Chongqing University of Technology

《MACHINE LEARNING》

DECISSION TREE

ALGORITHOMS

Student Name: Riffat (陈 琼 蕊)
Student ID: 62017010090
Class-6201703L1
Final ML

Table of Contents

1. Introduction to ML

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs. Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers. Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

2. Decision Tree Algorithm

For general use, decision trees are employed to visually represent decisions and show or inform decision making. When working with machine learning and data mining, decision trees are used as a predictive model. These models map observations about data to conclusions about the data’s target value. A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.

2.1 Introduction

The goal of decision tree learning is to create a model that will predict the value of a target based on input variables. In the predictive model, the data's attributes that are determined through observation are represented by the branches, while the conclusions about the data's target value are represented in the leaves. When "learning" a tree, the source data is divided into subsets based on an attribute value test, which is repeated on each of the derived subsets recursively. Once the subset at a node has the equivalent value as its target value has, the recursion process will be complete.

2.2 Important Terminology related to Decision Trees

- **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
- **Splitting:** It is a process of dividing a node into two or more sub-nodes.
- **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
- **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
- **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.

- **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
- **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.

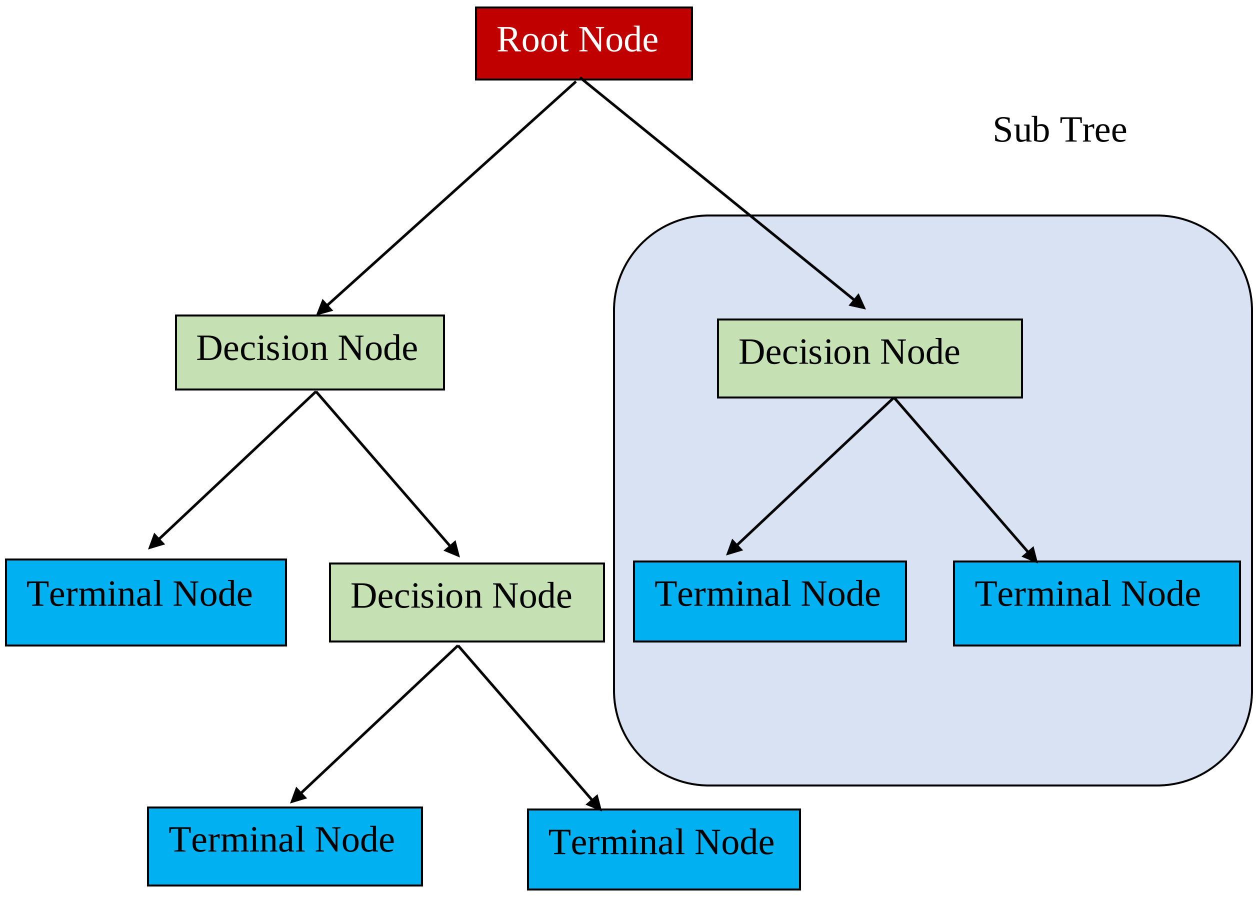


Figure 2.1: Sample of Decision tree

Decision trees classify the examples by sorting them down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the example. Each node in the tree acts as a test case for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new node.

