A1

February 16, 2019

In [13]:
```
import pandas as pd
import numpy as np
import sklearn as sk
import matplotlib.pyplot as plt
from os import listdir
from os.path import isfile, join
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier

%matplotlib inline
```

In [117]:
```
DATASET_PATH = "datasets/"
datasets = [f for f in listdir(DATASET_PATH) if isfile(join(DATASET_PATH, f)) and f
datasets
```

Out[117]:
```
['agaricus-lepiota.data.txt',
 'primary-tumor.data.txt',
 'hayes-roth.data.txt',
 'monks-2.test.txt',
 'nursery.data.txt',
 'lymphography.data.txt',
 'soybean-large.data.txt',
 'car.data.txt',
 'SPECT.test.txt',
 'adult+stretch.data.txt',
 'balance-scale.data.txt']
```

In [86]:
```
DATASET_PATH2 = "nonworking/"
datasets = [f for f in listdir(DATASET_PATH2) if isfile(join(DATASET_PATH2, f)) and f
datasets
```

Out[86]:
```
[]
```

In [91]:
```
dataset = 'tic-tac-toe.data.txt'
# data = pd.read_csv(DATASET_PATH + dataset, header = None)
# data = data.replace('?', np.NaN)
# data = data.dropna(axis=1)
```

```
        # data = data.drop([0], axis=0)
        # data.to_csv(dataset, index=False)
        # data

In [123]: def get_info_for_data(dataset_name, data_dict_list, n_splits=10):
              data_dict = {}

              data = pd.read_csv(DATASET_PATH + dataset_name, header = None)
              data = data.replace('?', np.NaN)
              data = data.dropna(axis=1)
              data = data.apply(lambda x: pd.factorize(x)[0])

              data_dict = {"name": dataset_name, "num_instances": data.shape[0], "num_attribut

              print()
              print(len(data_dict_list))
              print(dataset_name)
              print(data.shape)
              print()
              # print(data.head())
              # print()

              # number of attributes, Random average height, and Random average accuracy

              y_col = 0
              X = data.drop(y_col, 1)
              y = data[y_col]
              skf = StratifiedKFold(n_splits=n_splits)
              accuracies = []
              tree_depths = []
              accuracies2 = []
              tree_depths2 = []
              data_split = skf.split(X, y)
              # try:
              for train, test in data_split:

                  # C4.5 Tree:
                  model = DecisionTreeClassifier(criterion='entropy')
                  model.fit(X.loc[train] , y.loc[train])
                  y_predict = model.predict(X.loc[test])
                  accuracy = accuracy_score(y.loc[test], y_predict)
                  accuracies.append(accuracy)
                  tree_depth = dectree_max_depth(model.tree_)
                  tree_depths.append(tree_depth)

                  # Random Tree:
                  model2 = ExtraTreeClassifier(criterion='entropy', max_features=1)
                  model2.fit(X.loc[train] , y.loc[train])
```

```
                y_predict = model2.predict(X.loc[test])
                accuracy = accuracy_score(y.loc[test], y_predict)
                accuracies2.append(accuracy)
                tree_depth = dectree_max_depth(model2.tree_)
                tree_depths2.append(tree_depth)
            data_dict["C4.5 Avg Accuracy"] = sum(accuracies) / len(accuracies)
            data_dict["C4.5 Avg Height"] = sum(tree_depths) / len(tree_depths)
            data_dict["Random Avg Accuracy"] = sum(accuracies2) / len(accuracies2)
            data_dict["Random Avg Height"] = sum(tree_depths2) / len(tree_depths2)
            data_dict_list.append(data_dict)
            print(data_dict)
            print()
            # except:
            #    print("Exception!")


In [58]: def dectree_max_depth(tree):
             n_nodes = tree.node_count
             children_left = tree.children_left
             children_right = tree.children_right

             def walk(node_id):
                 if (children_left[node_id] != children_right[node_id]):
                     left_max = 1 + walk(children_left[node_id])
                     right_max = 1 + walk(children_right[node_id])
                     return max(left_max, right_max)
                 else: # leaf
                     return 1

             root_node_id = 0
             return walk(root_node_id)
         # Use: t.tree_
         # From: https://stackoverflow.com/questions/26602369/how-to-find-out-the-size-of-a-sk

In [124]: # Use a table to present your experiment results. The table should include the follo
          # dataset name, number of instances, number of attributes, number of classes, avg he

          data_dict_list = []
          for dataset in datasets: # datasets[7:-1]:
              get_info_for_data(dataset, data_dict_list, 10)


0
agaricus-lepiota.data.txt
(8124, 22)

{'name': 'agaricus-lepiota.data.txt', 'num_instances': 8124, 'num_attributes': 22, 'num_classes
```

```
1
primary-tumor.data.txt
(339, 13)

{'name': 'primary-tumor.data.txt', 'num_instances': 339, 'num_attributes': 13, 'num_classes': 4



/Users/jm/anaconda3/envs/dm/lib/python3.7/site-packages/sklearn/model_selection/_split.py:652:
  % (min_groups, self.n_splits)), Warning)



2
hayes-roth.data.txt
(133, 5)

{'name': 'hayes-roth.data.txt', 'num_instances': 133, 'num_attributes': 5, 'num_classes': 19,


3
monks-2.test.txt
(433, 7)

{'name': 'monks-2.test.txt', 'num_instances': 433, 'num_attributes': 7, 'num_classes': 23, 'C4


4
nursery.data.txt
(12960, 9)

{'name': 'nursery.data.txt', 'num_instances': 12960, 'num_attributes': 9, 'num_classes': 32, 'C


5
lymphography.data.txt
(148, 19)

{'name': 'lymphography.data.txt', 'num_instances': 148, 'num_attributes': 19, 'num_classes': 63


6
soybean-large.data.txt
(307, 2)

{'name': 'soybean-large.data.txt', 'num_instances': 307, 'num_attributes': 2, 'num_classes': 2
```

```
7
car.data.txt
(1728, 7)



/Users/jm/anaconda3/envs/dm/lib/python3.7/site-packages/sklearn/model_selection/_split.py:652:
  % (min_groups, self.n_splits)), Warning)
/Users/jm/anaconda3/envs/dm/lib/python3.7/site-packages/sklearn/model_selection/_split.py:652:
  % (min_groups, self.n_splits)), Warning)


{'name': 'car.data.txt', 'num_instances': 1728, 'num_attributes': 7, 'num_classes': 25, 'C4.5 A


8
SPECT.test.txt
(187, 23)

{'name': 'SPECT.test.txt', 'num_instances': 187, 'num_attributes': 23, 'num_classes': 46, 'C4.5


9
adult+stretch.data.txt
(20, 5)

{'name': 'adult+stretch.data.txt', 'num_instances': 20, 'num_attributes': 5, 'num_classes': 10


10
balance-scale.data.txt
(625, 5)

{'name': 'balance-scale.data.txt', 'num_instances': 625, 'num_attributes': 5, 'num_classes': 23



In [127]: # Use a table to present your experiment results. The table should include the follo
          df = pd.DataFrame(data_dict_list)
          df = df[['name', 'num_instances', 'num_attributes', 'num_classes', 'C4.5 Avg Accuracy
          df

Out[127]:                         name  num_instances  num_attributes  num_classes  \
          0   agaricus-lepiota.data.txt           8124              22          114
          1      primary-tumor.data.txt            339              13           46
          2         hayes-roth.data.txt            133               5           19
          3            monks-2.test.txt            433               7           23
          4            nursery.data.txt          12960               9           32
```

