**Czech University of Life Sciences Prague**

**Faculty of Economics and Management**

**Department of Information Engineering (FEM)**

# Master's Thesis

## Artificial Intelligence methods for decision making

**Mithun Shivashankar**

# CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

# DIPLOMA THESIS ASSIGNMENT

Mithun Bharadwaj Shivashankar, B.E.

Systems Engineering and Informatics
Informatics

Thesis title

**Artificial Intelligence methods for decision making**

---

**Objectives of thesis**

Give an overview of Artificial Intelligence methods that are used to support decision making. Design a suitable solution method for the selected specific decision problem and implement it using publicly available software and data.

**Methodology**

The student will give an overview of Artificial Intelligence methods that are used to support decision making. Then he will focus on solving a specific decision problem and use one of these methods to solve it. The proposed solution will be verified on data publicly available in a database intended for testing Artificial Intelligence methods. For testing the student will use freely available software.

**The proposed extent of the thesis**

60 pages

**Keywords**

Artificial intelligence, neural networks, machine learning, decision making

---

**Recommended information sources**

GÉRON, A. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.* Beijing ; Boston ; Farnham ; Sevastopol ; Tokyo: O'Reilly, 2019. ISBN 978-1-492-03264-9.

Gupta I., Nagpal G.: Artificial Intelligence and Expert systems, 2020, Boston

---

**Expected date of thesis defence**

2020/21 SS – FEM

**The Diploma Thesis Supervisor**

doc. Ing. Arnošt Veselý, CSc.

**Supervising department**

Department of Information Engineering

Electronic approval: 23. 11. 2021

**Ing. Martin Pelikán, Ph.D.**

Head of department

Electronic approval: 25. 11. 2021

**Ing. Martin Pelikán, Ph.D.**

Dean

Prague on 29. 03. 2022

**Declaration**

I declare that I have worked on my master's thesis titled Artificial Intelligence methods for decision making by myself and I have used only the sources mentioned at the end of the thesis. As the author of the master's thesis, I declare that the thesis does not break any copyrights.

In Prague, 29th March 2022 Mithun Shivashankar

**Acknowledgement**

I would like to thank doc. Ing. Arnošt Veselý, CSc. and all other persons, for their advice and support during my work on this thesis.

# Artificial Intelligence Methods for Decision Making

**Abstract**

In today's parlance, data is the underpinning ingredient of the edifice of the modern organisation. But the data that we have is available in billions of units, which would make it impossible for the human brain to process them and understand the underlying subtleties. Hence enter artificial intelligence for decision making.

With the aid of artificial intelligence (AI) for decision making, many large organisations use AI in the company's processes to make swift, precise and reliable decisions by taking the advantage of the dataset combined with AI. Artificial intelligence is the best way to get the best possible decision making processes by its unique ability to teach itself.

AI algorithms have the ability to teach themselves with the help of the data sets. They allow the companies to perform predictions, classification, and recommendations in the real time. That helps businesses to make good decisions where they can benefit commercially and increase their market share.

**Keywords:** Artificial intelligence, algorithms, regression, classification, prediction, datasets, decision making.

# Metody Umělé Inteligence pro Rozhodování

**Abstrakt**

V dnešní době jsou data klíčovou součástí majetku moderních organizací. K dispozici máme obrovské množství dat. Dat je již tolik, že je lidský mozek nedokáže zpracovat, natož porozumět jednotlivým detailům a souvislostem. Pro rozhodování je tedy třeba používat inteligenci umělou.

Mnoho velkých organizací využívá kombinaci umělé inteligence a velkých datových souborů ve svých postupech a činnostech. Organizace tak mohou činit rychlá, přesná a spolehlivá rozhodnutí. Umělá inteligence se svou unikátní schopností sama se učit, je nejlepší způsob, jakým je možné získat rozhodnutí nejkvalitnější.

Algoritmy uměle inteligence jsou schopny samy se učit s pomocí datových sad. Organizace tak mohou využívat výhod okamžitých předpovědí, klasifikací a doporučení, čímž mohou dosáhnout vyššího zisku a zvětšit svůj podíl na trhu.

**Klíčová slova:** Umělá inteligence, algoritmy, regrese, klasifikace, redukce, soubory dat, rozhodování.

# Table of Contents

1 Introduction....12

1.1 Machine Learning Core Concepts....12

1.2 Supervised Learning....13

1.2.1 Classification Problems....14

1.2.2 Regression Problems....14

1.3 Unsupervised Learning....15

1.3.1 Clustering....15

1.3.2 Dimensionality Reduction....15

2 Objectives and Methodology....16

2.1 Objectives....16

2.2 Methodology....16

2.2.1 Method Description....17

3 Literature Review....19

3.1 Training, validating, and testing sets....19

3.1.1 Training set....19

3.1.2 Validation set....20

3.1.3 Testing set....20

3.1.4 Rule of thumb – Accuracy....20

3.2 The Python Programming Language....20
3.2.1 Simple and Consistent....20
3.2.2 Extensive selection of libraries and frameworks....21
3.2.3 Platform independence....21
3.2.4 Great community and popularity....21
3.2.5 Why is Python used the most?....22
3.3 The Pandas Library....23
3.3.1 What is Pandas?....23
3.3.2 The biggest benefits....24
3.3.3 How does Pandas work?....24
3.3.4 Basic Pandas operations....24
3.4 Machine Learning Libraries....24
3.4.1 Sklearn....24
3.5 Machine Learning Algorithms....25
3.5.1 Support Vector Machine....25
3.5.2 Linear Regression....29
3.5.3 Neural Networks....32
3.5.4 Decision Tree....36
3.5.5 Cross validation....39
4 Practical Part....44

4.1 Implementation....44
4.1.1 Data Sets....44
4.1.2 Information about the dataset....47
4.1.3 Setting up the dataset for ML training....47
4.1.4 Code in Python....49
5 Results....55
5.1 Overview....55
5.2 Application outputs....55
5.2.1 Support vector machine....55
5.2.2 Logistic regression....56
5.2.3 Neural Network....56
5.2.4 Decision tree....56
6 Conclusion....57
6.1 Cross Validation....57
6.2 Tabulated Results....59
6.3 Tabulated Results with Cross-Validation....59
7 References....60
8 Appendix A – Full source code....61
9 Appendix B – Data Set....63

# 1 Introduction

Machine learning (aka ML) is a department of computer algorithms which aims to ameliorate axiomatically by way of experience and by the proper utilisation of data. It has been commonly associated with artificial intelligence. Machine learning algorithms construct a model from sample data, also called "training data", with the aim to make predictions or decisions which usually are not explicitly programmed to do so. Machine learning algorithms have been used in a broad range of applications, such as email filtering, speech recognition, self-driving cars, online fraud detection, computer vision, stock market trading etc. where it is hard or impractical to unfold traditional algorithms to carry out the necessary tasks.

Machine Learning (ML) is a vital aspect of new age business and its research. It utilises algorithms and neural network models in order to help computer systems steadily enhance its performance.

Hebb once wrote, “When one cell regularly aids in triggering another, the axon of the first cell develops synaptic knobs when it comes in contact with soma of the 2nd cell.” During the processes of rendering Hebb’s notions into artificial NN and other being the artificial neurons, Hebb’s model could be outlined as a system of method of altering the association amongst artificial neurons and the different variants to discrete neurons. The association amongst two neurons/nodes is bolstered if the two neurons/nodes are triggered simultaneously and debilitated if they are activated individually. The term “weight” is used to outline all these relationships, and nodes/neurons which can have both positive and negative are delineated as having rich positive weights.

## 1.1 Machine Learning Core Concepts

Several types of machine learning algorithm exists, several thousands of them are dished out each day, and they are generally teamed by either learning style or by similar resemblance in the form or functionality.

Irrespective of learning style or functionality, all types of machine learning algorithms contains the following:

- Representation
- Evaluation (aka objective/scoring function)
- Optimization

## 1.2 Supervised Learning

Supervised learning can be defined as task of understanding and learning a function that associates an input from the original dataset to an output with complete dependence on example input-output pairs.


The approach to supervised learning can be embraced when a dataset contains the records of the response variable values (Based on the context, this data alongside labels is generally referenced as "labelled data" and "training data.")

Illustration 1: When we seek to foresee an individual height from his weight, age, and gender, we will require the training data that has the individual's weight, age, gender information alongside with their real heights. This would allow the machine learning algorithm to find out the association between height and the other relevant variables. Then, using this awareness, the model can then foresee the height of an individual.

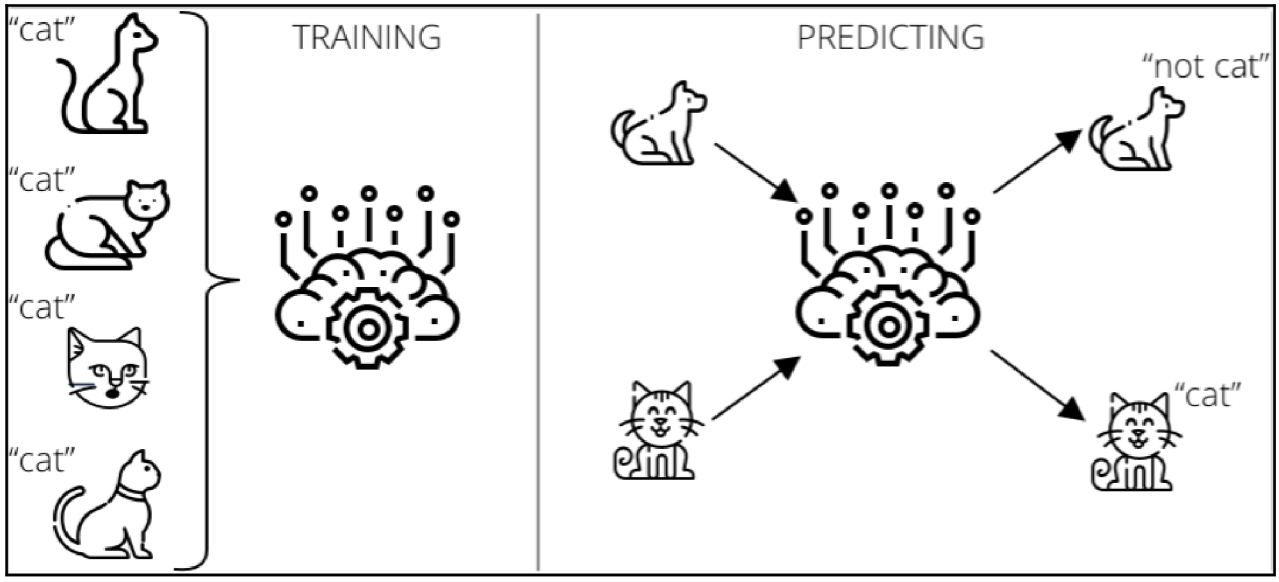
Example 2: Another case could be that we could mark emails either as 'spam' or 'not-spam' from the distinctive features of the earlier seen spam and that are not-spam emails, like the lengths of the e-mails and by the usage of specific keywords in the emails. Learning from training data goes on till the model reaches a peak level of precision on the training data.

Two main supervised learning problems are:

1. Classification Problems
2. Regression Problems

### 1.2.1 Classification Problems

In classification related issues, the models will learn to classify an observation from the variable values. Throughout the learning procedure, the model is wide opened to a plethora of observations alongside their labels. For instance, after watching several thousands of patrons and their shopping routines and gender info, a model may correctly foretell the gender of a next patron from on his/her shopping habits. Binary classification is a word commonly called for sorting under two labels, for example as male and female. Let's take another binary classification example that will be predicting if the animal in an image is a ‘cat’ or ‘not cat,’ as shown in the figure below.



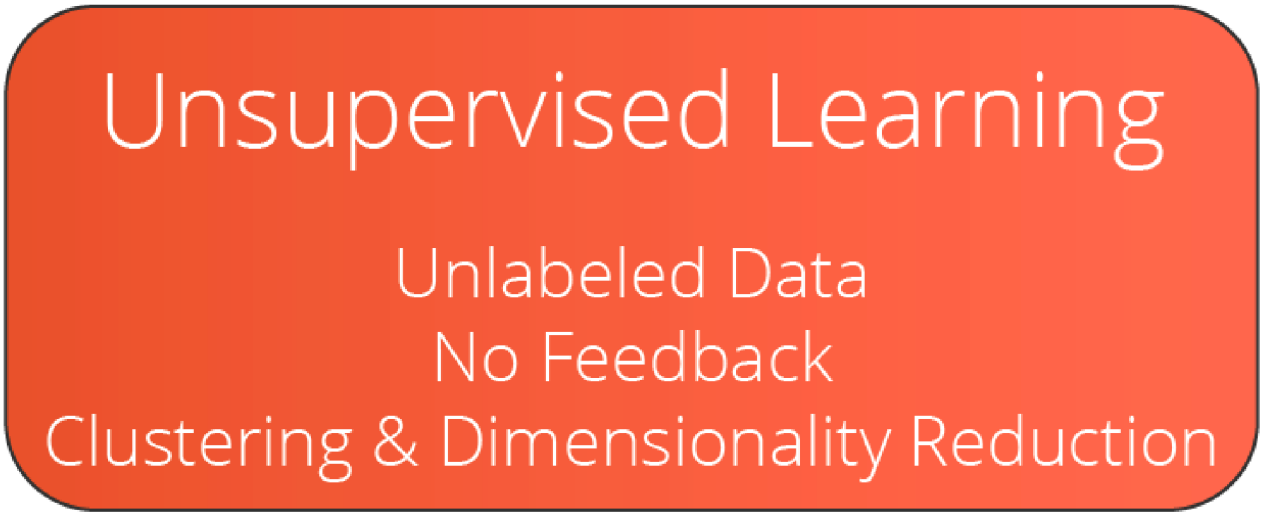

On the contrary, with the existence of more than two labels multi-label classification is often used. Detecting and foretelling letters written by hand and numbers on a picture could be an illustration of multi-label classification.

### 1.2.2 Regression Problems

For regression problems, the aim is to estimate a value from taking benefit of the correspondence between the other variables (*that is, the independent variables, explanatory variables*) and *dependent variables*. The advantage of the correspondence between the target variable and the other variables is a vital element of the prediction value. Forecasting how much a patron would be willing to spend from its prior data is a regression problem.

## 1.3 Unsupervised Learning

Unsupervised learning can be defined as a particular type of ML set of algorithms that seeks for previously patterns that go undetected/unidentified in a new data set that has no labels that was defined before and with a bare minimum of human interference.




### 1.3.1 Clustering

Unsupervised learning is mostly employed in clustering analysis. Clustering analysis is defined due to its grouping effort where the group members aka clusters are super identical similar to each other compared to the members of the remaining other clusters.

There are several clustering techniques available. They normally use a kind of familiarity measure from selected metrics like Euclidean or probabilistic distance. Bioinformatic sequence analysis, pattern mining and object recognition are few of the many clustering problems that could be confronted with the unsupervised learning technique.

