



k -NN, DECISION TREES, AND RANDOM FOREST FOR REGRESSION AND CLASSIFICATION

RESMI SURESH

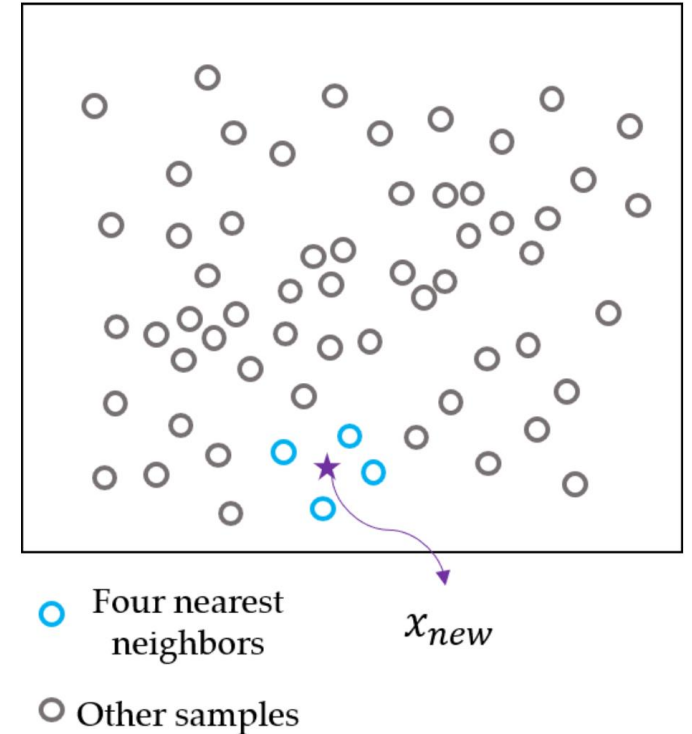
ASSISTANT PROFESSOR, IIT GUWAHATI

k -NEAREST NEIGHBORS

REGRESSION

FUNCTIONAL APPROXIMATION USING k -NEAREST NEIGHBOUR

- k -NN is a non-parametric supervised learning method
- Philosophy: If you want to know someone, understand the neighbors
- It is also called 'Lazy algorithm' as there's no training phase, just holding on to x, y data. – No model is built during training phase
- A distance metric (usually Euclidean) is needed to identify the nearest neighbours
- A tunable parameter ' k ' is chosen to get the best performance
- For a new point x_{new} , the k -NN algorithm finds the k nearest neighbours and predicts the output $\hat{y} = \bar{y}$, where the average is done over the identified k neighbours
- While k -NN method is super-easy to implement, it can be susceptible to noise, as each & every distance need to be calculated



PRACTICE EXAMPLE OF k -NN

Example: Consider the following dataset.

Height (h)	1.5	1.7	1.6	1.5	1.4	1.6	1.4	1.9	2	1.4	1.8	1.5
Weight (w)	71.1	103.3	26.4	27.8	21.8	94.9	90	98.3	108.1	91.9	61.5	90.2
BMI ($y = w/h^2$)	31.6	35.74	10.31	12.36	11.12	37.07	45.92	27.23	27.23	46.89	18.98	40.09

If we use K-NN for predicting BMI from height and weight assuming the true model ($y = w/h^2$) is unknown, find BMI for a person with height 1.3m and weight 32kg. Use K-NN with 4 neighbours and Euclidean distance as the metric for choosing neighbours.

PRACTICE EXAMPLE OF k -NN

- Euclidian distance of point (1.3,32) from all points are-
- $d = [39.1, 71.3, 5.61, 4.2, 10.2, 62.9, 58, 66.3, 76.1, 59.9, 29.5, 58.2]$
- The 4 neighbours would be samples 4,3,5 and 11
- Now the predicted BMI for the given test sample would be the average BMI of the 4 neighbours
- i.e. $\hat{y} = 13.1924$
- True BMI = 18.94
- Squared error prediction = 33.035

OPTIMAL VALUE OF K

- No structured way to find best k
 - Trial and error approach
 - Compare the models based on cross-validation performance
- Small value of k – noise will have high impact on the predictions, high SSE
- High value of k – Computationally more expensive
- A general practice: choose $k = \sqrt{n}$ where n is the number of samples in the training set

DISTANCE METRICS

- Continuous variables
 - Euclidean distance: $\sqrt{\sum (x^i - x_{test})^2}$
 - Manhattan distance: $\sum |x^i - x_{test}|$
- Strings of equal length
 - Hamming distance: Number of positions in which two strings differ. Example: Hamming distance between 'A23D' and 'A13E' is 2
- Strings of unequal length (eg: categories like lion, dog, cheetah etc.)
 - Reformulate features using
 - Integer encoding

Each category is given a number, like lion – 1, dog -2, cheetah – 3 etc.
 - One-hot encoding

The concerned feature is converted into multiple binary features

For the above example, if we keep a separate binary variable for lion, dog, and cheetah

k -NEAREST NEIGHBORS

CLASSIFICATION

CLASSIFICATION USING k -NN

- The procedure of Classification using k -NN is quite similar to Functional Approximation using k -NN
- However, to predict an output for $X(\text{new})$ a majority vote of k neighbours is used
- It is useful for binary and multiclass classification problems
- As the number of data points are increased, using k -NN becomes computationally hard
- Odd numbers preferred for k in case of binary classification

EXAMPLE

- **Question:** Find the species for a new sample with $(x_1, x_2, x_3, x_4) = (6.3, 2.5, 4.9, 1.5)$
- Let $k = 3$
- Nearest neighbors based on Euclidean distance

Sl No.	Nearest training data samples	Distance	Class Label
1	(6.8, 2.8, 4.8, 1.4)	0.6	1
2	(5.7, 2.8, 4.5, 1.3)	0.806	1
3	(5.6, 3, 4.5, 1.5)	0.95	1