2.3 Creating Decision Tree

Below are some of the assumptions we make while using Decision tree:

- ❖ In the beginning, the whole training set is considered as the root.
- ❖ Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- ❖ Records are distributed recursively on the basis of attribute values.
- ❖ Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

Decision Trees follow Sum of Product (SOP) representation. The Sum of product is also known as Disjunctive Normal Form. For a class, every branch from the root of the tree to a leaf node having the same class is conjunction (product) of values, different branches ending in that class form a disjunction. The primary challenge in the decision tree implementation is to identify which attributes do we need to consider as the root node and each level. Handling this is to know as the attributes selection. We have different attributes selection measures to identify the attribute which can be considered as the root note at each level.

2.4 Algorithms used in Decision Trees

- ❖ D3: Extension of D3
- ❖ C4.5: successor of ID3
- ❖ CART: Classification and Regression Tree

- ❖ CHAID: Chi-square automatic interaction detection Performs multi-level splits when computing classification trees.
- ❖ MARS: multivariate adaptive regression splines

The ID3 algorithm builds decision trees using a top-down greedy search approach through the space of possible branches with no backtracking. A greedy algorithm, as the name suggests, always makes the choice that seems to be the best at that moment.

- It begins with the original set S as the root node.
- On each iteration of the algorithm, it iterates through the very unused attribute of the set S and calculates Entropy(H) and Information gain(IG) of this attribute.
- It then selects the attribute which has the smallest Entropy or Largest Information gain.
- The set S is then split by the selected attribute to produce a subset of the data.
- The algorithm continues to recur on each subset, considering only attributes never selected before.