### 1.3.2 Dimensionality Reduction

A different use case of unsupervised machine learning could be something called a dimensionality reduction. Dimensionality is analogous to the total amount of features available in a particular dataset. While in few datasets, you could detect few hundreds of potential features embedded in each columns. In all of the datasets, few of the columns are largely correspond to other columns. So, we should pick the best and the safest ones, i.e., *feature selection.*

# 2 Objectives and Methodology

## 2.1 Objectives

The objectives of the thesis are the following:

- Research to gather the data that would be fed to the Artificial Intelligence algorithm
- Parse the CSV data and convert it to a new CSV that will be processed by an AI algorithm
- Feed the data to the following algorithms that would make the predictions
    1. Support Vector machine
    2. Logistic regression
    3. Neural Network
    4. Decision tree
- Evaluate the following metrics from the model
    1. Accuracy of the model
    2. Accuracy of the model with CV
    3. Accuracy of the test set
    4. Accuracy of the train set
- Choose the best model that predicts the problem on hand

## 2.2 Methodology

- An overview of Artificial Intelligence methods that are used to support decision making is given
- The thesis is focused on solving a specific decision problem and one of the methods to solve the problem is used
- The proposed solution is verified on data publicly available in a database intended for testing Artificial Intelligence methods
- For testing, a freely available software is used

### 2.2.1 Method Description

#### Obtain the Dataset

The dataset is freely available. We will use Python library to read the dataset.

#### Managing the missing data

Missing data is a common occurrence in datasets. When we are dealing with real-time datas, missing data is quite frequent.

In order to train the data in an orderly fashion, we should do something about this missing data. Or, the AI model can misinterpret this missing information.

#### Transform the dataset into Training and Testing Sets

We should cut the dataset into ratio of 80% Training Data and 20% into test data. So, for our case, we should divided the dataset into 80% for training data and 20% for test data.

#### Training the model

With scikit-learn, the estimator is an instance that fits a model depending on the type of the input data (i.e. training data) and renders specific estimations that leads to properties on newly created unseen data. In so many words, an estimator can act as an a regressor or also a classifier.

The frame work comes with the base class *BaseEstimator* located at sklearn.base.BaseEstimator and a the remaining estimators must inherit from that specific class to do further calculations.When in packaged conditions the base class is shipped with two methods, called *get_params()* and *set_params()* as the method suggests helps to retrieve and set the parameters of an estimator respectively.

#### Fit method signature

fit(X, y, sample_weight=None)

We will fit it the Support Vector model based on the given training dataset.

| Symbol | Explanation |
|---|---|
| X | A Training vector, where n_samples denotes total number of samples and n_features is denotes total number of features |
| Y | Target values |
| sample_weight | Indicates per sample weights |

The estimators must explicitly pass all of the necessary parameters in the constructor method (i.e. the *__init__* method) that will be used in the initialization.

The *fit()* method is used by almost all the estimators and it takes an input for the sample data (X) and for supervised models it also accepts an argument for labels (that is. target data y ). Additionally as an available option, it takes in additional parameters such as like weights etc.

Generally speaking the fit methods are the reasons behind several operations/functions. Normally, they must eliminate attributes that have already stored inside the estimator object and then perform necessary validations. Attributes with in the input data are also eliminated by the fit() method and also they store the parameters and attributes of the particular model and return value is estimator.

### Predicting the model

Since now the training of the model is complete, the following step is performing predictions with the remaining testing set. In order to accomplish this job, we need to pass the parameters to predict method *predict()* that will perform the predictions with the help of learned parameters by *fit()* on the new data points and unseen test data points.

Typically, the *predict()* method will render a predictions for each available test instance and it typically takes only a single input (X).

### Cross Validation

When fine tuning the various models our goal is to increase elevate the efficiency and performance that data that cannot be seen. The performance of the tests sets can be increased significantly by Hyper-parameter tuning. But enhancing the parameters to the test set could cause exhalation of the information causing for the model to underperform. To fix this particular issue we can do cross validation. Also we use cross validation in this particular scenario because we have a smaller dataset

**Confusion matrix**

It is an actually a table that is particularly used in classification related problems to estimate where errors in the model were made.

The rows indicates the total number of actual classes the outcomes should have been. While the columns represent the predictions we have made. Using this table it is easy to see which predictions are wrong.

True indicates that the values were accurately predicted, False means the prediction that were erroneous.

We can calculate Confusion Matrix by running the following code, we can calculate different measures to quantify the quality of the model.

**Models used**

- logistic regression
- support vector machines
- neural networks
- binary decision tree

# 3 Literature Review

In 2022, the sources of information about artificial intelligence are plentiful and diverse, significant amount of the information is available in form of articles at websites that specialize on the topic or related topics.

## 3.1 Training, validating, and testing sets

### 3.1.1 Training set

The training data set is  used in order to train the AI models. It can witness and understand from the best suitable permutations and combinations that can produce a firm understanding ML learning model.

### 3.1.2 Validation set

The validation data set can be used in order to train the ML model with the main goal of selecting the best machine learning model and improving it it. ML engineers make use of the validation information to refine the model (TechTarget, 2022).

### 3.1.3 Testing set

The testing set is can be used in a situation where the model all the machine learning models completely trained with the help of training datasets and the validation datasets. It can also be used to assess the performance of the machine learning model.

### 3.1.4 Rule of thumb – Accuracy

Accuracy indicates the accurate predictions made in classification related tasks tasks.

## 3.2 The Python Programming Language

What makes Python the best programming language for machine learning and the best programming language for AI?

### 3.2.1 Simple and Consistent

Python delivers succinct and human readable code. Whereas convoluted algorithms and adaptable workflows bolsters machine learning and Artificial intelligence, Python's coherence makes way for software developers to build systems that are reliable. Software Devs could put more of their work into solving complex Machine Learning related problems rather than concentrating on the technical aspects of the language.

Furthermore, Python is attracting a lot of software devs since it is a lot easier to learn compared to its counterparts. Python code is comprehensible by humans, which will help software developers to build models for machine learning.

Lot of developers say that Python is a lot more instinctive when compared to other programming languages. Some others point out a lot of frameworks and libraries that are available which simplifies the implementation of convoluted functionalities. It's usually assumed that Python is made for collaborative execution when numerous software developers are galvanised. As Python is a general-purpose programming language, it can perform a set of intricate machine learning tasks

and allow you to create prototypes swiftly that enable you to test the application for machine learning related things.

### 3.2.2 Extensive selection of libraries and frameworks

Implementation of Artificial intelligence and Machine Learning algorithms can get a little bit tricky and it will eat up a lot of time. It is important to have a proper environment that enables software devs to produce best coding solutions.

To minimise the development cycle time, software devs will use a variety of Python frameworks and libraries. A library is a pre-written piece of code that devs will use in order to solve common and repetitive tasks.

Why is Python great? Here is a table of common Artificial intelligence use cases.

| | |
|---|---|
| DATA ANALYSIS AND VISUALIZATION | NUMPY, SCIPY, PANDAS, SEABORN |
| Machine learning | TensorFlow, Keras, Scikit-learn |
| Computer vision | OpenCV |
| Natural language processing | NLTK, spaCy |

### 3.2.3 Platform independence

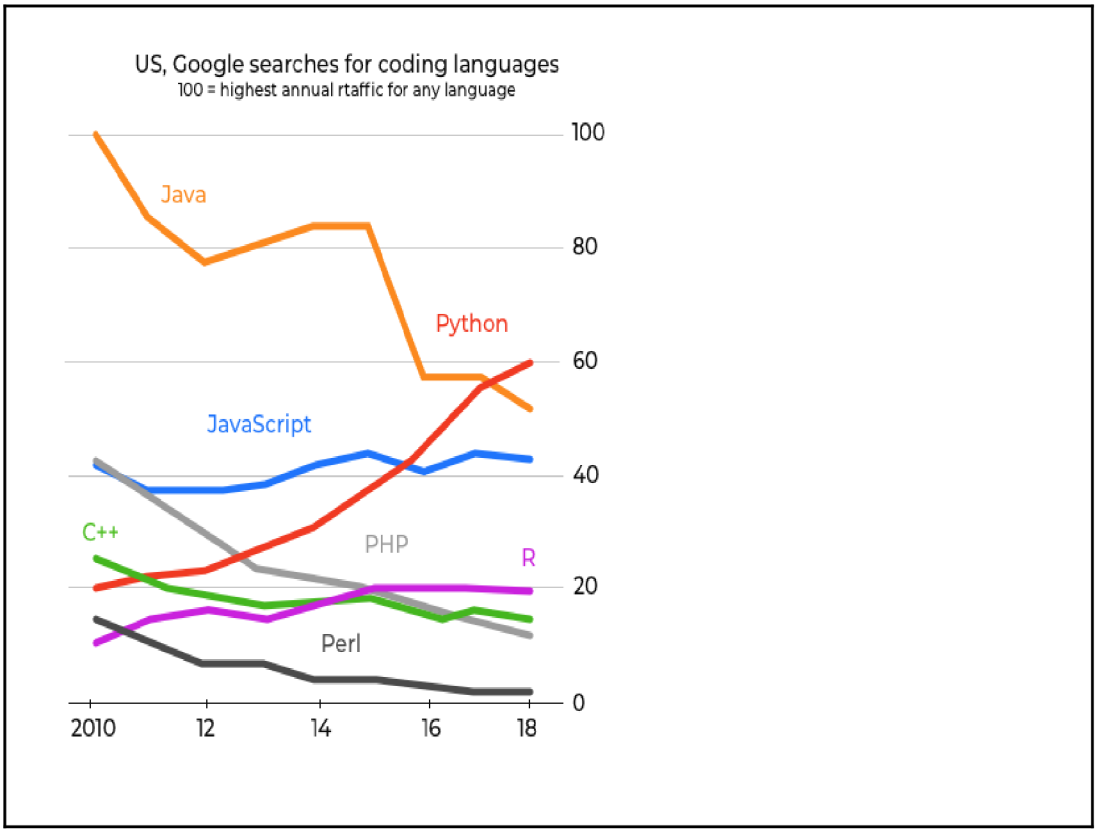
Platform independence means that a particular programming language allows software devs to enact things on a particular machine and easily migrate them on another machine without any changes/minimum changes. Another key to Python's popular recognition is that the language itself is platform independent. Python is known to support a wide variety of platforms including such as Linux, Windows, and macOS.

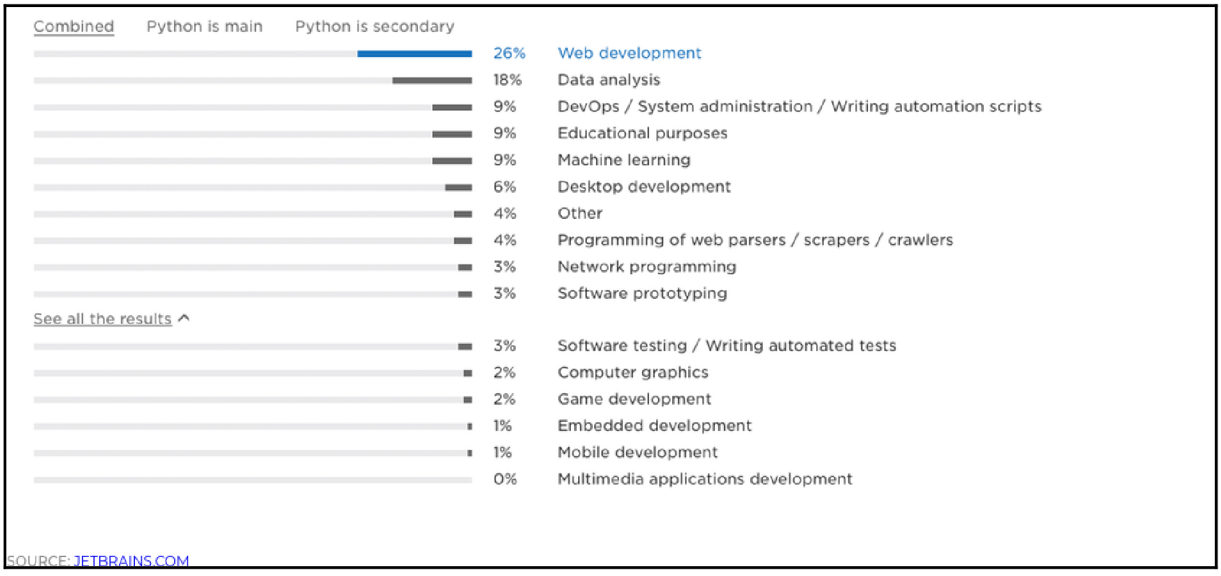
### 3.2.4 Great community and popularity

Developers survey performed in 2018 by Stack Overflow (a popular website where devs seek for help from each other), Python one of the top 10 most popular programming languages, which obviously means that it is easy to find and python developers with the necessary skills to build

AI/ML projects. Have a look at the below image, you'll witness that Python is a programming language that people Google search compared to any other programming language.

The growing popularity of the Python programming language is shown in the charts below.




### 3.2.5 Why is Python used the most?

Repositories online have more than 140,000 built-in Python related software packages. Python packages that are Scientific ilk Numpy, Scipy, and Matplotlib could be installed in an application

running on Python. These packages drive ML/AI and aid developers detect patterns in big sets of data. Google extensively uses Python for crawling web pages, syndicate Pixar makes use of Python in creating films, and recommending song engines is written in Python  Spotify.

It is a widely known fact that the Python Machine learning/Artificial Intelligence community has significantly grown over the globe. You can find advice and guidance from developers from a plethora of forums that are available.

## 3.3 The Pandas Library

The *pandas* package is a vital tool at the hands of a Data Scientist and Analysts working in Python as of today. The existing machine learning/artificial intelligence tools and fascinating visualisation tools might get all of developers  attention, but pandas is the foundation of  most ML/AI projects.

> [pandas] is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. — Wikipedia

If you ever plan to choose a data scientist or AI/ML engineer as a career, then it becomes really important to learn Pandas.

### 3.3.1 What is Pandas?


A Python Pandas data frame is more than  just an array. The Pandas package is a monstrous of a tool that allows you:

- Turn a JSON, CSV, lists, dictionaries, to a row and column format and work using names instead of indexes.

- Putting things plainly, Pandas is sort of like a spreadsheet, but one you work with using code, not Microsoft Excel.

### 3.3.2 The biggest benefits

- Pandas makes extremely complicated data transformations easy and natural.
- It includes a wealth of maths, analytics, and other functions.

### 3.3.3 How does Pandas work?

Pandas framework/library is built on top of two more popular framework one of them being NumPy and the other one is Matplotlib. Henceforth, Pandas can:

- Efficiently work with large n-dimensional arrays (
- Pickup chunks and convert them into several various shapes (NumPy)
- Draw charts (Matplotlib)

The other popular framework NumPy is the base framework for most of the existing Python machine learning SDKs.

### 3.3.4 Basic Pandas operations

Now, let's take the discussion further and move towards an easy explanation that delineates the basics of pandas

Create a DataFrame from an array

First create a DataFrame from an array.

## 3.4 Machine Learning Libraries

### 3.4.1 Sklearn

Sklearn is a popular open-source Python library that has built-in with a wide variety of machine learning like pre-processing, cross-validation, and visualisation with an integrated interface.

Popular features that scikit-learn ships are:

- It is an easy to use and efficient tool for mining the data and performing data analysis.