- Predicted class: 1

Dataset

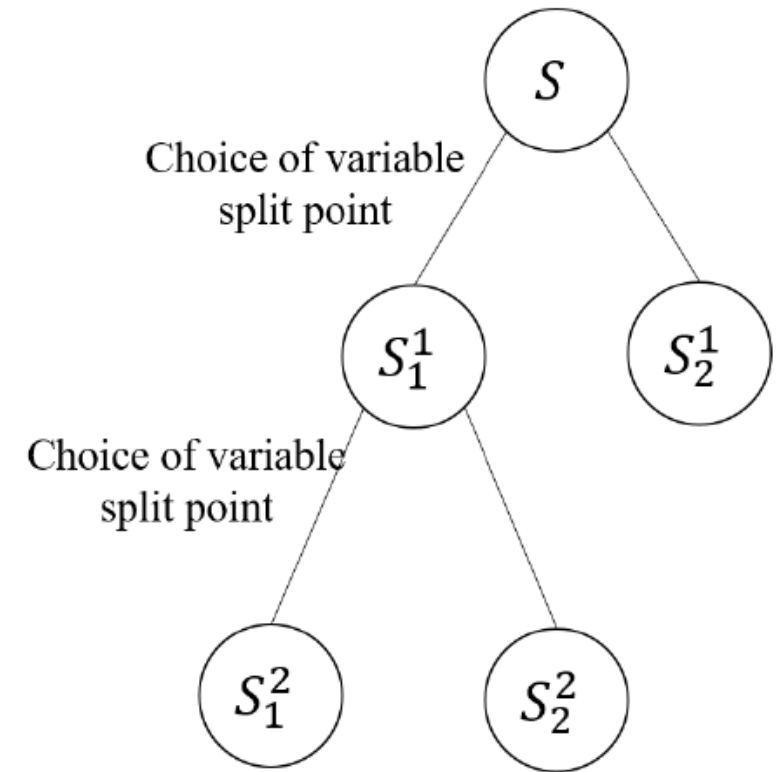
Sepal length (x1)	Sepal width (x2)	Petal length (x3)	Petal width (x4)	Species Class (y)
7.4	2.8	6.1	3	2
5.5	2.4	3.8	1.9	1
5.6	3	4.5	1.5	1
4.8	3	1.4	0.1	0
5	3.4	1.5	0.2	0
6.2	3.4	5.4	2.3	2
7.9	3.8	6.4	2	2
6.4	2.8	5.6	2.2	2
5.4	3.9	1.7	0.4	0
4.9	3.6	1.4	0.1	0
5.7	2.8	4.5	1.3	1
5.1	3.5	1.4	0.3	0
6.5	3	5.8	2.2	2
6.8	2.8	4.8	1.4	1
5.5	4.2	1.4	0.2	0
5.8	2.7	5.1	1.9	2
4.8	3	1.4	0.3	0

DECISION TREES

REGRESSION

DECISION TREES

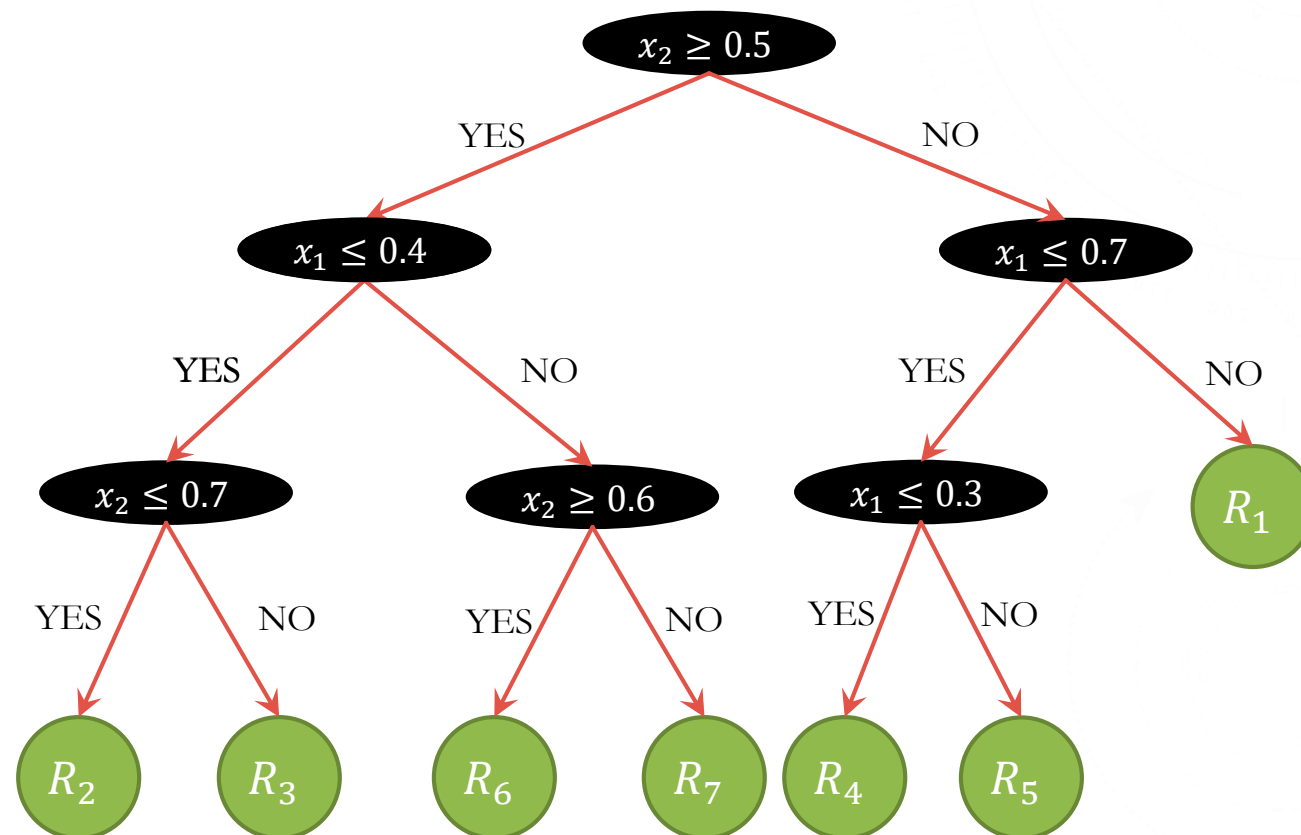
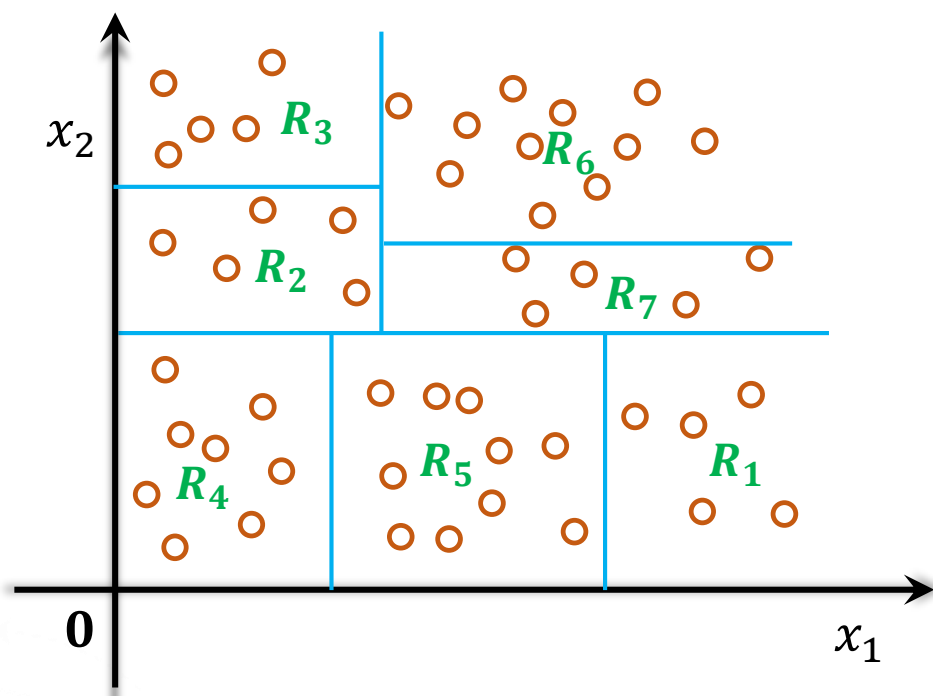
- Mimic the manner in which decisions are made by most of us
- Every decision point is represented as a node
- At each node, one variable is chosen from all the input variables and a decision is made based on the value of the variable
- Can be used for function approximation and classification
- Choice of variable and splitting value at each node have to be chosen based on our objective function:
 - Minimum prediction error for function approximation
 - Minimum classification error for classification