3. Implementation Decision Tree

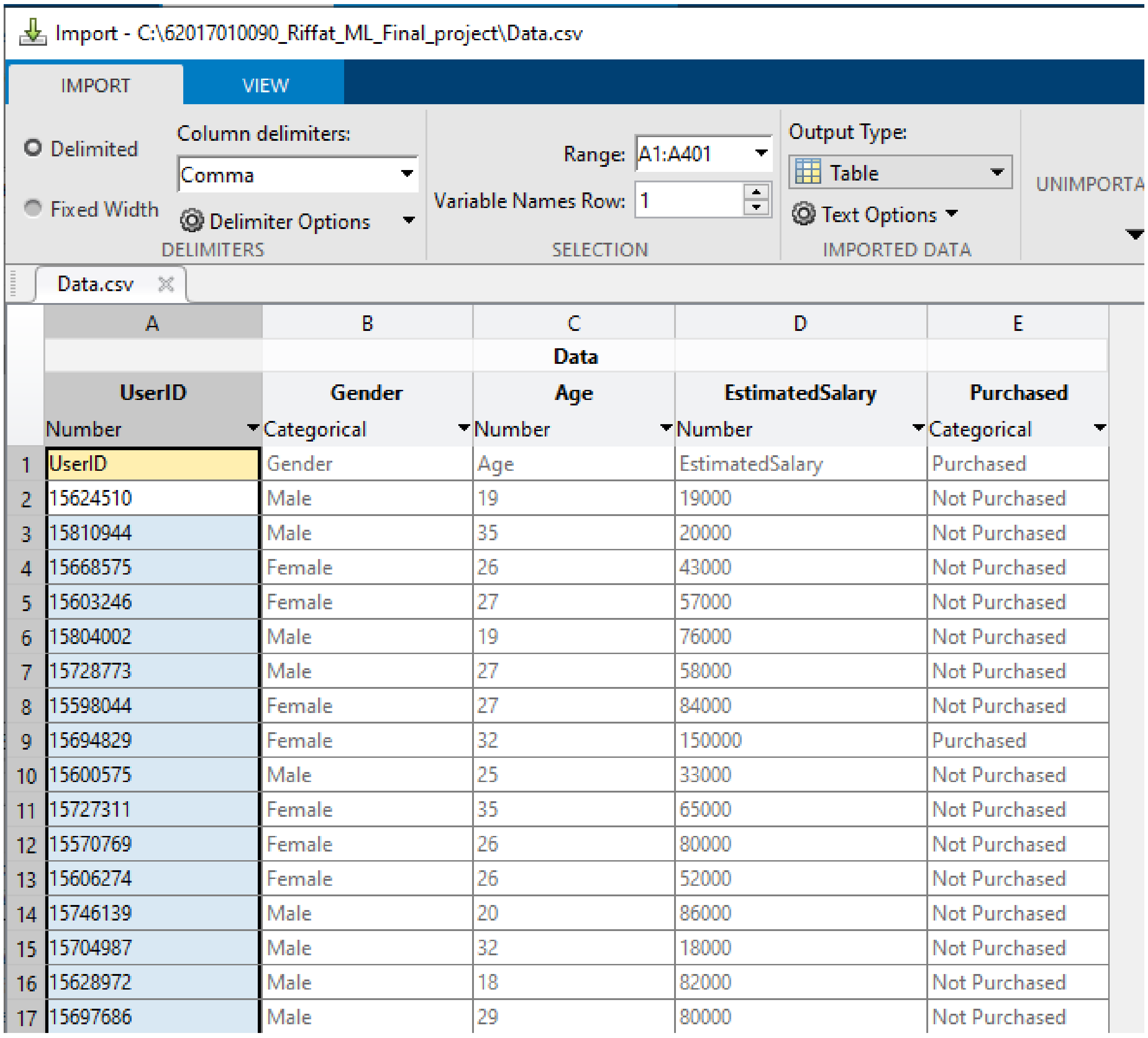
Now we will implement the Decision tree using MATLAB Programming Language. For this, we will use the dataset "**Data.csv**. By using the dataset, we can implement the Decision tree.

3.1 Importing Data Set

The code below is for Importing Data.

```
% Importing Dataset
```

```
data = readtable('Data.csv');

%-Feature Scaling (Standardization Method)
stand_age = (data.Age - mean(data.Age))/std(data.Age);
data.Age = stand_age;

stand_estimted_salary = (data.EstimatedSalary -
mean(data.EstimatedSalary))/std(data.EstimatedSalary);
data.EstimatedSalary = stand_estimted_salary;
```


Import - C:\62017010090_Riffat_ML_Final_project\Data.csv

Delimited; Column delimiters: Comma; Range: A1:A401; Variable Names Row: 1; Output Type: Table

	A	B	C	D	E
			Data		
	UserID	Gender	Age	EstimatedSalary	Purchased
	Number	Categorical	Number	Number	Categorical
1	UserID	Gender	Age	EstimatedSalary	Purchased
2	15624510	Male	19	19000	Not Purchased
3	15810944	Male	35	20000	Not Purchased
4	15668575	Female	26	43000	Not Purchased
5	15603246	Female	27	57000	Not Purchased
6	15804002	Male	19	76000	Not Purchased
7	15728773	Male	27	58000	Not Purchased
8	15598044	Female	27	84000	Not Purchased
9	15694829	Female	32	150000	Purchased
10	15600575	Male	25	33000	Not Purchased
11	15727311	Female	35	65000	Not Purchased
12	15570769	Female	26	80000	Not Purchased
13	15606274	Female	26	52000	Not Purchased
14	15746139	Male	20	86000	Not Purchased
15	15704987	Male	32	18000	Not Purchased
16	15628972	Male	18	82000	Not Purchased
17	15697686	Male	29	80000	Not Purchased