- It is packed with several classification algorithms, regression, as well as clustering algorithms, and also supports vector machines, random forests, k-means.
- Readily available to all and can be used in different contexts.
- The base framework that is Sklearn is built on the top are NumPy, SciPy, and Matplotlib.
- It is an open-source framework and could be used commercially.

Installation:

The Prerequisites are the following:

- NumPy
- SciPy and all its related dependencies.

In order to install sci-kit-learn, we will make sure Numpy and SciPy are installed to the machine that runs the Machine learning programs. Once they are installed the quickest way to install NumPy and SciPy

While installing the framework sci-kit-learn, make sure that the libraries such as NumPy and SciPy installed to the local machine. Once you have a working installation of NumPy and SciPy, the easiest way to install sci-kit-learn is with the help of the command pip.

PIP[https://en.wikipedia.org/wiki/Pip_(package_manager)]

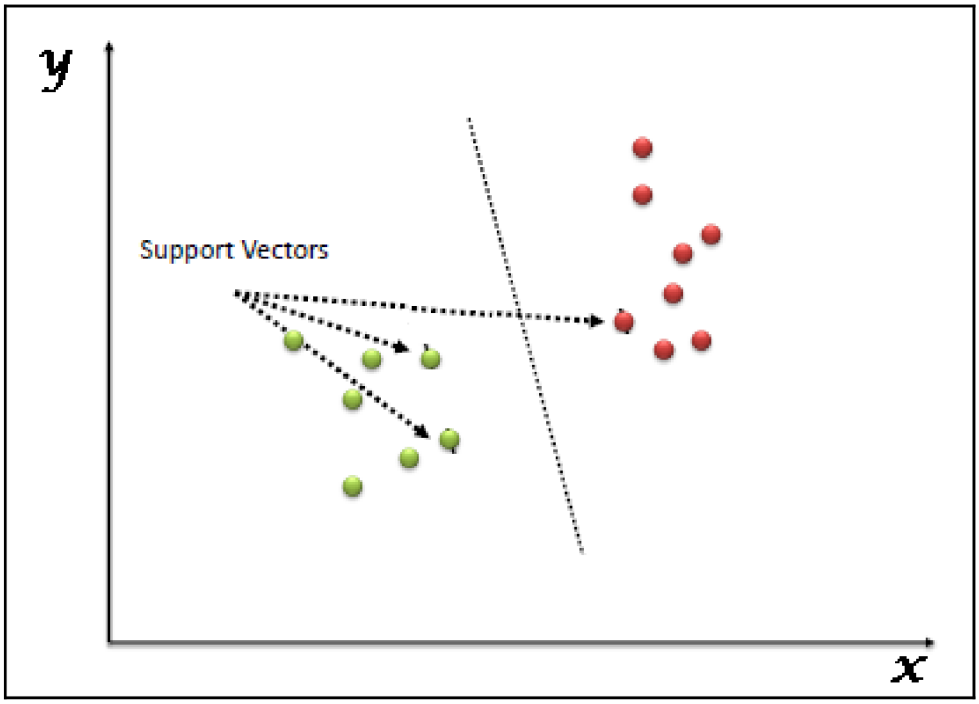
Pip is a package manager written in Python itself that is used to install and manage python packages software packages which will be imported into the python application. PIP communicates to an online repository where all the public packages are stored known as the Python Package Index. PIP could be modified to communicate with other package repositories (that are local or remote), assuming that they agree with Python Enhancement Proposal 503.

## 3.5 Machine Learning Algorithms

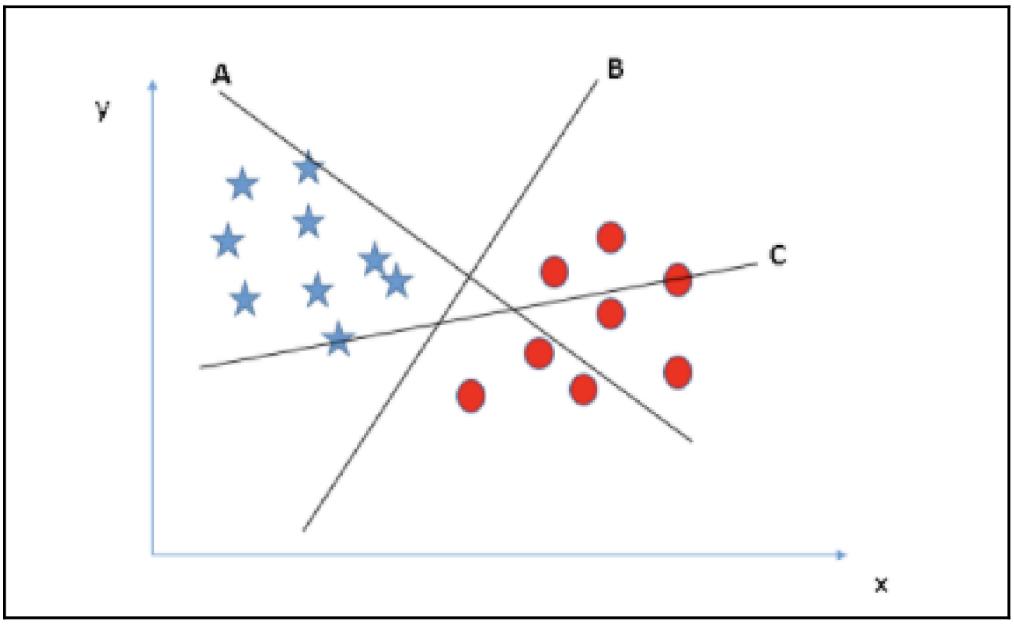
### 3.5.1 Support Vector Machine

The “Support Vector Machine” short for (SVM) is a supervised machine learning algorithm that could be used for a problem that faces classification or regression. But it is most likely used in classification-related issues. In the Support Vector Machine algorithm, we plot an individual data point as a point in n-dimensional space (where n represents a total number of features that we have)

the value of every feature denotes a value of some coordinate in space. Only After, we calculate classification by searching for the hyper-plane that separates the two known classes very well.



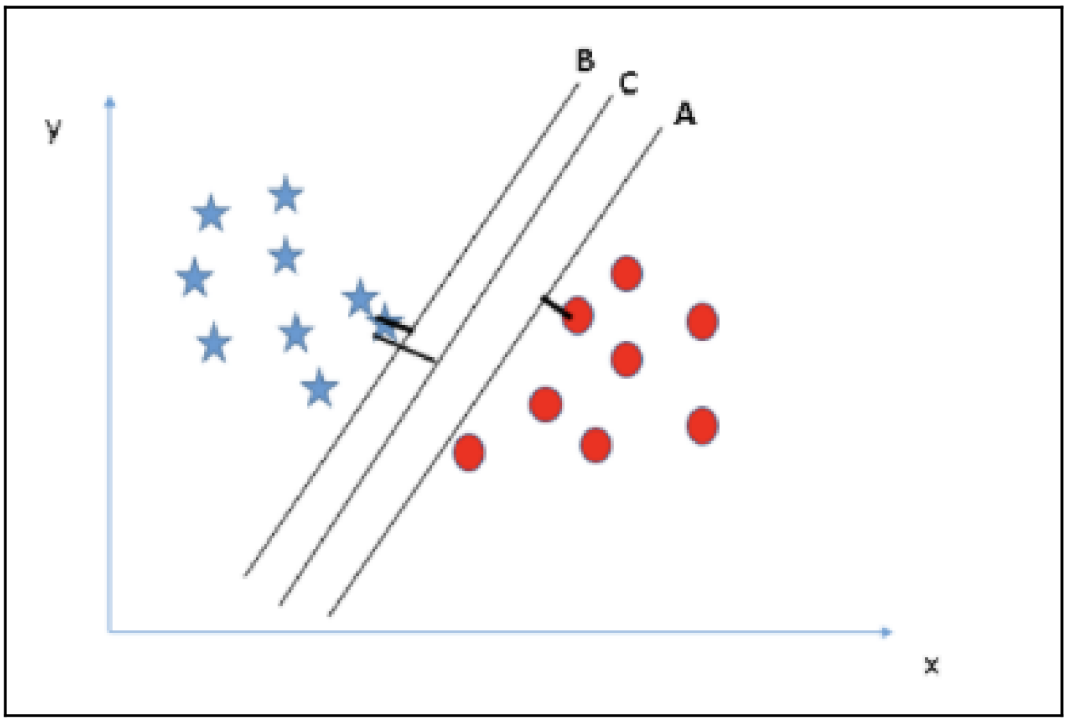

Support Vectors machines just simply represent the orientation of each observation. The Support Vector Machine classifier is a partition that best separates the two classes (hyper-plane or line).



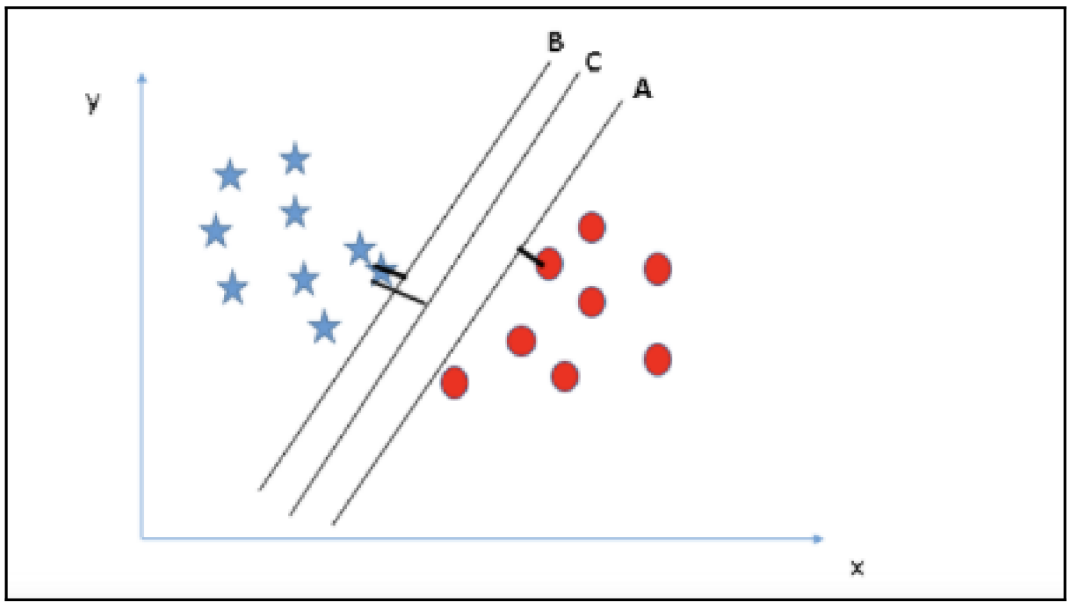

- We will need to have a golden rule to find the correct hyper-plane
  "Selects the hyperplane which segregates the two available classes way stronger". In this current case, hyper-plane "B" has done the job very well.

- Let's take Scenario-2

  Here, we got ourselves three hyper-planes (A, B, and C) and every hyperplane are separating the classes very well.

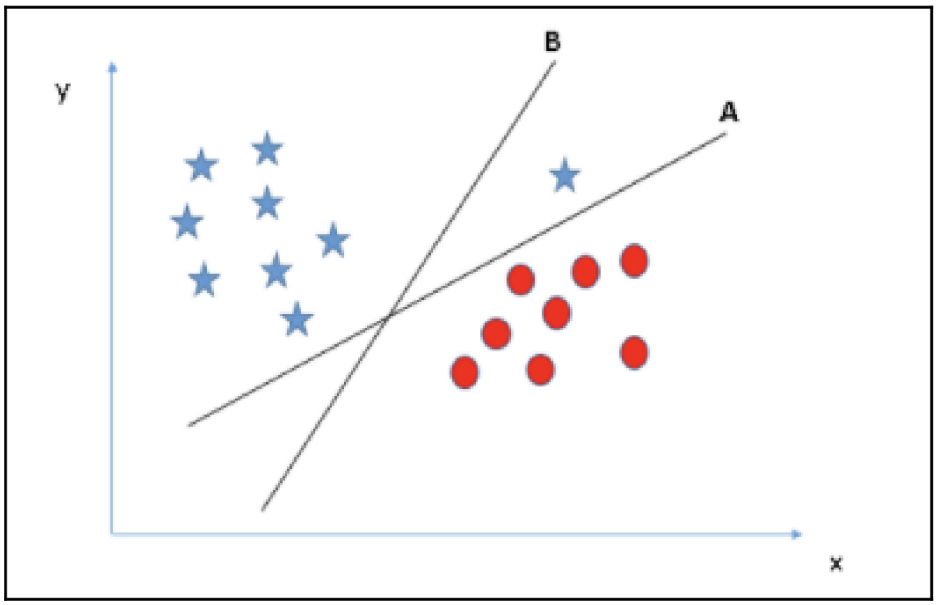



Increasing the space between nearest data nodes and hyper-plane will aid us to make conclusions about the proper hyper-plane. This distance or space is called Margin. Let's look at the below snapshot:



- Let's take Scenario-3

  Suggestion: Make sure to use the rules as talked about in above section to detect the correct hyper-plane.




In this case the hyper-plane B has what is called classification error and hyper-plane A has been classified properly. Hence the correct hyper-plane is A.

Pros and Cons associated with SVM

- Pros:
  - It works perfectly well with a clear margin of separation
  - It works the best in high dimensional spaces.
  - It is effective in scenarios where the total number of dimensions is more prominent than the total number of samples.
  - It makes use of a subset of training points in the decision function known as support vectors, so it is memory efficient.
- Cons:
  - It doesn't do well when we have a big data set because the necessary training is increased significantly.
  - It also doesn't perform very well, when the data set contains a lot of noise.

- Support Vector Machines will not, in straight forward manner, give us probability matrix.

### 3.5.2 Linear Regression

Putting it simply, Linear Regression belongs to a supervised ML category where the model finds itself as the best fit linear line amongst the independent and dependent variable which means the model holds linear association between the dependent variable and independent variable.

Linear Regression is of two types:

- Simple Linear Regression
  where only one independent variable is present and the model has to find the linear relationship with the dependent variable
- Multiple Linear Regression
  There are many independent variables in the model to find the relationship.

Equation of Simple Linear Regression

$$y = b_o + b_1 x$$

where $b_o$ represent intercept, b1 represent coefficient also known as slope, x represent the independent parameter and y represent the dependent variable.

