

CUSTOMER CHURN PROJECT



GROUP MEMBER

ALIPTA BASAK , GOVT. COLLEGE OF ENGINEERING & CERAMIC TECHNOLOGY, REG NO: 171130110040 OF 2017-18

ANTARA RAY, GOVT. COLLEGE OF ENGINEERING & CERAMIC TECHNOLOGY, REG NO: 171130110045 OF 2017-18

BRATATI GHOSH , GOVT. COLLEGE OF ENGINEERING & CERAMIC TECHNOLOGY, REG NO: 171130110052 OF 2017-18

MOHINI DAS , GOVT. COLLEGE OF ENGINEERING & CERAMIC TECHNOLOGY, REG NO: 171130110056 OF 2017-18

Table of Contents

* Acknowledgement
* Project Objective
* Project Scope
* Definition of machine learning
* Classification
* Features
* Real life application
* limitations
* Data Description
* Model Building
* Code
* Output
* Future Scope of Improvements
* Project Certificate

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty (TITAS ROYCHOWDHURY) for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

ALIPTA BASAK

ANTARA RAY

BRATATI GHOSH

MOHINI DAS

(Teacher’s signature)

Project Objective

• DESCRIPTION OF THE PROBLEM :

Customer Churn Analysis

This project gives an opportunity to predict churnes and non-churners of a customer.

* **Data Description**:

Here we use some code which have specific meanings. They are-

st - state

acclen - account length

arcode - area code

phnum - phone number

intplan - internet plan (yes/no)

voice - voice

nummailmes - no of email messages

tdmin - total day messages

tdcal - total day time calls

tdchar - total day time charges

temin - total evening time minutes

tecal - total evening time calls

tecahr - total evening time charges

tnmin - total night time minutes

tn cal - total night time calls

tnchar - total night time charges

timin - total international minutes

tical - total international calls

tichar - total international charges

ncsc - no. of customer services calls

label- churned(True/False)

This is the problem to find whether a customer is churned or not.

**OBJECTIVE** :

Here , in this problem two different data sets are given – one for the train purpose and another one for the test purpose. Train set has 3333 records and 21 features and the test set has 1667 records and 21 features. The main objective of the project is to predict that a customer is churned or not using the information. Here using the train data set we have to make a model which is fit the parameters of the classifier and then test the validation of the model using the test data set. If the model is tested with the test data set then we can conclude that the model is now appropriate to find that a customer is churned or not automatically using necessary information about the customer.

**PLANNING TO SOLVE**

We have planned to solve the problem using machine learning .Now there arise a question that what is machine learning.

* MACHINE LEARNING :

Machine learning(ML) is the scientific study of algorithms and statistical models that computer system use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data , known as “training data”, in order to make predictions or decisions without being explicitly programmed to perform the task . Machine learning algorithms are used in a wide variety of applications , such as email filtering , and computer vision , where it is infeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related computational statistics ,which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning .In this application across business problem ,machine learning is also known as predictive analysis .

* THEORY:

A core objective of a learner is to generalize from its experience. Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of [theoretical computer science](https://en.wikipedia.org/wiki/Theoretical_computer_science) known as [computational learning theory](https://en.wikipedia.org/wiki/Computational_learning_theory). Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The [bias–variance decomposition](https://en.wikipedia.org/wiki/Bias%E2%80%93variance_decomposition) is one way to quantify generalization [error](https://en.wikipedia.org/wiki/Errors_and_residuals).

For the best performance in the context of generalization, the complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, then the model has underfit the data. If the complexity of the model is increased in response, then the training error decreases. But if the hypothesis is too complex, then the model is subject to [overfitting](https://en.wikipedia.org/wiki/Overfitting) and generalization will be poorer.

In addition to performance bounds, learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in [polynomial time](https://en.wikipedia.org/wiki/Time_complexity#Polynomial_time). There are two kinds of [time complexity](https://en.wikipedia.org/wiki/Time_complexity) results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

* CHARECTERISTICS :

1. Branch of Artificial Intelligence
2. A system that can learn from data
3. Predict / classify data after learning
4. Generalize – the ability to classify / predict unseen data set

* WHY MACHINE LEARNING :
* Automatically adapt and customize to individual users.

*(Personalized news ,mail filters, movie/ book recommendation)*

* Discover new knowledge from huge amount of data

(*Market analysis*)

* Perform repetitive monotonous tasks of humans which requires intelligence and experience

(*Recognize signatures or hand -written characters*

*Driving a car , flying a plane*)

* Rapidly changing phenomena

(*credit scoring, financial modelling, diagnosis, fraud detection*)

* No human experts

(*industrial / manufacturing control , mass spectrometer analysis, drug design*)

* CONCEPT OF MACHINE LEARNING :
* Learning = improve task “T” with respect to performance measure “P” based on experience “E”
* Example : Spam Filtering

1. T : identity spam emails
2. P : % of spam emails filtered correctly , % of non spam emails that were filtered incorrectly(False Positive)
3. E : database of emails labelled manually by users

* Example : signature matching

1. T : determine if signature belongs to correct person
2. P : % of signatures that were correctly matched , % of valid signatures that were incorrectly labelled as not matching
3. E : database of signatures known to be of that person

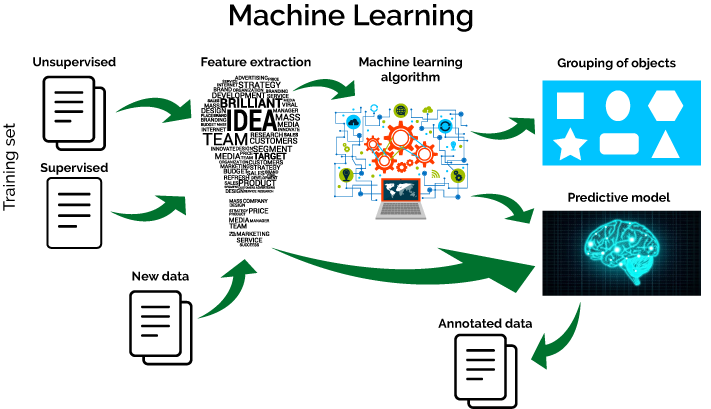


IMAGE OF MACHINE LEARNING CONCEPT

**DIMENSION OF LEARNING SYSTEM**

* Type of feedback

1)supervised(labelled examples)

2)unsupervised( unlabelled examples)

3)reinforcement(reward - based)

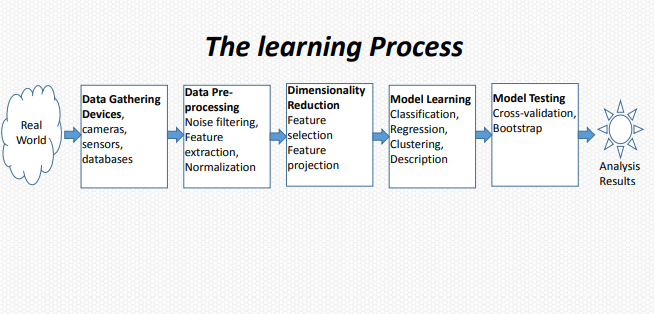
* Representation

1. Attribute – test(feature vector)
2. Relational (first – order logic)

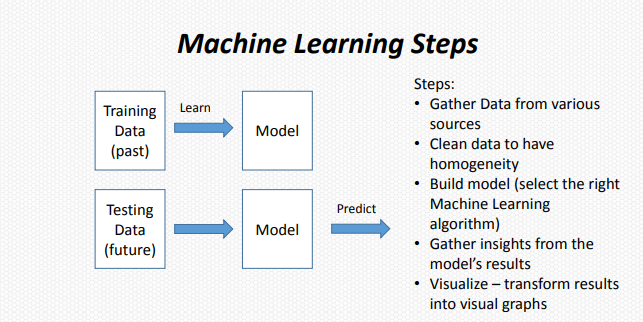
* Use of knowledge

1. Empirical(knowledge - free)
2. Analytical(knowledge – gui

**THE LEARNING PROCESS**



**MACHINE LEARNING STEPS**



**PROJECT SCOPE**

Customer churn analysis is an approach to maintain the interaction between the company and the old and potential customers.

This project helps the telecommunication company to understand which customers tend to churn. They can also recognise the non-churners.

This should help them to gain more profit.

In future they can add more features in the data-sets to predict more efficiently.

*PERFORMANCE EVALUATION*

* Randomly split examples into training set U and test set V.
* Used training set to learn a hypothesis H.
* Measure % of V correctly classified by H.
* Repeat for different random split and average results.

PROBLEMS OF OVERFITTING AND UNDERFITTING

* **OVERFITTING**:

Overfitting refers to a model that models the training data too well.

Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize.

Overfitting is more likely with nonparametric and nonlinear models that have more flexibility when learning a target function. As such, many nonparametric machine learning algorithms also include parameters or techniques to limit and constrain how much detail the model learns.

For example, decision trees are a nonparametric machine learning algorithm that is very flexible and is subject to overfitting training data. This problem can be addressed by pruning a tree after it has learned in order to remove some of the detail it has picked up.

* **UNDERFITTING**:

Underfitting refers to a model that can neither model the training data nor generalize to new data.

An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data.

Underfitting is often not discussed as it is easy to detect given a good performance metric. The remedy is to move on and try alternate machine learning algorithms. Nevertheless, it does provide a good contrast to the problem of overfitting..



MACHINE LEARNING TYPES

.

**TYPES OF LEARNING ALGORITHMS**

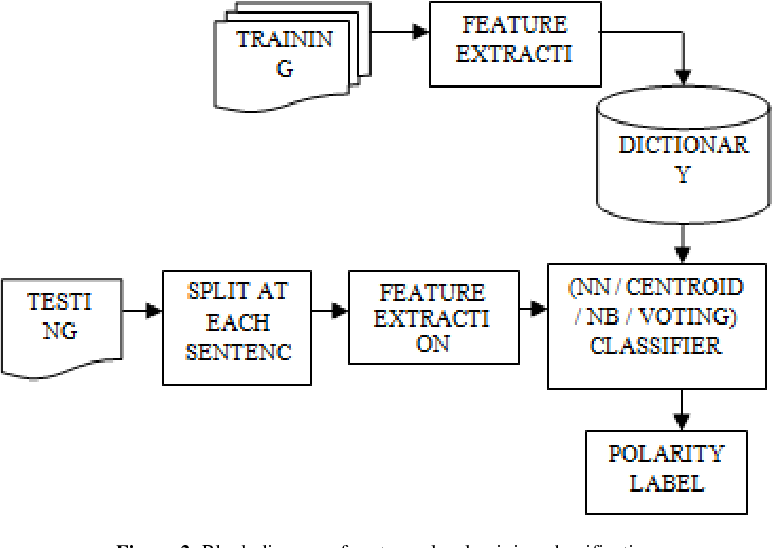
The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

* Supervised learning :

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as [training data](https://en.wikipedia.org/wiki/Training_data), and consists of a set of training examples. Each training example has one or more inputs and a desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an [array](https://en.wikipedia.org/wiki/Array_data_structure) or vector, sometimes called a feature vector, and the training data is represented by a [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)). Through iterative optimization of an [objective function](https://en.wikipedia.org/wiki/Loss_function), supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Supervised learning algorithms include [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis). Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. [Similarity learning](https://en.wikipedia.org/wiki/Similarity_learning) is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in [ranking](https://en.wikipedia.org/wiki/Ranking), [recommendation systems](https://en.wikipedia.org/wiki/Recommendation_systems), visual identity tracking, face verification, and speaker verification.

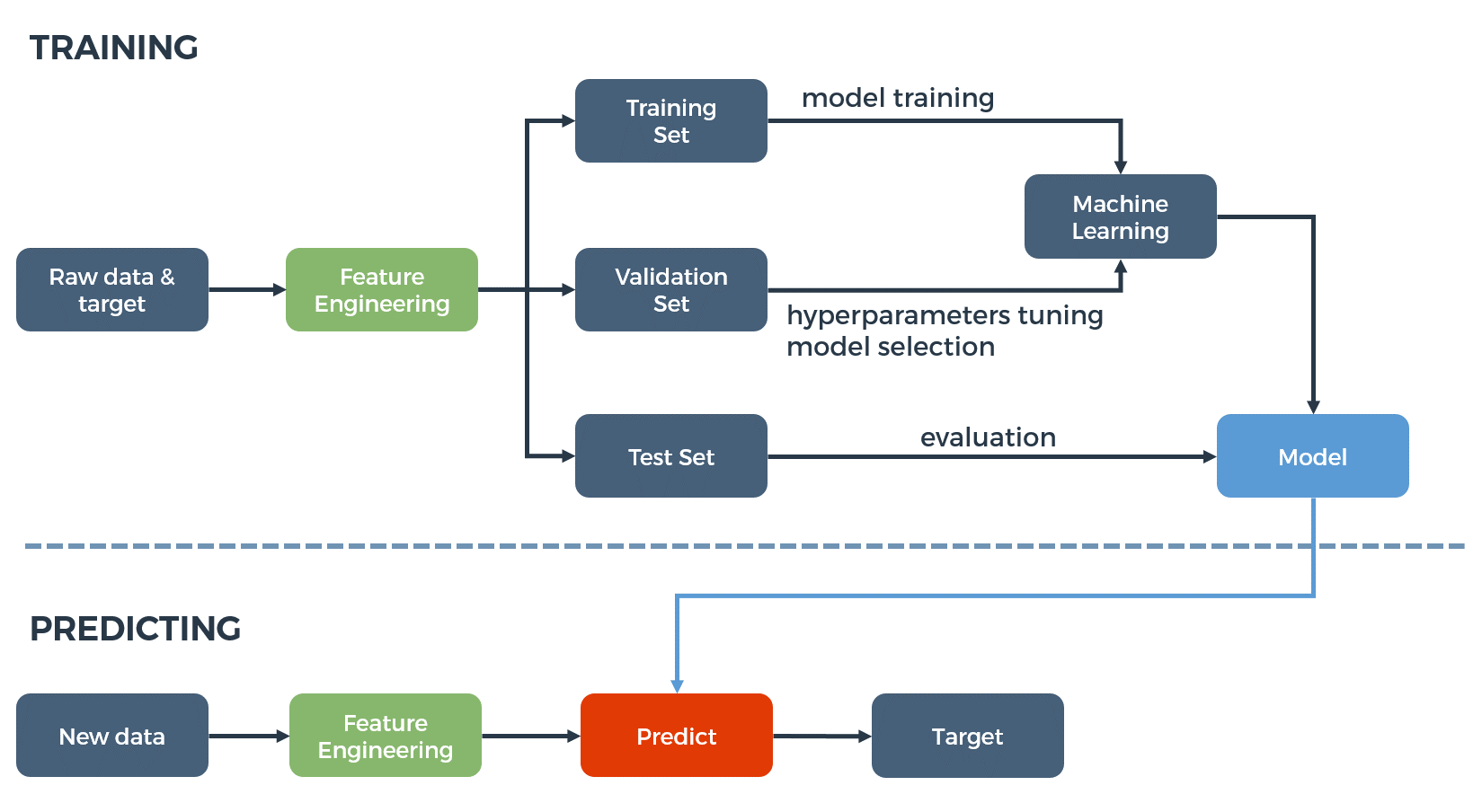
In the case of [semi-supervised](https://en.wikipedia.org/wiki/Semi-supervised_learning) learning algorithms, some of the training examples are missing training labels, but they can nevertheless be used to improve the quality of a model. In [weakly supervised learning](https://en.wikipedia.org/wiki/Weak_supervision), the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets.



* Unsupervised learning**:**

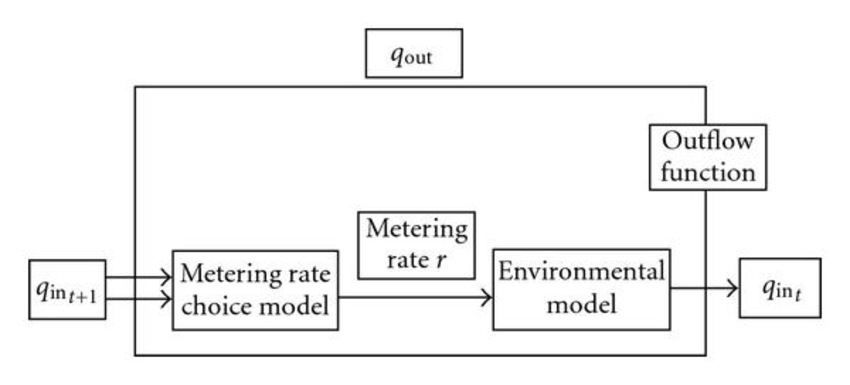
Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms therefore learn from test data that has not been labelled , classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of [density estimation](https://en.wikipedia.org/wiki/Density_estimation) in [statistics](https://en.wikipedia.org/wiki/Statistics), though unsupervised learning encompasses other domains involving summarizing and explaining data features.

Cluster analysis is the assignment of a set of observations into subsets (called *clusters*) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some *similarity metric* and evaluated, for example, by *internal compactness*, or the similarity between members of the same cluster, and *separation*, the difference between clusters. Other methods are based on *estimated density* and *graph connectivity*.



* Reinforcement learning**:**

Reinforcement learning is an area of machine learning concerned with how [software agents](https://en.wikipedia.org/wiki/Software_agent) ought to take [actions](https://en.wikipedia.org/wiki/Action_selection) in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as [game theory](https://en.wikipedia.org/wiki/Game_theory), [control theory](https://en.wikipedia.org/wiki/Control_theory), [operations research](https://en.wikipedia.org/wiki/Operations_research), [information theory](https://en.wikipedia.org/wiki/Information_theory), [simulation-based optimization](https://en.wikipedia.org/wiki/Simulation-based_optimization), [multi-agent systems](https://en.wikipedia.org/wiki/Multi-agent_system), [swarm intelligence](https://en.wikipedia.org/wiki/Swarm_intelligence), [statistics](https://en.wikipedia.org/wiki/Statistics) and [genetic algorithms](https://en.wikipedia.org/wiki/Genetic_algorithm). In machine learning, the environment is typically represented as a [Markov Decision Process](https://en.wikipedia.org/wiki/Markov_Decision_Process) (MDP). Many reinforcement learning algorithms use [dynamic programming](https://en.wikipedia.org/wiki/Dynamic_programming) techniques. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.



* Feature learning**:**

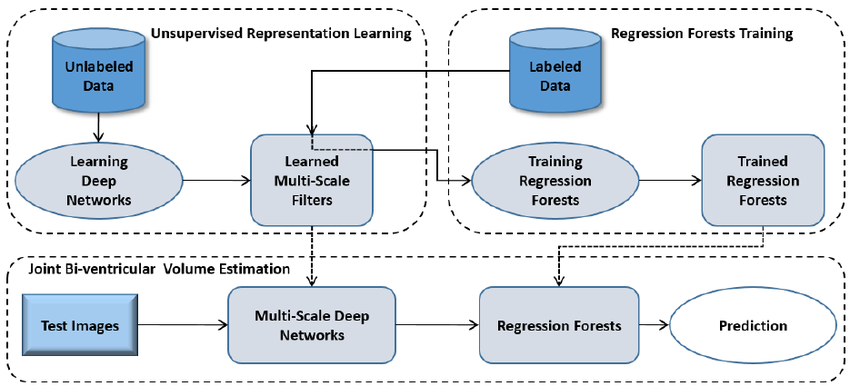
Several learning algorithms aim at discovering better representations of the inputs provided during training. Classic examples include [principal components analysis](https://en.wikipedia.org/wiki/Principal_components_analysis) and cluster analysis. Feature learning algorithms, also called representation learning algorithms, often attempt to preserve the information in their input but also transform it in a way that makes it useful, often as a pre-processing step before performing classification or predictions. This technique allows reconstruction of the inputs coming from the unknown data-generating distribution, while not being necessarily faithful to configurations that are implausible under that distribution. This replaces manual [feature engineering](https://en.wikipedia.org/wiki/Feature_engineering), and allows a machine to both learn the features and use them to perform a specific task.

Feature learning can be either supervised or unsupervised. In supervised feature learning, features are learned using labelled input data. Examples include [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network), [multilayer perceptrons](https://en.wikipedia.org/wiki/Multilayer_perceptron) , and supervised [dictionary learning](https://en.wikipedia.org/wiki/Dictionary_learning). In unsupervised feature learning, features are learned with unlabelled input data. Examples include dictionary learning, [independent component analysis](https://en.wikipedia.org/wiki/Independent_component_analysis), [autoencoders](https://en.wikipedia.org/wiki/Autoencoder), [matrix factorization](https://en.wikipedia.org/wiki/Matrix_decomposition) and various forms of [clustering](https://en.wikipedia.org/wiki/Cluster_analysis).

[Manifold learning](https://en.wikipedia.org/wiki/Manifold_learning) algorithms attempt to do so under the constraint that the learned representation is low-dimensional. [Sparse coding](https://en.wikipedia.org/wiki/Sparse_coding) algorithms attempt to do so under the constraint that the learned representation is sparse, meaning that the mathematical model has many zeros. [Multilinear subspace learning](https://en.wikipedia.org/wiki/Multilinear_subspace_learning) algorithms aim to learn low-dimensional representations directly from [tensor](https://en.wikipedia.org/wiki/Tensor) representations for multidimensional data, without reshaping them into higher-dimensional vectors.[Deep learning](https://en.wikipedia.org/wiki/Deep_learning) algorithms discover multiple levels of representation, or a hierarchy of features, with higher-level, more abstract features defined in terms of (or generating) lower-level features. It has been argued that an intelligent machine is one that learns a representation that disentangles the underlying factors of variation that explain the observed data.

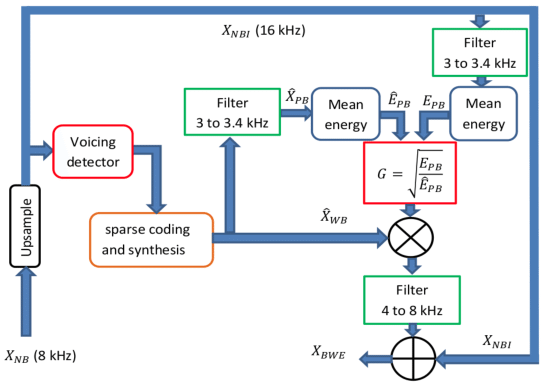
Feature learning is motivated by the fact that machine learning tasks such as classification often require input that is mathematically and computationally convenient to process. However, real-world data such as images, video, and sensory data has not yielded to attempts to algorithmically define specific features. An alternative is to discover such features or representations through examination, without relying on explicit algorithms

.



* Sparse dictionary learning**:**

Sparse dictionary learning is a feature learning method where a training example is represented as a linear combination of [basis functions](https://en.wikipedia.org/wiki/Basis_function" \o "Basis function), and is assumed to be a [sparse matrix](https://en.wikipedia.org/wiki/Sparse_matrix). The method is [strongly NP-hard](https://en.wikipedia.org/wiki/Strongly_NP-hard) and difficult to solve approximately. A popular [heuristic](https://en.wikipedia.org/wiki/Heuristic) method for sparse dictionary learning is the [K-SVD](https://en.wikipedia.org/wiki/K-SVD) algorithm. Sparse dictionary learning has been applied in several contexts. In classification, the problem is to determine to which classes a previously unseen training example belongs. For a dictionary where each class has already been built, a new training example is associated with the class that is best sparsely represented by the corresponding dictionary. Sparse dictionary learning has also been applied in [image de-noising](https://en.wikipedia.org/wiki/Image_de-noising). The key idea is that a clean image patch can be sparsely represented by an image dictionary, but the noise cannot.

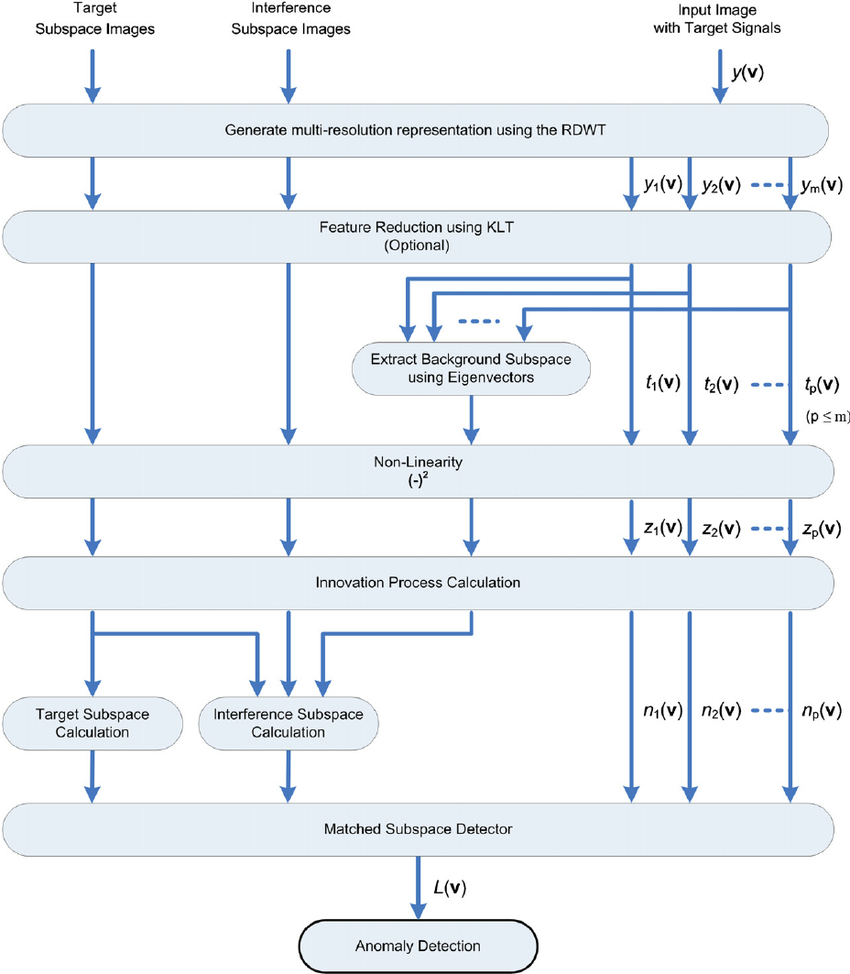


* Anomaly detection**:**

In [data mining](https://en.wikipedia.org/wiki/Data_mining), anomaly detection, also known as outlier detection, is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data. Typically, the anomalous items represent an issue such as [bank fraud](https://en.wikipedia.org/wiki/Bank_fraud), a structural defect, medical problems or errors in a text. Anomalies are referred to as [outliers](https://en.wikipedia.org/wiki/Outlier), novelties, noise, deviations and exceptions.

In particular, in the context of abuse and network intrusion detection, the interesting objects are often not rare objects, but unexpected bursts in activity. This pattern does not adhere to the common statistical definition of an outlier as a rare object, and many outlier detection methods (in particular, unsupervised algorithms) will fail on such data, unless it has been aggregated appropriately. Instead, a cluster analysis algorithm may be able to detect the micro-clusters formed by these patterns.

Three broad categories of anomaly detection techniques exist. Unsupervised anomaly detection techniques detect anomalies in an unlabelled test data set under the assumption that the majority of the instances in the data set are normal, by looking for instances that seem to fit least to the remainder of the data set. Supervised anomaly detection techniques require a data set that has been labelled as "normal" and "abnormal" and involves training a classifier (the key difference to many other statistical classification problems is the inherent unbalanced nature of outlier detection). Semi-supervised anomaly detection techniques construct a model representing normal behaviour from a given normal training data set, and then test the likelihood of a test instance to be generated by the model.



* Association rules**:**

Association rule learning is a [rule-based machine learning](https://en.wikipedia.org/wiki/Rule-based_machine_learning) method for discovering relationships between variables in large databases. It is intended to identify strong rules discovered in databases using some measure of "interestingness".

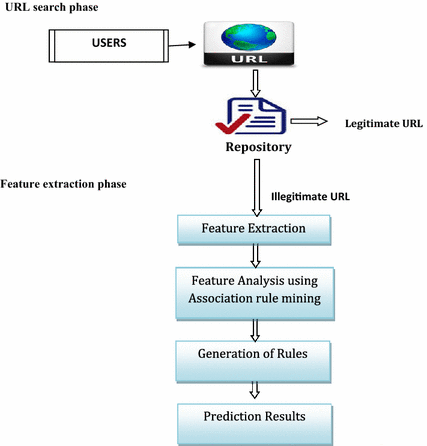
Rule-based machine learning is a general term for any machine learning method that identifies, learns, or evolves "rules" to store, manipulate or apply knowledge. The defining characteristic of a rule-based machine learning algorithm is the identification and utilization of a set of relational rules that collectively represent the knowledge captured by the system. This is in contrast to other machine learning algorithms that commonly identify a singular model that can be universally applied to any instance in order to make a prediction. Rule-based machine learning approaches include [learning classifier systems](https://en.wikipedia.org/wiki/Learning_classifier_system), association rule learning, and [artificial immune systems](https://en.wikipedia.org/wiki/Artificial_immune_system).

Based on the concept of strong rules, [Rakesh Agrawal](https://en.wikipedia.org/wiki/Rakesh_Agrawal_(computer_scientist)), [Tomasz Imieliński](https://en.wikipedia.org/wiki/Tomasz_Imieli%C5%84ski) and Arun Swami introduced association rules for discovering regularities between products in large-scale transaction data recorded by [point-of-sale](https://en.wikipedia.org/wiki/Point-of-sale) (POS) systems in supermarkets. For example, the rule {\displaystyle \{\mathrm {onions,potatoes} \}\Rightarrow \{\mathrm {burger} \}} found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, they are likely to also buy hamburger meat. Such information can be used as the basis for decisions about marketing activities such as promotional [pricing](https://en.wikipedia.org/wiki/Pricing) or [product placements](https://en.wikipedia.org/wiki/Product_placement). In addition to [market basket analysis](https://en.wikipedia.org/wiki/Market_basket_analysis), association rules are employed today in application areas including [Web usage mining](https://en.wikipedia.org/wiki/Web_usage_mining), [intrusion detection](https://en.wikipedia.org/wiki/Intrusion_detection), [continuous production](https://en.wikipedia.org/wiki/Continuous_production), and [bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics). In contrast with [sequence mining](https://en.wikipedia.org/wiki/Sequence_mining), association rule learning typically does not consider the order of items either within a transaction or across transactions.

Learning classifier systems (LCS) are a family of rule-based machine learning algorithms that combine a discovery component, typically a [genetic algorithm](https://en.wikipedia.org/wiki/Genetic_algorithm), with a learning component, performing either [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning), [reinforcement learning](https://en.wikipedia.org/wiki/Reinforcement_learning), or [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning). They seek to identify a set of context-dependent rules that collectively store and apply knowledge in a [piecewise](https://en.wikipedia.org/wiki/Piecewise) manner in order to make predictions.

Inductive logic programming (ILP) is an approach to rule-learning using [logic programming](https://en.wikipedia.org/wiki/Logic_programming) as a uniform representation for input examples, background knowledge, and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program that [entails](https://en.wikipedia.org/wiki/Entailment) all positive and no negative examples. [Inductive programming](https://en.wikipedia.org/wiki/Inductive_programming) is a related field that considers any kind of programming languages for representing hypotheses (and not only logic programming), such as [functional programs](https://en.wikipedia.org/wiki/Functional_programming).

