39
0	19910108	X126	TVGUIDE	25503	YES	KEY	YES	BENTON	GALLATIN	…	UNCOATED	…	36.4	0	0	2.5	1	34	40	105	100	band
1	19910109	X266	TVGUIDE	25503	YES	KEY	YES	BENTON	GALLATIN	…	UNCOATED	…	38.5	0	0	2.5	0.7	34	40	105	100	noband
2	19910104	B7	MODMAT	47201	YES	KEY	YES	BENTON	GALLATIN	…	UNCOATED	…	39.8	0	0	2.8	0.9	40	40	103.87	100	noband
3	19910104	T133	MASSEY	39039	YES	KEY	YES	BENTON	GALLATIN	…	UNCOATED	…	38.8	0	0	2.5	1.3	40	40	108.06	100	noband
4	19910111	J34	KMART	37351	NO	KEY	YES	BENTON	GALLATIN	…	UNCOATED	…	42.5	5	0	2.3	0.6	35	40	106.67	100	noband
..	…	…	…	…	…	…	…	…	…	…	…	…	…	..	..	…	…	..	..	…	…	…
536	19941005	aa66	kmart	85813	?	key	?	?	gallatin	…	super	…	?	?	?	1	1	?	40	112.5	100	band
537	19941009	j44	best	38064	?	key	?	?	gallatin	…	super	…	?	?	?	0	0	?	40	110	100	band
538	19941009	aa58	kmart	85814	?	key	?	?	gallatin	…	super	…	?	?	?	2.7	2.8	?	40	108	100	band
539	19941010	aa70	kmart	85814	?	key	?	?	gallatin	…	super	…	?	?	?	1.5	2.3	?	40	108	100	band
540	19941010	j70	best	38064	?	key	?	?	gallatin	…	super	…	?	?	?	2.5	1	?	40	108.1	100	band

[541 rows x 40 columns]

[48]:
```
dataset4.replace('?', np.NaN, inplace=True)
dataset4.dropna(inplace=True)
dataset4.drop(columns=0, inplace=True)
dataset4
```

[48]:

	1	2	3	4	5	6	7	8	9	…	30	31	32	33	34	35	36	37	38	39
0	X126	TVGUIDE	25503	YES	KEY	YES	BENTON	GALLATIN	UNCOATED	…	36.4	0	0	2.5	1	34	40	105	100	band
3	T133	MASSEY	39039	YES	KEY	YES	BENTON	GALLATIN	UNCOATED	…	38.8	0	0	2.5	1.3	40	40	108.06	100	noband
5	T218	MASSEY	38039	YES	KEY	YES	BENTON	GALLATIN	UNCOATED	…	40	0	0	2.5	1	34	40	105	100	band
6	X249	ROSES	35751	NO	KEY	YES	BENTON	GALLATIN	COATED	…	40	0	0	2.5	1	34	40	105	100	band

```
7    X788    ROSES  35751   NO  KEY  YES  BENTON  GALLATIN    COATED
..    …       …     …  …   …  …     …       …        …
424  X242     AMES  34590   NO  KEY  YES  BENTON  GALLATIN    COATED
426  X108  ECKERDS  34693   NO  KEY  YES  BENTON  GALLATIN    COATED
427   X80  ECKERDS  34694   NO  KEY  YES  BENTON  GALLATIN    COATED
428  F482    DOWNS  35525  YES  KEY  YES  BENTON  GALLATIN  UNCOATED
429  X388  TVGUIDE  25502  YES  KEY  YES  BENTON  GALLATIN  UNCOATED

            10  …    30   31 32   33   34  35  36      37   38      39
0    UNCOATED  …  36.4    0  0  2.5    1  34  40     105  100    band
3    UNCOATED  …  38.8    0  0  2.5  1.3  40  40  108.06  100  noband
5    UNCOATED  …  37.6    5  0  2.5  0.8  40  40  103.87  100  noband
6      COATED  …  37.5    6  0  2.5  0.6  30  40  106.67  100  noband
7      COATED  …  37.5    6  0  2.5  1.1  30  40  106.67  100  noband
..        …  …   …  … ..  …  …  ..  ..     …  …     …
424    COATED  …  41.2    8  0    3    1  33  40  106.45  100  noband
426    COATED  …  37.5    1  0  2.5  1.5  30  40  106.45  100  noband
427    COATED  …  39.5  4.5  0  1.9  1.3  30  40  114.28  100  noband
428  UNCOATED  …  36.1    4  0    3    1  40  40  117.85  100  noband
429  UNCOATED  …  53.4    0  0    3  0.9  34  40  103.22  100    band

[277 rows x 39 columns]
```

[49]:
```
le = preprocessing.LabelEncoder()
for i in range(1,40):
    dataset4[i] = le.fit_transform(dataset4[i])

dataset4
```

[49]:
```
      1   2    3   4   5   6   7   8   9   10  …  30  31  32  33  34  35  \
0    143  45   17   1   0   1   0   0   1   2  …  37   0   0  18  11   4
3    117  36  139   1   0   1   0   0   1   2  …  55   0   0  18  16   7
5    119  36  135   1   0   1   0   0   1   2  …  47  12   0  18   8   7
6    167  40   72   0   0   1   0   0   0   0  …  46  13   0  18   5   1
7    211  40   72   0   0   1   0   0   0   0  …  46  13   0  18  13   1
..   …   ..   …  ..  ..  ..  ..  ..  ..  ..  …  ..  ..  ..  ..  ..  ..
424  165   1   46   0   0   1   0   0   0   0  …  74  14   0  25  11   3
426  139  18   50   0   0   1   0   0   0   0  …  46   3   0  18  19   1
427  216  18   51   0   0   1   0   0   0   0  …  62  11   0  10  16   1
428   33  15   67   1   0   1   0   0   1   2  …  34  10   0  25  11   7
429  192  45   16   1   0   1   0   0   1   2  …  95   0   0  25  10   4

     36  37  38  39
0     4  10   0   0
3     4  21   0   1
5     4   9   0   1
6     4  17   0   1
```

```
7     4  17   0   1
..   ..  ..  ..  ..
424   4  14   0   1
426   4  14   0   1
427   4  27   0   1
428   4  30   0   1
429   4   6   0   0

[277 rows x 39 columns]
```