Figure 3.1: Importing Data

3.2 Classifying Data

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

```
% Classifying Data
classification_model = fitc-
tree(data,'Purchased~Age+EstimatedSalary');
% Max number of branch nodes / splits
classification_model = fitctree(data, 'Pur-
chased~Age+EstimatedSalary','MaxNumSplits',7);
% Min size of parent
classification_model = fitctree(data, 'Pur-
chased~Age+EstimatedSalary','MinParentSize',20);
classification_model = fitctree(data, 'Pur-
chased~Age+EstimatedSalary','SplitCriterion','gdi');
classification_model = fitctree(data, 'Pur-
chased~Age+EstimatedSalary','SplitCriterion','twoing');
classification_model = fitctree(data, 'Pur-
chased~Age+EstimatedSalary','SplitCriterion','deviance');
```

3.3 Partitioning

To find the optimal branching of a nominal attribute at a node in an L-side decision tree, one is often forced to search over all possible L-side partitions for the one that yields the minimum impurity measure. For binary trees ($L = 2$) when there are just two classes a short-cut search is possible that is linear in n, the number of distinct values of the attribute. For the general case in which the number of classes, k, may be greater than two, Burstein et al. have shown that the optimal partition satisfies a condition that involves the existence of 2 L hyperplanes in the class probability space. We derive a property of the optimal partition for concave impurity measures in terms of the existence of vectors in the dual of the class probability space, which implies the earlier condition. Unfortunately, these insights still do not offer a practical search method when n and k are large, even for binary trees. We therefore present a new heuristic search algorithm to find a good partition. It is based on ordering the attribute's values according to their principal component scores in the class probability space, and is linear in n.

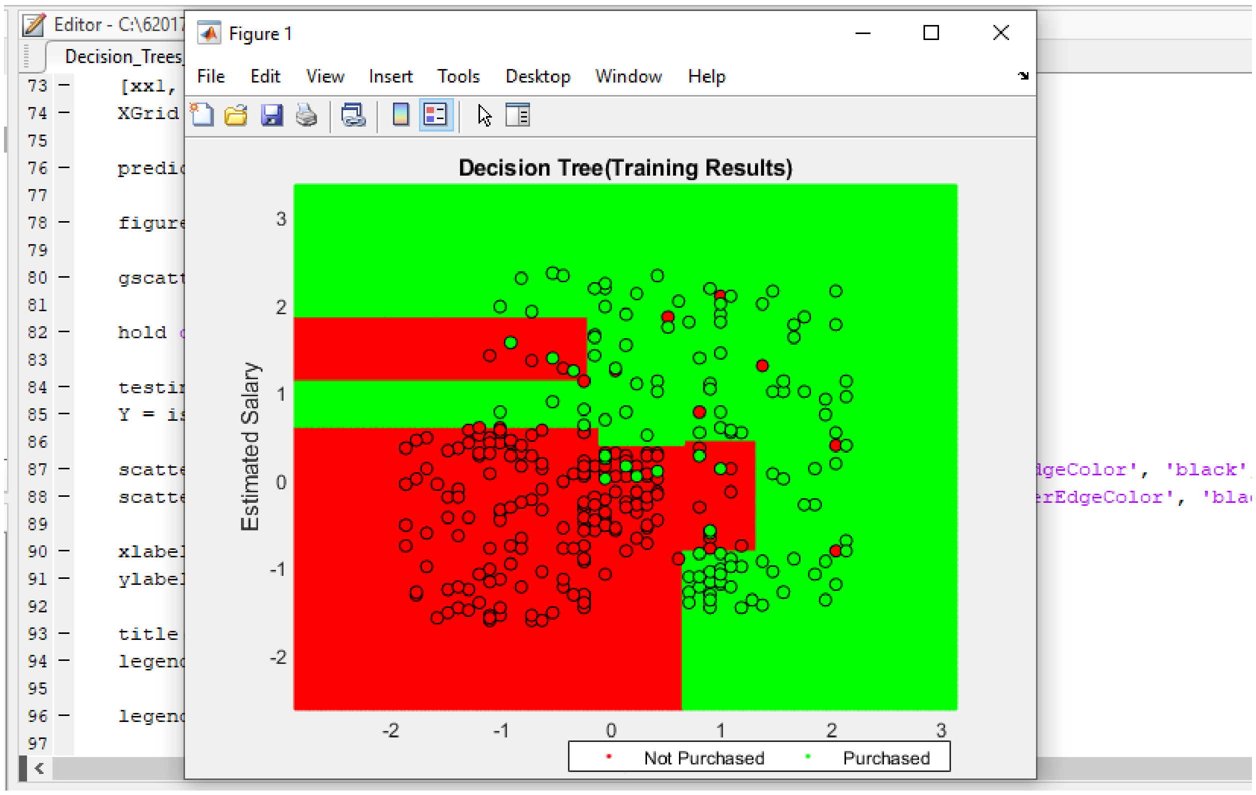
```
% Partitioning
cv = cvpartition(classification_model.NumObservations,
'HoldOut', 0.2);
cross_validated_model =
crossval(classification_model,'cvpartition',cv);

% Predictions
Predictions = pre-
dict(cross_validated_model.Trained{1},data(test(cv),1:end-
1));

% Analyzing the Results
Results = confusion-
mat(cross_validated_model.Y(test(cv)),Predictions);
```

