DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

III BTECH I SEMESTER 2021-22

DATA ANALYTICS UNIT-IV

Object Segmentation

- Object segmentation is the process of splitting up an object into a collection of smaller fixed-size objects in order to optimize storage and resources usage for large objects.

Regression vs. Segmentation

Regression analysis focuses on finding a relationship between a dependent variable and one or more independent variables.

- ✓ Predicts the value of a dependent variable based on the value of at least one independent variable.
- ✓ Explains the impact of changes in an independent variable on the dependent variable.
- ✓ We use linear or logistic regression technique for developing accurate models for predicting an outcome of interest. Often, we create separate models for separate segments.

Segmentation methods such as CHAID or CRT is used to judge their effectiveness

- ✓ Creating separate model for separate segments may be time consuming and not worth the effort.
- ✓ But, creating separate model for separate segments may provide higher predictive power.

How to create segments for model development?

- ✓ Commonly adopted methodology
- ✓ Let us consider an example.
- ✓ Here we'll build a logistic regression model for predicting likelihood of a customer to respond to an offer.
- ✓ A very similar approach can also be used for developing a linear regression model.

Logistic regression uses 1 or 0 indicator in the historical campaign data, which indicates whether the customer has responded to the offer or not.

- ✓ Usually, one uses the target (or 'Y' known as dependent variable) that has been identified for model development to undertake an objective segmentation.
- ✓ Remember, a separate model will be built for each segment.
- ✓ A segmentation scheme which provides the maximum difference between the segments with regards to the objective is usually selected.

What is Unsupervised Learning?

Unsupervised learning is a machine learning technique, where you do not need to supervise the model. Instead, you need to allow the model to work on its own to discover information. It mainly deals with the unlabelled data.

Unsupervised learning algorithms allow you to perform more complex processing tasks compared to supervised learning. Although, unsupervised learning can be more unpredictable compared with other natural learning deep learning and reinforcement learning methods.

Why Supervised Learning?

- Supervised learning allows you to collect data or produce a data output from the previous experience.
- Helps you to optimize performance criteria using experience
- Supervised machine learning helps you to solve various types of real-world computation problems.

Why Unsupervised Learning?

Here, are prime reasons for using Unsupervised Learning:

- Unsupervised machine learning finds all kind of unknown patterns in data.
- Unsupervised methods help you to find features which can be useful for categorization.
- It is taken place in real time, so all the input data to be analyzed and labeled in the presence of learners.
- It is easier to get unlabeled data from a computer than labeled data, which needs manual intervention.

How Supervised Learning works?

For example, you want to train a machine to help you predict how long it will take you to drive home from your workplace. Here, you start by creating a set of labeled data. This data includes

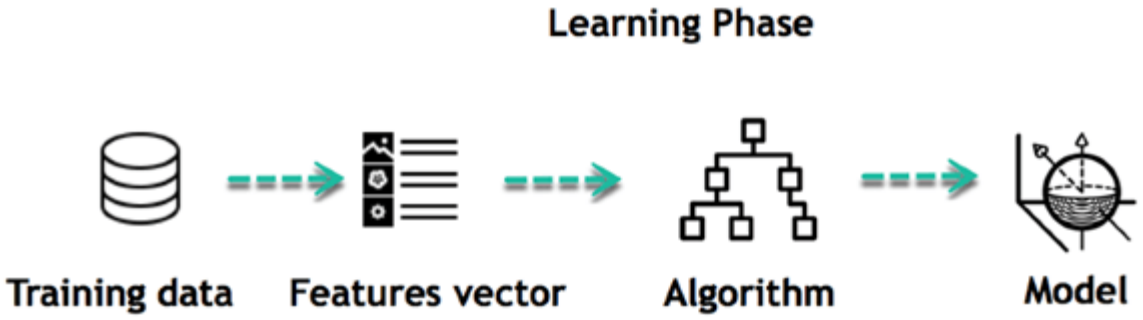
- Weather conditions
- Time of the day
- Holidays

All these details are your inputs. The output is the amount of time it took to drive back home on that specific day.You instinctively know that if it's raining outside, then it will take you longer to drive home. But the machine needs data and statistics.

Let's see now how you can develop a supervised learning model of this example which help the user to determine the commute time. The first thing you requires to create is a training data set. This training set will contain the total commute time and corresponding factors like weather, time, etc. Based on this training set, your machine might see there's a direct relationship between the amount of rain and time you will take to get home.

So, it ascertains that the more it rains, the longer you will be driving to get back to your home. It might also see the connection between the time you leave work and the time you'll be on the road.

The closer you're to 6 p.m. the longer time it takes for you to get home. Your machine may find some of the relationships with your labeled data.

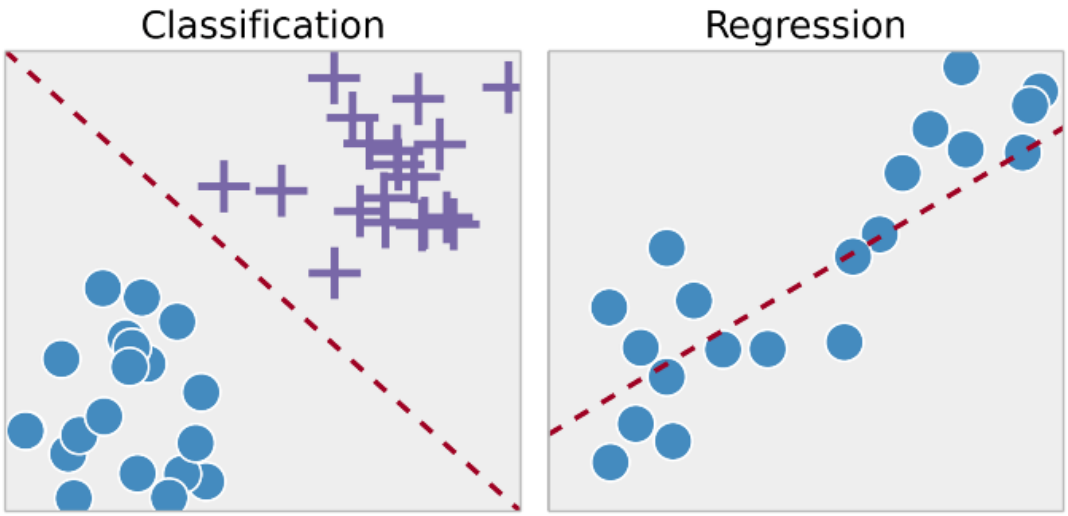


This is the start of your Data Model. It begins to impact how rain impacts the way people drive. It also starts to see that more people travel during a particular time of day.

How Unsupervised Learning works?

Let's, take the case of a baby and her family dog. She knows and identifies this dog. A few weeks later a family friend brings along a dog and tries to play with the baby. Baby has not seen this dog earlier. But it recognizes many features (2 ears, eyes, walking on 4 legs) are like her pet dog. She identifies a new animal like a dog. This is unsupervised learning, where you are not taught but you learn from the data (in this case data about a dog.) Had this been supervised learning, the family friend would have told the baby that it's a dog.

