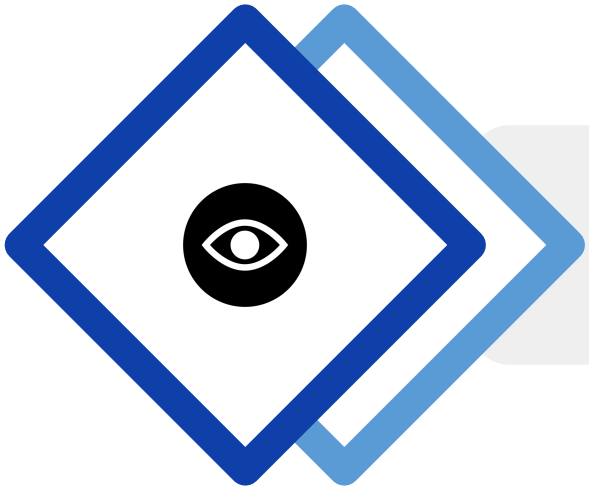
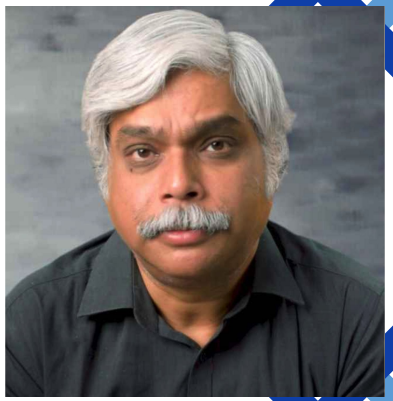


Decision Trees

Meet Your Speaker



Dr. Abhinanda Sarkar
Academic Director at Great Learning

- Alumnus - Indian Statistical Institute, Stanford University
- Faculty - MIT, Indian Institute of Management, Indian Institute of Science
- Experienced in applying probabilistic models, statistical analysis and machine learning to diverse areas
- Certified Master Black Belt in Lean Six Sigma and Design for Six Sigma in GE

Learning Objectives

By the end of this session, you should be able to:

- Summarize the working of decision trees for decision-making
- Compare different impurity measures and identify the optimal splitting criteria for decision-making
- Evaluate the performance of decision trees for classification using key metrics and identify the driving factors behind decision-making
- Experiment with different pruning techniques to improve the model performance and generalization
- Apply decision trees to real-world problems to derive actionable insights for informed decision-making

Agenda

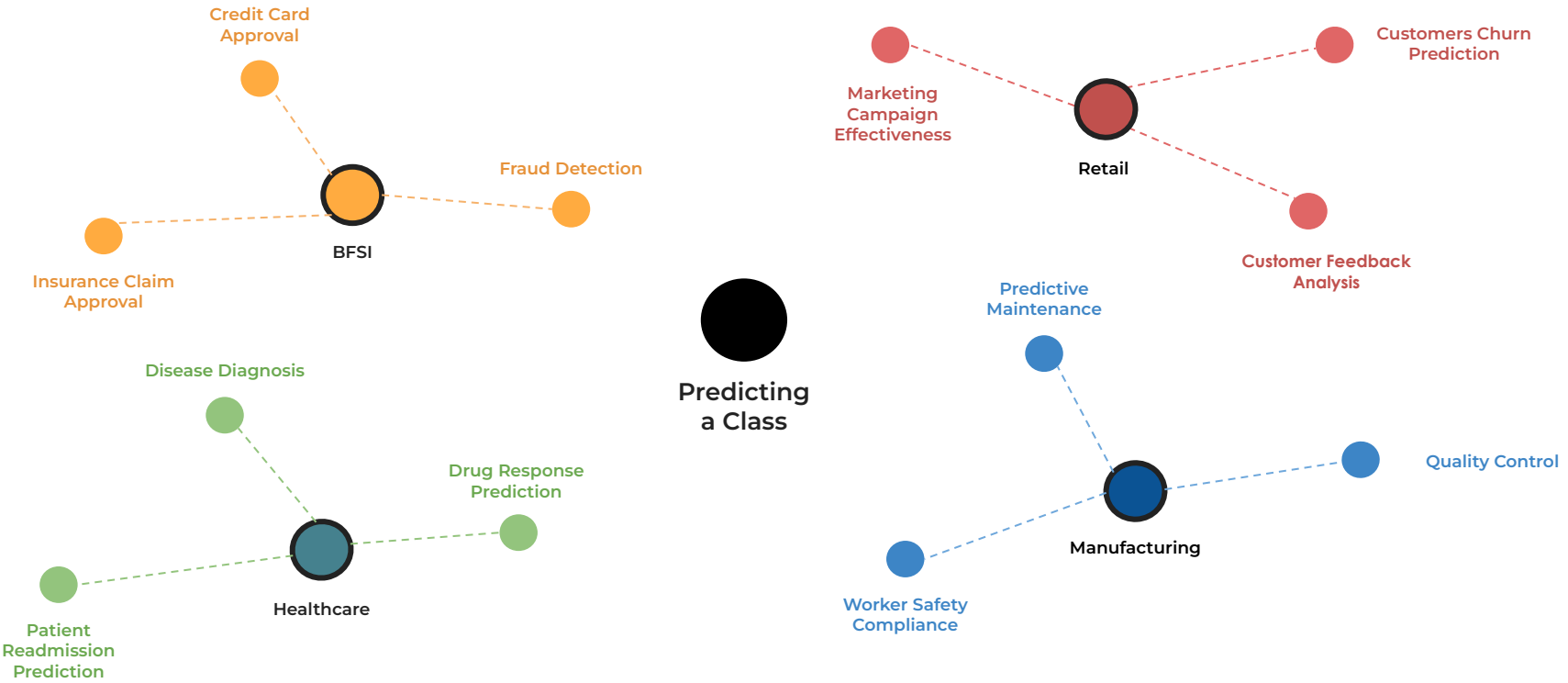
In this section, we'll discuss:

- Business Problems and Solution Space
- Overview of Decision Trees
- Impurity Measures and Splitting Criteria
- Evaluation Metrics for Classification
- Pruning Techniques
- Decision Trees for Regression

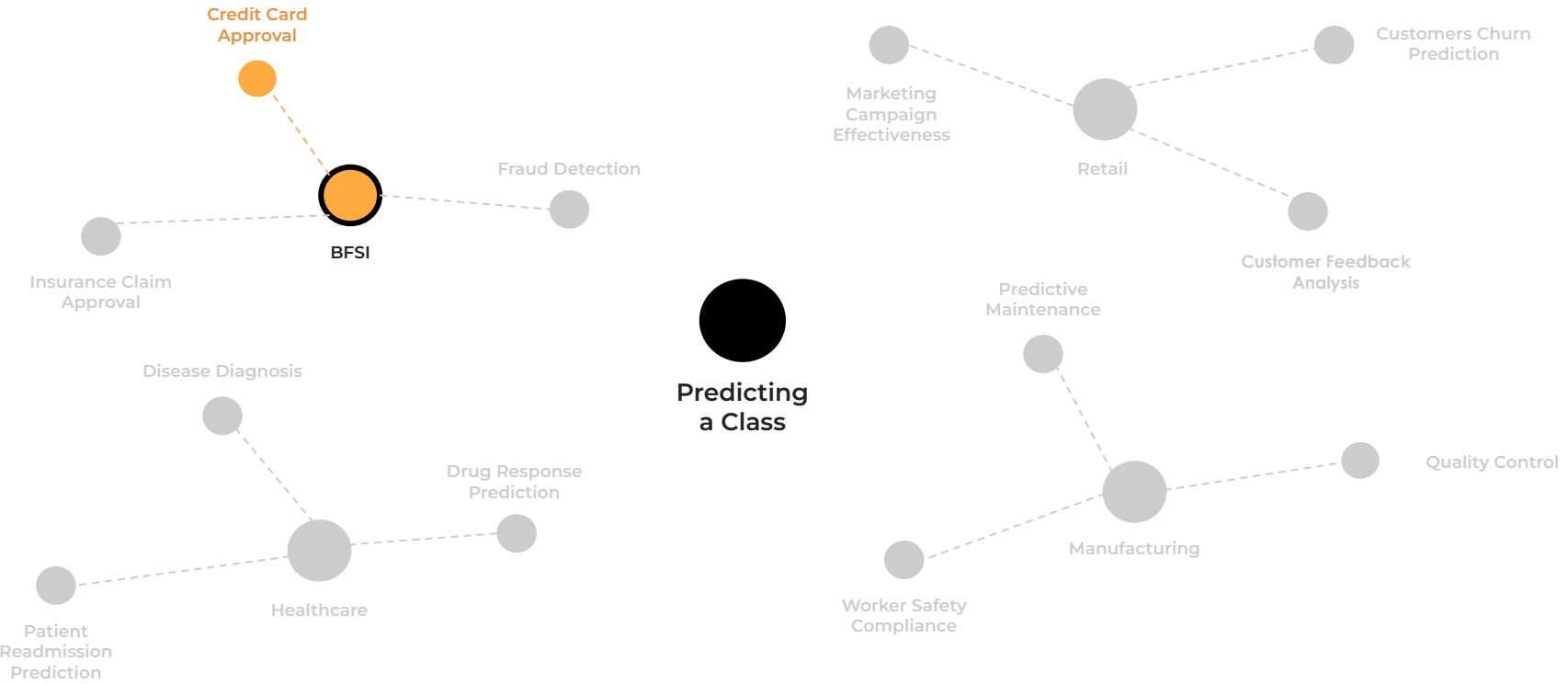
Common Business Questions

- How do we predict the likelihood of heart disease using patient data like cholesterol levels, blood pressure, and smoking habits?
- How can we assess the creditworthiness of credit card applicants based on their financial details, income, and demographic information?
- How can we predict which customers are likely to churn based on their purchase history and interaction with the company?
- How can we identify defective products on the production line based on sensor data and quality inspection results?

