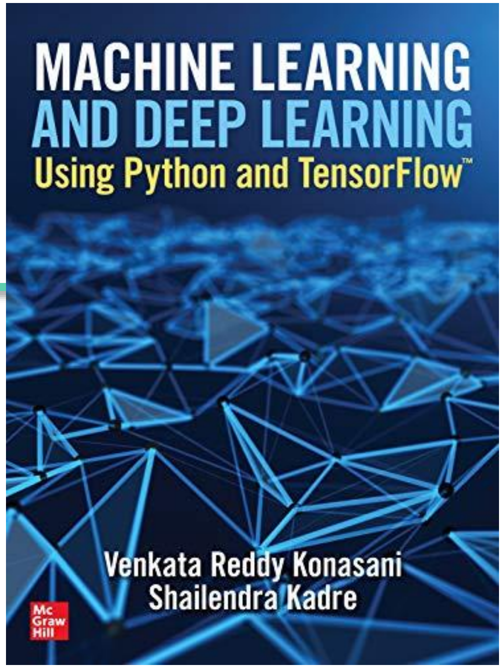
statinfer

Decision Trees

Venkat Reddy

Chapter 4 in the book



Introduction

Contents

* What is segmentation
* What is a Decision tree
* Decision Trees Algorithm
* Best Splitting attribute
* Building decision Trees
* Tree validation
* Pruning
* Prediction using the model

The Business Problem

Old Data

Gender	Marital Status	Ordered the product
M	Married	No
F	Unmarried	Yes
M	Married	No
M	Married	No
M	Married	No
M	Married	No
F	Unmarried	Yes
M	Unmarried	Yes
F	Married	No
M	Married	No
F	Married	No
M	Unmarried	No
F	Married	No
F	Unmarried	Yes

New Data

Gender	Marital Status	Product order
M	Married	??
F	Unmarried	??

The Business Problem

Old Data

Sr No	Gender	Marital Status	Ordered the product
1	M	Married	No
2	F	Unmarried	Yes
3	M	Married	No
4	M	Married	No
5	M	Married	No
6	M	Married	No
7	F	Unmarried	Yes
8	M	Unmarried	Yes
9	F	Married	No
10	M	Married	No
11	F	Married	No
12	M	Unmarried	No
13	F	Married	No
14	F	Unmarried	Yes

New Data

Gender	Marital Status	Product order
M	Married	??
F	Unmarried	??

The Decision Tree Philosophy

The Data

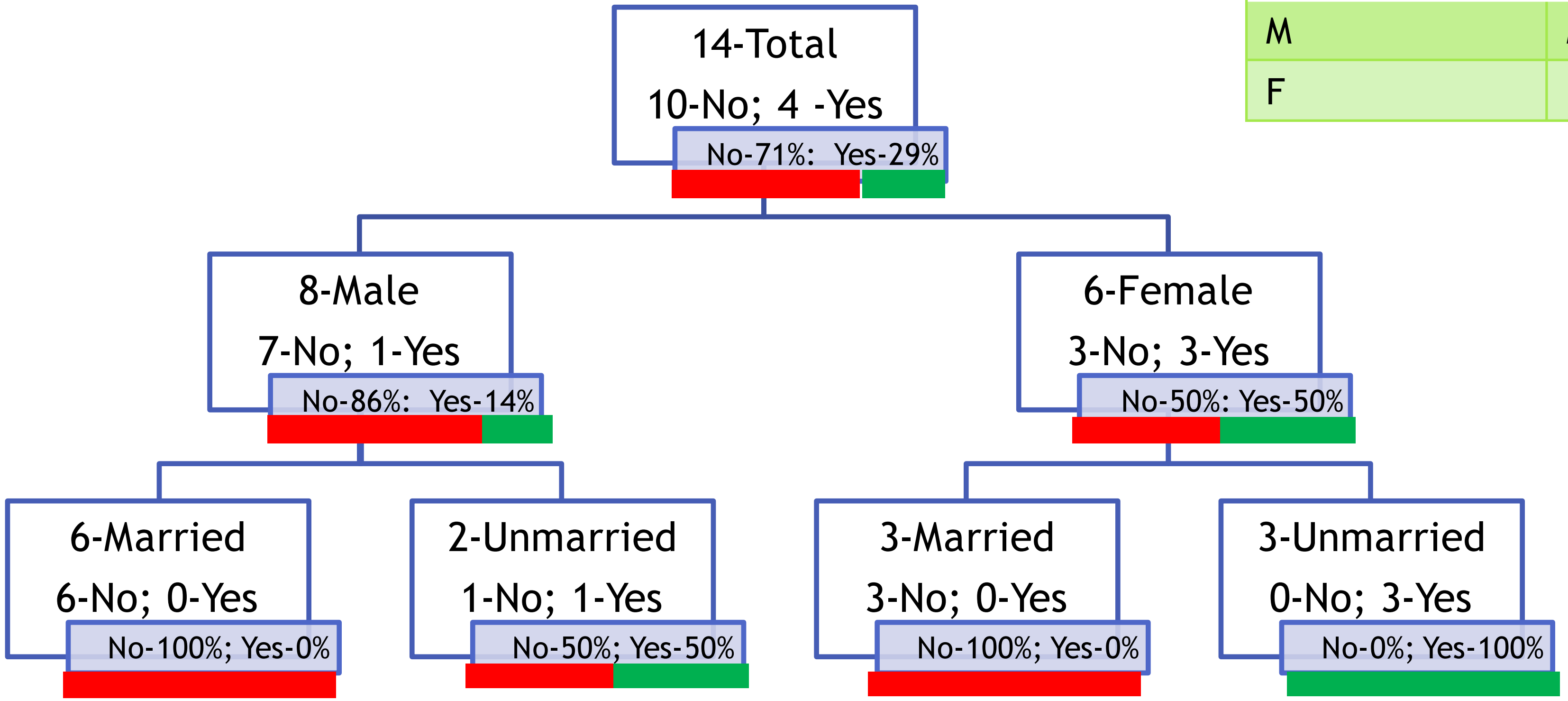
Old Data

Sr No	Gender	Marital Status	Ordered the product
1	M	Married	No
2	F	Unmarried	Yes
3	M	Married	No
4	M	Married	No
5	M	Married	No
6	M	Married	No
7	F	Unmarried	Yes
8	M	Unmarried	Yes
9	F	Married	No
10	M	Married	No
11	F	Married	No
12	M	Unmarried	No
13	F	Married	No
14	F	Unmarried	Yes

New Data

Gender	Marital Status	Product order
M	Married	??
F	Unmarried	??

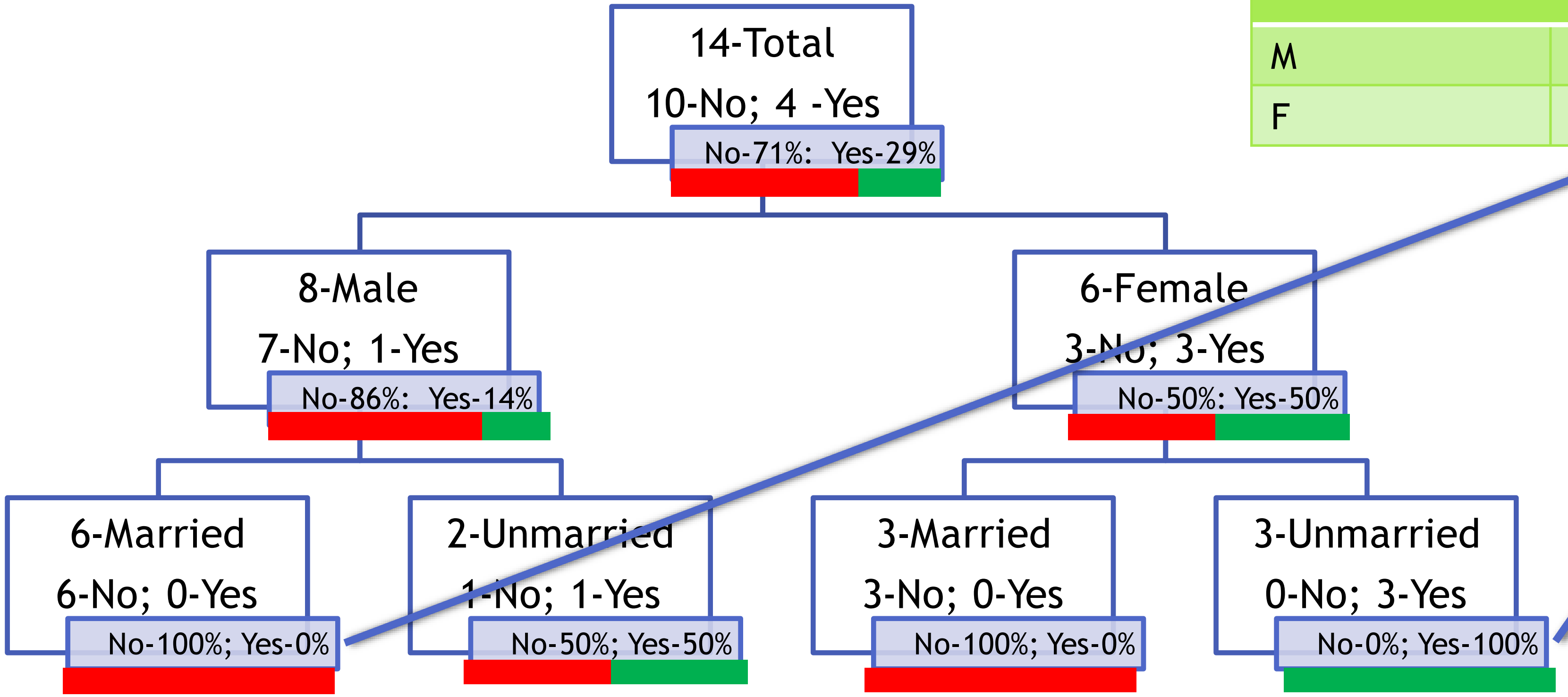
Re-Arranging the data



New Data

Gender	Marital Status	Product order
M	Married	??
F	Unmarried	??