3.4 Visualizing The Training Set Result

Here we will visualize the training set result. To visualize the training set result, I will plot a graph for the decision tree classifier. The classifier will predict yes or No for the users who have either Purchased or Not purchased the SUV car as we did in Logistic Regression. Below is the code for it.

```
% Creating View
%view(cross_validated_model.Trained{1}) %if statements for
tree in console
view(cross_validated_model.Trained{1}, 'Mode', 'Graph');
% Visualizing Test Results
```

```
labels = unique(data.Purchased);
classifier_name = 'Decision Tree (Testing Results)';

Age_range = min(data.Age(training(cv)))-
1:0.01:max(data.Age(training(cv)))+1;
Estimated_salary_range =
min(data.EstimatedSalary(training(cv)))1:0.01:max(data.Esti
matedSalary(training(cv)))+1;

[xx1, xx2] = meshgrid(Age_range,Estimated_salary_range);
XGrid = [xx1(:) xx2(:)];

predictions_meshgrid = pre-
dict(cross_validated_model.Trained{1},XGrid);
figure
gscatter(xx1(:), xx2(:), predictions_meshgrid,'rgb');
```



```
    [xx1,
    XGrid

    predi

    figure

    gscatt

    hold o

    testi
    Y = is

    scatte                                    dgeColor', 'black'
    scatte                                    erEdgeColor', 'bla

    xlabel
    ylabel

    title
    legend

    legend
```

Figure 3.4: Training Set Result

3.5 Visualizing The Test Set Result

Finally, the test dataset is a dataset used to provide an unbiased evaluation of a *final* model fit on the training dataset. If the data in the test dataset has never been used in training the test dataset is also called a holdout dataset. The term "validation set" is sometimes used instead of "test set" in some literature (if the original dataset was partitioned into only two subsets, the test set might be referred to as the validation set).Visualization of test set result will be similar to the visualization of the training set except that the training set will be replaced with the test set.

```
hold on
testing_data =  data(test(cv),:);
Y = ismember(testing_data.Purchased,labels{1});

scat-
ter(testing_data.Age(Y),testing_data.EstimatedSalary(Y),
'o' , 'MarkerEdgeColor', 'black', 'MarkerFaceColor',
'red');
scat-
ter(testing_data.Age(~Y),testing_data.EstimatedSalary(~Y) ,
'o' , 'MarkerEdgeColor', 'black', 'MarkerFaceColor',
'green');

xlabel('Age');
ylabel('Estimated Salary');

title(classifier_name);
legend off, axis tight

legend(labels,'Location',
[0.45,0.01,0.45,0.05],'Orientation','Horizontal');
```

