**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. C043 | Name: Om Kamath |
| Class : B | Batch : B2 |
| Date of Experiment: 29/9/23 | Date of Submission: 29/9/23 |
| Grade : |  |

**B.1 Task1**

# %%

# Name: Om Kamath

# Roll no: C043

# 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

data = pd.read\_csv('datasets/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')

df

# %%

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

# 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'])

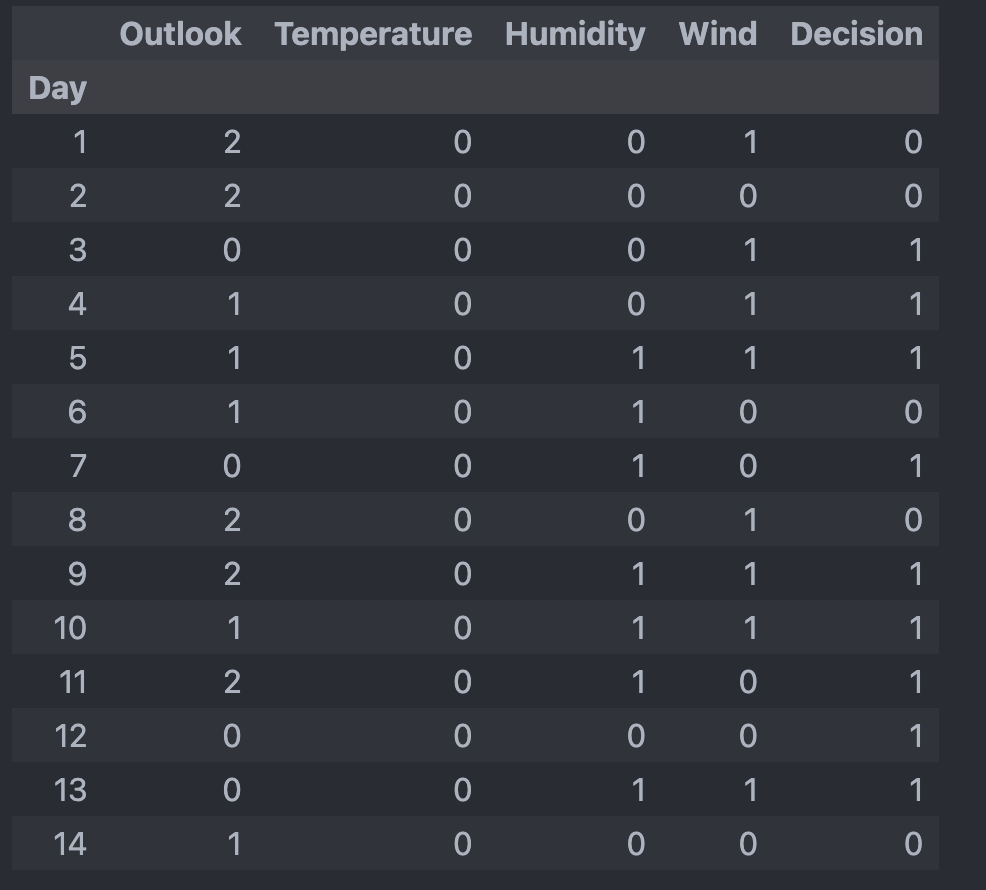
conf\_matrix = confusion\_matrix(y, y\_pred)

