Introduction:

It is an individual project implementing the Decision Tree Algorithm, coded by C#. Decision Tree Algorithm is helpful to take decision within a dataset by determining the entropy and information gain finding the highest in all attributes and then continue splitting until the dead end/pure set, so a decision can be taken through. My project is divided into 3 parts, where the first is the insertion of the dataset which includes columns and rows, while between part 1 and part 2 there is a huge communication between functions to calculate and assign the useful lists that will be used to draw the decision tree. Then, in part 2, a decision tree can be generated as a view, and part 3 is the manually classification or adding an extra row to the data and see the result.

Dive into the Implementation:

Starting with the dataset insertion, a textbox is used to create the columns/attributes and these columns is inserted in **pList** which is a List of Attributes class and this class has attributes: **name** and **iGained**. And when the user finishes, he click on the finish button and a function named changeTheme1() is called to change the theme to the next stage. Then, a manually coded textboxes will appear with respect the number of attributes he entered, so now the data can be inserted as well, these data is inserted also in a list called **rowsList** which is a list of list of Strings, where **rowsList**[0] means just a row, while **rowsList**[0][0] is a data. So once the user finished inserting the dataset, rowsList list is now ready and assigned, a function named changeTheme2() is called to change the theme and appear the dataset in front of the user, which is dynamically coded “Some labels and Rectangles”. Then, a function named **setUniqueRows**() is called to assign a list called **uniqueList**. This list do as its name says, set a unique rows for each column, for example: if a column

called Name and have data like Mahmoud, Ahmed, Ali, Kareem, Ali, Ali , Ahmed; it will only let these data to Mahmoud, Ahmed, Ali and Kareem; no duplicates. This list is a List of List of UniqueRows class, and this class has properties: **name**, **entropy** and **ct** "count". Then, a **main** 4 functions is called which are:

- **calculateAllEntropy**() -> Calculating the entropy for all the data
- **calculateEachEntropy**() -> Calculating Entropies for the unique rows to help the next function.
- **calculateIGain**() -> Calculating the Information Gain for each column
- **descIGain**() -> Which puts in an array the descending order of the columns so the tree can start with.

All these functions get helped and used **rowsList**, **uniqueList**, **pList**, an array called **descArray** and variable called **allEntropy.**

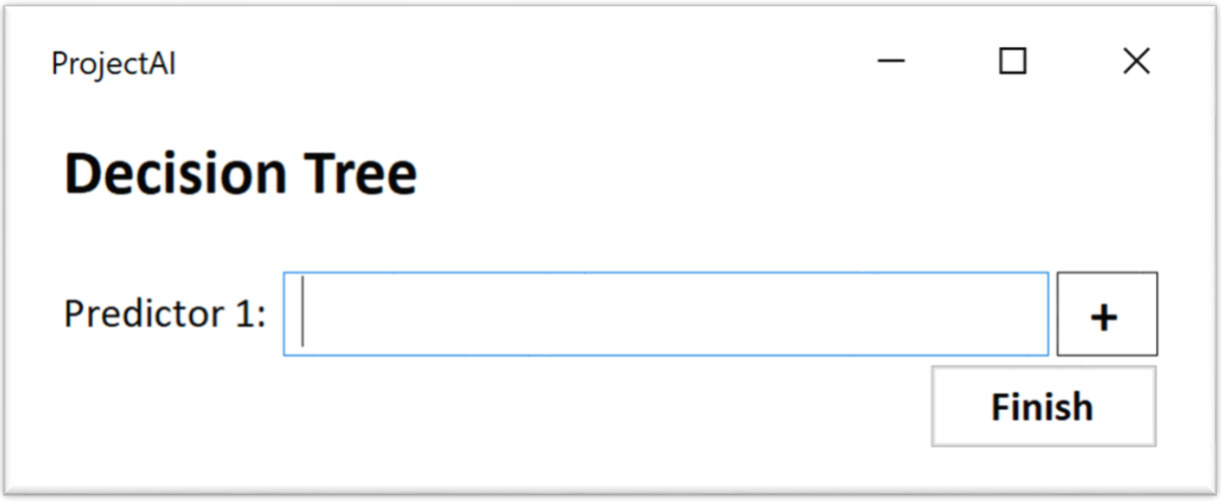
Then, an important function is called which named **createDecisionTree**(). This function's duty is creating an important list called **finalist**, this list is List of List of **AttributesTree** class, this class include properties which are: **name** and **bList** which is a List of **Branches** class, this class has properties which are: **name**, **end** "String" and **aList** which is List of **AttributesTree** class. So, this is the data structure I used to be able to draw the tree with respect the Decision Tree Rules, it is like a "data structure loop", and this happened because for every attribute there are branches and each branch can open another attribute if its not the dead end, so an attribute can have branches and these branches might have attributes and so on. This function will do its job to create **finalList,** it starts with the largest Information Gain and its saved in the array **descArray** as I said above, then it have 2 conditions, which are is it a pure set or not, if yes then **finalList** will add a new node with **name** and **end** "yes or no", if not then an infinite loop is joined and repeating the

