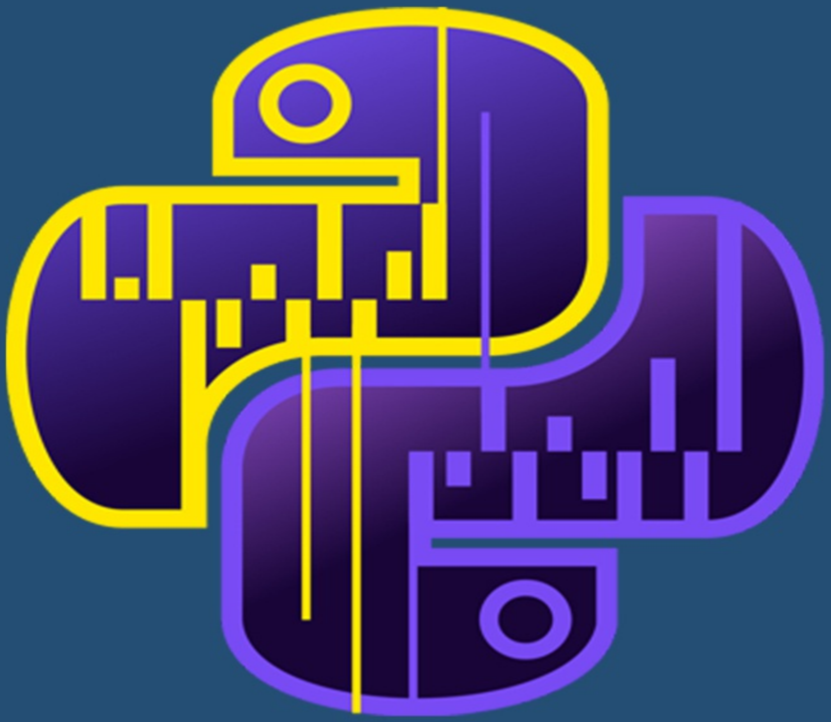
PYTHON

MACHINE LEARNING



THE BEGINNER'S GUIDE TO LEARN PYTHON MACHINE LEARNING INCLUDING KERAS, NUMPY, SCIKIT LEARN AND PYTORCH.

LILLY TRINITY

Table of Contents

ABOUT THIS BOOK
CHAPTER ONE INTRODUCTION
    Why Start With Python?
    Understand Why Python Is So Cool
    Discover the Reasons to Use Python
CHAPTER TWO THE CONCEPT OF PYTHON MACHINE LEARNING
    What Is Machine Learning?
    Applications of Machine Learning Algorithms
    Five Steps In Implementing Machine Learning
CHAPTER THREE ENVIRONMENTAL CONFIGURATION
    Libraries and Packages and Datasets
    Installation
CHAPTER FOUR DATA PROCESSING, ANALYSIS, AND VISUALIZATION
    Training Data and Test Data
CHAPTER FIVE TECHNIQUES AND ALGORITHMS
    Techniques Used In Machine Learning

ABOUT THIS BOOK

Python is high-level programming, general- purpose language that is increasingly used in data science and the design of machine learning algorithms. This book gives a quick introduction to Python and its libraries such as numpy, scipy, pandas, matplotlib, and how it can be used to develop machine learning algorithms that solve real problems. This book begins with an introduction to machine learning and the Python language and explains how to configure Python and its packages. It also covers all important concepts such as exploratory data analysis, data pre-processing, feature extraction, data visualization, and grouping, classification, regression, and performance evaluation of the model. This book also features several projects that teach techniques and features such as sorting news topics, detecting junk e-mail, forecasting online ad clicks, stock price forecasting, and several other machine learning algorithms.

We have written this book for professionals who are willing to learn the basics of Python and develop applications and software by making use of the machine learning techniques such as grouping, recommendation, and classification. In this book, you will be taught how to solve data problems and implement your solutions using the powerful but simple Python programming language and its packages. After reading this book, you will get a broad overview of the machine learning environment and best practices for machine learning techniques.

CHAPTER ONE

INTRODUCTION

On a technical level, Python is a high-level object-oriented programming language with integrated dynamic semantics, mainly for the development of Web and Web applications. It is extremely attractive in the rapid application development field because it offers dynamic typing and linking options.

Python is very easy to learn because it is relatively simple, but it requires a different syntax, focused on readability. Developers can easily read and translate Python code than other languages. This, in turn, reduces maintenance and program development costs by enabling teams to work collaboratively without significant language and language barriers.

Also, Python supports the use of modules and packages, which means that programs can be designed in a modular style, and that code can be reused in multiple projects. After developing a module or package that you need, you can size it for use in other projects, and you can easily import or export the modules.

One of the advantages of Python is that the standard library and interpreter are freely available in both source and binary form. There is also no exclusivity because Python and all the necessary tools are available on all major platforms. This is an attractive option for developers who do not want to worry about paying high development costs.

If this description of Python on your head, do not worry. You will have understood early enough. What you need to understand in this part is that Python is a high-level programming language which is used to develop software both in an application and on the web, including mobile devices. It is relatively easy to learn, and the necessary tools are available for free for all.

Python is a popular platform which is used for research and development of production systems. It's a vast language with many modules, packages, and libraries that offer different ways to accomplish a task.

Python and its libraries, such as NumPy, SciPy, Scikit-Learn, Matplotlib, are used in science and data analysis. They are also widely used to create scalable machine learning algorithms. Python makes use of popular machine learning techniques such as recommendation, classification, regression, and clustering.

Python provides a structure ready to efficiently perform data mining tasks on large volumes of data in less time. It includes several implementations performed using algorithms such as linear regression, logistic regression, Naïve Bayes, k-averages, the nearest neighbor K, and the random forest.

Machine learning, a leading subject in the field of artificial intelligence, has been in the spotlight for some time now. This field can offer an interesting opportunity, and starting a career in this field is not as challenging as it seems at first. Even if you have no experience in mathematics or programming, this is not an issue. The most crucial element of your success is purely your interest and your motivation to learn all these things.

Are you a beginner that does not know where to start his studies? Why you need Machine Learning and why you are gaining popularity, you have come to the right place! We have made provisions for all the relevant information and resources to help you acquire new knowledge and carry out your first projects.

Why Start With Python?

If your goal is to be a successful programmer, you need to know a lot. But for Machine Learning and Data Science, it is pretty enough to master at least one coding language and use it with confidence. So calm down, you do not have to be a genius of programming.

For successful machine learning, you must choose the correct coding language from the beginning, as your choice will determine your future. At this point, you need to think strategically, organize priorities correctly, and not waste time doing unnecessary things.

My opinion - Python is the ideal choice for beginners in the field of machine learning and data science. It is a minimalist and intuitive language with a complete library line (also called framework) which considerably reduces the time needed to obtain the first results.

Understand Why Python Is So Cool

Many programming languages are available today. A student can spend an entire semester at university and study computer languages without always hearing from everyone. (That's what I did during my college years.) One would think that programmers would be satisfied with all these programming languages and would choose only one to talk to the computer, but they would continue to invent more.

Programmers continue to create new languages for a good reason. Every language has something special to offer - something that does exceptionally well. Also, as computer technology evolves, programming languages evolve. Because creating an application relies on effective communication, many programmers are familiar with various programming languages to choose the right language for a specific task. One language may work better to obtain data from one database, and another may very well create user interface elements.

As in any other programming language, Python makes some exceptions, and you need to know what they are before you start using it. You might be surprised by the cool things you can do with Python. Knowing the strengths and weaknesses of a programming language allows you to use it better while avoiding the frustration of not using language for tasks that do not work well. The following sections help you make these kinds of decisions about Python.

Discover the Reasons to Use Python

Most programming languages are designed with specific goals. These goals help you define language characteristics and determine what you can do with the language. There is no way to create a programming language that does everything because people have conflicting goals and needs when designing applications. The main goal of Python was to create a programming language that would make programmers efficient and productive. With this in mind, here are the reasons why you want to use Python when creating an application:

Less application development time:
Python code is typically 2 to 10 times smaller than comparable code written in languages such as C / C ++ and Java, which means you spend less time writing your application and more time using it.

Easy to Read:
A programming language is like any other language - you must be able to read it to know what it does. Python language tends to be less difficult to read than code written in other languages, which means you spend less time interpreting and more time making necessary changes.

Learning time is Minimal:
The creators of Python wanted to create a programming language with fewer strange rules making learning difficult. After all, programmers want to create applications and not learn obscure and difficult languages.

It is important to realize that although Python is a popular language, it is not one of the most popular languages. It currently ranks eighth on websites such as TIOBE, an organization that records usage statistics. (among others).). If you are looking for a language only to get a job, Python is a good option, but C / C ++, Java, C # or Visual Basic would be better choices. Make sure you choose a language that's right for you and meets your application development needs, but also what you want to accomplish. Python was the best programming language in 2007 and 2010 and was ranked the fourth most popular language in February 2011. So, it's a good choice if you're searching for a job, but not necessarily the best choice. However, you may be astonished to find out that many colleges are now using python to teach coding, which has become the most popular language at this location.

CHAPTER TWO

THE CONCEPT OF PYTHON MACHINE LEARNING

Python is made up of libraries that allow developers to use optimized algorithms. Python implements common machine learning techniques such as recommendation, sorting, and grouping. Therefore, it is necessary we talk about a brief introduction to machine learning before we proceed.

What Is Machine Learning?

Machine Learning is now one of the most important topics among development organizations looking for innovative ways to leverage data resources to help businesses reach a new level of understanding. Why add machine learning to the mix? With appropriate machine learning models, companies can continually forecast business changes to predict next steps better. As data is added continuously, machine learning models ensure that the solution is continually updated. The value is simple: If you use the most variable and changing data sources in the context of machine learning, you can probably predict the future.

Machine learning is one of the artificial intelligence that allows a system to learn with information rather than explicit programming. However, machine learning is not an easy process.

Machine learning makes uses of a variety of algorithms that learn from iteratively to improve, describe, and predict results. As the algorithms absorb training data, it is possible to provide more accurate models from these data. A machine learning model is a result obtained after driving your machine learning algorithm with data. After coaching, when you provide a model with an entry, you receive an exit. For example, a program of predictive algorithms can produce a predictive model. Then, after proposing the predictive model with data, you will receive a forecast based on the data that led to the model. Machine learning is now essential for creating analysis models.

You probably interact with machine learning applications without realizing it. For example, when you visit an e-commerce website, and you start seeing products and reading reviews, you will probably see other similar products that might interest you. These recommendations are not coded by an army of developers. Suggestions are posted on the site via a machine learning model. The model ingests your browsing history, as well as navigation data and purchases from other customers, to present other related products that you may wish to purchase.

Artificial Intelligence, Data Science, and Machine learning are among the most popular topics in the world of technology. Data mining and Bayesian analysis are trends, which increases the demand for machine learning. This book is your guide into the machine learning world.

Machine learning is one of the disciplines that deal with system programming to enable them to learn and improve with experience automatically. Here, learning involves the recognition and understanding of input data and making informed decisions based on the data provided. It is very difficult to take into account all decisions based on all available inputs. To solve this problem, algorithms that develop knowledge from specific data and previous experiences were developed, applying the principles of statistics, probability, logic, mathematical optimization, learning by reinforcement, and control theory.

Applications of Machine Learning Algorithms

Machine Learning and Artificial intelligence (AI) are everywhere. You likely use them, and you are not even aware of them. In Machine Learning (ML), computers, software, and peripherals work through cognition similar to that of the human brain.

Typical successful machine language applications include programs that decode handwriting, facial recognition, speech recognition, speech recognition, pattern recognition, spam detection programs, weather forecasts, forecasts, and stock market analysis, and so on.

Facial Recognition

Facial recognition is among the most common areas where machine learning is being applied. There are many situations to use facial recognition. For example, high-quality cameras on mobile devices have made facial recognition a viable option for authentication and identification. Apple iPhone X is an example. The facial recognition application works on the software identifies 80 nodal points on a human face. And the nodal points are the endpoints used to measure variables in a person's face, such as the width or length of the nose, the depth of the orbits and the shape of the cheekbones.

Virtual Personal Assistants

Siri, Google Now, Alexa are some common examples of virtual personal assistants. These apps help find information when you are asked about the voice. Just activate them and ask questions such as "What are my appointments for today?", "What flights are available between Canada and New York?". To respond to these queries, the application searches for information retrieves its previous queries and accesses other resources to collect relevant information. You can even ask these assistants to perform certain tasks such as "Set an alarm at 5:30 the next morning," "Remind me to go to the passport office tomorrow at 10:30 am".

Analysis and forecast of traffic congestion

GPS navigation services monitor the user's position and speed and use them to create a map of the current traffic. This helps prevent traffic jams. Machine learning in these scenarios helps to estimate areas where congestion can be found based on previous records.

Automated video surveillance

Current video surveillance systems are based on AI, and machine learning is the technology that can detect and prevent crimes before they happen. They follow the strange and suspicious behavior of people and send alerts to human servants, who can ultimately help with accidents and crimes.

Social media
Facebook constantly monitors the friends you are in contact with, your interests, your workplace or a group you and someone else share, etc. Based on continuous learning; a Facebook users list is provided as suggestions from friends.

Facial Recognition
You upload a picture of yourself with a friend, and Facebook immediately recognizes that friend. Machine learning is at the heart of Computer Vision, a technique for extracting useful information from videos and images. Pinterest makes use of computer vision to identify pins or objects in images and recommend pins similar to their users.

Filtering unwanted email and malware
Machine learning is widely used in malware filtering and spam detection, and the databases of malware and spam continue to be updated to be processed efficiently.

Online customer support
Currently, on multiple sites, you have the opportunity to talk to the customer support representative when users are browsing the site. In most cases, instead of a real leader, you talk to a chatbot. These robots extract information from the site and provide it to customers to help them. Over some time, chatbots learn to understand users' queries better and provide them with better answers, which is possible thanks to machine learning algorithms.

Refinement of search engine results
Google and similar search engines use machine learning to improve search results for their users. Whenever a search is run, the backend algorithms control the response of users to the results. According to user responses, background algorithms improve search results.

Product recommendations
If a user buys or searches for a product online, they will continue to receive emails with purchase suggestions and announcements about that product. Depending on the user's previous behavior, a site/application, past purchases, items you liked or added to the shopping cart, brand

preferences, etc., product recommendations are sent to the user.

Online fraud detection

Machine learning is used to track online monetary fraud. For example, Paypal uses ML to prevent money laundering. The company uses a set of tools to compare millions of transactions and distinguish between legal and illegal transactions between buyers and sellers.

Speech Recognition

Speech recognition in another speech recognition Speech is a voice / a converted text. It is also referred to as automatic voice recognition (ASR). We may be looking for examples to help Google, Cortana, Siri, and Alexa. Speech recognition is one of the deep learning categories. The system analyzes the specific human voice and uses it to adjust the speech recognition of that person, thus providing greater accuracy. Simple voice commands can be converted to decoding and can be used to make phone calls, select radio stations, or listen to music from a compatible smartphone, MP3 player, or USB flash drive.

For example, use your smart device and open the Google (Android) or Siri (IOS) wizard and say "Hello Google / Siri!". And the device will answer you. By measuring user sounds while speaking, voice recognition software can measure the unique biological factors that combine to produce their voice.

Financial and Banking services

Machine learning is responsible for a lot in the banking and financial sectors. Help the financial institution of the banks.

Taaffeite Capital markets its products in a completely systematic and automated way using exclusive machine learning systems. Here is a list of funds and companies that use machine learning.

Design new products and service offerings for the right customers through the right data mining that supports simple, flexible and integrated processes to understand customers' buying habits and which channels the customer engages with and which are the determining factors. The application of machine learning to the creation of personalized products is essential for the next-generation banking sector.

Identify a client's risk score by nationality, occupation, salary, experience, industry, credit history, etc. This is very important for banks even before offering a product or customer service. This risk score is an important key performance indicator for banks that decide the interest rate and other product behaviors for the customer.

Health care

The health sector is made up of several many manual processes. In this field, the technology always helps to improve understanding of the patient's situation. By using these types of advanced analytics, we can provide better information to physicians when it comes to patient care. Have easy access to the patient's blood pressure, heart rate, lab tests, DNA tests, etc.

Image Recognition

Image recognition is one of the major advances of intelligent artificial algorithms. Facebook can detect your face that of your friends and probably even your dog, then assign the appropriate badge!

Image recognition or object analysis in images has progressed rapidly through the use of a subset of learning algorithms called convolutional neural networks. We do not explore what this means in this article but remember that it is a way of finding details that constitute an object.

For instance, if you want to detect a face, the algorithm must first learn to detect contours. As contour detection looks good, you may be able to detect angles or arcs. Then you can detect shapes like ovals or spheres.

This type of learning, if you will, is what happens under the hood of these convolutional neural networks (also called CNN). These networks (CNN) are a subset of deep learning algorithms. ThSo, when you or your organization start adding these features to the workflow, it's highly likely that developers will use CNN for image use cases! These use cases can range from face detection to object detection and image segmentation.

Five Steps In Implementing Machine Learning

Imagine that your company was planning to move to Industry 4.0. Now imagine that it's up to you to implement the big data analytics, machine learning, and artificial intelligence technologies required in the enterprise environment. You will need to know: where to start, what types of problems to expect, and how specific projects and associated services differ from those you are used to.

If you have encountered this article, you may already be in this position. However, this is likely to become your situation shortly, and it is now necessary to take advantage of someone else's experience to prepare. Although a lot of theory exists about enterprise applications for data analysis; there is a significant lack of practical and real experience in real life. This is because the adoption of these technologies for many industries is new and that the results of the pilot projects are emerging. Based on our work with one of the largest steel producers in the world, I will detail here some of our most useful and practical learnings.

The decrease in steel production costs with Magnitogorsk Steel and Magnetic Plants (MMK)

Machine learning technologies are being used successfully in forecasting and referral services. The Precise Forecast Base is historical data used as a training set. The result of this work is one or more models capable of predicting the most likely outcome of the technical process or set of options from which the best is chosen.

For example, Yandex Data Factory has developed a recommendation service for the Magnitogorsk Steel and Magnetic Plants (MMK), which reduces the use of iron alloys by an average of 5% at the oxygen converted stage of steel production. This not only saves about 5% of the ferro-alloys but above all to maintain safely and permanently the high quality of the steel obtained.

When you choose to apply machine learning technologies to tasks, you must choose the one that has measurable results and an economic effect. Also, there is a need for data and an understanding of how these recommendations and forecasts should be used in practice.

However, as we have learned, finding a solution is not just a leap forward. The process of creating a predictive project or recommendation involves several steps.

Step 1 - Determination of Objectives, Parameters, and Constraints

The first and very crucial step is to determine the objectives and constraints used in the modeling process. In the case of the ferroalloy optimization service, the main constraint is the need to respect the target chemical composition of the steel obtained using ferroalloys available in this specific melt. The objectives are the lowest possible cost of the ferroalloys used and the maximum proportion of recommendations accepted to be executed by the operator. The second objective is essential because, if the recommendations appear aggressive or sudden, they are likely to be rejected by the operator responsible for the management of the merger. For each objective, a metric must be chosen, and the model must be specially trained for that purpose. Its success will be determined according to this specific metric. So, choosing the right metric is a critical success factor. If the metric is poorly chosen, all the work on the model goes in the wrong direction.

Step 2 - Evaluating the data

The next step is to evaluate the available data sources, to estimate the volume and composition of the available data. Depending on the task, a set of parameters and fields, and the historical depth of the data, the experts decide how realistic the task is. An important difference between machine learning technologies is that the most valuable data is raw data, without aggregation or preprocessing. For modeling, it is better to have a larger amount of data, even inconsistent and contain errors and omissions, instead of a small amount of "net" data. If the data is not sufficient at this stage, you can define which data will help you solve the task and start collecting it.

When generating a model for MMK, the data used consisted of 200,000 smelter entries collected over seven years. The dataset included scrap metal and scrap materials, focused on the chemical composition of the steel obtained, the technical parameters of the oxygen conversion and refining steps, and the results of the chemical analysis.