Equation of Multiple Linear Regression

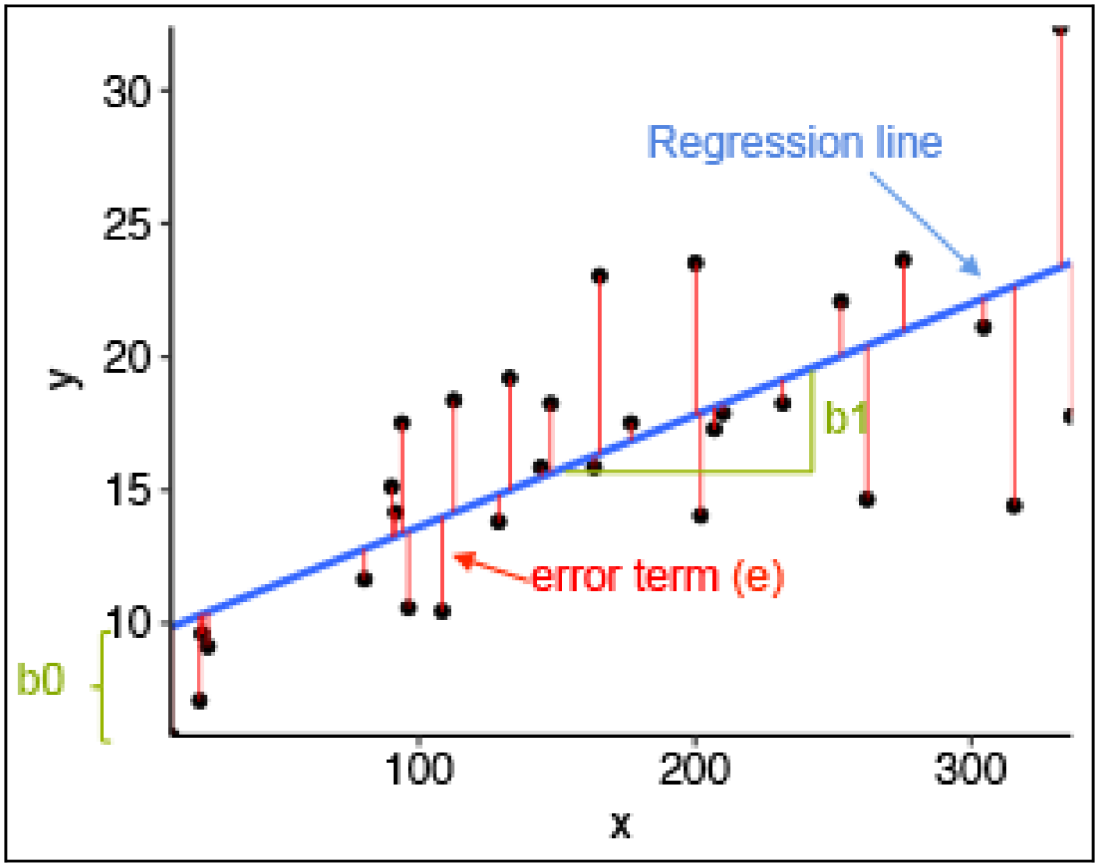
$$y = b_o + b_1 x_1 + b_2 x_2 + b_3 x_3 \ldots . + b_n x_n$$

where $b_o$ is the intercept, $b_1, b_2, b_3, b_4, \ldots, b_n$ are coefficients or slopes of the independent variables $x_1, x_2, x_3, x_4, \ldots, x_n$ and y is the dependent variable.

The goal of the linear regression model is to identify the finest fit linear line and the most desirable values of intercept and coefficients such that the error is minimised.

The residue between the actual and the predicted value is called error and the aim is to all pacify this difference.

Let's try to understand with the help of the figure below:




In the above diagram,

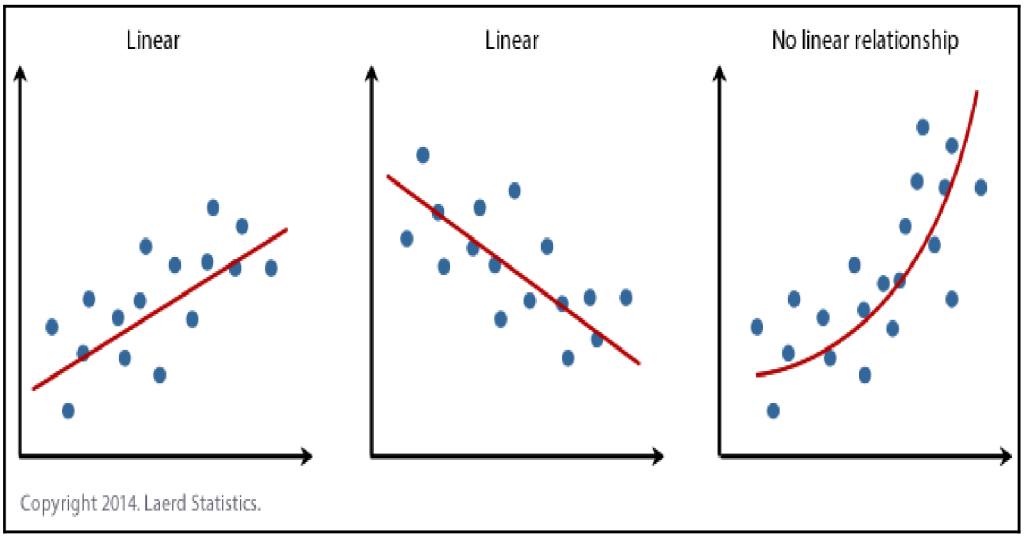
- x is the dependent variable which is delineate on the x-axis and the remaining dependent variable is y which is drawn on the y-axis.
- Black dots represent the real data points.
- $b_o$ represents the intercept that is 10 and $b_1$ constitutes the slope of the x variable.
- The best fit line identified by the model is the blue line which means the predicted values exist on the blue line.

## Assumptions of Linear Regression model

1. Linearity

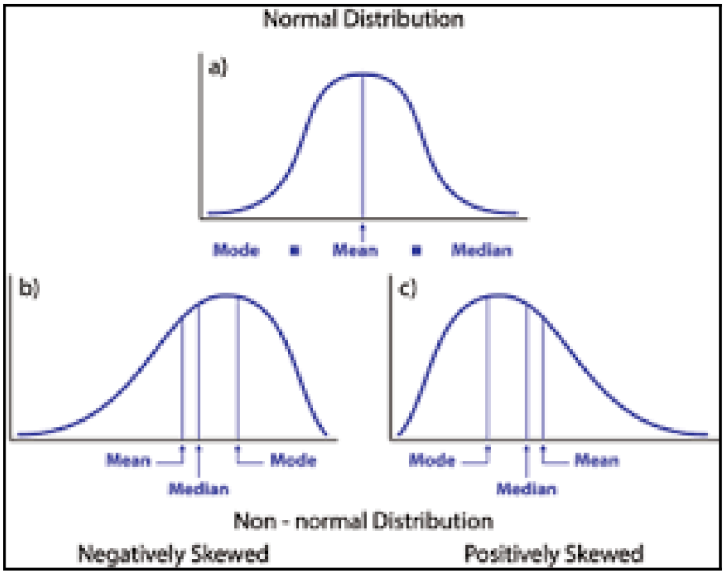
   It asserts that the dependent variable Y must be linearly associated with independent variables. Linearity assumption can be verified by drawing a scatter plot between dependent

and both independent.



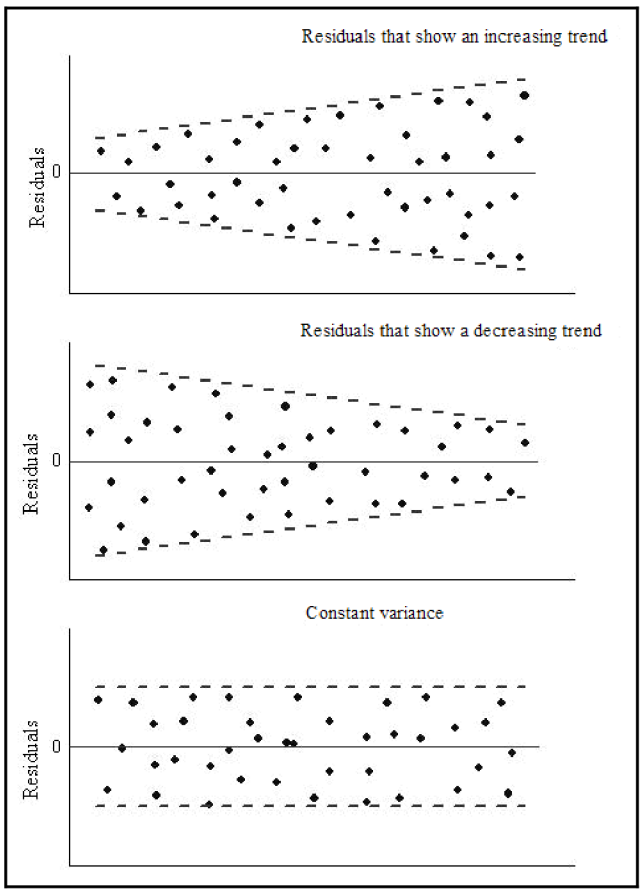

2. Normality

   Two known X and Y parameters must be distributed normally. Plots like the Histograms, KDE plots, Q-Q plots must be utilised properly to verify the Normality assumption.




3. Heteroscedasticity

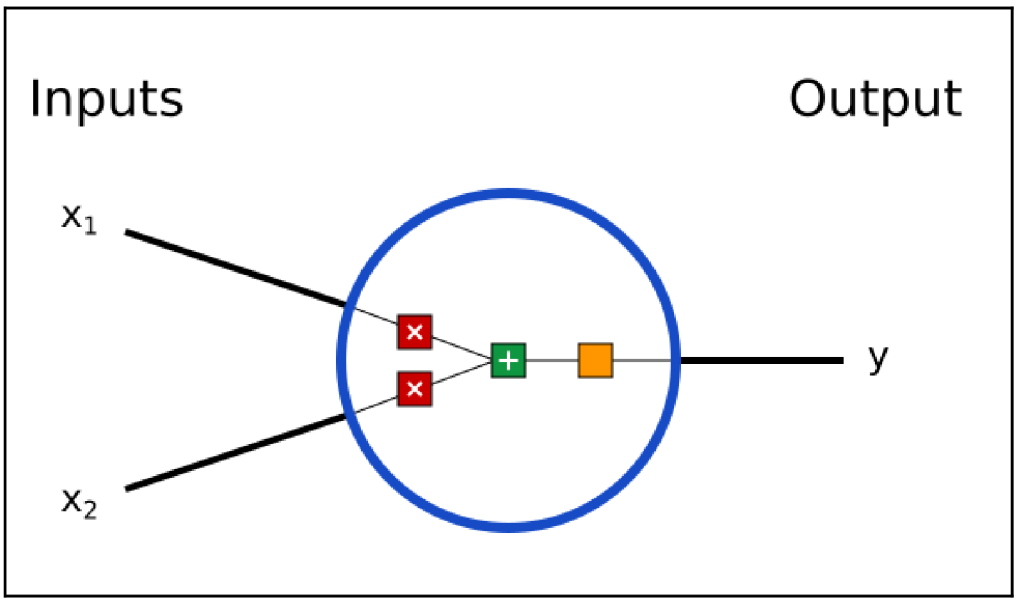
   The difference in the error terms must be constant i.e the spread of residuals should be constant for all values of X. This particular assumption can be identified by drawing a residual plot. In case this assumption is infringed then the points form themselves into funnel shape or they will remain constant.




### 3.5.3 Neural Networks

#### Building Blocks: Neurons

Before we even speak about Neural-networks, we will first have to comprehend about concept of neurons, the rudimentary axiom of a neural network. The every single neuron consumes what is loosely called inputs, does some sort of calculations in between, and generates an output. Below is a figure of two input neurons




There are three things happening in this scenario. First, neurons are multiplied by what is known as a pre defined weight:

$$x_1 \rightarrow x_1 * w_1$$

$$x_2 \rightarrow x_2 * w_2$$

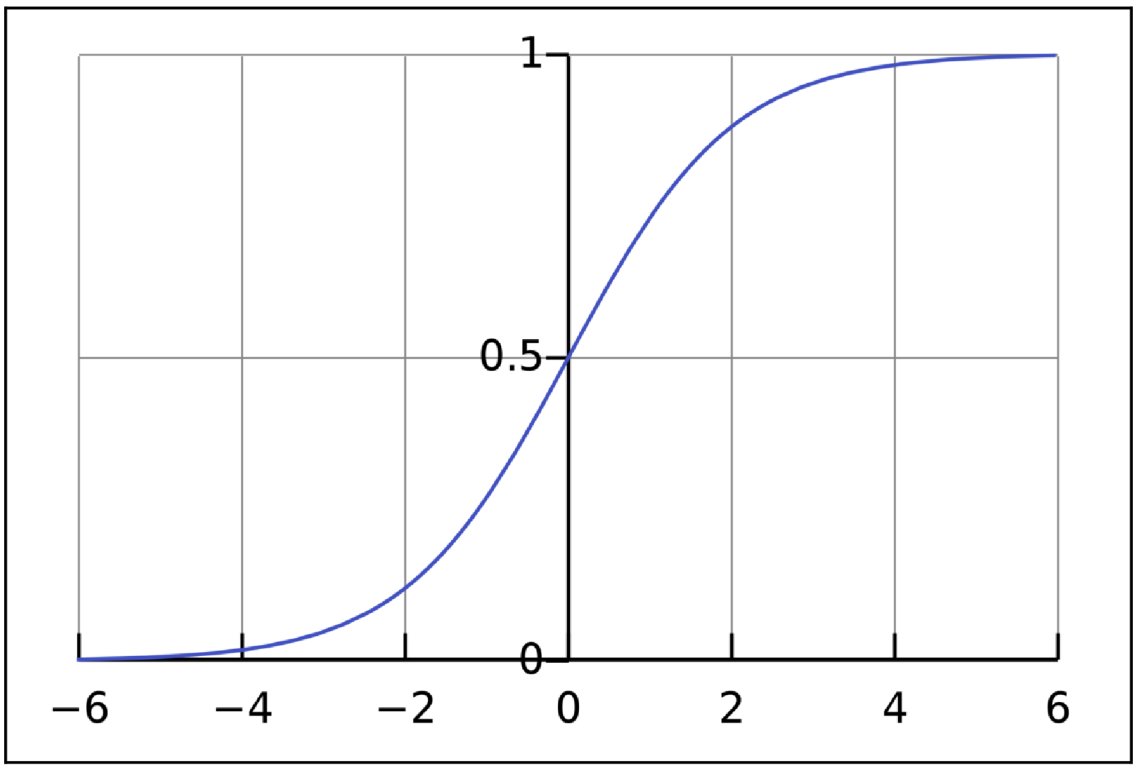
And next, inputs that weighted are appended all together with variable known as bias b:

$$(x_1 * w_1) + (x_2 * w_2) + b$$

Ito conclude, the overall calculated sum is travel's through function called an activation function:

$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$

The unbounded input is turned into an output that has a well rounded and predictable form with the help of an activation function. The most popular activation function that is used is sigmoid function:




The sigmoid function generates some sort of outputs in the positive numbers realm ranging from 0 to 1. We can comprehend it as squashing (−∞,+∞) to (0,1) — large size negative numbers turn ~0, and large size positive numbers turn ~1.