Re-Arranging the data



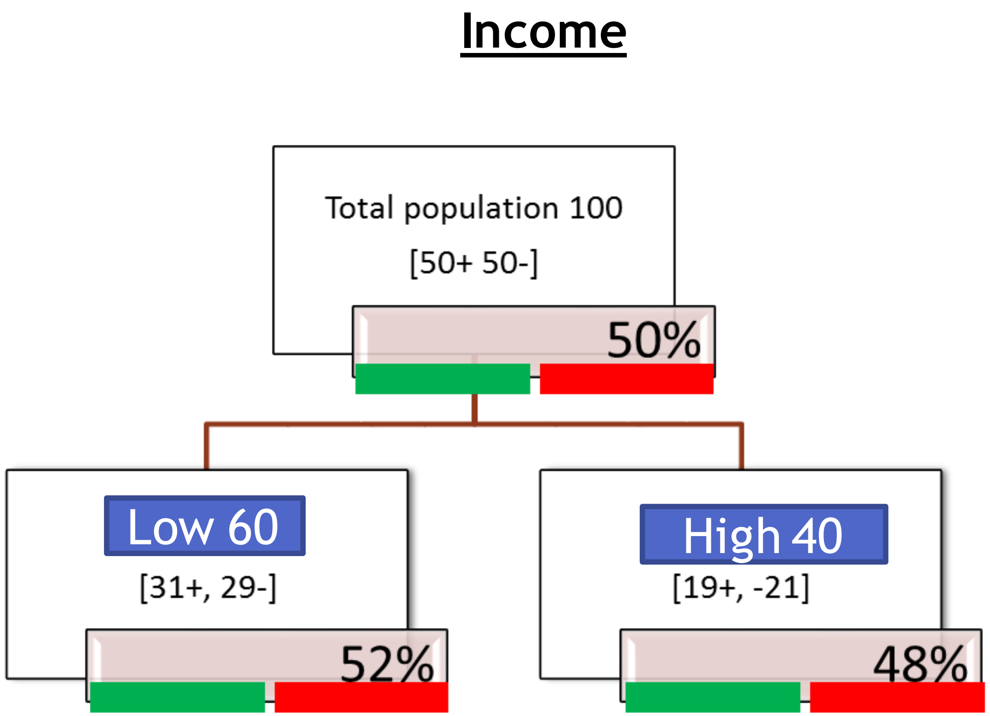
New Data		
Gender	Marital Status	Product order
M	Married	??
F	Unmarried	??

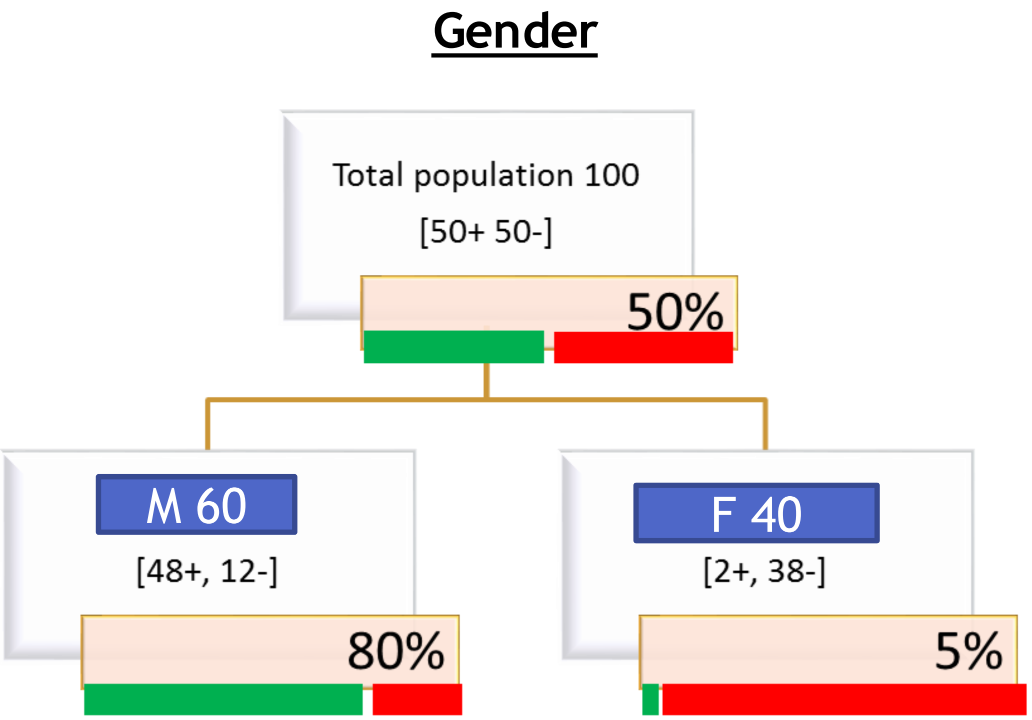
The Decision Tree Approach

The Decision Tree Approach

- The aim is to divide the whole population or the data set into segments
- The segmentation need to be useful for business decision making.
- If one class is really dominating in a segments
  - Then it will be easy for us to classify the unknown items
  - Then its very easy for applying business strategy
- For example:
  - It takes no great skill to say that the customers have 50% chance to buy and 50% chance to not buy.
  - A good splitting criterion segments the customers with 90% -10% buying probability, say Gender="Female" customers have 5% buying probability and 95% not buying

Example Sales Segmentations





The Splitting Criterion

The Splitting Criterion

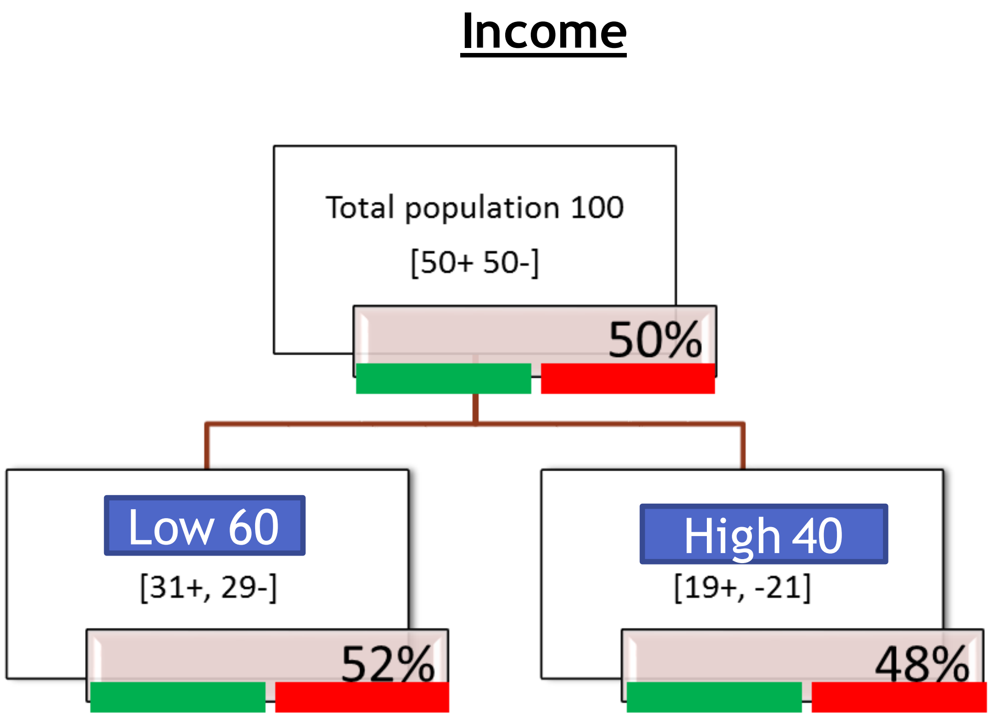
- The best split is
  - The split does the best job of separating the data into groups
    - Where a single class(either 0 or 1) predominates in each group

