Decision Trees

Case Study : Predicting loan defaults with decision trees

- Requirement : Need to buy a house. Need Loan
- Risk Assessment:
  - Credit - Previous payment of loan on time (excellent/good/fair)
  - Income
  - Term - Loan term - Needed to repay the loan.
  - Personal Information - Age, Gender, etc

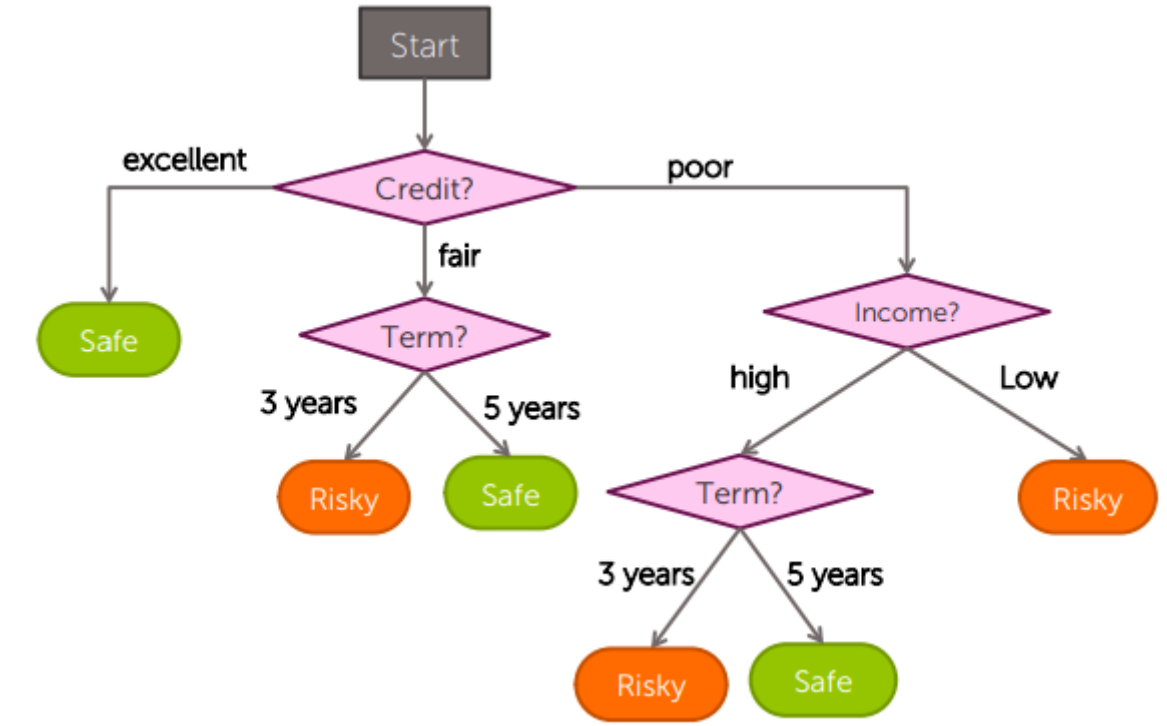
Intelligent application

- Loan Applications are sent into the **Intelligent loan application review system** and classified as **Safe** / **Risky**.

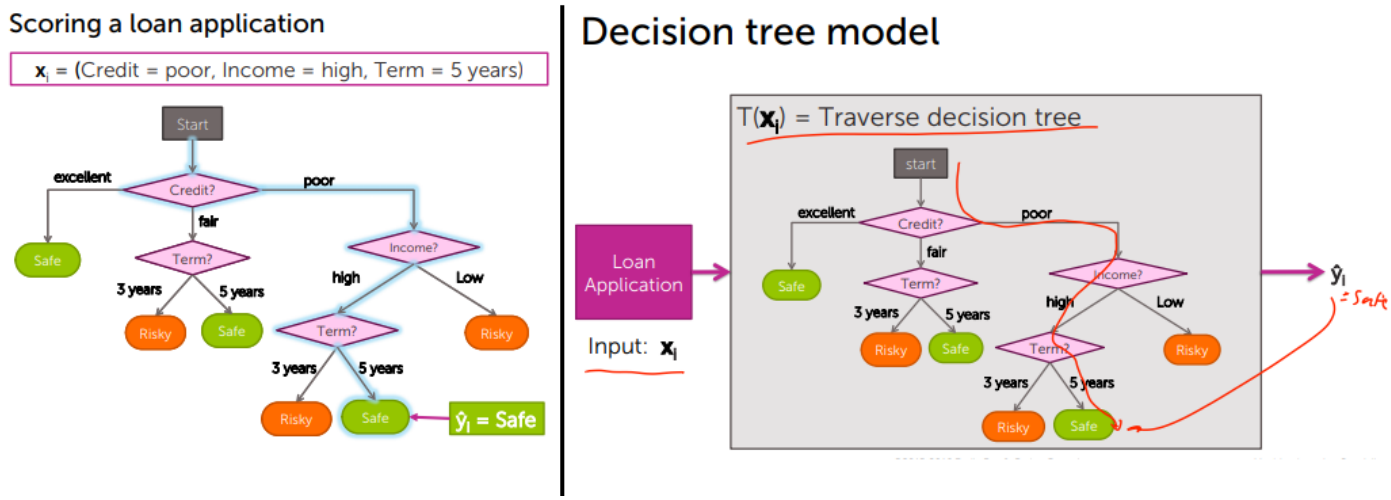
Classifier Review

```
                                        ----> Safe (y-hat = 1)
                                        |
Loan Application --> Classifier Model--- (Output:y-hat predicted class)
 (Input-xi)                             |
                                        ----> Risky (y-hat = -1)
```

