به نام خدا

حسین غلامی تمرین سری دو یادگیری ماشین – درخت تصمیم

توضحیات

در انجام این تمرین از آن جایی که فرمت اطلاعات به صورت ssv بودند به جهت ساده سازی و استفاده از متد های pandas اطلاعات را به فرمت csv در اوردم

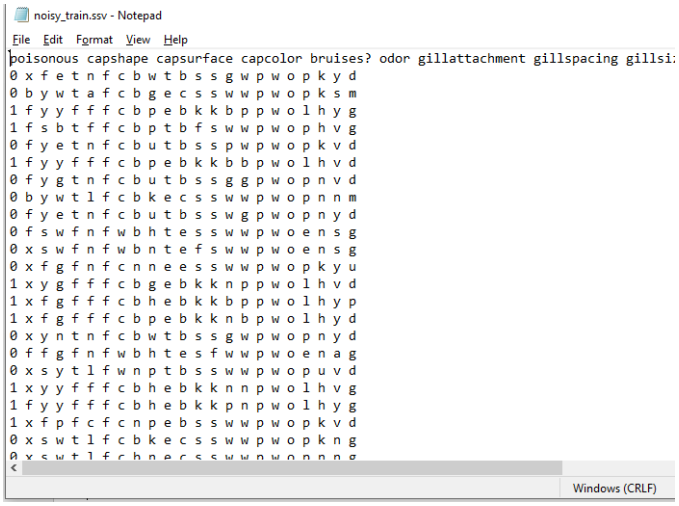
نحوه تبدیل:

ابتدا فایل را با notpad باز کرده ، و برای اینکه به فرمت dataframe، pandas در بیاید ، فایل های اضافی را حذف نموده

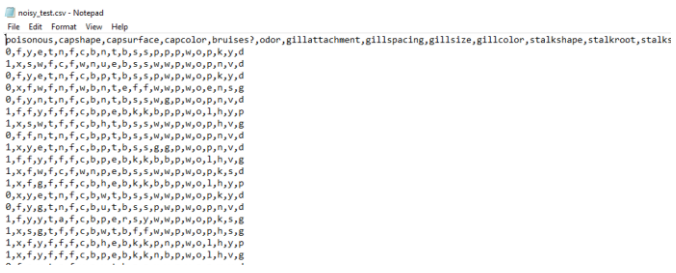
سپس فاصله را در این فرمت با کاما (,)

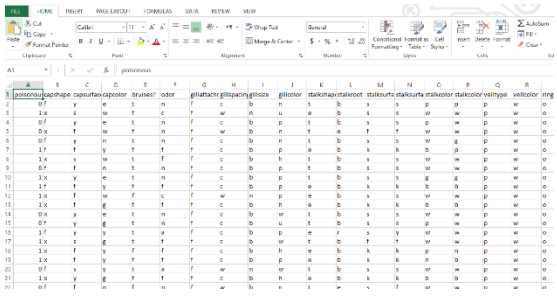
جایگزین نموده و فایل را یا فرمت csv

ذخیره میکنیم



خروجی به شکل زیر حاصل شد از آن جایی که فایل csv را در اکسل نیز میتوان مشاهده کرد ، تصویر آن نیز قرار گرفت





پیاده سازی الگوریتم :

در سه مرحله انجام شد:

1- پیاده سازی اولیه
2- بررسی کتابخانه ها و عملکردشان و نوع کارکردن با آن ها
3- پیاده سازی نهایی