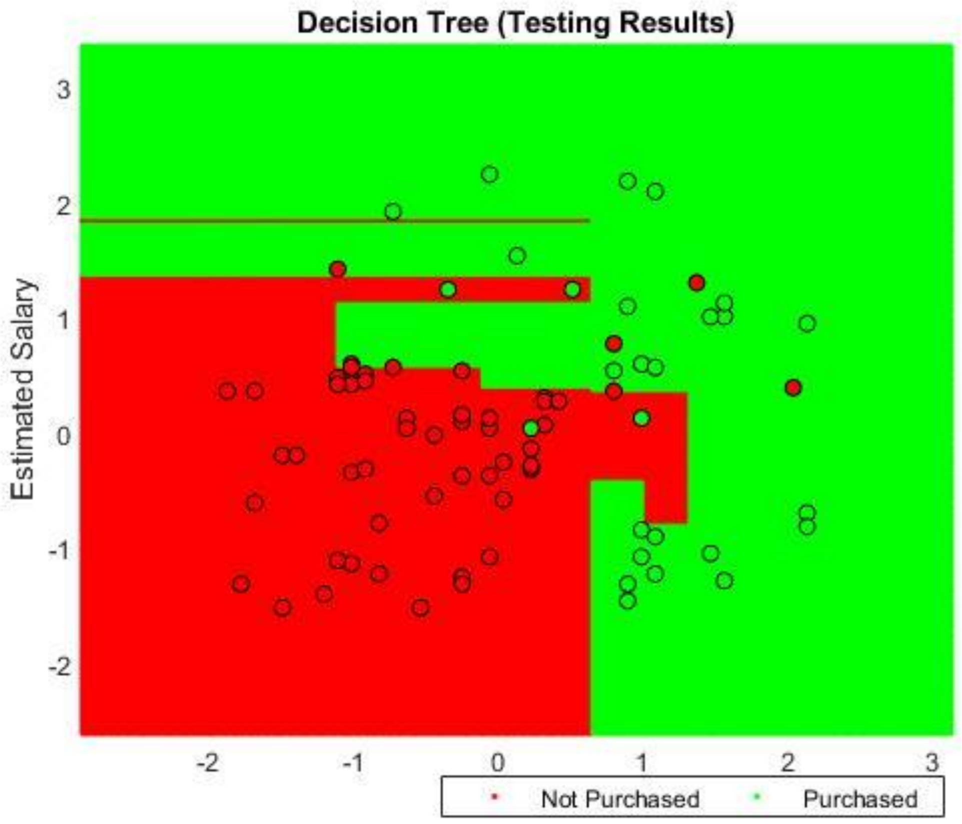


Figure 3.5: Test Data Set Visualizing

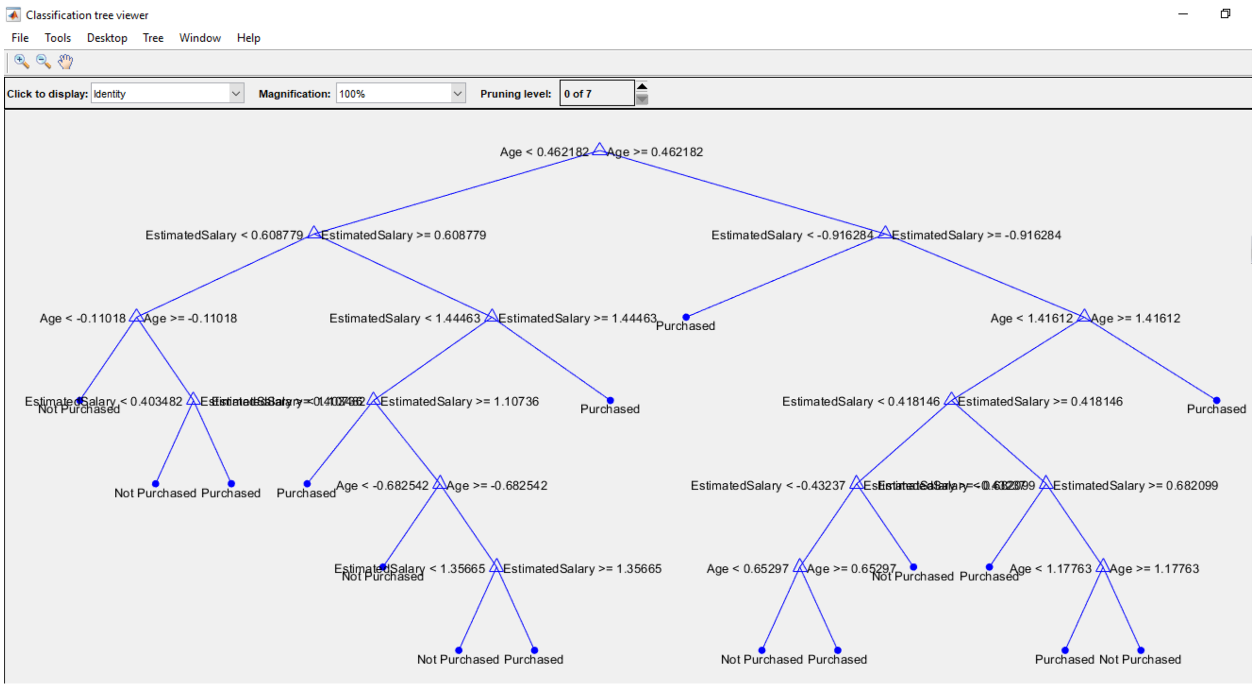
3.6 Final Result

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

```
% Decision Tree Algorithm
% Importing Dataset
data = readtable('Data.csv');

%-Feature Scaling (Standardization Method)
stand_age = (data.Age - mean(data.Age))/std(data.Age);
data.Age = stand_age;

stand_estimted_salary = (data.EstimatedSalary -
mean(data.EstimatedSalary))/std(data.EstimatedSalary);
data.EstimatedSalary = stand_estimted_salary;

% Classifying Data
classification_model = fitctree(data,'Purchased~Age+EstimatedSalary');
% Max number of branch nodes / splits
classification_model = fitctree(data, 'Pur-
chased~Age+EstimatedSalary','MaxNumSplits',7);
% Min size of parent
classification_model = fitctree(data, 'Pur-
chased~Age+EstimatedSalary','MinParentSize',20);
classification_model = fitctree(data, 'Pur-
chased~Age+EstimatedSalary','SplitCriterion','gdi');
classification_model = fitctree(data, 'Pur-
chased~Age+EstimatedSalary','SplitCriterion','twoing');
classification_model = fitctree(data, 'Pur-
chased~Age+EstimatedSalary','SplitCriterion','deviance');

% Partitioning
cv = cvpartition(classification_model.NumObservations, 'HoldOut', 0.2);
cross_validated_model = crossval(classification_model,'cvpartition',cv);

% Predictions
Predictions = predict(cross_validated_model.Trained{1},data(test(cv),1:end-1));

% Analyzing the Results
Results = confusionmat(cross_validated_model.Y(test(cv)),Predictions);

% Visualizing Training Results
labels = unique(data.Purchased);
classifier_name = 'Decision Tree(Training Results)';

Age_range = min(data.Age(training(cv)))-1:0.01:max(data.Age(training(cv)))+1;
Estimated_salary_range = min(data.EstimatedSalary(training(cv)))-
1:0.01:max(data.EstimatedSalary(training(cv)))+1;
```

```
[xx1, xx2] = meshgrid(Age_range,Estimated_salary_range);