Decision Tree -> Loan Application



Intution behind decision trees



Decision Tree Learning Task

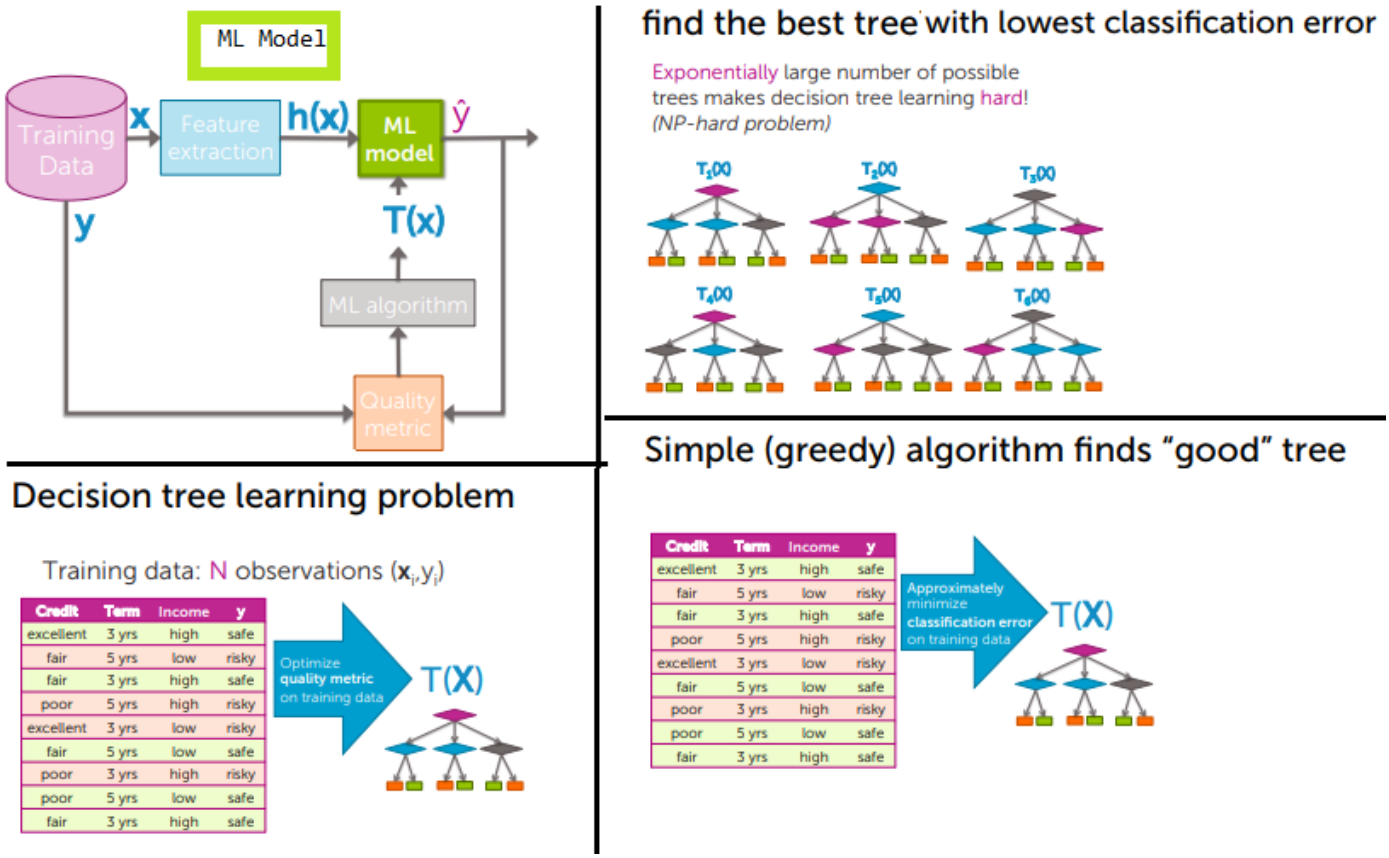
- The training data has N observations (xi, yi).
- **Quality Metrics : Classification Error**
  - Error = #incorrect predictions / #examples
  - Best Value = 0.0
  - Worst Value = 1.0