DECISION TREES

Input features: x_1 and x_2

Output feature: y



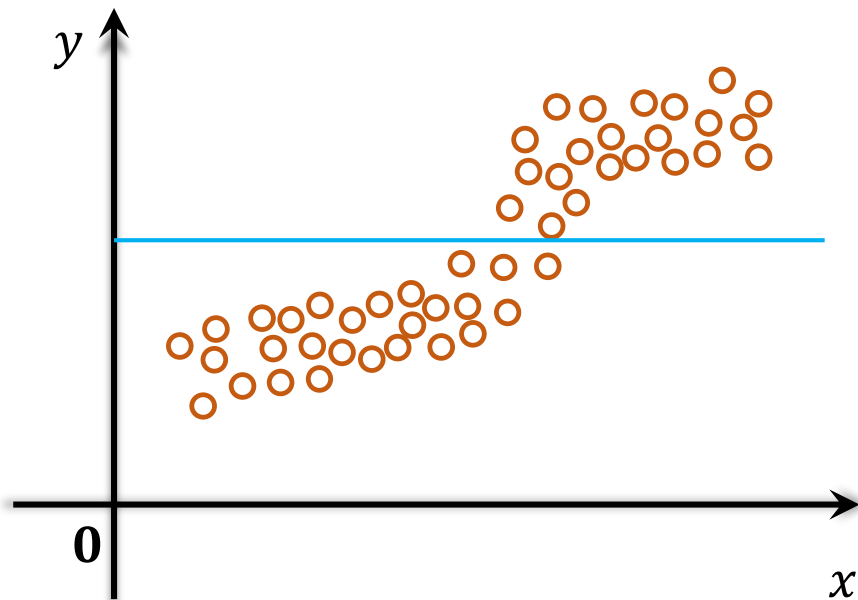
DECISION TREES

- Each end-node has a subset of samples – given data is split into multiple subsets
- Partitions happen only in the input space
- Outputs are used in computing the predicted value for each of the regions that is generated
- Output for a new test sample depends on the end-node to which the new point would fall
- One can keep growing the tree to a large number of nodes
 - Although performance on training set improves, may result in over-fitting
 - Size of tree might become computationally unwieldy
- Make choices that limit tree complexity
 - Tree depth
 - Number of end-nodes
 - Minimum size of each node (cardinality)
- Growing a tree is continued until a certain accuracy requirement is met and/or until the complexity becomes too much