Step 3 - Model Training

Modeling, unlike conventional software development, does not require pre-development of rules and algorithms. The data analyst determines the range of factors that can affect the modeled process. This is usually a long process of not losing important factors. At this point, it is necessary for the experts, knowing the subject and the process being modeled, to cooperate with the analysts who form the model and have the necessary tools. The training process is iterative; we go from 1 to 2 months to the design of the model, where the level of precision of this model is constantly increasing.

Unfortunately, the quality of the model can only be estimated after the end of your training and your tests. For many tasks, it is recommended to combine a particular formula, reflecting the known properties of the process, with refinement models based on machine learning technologies.

In this case, the generalized calculation, using the formula, is defined by the machine learning model. This combination allows you to take into account the general characteristics of the process and the local variations that can not be included in the formulas. This approach allowed us to ensure both an appropriate proportion of the accepted recommendations and significant savings in ferroalloys simultaneously in the MMK project.

However, some may find in this disadvantage that even after successful training, machine learning models do not interpret their results as to why a recommendation was recommended, such as business analysis. Although the quality of the predictive model is measurable and stable, it does not generate any new knowledge. Model training is performed in a limited data set and requires a great deal of computing power.

Step 4 - Integration and Testing

After forming the model, it is integrated into the customer management system. This integration is simplified because the model fulfills only one function: it predicts or recommends. In this respect, the interface is very simple, and the integration is reduced to the transfer of data and the display of these recommendations.

Practical tests are conducted with the participating client-side specialists. During the test, it is necessary to measure the accuracy of the model and to obtain an economic effect. In the service designed for MMK, the expert committee involved in the trial waived any liability of the operator if it did not fully agree with the recommendation made. This allowed us to estimate the effect of using the model and to determine the changes needed for its use in production. During testing, the model showed a decrease in ferroalloy use of about 5% and an expected savings of $ 4.3 million per year.

Step 5 - Model Monitoring

The last step is to use the production service that has passed the tests. The use of production requires constant quality control of the model and regular additional training on newly collected data. Unfortunately, fully autonomous machine learning systems are currently in the research phase, and their application for commercial and industrial purposes is not possible.

The most effective way to get to grips with Python for machine learning is to work with an end-to-end project and cover key milestones such as data loading, data synthesis, algorithm evaluation, and some predictions. This provides a replicable method that can be used in the dataset after the dataset. You can also add more data and improve the results.

CHAPTER THREE

ENVIRONMENTAL CONFIGURATION

Libraries and Packages and Datasets

To understand machine learning, you must have basic Python programming knowledge. Also, there are several libraries and packages commonly used to perform various machine learning tasks, as shown below -

TensorFlow
If you are currently working on a Python machine learning project, you may have heard of this popular open source library called TensorFlow.

This library was developed by Google and the Brain Team. TensorFlow is used in almost every Google machine learning application.

TensorFlow works as a computer library to write new algorithms that involve a large number of tensor operations because neural networks can easily be expressed as computer graphics that can be implemented using TensorFlow as a series of algorithms. Tensor operations. Also, the tensors are N-dimensional tables that represent their data.

TensorFlow Features
TensorFlow is optimized for speed and uses techniques such as XLA for fast linear algebra operations.

1. Reactive construction
With TensorFlow, we can easily visualize every part of the graph, which is not an option with Numpy or SciKit.

2. Flexible

One of the very important features of Tensorflow is that it is flexible in its operation, which means that it is modular and that you want to make the parts of it independent, it offers this option.

3. Easily drivable

It is easy to train in processors and GPUs for distributed computing.

4. Parallel training to the neural network

TensorFlow offers pipeline solutions in that you can form multiple neural networks and multiple GPUs, making models highly efficient in large-scale systems.

5. Great community

Needless to say, if it was developed by Google, there is already a large team of software engineers working continuously to improve stability.

6. Open Source

The fact that the machine learning library is an open source is one of the best things about it and that anyone can use it as long as it is connected to the Internet.

Where is TensorFlow used?

You use TensorFlow daily, but indirectly with applications such as Google Voice Search or Google Photos. These applications are developed using this library.

All libraries created in TensorFlow are coded in C and C ++. However, it has a complex interface for Python. Your Python code will be compiled and run on the TensorFlow distributed runtime engine, built in C and C ++.

The number of TensorFlow apps is unlimited, and that's the beauty of TensorFlow.

Scikit-Learn

It's a Python library associated with NumPy and SciPy. It is regarded as one of the best libraries for working with complex data.

Many changes have been made to this library. One change is the cross-validation feature, offering the ability to use multiple metrics. Many training methods, like the nearest neighbors and logistic regression, have been slightly improved.

Features of Scikit-Learn

1. Cross-validation: There are several methods to verify the accuracy of supervised models in invisible data.

2. Unsupervised learning algorithms: Again, the offer presents a large dispersion of algorithms: cluster, factor analysis, principal components analysis, and unsupervised neural networks.

3. Resource Extraction: useful for extracting image and text resources (for example, a word pocket)

Where is Scikit-Learn used?

It contains a large portion of algorithms to implement standard machine learning and data mining tasks, such as dimensionality reduction, classification, regression, clustering, and model selection.

Numpy

Numpy is regarded as one of the most popular machine learning libraries in Python.

TensorFlow and other libraries make use of Numpy internally to perform various tensor operations. The table interface is the most important and important feature of Numpy.

Numpy Resources

- Interactive: Numpy is easy to use and very interactive.
- Mathematics: simplifies complex mathematical implementations.
- Intuitive: facilitates coding and facilitates understanding of concepts.
- A lot of interaction: so much of the open source contribution has been widely used.

Where is Numpy used?

This interface can be used to express sound waves, images, and other binary raw streams as an array of N-dimensional real numbers.

To implement this library for machine learning, it is important that Full Pile developers have Numpy knowledge.

Keras

Keras is considered one of the best machine learning libraries in Python. It provides an easier mechanism for expressing neural networks. Keras also gives some of the best utilities for template compilation, dataset rendering, graphics visualization, and more.

In the background, Keras uses Theano or TensorFlow internally. Some of the most popular neural networks, such as CNTK, can also be used. Keras is relatively slow compared to other machine learning libraries. Because it creates a calculation graph using a backbone infrastructure and then uses it to perform operations. All models of Keras are portable.

Keras resources

- Works seamlessly on the processor and GPU.
- Keras supports almost all neural network models - fully connected, convoluted, grouped, recurrent, integrated, etc. Also, these models can be combined to build more complex models.
- Keras, modular in nature, are incredibly expressive, flexible, and adapted to innovative research.
- Keras is a fully Python-based framework that facilitates debugging and exploration.

Where is Keras used?

You're already in constant interaction with the features developed with Keras - it's used on Square, Netflix, Uber, Yelp, Instacart, ZocDoc, and many others. It is particularly popular among startups who place deep learning at the heart of their products.

Keras contains many commonly used neural network building block implementations such as layers, goals, activation functions, optimizers, and several tools that make it easy to work with image and text data.

Also, it provides numerous pretreated datasets and pre-trained templates such as MNIST, VGG, Inception, SqueezeNet, ResNet, and so on.

Keras is also one of the favorites among deep learning researchers, coming in second place. Keras has also been adopted by researchers from leading scientific organizations, mainly from CERN and NASA.

PyTorch

The largest machine learning library which allows developers to perform GPU-accelerated tensor calculations is known as PyTorch, create dynamic calculation graphs, and automatically calculate gradients. Also, PyTorch provides rich APIs for solving application problems that relate to neural networks.

This library is based on Torch, an open source machine library implemented in C with a Lua wrapper.

This Python machine library was released in 2017 and, since its inception, has grown in popularity and is attracting a growing number of machine learning developers.

PyTorch Features

Hybrid front

A new hybrid front end provides ease of use and flexibility in fast mode, while seamlessly moving to graphics for speed, optimization, and functionality in C ++ runtime environments.

Distributed training

Maximize search and production performance by leveraging native support for asynchronous collective action execution and Peer-to-peer communication accessible from Python and C ++.

Python first

PyTorch is not a Python binding in a C ++ unified framework. It was created to be deeply integrated with Python so that it can be used with libraries and popular packages such as Cython and Numba.

Libraries and Tools

An active community of researchers and developers has built a rich ecosystem of tools and libraries to extend PyTorch and support development in areas ranging from computational insight to reinforcement learning.

Where is PyTorch used?

PyTorch is mainly used for applications such as natural language processing.

It is primarily developed by Facebook's artificial intelligence research group, and Uber's "Pyro" software for probabilistic programming is built on it.

PyTorch is outperforming TensorFlow in many ways and is gaining a lot of attention in the last few days.

LightGBM

Gradient Boosting is among the most popular and best machine learning libraries that help developers build new algorithms using redefined elementary models and specifically decision trees. Therefore, some special libraries are designed for fast and efficient implementation of this method.

These libraries are LightGBM, XGBoost, and CatBoost. All of these libraries are competitors that help solve a common problem and can be used almost in the same way.

LightGBM Features

- Very fast computing ensures high production efficiency.
- Intuitive therefore makes the user-friendly.
- Training faster than many other deep learning libraries.
- It will not produce errors when you consider NaN values and other canonical values.

Where is LightGBM used?

These libraries provide highly scalable, optimized, and rapid gradient enhancement implementations, making it popular with machine learning developers. Because most full-stack machine learning developers have won machine learning competitions using these algorithms.

Eli5

Most of the time, machine learning model prediction results are not accurate, and the Eli5 machine learning library, built in Python, allows us to overcome this challenge. It is a combination of debugging and visualization of all machine learning models and tracks all the working steps of an algorithm.

Eli5 resources

Also, Eli5 supports the XGBoost, lightning, scikit-learn, and sklearn-crfsuite wGB libraries. All the libraries stated above can be used to perform different tasks using each one.

Where is Eli5 used?

Scientific applications that require a lot of computation in a short time.

Eli5 plays a vital role where dependencies exist with other Python packages.

Legacy applications and implementation of the latest methodologies in various fields.

Then, on the blog "Top 10 Python Libraries", we have the SciPy!

SciPy

SciPy is a machine learning library for engineers and application developers. However, you must know the difference between the SciPy stack and the SciPy library. The SciPy library contains modules for statistics, optimization, linear algebra, and integration.

SciPy Resources

The main feature of the SciPy library is that it is developed using NumPy, and its array makes the most use of NumPy.

Also, SciPy provides all efficient numerical routines, like numerical integration, optimization, and many others using its specific submodules.

All functions embedded in all SciPy submodules are properly documented.

Where is SciPy used?

SciPy is a library that uses NumPy to solve mathematical functions. SciPy uses NumPy arrays as a basic data structure and comes with modules for various tasks commonly used in scientific programming.

Tasks including linear algebra, integration (calculation), resolution of ordinary differential equations, and signal processing are easily handled by SciPy.

Theano

Theano is a machine learning library of a computational framework in Python for multidimensional array computing. Theano works similarly to TensorFlow but is not as efficient as TensorFlow. Because of their inability to fit into production environments.

Also, Theano can be used in distributed or parallel environments similar to TensorFlow.

Theano resources

Strong integration with NumPy - Ability to use completely NumPy arrays in functions compiled by Theano.

Transparent use of a GPU - Perform data-intensive calculations much faster than on a CPU.

Effective Symbolic Differentiation - Theano manufactures its derivatives for functions with one or more inputs.

Speed and Stability Optimizations - Get the correct answer for log (1 + x) even when x is too small. This is just one of the examples to show the stability of Theano.

Dynamic code generation C: Evaluate expressions faster than ever before, increasing efficiency by far.

In-depth unit testing and automatic verification - Detect and diagnose various types of errors and ambiguities in the model.

Where is Theano used?

The exact syntax of Theano expressions is symbolic and can be introduced for beginners accustomed to normal software development. More precisely, the expression is defined in the abstract sense, compiled and used later to perform calculations.

It has been specially designed to handle the types of computing required for large neural network algorithms used in Deep Learning. It was one of the first such libraries (development began in 2007) and is termed as an industry standard for deep learning development.

Theano is currently used in various neural network projects, and Theano's popularity is growing over time.

Pandas

Pandas is a Python machine learning library that provides high-level data structures and a wide variety of analysis tools. One of the main characteristics of this library is the ability to translate complex operations with data using one or two commands. Pandas have many internal methods of grouping, combining data and filtering, as well as time series features.

All of these are followed by excellent speed indicators.

Characteristics of pandas

Pandas facilitate the entire process of data manipulation. Support for operations such as reindexing, iteration, sorting, aggregations, concatenations, and views are some of the highlights of the Pandas feature.

Where are the pandas used?

There are fewer versions of the pandas library, which include hundreds of new features, bug fixes, improvements, and API changes. Panda enhancements take into account their ability to group and sort data, select the most relevant result for the method of application, and support custom-type operations.

Data analysis among all others is at the forefront of pandas use. But when used with other libraries and tools, Pandas guarantees high functionality and flexibility.

Installation

To create applications, you need another application unless you want to get a low level and write applications in machine code - a tough experience that even real programmers avoid as much as possible. Writing an application using the Python programming language requires some important applications. These applications allow you to work with Python by creating Python code, providing the necessary help information, and allowing you to execute the code you are writing. This chapter helps you get a copy of the Python application, install it on your hard drive, look for installed applications so you can use them, and test your installation to see how it works.

Download the version you need

Each platform (a combination of hardware and operating system software) is governed by special rules when running applications. The Python application hides these details. You enter the code that runs on any platform supported by Python, and Python applications translate that code into something that the platform can understand. However, for the translation to take place, you must have a version of Python that works on your specific platform. Python supports these platforms:

Amiga Research OS (AROS)

IBM Advanced Unix (AIX)

400 Application System (AS / 400)

Hewlett-Packard Unix (HP-UX)

BeOS

Linux

Microsoft Disk operating system (MS-DOS)

Mac OS X (pre-installed with the operating system)

MorphOS

OS 390 (OS / 390) and z / OS

Operating system 2 (OS / 2)

PalmOS

Psion

playground

QNX

series 60

Windows CE / Pocket PC

RISC OS (originally Acorn)

Solaris

32-bit Windows (XP and later)

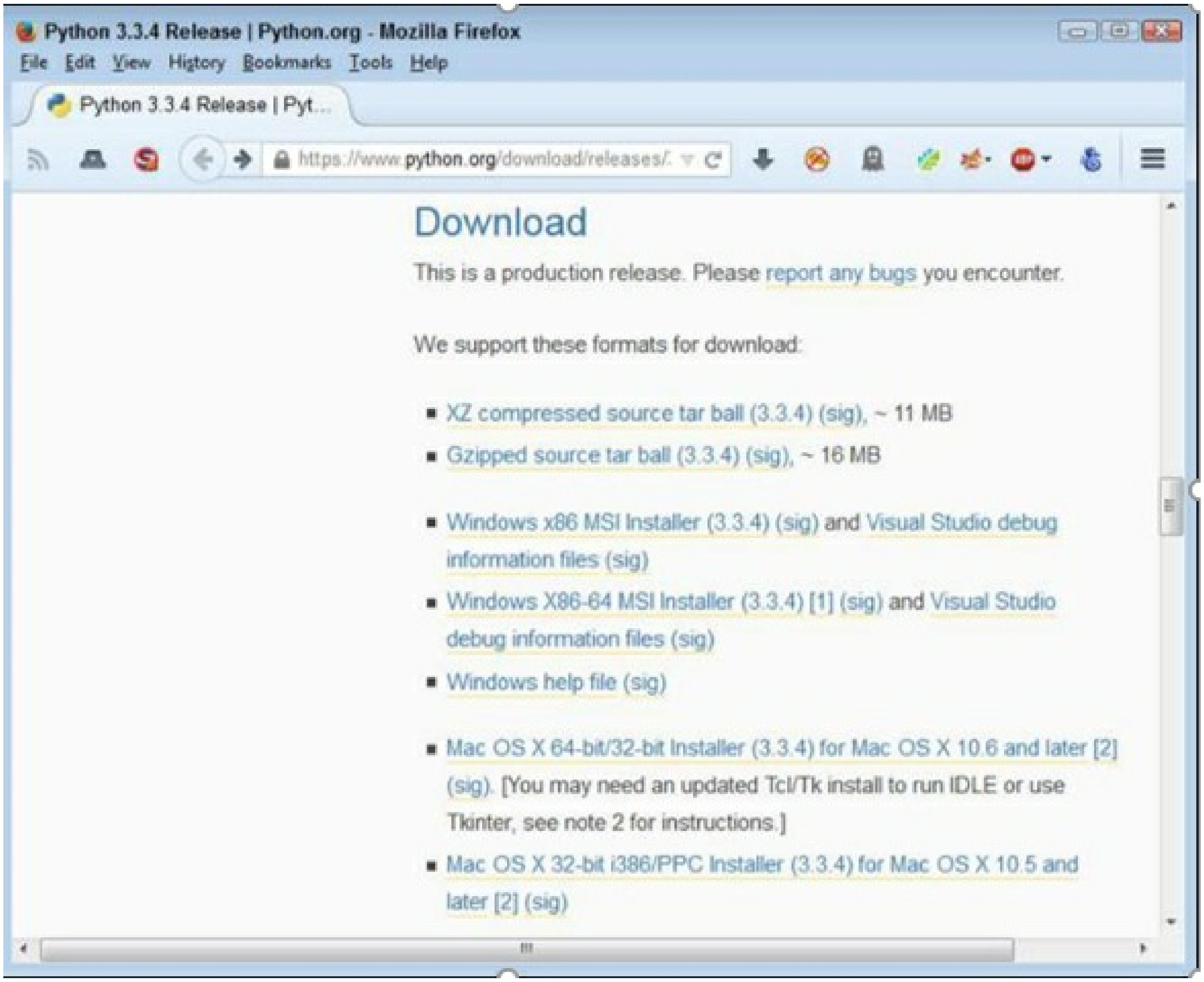
Virtual memory system (VMS)

64-bit Windows

Wow, that's a lot of different platforms! This book has been tested with Windows, Mac OS X, and Linux platforms. However, the examples can also work with these other platforms because they do not depend on any code specific to that platform.

To get the correct version for your platform, you need to access HTTP: //

www. python.org/download/releases/3.3.4/. Since the download section is initially hidden, you must scroll down the page.