Inductive logic programming is particularly useful in [bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics) and [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing). [Gordon Plotkin](https://en.wikipedia.org/wiki/Gordon_Plotkin) and [Ehud Shapiro](https://en.wikipedia.org/wiki/Ehud_Shapiro) laid the initial theoretical foundation for inductive machine learning in a logical setting. Shapiro built their first implementation (Model Inference System) in 1981: a Prolog program that inductively inferred logic programs from positive and negative examples. The term *inductive* here refers to [philosophical](https://en.wikipedia.org/wiki/Inductive_reasoning) induction, suggesting a theory to explain observed facts, rather than [mathematical](https://en.wikipedia.org/wiki/Mathematical_induction) induction, proving a property for all members of a well-ordered set .



**MACHINE LEARNING CO -ORDINATES**

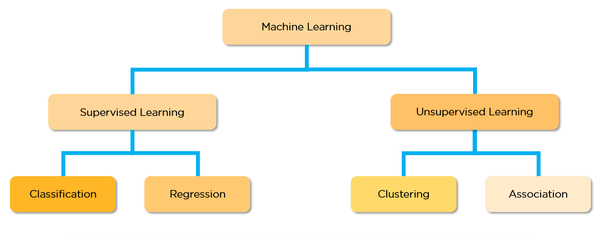


IMAGE OF CLASSIFICATION OF MACHINE LEARNING

*CLASSIFICATION*

A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes.  
For example, when filtering emails “spam” or “not spam”, when looking at transaction data, “fraudulent”, or “authorized”. In short Classification either predicts categorical class labels or classifies data (construct a model) based on the training set and the values (class labels) in classifying attributes and uses it in classifying new data. There are a number of classification models. Classification models include logistic regression, decision tree, random forest, gradient-boosted tree, multilayer perceptron, one-vs-rest, and Naive Bayes.

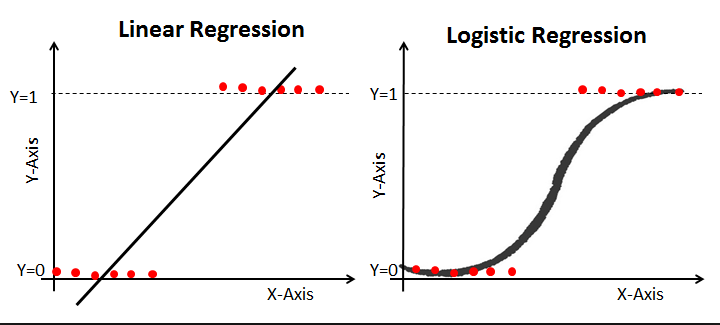
**For example :  
Which of the following is/are classification problem(s)?**

* Predicting the gender of a person by his/her handwriting style
* Predicting house price based on area
* Predicting whether monsoon will be normal next year
* Predict the number of copies a music album will be sold next month

Solution : Predicting the gender of a person Predicting whether monsoon will be normal next year. The other two are regression.  
As we discussed classification with some examples. Now there is an example of classification in which we are performing classification on the iris dataset using *RandomForestClassifier* in python.

*REGRESSION*

A regression problem is when the output variable is a real or continuous value, such as “salary” or “weight”. Many different models can be used, the simplest is the linear regression. It tries to fit data with the best hyper-plane which goes through the points.

**Types of Regression Models:**  


**For Examples:**  
**Which of the following is a regression task?**

* Predicting age of a person
* Predicting nationality of a person
* Predicting whether stock price of a company will increase tomorrow
* Predicting whether a document is related to sighting of UFOs?

**Solution :**Predicting age of a person (because it is a real value, predicting nationality is categorical, whether stock price will increase is discreet-yes/no answer, predicting whether a document is related to UFO is again discreet- a yes/no answer).

*CLUSTERING*

**Cluster analysis** or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense) to each other than to those in other groups (clusters). It is a main task of exploratory [data mining](https://en.wikipedia.org/wiki/Data_mining), and a common technique for [statistical](https://en.wikipedia.org/wiki/Statistics) [data analysis](https://en.wikipedia.org/wiki/Data_analysis), used in many fields, including [machine learning](https://en.wikipedia.org/wiki/Machine_learning), [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition), [image analysis](https://en.wikipedia.org/wiki/Image_analysis), [information retrieval](https://en.wikipedia.org/wiki/Information_retrieval), [bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics), [data compression](https://en.wikipedia.org/wiki/Data_compression), and [computer graphics](https://en.wikipedia.org/wiki/Computer_graphics).

Cluster analysis itself is not one specific [algorithm](https://en.wikipedia.org/wiki/Algorithm), but the general task to be solved. It can be achieved by various algorithms that differ significantly in their understanding of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small [distances](https://en.wikipedia.org/wiki/Distance_function) between cluster members, dense areas of the data space, intervals or particular [statistical distributions](https://en.wikipedia.org/wiki/Statistical_distribution). Clustering can therefore be formulated as a [multi-objective optimization](https://en.wikipedia.org/wiki/Multi-objective_optimization) problem. The appropriate clustering algorithm and parameter settings (including parameters such as the [distance function](https://en.wikipedia.org/wiki/Metric_(mathematics)) to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of [knowledge discovery](https://en.wikipedia.org/wiki/Knowledge_discovery) or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data preprocessing and model parameters until the result achieves the desired properties.

Besides the term *clustering*, there are a number of terms with similar meanings, including *automatic*[*classification*](https://en.wikipedia.org/wiki/Statistical_classification), [*numerical taxonomy*](https://en.wikipedia.org/wiki/Numerical_taxonomy), *botryology* , *typological analysis*, and [*community detection*](https://en.wikipedia.org/wiki/Community_structure). The subtle differences are often in the use of the results: while in data mining, the resulting groups are the matter of interest, in automatic classification the resulting discriminative power is of interest.

The notion of a "cluster" cannot be precisely defined, which is one of the reasons why there are so many clustering algorithms. There is a common denominator: a group of data objects. However, different researchers employ different cluster models, and for each of these cluster models again different algorithms can be given. The notion of a cluster, as found by different algorithms, varies significantly in its properties. Understanding these "cluster models" is key to understanding the differences between the various algorithms. Typical cluster models include:

* *Connectivity models*: for example, [hierarchical clustering](https://en.wikipedia.org/wiki/Hierarchical_clustering) builds models based on distance connectivity.
* *Centroid models*: for example, the [k-means algorithm](https://en.wikipedia.org/wiki/K-means_algorithm) represents each cluster by a single mean vector.
* *Distribution models*: clusters are modeled using statistical distributions, such as [multivariate normal distributions](https://en.wikipedia.org/wiki/Multivariate_normal_distribution) used by the [expectation-maximization algorithm](https://en.wikipedia.org/wiki/Expectation-maximization_algorithm).
* *Density models*: for example, [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN) and [OPTICS](https://en.wikipedia.org/wiki/OPTICS) defines clusters as connected dense regions in the data space.
* *Subspace models*: in [biclustering](https://en.wikipedia.org/wiki/Biclustering" \o "Biclustering) (also known as co-clustering or two-mode-clustering), clusters are modeled with both cluster members and relevant attributes.
* *Group models*: some algorithms do not provide a refined model for their results and just provide the grouping information.
* *Graph-based models*: a [clique](https://en.wikipedia.org/wiki/Clique_(graph_theory)), that is, a subset of nodes in a [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) such that every two nodes in the subset are connected by an edge can be considered as a prototypical form of cluster. Relaxations of the complete connectivity requirement (a fraction of the edges can be missing) are known as quasi-cliques, as in the [HCS clustering algorithm](https://en.wikipedia.org/wiki/HCS_clustering_algorithm).
* *Signed graph models*: Every [path](https://en.wikipedia.org/wiki/Path_(graph_theory)) in a [signed graph](https://en.wikipedia.org/wiki/Signed_graph) has a [sign](https://en.wikipedia.org/wiki/Sign_(mathematics)) from the product of the signs on the edges. Under the assumptions of [balance theory](https://en.wikipedia.org/wiki/Balance_theory), edges may change sign and result in a bifurcated graph. The weaker "clusterability axiom" (no [cycle](https://en.wikipedia.org/wiki/Cycle_(graph_theory)) has exactly one negative edge) yields results with more than two clusters, or subgraphs with only positive edges.
* *Neural models*: the most well known [unsupervised](https://en.wikipedia.org/wiki/Unsupervised_learning) [neural network](https://en.wikipedia.org/wiki/Neural_network) is the [self-organizing map](https://en.wikipedia.org/wiki/Self-organizing_map) and these models can usually be characterized as similar to one or more of the above models, and including subspace models when neural networks implement a form of [Principal Component Analysis](https://en.wikipedia.org/wiki/Principal_Component_Analysis) or [Independent Component Analysis](https://en.wikipedia.org/wiki/Independent_Component_Analysis).

A "clustering" is essentially a set of such clusters, usually containing all objects in the data set. Additionally, it may specify the relationship of the clusters to each other, for example, a hierarchy of clusters embedded in each other. Clusterings can be roughly distinguished as:

* *Hard clustering*: each object belongs to a cluster or not
* *Soft clustering* (also: [*fuzzy clustering*](https://en.wikipedia.org/wiki/Fuzzy_clustering)): each object belongs to each cluster to a certain degree (for example, a likelihood of belonging to the cluster)

**ALGORITHMS**

As listed above, clustering algorithms can be categorized based on their cluster model. The following overview will only list the most prominent examples of clustering algorithms, as there are possibly over 100 published clustering algorithms. Not all provide models for their clusters and can thus not easily be categorized. An overview of algorithms explained in Wikipedia can be found in the [list of statistics algorithms](https://en.wikipedia.org/wiki/List_of_algorithms#Statistics).

There is no objectively "correct" clustering algorithm, but as it was noted, "clustering is in the eye of the beholder." The most appropriate clustering algorithm for a particular problem often needs to be chosen experimentally, unless there is a mathematical reason to prefer one cluster model over another. An algorithm that is designed for one kind of model will generally fail on a data set that contains a radically different kind of model. For example, k-means cannot find non-convex clusters.

### Connectivity-based clustering (hierarchical clustering)

Connectivity-based clustering, also known as [*hierarchical clustering*](https://en.wikipedia.org/wiki/Hierarchical_clustering), is based on the core idea of objects being more related to nearby objects than to objects farther away. These algorithms connect "objects" to form "clusters" based on their distance. A cluster can be described largely by the maximum distance needed to connect parts of the cluster. At different distances, different clusters will form, which can be represented using a [dendrogram](https://en.wikipedia.org/wiki/Dendrogram), which explains where the common name "hierarchical clustering" comes from: these algorithms do not provide a single partitioning of the data set, but instead provide an extensive hierarchy of clusters that merge with each other at certain distances. In a dendrogram, the y-axis marks the distance at which the clusters merge, while the objects are placed along the x-axis such that the clusters don't mix.

Connectivity-based clustering is a whole family of methods that differ by the way distances are computed. Apart from the usual choice of [distance functions](https://en.wikipedia.org/wiki/Distance_function), the user also needs to decide on the linkage criterion (since a cluster consists of multiple objects, there are multiple candidates to compute the distance) to use. Popular choices are known as [single-linkage clustering](https://en.wikipedia.org/wiki/Single-linkage_clustering) (the minimum of object distances), [complete linkage clustering](https://en.wikipedia.org/wiki/Complete_linkage_clustering) (the maximum of object distances), and [UPGMA](https://en.wikipedia.org/wiki/UPGMA) or [WPGMA](https://en.wikipedia.org/wiki/WPGMA) ("Unweighted or Weighted Pair Group Method with Arithmetic Mean", also known as average linkage clustering). Furthermore, hierarchical clustering can be agglomerative (starting with single elements and aggregating them into clusters) or divisive (starting with the complete data set and dividing it into partitions).

These methods will not produce a unique partitioning of the data set, but a hierarchy from which the user still needs to choose appropriate clusters. They are not very robust towards outliers, which will either show up as additional clusters or even cause other clusters to merge (known as "chaining phenomenon", in particular with [single-linkage clustering](https://en.wikipedia.org/wiki/Single-linkage_clustering)). In the general case, the complexity is {\displaystyle {\mathcal {O}}(n^{3})} for agglomerative clustering and {\displaystyle {\mathcal {O}}(2^{n-1})} for [divisive clustering](https://en.wikipedia.org/wiki/Divisive_clustering), which makes them too slow for large data sets. For some special cases, optimal efficient methods (of complexity {\displaystyle {\mathcal {O}}(n^{2})}) are known: SLINK for single-linkage and CLINKfor complete-linkage clustering. In the [data mining](https://en.wikipedia.org/wiki/Data_mining) community these methods are recognized as a theoretical foundation of cluster analysis, but often considered obsolete. They did however provide inspiration for many later methods such as density based clustering.

### Centroid-based clustering

In centroid-based clustering, clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to *k*, [*k*-means clustering](https://en.wikipedia.org/wiki/K-means_clustering) gives a formal definition as an optimization problem: find the *k* cluster centers and assign the objects to the nearest cluster center, such that the squared distances from the cluster are minimized.

The optimization problem itself is known to be [NP-hard](https://en.wikipedia.org/wiki/NP-hard), and thus the common approach is to search only for approximate solutions. A particularly well known approximate method is [Lloyd's algorithm](https://en.wikipedia.org/wiki/Lloyd%27s_algorithm), often just referred to as "*k-means algorithm*" (although [another algorithm introduced this name](https://en.wikipedia.org/wiki/K-means_clustering#History)). It does however only find a [local optimum](https://en.wikipedia.org/wiki/Local_optimum), and is commonly run multiple times with different random initializations. Variations of *k*-means often include such optimizations as choosing the best of multiple runs, but also restricting the centroids to members of the data set ([*k*-medoids](https://en.wikipedia.org/wiki/K-medoids)), choosing [medians](https://en.wikipedia.org/wiki/Median) ([*k*-medians clustering](https://en.wikipedia.org/wiki/K-medians_clustering)), choosing the initial centers less randomly ([*k*-means++](https://en.wikipedia.org/wiki/K-means%2B%2B)) or allowing a fuzzy cluster assignment ([fuzzy c-means](https://en.wikipedia.org/wiki/Fuzzy_clustering)).

Most *k*-means-type algorithms require the [number of clusters](https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set) – *k* – to be specified in advance, which is considered to be one of the biggest drawbacks of these algorithms. Furthermore, the algorithms prefer clusters of approximately similar size, as they will always assign an object to the nearest centroid. This often leads to incorrectly cut borders of clusters (which is not surprising since the algorithm optimizes cluster centers, not cluster borders).

K-means has a number of interesting theoretical properties. First, it partitions the data space into a structure known as a [Voronoi diagram](https://en.wikipedia.org/wiki/Voronoi_diagram). Second, it is conceptually close to nearest neighbour classification, and as such is popular in [machine learning](https://en.wikipedia.org/wiki/Machine_learning). Third, it can be seen as a variation of model based clustering, and Lloyd's algorithm as a variation of the [Expectation-maximization algorithm](https://en.wikipedia.org/wiki/Expectation-maximization_algorithm) for this model discussed below.

### Distribution-based clustering:

The clustering model most closely related to statistics is based on [distribution models](https://en.wikipedia.org/wiki/Probability_distribution). Clusters can then easily be defined as objects belonging most likely to the same distribution. A convenient property of this approach is that this closely resembles the way artificial data sets are generated: by sampling random objects from a distribution.

While the theoretical foundation of these methods is excellent, they suffer from one key problem known as [overfitting](https://en.wikipedia.org/wiki/Overfitting), unless constraints are put on the model complexity. A more complex model will usually be able to explain the data better, which makes choosing the appropriate model complexity inherently difficult.

One prominent method is known as Gaussian mixture models (using the [expectation-maximization algorithm](https://en.wikipedia.org/wiki/Expectation-maximization_algorithm)). Here, the data set is usually modeled with a fixed (to avoid overfitting) number of [Gaussian distributions](https://en.wikipedia.org/wiki/Gaussian_distribution) that are initialized randomly and whose parameters are iteratively optimized to better fit the data set. This will converge to a [local optimum](https://en.wikipedia.org/wiki/Local_optimum), so multiple runs may produce different results. In order to obtain a hard clustering, objects are often then assigned to the Gaussian distribution they most likely belong to; for soft clusterings, this is not necessary.

Distribution-based clustering produces complex models for clusters that can capture [correlation and dependence](https://en.wikipedia.org/wiki/Correlation_and_dependence) between attributes. However, these algorithms put an extra burden on the user: for many real data sets, there may be no concisely defined mathematical model (e.g. assuming Gaussian distributions is a rather strong assumption on the data).

**MODELS**

### Models:

Performing machine learning involves creating a [model](https://en.wikipedia.org/wiki/Statistical_model), which is trained on some training data and then can process additional data to make predictions. Various types of models have been used and researched for machine learning systems.

#### 1)Artificial neural networks:

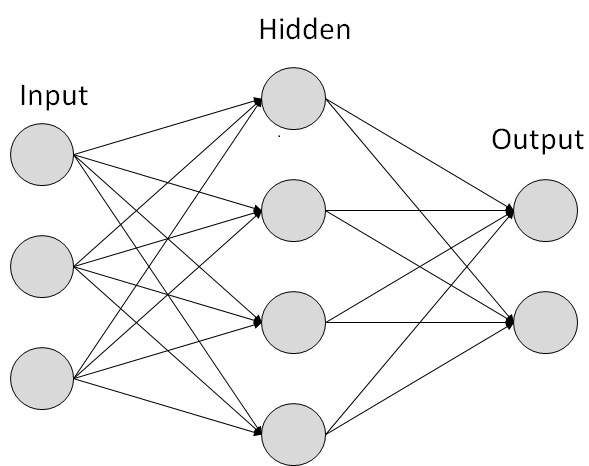
An artificial neural network is an interconnected group of nodes, akin to the vast network of [neurons](https://en.wikipedia.org/wiki/Neuron) in a [brain](https://en.wikipedia.org/wiki/Brain). Here, each circular node represents an [artificial neuron](https://en.wikipedia.org/wiki/Artificial_neuron) and an arrow represents a connection from the output of one artificial neuron to the input of another.

Artificial neural networks (ANNs), or [connectionist](https://en.wikipedia.org/wiki/Connectionism) systems, are computing systems vaguely inspired by the [biological neural networks](https://en.wikipedia.org/wiki/Biological_neural_network) that constitute animal [brains](https://en.wikipedia.org/wiki/Brain). Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

An ANN is a model based on a collection of connected units or nodes called "[artificial neurons](https://en.wikipedia.org/wiki/Artificial_neuron)", which loosely model the [neurons](https://en.wikipedia.org/wiki/Neuron) in a biological [brain](https://en.wikipedia.org/wiki/Brain). Each connection, like the [synapses](https://en.wikipedia.org/wiki/Synapse) in a biological [brain](https://en.wikipedia.org/wiki/Brain), can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN implementations, the signal at a connection between artificial neurons is a [real number](https://en.wikipedia.org/wiki/Real_number), and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called "edges". Artificial neurons and edges typically have a [weight](https://en.wikipedia.org/wiki/Weight_(mathematics)) that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

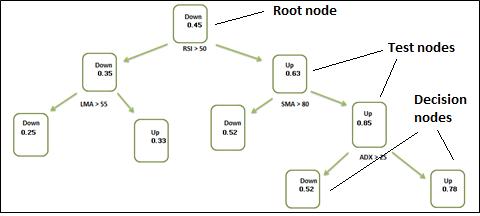
The original goal of the ANN approach was to solve problems in the same way that a [human brain](https://en.wikipedia.org/wiki/Human_brain) would. However, over time, attention moved to performing specific tasks, leading to deviations from [biology](https://en.wikipedia.org/wiki/Biology). Artificial neural networks have been used on a variety of tasks, including [computer vision](https://en.wikipedia.org/wiki/Computer_vision), [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition), [machine translation](https://en.wikipedia.org/wiki/Machine_translation), [social network](https://en.wikipedia.org/wiki/Social_network) filtering, [playing board and video games](https://en.wikipedia.org/wiki/General_game_playing) and [medical diagnosis](https://en.wikipedia.org/wiki/Medical_diagnosis).

[Deep learning](https://en.wikipedia.org/wiki/Deep_learning) consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are [computer vision](https://en.wikipedia.org/wiki/Computer_vision) and [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition).



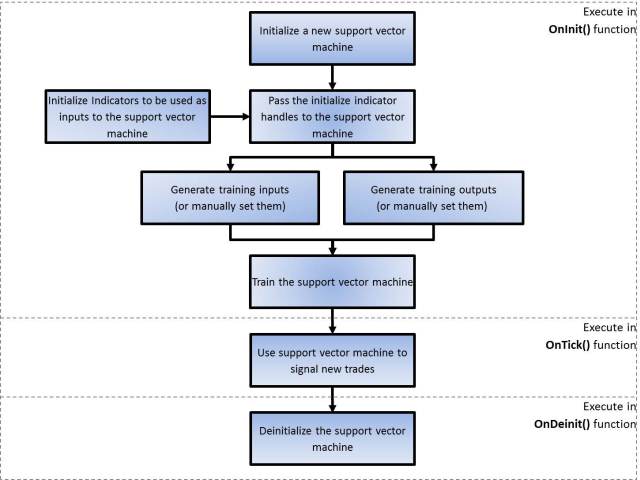
#### 2)Decision trees

Decision tree learning uses a [decision tree](https://en.wikipedia.org/wiki/Decision_tree) as a [predictive model](https://en.wikipedia.org/wiki/Predictive_modelling) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, [leaves](https://en.wikipedia.org/wiki/Leaf_node) represent class labels and branches represent [conjunctions](https://en.wikipedia.org/wiki/Logical_conjunction) of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically [real numbers](https://en.wikipedia.org/wiki/Real_numbers)) are called regression trees. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and [decision making](https://en.wikipedia.org/wiki/Decision_making). In data mining, a decision tree describes data, but the resulting classification tree can be an input for decision making.



#### 3)Support vector machines

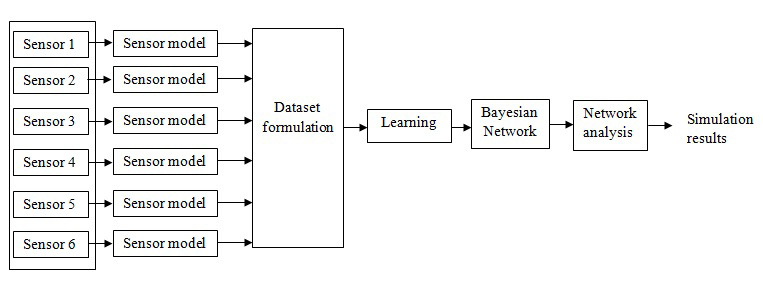
Support vector machines (SVMs), also known as support vector networks, are a set of related [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. An SVM training algorithm is a non-[probabilistic](https://en.wikipedia.org/wiki/Probabilistic_classification), [binary](https://en.wikipedia.org/wiki/Binary_classifier), [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier), although methods such as [Platt scaling](https://en.wikipedia.org/wiki/Platt_scaling) exist to use SVM in a probabilistic classification setting. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the [kernel trick](https://en.wikipedia.org/wiki/Kernel_trick), implicitly mapping their inputs into high-dimensional feature spaces.



#### 4)Bayesian networks

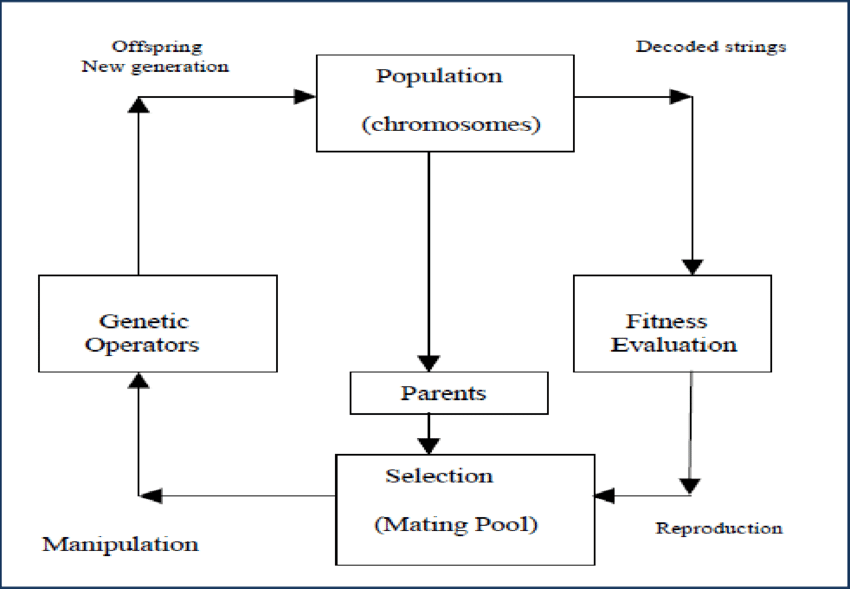
A simple Bayesian network. Rain influences whether the sprinkler is activated, and both rain and the sprinkler influence whether the grass is wet.

A Bayesian network, belief network or directed acyclic graphical model is a probabilistic [graphical model](https://en.wikipedia.org/wiki/Graphical_model) that represents a set of [random variables](https://en.wikipedia.org/wiki/Random_variables) and their [conditional independence](https://en.wikipedia.org/wiki/Conditional_independence) with a [directed acyclic graph](https://en.wikipedia.org/wiki/Directed_acyclic_graph) (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform [inference](https://en.wikipedia.org/wiki/Inference) and learning. Bayesian networks that model sequences of variables, like [speech signals](https://en.wikipedia.org/wiki/Speech_recognition) or [protein sequences](https://en.wikipedia.org/wiki/Peptide_sequence), are called [dynamic Bayesian networks](https://en.wikipedia.org/wiki/Dynamic_Bayesian_network). Generalizations of Bayesian networks that can represent and solve decision problems under uncertainty are called [influence diagrams](https://en.wikipedia.org/wiki/Influence_diagram).



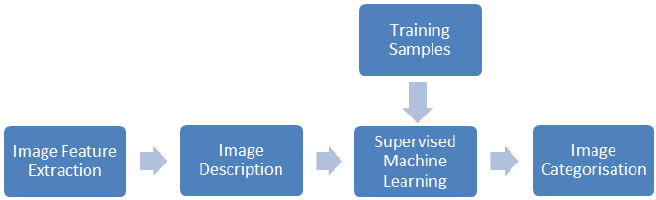
#### 5)Genetic algorithms

A genetic algorithm (GA) is a [search algorithm](https://en.wikipedia.org/wiki/Search_algorithm) and [heuristic](https://en.wikipedia.org/wiki/Heuristic_(computer_science)) technique that mimics the process of [natural selection](https://en.wikipedia.org/wiki/Natural_selection), using methods such as [mutation](https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)) and [crossover](https://en.wikipedia.org/wiki/Crossover_(genetic_algorithm)) to generate new [genotypes](https://en.wikipedia.org/wiki/Chromosome_(genetic_algorithm)) in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms were used in the 1980s and 1990s. Conversely, machine learning techniques have been used to improve the performance of genetic and [evolutionary algorithms](https://en.wikipedia.org/wiki/Evolutionary_algorithm).



### 6)Training models

Usually, machine learning models require a lot of data in order for them to perform well. Usually, when training a machine learning model, one needs to collect a large, representative sample of data from a training set. Data from the training set can be as varied as a corpus of text, a collection of images, and data collected from individual users of a service. [Overfitting](https://en.wikipedia.org/wiki/Overfitting) is something to watch out for when training a machine learning model.



#### 7)Federated learning

Federated learning is a new approach to training machine learning models that decentralizes the training process, allowing for users' privacy to be maintained by not needing to send their data to a centralized server. This also increases efficiency by decentralizing the training process to many devices. For example, [Gboard](https://en.wikipedia.org/wiki/Gboard" \o "Gboard) uses federated machine learning to train search query prediction models on users' mobile phones without having to send individual searches back to [Google](https://en.wikipedia.org/wiki/Google).

**LIMITATIONS**

### Bias:

Machine learning approaches in particular can suffer from different data biases. A machine learning system trained on current customers only may not be able to predict the needs of new customer groups that are not represented in the training data. When trained on man-made data, machine learning is likely to pick up the same constitutional and unconscious biases already present in society. Language models learned from data have been shown to contain human-like biases. Machine learning systems used for criminal risk assessment have been found to be biased against black people. In 2015, Google photos would often tag black people as gorillas, and in 2018 this still was not well resolved, but Google reportedly was still using the workaround to remove all gorilla from the training data, and thus was not able to recognize real gorillas at all. Similar issues with recognizing non-white people have been found in many other systems. In 2016, Microsoft tested a [chatbot](https://en.wikipedia.org/wiki/Chatbot) that learned from Twitter, and it quickly picked up racist and sexist language. Because of such challenges, the effective use of machine learning may take longer to be adopted in other domains. Concern for reducing bias in machine learning and propelling its use for human good is increasingly expressed by artificial intelligence scientists, including [Fei-Fei Li](https://en.wikipedia.org/wiki/Fei-Fei_Li), who reminds engineers that "There’s nothing artificial about AI...It’s inspired by people, it’s created by people, and—most importantly—it impacts people. It is a powerful tool we are only just beginning to understand, and that is a profound responsibility.”

## **Model assessments**

Classification machine learning models can be validated by accuracy estimation techniques like the [Holdout](https://en.wikipedia.org/wiki/Test_set) method, which splits the data in a training and test set (conventionally 2/3 training set and 1/3 test set designation) and evaluates the performance of the training model on the test set. In comparison, the K-fold-[cross-validation](https://en.wikipedia.org/wiki/Cross-validation_(statistics)) method randomly partitions the data into K subsets and then K experiments are performed each respectively considering 1 subset for evaluation and the remaining K-1 subsets for training the model. In addition to the holdout and cross-validation methods, [bootstrap](https://en.wikipedia.org/wiki/Bootstrapping), which samples n instances with replacement from the dataset, can be used to assess model accuracy.

In addition to overall accuracy, investigators frequently report [sensitivity and specificity](https://en.wikipedia.org/wiki/Sensitivity_and_specificity) meaning True Positive Rate (TPR) and True Negative Rate (TNR) respectively. Similarly, investigators sometimes report the [False Positive Rate](https://en.wikipedia.org/wiki/False_Positive_Rate) (FPR) as well as the [False Negative Rate](https://en.wikipedia.org/wiki/False_Negative_Rate) (FNR). However, these rates are ratios that fail to reveal their numerators and denominators. The [Total Operating Characteristic](https://en.wikipedia.org/wiki/Total_Operating_Characteristic) (TOC) is an effective method to express a model's diagnostic ability. TOC shows the numerators and denominators of the previously mentioned rates, thus TOC provides more information than the commonly used [Receiver Operating Characteristic](https://en.wikipedia.org/wiki/Receiver_Operating_Characteristic) (ROC) and ROC's associated Area Under the Curve (AUC).

**REAL LIFE EXAMPLE OF MACHINE LEARNING**

Machine learning is one modern innovation that has helped man enhance not only many industrial and professional processes but also advances everyday living. But what is machine learning? It is a subset of artificial intelligence, which focuses on using statistical techniques to build intelligent computer systems in order to learn from databases available to it. Currently, [machine learning](https://bigdata-madesimple.com/why-every-small-business-should-use-machine-learning/) has been used in multiple fields and industries. For example, medical diagnosis, image processing, prediction, classification, learning association, regression etc.

The intelligent systems built on machine learning algorithms have the capability to learn from past experience or historical data. [Machine learning applications](https://bigdata-madesimple.com/how-do-you-explain-machine-learning-and-data-mining-to-a-layman/) provide results on the basis of past experience. In this article, we will discuss 10 real-life examples of how machine learning is helping in creating better technology to power today’s ideas.