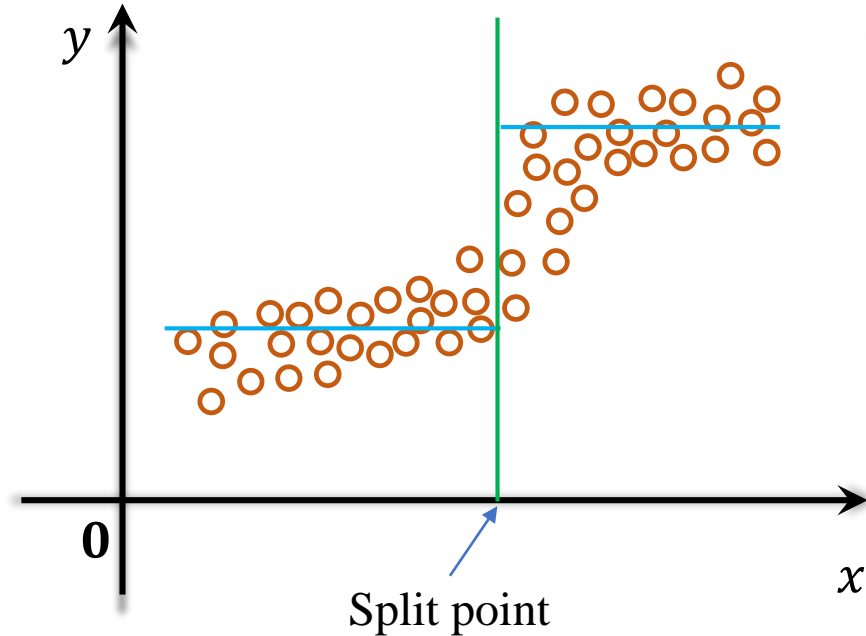
DECISION TREES FOR REGRESSION

ONE INPUT FEATURE

Without decision tree



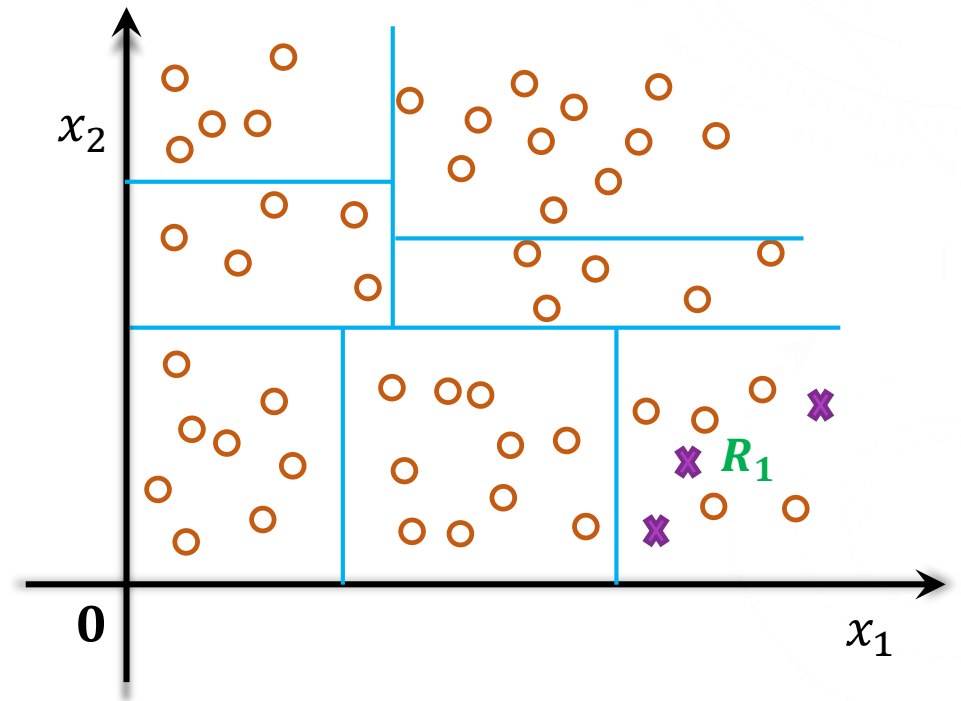
With decision tree (one node)



DECISION TREES FOR REGRESSION

GENERAL CASE

- ❑ For regression, to predict the output for a new test sample ' x_{new} ',
 - ❑ identify the set (or rectangle) R_j corresponding to sample x_{new} and
 - ❑ \hat{y}_{new} would be the average of the output values of all samples in set R_j
- ❑ Example: consider the new point marked by 'x'
$$\hat{y}_{new} = \text{mean} \left(y(x^i) \right) \text{ such that } x^i \in R_1$$
 - ❑ Same prediction for any point within the rectangle R_1
- ❑ How do we find the optimal split point?
 - ❑ Based on minimum squared error, $\sum (y^i - \hat{y}^i)^2$



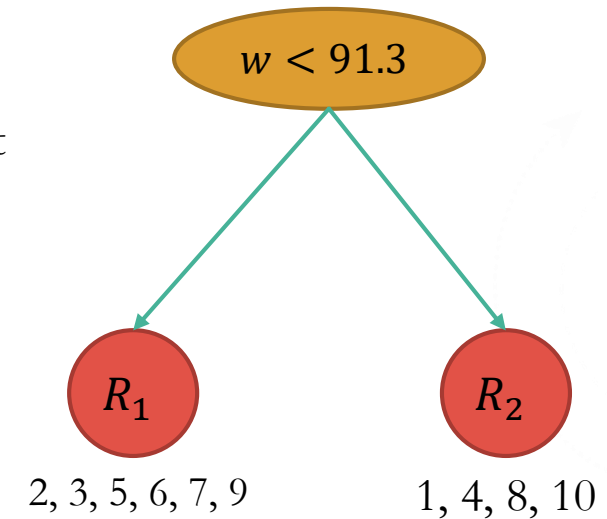
DECISION TREES FOR REGRESSION

- Consider the data relating weight to BMI
- If a single node decision tree is used, let us see how to find the best split point
- If 's' is split point, let the decision be $w < s$.

Sample No	1	2	3	4	5	6	7	8	9	10
Weight (w)	91.3	67.6	71.1	103.3	26.4	27.8	21.8	94.9	90	98.3
BMI (y)	40.58	23.39	27.77	45.91	13.47	10.86	11.12	26.29	22.5	50.15

Split point (w)	Samples in R_1	Samples in R_2	\bar{y}_{R_1}	\bar{y}_{R_2}	SSE
91.3	2,3,5,6,7,9	1,4,8,10	18.185	40.7325	587.49
67.6	5,6,7	1,2,3,4,8,9,10	11.8167	33.7986	792.9
71.1	2,5,6,7	1,3,4,8,9,10	14.71	35.5333	766.96
103.3	1,2,3,5,6,7,8,9,10	4	25.1256	45.91	1418.83
26.4	7	1,2,3,4,5,6,8,9,10	11.12	28.9911	1520.18
27.8	5,7	1,2,3,4,6,8,9,10	12.295	30.9313	1251.9
21.8	-	1,2,3,4,5,6,7,8,9,10	-	27.204	1807.6
94.9	1,2,3,5,6,7,9	4,8,10	21.3843	40.7833	1017.3
90	2,3,5,6,7	1,4,8,9,10	17.322	37.086	831.0856
98.3	1,2,3,5,6,7,8,9	4,10	21.9975	48.03	723.3192