[50]:
```
X, Y = splitdataset(dataset4)
```

[51]:
```
CV_Gini()
```

```
Decision Trees using Gini Index:

Best stats achieved:  89.28571428571429
```

[52]:
```
CV_Gini_Prune()
```

```
Decision Trees using Gini Index:

Best stats achieved:  82.14285714285714
```

[53]:
```
CV_Entropy()
```

```
Decision Trees using Entropy:

Best stats achieved:  89.28571428571429
```

[54]:
```
CV_Entropy_Prune()
```

```
Decision Trees using Entropy:

Best stats achieved:  85.71428571428571
```

[55]:
```
HO_Gini()
```

```
Decision Trees using Gini Index:

Best stats achieved:  78.57142857142857
```

[56]:
```
HO_Gini_Prune()
```

```
Decision Trees using Gini Index:

Best stats achieved:  77.38095238095238
```

[57]:
```
HO_Entropy()
```

```
Decision Trees using Entropy:

Best stats achieved:  77.38095238095238
```

[58]:
```
HO_Entropy_Prune()
```

```
Decision Trees using Entropy:

Best stats achieved:  73.80952380952381
```

0.4.5 Dataset 5 Flags

[59]:
```
dataset5 = pd.read_csv(
'https://archive.ics.uci.edu/ml/machine-learning-databases/flags/flag.data',
sep= ',', header = None)

# Printing the dataset shape
print("Dataset Length: ", len(dataset5))
print("Dataset Shape: ", dataset5.shape)

# Printing the dataset obseravtions
dataset5.head()
```

```
Dataset Length:  194
Dataset Shape:  (194, 30)
```

[59]:

	0	1	2	3	4	5	6	7	8	9	…	20	21	22	23	\
0	Afghanistan	5	1	648	16	10	2	0	3	5	…	0	0	1	0	
1	Albania	3	1	29	3	6	6	0	0	3	…	0	0	1	0	
2	Algeria	4	1	2388	20	8	2	2	0	3	…	0	0	1	1	
3	American-Samoa	6	3	0	0	1	1	0	0	5	…	0	0	0	0	
4	Andorra	3	1	0	0	6	0	3	0	3	…	0	0	0	0	

	24	25	26	27	28	29
0	0	1	0	0	black	green
1	0	0	1	0	red	red
2	0	0	0	0	green	white
3	1	1	1	0	blue	red
4	0	0	0	0	blue	red

```
[5 rows x 30 columns]
```

[60]:
```
for i in range(0,30):
    dataset5[i] = le.fit_transform(dataset5[i])
dataset5
```

[60]:

	0	1	2	3	4	5	6	7	8	9	…	20	21	22	23	24	25	\
0	0	4	0	97	16	9	2	0	3	4	…	0	0	1	0	0	1	
1	1	2	0	20	3	5	6	0	0	2	…	0	0	1	0	0	0	
2	2	3	0	126	19	7	2	2	0	2	…	0	0	1	1	0	0	
3	3	5	2	0	0	0	1	0	0	4	…	0	0	0	0	1	1	
4	4	2	0	0	0	5	0	3	0	2	…	0	0	0	0	0	0	
..	…	..	..	…	..	..	..	..	..	..	…	..	..	..	..	..	..	
189	189	5	2	3	0	0	1	0	0	2	…	0	1	5	0	0	0	
190	190	2	0	68	20	5	6	0	3	3	…	0	0	1	0	0	0	
191	191	3	1	104	22	9	5	0	0	3	…	0	0	0	0	0	1	
192	192	3	1	99	6	9	5	3	0	3	…	0	0	0	0	0	0	
193	193	3	1	82	8	9	5	0	7	4	…	0	0	1	0	1	1	

	26	27	28	29
0	0	0	0	4
1	1	0	5	6
2	0	0	3	7
3	1	0	1	6
4	0	0	1	6
..	..	..	..	..
189	0	0	1	6
190	0	0	1	6
191	1	0	3	4
192	1	0	3	2
193	1	0	3	4

[194 rows x 30 columns]

[61]:
```
X, Y = splitdataset(dataset5)
```

[62]:
```
CV_Gini()
```

Decision Trees using Gini Index:

Best stats achieved: 75.0

[63]:
```
CV_Gini_Prune()
```

Decision Trees using Gini Index:

Best stats achieved: 84.21052631578947

[64]:
```
CV_Entropy()
```

Decision Trees using Entropy:

Best stats achieved: 78.94736842105263

[65]:
```
CV_Entropy_Prune()
```

```
Decision Trees using Entropy:

Best stats achieved:  73.68421052631578
```

[66]:
```
HO_Gini()
```

```
Decision Trees using Gini Index:

Best stats achieved:  59.32203389830508
```

[67]:
```
HO_Gini_Prune()
```

```
Decision Trees using Gini Index:

Best stats achieved:  67.79661016949152
```

[68]:
```
HO_Entropy()
```

```
Decision Trees using Entropy:

Best stats achieved:  64.40677966101694
```

[69]:
```
HO_Entropy_Prune()
```

```
Decision Trees using Entropy:

Best stats achieved:  69.49152542372882
```

[ ]:
```

```