Problem Space

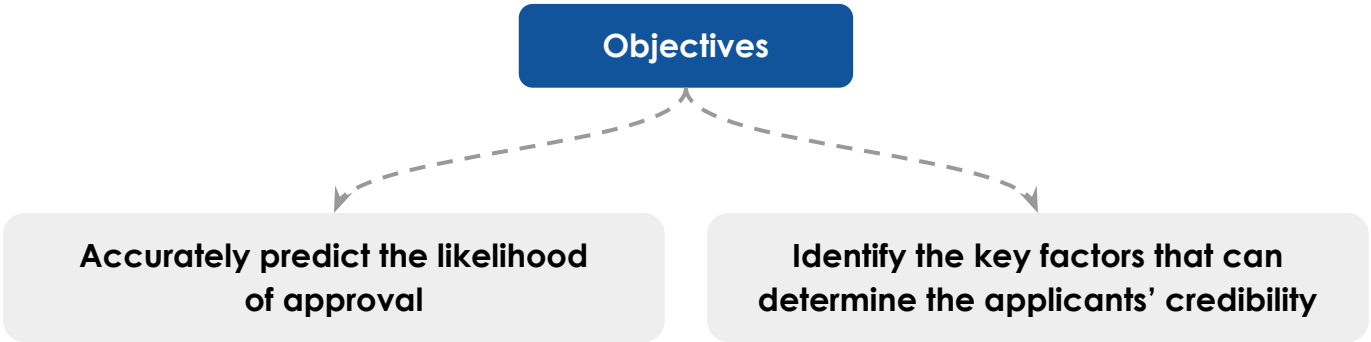


Problem Space

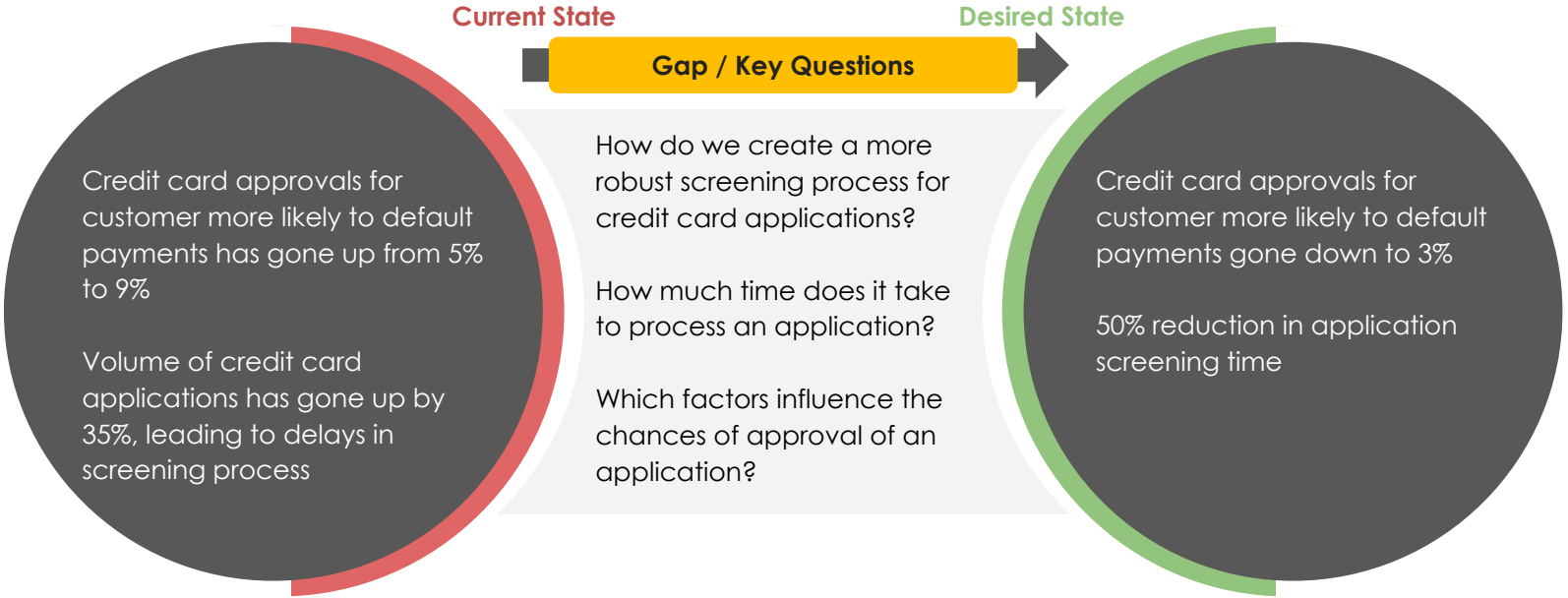


Problem Statement

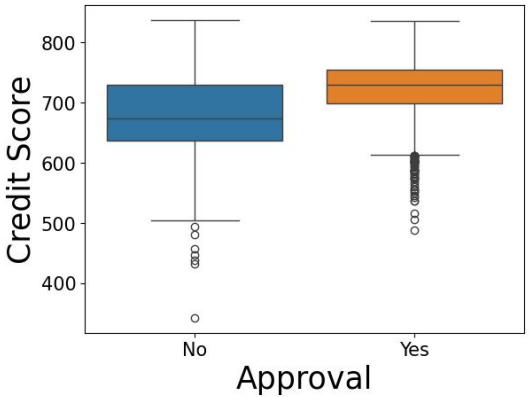
- Consider a bank offering credit cards to the applicants.
- Essential to accurately assess risk involved and approve credit to financially reliable applicants.
- Important to understand the customer attributes that drive eligibility for availing credit cards.

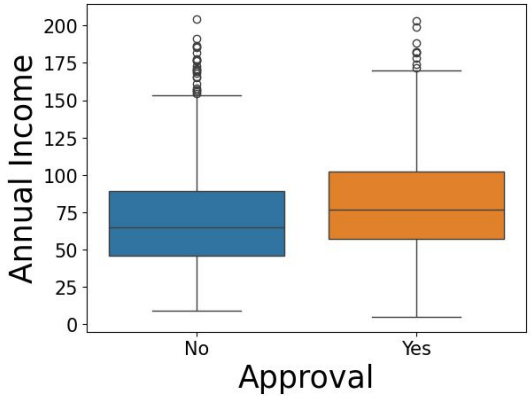


Credit Card Approval



Visualizing Relationships





Credit Score ↑ **Chances of Approval** ↑

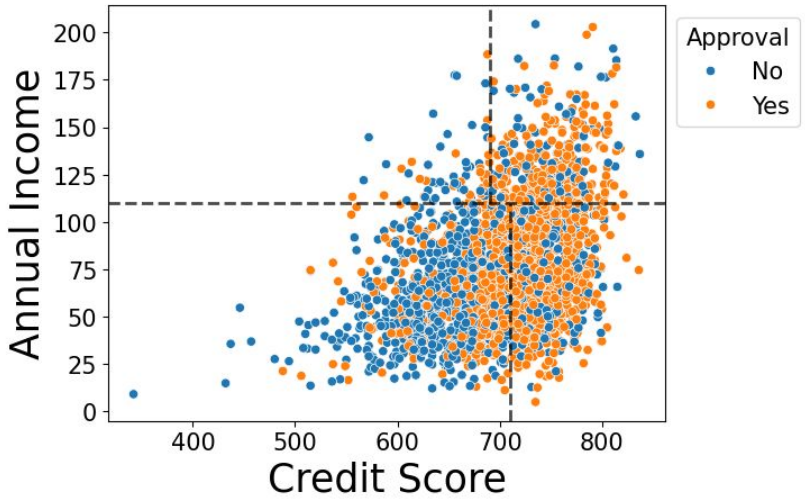
Annual Income ↑ **Chances of Approval** ↑

- Cannot observe the correlation between credit score and annual income here

Visualizing Relationships

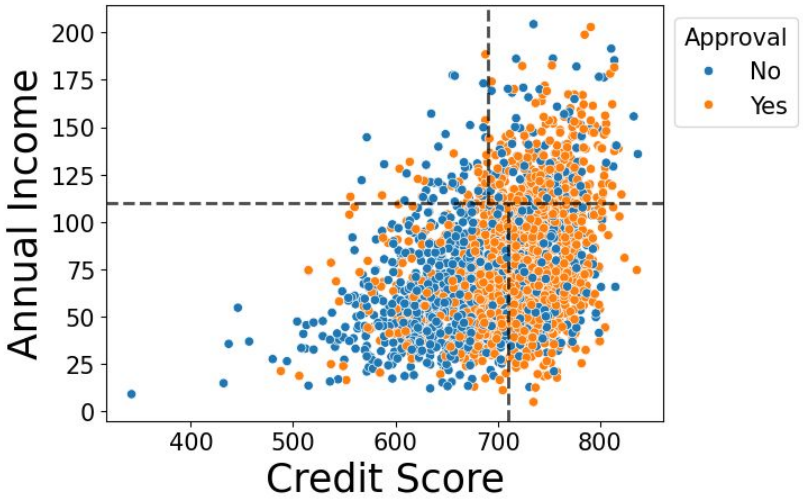
- Let's split the graph into two parts based on Annual Income
- Any applicant with **>$110k income** is more likely to get their application approved
- This alone doesn't give us a clear picture of when applications are likely to be approved

Visualizing Relationships



- Let's split the plot again based on Credit Score
- For **Annual Income >$110k**, if **Credit Score >690**, application is likely to get approved
- For **Annual Income <=$110k**, if **Credit Score >710**, application is likely to get approved

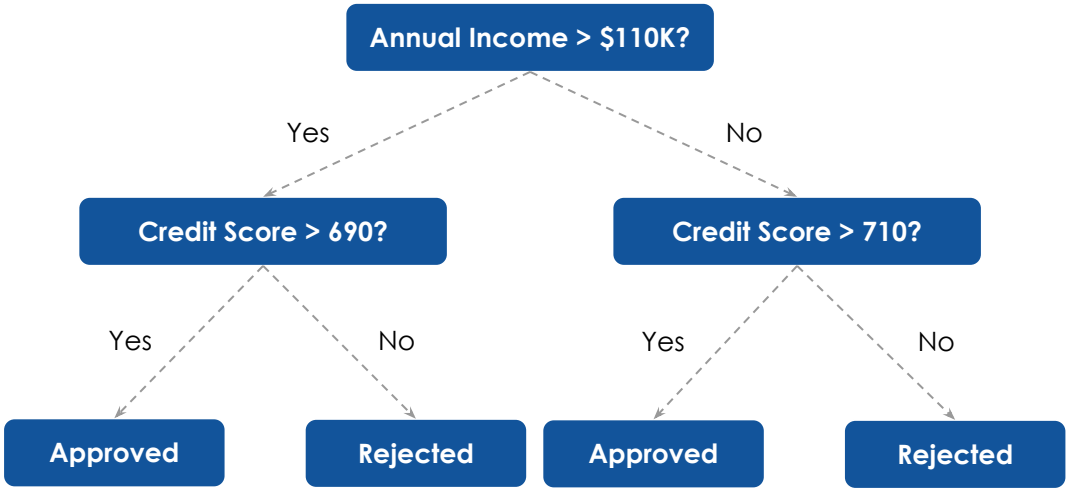
Visualizing Relationships



- Four different regions in the scatterplot
- Each contains a distribution of approvals and rejections

Decision Tree

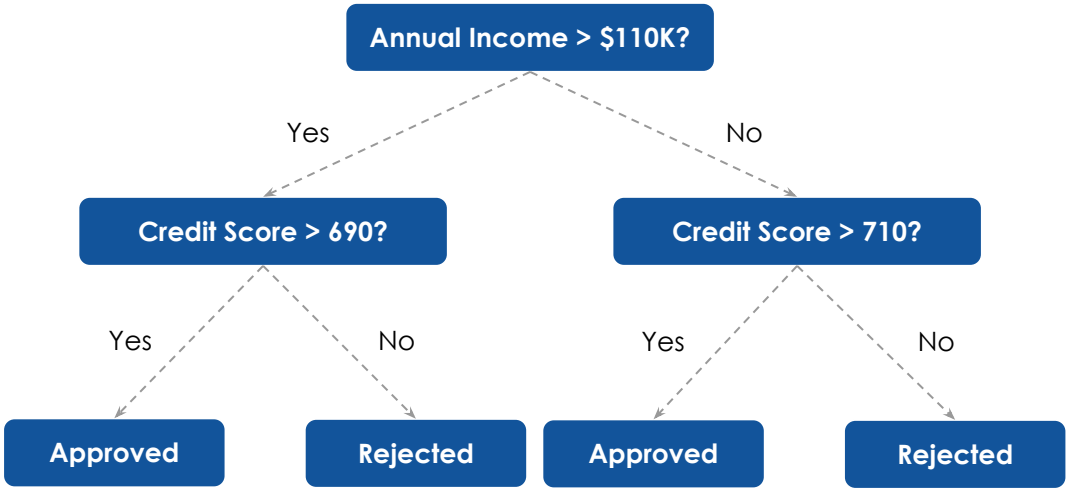
- Based on our observations, we can make a rough decision on application approval



This is the **mechanism** followed by **decision trees**

Decision Tree

- A flowchart-like structure for making decisions or predictions based on certain conditions



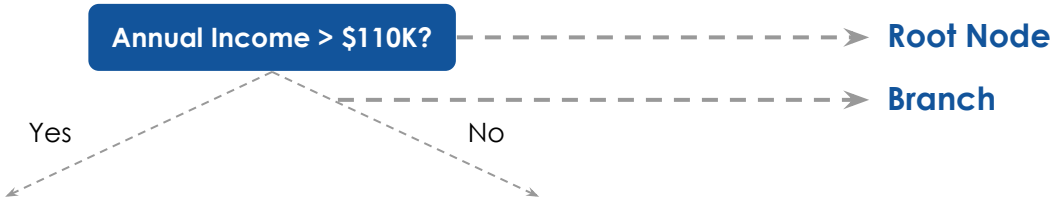
- The data is split into subsets based on the selected attributes
- The process is repeated recursively for each subset