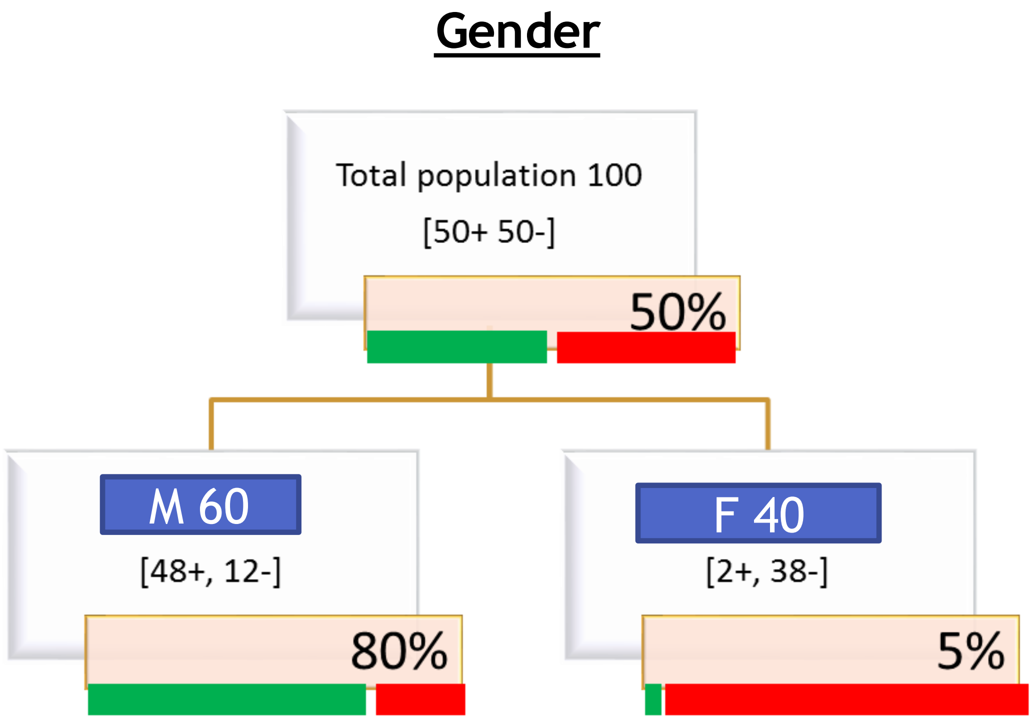
Main questions

- Ok we are looking for pure segments
- Dataset has many attributes
- Which is the right attribute for pure segmentation?
- Can we start with any attribute?
- Which attribute to start? – The best separating attribute
- Customer Age can impact the sales, gender can impact sales , customer place and demographics can impact the sales. How to identify the best attribute and the split?

Impurity (Diversity) Measures:

- We are looking for a impurity or diversity measure that will give high score for this Age variable(high impurity while segmenting), Low score for Gender variable(Low impurity while segmenting)





statinfer

Impurity (Diversity) Measures:

- **Entropy:** Characterizes the impurity/diversity of segment
- Measure of uncertainty/Impurity
- Entropy measures the information amount in a message
- S is a segment of training examples, p_+ is the proportion of positive examples, p_- is the proportion of negative examples
- Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
  - Where p_+ is the probabailty of positive class and p_- is the probabailty of negative class
- Entropy is highest when the split has p of 0.5.
- Entropy is least when the split is pure .ie p of 1

Entropy is highest when the split has p of 0.5

- Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
- Entropy is highest when the split has p of 0.5
- 50-50 class ratio in a segment is really impure, hence entropy is high
  - Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
  - Entropy(S) = $-0.5*\log_2(0.5) - 0.5*\log_2(0.5)$
  - Entropy(S) = 1

```
import math
entropy=-0.5*math.log2(0.5) -0.5*math.log2(0.5)
print(entropy)
```

Entropy is least when the split is pure .ie p of 1

- Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
  - Entropy is least when the split is pure .ie p of 1
  - 100-0 class ratio in a segment is really pure, hence entropy is low
    - Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
    - Entropy(S) = $-1*\log_2(1) - 0*\log_2(0)$
    - Entropy(S) = 0

```
import math
-0.0001*math.log2(0.0001) -0.9999*math.log2(0.9999)
```

The less the entropy, the better the split

- The less the entropy, the better the split
- Entropy is formulated in such a way that, its value will be high for impure segments

LAB: Entropy

- Calculate Entropy for 50%-50%
- Calculate Entropy for 0%-100%
- Calculate entropy for 45%-55%
- Calculate entropy for 5%-95%

Code: Entropy

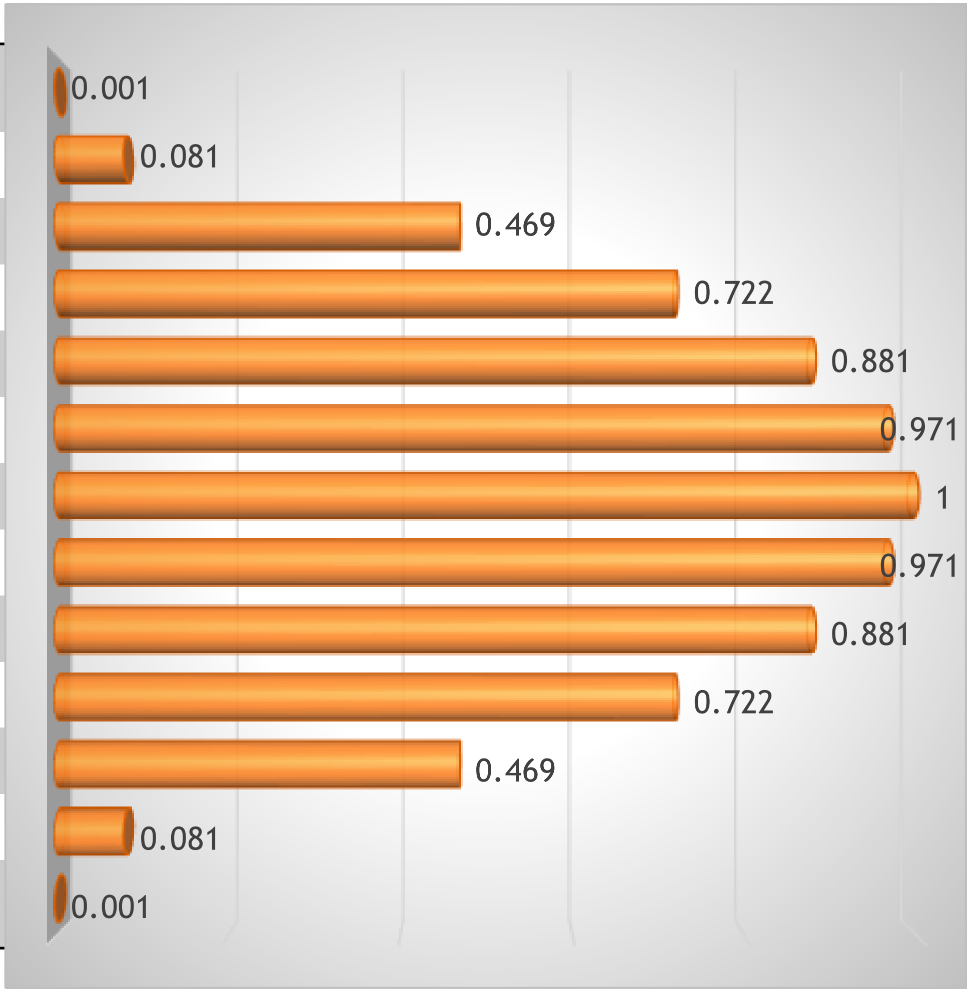
```
import math
entropy=-0.45*math.log2(0.45) -0.55*math.log2(0.55)
print("Entropy for 45%-55% case ==>", entropy)

entropy=-0.05*math.log2(0.05) -0.95*math.log2(0.95)
print("Entropy for 5%-95% case ==>", entropy)
```

```
Entropy for 45%-55% case ==> 0.9927744539878083
Entropy for 5%-95% case ==> 0.2863969571159625
```

statinfer

Code: Entropy

Segment	P_1	P_2	Entropy
S1	0.0001	0.9999	0.001
S2	0.01	0.99	0.081
S3	0.1	0.9	0.469
S4	0.2	0.8	0.722
S5	0.3	0.7	0.881
S6	0.4	0.6	0.971
S7	0.5	0.5	1.000
S8	0.6	0.4	0.971
S9	0.7	0.3	0.881
S10	0.8	0.2	0.722
S11	0.9	0.1	0.469
S12	0.99	0.01	0.081
S13	0.9999	0.0001	0.001



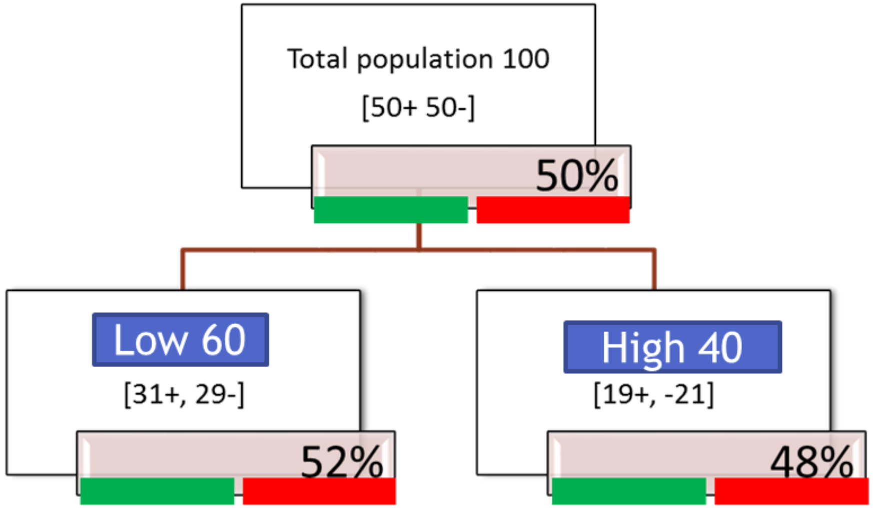
Information Gain

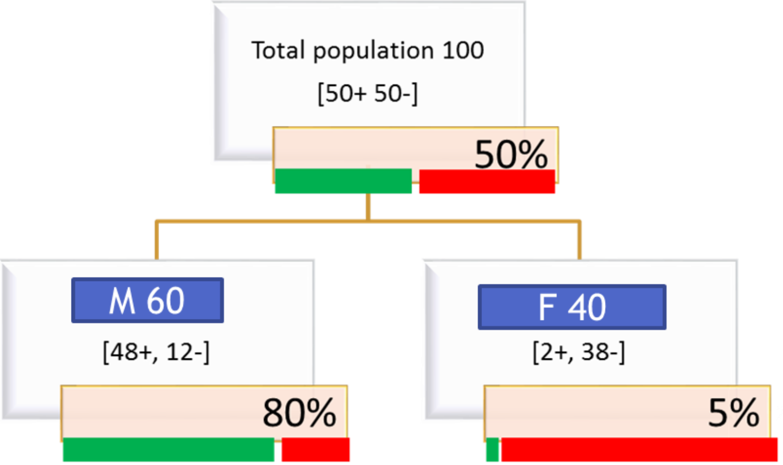
Information Gain

- Information Gain= entropyBeforeSplit – entropyAfterSplit
- Easy way to understand Information gain= (overall entropy at parent node) – (sum of weighted entropy at each child node)
- Attribute with maximum information is best split attribute