XGrid = [xx1(:) xx2(:)];

predictions_meshgrid = predict(cross_validated_model.Trained{1},XGrid);

gscatter(xx1(:), xx2(:), predictions_meshgrid,'rgb');

hold on

training_data = data(training(cv),:);
Y = ismember(training_data.Purchased,labels{1});

scatter(training_data.Age(Y),training_data.EstimatedSalary(Y), 'o' , 'MarkerEdgeColor',
'black', 'MarkerFaceColor', 'red');
scatter(training_data.Age(~Y),training_data.EstimatedSalary(~Y) , 'o' , 'MarkerEdgeCol-
or', 'black', 'MarkerFaceColor', 'green');

xlabel('Age');
ylabel('Estimated Salary');

title(classifier_name);
legend off, axis tight

legend(labels,'Location',[0.45,0.01,0.45,0.05],'Orientation','Horizontal');

% Creating View
%view(cross_validated_model.Trained{1}) %if statements for tree in console
view(cross_validated_model.Trained{1}, 'Mode', 'Graph');
% Visualizing Test Results
labels = unique(data.Purchased);
classifier_name = 'Decision Tree (Testing Results)';

Age_range = min(data.Age(training(cv)))-1:0.01:max(data.Age(training(cv)))+1;
Estimated_salary_range = min(data.EstimatedSalary(training(cv)))-
1:0.01:max(data.EstimatedSalary(training(cv)))+1;

[xx1, xx2] = meshgrid(Age_range,Estimated_salary_range);
XGrid = [xx1(:) xx2(:)];

predictions_meshgrid = predict(cross_validated_model.Trained{1},XGrid);

figure
```

```
gscatter(xx1(:), xx2(:), predictions_meshgrid,'rgb');

hold on

testing_data =  data(test(cv),:);
Y = ismember(testing_data.Purchased,labels{1});

scatter(testing_data.Age(Y),testing_data.EstimatedSalary(Y), 'o' , 'MarkerEdgeColor',
'black', 'MarkerFaceColor', 'red');
scatter(testing_data.Age(~Y),testing_data.EstimatedSalary(~Y) , 'o' , 'MarkerEdgeColor',
'black', 'MarkerFaceColor', 'green');

xlabel('Age');
ylabel('Estimated Salary');

title(classifier_name);
legend off, axis tight

legend(labels,'Location',[0.45,0.01,0.45,0.05],'Orientation','Horizontal');
```


