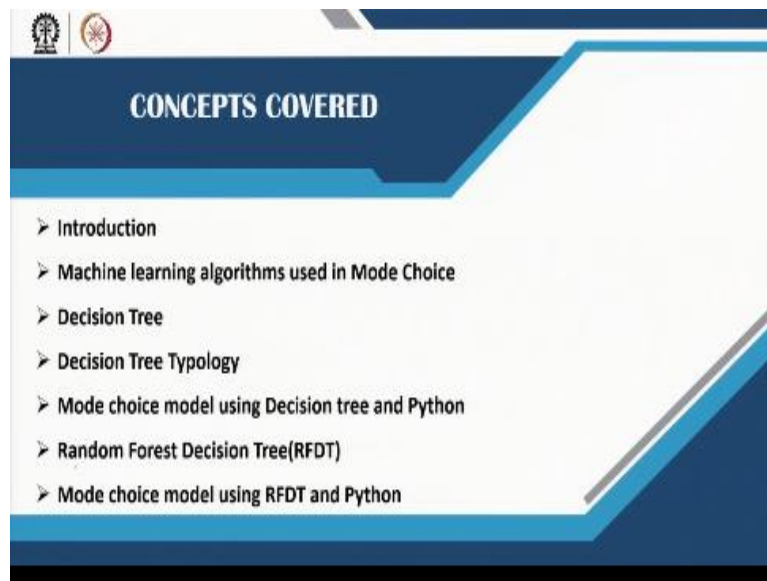


**Urban Landuse and Transportation Planning**  
**Prof. Debapratim Pandit**  
**Department of Architecture and Regional Planning**  
**Indian Institute of Technology – Kharagpur**

**Lecture 59**  
**Mode Choice using Machine Learning**

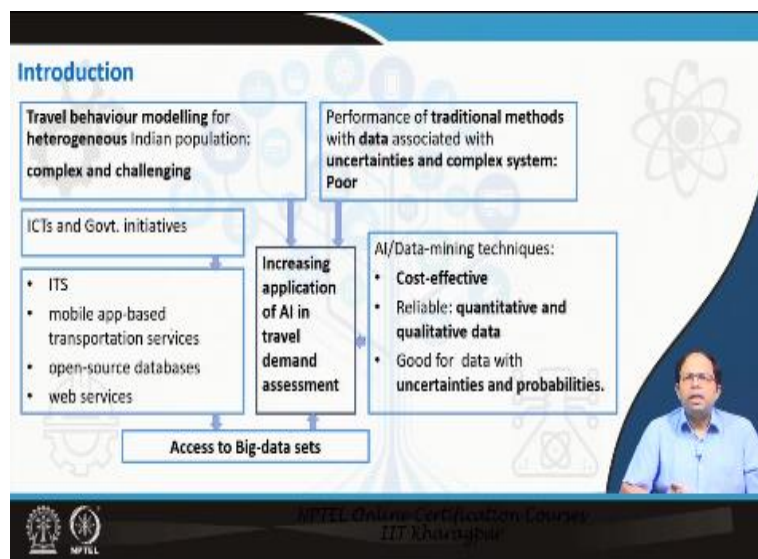
Welcome back. In lecture 59, Mode choice models using machine learning will be addressed.

**(Refer Slide Time: 00:28)**



The different concepts that are covered include Introduction to machine learning, Machine learning algorithms used in mode choice, Decision tree, Different types of decision tree, Mode choice model using decision tree and Python, Random forest decision tree, Mode choice model using random forests decision tree and Python.

**(Refer Slide Time: 00:55)**



In the last lecture one of the popular activity-based model model, Albatross was discussed. It uses decision trees to determine the different choices or steps in the model. Any kind of choices could be modelled using decision trees. In this particular lecture, mode choice will be covered.

## **Introduction**

Travel behaviour modelling for heterogeneous population and particularly for India is becoming complex and challenging as a lot of data is involved. Also, there are a lot of relationships which needs to be determined. So, it is necessary to look into different kinds of model systems, which makes it easier to build complex models, unlike the choice models that we have built earlier. There are performance-related issues concerning traditional methods associated with uncertainties and difficulty in building complex systems reinforcing the need for better models.

Government of India has taken a lot of initiatives promoting Internet and communication technologies to be adopted for urban areas as the cities are becoming smart. Intelligent transportation systems collect data every day, like locations of buses at different times of the day and many other types of data. Such data collection helps in understanding the speed along a particular corridor. These are big data sets. There is also mobile based data on transportation services, open source databases, web services, etc.

Artificial intelligence (AI) can be used in managing such big data and building complex models. Increasing application of AI is noted for travel demand assessment. This is adopted for its cost effectiveness, ability in managing big data and finding trends or patterns in them, reliability it provides for quantitative and qualitative data and since it is good for data with uncertainties and probabilities.

**(Refer Slide Time: 03:42)**

**Machine learning algorithms applied in mode choice**

Machine learning methods are an alternative to statistical approaches for modeling travel mode choice. These models can determine and present complex relationships from data without making strict assumptions about it.