Best split point



DECISION TREES

CLASSIFICATION

DECISION TREES FOR CLASSIFICATION

- ❑ Similar to function approximation, except for
 - how we find the best choice of split variable and split point
 - Minimization of error for function approximation while other metrics like maximization of Gini split index for classification
 - how we make predictions
 - Average of output values for function approximation while majority voting for classification
- ❑ Consider j^{th} node at the k^{th} level of the tree. Let the set of points in the node be $\{S_j^k\}$. Assume that n samples are present in set $\{S_j^k\}$. Let there be p classes in the data.
- ❑ $f_i(j, k)$ – fraction of data points in $\{S_j^k\}$ that belong to class i
- ❑ Gini impurity

$$GI(j, k) = 1 - \sum_{i=1}^p f_i(j, k)^2$$

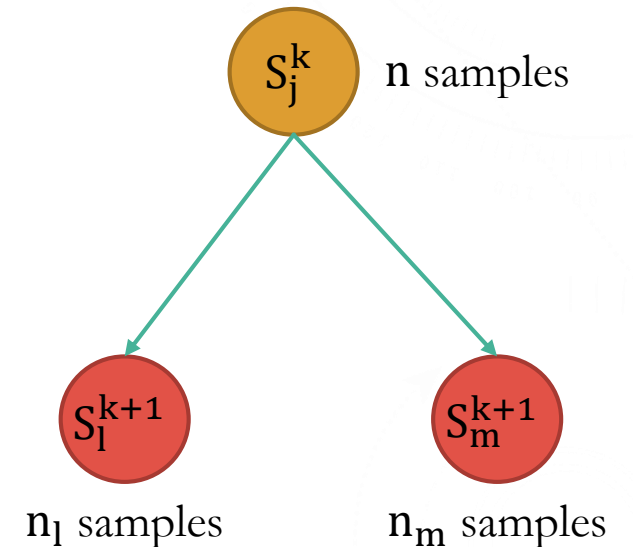
- ❑ When a node has data from only one class, $GI = 0 \rightarrow$ pure node

DECISION TREES FOR CLASSIFICATION

- ❑ When choosing the variable to branch on and split point, aim is to get pure or close to pure child nodes
- ❑ Let child nodes be S_l^{k+1} (with n_l samples) and S_m^{k+1} (with n_m samples)
- ❑ Gini split index

$$GSI = GI(j, k) - \frac{n_l}{n} GI(l, k + 1) - \frac{n_m}{n} GI(m, k + 1)$$

- ❑ If both child nodes are pure, $GSI = GI(j, k)$
- ❑ For any other split, GSI would be less than $GI(j, k)$
- ❑ Since the best case scenario is to get pure nodes on splitting, variable and split point that maximizes GSI is selected



$$n = n_l + n_m$$

DECISION TREES FOR CLASSIFICATION

- Consider the data relating hours studied to result

Sample No.	1	2	3	4	5	6	7	8
Hours studied	10.5	2.5	14	8.2	10.4	5	6.7	14.7
Pass (1)/Fail(0)	1	0	1	1	1	0	0	1

- If a single node decision tree is used, let us see how to find the best split point
- Gini impurity (GI) for the given data:

$$GI = 1 - f_0^2 - f_1^2 = 1 - \left(\frac{3}{8}\right)^2 - \left(\frac{5}{8}\right)^2 = 0.4688$$

DECISION TREES FOR CLASSIFICATION

Sample No.	1	2	3	4	5	6	7	8
Hours studied	10.5	2.5	14	8.2	10.4	5	6.7	14.7
Pass (1)/Fail(0)	1	0	1	1	1	0	0	1

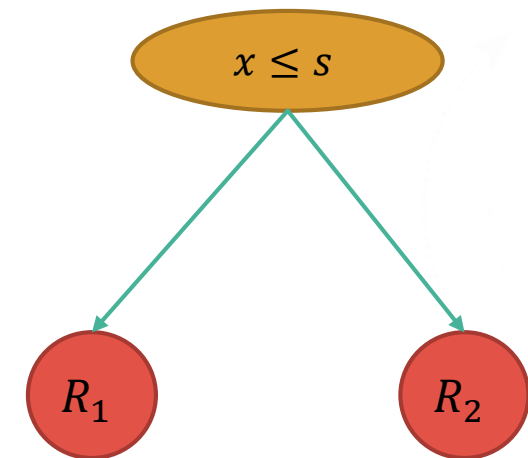
- Let 's' be a split point such that set R_1 contains all points such that $x \leq s$ and set R_2 contains all points such that $x > s$
- Let $x = 10.4$ be the split point, then R_1 contains samples $\{2, 4, 5, 6, 7\}$, and R_2 contains samples $\{1, 3, 8\}$
- Since 3 out of 5 samples in R_1 have label 0 and the remaining 2 have label 1.

$$f_0 \text{ for } R_1 = 0.6$$

$$f_1 \text{ for } R_1 = 0.4$$

$$GI \text{ for } R_1 = 1 - 0.6^2 - 0.4^2 = 0.48$$

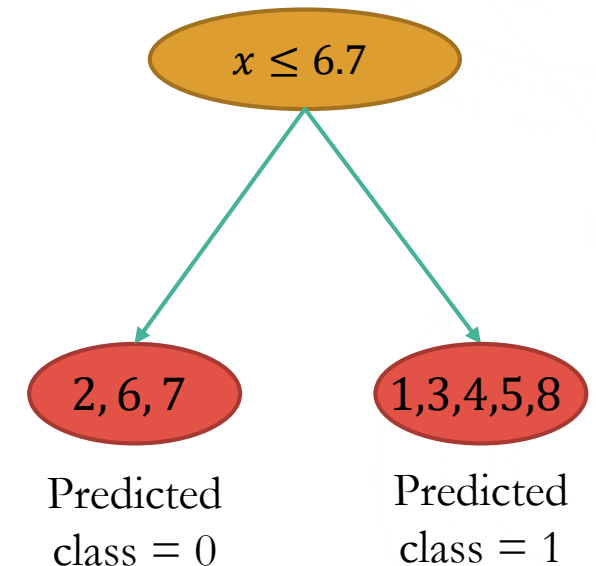
- R_2 is a pure node, implies $GI = 0$ for R_2
- GSI for split point $10.4 = 0.4688 - \left(\frac{5}{8}\right) 0.48 - 0 = 0.1688$