Python 3.3.4 Release | Python.org - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Python 3.3.4 Release | Pyt...
https://www.python.org/download/releases/

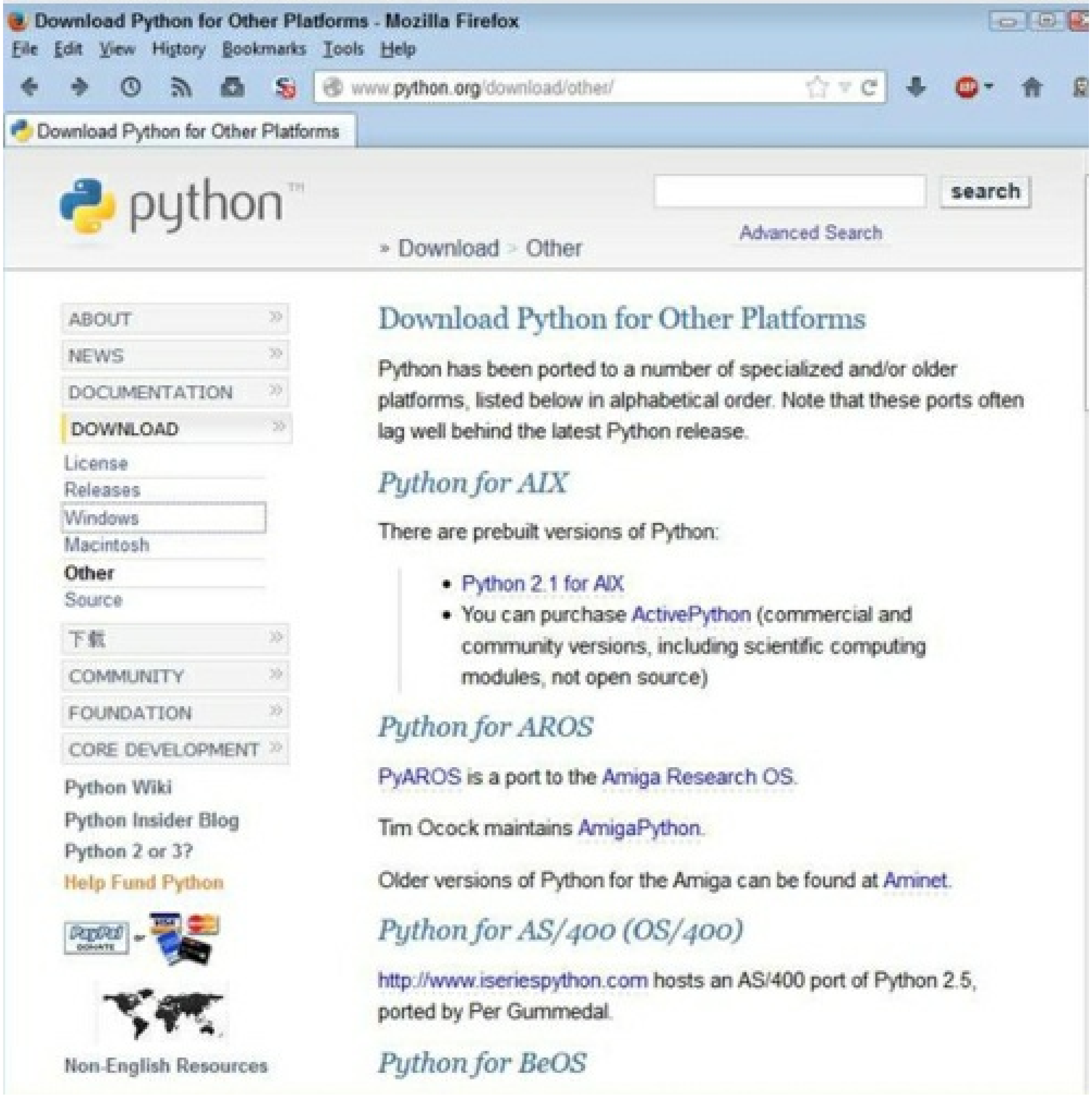
Download

This is a production release. Please report any bugs you encounter.

We support these formats for download:

- XZ compressed source tar ball (3.3.4) (sig), ~ 11 MB
- Gzipped source tar ball (3.3.4) (sig), ~ 16 MB
- Windows x86 MSI Installer (3.3.4) (sig) and Visual Studio debug information files (sig)
- Windows X86-64 MSI Installer (3.3.4) [1] (sig) and Visual Studio debug information files (sig)
- Windows help file (sig)
- Mac OS X 64-bit/32-bit Installer (3.3.4) for Mac OS X 10.6 and later [2] (sig). [You may need an updated Tcl/Tk install to run IDLE or use Tkinter, see note 2 for instructions.]
- Mac OS X 32-bit i386/PPC Installer (3.3.4) for Mac OS X 10.5 and later [2] (sig)

If you want to use another platform, click the second link on the left side of the page. You discover a list of Python installations for other platforms. Many of these installations are run by volunteers and not by people who create Python versions for Windows, Mac OS X, and Linux. Be sure to contact these people when you have questions about the installation, as they know how to help you get a good setup on your platform.



Download Python for Other Platforms

Python has been ported to a number of specialized and/or older platforms, listed below in alphabetical order. Note that these ports often lag well behind the latest Python release.

Python for AIX

There are prebuilt versions of Python:

- Python 2.1 for AIX
- You can purchase ActivePython (commercial and community versions, including scientific computing modules, not open source)

Python for AROS

PyAROS is a port to the Amiga Research OS.

Tim Ocock maintains AmigaPython.

Older versions of Python for the Amiga can be found at Aminet.

Python for AS/400 (OS/400)

http://www.iseriespython.com hosts an AS/400 port of Python 2.5, ported by Per Gummedal.

Python for BeOS

Install Python

After downloading your copy of Python, it's time to install it on your system.

The downloaded file contains everything you need to get started:

- Python interpreter
- Help files (documentation)
- Command Line Access
- IDLE (Integrated Development Environment) application
- Uninstaller (only on platforms that need it)

Work with Windows

The process of installation on a Windows system follows the same procedure that is used for other types of applications. The main difference is finding the file you downloaded so that you can start the installation process. The following procedure should work correctly on any Windows system, whether you use the 32-bit or 64-bit version of Python.

Find the downloaded copy of Python on your system.

The name of this file varies, but it usually appears under the following names: python-3.3.4.amd64.msi for 64-bit systems and python-3.3.4.msi for 32-bit systems. The version number is inserted in the file name. In this case, the file name refers to version 3.3.4, which is the version used for this book.

Double-click on the installation file.

(You can see an Open File - Security Warning dialog box asking if you want to run this file.) Click Run if this dialog box appears.) A Python Setup dialog box similar to that shown in Figure 2 is displayed. 3 The precise dialog box you see depends on the version of the Python installer you download.

Choose a user installation option, and then click Next.

The installation prompts you for the name of an installation directory for Python. Using the default destination will save you effort and time later.

However, you can install Python anywhere.

Using the Windows \ Program Files (x86) folder or Program Files is problematic for two reasons. First, the folder name has space, which makes access difficult from the application. Second, the folder usually requires administrator access. You will have to constantly fight with the Windows User Account Control (UAC) feature if you install Python in any folder.

Type a destination folder name, if needed, and then click Next. Python asks you to customize your installation.

Enabling the Add python.exe option to the path will save you time. This feature allows you to access Python from the command prompt window. Do not worry too much about how you use this feature at the moment, but it's a good feature to install. The book assumes that you have enabled this feature. Do not worry about the other features you see in Figure 2-5. They are all enabled by default, giving you maximum access to Python features.

(Optional) Select on the down arrow next to the Add python.exe to path option and choose the options and will be installed on the local drive.

Click Next.

You see the installation process begin. A User Account Control box may appear asking you if you want to perform the installation. If you notice this dialog box, click Yes. The installer continues, and a Setup Complete dialog box appears.

Click Finish.

Python is ready to use.

Work with Mac

Python is probably on your Mac system. However, this installation usually takes a few years, regardless of the age of your system. For this book, the installation will probably work properly. You will not test the limits of Python programming technology - you will learn how to use Python.

The latest version of OS X at the time of this publication (Mavericks, or 10.9) comes with Python 2.7, which is very useful for working with book examples.

Depending on how you use Python, you may want to update your installation at some point. Part of this process involves installing the GCC (GNU Compiler Collection) tools so that Python has access to the low-level resources you need. The following steps begin installing a new version of Python on the Mac OS X system.

Click on the link for your version of OS X:

- Python 3.3.5 Mac OS X 32-bit i386 / PPC installation program for 32-bit versions on the Power PC processor
- Python 3.3.5 Mac OS X 32-bit / 64-bit x86-64 / i386 installation program for 32-bit or 64-bit versions on Intel

The Python disk image starts to download. Be patient: downloading the disk image takes several minutes. You can easily see how long it will take to the download because most browsers provide a method to monitor the download process. Once the download is complete, Mac will automatically open the disk image for you.

The disk image looks like a folder. In this folder, you see several files like python.mpkg. The python.mpkg file contains the Python application. Text files contain information about the latest compilation, licenses, and annotations.

Double-click on python.mpkg.

You see a welcome dialog that informs you about this particular Python build.

Click Continue three times.

The installer displays the latest notes on Python, the license information (click Accept when asked about license information), and finally a target dialog box.

Select the volume (hard disk or other media) that you want to use to install Python, and then click Continue.

The Installation Type dialog box appears. This dialog box performs two tasks:

Click Customize to change the feature set installed on your system.

Click Change Installation Location to change the location where the installer places Python.

The book assumes you are performing a default installation and have not changed the installation location. However, you can make use of these options in case you want to use them.

Click Install.

The installer can request your administrator password. Enter the administrator username and password, if necessary, in the dialog box and click OK. You see a Python Installation dialog box. The contents of this dialog will change as the installation process progresses. This will tell you which part of Python works with the installer.

When you have successfully installed the software, you will see a Successful Setup dialog box.

Click Close.

Python software is ready to use. (You may decide to close the disk image at this junction and delete it from your system.)

Work with Linux

Python software some with some versions of Linux. For example, if you have an RPM (Red Hat Package Manager) -based distribution (such as CentOS, SUSE, Yellow Dog, Red Hat, and Fedora Core), you probably already have Python on your system, and you have nothing else to do.

Depending on the version of Linux that you use, the version of Python varies, and some systems do not include the Interactive Development Environment (IDE) application. If you have an earlier version of Python (version 2.5.1 or earlier), you may want to install a newer version to access IDLE. Most book exercises require the use of IDLE.

Using the default Linux installation

The default installation of Linux runs on any system. However, you must work in the terminal and enter the commands to complete it. Some of the actual commands may vary depending on the version of Linux. The information on http://docs.python.org/3/install/ provides useful tips that you can use in addition to the following procedure.

Click on the link that matches your Linux version:

Compressed source archive Python 3.3.4 (any version of Linux)

Python 3.3.3 xzip python fonts (better compression and faster download)

You will be prompt to either open or save the file, choose Save.

Python source files are being downloaded. Be patient: downloading source files take a minute or two.

Double-click on the downloaded file.

The Archive Manager window opens. Once the files are extracted, you see the Python 3.3.4 folder in the file manager window.

Double-click the Python 3.3.4 folder.

The file manager extracts the files from the Python 3.3.4 subfolder from your folder.

Open a copy of the terminal.

The terminal window is displayed. If you have never created software on your system before, you must install the basics of the compilation, SQLite and bzip2. Otherwise, the installation of Python will fail. Otherwise, you can go to step 10 to start using Python without any delay.

Press Enter after typing the following "sudo apt-get install build-essential."

Linux installs the necessary Build Essential support for creating packages (see https://packages.debian.org/squeeze/build-essential for more details).

Press Enter after typing the following "sudo apt-get install libsqlite3-dev."

The SQLite support needed by Python software for database manipulation is installed by Linux (see https://packages.debian.org/squeeze/libsqlite3-dev).

Press Enter after typing the following "sudo apt-get install libbz2-dev."

The bzip2 support required by Python software for file manipulation is installed by Linux (see https://packages.debian.org/sid/libbz2-dev for more details).

Type CD Python 3.3.5 in the Terminal window and press Enter. The terminal changes directories in the Python 3.3.5 folder of your system.

Type ./configure and press Enter.

The script starts by checking the type of system build, then performs a series of tasks depending on the system you are using. This process can take one or two minutes because there is a long list of things to check.

Type make and press Enter.

Linux runs the creation script to create the Python application software. The manufacturing process may take a minute - this depends on the processing rate of your system.

Type sudo make altinstall and press Enter.

The system may prompt you for your administrator password. Enter your password and press Enter. At this point, several tasks occur when the system installs Python on your system.

CONCEPTS OF LEARNING

Learning involves the process of transforming experience into knowledge or experience.

As indicated below, learning can be broadly categorized into three categories, depending on the nature of the learning data and the interaction between the student and the environment.

- supervised learning
- Unsupervised learning
- Semi-supervised learning

Likewise, there are four categories of machine learning algorithms, as shown below -

- supervised learning
- unsupervised learning
- semi-supervised learning
- reinforcement learning

However, the most used are supervised and unsupervised learning.

Supervised learning

Supervised learning is commonly used in real-world applications such as facial and voice recognition, product or film recommendations, and sales forecasting. Supervised learning is classified into two types - Regression and Classification.

The regression drives and predicts a continuous value response, for example, the prediction of real estate prices.

The classification attempts to find the appropriate class label, for example, by analyzing positive/negative sentiment, men and women, benign and malignant tumors, secure and unsecured loans, and so on.

In supervised learning, the learning data is accompanied by a description, labels, goals, or desired results. The goal is to find a general rule that maps inputs to outputs. This type of learning data is called labeled data. The learned rule is then used to tag new data with unknown outputs.

Supervised learning entails building a machine learning model based on labeled samples. For instance, if we build a system to estimate the price of land or a house based on various characteristics such as size, location, etc., we must first create a database and then 'label. We need to teach the algorithm which features match what prices. According to this data, the algorithm will master how to calculate the price of the property using the values of the input resources.

Supervised learning involves learning a function from available training data. Here, a learning algorithm analyzes the learning data and produces a derived function that can be used to map new examples. There are many supervised learning algorithms, which are: Naive Bayes classifiers, logistic regression, neural networks, and support vector machines.

Common examples of supervised learning include sorting e-mails into categories of spam and unwanted junk mail, labeling web pages based on their content, and voice recognition.

Unsupervised learning

Unsupervised learning is used to detect anomalies, exceptions, such as fraud or defective equipment, or to group customers with similar behaviors into a sales campaign. It's the opposite of supervised learning. There is no data labeled here.

When the training data contains only a few indications without a description or tag, it is up to the coder or algorithm to look for the underlying data structure, to determine how to describe the data, or to discover hidden patterns. This type of learning data is called unbranded data.

Suppose we have multiple data points and want to classify them into several groups. We can not know exactly what the classification criteria would be. Thus, an unsupervised learning algorithm attempts to ideally classify the given data set into a given number of groups.

Unsupervised learning algorithms are remarkably powerful tools for analyzing data and identifying patterns and trends. They are most often used to group similar entries into logical groups. Unsupervised learning algorithms are Kmeans, Random Forests, Hierarchical Clustering, and so on.

Semi-supervised learning

If some learning samples are tagged, but others are not, this is semi-supervised learning. It uses a large amount of uncollected learning data and a small amount of tagged data for testing. Semi-supervised learning is applied in cases where acquiring a fully labeled dataset is expensive, although it is more practical to label a small subset. For example, it is usually necessary for qualified experts to classify some remote sensing images and many field experiments to locate oil at a specific location while acquiring unlabelled data is relatively easy.

Reinforcement learning

Here, the training data provides the system with feedback that allows it to adapt to the dynamic conditions for achieving a given goal. The system evaluates its performance according to the responses received and reacts accordingly. The best-known examples include autonomous cars and the AlphaGo main algorithm.

Purpose of machine learning

Machine learning can be considered a branch of artificial intelligence or artificial intelligence because the ability to transform experience into an experience or to detect patterns in complex data is a hallmark of human intelligence or animal.

As a scientific domain, machine learning shares common concepts with other disciplines, such as statistics, information theory, game theory, and optimization.

As a sub-domain of information technology, your goal is to program machines to enable them to learn.

However, it should be noted that the purpose of machine learning is not to create automated duplication of intelligent behavior, but to use the power of computers to supplement and supplement human intelligence. For example, machine learning programs can analyze and process huge databases, detecting patterns that go beyond human perception.

CHAPTER FOUR

DATA PROCESSING, ANALYSIS, AND VISUALIZATION

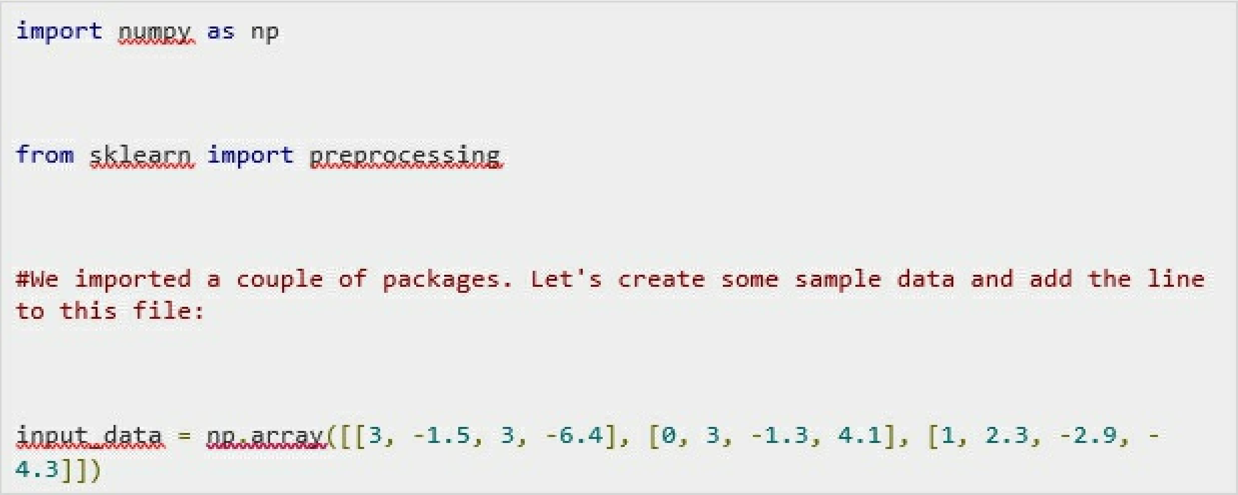
In the real world, many raw data are not adequate for quick processing by machine learning algorithms. We need to pre-process the raw data before introducing it into various machine learning algorithms. This chapter discusses various data preprocessing techniques in Python machine learning.

Pre-Processing of data

How we process data in Python will be discussed in this section.

First, open a file with a .py extension, such as prefoo.py, in a text editor such as Notepad.

Then add the following code snippet to this file –



```
import numpy as np


from sklearn import preprocessing


#We imported a couple of packages. Let's create some sample data and add the line
to this file:


input_data = np.array([[3, -1.5, 3, -6.4], [0, 3, -1.3, 4.1], [1, 2.3, -2.9, -
4.3]])
```

We are now ready to exploit this data.

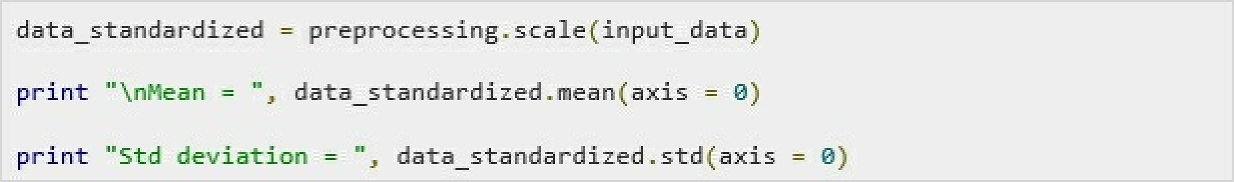
Pre-processing techniques

The data can be pre-processed using various techniques, as shown here:

Mean Removal

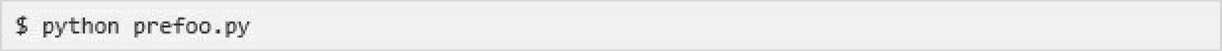
This involves removing the average of each resource so that it is centered at zero. Average elimination helps eliminate any tendency of resources.

You can utilize the below code snippet for mean removal –



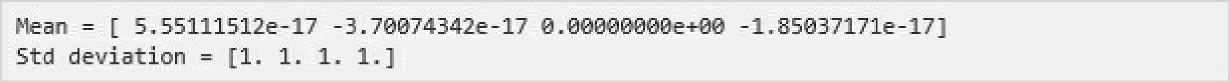
```
data_standardized = preprocessing.scale(input_data)

print "\nMean = ", data_standardized.mean(axis = 0)

print "Std deviation = ", data_standardized.std(axis = 0)
```

Now run the below command on the terminal –



```
$ python prefoo.py
```

You can check the following output –



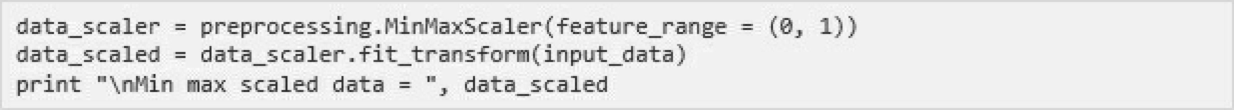
```
Mean = [ 5.55111512e-17 -3.70074342e-17 0.00000000e+00 -1.85037171e-17]
Std deviation = [1. 1. 1. 1.]
```

Note that at the output, the average is almost equal to 0, and the standard deviation is 1.