**Artificial Neural networks:**

- ❑ Artificial neurons and directed connections in between.
- ❑ The model is trained using data with both independent variables and the outcome (dependent variable) to determine the weights for these connections while being processed by the neurons in the hidden layer.
- ❑ The number of neurons in the input layer equals the number of input variables and number of neurons in the output layer equals the number of outputs. — *n*
- ❑ Number of neurons in the hidden layer depends on the number of decision boundaries we need to draw between the different classes/categories that need to be predicted.

## Machine learning algorithms applied in mode choice

Among AI or machine learning, we usually use the term machine learning. These algorithms are an alternative to the existing statistical approaches for mode choice modelling and particularly multinomial logit choice. These models can determine and present complex relationships from data without making strict assumptions about it, unlike MNL model. In the MNL model, it is assumed that the error was distributed following gumbell distribution or normal distribution. Such assumptions are not present for AI models and avoiding these assumptions results in robust models.

### **Artificial neural networks**

*Artificial neurons and direct connections* - Artificial neural network or artificial neurons, is like a decision box or a decision window that gives output based on an input given and based on similar previous examples. Multiple artificial neurons can be used based on the number of decisions needed and these decision windows have directed connections, that makes one decision window getting connected to other or a set of them connecting in a particular order. So, as represented in the above figure, neurons are trained using previous data on the direction it should proceed. Thus, the output is based on different attributes or parameters and the weights of the parameters. It can be said that the predictor variables are similar to explanatory variables and the output variable is similar to the dependent variable.

*The model is trained using data with both independent variables and the outcome, which is the dependent variable to determine the weights for these connections while being processed*



## **Decision trees (DT) or classification trees (CT)**

*This uses a tree like data structure for classification and the nodes of the tree represent binary decision rules, edges the answers to the question and leaves represent the classes –* In the figure given above, the leaves are the final results of a mode choice question which are the three classes or outputs such as the bus, auto-rickshaw or car. Then, there is the Root node, which is the first decision window which is about car ownership. Here, it says whether a person has no vehicle or less than 2 vehicles and it becomes a decision point. If it is the no vehicle case, they will either use bus or auto and if it is the less than 2 vehicles is owned case, then, that means, they can have at least one vehicle, and that some of them may have used a car, and some of them use bus and auto. So, certain attributes are considered and questions are asked on those attributes and based on the questions, the data is segregated into different groups.

*Decision trees are nonlinear decision making with simple linear decision surface -* It solves nonlinear decisions and but linear decision surfaces are used.

Decision trees are predictive models constructed using algorithms which splits the data set *based on different conditions* - The different conditions are the decision rules which is done by *if then else* statement as explained in the decision tree for mode choice given in the above figure. If the answer of the question is yes, then a particular choice is decided and if it is not yes then another choice is decided.

*Nonparametric supervised learning method used for both classification and regression tasks.* Learning indicates that we have to train the algorithm based on the data set and the existing results and accordingly, it identifies the questions that need to be asked to segregate the data so, that one can arrive at results very fast. Consider a case, where it has to be predicted whether a person uses a car based on parameters like income, vehicle ownership and some other known parameters. A series of questions have to be asked based on the answer of previous questions such as what is the income, vehicle ownership status etc. so that the ultimate mode choice is predicted. Decision trees give the sequence in which such questions have to be asked. Root node represents the entire population or sample from where the first splitting starts and a decision node are the intermediate nodes where further splitting is possible. That means one cannot keep on splitting the data because it may make the model

more complex with so many subcategories. Splitting is dividing the root node and the decision node into further decision nodes or leaf nodes based on decision rules.

**(Refer Slide Time: 13:26)**

Tree-based ensemble techniques:

**Boosting (BOOST):**  
DTs are built in sequence.  
Successive tree attempts to correct wrong classifications. Finally, weighted voting among all trees.

**Bagging (BAG):**  
DTs are trained in parallel using sample data(bootstrap)  
Majority voting among all trees for class assignment.

**Random Forest(RF):**  
DTs are trained in parallel using sample data(bootstrap)  
Splits at the nodes are determined by a random subset of variables.  
Majority voting among all trees for class assignment.

NPTEL Online Certification Course  
IIT Kharagpur

## Tree based ensemble techniques

Ensemble techniques refer to combining different techniques, to gain the strength of each one of these techniques.

*Boosting (BOOST)* - where decision trees are built-in sequence. Here, successive tree attempts to correct wrong classification or wrong specifications done by the previous tree. Finally, weighted voting among all trees is considered to get the final result.

*Bagging (BAG)* – where decision trees are trained in parallel using sample data, which is generated by bootstrapping. Bootstrapping is a method that can create many samples from a few samples following certain distributions. Multiple decision trees are trained based on this particular data in parallel and majority voting among all the trees are taken for a class assignment.

*Random forest (RF)* – Here also, the decision trees are trained in parallel using sample data, generated using the bootstrapping method. Splits or divisions are determined by a random subset of variables. Majority voting on all trees is done for a class assignment. Since the feedback from many trees is considered, the result improves compared to a normal decision



tree. So, in many examples, random forest decision trees have performed very well in case of the mode choice model.

