LAB-02

10/03/2025

PAGE NO : 5
DATE :

ID3 :- (Iterative Dichotomiser 3).

- used in machine learning to create decision trees.. that classifies data efficiently.
- it does so by repeatedly splitting the dataset based on features that provide highest information gain

Steps:

① Information Gain :

algorithm concern of entropy (measure of uncertainity) & information gain to decide which feature to split, on.

Information Gain = Entropy(Parent) - Σ(weighted Entropy of children).

② Spliting the Data

- based on selected features possible values.
- process continous recursily for each subset until certain stopping.

③ constructing Decision tree.

- Start with root node
- each branch represents a possible value of feature, and leaf nodes represent the class labels.

LAB-08)

17/08/2023

PAGE NO :
DATE :

ID3: Implementation

Code:-

```
import numpy as np
import pandas as pd
from collections import Counter

df = pd.read_csv("weather.csv")

def entropy(target_col):
    values, counts = np.unique(target_col, return_
                               counts = true)
    probs = counts / counts.sum()
    return -np.sum(probs * np.log2(probs))

def information_gain(df, feature, target = "Decision"):
    total_entropy = entropy(df[target])
    values, counts = np.unique(df[feature], return_
                               counts = True)
    weighted_entropy = sum(
        (counts[i] / sum(counts)) * entropy(df[df
            [feature] == values[i]][target])
        for i in range(len(values))
    )
    return total_entropy - weighted_entropy

def id3(df, features, target = "Decision"):
    unique_classes = np.unique(df[target])

    if len(unique_classes) == 1:
        return unique_classes = np.unique(df[target])

    if len(features) == 0:
        return Counter(df[target]).most_common(1)
```

```
gains = {feature : information_gain(df, feature,
        target) for feature in features}

best_feature = max(gains, key = gains.get)

tree = {best_feature : {}}
for value in np.unique(df[best_feature]):
    subset = df[df[best_feature] == value].
        drop(columns = [best_feature])
    tree[best_feature][value] = id3(subset,
        [f for f in features if f != best_feature],
                target)
return tree

features = list(df.columns[:-1])
decision_tree = id3(df, features)

import pprint
pprint.pprint(decision_tree)
```

17/3/25

output :

```
{'outlook': {'over cast': 'Yes',
             'Rain': {'wind': {'strong': 'No',
                               'weak': 'Yes'}},
             'Sunny': {'Humidity': {'High': 'No',
                                    'Normal': 'Yes'}}}}
```

tree :