پیاده سازی اولیه

به این صورت که در ابتدا ریشه درخت مشخص میشود و در ادامه شاخه ها و نود ها پس از هم محاسبه میشوند. ولی این که پس از تمام شدن درخت چگونه از روی شاخه ها بازگشت ، قابل پیاده سازی نبود ، چرا که نام ویژگی های یکسانی وجود داشت و پیاده سازی امکان پذیر نبود

```
import pandas as pd
from math import log

def entr(x,y):# x:pos y:neg
    return -((x/(x+y))*log((x/(x+y))))-((y/(x+y))*log((y/(x+y))))


path='F:/Data/noisy_train.csv'
df=pd.read_csv(path)

entropy=list()
gain=list()
tree=list()

index = df.columns.tolist()
index.remove('poisonous')
for feather in index:
    for types ,z in df.groupby([feather]):
        if(((0,types) in df.groupby(['poisonous',feather]).groups.keys())
          &((1,types) in df.groupby(['poisonous',feather]).groups.keys())):
            entropy.append(
                        entr(df.groupby(['poisonous',feather]).size()[0][types]
                            ,df.groupby(['poisonous',feather]).size()[1][types])
                        *
                        (df.groupby([feather]).size()[types]
                        /df.groupby([feather]).size().sum() )
                        )
        else:
            entropy.append(0)
    gain.append([sum(entropy),feather])
   # print(entropy)
    for types in range(0,len(entropy)):
        entropy.pop()
gain.sort()
print(gain)
root=gain[0][1]
for types in range(0,len(gain)):
    gain.pop()
```

```
#root has found

zanjir=list()
print('_____________\n')
stop=0
count =0
t=1
index.remove(root)

tree_lvl=-1

internal_counter=-1

node=root

while(stop==0):
    #mohasebe zanjir ha
    if( tree_lvl>0 ):
        stop=1

    if(count > 0):

        if(internal_counter<0):
            internal_counter=len(df.groupby([node]).size())
            tree_lvl=tree_lvl+1



        internal_counter=internal_counter-1
        if(node in index ):
            index.remove(node)


        if(len(zanjir[tree_lvl])==6):
            node=zanjir[internal_counter][3]
            index.remove(node)
        else:
            node=zanjir[count][3]
```

```
81      count =count+1
82          ##calculate zanjir
83      for branch ,z1 in df.groupby([node]):
84              for feather in index:
85                  for types ,z2 in df.groupby([feather]):
86                      if(((1,types,branch) in df.groupby(['poisonous',feather,node]).groups.keys())
87                        &((0,types,branch) in df.groupby(['poisonous',feather,node]).groups.keys())):
88                          entropy.append(
89                                  entr(df.groupby(['poisonous',feather,node]).size()[0][types][branch]
90                                      ,df.groupby(['poisonous',feather,node]).size()[1][types][branch])
91                                  *
92                                  (df.groupby([feather]).size()[types]
93                                  /df.groupby([feather]).size().sum() )
94                                  )
95                      else:
96                          entropy.append(0)
97                  GAIN=sum(entropy)
98                  for types in range(0,len(entropy)):
99                      entropy.pop()
100                 #label zadan
101                 if(GAIN==0):
102                     t=1
103                 else:
104                     gain.append([GAIN,feather])
105
106             gain.sort()
107             #print(branch,gain)
108             try:
109                 if(t==0):
110                     zanjir.append((tree_lvl,node,branch,gain[0][1]))
111                 if(t==1):
112                     zanjir.append((tree_lvl,node,branch,gain[0][1],'end'))
113                 t=0
114
115                 print(count,node,branch,gain[0][1])
116                 for types in range(0,len(gain)):
117                     gain.pop()
```