(Refer Slide Time: 15:37)

### Machine learning algorithms in mode choice - Summary

Decision Tree	Neural Network	Fuzzy Model	Hybrid Models	Advanced ML Algorithms
<ul style="list-style-type: none"><li>White box model</li><li>Supervised classification approach</li><li>Chances of over fitting</li><li>Error due to bias and variance both may be present</li></ul>	<ul style="list-style-type: none"><li>Inspired by Human Neural system</li><li>"Black box" model – difficult to interpret results</li></ul>	<ul style="list-style-type: none"><li>Good at dealing with 'vague' / linguistic expression</li><li>Lack of learning ability</li></ul>	<ul style="list-style-type: none"><li>Neuro-fuzzy model: ANN and FIS models complement each other</li><li>Random forest decision tree: Reduce error in DT model due to bias and variance</li></ul>	<ul style="list-style-type: none"><li>Support Vector Machine: Supervised learning model for classification and regression</li></ul>

#### *Decision trees*

- White box models, where one can see what questions are asked.
- It is a supervised classification approach, that is, training the model with the existing data is necessary for making predictions. However, when there is a situation where it has not been trained, the prediction will be difficult.
- There are chances of overfitting of the data.
- Also, error due to bias and variance may be present in a decision tree.

#### *Neural networks*

Blackbox models, ie, it is difficult to interpret the results. The steps are also difficult to identify however this can be adopted if the concern is only about the final outcome.

#### *Fuzzy model*

- It is good at dealing with vague or linguistic expressions and so on.
- There is a lack of learning ability in this kind of model.

#### *Hybrid models*

- Neural fuzzy model, where ANN and FIS models complement each other.
- Random forest decision tree where one can reduce the error in the decision tree model due to bias and variance by increasing the number of trees which are considered.

### *Advanced ML algorithms*

- Support vector machine – here, different decision planes are created with the same set of data and using that, classification is done.
- This is again a supervised learning model for classification and regression.

These are the different machine learning techniques, which has been used in previous research for mode choice models.

### **Decision tree**

**(Refer Slide Time: 17:38)**

**Decision Tree**

- Determination of root value.
- Complex decision trees can be simplified by pruning sub-trees at a selected leaf node. This increases quality of the classification model.

**Data Pruning Method**

- Pre-pruning
- Post-pruning
- Pessimistic pruning

**Pre-pruning** a decision tree involves using a 'termination condition' to decide when it is desirable to terminate some of the branches prematurely as the tree is generated.

**In Post-pruning** decision tree is generated first then non-significant branches are removed.

**In Pessimistic pruning** a sequence of trees are built by removing one rule in each step (lowest error among all possible removals). This results in the smallest tree with lowest error.

NPTEL Online Certification Course  
IIT Kharyatpur

Determination of root value – is the first thing to be done such that one arrives at the question to be asked such that the decision tree splits in the proper. If the top root is ignored then, the decision node becomes the root node for a particular subset of data and again a strategy has to be identified to ask questions.



## Pruning

Complex decision trees can be simplified by pruning sub trees at selected leaf nodes and this increases quality of the classification model. So, a lot of classification or the branches is also not good and so pruning techniques are also followed. There are 3 ways.

- *Pre-pruning* – where a decision tree involves using a termination condition to decide when it is desirable to terminate some of the branches prematurely as the tree is generated.
- *Post-pruning* - decision tree is generated first and then non-significant branches are removed.
- *Pessimistic pruning* – a sequence of trees are built by removing one rule in each step and lowest error among all the possible removals out. This result in the smallest tree with the lowest error.

These are the three ways to prune a tree however it is important how the root value is determined because not all decision trees do pruning.

**(Refer Slide Time: 19:34)**

**Decision Tree**

- ❑ **Calculation of Entropy**
  - ❑ Measure of randomness in the sample.
  - ❑ Its value lies between 0 to 1 (Higher the value of entropy the randomness)
  - ❑ **Entropy calculation:**  
$$\text{Entropy}(\text{set}) = E(S) = -\sum_{i=1}^m p_i \log_2(p_i)$$
  
Where,  
 $p_i = |C_i, D| / |D|$  is calculated with the following parameters;  
 $P_i$ : the probability that an arbitrary tuple in D belongs to class  $C_i$ .  
D: training set of class-labeled tuples(row of a dataset).  
 $C_i$ : the class label attribute has m distinct values for  $i=1, \dots, m$ .  
 $C_i, D$ : the set of tuples of class  $C_i$  in D;  $|D|, |C_i, D|$ : denote the number of tuples in D and  $C_i, D$ .

$E(S) = \text{Info}(D)$

At probability (0.5) entropy = 1

The slide includes a graph of Entropy vs Probability. The x-axis is labeled 'Probability' and ranges from 0.0 to 1.0. The y-axis is labeled 'Entropy' and ranges from 0.0 to 1.0. A curve starts at (0,0), rises to a peak of 1.0 at probability 0.5, and then falls back to 0 at probability 1.0. A dashed vertical line is drawn at probability 0.5, and a horizontal dashed line is drawn from the peak of the curve to the y-axis at 1.0. A small inset box points to the peak with the text 'At probability (0.5) entropy = 1'. In the bottom right corner of the slide, there is a small video inset showing a man in a blue shirt speaking.

## Calculation of entropy

When a decision tree is developed, the decision point has to be identified. Thus there is a need to calculate the entropy. Entropy refers to how the data is split at a decision node or at the root node and the amount of randomness in the sample after the split. The randomness could be measured using a value of 0 to 1; higher the value of entropy, higher is the

randomness. The entropy value could be 0 to 1. From the graph given in the above figure, it can be understood that a probability value of 0.5 corresponds to an entropy value of 1, that means, there is a maximum mix, 50% of one group and 50% of another group. So, when entropy is lower then the probability is also lower. So, the amount of randomness varies with different splits.

