```python
from google.colab import files
uploaded=files.upload()

#Calculate Entropy
def entropy(probs):
  import math
  return sum(-prob*math.log(prob,2) for prob in probs)
```

```python
import pandas as pd
df=pd.read_csv("lab2dataset.csv")
print(df)
```

```python
# Calculating the Probability of Positive and negative examples
def entropy_of_list(a_list):
  from collections import Counter
  cnt = Counter (x for x in a_list)
  num_instances =len(a_list)
  probs=[x/num_instances for x in cnt.values()]
  return entropy(probs)
```

```python
total_entropy= entropy_of_list(df['PlayTime'])
print(total_entropy)
```

```python
#Calculating information Gain Function
def information_gain(df,split_attribute_name, target_attribute_name, trace=0):
  #print ("Information gain calculation of", split_attribute_name)
  #print("target_attribute_name",target_attribute_name)
  df_split =df.groupby(split_attribute_name)
  for name,group in  df_split:
    #print("Name", name)
    #print("Group",group)
    nobs=len(df.index)*1.0
    #print(nobs)
    # Calculating Entropy of an attribute  and probability part of formula
    df_agg_ent=df_split.agg({target_attribute_name: [entropy_of_list,lambda x:
len(x)/nobs] })[target_attribute_name]
    #print("df_agg_ent", df_agg_ent)
    #print(df_agg_ent.columns)
    #calculate information gain
    avg_info=sum(df_agg_ent['entropy_of_list'] * df_agg_ent['<lambda_0>'])
    old_entropy=entropy_of_list(df[target_attribute_name])
    return old_entropy-avg_info
#print('Ingo-gain for day is'+str(information_gain(df,'Day', 'PlayTime')))

def id3DT(df, target_attribute_name, attribute_names, default_class=None):
  from collections import Counter
  cnt = Counter(x for x in df[target_attribute_name])
  if len(cnt)==1:
    return next(iter(cnt))
  elif df.empty or (not attribute_names):
    return default_class
  else:
    default_class =max(cnt.keys())
    #print("attributes_names:",attribute_names)
    gainz=[information_gain(df,attr, target_attribute_name) for attr in attribute_names]
    index_of_max=gainz.index(max(gainz))
    best_attr=attribute_names[index_of_max]
    tree={best_attr:{}}
    remaining_attributes_names=[i for i in attribute_names if i != best_attr]
    for attr_val, data_subset in df.groupby(best_attr):
      subtree=id3DT(data_subset,target_attribute_name,remaining_attributes_names,default_class)
      tree[best_attr][attr_val]=subtree
  return tree
attribute_names=list(df.columns)
attribute_names.remove('PlayTime')

from pprint import pprint
tree= id3DT(df,'PlayTime',attribute_names)
print("The Resultant Decision Tree is ")
pprint(tree)
attribute=next(iter(tree))
print("Best Attribute: \n", attribute)
print("Tree Keys\n ", tree[attribute].keys())
```

```
# Classifying new sample
def classify(instance, tree, default=None):
  attribute=next(iter(tree))
  print("Key:",tree.keys())
  print("Attribute",attribute)
  if instance[attribute] in tree[attribute].keys():
    result=tree[attribute][instance[attribute]]
    print("Instance Attribute:",instance[attribute], "TreeKeys:",tree[attribute].keys())
    if isinstance(result,dict):
      return classify(instance,result)
    else:
      return result
  else:
    return default

tree1={'Outlook':['Rainy','Sunny'],'Temperature':['Mild','Hot'],'Humidity':['Normal','High'],'
Windy':['Weak','Strong']}
df2=pd.DataFrame(tree1)
df2['Predicted']=df2.apply(classify,axis=1, args=(tree,'No'))
print(df2)
```