Scaling

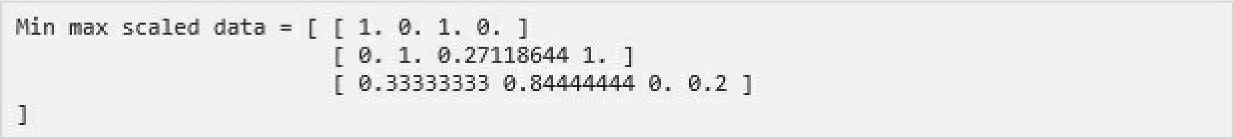
The values of each resource in a data point can vary between random values. Therefore, it is important to scale them to match the specified rules.

For scaling, make use of the below code snippet –



```
data_scaler = preprocessing.MinMaxScaler(feature_range = (0, 1))
data_scaled = data_scaler.fit_transform(input_data)
print "\nMin max scaled data = ", data_scaled
```

Now run the code above, and you can observe the following output –



```
Min max scaled data = [ [ 1. 0. 1. 0. ]
                        [ 0. 1. 0.27118644 1. ]
                        [ 0.33333333 0.84444444 0. 0.2 ]
]
```

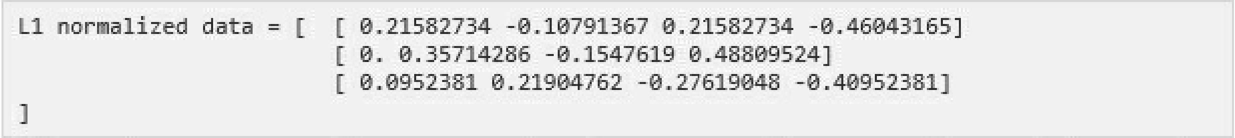
Note that all values have been scaled between the given interval.

Normalization

Normalization consists of adjusting the values of the resource vector to measure them on a common scale. In normalization, the values of a feature vector are adjusted to add 1. We add the below lines to the prefoo.py file -

You can utilize the following code for normalization –

Now run the code above, and you can observe the following output –

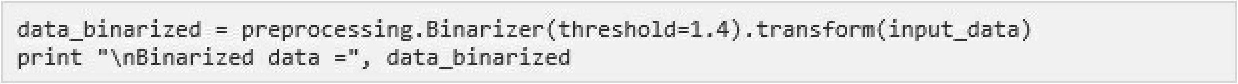


```
L1 normalized data = [  [ 0.21582734 -0.10791367 0.21582734 -0.46043165]
                        [ 0. 0.35714286 -0.1547619 0.48809524]
                        [ 0.0952381 0.21904762 -0.27619048 -0.40952381]
]
```

Standardization is used to ensure that data points are not increased due to the nature of their resources.

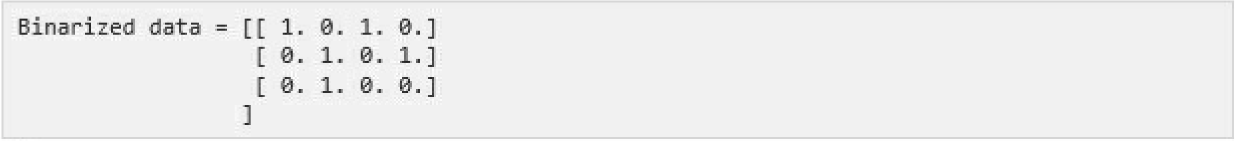
Binarization

Binarization is used to convert a digital resource vector into a Boolean vector. For binarization, you can use the following code –



```
data_binarized = preprocessing.Binarizer(threshold=1.4).transform(input_data)
print "\nBinarized data =", data_binarized
```

Now run the code above, and you can observe the following output –



```
Binarized data = [[ 1. 0. 1. 0.]
                  [ 0. 1. 0. 1.]
                  [ 0. 1. 0. 0.]
                 ]
```

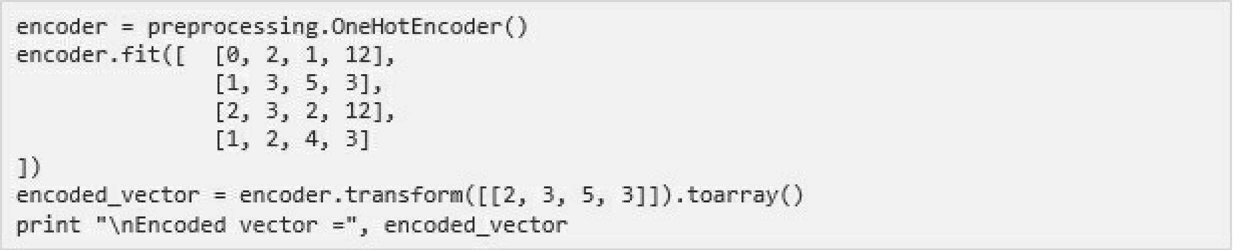
This technique is useful when we have prior knowledge of the data.

One Hot Encoding

You may need to handle numeric values that are few and far between, and you may not need to store these values. In such situations, you can use the One Hot Encoding technique.

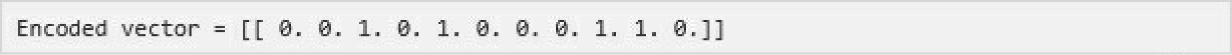
If K is the number of distinct values, the resource will be transformed into a vector of dimension k, where a single value is 1, and all other values are 0.

You can use the following code for a hot encoding –



```
encoder = preprocessing.OneHotEncoder()
encoder.fit([  [0, 2, 1, 12],
               [1, 3, 5, 3],
               [2, 3, 2, 12],
               [1, 2, 4, 3]
])
encoded_vector = encoder.transform([[2, 3, 5, 3]]).toarray()
print "\nEncoded vector =", encoded_vector
```

Now run the code above, and you can observe the below output –



```
Encoded vector = [[ 0. 0. 1. 0. 1. 0. 0. 0. 1. 1. 0.]]
```

In the example above, we will consider the third resource in each resource vector. The values are 1, 3, 5, and 3.

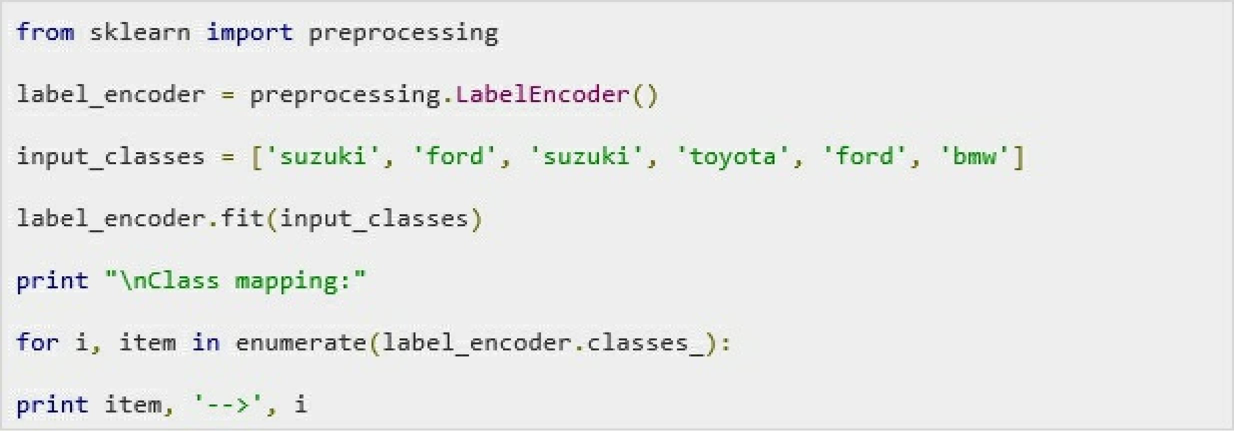
There are four distinct values here, which means that the hot-coded vector will be of length 4. If we want to encode the value 3, it will be a vector [0, 1, 0, 0]. Only a single value can be 1 in this vector. The second element is 3, which indicates that the value is 5.

Label Encoding

In supervised learning, there is usually a variety of labels that can be in the form of numbers or words. If they are digital, they can be used directly by the algorithm. However, labels must often be in a readable format. Thus, the training data is generally labeled with words.

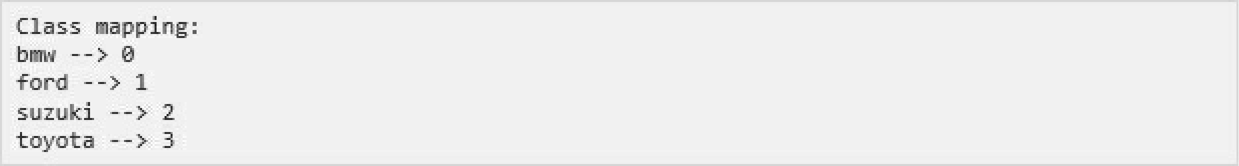
Label coding involves changing the labels of words into numbers so that algorithms can understand how to use them. We will understand in detail how to perform label encoding -

Create a fresh Python file and import the pre-processing package –



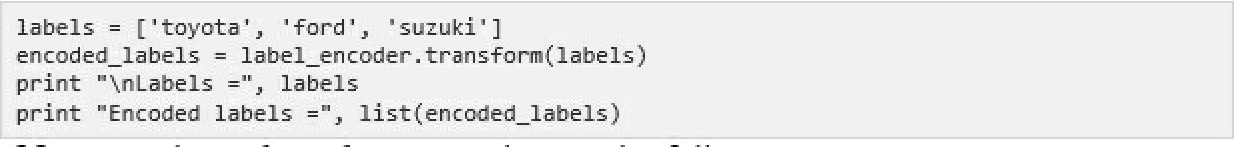
```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()

input_classes = ['suzuki', 'ford', 'suzuki', 'toyota', 'ford', 'bmw']

label_encoder.fit(input_classes)

print "\nClass mapping:"

for i, item in enumerate(label_encoder.classes_):

print item, '-->', i
```

Now run the code above, and you can observe the following output –



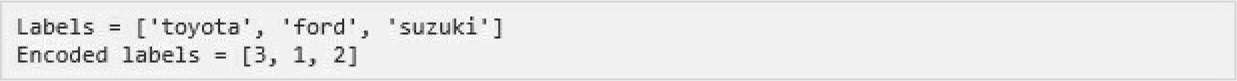
```
Class mapping:
bmw --> 0
ford --> 1
suzuki --> 2
toyota --> 3
```

As displayed in the above output, the words have been altered into 0-indexed numbers. Now, a set of labels is what we deal with; they can be transformed as follows –



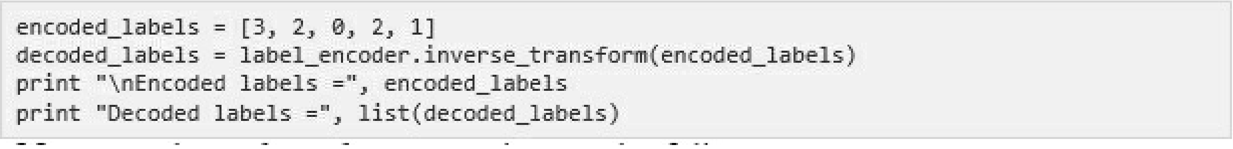
```
labels = ['toyota', 'ford', 'suzuki']
encoded_labels = label_encoder.transform(labels)
print "\nLabels =", labels
print "Encoded labels =", list(encoded_labels)
```

Now run the code above, and you can observe the following output –



```
Labels = ['toyota', 'ford', 'suzuki']
Encoded labels = [3, 1, 2]
```

This is effective only to manually maintain the correspondence between words and numbers. You can check how to turn the numbers into word labels, as shown in the code, here –



```
encoded_labels = [3, 2, 0, 2, 1]
decoded_labels = label_encoder.inverse_transform(encoded_labels)
print "\nEncoded labels =", encoded_labels
print "Decoded labels =", list(decoded_labels)
```

Now run the code above, and you can observe the following output –

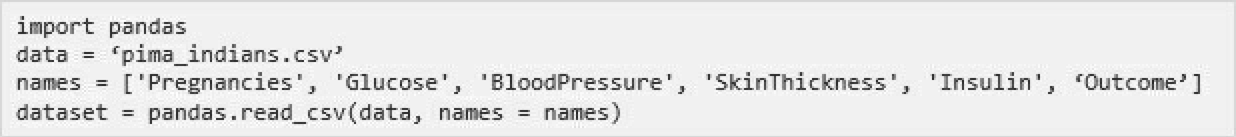
Upon leaving, you will notice that the cartography is perfectly preserved.

Data analysis

This section details the analysis of data in machine learning in Python

Loading the dataset

We can download data directly from the UCI Machine Learning repository. Note that we use pandas here to load the data. We will also use pandas to explore the data, both with descriptive statistics and data visualization. Note the following code and note that we specify the names of each column when loading the data.



```
import pandas
data = ‘pima_indians.csv’
names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', ‘Outcome’]
dataset = pandas.read_csv(data, names = names)
```

After running the code, you may notice that the dataset is loaded and ready for analysis. Here we downloaded the pima_indians.csv file, moved it to our working directory and loaded it using the local file name.

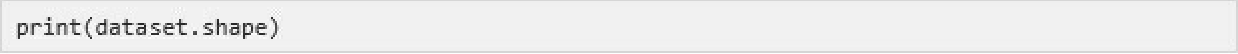
Summarize the dataset

In summary, the data can be done in several ways, as follows:

- Check the dimensions of the dataset
- Full data list
- View the statistical summary of all attributes
- Splitting of data by class variable

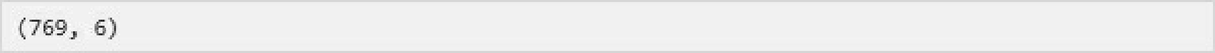
Dimensions of the dataset

You can utilize the below command to check the number of instances (rows) and attributes (columns) that the data have with the shape property.



```
print(dataset.shape)
```

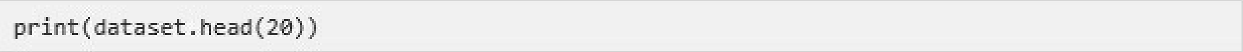
Then, concerning the code we have talked about, we can see 759 instances and six attributes –



```
(769, 6)
```

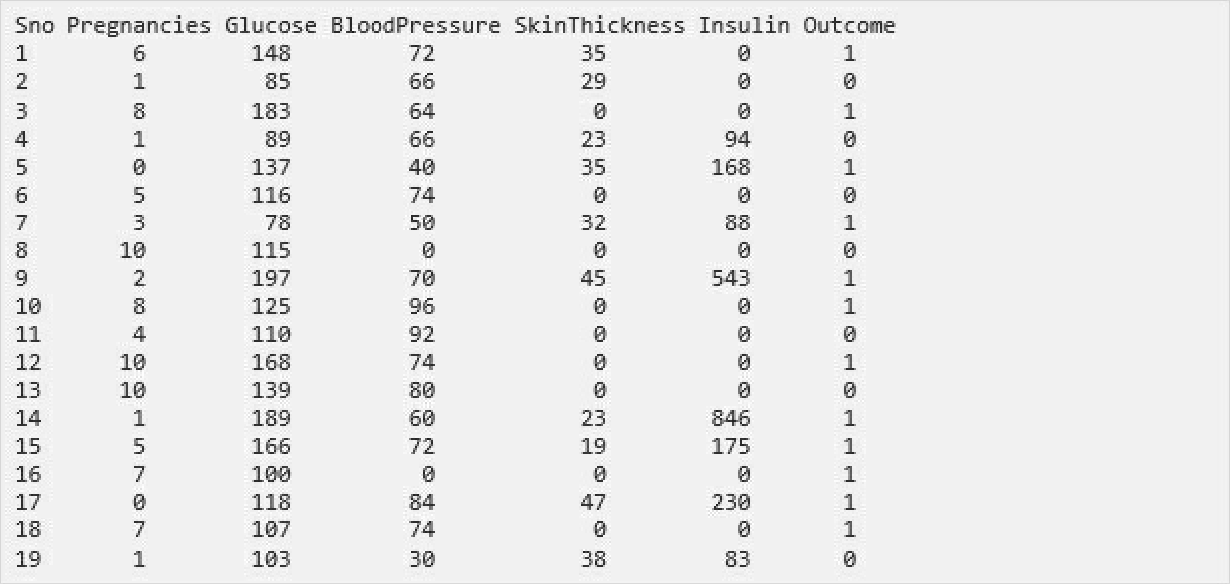
List the Entire Data

You can check out the entire data and understand its summary –



```
print(dataset.head(20))
```

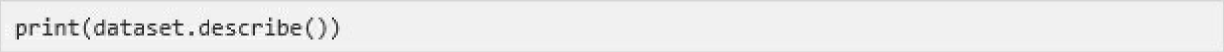
This command prints the first 30 rows of the data as shown –



```
Sno Pregnancies Glucose BloodPressure SkinThickness Insulin Outcome
1        6        148         72           35          0       1
2        1         85         66           29          0       0
3        8        183         64            0          0       1
4        1         89         66           23         94       0
5        0        137         40           35        168       1
6        5        116         74            0          0       0
7        3         78         50           32         88       1
8       10        115          0            0          0       0
9        2        197         70           45        543       1
10       8        125         96            0          0       1
11       4        110         92            0          0       0
12      10        168         74            0          0       1
13      10        139         80            0          0       0
14       1        189         60           23        846       1
15       5        166         72           19        175       1
16       7        100          0            0          0       1
17       0        118         84           47        230       1
18       7        107         74            0          0       1
19       1        103         30           38         83       0
```

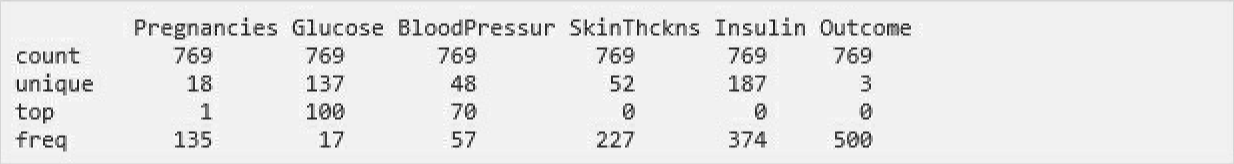
See the statistical summary

You can view the statistical summary for each attribute, including count, unique, top, and freq, using the command below.



```
print(dataset.describe())
```

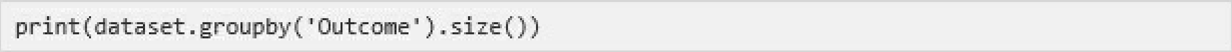
This command provides you with the below output that shows each attribute's statistical summary. –



```
        Pregnancies Glucose BloodPressur SkinThckns Insulin Outcome
count       769       769       769          769       769     769
unique       18       137        48           52       187       3
top           1       100        70            0         0       0
freq        135        17        57          227       374     500
```

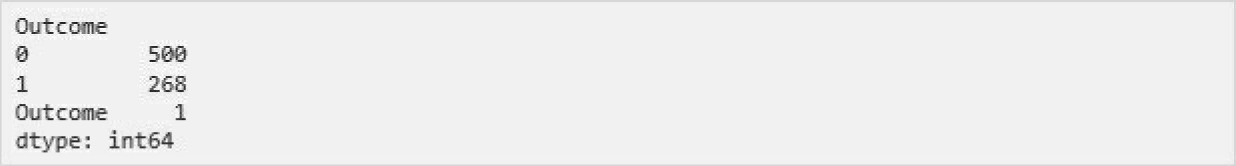
Breakdown the Data via Class Variable

You can additionally check the number of instances (rows) that relate to each result as an absolute number using the command specified here -



```
print(dataset.groupby('Outcome').size())
```

Then you are shown the number of the output of instances as shown –



```
Outcome
0         500
1         268
Outcome     1
dtype: int64
```

