# EXPERIMENT-24

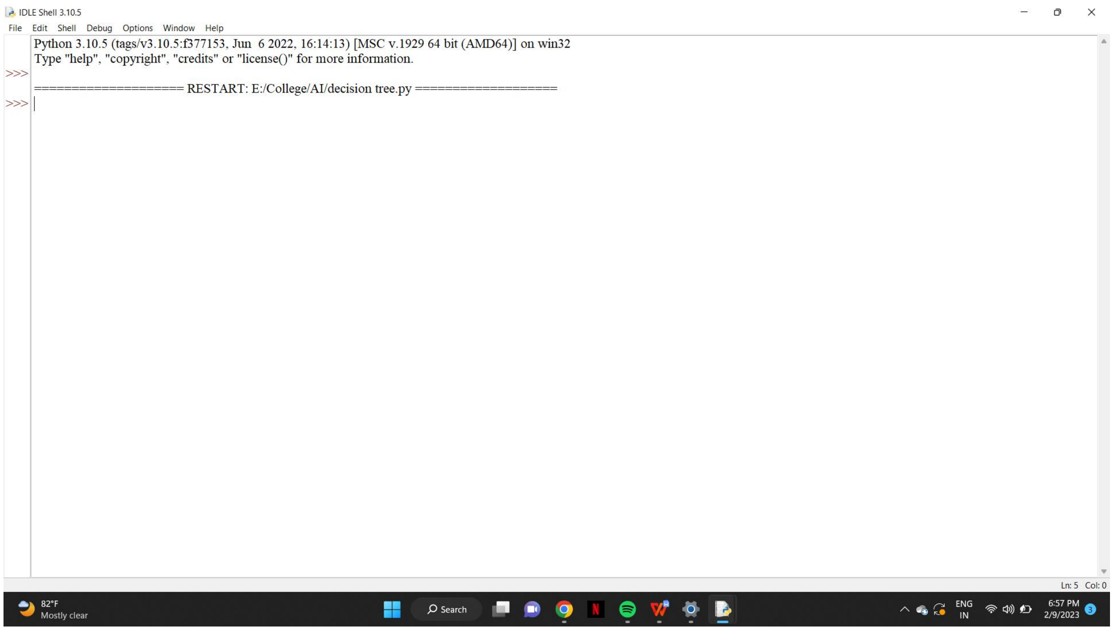
Name: S.G.DEVSACHIN
Reg.No: 192111088
Course: CSA1789 Artificial Intelligence

Q) Write the python program to implement Decision Tree

Program:

```
class DecisionTree:
    def __init__(self, feature, value):
        self.feature = feature
        self.value = value
        self.left = None
        self.right = None
        self.leaf = None

    def predict(self, x):
        if self.leaf is not None:
            return self.leaf
        if x[self.feature] <= self.value:
            return self.left.predict(x)
        else:
            return self.right.predict(x)

    def fit(self, X, y, depth=0, max_depth=10):
        if len(set(y)) == 1:
            self.leaf = y[0]
            return
        if depth == max_depth:
            self.leaf = max(set(y), key=y.count)
            return

        best_feature = None
        best_gain = 0
        n_features = X.shape[1]
        for feature in range(n_features):
            feature_values = X[:, feature]
            unique_values = set(feature_values)
            for value in unique_values:
                left_index = feature_values <= value
                right_index = feature_values > value
                X_left, y_left = X[left_index], y[left_index]
                X_right, y_right = X[right_index], y[right_index]
                gain = info_gain(y, y_left, y_right)
                if gain > best_gain:
                    best_gain = gain
                    best_feature = feature
                    best_value = value

        if best_gain > 0:
            self.feature = best_feature
            self.value = best_value
            left_index = X[:, best_feature] <= best_value
            right_index = X[:, best_feature] > best_value
            X_left, y_left = X[left_index], y[left_index]
            X_right, y_right = X[right_index], y[right_index]
            self.left = DecisionTree(None, None)
            self.right = DecisionTree(None, None)
            self.left.fit(X_left, y_left, depth=depth+1, max_depth=max_depth)
            self.right.fit(X_right, y_right, depth=depth+1,
max_depth=max_depth)

    def __repr__(self):
        return f'DecisionTree(feature={self.feature}, value={self.value},
leaf={self.leaf})'

def info_gain(parent, left, right):
    # Calculate the information gain of the split
    Pass
```

OUTPUT:



```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==================== RESTART: E:/College/AI/decision tree.py ===================
>>> 
```