Randomness means a mixture of different outcomes. Entropy can be calculated with the following equation.

$$\text{Entropy}(\text{set}) = E(S) = -\sum_{i=1}^m p_i \log_2(p_i)$$

$p_i = |C_i, D| / |D|$  is calculated with the following parameters;

$P_i$ : the probability that an arbitrary tuple in  $D$  which belongs to class  $C_i$ .

$D$ : training set of class-labeled tuples (row of a dataset),

$C_i$ : the class label attribute has  $m$  distinct values for  $i=1, \dots, m$

$C_i, D$ : the set of tuples of class  $C_i$  in  $D$ ;  $|D|, |C_i, D|$ : denote the number of tuples in  $D$  and  $C_i, D$ .

$E(S) = \text{Info}(D)$

Tuple involves all the attributes; it is a row of data. So, there is an outcome and then there are attributes to that particular outcome. And using this, one can determine the probability of this  $C_i, D / D$ . This can be called as information. In the next step, we will determine what is the gain in information. So, first the total entropy that is there for this particular split is determined.

**(Refer Slide Time: 22:51)**

**Decision Tree**

□ **Calculation of Information Gain**  
 The parameter which has the biggest value of information gain become root node.  
**Calculation:**  
 $\text{Info } A | D = \sum_{j=1}^v (|D_j| / |D|) * \text{Info}(D_j)$   
 The gain of an interested element  $A$  is calculated by:  $\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$   
 $\text{Info}_A(D)$  denotes the entropy of  $A$ ; where  $A$ : the element of  $D$  training set;  $|D_j| / |D|$ : the weight of the  $j^{\text{th}}$  partition.  $v$ : number of distinct values for element  $A$  for  $j=1 \dots v$   
 This process is iterated by excluding the newly assign root node at each step until last and bottom leaf parameter becomes target value.

□ **Gain Ratio**  
 • Gain ratio is used for partitioning in practical application.  
 • The attribute with the maximum gain ratio is selected as the splitting attribute.  
**Calculation:**  $\text{Gain Ratio}(A) = \text{Gain}(A) / \text{SplitInfo}(A)$   
 $\text{SplitInfo}(A) = \sum_{j=1}^v (|D_j| / |D|) * \log_2(|D_j| / |D|)$

MPTIL Online Certification Course  
 IIT Kharagpur

### Calculation of information gain

For calculating the information gain, the parameter which has got the biggest value of information gain is selected and considered as the root node. In the next step, for whatever data is got in the split, another parameter becomes root node for that particular subgroup and using the same process is done repeatedly and iteratively. This process is iterated by excluding the newly assigned root nodes at each step until the last bottom leaf parameter becomes the target value. So, the total amount of information that is gained is estimated. Info A(D), which is the gain of an interested element A (the candidate for splitting the particular decision tree) is calculated. So, one can use any parameter to split the data to get the final outcome based on the likely information gain. Consider multiple characteristics such as income and car ownership. The selection of a parameter to split the data at first has to be based on which results in more information gain. So, when income is used, which is the interested element A, the total amount of information gain has to be calculated, which is info D.

$$\text{Info A (D)} = \sum_{j=1}^v (|D_j| / |D|) * \text{info (D}_j)$$

The gain of an interested element A is calculated by:  $\text{Gain (A)} = \text{Info (D)} - \text{Info}_A(\text{D})$

$\text{Info}_A(\text{D})$  denotes the entropy of A ; where A: the element of D training set ;

$|D_j|/|D|$  : the weight of the  $j^{\text{th}}$  partition.

V: number of distinct values for element A for  $j=1 \dots v$

This process is iterated by excluding the newly assigned root node at each step until the last and bottom leaf parameter becomes target value. Suppose income has 3 categories, the data can be divided into 3 different groups. So, each group has a mixture of positives and negatives and then for each split, the information gain can be calculated. Once the information gain is calculated, the parameter which shows the highest information gain can be used to split the data. Instead of determining information gain, the parameter gain ratio can also be used.

### Gain Ratio

Gain ratio is used also for partitioning in practical applications. The attribute with the maximum gain ratio is selected as the splitting attribute and calculation of gain ratio is:

$$\text{Gain Ratio (A)} = \text{Gain (A)} / \text{Splitinfo (A)}$$

$$\text{Splitinfo (A)} = \sum_{j=1}^v (|D_j| / |D|) * \log_2 (|D_j| / |D|)$$

(Refer Slide Time: 27:07)

**Decision Tree**

- **Gini Impurity**
  - Measure impurity in the classified outcome like entropy
  - **Gini Index Calculation:**  $I_G = 1 - \sum_{j=1}^c p_j^2$ ;
  - $p_j$ : proportion of the samples that belongs to class c for a particular node
- Both entropy calculation and Gini Impurity help to determine good split point for root/decision nodes on classification/regression trees

The diagram shows a 2D scatter plot of data points (orange circles and green triangles) separated by a diagonal decision boundary. The region above the boundary is labeled 'Low Impurity' and the region below is labeled 'High Impurity'. Red checkmarks and a minus sign are next to these labels. The slide footer includes logos for IIT Kharagpur and NPTEL.