Types of Supervised Machine Learning Techniques



Regression:

Regression technique predicts a single output value using training data.
Example: You can use regression to predict the house price from training data. The input variables will be locality, size of a house, etc.

Classification:

Classification means to group the output inside a class. If the algorithm tries to label input into two distinct classes, it is called binary classification. Selecting between more than two classes is referred to as multiclass classification.

Example: Determining whether or not someone will be a defaulter of the loan.

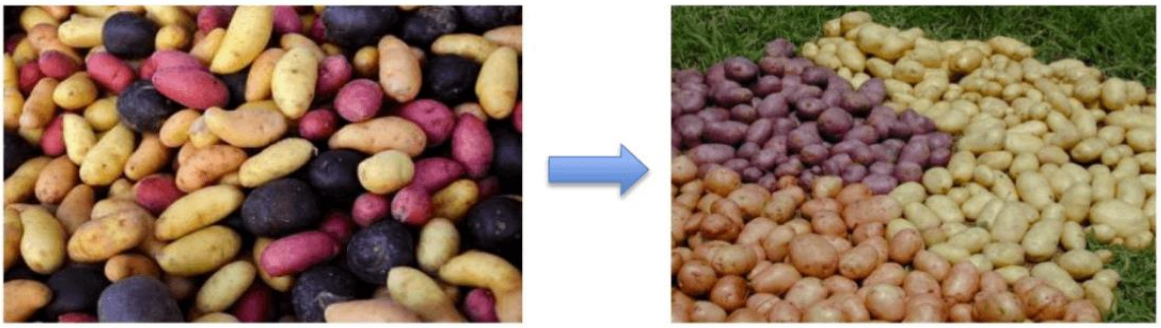
Strengths: Outputs always have a probabilistic interpretation, and the algorithm can be regularized to avoid overfitting.

Weaknesses: Logistic regression may underperform when there are multiple or non-linear decision boundaries. This method is not flexible, so it does not capture more complex relationships.

Types of Unsupervised Machine Learning Techniques

Unsupervised learning problems further grouped into clustering and association problems.

Clustering



sample Cluster/group

Clustering is an important concept when it comes to unsupervised learning. It mainly deals with finding a structure or pattern in a collection of uncategorized data. Clustering algorithms will process your data and find natural clusters(groups) if they exist in the data. You can also modify how many clusters your algorithms should identify. It allows you to adjust the granularity of these groups.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Association

Association rules allow you to establish associations amongst data objects inside large databases. This unsupervised technique is about discovering exciting relationships between variables in large databases. For example, people that buy a new home most likely to buy new furniture.

Other Examples:

- A subgroup of cancer patients grouped by their gene expression measurements
- Groups of shopper based on their browsing and purchasing histories
- Movie group by the rating given by movies viewers

Supervised and Unsupervised Learning

There are two broad set of methodologies for segmentation:

- ✓ Objective (supervised) segmentation
- ✓ Non-Objective (unsupervised) segmentation

Objective Segmentation

- ✓ Segmentation to identify the type of customers who would respond to a particular offer.
- ✓ Segmentation to identify high spenders among customers who will use the e-commerce channel for festive shopping.
- ✓ Segmentation to identify customers who will default on their credit obligation for a loan or credit card.

Non-Objective Segmentation

- ✓ Segmentation of the customer base to understand the specific profiles which exist within the customer base so that multiple marketing actions can be personalized for each segment
- ✓ Segmentation of geographies on the basis of affluence and lifestyle of people living in each geography so that sales and distribution strategies can be formulated accordingly.
- ✓ Segmentation of web site visitors on the basis of browsing behavior to understand the level of engagement and affinity towards the brand.
- ✓ Hence, it is critical that the segments created on the basis of an objective segmentation methodology must be different with respect to the stated objective (e.g. response to an offer).
- ✓ However, in case of a non-objective methodology, the segments are different with respect to the "generic profile" of observations belonging to each segment, but not with regards to any specific outcome of interest.
- ✓ The most common techniques for building non-objective segmentation are cluster analysis, K nearest neighbor techniques etc.
- ✓ Each of these techniques uses a distance measure (e.g. Euclidian distance, Manhattan distance, Mahalanobis distance etc.)

- ✓ This is done to maximize the distance between the two segments.
- ✓ This implies maximum difference between the segments with regards to a combination of all the variables (or factors).

Supervised vs. Unsupervised Learning

Parameters	Supervised machine learning technique	Unsupervised machine learning technique
Process	In a supervised learning model, input and output variables will be given.	In unsupervised learning model, only input data will be given
Input Data	Algorithms are trained using labelled data.	Algorithms are used against data which is not labelled
Algorithms Used	Support vector machine, Neural network, Linear and logistics regression, random forest, and Classification trees.	Unsupervised algorithms can be divided into different categories: like Cluster algorithms, K-means, Hierarchical clustering, etc.
Computational Complexity	Supervised learning is a simpler method.	Unsupervised learning is computationally complex
Use of Data	Supervised learning model uses training data to learn a link between the input and the outputs.	Unsupervised learning does not use output data.
Accuracy of Results	Highly accurate and trustworthy method.	Less accurate and trustworthy method.
Real Time Learning	Learning method takes place offline.	Learning method takes place in real time.
Number of Classes	Number of classes is known.	Number of classes is not known.
Main Drawback	Classifying big data can be a real challenge in Supervised Learning.	You cannot get precise information regarding data sorting, and the output as data used in unsupervised learning is labelled and not known.

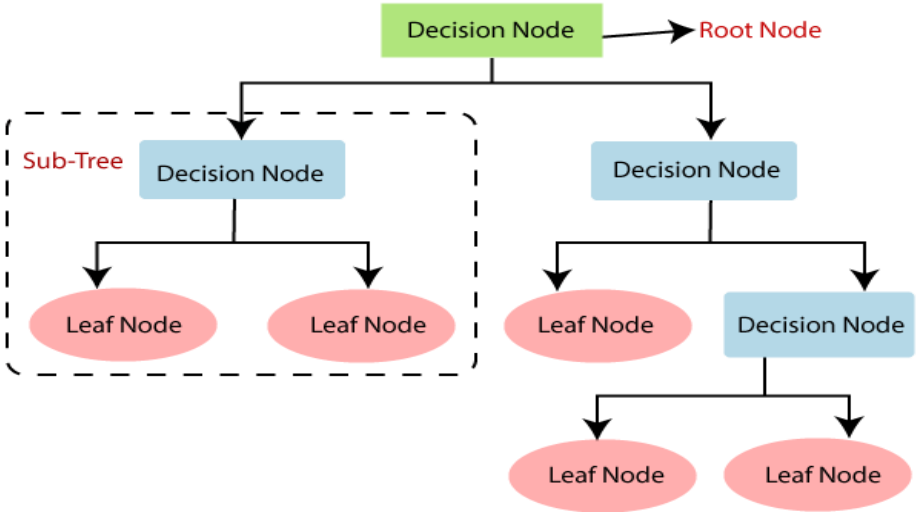
Decision Tree

- Decision Tree is a **supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-

structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**

- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:

Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

- ✓ **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- ✓ **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- ✓ **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- ✓ **Branch/Sub Tree:** A tree formed by splitting the tree.
- ✓ **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- ✓ **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

How does the Decision Tree algorithm Work?