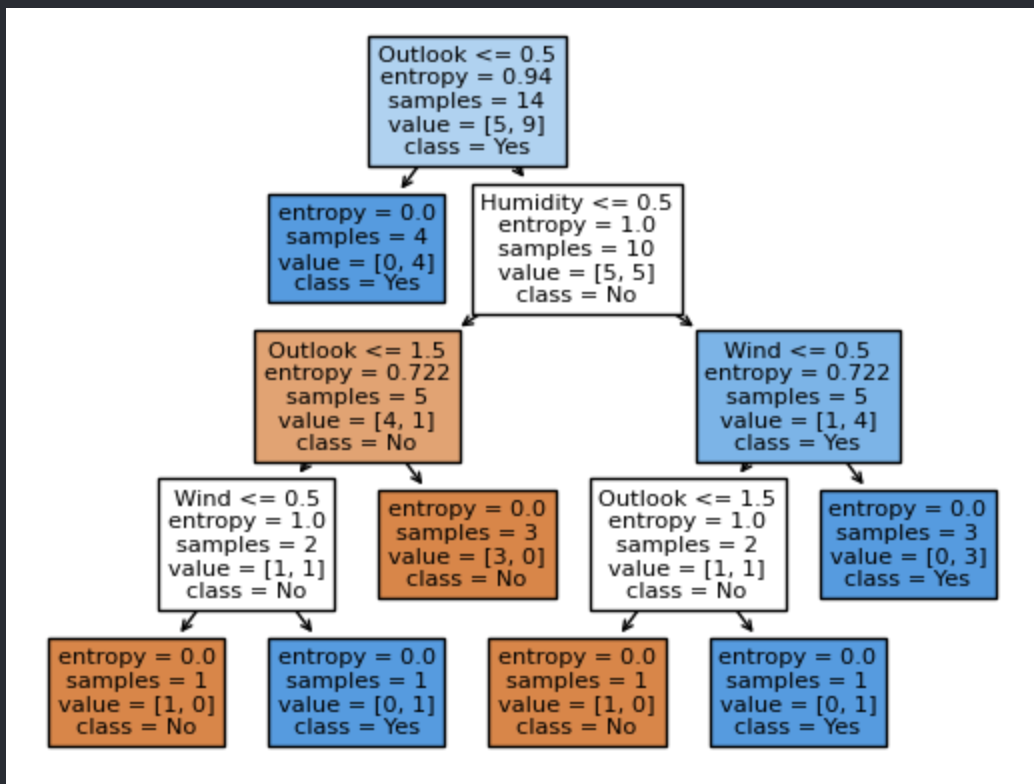
print('Accuracy:', accuracy)

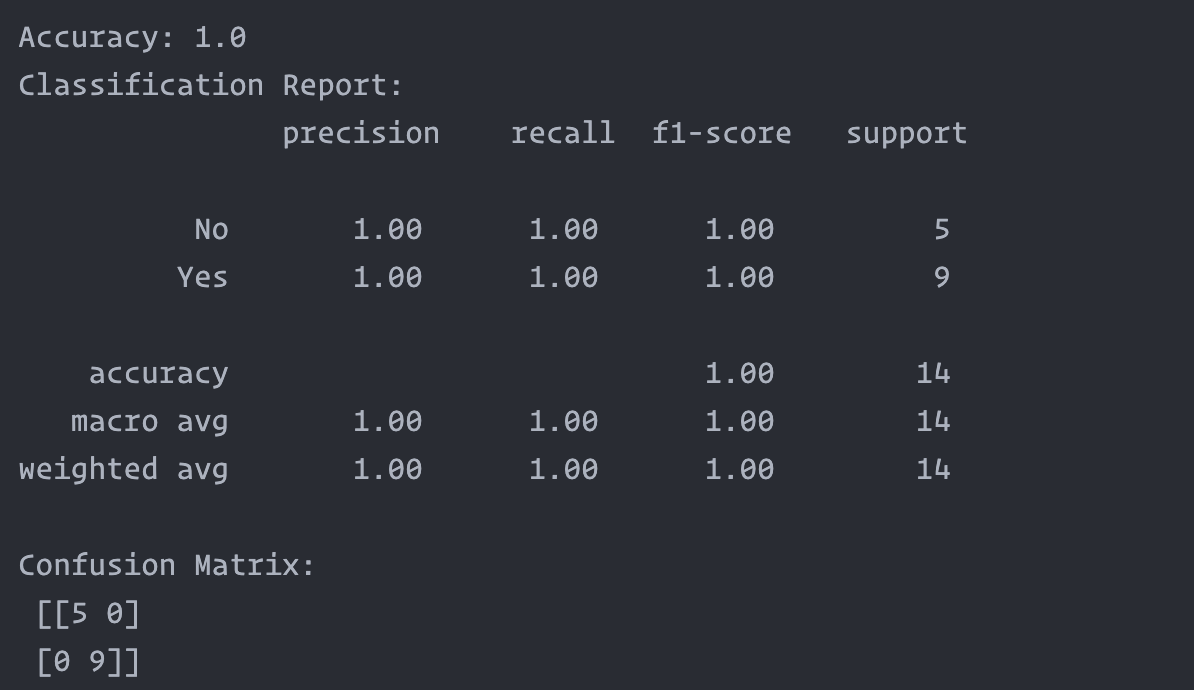
print('Classification Report:\n', class\_report)