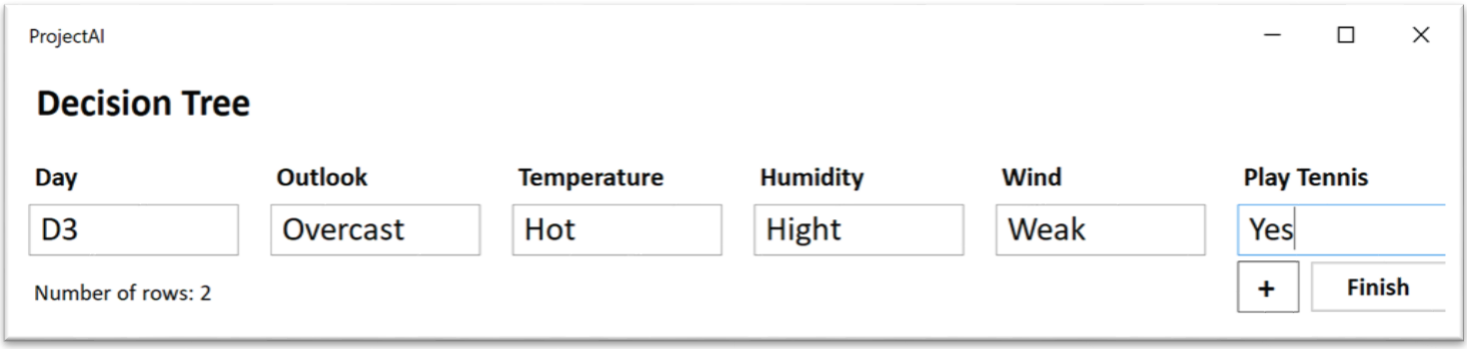
process until it finds a dead-end, and when it finishes, it will add in the finalList.
And for sure there are 2 important functions which are:

- **checkIfEnded**() -> To check for the dead-end “boolean”
- **createNewDataset**() -> To create new dataset with the specific value while looping “Creating new dataset is depending on the 4 main functions which are **calculateAllEntropy**(), **calculateEachEntropy**(), **calculateIGain**() and **descIGain**(), so they are called again but with respect to keep the original dataset safe”.