Classification tree viewer
File Tools Desktop Tree Window Help
Age < 0.462182 Age >= 0.462182
EstimatedSalary < 0.608779 EstimatedSalary >= 0.608779
EstimatedSalary < -0.916284 EstimatedSalary >= -0.916284
Age < -0.11018 Age >= -0.11018
EstimatedSalary < 1.44463 EstimatedSalary >= 1.44463
Purchased
Age < 1.41612 Age >= 1.41612
Not Purchased
EstimatedSalary < 0.403482 EstimatedSalary >= 0.403482
EstimatedSalary < 1.10736 EstimatedSalary >= 1.10736
Purchased
EstimatedSalary < 0.418146 EstimatedSalary >= 0.418146
Purchased
Not Purchased Purchased
Purchased
Age < -0.682542 Age >= -0.682542
EstimatedSalary < -0.43237 EstimatedSalary >= -0.43237
EstimatedSalary < 0.682099 EstimatedSalary >= 0.682099
Not Purchased
EstimatedSalary < 1.35665 EstimatedSalary >= 1.35665
Age < 0.65297 Age >= 0.65297
Not Purchased Purchased
Age < 1.17763 Age >= 1.17763
Not Purchased Purchased
Not Purchased Purchased
Purchased Not Purchased

Click to display: Identity
Magnification: 100%
Pruning level: 0 of 7

Figure 3.6: Final Result

4. Advantages of the Decision Tree

- ❖ It is simple to understand as it follows the same process which a human follows while making any decision in real-life.
- ❖ It can be very useful for solving decision-related problems.
- ❖ It helps to think about all the possible outcomes for a problem.
- ❖ There is less requirement of data cleaning compared to other algorithms.

5. Disadvantages of the Decision Tree

- ❖ The decision tree contains lots of layers, which makes it complex.
- ❖ It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
- ❖ For more class labels, the computational complexity of the decision tree may increase.

6. Summary

Decision trees are composed of three main parts decision nodes (denoting choice), chance nodes (denoting probability), and end nodes (denoting outcomes). Decision trees can be used to deal with complex datasets, and can be pruned if necessary to avoid overfitting. Despite having many benefits, decision trees are not suited to all types of data, e.g. continuous variables or imbalanced datasets. They are popular in data analytics and machine learning, with practical applications across sectors from health, to finance, and technology.