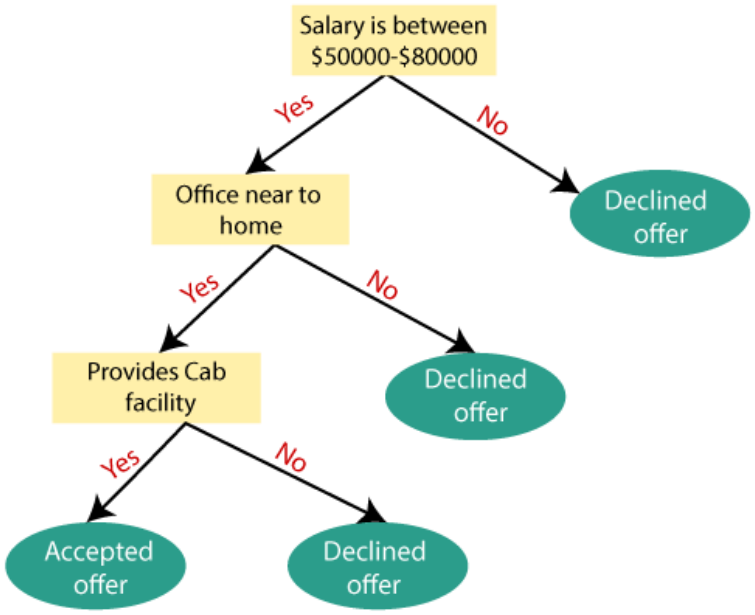
In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and

one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

To understand the concept of a Decision Tree, consider the below example.

Let's say on a particular day we want to play tennis, say Monday. How do you know whether or not to play? Let's say you go out to check whether it's cold or hot, check the pace of wind and humidity, what the weather is like, i.e., sunny, snowy, or rainy. To decide whether you want to play or not, you take into consideration all these variables.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Table 1: Weather Data: Play or not Play?

Now, to determine whether to **play or not**, you will use the table. What if Monday's weather pattern doesn't follow all of the rows in the chart? Maybe that's a concern. A decision tree can be a perfect way to represent data like this. When adopting a tree-like structure, it considers all possible directions that can lead to the final decision by following a tree-like structure.

Why Entropy and Information Gain?

Given a set of data, and we want to draw a Decision Tree, the very first thing that we need to consider is how many attributes are there and what the target class is, whether binary or multi-valued classification.

In the Weather dataset, we have four attributes (**outlook, temperature, humidity, wind**). From these four attributes, we have to select the root node. Once we choose one particular feature as the **root note**, which is the following attribute, we should choose as the next level root and so on. That is the first question we need to answer.

So to answer the particular question, we need to calculate the **Information Gain** of every attribute. **Once we calculate the Information Gain of every attribute, we can decide which attribute has maximum importance. We can select that attribute as the Root Node.**

If we want to calculate the Information Gain, the first thing we need to calculate is entropy. **So given the entropy, we can calculate the Information Gain. Given the Information Gain, we can select a particular attribute as the root node.**

What is Entropy?

Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data.

Entropy measures homogeneity of examples.

Defined over a collection of training data, ***S***, with a Boolean target concept, the **entropy of *S*** is defined as

$$Entropy(S) = p_- \log_2 p_- - p_+ \log_2 p_+$$

where

S is a sample of training examples,

$\mathbf{p_+}$ is the proportion of positive example in S,

$\mathbf{p\_}$ is the proportion of negative examples in S.

How to calculate Entropy?

Entropy([14+, 0-]) = -14/14 $\log_2$(14/14) — 0$\log_2$(0)=0.

Entropy([7+, 7-]) = -7/14 $\log_2$(7/14) — 7/14$\log_2$(7/14)=1.

Entropy([9+, 5-]) = -9/14 $\log_2$(9/14) — 5/14log2(5/14)=0.94.

Important Characteristics of Entropy

- Only positive examples, or only negative examples, Entropy=**0.**
- Equal number of positive & negative example, Entropy=**1.**
- Combination of positive & negative example, use **Formula.**

We have learned how to calculate the entropy for a given data. Now, we will try to understand how to calculate the Information Gain.

Information Gain

Information gain is a measure of the effectiveness of an attribute in classifying the training data.

Given entropy as a measure of the impurity in a collection of training examples, the **information gain** is simply the expected reduction in entropy caused by partitioning the samples according to an attribute.

More precisely, the information gain, ***Gain(S, A)*** of an attribute **A**, relative to a collection of example ***S***, is defined as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|}$$

Where,

S — a collection of examples,

A — an attribute,

Values(A) — possible value of attribute A,

S_v — the subset of S for which attribute A has a value v.

How to find Information Gain?

We will consider the Weather dataset in Table 1. There are four attributes (**outlook, temperature, humidity & wind**) in the dataset, and we need to calculate information gain of all the four attributes.

Here, we will illustrate how to calculate the information gain of wind.

$$\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}$$

In the weather dataset, we only have two classes, ***Weak*** and ***Strong***. There are a total of ***14*** data points in our dataset with ***9*** belonging to the positive class and ***5*** belonging to the negative class.

The entropy here is approximately **0.048.**

This is how; we can calculate the information gain.

Once we have calculated the information gain of every attribute, we can decide which attribute has the maximum importance and then we can select that particular attribute as the root node. We can then start building the decision tree.

Advantages

- Decision trees generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are capable of handling both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

Disadvantages

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many classes and a relatively small number of training examples.
- Decision trees can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. **Pruning algorithms** can also be expensive since many candidate sub-trees must be formed and compared.

Decision-tree algorithms:

- ✓ ID3 (Iterative Dichotomiser 3)
- ✓ C4.5 (successor of ID3)
- ✓ CART (**Classification and Regression Tree**)
- ✓ CHAID (CHI-squared Automatic Interaction Detector). Performs multi-level splits when computing **classification trees**.
- ✓ MARS: extends decision trees to handle numerical data better. Conditional Inference Trees.

- Statistics-based approach that uses non-parametric tests as splitting criteria, corrected for multiple testing to avoid over fitting.
- This approach results in unbiased predictor selection and does not require pruning.
- ID3 and CART follow a similar approach for learning decision tree from training tuples.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Tree-based learning algorithms are considered to be one of the best and mostly used supervised learning methods as they empower predictive models with high accuracy, stability, and ease of interpretation. Unlike linear models, they map non-linear relationships quite well. They are adaptable at solving any kind of problem at hand (classification or regression).

CHAID and CART are the two oldest types of Decision trees. They are also the most common types of Decision trees used in the industry today as they are super easy to understand while being quite different from each other. In this post, we'll learn about all the fundamental information required to understand these two types of decision trees.