A Simple illustration:

Let us consider we have with yourself a two-input neuron that make use of sigmoid activation function and has the following specification:

$$w = [0, 1]$$

$$b = 4$$

w=[0, 1] translates or gets converted into w1=0, w2=1 in the version that will be understood by a vector. Let us consider that the neuron has the following input of x=[2, 3] which means w1=2, w2 =3 We will make use of dot product to make things more simple:

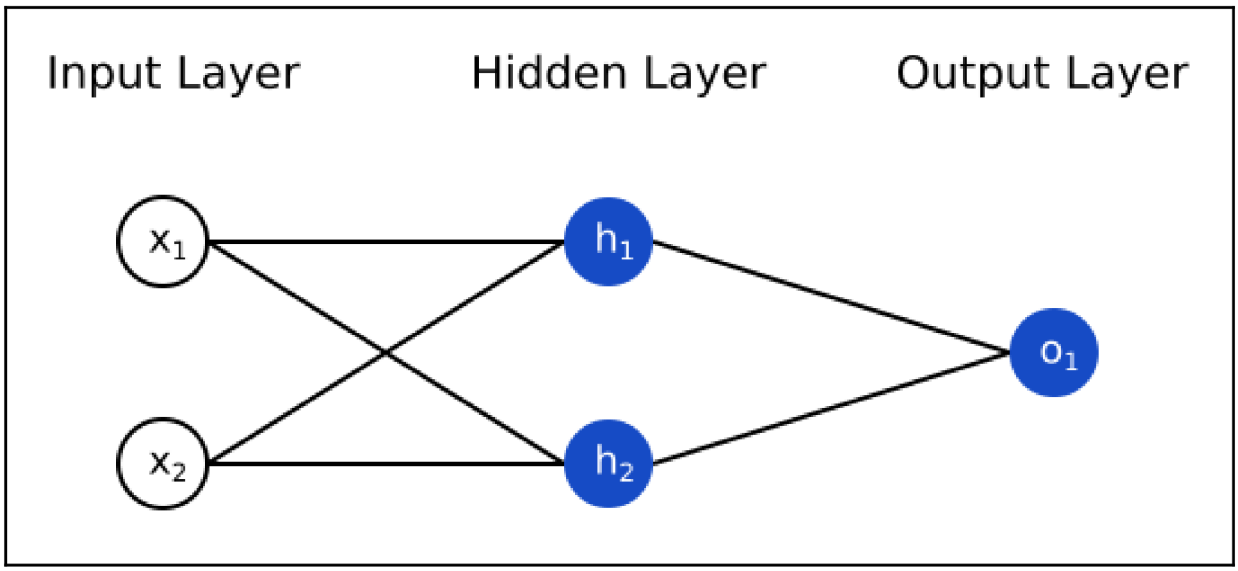
$$
\begin{aligned}
(w \cdot x) + b &= ((w_1 * x_1) + (w_2 * x_2)) + b \\
&= 0 * 2 + 1 * 3 + 4 \\
&= 7
\end{aligned}
$$

$$
y = f(w \cdot x + b) = f(7) = \boxed{0.999}
$$

The output of the neurons generated by the function is 0.999 for the inputs weights w1=2, w2=3. That is it! This whole process of throwing inputs forward to generate an output is called feed-forward.

### Combining Neurons into a Neural Network

A simple  neural network is nothing but a bunch of neurons cascaded all together. Diagram below is is a basic neural network




This above mentioned network has two available inputs one called as a hidden layer that has two neurons one of them is $h1$ and the other $h2$, and an output layer with 1 neuron ($o1$). Notice that the inputs for $o1$ are the outputs from $h1$ and $h2$ — that's what makes this a network.

A hidden layer is sandwiched between the first input layer and last output layer. Multiple hidden layers can be more than one

Illustration of Feed-forward

Let's consider the network in the diagram above and let us assume all the neurons have the default weights $w$=[0,1] and bias $b$=0, and the similar sigmoid activation function. Parameters $h1$, $h2$, $o1$ indicate the *outputs* of the neurons they depict.

What will happen if we pass in the input $x$=[2, 3]?

$$\begin{aligned} h_1 = h_2 &= f(w \cdot x + b) \\ &= f((0 * 2) + (1 * 3) + 0) \\ &= f(3) \\ &= 0.9526 \end{aligned}$$

$$\begin{aligned} o_1 &= f(w \cdot [h_1, h_2] + b) \\ &= f((0 * h_1) + (1 * h_2) + 0) \\ &= f(0.9526) \\ &= \boxed{0.7216} \end{aligned}$$

The outcome of the neural network for input $x1$ =2, x2= 3 is 0.7216. easy, is it not?

A neural network can come with any more than one layer and can accommodate any number of neurons in those layers. The strategy remains the same: throw some input(s) forward via the neurons in the given network to obtain some output(s).

### 3.5.4 Decision Tree

Decision Tree belongs to the Supervised Machine Learning Algorithm that makes use of a set of rules in order to make proper decisions, in a manner to how humans envision the decision.

This particular *model foresees* the class of the new, never known input but, but in reality behind the scenes, the algorithm has to *come up with* which class to designate.

We humans make use of rule-based decisions every time.

Whenever we plan our vacation, we make use of a rule-based approach. We may pick a different destination based on the duration of the vacation, also based on the budget allocated to vacation or if our friends and relatives are coming along.

The answer to all the above questions is the deciding factor for the final decision. And if you continually narrow down the available vacation destinations based on how you answer each question, you can visualise this decision process as a (decision) tree.

## A tree that makes decisions

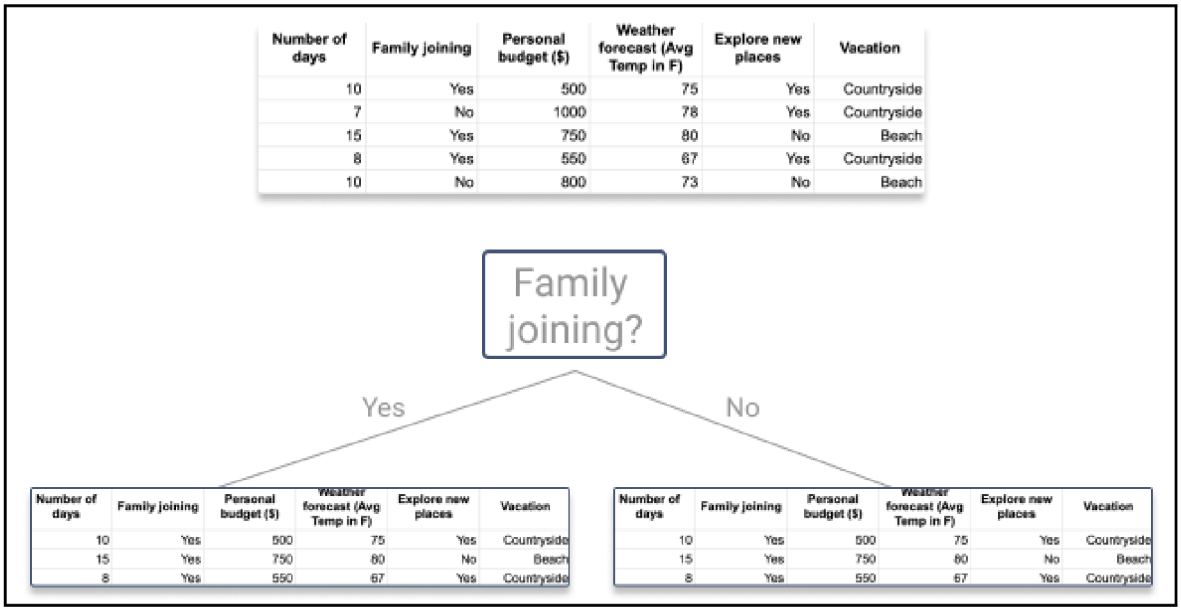
Decision trees can render both classification and regression tasks, so you'll see authors refer to them as the CART algorithm: Classification and Regression Tree. This term is universal, pretty much applies to all the available tree-based machine learning algorithms,.

But let's focus on decision trees for classification.

The intuition behind Decision Trees is that you use the dataset features to create *yes/no* questions and continually split the dataset until you isolate all data points belonging to each class.

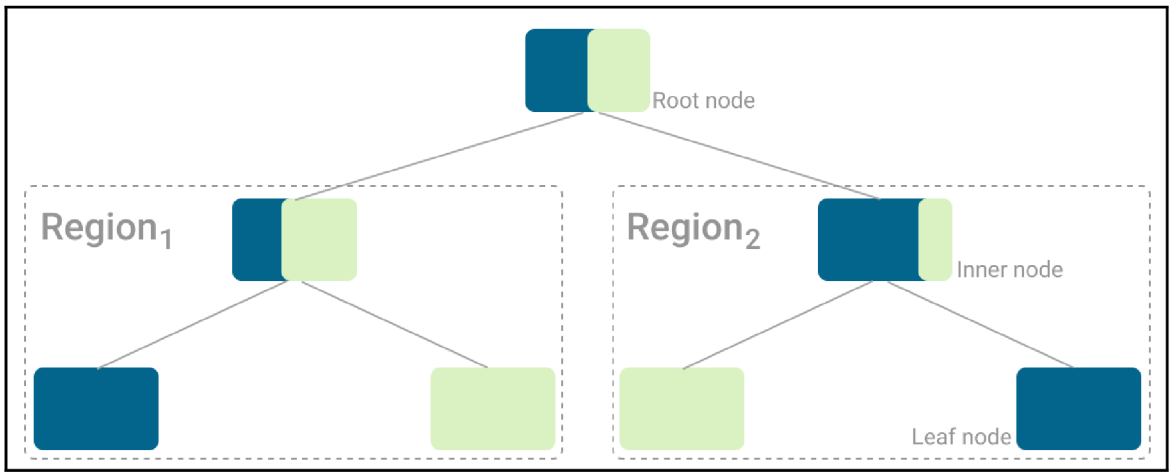
With this overall steps we construct the data into a tree based structure.

- Every time you *ask a question* you're adding a node to the tree. The very first node is popularly known as the root node.
- The result of *asking a question* splits the dataset based on the value of a feature, and creates new nodes.
- Whenever we decide to halt the process after the chop, the renaming nodes that gets created is called leaf nodes.



A simple illustration of a decision tree with lets say with tree nodes, the first node being root node and the reaming two leaf nodes.

Each time we begin to answer a basic question, we also construct the branches and segmenting and the feature space something into a region called disjoint.




Example of the different regions and types of nodes in a tree. (Image by author)

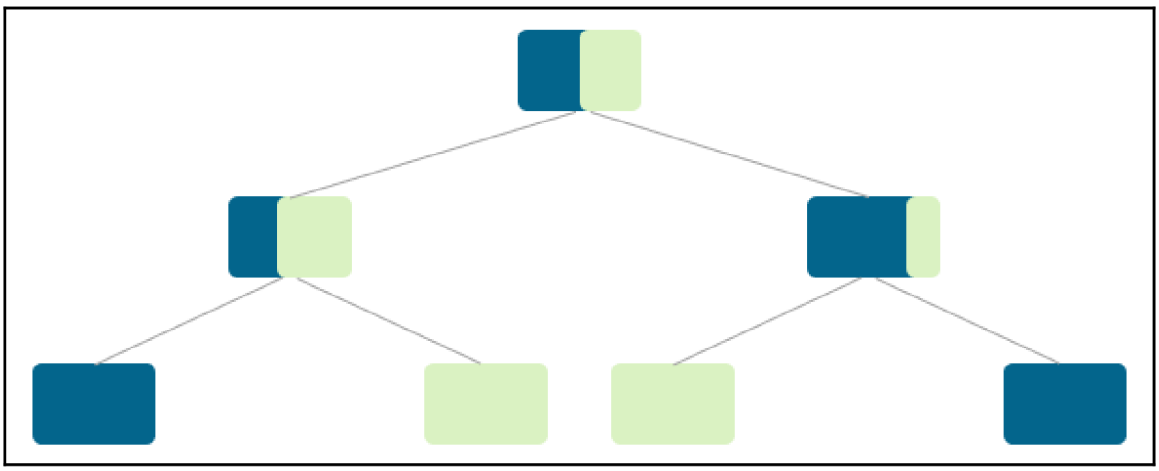
In this current way we reduce the feature space with every chop or branch created in the tree, and every data point will associate itself to each region.

The goal is to continue splitting the feature space, and applying rules, until you don't have any more rules to apply or no data points left.

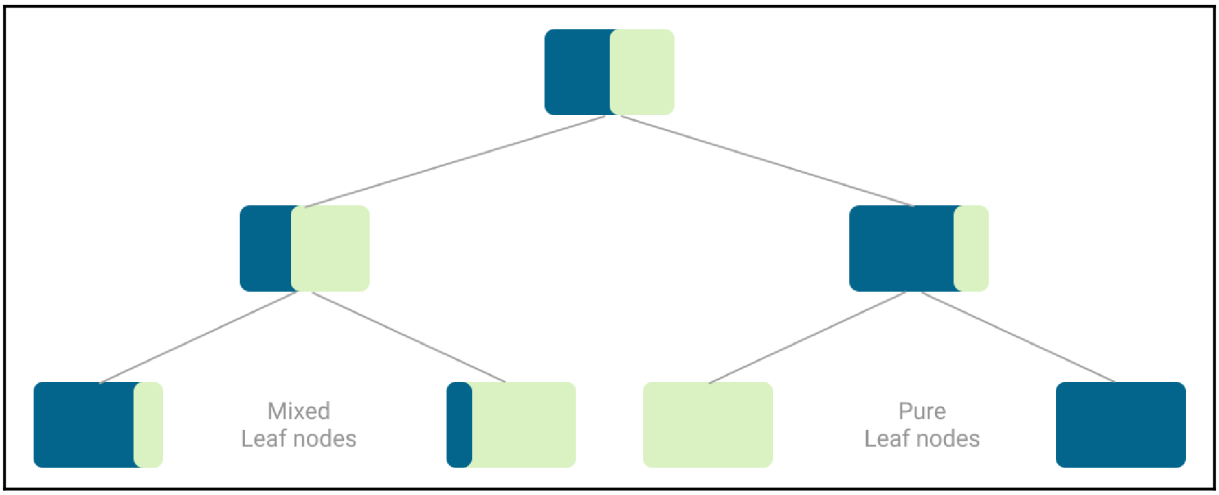
After that, it's time to associate a class to each and every data points to every leaf node.

Assigning a class

This current artificial intelligence algorithm will try to absolutely segregate the original dataset such that every other leaf nodes, that is, the all the nodes that did not split the data further down the line, that associate itself to a single class. These are called pure leaf nodes.



But most times you end up with mixed leaf nodes, where not all data points have the same class.




Example of a tree with pure and mixed leaf nodes, before assigning the final classes to each node. (Image by author)

With pure leaf nodes that are already taken care of, because all data points in that node have the same class.