Decision Tree

The topmost node in a decision tree

Represents the initial point for splitting the entire dataset

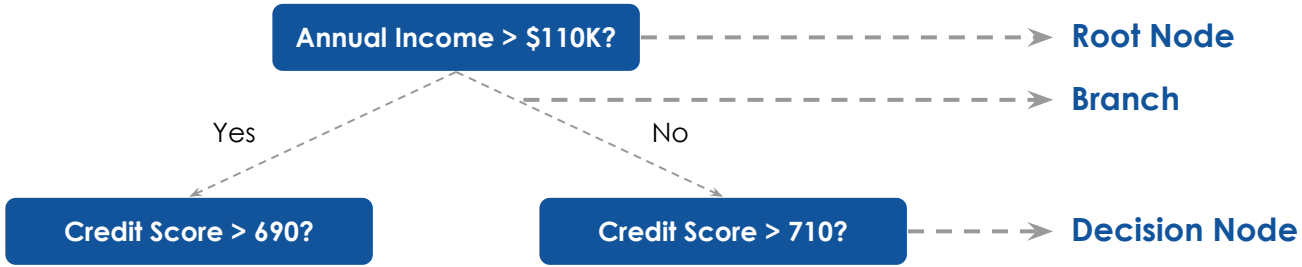
Decision Tree



Represent possible outcomes of a decision

Leads to further nodes

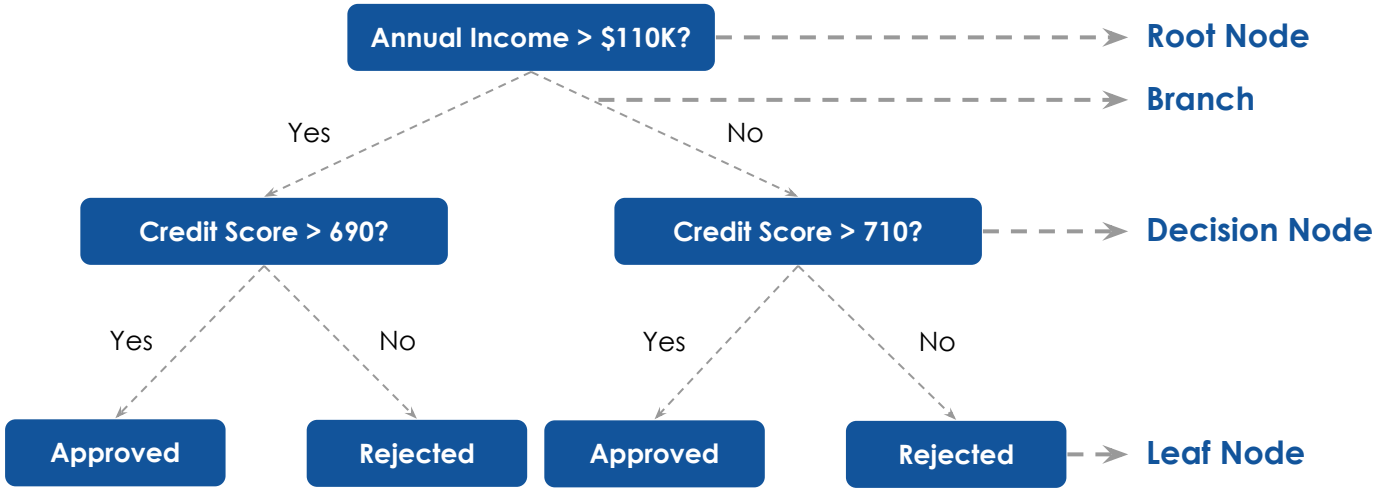
Decision Tree



Node obtained as a result of splitting a previous node

Data is further splitted at this node

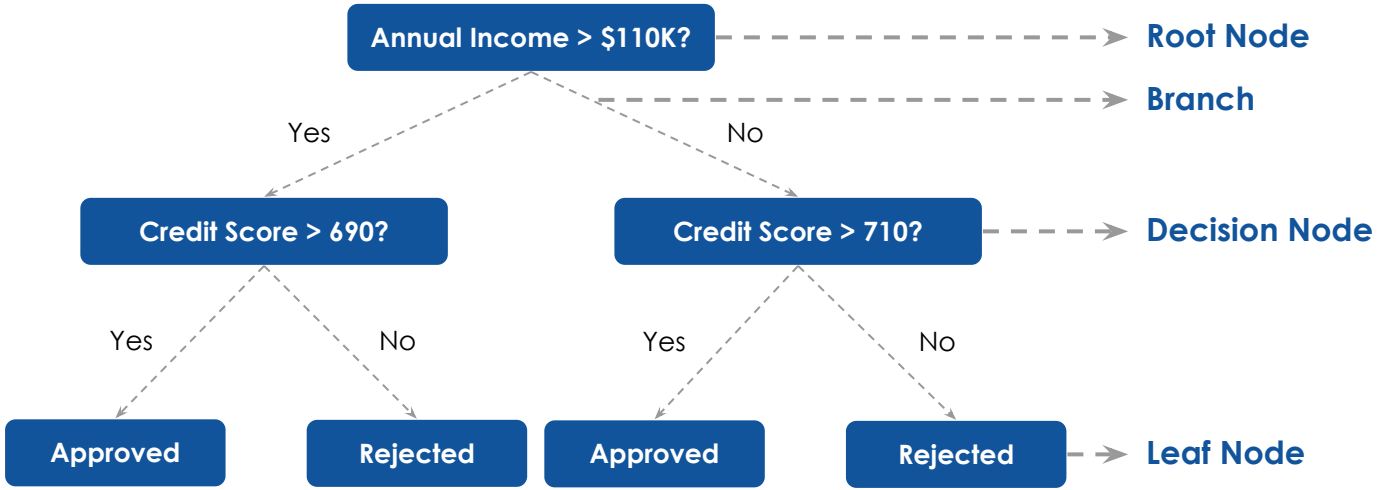
Decision Tree



Node where no further splits occur

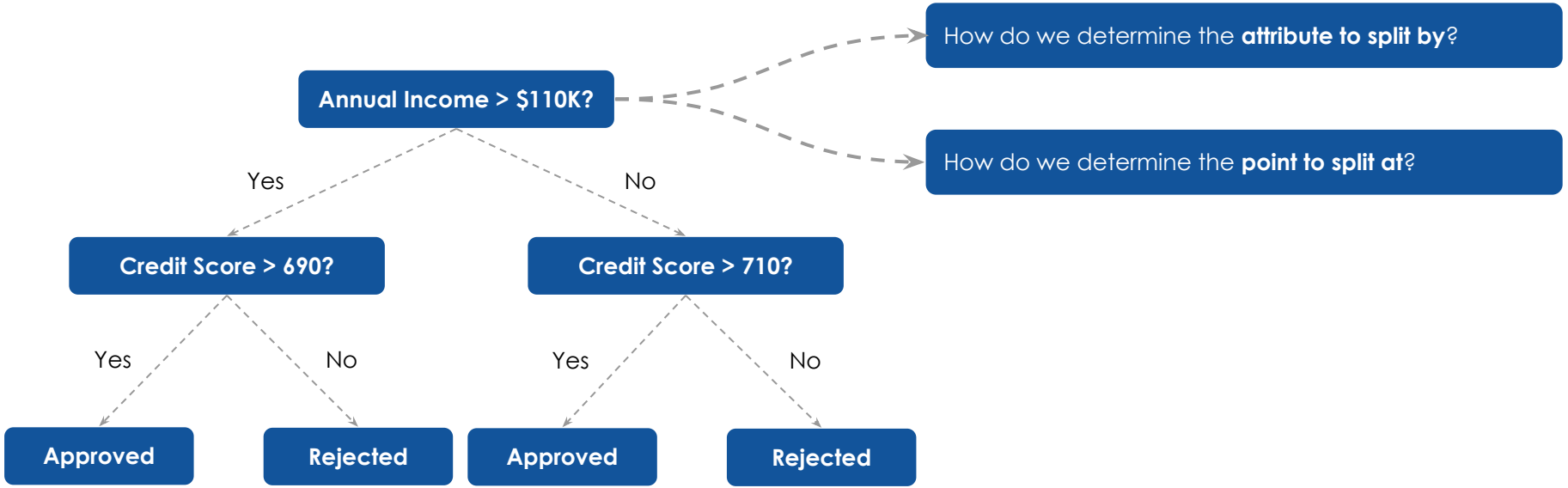
Provides the final prediction

Decision Tree

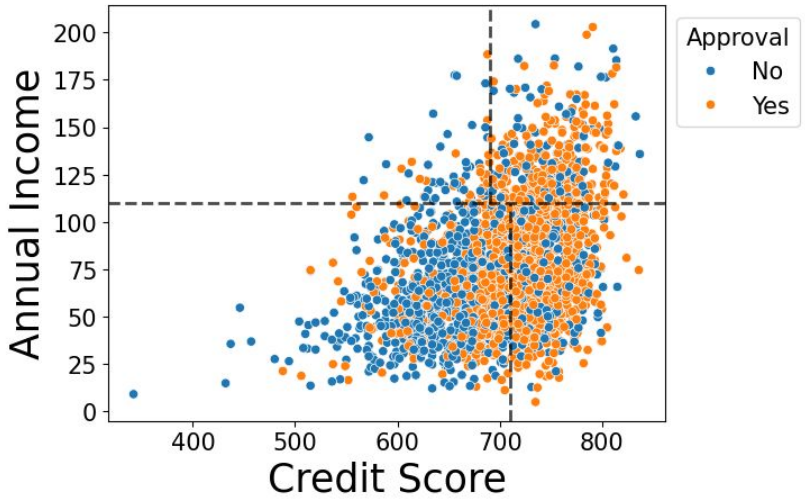


- At the **leaf node**, find the **proportion of each class**
- The **class** with the **higher proportion** is the **prediction**

Decision Tree

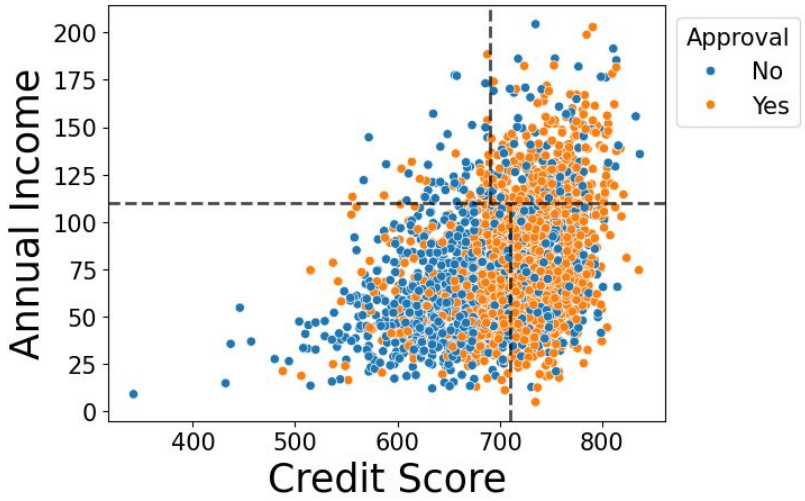


Impurity Measures



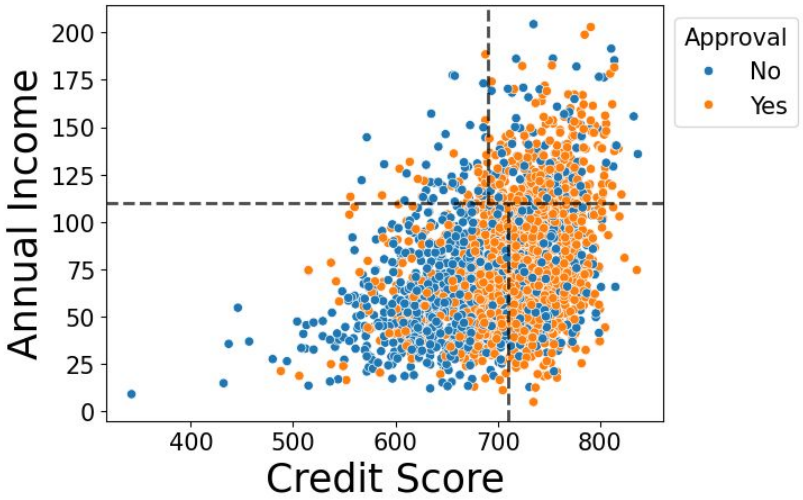
- In the lower right region, roughly 80% of applications are accepted
- Most of the applications are similar in nature - approvals
- Low ‘impurity’

Impurity Measures



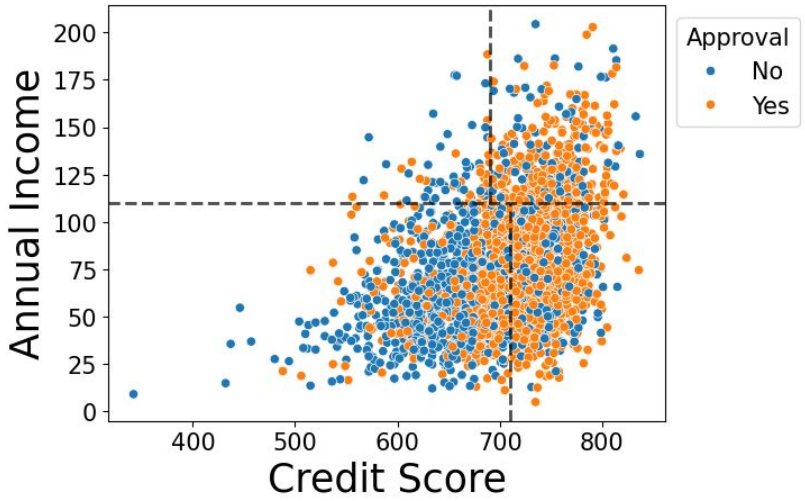
- In the lower left region, roughly 60% applications are rejected
- 60% of the applications are similar in nature - rejections
- High ‘impurity’

Impurity Measures



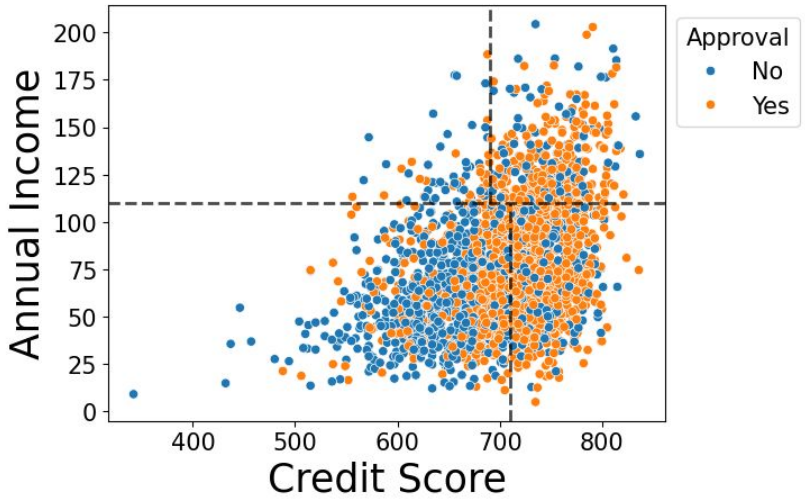
- In the upper left region, roughly 80% of the applications are rejected
- Most of the applications are similar in nature - rejections
- Low ‘impurity’

Impurity Measures



- In the upper right region, roughly 60% of the applications are approved
- 60% of the applications are similar in nature - approvals
- High ‘impurity’

Impurity Measures



- ‘Impurity’ determines the uncertainty of making a decision - approve or reject
- Lower the impurity, easier the decision
- Make splits to reduce impurity at each step
- Build the tree in the order of decreasing impurity

Impurity Measures

Impurity measures measure the **homogeneity of the target variable** within a **subset of the data**

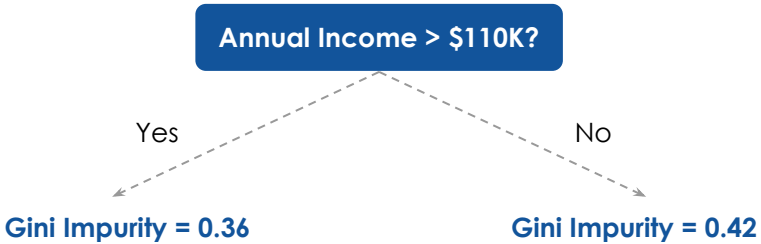
- Compute impurity measure for each candidate subset
- Combine the impurity values for all candidate subsets to get the impurity of the split
- This will determine the quality of the split made - similarity between the data points within each subset
- The goal is to split to decrease the impurity within each subset
- Two common impurity measures - **Gini Impurity** and **Entropy**

Impurity Measures

Gini Impurity	Entropy
Measures the probability of misclassifying a randomly selected data point if we label it randomly	Measures how mixed up or organized data points are in a certain group.
Higher the Gini Impurity, lower the homogeneity - worse the split	Higher the Entropy, lower the homogeneity - worse the split
Ranges between 0 (pure) and 0.5 (impure)	Ranges between 0 (pure) and 1 (impure)

Splitting Mechanism

- How do we determine the attribute to split by?