که به خروجی زیر رسیدم که شاخه های درخت را تشکیل میداد ولی برای محاسبه درخت مشکل داشت:

```
In [7]: zanjir
Out[7]:
[(-1, 'odor', 'a', 'stalkcolorbelowring', 'end'),
 (-1, 'odor', 'c', 'gillsize'),
 (-1, 'odor', 'f', 'sporeprintcolor'),
 (-1, 'odor', 'l', 'stalkcolorbelowring'),
 (-1, 'odor', 'm', 'gillattachment'),
 (-1, 'odor', 'n', 'sporeprintcolor'),
 (-1, 'odor', 'p', 'gillsize'),
 (0, 'gillsize', 'b', 'sporeprintcolor'),
 (0, 'gillsize', 'n', 'gillsize'),
 (0, 'sporeprintcolor', 'h', 'sporeprintcolor'),
 (0, 'sporeprintcolor', 'k', 'sporeprintcolor'),
 (0, 'sporeprintcolor', 'n', 'sporeprintcolor'),
 (0, 'sporeprintcolor', 'r', 'sporeprintcolor'),
 (0, 'sporeprintcolor', 'u', 'sporeprintcolor'),
 (0, 'sporeprintcolor', 'w', 'sporeprintcolor'),
 (0, 'stalkcolorbelowring', 'b', 'stalkcolorbelowring'),
 (0, 'stalkcolorbelowring', 'c', 'gillattachment'),
 (0, 'stalkcolorbelowring', 'g', 'stalkcolorbelowring'),
 (0, 'stalkcolorbelowring', 'n', 'stalkcolorbelowring'),
 (0, 'stalkcolorbelowring', 'p', 'stalkcolorbelowring'),
 (0, 'stalkcolorbelowring', 'w', 'stalkcolorbelowring'),
 (0, 'gillattachment', 'a', 'gillattachment', 'end'),
 (0, 'gillattachment', 'f', 'ringtype'),
 (0, 'sporeprintcolor', 'h', 'capcolor'),
 (0, 'sporeprintcolor', 'k', 'stalksurfaceabovering'),
 (0, 'sporeprintcolor', 'n', 'stalksurfaceabovering'),
 (0, 'sporeprintcolor', 'r', 'ringnumber'),
 (0, 'sporeprintcolor', 'u', 'gillspacing'),
 (0, 'sporeprintcolor', 'w', 'gillcolor'),
 (0, 'gillsize', 'b', 'stalksurfaceabovering'),
 (0, 'gillsize', 'n', 'stalkcolorabovering'),
 (0, 'sporeprintcolor', 'h', 'capcolor'),
 (0, 'sporeprintcolor', 'k', 'stalksurfaceabovering'),
 (0, 'sporeprintcolor', 'n', 'stalksurfaceabovering'),
 (0, 'sporeprintcolor', 'r', 'ringnumber'),
 (0, 'sporeprintcolor', 'u', 'gillspacing'),
 (0, 'sporeprintcolor', 'w', 'gillcolor'),
 (0, 'gillsize', 'b', 'stalksurfaceabovering'),
 (0, 'gillsize', 'n', 'stalkcolorabovering'),
 (1, 'sporeprintcolor', 'h', 'capcolor'),
 (1, 'sporeprintcolor', 'k', 'stalksurfaceabovering'),
 (1, 'sporeprintcolor', 'n', 'stalksurfaceabovering'),
 (1, 'sporeprintcolor', 'r', 'ringnumber'),
 (1, 'sporeprintcolor', 'u', 'gillspacing'),
 (1, 'sporeprintcolor', 'w', 'gillcolor'),
 (1, 'sporeprintcolor', 'h', 'capcolor'),
 (1, 'sporeprintcolor', 'k', 'stalksurfaceabovering'),
 (1, 'sporeprintcolor', 'n', 'stalksurfaceabovering'),
 (1, 'sporeprintcolor', 'r', 'ringnumber')
```

.....

2-1)در مرحله بعد برای بررسی دقیق سایرین و فرایندی که درآینده استفاده خواهد شد ، به بررسی کتابخانه های موجود پرداختم که با جست و جویی ساده به دو کتابخانه Sklean.tree و Id3 رسیدم ، که هر کدام را توضیح و مزایا و معایب آن ها را از دیدگاه خودم بررسی میکنم

Sklearn.tree کتابخانه ای کامل که علاوه بر روش entropy میتوان از روش gini نیز استفاده نمود. توانایی pruning وجود ندارد. همچنین یکی از مشکلات اصلی بحث Passing categorical data to Sklearn Decision Tree است که نمیتوان به این تابع مقدار کمی مانند 'a' نسبت داد و باید همه اطلاعات عدد باشند. راه حل آن این است که از

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(["paris", "paris", "tokyo", "amsterdam"])
le.transform(["tokyo", "tokyo", "paris"])
```

استفاده کرد و اطلاعات را به عدد تبدیل نمود و سپس از طریق