CART (Classification And Regression Tree)

- ✓ CART algorithm was introduced in Breiman et al. (1986).
- ✓ A CART tree is a binary decision tree that is constructed by splitting a node into two child nodes repeatedly, beginning with the root node that contains the whole learning sample.
- ✓ The CART growing method attempts to maximize within-node homogeneity.
- ✓ The extent to which a node does not represent a homogenous subset of cases is an indication of impurity.
- ✓ For example, a terminal node in which all cases have the same value for the dependent variable is a homogenous node that requires no further splitting because it is "pure."
- ✓ For categorical (nominal, ordinal) dependent variables the common measure of impurity is Gini, which is based on squared probabilities of membership for each category.
- ✓ Splits are found that maximize the homogeneity of child nodes with respect to the value of the dependent variable.

Impurity Measure:

- ✓ GINI Index Used by the CART (classification and regression tree) algorithm, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset.
- ✓ Gini impurity can be computed by summing the probability fi of each item being chosen times the probability 1-fi of a mistake in categorizing that item.
- ✓ It reaches its minimum (zero) when all cases in the node fall into a single target category.
- ✓ To compute Gini impurity for a set of items, suppose i ε {1, 2... m}, and let fi be the fraction of items labeled with value i in the set.

$$I_G(f) = \sum_{i=1}^{m} f_i(1-f_i) = \sum_{i=1}^{m}(f_i - f_i^2) = \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} f_i^2 = 1 - \sum_{i=1}^{m} f_i^2 = \sum_{i \neq k} f_i f_k$$

Advantages of Decision Tree:

- ✓ ***Simple to understand and interpret***. People are able to understand decision tree models after a brief explanation.
- ✓ ***Requires little data preparation***. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed.
- ✓ ***Able to handle both numerical and categorical data*.** Other techniques are usually specialized in analysing datasets that have only one type of variable.
- ✓ ***Uses a white box model*.** If a given situation is observable in a model the explanation for the condition is easily explained by Boolean logic.
- ✓ ***Possible to validate a model using statistical tests*.** That makes it possible to account for the reliability of the model.
- ✓ ***Robust.*** Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.
- ✓ **Performs well with large datasets.** Large amounts of data can be analyzed using standard computing resources in reasonable time.

Tools used to make Decision Tree:

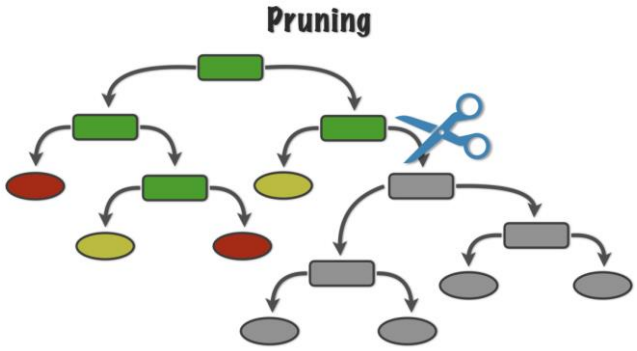
Many data mining software packages provide implementations of one or more decision tree algorithms.

Several examples include:

- o Salford Systems CART
- o IBM SPSS Modeler
- o Rapid Miner
- o SAS Enterprise Miner
- o Matlab
- o R (an open source software environment for statistical computing which includes several CART implementations such as rpart, party and random Forest packages)
- o Weka (a free and open-source data mining suite, contains many decision tree algorithms)
- o Orange (a free data mining software suite, which includes the tree module orngTree)
- o KNIME
- o Microsoft SQL Server
- o Scikit-learn (a free and open-source machine learning library for the Python programming language).

Tree Pruning

It is the process of removal of sub nodes which contribute less power to the decision tree model is called as Pruning. Here we reduce the unwanted branches of the tree which reduces complexity and unwanted branches of the tree and reduces over fitting.



Tree Pruning Approaches

There are two approaches to prune a tree −

Pre-pruning

In this approach we stop growing a branch when the information becomes unreliable. It means it is decided not to further partition the branches. The attribute selection measures are used to find out the weightage of the split. Threshold values are prescribed to decide which splits are regarded as useful. If the portioning of the node results in splitting by falling below threshold then the process is halted

Post pruning

In this approach we first let the tree fully grown and then discard unreliable parts of the branch from it. This technique requires more computation than pre pruning; however, it is more reliable.

Cost Complexity

The cost complexity is measured by the following two parameters −

- Number of leaves in the tree, and
- Error rate of the tree.

Overfitting

Overfitting is a concept in data science, which occurs when a statistical model fits exactly against its training data. When this happens, the algorithm unfortunately cannot perform accurately against unseen data,

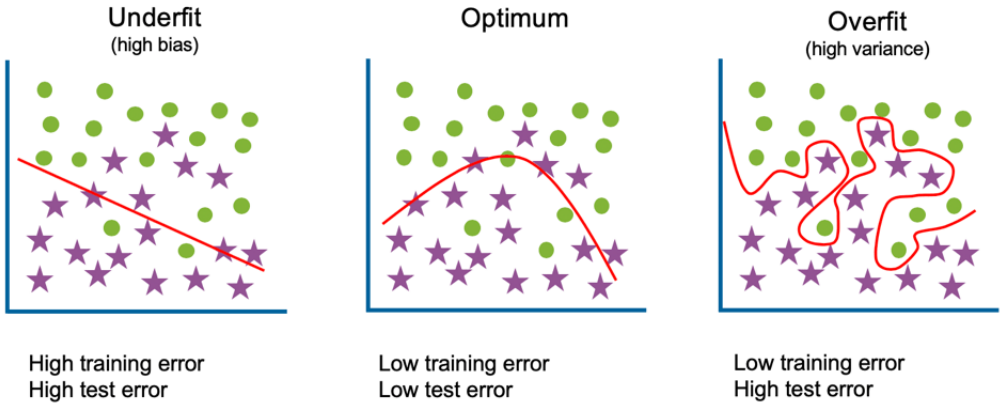
defeating its purpose. Generalization of a model to new data is ultimately what allows us to use machine learning algorithms every day to make predictions and classify data. In reality, the data often studied has some degree of error or random noise within it. Thus, attempting to make the model conform too closely to slightly inaccurate data can infect the model with substantial errors and reduce its predictive power.

When machine learning algorithms are constructed, they leverage a sample dataset to train the model. However, when the model trains for too long on sample data or when the model is too complex, it can start to learn the "noise," or irrelevant information, within the dataset. When the model memorizes the noise and fits too closely to the training set, the model becomes "overfitted," and it is unable to generalize well to new data. If a model cannot generalize well to new data, then it will not be able to perform the classification or prediction tasks that it was intended for.

Low error rates and a high variance are good indicators of overfitting. In order to prevent this type of behavior, part of the training dataset is typically set aside as the "test set" to check for overfitting. If the training data has a low error rate and the test data has a high error rate, it signals overfitting.