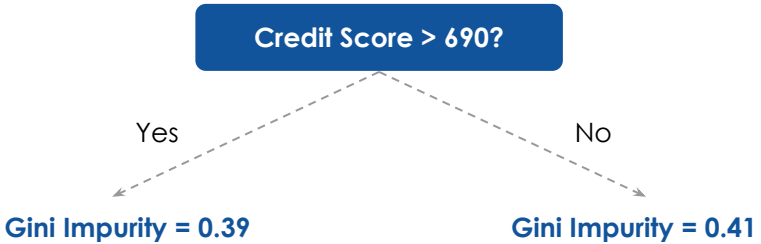


Gini Impurity = 0.43

Gini Impurity of Split = 0.39

Splitting Mechanism

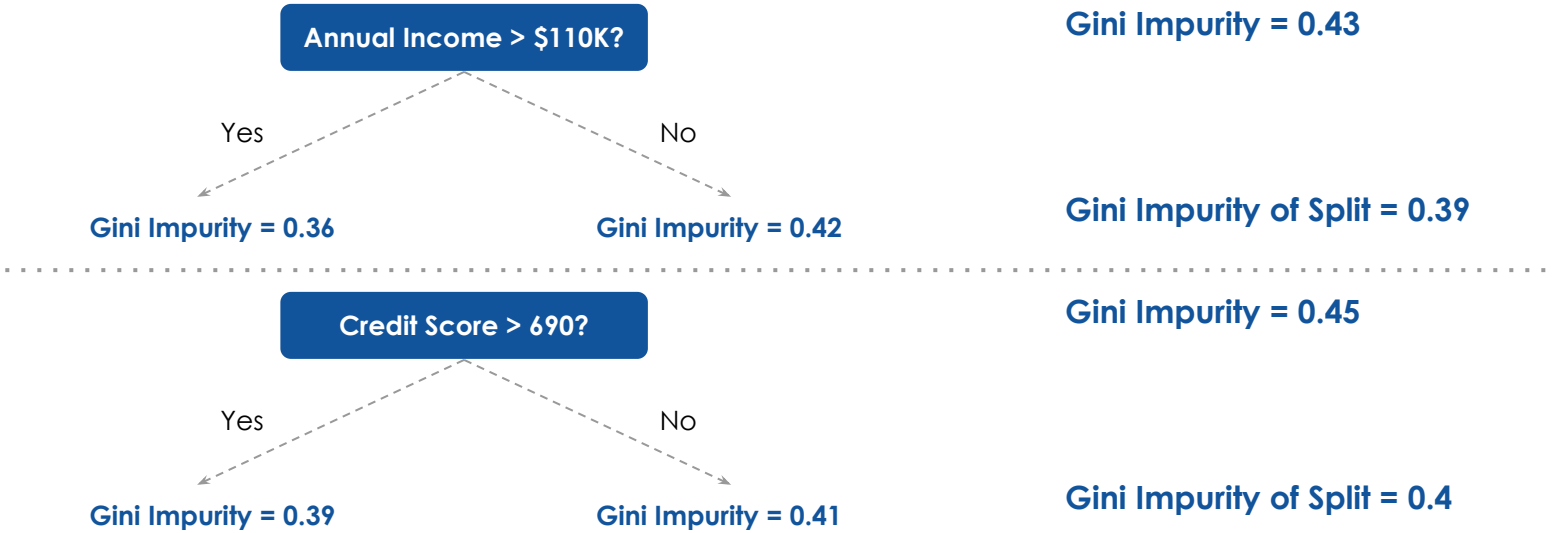
- How do we determine the attribute to split by?



Gini Impurity = 0.45

Gini Impurity of Split = 0.4

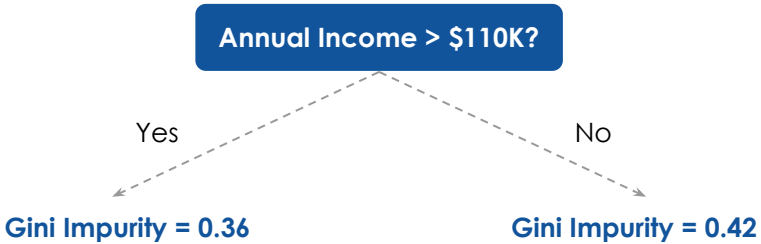
Splitting Mechanism



- Gini Impurity of Split (Annual Income) < Gini Impurity of Split (Credit Score)
- Annual Income will be the first attribute to split by

Splitting Mechanism

- How do we determine the point to split at?

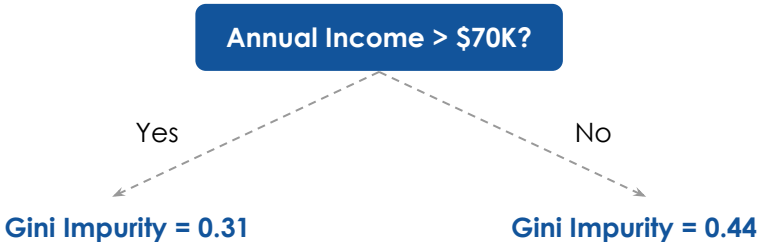


Gini Impurity = 0.43

Gini Impurity of Split = 0.39

Splitting Mechanism

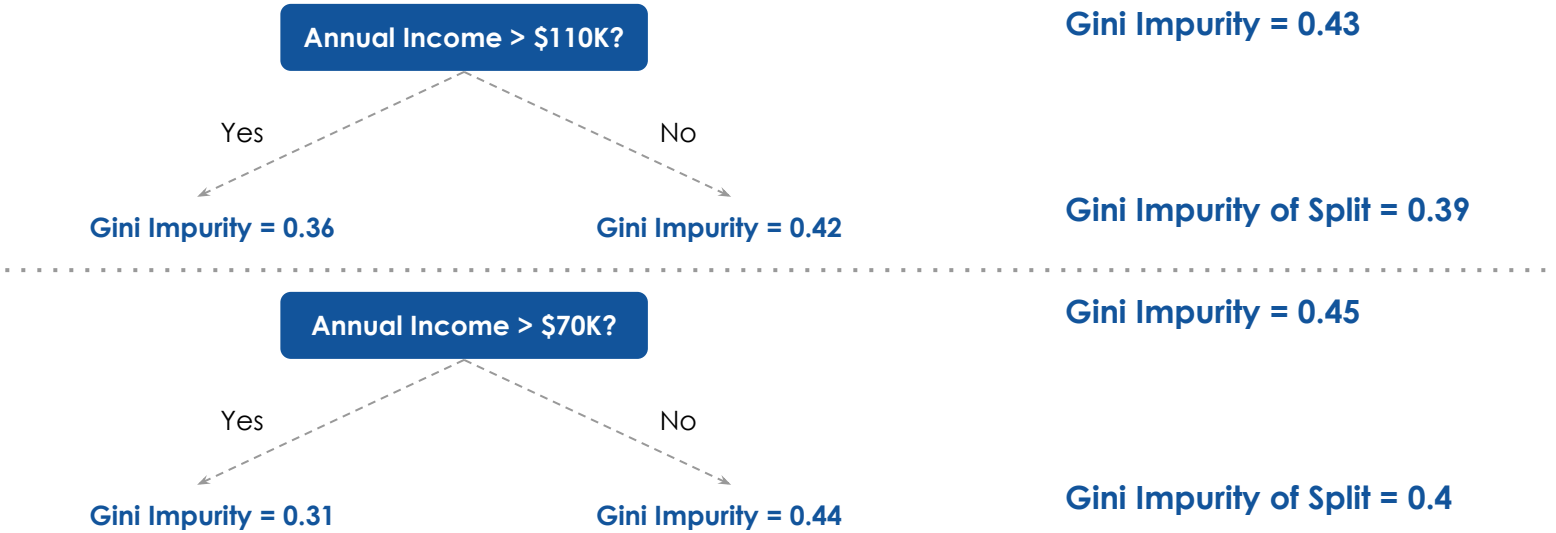
- How do we determine the point to split at?



Gini Impurity = 0.45

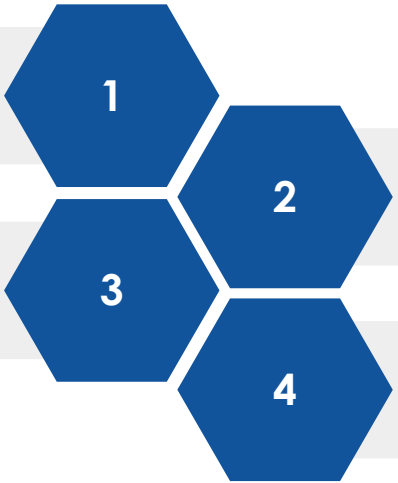
Gini Impurity of Split = 0.4

Splitting Mechanism



- Gini Impurity of Split (Annual Income: $110k) < Gini Impurity of Split (Annual Income: $100k)
- Split for Annual Income at $110K

Splitting Mechanism



Split the data by all attributes at different thresholds

Calculate the Gini Impurity of Split for each attribute for each threshold

Identify the attribute-threshold combination with lowest Gini Impurity

Split the dataset using this attribute-threshold combination

- Applicable for root node as well as decision nodes

Gini Impurity

Gini impurity measures the **likelihood of incorrectly classifying** a randomly chosen element, indicating how mixed the classes are in a node.

$$Gini\ Impurity = 1 - \sum_{i=1}^{k} p_i^2$$

- ***k*** denotes the total number of classes

Gini Impurity

Gini impurity measures the **likelihood of incorrectly classifying** a randomly chosen element, indicating how mixed the classes are in a node.

$$Gini\ Impurity\ (Split) = \frac{N_{Left}}{N}\left(Gini_{Left}\right) + \frac{N_{Right}}{N}\left(Gini_{Right}\right)$$

- ***N*** denotes the total number of datapoints
- N_{Left}, N_{Right} denote the no. of data points in the left, right child nodes respectively
- $Gini_{Left}$, $Gini_{Right}$ denote the Gini Impurity of the left, right child nodes respectively

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider the sample data here to understand how Gini Impurity computations are done

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's compute the Gini Impurity at the root node

No. of Approvals: 6
No. of Rejections: 4

$$\text{Gini Impurity} = 1 - (6/10)^2 - (4/10)^2$$
$$= 1 - 0.36 - 0.16$$
$$= 1 - 0.52$$
$$= 0.48$$

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Annual Income** of **$110K**

Annual Income > $110k

No. of Approvals: 4
No. of Rejections: 1

Gini Impurity = $1 - (1/5)^2 - (4/5)^2$
$= 1 - 0.04 - 0.64$
$= 1 - 0.68$
$= 0.32$

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Annual Income** of **$110K**

Annual Income <= $110k

No. of Approvals: 2
No. of Rejections: 3

Gini Impurity = $1 - (2/5)^2 - (3/5)^2$
$= 1 - 0.16 - 0.36$
$= 1 - 0.52$
$= 0.48$

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Annual Income** of **$110K**

Gini Impurity (Split) at $110K

Gini Impurity = (5/10) * 0.32 + (5/10) * 0.48
= 0.16 + 0.24
= 0.4

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Annual Income** of **$70K**

Annual Income > $70k

No. of Approvals: 5
No. of Rejections: 3

Gini Impurity = $1 - (5/8)^2 - (3/8)^2$
$= 1 - 0.39 - 0.14$
$= 1 - 0.53$
$= 0.47$

