ID3

Algorithm

```
import numpy as np
import Pandas as pd
from graphviz import Digraph.

def Entropy (dataset):
    class_Count = dataset .iloc [:, -1].value_counts()
    Prob = class_Counts / len (dataset)
    return -np.Sum(Prob * np.log2(Prob))

def information_gain (dataset, feature):
    total_entropy = entropy (dataset)
    feature_values = dataset [feature].value_counts()
        weighted_Entropy = 0
    for value, Count in feature_value.Items():
            Subset = dataset [dataset [features] == value]
        weighted_Entropy += (Count / len (dataset))*
                    Entropy (Subset)
~~for~~
    return total_Entropy - weighted_entropy

def best_feature (dataset):
    features = dataset. Columns [:-1]
    best_feature = None
    for feature in features: ~~for~~
        info_gain = information_gain (dataset, feature)
        if info_gain > best_info_gain:
        best_feature = feature
    return best_feature
```

```
dy id3 (dotaset ,max_depth = None , depth = 0);
    if len (dotaset . iloc [:, -1 ]. unique ())==1:
        return dotaset .iloc [0, -1]

    if len (dataset . columns) == 1:
        return dotaset .iloc [:0, -1]. mode () [0]

    if max_depth is not None and depth >= max_depth
    return dotaset . iloc [:, -1]. mode ( ) [0]
    best = best_feature (dotaset)
        tree = {best : {}}

    for value in dataset [best] unique ();
        subset = dotaset [dataset [best]== value]
    tree [best ] [value ] = id3 (subset .drop (columns=
            [best ] , max_depth= max_depth,
        depth= depth+1 )

dy create_tree_diagram (tree, dot = None ,
    Parent_name = "Root", Parent_value = " ");
    if dot is None:
        dot = Digraph (format = "png" , engine= "dot")

    if isinstance (tree ; dict):

        for feature , branches in tree . items ():
          feature_name = f"{Parent_name}_{feature}"
        dot.node (feature_name ~~feature~~ ) ,
                                   feature
        dot. node (Parent_name, feature_name ,
          label = str (value))
    createe_tree_diagrame ( subtree ,dot, value_name,
                        str (value)]
```

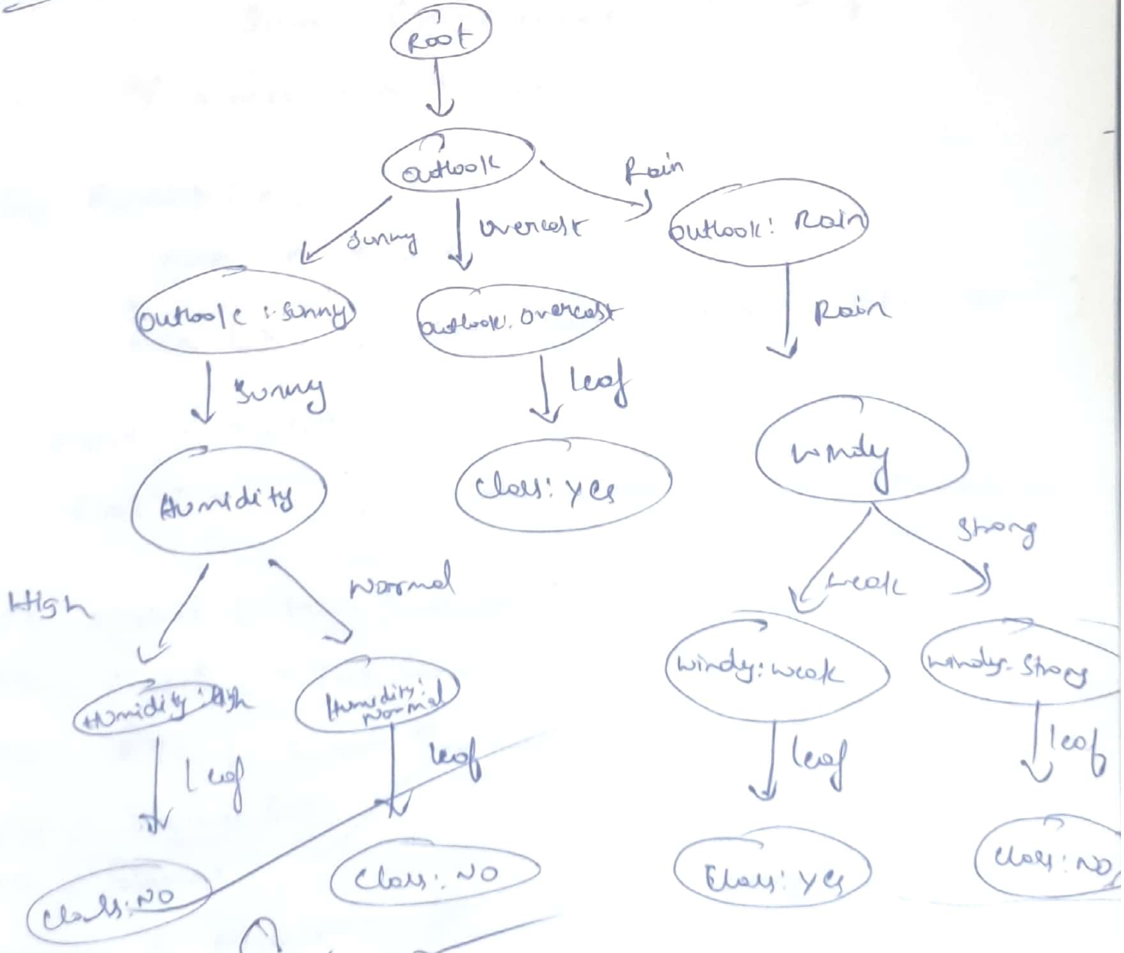
Ex:

```
dot.node (Parent_name + "_class", f"class:
              {tree}")

dot.edge (parent_name, parent_name +
      "_class", label = "leaf")
   return dot

data = pd.read_csv ("/content/weather_forecast.csv")
  df = pd.DataFrame (data)
  tree = id3 (df, max_depth = 3)
   dot = create_tree_diagrams (tree)
  dot.render ("decision_tree", view = True)
```

Output:



End to End
machine learning Project.

→ frame the Problem

→ Get the data

→ Discover and visualize the data to gain insight

→ Prepare the data for ML algorithms

→ Select a model and train it

→ fine-tune your model

→ Present your solution

→ launch, monitor, and maintain your System.