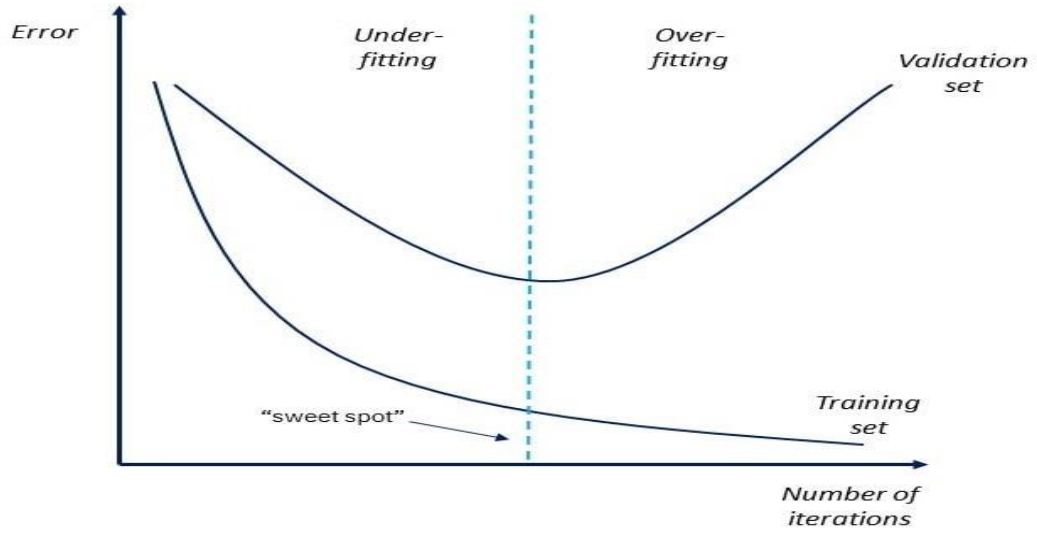
Overfitting Vs. Underfitting

If overtraining or model complexity results in overfitting, then a logical prevention response would be either to pause training process earlier, also known as, "early stopping" or to reduce complexity in the model by eliminating less relevant inputs. However, if you pause too early or exclude too many important features, you may encounter the opposite problem, and instead, you may underfit your model. Underfitting occurs when the model has not trained for enough time or the input variables are not significant enough to determine a meaningful relationship between the input and output variables.



In both scenarios, the model cannot establish the dominant trend within the training dataset. As a result, underfitting also generalizes poorly to unseen data. However, unlike overfitting, underfitted models experience high bias and less variance within their predictions. This illustrates the bias-variance tradeoff, which occurs when as an underfitted model shifted to an overfitted state. As the model learns, its bias reduces, but it can increase in variance as becomes overfitted. When fitting a model, the goal is to find the

"sweet spot" in between underfitting and overfitting, so that it can establish a dominant trend and apply it broadly to new datasets.



How to Avoid Overfitting In Machine Learning?

There are several techniques to avoid overfitting in Machine Learning altogether listed below.

- ✓ Cross-Validation
- ✓ Training With More Data
- ✓ Removing Features
- ✓ Early Stopping
- ✓ Regularization
- ✓ Ensembling

Time Series Analysis

Time series is a sequence of observations of categorical or numeric variables indexed by a date, or timestamp. A clear example of time series data is the time series of a stock price. In the following table, we can see the basic structure of time series data. In this case the observations are recorded every hour.

Timestamp	Stock - Price
2015-10-11 09:00:00	100
2015-10-11 10:00:00	110
2015-10-11 11:00:00	105

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

2015-10-11 12:00:00	90
2015-10-11 13:00:00	120

Normally, the first step in time series analysis is to plot the series, this is normally done with a line chart.

- **Time Series Data** are data points collected over a period of time as a sequence of time gap.
- **Time Series data Analysis** means analyzing the available data to find out the pattern or trend in the data to predict values which will, in turn, help more effective and optimize business decisions.
- Time series data can be found in **economics, social sciences, finance, epidemiology,** and the **physical sciences.**

Field	Example topics	Example dataset
Economics	Gross Domestic Product (GDP), Consumer Price Index (CPI), S&P 500 Index, and unemployment rates	U.S. GDP from the Federal Reserve Economic Data
Social sciences	Birth rates, population, migration data, political indicators	Population without citizenship from Eurostat
Epidemiology	Disease rates, mortality rates, mosquito populations	U.S. Cancer Incidence rates from the Center for Disease Control
Medicine	Blood pressure tracking, weight tracking, cholesterol measurements, heart rate monitoring	MRI scanning and behavioral test dataset
Physical sciences	Global temperatures, monthly sunspot observations, pollution levels.	Global air pollution from the Our World in Data

The most common application of time series analysis is forecasting future values of a numeric value using the temporal structure of the data. This means, the available observations are used to predict values from the future.

The temporal ordering of the data implies that traditional regression methods are not useful. In order to build robust forecast, we need models that take into account the temporal ordering of the data.

The most widely used model for Time Series Analysis is called **Autoregressive Moving Average** (ARMA).

The model consists of two parts,

- an **autoregressive** (AR) part and

- a **moving average** (MA) part.

The model is usually then referred to as the *ARMA (p, q)* model where *p* is the order of the autoregressive part and *q* is the order of the moving average part.

Autoregressive Model

- The *AR(p)* is read as an autoregressive model of order p. Mathematically it is written as –

$$X_t = c + \sum_{i=1}^{P} \phi_i X_{t-i} + \varepsilon_t$$

Where {ϕ_1.. ϕ_p } are parameters to be estimated, c- constant, random variable € representsthe white noise.

Moving Average

- The notation *MA(q)* refers to the moving average model of order *q*

$$X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

Where ϕ_q are the parameters of the model

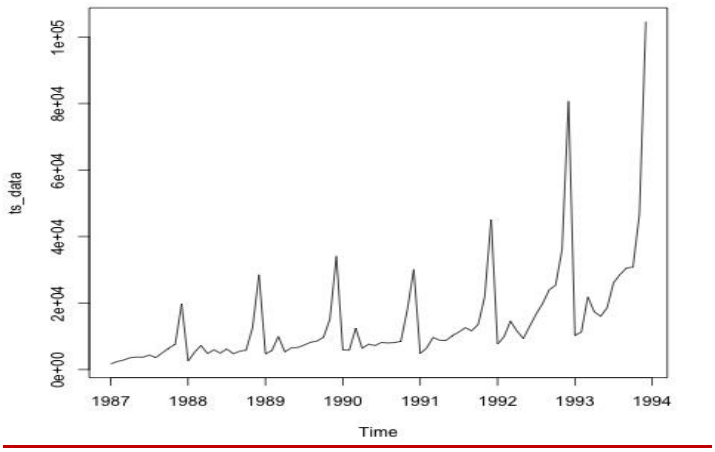
μ is the expectation of X_t and ϵ_t, ϵ_{t-1}.. Are white noise error terms.

Autoregressive Moving Average

The *ARMA(p, q)* model combines p autoregressive terms and q moving-average terms. Mathematically the model is expressed with the following formula

$$X_t = c + \varepsilon_t + \sum_{i=1}^{P} \phi_i X_{t-1} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

-
- Plotting the data is normally the first step to find out if there is a temporal structure in the data. We can see from the plot that there are strong spikes at the end of each year.