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Annual Income** of **$70K**

Annual Income <= $70k

No. of Approvals: 1
No. of Rejections: 1

Gini Impurity = $1 - (1/2)^2 - (1/2)^2$
$= 1 - 0.25 - 0.25$
$= 0.5$

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Annual Income** of **$70K**

Gini Impurity (Split) at $70K

Gini Impurity = (8/10) * 0.47 + (2/10) * 0.5
= 0.376 + 0.1
= 0.476

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Credit Score** of **690**

Credit Score > 690

No. of Approvals: 4
No. of Rejections: 1

Gini Impurity = $1 - (4/5)^2 - (1/5)^2$
= 1 - 0.64 - 0.04
= 1 - 0.68
= 0.32

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Credit Score** of **690**

Credit Score <= 690

No. of Approvals: 2
No. of Rejections: 3

Gini Impurity = $1 - (2/5)^2 - (3/5)^2$
$= 1 - 0.16 - 0.36$
$= 1 - 0.52$
$= 0.48$

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Credit Score** of **690**

Gini Impurity (Split) at 690

Gini Impurity = (5/10) * 0.32 + (5/10) * 0.48
= 0.16 + 0.24
= 0.4

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Credit Score** of **710**

Credit Score > 710

No. of Approvals: 1
No. of Rejections: 1

Gini Impurity = $1 - (1/2)^2 - (1/2)^2$
$= 1 - 0.25 - 0.25$
$= 1 - 0.5$
$= 0.5$

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Credit Score** of **710**

Credit Score <= 710

No. of Approvals: 5
No. of Rejections: 3

Gini Impurity = $1 - (5/8)^2 - (3/8)^2$
$= 1 - 0.39 - 0.14$
$= 1 - 0.53$
Gini Impurity = 0.47

Gini Impurity Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Credit Score** of **710**

Gini Impurity (Split) at 710

Gini Impurity = (2/10) * 0.5 + (8/10) * 0.47
= 0.1 + 0.376
= 0.476

Gini Impurity Computation

- Let's compare the Gini Impurity of all the splits to select the optimal one

Attribute-Threshold Combination	Gini Impurity (Root)	Gini Impurity (Split)	Difference
Annual Income: $110K	0.48	0.4	0.08
Annual Income: $70K	0.48	0.476	0.004
Credit Score: 690	0.48	0.4	0.08
Credit Score: 710	0.48	0.476	0.004

- Both Annual Income at $110k and Credit Score at 690 have lowest Gini impurity
- We can choose either one to start with

Entropy

Entropy is defined as the **randomness** or **measure of the disorder** in the data being processed, i.e., the unpredictability or impurity in the system.

$$Entropy = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

- ***k*** denotes the total number of classes

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's compute the Entropy at the root node

No. of Approvals: 6
No. of Rejections: 4

Entropy = $- (6/10) \log_2(6/10) - (4/10) \log_2 (4/10)$
$= - 0.6 * (-0.737) - 0.4 * (-1.322)$
$= 0.442 + 0.528$
$= 0.97$

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Annual Income** of **$110K**

Annual Income > $110k

No. of Approvals: 4
No. of Rejections: 1

Entropy = $- (1/5) \log_2 (1/5) - (4/5) \log_2 (4/5)$
$= - 0.2 * (-2.32) - 0.8 * (-0.32)$
$= 0.464 + 0.256$
$= 0.72$

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Annual Income** of **$110K**

Annual Income <= $110k

No. of Approvals: 2
No. of Rejections: 3

Entropy = $- (2/5) \log_2 (2/5) - (3/5) \log_2 (3/5)$
$= - 0.4 * (- 1.32) - 0.6 * (- 0.73)$
$= 0.528 + 0.438$
$= 0.966$

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Annual Income** of **$110K**

Entropy (Split) at $110K

Entropy = (5/10) * 0.72 + (5/10) * 0.96
= 0.36 + 0.48
= 0.84

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Annual Income** of **$70K**

Annual Income > $70k

No. of Approvals: 5
No. of Rejections: 3

Entropy = $- (5/8) \log_2 (5/8) - (3/8) \log_2 (3/8)$
$= - 0.625 * (- 0.67) - 0.375 * (- 1.41)$
$= 0.41 + 0.52$
$= 0.93$

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Annual Income** of **$70K**

Annual Income <= $70k

No. of Approvals: 1
No. of Rejections: 1

Entropy = $- (1/2) \log_2 (1/2) - (1/2) \log_2 (1/2)$
$= - 0.5 * (-1) - 0.5 * (-1)$
$= 1$

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	697	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Annual Income** of **$70K**

Entropy (Split) at $70K

Entropy = (8/10) * 0.93 + (2/10) * 1
= 0.74 + 0.2
= 0.94

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Credit Score** of **690**

Credit Score > 690

No. of Approvals: 4
No. of Rejections: 1

Entropy = $- (4/5) \log_2 (4/5) - (1/5) \log_2 (1/5)$
$= - 0.8 * (- 0.32) - 0.2 * (- 2.32)$
$= 0.256 + 0.464$
$= 0.72$

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Credit Score** of **690**

Credit Score <= 690

No. of Approvals: 2
No. of Rejections: 3

Entropy = $- (2/5) \log_2 (2/5) - (3/5) \log_2 (3/5)$
$= - 0.4 * (- 1.32) - 0.6 * (- 0.73)$
$= 0.528 + 0.438$
$= 0.96$

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Credit Score** of **690**

Entropy (Split) at 690

Entropy = (5/10) * 0.72 + (5/10) * 0.96
= 0.36 + 0.48
= 0.84

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Credit Score** of **710**

Credit Score > 710

No. of Approvals: 1
No. of Rejections: 1

Entropy = $- (1/2) \log_2 (1/2) - (1/2) \log_2 (1/2)$
$= - 0.5 * (-1) - 0.5 * (-1)$
$= 0.5 + 0.5$
$= 1$

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Credit Score** of **710**

Credit Score <= 710

No. of Approvals: 5
No. of Rejections: 3

Entropy = $- (5/8) \log_2 (5/8) - (3/8) \log_2 (3/8)$
$= - 0.625 * (- 0.67) - 0.375 * (- 1.41)$
$= 0.41 + 0.52$
$= 0.93$

Entropy Computation

Annual Income	Credit Score	Approval Status
69.72	702	Yes
60.87	615	No
70.45	587	No
77.9	712	No
84.38	691	Yes
112.74	723	Yes
110.54	707	Yes
134.59	607	Yes
142.06	669	Yes
161.52	618	No

Let's consider splitting the root node at **Credit Score** of **710**

Entropy (Split) at 710

Entropy = (2/10) * 1 + (8/10) * 0.93
= 0.2 + 0.74
= 0.94

Entropy Computation

- Let's compare the Entropy of all the splits to select the optimal one

Attribute-Threshold Combination	Entropy (Root)	Entropy (Split)	Difference
Annual Income: $110K	0.97	0.84	0.11
Annual Income: $70K	0.97	0.94	0.01
Credit Score: 690	0.97	0.84	0.11
Credit Score: 710	0.97	0.94	0.01

- Both Annual Income at $110k and Credit Score at 690 have lowest entropy
- We can choose either one to start with

Evaluation Metrics

- Model made **predictions**
- **Four possible outcomes**

Case 1	Case 2	Case 3	Case 4
The application was approved and the model also predicted that it would be approved	The application was approved but the model predicted that it wouldn't be approved	The application wasn't approved but the model predicted that it would be approved	The application wasn't approved and the model also predicted that it wouldn't be approved

Evaluation Metrics

- Consider an **approved application** to be a **positive outcome**

Actual Positive Predicted Positive	Actual Positive Predicted Negative	Actual Negative Predicted Positive	Actual Negative Predicted Negative
Case 1	**Case 2**	**Case 3**	**Case 4**
The application was approved and the model also predicted that it would be approved	The application was approved but the model predicted that it wouldn't be approved	The application wasn't approved but the model predicted that it would be approved	The application wasn't approved and the model also predicted that it wouldn't be approved

Evaluation Metrics

True Positive (TP)	False Negative (FN)	False Positive (FP)	True Negative (TN)
Actual Positive Predicted Positive	Actual Positive Predicted Negative	Actual Negative Predicted Positive	Actual Negative Predicted Negative
Case 1	**Case 2**	**Case 3**	**Case 4**
The application was approved and the model also predicted that it would be approved	The application was approved but the model predicted that it wouldn't be approved	The application wasn't approved but the model predicted that it would be approved	The application wasn't approved and the model also predicted that it wouldn't be approved

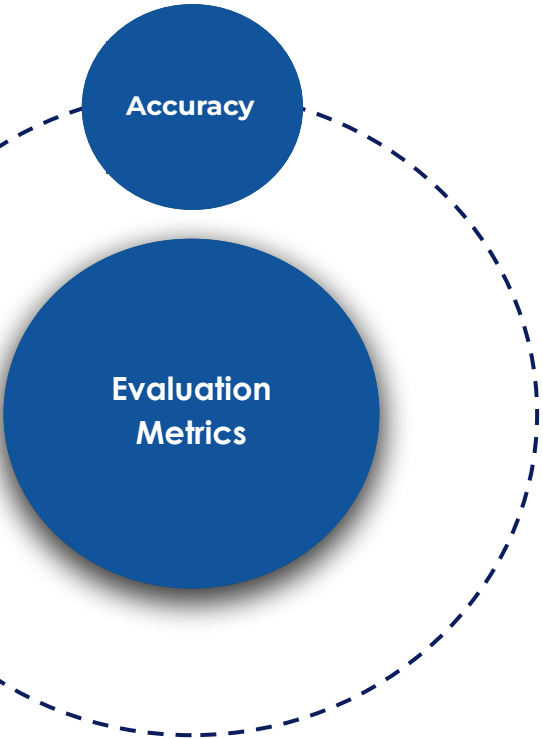
Evaluation Metrics

- Outcomes can be represented using the **confusion matrix**

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

But what **proportion** of the **predictions** are **correct**?

Evaluation Metrics

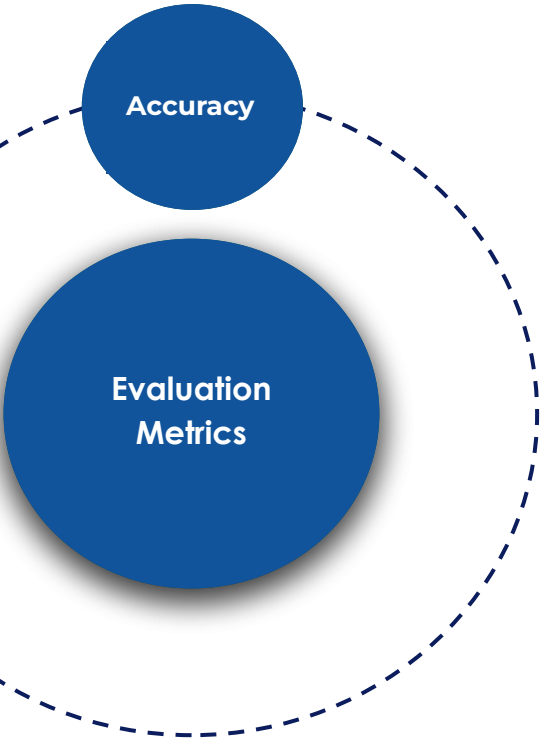


- Most intuitive and widely used metric

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Represents the **proportion of correct predictions** out of the total number of predictions
- Ranges from 0 to 1

Evaluation Metrics



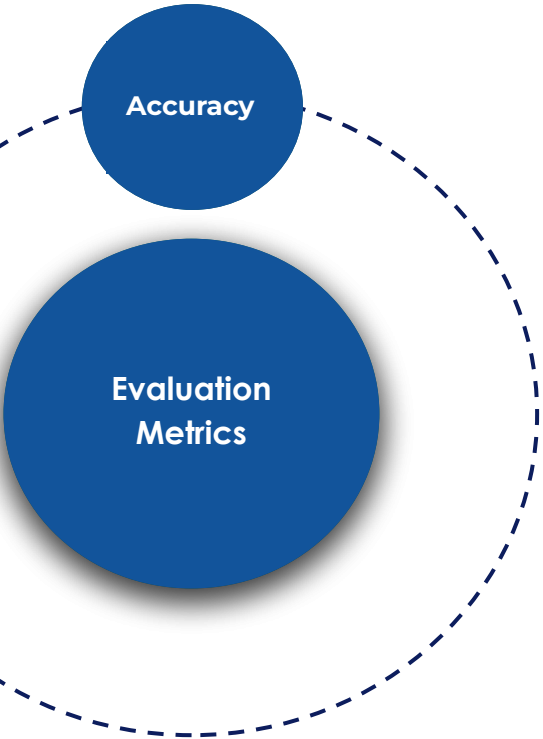
- Most intuitive and widely used metric

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Represents the **proportion of correct predictions** out of the total number of predictions
- Ranges from 0 to 1

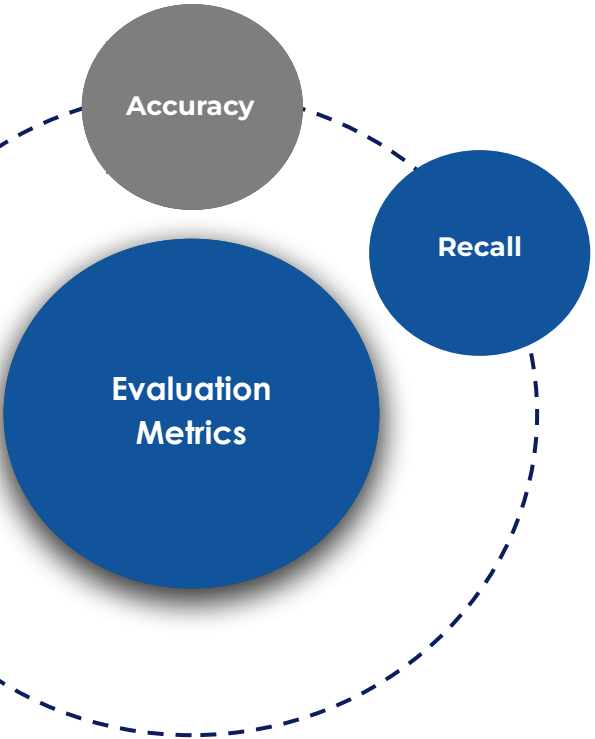
Is accuracy a good metric for all scenarios?