### Gini impurity

This measures the impurity in the classified outcome like entropy. Gini index calculation is as follows:

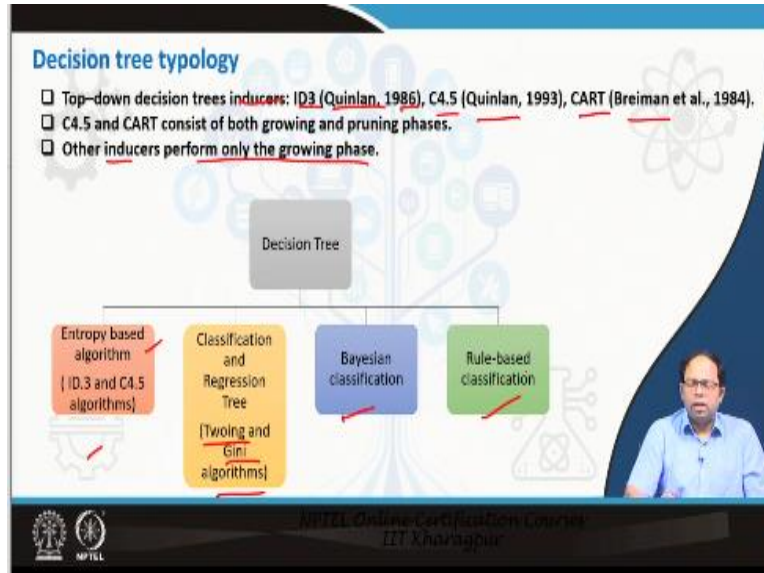
$$I_G = 1 - \sum_{j=1}^c p_j^2$$

$p_j$ : proportion of the samples that belongs to class c for a particular node

Both entropy calculation and Gini impurity help to determine a good split point for root decision nodes on classification regression trees. The graph given in the above figure represents the final outcome. It can be noted that on one side there is low impurity and the other side there is high impurity. The low impurity side of the decision plane or division window would have higher information gain. Overall information gain is the summation of gain on both sides of the decision plane multiplied with the proportion of data points that are there. So, these processes help to decide on the split and identifies which parameter to be considered at the beginning as well as subsequent points.

## Decision tree typology

(Refer Slide Time: 28:35)



Decision tree are of different types such as

- Top-down decision trees inducers or algorithms;
- ID3 algorithm introduced by Quinlan in 1986.
- C4.5 algorithm by Quinlan in 1993.
- CART algorithm by Breiman in 1984 and

C4.5 and CART consist of both growing and pruning phases, unlike other algorithms, which can perform only the growing phase.

Decision trees could be built in different ways. It could be using

Entropy based algorithms like ID3 and C4.5

Twoing and Gini algorithms which come under Classification and regression tree

Bayesian classification

Rule based classification which refers to building the decision tree based on that means based on certain decision rules.

## Mode choice model using Decision tree and Python

(Refer Slide Time: 29:58)

The slide features a blue header with the title 'Mode choice model using Decision Tree and Python'. Below the title, there are three sections: 'Problem statement', 'Algorithm', and 'Software'. The 'Problem statement' section describes a mode choice model for workers, mentioning parameters like monthly household income level and car ownership, and lists three income categories: 'Higher' (>50000 INR), 'Middle' (20000 - 50000 INR), and 'Lower' (<20000 INR). It also states that the number of car ownership is 0 if no vehicle is owned, and otherwise, the number of vehicles owned is mentioned. Available modes are car, bus, and auto, and 70% of data is used for training while 30% is used for testing. The 'Algorithm' section states that the CART algorithm is used. The 'Software' section lists Python 3.8.7 and Jupyter notebook. On the right side of the slide, there is a small video inset of a man in a blue shirt. At the bottom, there are logos for NPTEL and IIT Kharagpur.

### *Problem statement*

- Mode choice model for workers is being developed.
- Two parameters have been selected, monthly household income level and car ownership. Income level is divided into 3 categories; higher (>50,000 INR), middle (20,000 to 50,000 INR) and lower (<20,000 INR).
- Number of car ownership: 0 is used in case no vehicle ownership; else the number of vehicles owned is mentioned.
- Available modes are car, bus and auto-rickshaw.
- 70% data is used for training and 30% is used for testing. In any kind of supervised learning algorithms, one has to use some of the data to first train the model and based on that, all the decision window, decision criteria can be created. Then, it can also be used to test that if a model is performing properly. Certain testing criteria as well as validation criteria are used to test if the model is giving a good result. For example, a confusion matrix can be used to determine the final outcome; whether the prediction by the model is correct.

*Algorithm* - CART algorithm is used; it is available in Python

*Software* - Python 3.8.7 and Jupyter Notebook is used.