Finding the best decision tree

- Give the feature set, multiple decision trees can be formulated and therefore identifying the best decision tree could be a tedious task.
  - Exponentially large number of possible trees hence - makes decision tree learning hard.

Simple (greedy) algorithm

- It is employed to find "good" trees.
- It minimizes classification error on the training data.
- Here incrementally the tree is built, one layer each time in order to get the best classification error each time.



Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

Recursive greedy algorithm

Algorithm:

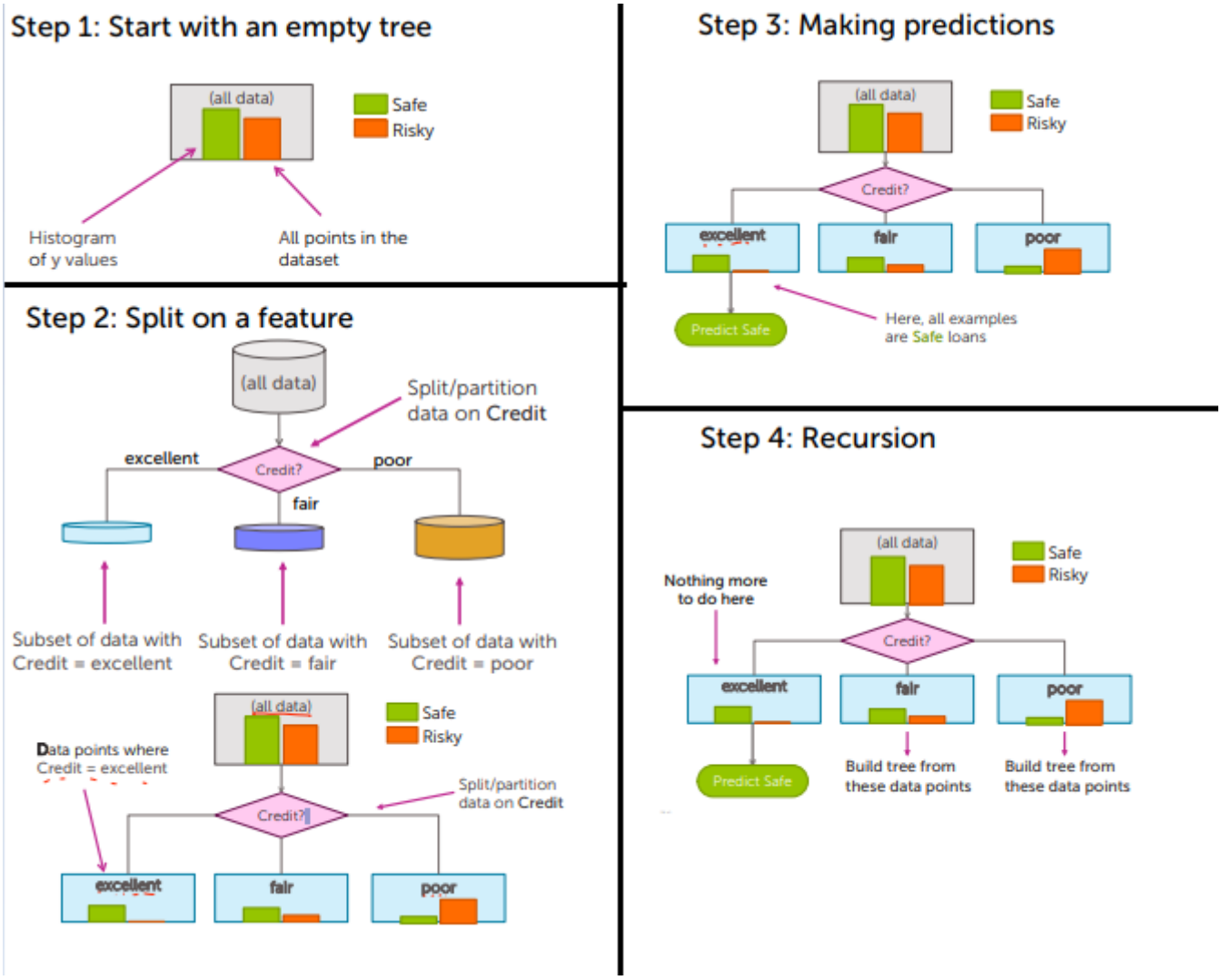
- Step 1 - Start with an empty tree
- Step 2 - Select a feature to split data
- For each split of the tree:
  - Step 3 - If nothing more to do, make predictions.
  - Step 4 - Else, Step 2 and continue recurse on split.