```
list(le.inverse_transform([2, 2, 1]))
```

به حالت اولیه برگرداند.

یک نمونه در ضمیمه قرار گرفت استفاده در ضمیمه قرار گرفت.

2-2) id3 کتابخانه و کلاس بعدی است که میتوان با نام decision-tree-id3 آن را نصب نمود. که بعد از اجرا خروجی dot. میدهد ، همچنین میتوان خروجی dot. را با graphviz به فرمت png برده و نمایش داد.

از لینک زیر برای نصب graphviz می‌توان استفاده کرد :

https://bobswift.atlassian.net/wiki/spaces/GVIZ/pages/20971549/How+to+install+Graphviz+software

باتوجه به نوع اطلات موجود استفاده از id3 مناسب تر می باشد چرا که علاوه پذیرفتن و قبول کردن اطلاعات کتگورکال میتواند pruning را انجام دهد.

حال به بررسی اطلاعات و یافتن بهترین سطح هرس میپردازیم

```
feature_names = tr.columns.tolist()
feature_names.remove('poisonous')
#_____________________________________________

Xtr=tr.values.tolist()
for i in range(0,len(tr.values)):
    del Xtr[i][0]
ytr=list(tr['poisonous'])
#_____________
Xva=va.values.tolist()
for i in range(0,len(va.values)):
    del Xva[i][0]
yva=list(va['poisonous'])
#_____________________________________________

Xtr_train, Xtr_test, ytr_train, ytr_test = train_test_split(Xtr, ytr
                                                ,test_size=0.0,shuffle=False)

Xva_train, Xva_test, yva_train, yva_test = train_test_split(Xva, yva
                                                ,test_size=0.0,shuffle=False)
#_____________________________<< start learning >>_____________________________

estimator = Id3Estimator(max_depth=None, min_samples_split=1, prune=False,
            gain_ratio=False, min_entropy_decrease=0.0, is_repeating=False)

estimator.fit(Xtr_train, ytr_train, check_input=False)
export_graphviz(estimator.tree_, 'tree.dot', feature_names,extensive=True)

y_predtr=estimator.predict(Xtr_train)


y_predva=estimator.predict(Xva_train)


print('Accuracy Score on train data: ',
      accuracy_score(y_true = ytr_train  , y_pred=y_predtr )
      )
print('Accuracy Score on test data: ',
      accuracy_score(y_true=yva_train,y_pred=y_predva)
      )

```

بدون prune نتیجه به روی داده validation به شرح زیر است ،

```
Accuracy Score on train data:  1.0
Accuracy Score on test data:  0.8038984051978736
```

حال prune را فعال نموده ونتیجه :

```
Accuracy Score on train data:  0.9384138236597253
Accuracy Score on test data:  0.8411104548139398
```

حال purne را مجدد غیرفعال نموده و محدودیت را روی تعداد عمق درخت میبریم ، عمق درخت ما 11 است. پس به کم کردن آن میپردازیم تا جایی که میزان دقت اطلاعات validation حداکثر شود. اگر عمق درخت را 2 قرار دهیم به

```
Accuracy Score on train data:  0.898980948161276
Accuracy Score on test data:  0.8907265209686946
```

میرسیم که بیشترین میزان دقت در داده تست است.

حالت های مختلفی بررسی شد ، که بهرین نتیجه را بر روی داده validation از طریق کم کردن شاخه ها دریافت شد.

حال به رسم نتایج میپردازیم

فایل dot. را با کمک graphiz که پیشتر نصبش توضیح داده شد ، توسط command

dot file.dot -Tpng -o image.pngt در cmd به فایل png تبدیل میکنیم

درخت شماره 1

درخت شماره2

درخت شماره3

** تصاویر در ضمیمه قرار گرفت ، اینجا نمیشد بزرگ کرد.

حال اطلاعات train شده را برای بهترین حالت یادگیری (تغییر عمق شاخه بررسی می‌کنیم) که نتیجه زیر حاصل شد:

```
Accuracy Score on train data:  0.898980948161276
Accuracy Score on test data:  0.9037780401416765
```

روش سوم

با توجه به مرحله قبل دریافتیم که بهترین پاسخ را میتوان از تعیین عمق شاخه بدست آورد.

پس به پیاده سازی کامل بدون استفاده از کتابخانه id3 پرداخته شد . به صورتی که تابعی با ورودی های 1- دیتا فریم 2-لیست ویژگی ها 3- عمق نفوذ ، نوشته شد، برای محاسبه بهترین عمق نفوذ ، ابتدا نتیجه را روی داده validation بررسی کرده و درجایی که بهترین نتیجه را بگیریم ، روی داده تست اجرا میکنیم:

برای بررسی بهترین عمق نفوذ چند مثال را بررسی میکنیم:

مثال 1:

```
tree = id3(df_shroom, attribute_names,depth_of_tree=None)
```

که خروجی آن:

```
Accuracy of train is  1.0
Accuracy of validation is  0.7672770230360307
```

مثال 2:

```
tree = id3(df_shroom, attribute_names,depth_of_tree=10)
```

که نتایج زیر حاصل شد:

```
Accuracy of train is  0.9091714665485158
Accuracy of validation is  0.8470171293561725
```

مثال 3:

```
tree = id3(df_shroom, attribute_names,depth_of_tree=5)
```

که نتایج زیر حاصل شد:

```
Accuracy of train is  0.9020824102791316
Accuracy of validation is  0.8865918487891317
```

مثال 4:

```
tree = id3(df_shroom, attribute_names,depth_of_tree=3)
```

که نتایج زیر حاصل شد:

```
Accuracy of train is  0.898980948161276
Accuracy of validation is  0.8907265209686946
```

مثال 5:

```
tree = id3(df_shroom, attribute_names,depth_of_tree=2)
```

که نتایج زیر حاصل شد:

```
Accuracy of train is  0.898980948161276
Accuracy of validation is  0.8907265209686946
```

مثال 5:

```
tree = id3(df_shroom, attribute_names,depth_of_tree=1)
```

که نتایج زیر حاصل شد:

```
Accuracy of train is  0.8932210899424015
Accuracy of validation is  0.8753691671588896
```

همانگونه که مشاهده میشود ، برای درختی دو یا سه سطحی ، بهترین نتیجه را داریم ،در زیر درخت دو سطحی نمایش داده میشود :

```
{'odor': {'a': {'sporeprintcolor': {'k': 0, 'n': 0, 'u': 0}},
        'c': {'sporeprintcolor': {'k': 1, 'n': 1}},
        'f': {'sporeprintcolor': {'h': 1}},
        'l': {'sporeprintcolor': {'k': 0, 'n': 0, 'u': 0}},
        'm': {'sporeprintcolor': {'w': 1}},
        'n': {'sporeprintcolor': {'k': 0, 'n': 0, 'r': 1, 'w': 0}},
        'p': {'sporeprintcolor': {'k': 1, 'n': 1}}}}
```

حال به بررسی اطلاعات مربوط به دادگان تست میپردازیم :

که نتیجه زیر حاصل شد :

```
Accuracy of train is  0.898980948161276
Accuracy of validation is  0.8907265209686946
Accuracy of test is  0.9037780401416765
```

که درصد ملاحظه شده ای نسبت به درخت بدون هرس ، یا عمق نفوذ بهبود یافته

همچنین کد بخش آخر در ادامه آمده است :

```python
import pandas as pd
from collections import Counter
from math import log2
from pprint import pprint

path='F:/Data/noisy_train.csv'
df_shroom=pd.read_csv(path)

path1='F:/Data/noisy_valid.csv'
df_shroom_valid=pd.read_csv(path1)

path2='F:/Data/noisy_test.csv'
df_shroom_test=pd.read_csv(path2)


def entropy(prob):
 return sum( [-p*log2(p) for p in prob] )

def entropy_of_list(a_list):
    cnt = Counter(x for x in a_list)

    num_instances = len(a_list)*1.0
    #mohasebe ehtemal ha
    probs = [x / num_instances for x in cnt.values()]

    return entropy(probs)
def gain(feature_name):
    # dar avardan category haye featue
    df_categoty = df_shroom.groupby(feature_name)