(Refer Slide Time: 31:27)




### Example of Decision Tree - Python

The hypothetically generated sample data is as follows :

Sample No	No. of car owned	Monthly Household Income (INR)	Level of Income	Mode choice for work trip
11	0	85000	Higher	Auto
12	0	15000	Lower	Auto
13	1	45000	Middle	Car
14	1	40000	Middle	Auto
15	0	11000	lower	Bus
16	0	20000	Middle	Bus
17	2	90000	Higher	Car
18	1	39000	Middle	Auto
19	1	42000	Middle	Car
20	1	85000	Higher	Auto

■ Training Data  
■ Testing Data

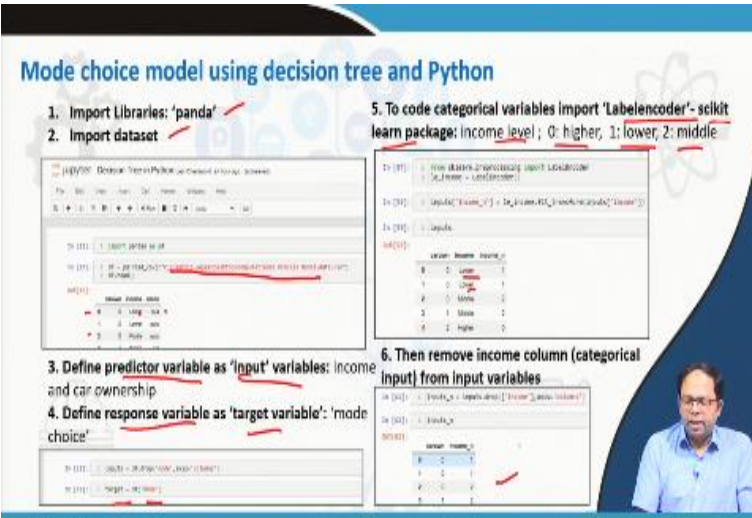


The table given in the above figure represents the hypothetically generated sample data. Details regarding car ownership, monthly household income and level of income, and mode choices for work trip is given. Here, income is converted into categorical variables. Mode choice has 3 classes. The rows up to 14 constitute the training data and then the rest forms the testing data and is differentiated with colour coding for the small data set of 20 samples. Both qualitative data as well as quantitative data can be used in such an analysis.

(Refer Slide Time: 32:18)

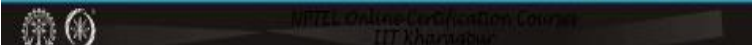
### Mode choice model using decision tree and Python

1. Import Libraries: 'panda'
2. Import dataset
3. Define predictor variable as 'input' variables: income and car ownership
4. Define response variable as 'target variable': 'mode choice'
5. To code categorical variables import 'Labelencoder' - scikit learn package: Income level; 0: higher, 1: lower, 2: middle
6. Then remove income column (categorical input) from input variables



In [1]: from sklearn.preprocessing import LabelEncoder  
 In [2]: le = LabelEncoder()  
 In [3]: le.fit\_transform(X['income'])  
 In [4]: le.classes\_

Out[4]: array(['higher', 'lower', 'middle'], dtype=object)



After Python is opened, the Jupyter Notebook can be imported; The panda library can be imported in code, then the data set can be imported. The different input variables such as

vehicle ownership, income and the output can be seen in the window shown in the above figure. Next step is to define the predictor variable (input variable) and the response variable (target variable). Also, the categorical variables can be coded with 'labelencoder' in scikit learn package. After using the conversion, the data set represents income as shown in the window given in the above figure. Also, some of the categorical variables can be converted into dummy variables.

(Refer Slide Time: 34:09)

**Mode choice model using decision tree and Python**

**7. Import Libraries: 'decision tree classifier'**

```
In [101]: from sklearn import tree
Out[101]: tree
In [102]: from sklearn.preprocessing import
```

**8. Model fitting : 'gini impurity' calculated**

```
In [104]: model = DecisionTreeClassifier(criterion='gini', max_depth=None,
Out[104]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
max_features=None, max_leaf_nodes=None,
min_impurity_split=0.01, min_samples_leaf=1,
min_samples_split=1, min_weight_fraction=0.0,
min_weight_fraction_leaf=0.0, presorted=False,
random_state=None, splitter='best')
```

**9. Model score : depicts prediction accuracy; can be increased with more sample size; the higher the complexity of the sample lower the score.**

```
In [105]: model.score(input_data,target)
Out[105]: 0.6
```

**10. prediction result: check for samples**

Sample No.	No. of car owned	Level of Income	Mode choice (actual)	Predicted mode choice
2	0	Lower	Auto	Auto
1	0	Medium	Auto	Auto
4	1	Medium	Car	Car
5	2	Higher	Bus	Bus
10	0	Higher	Car	Auto

Sample no. 2, 3, 4, 5, 10

NPTEL Online South Indian Institute of Technology Madras

Once the data set is ready, the next is to call the decision tree classifier. And then the model fitting criteria can be given. Many criteria can be used, here, Gini impurity is used for classification. Then, the model score can be generated. Model score depicts the prediction accuracy which can be increased with more sample size. Higher the complexity of the sample, lower is the score. Here, the model score is 0.6. The predictions are done, for instance, for sample number 2, car owned is 0; income level is lower, mode choice is auto and it is predicted correctly. Other predictions are done as shown in the table above. Most of the predictions are correctly done. This explains how decision trees can be developed using simple code with existing algorithms such as the built-in CART algorithm.