After the **finalList** is ready, so its passed to another form which is the form to view the drawing for the decision tree, a function named **createTheToolsForDrawing**() is called and has a parameter which is List of **Attributes** class. This function has the responsibility to assign the last list which **treeList** is List of **DrawTree** class, and this class has properties which are: **X**, **Y**, **X1**, **Y1**, **X2**, **Y2** -> integer and **name**, **type** -> String.
This function have 2 conditions, which are if **finalList[i].bList[j].aList.count** > 0 then this branch has another attribute and not a pure set, so in this case, a recursion occur, the function call itself, so it will start from the beginning which can create a new Parent and this parent has branches and so on until **finalList[i].bList[j].aList.count** is equal to **zero,** so in this case, it is a pure set and it has ended.
So, finally, **treeList** now is assigned and I can used it to draw by C# Graphics.
Also, a textbox is generated with a button to manually classify a new row and check the result and this also helped by the function **createTheToolsForDrawing()** in the case that if **finalList[i].bList[j].aList.count** is equal to **zero,** so I cant print the **finalList[i].bList[j].end** if **finalList[i].bList[j].name =** The input of the user.

Screenshots:





ProjectAI

Decision Tree

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Generate Tree

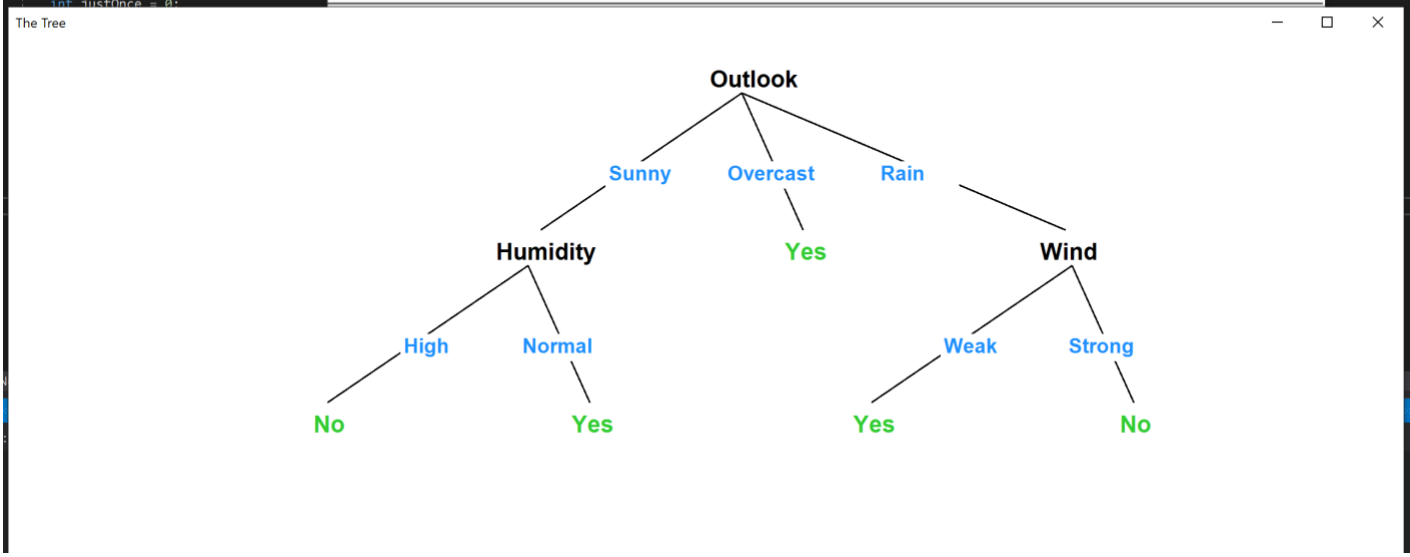
Classify

```
int[] descArray;
int[] tempDescArray;
TextBox newTextBox = new TextBox();
int justOnce = 0;
```

Decision Tree

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No

The Tree



D7	Overcast	Cool	Normal	Strong
D8	Sunny	Mild	High	Weak
D9	Sunny	Cool	Normal	Weak
D10	Rain	Mild	Normal	Weak
D11	Sunny	Mild	Normal	Strong
D12	Overcast	Mild	High	Strong
D13	Overcast	Hot	Normal	
D14	Rain	Mild	High	

Generate Tree

Overcast Mild Low

Classify

Yes

OK