Information Gain- Calculation

Information Gain= entropy Before Split – entropy After Split





$-31/60 \log_2 31/60 - 29/60 \log_2 29/60$ $\quad -19/40 \log_2 19/40 - 21/40 \log_2 21/40$

- Entropy Ovearll = 100% (Impurity)
- Entropy Low income Segment = 99%
- Entropy High Sgment = 99%
- Information Gain for Income =100-(0.6*99+0.4*99)=**1**

- Entropy Ovearll = 100% (Impurity)
- Entropy Male Segment = 72%
- Entropy Female Sgment = 29%
- Information Gain for Gender =100-(0.6*72+0.4*29)=**45.2**

The Decision tree Algorithm

The Decision tree Algorithm

- The major step is to identify the best split variables and best split criteria
- Once we have the split then we have to go to segment level and drill down further

The Decision tree Algorithm

Until stopped:

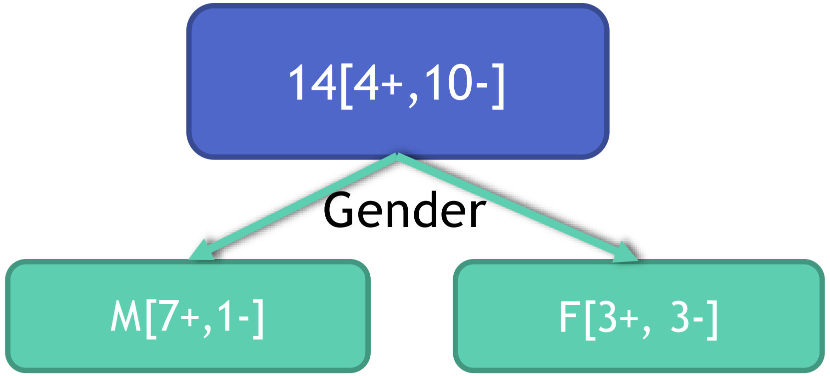
1. Select a leaf node
2. Find the best splitting attribute
3. Spilt the node using the attribute
4. Go to each child node and repeat step 2 & 3

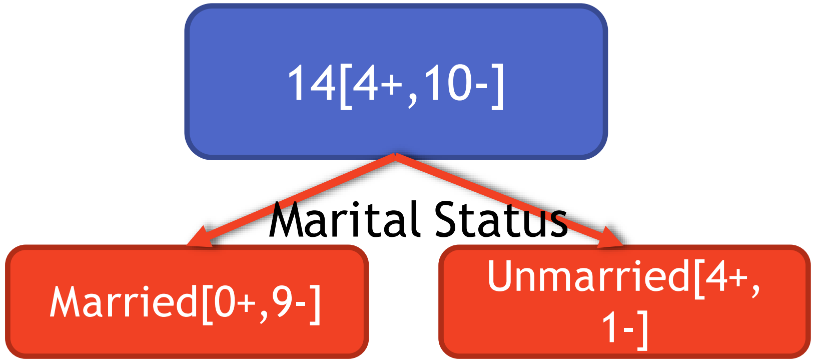
Stopping criteria:

- Each leaf-node contains examples of one type
- Algorithm ran out of attributes
- No further significant information gain

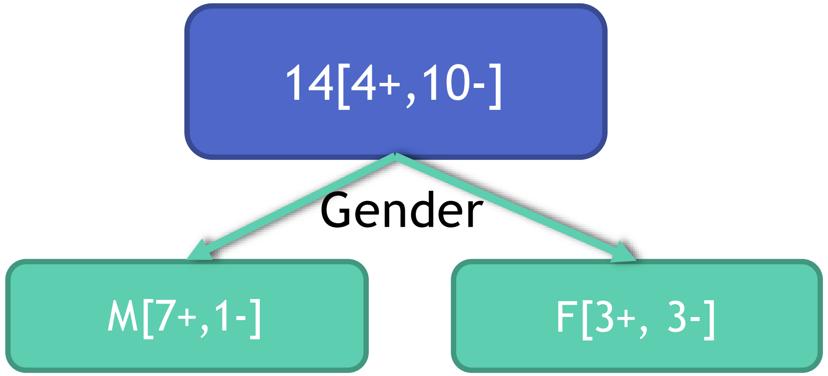
The Decision tree Algorithm- Demo

Sr No	Gender	Marital Status	Ordered the product
1	M	Married	No
2	F	Unmarried	Yes
3	M	Married	No
4	M	Married	No
5	M	Married	No
6	M	Married	No
7	F	Unmarried	Yes
8	M	Unmarried	Yes
9	F	Married	No
10	M	Married	No
11	F	Married	No
12	M	Unmarried	No
13	F	Married	No
14	F	Unmarried	Yes



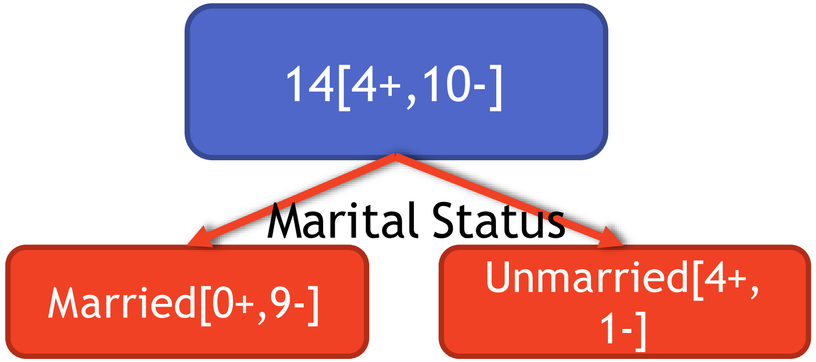


The Decision tree Algorithm- Demo



- Entropy([4+,10-]) Ovearll = 86.3% (Impurity)

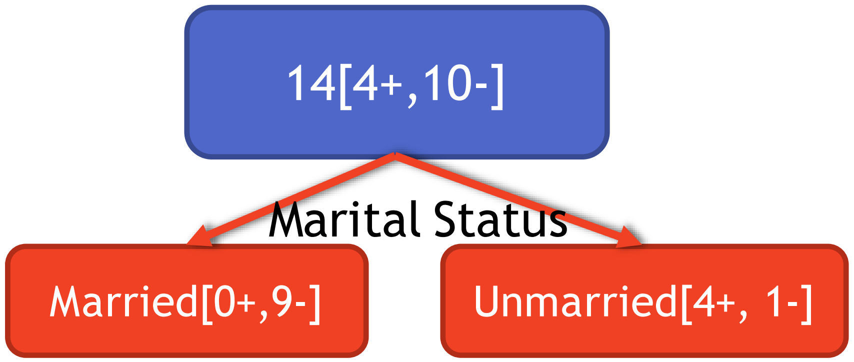
- Entropy([7+,1-]) Male= 54.3%
- Entropy([3+,3-]) Female = 100%
- Information Gain for Gender=86.3-((8/14)*54.3+(6/14)*100) =12.4



- Entropy([0+,9-]) Married = 0%
- Entropy([4+,1-]) Un Married= 72.1%
- Information Gain for Marital Status=86.3-((9/14)*0+(5/14)*72.1)=**60.5**

The Decision tree Algorithm- Demo

- The information gain for Marital Status is high, so it has to be the first variable for segmentation



- Now we consider the segment "Married" and repeat the same process of looking for the best splitting variable for this sub segment

The Decision tree Algorithm

Until stopped:

1. Select a leaf node
2. Find the best splitting attribute
3. Spilt the node using the attribute
4. Go to each child node and repeat step 2 & 3

Stopping criteria:

- Each leaf-node contains examples of one type
- Algorithm ran out of attributes
- No further significant information gain

Gini

$$Gini\ Index(S) = 1 - (p_1^2 + p_2^2)$$

- There are a few more alternatives available to us.
- Gini index can be used as an alternative to entropy.
- Both entropy and Gini calculate the impurity in a segment. Both Entropy and Gini are a measure of impurity.
- If a segment is pure, both Gini and Entropy are near to their lower limit and vice versa.
- Gini has a different formula, but the interpretation is all the same as that of Entropy.