## Random Forest Decision Tree (RFDT)

(Refer Slide Time: 35:45)

**Random Forest Decision Tree (RFDT)**

- It is an ensemble technique
- It has the ability to limit overfitting without substantially increasing error due to bias
- More accurate and stable prediction
- If features weren't chosen randomly, base trees in our forest could become highly correlated

Decision Tree 1

Decision Tree n

Assume: Available modes are Bus, auto, car

Root Node, Decision Node, Leaf Node

Majority Voting from 'n' DT → Final RFDT

The slide illustrates the Random Forest Decision Tree (RFDT) concept. It features two decision trees, labeled 'Decision Tree 1' and 'Decision Tree n'. Both trees start with a root node labeled 'income'. Decision Tree 1 splits 'income' into 'Lower' and 'Middle'. The 'Lower' branch leads to a decision node 'Car owned?'. If 'No', it leads to a leaf node 'Car'. If 'Yes', it leads to a leaf node 'Auto/Rick'. The 'Middle' branch leads to a leaf node 'Bus'. Decision Tree n splits 'income' into 'Lower' and 'Higher'. The 'Lower' branch leads to a leaf node 'Bus'. The 'Higher' branch leads to a decision node 'Car owned?'. If 'Yes', it leads to a leaf node 'Car'. If 'No', it leads to a leaf node 'Auto/Rick'. A legend at the bottom identifies 'Root Node' (black circle), 'Decision Node' (blue rectangle), and 'Leaf Node' (green rectangle). An orange arrow labeled 'Majority' points from the trees to a box labeled 'Final RFDT', with the text 'Voting from 'n' DT' below it. The slide also includes the NPTEL logo and the text 'NPTEL Online Certification Course IIT Khargpur'.

- It is an ensemble technique.
- It can limit over-fitting without substantially increasing error due to bias
- More accurate and stable prediction compared to standard decision trees.
- If features were not chosen randomly, base trees in the forest could become highly correlated.

Features are randomly selected as shown in the decision trees in the above figure. For instance, in the decision tree one, the income is split into lower and middle categories whereas for the second decision tree, it is split into lower and higher and so on. Hence, the decision criteria are randomly chosen. Then, many trees are automatically developed and the majority voting is considered to get the final result.

## Model choice using RFDT and Python

(Refer Slide Time: 37:05)

**Mode choice model using RFDT and Python**

1. Import Libraries: 'panda'
2. Import dataset
3. To code categorical variables import 'OneHotEncoder': scikit learn preprocessing package: income level ; 0: higher, 1: lower, 2: middle

```
In [1]: import pandas as pd
In [2]: df = pd.read_csv('Libraries\data\car_data\car_data.csv')
In [3]: df.head()
```

income	choice	mode
2	0	low
1	0	low
2	0	low
1	1	mid
4	0	low

```
In [4]: from sklearn.preprocessing import OneHotEncoder
In [5]: ohe = OneHotEncoder(sparse=False)
In [6]: First_OneHot_Encoder=ohe.fit_transform(df[['income', 'choice', 'mode'])
In [7]: ohe.categories_
Out[7]: [array([0, 1, 2]),
array(['low', 'mid', 'high'], dtype=object),
array(['low', 'mid', 'high'], dtype=object)]
```

NPTEL Online Certification Courses  
IIT Kharagpur

In the Jupyter Notebook, the panda library can be imported. The data set is imported after that. Then, the OneHotEncoder is used to do categorization of the different income categories.

(Refer Slide Time: 37:43)

**Mode choice model using RFDT and Python**

4. Define predictor variable as 'x' variables from created array: income and car ownership
5. Define response variable as 'y' from created array: 'mode choice'
6. Split datasets into test data and train data: consider 80% as training data
7. Import Random Forest classifier
8. Model fit

```
In [8]: x = df[['income', 'choice', 'mode']]
In [9]: y = df['mode']
```

```
In [10]: from sklearn.model_selection import train_test_split
In [11]: X_train, X_test, y_train, y_test = train_test_split(x, y, train_size=0.8)
```

```
In [12]: from sklearn.ensemble import RandomForestClassifier
In [13]: model = RandomForestClassifier()
In [14]: model.fit(X_train, y_train)
```

```
In [15]: from sklearn.metrics import accuracy_score
In [16]: accuracy_score(y_test, model.predict(X_test))
```

income	choice	mode
2	0	low
1	0	low
2	0	low
1	1	mid
4	0	low

NPTEL Online Certification Courses  
IIT Kharagpur

The predictor variables and the response variables are defined. The data set is then divided considering 80% as training data and the rest for testing. The code for such a split can be observed from the above slide as  $x_{train}$   $x_{test}$   $y_{train}$   $y_{test}$ . Once this is done, the random forest classifier is imported and the different criteria are set such as Gini. Other criteria



involves the number of estimators which is 10 here. This indicates the number of random decision trees that will be created.