DECISION TREES FOR CLASSIFICATION

Sample No.	1	2	3	4	5	6	7	8
Hours studied	10.5	2.5	14	8.2	10.4	5	6.7	14.7
Pass (1)/Fail(0)	1	0	1	1	1	0	0	1

Split point (s)	Samples in R1	Samples in R2	GI for R1	GI for R2	GSI
10.5	1,2,4,5,6,7	3,8	0.5	0	0.094
2.5	2	1,3,4,5,6,7,8	0	0.408	0.112
14	1,2,3,4,5,6,7	8	0.49	0	0.04
8.2	2,4,6,7	1,3,5,8	0.375	0	0.28
10.4	2,4,5,6,7	1,3,8	0.48	0	0.169
5	2,6	1,3,4,5,7,8	0	0.28	0.26
Best split point 6.7	2,6,7	1,3,4,5,8	0	0	0.469
14.7	1,2,3,4,5,6,7,8	-	0.469	-	0

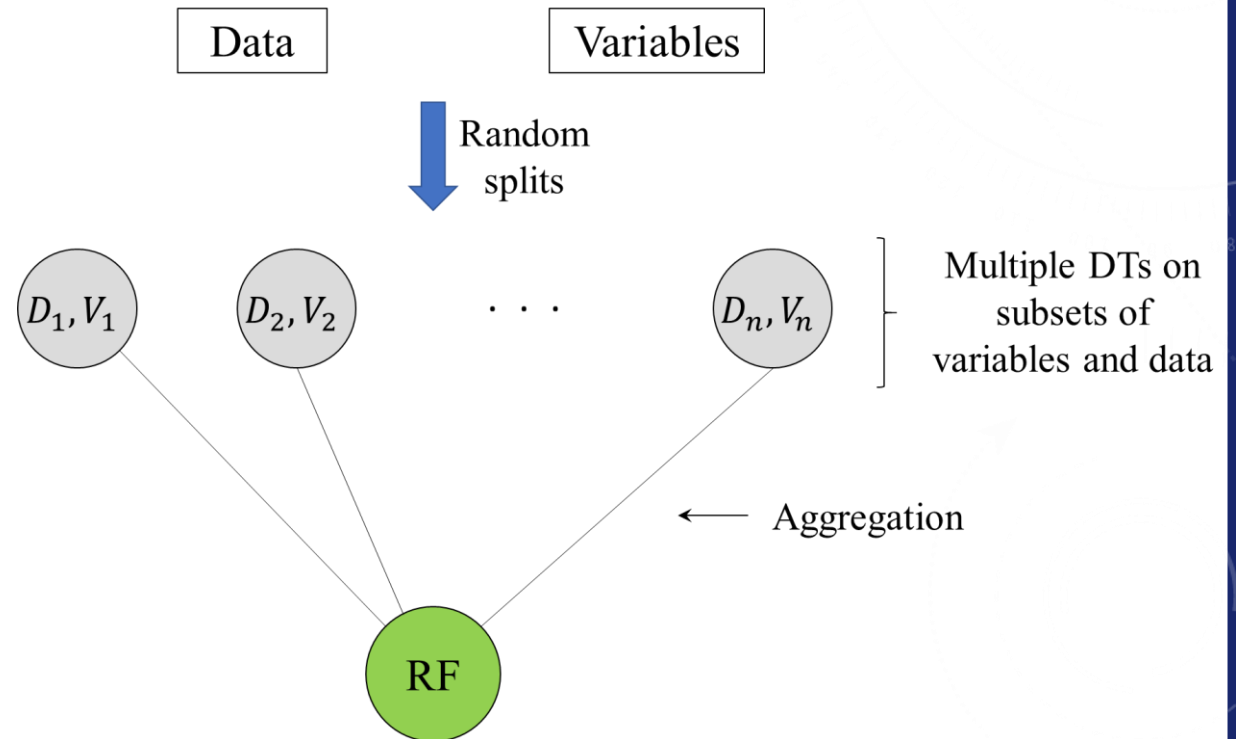


RANDOM FOREST

REGRESSION AND CLASSIFICATION

RANDOM FORESTS

- ❑ Simple extension of decision trees
- ❑ Forest comprise of many trees; equivalently, random forests consists of many decision trees
- ❑ Decision tree may give considerably different results for minor changes in the data or construction procedure
- ❑ Random forests help to avoid this by building multiple trees
- ❑ Each tree is realized by choosing a subset of datapoints or variables and building a decision for each sub-selected data matrix



RANDOM FORESTS

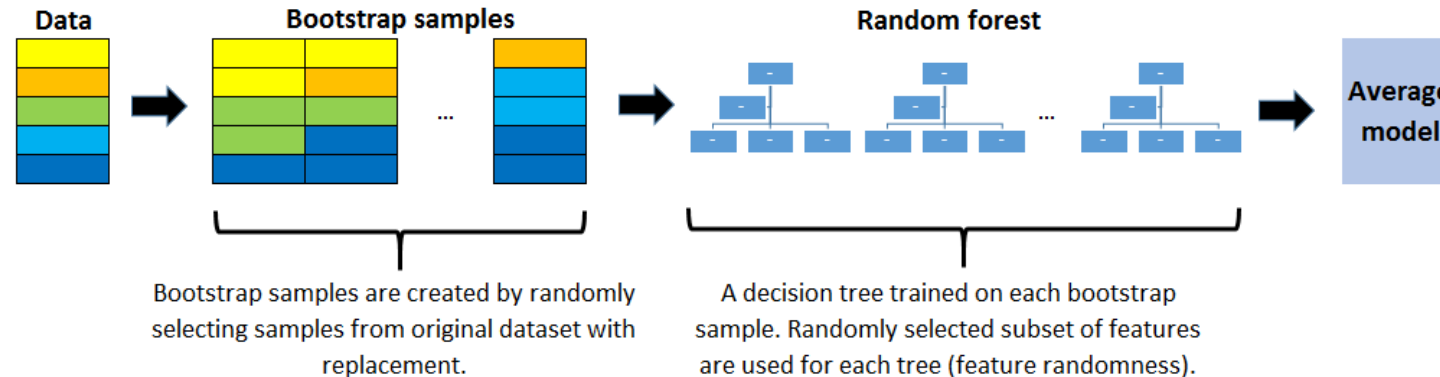
○ Predictions

- Average of predictions from multiple trees for function approximation
- Majority voting (or other metrics) for classification