Gini

Let S1 be an impure segment; then P_1=0.5 and P_2=0.5

$$Gini\ Index(S1) = \ 1 - (0.5^2 + 0.5^2)$$

$$Gini\ Index(S1) = \ 1 - (0.25 + 0.25)$$

$$Gini\ Index(S1) = \ 1 - (0.5)$$

$$Gini\ Index(S1) = \ 0.5$$

Lest S2 be a pure segment; then P_1=1 and P_2=0

$$Gini\ Index(S2) = \ 1 - (1^2 + 0^2)$$

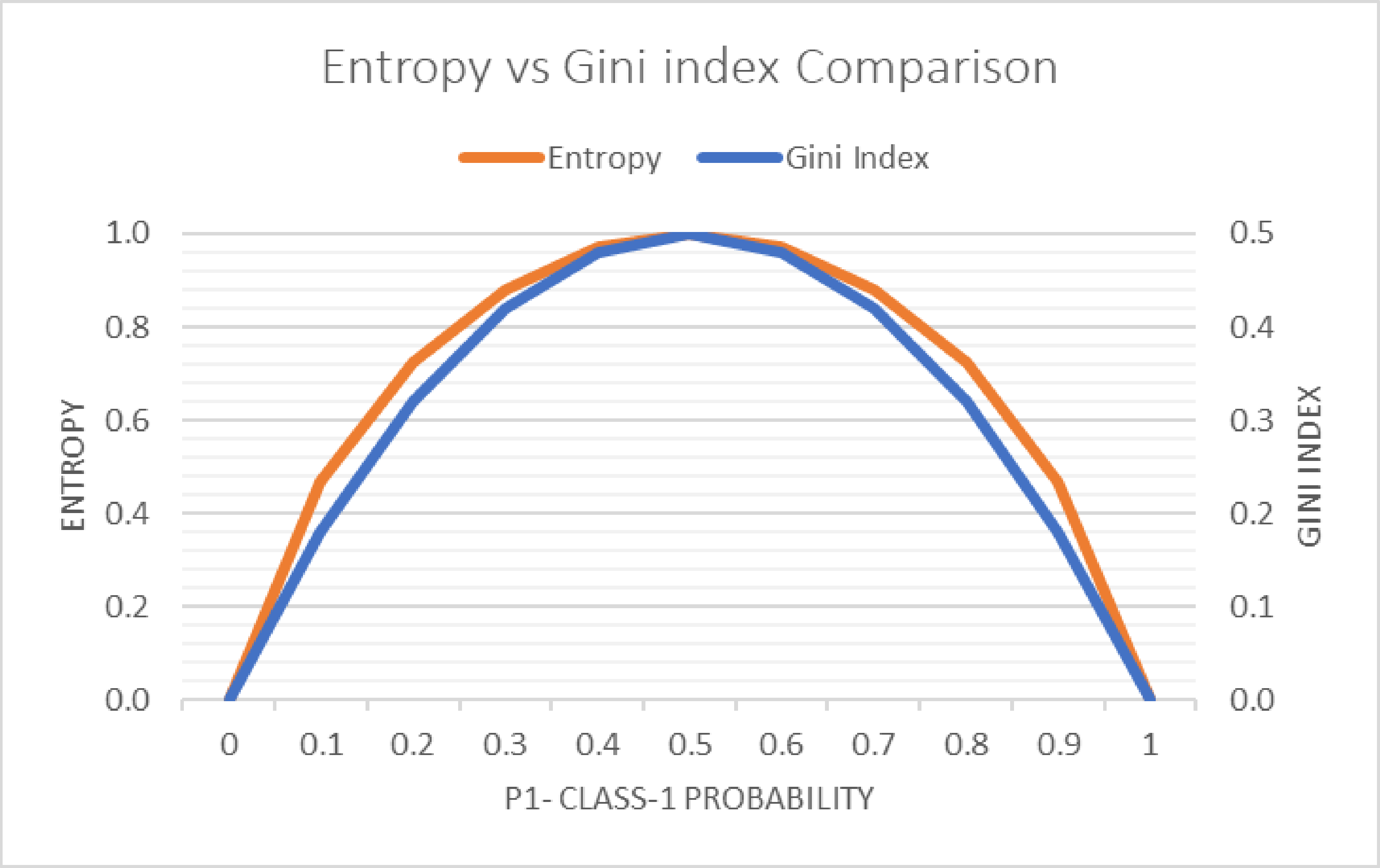
$$Gini\ Index(S2) = \ 1 - (1)$$

$$Gini\ Index(S2) = \ 0$$

Gini vs. Entropy Comparison

Segment	P_1	P_2	Entropy	Gini Index
S1	0.0001	0.9999	0.001	0.000
S2	0.01	0.99	0.081	0.020
S3	0.1	0.9	0.469	0.180
S4	0.2	0.8	0.722	0.320
S5	0.3	0.7	0.881	0.420
S6	0.4	0.6	0.971	0.480
S7	0.5	0.5	1.000	0.500
S8	0.6	0.4	0.971	0.480
S9	0.7	0.3	0.881	0.420
S10	0.8	0.2	0.722	0.320
S11	0.9	0.1	0.469	0.180
S12	0.99	0.01	0.081	0.020
S13	0.9999	0.0001	0.001	0.000

Gini vs. Entropy Comparison



LAB: Decision Tree Building

LAB: Decision Tree Building

- Data:Ecom_Cust_Relationship_Management/Ecom_Cust_Survey.csv
- How many customers have participated in the survey?
- Overall most of the customers are satisfied or dis-satisfied?
- Can you segment the data and find the concentrated satisfied and dis-satisfied customer segments ?
- What are the major characteristics of satisfied customers?
- What are the major characteristics of dis-satisfied customers?

Notes: Decision Tree Building

- Decision Tree building in python uses sci-kit learn
- The package expects the data to be in a specific format
- We need to do lot of data preparation before building the actual tree
  - Converting all variables into numerical variables
  - Column names also need to be formatted
  - Create predictor variables matrix and dependent variables

Code: Decision Tree Building

```
##Ecom_Cust_Survey = pd.read_csv('...',header = 0)
Ecom_data = pd.read_csv("https://raw.githubusercontent.cc

Ecom_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11805 entries, 0 to 11804
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Cust_num              11805 non-null  int64 
 1   Region                11805 non-null  int64 
 2   Age                   11805 non-null  int64 
 3   Order_Quantity        11805 non-null  int64 
 4   Customer_Type         11805 non-null  object
 5   Overall_Satisfaction  11805 non-null  object
dtypes: int64(4), object(2)
memory usage: 553.5+ KB
```

Code: Decision Tree Building

```
Ecom_data['Customer_Type_num'] = Ecom_data['Customer_Type'].map({'Prime': 1, 'Non_Prime': 0}).astype(int)
print(Ecom_data['Customer_Type'].value_counts())
print(Ecom_data['Customer_Type_num'].value_counts())
```

```
Prime        6804
Non_Prime    5001
Name: Customer_Type, dtype: int64
1    6804
0    5001
Name: Customer_Type_num, dtype: int64
```

```
Ecom_data['Overall_Satisfaction_num'] = Ecom_data['Overall_Satisfaction'].map( {'Dis Satisfied': 0, 'Satisfied': 1} ).astype(int).a
print(Ecom_data['Overall_Satisfaction'].value_counts())
print(Ecom_data['Overall_Satisfaction_num'].value_counts())
```

```
Dis Satisfied    6408
Satisfied        5397
Name: Overall_Satisfaction, dtype: int64
0    6408
1    5397
Name: Overall_Satisfaction_num, dtype: int64
```

Code: Decision Tree Building

```
from sklearn import tree

features= ['Region', 'Age', 'Order_Quantity',  'Customer_Type_num']
print("Feacures",features)

X = Ecom_data[features]
print("X shape", X.shape)
y = Ecom_data['Overall_Satisfaction']
print("Y shape", y.shape)
```

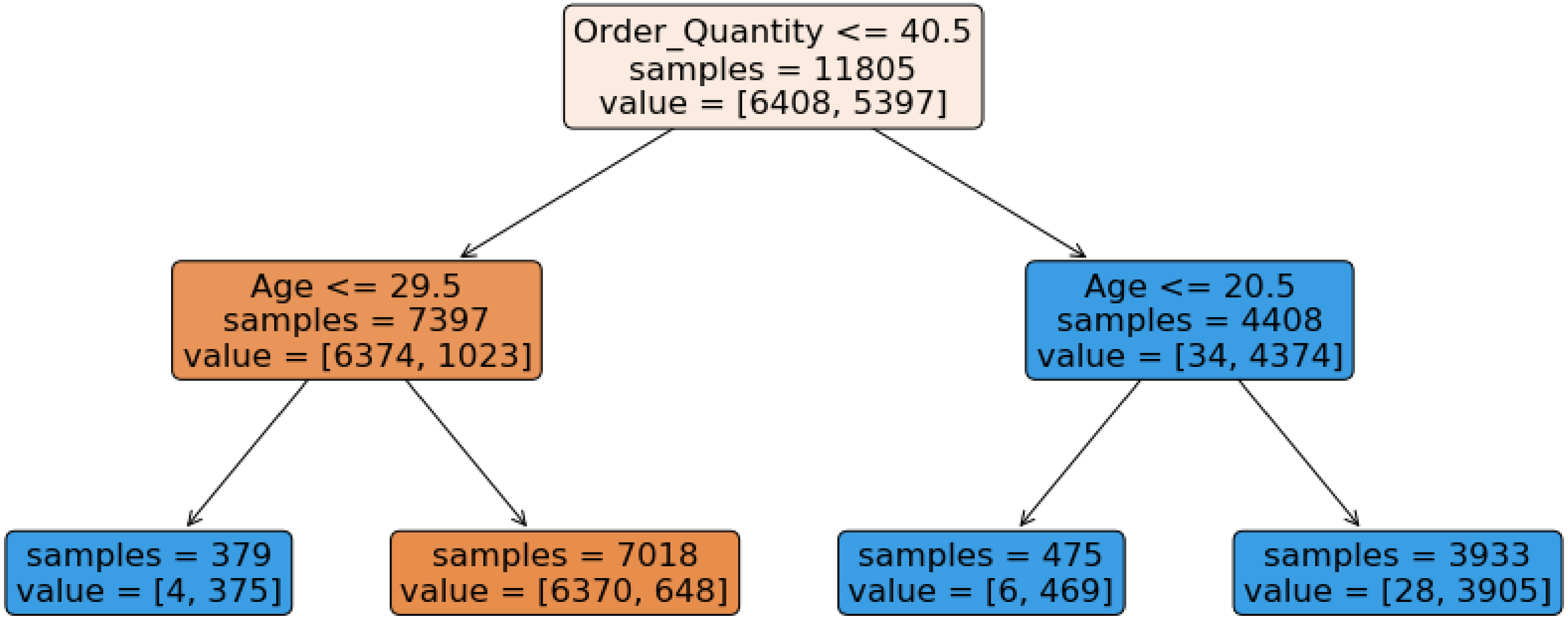
```
Feacures ['Region', 'Age', 'Order_Quantity', 'Customer_Type_num']
X shape (11805, 4)
Y shape (11805,)
```