5	lymphography.data.txt	148	19	63
6	soybean-large.data.txt	307	2	21
7	car.data.txt	1728	7	25
8	SPECT.test.txt	187	23	46
9	adult+stretch.data.txt	20	5	10
10	balance-scale.data.txt	625	5	23

	C4.5 Avg Accuracy	C4.5 Avg Height	Random Avg Accuracy	Random Avg Height
0	0.968512	7.7	0.965679	17.8
1	0.311928	12.2	0.298264	13.6
2	0.193077	10.5	0.213718	10.1
3	0.405761	10.5	0.359619	12.4
4	0.439815	18.3	0.398611	18.7
5	0.785516	8.9	0.653393	13.9
6	0.163250	2.0	0.163250	2.0
7	0.356052	14.0	0.321987	14.6
8	0.886725	11.5	0.893129	14.4
9	0.500000	4.3	0.500000	4.3
10	0.671442	12.6	0.678431	14.5

0.0.1 Sources

http://www.cs.uvm.edu/~icdm/algorithms/10Algorithms-08.pdf

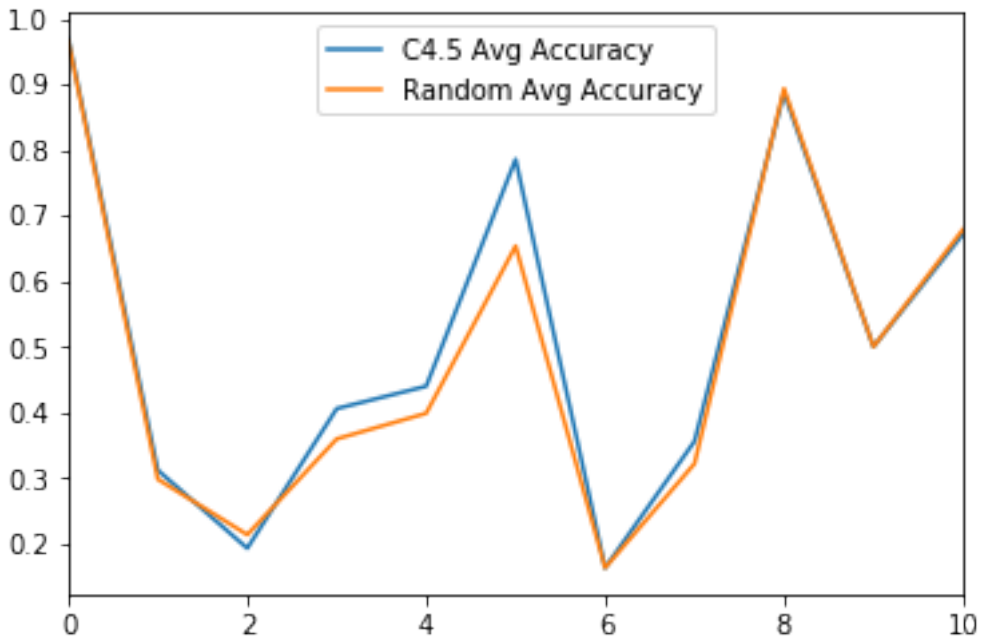
"CART uses the Gini diversity index to rank tests, whereas C4.5 uses information-based criteria"

https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart

"CART (Classification and Regression Trees) is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets." scikit-learn uses an optimised version of the CART algorithm

```
In [130]: df[['C4.5 Avg Accuracy', 'Random Avg Accuracy']].plot()
```

```
Out[130]: <matplotlib.axes._subplots.AxesSubplot at 0x1a22353ac8>
```




```
In [131]: df[['C4.5 Avg Height', 'Random Avg Height']].plot()
```

```
Out[131]: <matplotlib.axes._subplots.AxesSubplot at 0x10cc32128>
```