○ Advantages of random forest

Stochasticity is included in various forms

- Splitting on the basis of random subsets of samples
- Splitting on the basis of random features
- Bagging (Bootstrap aggregating) decision trees – random samples with replacement



Other approaches for introducing stochasticity: Boosting decision trees – trees added sequentially

PERFORMANCE MEASURES FOR CLASSIFICATION

CONFUSION MATRIX

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity/Recall $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$

- TP – Correct identification of positive labels
- TN – Correct identification of negative labels
- FP – Incorrect identification of positive labels
- FN – Incorrect identification of negative labels

- Accuracy: Overall effectiveness of a classifier
- Total number of actual positive samples – TP+FN
- Total number of actual negative samples – TN+FP
- Sensitivity/Recall – Effectiveness of a classifier to identify positive labels

$$S_e = \frac{TP}{TP + FN}$$

- Specificity – Effectiveness of a classifier to identify negative labels

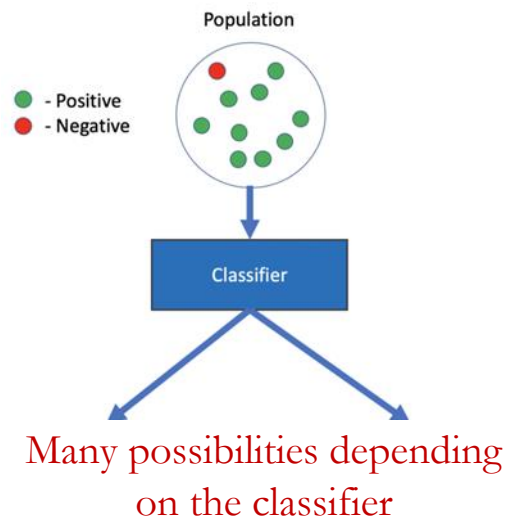
$$S_p = \frac{TN}{FP + TN}$$

- Accuracy, specificity and sensitivity – maximum value is 1
- Balanced accuracy = 0.5(Specificity+Sensitivity)
- Maximizing recall may adversely affect precision and vice versa

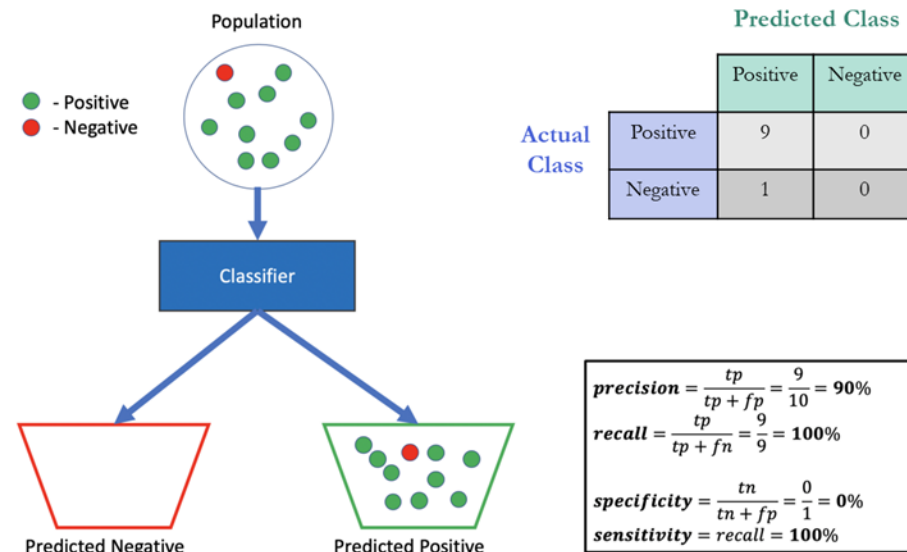
- F1-score $F1 - score = \frac{2(Precision \times Recall)}{Precision + Recall}$

PERFORMANCE MEASURES

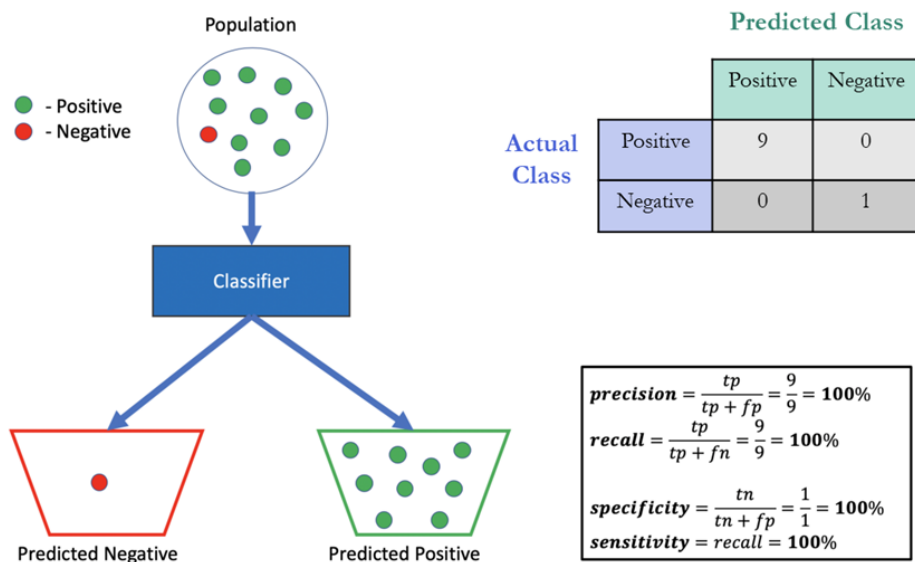
		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity/Recall $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$