Code: Decision Tree Building

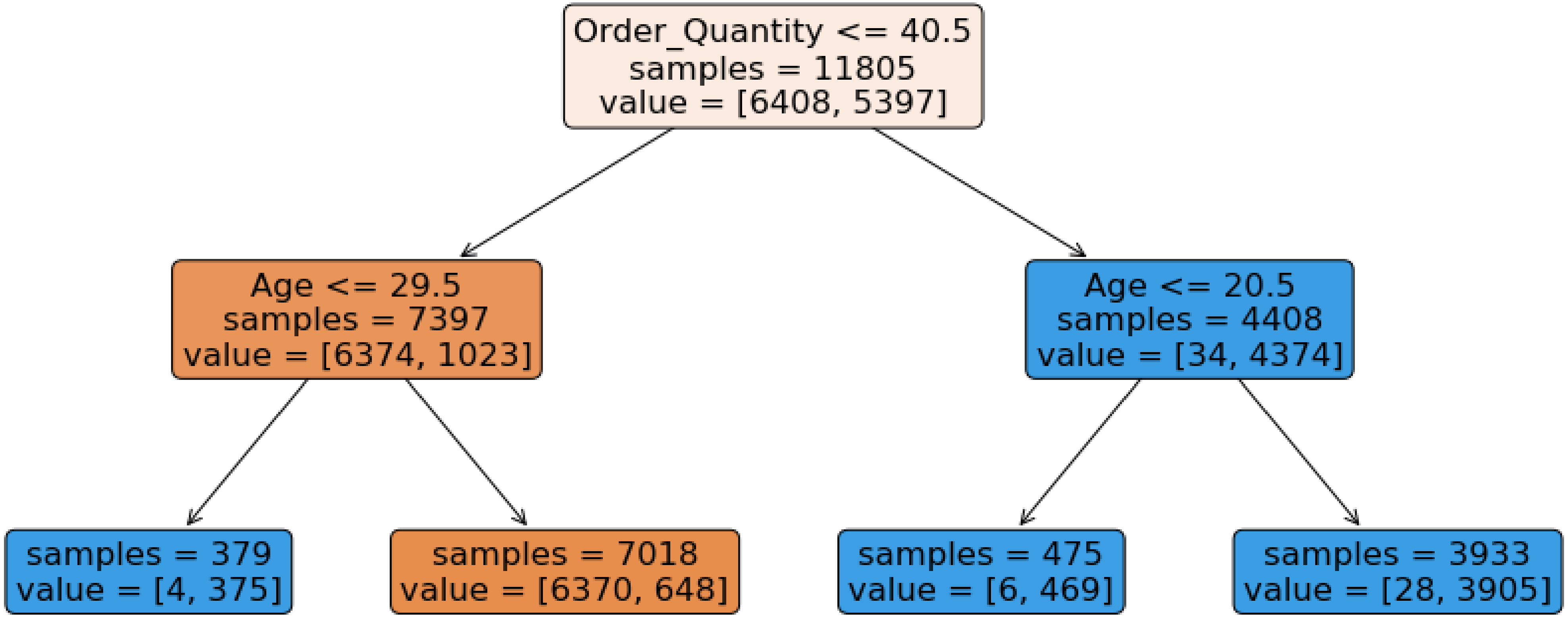
```
#Building Tree Model
DTree = tree.DecisionTreeClassifier(max_depth=2)
DTree.fit(X,y)

##Plotting the trees
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree, export_text
plt.figure(figsize=(15,7))
plot_tree(DTree, filled=True,
                     rounded=True,
                     impurity=False,
                     feature_names = features)
print( export_text(DTree, feature_names = features))
```

Output

Code: Decision Tree Building



Code: Decision Tree Building

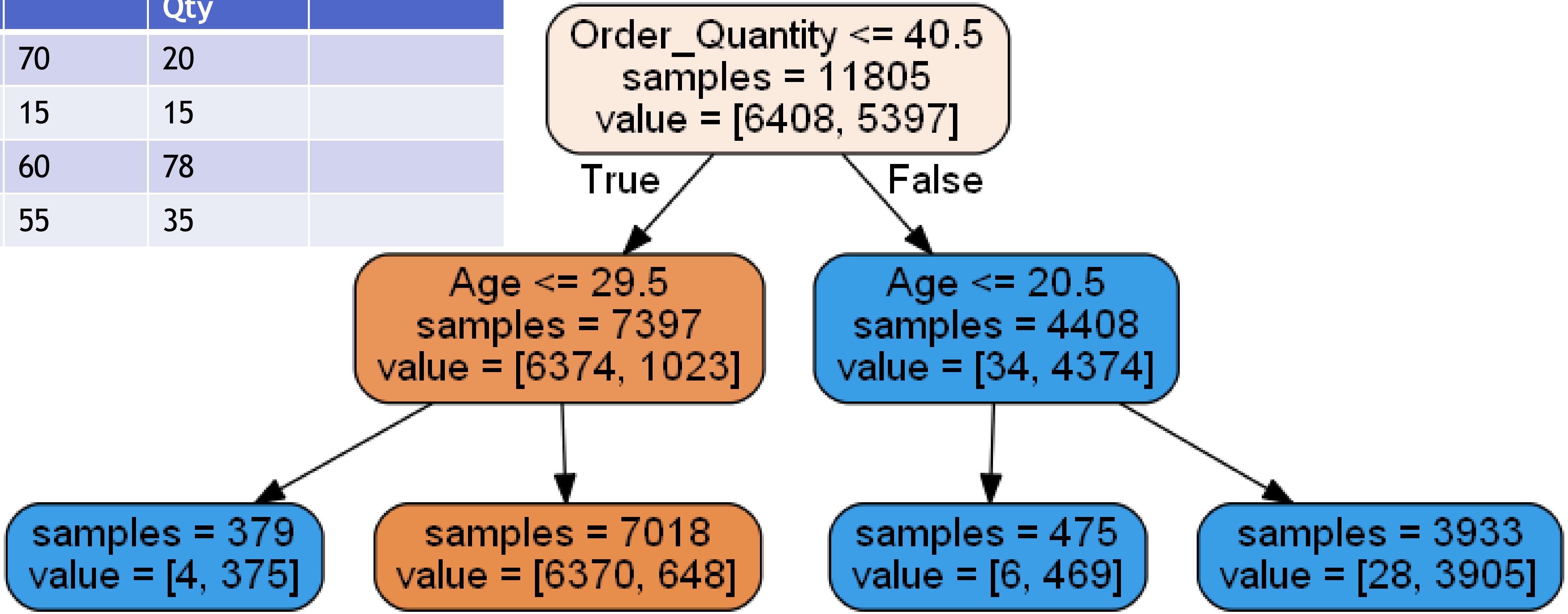


What is the final output?

- What is the final output of a machine learning model?
  - Is it the python code file?
  - Is it the data set?
  - Is it the predicted values file?
- What is the output of a regression model?
  - The equation /beta coefficients
- What is the output of a logistic regression model?
  - The equation / beta coefficients
- What is the output of a decision tree model?
  - The rules created by the leaf nodes.