Algorithm - w.r.t case study - Loan defaulty Prediction

- Step 1 : Start with an empty tree.
  - Have all the data -> dataset y values are represented as histogram.
- Step 2 : Split on a feature
  - Split on the credit feature. The values can be -> **Excellent / Fair / Poor**
  - Represent the split datasets as histogram of the y values.
- Step 3 : since all the data in excellent subset are **Safe**. The loan can be guaranteed and no further recursion is needed.
- Step 4 : On the otherhand - Fair and poor subset needed to be further split on other features inorder to classify them. It can be classified on **Term** / **Income**, etc.

Problems

1. Feature split selection
2. Stopping condition -> when to stop recursion and predict the value.



Feature split learning => Decision stump learning

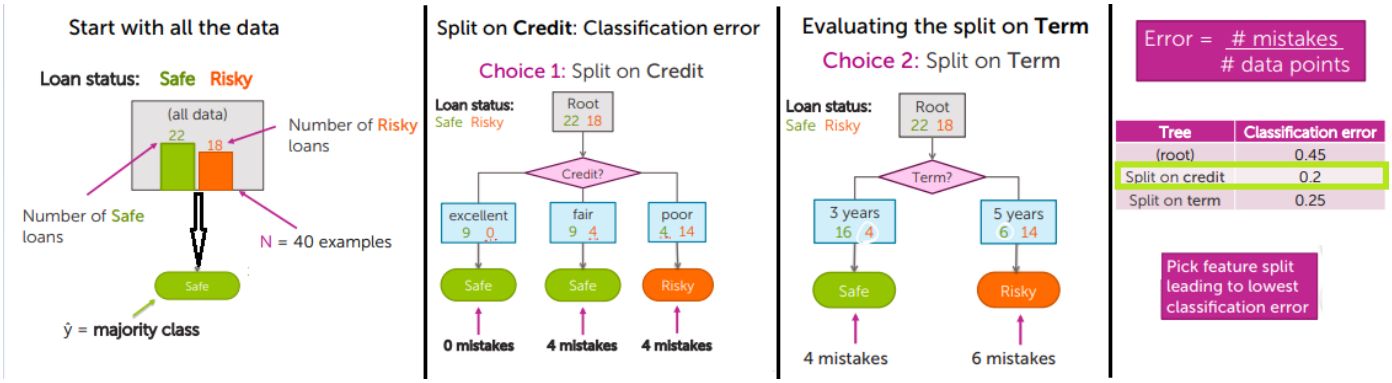
- Decision Stumps are **Sngle Level Trees**.
- The selection of feature to split on is based on classification error that is obtain on the selection of the feature. The feature with least classification must be choosen to split.
- For each intermediate setp -> The predicted y-hat is the majority value of the result. E.g For **Fair** subset -> Safe = 9, Risky = 4; Since the majority is **Safe** all the loan applications are classified as **Safe**.

Feature split selection algorithm

- Given a subset of data M (a node of the tree)
- For each feature hi(x): (Eg - Credit, Term, Income, etc)
  1. Split data of M according to feature hi(x)
  2. Compute the classification error split.
- Choose feature h * (x) with lowest classification error. (Credit).