Data visualization

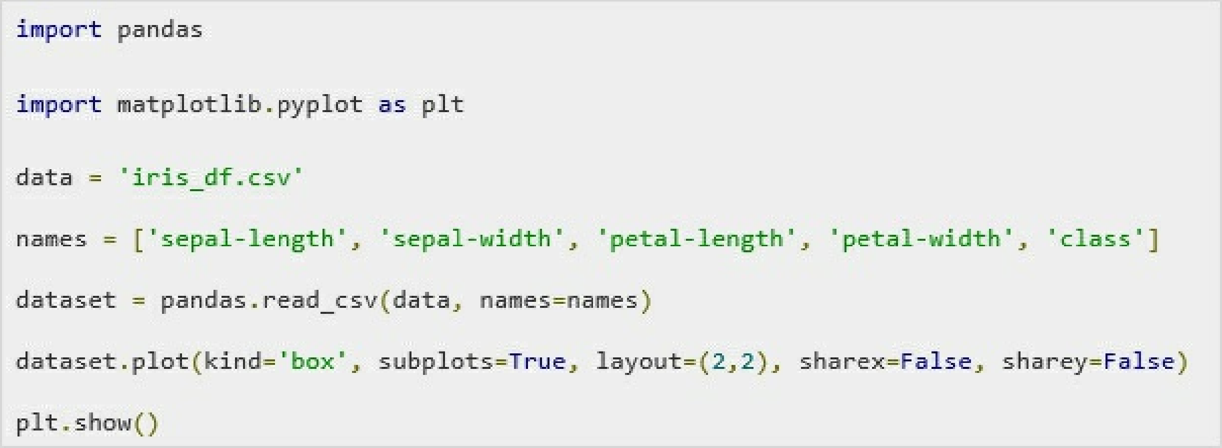
You can display the data using two types of graphics, as shown -

Univariate graphs to understand each attribute

Multivariate graphs to understand the relationships that exist between attributes

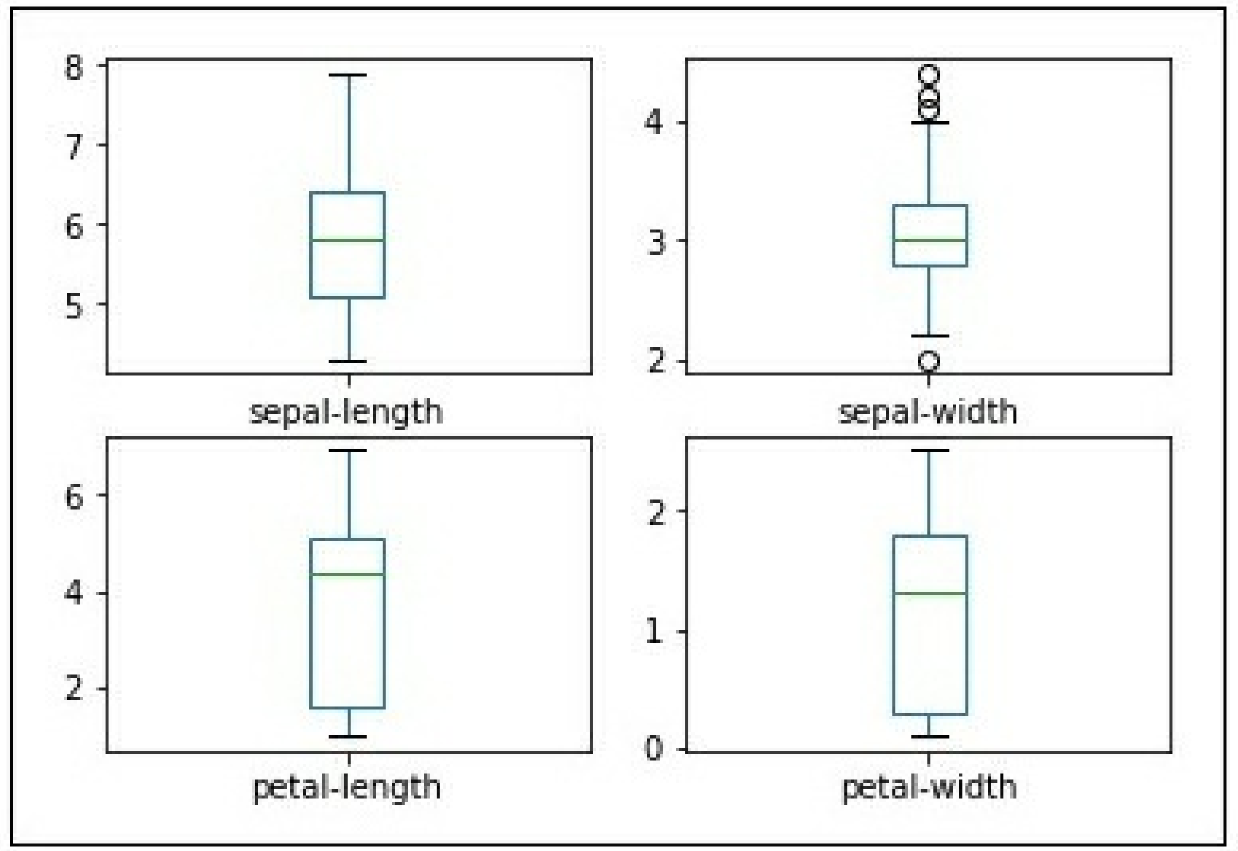
Univariate Plots

These are plots of each variable. Consider the case where input variables are numeric and where we need to create box and mustache graphics. The following code can be used for this purpose.



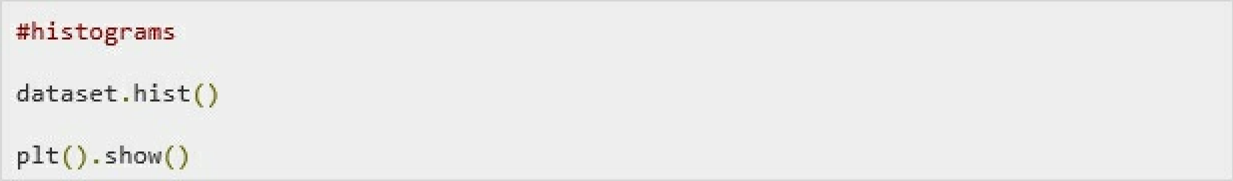
```
import pandas

import matplotlib.pyplot as plt

data = 'iris_df.csv'

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

dataset = pandas.read_csv(data, names=names)

dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)

plt.show()
```

The output can be seen with a clearer idea of the distribution of input attributes, as shown -

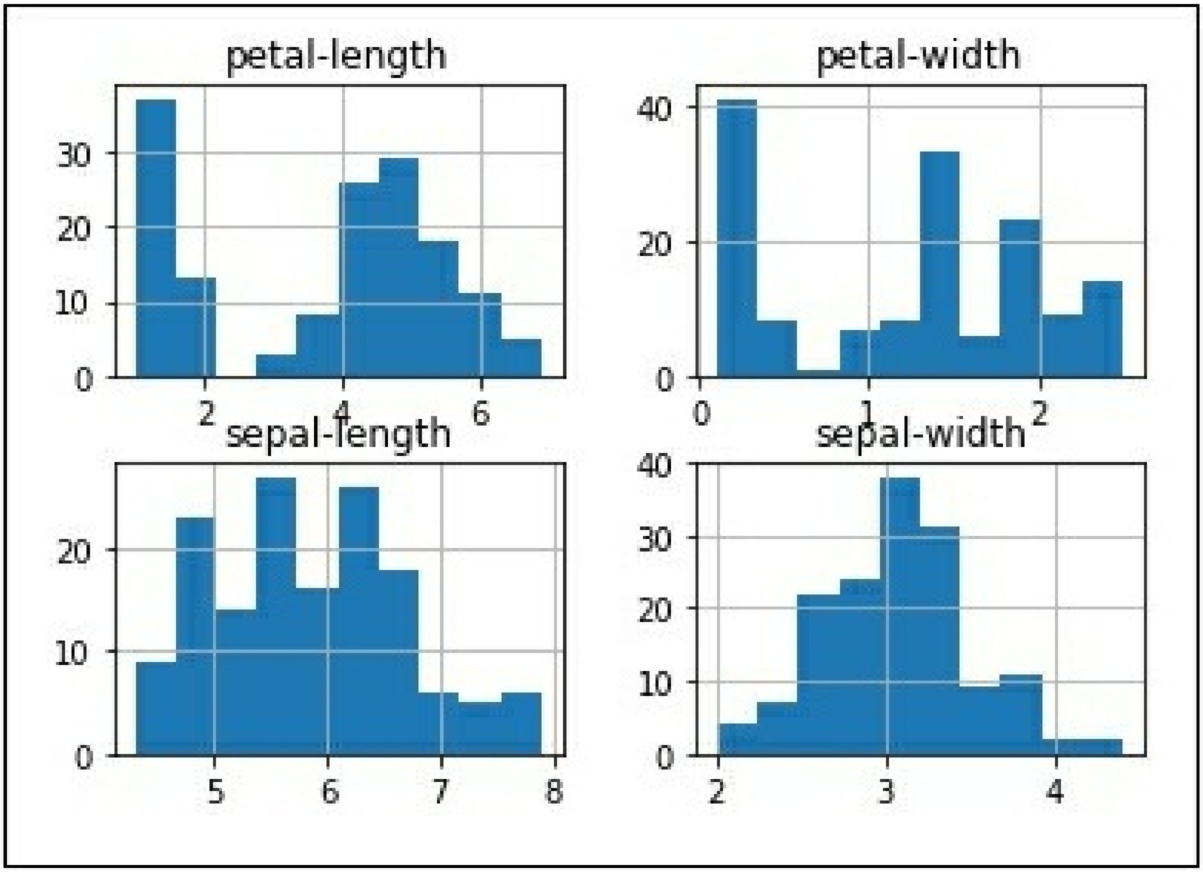


Box and Whisker Plots

A histogram of each input variable can be created to get the distribution idea by making use of the commands below –



```
#histograms

dataset.hist()

plt().show()
```



In the output, it can be seen that two of the input variables have a Gaussian distribution. Thus, these tables help to give an idea of the algorithms that we can use in our program.

Multivariate Plots

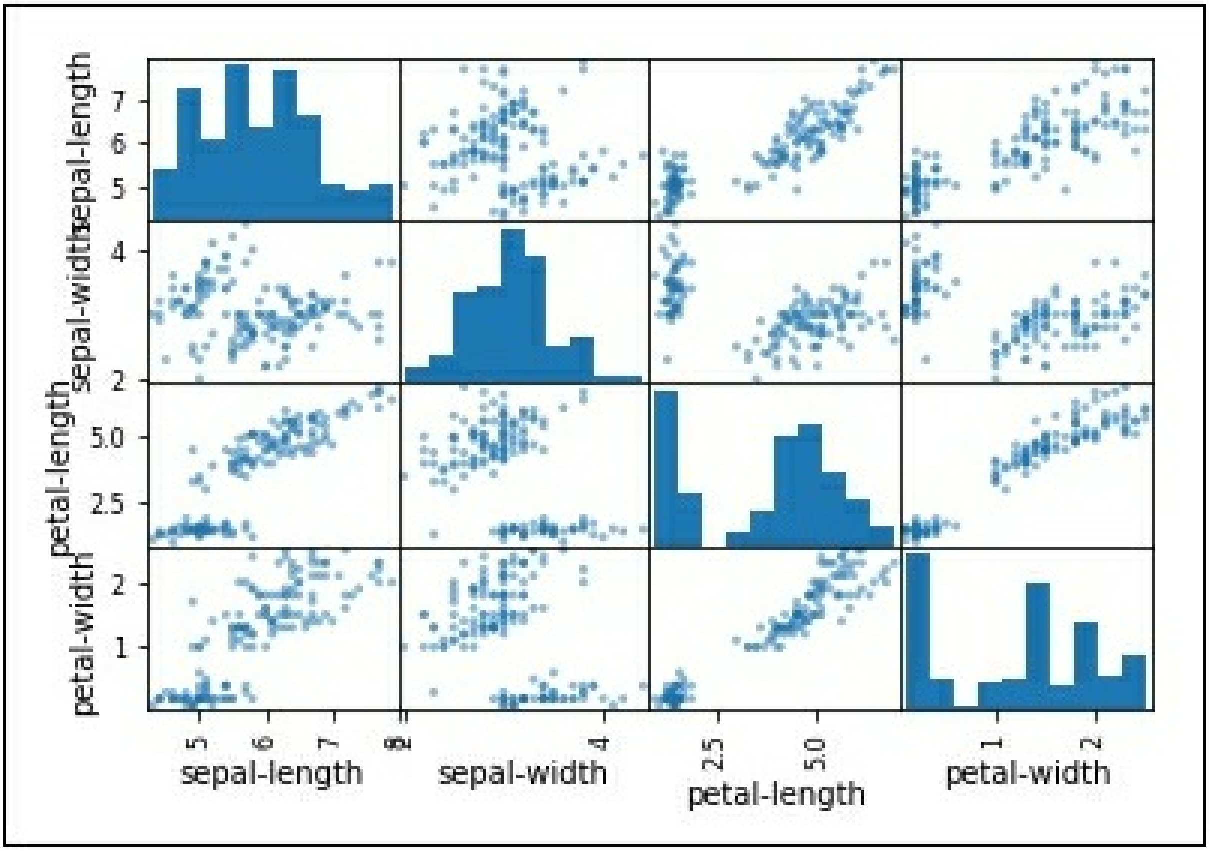
Multivariate graphs help us understand the interactions between variables.

Scatter Plot matrix

First, let's examine the scatter plots of all attribute pairs. This can be useful for identifying structured relationships between input variables.

```
from pandas.plotting import scatter_matrix
scatter_matrix(dataset)
plt.show()
```

You can observe the result as shown -



Note that in the output, there is a diagonal grouping of certain pairs of attributes. This indicates a strong correlation and a predictable relationship.

Training Data and Test Data

Learning data and test data are two important concepts of machine learning. This section discusses them in detail.

Training data

The observations of the learning set constitute the experience that the algorithm uses to learn. In supervised learning issues, each observation is made up of an observed output variable and one or more observed input variables.

Test data

The set of tests is a set of observations used to evaluate model performance using certain performance metrics. It is important that no observations of the training set are included in the test set. If the test suite contains examples of the training set, it will be difficult to evaluate if the algorithm has learned to generalize from the learning set or to memorize it.

A program which generalizes properly will be able to execute a task with new data efficiently. On the other hand, a program that memorizes learning data by learning a model that is too complex could accurately predict the response variable values for the training set, but can not predict the response variable value for new examples. The memorization of the drive is called over-adjustment. A program that memorizes your observations may not work properly because you can memorize relationships and structures that are noise or coincidences. Memory balancing and generalization, or excessive tuning and low tuning, is a common problem for many machine learning algorithms. The setting can be applied to many models to reduce over-adjustment.

In addition to the learning and testing data, the third set of observations, called validation or waiting, is sometimes necessary. The validation set is used to adjust variables called hyperparameters, which control how the model is learned. The program is always evaluated in the test suite to provide an estimate of its performance in the real world; its performance in the validation suite should not be used as an estimate of real-world model performance, as the program has been specifically adapted to validation data. It is mostly done to partition a single set of supervised observations into learning, validation, and test sets. Partition size is not required and may vary depending on the amount of data available. It is common to allocate 50% or more of the data for the learning set, 25% for the test set, and the rest of the data for the validation set.

Some training sets may include only a few thousand observations, while others may include millions. Cheap storage, increased network connectivity, ubiquitous sensor-based smartphones, and changing mindsets for privacy have all contributed to the current status of Big Data or training packages containing millions or billions of examples.

However, machine learning algorithms also track the maximum "waste input." A student who is studying for a test reading a big and confusing book that contains many errors will probably not do better than a student reading a small but well-written book. Similarly, an algorithm formed on a large collection of noisy, irrelevant, or mislabeled data will not work better than an algorithm-driven on a smaller dataset more representative of real problems.

Many supervised pieces of training are prepared manually or through semi-automated processes. Creating an enormous collection of supervised data can be expensive in some areas. Fortunately, several datasets are bundled with scikit-learn, which allows developers to focus on experimenting with models.

During development, and especially when learning data is scarce, a practice called cross-validation can be adopted to train and validate the algorithm on the same data. In cross-validation, the training data is partitioned. The algorithm is formed using all but one partition and tested on the remaining partition. The partitions will be rotated several times so that the algorithm is formed and evaluated on all the data.

Suppose, for example, that the original dataset is partitioned into five subsets of equal size, labeled from A to E. Initially, the model is formed on partitions B and E and tested on partition A. To the next repetition, the model is: trained on partitions A, B, D and E and tested on partition C. The partitions are rotated until the models are trained and tested on all partitions. Cross-validation provides a more accurate estimate of model performance than testing a single data partition.

Performance measures - Bias and Variance

Many indicators can be used to determine if a program is learning to do its job more efficiently. For supervised learning problems, many performance measures measure the number of prediction errors.

There are two primary causes of prediction error for a model: bias and variance. Suppose you have many unique training packages, but also representative of the population. A high biased model will produce similar errors for entry, regardless of the training game with which it was formed; the model polarizes its assumptions about the real relationship with the relationship demonstrated in the learning data. A high variance model, on the other hand, will produce different errors for entry, depending on the training set with which it was formed. A high bias model is inflexible, but a high variance model can be so flexible that it can model the noise in the training set. In other words, a model with a high variance corresponds to the training data, while a high bias model corresponds to the training data.

Ideally, a model will have both small bias and a small variance, but efforts to reduce one will often increase the other. This is called the bias-variance tradeoff. We may need to consider the bias-variance offsets of various models presented in this tutorial. Unsupervised learning issues do not have an error signal to measure; Instead, performance metrics for unsupervised learning problems measure some attributes of the discovery structure in the data. Most performance measures can only be performed for a specific type of task.

Machine learning systems ought to be assessed by making use of performance measures that represent the cost of actual errors. Although this seems trivial, the following example illustrates the use of a performance measure appropriate for the overall task, but not for the specific application.

Accuracy, precision, and recall

Let’s consider a classification task where a machine learning system detects tumors and must determine whether these tumors are benign or malignant. Precision, or the fraction of correctly ranked instances, is an obvious measure of program performance. Although accuracy does not measure program performance, it does not distinguish between malignant tumors classified as benign and benign tumors classified as malignant. In some applications, the costs generated for all types of errors may be the same. In this problem, however, not identifying malignancies is a more serious mistake than classifying benign tumors as malignant by mistake.

Each of the possible prediction results can be measured to create different snapshots of classifier performance. After the system has correctly classified a tumor as malignant, the prediction is called true positive. When the system has incorrectly classified a benign tumor as malignant, the prediction is a false positive. Similarly, a false negative is a wrong prediction that the tumor is benign, and a true negative is the right prediction that a tumor is benign. These four results can be used to calculate several common measures of classification performance, such as accuracy, precision, recall, and so on.

The precision is calculated with the following formula -

$$ACC = (TP + TN) / (TN + TP + FN+ FP)$$

Where TN stands for the number of true negatives

TP is the number of true positives

FN is the number of false negatives.

FP is the number of false positives

Accuracy is the fraction of malignant and truly malignant predicted tumors. The precision is calculated with the following formula -

$$PREC = TP / (TP + FP)$$

Remember that this is the fraction of malignant tumors identified by the system. The recall is calculated with the following formula -

$$R = TP / (TP + FN)$$

In this example, accuracy measures the fraction of tumors that are presumed to be malignant and, in fact, malignant. The booster measures the fraction of truly malignant tumors detected. Precision and recall measurements can reveal that a classifier with impressive accuracy cannot detect most malignant tumors. If most tumors are non-threatening, even a classifier who never predicts malignancy can have great accuracy. A different classifier with less precision and a larger reminder may be more suited to the task as it will detect more malignant tumors. Many other performance measures for ranking can also be used.

CHAPTER FIVE

TECHNIQUES AND ALGORITHMS

Techniques Used In Machine Learning

In this chapter, we explain in detail each of the techniques used in machine learning.