Code: Decision Tree Building

	Age	Order Qty	Predicted
C1	70	20	
C2	15	15	
C3	60	78	
C4	55	35	



Just printing the rules

```
print(export_text(clf, feature_names = features))
```

```
In [82]: print(export_text(clf, feature_names = features))
|--- Order_Quantity <= 40.50
|   |--- Age <= 29.50
|   |   |--- class: 1
|   |--- Age >  29.50
|   |   |--- class: 0
|--- Order_Quantity >  40.50
|   |--- Age <= 20.50
|   |   |--- class: 1
|   |--- Age >  20.50
|   |   |--- class: 1
```

Tree Validation

Classification Table & Accuracy

Actual Classes \ Predicted Classes	0(Positive)	1(Negative)
0(Positive)	True positive (TP) Actual condition is Positive, it is truly predicted as positive	False Negatives(FN) Actual condition is Positive, it is falsely predicted as negative
1(Negative)	False Positives(FP) Actual condition is Negative, it is falsely predicted as positive	True Negatives(TN) Actual condition is Negative, it is truly predicted as negative

- Accuracy=(TP+TN)/(TP+FP+FN+TN)
- Misclassification Rate=(FP+FN)/(TP+FP+FN+TN)

LAB: Tree Validation

LAB: Tree Validation

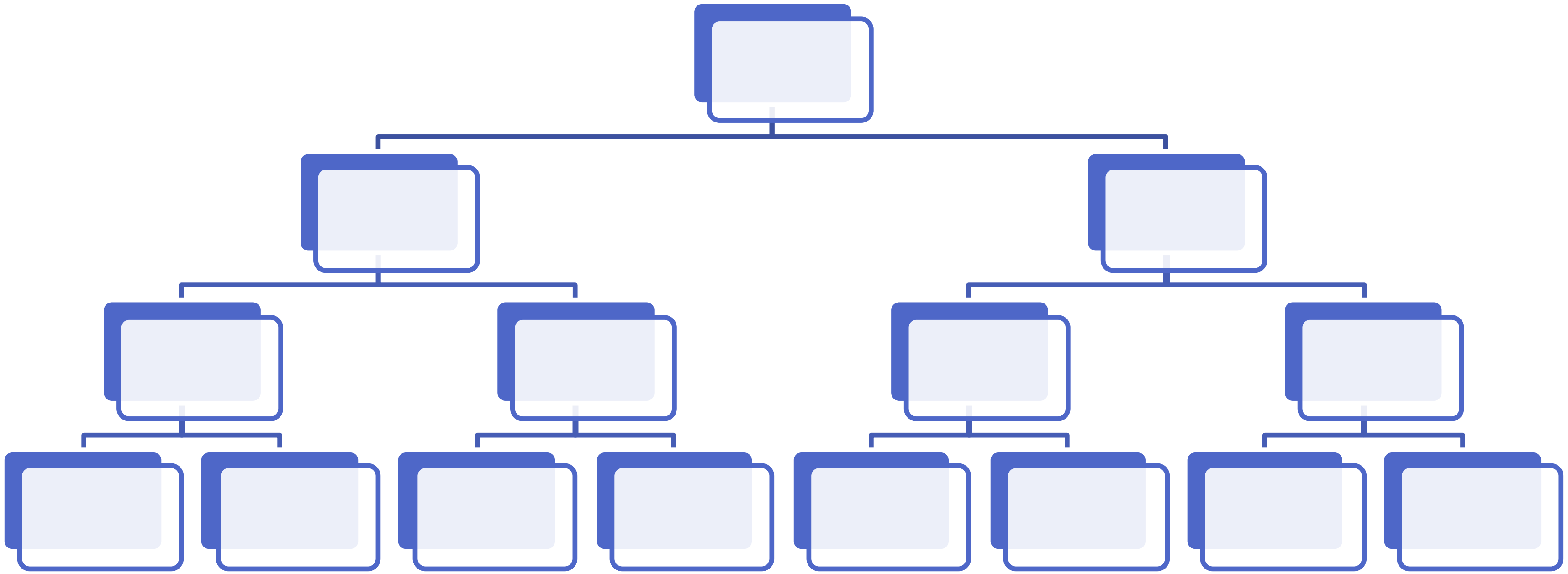
- Create the confusion matrix for the model
- Find the accuracy of the classification for the Ecom_Cust_Survey model

Code: Tree Validation

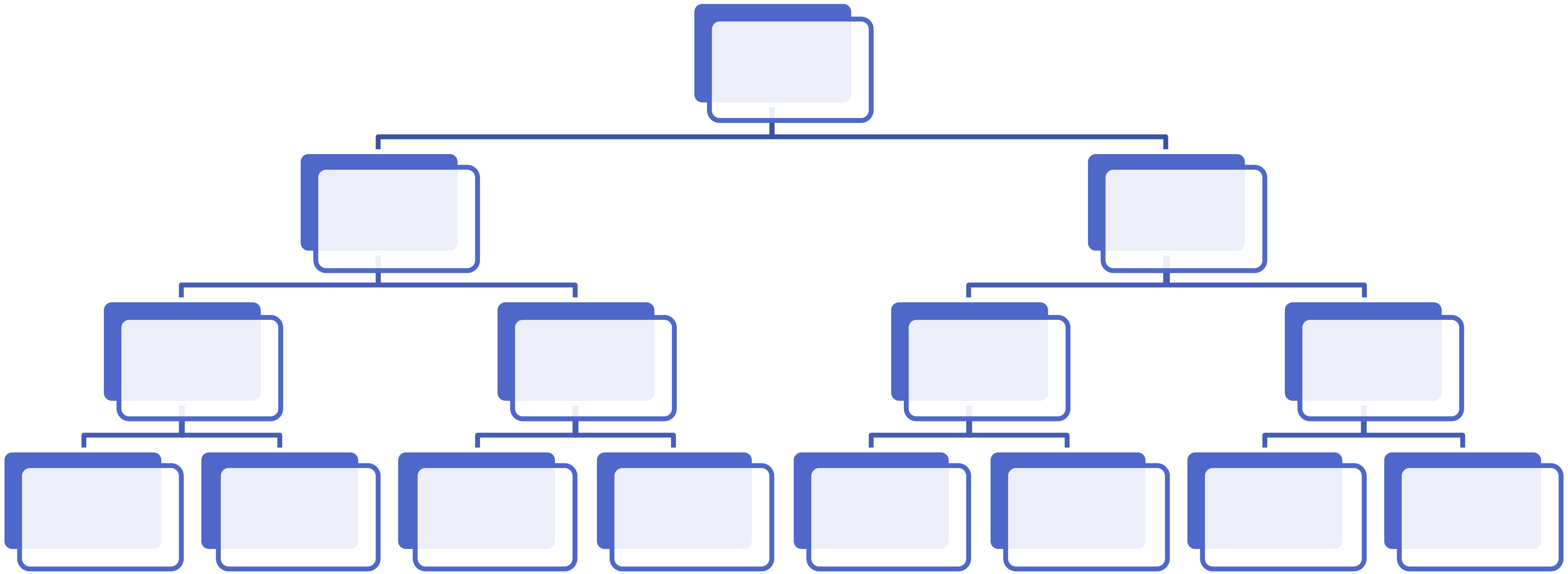
```
##########Tree Validation
#Tree Validation
predict1 = DTree.predict(X)

from sklearn.metrics import confusion_matrix ###for using confu
cm = confusion_matrix(y, predict1)
print (cm)

total = sum(sum(cm))
#####from confusion matrix calculate accuracy
accuracy = (cm[0,0]+cm[1,1])/total
print(accuracy)
```

```
[[6370   38]
 [ 648 4749]]
0.9418890300720034
```

Really Large Tree



Really Large Tree – Problem?



The Problem of Overfitting

LAB: The Problem of Overfitting

- Diabetes prediction based on Count_Pregnancies Glucose_level BP SkinThickness_index Insulin_level BMI DiabetesPedigreeFunction Age
- Dataset: “pima diabetes/Train_data.csv”
- Import both test and training data
- Build a decision tree model on training data
- Find the accuracy on training data
- Find the predictions for test data
- What is the model prediction accuracy on test data?

Code: Accuracy on training and test data

```
clf = tree.DecisionTreeClassifier()
clf.fit(X_train,y_train)

predict1 = clf.predict(X_train)
predict2 = clf.predict(X_test)

#On Train Data
cm1 = confusion_matrix(y_train,predict1)
total1 = sum(sum(cm1))
accuracy1 = (cm1[0,0]+cm1[1,1])/total1
print("Train Accuracy", accuracy1)

#On Test Data
cm2 = confusion_matrix(y_test,predict2)
total2 = sum(sum(cm2))
accuracy2 = (cm2[0,0]+cm2[1,1])/total2
print("Test Accuracy", accuracy2)
```

```
Train Accuracy 1.0
Test Accuracy 0.7597402597402597
```

statinfer

The Problem of Overfitting

- If we further grow the tree we might even see each row of the input data table as the final rules
- The model will be really good on the training data but it will fail to validate on the test data
- Growing the tree beyond a certain level of complexity leads to overfitting
- A really big tree is very likely to suffer from overfitting.

Pruning

Pruning to Avoid Overfitting

- Pruning helps us to avoid overfitting
- Generally it is preferred to have a simple model, it avoids overfitting issue
- Any additional split that does not add significant value is not worth while.
- We can avoid overfitting by changing the parameters like
  - max_depth
  - max_leaf_nodes

Pruning Parameters

- max_depth
  - Reduce the depth of the tree to build a generalized tree
  - Set the depth of the tree to 3, 5, 10 depending after verification on test data
- max_leaf_nodes
  - Reduce the number of leaf nodes

The problem of under fitting

The problem of under-fitting

- Simple models are better. Its true but is that always true? May not be always true.
- We might have given it up too early. Did we really capture all the information?
- Did we do enough research and future reengineering to fit the best model? Is it the best model that can be fit on this data?
- By being over cautious about variance in the parameters, we might miss out on some patterns in the data.
- Model need to be complicated enough to capture all the information present.