## **Image Recognition**

Image recognition is one of the most common uses of machine learning. There are many situations where you can classify the object as a [digital image](https://roboticsbiz.com/machine-vision-to-open-a-plethora-of-opportunities-for-companies/). For example, in the case of a black and white image, the intensity of each pixel is served as one of the measurements. In colored images, each pixel provides 3 measurements of intensities in three different colors – red, green and blue (RGB).

Machine learning can be used for [face detection in an image](https://www.webtunix.com/blog/real-time-tracking-face-detection-and-recognition-using-ai) as well. There is a separate category for each person in a database of several people. Machine learning is also used for character recognition to discern handwritten as well as printed letters.  We can segment a piece of writing into smaller images, each containing a single character.

## **Speech Recognition**

[Speech recognition](https://medium.com/@ageitgey/machine-learning-is-fun-part-6-how-to-do-speech-recognition-with-deep-learning-28293c162f7a) is the translation of spoken words into the text. It is also known as computer speech recognition or automatic speech recognition. Here, a software application can recognize the words spoken in an audio clip or file, and then subsequently convert the audio into a text file. The measurement in this application can be a set of numbers that represent the speech signal. We can also segment the speech signal by intensities in different time-frequency bands.

Speech recognition is used in the applications like voice user interface, voice searches and more. Voice user interfaces include voice dialing, call routing, and appliance control. It can also be used a simple data entry and the preparation of structured documents.

## **Medical diagnosis**

Machine learning can be used in the techniques and tools that can help in the [diagnosis of diseases](https://www.webtunix.com/blog/why-artificial-intelligence-important-for-healthcare). It is used for the analysis of the clinical parameters and their combination for the prognosis example prediction of disease progression for the extraction of medical knowledge for the outcome research, for therapy planning and patient monitoring. These are the successful implementations of the machine learning methods. It can help in the integration of computer-based systems in the healthcare sector.

## **Statistical Arbitrage**

In finance, arbitrage refers to the automated trading strategies that are of a short-term and involve a large number of securities. In these strategies, the user focuses on implementing the trading algorithm for a set of securities on the basis of quantities like historical correlations and the general economic variables. Machine learning methods are applied to obtain an index arbitrage strategy. We apply linear regression and the Support Vector Machine to the prices of a stream of stocks.

## **Learning associations**

Learning associations is the process of developing insights into the various associations between the products. A good example is how the unrelated products can be associated with one another. One of the applications of machine learning is studying the associations between the products that people buy. If a person buys a product, he will be shown similar products because there is a relation between the two products. When any new products are launched in the market, they are associated with the old ones to increase their sales.

## **Classification**

A classification is a process of placing each individual under study in many classes. Classification helps to analyze the measurements of an object to identify the category to which that object belongs. To establish an efficient relation, analysts use data. For example, before a bank decides to distribute loans, it assesses the customers on their ability to pay loans. By considering the factors like customer’s earnings, savings, and financial history, we can do it. This information is taken from the past data on the loan.

## **Prediction**

Machine learning can also be used in the prediction systems. Considering the loan example, to compute the probability of a fault, the system will need to classify the available data in groups. It is defined by a set of rules prescribed by the analysts. Once the classification is done, we can calculate the probability of the fault. These computations can compute across all the sectors for varied purposes. Making predictions is one of the best machine learning applications.

## **Extraction**

Extraction of information is one of the [best applications of machine learning](https://bigdata-madesimple.com/a-list-of-best-data-science-machine-learning-projects-at-github/). It is the process of extracting structured information from the unstructured data. For example, the web pages, articles, blogs, business reports, and emails. The relational database maintains the output produced by the information extraction. The process of extraction takes a set of documents as input and outputs the structured data.

## **Regression**

We can also implement machine learning in the regression as well. In regression, we can use the principle of machine learning to optimize the parameters. It can also be used to decrease the approximation error and calculate the closest possible outcome. We can also use the machine learning for the function optimization. We can also choose to alter the inputs in order to get the closest possible outcome.

## **Financial Services**

Machine learning has a lot of potential in the [financial and banking sector](https://bigdata-madesimple.com/banking-in-the-time-of-emerging-tech-blockchain-ai-personalization/). It is the driving force behind the popularity of the financial services. Machine learning can help the banks, financial institutions to make smarter decisions. Machine learning can help the financial services to spot an account closure before it occurs. It can also track the spending pattern of the customers. Machine learning can also perform the market analysis. Smart machines can be trained to track the spending patterns. The algorithms can identify the tends easily and can react in real time.

### **Conclusion**

In a nutshell, we can say that machine learning is an incredible breakthrough in the field of artificial intelligence. And while machine learning has some frightening implications, these machine learning applications are one of the ways through which technology can improve our lives.

**DATA DESCRIPTION**

* **SOURCE OF DATA**:

Data got from globsyn.

* **DATA ANALYSIS**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FEATURE | RECORD | NULL | TYPE | X OR Y | OUTLIER |
| ST | 3333 | NON-NULL | OBJECT  (DISCRETE) | X | NO |
| ACCLEN | 3333 | NON-NULL | INT64  (CONTINUOUS) | X | YES |
| ARCODE | 3333 | NON-NULL | INT64  (CONTINUOUS) | X | NO |
| INTPLAN | 3333 | NON-NULL | OBJECT  (DISCRETE) | X |  |
| VOICE | 3333 | NON-NULL | OBJECT  (DISCRETE) | X |  |
| NUMMAILMES | 3333 | NON-NULL | INT64  (DISCRETE) | X | NO |
| TDMIN | 3333 | NON-NULL | FLOAT64  (CONTINUOUS) | X | YES |
| TDCAL | 3333 | NON-NULL | CONTINUOUS | X | YES |
| TDCHAR | 3333 | NON-NULL | FLOAT64  (CONTINUOUS) | X | YES |
| TEMIN | 3333 | NON-NULL | FLOAT64  (CONTINUOUS) | X | YES |
| TECAL | 3333 | NON-NULL | INT64  (CONTINUOUS) | X | YES |
| TECAHR | 3333 | NON-NULL | FLOAT64  (CONTINUOUS) | X | YES |
| TNMIN | 3333 | NON-NULL | FLOAT64  (CONTINUOUS) | X | YES |
| TNCAL | 3333 | NON-NULL | INT64  (CONTINUOUS) | X | YES |
| TNCHAR | 3333 | NON-NULL | FLOAT64  (CONTINUOUS) | X | YES |
| TIMIN | 3333 | NON-NULL | FLOAT64  (CONTINUOUS) | X | YES |
| TICAL | 3333 | NON-NULL | INT64  (CONTINUOUS) | X | YES |
| TICHAR | 3333 | NON-NULL | FLOAT64  (CONTINUOUS) | X | YES |
| NCSC | 3333 | NON-NULL | INT64  (DISCRETE) | X | YES |
| LABEL | 3333 | NON-NULL | OBJECT  (DISCRETE) | Y |  |
| PHNUM | 3333 | NON-NULL | OBJECT  (DISCRETE) |  | NO |

**BOXPLOT**

For some distributions/datasets, you will find that you need more information than the measures of central tendency (median, mean, and mode). You need to have information on the variability or dispersion of the data. A boxplot is a graph that gives you a good indication of how the values in the data are spread out. Although box lots may seem primitive in comparison to a [histogram](https://datavizcatalogue.com/methods/histogram.html) or [density plot](https://datavizcatalogue.com/methods/density_plot.html), they have the advantage of taking up less space, which is useful when comparing distributions between many groups or datasets.

Boxplots are a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”).

**median (Q2/50th Percentile)**: the middle value of the dataset.

**first quartile (Q1/25th Percentile)**: the middle number between the smallest number (not the “minimum”) and the median of the dataset.

**third quartile (Q3/75th Percentile)**: the middle value between the median and the highest value (not the “maximum”) of the dataset.

**interquartile range (IQR)**: 25th to the 75th percentile.

**whiskers (shown in blue)**

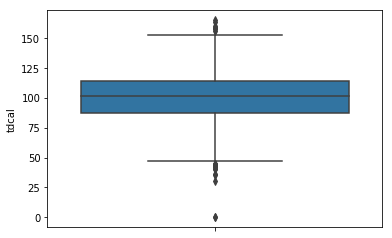
**outliers (shown as green circles)**

**“maximum”**: Q3 + 1.5\*IQR

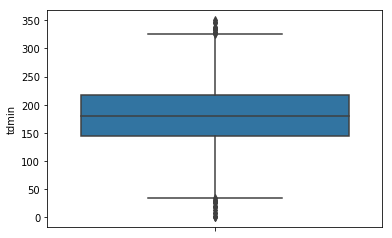
**“minimum”**: Q1 -1.5\*IQR

* **BOX PLOT OF DIFFERENT FEATURES:**

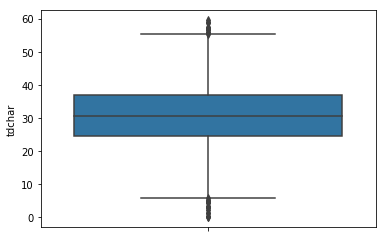
Here , the box plot is of tdcal feature. Here, we can see that there are max and min outlier seen in the graph.



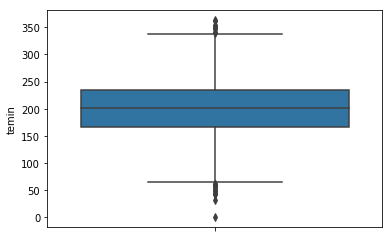
2) Here , tdmin is box plotted and we can see the max and min both outlier is visible.



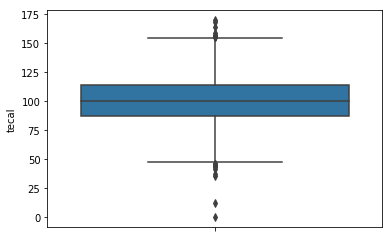
3) Here , tdchar is box plotted and both max and min outlier is visible.



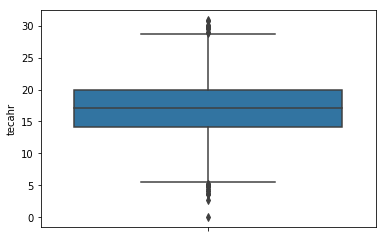
4)Here , temin is box plotted and both max and min outlier is visible.



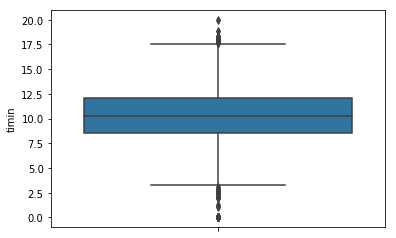
5)Here , tecal is box plotted and both max and min outlier is visible.



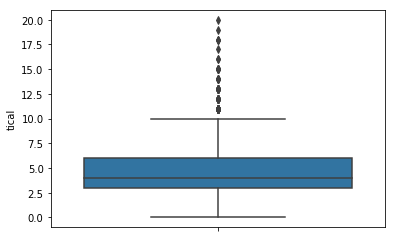
6)Here , tecahr is box plotted and both max and min outlier is visible.



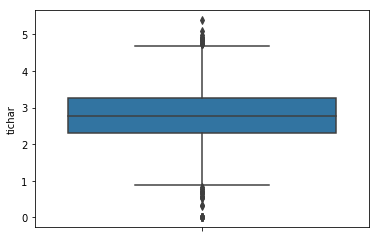
7)Here , timin is box plotted and both max and min outlier is visible.



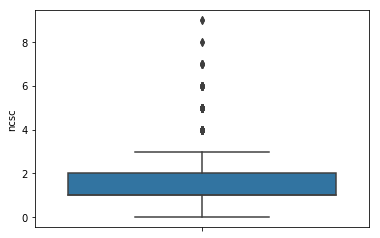
8) Here , tical is box plotted and only max outlier is visible.



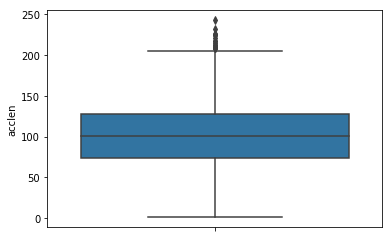
9)Here , tichar is box plotted and both max and min outlier is visible.



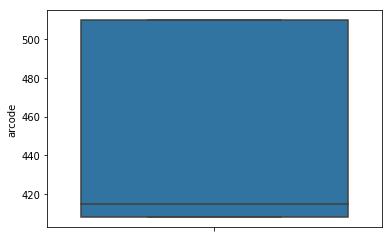
10)Here , ncsc is box plotted and only max outlier is visible.



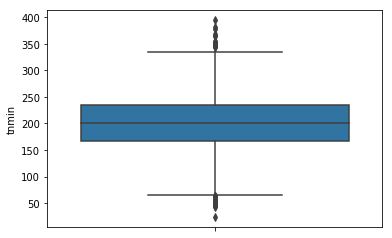
11)Here , acclen is box plotted and only min outlier is visible.



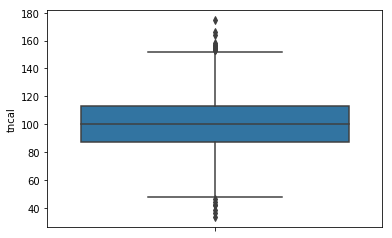
12)Here , arcode is box plotted .



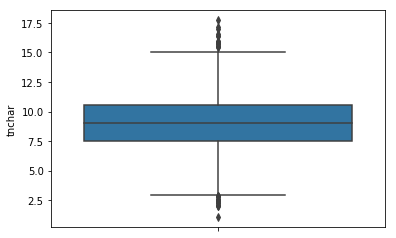
13) Here , tnmin is box plotted and both max and min outlier is visible.



14) Here , tncal is box plotted and both max and min outlier is visible.

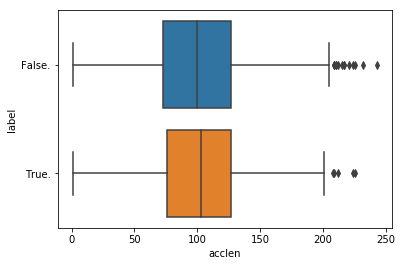


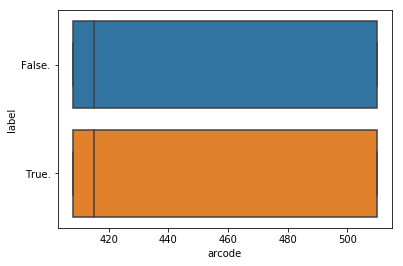
15)Here , tnchar is box plotted and both max and min outlier is visible.

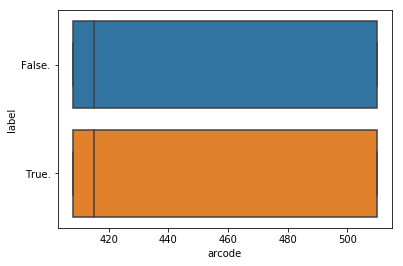


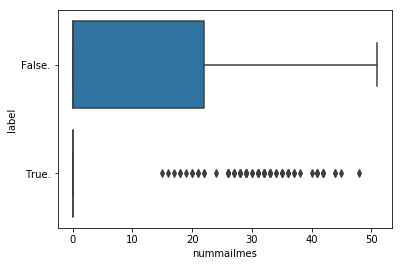
* **BOXPLOT OF BIVARIATE :**

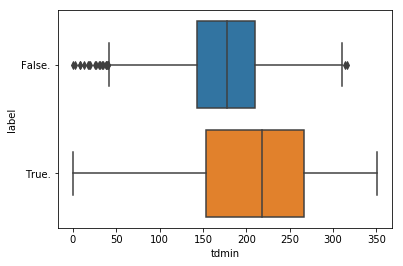
We propose the bagplot, a bivariate generalization of the univariate boxplot. The key notion is the half space location depth of a point relative to a bivariate dataset, which extends the univariate concept of rank. The “depth median” is the deepest location, and it is surrounded by a “bag” containing the n/2 observations with largest depth. Magnifying the bag by a factor 3 yields the “fence” (which is not plotted). Observations between the bag and the fence are marked by a light gray loop, whereas observations outside the fence are flagged as outliers. The bagplot visualizes the location, spread, correlation, skewness, and tails of the data. It is equivariant for linear transformations, and not limited to elliptical distributions. Software for drawing the bagplot is made available for the S-Plus and MATLAB environments. The bagplot is illustrated on several datasets—for example, in a scatterplot matrix of multivariate data.

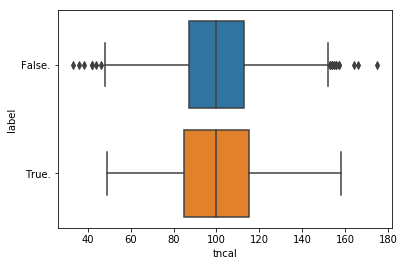


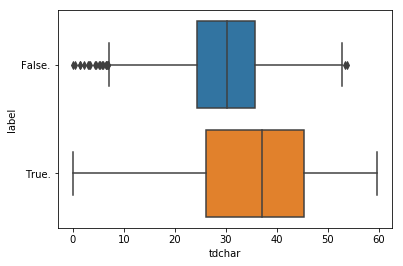


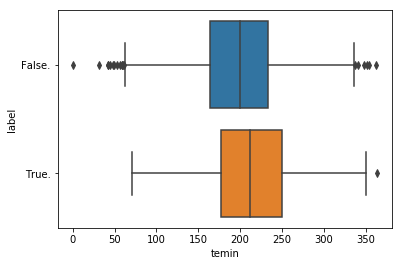


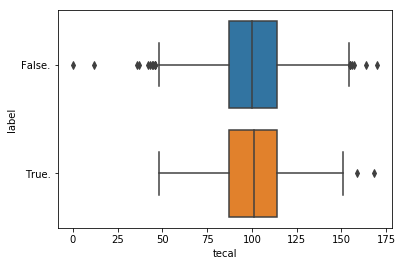


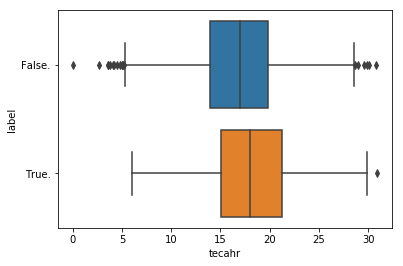


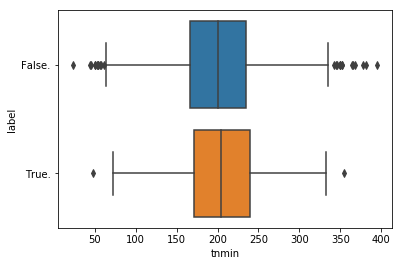


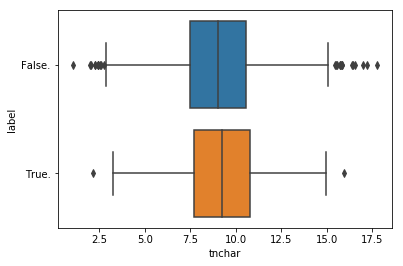


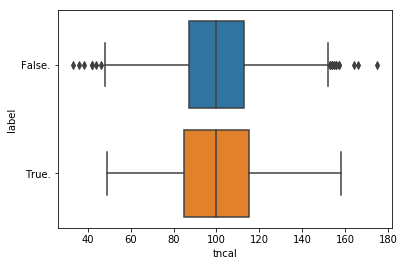


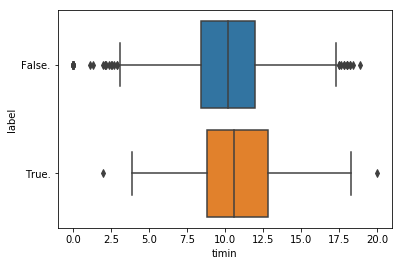


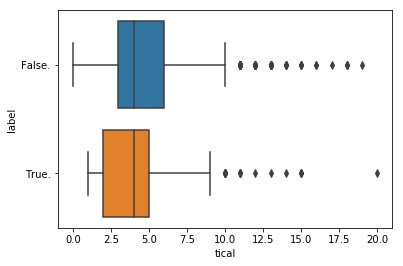


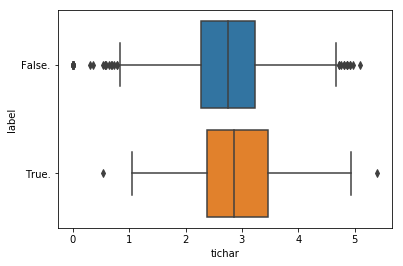


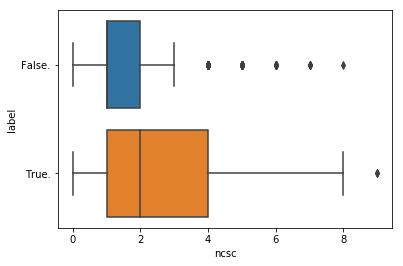












* **TYPES OF BOX PLOT :**

Box and whisker plots [quartiles](https://en.wikipedia.org/wiki/Quartile), and the band inside the box is always the second [quartile](https://en.wikipedia.org/wiki/Quartile) (the [median](https://en.wikipedia.org/wiki/Median)). But the ends of the whiskers can represent several possible alternative values, among them:

* the minimum and maximum of all of the data
* the lowest datum still within 1.5 [IQR](https://en.wikipedia.org/wiki/Interquartile_range) of the lower quartile, and the highest datum still within 1.5 IQR of the upper quartile (often called the Tukey boxplot)
* one [standard deviation](https://en.wikipedia.org/wiki/Standard_deviation) above and below the mean of the data
* the 9th [percentile](https://en.wikipedia.org/wiki/Percentile) and the 91st [percentile](https://en.wikipedia.org/wiki/Percentile)
* the 2nd [percentile](https://en.wikipedia.org/wiki/Percentile) and the 98th [percentile](https://en.wikipedia.org/wiki/Percentile).

Any data not included between the whiskers should be plotted as an outlier with a dot, small circle, or star, but occasionally this is not done.

Some box plots include an additional character to represent the mean of the data (see also Marmolejo-Ramos and Tian, 2010).

On some box plots a crosshatch is placed on each whisker, before the end of the whisker.

Rarely, box plots can be presented with no whiskers at all.

Because of this variability, it is appropriate to describe the convention being used for the whiskers and outliers in the caption for the plot.

The unusual percentiles 2%, 9%, 91%, 98% are sometimes used for whisker cross-hatches and whisker ends to show the [seven-number summary](https://en.wikipedia.org/wiki/Seven-number_summary). If the data are [normally distributed](https://en.wikipedia.org/wiki/Normal_distribution), the locations of the seven marks on the box plot will be equally spaced.

* **VARIATIONS**:

Since the mathematician [John W. Tukey](https://en.wikipedia.org/wiki/John_W._Tukey) introduced this type of visual data display in 1969, several variations on the traditional box plot have been described. Two of the most common are variable width box plots and notched box plots .

Variable width box plots illustrate the size of each group whose data is being plotted by making the width of the box proportional to the size of the group. A popular convention is to make the box width proportional to the square root of the size of the group.

Notched box plots apply a "notch" or narrowing of the box around the median. Notches are useful in offering a rough guide to significance of difference of medians; if the notches of two boxes do not overlap, this offers evidence of a statistically significant difference between the medians. The width of the notches is proportional to the interquartile range (IQR) of the sample and inversely proportional to the square root of the size of the sample. However, there is uncertainty about the most appropriate multiplier (as this may vary depending on the similarity of the variances of the samples). One convention is to use {\displaystyle \pm {\frac {1.58IQR}{\sqrt {n}}}}.

Adjusted box plots are intended for [skew distributions](https://en.wikipedia.org/wiki/Skewness). They rely on the [medcouple](https://en.wikipedia.org/wiki/Medcouple" \o "Medcouple) statistic of skewness. For a medcouple value of MC, the lengths of the upper and lower whiskers are respectively defined to be

{\displaystyle {\begin{matrix}1.5IQR\cdot e^{3MC},&1.5IQR\cdot e^{-4MC}{\text{ if }}MC\geq 0\\1.5IQR\cdot e^{4MC},&1.5IQR\cdot e^{-3MC}{\text{ if }}MC\leq 0\end{matrix}}}Observe that for symmetrical distributions, the medcouple will be zero, and this reduces to Tukey's boxplot with equal whisker lengths of {\displaystyle 1.5IQR} both whiskers.

**COUNT PLOT OF CATEGORICAL FEATURES**

A familiar style of plot that accomplishes this goal is a count plot. In seaborn, the count plot() function operates on a full dataset and applies a function to obtain the estimate (taking the mean by default). When there are multiple observations in each category, it also uses bootstrapping to compute a confidence interval around the estimate and plots that using error bars.

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for **[barplot()](https://seaborn.pydata.org/generated/seaborn.barplot.html" \l "seaborn.barplot" \o "seaborn.barplot)**, so you can compare counts across nested variables.

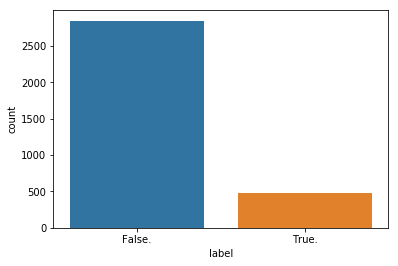
Input data can be passed in a variety of formats, including:

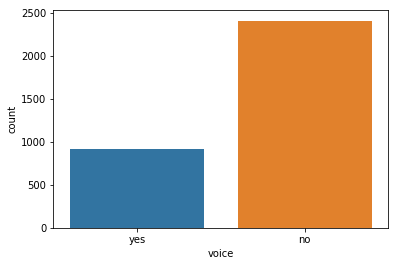
* Vectors of data represented as lists, numpy arrays, or pandas Series objects passed directly to the x, y, and/or hue parameters.
* A “long-form” DataFrame, in which case the x, y, and hue variables will determine how the data are plotted.
* A “wide-form” DataFrame, such that each numeric column will be plotted.
* An array or list of vectors.

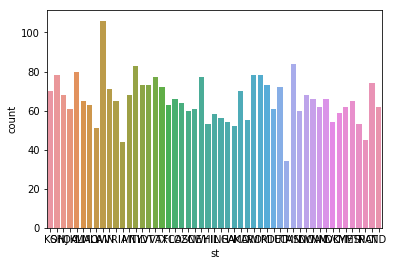
In most cases, it is possible to use numpy or Python objects, but pandas objects are preferable because the associated names will be used to annotate the axes. Additionally, you can use Categorical types for the grouping variables to control the order of plot elements.

This function always treats one of the variables as categorical and draws data at ordinal positions (0, 1, … n) on the relevant axis, even when the data has a numeric or date type.

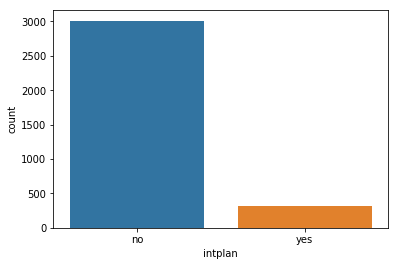
|  |  |
| --- | --- |
| **Parameters:** | **x, y, hue** : names of variables in data or vector data, optional  Inputs for plotting long-form data. See examples for interpretation.  **data** : DataFrame, array, or list of arrays, optional  Dataset for plotting. If x and y are absent, this is interpreted as wide-form. Otherwise it is expected to be long-form.  **order, hue\_order** : lists of strings, optional  Order to plot the categorical levels in, otherwise the levels are inferred from the data objects.  **orient** : “v” | “h”, optional  Orientation of the plot (vertical or horizontal). This is usually inferred from the dtype of the input variables, but can be used to specify when the “categorical” variable is a numeric or when plotting wide-form data.  **color** : matplotlib color, optional  Color for all of the elements, or seed for a gradient palette.  **palette** : palette name, list, or dict, optional  Colors to use for the different levels of the hue variable. Should be something that can be interpreted by **[color\_palette()](https://seaborn.pydata.org/generated/seaborn.color_palette.html" \l "seaborn.color_palette" \o "seaborn.color_palette)**, or a dictionary mapping hue levels to matplotlib colors.  **saturation** : float, optional  Proportion of the original saturation to draw colors at. Large patches often look better with slightly desaturated colors, but set this to 1 if you want the plot colors to perfectly match the input color spec.  **dodge** : bool, optional  When hue nesting is used, whether elements should be shifted along the categorical axis.  **ax** : matplotlib Axes, optional  Axes object to draw the plot onto, otherwise uses the current Axes.  **kwargs** : key, value mappings  Other keyword arguments are passed to plt.bar. |







4)



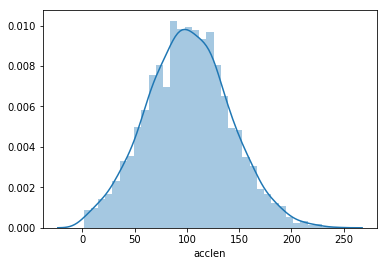
**DIST PLOT TO CHECK NORMALITY**

**seaborn.distplot(*a*, *bins=None*, *hist=True*, *kde=True*, *rug=False*, *fit=None*, *hist\_kws=None*, *kde\_kws=None*, *rug\_kws=None*, *fit\_kws=None*, *color=None*, *vertical=False*, *norm\_hist=False*, *axlabel=None*, *label=None*, *ax=None*)**

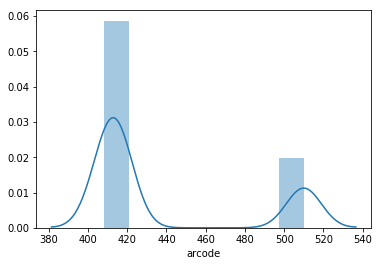
Flexibly plot a univariate distribution of observations.

This function combines the matplotlib hist function (with automatic calculation of a good default bin size) with the seaborn **[kdeplot()](https://seaborn.pydata.org/generated/seaborn.kdeplot.html" \l "seaborn.kdeplot" \o "seaborn.kdeplot)** and **[rugplot()](https://seaborn.pydata.org/generated/seaborn.rugplot.html" \l "seaborn.rugplot" \o "seaborn.rugplot)** functions. It can also fit scipy.stats distributions and plot the estimated PDF over the data.

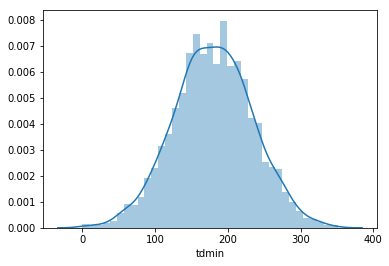
|  |  |
| --- | --- |
| **Parameters:** | **a** : Series, 1d-array, or list.  Observed data. If this is a Series object with a name attribute, the name will be used to label the data axis.  **bins** : argument for matplotlib hist(), or None, optional  Specification of hist bins, or None to use Freedman-Diaconis rule.  **hist** : bool, optional  Whether to plot a (normed) histogram.  **kde** : bool, optional  Whether to plot a gaussian kernel density estimate.  **rug** : bool, optional  Whether to draw a rugplot on the support axis.  **fit** : random variable object, optional  An object with *fit* method, returning a tuple that can be passed to a *pdf* method a positional arguments following an grid of values to evaluate the pdf on.  **{hist, kde, rug, fit}\_kws** : dictionaries, optional  Keyword arguments for underlying plotting functions.  **color** : matplotlib color, optional  Color to plot everything but the fitted curve in.  **vertical** : bool, optional  If True, observed values are on y-axis.  **norm\_hist** : bool, optional  If True, the histogram height shows a density rather than a count. This is implied if a KDE or fitted density is plotted.  **axlabel** : string, False, or None, optional  Name for the support axis label. If None, will try to get it from a.namel if False, do not set a label.  **label** : string, optional  Legend label for the relevent component of the plot  **ax** : matplotlib axis, optional  if provided, plot on this axis. |



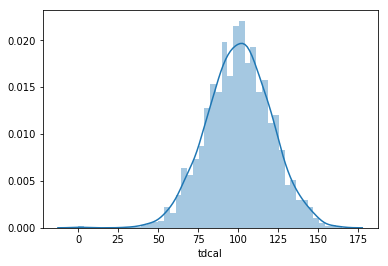
Hence the acclen is normaly distributed.