Evaluation Metrics



- Consider the scenario where we want to diagnose cancer
- Data is provided for 1000 patients, out of which 10 have cancer
- The model predicts that none of the patients have cancer
- The model would have a **99% accuracy!**
- But the model **missed the main point** - could not do what it was supposed to

Evaluation Metrics



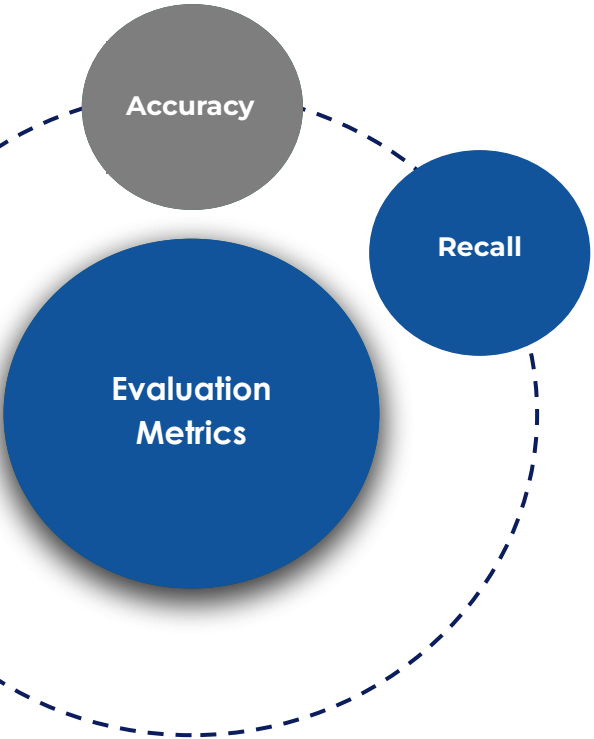
- The ratio of true positives to total actual positives

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

- Tells us out of all the 'positive' instances, how many did the model remember correctly
- Ranges from 0 to 1

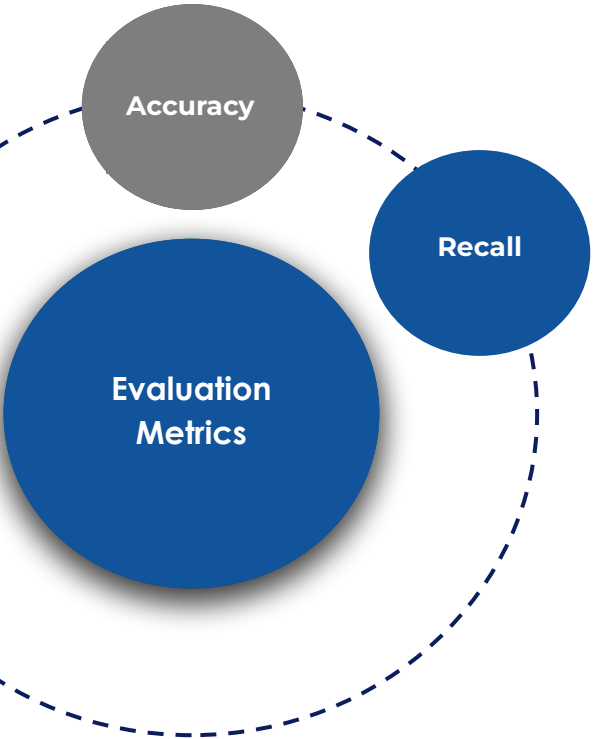
Recall ↑ **False Negatives** ↓

Evaluation Metrics



- For the cancer diagnosis use case, the model achieved 99% accuracy by predicting that no one has cancer
- For the same model, **recall will be 0** - none of the patients with cancer were diagnosed
- Recall a better metric in this case - forces the model to distinguish cancer patients

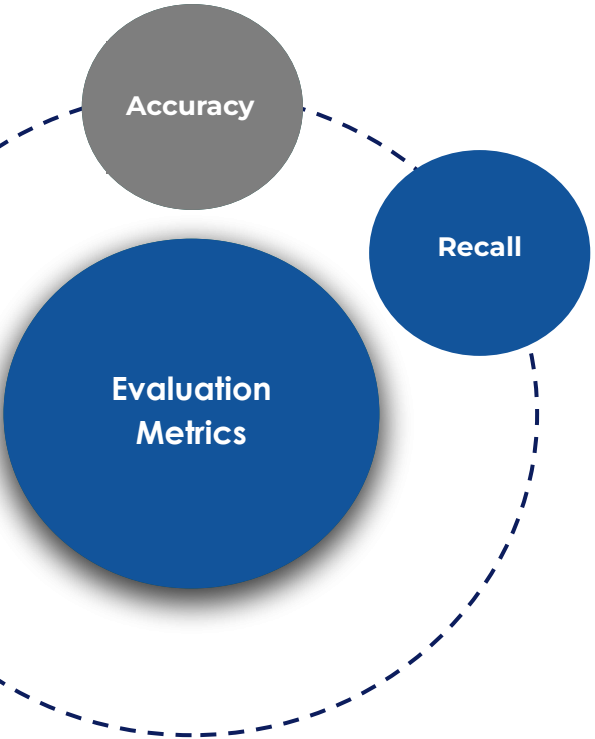
Evaluation Metrics



- For the cancer diagnosis use case, the model achieved 99% accuracy by predicting that no one has cancer
- For the same model, **recall will be 0** - none of the patients with cancer were diagnosed
- Recall a better metric in this case - forces the model to distinguish cancer patients

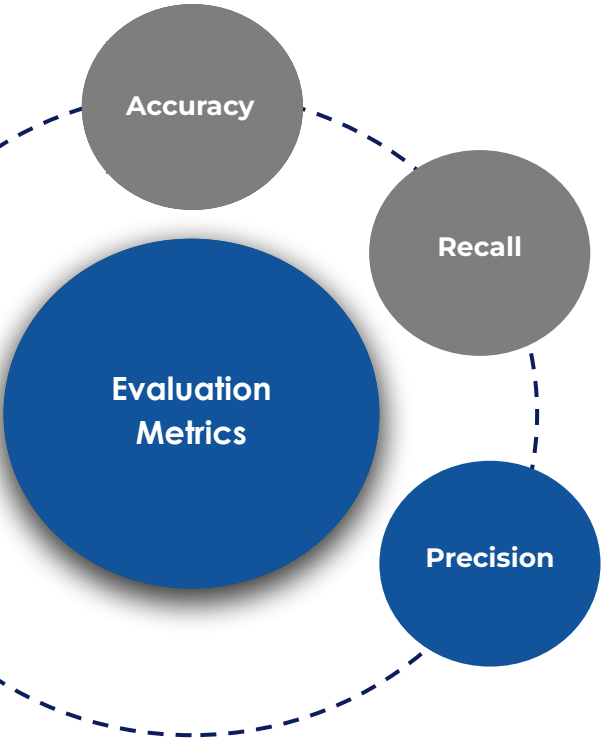
Does recall solve the problem with accuracy for all scenarios?

Evaluation Metrics



- Consider the scenario of estimating the likelihood of customers engaging with a marketing campaign
- Data is provided for 1000 users, 600 engaged
- The model predicts that all customers will engage
- **Recall will be 1** - indicating great performance
- But the model considered customers that are unlikely to engage - will require more effort for nurturing the leads

Evaluation Metrics



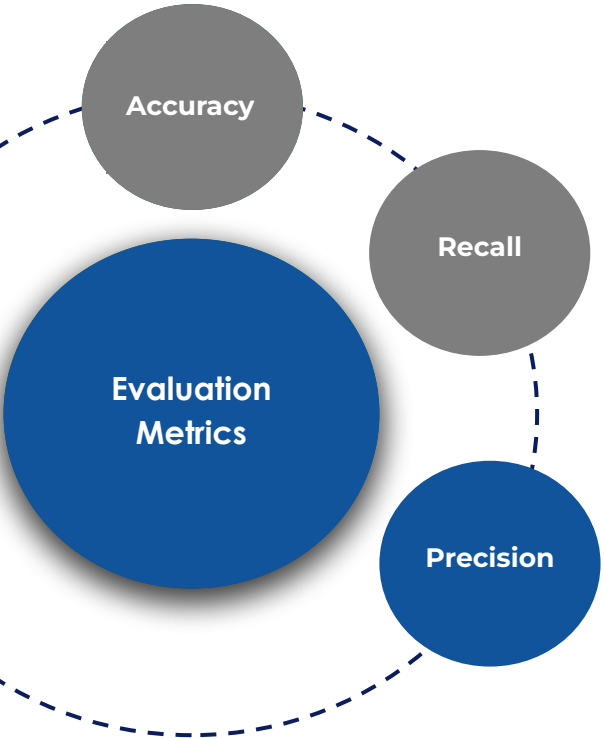
- The ratio of true positives to total predicted positives

$$Precision = \frac{TP}{TP + FP}$$

- Tells us out of all the predicted ‘positive' instances, how many are actually ‘positive’
- Ranges from 0 to 1

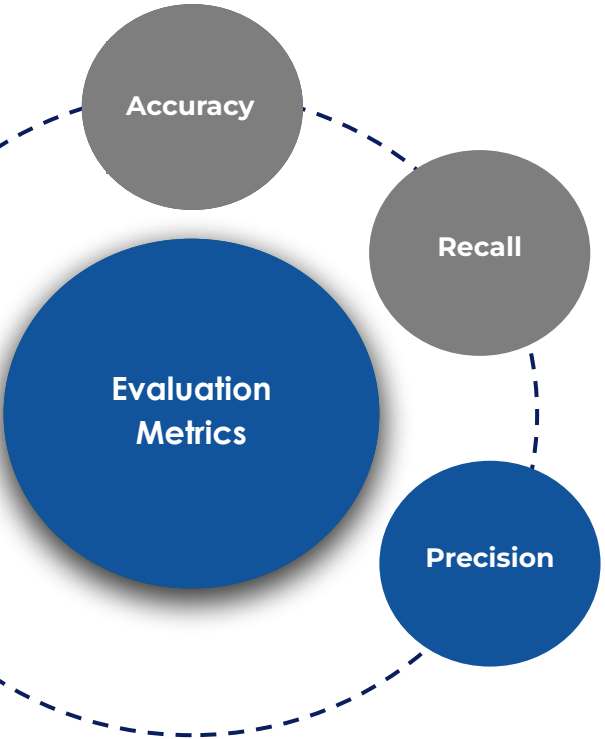
Precision ↑ **False Positives** ↓

Evaluation Metrics



- For the marketing campaign engagement prediction use case, the model had a 100% recall by predicting everyone will engage
- For the same model, **precision will be 60%** - more focused effort in nurturing leads
- Precision is a better metric in this case - forces the model to distinguish customers who will engage