**(Refer Slide Time: 38:40)**

**7. Model Performance Measurement: run with different estimators (Number of DT) to find the best fit model**

a. Run with 20 estimators : Model score = 0.25

```
In [102]: from sklearn.metrics import random_forest_classifier
model = random_forest_classifier(n_estimators=20)
OUT[102]: RandomForestClassifier(n_estimators=20)
In [103]: model.score(X_test, y_test)
OUT[103]: 0.25
```

b. Run with 30 estimators : Model score = 0.25

```
In [102]: from sklearn.metrics import random_forest_classifier
model = random_forest_classifier(n_estimators=30)
OUT[102]: RandomForestClassifier(n_estimators=30)
In [103]: model.score(X_test, y_test)
OUT[103]: 0.25
```

c. Run with 10 estimators : Model score = 0.75, Highest one

```
In [100]: from sklearn.metrics import random_forest_classifier
model = random_forest_classifier(n_estimators=10)
OUT[100]: RandomForestClassifier(n_estimators=10)
In [101]: model.score(X_test, y_test)
OUT[101]: 0.75
```

d. Run with 5 estimators : Model score = 0.5

```
In [102]: from sklearn.metrics import random_forest_classifier
model = random_forest_classifier(n_estimators=5)
OUT[102]: RandomForestClassifier(n_estimators=5)
In [103]: model.score(X_test, y_test)
OUT[103]: 0.5
```

**8. Model Performance Measurement: Confusion matrix.**

NPTEL Online Certification Course  
ITIT Hyderabad

For 20 estimators, the model score is obtained as 0.25; for 10 estimators, it is 0.75; for 30 estimators it is 0.25 and for 5 estimators, the model score is 0.5. So 10 estimators prove to be the best choice for the given dataset. For bigger datasets, it may be noted that the results improve with the increase in the number of estimators, however, too many estimators may not improve results at times. To measure the model performance, the confusion matrix can be printed to see how many are predicted in the right class

**References**

**(Refer Slide Time: 39:49)**

**REFERENCES**

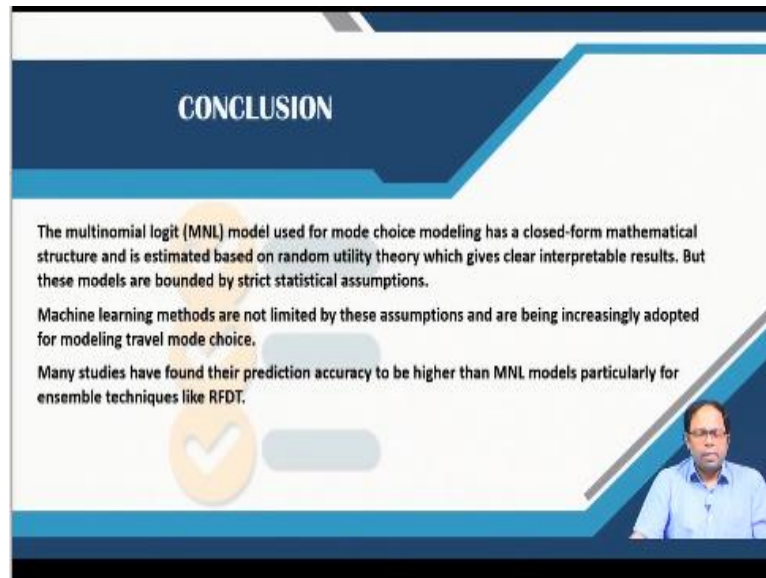
- [https://github.com/random-forests/tutorials/blob/master/decision\\_tree.py](https://github.com/random-forests/tutorials/blob/master/decision_tree.py), accessed on 30<sup>th</sup> May 2020
- [https://github.com/codebasics/py/blob/master/ML/11\\_random\\_forest/11\\_random\\_forest.ipynb](https://github.com/codebasics/py/blob/master/ML/11_random_forest/11_random_forest.ipynb), accessed on 30<sup>th</sup> May 2020
- Cantarella, G. E., & De Luca, S. (2003). Modeling transportation mode choice through artificial neural networks. 4th International Symposium on Uncertainty Modeling and Analysis, ISUMA 2003. <https://doi.org/10.1109/ISUMA.2003.1236145>
- Hagenauer, J., & Helbich, M. (2017). A comparative study of machine learning classifiers for modeling travel mode choice. Expert Systems with Applications. <https://doi.org/10.1016/j.eswa.2017.01.057>
- Omrani, H. (2015). Predicting travel mode of individuals by machine learning. Transportation Research Procedia. <https://doi.org/10.1016/j.trpro.2015.09.037>
- Sekhar, C. R., Minal, & Madhu, E. (2016). Mode Choice Analysis Using Random Forrest Decision Trees. Transportation Research Procedia. <https://doi.org/10.1016/j.trpro.2016.11.119>

NPTEL Online Certification Course  
ITIT Hyderabad

The first reference can give access to elaborate codes and using it one can you also draw the decision trees for the outcome. The actual structure of the tree can be seen apart from the prediction of the final outcome.

## Conclusion

(Refer Slide Time: 40:24)



**CONCLUSION**

The multinomial logit (MNL) model used for mode choice modeling has a closed-form mathematical structure and is estimated based on random utility theory which gives clear interpretable results. But these models are bounded by strict statistical assumptions.

Machine learning methods are not limited by these assumptions and are being increasingly adopted for modeling travel mode choice.

Many studies have found their prediction accuracy to be higher than MNL models particularly for ensemble techniques like RFDT.

- The multinomial logit model used for mode choice modelling has a closed form mathematical structure and is estimated based on random utility theory which gives clear interpretable results, but these models are bounded by the strict statistical assumptions.
- Machine learning methods are not limited by these assumptions and are increasingly adopted for modelling travel mode choice.
- Many studies have found their prediction accuracy to be higher than MNL models particularly for ensemble techniques like RFDT.

Thank you.