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

# %%

****

****

****

**B.4 Conclusion:**

*(Students must write the conclusion in their own words.)*

The ID3 algorithm is a foundational decision tree learning method that has limitations like overfitting and poor handling of numerical attributes, but these can be mitigated through techniques like pruning and feature engineering.

**B.5 Questions of Curiosity**

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

Issues:

1. **Overfitting**: ID3 can create overly complex trees that fit the noise in the data, thereby affecting the model's ability to generalize.
2. **Binary Splits**: ID3 only performs binary splits, which may not be efficient for categorical variables with multiple levels.
3. **Lack of Pruning**: ID3 doesn't perform any pruning by default, contributing to the risk of overfitting.
4. **Greedy Nature**: It selects attributes based on information gain in a greedy manner, which might not lead to the optimal tree.
5. **Handling Missing Values**: ID3 doesn't have a native mechanism for handling missing values.
6. **Non-handling of Numeric Attributes**: The standard ID3 algorithm is not designed to handle numerical attributes effectively.

Resolutions:

1. **Pruning**: Reducing the complexity of the tree after its creation can combat overfitting.
2. **Multi-way Splits**: Modifying the algorithm to handle multi-way splits can make it more efficient.
3. **Random Forests**: Using an ensemble of decision trees (like Random Forest) can improve generalization.
4. **Non-greedy Algorithms**: Alternatives like CART can be used, which are not purely greedy.
5. **Imputation**: Missing values can be handled by various imputation techniques before feeding the data into ID3.
6. **Discretization**: Numerical attributes can be converted into categorical ones through discretization.

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

What is Overfitting?

Overfitting occurs when a model learns the detail and noise in the training data to the extent that it negatively impacts its performance on new, unseen data.

How to Resolve?

1. **Simplifying the Model**: Using fewer features or a simpler architecture can reduce overfitting.
2. **Regularization**: Introducing a penalty term can prevent the coefficients from fitting too perfectly.
3. **Cross-Validation**: Using cross-validation can give a more accurate estimate of a model's performance.
4. **Pruning**: In the case of decision trees, pruning techniques can be applied to simplify the final model.
5. **Use More Data**: Having more data can help the algorithm detect the signal better.
6. **Early Stopping**: In iterative algorithms like gradient boosting, stopping the training early can prevent overfitting.

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

What is Pruning?

Pruning involves removing the branches from a decision tree that provide little power in predicting target variables.

Conditions for Pruning

Pruning is generally done to:

1. Improve model's ability to generalize.
2. Reduce complexity and size of the tree.

How It's Done in C4.5

In C4.5, the pruning is done by replacing a subtree with a leaf node. The replaced leaf node is then assigned the most common classification of the training samples associated with that node. During the tree-building process, C4.5 uses a statistical test to check the confidence level of the subtree replacement. If the confidence level is acceptable, the subtree is pruned.

Here's a simplified explanation:

1. Start at the leaves and move towards the root of the decision tree.
2. For each node, calculate the statistical confidence of the error rate.
3. If the confidence is within a threshold, prune the node and replace it with a leaf node with the most frequent category.