Dilnoza Saidova

TCSS455: Machine Learning

February 2, 2023

Homework #2

**Other resources**:

https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.cross\_val\_score.html

1a. New lines of coded added to original document are made **bold**.

#################################################################  
#  
# ML - HW 2  
#  
# Martine De Cock  
#  
#################################################################  
import pandas as pd  
from sklearn import tree  
from sklearn import metrics

**from sklearn.model\_selection import cross\_val\_score**

# Read the dataset into a dataframe and map the labels to numbers  
df = pd.read\_csv('iris.csv')  
map\_to\_int = {'setosa':0, 'versicolor':1, 'virginica':2}  
df["label"] = df["species"].replace(map\_to\_int)  
print(df)

# Separate the input features from the label  
features = list(df.columns[:4])  
X = df[features]  
y = df["label"]

# Train a decision tree and compute its training accuracy  
clf = tree.DecisionTreeClassifier(max\_depth=2, criterion='entropy')  
clf.fit(X, y)  
print(metrics.accuracy\_score(y,clf.predict(X)))

**scores = cross\_val\_score(clf, X, y, cv=10)**

**print(“Accuracy: %0.2f (+/- %0.2f)” % (scores.mean(), scores.std()\*2))**

**cv\_scores = cross\_val\_score(clf, X, y, cv=10)**

**print("Cross-validation scores: {}".format(cv\_scores))**

**print("Average accuracy: {:.2f}".format(cv\_scores.mean()))**

**accuracy = cross\_val\_score(clf, X, y, cv=10)**

**print("Accuracy using 10-fold cross-validation: %0.2f (+/- %0.2f)" %**

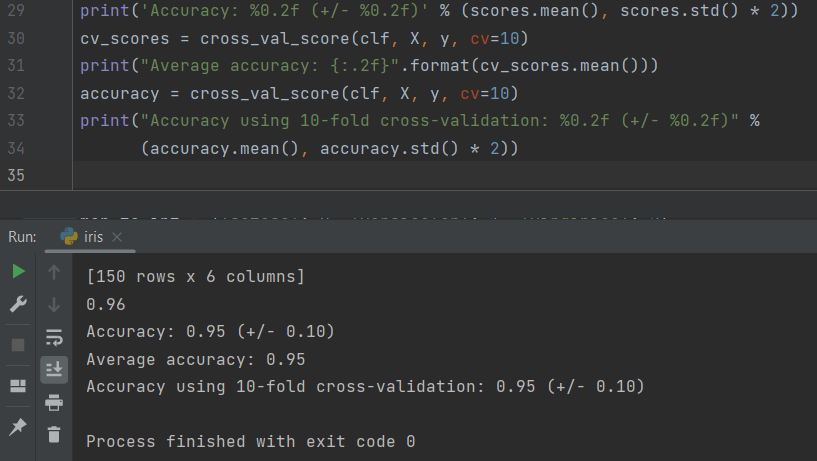
**(accuracy.mean(), accuracy.std() \* 2))**

1b.

Training accuracy: 0.96

10 – fold cross – validation accuracy: 0.95

There isn’t much difference between training and 10-fold cross-validation accuracies. Yet, 10-fold cross-validation has the lowest accuracy due to the fact that k-fold cross-validation isn’t meant to increase the performance but to provide more accurate measure of the performance.

****

2.

Text, letter

Description automatically generated

A picture containing text

Description automatically generated

# TCSS455: Machine Learning  
# Homework 2  
# Name: Martine De Cock, Dilnoza Saidova  
# Collaborator: AJ Garcia  
# Description: Training and testing decision trees with discrete-values attributes  
  
import sys  
import math  
import pandas as pd  
import operator  
from collections import Counter  
  
class DecisionNode:  
  
 # A DecisionNode contains an attribute and a dictionary of children.  
 # The attribute is either the attribute being split on,  
 # or the predicted label if the node has no children.  
 def \_\_init\_\_(self, attribute):  
 self.attribute = attribute  
 self.children = {}  
  