### 3.5.5 Cross validation

If we realise about machine learning we already know that during the construction of machine learning edifice, we require data at our disposal in order to train it. How do we achieve that? We chop of the whole data set in two asymmetrical sets. Let us just say eight percent of the data as a training set and the rest twenty percent as a test data set.. Henceforth, we will train the opted machine learning algorithm model on the training set and in order to estimate the model actual performance we try out the performance of the model on the obscured test data that we have it

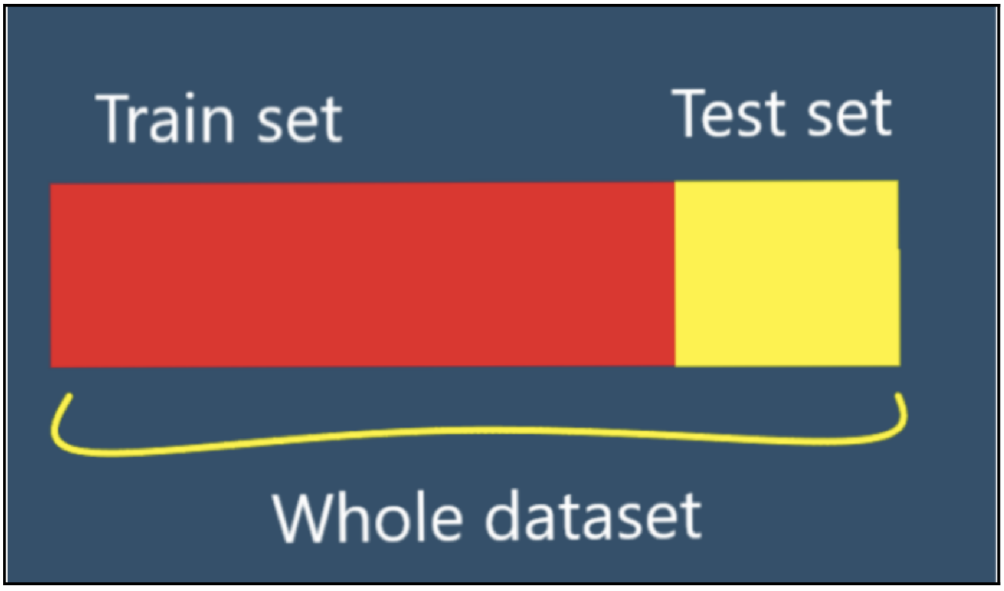
around the corner for further verification. Cross validation is a way for calculating the machine learning model and testing its performance.. It is a super simple model and very easy to implement.

Different types of cross validation techniques

There are a plethora of cross validation(CV) techniques that exists..But only few of them are used in the machine learning ecosystem, others cross validation(CV) techniques just exist. Let's discuss all of them:

- Hold-out cross validation(CV)
- K-fold CV
- Leave-p-out cross validation(CV)
- Stratified k-folds
- Repeated k-folds
- Nested k-folds
- Time series CV
- Validation set approach.

**Hold-out**

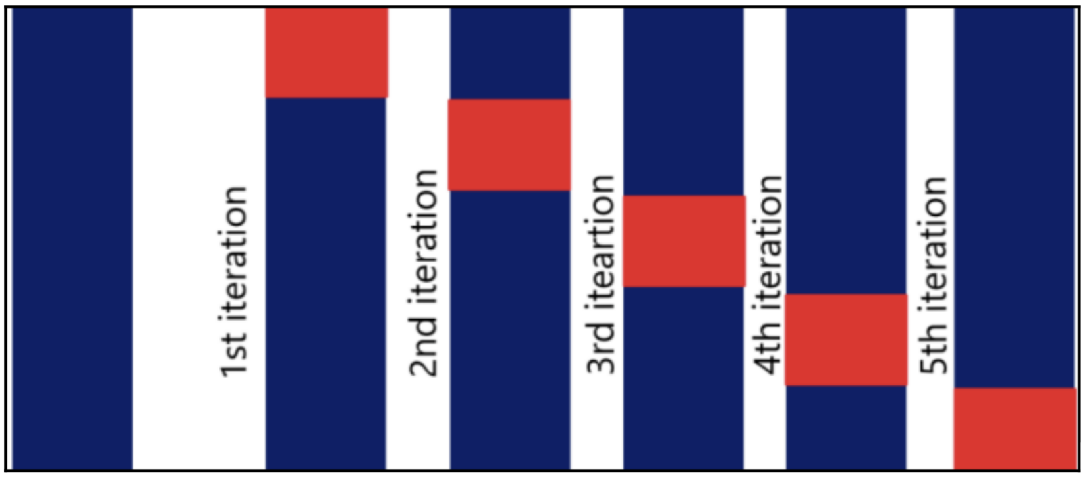



1. Chop the whole data set into two different parts. Basically eighty percent of the data goes towards the training set and the remaining twenty percent move towards the testing set. This way of splitting is not universal.

2. Train the machine learning cross validation model on training set
3. The model should be validated on the test data set.
4. Calculate the efficiency/performance of the cross validation model
5. Complete!

It is so rudimentary that it looks super trivial. It is especially handy when we have a big amount of data samples and it is necessary to train the model only once.

**K-Fold cross validation**




K-fold is one of the most commonly used cross validation techniques.

In this current technique we split the complete data set into k number groups of identical proportions. These are called Folds.

The technique works the following way:

1. Pick the value for the variable ‘k’ on your own. Ideally speaking it is assumed to be in the values that could be 5 or 10.
2. Divide the complete dataset into k numbers of equal parts.
3. Pick one test data set and the remaining as k-1 folds as a training data set.
4. With the help of k-1 training sets, train the model.
5. Validation is performed using a test set.
6. Estimate the performance and make sure to save the result.

7. Make sure to repeat the steps 3rd to 5th with a whole new k fold as the test dataset and the remaining as training dataset.
8. Evaluate the arithmetic mean for the sake of performance.

When we train and test the models on various parts of the data set it will result in a lot more stable edifice. If we need to significantly elevate performance of the ML model all we need to to do is increase the value of k.

The parameter K can be picked up in any number of manners.

- Pick the k in such a manner that both the train dataset and test dataset is big enough that they statistically delineate the main data set.
- K = 10. As per many experiments it is a good choice since it results in the ML model with lowest bias and little to no variance.
- K = n.

One cons of the ML model k-fold Cross Validation is that when the parameter value k is increased the computational cost is extremely high and it has greater time complexity as we are training many models (k).

### Leave p-out cross validation model

In the present approach we pick p number of samples and convert it into a test dataset and remaining samples (n-p) into a training set. We can pick the number of samples (p) from n samples in nCp ways. In contrast to k-fold cross validation the test sets can overlap.

1. Pick p number of samples and this is our test data set
2. Pick the remaining of the samples (n-p) as our training dataset.
3. Model is trained with the help of a training set. Be sure that with every iteration a new model should be trained
4. Test set is used to validate the model.
5. Result must be saved.
6. Rinse and Repeat from step-3 to step-5 a number of nCp times.
7. Average the results

The problem with this technique is that for larger number p it can be very expensive to compute the model.

### Stratified k-fold

When dealing with imbalanced or asymmetrical data such as a regression problem the data we have at our disposal is the price of a similar type of product. In the case when the price of some items is high and the remaining items are very low, such low price items are greater in number than the costly items. Let's take the case of binary classification where samples of one particular class a pure case of asymmetrical where are higher than the other class. Let's consider a dataset of 2000 samples about lions and tigers. There are 1400 lions and 600 tigers. When this kind of asymmetrical dataset the k-fold CV technique might have some problems. In order to solve this issue they have come up with a new version/variant of k- fold technique CV.

### Stratified k-fold Cross Validation.

It renders almost in similar way as that k-fold Cross Validation operates. The main distinction is that the Stratified k-folds chop the data set in a way that every fold has the identical percentages of samples of every target like in the original data set. For the case of regression, Stratified k-fold makes sure that the mean target results are more or less equal in all folds.

Stratified k-fold deals with bias and variance well.

### Repeated k-folds

In this technique the parameter k is not the number of folds like in the standard k-fold technique. K represents the total number of times we will train the model.

Let's assume that we chose twenty percent of the data to be our test dataset and remaining are the training dataset. Now the technique randomly selects twenty percent of the data from the main dataset and trains the model with the remaining of the samples and validates the model on the twenty percent samples that it took aside. It does this k times and averages the result

Advantage of this method is that it is more robust since it chooses the train and test set at random. Disadvantage is that there is no guarantee that all the samples will be selected at least once for testing.

### Nested k-fold

This is where k-fold cross-validation technique is rendered with every fold of each cross-validation, frequently to render hyper parameter fine-tuning during the processes of model estimation.

### Time series CV

This is a separate discussion in itself. Let's consider a scenario where we have a sequential data like time series one. In a scenario like this we should not randomly data point to either train dataset or test dataset.

### Validation set approach

In this way of approach we will segregate the main data set into two equal symmetrical parts training dataset and test data set or also known as validation set.

Main cons of this current approach is that only fifty percent data will be trained as per the model. So, what might actually occur is that the model might fail to detect numerous patterns which will be in the original/main dataset but will not be present in the training data set.

# 4 Practical Part

## 4.1 Implementation

### 4.1.1 Data Sets

Qualitative Bankruptcy Database

**Information about the data sour**

- Created by Martin. A Uthayakumar.

**Usage in the past**

- The attributes or parameters which we used for collecting the dataset is referred from the paper

**Cardinality of the data**

- Number of Instances: 250
- Number of Attributes: 6, each corresponding to Qualitative Parameters in Bankruptcy

**Attribute-information**

(P=Positive, A-Average, N-negative, B-Bankruptcy, NB-Non-Bankruptcy)

- Industrial Risk: {P, A, N}
- Management Risk: {P, A, N}
- Financial Flexibility: {P, A, N}
- Credibility: {P, A, N}
- Competitiveness: {P, A, N}
- Operating Risk: {P, A, N}
- Class: {B, NB}

**Internal Risks**

- Industry risk (IR):
- Government policies and International agreements
- Cyclicality
- Degree of competition
- The size and growth of market demand
- The ability to alterations in macroeconomic factors
- Domestic and international competitive power
- Product Life Cycle.

**Management risk(MR)**

- Ability and competence of management
- Stability of management
- The relationship between management/ owner
- Human resources management

- Growth process/business performance
- achievement and feasibility

**Financial Flexibility(FF):**

- Direct financing
- Indirect financing
- Other financing

**Credibility (CR)**

- Credit history
- Reliability of information
- The relationship with financial institutes

**Competitiveness (CO)**

- Market position
- Level of core capacities
- Differentiated strategy

**Operating Risk (OP)**

- The stability and diversity of procurement
- The stability of transaction
- The efficiency of production
- The prospects for demand for product and service
- Sales diversification
- Sales price and settlement condition,
- Collection of A/R
- Effectiveness of the sales network.

**Missing Parameters and Class distribution**

- Vales of the parameters missing: None

- Class Distribution information: [there are (one forty three)143 samples For Non-Bankruptcy] [(Hundred and seven)107 s For Bankruptcy]

### 4.1.2 Information about the dataset

```
CLASS-TYPE: case of nominal
@relation bankrupt_qualitative
@attribute IR {P,A,N}
@attribute MR {P,A,N}
@attribute FF {P,A,N}
@attribute CR {P,A,N}
@attribute CO {P,A,N}
@attribute OP {P,A,N}
@attribute Class {B,NB}
@data

DATA SAMPLE:
P, P, A, A, A, P, NB
N, N, A, A, A, N, NB
A, A, A, A, A, A, NB
P, P, P, P, P, P, NB
N, N, P, P, P, N, NB
A, A, P, P, P, A, NB
P, P, A, P, P, P, NB
P, P, P, A, A, P, NB
P, P, A, P, A, P, NB
P, P, A, A, P, P, NB
P, P, P, P, A, P, NB
P, P, P, A, P, P, NB
N, N, A, P, P, N, NB
N, N, P, A, A, N, NB
```

### 4.1.3 Setting up the dataset for ML training

#### Step 1: Obtain the the Dataset

We now should create python module.

Let's name it main.py and import all the necessary libraries.

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn import tree
from sklearn import datasets
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
```

We are using Python-3. When we run the main.py with the following code the return code should be zero.

## Step 2: Managing the missing data

Missing data is a common occurrence in datasets. When we are dealing with real-time datas, missing data is quite frequent.

In order to train the data in an orderly fashion, we should do something about this missing data. Or, the AI model can misinterpret this missing information.

In order to overcome this problem we have libraries that are easily available.

We should also segregate datasets into two parts  X and Y-axis.

Okay, now write the following code after importing the libraries.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
data = pd.read_csv("data.csv")
```

We have these two methods replaces

1. rep_class

2. rep_column

Their fundamental function is to go though each value of calss and column and replace the missing values.

```
def rep_class(x):
    if x == 'NB':
        return 0
    else:
        return 1
def rep_column(x):
    if x == 'P':
        return float(1)
    elif x == 'A':
        return float(2)
    else:
        return float(3)
```

When should run the main.py again with the return code being zero. Okay, so we have included our initial dataset, and you can see it here.

## Step 3: Transform the dataset into Training and  Testing Sets

We should cut the dataset into ratio of 80%  Training Data and 20% into test data. So, for our case, we should divided the dataset into 80% for training data and 20% for test data.

```
# Split the data between the Training Data and Test Data
#Perform data split 80% tranin and 20% test data
[X_train,X_test,y_train , y_test] = train_test_split(X, y, test_size=0.2, random_state=0)
#Perform data split 90% tranin and 10% test data
```

```
[X_train,X_test,y_train , y_test] = train_test_split(X, y, test_size=0.2, random_state=0)
X,y = datasets.load_iris(return_X_y=True)
```

Now we have successfully prepared Dataset For Machine Learning in Python.

### 4.1.4 Code in Python

These are the library imports that are used by the application. The main libraries that are used are scikit-learn and Pandas.

```
import pandas as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
import numpy as np
from sklearn.svm import SVC
from sklearn import tree
from sklearn import datasets
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
```

We read the data set and drop the unnecessary columns.

```
data = pd.read_csv("data.csv")
X = np.array(data.drop(["Class"],axis=1))
y = np.array(data["Class"])
```

We use the rep method to create the necessary values in Class column

```
def rep(x):
    if x == 'NB':
        return 0
else:
        return 1
```

We use the rep_column method to replace the necessary values in remaining column

```
def rep_column(x):
    if x == 'P':
        return float(1)
    elif x == 'A':
        return float(2)
     else:
         return float(3)

data["Class"] = data["Class"].map(rep)
data["Industrial Risk"] = data["Industrial Risk"].map(rep_column)
data["Industrial Risk.1"] = data["Industrial Risk.1"].map(rep_column)
data[" Financial Flexibility"] = data[" Financial Flexibility"].map(rep_column)
data["Credibility"] = data["Credibility"].map(rep_column)
data["Competitiveness"] = data["Competitiveness"].map(rep_column)
data["Operating Risk"] = data["Operating Risk"].map(rep_column)
```

With the help of this line we convert unparsed csv to a parsed csv which is a new line.

```
data.to_csv("data_new.csv",index=None,header=True)
```

Perform data split 80% data split and 20% test

```
[X_train,X_test,y_train , y_test] = train_test_split(X, y, test_size=0.2, random_state=0)
```

We now Build Support Vector machine Classifiers using the following code.

```
Classifier = SVC(kernel='linear')
Now we train our model
model = Classifier.fit(X_train,y_train)
```

## Training the model

With scikit-learn, the estimator is an instance that fits a model depending on the type of the input data (i.e. training data) and renders specific estimations that leads to properties on newly created unseen data. In so many words, an estimator can act as an a regressor or also a classifier.

The frame work comes with the base class *BaseEstimator* located at sklearn.base.BaseEstimator and a the remaining estimators must inherit from that specific class to do further calculations.When in packaged conditions the base class is shipped with two methods, called *get_params()* and *set_params()* as the method suggests helps to retrieve and set the parameters of an estimator respectively.

