# ASSIGNMENT 1

## Predicting Sinking Titanic Survivors using Decision Tree based Learning Model

**[ Project Code: TSDT ]**

**BY**
**GROUP NO. - 6**
**Khyathi Nalluri - 19EC37002**
**Priyanshi Dixit - 20CS10047**
**Vaishnavi Raghvendra Mughate - 18CS30045**

## Procedure :

Our design of Algorithm has used the standard ID3 Algorithm for designing the Decision Tree . We were given the titanic dataset and the aim is to predict the survival of passengers. The dataset is provided in .data format and data is shuffled . We consider the split of 80:20 as train,test set respectively for building decision tree. The data has the following attributes :

- PassengerId:
- Pclass: ticket class 1 = 1st, 2 = 2nd, 3 = 3rd
- Name: name of passenger
- Sex: male ,female
- Age: age of passenger
- SibSp: # of siblings / spouses aboard the Titanic
- Parch: # of parents / children aboard the Titanic
- Ticket: ticket number
- Cabin: cabin number
- Embarked: Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton
- Output Class – Survived: 0 = No, 1 = Yes

We have dropped the following attributes from the dataset :

- 'PassengerId'
- 'Name'
- 'Ticket'
- 'Fare'
- 'Cabin'
- 'Embarked'

**Major Functions :** This contains the procedure followed and the function used in brief

- ***get_df*** : Takes the dataframe as input and returns the training and test test data split at 80/20 ratio
- ***check_purity*** : Checks whether the given data is in pure form ( that is all data has same dependant value)
- ***classify_data*** : Identified the unique dependent classes in the given data
- ***get_potential_splits*** : Returns all the values that lie exactly in the middle between two dots as potential splits
- ***split_data*** : To determine where we have to split the given data using parameter along which feature will be used to split the data
- ***calculate_entropy*** : Function used to determine the each individual entropy of the split
- ***calculate_overall_entropy*** : Function used to calculate the overall entropy of the split
- ***determine_best_split_igain*** : Function used to find best potential split out of all those potential splits using information gain
- ***decision_tree_algorithm_igain*** : Uses the helper functions to build the Decision Tree. It is a recursive method, which chooses an attribute at each height to classify the current node into two children nodes. The attribute to be chosen is decided on the basis of the least entropy or highest information gain from the current node.
- ***classify_example*** : Classify one example of liver disorder using the built decision tree
- ***calculate_accuracy*** : Calculate accuracy to see how good the tree is classifying new liver disorder data

- ***predict_example*** : Function to predict target(dependent) value of one example of liver disorder using the built decision tree
- ***make_predictions*** : Function to predict the target (dependent) value of the whole input tree and given input liver disorder data .
- ***filter_df*** : Function used to split the data at a particular node

- ***determine_leaf*** : Function used to determine whether the given node is leaf node or another (parent node)

- ***determine_errors*** : Function used to determine errors by reduced-error pruning method

- ***pruning_result*** : Compares the training data and validation data and prunes the node

- ***post_pruning*** : After generating the Decision Tree using the Decision_Tree_Algorithm (igain). In this function we use a bottom up approach to prune the tree. Returns pruned tree after pruning using reduced-error pruning method

## Pruning

Steps followed:

- We constructed the entire Decision Tree with the height provided in construct_tree . We take a bottom-up approach.

- Pruning begins from the parent of leaf nodes. We consider a node. We temporarily remove the children of the node, say it New_Temporary_Tree . Evaluate the mean-squared error ( MSE ) of the New_Temporary_Tree on the validation set.

- If the MSE decreases on the validation set, then we remove the children of that node and update the attributes of the current node, thus updating the Tree.

- We repeat these steps until no more pruning is possible (overfitting has reduced significantly).

The above pruning approach is also termed as reduced-error pruning . To summarize, we have used the train set for building the tree, validation set for pruning, and test set for estimating the final accuracy.

# Results :

### For first tree :

Confusion Matrix:
[[112  12]
 [ 14  41]]
Accuracy:
85.47486033519553
Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.89 | 0.90 | 0.90 | 124 |
| 1 | 0.77 | 0.75 | 0.76 | 55 |
| accuracy | | | 0.85 | 179 |
| macro avg | 0.83 | 0.82 | 0.83 | 179 |
| weighted avg | 0.85 | 0.85 | 0.85 | 179 |

### For second tree :

Confusion Matrix:
[[94 16]
 [19 49]]
Accuracy:
80.33707865168539
Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.85 | 0.84 | 110 |
| 1 | 0.75 | 0.72 | 0.74 | 68 |
| accuracy | | | 0.80 | 178 |
| macro avg | 0.79 | 0.79 | 0.79 | 178 |
| weighted avg | 0.80 | 0.80 | 0.80 | 178 |

### For third tree :

Confusion Matrix:
[[89 16]
 [25 48]]
Accuracy:
76.96629213483146
Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 0.85 | 0.81 | 105 |
| 1 | 0.75 | 0.66 | 0.70 | 73 |
| accuracy | | | 0.77 | 178 |

| | | | | |
|---|---|---|---|---|
| macro avg | 0.77 | 0.75 | 0.76 | 178 |
| weighted avg | 0.77 | 0.77 | 0.77 | 178 |

### For fourth tree :

Confusion Matrix:
[[85 15]
 [23 55]]
Accuracy:
78.65168539325843
Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.85 | 0.82 | 100 |
| 1 | 0.79 | 0.71 | 0.74 | 78 |
| accuracy | | | 0.79 | 178 |
| macro avg | 0.79 | 0.78 | 0.78 | 178 |
| weighted avg | 0.79 | 0.79 | 0.78 | 178 |

### For fifth tree :

Confusion Matrix:
[[100  10]
 [ 20  48]]
Accuracy:
83.14606741573034
Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.91 | 0.87 | 110 |
| 1 | 0.83 | 0.71 | 0.76 | 68 |
| accuracy | | | 0.83 | 178 |
| macro avg | 0.83 | 0.81 | 0.82 | 178 |
| weighted avg | 0.83 | 0.83 | 0.83 | 178 |

**For the best tree(with high accuracy) after pruning :**

Confusion Matrix:

```
[[113  11]
 [ 12  43]]
```

Accuracy:

87.15083798882681

Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.91 | 0.91 | 124 |
| 1 | 0.80 | 0.78 | 0.79 | 55 |
| accuracy | | | 0.87 | 179 |
| macro avg | 0.85 | 0.85 | 0.85 | 179 |
| weighted avg | 0.87 | 0.87 | 0.87 | 179 |

**Accuracies of Trees corresponding to 5 data splits :**

- tree-1 : 0.855
- tree-2 : 0.803
- tree-3 : 0.77
- tree-4 : 0.787
- tree-5 : 0.831
- tree with best accuracy - 0.8547

best tree with best accuracy - 0.85 after pruning [pruned accuracy - 0.87]