Classification

Classification is an automatic learning technique that uses known data to determine how to classify new data into a set of existing categories.

Consider the following examples to understand the classification technique -

A credit card company accepts tens of thousands of applications for new credit cards. These apps contain information about different characteristics such as age, location, gender, annual salary, credit history, and so on. The algorithm's task here is to classify card applicants such as those with a good credit history, a bad credit history and those with a mixed credit history.

In a hospital, the emergency department has more than 15 characteristics (age, blood pressure, heart disease, the severity of illness, etc.) to analyze before deciding whether a particular patient should be placed in an intensive care unit in the hospital because it is an expensive proposition and only those patients who can survive and bear the costs are considered a top priority. The problem here is to classify patients as high-risk, low-risk patients based on available characteristics or parameters.

When classifying a given dataset, the classifier carries out the below actions:

Firstly, a new data model is prepared using one of the learning algorithms.

Then, the prepared data model is tested.

Subsequently, this data model is used to examine the new data and determine their class.

Classification, also known as categorization, is an automatic learning technique that uses known data to determine how to classify new data into an existing set of labels /categories/classes.

In classification issues, the program must learn to predict discrete values for dependent variables or outputs of one or more independent or input variables. In other words, the program must provide the class, category, or label most likely to be used for new observations. Classification applications include predicting whether a day will rain or not, or predicting whether the stock price of a given company will go up or down, or whether an item belongs to the sports or recreation section.

Classification is a form of supervised learning. Mail service providers, such as Gmail, Yahoo, and others, use this technique to classify a new email as spam or not. The classification algorithm is information; the classifier decides whether a new email should be sent to the inbox or the spam folder.

Applications of Classification

Credit card fraud detection - The sorting method is used to predict credit card fraud. Using historical past fraud records, the classifier can predict which future transactions may turn into a fraud.

Email spam - Depending on the features of previous spam, the classifier determines if a new email should be sent to the spam folder.

Naive Bayes classifier technique

Classification techniques include the Naive Bayes classifier, which is a simple technique for constructing classifiers. It is not an algorithm to train these classifiers, but a group of algorithms. A Bayes classifier builds models to classify problem instances. These assessments are made using available data.

An important feature of the naive Bayes classifier is that it requires little training data to estimate the parameters required for classification. For some types of models, Bayes naive classifiers can be trained very effectively in a supervised learning environment.

Despite their simplistic assumptions, Bayes naive classifiers have worked effectively in many complex real-world circumstances. These have worked well in document sorting and spam filtering.

The regression

In the regression, the program predicts the value of continuous or variable output response. Regression problems include forecasting sales of a new product or salary for a job based on what it entails. Similar to classification, regression issues require supervised learning. In regression tasks, the program guesses the value of an output variable or a continuous response from the input or explanatory variables.

Recommendation

The recommendation is a popular method that provides rough recommendations based on user information, such as purchase history, ratings, and clicks. Amazon and Google use this method to display recommended item lists for their users, based on info from their previous actions. There are recommendation engines that operate in the background to capture the user's behavior and recommend the selected items based on the user's previous actions. Facebook also uses the recommendation method to identify and recommend people and send suggestions from friends to their users.

A recommendation mechanism is a model that is of interest to a user based on their previous registration and behavior. When applied in the context of movies, it becomes a mechanism for recommending movies. We filter the elements of the movie database by predicting how a user can rate them. This helps us connect users with the right content of the movie database. This method is useful in two ways: If you have a massive database of movies, you may or may not find content that matches your choices. Also, by recommending relevant content, we can increase consumption and attract more users.

Amazon Prime, Netflix, and similar movie rental firms depend heavily on sponsorship mechanisms to retain their users. Typically, recommendation engines produce a list of recommendations using collaborative filtering or content-based filtering. The difference between the two types lies in the way the recommendations are extracted. Collaborative filtering builds a model based on the past behavior of the current user, as well as the ratings assigned by other users. This model is used to predict what might interest this user.

In contrast, content-based filtering uses the functionality of the element to recommend more items to the user. The similarity between the articles is the main motivation here. Collaborative filtering is often used more often in such recommendation methods.

Clustering

The associated groups of observations are known as clusters. A general unsupervised learning duty is to find clusters in the training data.

We can also define grouping as a way to organize elements of a particular collection into groups based on similar features. For example, online news publishers group their articles using clustering.

Applications of Clustering

Clustering has applications in many areas, such as market research, pattern recognition, data analysis, and image processing. As discussed here -

Helps marketers to find out distinct groups in their customer base and characterize their customer groups based on their buying habits.

In biology, it is often used to derive taxonomies of plants and animals, to classify genes with similar functionality and to better understand the structures inherent in populations.

It helps identify similar land use areas in an Earth observation database.

Helps sort documents on the web for information discovery.

It is used in different detection applications such as credit card fraud detection.

Cluster Analysis serves as a data mining tool to obtain information about the distribution of data to observe the characteristics of every cluster.

The task, called cluster analysis or clustering, assigns observations to groups so that observations within groups are more similar in terms of a similarity measure than observations from other groups.

The cluster is often used to explore a dataset. For example, considering a collection of movie reviews, a grouping algorithm can discover sets of positive and negative reviews. The system can not label clusters as "positive" or "negative"; without supervision, it will be appreciated that grouped observations are similar to each other, to a certain extent. A common application of clustering is to discover customer segments within a market for a product. By understanding which attributes are common to particular customer groups, marketers can choose which aspects of their campaigns to highlight. The cluster is also used by Internet radio services; For example, with a set of songs, a grouping algorithm can group songs according to their genre. Using different similarity measures, the same grouping algorithm can group songs according to their keys or the instruments they contain.

Unsupervised learning tasks include grouping, in which observations are organized into groups based on similar resources. Clustering is used to form clusters or groups of similar data based on common characteristics.

Grouping is a form of unsupervised learning. Search engines like Bing, Google, and Yahoo! Use grouping techniques to group data with similar characteristics. Newsgroups use grouping techniques to group multiple articles based on related topics.

The grouping mechanism completely traverses the input data and, depending on the characteristics of the data, determines the cluster to be grouped. The following points can be observed when grouping -

An appropriate cluster algorithm must be selected to group the elements of a cluster.

A rule is needed to check the similarity between newly found items and group items.

A stop condition is required to define the point at which no clustering is required.

Cluster types

There are two types of clustering: the simple cluster and hierarchical clustering.

The simple cluster creates a simple set of clusters with no clear structure that can link clusters to each other. Hierarchical clustering creates a hierarchy of clusters. Hierarchical clustering provides an output cluster hierarchy, a structure that generates more information than the unstructured cluster returned by simple clustering. Hierarchical clustering does not require that we specify the number of clusters in advance. The advantages of hierarchical grouping are less profitable.

In general, we select simple clustering when efficiency is important and hierarchical clustering when one of the possible problems of simple clustering is a problem. Also, many researchers believe that hierarchical clusters produce better clusters than single clusters.

Cluster Algorithms

You need clustering algorithms to group a given piece of data. Two algorithms are frequently used: clustering of the canopy and clustering of K-media.

The canopy grouping algorithm is an unsupervised pre-classification algorithm often used as a pre-processing step for the K-means algorithm or hierarchical classification algorithm. It is used to speed up cluster operations on large data sets where the use of another algorithm may not be possible due to the large size of the data sets.

K-means clustering is an important clustering algorithm. The k-means clustering algorithm represents the number of clusters in which the data will be divided. For example, if the value of k specified in the algorithm is 3, this algorithm will split the data into 3 clusters.

Each object is signified as a vector in space. Firstly, k points are randomly selected by the algorithm and treated as centers, with each object closest to each center grouped. The k-means algorithm requires input vector files, so we need to create vector files. After creating the vectors, we proceed with the k-means algorithm.

ALGORITHMS

We can classify Machine learning algorithms into two types: supervised and unsupervised. This chapter discusses them in detail.

Supervised learning

The algorithm includes a target or predicted outcome or variable dependent on a given set of independent variables or independent variables. Using this set of variables, we generate a function that maps the input variables to the desired output variables. The learning process continues until the model reaches the desired level of precision in the training data.

Examples of supervised learning - KNN, regression, logistic regression, decision tree, random forest, etc.

Unsupervised learning

In Unsupervised learning algorithm, there is no objective or result or dependent variable to predict or estimate. It is used to group a given dataset into different groups, which is widely used to target clients in different groups for a specific intervention. The apriori algorithm and the K means are some examples of an unsupervised learning algorithm.

Reinforcement learning

Making use of this algorithm, the machine is taught to make specific decisions. Here, the algorithm is continuously formed using test and error methods and return methods. This machine picks up from past experiences and tries to capture the best knowledge possible for making accurate business decisions.

An example of reinforcement learning is the Markov decision process.

List of some machine learning algorithms

Here is the list of generally used machine learning algorithms which can be applied to almost any data issue -

- Linear regression
- Logistic regression
- Decision tree
- SVM
- Naive Bayes
- KNN
- K-Means
- Random Forest
- Dimensionality reduction algorithms
- Gradient Boosting algorithms such as GBM, XGBoost, LightGBM and CatBoost

Linear regression

Linear regression is used to estimate real-world values such as housing cost, number of calls, total sales, etc., as a function of continuous variables. Here, we establish relationships between the dependent and independent variables, adjusting the best line. This line of best fit is called the regression line and is represented by the linear equation Y = a * X + b.

In this equation -

Y - Dependent variable

a - inclination

X - independent variable

b - Interception

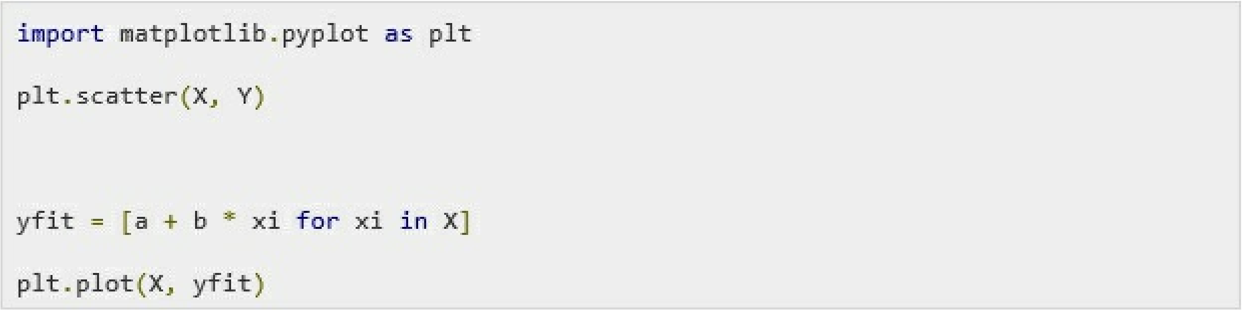
These coefficients(a & b) are derived based on the minimization of the sum of the quadratic difference of the distance between the data points and the regression line.

Example

The best way to know linear regression is to take an example. Suppose we are asked to organize students in a class in ascending order of their weights. By looking at students and visually analyzing their heights and constructions, we can organize them according to need, using a combination of these parameters, namely height and compilation. This is an example of a linear regression of the real world. We found that height and construction correlate with the weight of a relationship, which is similar to the equation above.

Types of linear regression

Linear regression is mainly of two types - simple linear regression and multiple linear regression. Simple linear regression is characterized by an independent variable, whereas multiple linear regression is characterized by more than one independent variable. By finding the best-fitting line, you can adjust a curvilinear or polynomial regression. You can make use of the code below for this purpose.

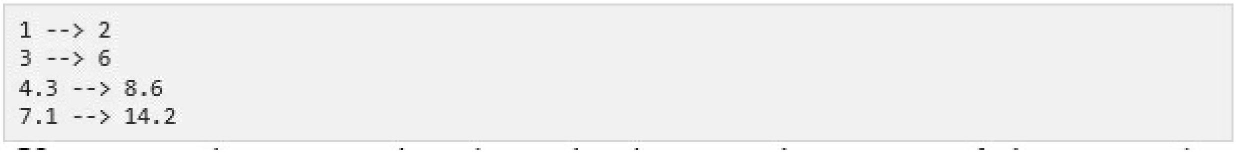


```
import matplotlib.pyplot as plt

plt.scatter(X, Y)



yfit = [a + b * xi for xi in X]

plt.plot(X, yfit)
```

Constructing a linear regressor

Regression involves the process of assessing the relationship between input data and continuous value output data. These data are typically in the form of real numbers, and our goal is to estimate the underlying function that controls the mapping from input to output.

Consider this mapping between input and output as shown –



```
1 --> 2
3 --> 6
4.3 --> 8.6
7.1 --> 14.2
```

You can easily estimate the relationship between inputs and outputs by analyzing the model. We can see that the output is twice the input value in each case so that the transformation would be - f (x) = 2x

Linear regression refers to the estimation of the relevant function using a linear combination of input variables. The previous example was an example composed of an input variable and an output variable.

Linear regression aims to remove the important linear model that links the input variable to the output variable. This minimizes the sum of squares of differences between the actual output and the expected output using a linear function. This method is called ordinary least squares. You may assume that a curvilinear line is best for these points, but linear regression does not allow it. The main benefit of linear regression is that it is not complex. You can also find more accurate models in nonlinear regression, but they will not be quick. Here, the model attempts to approximate the input data points by making use of a straight line.

Let's see how to build a linear regression model in Python.

Consider that you received a data file called data_singlevar.txt. Contains the lines that separated with commas, where the first component is the input value, and the second component is the output value that corresponds to this input value. You should use this as an input argument -

Assuming the best fit line for a set of points is:

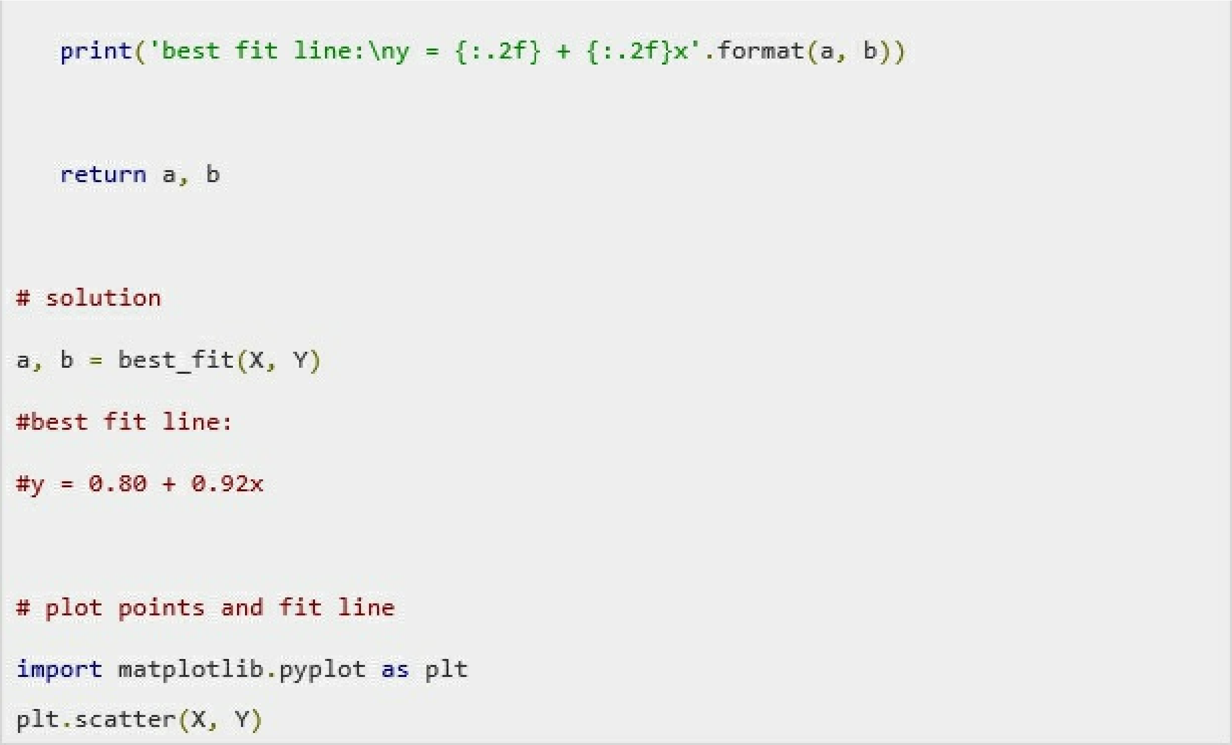
y = c + d * x

where d = (sum (xi * yi) - n * xbar * ybar) / soma ((xi - xbar) ^ 2)

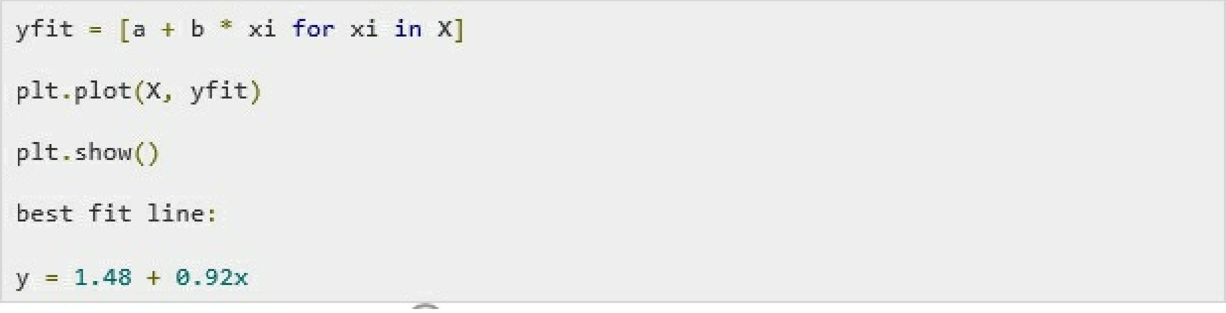
c = ybar - d * xbar

Use the following code for this purpose –

```
# sample points
X = [0, 6, 11, 14, 22]
Y = [1, 7, 12, 15, 21]

# solve for a and b
def best_fit(X, Y):
   xbar = sum(X)/len(X)
   ybar = sum(Y)/len(Y)
   n = len(X) # or len(Y)

   numer = sum([xi*yi for xi,yi in zip(X, Y)]) - n * xbar * ybar
   denum = sum([xi**2 for xi in X]) - n * xbar**2

   b = numer / denum
   a = ybar - b * xbar
```