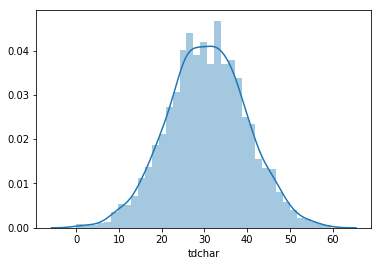
Hence arcode is normally distributed.



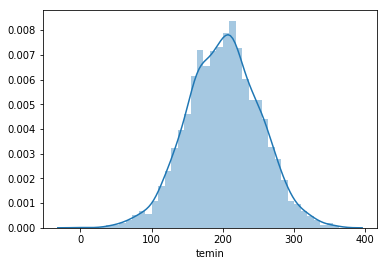
Hence tdmin is normally distributed.



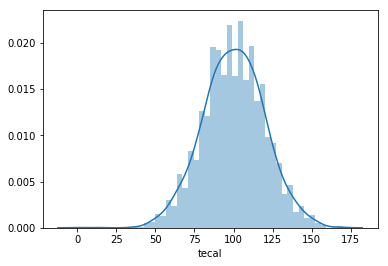
Tdcal is normally distributed.so it is continuous.



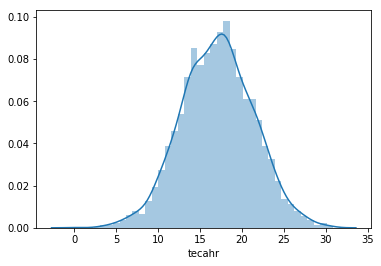
Hence tdchar is normally distributed and continuous.



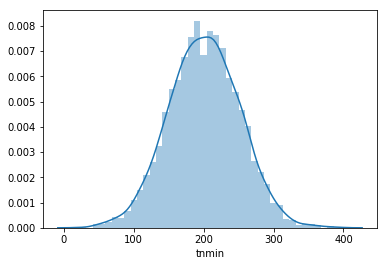
Hence, temin is normally distributed and continuous .



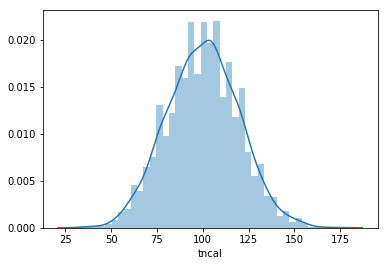
Hence tecal is normally distributed and continuous



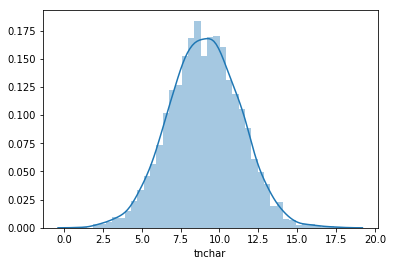
Hence tecahr is normally distributed and continuous.



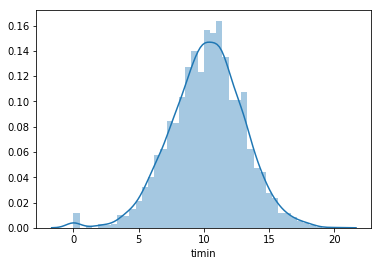
Hence tnmin is normally distributed and continuous



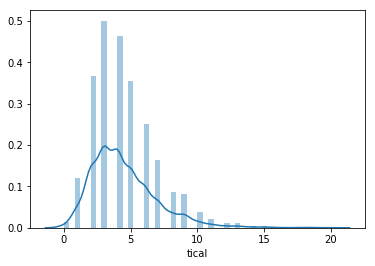
Hence tncal is normally distributed and continuous



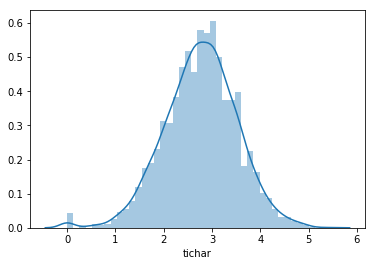
Hence tnchar is normally distributed and continuous



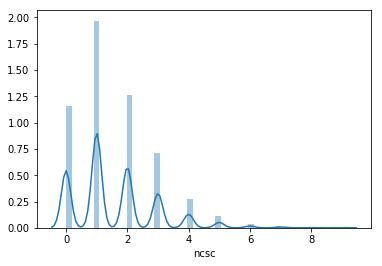
Hence timin is normally distributed and continuous



Hence tical is normally distributed and continuous



Hence tichar is normally distributed and continuous

****

Hence ncsc is not normally distributed and it is discrete.

**5 NOS IN MACHINE LEARNING**

* 1. ACCLEN

count 3333.000000

mean 101.064806

std 39.822106

min 1.000000

25% 74.000000

50% 101.000000

75% 127.000000

max 243.000000

* 1. ARCODE

count 3333.000000

mean 437.182418

std 42.371290

min 408.000000

25% 408.000000

50% 415.000000

75% 510.000000

max 510.000000

3)PHNUM

count 3333

unique 3333

top 394-2445

freq 1

4)TDMIN

count 3333.000000

mean 179.775098

std 54.467389

min 0.000000

25% 143.700000

50% 179.400000

75% 216.400000

max 350.800000

5)TDCAL

count 3333.000000

mean 100.435644

std 20.069084

min 0.000000

25% 87.000000

50% 101.000000

75% 114.000000

max 165.000000

6)TDCHAR

count 3333.000000

mean 30.562307

std 9.259435

min 0.000000

25% 24.430000

50% 30.500000

75% 36.790000

max 59.640000

7)TEMIN

count 3333.000000

mean 200.980348

std 50.713844

min 0.000000

25% 166.600000

50% 201.400000

75% 235.300000

max 363.700000

8)TECAL

count 3333.000000

mean 100.114311

std 19.922625

min 0.000000

25% 87.000000

50% 100.000000

75% 114.000000

max 170.000000

9)TECAHR

count 3333.000000

mean 17.083540

std 4.310668

min 0.000000

25% 14.160000

50% 17.120000

75% 20.000000

max 30.910000

10)TNMIN

count 3333.000000

mean 200.872037

std 50.573847

min 23.200000

25% 167.000000

50% 201.200000

75% 235.300000

max 395.000000

11)TNCAL

count 3333.000000

mean 100.107711

std 19.568609

min 33.000000

25% 87.000000

50% 100.000000

75% 113.000000

max 175.000000

12)TNCHAR

count 3333.000000

mean 9.039325

std 2.275873

min 1.040000

25% 7.520000

50% 9.050000

75% 10.590000

max 17.770000

13)TIMIN

count 3333.000000

mean 10.237294

std 2.791840

min 0.000000

25% 8.500000

50% 10.300000

75% 12.100000

max 20.000000

14)TICAL

count 3333.000000

mean 4.479448

std 2.461214

min 0.000000

25% 3.000000

50% 4.000000

75% 6.000000

max 20.000000

15)TICHAR

count 3333.000000

mean 2.764581

std 0.753773

min 0.000000

25% 2.300000

50% 2.780000

75% 3.270000

max 5.400000

16)NCSC

count 3333.000000

mean 1.562856

std 1.315491

min 0.000000

25% 1.000000

50% 1.000000

75% 2.000000

max 9.000000

17)INTPLAN

count 3333

unique 2

top no

freq 3010

no 3010

yes 323

18)VOICE

count 3333

unique 2

top no

freq 2411

NO 2411

YES 922

19)LABEL

False. 2850

True. 483

20)ST

WV 106

MN 84

NY 83

AL 80

OR 78

OH 78

WI 78

VA 77

WY 77

CT 74

MI 73

ID 73

VT 73

TX 72

UT 72

IN 71

KS 70

MD 70

NC 68

MT 68

NJ 68

CO 66

WA 66

NV 66

RI 65

MA 65

MS 65

AZ 64

FL 63

MO 63

ND 62

NM 62

ME 62

NE 61

DE 61

OK 61

SD 60

SC 60

KY 59

IL 58

NH 56

AR 55

DC 54

GA 54

HI 53

TN 53

AK 52

LA 51

PA 45

IA 44

CA 34

* **DATA ANALYSIS USING FEATURE\_SELECTION**:

We all may have faced this problem of identifying the related features from a set of data and removing the irrelevant or less important features with do not contribute much to our target variable in order to achieve better accuracy for our model.

**Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model.** The data features that you use to train your machine learning models have a huge influence on the performance you can achieve.

Irrelevant or partially relevant features can negatively impact model performance.

Feature selection and Data cleaning should be the first and most important step of your model designing.

In this post, you will discover feature selection techniques that you can use in Machine Learning.

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in.

Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

**How to select features and what are Benefits of performing feature selection before modeling your data?**

· **Reduces Overfitting**: Less redundant data means less opportunity to make decisions based on noise.

· **Improves Accuracy**: Less misleading data means modeling accuracy improves.

· **Reduces Training Time**: fewer data points reduce algorithm complexity and algorithms train faster.

**Feature Selection Methods:**

I will share 3 Feature selection techniques that are easy to use and also gives good results.

1. Univariate Selection

2. Feature Importance

3.Correlation Matrix with Heatmap

**1. Univariate Selection**

Statistical tests can be used to select those features that have the strongest relationship with the output variable.

The scikit-learn library provides the [SelectKBest](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html" \l "sklearn.feature_selection.SelectKBest" \t "_blank) class that can be used with a suite of different statistical tests to select a specific number of features.

The example below uses the chi-squared (chi²) statistical test for non-negative features to select 10 of the best features from the Mobile Price Range Prediction Dataset.

#### 2. Feature Importance

You can get the feature importance of each feature of your dataset by using the feature importance property of the model.

Feature importance gives you a score for each feature of your data, the higher the score more important or relevant is the feature towards your output variable.

Feature importance is an inbuilt class that comes with Tree Based Classifiers, we will be using Extra Tree Classifier for extracting the top 10 features for the dataset.

#### **3.Correlation Matrix with Heatmap**

Correlation states how the features are related to each other or the target variable.

Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable)

Heatmap makes it easy to identify which features are most related to the target variable, we will plot heatmap of correlated features using the seaborn library.

Have a look at the last row i.e price range, see how the price range is correlated with other features, ram is the highly correlated with price range followed by battery power, pixel height and width while m\_dep, clock\_speed and n\_cores seems to be least correlated with price\_range.

In this article we have discovered how to select relevant features from data using Univariate Selection technique, feature importance and correlation matrix.

## **Subset selection**

Subset selection evaluates a subset of features as a group for suitability. Subset selection algorithms can be broken up into Wrappers, Filters and Embedded. Wrappers use a search algorithm to search through the space of possible features and evaluate each subset by running a model on the subset. Wrappers can be computationally expensive and have a risk of over fitting to the model. Filters are similar to Wrappers in the search approach, but instead of evaluating against a model, a simpler filter is evaluated. Embedded techniques are embedded in and specific to a model.

Many popular search approaches use [greedy](https://en.wikipedia.org/wiki/Greedy_algorithm) [hill climbing](https://en.wikipedia.org/wiki/Hill_climbing), which iteratively evaluates a candidate subset of features, then modifies the subset and evaluates if the new subset is an improvement over the old. Evaluation of the subsets requires a scoring [metric](https://en.wikipedia.org/wiki/Metric_(mathematics)) that grades a subset of features. Exhaustive search is generally impractical, so at some implementor (or operator) defined stopping point, the subset of features with the highest score discovered up to that point is selected as the satisfactory feature subset. The stopping criterion varies by algorithm; possible criteria include: a subset score exceeds a threshold, a program's maximum allowed run time has been surpassed, etc.

Alternative search-based techniques are based on [targeted projection pursuit](https://en.wikipedia.org/wiki/Targeted_projection_pursuit) which finds low-dimensional projections of the data that score highly: the features that have the largest projections in the lower-dimensional space are then selected.

Search approaches include:

* Exhaustive
* [Best first](https://en.wikipedia.org/wiki/Best-first_search)
* [Simulated annealing](https://en.wikipedia.org/wiki/Simulated_annealing)
* [Genetic algorithm](https://en.wikipedia.org/wiki/Genetic_algorithm)
* [Greedy](https://en.wikipedia.org/wiki/Greedy_algorithm) forward selection
* Greedy backward elimination
* [Particle swarm optimization](https://en.wikipedia.org/wiki/Particle_swarm_optimization)
* [Targeted projection pursuit](https://en.wikipedia.org/wiki/Targeted_projection_pursuit)
* Scatter Search
* [Variable Neighborhood Search](https://en.wikipedia.org/wiki/Variable_Neighborhood_Search)

Two popular filter metrics for classification problems are [correlation](https://en.wikipedia.org/wiki/Correlation) and [mutual information](https://en.wikipedia.org/wiki/Mutual_information), although neither are true [metrics](https://en.wikipedia.org/wiki/Metric_(mathematics)) or 'distance measures' in the mathematical sense, since they fail to obey the [triangle inequality](https://en.wikipedia.org/wiki/Triangle_inequality) and thus do not compute any actual 'distance' – they should rather be regarded as 'scores'. These scores are computed between a candidate feature (or set of features) and the desired output category. There are, however, true metrics that are a simple function of the mutual information;

Other available filter metrics include:

* Class separability
  + Error probability
  + Inter-class distance
  + Probabilistic distance
  + [Entropy](https://en.wikipedia.org/wiki/Entropy_(Information_theory))
* Consistency-based feature selection
* Correlation-based feature selection

## **Optimality criteria**

The choice of optimality criteria is difficult as there are multiple objectives in a feature selection task. Many common criteria incorporate a measure of accuracy, penalised by the number of features selected. Examples include [Akaike information criterion](https://en.wikipedia.org/wiki/Akaike_information_criterion) (AIC) and [Mallows's](https://en.wikipedia.org/wiki/Mallows%27s_Cp" \o "Mallows's Cp)*[C](https://en.wikipedia.org/wiki/Mallows%27s_Cp" \o "Mallows's Cp)[p](https://en.wikipedia.org/wiki/Mallows%27s_Cp" \o "Mallows's Cp)*, which have a penalty of 2 for each added feature. AIC is based on [information theory](https://en.wikipedia.org/wiki/Information_theory), and is effectively derived via the [maximum entropy principle](https://en.wikipedia.org/wiki/Maximum_entropy_principle).

Other criteria are [Bayesian information criterion](https://en.wikipedia.org/wiki/Bayesian_information_criterion) (BIC), which uses a penalty of {\displaystyle {\sqrt {\log {n}}}} for each added feature, [minimum description length](https://en.wikipedia.org/wiki/Minimum_description_length) (MDL) which asymptotically uses {\displaystyle {\sqrt {\log {n}}}}, [Bonferroni](https://en.wikipedia.org/wiki/Bonferroni) / RIC which use {\displaystyle {\sqrt {2\log {p}}}}, maximum dependency feature selection, and a variety of new criteria that are motivated by [false discovery rate](https://en.wikipedia.org/wiki/False_discovery_rate) (FDR), which use something close to {\displaystyle {\sqrt {2\log {\frac {p}{q}}}}}. A maximum [entropy rate](https://en.wikipedia.org/wiki/Entropy_rate)criterion may also be used to select the most relevant subset of features.

REMOVING FEATURES OF LOW VARIENCE:

[**VarianceThreshold**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html#sklearn.feature_selection.VarianceThreshold) is a simple baseline approach to feature selection. It removes all features whose variance doesn’t meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples.

As an example, suppose that we have a dataset with boolean features, and we want to remove all features that are either one or zero (on or off) in more than 80% of the samples. Boolean features are Bernoulli random variables, and the variance of such variables is given by

Var[X]=p(1−p)

so we can select using the threshold .8 \* (1 - .8):

>>>

**>>> from** **sklearn.feature\_selection** **import** VarianceThreshold

**>>>** X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0], [0, 1, 1]]

**>>>** sel = VarianceThreshold(threshold=(.8 \* (1 - .8)))

**>>>** sel.fit\_transform(X)

array([[0, 1],

[1, 0],

[0, 0],

[1, 1],

[1, 0],

[1, 1]])

As expected, VarianceThreshold has removed the first column, which has a probability p=5/6>.8 of containing a zero.

## 1.13.2. Univariate feature selection[¶](https://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection)

Univariate feature selection works by selecting the best features based on univariate statistical tests. It can be seen as a preprocessing step to an estimator. Scikit-learn exposes feature selection routines as objects that implement the transformmethod:

* [**SelectKBest**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest) removes all but the k highest scoring features
* [**SelectPercentile**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectPercentile.html#sklearn.feature_selection.SelectPercentile) removes all but a user-specified highest scoring percentage of features
* using common univariate statistical tests for each feature: false positive rate **[SelectFpr](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFpr.html" \l "sklearn.feature_selection.SelectFpr" \o "sklearn.feature_selection.SelectFpr)**, false discovery rate **[SelectFdr](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFdr.html" \l "sklearn.feature_selection.SelectFdr" \o "sklearn.feature_selection.SelectFdr)**, or family wise error **[SelectFwe](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFwe.html" \l "sklearn.feature_selection.SelectFwe" \o "sklearn.feature_selection.SelectFwe)**.
* [**GenericUnivariateSelect**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.GenericUnivariateSelect.html#sklearn.feature_selection.GenericUnivariateSelect) allows to perform univariate feature selection with a configurable strategy. This allows to select the best univariate selection strategy with hyper-parameter search estimator.

For instance, we can perform a χ2 test to the samples to retrieve only the two best features as follows:

>>>

**>>> from** **sklearn.datasets** **import** load\_iris

**>>> from** **sklearn.feature\_selection** **import** SelectKBest

**>>> from** **sklearn.feature\_selection** **import** chi2

**>>>** iris = load\_iris()

**>>>** X, y = iris.data, iris.target

**>>>** X.shape

(150, 4)

**>>>** X\_new = SelectKBest(chi2, k=2).fit\_transform(X, y)

**>>>** X\_new.shape

(150, 2)

These objects take as input a scoring function that returns univariate scores and p-values (or only scores for **[SelectKBest](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html" \l "sklearn.feature_selection.SelectKBest" \o "sklearn.feature_selection.SelectKBest)**and **[SelectPercentile](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectPercentile.html" \l "sklearn.feature_selection.SelectPercentile" \o "sklearn.feature_selection.SelectPercentile)**):

* For regression: **[f\_regression](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_regression.html" \l "sklearn.feature_selection.f_regression" \o "sklearn.feature_selection.f_regression)**, **[mutual\_info\_regression](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_regression.html" \l "sklearn.feature_selection.mutual_info_regression" \o "sklearn.feature_selection.mutual_info_regression)**
* For classification: [**chi2**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html#sklearn.feature_selection.chi2), **[f\_classif](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html" \l "sklearn.feature_selection.f_classif" \o "sklearn.feature_selection.f_classif)**, **[mutual\_info\_classif](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html" \l "sklearn.feature_selection.mutual_info_classif" \o "sklearn.feature_selection.mutual_info_classif)**

The methods based on F-test estimate the degree of linear dependency between two random variables. On the other hand, mutual information methods can capture any kind of statistical dependency, but being nonparametric, they require more samples for accurate estimation.

**Feature selection with sparse data**

If you use sparse data (i.e. data represented as sparse matrices), [**chi2**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html#sklearn.feature_selection.chi2), [**mutual\_info\_regression**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_regression.html#sklearn.feature_selection.mutual_info_regression), [**mutual\_info\_classif**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html#sklearn.feature_selection.mutual_info_classif) will deal with the data without making it dense.

## 1.13.3. Recursive feature elimination

Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), recursive feature elimination ([**RFE**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html#sklearn.feature_selection.RFE)) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through a coef\_ attribute or through a feature\_importances\_ attribute. Then, the least important features are pruned from current set of features.That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

[**RFECV**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html#sklearn.feature_selection.RFECV) performs RFE in a cross-validation loop to find the optimal number of features.

**Examples:**

* [Recursive feature elimination](https://scikit-learn.org/stable/auto_examples/feature_selection/plot_rfe_digits.html#sphx-glr-auto-examples-feature-selection-plot-rfe-digits-py): A recursive feature elimination example showing the relevance of pixels in a digit classification task.
* [Recursive feature elimination with cross-validation](https://scikit-learn.org/stable/auto_examples/feature_selection/plot_rfe_with_cross_validation.html#sphx-glr-auto-examples-feature-selection-plot-rfe-with-cross-validation-py): A recursive feature elimination example with automatic tuning of the number of features selected with cross-validation.

## 1.13.4. Feature selection using SelectFromModel

[**SelectFromModel**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html#sklearn.feature_selection.SelectFromModel) is a meta-transformer that can be used along with any estimator that has a coef\_ or feature\_importances\_ attribute after fitting. The features are considered unimportant and removed, if the correspondingcoef\_ or feature\_importances\_ values are below the provided threshold parameter. Apart from specifying the threshold numerically, there are built-in heuristics for finding a threshold using a string argument. Available heuristics are “mean”, “median” and float multiples of these like “0.1\*mean”.

For examples on how it is to be used refer to the sections below.

**Examples**

* [Feature selection using SelectFromModel and LassoCV](https://scikit-learn.org/stable/auto_examples/feature_selection/plot_select_from_model_boston.html#sphx-glr-auto-examples-feature-selection-plot-select-from-model-boston-py): Selecting the two most important features from the Boston dataset without knowing the threshold beforehand.

### 1.13.4.1. L1-based feature selection

[Linear models](https://scikit-learn.org/stable/modules/linear_model.html#linear-model) penalized with the L1 norm have sparse solutions: many of their estimated coefficients are zero. When the goal is to reduce the dimensionality of the data to use with another classifier, they can be used along with **[feature\_selection.SelectFromModel](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html" \l "sklearn.feature_selection.SelectFromModel" \o "sklearn.feature_selection.SelectFromModel)** to select the non-zero coefficients. In particular, sparse estimators useful for this purpose are the **[linear\_model.Lasso](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html" \l "sklearn.linear_model.Lasso" \o "sklearn.linear_model.Lasso)** for regression, and of **[linear\_model.LogisticRegression](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html" \l "sklearn.linear_model.LogisticRegression" \o "sklearn.linear_model.LogisticRegression)** and **[svm.LinearSVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html" \l "sklearn.svm.LinearSVC" \o "sklearn.svm.LinearSVC)** for classification:

>>>

**>>> from** **sklearn.svm** **import** LinearSVC

**>>> from** **sklearn.datasets** **import** load\_iris

**>>> from** **sklearn.feature\_selection** **import** SelectFromModel

**>>>** iris = load\_iris()

**>>>** X, y = iris.data, iris.target

**>>>** X.shape

(150, 4)

**>>>** lsvc = LinearSVC(C=0.01, penalty="l1", dual=**False**).fit(X, y)

**>>>** model = SelectFromModel(lsvc, prefit=**True**)

**>>>** X\_new = model.transform(X)

**>>>** X\_new.shape

(150, 3)

With SVMs and logistic-regression, the parameter C controls the sparsity: the smaller C the fewer features selected. With Lasso, the higher the alpha parameter, the fewer features selected.

**Examples:**

* [Classification of text documents using sparse features](https://scikit-learn.org/stable/auto_examples/text/plot_document_classification_20newsgroups.html#sphx-glr-auto-examples-text-plot-document-classification-20newsgroups-py): Comparison of different algorithms for document classification including L1-based feature selection.

**L1-recovery and compressive sensing**

For a good choice of alpha, the [Lasso](https://scikit-learn.org/stable/modules/linear_model.html#lasso) can fully recover the exact set of non-zero variables using only few observations, provided certain specific conditions are met. In particular, the number of samples should be “sufficiently large”, or L1 models will perform at random, where “sufficiently large” depends on the number of non-zero coefficients, the logarithm of the number of features, the amount of noise, the smallest absolute value of non-zero coefficients, and the structure of the design matrix X. In addition, the design matrix must display certain specific properties, such as not being too correlated.

There is no general rule to select an alpha parameter for recovery of non-zero coefficients. It can by set by cross-validation (**LassoCV** or **LassoLarsCV**), though this may lead to under-penalized models: including a small number of non-relevant variables is not detrimental to prediction score. BIC (**LassoLarsIC**) tends, on the opposite, to set high values of alpha.

**Reference** Richard G. Baraniuk “Compressive Sensing”, IEEE Signal Processing Magazine [120] July 2007<http://users.isr.ist.utl.pt/~aguiar/CS_notes.pdf>

### 1.13.4.2. Tree-based feature selection

Tree-based estimators (see the **[sklearn.tree](https://scikit-learn.org/stable/modules/classes.html" \l "module-sklearn.tree" \o "sklearn.tree)** module and forest of trees in the **[sklearn.ensemble](https://scikit-learn.org/stable/modules/classes.html" \l "module-sklearn.ensemble" \o "sklearn.ensemble)** module) can be used to compute feature importances, which in turn can be used to discard irrelevant features (when coupled with the **[sklearn.feature\_selection.SelectFromModel](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html" \l "sklearn.feature_selection.SelectFromModel" \o "sklearn.feature_selection.SelectFromModel)** meta-transformer):

>>>

**>>> from** **sklearn.ensemble** **import** ExtraTreesClassifier

**>>> from** **sklearn.datasets** **import** load\_iris

**>>> from** **sklearn.feature\_selection** **import** SelectFromModel

**>>>** iris = load\_iris()

**>>>** X, y = iris.data, iris.target

**>>>** X.shape

(150, 4)

**>>>** clf = ExtraTreesClassifier(n\_estimators=50)

**>>>** clf = clf.fit(X, y)

**>>>** clf.feature\_importances\_

array([ 0.04..., 0.05..., 0.4..., 0.4...])

**>>>** model = SelectFromModel(clf, prefit=**True**)

**>>>** X\_new = model.transform(X)

**>>>** X\_new.shape

(150, 2)

**Examples:**

* [Feature importances with forests of trees](https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html#sphx-glr-auto-examples-ensemble-plot-forest-importances-py): example on synthetic data showing the recovery of the actually meaningful features.
* [Pixel importances with a parallel forest of trees](https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances_faces.html#sphx-glr-auto-examples-ensemble-plot-forest-importances-faces-py): example on face recognition data.

## 1.13.5. Feature selection as part of a pipeline

Feature selection is usually used as a pre-processing step before doing the actual learning. The recommended way to do this in scikit-learn is to use a **[sklearn.pipeline.Pipeline](https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html" \l "sklearn.pipeline.Pipeline" \o "sklearn.pipeline.Pipeline)**:

clf = Pipeline([

('feature\_selection', SelectFromModel(LinearSVC(penalty="l1"))),

('classification', RandomForestClassifier())

])

clf.fit(X, y)

In this snippet we make use of a **[sklearn.svm.LinearSVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html" \l "sklearn.svm.LinearSVC" \o "sklearn.svm.LinearSVC)** coupled with **[sklearn.feature\_selection.SelectFromModel](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html" \l "sklearn.feature_selection.SelectFromModel" \o "sklearn.feature_selection.SelectFromModel)** to evaluate feature importances and select the most relevant features. Then, a **[sklearn.ensemble.RandomForestClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html" \l "sklearn.ensemble.RandomForestClassifier" \o "sklearn.ensemble.RandomForestClassifier)** is trained on the transformed output, i.e. using only relevant features. You can perform similar operations with the other feature selection methods and also classifiers that provide a way to evaluate feature importances of course. See the **[sklearn.pipeline.Pipeline](https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html" \l "sklearn.pipeline.Pipeline" \o "sklearn.pipeline.Pipeline)** examples for more detail

**CODE**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt # ploting , visualization

import seaborn as sns # ploting

from sklearn import model\_selection #scikit learn

from sklearn import linear\_model

from sklearn import metrics

from sklearn import preprocessing

from sklearn import utils

from sklearn import tree

from sklearn import feature\_selection

df=pd.read\_csv("c:/projectdata\_machinelearning/customer\_churn/churn\_train.csv")

df.info()

f=open("c:\\projectdata\_machinelearning\\customer\_churn\\churn\_test.txt","r")

records=f.readlines()

print(records)

import re

nf=open("c:\\projectdata\_machinelearning\\customer\_churn\\churn\_test.csv","w")

with open("c:\\projectdata\_machinelearning\\customer\_churn\\churn\_test.txt","r") as f:

for record in f:

newrec=re.sub('\s+','',record)

nf.write(newrec+ "\n")

nf.close()

df1=pd.read\_csv("c:/projectdata\_machinelearning/customer\_churn/churn\_test.csv")

df1.info()

df.voice.value\_counts()

repldict={"yes":1,"no":0}

df.voice.replace(repldict,inplace=True)

df.voice.value\_counts()

df.intplan.value\_counts()

repldict1={"yes":1,"no":0}

df.intplan.replace(repldict1,inplace=True)

df.intplan.value\_counts()

df1.voice.value\_counts()

repldict2={"yes":1,"no":0}

df1.voice.replace(repldict2,inplace=True)

df1.voice.value\_counts()

df1.intplan.value\_counts()

repldict3={"yes":1,"no":0}

df1.intplan.replace(repldict3,inplace=True)

df1.intplan.value\_counts()

df1.label.value\_counts()

repldict4={"True.":1,"False.":0}

df1.label.replace(repldict4,inplace=True)

df1.label.value\_counts()

df.acclen.describe()

df.arcode.describe()

df.phnum.describe()

df.tdmin.describe()### min 0

df.tdmin.isnull().sum()

df.tdcal.describe()##min 0

df.tdcal.isnull().sum()

sns.boxplot(y=df["acclen"])## min and max outlier

sns.boxplot(y=df["arcode"])## min and max outlier

df.tdchar.describe()

sns.boxplot(y=df["tdchar"])### outlier

df.temin.describe()

sns.boxplot(y=df["temin"])##outlier

df.tecal.describe()

sns.boxplot(y=df["tecal"])##outlier

df.tecahr.describe()

sns.boxplot(y=df["tecahr"])##outlier

df.tnmin.describe()

df.tncal.describe()

df.tnchar.describe()

df.timin.describe()

sns.boxplot(y=df["tnchar"])##outlier

df.tical.describe()

sns.boxplot(y=df["tical"])##outlier(max)

df.tichar.describe()

sns.boxplot(y=df["tichar"])##outlier

df.ncsc.describe()

sns.boxplot(y=df["voice"])##outlier(max)

df.intplan.describe()

df.ncsc.describe()

df.voice.describe()

sns.countplot(x="label",data=df)

sns.countplot(x="voice",data=df)

sns.countplot(x="st",data=df)

sns.countplot(x="intplan",data=df)

df.label.value\_counts()## f=>2850,t=>483

df.voice.value\_counts()## no=>2411,yes=>922

df.st.value\_counts()

df.intplan.value\_counts()## no=>3010,yes=>323

un=df.st.unique()###51 unique state

un

df.ncsc.value\_counts()

sns.boxplot(x="acclen",y="label",data=df)

sns.boxplot(x="arcode",y="label",data=df)

sns.boxplot(x="phnum",y="label",data=df)## numeric variable not appeared

sns.boxplot(x="intplan",y="label",data=df)### numeric variable not present

sns.boxplot(x="voice",y="label",data=df)## numeric variable not present

sns.boxplot(x="arcode",y="label",data=df)

sns.boxplot(x="nummailmes",y="label",data=df)

sns.boxplot(x="tdmin",y="label",data=df)

sns.boxplot(x="tdcal",y="label",data=df)

sns.boxplot(x="tdchar",y="label",data=df)

sns.boxplot(x="temin",y="label",data=df)

sns.boxplot(x="tecal",y="label",data=df)

sns.boxplot(x="tecahr",y="label",data=df)

sns.boxplot(x="tnmin",y="label",data=df)

sns.boxplot(x="tnchar",y="label",data=df)

sns.boxplot(x="tncal",y="label",data=df)

sns.boxplot(x="timin",y="label",data=df)

sns.boxplot(x="tical",y="label",data=df)

sns.boxplot(x="tichar",y="label",data=df)

sns.boxplot(x="ncsc",y="label",data=df)

df.label.value\_counts()

repldict5={"True.":1,"False.":0}

df.label.replace(repldict5,inplace=True)

df.label.value\_counts()

df.ncsc.value\_counts()

replidict6={5:4,6:4,7:4,8:4,9:4}

df.ncsc.replace(replidict6,inplace=True)

df.ncsc.value\_counts()

df1.ncsc.value\_counts()

replidict7={5:4,6:4,7:4,8:4,9:4}

df1.ncsc.replace(replidict7,inplace=True)

df.ncsc.value\_counts()