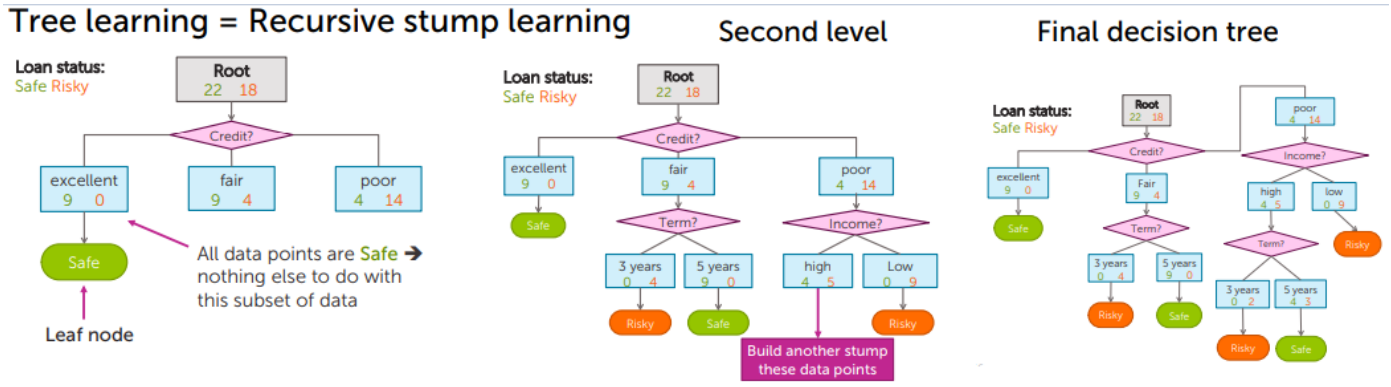
Greedy decision tree algorithm

- Step 1 - Start with an empty tree.
- Step 2 - Select a feature to split data. **Pick feature split that leads to lowest classification error**.
- For each split of the tree:
  - Step 3 : If nothing more to split, make predictions.
  - Step 4 : Otherwise, go to Step 2 and continue feature split.



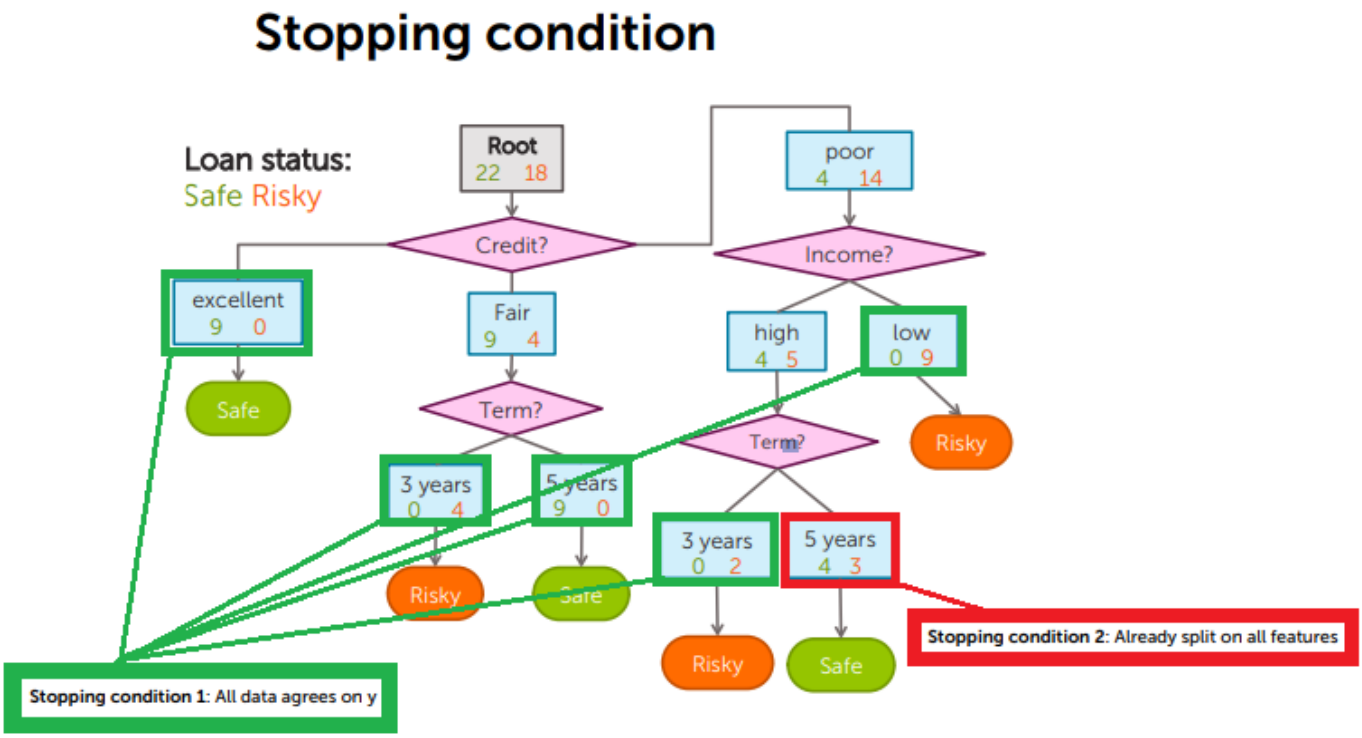
Recursion & Stopping conditions

- Stopping conditions
  1. All data agrees on y(expected output).
  2. Already split on all features.
     - In the above case, for excellent subset all the data agree hance no need to recurse further but for other two subset need to recurse further.
     - Select a feature with lowest classification error on each subset and split and continue until all agree to y, or run out of features.