    # mohasebe entropy ha dar dataframe roye target_attribute_name
    n = len(df_shroom.index) * 1.0
    df_agg_ent=df_categoty.agg({'poisonous':
            [entropy_of_list, lambda x: len(x)/n] })['poisonous']


    df_agg_ent.columns = ['entropy', 'prob']

    #jamh entropy ha va zarb dar ehtemaleshon
    return sum( df_agg_ent['entropy'] * df_agg_ent['prob'] )

#_________ta in ja be nahvi dar code aval neveshte shod ___________________
```

```
def id3(df , feature_names , depth_of_tree, last_lable=None):
    #baraye moshakhas kardan depth tree
    #aya dade ya na
    if(depth_of_tree!=None):
        dpt=depth_of_tree-1
    else:
        dpt=None
    #bara hame lable ha :
    cnt = Counter(x for x in df['poisonous'])

    #sharayet tvaghof tree
    ##1:agr hame label ha az yek model bashand
    if len(cnt) == 1:
        a=list(cnt.keys())
        return a[0]
    ##2:agr feature ha tamam shode bashand ya az hameye data estefade karde bashim
    elif df.empty or (not feature_names):
        return last_lable
     ###hala bia va derakh ra beshkan
    else:
        # label aksariyat in ja ra baraye laye badi dashte bash
        #voting
        a=list(cnt.values())
        index_of_max = a.index(max(cnt.values()))
        a=list(cnt.keys())
        last_lable = a[index_of_max]
        #mohasebe gain
        #bia gain ha ra hesab kon va kamtrin ra bardar
            #choon az entropy ghabl kam nakrdi kamtarin gain bishtarin data ra darad
        gainz = [gain(feature) for feature in feature_names]
        index_of_max_gain = gainz.index(min(gainz))

        best_feature = feature_names[index_of_max_gain]

     ##3:agr depth baraye tree bashad va be on reside bashad
        if(dpt!=None):
            if(dpt<0):
                return last_lable

        # ezafe kardan shakhe khali daroon dict ghabli
        tree = {best_feature:{}}
        remaining_feature_names=[i for i in feature_names if i != best_feature]


#seda kardan recursively in algorithm ta yeki az sharayet tavaghof rokh dahad
        for feature_val, data_subset in df.groupby(best_feature):

            subtree = id3(data_subset,remaining_feature_names,dpt)

            tree[best_feature][feature_val] = subtree
        return tree
```

```
attribute_names = list(df_shroom.columns)
attribute_names.remove('poisonous')

tree = id3(df_shroom, attribute_names,depth_of_tree=3)

pprint(tree)
```

```
#func komaki baraye conv tree be list
def tree_to_list(d_pandas, tree):

    a = list(tree.keys())
    feature = a[0]
    if d_pandas[feature] in tree[feature].keys():
        result = tree[feature][d_pandas[feature]]
        if isinstance(result, dict):
            # tosh ye dic dg hast : derakh bayad shekaste shavad
            return tree_to_list(d_pandas, result)
        else:
            return result # this is a label
    else:
        return None
```

```
#__test of train_____________________________________________________________

train_data  = df_shroom
#applay kardan func tree_to_list be tree va rikhtan ro  predicted_train
train_data['predicted_train'] = train_data.apply(tree_to_list,axis='columns',args=(tree,) )

print ( 'Accuracy of train is ' ,
       sum(train_data['poisonous']==train_data['predicted_train'])/ (len(train_data.index)*1.0)
       )
#__test of valid_____________________________________________________________

valid_data = df_shroom_valid
valid_data['predicted_valid'] = valid_data.apply(tree_to_list,axis='columns',args=(tree,) )

print ( 'Accuracy of validation is ' ,
       sum(valid_data['poisonous']==valid_data['predicted_valid'])/ (len(valid_data.index)*1.0)
       )

#__test of test_____________________________________________________________
test_data  = df_shroom_test
test_data['predicted_test'] = test_data.apply(tree_to_list,axis='columns',args=(tree,) )

print ( 'Accuracy of test is ' ,
       sum(test_data['poisonous']==test_data['predicted_test'])/ (len(test_data.index))
       )
#____________________________________________________________________________
```