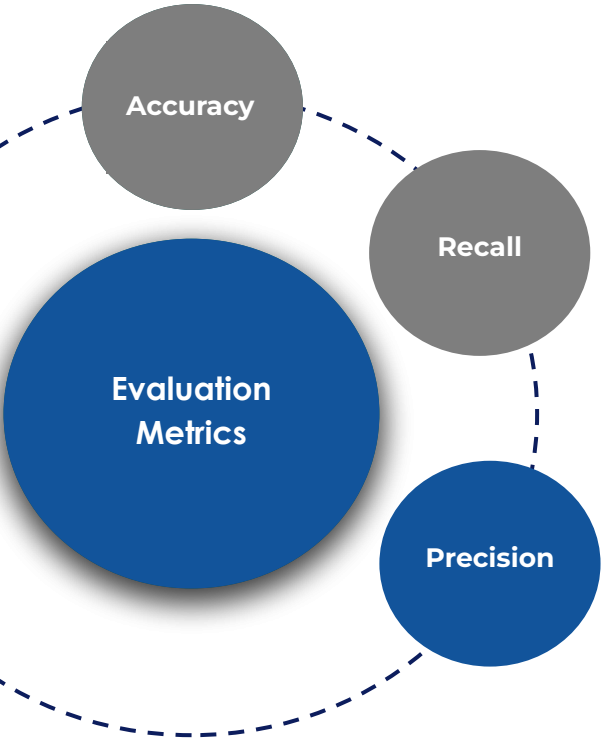
Evaluation Metrics



- For the marketing campaign engagement prediction use case, the model had a 100% recall by predicting everyone will engage
- For the same model, **precision will be 60%** - more focused effort in nurturing leads
- Precision is a better metric in this case - forces the model to distinguish customers who will engage

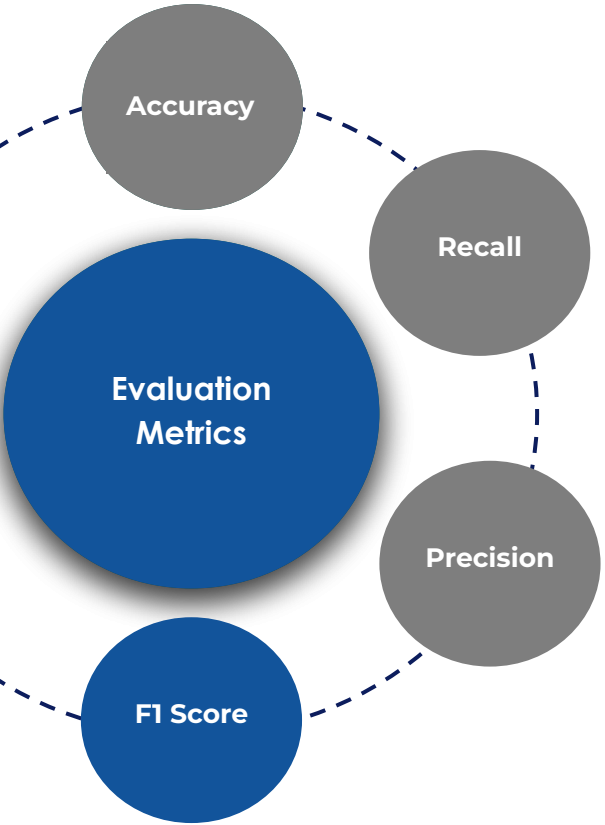
Does precision solve the problems with accuracy and recall for all scenarios?

Evaluation Metrics



- Consider our context of credit card approval
- False Negatives - credible applications rejected
- False Positives - non-credible applications approved
- Need to reduce both these errors simultaneously

Evaluation Metrics



- Maintains a balance between False Positives and False Negatives

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- Ranges from 0 to 1

F1 Score ↑ **False Positives** ↓ **False Negatives** ↓

Model Complexity vs Performance

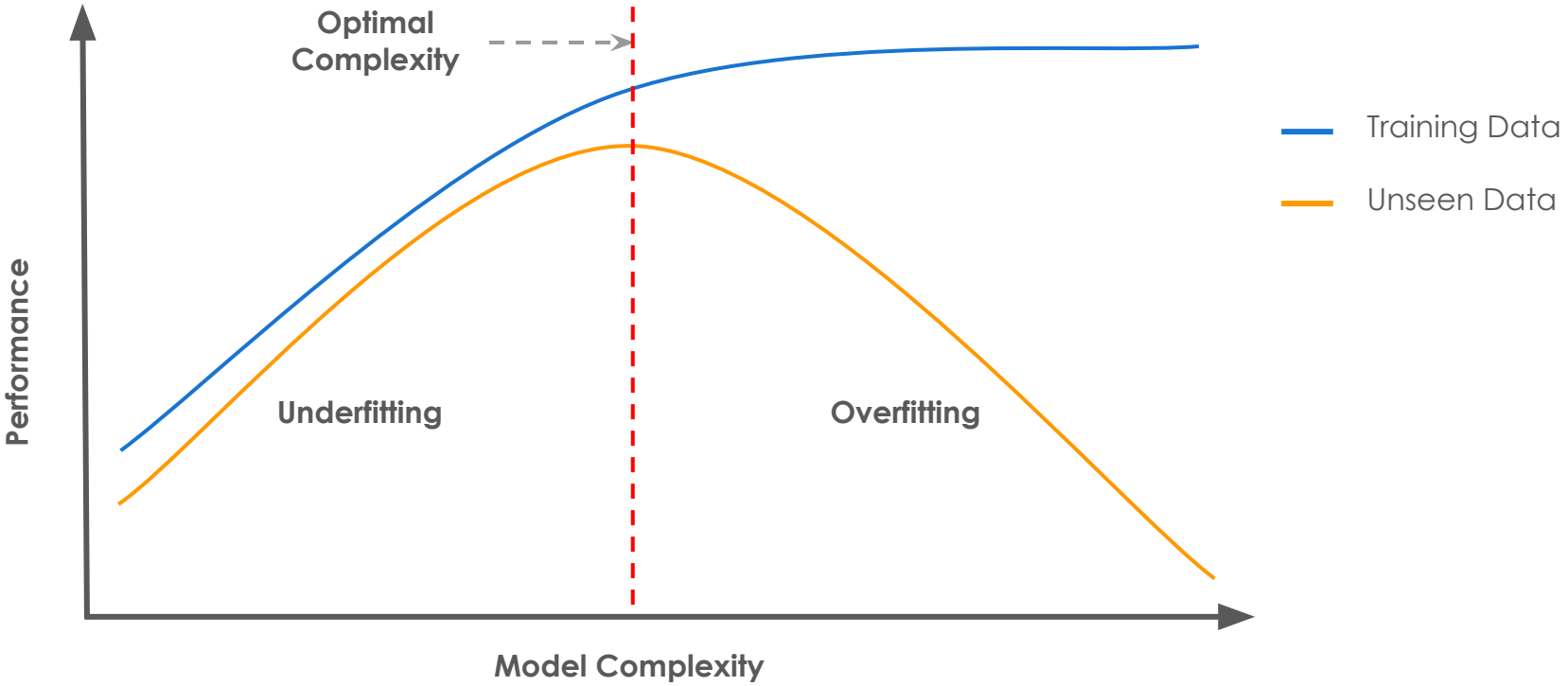
- The complexity of a decision tree depends a lot on the depth of the tree *
- More the depth a tree, more complex it is *

But does a more **complex tree** always **capture more information**?

Will a **complex decision tree** continue to **perform well** on **new (unseen) data**?

* There are other parameters too that determine the complexity of a tree.

Model Complexity vs Performance



Overfitting and Underfitting

Overfitting	Underfitting
Model performs well on training data but poorly on test data	Model performs poorly on both training and test data
Model captures the noise in the training data in addition to identifying underlying patterns	Model fails to capture the underlying patterns in the training data
Generally observed in high complexity models	Generally observed in low complexity models

Overfitting and Underfitting

Overfitting	Underfitting
Model performs well on training data but poorly on test data	Model performs poorly on both training and test data
Model captures the noise in the training data in addition to identifying underlying patterns	Model fails to capture the underlying patterns in the training data
Generally observed in high complexity models	Generally observed in low complexity models

How can we overcome the problem of overfitting in complex decision trees?

Pruning

- Don’t let the decision tree **grow beyond a certain point!**

Pruning is a data compression technique that **reduces the size** of decision trees by **removing** sections of the tree that are **non-critical and redundant** to classify instances.

- Helps optimize the performance by **reducing complexity** and **improving generalization**
- Prevents the tree from **memorizing noise** in the training data
- **Simplifies** decision trees, making them **easier to interpret**

How do we **prune decision trees**?

Pre-pruning

Pre-pruning is the method which **prevents** the tree from **growing fully** by introducing a **stopping criterion**

- Constrain the tree growth before it gets too complex
- Faster and computationally efficient
- Done by specifying parameters like tree depth, minimum leaves per node, and minimum data points to consider for splits

Pre-pruning

Pre-pruning is the method which **prevents** the tree from **growing fully** by introducing a **stopping criterion**

Max Depth	Min Samples for Leaf	Min Samples to Split
Sets the maximum depth to which a tree can grow	Sets the minimum number of data points that need to be present for a node to be a leaf node	Sets the minimum number of data points that need to be present for a node to be split further

Post-pruning

Post-pruning is a process in which **nodes and subtrees** are **replaced** with **leaves** to reduce complexity.

- Allow the tree to grow fully and then remove unnecessary branches and/or leaves
- Slower and computationally expensive
- Done by calculating the importance of each part of the tree using methods like cost complexity pruning and reduced error pruning

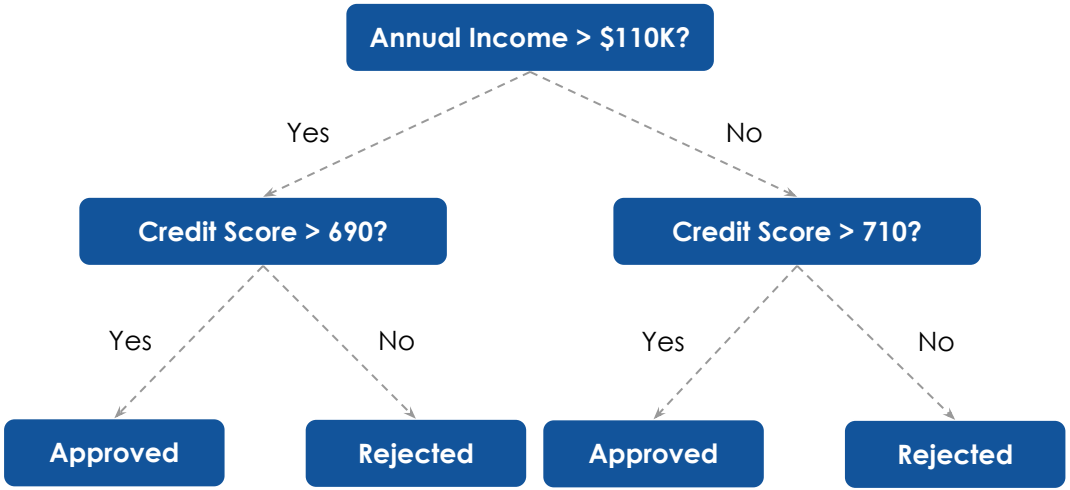
Post-pruning

Post-pruning is a process in which **nodes and subtrees** are **replaced** with **leaves** to reduce complexity.

Cost Complexity Pruning	Reduced Error Pruning
Balances tree complexity with classification accuracy by penalizing the number of nodes	Identifies subtrees that do not improve accuracy on the validation set and prunes them

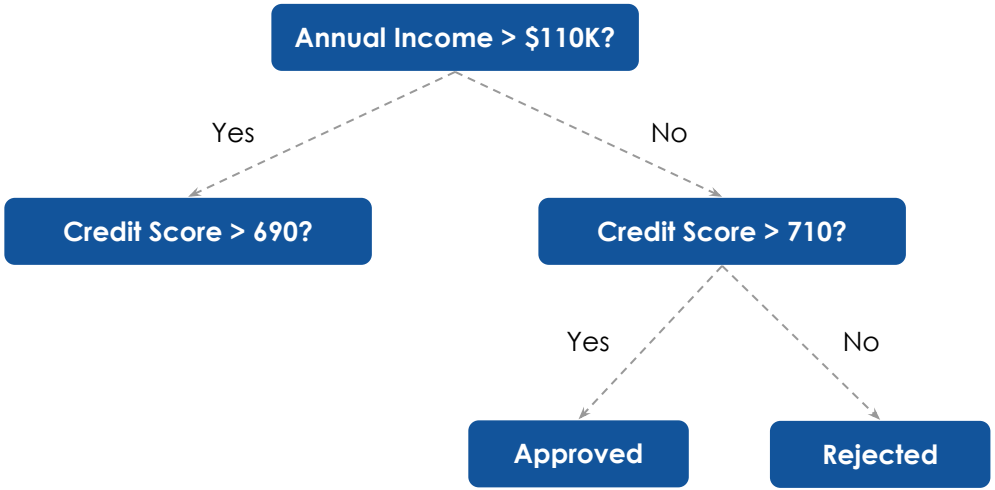
Cost Complexity Pruning

- Consider the decision tree we built



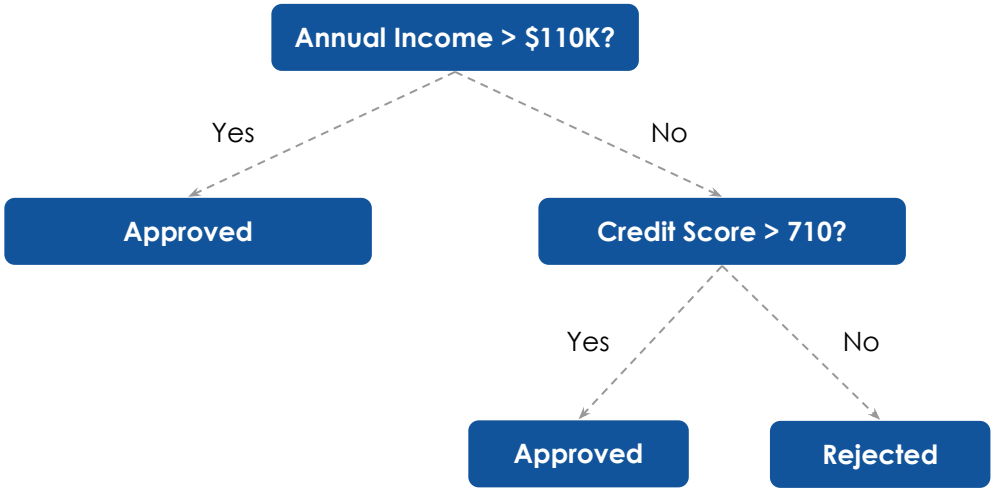
Cost Complexity Pruning

- What happens if we remove the bottom left branches?