Greedy decision tree algorithm

- Step 1 - Start with an empty tree.
- Step 2 - Select a feature to split data. **Pick feature split that leads to lowest classification error**.
- For each split of the tree:
  - Step 3 : If nothing more to split, make predictions. **Stopping condition - (All data agree to y, all feature splits done)**.
  - Step 4 : Otherwise, go to Step 2 and continue feature split. **(Recursion)**.



Decision Tree Prediction Algorithm

- **predict(tree_node, input)**
  - If current tree_node is a leaf:
    - return majority class of data points in leaf
  - else:
    - next_node = child node of tree_node whose feature value agrees with input.
    - return **predict(next_node, input)**

Multiclass decision stump

- The predicted label can be based on the majority label in each subset.
- The prediction can be done based on probability as well.

Learning decision trees with continuous data

- Consider Income to be a feature to split the dataset into loan default classification.
- Income is a real value and can vary vastly and assessed variedly.

```
- $30 K -> risky
- $34 K -> safe
- $40 K -> risky
```

- Spliting on each of the numeric value and predicting the label could be dangerous, since each node may contain only one data point. It could also lead to **overfitting**.

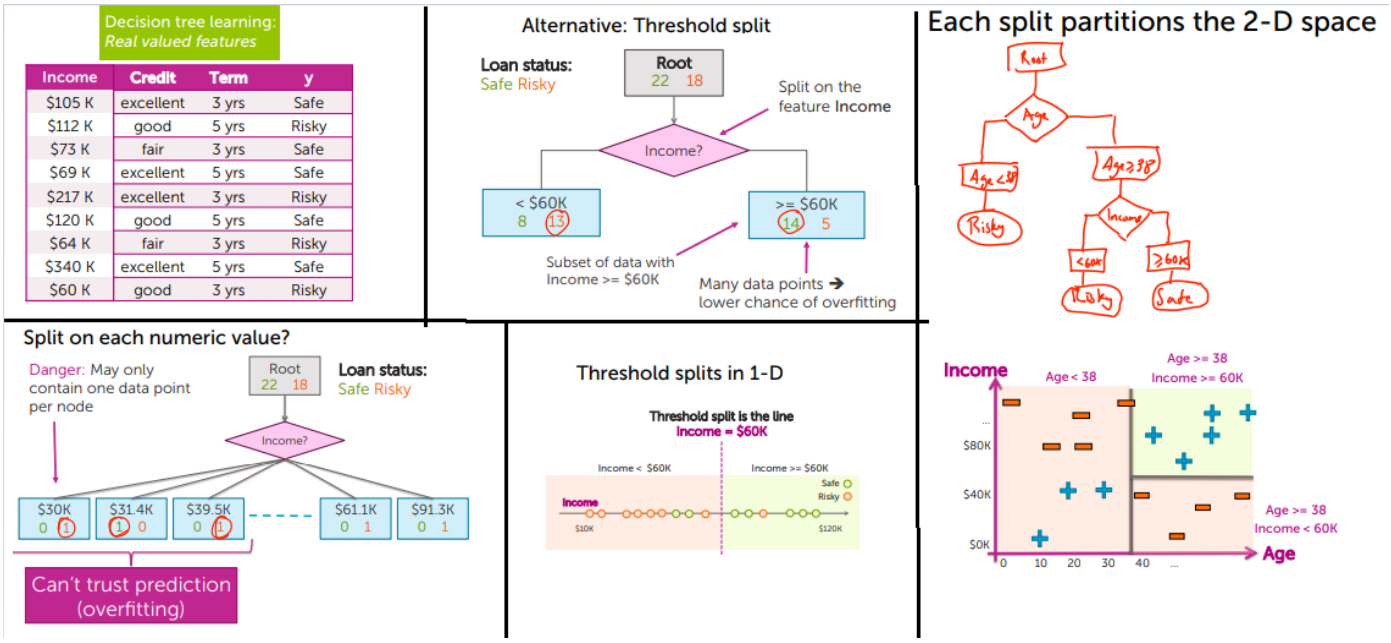
Alternative : Threshold split

- Rather than splitting on each of the numerical income value post sorting it is
  - tedious process
  - could lead to overfitting
  - many nodes could contain only one datapoint

- Therefore Threshold split is implemented.
  - Where the splitting is based on a threshold value (eg - 60K.)
  - Based on the classification error on each step -> further classification are made.
  - The same threhold value can be used to make further splits, unlike previously where one feature was used only once to split the dataset.