Components of Time Series

- **Long term trend** – The smooth long term direction of time series where the data can increase or decrease in some pattern.
- **Seasonal variation** – Patterns of change in a time series within a year which tends to repeat every year.
- **Cyclical variation** – Its much alike seasonal variation but the rise and fall of time series• over periods are longer than one year.
- **Irregular variation** – Any variation that is not explainable by any of the three above mentioned components. They can be classified into – stationary and non – stationary variation.
- When the data neither increases nor decreases, i.e. it's completely random it's called stationary variation.

When the data has some explainable portion remaining and can be analysed further then such case is called non – stationary variation.

ARIMA (Autoregressive Integrated Moving Average)

- ✓ ARIMA model is a generalization of an autoregressive moving average (ARMA) model, in time series analysis,
- ✓ These models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting).
- ✓ They are applied in some cases where data show evidence of non-stationary, wherein initial differencing step (corresponding to the "integrated" part of the model) can be applied to reduce the non-stationary.

Non-seasonal ARIMA models

- ✓ These are generally denoted ARIMA(p, d, q) where parameters p, d, and q are non-negative integers, p is the order of the Autoregressive model, d is the degree of differencing, and q is the order of the Moving-average model.

Seasonal ARIMA models

- ✓ These are usually denoted ARIMA(p, d, q)(P, D, Q)_m, where m refers to the number of periods in each season, and the uppercase P, D, Q refer to the autoregressive, differencing, and moving average terms for the seasonal part of the ARIMA model.

- ✓ ARIMA models form an important part of the Box-Jenkins approach to time-series modeling.

Applications

- ✓ ARIMA models are important for generating forecasts and providing understanding in all kinds of time series problems from economics to health care applications.

In quality and reliability, they are important in process monitoring if observations are correlated.

- ✓ Designing schemes for process adjustment
- ✓ Monitoring a reliability system over time
- ✓ Forecasting time series
- ✓ Estimating missing values
- ✓ Finding outliers and atypical events
- ✓ Understanding the effects of changes in a system

Measure of Forecast Accuracy

- **Forecasting** is the process of making predictions based on past and present data and most commonly by **analysis of trends**. A commonplace **example** might be estimation of some variable of interest at some specified future date.

- **Forecast accuracy** is the deviation of the actual demand from the forecasted demand. If you can calculate the level of **error** in your previous demand **forecasts**, you can factor this into future ones and make the relevant adjustments to your planning.

Why do we need forecast accuracy?

- For measuring accuracy, we compare the existing data with the data obtained by running the prediction model for existing periods. The difference between the actual and predicted value is also known as ***forecast error.*** Lesser the forecast error, the more accurate our model is.

- By selecting the method that is most accurate for the data already known, it increases the probability of accurate future values.

Forecast Accuracy can be defined as the deviation of Forecast or Prediction from the actual results.

Error = Actual demand – Forecast

OR

ei = At – Ft

There are several measures to measure forecast accuracy:

- Mean Forecast Error (MFE)
- Mean Absolute Error (MAE) or Mean Absolute Deviation (MAD)
- Root Mean Square Error (RMSE)
- Mean Absolute Percentage Error (MAPE)

Calculating Forecast Error

The difference between the actual value and the forecasted value is known as ***forecast error***.

The table shows the weekly sales volume of a store.

Note:

- A positive value of forecast error signifies that the model has underestimated the actual value of the period.
- A negative value of forecast error signifies that the model has overestimated the actual value of the period.

	A	B
1	Week	Actual Sales (A)
2	1	17
3	2	21
4	3	24
5	4	17
6	5	23
7	6	18
8	7	14
9	8	15
10	9	22
11	10	20
12	11	23
13	12	25
14	13	18
15	14	20

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Mean Forecast Error (MFE):

- A simple measure of forecast accuracy is the mean or average of the forecast error, also known as **Mean Forecast Error**.
- In this example,

Calculate the average of all the forecast Errors to get mean forecast error:

	A	B	C	D
1	Week	Actual Sales (A)	Forecast (F)	Forecast Error FE = (A - F)
2	1	17		
3	2	21	17	4
4	3	24	21	3
5	4	17	24	-7
6	5	23	17	6
7	6	18	23	-5
8	7	14	18	-4
9	8	15	14	1
10	9	22	15	7
11	10	20	22	-2
12	11	23	20	3
13	12	25	23	2
14	13	18	25	-7
15	14	20	18	2
16				
17			=AVERAGE(D3:D15)	
18				
19				Mean Forecast Error

The MFE for this forecasting method is 0.2.

Mean Absolute Deviation (MAD) or Mean Absolute Error (MAE):

This method avoids the problem of positive and negative forecast errors. As the name suggests, the mean absolute error is the average of the absolute values of the forecast errors.

	A	B	C	D	E
1	Week	Actual Sales (A)	Forecast (F)	Forecast Error FE = (A - F)	Absoulte FE ABS(FE)
2	1	17			
3	2	21	17	4	=ABS(D3)
4	3	24	21	3	3
5	4	17	24	-7	7
6	5	23	17	6	6
7	6	18	23	-5	5
8	7	14	18	-4	4
9	8	15	14	1	1
10	9	22	15	7	7
11	10	20	22	-2	2
12	11	23	20	3	3
13	12	25	23	2	2
14	13	18	25	-7	7
15	14	20	18	2	2
16					
17				0.2	4.08
18					
19				Mean Forecast Error	Mean Absolute Error

MAD for this forecast model is 4.08

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Mean Squared Error (MSE)

- Mean Squared Error also avoids the challenge of positive and negative forecast errors offsetting each other. It is obtained by:

First, calculating the square of the forecast error Then, taking the average of the squared forecast error

	A	B	C	D	E	F
1	Week	Actual Sales (A)	Forecast (F)	Forecast Error FE = (A - F)	Absoulte FE ABS(FE)	FE ^2
2	1	17				
3	2	21	17	4	4	=D3^2
4	3	24	21	3	3	9
5	4	17	24	-7	7	49
6	5	23	17	6	6	36
7	6	18	23	-5	5	25
8	7	14	18	-4	4	16
9	8	15	14	1	1	1
10	9	22	15	7	7	49
11	10	20	22	-2	2	4
12	11	23	20	3	3	9
13	12	25	23	2	2	4
14	13	18	25	-7	7	49
15	14	20	18	2	2	4
16						
17				0.2	4.08	20.85
18						
19				Mean Forecast Error	Mean Absolute Error	Mean Squared Error

Root Mean Squared Error (RMSE):

Root Mean Squared Error is the square root of Mean Squared Error (MSE). It is a useful metric for calculating forecast accuracy.

RMSE for this forecast model is 4.57. It means, on average, the forecast values were 4.57 values away from the actual.

