**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

Program: B.Tech(CSBS)/ BTech(CSDS)-311

**Course: Machine Learning**

**Experiment No.05**

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical.)***

|  |  |
| --- | --- |
| Roll No.: C026 | Name: Anirbaan Ghatak |
| Class : B | Batch : EB1 |
| Date of Experiment: 29/08/2023 | Date of Submission: |
| Grade : |  |

**B.1 Task1**

# Name: Anirbaan Ghatak

# Roll no: C026

# Aim: :  Implementation of ID3(Decision Tree) Classifier. Also to find the performance metrics for the given dataset.

from sklearn.preprocessing import LabelEncoder

import pandas as pd

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

data = pd.read\_csv('weather\_data.csv', index\_col='Day')

# Convert the list of dictionaries to a DataFrame

headers = ['Outlook', 'Temperature', 'Humidity', 'Wind', 'Decision']

df = pd.DataFrame(data, columns=headers)

# Encode label categories to numbers, This is necessary because scikit-learn's decision tree implementation works with numerical data

label\_encoders = {}

for column in df.columns:

    le = LabelEncoder()

    df[column] = le.fit\_transform(df[column])

    label\_encoders[column] = le

# Separate features and target variable from the DataFrame

X = df.drop('Decision', axis=1)  # Features (exclude the 'Decision' column)

y = df['Decision']  # Target variable ('Decision')

# Initialize the DecisionTreeClassifier with criterion as 'entropy' to simulate ID3

clf = DecisionTreeClassifier(criterion='entropy', random\_state=42)

# Fit the model

clf.fit(X, y)

# Display the decision tree

tree.plot\_tree(

    clf, feature\_names=headers[:-1], class\_names=['No', 'Yes'], filled=True)

y\_pred = clf.predict(X)

accuracy = accuracy\_score(y, y\_pred)

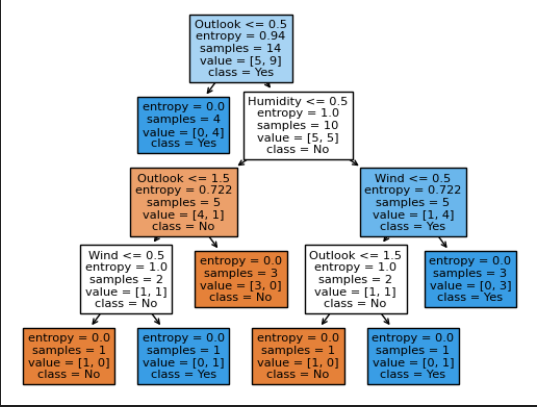
class\_report = classification\_report(y, y\_pred, target\_names=['No', 'Yes'])

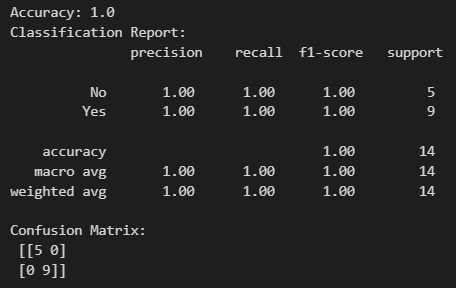
conf\_matrix = confusion\_matrix(y, y\_pred)

print('Accuracy:', accuracy)

print('Classification Report:\n', class\_report)

print('Confusion Matrix:\n', conf\_matrix)

****

****

**B.4 Conclusion:**

*ID3 is a fundamental decision tree learning method that excels with categorical attributes but struggles with numerical ones, leading to overfitting. However, these limitations can be addressed through pruning to simplify the tree and feature engineering to enhance its adaptability and performance.*

**B.5 Questions of Curiosity**

Q1.What are the issues in ID3 classification? How it can be resolved.

**Issues:**

**Overfitting**: ID3 can create overly complex trees that fit the noise in the data, affecting the model's ability to generalize.

**Binary Splits**: ID3 only performs binary splits, which may not be efficient for categorical variables with multiple levels.

**Lack of Pruning**: ID3 doesn't perform any pruning by default, contributing to the risk of overfitting.

**Greedy Nature**: It selects attributes based on information gain in a greedy manner, which might not lead to the optimal tree.

**Handling Missing Values**: ID3 doesn't have a native mechanism for handling missing values.

**Resolutions:**

**Pruning**: Reduce the complexity of the tree after its creation to combat overfitting.

**Multi-way Splits**: Modify the algorithm to handle multi-way splits efficiently.

**Non-greedy Algorithms** (e.g., CART): Consider alternatives that are not purely greedy in attribute selection.

**Imputation**: Handle missing values with techniques like mean imputation or more advanced methods before applying ID3.

**Discretization**: Convert numerical attributes into categorical ones through discretization to make them suitable for ID3.

Q2. What is overfitting? How can it be resolved?

Overfitting occurs when a machine learning model learns the training data too well, capturing noise or random fluctuations rather than the underlying patterns. It leads to poor generalization to new, unseen data.

Resolutions for overfitting include:

**Pruning**: Trim branches of decision trees or neural networks.

**Data Augmentation**: Increase the amount of training data.

**Simplifying the Model**: Use a simpler model with fewer parameters.

**Regularization**: Add penalties for complex models, like L1 or L2 regularization.

**Cross-Validation**: Evaluate the model's performance on validation data and use techniques like early stopping.

**Ensemble Methods**: Combine multiple models to reduce overfitting, e.g., Random Forests.

**Feature Selection**: Choose relevant features and remove irrelevant ones.

**Reduce Model Complexity**: Decrease the depth of decision trees or the number of layers in neural networks.

Q3.What is pruning? What is the condition for pruning? How it is done in C4.5 algorithm.

Pruning is a technique used in decision tree algorithms to reduce the size of the tree by removing branches that do not provide significant improvements in predictive accuracy. Pruning helps prevent overfitting.

Condition for Pruning: Pruning is typically performed when a subtree's estimated error rate or impurity measure (such as Gini index or entropy) on a validation dataset is not significantly worse than the unpruned subtree.

C4.5 Algorithm Pruning: In the C4.5 algorithm, pruning is done as follows:

Generate a complete decision tree using the training data.

**Evaluate each subtree**: For each subtree of the complete tree, calculate its estimated error rate or impurity on a validation dataset.

**Compare with unpruned tree**: Compare the error rate or impurity of each subtree with the corresponding metric for the unpruned tree.

**Prune if condition met**: If the subtree's error rate or impurity is not significantly worse than the unpruned tree (based on a statistical test like chi-squared), prune the subtree, replacing it with a leaf node that represents the majority class.

This process is recursively applied to different subtrees until no further pruning can be performed while maintaining or improving the model's accuracy on the validation data.