Xtrain=df.drop(["label","st","phnum","tncal","arcode","tecal","acclen","tdcal","tnmin"],axis=1)

Xtrain.info()

Xtrain.shape

Xtest=df1.drop(["label","st","phnum","tncal","arcode","tecal","acclen","tdcal","tnmin"],axis=1)

Xtest.info()

Ytrain=df["label"]

Ytrain[:5]

Ytest=df1["label"]

Ytest[:5]

Xtrain.corr()

## sns.heatmap(Xtrain.corr())

sns.distplot(Xtrain["acclen"])

sns.distplot(Xtrain["arcode"])

sns.distplot(Xtrain["nummailmes"])

sns.distplot(Xtrain["tdmin"])

sns.distplot(Xtrain["tdcal"])

sns.distplot(Xtrain["tdchar"])

sns.distplot(Xtrain["temin"])

sns.distplot(Xtrain["tecal"])

sns.distplot(Xtrain["tecahr"])

sns.distplot(Xtrain["tnmin"])

sns.distplot(Xtrain["tncal"])

sns.distplot(Xtrain["tnchar"])

sns.distplot(Xtrain["timin"])

sns.distplot(Xtrain["tical"])

sns.distplot(Xtrain["tichar"])

sns.distplot(Xtrain["ncsc"])

df.ncsc.value\_counts()

Xtrain.info()

Xtest.info()

Xtrain.info()

Xtest.info()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=17)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=16)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=15)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=14)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=13)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=1)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=9)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=12)

obj.fit(Xtrain,Ytrain)

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=8)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=7)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=6)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=5)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=4)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=3)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

obj=feature\_selection.SelectKBest(feature\_selection.f\_classif,k=2)

obj.fit(Xtrain,Ytrain)

obj.get\_support()

from sklearn import tree

from sklearn import neighbors

from sklearn import naive\_bayes

def modelstats(Xtrain,Xtest,Ytrain,Ytest):

stats=[]

modelnames=["LR","DecisionTree","KNN","NB"]

models=list()

models.append(linear\_model.LogisticRegression())

models.append(tree.DecisionTreeClassifier())

models.append(neighbors.KNeighborsClassifier())

models.append(naive\_bayes.GaussianNB())

for name,model in zip(modelnames,models):

model.fit(Xtrain,Ytrain)

trainprediction=model.predict(Xtrain)

testprediction=model.predict(Xtest)

scores=list()

scores.append(name+"-train")

scores.append(metrics.accuracy\_score(Ytrain,trainprediction))

scores.append(metrics.precision\_score(Ytrain,trainprediction))

scores.append(metrics.recall\_score(Ytrain,trainprediction))

scores.append(metrics.roc\_auc\_score(Ytrain,trainprediction))

stats.append(scores)

scores=list()

scores.append(name+"-test")

scores.append(metrics.accuracy\_score(Ytest,testprediction))

scores.append(metrics.precision\_score(Ytest,testprediction))

scores.append(metrics.recall\_score(Ytest,testprediction))

scores.append(metrics.roc\_auc\_score(Ytest,testprediction))

stats.append(scores)

colnames=["MODELNAME","ACCURACY","PRECISION","RECALL","AUC"]

return pd.DataFrame(stats,columns=colnames)

resultprint=modelstats(Xtrain,Xtest,Ytrain,Ytest)

resultprint

alldepths=[i for i in range(2,120)]

grid={"max\_depth":alldepths}

model=tree.DecisionTreeClassifier()

gridobj=model\_selection.GridSearchCV(estimator=model,param\_grid=grid,scoring="recall")

gridobj.fit(Xtrain,Ytrain)

bestmodel=gridobj.best\_estimator\_

gridobj.best\_params\_

print("recal: ",metrics.recall\_score(Ytest,bestmodel.predict(Xtest)))

print("AUC: ",metrics.roc\_auc\_score(Ytest,bestmodel.predict(Xtest)))

fpr,tpr,threshold=metrics.roc\_curve(Ytest,bestmodel.predict(Xtest))

plt.plot(fpr,tpr)

model.fit(Xtrain,Ytrain)

predict\_train=model.predict(Xtrain)

metrics.roc\_auc\_score(Ytrain,predict\_train)

fpr1,tpr1,threshhold=metrics.roc\_curve(Ytrain,predict\_train)

**OUTPUT**

1)TRAIN DATAFRAME:

df1=pd.read\_csv("C:\\Users\\hp\\Desktop\\python\\projectdata\_machinelearning\\customer\_churn\\churn\_test.csv")

df1[:5]

Out[91]:

st acclen arcode phnum ... tical tichar ncsc label

0 HI 101 510 354-8815 ... 3 2.86 3 False.

1 MT 137 510 381-7211 ... 7 2.57 0 False.

2 OH 103 408 411-9481 ... 6 3.70 1 False.

3 NM 99 415 418-9100 ... 2 4.24 1 False.

4 SC 108 415 413-3643 ... 4 2.08 2 False.

2)TEST DATAFRAME:

df1[:5]

Out[92]:

st acclen arcode phnum ... tical tichar ncsc label

0 HI 101 510 354-8815 ... 3 2.86 3 False.

1 MT 137 510 381-7211 ... 7 2.57 0 False.

2 OH 103 408 411-9481 ... 6 3.70 1 False.

3 NM 99 415 418-9100 ... 2 4.24 1 False.

4 SC 108 415 413-3643 ... 4 2.08 2 False.

* 1. TO CHANGE STRING INTO INTEGER:

df.voice.value\_counts()

Out[94]:

0 2411

1 922

Name: voice, dtype: int64

df.intplan.value\_counts()

Out[13]:

0 3010

1. 323

df.label.value\_counts()

Out[17]:

0 2850

1. 483

df1.voice.value\_counts()

Out[21]:

0 1266

1. 401

df1.intplan.value\_counts()

Out[22]:

0 1517

1. 150

df1.intplan.value\_counts()

Out[25]:

0 1517

1. 150

df1.label.value\_counts()

Out[29]:

0 1443

1. 224

4) DESCRIPTION OF XTRAIN,YTRAIN:

Xtrain.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 3333 entries, 0 to 3332

Data columns (total 19 columns):

st 3333 non-null object

acclen 3333 non-null int64

arcode 3333 non-null int64

intplan 3333 non-null object

voice 3333 non-null object

nummailmes 3333 non-null int64

tdmin 3333 non-null float64

tdcal 3333 non-null int64

tdchar 3333 non-null float64

temin 3333 non-null float64

tecal 3333 non-null int64

tecahr 3333 non-null float64

tnmin 3333 non-null float64

tncal 3333 non-null int64

tnchar 3333 non-null float64

timin 3333 non-null float64

tical 3333 non-null int64

tichar 3333 non-null float64

ncsc 3333 non-null int64

dtypes: float64(8), int64(8), object(3)

(3333, 12)

XTEST:

Data columns (total 12 columns):

intplan 1667 non-null int64

voice 1667 non-null int64

nummailmes 1667 non-null int64

tdmin 1667 non-null float64

tdchar 1667 non-null float64

temin 1667 non-null float64

tecahr 1667 non-null float64

tnchar 1667 non-null float64

timin 1667 non-null float64

tical 1667 non-null int64

tichar 1667 non-null float64

ncsc 1667 non-null int64

dtypes: float64(7), int64(5)

YTRAIN:

0 0

1 0

2 0

3 0

4 0

YTEST:

0 0

1 0

2 0

3 0

4 0

CORRELATION:

intplan voice ... tichar ncsc

intplan 1.000000 0.006006 ... 0.045780 -0.024522

voice 0.006006 1.000000 ... -0.001276 -0.017824

nummailmes 0.008745 0.956927 ... 0.002884 -0.013263

tdmin 0.049396 -0.001684 ... -0.010092 -0.013423

tdchar 0.049398 -0.001686 ... -0.010094 -0.013427

temin 0.019100 0.021545 ... -0.011067 -0.012985

tecahr 0.019106 0.021559 ... -0.011074 -0.012987

tnchar -0.028913 0.006064 ... -0.015186 -0.009277

timin 0.045871 -0.001318 ... 0.999993 -0.009640

tical 0.017366 0.007618 ... 0.032372 -0.017561

tichar 0.045780 -0.001276 ... 1.000000 -0.009675

ncsc -0.024522 -0.017824 ... -0.009675 1.000000

5)FEATURE SELECTION:

1. INTPLAN
2. NCSC
3. TDMIN
4. TDCHAR
5. TEMIN
6. VOICE
7. TECHAR
8. NUMMAILMES
9. TICHAR
10. TIMIN
11. TICAL
12. TECAHR
13. TNMIN
14. TDCAL
15. ACCLEN
16. TECAL
17. ARCODE
18. TNCAL

6) MODEL:

MODELNAME ACCURACY PRECISION RECALL AUC

LR-train 0.862586 0. 583893 0.180124 0.579185

LR-test 0.873425 0.597015 0.178571 0.579930

DecisionTree-

train 1.000000 1.000000 1.000000 1.000000

DecisionTree-

Test 0.923815 0.713656 0.723214 0.839085

KNN-train 0.904890 0.834677 0.428571 0.707093

KNN-test 0.904019 0.796296 0.383929 0.684341

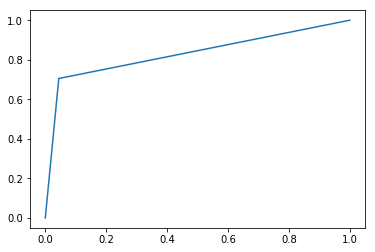
NB-train 0.867087 0.541494 0.540373 0.731414

NB-test 0.877624 0.545045 0.540179 0.735093

HERE WE SELECT THE DECISION-TREE MODEL BECAUSE IN OUR PROBLEM WE HAVE TO INCREASE RECALL TO DECRESE THE LOSS.SO WE CHOOSE DECISION TREE IS BEST FOR OUR PROBLEM AS IN THIS MODEL RECAL IS MORE THAN THE OTHERS.

SO , WE SELECT THE DECISION TREE MODEL.

* **ROC CURVE** :



ROC CURVE FOR TEST SET

### **What is customer churn?**

**Customer churn (or customer attrition)**is a tendency of customers to abandon a brand and stop being a paying client of a particular business.

The percentage of customers that discontinue using a company’s products or services during a particular time period is called a customer churn (attrition) rate.

One of the ways to calculate a churn rate is to divide the number of customers lost during a given time interval by the number of acquired customers, and then multiply that number by 100 percent. For example, if you got 150 customers and lost three last month, then your monthly churn rate is 2 percent.

Churn rate is a health indicator for businesses whose customers are subscribers and paying for services on a recurring basis, notes head of data analytics department at ScienceSoft [Alex Bekker](https://twitter.com/alexlbekker), “Customers [of subscription-driven businesses] opt for a product or a service for a particular period, which can be rather short – say, a month. Thus, a customer stays open for more interesting or advantageous offers. Plus, each time their current commitment ends, customers have a chance to reconsider and choose not to continue with the company. Of course, some natural churn is inevitable, and the figure differs from industry to industry. But having a higher churn figure than that is a definite sign that a business is doing something wrong.”

There are many things brands may do wrong, from complicated onboarding when customers aren’t given easy-to-understand information about product usage and its capabilities to poor communication, e.g. the lack of feedback or delayed answers to queries. Another situation: Longtime clients may feel unappreciated because they don’t get as many bonuses as the new ones.

In general, it’s the overall customer experience that defines brand perception and influences how customers recognize value for money of products or services they use.

#### **Impact of customer churn on businesses**

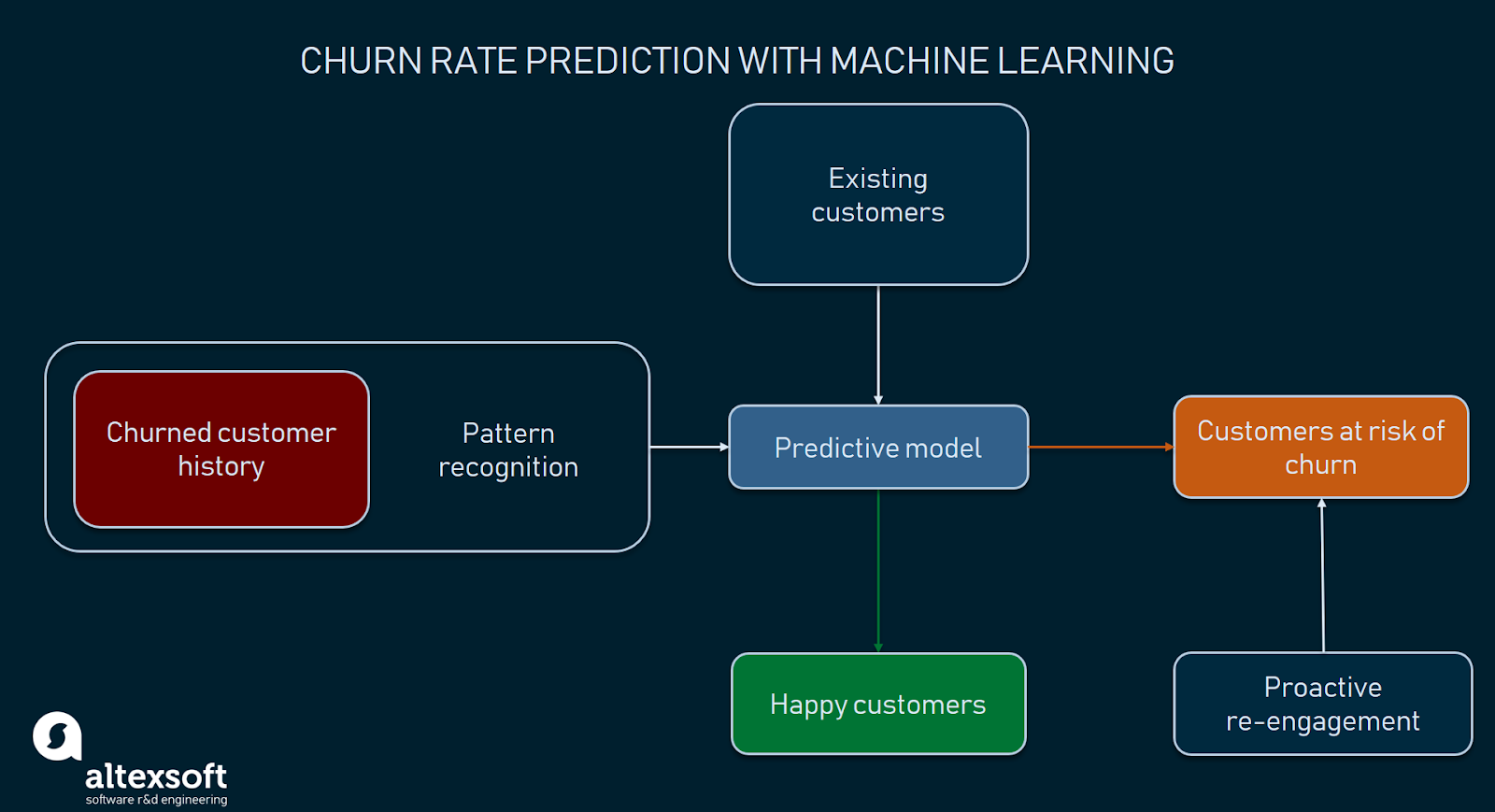
Don’t underestimate the impact of even a tiny percentage of churn, says [Michael Redbord](https://www.linkedin.com/in/mredbord/?lipi=urn%3Ali%3Apage%3Ad_flagship3_search_srp_top%3BL0kyvSq6RtG31Y%2BiJAjWWg%3D%3D&licu=urn%3Ali%3Acontrol%3Ad_flagship3_search_srp_top-search_srp_result&lici=e1pUKpgpST6L9WptRqo6Qg%3D%3D), general manager of Service Hub at HubSpot. “In a subscription-based business, even a small rate of monthly/quarterly churn will compound quickly over time. Just 1 percent monthly churn translates to almost 12 percent yearly churn. Given that it’s far more expensive to acquire a new customer than to retain an existing one, businesses with high churn rates will quickly find themselves in a financial hole as they have to devote more and more resources to new customer acquisition.”

Many surveys focusing on customer acquisition and retention costs are available online. According to this one by [Invesp](https://www.invespcro.com/blog/customer-acquisition-retention/), conversion rate optimization company, getting a new customer may cost up to five times more than retaining an existing customer.

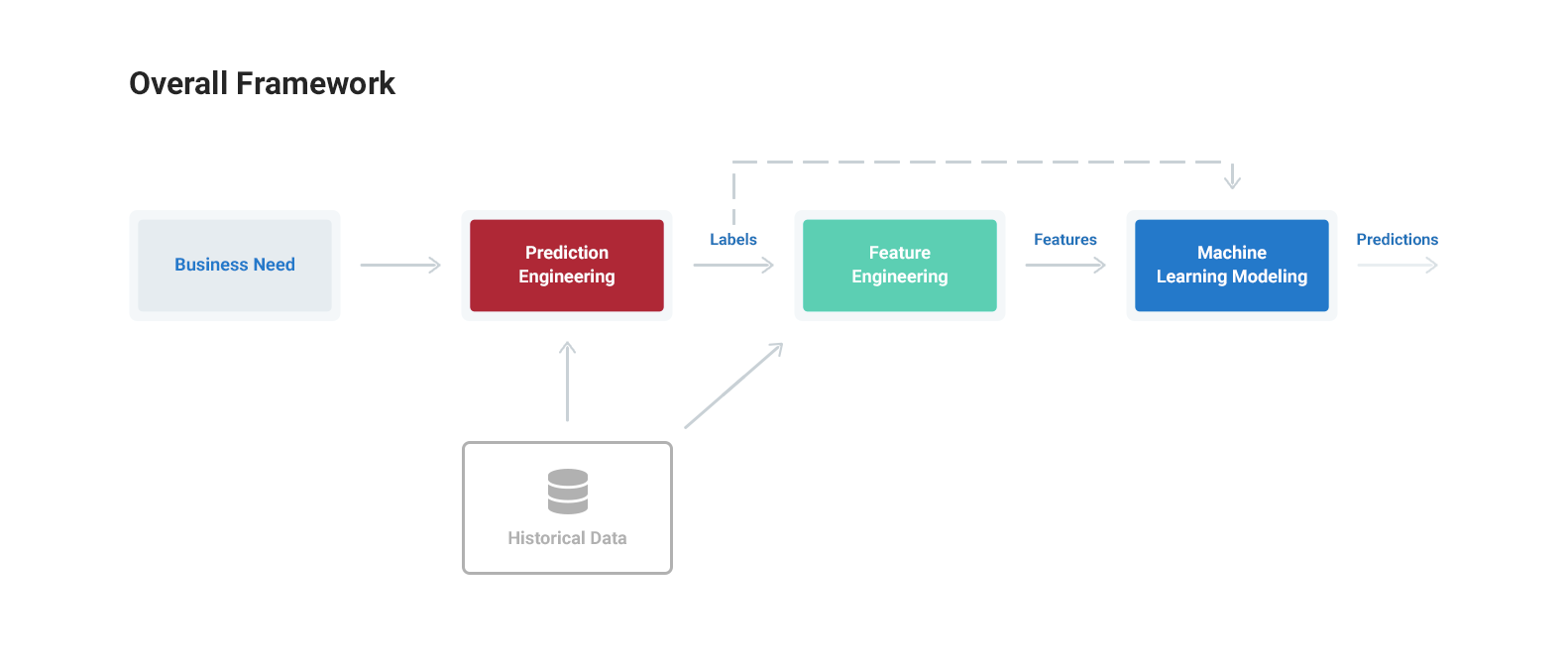
Churn rates do correlate with lost revenue and increased acquisition spend In addition, they play a more nuanced role in a company’s growth potential, continues Michael,“Today’s buyers aren’t shy about sharing their experiences with vendors through channels like review sites and social media, as well as peer-to-peer networks. [HubSpot Research](https://blog.hubspot.com/news-trends/customer-acquisition-study) found that 49 percent of buyers reported sharing an experience they had with a company on social media. In a world of eroding trust in businesses, word of mouth plays a more critical role in the buying process than ever before. From the same HubSpot Research study, 55 percent of buyers no longer trust the companies they buy from as much as they used to, 65 percent don’t trust company press releases, 69 percent don’t trust advertisements, and 71 percent don’t trust sponsored ads on social networks.”

The expert concludes that companies with high churn rates aren’t only failing to deliver in their relationships with ex-customers but also damage their future acquisition efforts by creating negative word-of-mouth around their products.

CallMiner conversational analytics solutions provider interviewed 1000 adults to learn why and how they interact with companies. The [survey](https://learn.callminer.com/whitepapers/callminerindex-us-consumers-switch-by-sector) revealed that US businesses lose about $136 billion a year due to customer attrition. What’s more, the company behaviors that caused customers to cut ties with brands could have been corrected.



# A Machine Learning Framework with an Application to Predicting Customer Churn



The end outcome is a both a specific solution to a customer churn use case, with a reduction in revenue lost to churn of more than 10%, as well as a general approach you can use to solve your own problems with machine learning.

# Framework Steps

1. **Prediction engineering**

* State business need
* Translate business requirement into machine learning task by specifying problem parameters
* Develop set of labels along with cutoff times for supervised machine learning

1. **Feature Engineering**

* Create features - predictor variables - out of raw data
* Use cutoff times to make valid features for each label
* Apply automated feature engineering to automatically make hundreds of relevant, valid features

1. **Modeling**

* Train a machine learning model to predict labels from features
* Use a pre-built solution with common libraries
* Optimize model in line with business objectives

Machine learning currently is an ad-hoc process requiring a custom solution for each problem. Even for the same dataset, a slightly different prediction problem requires an entirely new pipeline built from scratch. This has made it too difficult for many companies to take advantage of the benefits of machine learning. The standardized procedure presented here will make it easier to solve meaningful problems with machine learning, allowing more companies to harness this transformative technology.

**Technique choice**

 Numerous factors can influence the number of required models in production and their type. Although each company’s case is unique, but generally approaches to managing customer data and business needs do have weight. The choice of a prediction technique may depend on:

**Customer lifecycle stage.** HubSpot specialists, for instance, concluded that the model choice may depend on the stage of interaction between a customer and a brand. “Customers in onboarding don’t usually display the same value metrics as those customers who have been using HubSpot for greater than a year. Thus, a model trained on customers older than one year may work really great for those customers, but not be accurate when applied to customers still in onboarding,” explains Michael of HubSpot.

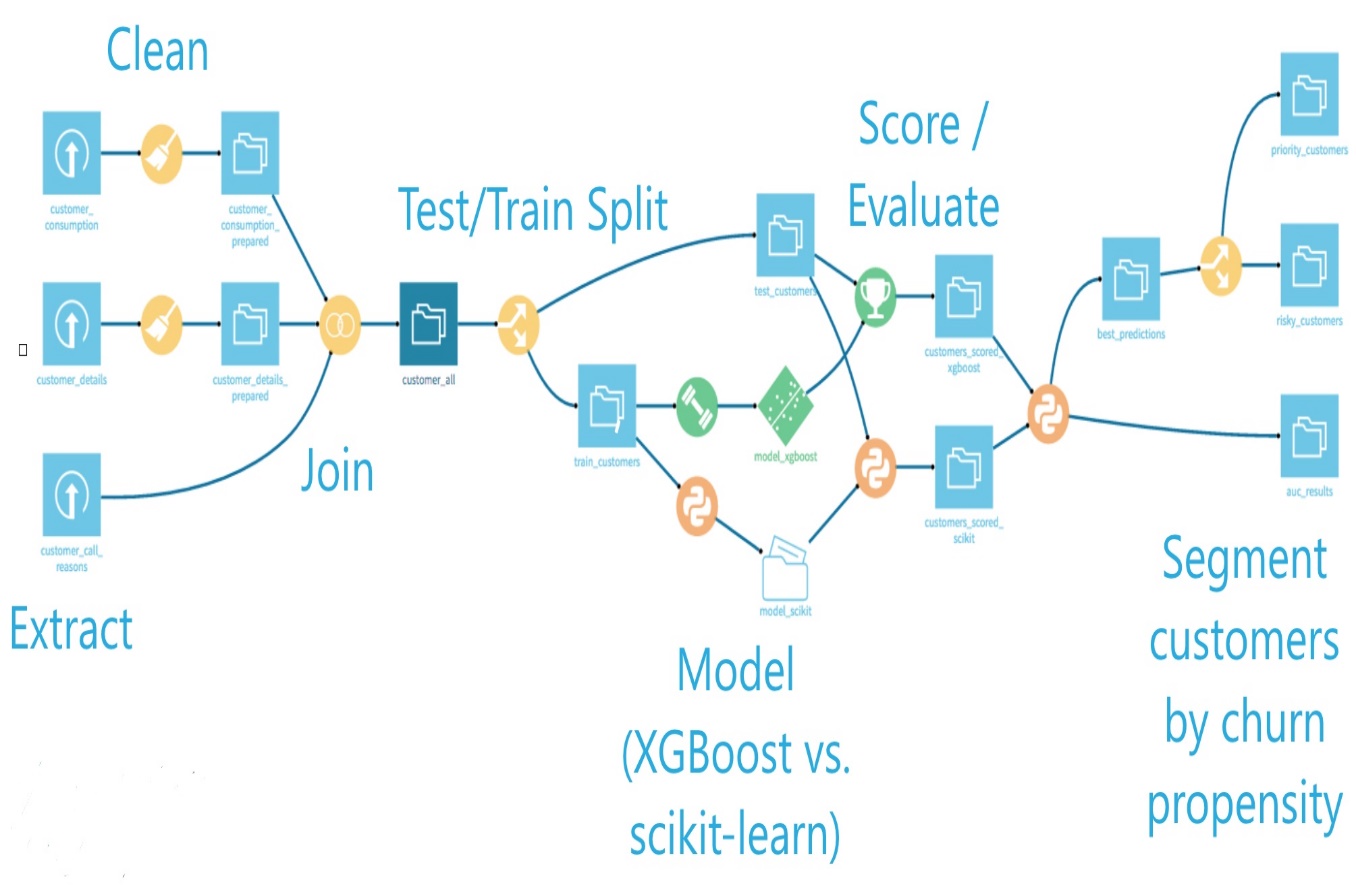
**The need for output explanation.**When company representatives (e.g. customer success managers) must understand the reasons for churn, so-called white box techniques like decision trees, random forest, or logistics regression can be used. Increased interpretability is one of the main reasons HubSpot opts for random forest. Sometimes it’s enough just to detect churn, for instance when company management needs to estimate budgeting for the next year while taking into consideration possible losses due to customer churn. In these cases, less interpretable models would work.

**Customer persona.** Think of a company providing numerous products, each of them designed a specific user type. Since different customer personas may have typical behavior patterns, using dedicated models to predict the likelihood of them churning seems reasonable. Michael Redbord adds: “In a growing business, the nature of the customer base will evolve, especially when new products are introduced. Models built on one set of customers may not work as well when a new customer persona enters the customer base. Thus, when we’ve introduced a new product line we’ve typically built new models to predict churn of those customers.”

#### **Deployment and monitoring**

And now, the final stage of the churn prediction project workflow. The selected model/models need to be put into production. A model may be incorporated into existing software or become a core of a new program. However, the deploy-and-forget scenario won’t work: Data scientists must keep track of a model’s accuracy levels and improve it if needed.

“Predicting customer churn with machine learning and artificial intelligence is an iterative process that never ends. We monitor model performance and adjust features as necessary to improve accuracy when customer-facing teams give us feedback or new data becomes available. At the point of any human interaction – a support call, a CSM QBR [quarterly business review], a Sales discovery call – we monitor and log the human interpretation of customer help, which augments the machine learning models and increases the accuracy of our health prediction for each customer,” summarizes Michael.

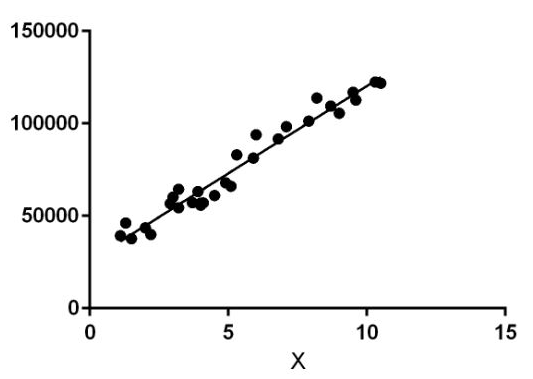
The frequency with which a model performance is tested depends on how fast data becomes outdated in an organization.

# **MODEL BUILDING**

* *SHORT DESCRIPTION OF ESCH MODEL:*

**LINEAR REGRESSION:**

**Linear Regression** is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.  
In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

**Hypothesis function for Linear Regression :**  

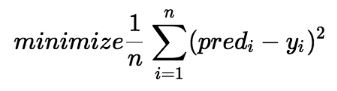

*While training the model we are given :*

**x:** input training data (univariate – one input variable(parameter))  
**y:** labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.  
**θ1:** intercept  
**θ2:** coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

***Cost Function (J):***  
By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ1 and θ2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y).





Cost function(J) of Linear Regression is the **Root Mean Squared Error (RMSE)** between predicted y value (pred) and true y value (y).

[**Gradient Descent**](https://www.geeksforgeeks.org/gradient-descent-in-linear-regression/)**:**  
To update θ1 and θ2 values in order to reduce Cost function (minimizing RMSE value) and achieving the best fit line the model uses Gradient Descent. The idea is to start with random θ1 and θ2 values and then iteratively updating the values, reaching minimum cost.

**LOGISTIC REGRESSION:**

This article discusses the basics of Logistic Regression and its implementation in Python. Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable or output), y, can take only discrete values for given set of features (or inputs), X.

Contrary to popular belief, logistic regression IS a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”. Just like Linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function.



Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. The setting of the threshold value is a very important aspect of Logistic regression and is dependent on the classification problem itself.