	A	B	C	D	E	F
1	Week	Actual Sales (A)	Forecast (F)	Forecast Error FE = (A - F)	Absoulte FE ABS(FE)	FE ^2
2	1	17				
3	2	21	17	4	4	16
4	3	24	21	3	3	9
5	4	17	24	-7	7	49
6	5	23	17	6	6	36
7	6	18	23	-5	5	25
8	7	14	18	-4	4	16
9	8	15	14	1	1	1
10	9	22	15	7	7	49
11	10	20	22	-2	2	4
12	11	23	20	3	3	9
13	12	25	23	2	2	4
14	13	18	25	-7	7	49
15	14	20	18	2	2	4
16						
17				0.2	4.08	20.85
18						
19				Mean Forecast Error	Mean Absolute Error	Mean Squared Error
20						
21					RMSE	=SQRT(F17)

Mean Absolute Percentage Error (MAPE)

The size of MAE or RMSE depends upon the scale of the data. As a result, it is difficult to make comparisons for a different time interval (such as comparing a method of forecasting monthly sales to a method forecasting a weekly sales volume). In such cases, we use the mean absolute percentage error (MAPE).

Steps for calculating MAPE:

- By dividing the absolute forecast error by the actual value.
- Calculating the average of individual absolute percentage Error.

	A	B	C	D	E	F	G
1	Week	Actual Sales (A)	Forecast (F)	Forecast Error FE = (A - F)	Absoulte FE ABS(FE)	FE ^2	Absolute Percentage Error ABS(FE)/A
2	1	17					
3	2	21	17	4	4	16	=E3/B3
4	3	24	21	3	3	9	13%
5	4	17	24	-7	7	49	41%
6	5	23	17	6	6	36	26%
7	6	18	23	-5	5	25	28%
8	7	14	18	-4	4	16	29%
9	8	15	14	1	1	1	7%
10	9	22	15	7	7	49	32%
11	10	20	22	-2	2	4	10%
12	11	23	20	3	3	9	13%
13	12	25	23	2	2	4	8%
14	13	18	25	-7	7	49	39%
15	14	20	18	2	2	4	10%
16							
17				0.2	4.08	20.85	21%
18							
19				Mean Forecast Error	Mean Absolute Error	Mean Squared Error	Mean Absolute Percentage Error
20							
21					RMSE	4.57	

The MAPE for this model is 21%.

ETL Approach

- Extract, Transform and Load (ETL) refers to a process in database usage and especially in data warehousing that:
  - Extracts data from homogeneous or heterogeneous data sources
  - Transforms the data for storing it in proper format or structure for querying and analysis purpose
  - Loads it into the final target (database, more specifically, operational data store, data mart, or data warehouse)
- Usually all the three phases execute in parallel since the data extraction takes time, so while the data is being pulled another transformation process executes, processing the already received data and prepares the data for loading and as soon as there is some data ready to be loaded into the target, the data loading kicks off without waiting for the completion of the previous phases.
- ETL systems commonly integrate data from multiple applications (systems), typically developed and supported by different vendors or hosted on separate computer hardware.
- The disparate systems containing the original data are frequently managed and operated by different employees.
- For example, a cost accounting system may combine data from payroll, sales, and purchasing.

- Commercially available ETL tools include:
  - ✓ Anatella
  - ✓ Alteryx
  - ✓ CampaignRunner
  - ✓ ESF Database Migration Toolkit
  - ✓ InformaticaPowerCenter
  - ✓ Talend
  - ✓ IBM InfoSphereDataStage
  - ✓ Ab Initio
  - ✓ Oracle Data Integrator (ODI)
  - ✓ Oracle Warehouse Builder (OWB)
  - ✓ Microsoft SQL Server Integration Services (SSIS)
  - ✓ Tomahawk Business Integrator by Novasoft Technologies.
  - ✓ Stambia
  - ✓ Diyotta DI-SUITE for Modern Data Integration
  - ✓ FlyData
  - ✓ Rhino ETL
  - ✓ SAP Business Objects Data Services
  - ✓ SAS Data Integration Studio
  - ✓ SnapLogic
  - ✓ Clover ETL opensource engine supporting only basic partial functionality and not server
  - ✓ SQ-ALL - ETL with SQL queries from internet sources such as APIs
  - ✓ North Concepts Data Pipeline

Various steps involved in ETL.

- ✓ Extract
- ✓ Transform
- ✓ Load

Extract

- ❖ The Extract step covers the data extraction from the source system and makes it accessible for further processing.
- ❖ The main objective of the extract step is to retrieve all the required data from the source system with as little resources as possible.
- ❖ The extract step should be designed in a way that it does not negatively affect the source system in terms or performance, response time or any kind of locking.

There are several ways to perform the extract:

- ✓ Update notification - if the source system is able to provide a notification that a record has been changed and describe the change, this is the easiest way to get the data.
- ✓ Incremental extract - some systems may not be able to provide notification that an update has occurred, but they are able to identify which records have been modified and provide an extract of such records. During further ETL steps, the system needs to identify changes and propagate it down. Note, that by using daily extract, we may not be able to handle deleted records properly.
- ✓ Full extract - some systems are not able to identify which data has been changed at all, so a full extract is the only way one can get the data out of the system. The full extract requires keeping a copy of the last extract in the same format in order to be able to identify changes. Full extract handles deletions as well.
- ✓ When using Incremental or Full extracts, the extract frequency is extremely important. Particularly for full extracts; the data volumes can be in tens of gigabytes.

Clean - The cleaning step is one of the most important as it ensures the quality of the data in the data warehouse. Cleaning should perform basic data unification rules, such as:

- Making identifiers unique (sex categories Male/Female/Unknown, M/F/null, Man/Woman/Not Available are translated to standard Male/Female/Unknown)
- Convert null values into standardized Not Available/Not Provided value
- Convert phone numbers, ZIP codes to a standardized form
- Validate address fields, convert them into proper naming, e.g. Street/St/St./Str./Str
- Validate address fields against each other (State/Country, City/State, City/ZIP code, City/Street).

Transform

- The transform step applies a set of rules to transform the data from the source to the target.
- This includes converting any measured data to the same dimension (i.e. conformed dimension) using the same units so that they can later be joined.
- The transformation step also requires joining data from several sources, generating aggregates, generating surrogate keys, sorting, deriving new calculated values, and applying advanced validation rules.

Load

- During the load step, it is necessary to ensure that the load is performed correctly and with as little resources as possible.

- The target of the Load process is often a database.
- In order to make the load process efficient, it is helpful to disable any constraints and indexes before the load and enable them back only after the load completes.
- The referential integrity needs to be maintained by ETL tool to ensure consistency.

Managing ETL Process

- The ETL process seems quite straight forward.
- As with every application, there is a possibility that the ETL process fails.
- This can be caused by missing extracts from one of the systems, missing values in one of the reference tables, or simply a connection or power outage

STL approach

STL is a versatile and robust method for decomposing time series. STL is an acronym for "**Seasonal and Trend decomposition is using Loess,**" while Loess is a method for estimating nonlinear relationships. The STL method was developed by R. B. Cleveland, Cleveland, McRae, & Terpenning (1990).