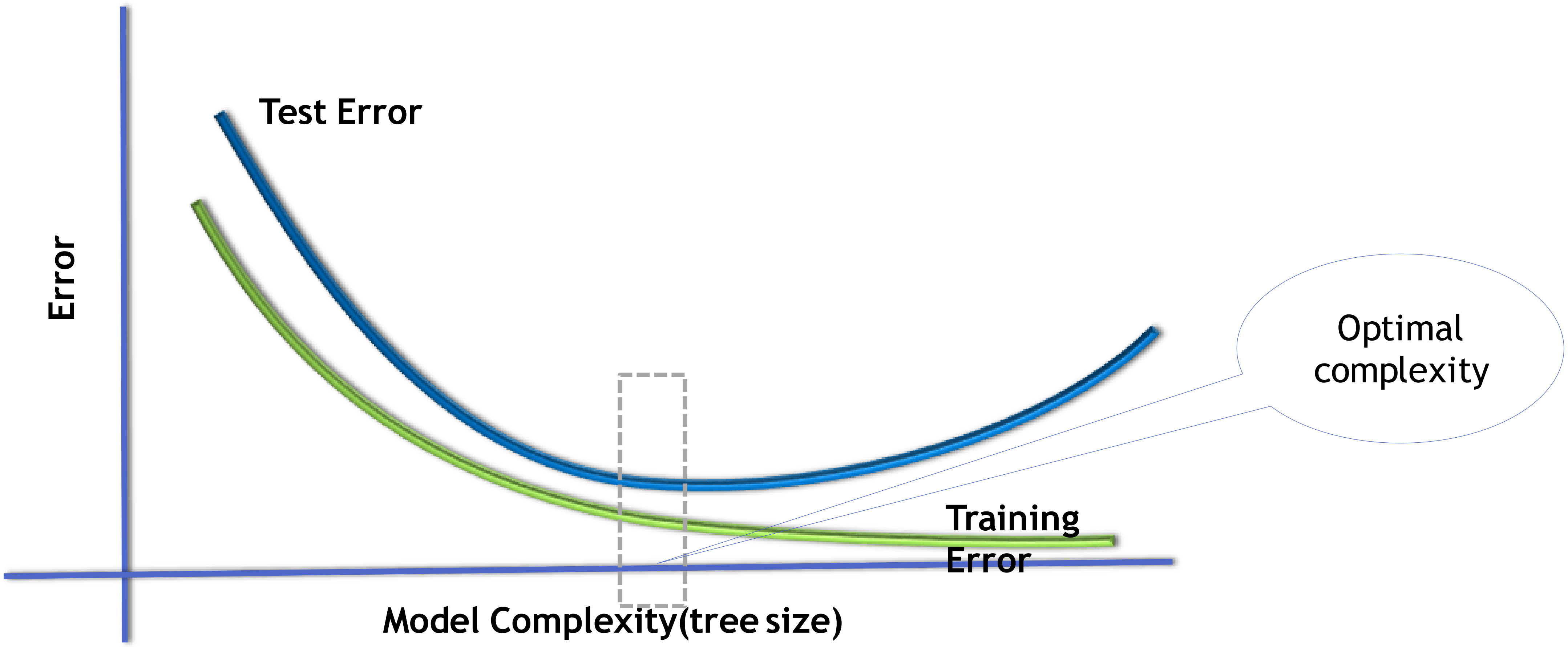
The problem of under-fitting

- If the training error itself is high, how can we be so sure about the model performance on unknown data?
- Most of the accuracy and error measuring statistics give us a clear idea on training error, this is one advantage of under fitting, we can identify it confidently.
- Under fitting
  - A model that is too simple
  - A mode with a scope for improvement

Overfitting and Underfitting

- Overfitted model - Variance
  - High training accuracy- Low test accuracy
  - Small changes in the data causes a lot of change the model parameters
  - Model with lot of variance
- Underfitted model - Bias
  - Low training accuracy
  - Model with inherent bias in its parameter estimates
  - Model with lot of bias

Bias-Variance trade off



How to Finetune Pruning Parameters?

2	7	15	22	30
Under fitting				Over fitting

LAB: Pruning

- Rebuild the model for above data
- Prune the decision tree or rebuild it with optimal parameters
- Calculate the training and test error
- Check whether there is an issue of overfitting in the final model

Code: Pruning

```
dtree = tree.DecisionTreeClassifier(max_leaf_nodes =10) #Try 3,4,5,6
```

```
#training Tree Model
clf = tree.DecisionTreeClassifier(max_leaf_nodes = 10)
clf.fit(X_train,y_train)

predict1 = clf.predict(X_train)
predict2 = clf.predict(X_test)

#On Train Data
cm1 = confusion_matrix(y_train,predict1)
total1 = sum(sum(cm1))
accuracy1 = (cm1[0,0]+cm1[1,1])/total1
print("Train Accuracy", accuracy1)

#On Test Data
cm2 = confusion_matrix(y_test,predict2)
total2 = sum(sum(cm2))
accuracy2 = (cm2[0,0]+cm2[1,1])/total2
print("Test Accuracy", accuracy2)
```

```
Train Accuracy 1.0
Test Accuracy 0.16666666666666666
```

Steps in Building the Decision Tree Model

1. Overall Data → Train(80)   Test(20) [70-30 ; 85 -15 ; 90-10]
2. Build the model on train data
3. Confusion matrix and accuracy (>80%)
   1. Train Accuracy
   2. Test Accuracy
4. Pruning
   1. Max depth (infinity)
      1. Too High – Overfitted
      2. Too Low – Underfitted
   2. Max Leaf Nodes (infinity)
      1. Too High – Over fitting
      2. Too Low – Underfitting
5. Finalize the model when you have highest accuracy on Training data and matching accuracy on test

LAB: Tree Building & Model Selection

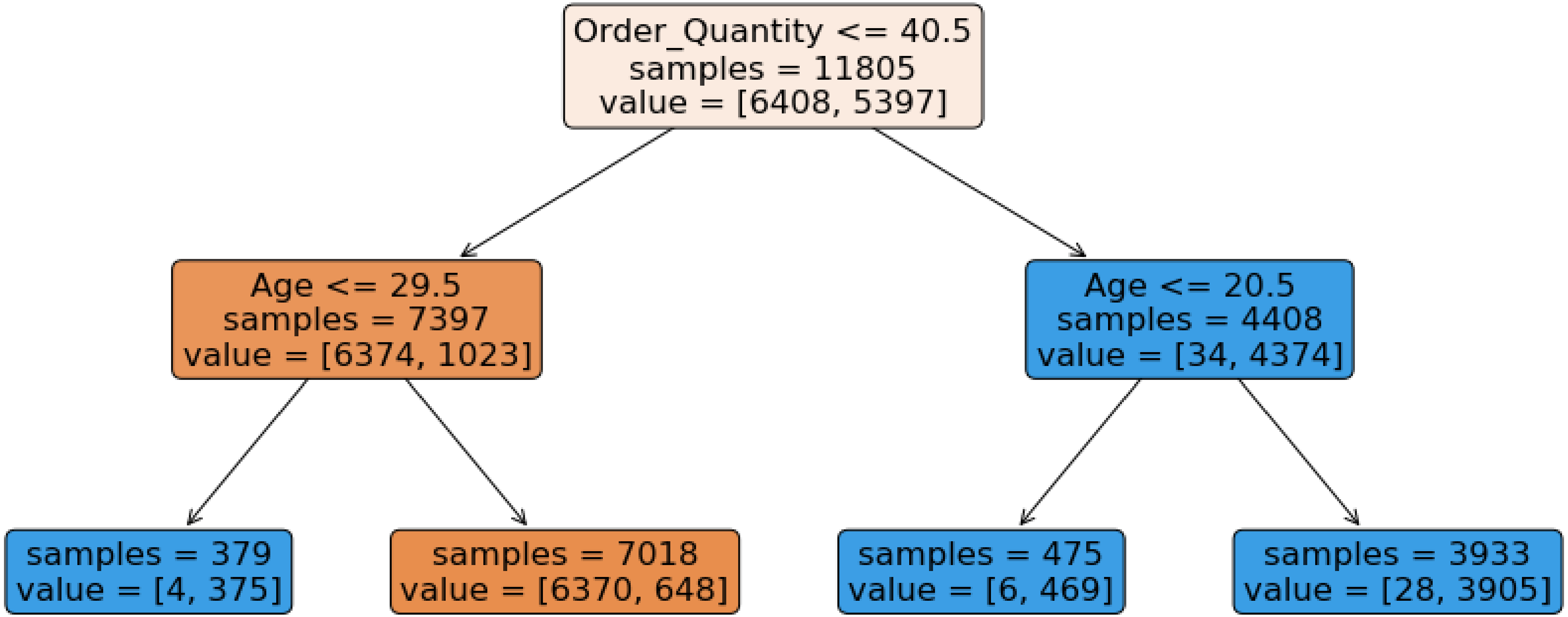
When to use which model?

- Output is Buying vs. Not buying and trying to set of customers who can buy
- Output is responder vs non responder and we are trying predict the response score of customers
- Output is Defaulter vs. Non Defaulter and trying to predict credit score
- Output is responder vs non responder and we are trying to target responder segments

LAB: Tree Building & Model Selection

- Import fiber bits data. This is internet service provider data. The idea is to predict the customer attrition based on some independent factors
- Build a decision tree model for fiber bits data
- Prune the tree if required
- Find out the final accuracy
- Is there any 100% active/inactive customer segment?

Decision Tres Applications



Decision Tres Applications

- Marketing segmentation for campaigning
- Customer Segmentation in Telecom
- Sales Segmentation
- Insurance segmentation

Conclusion

Conclusion

- Decision trees are powerful and very simple to represent and understand.
- One need to be careful with the size of the tree. Decision trees are more prone to overfitting than other algorithms
- Can be applied to any type of data, especially with categorical predictors
- One can use decision trees to perform a basic customer segmentation and build a different predictive model on the segments
- In python you will get overfitted models with default parameters. You need to adjust the parameters to avoid overfitting

statinfer

Thank you