**Fit method signature**

***fit(X, y, sample_weight=None):***

We will fit it the Support Vector model based on the given training dataset.

X — Is a Training vectors, where n_samples denotes total number of samples and n_features is denotes total no. of features.

y — Are Target values .

sample_weight — indicates Per-sample weights.

Note:

The estimators must explicitly pass all of the necessasry parameters in the constructor method (i.e. the *__init__* method) that will be used in the initialization.

The *fit()* method is used by almost all the estimators and it takes an input for the sample data (X) and for supervised models it also accepts an argument for labels (that is. target data y ). Additionaly as an available option, it takes in additional parameters such as like weights etc.

Generally speaking the fit methods are the reasons behind several operations/functions. Normally, they must eliminate attributes that have already stored inside the estimator object and then perform necessary validations. Attributes with in the input data are also eliminated by the fit() method and also they store the parameters and attributes of the particular model and return value is estimator.

Let us take a current situation a classification problem where Support vector Machine model will help to recognise occurrence of Bankruptcy. In the code below, we first load our data and then split it into training and testing sets.

### Predicting the model

Since now the training of the model is complete, the following step is performing predictions with the remaining testing set. In order to accomplish this job, we need to pass the parameters to predict method *predict()* that will perform the predictions with the help of learned parameters by *fit()* on the new data points and unseen test data points.

Typically, the *predict()* method will render a predictions for each available test instance and it typically takes only a single input (X).

```
Y_PRD = model.predict(X_test)
print("Predicions of SVC: " , Y_PRD)
```

### Get the accuracy of the model

```
accu = model.score(X_train,y_train)
print("Accuracy of SVC: " , accu)
```

### Cross Validation

When fine tuning the various models our goal is to increase elevate the efficiency and performance that data that cannot be seen. The performance of the tests sets can be increased significantly by Hyper-parameter tuning. But enhancing the parameters to the test set could cause exhalation of the information causing for the model to underperform. To fix this particular issue, we can do cross validation. Also we use cross validation in this particular scenario because we have a smaller dataset

```
score_cv = cross_val_score(model,X,y , cv=5)
print("%0.2f accuracy with a standard deviation of %0.2f" % (score_cv.mean(),
score_cv.std()))
print("Scores of CV: " ,score_cv)
print("Training accuracy")
```

### Get the accuracy of the train data

```
print(accuracy_score(y_train, model.predict(X_train)))
print("testing accuracy")
predictions = model.predict(X_test)
```

### Get the accuracy of the test data

```
print(accuracy_score(y_test, predictions))
```

## Confusion matrix

It is an actually a table that is particularly used in classification related problems to estimate where errors in the model were made.

The rows indicates the total number of actual classes the outcomes should have been. While the columns represent the predictions we have made. Using this table it is easy to see which predictions are wrong.

True indicates that the values were accurately predicted, False means the prediction that were erroneous.

We can calculate Confusion Matrix by running the following code, we can calculate different measures to quantify the quality of the model. First, let's look at Accuracy.

```
cm = metrics.confusion_matrix(y_test, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display_labels =
[False, True])
cm_display.plot()
plt.show()
print(cm)
```

## Build Logistic Regression

```
Classifier = LogisticRegression(solver='liblinear')
```

## Training our model

```
model = Classifier.fit(X_train,y_train)
```

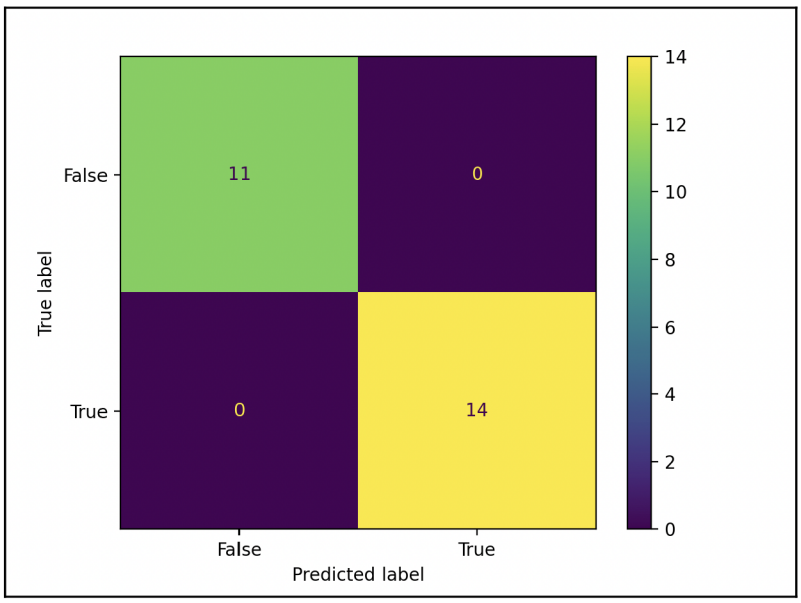
## Getting the accuracy of the model

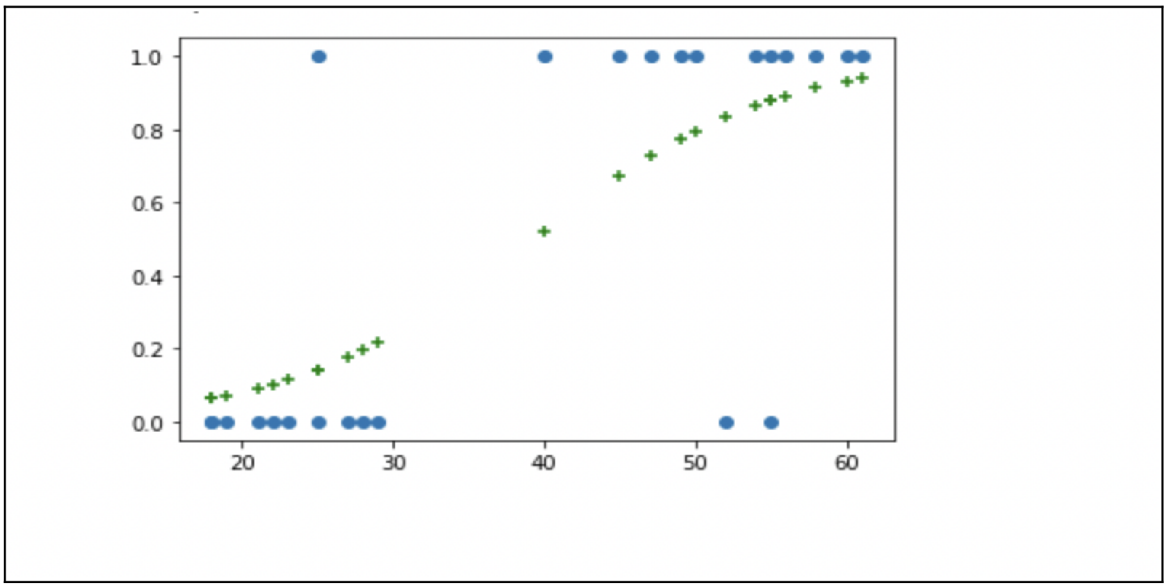
```
accu = model.score(X_train,y_train)
print("Accuracy of Logistic Regression: " , accu)
score_cv = cross_val_score(model,X,y , cv=5)

print("Scores of CV: " ,score_cv)
print("Training accuracy")
print(accuracy_score(y_train, model.predict(X_train)))
print("testing accuracy")
predictions = model.predict(X_test)
print(accuracy_score(y_test, predictions))

cm = metrics.confusion_matrix(y_test, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display_labels =
[False, True])
cm_display.plot()
plt.show()
print("Confusion matrix of Logistic Regression”)
print(cm)
```

## Accuracy Diagram




```
plt.scatter(X_train,X_train)
plt.plot(X_train,model.predict_proba(X_train[:1]))
```

## Build Neural Network

```
Classifier = MLPClassifier(max_iter=1000)
Now we train our model
model = Classifier.fit(X_train,y_train)
print("%0.2f accuracy with a standard deviation of %0.2f" % (score_cv.mean(),
score_cv.std()))
Get the accuracy of the model
accu = model.score(X_train,y_train)
```

## Cross Validation

```
print("Accuracy of Neural Network: ",accu)
```

```
score_cv = cross_val_score(model,X,y , cv=5)
print("Scores of CV: " ,score_cv)
print("Training accuracy")
```

## Get the accuracy of the train data

```
print(accuracy_score(y_train, model.predict(X_train)))
print("testing accuracy")
```

## Get the accuracy of the test data

```
predictions = model.predict(X_test)
print(accuracy_score(y_test, predictions))
print("%0.2f accuracy with a standard deviation of %0.2f" % (score_cv.mean(),
score_cv.std()))
```

## Calculate confusion matrix

```
predictions = model.predict(X_test)
cm = metrics.confusion_matrix(y_test, predictions)
print("Confusion matrix of Neural Network”)
print(cm)
```

## Build Decision tree

```
Classifier = tree.DecisionTreeClassifier()
model = Classifier.fit(X_train,y_train)
accu = model.score(X_train,y_train)
print()
print("Accuracy of Decision tree: ",accu)
score_cv = cross_val_score(model,X,y , cv=5)
print("%0.2f accuracy with a standard deviation of %0.2f" % (score_cv.mean(),
score_cv.std()))
```

## Cross Validation

```
print("Scores of CV: " ,score_cv)
print("Training accuracy")
print(accuracy_score(y_train, model.predict(X_train)))
print("testing accuracy")
predictions = model.predict(X_test)
print(accuracy_score(y_test, predictions))
```

## Calculate confusion matrix

```
predictions = model.predict(X_test)
print("Confusion matrix of decision tree: ", predictions)
cm = metrics.confusion_matrix(y_test, predictions)
print(cm)
```

# 5 Results

## 5.1 Overview

These are the results that are obtained from running the python script 'main.py'. The application does several things. It parses the csv file, and gets rid of unwarranted stuff from the csv file. Replaces the class variable with 0's and 1's and creates a new csv file. There are four artificial algorithms running

- Support vector machine
- Logistic regression
- Neural Network
- Decision tree.

For all the above mentioned algorithms. It does the following things.

- It obtains the algorithm object
- Trains the model.
- Calculates accuracy of the model.
- Estimates the accuracy of the test set.
- Estimates the accuracy of the train set.

## 5.2 Application outputs

### 5.2.1 Support vector machine

```
Accuracy of SVC:  0.975
0.98 accuracy with a standard deviation of 0.02
Scores of CV:  [0.96666667 1.         0.96666667 0.96666667 1.        ]
Training  accuracy:
0.975
Testing accuracy
1.0
Confusion matrix of SVC
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

### 5.2.2 Logistic regression

```
Accuracy of Logistic Regression:  0.9333333333333333
0.96 accuracy with a standard deviation of 0.04
Scores of CV:  [1.         0.96666667 0.93333333 0.9        1.        ]
Training accuracy
0.9333333333333333
testing accuracy
0.9666666666666667

Confusion matrix of Logistic Regression
[[11  0]
 [ 0 14]]
```

### 5.2.3 Neural Network

```
Accuracy of Neural Network:  0.9833333333333333
0.98 accuracy with a standard deviation of 0.03
Scores of CV:  [1.         1.         0.96666667 0.93333333 1.        ]
Training accuracy
0.9833333333333333
testing accuracy
1.0

Confusion matrix of Neural Network
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