STL has several advantages over the classical, SEATS and X11 decomposition methods:

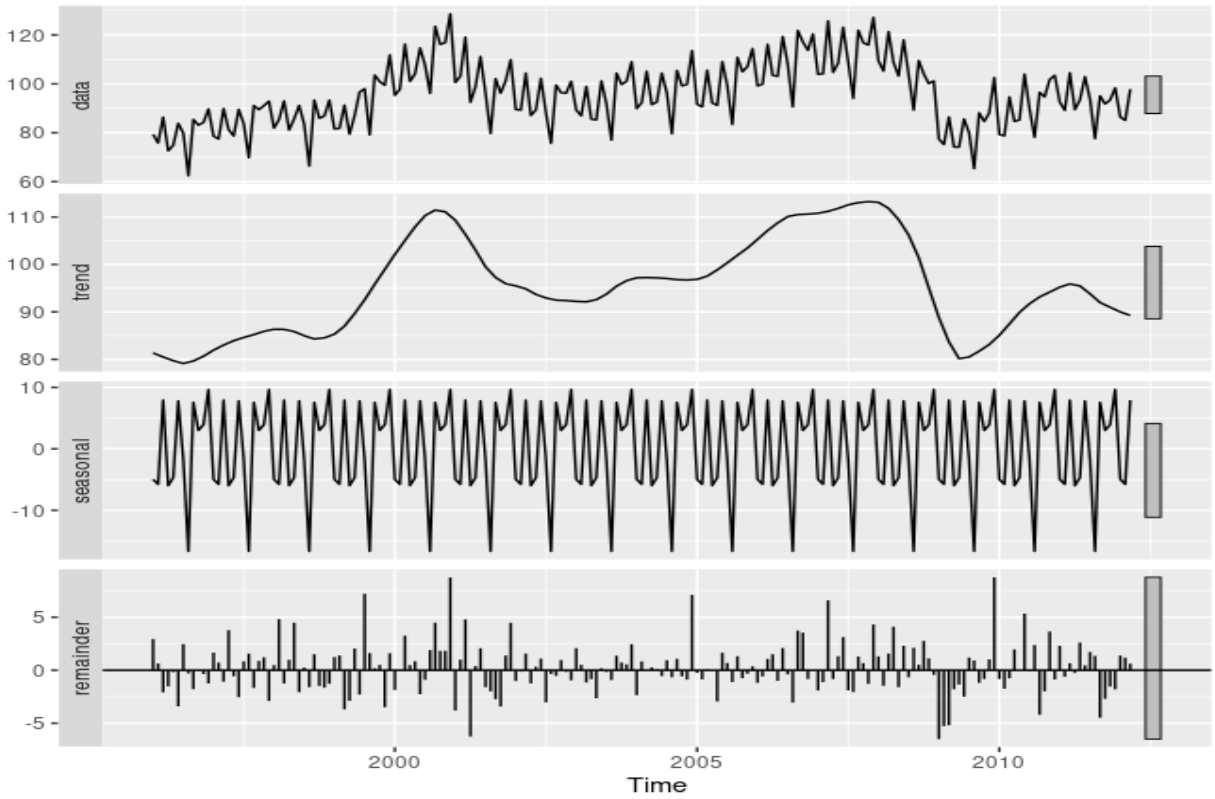
- Unlike SEATS and X11, STL will handle any type of seasonality, not only monthly and quarterly data.
- The seasonal component is allowed to change over time, and the rate of change can be controlled by the user.
- The smoothness of the trend-cycle can also be controlled by the user.
- It can be robust to outliers (i.e., the user can specify a robust decomposition), so that occasional unusual observations will not affect the estimates of the trend-cycle and seasonal components. They will, however, affect the remainder component.

On the other hand, STL has some disadvantages. In particular, it does not handle trading day or calendar variation automatically, and it only provides facilities for additive decompositions.

It is possible to obtain a multiplicative decomposition by first taking logs of the data, then back-transforming the components. Decompositions between additive and multiplicative can be obtained using a Box-Cox transformation of the data with $0<\lambda<10<\lambda<1$. A value of $\lambda=0\lambda=0$ corresponds to the multiplicative decomposition while $\lambda=1\lambda=1$ is equivalent to an additive decomposition.

Figure 6.13 shows an alternative STL decomposition where the trend-cycle is more flexible, the seasonal component does not change over time, and the robust option has been used. Here, it is more obvious that

there has been a down-turn at the end of the series, and that the orders in 2009 were unusually low (corresponding to some large negative values in the remainder component).



The two main parameters to be chosen when using STL are the trend-cycle window (t.window) and the seasonal window (s.window). These control how rapidly the trend-cycle and seasonal components can change. Smaller values allow for more rapid changes.

Both t.window and s.window should be odd numbers; t.window is the number of consecutive observations to be used when estimating the trend-cycle; s.window is the number of consecutive years to be used in estimating each value in the seasonal component. The user must specify s.window as there is no default. Setting it to be infinite is equivalent to forcing the seasonal component to be periodic (i.e., identical across years). Specifying t.window is optional, and a default value will be used if it is omitted.

The mstl() function provides a convenient automated STL decomposition using s.window=13, and t.window also chosen automatically. This usually gives a good balance between overfitting the seasonality and allowing it to slowly change over time. But, as with any automated procedure, the default settings will need adjusting for some time series.

Feature Extraction

Feature extraction deals with the problem of finding the most informative, distinctive, and reduced set of features, to improve the success of data storage and processing.

Important feature vectors remain the most common and suitable signal representation for the classification problems. Numerous scientists in diverse areas, who are interested in data modelling and classification are combining their effort to enhance the problem of feature extraction.

The current advances in both data analysis and machine learning fields made it possible to create a recognition system, which can achieve tasks that could not be accomplished in the past. Feature extraction lies at the center of these advancements with applications in data analysis (Guyon, Gunn, Nikravesh, & Zadeh, 2006; Subasi, 2019).

In *feature extraction*, we are concerned about finding a new set of *k* dimensions, which are combinations of the original *d* dimensions. The widely known and most commonly utilized feature extraction methods are *principal component analysis* and *linear discriminant analysis*, unsupervised and supervised learning techniques. Principal component analysis is considerably similar to two other unsupervised linear methods, *factor analysis* and *multidimensional scaling*. When we have not one but two sets of observed variables, *canonical correlation analysis* can be utilized to find the joint features, which explain the dependency between the two (Alpaydin, 2014).

Conventional classifiers do not contain a process to deal with class boundaries. Therefore, if the input variables (number of features) are big as compared to the number of training data, class boundaries may not overlap. In such situations, the generalization ability of the classifier may not be sufficient. Hence, to improve the generalization ability, usually a small set of features from the original input variables are formed by feature extraction, dimension reduction, or feature selection.

The most efficient characteristic in creating a model with high generalization capability is to utilize informative and distinctive sets of features. Nevertheless, as there is no effective way of finding an original set of features for a certain classification problem, it is essential to find a set of original features by trial and error. If the number of features is so big and every feature has an insignificant effect on the classification, it is more appropriate to transform the set of features into a reduced set of features. In data analysis, raw data are transformed into a set of features by means of a linear transformation.

If every feature in the original set of features has an effect on the classification, the set is reduced by feature extraction, feature selection, or dimension reduction. By feature selection or dimension reduction, ineffective or redundant features are removed in a way that the higher generalization performance and faster classification by the initial set of features can be accomplished.