The decision for the value of the threshold value is majorly affected by the values of [precision and recall.](https://www.geeksforgeeks.org/confusion-matrix-machine-learning/) Ideally, we want both precision and recall to be 1, but this seldom is the case. In case of a Precision-Recall trade off we use the following arguments to decide upon the threshold :-

1. **Low Precision/High Recall:** In applications where we want to reduce the number of false negatives without necessarily reducing the number false positives, we choose a decision value which has a low value of Precision or high value of Recall. For example, in a cancer diagnosis application, we do not want any affected patient to be classified as not affected without giving much heed to if the patient is being wrongfully diagnosed with cancer. This is because, the absence of cancer can be detected by further medical diseases but the presence of the disease cannot be detected in an already rejected candidate.

2. **High Precision/Low Recall:** In applications where we want to reduce the number of false positives without necessarily reducing the number false negatives, we choose a decision value which has a high value of Precision or low value of Recall. For example, if we are classifying customers whether they will react positively or negatively to a personalised advertisement, we want to be absolutely sure that the customer will react positively to the advertisemnt because otherwise, a negative reaction can cause a loss potential sales from the customer.

Based on the number of categories, Logistic regression can be classified as:

1. **binomial:** target variable can have only 2 possible types: “0” or “1” which may represent “win” vs “loss”, “pass” vs “fail”, “dead” vs “alive”, etc.
2. **multinomial:** target variable can have 3 or more possible types which are not ordered(i.e. types have no quantitative significance) like “disease A” vs “disease B” vs “disease C”.
3. **ordinal:** it deals with target variables with ordered categories. For example, a test score can be categorized as:“very poor”, “poor”, “good”, “very good”. Here, each category can be given a score like 0, 1, 2, 3.

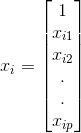
First of all, we explore the simplest form of Logistic Regression, i.e **Binomial Logistic Regression**.

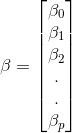
**Binomial Logistic Regression**

Consider an example dataset which maps the number of hours of study with the result of an exam. The result can take only two values, namely passed(1) or failed(0):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **HOURS(X)** | 0.50 | 0.75 | 1.00 | 1.25 | 1.50 | 1.75 | 2.00 | 2.25 | 2.50 | 2.75 | 3.00 | 3.25 | 3.50 | 3.75 | 4.00 | 4.25 | 4.50 | 4.75 | 5.00 | 5.50 |
| **PASS(Y)** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

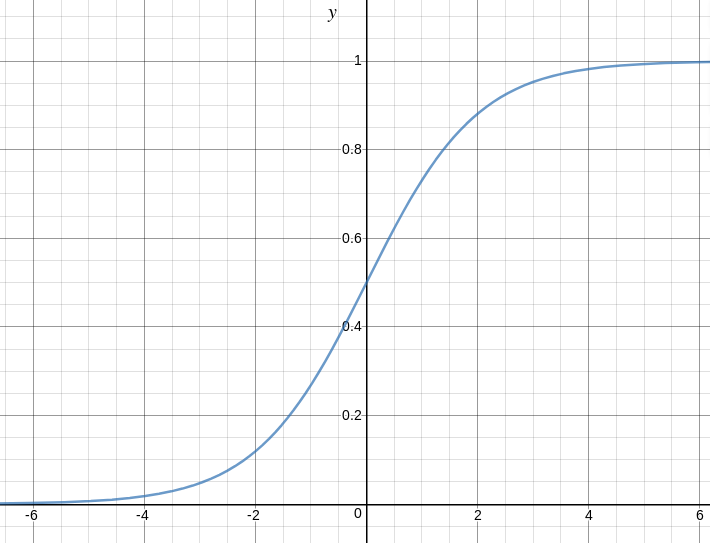
So, we have  
https://latex.codecogs.com/gif.latex?y%20%3D%20%5Cleft%5C%7B%5Cbegin%7Bmatrix%7D%200%2Cif%20fail%5C%5C%201%2Cif%20pass%5C%5C%20%5Cend%7Bmatrix%7D%5Cright.  
i.e. y is a categorical target variable which can take only two possible type:“0” or “1”.  
In order to generalize our model, we assume that:

* The dataset has ‘p’ feature variables and ‘n’ observations.
* The feature matrix is represented as:  
    
  Here,  denotes the values of  feature for  observation.  
  Here, we are keeping the convention of letting  = 1. (Keep reading, you will understand the logic in a few moments).
* The  observation, , can be represented as:  
  
* represents the predicted response for  observation, i.e. . The formula we use for calculating  is called **hypothesis**.

If you have gone though Linear Regression, you should recall that in Linear Regression, the hypothesis we used for prediction was:  
https://latex.codecogs.com/gif.latex?h%28x_i%29%20%3D%20%5Cbeta_0%20+%20%5Cbeta_1x_%7Bi1%7D%20+%20%5Cbeta_2x_%7Bi2%7D%20+%20.....%20+%20%5Cbeta_px_%7Bip%7D  
where,  are the regression coefficients.  
Let regression coefficient matrix/vector,  be:  
  
Then, in a more compact form,  
https://latex.codecogs.com/gif.latex?h%28x_i%29%20%3D%20%5Cbeta%5ETx_i

*The reason for taking  = 1 is pretty clear now.  
We needed to do a matrix product, but there was no  
actual  multiplied to  in original hypothesis formula. So, we defined  = 1.*

Now, if we try to apply Linear Regression on above problem, we are likely to get continuous values using the hypothesis we discussed above. Also, it does not make sense for to take values larger that 1 or smaller than 0.  
So, some modifications are made to the hypothesis for classification:  
https://latex.codecogs.com/gif.latex?h%28x_i%29%20%3D%20g%28%5Cbeta%5ET%20x_i%29%20%3D%20%5Cfrac%7B1%7D%7B1%20+%20e%5E%7B-%5Cbeta%5ET%20x_i%7D%7D  
where,  
https://latex.codecogs.com/gif.latex?g%28z%29%20%3D%20%5Cfrac%7B1%7D%7B1%20+%20e%5E%7B-z%7D%7D  
is called **logistic function** or the **sigmoid function**.

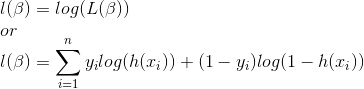
Here is a plot showing g(z):  
[](https://media.geeksforgeeks.org/wp-content/uploads/logistic-function.png)

We can infer from above graph that:

* g(z) tends towards 1 as
* g(z) tends towards 0 as
* g(z) is always bounded between 0 and 1

So, now, we can define conditional probabilities for 2 labels(0 and 1) for  observation as:  
https://latex.codecogs.com/gif.latex?%5Cnewline%20P%28y_i%20%3D%201%7Cx_i%3B%20%5Cbeta%29%20%3D%20h%28x_i%29%20%5Cnewline%20P%28y_i%3D0%7Cx_i%3B%20%5Cbeta%29%20%3D%201%20-%20h%28x_i%29  
We can write it more compactly as:  
https://latex.codecogs.com/gif.latex?P%28y_i%7Cx_i%3B%5Cbeta%29%20%3D%20%28h%28x_i%29%29%5E%7By_i%7D%281-h%28x_i%29%29%5E%7B1-y_i%7D  
Now, we define another term, **likelihood of parameters** as:

*Likelihood is nothing but the probability of data(training examples), given a model and specific parameter values(here, ). It measures the support provided by the data for each possible value of the . We obtain it by multiplying all for given .*

And for easier calculations, we take **log likelihood**:  
  
The **cost function** for logistic regression is proportional to inverse of likelihood of parameters. Hence, we can obtain an expression for cost function, J using log likelihood equation as:  
https://latex.codecogs.com/gif.latex?J%28%5Cbeta%29%20%3D%5Csum_%7Bi%3D1%7D%5E%7Bn%7D%20-%20y_ilog%28h%28x_i%29%29%20-%20%281-y_i%29log%281-h%28x_i%29%29  
and our aim is to estimate  so that cost function is minimized !!

**Using Gradient descent algorithm**

Firstly, we take partial derivatives of  w.r.t each  to derive the stochastic gradient descent rule(we present only the final derived value here):  
https://latex.codecogs.com/gif.latex?%5Cfrac%7B%5Cpartial%20J%28%5Cbeta%29%7D%7B%5Cpartial%20%5Cbeta_j%7D%20%3D%20%28h%28x%29%20-%20y%29x_j  
Here, y and h(x) represent the response vector and predicted response vector(respectively). Also,  is the vector representing the observation values for feature.  
Now, in order to get min ,  
  
where  is called **learning rate** and needs to be set explicitly.

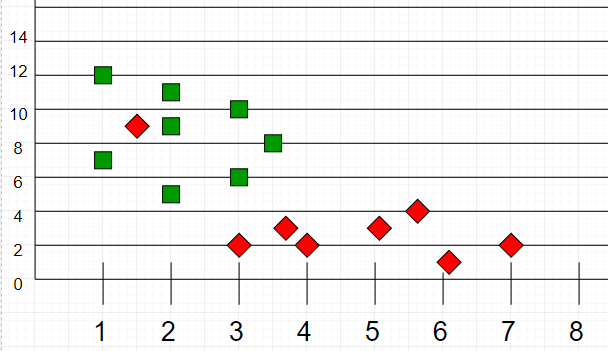
* **K-NEAREST NEIGHBOURS:**

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

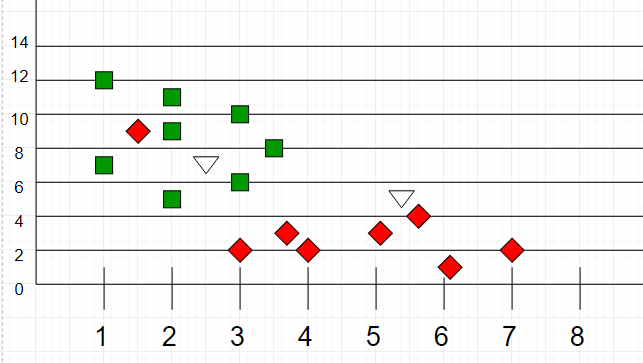
It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as [GMM](https://en.wikipedia.org/wiki/Mixture_model), which assume a Gaussian distribution of the given data).

We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

Now, given another set of data points (also called testing data), allocate these points a group by analyzing the training set. Note that the uncla ssified points are marked as ‘White’.



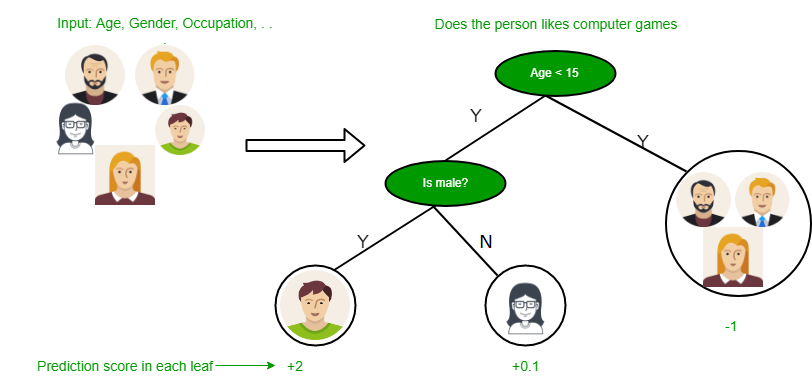
As an example, consider the following table of data points containing two features:



**Intuition**  
If we plot these points on a graph, we may be able to locate some clusters, or groups. Now, given an unclassified point, we can assign it to a group by observing what group its nearest neighbours belong to. This means, a point close to a cluster of points classified as ‘Red’ has a higher probability of getting classified as ‘Red’.

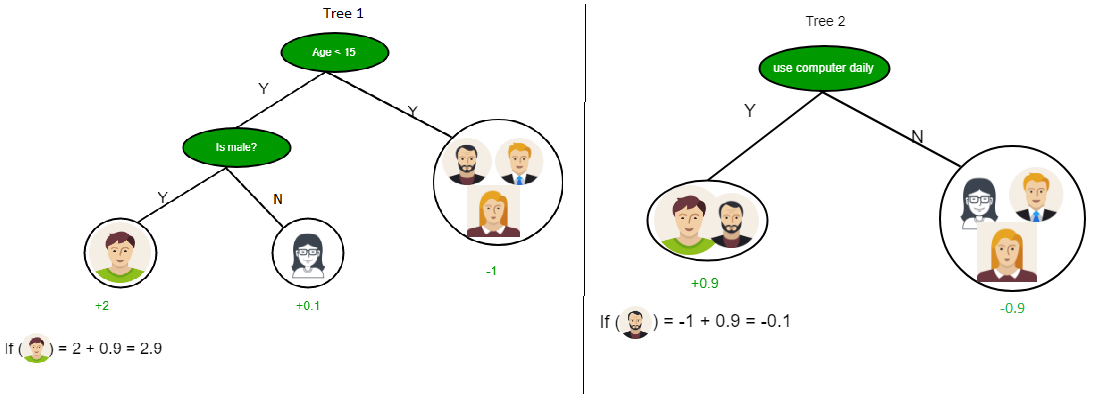
Intuitively, we can see that the first point (2.5, 7) should be classified as ‘Green’ and the second point (5.5, 4.5) should be classified as ‘Red’.

* **DECISION TREE:**
* Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.
* Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.
* We can represent any boolean function on discrete attributes using the decision tree.



**Below are some assumptions that we made while using decision tree:**

* At the beginning, we consider the whole training set as the root.
* Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
* On the basis of attribute values records are distributed recursively.
* We use statistical methods for ordering attributes as root or the internal node.



As you can see from the above image that Decision Tree works on the Sum of Product form which is also known as *Disjunctive Normal Form*. In the above image, we are predicting the use of computer in the daily life of the people.

In Decision Tree the major challenge is to identification of the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:

1. Information Gain
2. Gini Index

**1. Information Gain**  
When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy.  
***Definition***: Suppose S is a set of instances, A is an attribute, Sv is the subset of S with A = v, and Values (A) is the set of all possible values of A, then

**Entropy**  
Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.  
***Definition***: Suppose S is a set of instances, A is an attribute, Sv is the subset of S with A = v, and Values (A) is the set of all possible values of A, then  
  
Example:

For the set X = {a,a,a,b,b,b,b,b}

Total intances: 8

Instances of b: 5

Instances of a: 3

= -[0.375 \* (-1.415) + 0.625 \* (-0.678)]

=-(-0.53-0.424)

= 0.954

**Building Decision Tree using Information Gain**  
**The essentials:**

* Start with all training instances associated with the root node
* Use info gain to choose which attribute to label each node with
* *Note:* No root-to-leaf path should contain the same discrete attribute twice
* Recursively construct each subtree on the subset of training instances that would be classified down that path in the tree.

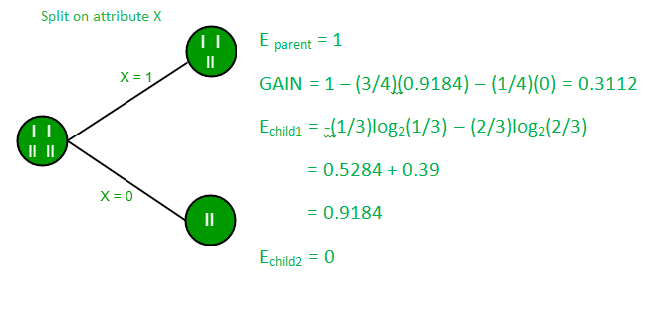
**The border cases:**

* If all positive or all negative training instances remain, label that node “yes” or “no” accordingly
* If no attributes remain, label with a majority vote of training instances left at that node
* If no instances remain, label with a majority vote of the parent’s training instances

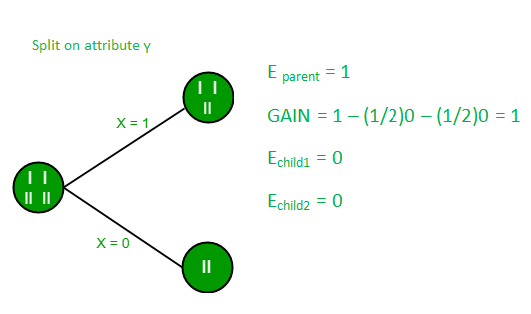
**Example:**  
Now, lets draw a Decision Tree for the following data using Information gain.

**Training set: 3 features and 2 classes**

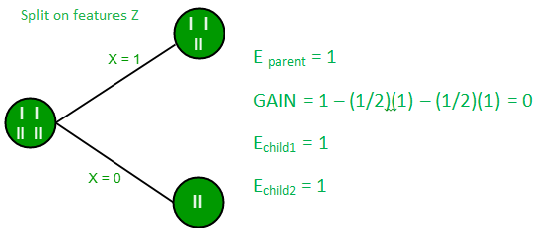
| **X** | **Y** | **Z** | **C** |
| --- | --- | --- | --- |
| 1 | 1 | 1 | I |
| 1 | 1 | 0 | I |
| 0 | 0 | 1 | II |
| 1 | 0 | 0 | II |

Here, we have 3 features and 2 output classes.  
To build a decision tree using Information gain. We will take each of the feature and calculate the information for each feature.  


**Split on feature X**

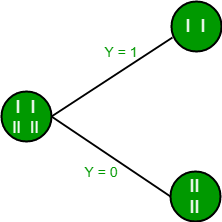


**Split on feature Y**



**Split on feature Z**

From the above images we can see that the information gain is maximum when we make a split on feature Y. So, for the root node best suited feature is feature Y. Now we can see that while spliting the dataset by feature Y, the child contains pure subset of the target variable. So we don’t need to further split the dataset.

The final tree for the above dataset would be look like this:  
  
**2. Gini Index**

* Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
* It means an attribute with lower Gini index should be preferred.
* Sklearn supports “Gini” criteria for Gini Index and by default, it takes “gini” value.
* The Formula for the calculation of the of the Gini Index is given below.

**Example:**  
Lets consider the dataset in the image below and draw a decision tree using gini index.

| **INDEX** | **A** | **B** | **C** | **D** | **E** |
| --- | --- | --- | --- | --- | --- |
| 1 | 4.8 | 3.4 | 1.9 | 0.2 | positive |
| 2 | 5 | 3 | 1.6 | 1.2 | positive |
| 3 | 5 | 3.4 | 1.6 | 0.2 | positive |
| 4 | 5.2 | 3.5 | 1.5 | 0.2 | positive |
| 5 | 5.2 | 3.4 | 1.4 | 0.2 | positive |
| 6 | 4.7 | 3.2 | 1.6 | 0.2 | positive |
| 7 | 4.8 | 3.1 | 1.6 | 0.2 | positive |
| 8 | 5.4 | 3.4 | 1.5 | 0.4 | positive |
| 9 | 7 | 3.2 | 4.7 | 1.4 | negative |
| 10 | 6.4 | 3.2 | 4.7 | 1.5 | negative |
| 11 | 6.9 | 3.1 | 4.9 | 1.5 | negative |
| 12 | 5.5 | 2.3 | 4 | 1.3 | negative |
| 13 | 6.5 | 2.8 | 4.6 | 1.5 | negative |
| 14 | 5.7 | 2.8 | 4.5 | 1.3 | negative |
| 15 | 6.3 | 3.3 | 4.7 | 1.6 | negative |
| 16 | 4.9 | 2.4 | 3.3 | 1 | negative |

In the dataset above there are 5 attributes from which attribute E is the predicting feature which contains 2(Positive & Negative) classes. We have an equal proportion for both the classes.  
In Gini Index, we have to choose some random values to categorize each attribute. These values for this dataset are:

A B C D

>= 5 >= 3.0 >= 4.2 >= 1.4

< 5 < 3.0 < 4.2 < 1.4

**Calculating Gini Index for Var A:**  
**Value >= 5: 12**  
Attribute A >= 5 & class = positive:   
Attribute A >= 5 & class = negative:   
Gini(5, 7) = 1 –   
**Value < 5: 4**  
Attribute A < 5 & class = positive:   
Attribute A < 5 & class = negative:   
Gini(3, 1) = 1 –   
By adding weight and sum each of the gini indices:

**Calculating Gini Index for Var B:**  
**Value >= 3: 12**  
Attribute B >= 3 & class = positive:   
Attribute B >= 5 & class = negative:   
Gini(5, 7) = 1 –   
**Value < 3: 4**  
Attribute A < 3 & class = positive:   
Attribute A < 3 & class = negative:   
Gini(3, 1) = 1 –   
By adding weight and sum each of the gini indices:

Using the same approach we can calculate the Gini index for C and D attributes.

Positive Negative

For A|>= 5.0 5 7

|<5 3 1

Ginin Index of A = 0.45825

Positive Negative

For B|>= 3.0 8 4

|< 3.0 0 4

Gini Index of B= 0.3345

Positive Negative

For C|>= 4.2 0 6

|< 4.2 8 2

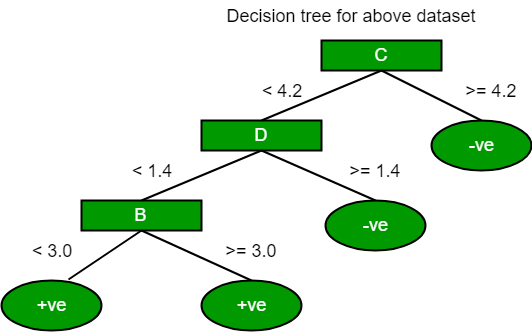
Gini Index of C= 0.2

Positive Negative

For D|>= 1.4 0 5

|< 1.4 8 3

Gini Index of D= 0.273



The most notable types of decision tree algorithms are:-

1. **Iterative Dichotomiser 3 (ID3):** This algorithm uses Information Gain to decide which attribute is to be used classify the current subset of the data. For each level of the tree, information gain is calculated for the remaining data recursively.

2. **C4.5:** This algorithm is the successor of the ID3 algorithm. This algorithm uses either Information gain or Gain ratio to decide upon the classifying attribute. It is a direct improvement from the ID3 algorithm as it can handle both continuous and missing attribute values.

3. **Classification and Regression Tree(CART):** It is a dynamic learning algorithm which can produce a regression tree as well as a classification tree depending upon the dependent vari.

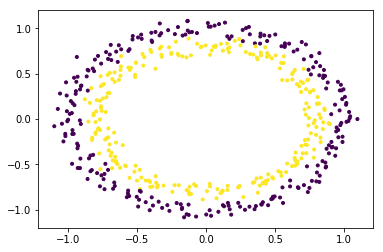
* **SVM:**

**Definition of a hyperplane and SVM classifier:**

For a linearly separable dataset having n features (thereby needing n dimensions for representation), a hyperplane is basically an (n – 1) dimensional subspace used for separating the dataset into two sets, each set containing data points belonging to a different class. For example, for a dataset having two features X and Y (therefore lying in a 2-dimensional space), the separating hyperplane is a line (a 1-dimensional subspace). Similarly, for a dataset having 3-dimensions, we have a 2-dimensional separating hyperplane, and so on.  
In machine learning, Support Vector Machine (SVM) is a non-probabilistic, linear, binary classifier used for classifying data by learning a hyperplane separating the data.

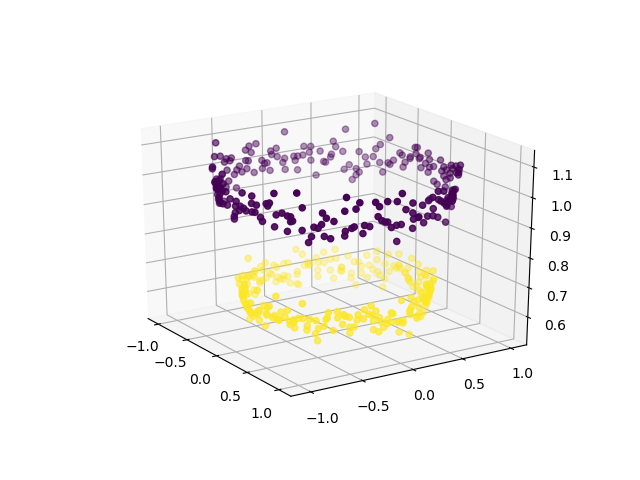
**Classifying a non-linearly separable dataset using a SVM – a linear classifier:**

As mentioned above SVM is a linear classifier which learns an (n – 1)-dimensional classifier for classification of data into two classes. However, it can be used for classifying a non-linear dataset. This can be done by projecting the dataset into a higher dimension in which it is linearly separable!

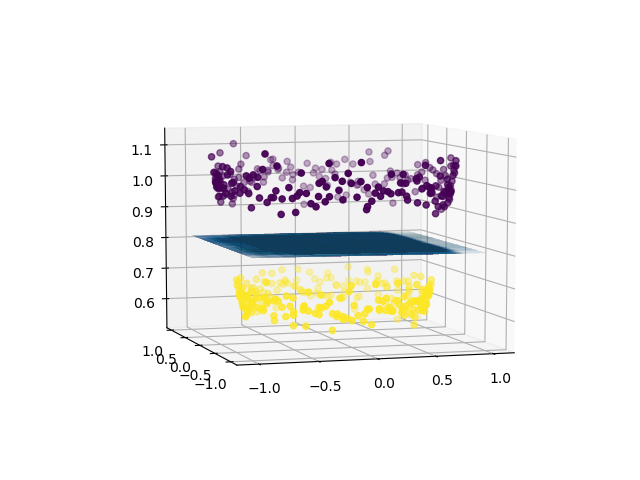


The dataset is clearly a non-linear dataset and consists of two features (say, X and Y).

In order to use SVM for classifying this data, introduce another feature Z = X2 + Y2 into the dataset. Thus, projecting the 2-dimensional data into 3-dimensional space. The first dimension representing the feature X, second representing Y and third representing Z (which, mathematically, is equal to the radius of the circle of which the point (x, y) is a part of). Now, clearly, for the data shown above, the ‘yellow’ data points belong to a circle of smaller radius and the ‘purple’ data points belong to a circle of larger radius. Thus, the data becomes linearly separable along the Z-axis.



Now, we can use SVM (or, for that matter, any other linear classifier) to learn a 2-dimensional separating hyperplane.



Thus, using a linear classifier we can separate a non-linearly separable dataset.

**A brief introduction to kernels in machine learning:**

In machine learning, a trick known as “kernel trick” is used to learn a linear classifier to classify a non-linear dataset. It transforms the linearly inseparable data into a linearly separable one by projecting it into a higher dimension. A kernel function is applied on each data instance to map the original non-linear data points into some higher dimensional space in which they become linearly separable.

* **NAÏVE BAYES:**

This article discusses the theory behind the Naive Bayes classifiers and their implementation.

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes’ Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

To start with, let us consider a dataset.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit(“Yes”) or unfit(“No”) for plaing golf.

Here is a tabular representation of our dataset.

|  | **OUTLOOK** | **TEMPERATURE** | **HUMIDITY** | **WINDY** | **PLAY GOLF** |
| --- | --- | --- | --- | --- | --- |
| 0 | Rainy | Hot | High | False | No |
| 1 | Rainy | Hot | High | True | No |
| 2 | Overcast | Hot | High | False | Yes |
| 3 | Sunny | Mild | High | False | Yes |
| 4 | Sunny | Cool | Normal | False | Yes |
| 5 | Sunny | Cool | Normal | True | No |
| 6 | Overcast | Cool | Normal | True | Yes |
| 7 | Rainy | Mild | High | False | No |
| 8 | Rainy | Cool | Normal | False | Yes |
| 9 | Sunny | Mild | Normal | False | Yes |
| 10 | Rainy | Mild | Normal | True | Yes |
| 11 | Overcast | Mild | High | True | Yes |
| 12 | Overcast | Hot | Normal | False | Yes |
| 13 | Sunny | Mild | High | True | No |

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

* Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are ‘Outlook’, ‘Temperature’, ‘Humidity’ and ‘Windy’.
* Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In above dataset, the class variable name is ‘Play golf’.

**Assumption:**

The fundamental Naive Bayes assumption is that each feature makes an:

* independent
* equal

contribution to the outcome.

With relation to our dataset, this concept can be understood as:

* We assume that no pair of features are dependent. For example, the temperature being ‘Hot’ has nothing to do with the humidity or the outlook being ‘Rainy’ has no effect on the winds. Hence, the features are assumed to be **independent**.
* Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can’t predict the outcome accuratey. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

**Note:** The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Now, before moving to the formula for Naive Bayes, it is important to know about Bayes’ theorem.

**Bayes’ Theorem**

Bayes’ Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes’ theorem is stated mathematically as the following equation:

P(A|B)= P(B|A)P(A)

P(B)

where A and B are events and P(B) ? 0.

* Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
* P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
* P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

Now, with regards to our dataset, we can apply Bayes’ theorem in following way:

P(y|X)= P(X|y)P(y)

P(X)

where, y is class variable and X is a dependent feature vector (of size *n*) where:

Just to clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of dataset)

X = (Rainy, Hot, High, False)

y = No

So basically, P(X|y) here means, the probability of “Not playing golf” given that the weather conditions are “Rainy outlook”, “Temperature is hot”, “high humidity” and “no wind”.

**Naive assumption**

Now, its time to put a naive assumption to the Bayes’ theorem, which is, **independence**among the features. So now, we split **evidence** into the independent parts.

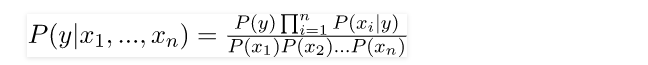
Now, if any two events A and B are independent, then,

P(A,B) = P(A)P(B)

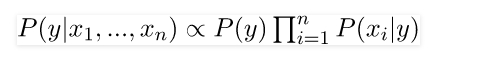
Hence, we reach to the result:



which can be expressed as:



Now, as the denominator remains constant for a given input, we can remove that term:



Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable y and pick up the output with maximum probability. This can be expressed mathematically as:



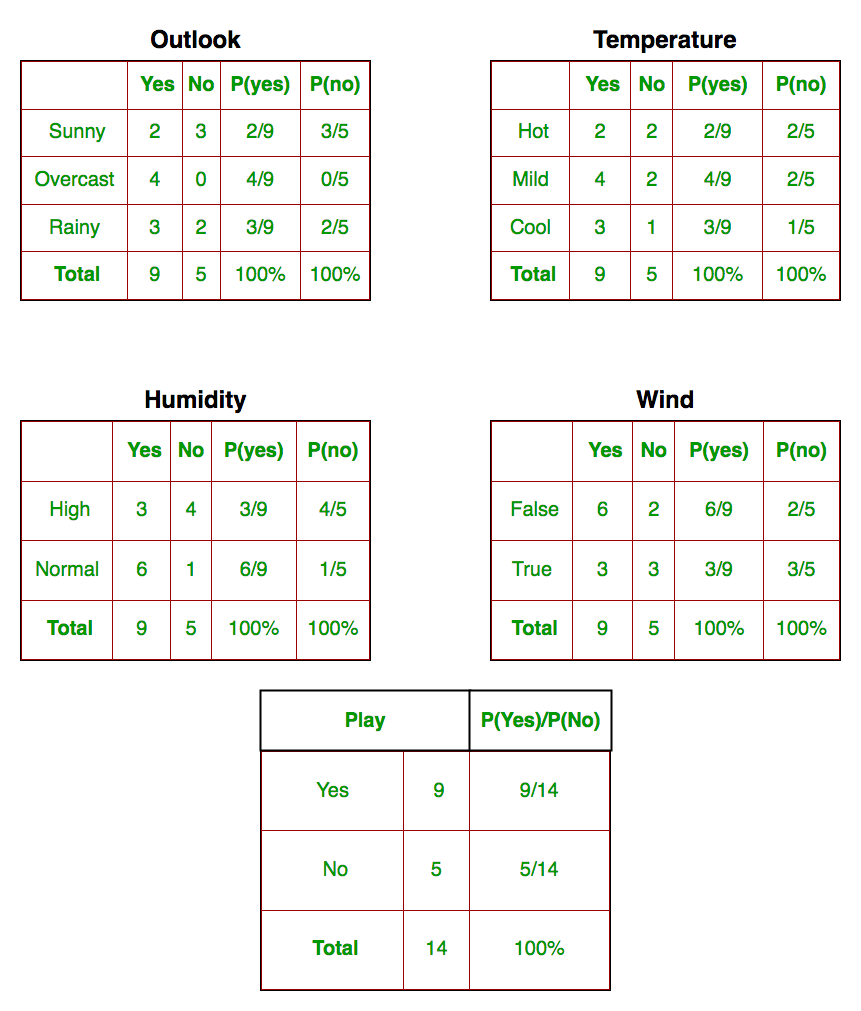
So, finally, we are left with the task of calculating P(y) and P(xi | y).