Finding the best threshold split

- Step 1 : Sort the values of a feature hj(x) (E.g: Income) Let {v1, v2, v3, ... vn} denote the sorted values.
- Step 2 :
  - For i = 1 ... N-1
    - Consider split ti = (vi + vi+1) / 2
    - Compute classification error for threshold split hj(x) >= ti.
  - Choose the t * with the lowest classification error.



Quiz

1. Questions 1 to 6 refer to the following common scenario:

Consider the following dataset:

x1	x2	x3	y
1	1	1	+1
0	1	0	-1
1	0	1	-1
0	0	1	+1

Let us train a decision tree with this data. Let's call this tree T1. What feature will we split on at the root?

- ( ) x1
- ( ) x2
- (•) x3

Explaination :

```
first split on x1 : error = 2/4
assume x1=0 predicts y=+1
    x1    x2    x3    y
    1    1    1    +1
    1    0    1    -1 [error]
    0    0    1    +1 [error]
    0    1    0    -1
NB: if we test x1=0 predicts y=-1 we also get error=2/4


first split on x2 : error = 2/4
assume x2=0 predicts y=+1
    x1    x2    x3    y
    1    1    1    +1
    0    1    0    -1 [error]
    1    0    1    -1
    0    0    1    +1 [error]
NB: if we test x2=0 predicts y=-1 we also get error=2/4


first split on x3 : error = 1/4
assume x3=0 predicts y=+-1
    x1    x2    x3    y
    0    1    0    -1
    1    0    1    -1 [error]
    0    0    1    +1
    1    1    1    +1
```

- splitting on x3 @ root yields the lowest number of errors.

---

2. Refer to the dataset presented in Question 1 to answer the following.

Fully train T1 (until each leaf has data points of the same output label). What is the depth of T1?

3

Explaination:

```
START

split level 1 :

X3=1
x1    x2    x3    y
1    1    1    +1
0    0    1    +1
1    0    1    -1

X3=0
x1    x2    x3    y
0    1    0    -1
no further split required on this leaf.

----------
starting point for this node
X3=1
x1    x2    x3    y
1    1    1    +1
0    0    1    +1
1    0    1    -1

split on x2
x2 =1
x1    x2    x3    y
1    1    1    +1
no further split required on this leaf.


x2=0
x1    x2    x3    y
0    0    1    +1
1    0    1    -1

------------------------------
starting point for this node
X3=1 & x2=0
x1    x2    x3    y
0    0    1    +1
1    0    1    -1

split on x1
x1=1
x1    x2    x3    y
1    0    1    -1
no further split required on this leaf.

x1=0
x1    x2    x3    y
0    0    1    +1
no further split required on this leaf.
```

```
conclusion: three (3) split levels required : X3, then X2, then X1.
Answer = 3
```

In [9]:

```
import graphlab
x = graphlab.SFrame({'x1':[1,0,1,0],'x2':['1','1','0','0'],'x3':['1','0','1','1'],'y':['1',
x
```

Out[9]:

x1	x2	x3	y
1	1	1	1
0	1	0	-1
1	0	1	-1
0	0	1	1

[4 rows x 4 columns]

In [10]:

```
features = ['x1','x2','x3']
target = 'y'

decision_tree_model = graphlab.decision_tree_classifier.create(x, validation_set=None,
                                target = target, features = features)
```

```
WARNING: The number of feature dimensions in this problem is very large in c
omparison with the number of examples. Unless an appropriate regularization
value is set, this model may not provide accurate predictions for a validati
on/test set.

Decision tree classifier:

--------------------------------------------------------

Number of examples          : 4

Number of classes           : 2

Number of feature columns   : 3

Number of unpacked features : 3

+-----------+--------------+-------------------+-------------------+

| Iteration | Elapsed Time | Training-accuracy | Training-log_loss |

+-----------+--------------+-------------------+-------------------+

| 1         | 0.000000     | 1.000000          | 0.634946          |

+-----------+--------------+-------------------+-------------------+
```

In [11]:

```
graphlab.canvas.set_target('ipynb')
decision_tree_model.show()
```

In [7]:

```
decision_tree_model.show(view="Tree")
```

---

3. Refer to the dataset presented in Question 1 to answer the following.

What is the training error of T1?