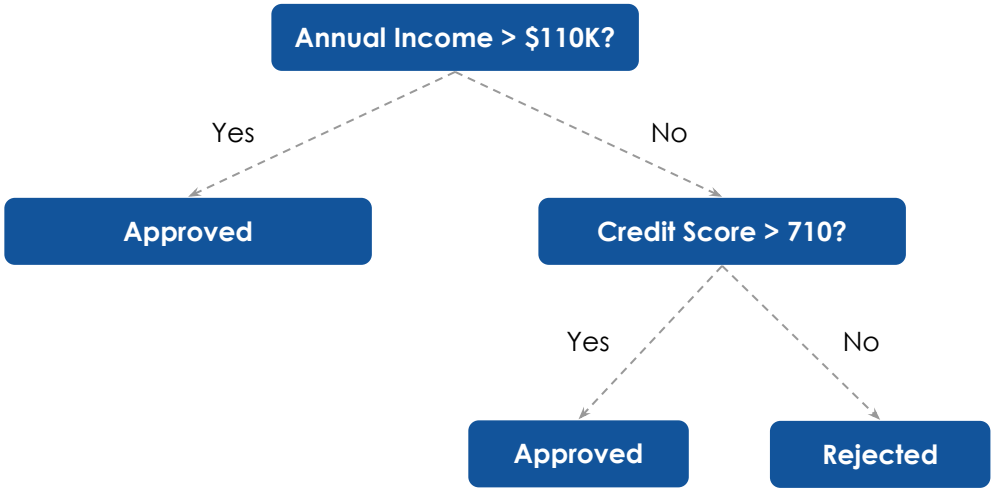
Cost Complexity Pruning

- That decision node will become a leaf node



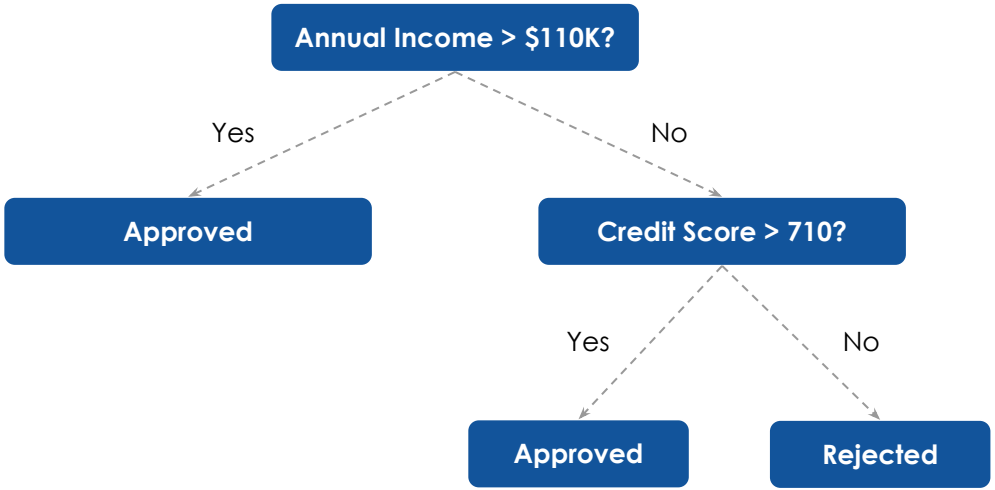
Cost Complexity Pruning

- This results in a dip in model performance on the training data, but not necessarily in unseen data



Cost Complexity Pruning

- How to measure the contribution of the subtree that was removed?



Cost Complexity Pruning

- One way would be to take compute the relative error decrease per node for that subtree

$$\alpha = \frac{Error\ (Pruned) - Error\ (Original)}{Number\ of\ nodes\ reduced}$$

- α controls the trade-off between the complexity of the tree and its performance

But how does this method work?

Cost Complexity Pruning

Tree	Error on Unseen Data	α
T_0	25.56	0.000162
T_1	23.37	0.000869
T_2	21.84	0.00134
T_3	20.15	0.00761
T_4	19.32	0.0147
T_5	22.06	0.0283
…	…	…
T_m	32.75	0.0645

- Start from the fully grown tree
- Consider all possible subtrees
- Calculate α and the error on unseen data corresponding to pruning each subtree
- Remove the subtree corresponding to the minimum α
- Repeat

Cost Complexity Pruning

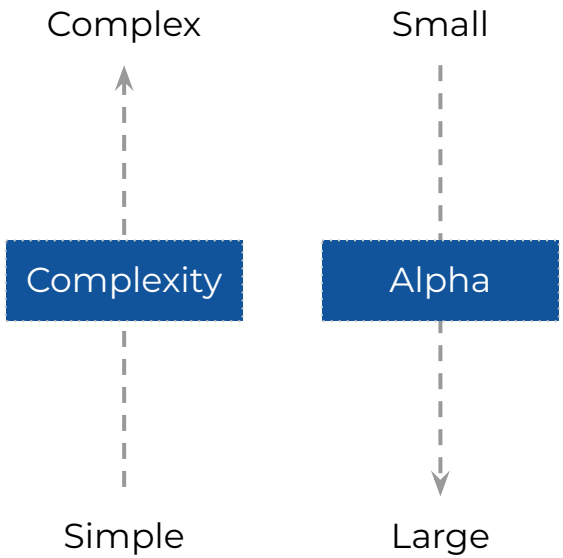
Tree	Error on Unseen Data	α
T_0	25.56	0.000162
T_1	23.37	0.000869
T_2	21.84	0.00134
T_3	20.15	0.00761
T_4	19.32	0.0147
T_5	22.06	0.0283
...	...	...
T_m	32.75	0.0645

- Start from the fully grown tree
- Consider all possible subtrees
- Calculate α and the error on unseen data corresponding to pruning each subtree
- Remove the subtree corresponding to the minimum α
- Repeat

But what is the optimum value of α?

Cost Complexity Pruning

Tree	Error on Unseen Data	α
T_0	25.56	0.000162
T_1	23.37	0.000869
T_2	21.84	0.00134
T_3	20.15	0.00761
T_4	19.32	0.0147
T_5	22.06	0.0283
...	...	...
T_m	32.75	0.0645



Cost Complexity Pruning

Tree	Error on Unseen Data	α
T_0	25.56	0.000162
T_1	23.37	0.000869
T_2	21.84	0.00134
T_3	20.15	0.00761
T_4	19.32	0.0147
T_5	22.06	0.0283
…	…	…
T_m	32.75	0.0645

- As we remove subtrees, the error on training data will always increase
- But the error on unseen data will first decrease and then increase after reaching a minimum

Optimum α value corresponds to the tree having the lowest error on unseen data

Decision Trees for Regression

- Consider the case where we have approved the credit card applications for several candidates
- Once an application is approved, we need to determine the credit limit to be granted
- We can use linear regression for this
- But linear regression assumes a linear relationship between the independent and dependent variables

Can we use decision trees for predicting the credit limit?

Decision Trees for Regression

- Let’s consider the same set of attributes as before

Annual Income < $50K?

Decision Trees for Regression

- How do we split?

Annual Income < $50K?

Gini Impurity

- Measures the chances of misclassification
- But we have a regression problem at hand

Decision Trees for Regression

- How do we split?

Annual Income < $50K?

Gini Impurity

- Better to measure the variation in the data
- Lower variation - more ‘purity’

Decision Trees for Regression

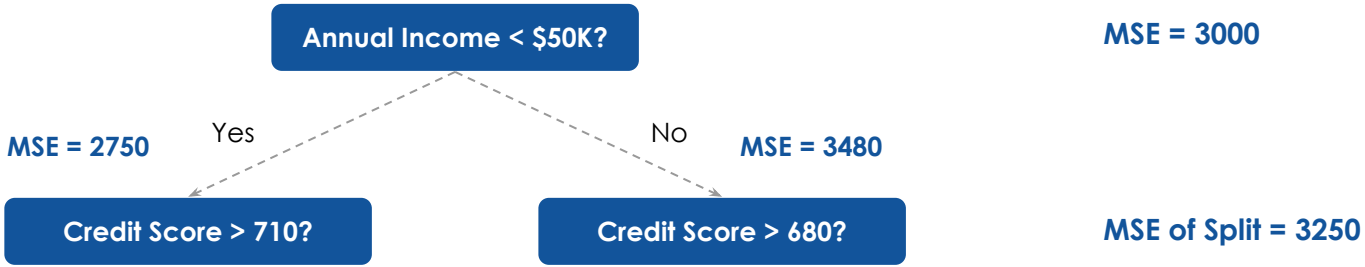
- How do we split?

Annual Income < $50K?

Mean Squared Error

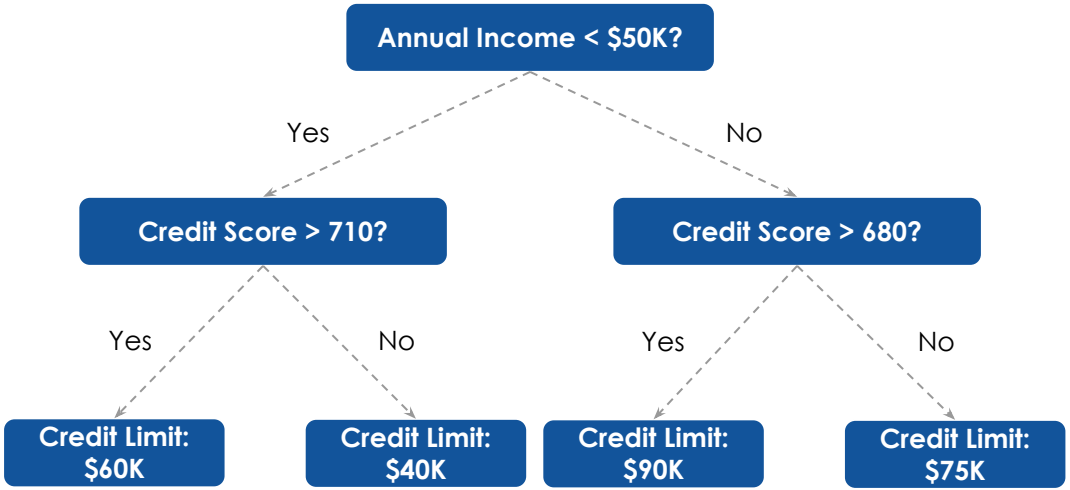
- Measures the variation in the data

Decision Trees for Regression



- MSE of Split is computed in the same way as Gini Impurity of Split

Decision Trees for Regression



MSE = 3000

MSE of Split = 3250

- At the **leaf node**, take the **average of all the data points** as the **prediction**

Decision Trees for Regression

Classification Trees	Regression Trees
Predict a categorical value (class label)	Predict a numerical value
Uses Impurity measures like Gini Impurity and Entropy for splitting	Uses measures like Mean Squared Error (MSE) for splitting
Focuses on reducing impurity with each split	Focuses on reducing variance with each split

Summary

Here’s a quick recap of what we've learned:

- **Business Problems and Solution Space**: Decision trees predict outcomes like customer churn and fraud detection across domains like retail and finance.
- **Overview of Decision Trees**: Decision trees split data based on feature values to create a model for classification and regression tasks.
- **Impurity Measures and Splitting Criteria**: Gini Impurity and Entropy measure classification homogeneity, while Mean Squared Error is used for regression splits.

Summary

- **Evaluation Metrics for Classification**: Metrics like accuracy, precision, recall, and confusion matrices evaluate classification model performance.
- **Pruning Techniques**: Pre-pruning sets growth constraints, and post-pruning removes non-critical branches to prevent overfitting.
- **Decision Trees for Regression**: Used for predicting continuous values, minimizing variance with measures like MSE to optimize splits.

Learning Outcomes

You should now be able to:

- Articulate the data partitioning mechanism and predictive process of decision trees
- Compare impurity measures and implement optimal splitting criteria
- Assess model performance with key metrics and identify crucial decision-making features
- Analyze and examine pruning techniques to enhance model performance and generalization
- Utilize decision trees to solve practical problems and derive actionable insights for informed decisions

Great Learning

Happy Learning !