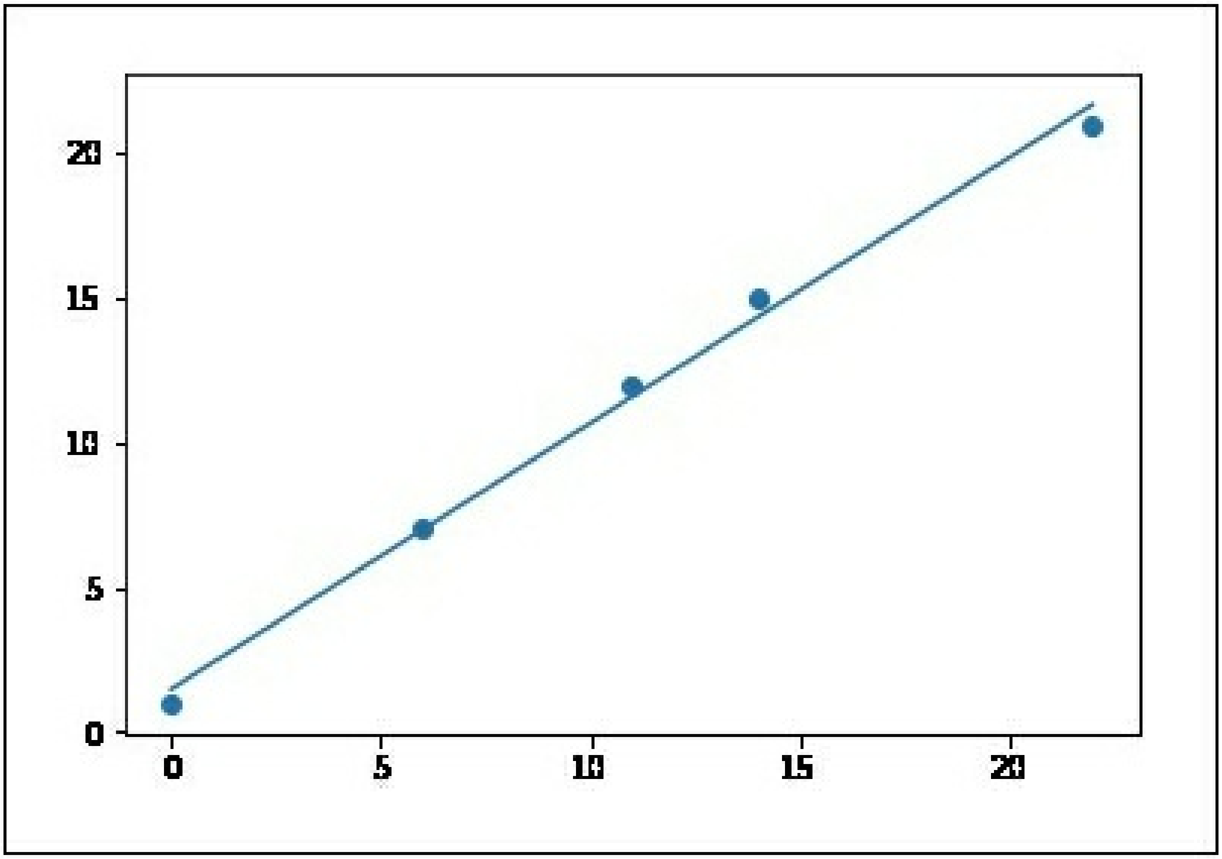


```
    print('best fit line:\ny = {:.2f} + {:.2f}x'.format(a, b))

    return a, b

# solution
a, b = best_fit(X, Y)
#best fit line:
#y = 0.80 + 0.92x

# plot points and fit line
import matplotlib.pyplot as plt
plt.scatter(X, Y)
```



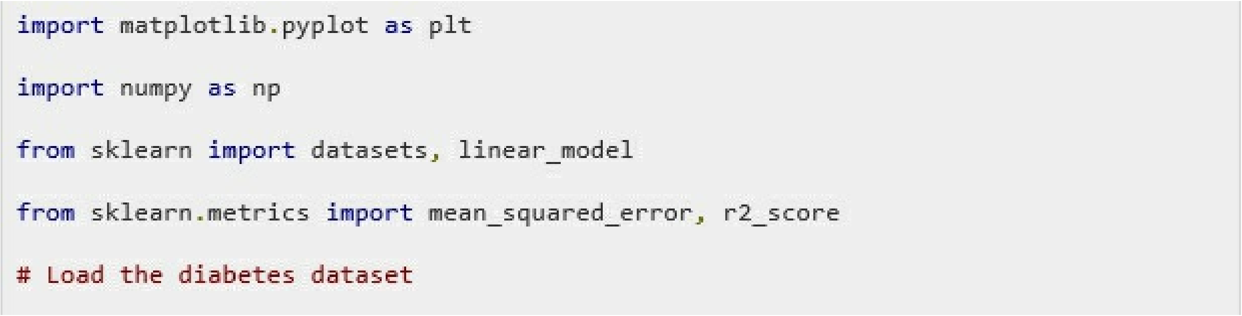
```
yfit = [a + b * xi for xi in X]
plt.plot(X, yfit)
plt.show()
best fit line:
y = 1.48 + 0.92x
```

If you run the code above, you can observe the output graph as shown –

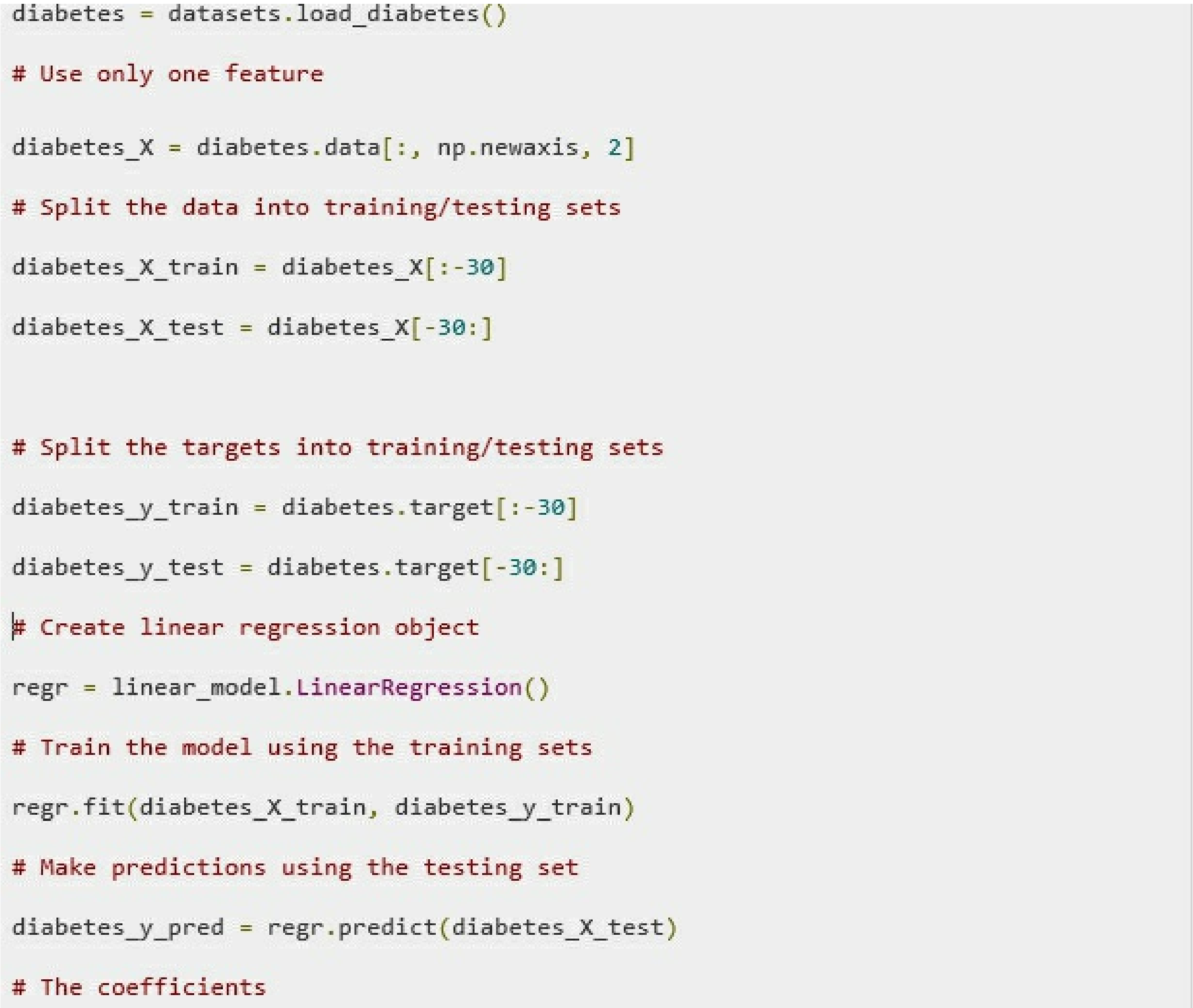


Note that this example makes use of only the first feature of the diabetes dataset to illustrate a two-dimensional graph of this regression method. The straight line is seen in the graph showing how linear regression attempts to draw a straight line that best minimizes the residual sum of squares between observed responses in the dataset and predicted linear responses.

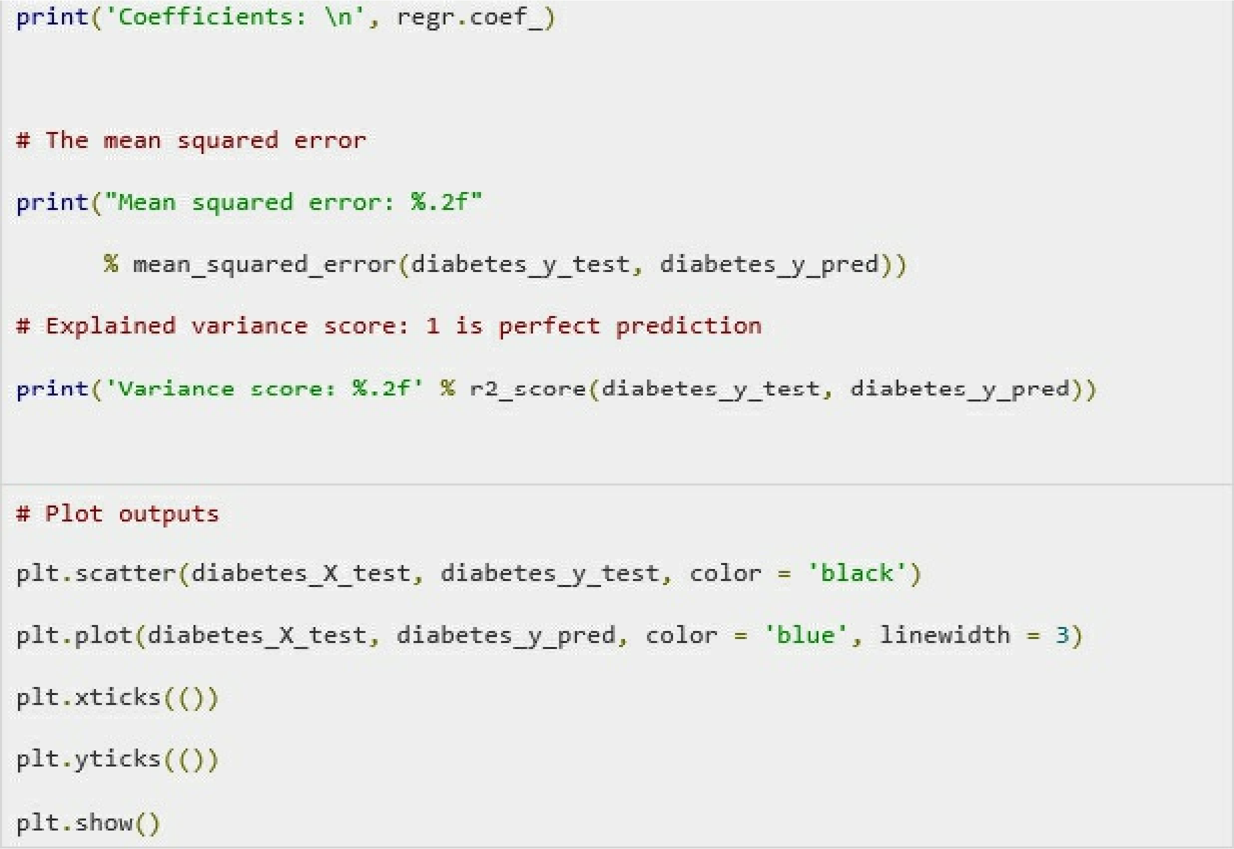
The coefficients can be calculated, the residual sum of the squares and the variance score using the program code below –



```
import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets, linear_model

from sklearn.metrics import mean_squared_error, r2_score

# Load the diabetes dataset
```



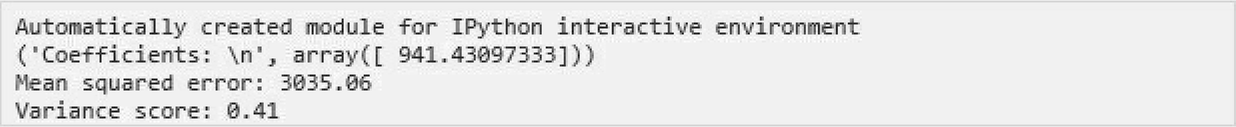
```
diabetes = datasets.load_diabetes()

# Use only one feature


diabetes_X = diabetes.data[:, np.newaxis, 2]

# Split the data into training/testing sets

diabetes_X_train = diabetes_X[:-30]

diabetes_X_test = diabetes_X[-30:]



# Split the targets into training/testing sets

diabetes_y_train = diabetes.target[:-30]

diabetes_y_test = diabetes.target[-30:]

# Create linear regression object

regr = linear_model.LinearRegression()

# Train the model using the training sets

regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set

diabetes_y_pred = regr.predict(diabetes_X_test)

# The coefficients
```



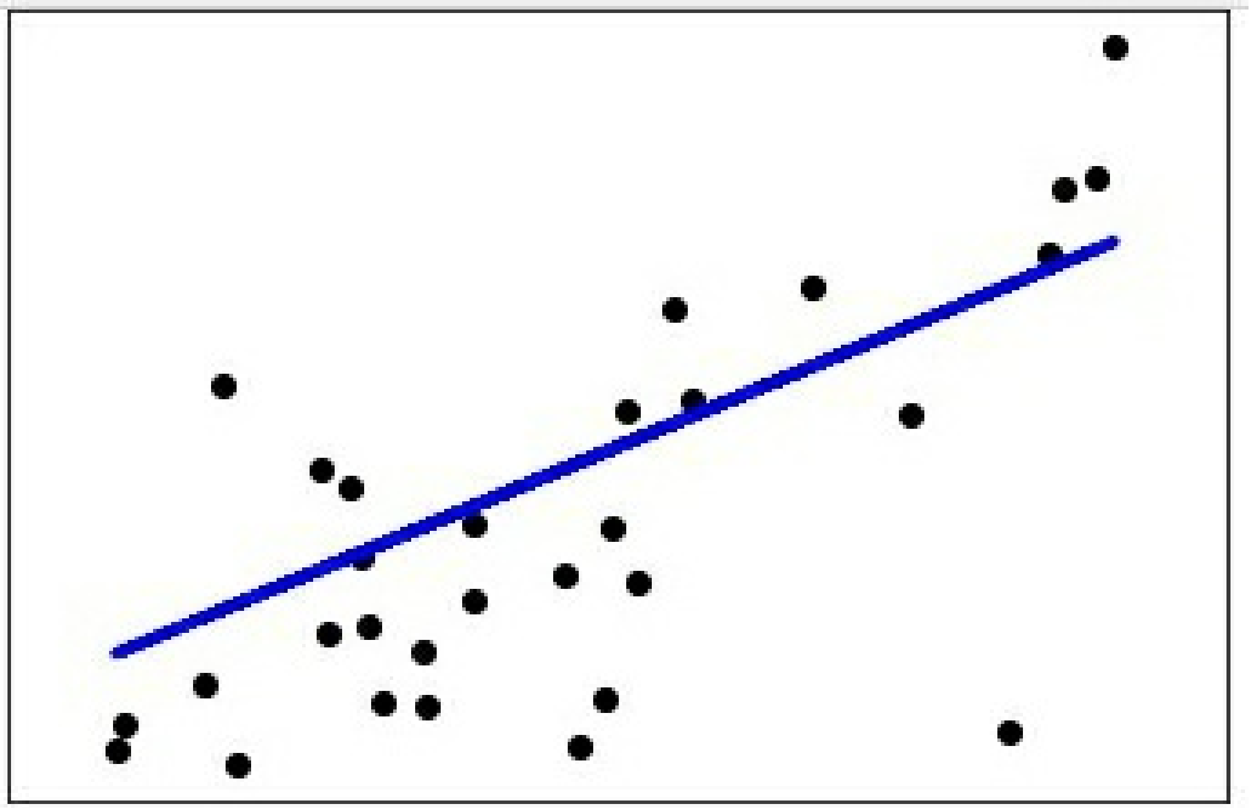
```
print('Coefficients: \n', regr.coef_)


# The mean squared error

print("Mean squared error: %.2f"

      % mean_squared_error(diabetes_y_test, diabetes_y_pred))

# Explained variance score: 1 is perfect prediction

print('Variance score: %.2f' % r2_score(diabetes_y_test, diabetes_y_pred))
```

```
# Plot outputs

plt.scatter(diabetes_X_test, diabetes_y_test, color = 'black')

plt.plot(diabetes_X_test, diabetes_y_pred, color = 'blue', linewidth = 3)

plt.xticks(())

plt.yticks(())

plt.show()
```

You can observe the following result after executing the code given above –



```
Automatically created module for IPython interactive environment
('Coefficients: \n', array([ 941.43097333]))
Mean squared error: 3035.06
Variance score: 0.41
```



Logistic regression

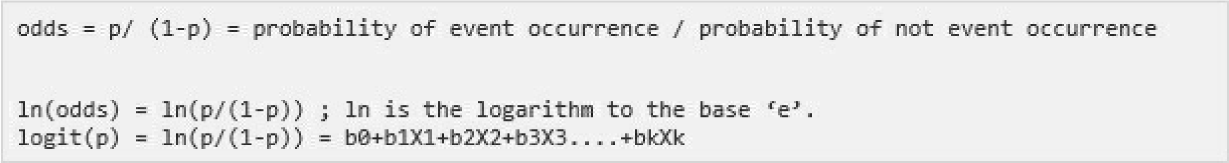
Logistic regression is another technique borrowed from machine learning from statistics. This is the preferred method for binary classification problems, that is, problems with two class values.

It's a classification algorithm and not a regression algorithm, as its name suggests. It is used to estimate values such as 0/1, Y / N, T / F, or discrete values depending on the set of independent variables. It predicts the probability of an event occurring by fitting the data to a logit function. Therefore, it is also called logit regression. As expected by the probability, its output values are between 0 and 1.

Example

Let's understand this algorithm through a simple example.

Suppose there is an enigma to solve that has only two result scenarios - either there is a solution or there is none. Now, suppose we have a wide variety of puzzles to test a person who is good on. The results may look like this: if a trigonometric riddle is given, a person can probably solve the problem at 80%. On the other hand, if a geographic puzzle is given, the person can only have a 20% probability of solving it. This is where logistic regression contributes to the solution. According to mathematics, the log probabilities of the result are expressed as a linear combination of the predictor variables.



```
odds = p/ (1-p) = probability of event occurrence / probability of not event occurrence


ln(odds) = ln(p/(1-p)) ; ln is the logarithm to the base ‘e’.
logit(p) = ln(p/(1-p)) = b0+b1X1+b2X2+b3X3....+bkXk
```

Take note that in p above is the probability of the presence of the characteristic of interest. He chooses parameters that maximize the probability of observing sample values, rather than minimizing the sum of square errors (as in ordinary regression).

Note that logging is one of the best mathematical ways to replicate a step function.

The following points can be remarkable when working on logistic regression -

This is similar to regression in that the objective is to find the values of the coefficients that weigh each input variable.

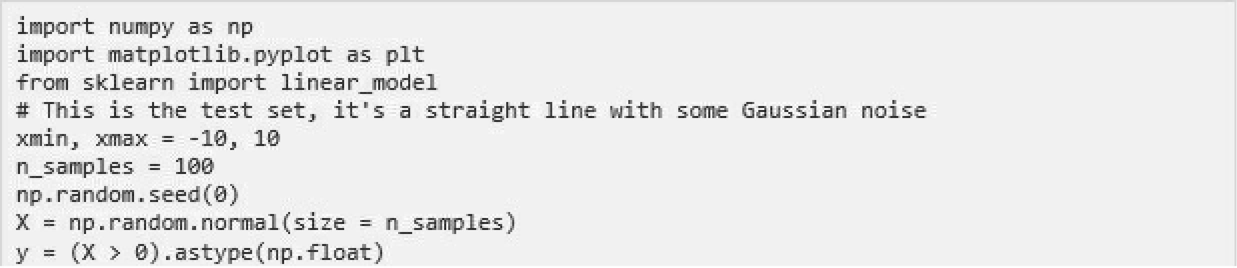
Not like the linear regression, the output prediction is obtained using a nonlinear function called the logistic function.

The logistics function appears as a large "S" and changes all values between 0 and 1. This is useful because a rule can be applied to the output of the logistics function to allocate values to 0 and 1 and expect a class value.