### 5.2.4 Decision tree

```
Accuracy of Decision tree:  1.0
0.97 accuracy with a standard deviation of 0.04
Scores of CV:  [0.96666667 0.96666667 0.9        1.         1.        ]
Training accuracy
1.0
testing accuracy
1.0

Confusion matrix of Decision tree
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

# 6 Conclusion

The objective of the whole diploma thesis is to bring out the best Artificial algorithm that solves the problem that we have at hand. The necessary research was done in order to gather the data. The data was retrieved from the University of California ,Irvine official website (https://archive.ics.uci.edu/ml/datasets/qualitative_bankruptcy). The title of the dataset is qualitative bankruptcy database. The variables that were used for feeding the information into dataset is is addressed in the paper "The discovery of experts decision related rules from qualitative bankruptcy data with genetic algorithms from the ceeator" by Myoung-Jong Kim*, Ingoo Han.

Let's take a case where gathering and efficiently processing a colossal number of data is not readily possible. This can be because of issues such as data regulations like privacy concerns and safety issues. It could be possible it may not be possible to acquire enough data right on time. When such problem is in front of us, it's not possible to acquire the data for machine learning

This also does not denote that fruitful machine learning is not going to achievable. In retrospect, models can adapt and learn successfully from little datasets.

It's is a great concept to learn that we require less data for machine learning related applications, at the same time we need to be cognizant of exactly how much the data is required for our own specific situation. Hence, every machine learning project should render its own ground for Proof of Concept development to witness weather the correct data is being gathered for calculating precise results.

Perhaps it might shock us at how many little fragments of data are required. For instance, based on research we infer in case of image classification problem as little as four samples per individual class are required to hit accuracy of 70%. Another objective of the thesis is to prove the we need little samples to reach desired accuracy to solve the bankruptcy problem

This particular research was conducted with the help of images of coins. Our own case may perhaps necessitates more or little data in the data sets.

## 6.1 Cross Validation

As witnessed above, the answer to the million dollar question: "What is the amount of data that is required?" relies on various factors like the wide-range of diversities of production data, the

existence of open-source datasets,  the performance is it anticipated by the system. When we have a limited dataset, the algorithms performance should be evaluated along with cross validation calculations. While fine tuning the all the models models our objective is to elevate elevate the efficiency and performance that data that cannot be seen or many times goes missing. The performance of the data sets  can be increased momently by Hyper-parameter tuning. To fix this issue we use cross-validation estimates.

The dataset that was gathered was parsed using the following steps:

- Feed the data to the following algorithms that would make the predictions
    1. Support Vector machine
    2. Logistic regression
    3. Neural Network
    4. Decision tree
- Evaluate the following metrics from the model
    1. Accuracy of the model
    2. Accuracy of the model with Cross Validation(CV)
    3. Accuracy of the test set
    4. Accuracy of the train set

At this point we have evaluated the performance of the model and estimated the accuracy of the testing set and training set. Cross validation accuracy for all the four models have been estimated.

## 6.2 Tabulated Results

| Accuracy/AI models | SVM | Logistic regression | Neural Network | Decision tree |
|---|---|---|---|---|
| Acc. of model | 0.975 | 0.975 | 0.983 | 1.00 |
| Acc. with Cross Validation | 0.980 | 0.960 | 0.980 | 0.97 |
| Acc. of train set | 0.975 | 0.933 | 0.983 | 1.00 |
| Acc. of test set | 1.000 | 0.966 | 1.000 | 1.00 |

## 6.3 Tabulated Results with Cross-Validation

| Accuracy/AI models with cross validation | SVM | Logistic regression | Neural Network | Decision tree |
|---|---|---|---|---|
| Acc. with Cross Validation | 0.980 | 0.960 | 0.980 | 0.97 |

The model that best predicts the Bankruptcy inferring from the above tabulated results are is the *Support vector machine* due to it's high model accuracy combined with higher test set accuracy.

# 7 References


Beklemysheva, A 2010, “Why Use Python for AI and Machine Learning?”,article, viewed January 2022,<https://steelkiwi.com/blog/python-for-ai-and-machine-learning/#:~:text=Python%20offers%20concise%20and%20readable,technical%20nuances%20of%20the%20language>

Bewaji, 2022, “Decision Tree Classifier”, article, viewed February 2022, <https://medium.com/@moibrahim002/decision-tree-classifier-d66c8b2c3a51>

Shubhendu, G 2022, “Cross validation in Data Science”, article, viewed March 2022, <https://medium.com/mlearning-ai/cross-validation-must-read-49b1b4c1154b>

Deepanshi, 2021, “All you need to know about your first Machine Learning model – Linear Regression”, article, viewed February 2022, <https://www.analyticsvidhya.com/blog/2021/05/all-you-need-to-know-about-your-first-machine-learning-model-linear-regression/>

Yadav, R 2020, “A Quick Introduction to Pandas”, article, viewed January 2022, <https://medium.com/@ritikayadav6/a-quick-introduction-to-pandas-500fe4e109fa>

Yalcin, OG 2020, “4 Machine Learning Approaches that Every Data Scientist Should Know”, article, viewed January 2022, <https://towardsdatascience.com/4-machine-learning-approaches-that-every-data-scientist-should-know-e3a9350ec0b9>

Dataflair, ”Machine Learning with Small Dataset”,article, viewed November 2022, <https://data-flair.training/blogs/machine-learning-with-small-dataset/>

# 8 Appendix A – Full source code

The full source code is available in a GIT repository

https://github.com/mithunbharadwaj/Bankruptcy

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
import numpy as np
from sklearn.svm import SVC
from sklearn import tree
from sklearn import datasets
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
#read the CSV
#

#read the CSV

data = pd.read_csv("data.csv")
X = np.array(data.drop(["Class"],axis=1))
y = np.array(data["Class"])

def rep(x):
    if x == 'NB':
        return 0
    else:
        return 1

def rep_column(x):
    if x == 'P':
        return float(1)
    elif x == 'A':
        return float(2)
    else:
        return float(3)


data["Class"] = data["Class"].map(rep)
data["Industrial Risk"] = data["Industrial Risk"].map(rep_column)
data["Industrial Risk.1"] = data["Industrial Risk.1"].map(rep_column)
data[" Financial Flexibility"] = data[" Financial Flexibility"].map(rep_column)
data["Credibility"] = data["Credibility"].map(rep_column)
data["Competitiveness"] = data["Competitiveness"].map(rep_column)
data["Operating Risk"] = data["Operating Risk"].map(rep_column)

data.to_csv("data_new.csv",index=None,header=True)



#Name columns
#data.columns = ["Industrial Risk","Industrial Risk"," Financial
Flexibility","Credibility","Competitiveness","Operating Risk","Class"]

#store file as data.csv

data.to_csv("data_new.csv",index=None,header=True)

data = pd.read_csv("data_new.csv")

X = np.array(data.drop(["Class"],axis=1))
```

```
y = np.array(data["Class"])

#Perform data split 90% tranin and 10% test data
[X_train,X_test,y_train , y_test] = train_test_split(X, y, test_size=0.1, random_state=0)

X,y = datasets.load_iris(return_X_y=True)


# Build Logistic Regression
Classifier = LogisticRegression(solver='liblinear')
model = Classifier.fit(X_train,y_train)
accu = model.score(X_train,y_train)


#Perform data split 70% data split and 30% test
[X_train,X_test,y_train , y_test] = train_test_split(X, y, test_size=0.2, random_state=0)

# Build SVC Classifier
Classifier = SVC(kernel='linear')
model = Classifier.fit(X_train,y_train)
accu = model.score(X_train,y_train)

print()
print("Accuracy of SVC: " , accu)
score_cv = cross_val_score(model,X,y , cv=5)
print("%0.2f accuracy with a standard deviation of %0.2f" % (score_cv.mean(),
score_cv.std()))
print("Scores of CV: " ,score_cv)
print("Traning accuracy")
print(accuracy_score(y_train, model.predict(X_train)))
print("testing accuracy")
predictions = model.predict(X_test)
print(accuracy_score(y_test, predictions))
print()

# Build Logistic Regression
Classifier = LogisticRegression(solver='liblinear')
model = Classifier.fit(X_train,y_train)
accu = model.score(X_train,y_train)

print("Accuracy of Logistic Regression: " , accu)
score_cv = cross_val_score(model,X,y , cv=5)
print("Scores of CV: " ,score_cv)
print("Traning accuracy")
print(accuracy_score(y_train, model.predict(X_train)))
print("testing accuracy")
predictions = model.predict(X_test)
print(accuracy_score(y_test, predictions))
print()


# Build Neural Network
Classifier = MLPClassifier(max_iter=1000)
model = Classifier.fit(X_train,y_train)
print("%0.2f accuracy with a standard deviation of %0.2f" % (score_cv.mean(),
score_cv.std()))
accu = model.score(X_train,y_train)

print()
print("Accuracy of Neural Network: ",accu)
score_cv = cross_val_score(model,X,y , cv=5)
print("Scores of CV: " ,score_cv)
print("Traning accuracy")
print(accuracy_score(y_train, model.predict(X_train)))
print("testing accuracy")
predictions = model.predict(X_test)
```

```
print(accuracy_score(y_test, predictions))
print("%0.2f accuracy with a standard deviation of %0.2f" % (score_cv.mean(),
score_cv.std()))
print()
print()
# Build Decision tree
Classifier = tree.DecisionTreeClassifier()
model = Classifier.fit(X_train,y_train)
accu = model.score(X_train,y_train)

print()
print("Accuracy of Decision tree: ",accu)
score_cv = cross_val_score(model,X,y , cv=5)
print("%0.2f accuracy with a standard deviation of %0.2f" % (score_cv.mean(),
score_cv.std()))
print("Scores of CV: " ,score_cv)
print("Traning accuracy")
print(accuracy_score(y_train, model.predict(X_train)))
print("testing accuracy")
predictions = model.predict(X_test)
print(accuracy_score(y_test, predictions))
print()
```

# 9 Appendix B – Data Set

```
Industrial Risk,Industrial Risk.1, Financial
Flexibility,Credibility,Competitiveness,Operating Risk,Class
N,N,A,A,A,N,NB
A,A,A,A,A,A,NB
P,P,P,P,P,P,NB
N,N,P,P,P,N,NB
A,A,P,P,P,A,NB
P,P,A,P,P,P,NB
P,P,P,A,A,P,NB
P,P,A,P,A,P,NB
P,P,A,A,P,P,NB
P,P,P,P,A,P,NB
P,P,P,A,P,P,NB
N,N,A,P,P,N,NB
N,N,P,A,A,N,NB
N,N,A,P,A,N,NB
N,N,A,P,A,N,NB
N,N,A,A,P,N,NB
N,N,P,P,A,N,NB
N,N,P,A,P,N,NB
A,A,A,P,P,A,NB
A,A,P,A,A,A,NB
A,A,A,P,A,A,NB
A,A,A,A,P,A,NB
A,A,P,P,A,A,NB
A,A,P,A,P,A,NB
P,N,A,A,A,P,NB
N,P,A,A,A,N,NB
P,N,A,A,A,N,NB
P,N,P,P,P,P,NB
N,P,P,P,P,N,NB
P,N,P,P,P,N,NB
N,N,A,P,P,P,NB
P,N,P,A,A,P,NB
N,P,A,P,A,P,NB
N,P,A,A,P,N,NB
P,N,P,P,A,N,NB
N,P,P,A,P,A,NB
A,N,A,P,P,A,NB
```

A,P,P,A,A,A,NB
A,A,A,P,A,P,NB
A,A,A,A,P,N,NB
N,A,P,P,P,N,NB
A,N,P,A,P,N,NB
A,P,P,A,P,N,NB
N,N,P,A,P,P,NB
N,N,P,P,P,P,NB
P,A,P,P,P,N,NB
A,P,P,P,P,N,NB
A,N,N,A,P,P,NB
A,P,P,A,P,P,NB
P,A,P,A,P,N,NB
A,N,A,A,P,P,NB
A,N,A,A,P,N,NB
A,N,A,A,P,A,NB
A,A,N,P,P,A,NB
A,A,N,P,P,P,NB
A,P,P,P,P,P,NB
A,P,A,A,A,P,NB
N,P,N,P,P,A,NB
N,N,P,P,P,A,NB
A,A,A,P,P,A,NB
A,A,A,P,P,P,NB
P,A,A,P,P,P,NB
A,N,N,P,P,A,NB
P,N,N,P,P,A,NB
A,A,N,P,P,P,NB
P,A,P,N,P,N,NB
P,A,N,A,P,N,NB
P,P,P,A,P,N,NB
P,A,P,N,P,N,NB
P,A,A,P,P,N,NB
P,P,A,P,P,P,NB
P,A,P,A,P,P,NB
P,P,A,A,P,N,NB
A,A,A,A,A,P,NB
P,P,A,P,P,N,NB
A,P,N,P,A,P,NB
A,P,A,P,A,P,NB
P,A,N,A,P,A,NB
A,P,N,P,A,P,NB
P,A,A,A,P,A,NB
A,P,A,P,A,P,NB
A,P,P,P,P,P,NB
P,A,A,N,P,P,NB
A,P,A,A,A,P,NB
A,N,N,N,P,P,NB
N,P,N,P,P,A,NB
P,N,A,N,A,P,NB
N,N,P,P,P,A,NB
A,A,A,P,P,A,NB
A,A,A,P,P,P,NB
P,A,A,P,P,P,NB
A,N,N,P,P,A,NB
A,A,N,P,P,P,NB
P,A,P,N,P,N,NB
P,P,P,A,P,N,NB
P,A,P,N,P,N,NB
P,A,A,P,P,N,NB
P,P,A,P,P,P,NB
P,A,P,A,P,P,NB
P,P,A,A,P,N,NB
A,A,A,A,A,P,NB
P,P,A,P,P,N,NB
A,P,N,P,A,P,NB
P,A,A,A,P,A,NB

A,P,A,P,A,P,NB
A,P,N,P,A,P,NB
A,P,A,P,A,P,NB
P,P,A,A,A,P,NB
N,N,A,A,A,N,NB
A,A,A,A,A,A,NB
P,P,P,P,P,P,NB
N,N,P,P,P,N,NB
A,A,P,P,P,A,NB
P,P,A,P,P,P,NB
P,P,P,A,A,P,NB
P,P,A,P,A,P,NB
P,P,A,A,P,P,NB
P,P,P,P,A,P,NB
P,P,P,A,P,P,NB
N,N,A,P,P,N,NB
N,N,P,A,A,N,NB
N,N,A,P,A,N,NB
N,N,A,P,A,N,NB
N,N,A,A,P,N,NB
N,N,P,P,A,N,NB
N,N,P,A,P,N,NB
A,A,A,P,P,A,NB
A,A,P,A,A,A,NB
A,A,A,P,A,A,NB
A,A,A,A,P,A,NB
A,A,P,P,A,A,NB
A,A,P,A,P,A,NB
P,N,A,A,A,P,NB
N,P,A,A,A,N,NB
P,N,A,A,A,N,NB
P,N,P,P,P,P,NB
N,P,P,P,P,N,NB
P,N,P,P,P,N,NB
N,N,A,P,P,P,NB
P,N,P,A,A,P,NB
N,P,A,P,A,P,NB
N,P,A,A,P,N,NB
A,N,N,N,N,A,B
P,N,N,N,N,N,B
N,P,N,N,N,N,B
A,P,N,A,N,N,B
N,N,N,N,N,N,B
N,N,N,A,N,A,B
N,N,N,N,N,P,B
N,N,N,N,N,A,B
N,N,N,A,N,P,B
N,N,N,A,N,N,B
N,N,A,N,N,N,B
P,N,N,N,N,N,B
A,N,N,N,N,N,B
N,N,N,N,N,N,B
P,N,N,N,A,A,B
A,N,N,N,N,A,B
A,N,N,N,N,A,B
A,A,N,N,N,P,B
A,N,N,N,N,N,B
P,A,N,N,N,A,B
P,N,N,N,N,P,B
P,A,N,N,N,N,B
P,N,N,N,N,N,B
N,A,N,N,N,P,B
N,N,N,N,N,A,B
A,A,N,N,N,N,B
A,A,N,N,N,N,B
P,P,N,N,N,N,B
A,P,N,N,N,N,B

P,A,N,N,N,N,B
A,N,N,N,N,N,B
N,N,N,N,N,N,B
N,A,N,N,N,N,B
P,N,N,N,N,N,B
N,P,N,N,N,N,B
A,P,N,A,N,N,B
N,N,N,P,N,P,B
N,N,N,N,N,N,B
N,N,N,A,N,A,B
N,N,N,P,N,N,B
N,N,N,N,N,P,B
N,N,N,N,N,A,B
N,N,N,A,N,P,B
N,N,N,A,N,N,B
N,N,A,N,N,N,B
P,N,N,N,N,N,B
A,N,N,N,N,N,B
N,N,N,N,N,N,B
P,N,N,N,A,A,B
P,N,N,N,A,A,B
A,N,N,N,N,A,B
N,N,N,N,N,A,B
N,P,N,N,N,N,B
A,P,N,A,N,N,B
N,N,N,N,N,N,B
N,N,N,A,N,A,B
N,N,N,P,N,N,B
N,N,N,N,N,P,B
N,N,N,N,N,A,B
N,N,N,A,N,P,B
P,N,N,N,N,P,B
P,A,N,N,N,N,B
P,N,N,N,N,N,B
N,A,N,N,N,P,B
N,N,N,N,N,A,B
A,A,N,N,N,N,B
A,A,N,N,N,N,B
N,N,N,A,N,N,B
N,N,A,N,N,N,B
P,N,N,N,N,N,B
A,N,N,N,N,N,B
N,N,N,N,N,N,B
N,A,P,A,N,P,B
P,N,N,N,N,N,B
N,A,N,N,N,P,B
N,N,N,N,N,A,B
A,A,N,N,N,N,B
A,A,N,N,N,P,B
A,N,N,N,N,N,B
P,A,N,N,N,A,B
P,N,N,N,N,P,B
P,A,N,N,N,N,B
P,N,N,N,N,N,B
N,A,N,N,N,P,B
N,N,N,N,N,A,B
A,A,N,N,N,N,B
A,A,N,N,N,N,B
A,P,N,N,N,N,B
P,A,N,N,N,N,B
A,N,N,N,N,N,B
N,N,N,N,N,N,B
N,A,N,N,N,N,B
A,N,N,N,N,A,B
P,N,N,N,N,N,B
N,P,N,N,N,N,B
A,P,N,A,N,N,B

```
N,N,N,N,N,N,B
N,N,N,A,N,A,B
N,N,N,N,N,P,B
N,N,N,N,N,A,B
N,N,N,A,N,P,B
N,N,N,A,N,N,B
N,N,A,N,N,N,B
P,N,N,N,N,N,B
A,N,N,N,N,N,B
N,N,N,N,N,N,B
P,N,N,N,A,A,B
```