 # Visualizes the tree  
 def display(self, level=0):  
 if self.children == {}: # reached leaf level  
 print(": ", self.attribute, end="")  
 else:  
 for value in self.children.keys():  
 prefix = "\n" + " " \* level \* 4  
 print(prefix, self.attribute, "=", value, end="")  
 self.children[value].display(level + 1)  
 # Predicts the target label for instance x  
 def predicts(self, x):  
 if self.children == {}: # reached leaf level  
 return self.attribute  
 value = x[self.attribute]  
 subtree = self.children[value]  
 return subtree.predicts(x)  
  
  
# Illustration of functionality of DecisionNode class  
def funTree():  
 myLeftTree = DecisionNode('humidity')  
 myLeftTree.children['normal'] = DecisionNode('no')  
 myLeftTree.children['high'] = DecisionNode('yes')  
 myTree = DecisionNode('wind')  
 myTree.children['weak'] = myLeftTree  
 myTree.children['strong'] = DecisionNode('no')  
 return myTree  
  
  
def id3(examples, target, attributes):  
 examples\_list = examples.loc[:, target].tolist()  
 if examples\_list.count(examples\_list[0]) == len(examples\_list):  
 return DecisionNode(examples\_list[0])  
 elif len(attributes) == 0:  
 example\_cnt = {}  
 for example in examples\_list:  
 if example not in example\_cnt.keys():  
 example\_cnt[example] = 1  
 example\_cnt[example] += 1  
 sorted\_cnt = sorted(example\_cnt.items(),  
 key=operator.itemgetter(1),  
 reverse=True)  
 return DecisionNode(sorted\_cnt[0][0])  
 else:  
 final\_gain = 0.0  
 attr\_index = 0  
 for num in range(len(attributes)):  
 entropy = getEntropy(examples, target)  
 attr\_cnt = Counter(examples.loc[:, attributes[num]])  
 new\_entropy = 0.0  
 for key in attr\_cnt:  
 temp = examples.loc[examples[attributes[num]] == key]  
 del temp[attributes[num]]  
 mod\_examples = temp  
 new\_entropy += (attr\_cnt[key] / sum(attr\_cnt.values()))\  
 \* getEntropy(mod\_examples, target)  
 new\_gain = (entropy - new\_entropy)  
 if new\_gain > final\_gain:  
 final\_gain = new\_gain  
 attr\_index = num  
 top\_attr = attributes[attr\_index]  
 tree = DecisionNode(top\_attr)  
 top\_attr\_cnt = Counter(examples.loc[:, top\_attr])  
 values = []  
 for key in top\_attr\_cnt:  
 if key not in values:  
 values.append(key)  
 for value in values:  
 temp = examples.loc[examples[top\_attr] == value]  
 del temp[top\_attr]  
 new\_ex = temp  
 new\_attr = attributes[:]  
 new\_attr.remove(top\_attr)  
 subtree = id3(new\_ex, target, new\_attr)  
 tree.children[value] = subtree  
 return tree

def getEntropy(examples, target):  
 cnt = Counter(examples.loc[:, target])  
 sum = 0  
 for num in cnt:  
 sum += -1.0 \* (cnt[num] / len(examples))\  
 \* math.log(cnt[num] / len(examples), 2)  
 return sum  
  
#################### MAIN PROGRAM ######################  
  
# Reading input data  
train = pd.read\_csv(sys.argv[1])  
test = pd.read\_csv(sys.argv[2])  
target = sys.argv[3]  
attributes = train.columns.tolist()  
attributes.remove(target)  
  
# Learning and visualizing the tree  
tree = id3(train, target, attributes)  
tree.display()  
  
# Evaluating the tree on the test data  
correct = 0  
for i in range(0, len(test)):  
 if str(tree.predicts(test.loc[i])) == str(test.loc[i, target]):  
 correct += 1  
print("\nThe accuracy is: ", correct / len(test))