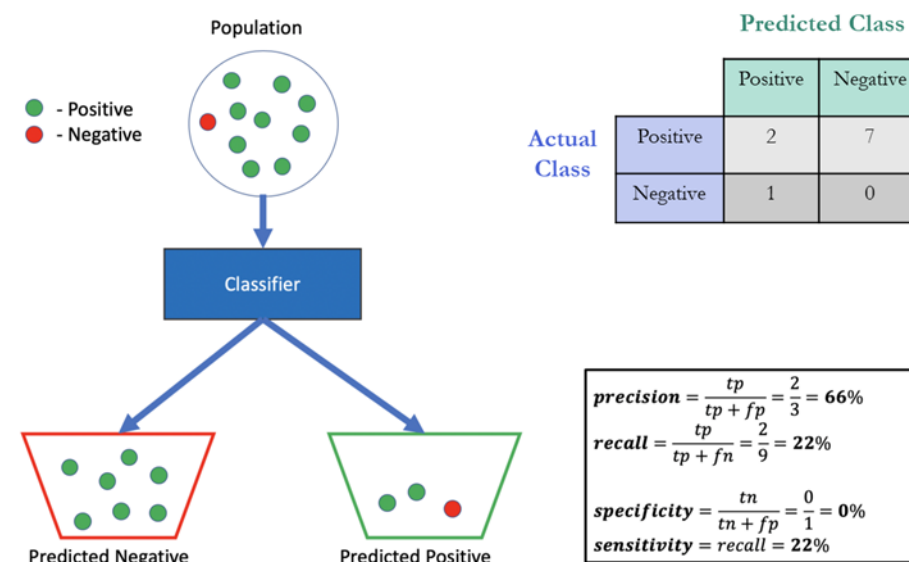
High precision and recall, low specificity



Best case scenario



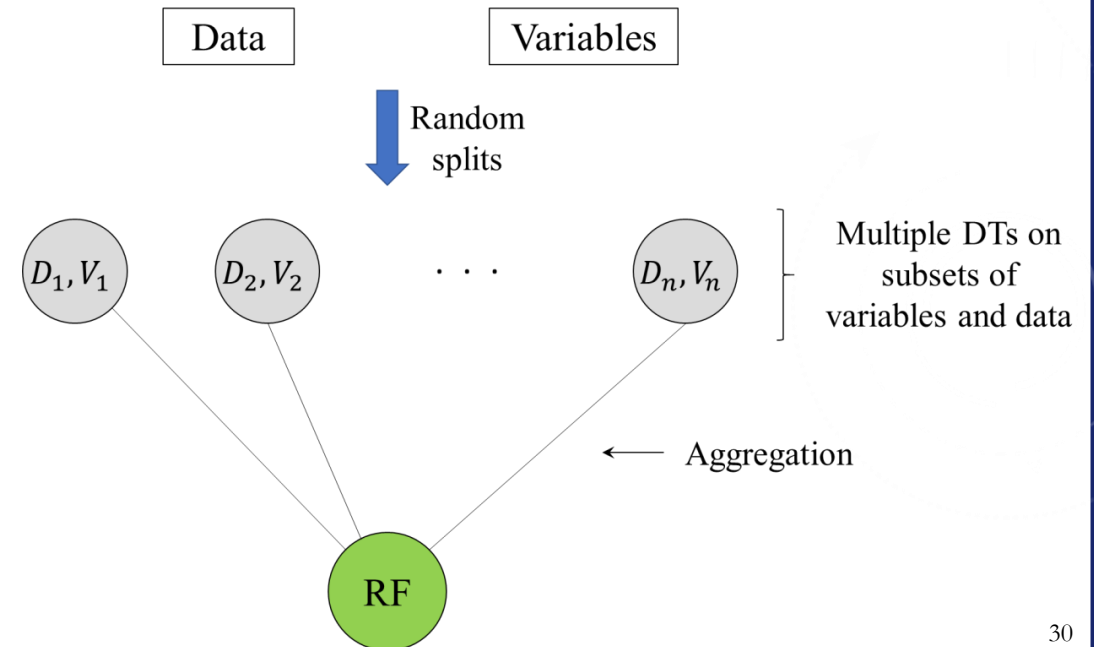
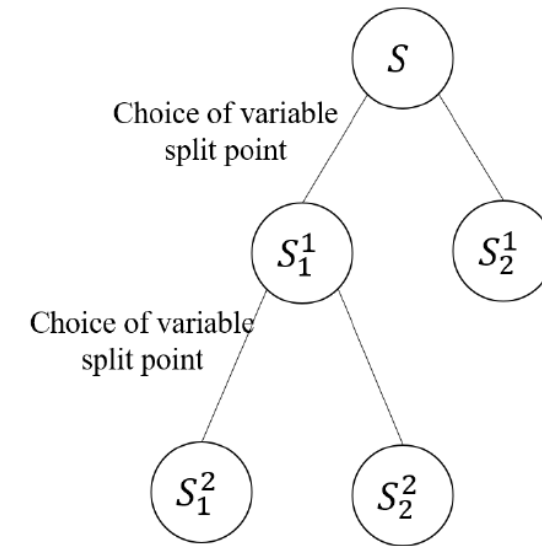
High precision, low recall and specificity



CONCLUSIONS

- ❑ Decision trees for both function approximation and classification
- ❑ Random forests for both function approximation and classification
- ❑ Performance measures for classification (using any algorithm)

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity/Recall $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$



```
        operation == "MIRROR_X":  
            mirror_mod.use_x = True  
            mirror_mod.use_y = False  
            mirror_mod.use_z = False  
        operation == "MIRROR_Y":  
            mirror_mod.use_x = False  
            mirror_mod.use_y = True  
            mirror_mod.use_z = False  
        operation == "MIRROR_Z":  
            mirror_mod.use_x = False  
            mirror_mod.use_y = False  
            mirror_mod.use_z = True
```

```
    #selection at the end -add  
    mirror_ob.select= 1  
    modifier_ob.select=1  
    context.scene.objects.active  
    one("Selected" + str(modifier_ob.name))  
    mirror_ob.select = 0  
    one = bpy.context.selected_objects[0]  
    data.objects[one.name].select  
    print("please select exactly one object")
```

WILLIAM CHAI

THANK YOU