Please note that P(y) is also called **class probability** and P(xi | y) is called **conditional probability**.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(xi | y).

Let us try to apply the above formula manually on our weather dataset. For this, we need to do some precomputations on our dataset.

We need to find P(xi | yj) for each xi in X and yj in y. All these calculations have been demonstrated in the tables below:



So, in the figure above, we have calculated P(xi | yj) for each xi in X and yj in y manually in the tables 1-4. For example, probability of playing golf given that the temperature is cool, i.e P(temp. = cool | play golf = Yes) = 3/9.

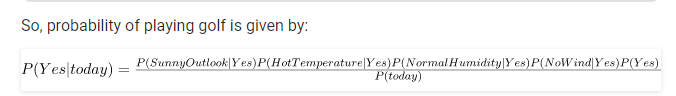
Also, we need to find class probabilities (P(y)) which has been calculated in the table 5. For example, P(play golf = Yes) = 9/14.

So now, we are done with our pre-computations and the classifier is ready!

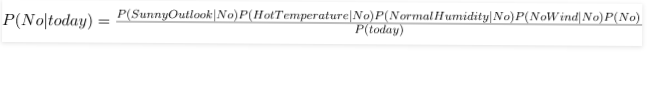
Let us test it on a new set of features (let us call it today):

today = (Sunny, Hot, Normal, False)

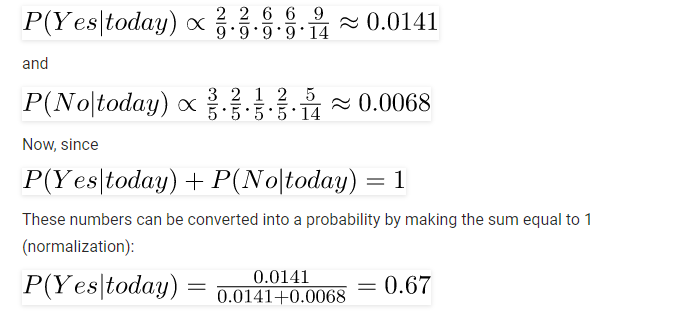
So, probability of playing golf is given by:

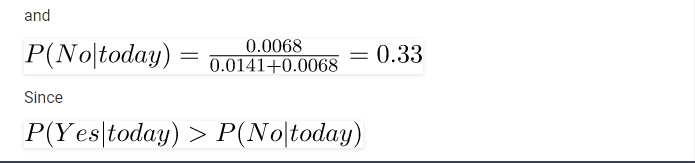


and probability to not play golf is given by:



Since, P(today) is common in both probabilities, we can ignore P(today) and find proportional probabilities as:





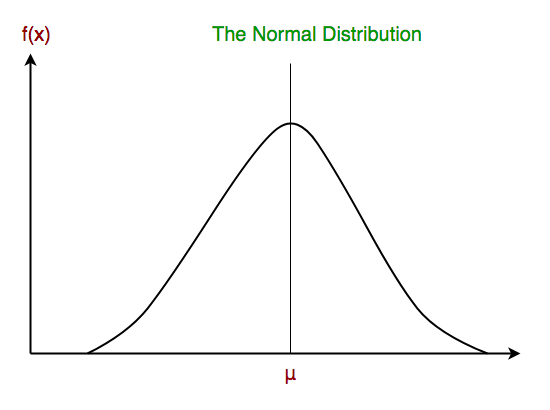
So, prediction that golf would be played is ‘Yes’.

The method that we discussed above is applicable for discrete data. In case of continuous data, we need to make some assumptions regarding the distribution of values of each feature. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(xi | y).

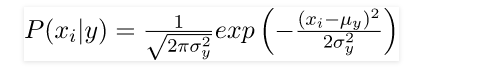
Now, we discuss one of such classifiers here.

**Gaussian Naive Bayes classifier**

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**. A Gaussian distribution is also called [Normal distribution](https://en.wikipedia.org/wiki/Normal_distribution). When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:



The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:



Other popular Naive Bayes classifiers are:

* **Multinomial Naive Bayes**: Feature vectors represent the frequencies with which certain events have been generated by a **multinomial distribution**. This is the event model typically used for document classification.
* **Bernoulli Naive Bayes**: In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence(i.e. a word occurs in a document or not) features are used rather than term frequencies(i.e. frequency of a word in the document).

As we reach to the end of this article, here are some important points to ponder upon:

* In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters.
* Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.
* **MODEL DESCRIPTION WHICH ARE TRIED:**

We have used in our model **LOGISTIC REGRESSION, KNN, NAÏVE-BAYES, DECISION TREE.**

***Logistic regression:***

In [statistics](https://en.wikipedia.org/wiki/Statistics), the **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be combined to model several classes of events such as determining whether an image contains a cat, dog, lion, etc... Each object being detected in the image would be assigned a probability between 0 and 1 and the sum adding to one.

Logistic regression is a [statistical model](https://en.wikipedia.org/wiki/Statistical_model) that in its basic form uses a [logistic function](https://en.wikipedia.org/wiki/Logistic_function) to model a [binary](https://en.wikipedia.org/wiki/Binary_variable) [dependent variable](https://en.wikipedia.org/wiki/Dependent_variable), although many more complex [extensions](https://en.wikipedia.org/wiki/Logistic_regression#Extensions) exist. In [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis), **logistic regression** (or **logit regression**) is [estimating](https://en.wikipedia.org/wiki/Estimation_theory) the parameters of a logistic model (a form of [binary regression](https://en.wikipedia.org/wiki/Binary_regression)). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an [indicator variable](https://en.wikipedia.org/wiki/Indicator_variable), where the two values are labelled "0" and "1". In the logistic model, the [log-odds](https://en.wikipedia.org/wiki/Log-odds) (the [logarithm](https://en.wikipedia.org/wiki/Logarithm) of the [odds](https://en.wikipedia.org/wiki/Odds)) for the value labelled "1" is a [linear combination](https://en.wikipedia.org/wiki/Linear_function_(calculus)) of one or more [independent variables](https://en.wikipedia.org/wiki/Independent_variable) ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a [continuous variable](https://en.wikipedia.org/wiki/Continuous_variable) (any real value). The corresponding [probability](https://en.wikipedia.org/wiki/Probability) of the value labelled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labelling; the function that converts log-odds to probability is the logistic function, hence the name. The [unit of measurement](https://en.wikipedia.org/wiki/Unit_of_measurement) for the log-odds scale is called a [*logit*](https://en.wikipedia.org/wiki/Logit), from ***log****istic un****it***, hence the alternative names. Analogous models with a different [sigmoid function](https://en.wikipedia.org/wiki/Sigmoid_function) instead of the logistic function can also be used, such as the [probit model](https://en.wikipedia.org/wiki/Probit_model" \o "Probit model); the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a *constant* rate, with each dependent variable having its own parameter; for a binary independent variable this generalizes the [odds ratio](https://en.wikipedia.org/wiki/Odds_ratio).

The binary logistic regression model has [extensions](https://en.wikipedia.org/wiki/Logistic_regression#Extensions) to more than two levels of the dependent variable: [categorical](https://en.wikipedia.org/wiki/Categorical_variable) outputs with more than two values are modelled by [multinomial logistic regression](https://en.wikipedia.org/wiki/Multinomial_logistic_regression), and if the multiple categories are [ordered](https://en.wikipedia.org/wiki/Level_of_measurement#Ordinal_type), by [ordinal logistic regression](https://en.wikipedia.org/wiki/Ordinal_logistic_regression), for example the proportional odds ordinal logistic model .The model itself simply models probability of output in terms of input, and does not perform [statistical classification](https://en.wikipedia.org/wiki/Statistical_classification) (it is not a classifier), though it can be used to make a classifier, for instance by choosing a cutoff value and classifying inputs with probability greater than the cut off as one class, below the cutoff as the other; this is a common way to make a [binary classifier](https://en.wikipedia.org/wiki/Binary_classifier).

***KNN:***

In [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition), the ***k*-nearest neighbours algorithm** (***k*-NN**) is a [non-parametric](https://en.wikipedia.org/wiki/Non-parametric_statistics) method used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis).[[1]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-1)In both cases, the input consists of the *k* closest training examples in the [feature space](https://en.wikipedia.org/wiki/Feature_space). The output depends on whether *k*-NN is used for classification or regression:

* In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbours, with the object being assigned to the class most common among its *k* nearest neighbours (*k* is a positive [integer](https://en.wikipedia.org/wiki/Integer), typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbour.
* In *k-NN regression*, the output is the property value for the object. This value is the average of the values of *k* nearest neighbours.

*k*-NN is a type of [instance-based learning](https://en.wikipedia.org/wiki/Instance-based_learning), or [lazy learning](https://en.wikipedia.org/wiki/Lazy_learning), where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms.

Both for classification and regression, a useful technique can be used to assign weight to the contributions of the neighbours, so that the nearer neighbours contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbour a weight of 1/*d*, where *d* is the distance to the neighbour.

The neighbours are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

***NAÏVE-BAYES:***

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), **naive Bayes classifiers** are a family of simple "[probabilistic classifiers](https://en.wikipedia.org/wiki/Probabilistic_classification)" based on applying [Bayes' theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem) with strong (naive) [independence](https://en.wikipedia.org/wiki/Statistical_independence) assumptions between the features.

Naive Bayes has been studied extensively since the 1960s. It was introduced (though not under that name) into the [text retrieval](https://en.wikipedia.org/wiki/Information_retrieval)community in the early 1960s,[[1]](https://en.wikipedia.org/wiki/Naive_Bayes_classifier#cite_note-1) and remains a popular (baseline) method for [text categorization](https://en.wikipedia.org/wiki/Text_categorization), the problem of judging documents as belonging to one category or the other (such as [spam or legitimate](https://en.wikipedia.org/wiki/Spam_filtering), sports or politics, etc.) with [word frequencies](https://en.wikipedia.org/wiki/Bag_of_words) as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine).[[2]](https://en.wikipedia.org/wiki/Naive_Bayes_classifier#cite_note-rennie-2) It also finds application in automatic [medical diagnosis](https://en.wikipedia.org/wiki/Medical_diagnosis).[[3]](https://en.wikipedia.org/wiki/Naive_Bayes_classifier#cite_note-rish-3)

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. [Maximum-likelihood](https://en.wikipedia.org/wiki/Maximum-likelihood_estimation) training can be done by evaluating a [closed-form expression](https://en.wikipedia.org/wiki/Closed-form_expression), which takes [linear time](https://en.wikipedia.org/wiki/Linear_time), rather than by expensive [iterative approximation](https://en.wikipedia.org/wiki/Iterative_method) as used for many other types of classifiers.

In the [statistics](https://en.wikipedia.org/wiki/Statistics) and [computer science](https://en.wikipedia.org/wiki/Computer_science) literature, naive Bayes models are known under a variety of names, including **simple Bayes** and **independence Bayes**.[[5]](https://en.wikipedia.org/wiki/Naive_Bayes_classifier#cite_note-idiots-5) All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a [Bayesian](https://en.wikipedia.org/wiki/Bayesian_probability) method.

***DECISION TREE:***

A **decision tree** is a [decision support](https://en.wikipedia.org/wiki/Decision_support_system) tool that uses a [tree-like](https://en.wikipedia.org/wiki/Tree_(graph_theory)) [model](https://en.wikipedia.org/wiki/Causal_model) of decisions and their possible consequences, including [chance](https://en.wikipedia.org/wiki/Probability) event outcomes, resource costs, and [utility](https://en.wikipedia.org/wiki/Utility). It is one way to display an [algorithm](https://en.wikipedia.org/wiki/Algorithm) that only contains conditional control statements.

Decision trees are commonly used in [operations research](https://en.wikipedia.org/wiki/Operations_research), specifically in [decision analysis](https://en.wikipedia.org/wiki/Decision_analysis), to help identify a strategy most likely to reach a [goal](https://en.wikipedia.org/wiki/Goal), but are also a popular tool in [machine learning](https://en.wikipedia.org/wiki/Decision_tree_learning).

A decision tree is a [flowchart](https://en.wikipedia.org/wiki/Flowchart)-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

In [decision analysis](https://en.wikipedia.org/wiki/Decision_analysis), a decision tree and the closely related [influence diagram](https://en.wikipedia.org/wiki/Influence_diagram) are used as a visual and analytical decision support tool, where the [expected values](https://en.wikipedia.org/wiki/Expected_value) (or [expected utility](https://en.wikipedia.org/wiki/Expected_utility)) of competing alternatives are calculated.

A decision tree consists of three types of nodes:[[1]](https://en.wikipedia.org/wiki/Decision_tree#cite_note-1)

1. Decision nodes – typically represented by squares
2. Chance nodes – typically represented by circles
3. End nodes – typically represented by triangles

Decision trees are commonly used in [operations research](https://en.wikipedia.org/wiki/Operations_research) and [operations management](https://en.wikipedia.org/wiki/Operations_management). If, in practice, decisions have to be taken online with no recall under incomplete knowledge, a decision tree should be paralleled by a [probability](https://en.wikipedia.org/wiki/Probability) model as a best choice model or online selection model [algorithm](https://en.wikipedia.org/wiki/Algorithm). Another use of decision trees is as a descriptive means for calculating [conditional probabilities](https://en.wikipedia.org/wiki/Conditional_probability).

Decision trees, [influence diagrams](https://en.wikipedia.org/wiki/Influence_diagrams), [utility functions](https://en.wikipedia.org/wiki/Utility_function), and other [decision analysis](https://en.wikipedia.org/wiki/Decision_analysis) tools and methods are taught to undergraduate students in schools of business, health economics, and public health, and are examples of operations research or [management science](https://en.wikipedia.org/wiki/Management_science) methods.

* **FOR CLASSIFICATION:**

***ACCURACY:***

Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right. Formally, accuracy has the following definition:

Accuracy=Number of correct predictions/Total number of predictions

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

Accuracy=TP+TN/TP+TN+FP+FN

Where *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives.

Let's try calculating accuracy for the following model that classified 100 tumors as [malignant](https://wikipedia.org/wiki/Malignancy) (the positive class) or [benign](https://wikipedia.org/wiki/Benign_tumor) (the negative class):

|  |  |
| --- | --- |
| True Positive (TP):   * Reality: Malignant * ML model predicted: Malignant * Number of TP results: 1 | False Positive (FP):   * Reality: Benign * ML model predicted: Malignant * Number of FP results: 1 |
| False Negative (FN):   * Reality: Malignant * ML model predicted: Benign * Number of FN results: 8 | True Negative (TN):   * Reality: Benign * ML model predicted: Benign * Number of TN results: 90 |

Accuracy=TP+TN/TP+TN+FP+FN

=1+901+90+1+8=0.91

Accuracy comes out to 0.91, or 91% (91 correct predictions out of 100 total examples). That means our tumor classifier is doing a great job of identifying malignancies, right?

Actually, let's do a closer analysis of positives and negatives to gain more insight into our model's performance.

Of the 100 tumor examples, 91 are benign (90 TNs and 1 FP) and 9 are malignant (1 TP and 8 FNs).

Of the 91 benign tumors, the model correctly identifies 90 as benign. That's good. However, of the 9 malignant tumors, the model only correctly identifies 1 as malignant—a terrible outcome, as 8 out of 9 malignancies go undiagnosed!

While 91% accuracy may seem good at first glance, another tumor-classifier model that always predicts benign would achieve the exact same accuracy (91/100 correct predictions) on our examples. In other words, our model is no better than one that has zero predictive ability to distinguish malignant tumors from benign tumors.

Accuracy alone doesn't tell the full story when you're working with a **class-imbalanced data set**, like this one, where there is a significant disparity between the number of positive and negative labels.

In the next section, we'll look at two better metrics for evaluating class-imbalanced problems: precision and recall.

***PRECISION:***

In the field of [information retrieval](https://en.wikipedia.org/wiki/Information_retrieval), precision is the fraction of retrieved documents that are [relevant](https://en.wikipedia.org/wiki/Relevance_(information_retrieval)) to the query:

**PRECISION= TP/(TP+FP)**

{\displaystyle {\text{precision}}={\frac {|\{{\text{relevant documents}}\}\cap \{{\text{retrieved documents}}\}|}{|\{{\text{retrieved documents}}\}|}}}

For example, for a text search on a set of documents, precision is the number of correct results divided by the number of all returned results.

Precision takes all retrieved documents into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. This measure is called *precision at n* or *P@n*.

Precision is used with [recall](https://en.wikipedia.org/wiki/Recall_(information_retrieval)), the percent of *all* relevant documents that is returned by the search. The two measures are sometimes used together in the [F1 Score](https://en.wikipedia.org/wiki/F1_Score) (or f-measure) to provide a single measurement for a system.

Note that the meaning and usage of "precision" in the field of information retrieval differs from the definition of [accuracy and precision](https://en.wikipedia.org/wiki/Accuracy_and_precision) within other branches of science and technology.

***RECALL:***

In information retrieval, recall is the fraction of the relevant documents that are successfully retrieved.

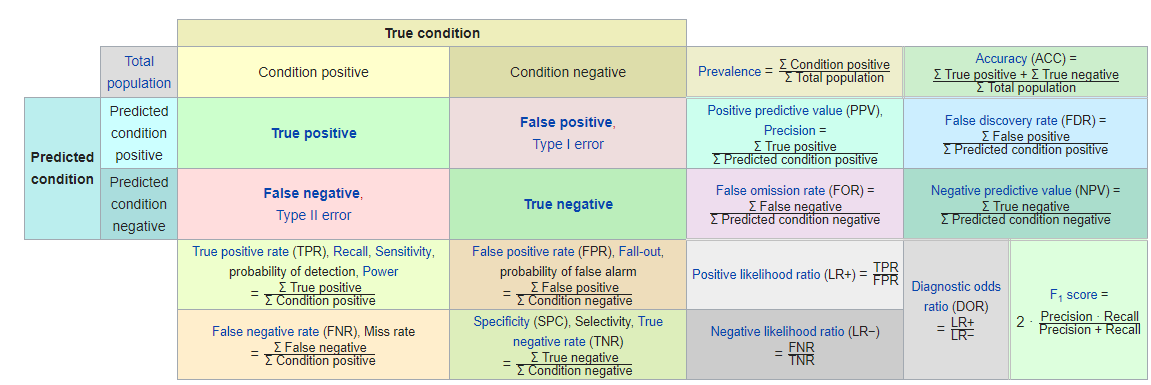
**RECALL=TP/(TP+FN)**

{\displaystyle {\text{recall}}={\frac {|\{{\text{relevant documents}}\}\cap \{{\text{retrieved documents}}\}|}{|\{{\text{relevant documents}}\}|}}}

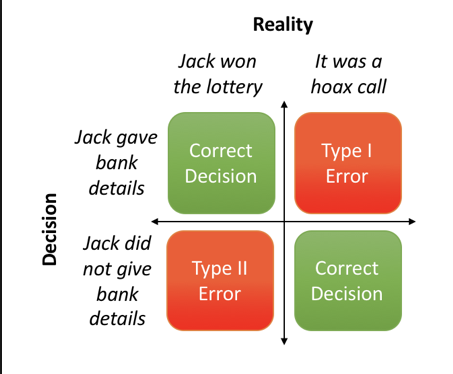
For example, for a text search on a set of documents, recall is the number of correct results divided by the number of results that should have been returned.

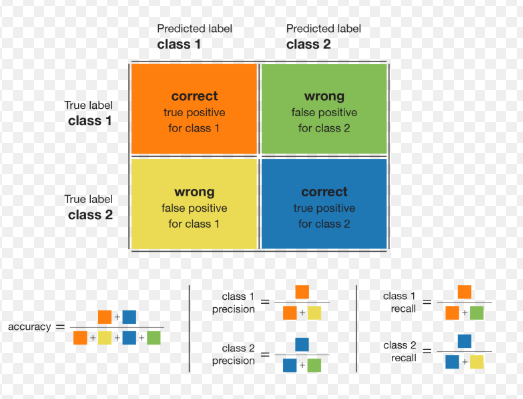
In binary classification, recall is called [sensitivity](https://en.wikipedia.org/wiki/Sensitivity_and_specificity#Sensitivity). It can be viewed as the probability that a relevant document is retrieved by the query.

It is trivial to achieve recall of 100% by returning all documents in response to any query. Therefore, recall alone is not enough but one needs to measure the number of non-relevant documents also, for example by also computing the precision.

******

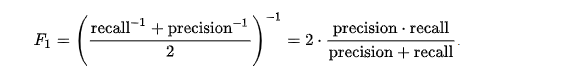
***PRECISION VS RECALL:***

******



***F1-SCORE:***

In [statistical](https://en.wikipedia.org/wiki/Statistics) analysis of [binary classification](https://en.wikipedia.org/wiki/Binary_classification), the **F1 score** (also **F-score** or **F-measure**) is a measure of a test's accuracy. It considers both the [precision](https://en.wikipedia.org/wiki/Precision_(information_retrieval)) *p* and the [recall](https://en.wikipedia.org/wiki/Recall_(information_retrieval)) *r* of the test to compute the score: *p* is the number of correct positive results divided by the number of all positive results returned by the classifier, and *r* is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F1 score is the [harmonic average](https://en.wikipedia.org/wiki/Harmonic_mean) of the [precision and recall](https://en.wikipedia.org/wiki/Precision_and_recall), where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

******

**F1-SCORE APPLICATION:**

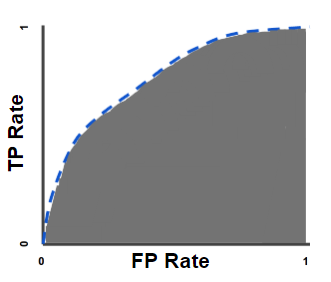
The F-score is often used in the field of [information retrieval](https://en.wikipedia.org/wiki/Information_retrieval) for measuring [search](https://en.wikipedia.org/wiki/Web_search), [document classification](https://en.wikipedia.org/wiki/Document_classification), and [query classification](https://en.wikipedia.org/wiki/Query_classification) performance.[[4]](https://en.wikipedia.org/wiki/F1_score#cite_note-4) Earlier works focused primarily on the F1 score, but with the proliferation of large scale search engines, performance goals changed to place more emphasis on either precision or recall[[5]](https://en.wikipedia.org/wiki/F1_score#cite_note-5) and so {\displaystyle F\_{\beta }} is seen in wide application.

The F-score is also used in [machine learning](https://en.wikipedia.org/wiki/Machine_learning). Note, however, that the F-measures do not take the true negatives into account, and that measures such as the [Matthews correlation coefficient](https://en.wikipedia.org/wiki/Matthews_correlation_coefficient), [Informedness](https://en.wikipedia.org/wiki/Informedness" \o "Informedness) or [Cohen's kappa](https://en.wikipedia.org/wiki/Cohen%27s_kappa) may be preferable to assess the performance of a binary classifier.

The F-score has been widely used in the natural language processing literature, such as the evaluation of [named entity recognition](https://en.wikipedia.org/wiki/Named_entity_recognition) and [word segmentation](https://en.wikipedia.org/wiki/Word_segmentation).

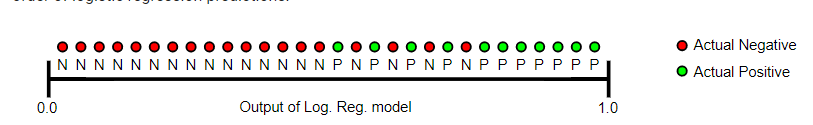
## **AUC: Area Under the ROC Curve**

**AUC** stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).



**AUC (Area under the ROC Curve).**

AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example. For example, given the following examples, which are arranged from left to right in ascending order of logistic regression predictions:



**Predictions ranked in ascending order of logistic regression score.**

AUC represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example.

AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

AUC is desirable for the following two reasons:

* AUC is **scale-invariant**. It measures how well predictions are ranked, rather than their absolute values.
* AUC is **classification-threshold-invariant**. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

However, both these reasons come with caveats, which may limit the usefulness of AUC in certain use cases:

* **Scale invariance is not always desirable.** For example, sometimes we really do need well calibrated probability outputs, and AUC won’t tell us about that.
* **Classification-threshold invariance is not always desirable.** In cases where there are wide disparities in the cost of false negatives vs. false positives, it may be critical to minimize one type of classification error. For example, when doing email spam detection, you likely want to prioritize minimizing false positives (even if that results in a significant increase of false negatives). AUC isn't a useful metric for this type of optimization.

# **Classification: ROC Curve and AUC**

## **ROC curve**

An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

* True Positive Rate
* False Positive Rate

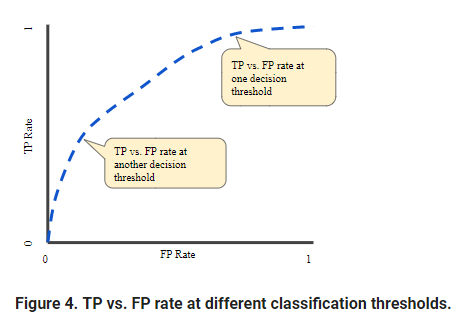
**True Positive Rate** (**TPR**) is a synonym for recall and is therefore defined as follows:

TPR=TP/TP+FN

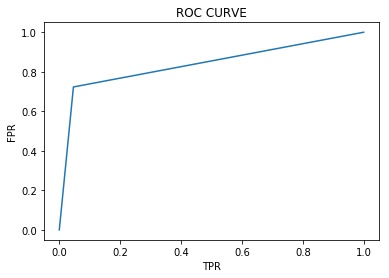
**False Positive Rate** (**FPR**) is defined as follows:

FPR=FP/FP+TN

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.



***FOR OUR MODEL(CUSTOMER CHURN)ROC CURVE:***

**

**Characteristics:**

In a Receiver Operating Characteristic (ROC) curve the true positive rate (Sensitivity) is plotted in function of the false positive rate for different cut-off points. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold.

***FINALLY ACCEPTED MODEL:***

***For our project we have to minimise the value of False Negative . Decrease in FN results in increase in Recall. So we need high Recall value for our project.***

***Decision Tree gives the best Recall value for this project. So we consider Decision Tree model to predict the value for the label(churn/nonchurn).***

# 

# 

A comparison of machine learning technique

for customer churn prediction

**Abstract:**

We present a comparative study on the most popular machine learning methods applied to the challenging problem of customer churning prediction in the telecommunications industry. In the first phase of our experiments, all models were applied and evaluated using cross-validation on a popular, public domain dataset. In the second phase, the performance improvement offered by boosting was studied. In order to determine the most efficient parameter combinations we performed a series of Monte Carlo simulations for each method and for a wide range of parameters. Our results demonstrate clear superiority of the boosted versions of the models against the plain (non-boosted) versions. The best overall classifier was the SVM-POLY using AdaBoost with accuracy of almost 97% and F-measure over 84%. Keywords: Churn prediction, machine learning techniques, boosting algorithm.

* 1. **Introduction:**

Customer Relationship Management (CRM) is a comprehensive strategy for building, managing and strengthening loyal and long-lasting customer relationships. It is broadly acknowledged and extensively applied to different.

fields, e.g. telecommunications, banking and insurance, retail market, etc. One of its main objectives is customer retention. The importance of this objective is obvious, given the fact that the cost for customer acquisition is much greater than the cost of customer retention (in some cases it is 20 times more expensive [1]). Thus, tools to develop and apply customer retention models (churn models) are required and are essential Business Intelligence (BI) applications. In the dynamic market environment, churning could be the result of low-level customer satisfaction, aggressive competitive strategies, new products, regulations, etc. Churn models aim to identify early churn signals and recognize customers with an increased likelihood to leave voluntarily. Over the last decade there has been increasing interest for relevant studies in areas including telecommunication industry [2, 3, 4, 5, 6, 7, 8, 9, 10], banking [1, 11, 12, 13], insurance companies [14], and gaming [15], among others. Several, very popular in research community machine learning algorithms have been proposed in order to tackle the churning prediction problem. Such methods include Artificial Neural Networks [4, 5, 16, 17, 18], Decision Trees learning [4, 5, 7, 9, 16, 17], Regression Analysis [16], Logistic Regression [5, 9], Support Vector Machines [17], Na¨ıve Bayes [4, 19], Sequential Pattern Mining and Market Basket Analysis [20], Linear Discriminant Analysis [13], and Rough Set Approach [21]. This work constitutes a comparison of five of the most widely used classification methods on the problem of customers’ churning in the telecommunication sector. In particular, we compare the performance of multi-layer Artificial Neural Networks, Decision Trees, Support Vector Machines, Na¨ıve Bayes classifiers, and Logistic Regression classifiers, compared to their boosting versions in an attempt to further improve their performance. The motivation behind our study is to evaluate the suitability of the state of the art machine learning methods on the problem of churning. This investigation is performed using Monte Carlo simulation at different settings of each classification method. We use the churn dataset originally from the UCI Machine Learning Repository (converted to MLC++ format1 ), which is now included in the package C50 of the R language2 , in order to test the performance of classification methods and their boosting versions. Data are artificial based on claims similar to the real world. The dataset has been used in numerous publications [22, 23, 24, 25, 26, 27, 28]. The remainder of the paper is organized as follows. In Sec. 2, we give a brief presentation of the machine learning techniques that were evaluated. Evaluation criteria and the proposed boosting algorithm are presented in Sec. 3 and 4. The simulation setup and results are given in Sec. 5, and in Sec. 6 we draw our

conclusions.

* 1. **Machine Learning Techniques-Classification Methods:**

In the following, we briefly present five well established and popular techniques used for churn prediction, taking into consideration reliability, efficiency and popularity in the research community [4, 7, 9, 16, 17, 19, 29, 30].

**2.1. *Artificial Neural Network:***

Artificial Neural Networks (ANN) is a popular approach to address complex problems, such as the churn prediction problem. Neural networks can be hardware-based (neurons are represented by physical components) or software-based (computer models), and can use a variety of topologies and learning algorithms. One popular supervised model is the Multi-Layer Perceptron trained with variations of the Back-Propagation algorithm (BPN). BPN is a feed-forward model with supervised learning. In the case of the customer churn problem, Au e.a. [31] have shown that neural networks achieve better performance compared to Decision Trees. Also, experimental results showed that ANN outperformed Logistic Regression and C5.0 for churn prediction [30].

**2.2. *Support Vector Machines:***

Support Vector Machines (SVM), also known as Support Vector Networks, introduced by Boser, Guyon, and Vapnik [32], are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. SVM is a machine learning technique based on structural risk minimization. Kernel functions have been employed for improving performance [4]. Research on selecting the best kernels or combinations of kernels is still under way. In the churn prediction problem, SVM outperform DT and sometimes ANN, depending mainly on the type of data and data transformation that takes place among them [29, 17].