0

Training error is zero since we recursively split until every leaf had one element or 100% accuracy.

---

4. Refer to the dataset presented in Question 1 to answer the following.

Now consider a tree T2, which splits on x1 at the root, and splits on x2 in the 1st level, and has leaves at the 2nd level. Note: this is the XOR function on features 1 and 2. What is the depth of T2?

2

Depth = number of splits : split on x1 and split on x2 : 2 splits.

---

5. Refer to the dataset presented in Question 1 to answer the following.

What is the training error of T2?

0

```
error = number of incorrect predictions
        ------------------------------
        number of examples.

tree T2: split on x1, then split on x2.

Starting point.
x1    x2    x3    y
1     1     1    +1
0     1     0    -1
1     0     1    -1
0     0     1    +1

------------
split on x1

x1=0
x1    x2    x3    y
0     1     0    -1
0     0     1    +1


x1=1
x1    x2    x3    y
1     1     1    +1
1     0     1    -1
-----------------
level 2 of tree: split on x2

start from x1=0
x1    x2    x3    y
0     1     0    -1
0     0     1    +1

x1=0 & x2=0
x1    x2    x3    y
0     0     1    +1 [correct]

x1=0 & x2=1
x1    x2    x3    y
0     1     0    -1 [correct]


start from x1=1
x1    x2    x3    y
1     1     1    +1
1     0     1    -1

x1=1 & x2=0
x1    x2    x3    y
1     0     1    -1 [correct]
```

```
x1=1 & x2=1
x1    x2    x3    y
1    1    1    +1 [correct]

4 correct predictions, zero incorrect predictions.
```

---

6. Refer to the dataset presented in Question 1 to answer the following.

Which has smaller depth, T1 or T2?

- ( ) T1
- (•) T2

T2 has smaller depth.

- simple algorythm chose splitting on x3 first because it gave lower error on first split.
- Above splitting on x3 from root results in more steps required - slower result, not as good algorythm.

---

7. (True/False) When deciding to split a node, we find the best feature to split on that minimizes classification error.

- (•) True
- ( ) False

---

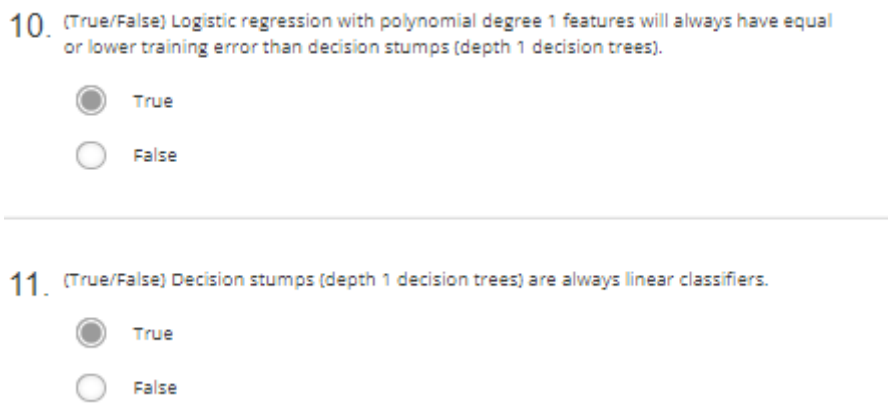
8. If you are learning a decision tree, and you are at a node in which all of its data has the same y value, you should

- ( ) find the best feature to split on
- (•) create a leaf that predicts the y value of all the data
- ( ) terminate recursions on all branches and return the current tree
- ( ) go back to the PARENT node and select a DIFFERENT feature to split on so that the y values are not all the same at THIS node

---

9. Consider two datasets D1 and D2, where D2 has the same data points as D1, but has an extra feature for each data point. Let T1 be the decision tree trained with D1, and T2 be the tree trained with D2. Which of the following is true?

- ( ) T2 has better training error than T1
- ( ) T2 has better test error than T1
- (•) Too little information to guarantee anything



http://web.engr.oregonstate.edu/~xfern/classes/cs534/notes/decision-tree-7-11.pdf (http://web.engr.oregonstate.edu/~xfern/classes/cs534/notes/decision-tree-7-11.pdf)

https://www.ime.unicamp.br/~dias/Intoduction%20to%20Statistical%20Learning.pdf (https://www.ime.unicamp.br/~dias/Intoduction%20to%20Statistical%20Learning.pdf)