**2.3. Decision Trees Learning:**

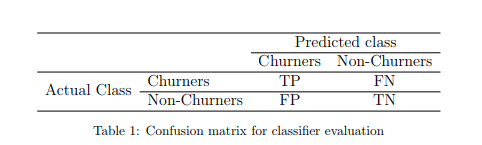
Decision Trees (DT) are tree-shaped structures representing sets of decisions capable to generate classification rules for a specific data set [33], or as Berry and Linoff noted “a structure that can be used to divide up a large collection of records into successively smaller sets of records by applying a sequence of simple decision rules” [34]. More descriptive names for such tree models are Classification Trees or Regression Trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. DT have no great performance on capturing complex and non-linear relationships between the attributes. Yet, in the customers churn problem, the accuracy of a DT can be high, depending on the form of the data [16].

**2.4. Naïve- Bayes:**

A Bayes classifier is a simple probabilistic classifier based on applying Bayes’ theorem with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be independent feature model. In simple terms, a Naive-Bayes (NB) classifier assumes that the presence (or absence) of a particular feature of a class (i.e., customer churn) is unrelated to the presence (or absence) of any other feature. The NB classifier achieved good results on the churn prediction problem for the wireless telecommunications industry [19] and it can also achieve improved prediction rates compared to other widely used algorithms, such as DT-C4.5 [4].

**2.5. Regression Analysis-Logistic :**

Regression Analysis Regression analysis is a statistical process for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables. In terms of customer churning, regression analysis is not widely used, and that is because linear regression models are useful for the prediction of continuous values. On the other hand, Logistic Regression or Logit Regression analysis (LR) is a type of probabilistic statistical classification model. It is also used to produce a binary prediction of a categorical variable (e.g. customer churn) which depends on one or more predictor variables (e.g. customers’ features). In the churn prediction problem, LR is usually used after proper data transformation is applied on initial data, with quite good performance [9] and sometimes performs as well as DT [7].



**3.Evaluation Measures:**

In order to evaluate classifiers performance in churn prediction for different schemes with their appropriate parameters, we use the measures of precision, recall, accuracy and F-measure, calculated from the contents of the confusion matrix, shown in Tab. 1. True positive and false positive cases are denoted as TP and FP, respectively, while true negative and false negative cases are denoted as TN and FN.

Precision is the proportion of the predicted positive cases that were correct and is calculated from the equation P

recision = T P/ T P + F P . (1)

Recall is the proportion of positive cases that were correctly identified and is calculated from the equation

Recall = T P/ T P + F N . (2)

Accuracy is the proportion of the total number of predictions that were correct and is calculated from the equation

Accuracy = T P + T N/ T P + F P + T N + F N . (3)

Precision or recall alone cannot describe the efficiency of a classifier since good performance in one of those indices does not necessarily imply good performance on the other. For this reason, F-measure, a popular combination is commonly used as a single metric for evaluating classifier performance. F measure is defined as the harmonic mean of precision and recall

F − measure = 2 × Precision × Recall /Precision + Recall . (4)

A value closer to one implies that a better combined precision and recall is achieved by the classifier [35].

**4.Boosting Algorithm:**

The main goal of boosting is to improve classification performance through the combination of decisions from many classification models, which are called weak classifiers (or weak learning algorithms). Boosting has been successfully applied to the prediction of customer churn in retail [36] and telecommunications companies [37]. In this paper, the effectiveness of boosting on a classifier will be measured by its ability to improve the respective F-measure. The weak classifiers are used as subroutines combined together in order to build an extremely accurate classifier in the train set. For each weak learner r, the boosting algorithm maintains a distribution of weights wr(i) on the training patterns i, so that each pattern can potentially have different contribution to the final training error of the learner. Initially all weights are set equally, but on each iteration, the weights of incorrectly classified elements are increased in order to force the weak classifier to focus on these hard cases. These continuously changing weights are called adaptive weights. Once the process has finished, the single classifiers obtained are combined into a final, theoretically highly accurate classifier in the train set. The final classifier therefore usually achieves a high degree of accuracy in the test set [11, 38, 39, 40].

There are a lot of forms of the boosting algorithms [19, 41], but the most popular is AdaBoost, where the weak classifiers are decision trees [42]. In this work, we use the AdaBoost.M1 with DT and BPN as weak classifiers. In more detail the AdaBoost.M1 is working as follows: For a training set

T Sn = [(x1, y1), . . . ,(xn, yn)] (5)

with labels yi ∈ Y = [1, . . . , l], a weight

wr(i) = 1 /n (6)

is initially assigned to every observation. These weights are recomputed, afterwards, according to weak classifier achievements. Iteratively, for r = 1, . . . , k, a weak classifier Cr(x) is trained on T Sn in order to minimize the following error

where I is the indicator function, equal to one when its argument is true, zero otherwise. After r iterations the weights are initially updated as follows

(8)

where . (9)

After the initial update the weights are re-normalized. The final boosted classifier is

Cfinal(x) = argmax j∈Y X k r=1 ar I(Cr(xi) = j). (10)

***5. Cross Validation:***

**5.1. Simulation Setup:**

Our main objective is to compare the most prominent classification methods on churn prediction. To that end, we implemented our simulation in two steps using the R language. In the first step, all tested classifiers are initially applied on the churning data. Their performance is evaluated using the F-measure criterion, discussed in Sec. 3. In the second step, a boosting algorithm is applied on the classifiers under inspection and their performance is measured again. For cross validation of our results, we generate 100 Monte Carlo realizations for different parameter scenarios in each classifier. The idea of Monte Carlo is the generation of a large number of synthetic data sets that are similar to experimental data. It is suggested to perform a test Monte Carlo with a small (e.g. 10-100) number of iterations in order to check the performance of the procedure. The tested classifiers were BPN, SVM, DT, NB and LR models. We employed boosting techniques in order to improve classification performance in three cases of ensemble models of BPN, SVM and DT. Ensembles of NB or LR models models were not formed sincethese models have no free parameters to tune. Therefore, no boosting has been applied in these cases. Below, we give a brief presentation of the parameters of each classifier used in the simulation. The combination of all the values of parameters for each classifier and a size of 100 Monte Carlo realizations for each case, gives overall 5000 cases.

**5.1.1. Artificial Neural Networks:**

The classic Back-Propagation algorithm is chosen to train a Multilayer Perceptron model with a single hidden layer (BPN). The number of hidden neurons, n, varies as n = 5 × i , for i = 1, 2, . . . , 9.

**5.1.2. Support Vector Machines:**

The definition of simulation parameters for SVM varies depending on the kernel function that is chosen. In our simulations we chose the Gaussian Radial Basis kernel function (SMV-RBF) and the Polynomial kernel (SMVPOLY). For the RBF kernel,

K(x, y) = exp {||x − y||2 2 σ 2 } , (11)

σ varies as σ = 0.001, 0.01, 0.1, and the constant C, as C = 10, 20, 30, 40, 50, 100. For the Polynomial kernel, K(x, y) = x T y + θ p , (12)

• θ takes the values θ = −2, −1, 0, 1, 2, although θ = 1 is preferable as it avoids problems with the Hessian becoming zero,

• the polynomial degree takes the values p = 2, 3, 4, 5,

• constant C = 100.

5.1.3. **Decision Trees:**

We choose the default C5.0 algorithm for the DT classification (DT-C5.0) the follow up version of the well-known C4.5 [43]. C5.0 algorithm supports boosting, a method that improves the overall construction of the trees and gives more accuracy.

5.1.4. **Naïve- Bayes:**

In the NB model the conditional (posterior) probabilities of a categorical class variable given independent predictor variables are computed, using the Bayes rule.

5.1.5**. Logistic Regression:**

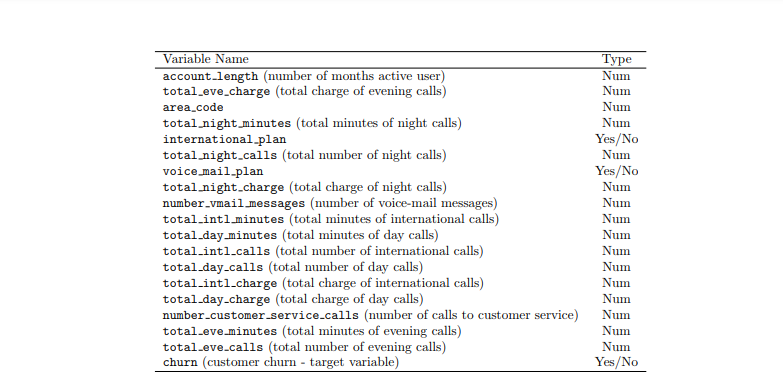
In the LR model case the conditional (posterior) probabilities of a discrete value variable are computed, given discrete or continuous variables.

5.2. **Simulation Results:**

To evaluate the performance of tested classifiers, we use the churn dataset from the UCI Machine Learning Repository, which is now included in the package C50 of the R language for statistical computing. In the corresponding version of the dataset there are 19 predictors, mostly numerical (num), plus the binary (yes/no) churn variable. In our study we do not consider the categorical state variable, since it’s quite specific and we are interested in a more general approach. Thus, we consider 18 predictors, plus the churn variable, presented in Tab. 2. The dataset contains a total of 5000 samples. In order to eliminate the bias, a 100-fold cross validation method is used. The training and testing data sets are randomly chosen with cross validation in a ratio of 2/3 and 1/3, respectively, for every Monte Carlo realization.

5.2.1. **Classifiers performance without boosting:**

For the BPN case of the ANN classifier, our simulation results showed that the use of 20 neurons (or less) in the hidden layer, achieves better precision and quite good recall compared to other cases (Tab. 3). Especially, in the case of 15 hidden neurons, the average F-measure on 100 Monte Carlo realizations is 77.48% and it has a downward trend as the size of hidden layer increases. A similar behavior is also observed for the recall and accuracy metrics where the largest values, 71.68% and 94.06%, respectively, are achieved for the same number of hidden neurons. These results are considered satisfactory, because the use of such size of hidden neurons minimizes the risk of overfitting. For the RBF case of the SVM classifier, the highest values for all evaluation measures are found for σ = 0.01 and C ≥ 40 (Tab. 4). Especially, in the case of C = 40, the F-measure becomes 73.16% on average, and the recall is 68.84%. The accuracy measure is 93.18% when C = 30. In the SMV-POLY



case, the highest values for most evaluation measures are found for θ = 1 (recall, accuracy and F-measure) and for p ≥ 4 (Tab. 5). Especially, for θ = 1 and p = 4, the accuracy is 93.04% and the F-measure is 73.11% with the corresponding value of recall at 67.17%. For p = 5, recall achieves 68.46% with the corresponding value of F-measure at 72.41%, a value that is quite close to the maximum F-measure of SMV-POLY case. For the DT case, the C5.0 algorithm achieves 77.04% on average for the F-measure and the corresponding recall and accuracy measures are 86.74% and 94.15% respectively, the largest evaluation measures compared to other classification methods (Tab. 6). For the NB classifier, simulation showed that it is not as effective as the other classification methods (F-measure is 53.31% on average). For the LR classifier the results are quite similar with NB, where despite the high accuracy, 87.94%, the average value of F-measure is only 14.46% (for 100 Monte Carlo iterations), due to low precision and recall (Tab. 6). To summarize, simulation results showed that BPN, with the number of hidden neurons of hidden layer to be less than 20, and DT-C5.0 classifiers are the most effective for the churn prediction problem on this specific dataset, with the SVM classifier to be very close. The performance of NB and LR classifiers fall short compared to BPN, SVM and DT.

5.2.2. **Classifiers performance with boosting:**

Next, we apply the AdaBoost.M1 algorithm to explore the impact of boosting to the performance of the classifiers. Adaboost.M1 algorithm can only be applied to BNP, SVM and DT classifiers. Boosting requires the free model parameters to tune something that cannot be done in the case of NB or LR classifiers (the lack of parameters implies that the application of the classifier to the dataset several times will always give the same result). For BPN with five internal weak learners, k = 5, accuracy is 95% and F-measure is nearly 81%. Similar results were obtained for ten internal weak learners, k = 10; accuracy is 94% and F-measure achieves 80.2% (Tab. 7) . In order to apply boosting for the case of the SVM classifier we use different parameter sets of equal length to the size of the internal weak learners. For the SVM-RBF classifier we used two sets for the parameter C, C = 10, 20, 30, 40, 50 and C = 60, 70, 80, 90, 100. For the first set the Fmeasure is 76.18%, similar to the result without the use of boosting. For the second set, the improvement of performance is significant since accuracy is 96.05% and F-measure achieves 84.22%. For SVM-POLY classifier, tuning of the parameter θ does not result to high performance, since the achieved F-measure is only 40.84%. On the other hand, when the polynomial degree parameter p is tuned, then the boosting classifiers has significant improvement on its performance, since accuracy is 96.85% and the F-measure is 84.57%. The results are presented in Tab. 8. In the DT case, the use of AdaBoost.M1 results in accuracy of 95.09% and the F-measure obtained is 83.87% (Tab. 9).

5.3. **Performance comparison** –

Discussion A performance comparison of the BPN, SVM and DT classifiers for the version with and without boosting is presented in Tab. 10. It is apparent that the use of boosting improves significantly the classifiers’ performance, especially the F-measure metric. The improvement of the accuracy metric is 1% in the case of the BPN, 3% for SVM-RBF, 4% for SVM-POLY, and almost 1% for DT. For the F-measure the improvement is substantially higher, 4.5% in the case of the BPN, more than 15% in the case of the SVM-RBF and SVM-POLY, and almost 9% in the case of DT. This improvement rate may be substantial in the case of a telecom provider where a typical database includes, on average, millions of customers. Our experiments suggest that a) the use of boosting can significantly improve the classification performance, and it is therefore recommended and b) SVM compare to the other methods is the most powerful tool for tackling the problem of customers churn prediction.

6. **Conclusion and Summary:**

Monte Carlo simulations were performed using five of the most popular, state-of-the-art classification methods for the churn prediction problem of telecom customers based on a publicly available dataset. Initially all methods were tested without the use of boosting under different settings. The two top performing methods in terms of corresponding testing error were the two-layer Back-Propagation Network with 15 hidden units and the Decision Tree classifier; both methods achieved accuracy 94% and F-measure 77%, approximately. The Support Vector Machines classifiers (RBF and POLY kernels) obtained accuracy of about 93% and an approximate F-measure of 73%. Na¨ıve Bayes and Logistic Regression methods fail short with approximate accuracy 86% and an F-measure of about 53% and 14%, respectively. Subsequently, we investigated the impact of the application of boosting to the corresponding classifiers using the AdaBoost.M1 algorithm. Na¨ıve Bayes and Logistic Regression classifiers cannot be boosted due to the lack of free parameters to be tuned. Comparative results showed performance improvement for all three remaining classifiers due to boosting. Accuracy has been improved between 1% and 4%, while F-measure between 4.5% and 15%. Overall, the best classifier was the boosted SVM (SVM-POLY with AdaBoost) with accuracy of almost 97% and F-measure over 84%. This work has shed some light on the performance of popular machine learning techniques for the churning prediction problem and supported the advantage of the application of boosting techniques. In future work we plan to explore additional simulation schemes for the parameters of the weak learners for the AdaBoost.M1 algorithm, and to explore the performance of additional boosting algorithms beyond AdaBoost. Also, to use a larger and more detailed data set from the telecom industry in order to maximize the statistical significance of our results.

## Some Important Discussions

In the telecommunications industry, several practices have emerged to analyze churn, including:

* Derive metrics for four fundamental categories:
  + **Entity (for example, a subscription)**. Provision basic information about the subscription and/or customer that is the subject of churn.
  + **Activity**. Obtain all possible usage information that is related to the entity, for example, the number of logins.
  + **Customer support**. Harvest information from customer support logs to indicate whether the subscription had issues or interactions with customer support.
  + **Competitive and business data**. Obtain any information possible about the customer (for example, can be unavailable or hard to track).
* Use importance to drive feature selection. This implies that the boosted decision tree model is always a promising approach.

The use of these four categories creates the illusion that a simple *deterministic* approach, based on indexes formed on reasonable factors per category, should suffice to identify customers at risk for churn. Unfortunately, although this notion seems plausible, it is a false understanding. The reason is that churn is a temporal effect and the factors contributing to churn are usually in transient states. What leads a customer to consider leaving today might be different tomorrow, and it certainly will be different six months from now. Therefore, a *probabilistic* model is a necessity.

This important observation is often overlooked in business, which generally prefers a business intelligence-oriented approach to analytics, mostly because it is an easier sell and admits straightforward automation.

However, the promise of self-service analytics by using Machine Learning Studio is that the four categories of information, graded by division or department, become a valuable source for machine learning about churn.

Another exciting capability coming in Azure Machine Learning Studio is the ability to add a custom module to the repository of predefined modules that are already available. This capability, essentially, creates an opportunity to select libraries and create templates for vertical markets. It is an important differentiator of Azure Machine Learning Studio in the market place.

We hope to continue this topic in the future, especially related to big data analytics.

Top 5 Challenges & Solution Strategies

Challenge 1: “Mixed” dataset

Problem: The dataset is made up of both categorical variables and numerical variables, with one variable

type not significantly dominating the other one. According to , “with mixed data, choices are

limited and you need to be cautious and creative with your choices.”

Strategy: A good strategy to overcome the shortcomings of “mixed” datasets could be found in this

posting regarding the k-Nearest Neighbor algorithm: “K-NN algorithm requires the features to

be either all categorical or continuous. In the case of mix of these two, the categorical variables may be

mapped to numerical values before applying the K-NN algorithm.” We believe that there is a stronger

case to “make” all features numeric instead of making them all categorical, however certain methods

such as Naive Bayes become practically inapplicable. We take this decision in favour of a “broad”

benchmark of algorithms with very different assumptions such as kNN, Logistic Regression and different

tree variants.

However, it would have been also a reasonable (and possibly better) approach to simply work with a

mixed dataset, as several tree variants can perfectly deal with both numeric and nominal variables. Also

Naïve Bayes algorithm can work well with a mixed dataset under certain considerations.

Challenge 2: Small dataset

Problem: The biggest problem with small datasets is that they may present high variance. In general,

variance specifies the estimate of change of target function on different training data. Ideally the

variance should remain small, meaning that the algorithm used is good at connecting the hidden

underlying mapping between the inputs and the outputs. Unfortunately, there can be numerous

negative consequences of high variance, such as:

- significant over-fitting (i.e. fitting to peculiarities of the training data),

- high danger of outliers,

- high danger of noise ( both in class variable or in features).

Strategies: We found excellent advice on what to do with small datasets on . Most relevant

strategies found there and actually applied in our project are:

- Usage of simple models with strong assumptions.

- Usage of small set of possible hypotheses.

- Reasonable feature selection/elimination, possible consideration of brute force approaches

such as greedy forward selection of subsets of features.

Challenge 3: Class imbalance

Problem: An excellent explication about the impact of class imbalance could be found on

“ML algorithms struggle with accuracy because of the unequal distribution in dependent variable. This

causes the performance of existing classifiers to get biased towards majority class. The algorithms are

accuracy driven i.e. they aim to minimize the overall error to which the minority class contributes very

little. ML algorithms assume that the dataset has balanced class distributions. They also assume that

errors obtained from different classes have same cost”.

Strategy: Usage of “Oversampling” method (see Chapter 2.5 for detailed discussion) consisting in

replicating the whole minority class n times, in order to “inflate” minority class and allow algorithms to

better perform by clearer discriminating “lost” customers against “active” customers.

Challenge 4: Optimal Churn factor

Problem: As stated in the former paragraph, “Oversampling” will be used to improve predictive

performance. However, there is no clear rule at all how exactly the Churn factor should be set, i.e. which

ratio between Churn and Not Churn data points in the dataset would be optimal. Also, it might be

interesting to validate the performance of the original dataset (without any sampling) against the

oversampled dataset, to better understand the advantages and disadvantages of oversampling.

Strategy: All chosen algorithms will be run several times against different versions of the same dataset.

to develop classification models although the optimal ratio may depend upon the dataset

characteristics.”

Based on the CHURN-specific results, a comparative evaluation will be elaborated based on all three

datasets, complementing the individual assessments of the algorithms in the context of just one dataset

(“one by one evaluation”). The idea behind this comparative evaluation is to observe how evaluation

metrics such as “AUC” change for each algorithm if sampling rate in dataset is increased, as motivated in a case study.

Challenge 5: Choice of evaluation metrics

Problem: In Churn Prediction, it is very important to put special emphasis on the correct interpretation

of the two common error types (False Negative / False Positive), where “False Negative” is the by far most important one.

common error types in Churn Prediction: Type I Error - False Negative: Failing to identify a customer who

has a high propensity to unsubscribe. From a business perspective, this is the least desirable error as the

customer is very likely to quit/cancel/abandon the business, thus adversely affecting its revenue. Type

II Error - False Positive: Classifying a good, satisfied customer as one likely to Churn. From a business

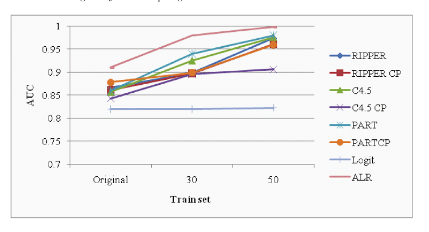
perspective, this is acceptable as it does not impact revenue”.

Strategy: When evaluating results, we will put special weight on minimizing False Negatives, without forgetting about meaning of overall accuracy as well as AUC values. It should be clear that “accuracy” cannot be the most important performance metric, as the dataset is not symmetric and costs of false positives and false negatives are not roughly the same. As we are working on a practical “real world” problem, interpretability of output will be another evaluation criterion, giving certain preference to rule-

based algorithms (such as Decision Trees) over “black box algorithms” (such as Logistic Regression).The general recommendation to use false negatives as most important performance metric.

usually the class of interest. Therefore, higher true positive rate and lower false positive rates are better

measures than the overall accuracExample of comparative performance evaluation



FUTURE SCOPE OF IMPROVEMENT OF THE PROJECT

The proposed models can be further enhanced, if the processes can be parallelizing.

This is feasible, by identifying operations that are independent to each other and propose a parallel architecture to improve the performance.

Amount of memory used loyalty assessment and action discovery is another area which can be analyzed in future.

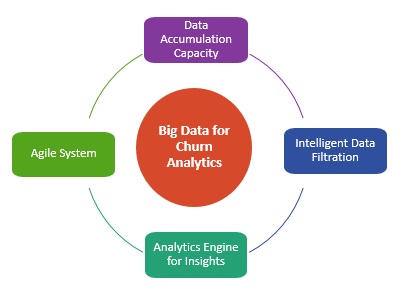
Classification process can be improved by using advanced techniques like ensemble clustering or ensemble classification.

Customer Churn is a term every business dreads, especially those which rely heavily on customer experience. Todays customers are well informed and armed with multiple options. They want competitive pricing, value for money and, above all, high quality service and wont hesitate to switch providers if they dont find what theyre looking for. As such, it becomes crucial to put in place a sustainable and robust strategy for customer retention to preserve customer lifetime value. Businesses that have a high acquisition low margin model are highly affected by customer churn and need to ensure quick real time decisions to lower the impact.

Current churn analysis techniques rely heavily on Customer Lifetime Value calculations, which are based on certain fixed metrics such as average monthly transactions, average gross margin, monthly retention rate, and so on. All of these metrics are highly number driven and not behavior based. However, in a volatile services market, it is imperative to not overlook the hidden metrics related to customer behavior and modifiers. Identifying the hidden value from vast layers of data through articulate filtering can be a differentiating factor. Thats where a well-defined Big Data analytics structure can be effectively leveraged. The more intelligent and accurate the structure is, the more predictive strategic business decisions would be able to be.

A well-structured Big Data analytics model will help re-define existing predictive churn analytics techniques. Businesses can now tap into non-traditional sources such as social media data analytics, customer touch point feedback, call center feedback and many more to create a holistic analytics model that deals with, not only the monetary value but also the behavioral patterns of the customer. This will allow the businesses to identify potential churning customers and act on preventing the churn. This proactive strategy can deliver better results than waiting for a trigger that indicates an inevitable churn and then trying to re-capture the customer. As such, any analytics model should have :

* The capacity to accumulate vast unstructured data from both traditional and non-traditional sources.
* The ability to superimpose intelligent filters to reduce noise and false alerts
* Business specific analytics engines to co-relate data, detect patterns, and generate real-time business insights
* Agility, through flexible tuning of rules and models for near real time streaming of data



An intelligent predictive churn analytics model, powered by Big Data analytics will allow businesses to process, analyze, and co-relate traditional and non-traditional metrics to achieve a holistic customer blueprint and effective insights that can trigger an alarm way before real damage is done. A simple example can be that of personalized retention incentives. Businesses can combine the insights from traditional churn analytics models such as average transaction value, monthly discount values, last transaction date, and so on with data from non-traditional sources such as brand or product sentiment on social media, number of complaints in the last month to the call center, competitor offers, and others. They can use this, to predict the churning intent of the said customer and quickly create a customized offer to try and retain them.

Predictive churn analytics is a small step towards automated personalization, which will be a critical business differentiator delivered by full-fledged Big Data adoption. However, businesses will have to start from small use cases, strategizing progressively to encompass complex multi-level use cases to realize the full potential of Big Data analytics. Businesses have to gear up and ensure that they can manage the speed and complexity of Big Data, establish well defined data points, and equip employees with enough training to handle the process complexities. Most importantly, businesses will have to shift the traditional Business Intelligence mindset of reporting and adapt to the real-time action mindset to successfully decipher the holistic customer insights that Big Data analytics is capable of providing.

## conclusion

Churn prediction is a function that involves systematic analysis of customer data for identifying and analyzing patterns and trends of customer loyalty and blend. The detected patterns and trends can be used by telecommunication industries to improve customer relationship and at the same time improve net profit. Identification of churners and nonchurners is a time consuming and critical task, that has to be performed carefully, as the future growth of the company relies on the result of such an analysis. This task is considered challenging because of two reasons, (i) customer information volume has increased and (ii) the data available is inconsistent and are incomplete thus making the task of formal analysis a difficult task. Further, due to its vast size, investigation and analysis of customer database takes longer duration due to the complexity of these issues. As information science and technology progress, sophisticated data mining and artificial intelligence tools are increasingly accessible to the telecommunication sector. These techniques combined with state-of-the-art computers can process thousands of instructions in seconds, saving precious time. In addition, installing and running software often costs less than hiring and training personnel. Computers are also less prone to errors than human investigators, especially those who work long hours. The current needs of telecom companies is a tool that can be used to help them to understand customer patterns and locate churners and possible actions that can be taken to convert the churners to non-churners. This tool is called as ‘Customer Loyalty Assessment Model and Actionable Knowledge Discovery System’ and the main goal is to provide timely and pertinent customer information to decision-makers in a company. The present research work focus on developing such a system that can be used by telecom industry easily discover customer patterns and trends, make forecasts, find relationships and possible explanations and identify possible churners. The proposed system proposes the use of data mining techniques during the design and development. To obtain an extra edge over competitive business, telecommunication industries are 175 relying more and more on CRM combined with data mining techniques. In this study, customer’s churning behaviour is predicted along with actionable knowledge discovery. The proposed system consists of three main steps, namely, data preprocessing, customer loyal assessment and actionable knowledge discovery. Each of the three steps is treated as a separate research phase and the phases are interconnected to each other, where the output of one phase is taken as input by the next phase. As incomplete dataset and presence of outliers during churn management process reduces the efficiency of prediction, a preprocessing step handling the missing values and outlier removal is used. In the first phase, preprocessing, a missing value handling procedure and outlier detection algorithm is proposed. The missing value handling procedure enhances the operation of a K-Nearest Neighbour imputation method by combining it with Learning Vector Quantization and Relevance Learning. A iterative procedure for handling missing value is also proposed. In the same phase, the density based Local Outlier Factor (LOF) algorithm is enhanced for detecting outliers in the customer data. This algorithm is enhanced through the use of speed optimizers and a data partitioning algorithm. The use of partitioning algorithm makes it scalable and the speed optimizers reduces the time complexity of the algorithm. In the second phase of the study, classification and clustering algorithms are analyzed for customer loyalty assessment. In this research work, a combination method that combines clustering with classification is proposed for this purpose. The proposed algorithms performs loyalty assessment in two steps. The first step performs clustering, while the second step performs classification. Clustering step identifies churners and nonchurners, while classification divides churners into three groups, namely, low, medium and high churners. In this system, instead of using clustering algorithm to improve the classification process, the clustering algorithm is used as a speed optimizer that reduces the dataset to have only churners. The first step, uses a hybrid clustering model that combines the advantages of SOM (Self Organizing Maps), KMeans and DBScan clustering algorithms to separate loyal 176 customers and churners. The second step analyzes the use of three classifiers, namely, Support Vector Machine (SVM), Back Propagation Neural Network (BPNN) and Decision Tree (DT) classifiers. The final phase of the study focuses on discovering action sets, which can be used as knowledge to convert churners to loyal customers and plan promotional activities to maintain loyal customers. For this purpose, the system uses dimensionality reduction algorithm, based on ACO (Ant Colony Optimization) with UDA (Uncorrelated Discriminant Analysis). Two classifiers, namely, a decision tree learning algorithm and Bayesian network classifiers are then used to build the customer profile. Finally, the last step performs a search operation to find optimal actions for each customer. The performance evaluation stage was conducted in four stages, each analyzing the performance of the missing value handling algorithm, outlier detection algorithm, customer loyalty assessment model and actionable knowledge discovery system. A telecom customer dataset from IDEA customer care was used during experiments. The missing value handling algorithm was evaluated using two performance metrics, namely, Normalized Root Mean Square and speed. Similarly, the outlier detection algorithm was evaluated using outlier detection rate and speed. The customer loyalty assessment procedure used accuracy, error rate and speed to analyze the algorithms. The actionable knowledge discovery system analysis was based on net profit achieved and speed of discovery. The various experiments conducted proved that the proposed algorithms and the proposed CRM system for customer loyalty assessment and actionable knowledge discovery are efficient. Experimental results showed that the system is effective in terms of analysis accuracy and speed in identifying common customer behaviour patterns and future churn prediction. The developed system has promising value in the current constantly changing telecommunication industry and can be used as effectively by companies to improve customer relationship and improve business opportunities.

**Certificate**

This is to certify that Ms. BRATATI GHOSH of GOVT. COLLEGE OF ENGINEERING AND CERAMIC TECHNOLOGY,

registration number: 171130110052 OF 2017-18, has successfully completed a project on CUSTOMER CHURN using MACHINE LEARNING under the guidance of Mr. TITAS ROWCHOWDHURY .

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

[Name of your faculty]

Globsyn Finishing School

**Certificate**

This is to certify that Ms. ALIPTA BASAK of GOVT. COLLEGE OF ENGINEERING AND CERAMIC TECHNOLOGY,

registration number: 171130110040 OF 2017-18, has successfully completed a project on CUSTOMER CHURN using MACHINE LEARNING under the guidance of Mr TITAS ROWCHOWDHURY .

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

[Name of your faculty]

Globsyn Finishing School

**Certificate**

This is to certify that Ms. ANTARA RAY of GOVT. COLLEGE OF ENGINEERING AND CERAMIC TECHNOLOGY,

registration number: 171130110045 OF 2017-18, has successfully completed a project on CUSTOMER CHURN using MACHINE LEARNING under the guidance of Mr TITAS ROWCHOWDHURY .

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

[Name of your faculty]

Globsyn Finishing School

**Certificate**

This is to certify that Ms. MOHINI DAS of GOVT. COLLEGE OF ENGINEERING AND CERAMIC TECHNOLOGY,

registration number: 171130110056 OF 2017-18, has successfully completed a project on CUSTOMER CHURN using MACHINE LEARNING under the guidance of Mr TITAS ROWCHOWDHURY .

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

[Name of your faculty]

Globsyn Finishing School