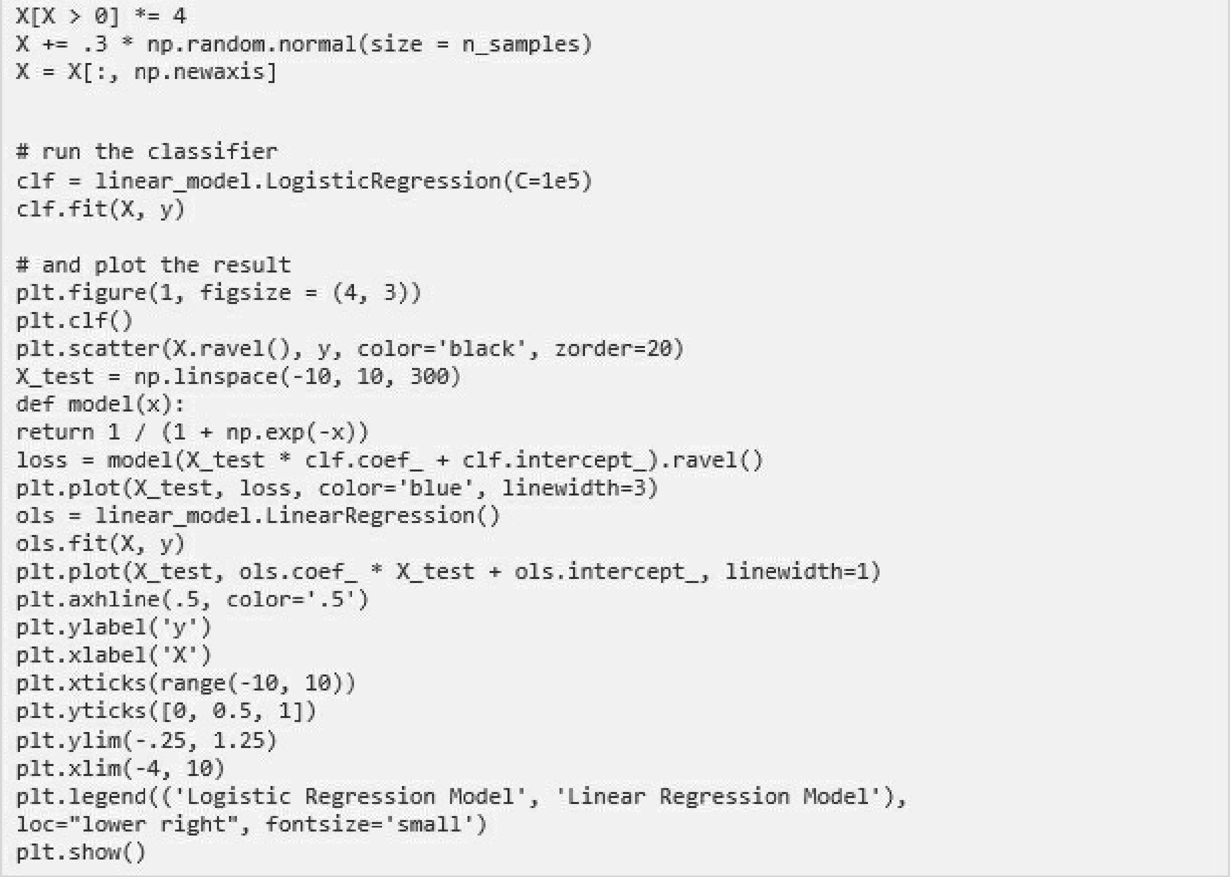
In the same way that the logistic regression model is learned, the predictions made by it can also be used as the probability that a particular data instance belongs to class 0 or class 1. This can be useful for problems in which you need to provide more reasoning for a forecast.

Like linear regression, logistic regression works best when unlinked attributes of the output variable are removed, and similar attributes are removed.

The following code illustrates how to develop a logistic expression graph in which a synthetic dataset is classified as 1 or 0, i.e., class one or two, using the logistic curve.

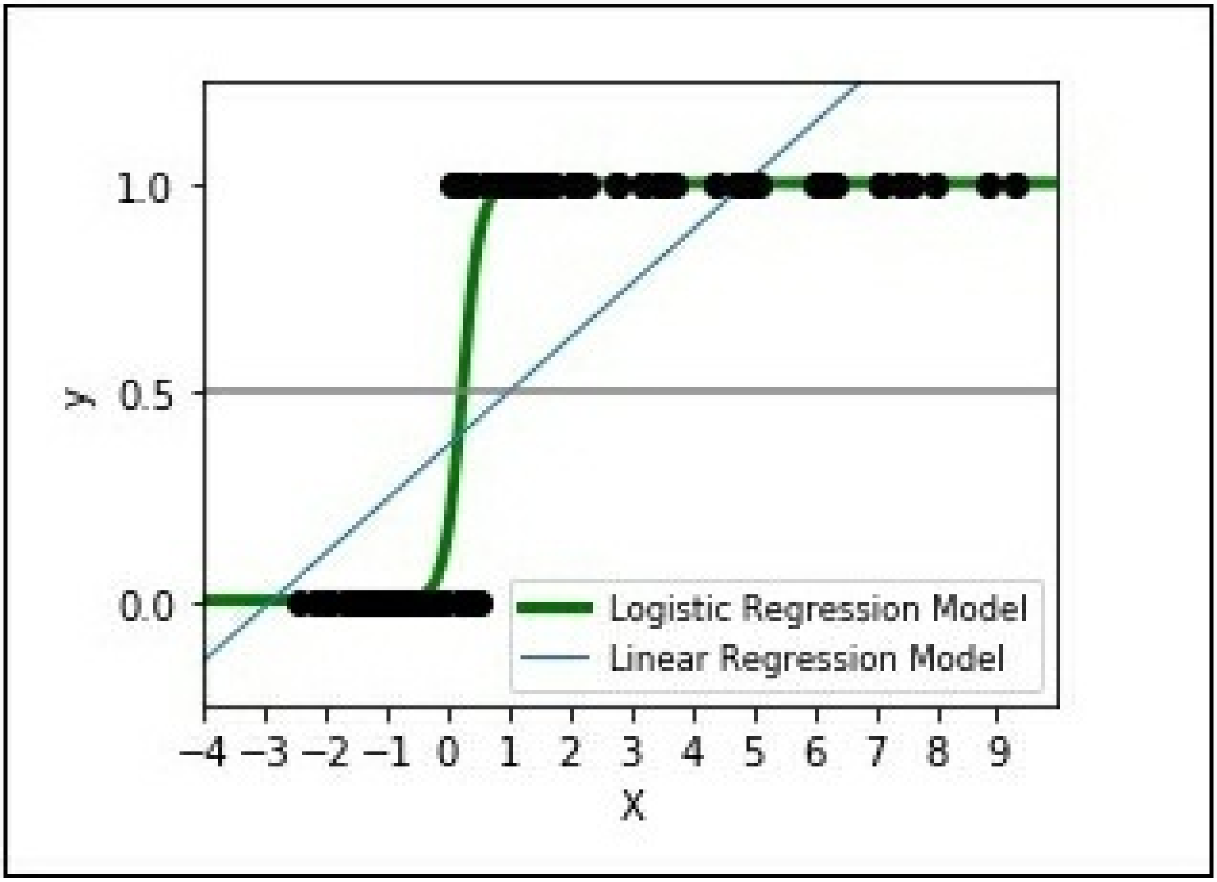


```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
# This is the test set, it's a straight line with some Gaussian noise
xmin, xmax = -10, 10
n_samples = 100
np.random.seed(0)
X = np.random.normal(size = n_samples)
y = (X > 0).astype(np.float)
```



```
X[X > 0] *= 4
X += .3 * np.random.normal(size = n_samples)
X = X[:, np.newaxis]


# run the classifier
clf = linear_model.LogisticRegression(C=1e5)
clf.fit(X, y)

# and plot the result
plt.figure(1, figsize = (4, 3))
plt.clf()
plt.scatter(X.ravel(), y, color='black', zorder=20)
X_test = np.linspace(-10, 10, 300)
def model(x):
return 1 / (1 + np.exp(-x))
loss = model(X_test * clf.coef_ + clf.intercept_).ravel()
plt.plot(X_test, loss, color='blue', linewidth=3)
ols = linear_model.LinearRegression()
ols.fit(X, y)
plt.plot(X_test, ols.coef_ * X_test + ols.intercept_, linewidth=1)
plt.axhline(.5, color='.5')
plt.ylabel('y')
plt.xlabel('X')
plt.xticks(range(-10, 10))
plt.yticks([0, 0.5, 1])
plt.ylim(-.25, 1.25)
plt.xlim(-4, 10)
plt.legend(('Logistic Regression Model', 'Linear Regression Model'),
loc="lower right", fontsize='small')
plt.show()
```

The output plot will be displayed like this –



Algorithm of the decision tree

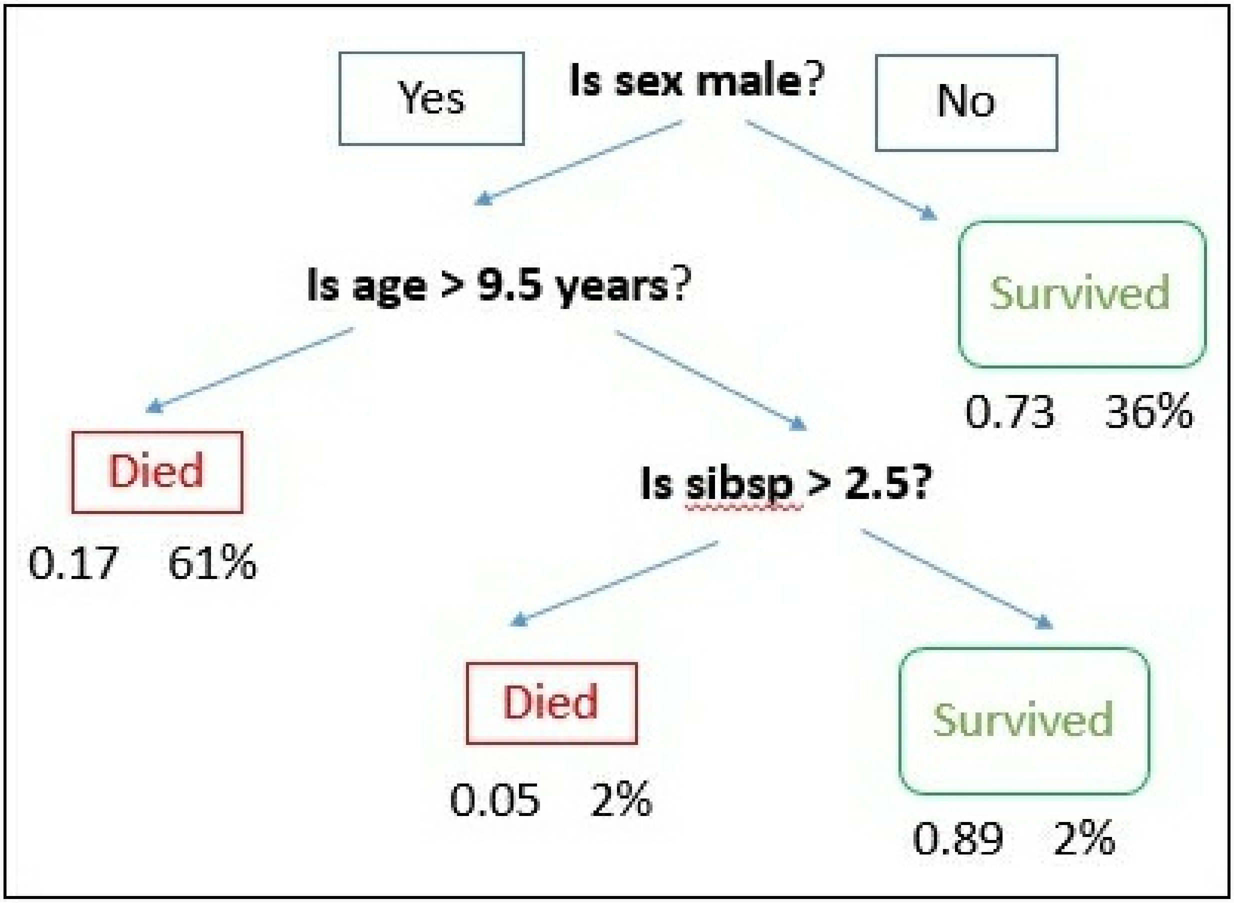
It is a supervised learning algorithm mainly used for classification problems. This works for discrete and continuous dependent variables. In this algorithm, we divide the population into two or more similar sets. This is done based on the most important attributes to create groups as distinct as possible.

Decision trees are popularly used in machine learning, covering both regression and classification. A decision tree can be called on to explicitly and visually represent decisions and decision-making in decision analysis. It uses a tree-shaped model of decisions. It uses a tree-shaped model of decisions.

A decision tree is sketched with its branches at the bottom and its root at the top. In the image, the bold text stands for a condition / internal node, based on which the tree is divided into branches/edges. The end of the branch that no longer divides is the decision/leaf.

Example

Let's take an example of using titanic data to predict whether a passenger will survive or not. The model below uses three characteristics/attributes/columns of the dataset, i.e., sex, age, and sibsp (not spouse/child). In this case, if the passenger is dead or has survived, the text is shown in red and green, respectively.

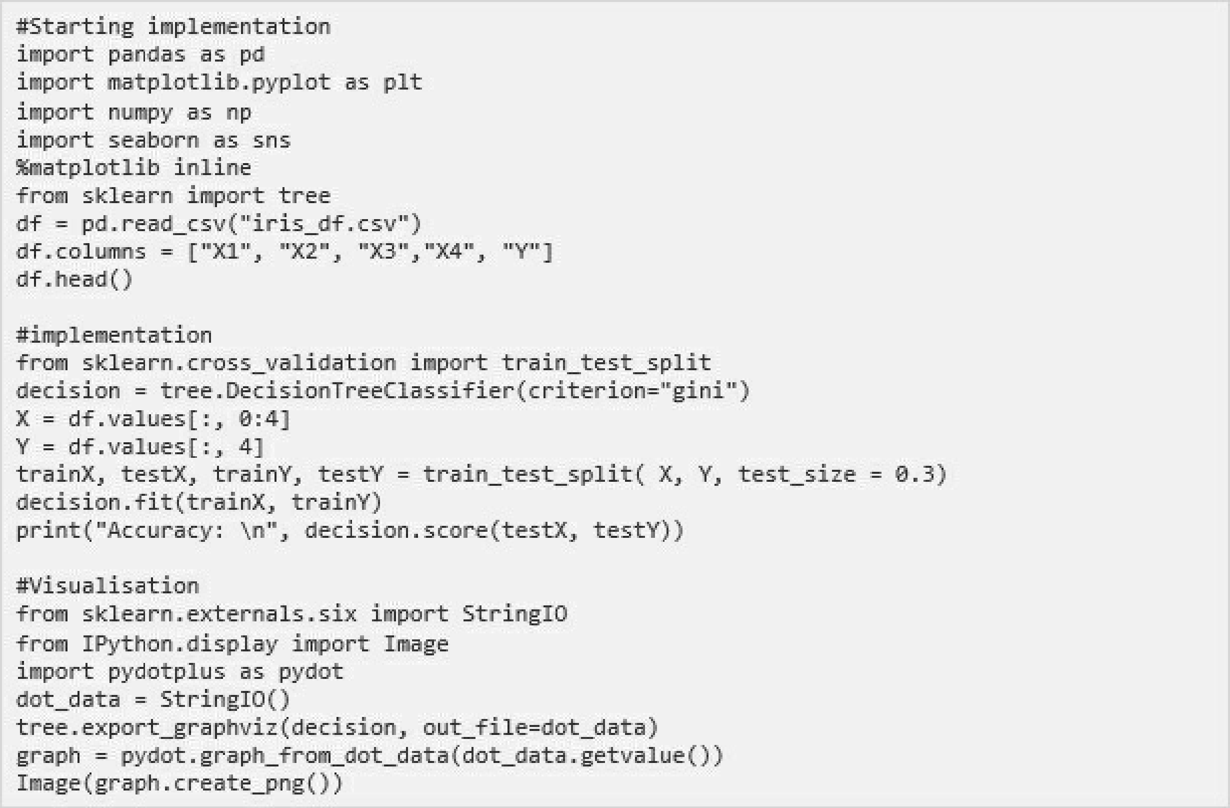


In some instances, we discover that the population can be classified into several groups based on various attributes to identify "whether they do something or not." To divide the population into different heterogeneous groups, he uses various techniques such as Gini, information gain, chi-squared, entropy, and so on.

The best way to understand how the decision tree works are to play Jezzball, a classic Microsoft game. In this Microsoft game, you will be provided with a room with moving walls, and you have to create walls so that the maximum area is removed without the balls.

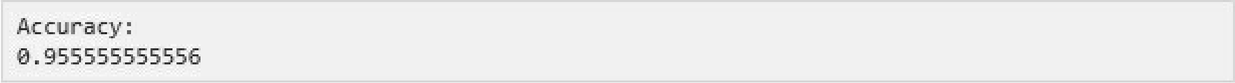
So, every time you divide the room by one wall, you try to create two different populations in the same room. Decision trees work in a very similar way by dividing a population into groups as different as possible.

Note the code and its output given below –



```
#Starting implementation
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
%matplotlib inline
from sklearn import tree
df = pd.read_csv("iris_df.csv")
df.columns = ["X1", "X2", "X3","X4", "Y"]
df.head()

#implementation
from sklearn.cross_validation import train_test_split
decision = tree.DecisionTreeClassifier(criterion="gini")
X = df.values[:, 0:4]
Y = df.values[:, 4]
trainX, testX, trainY, testY = train_test_split( X, Y, test_size = 0.3)
decision.fit(trainX, trainY)
print("Accuracy: \n", decision.score(testX, testY))

#Visualisation
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus as pydot
dot_data = StringIO()
tree.export_graphviz(decision, out_file=dot_data)
graph = pydot.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

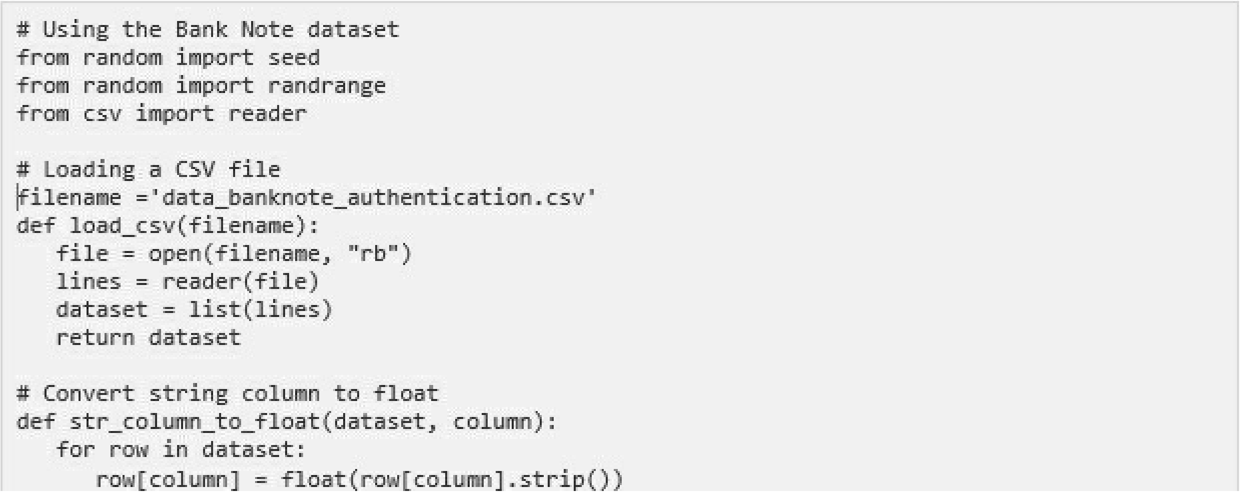
Output



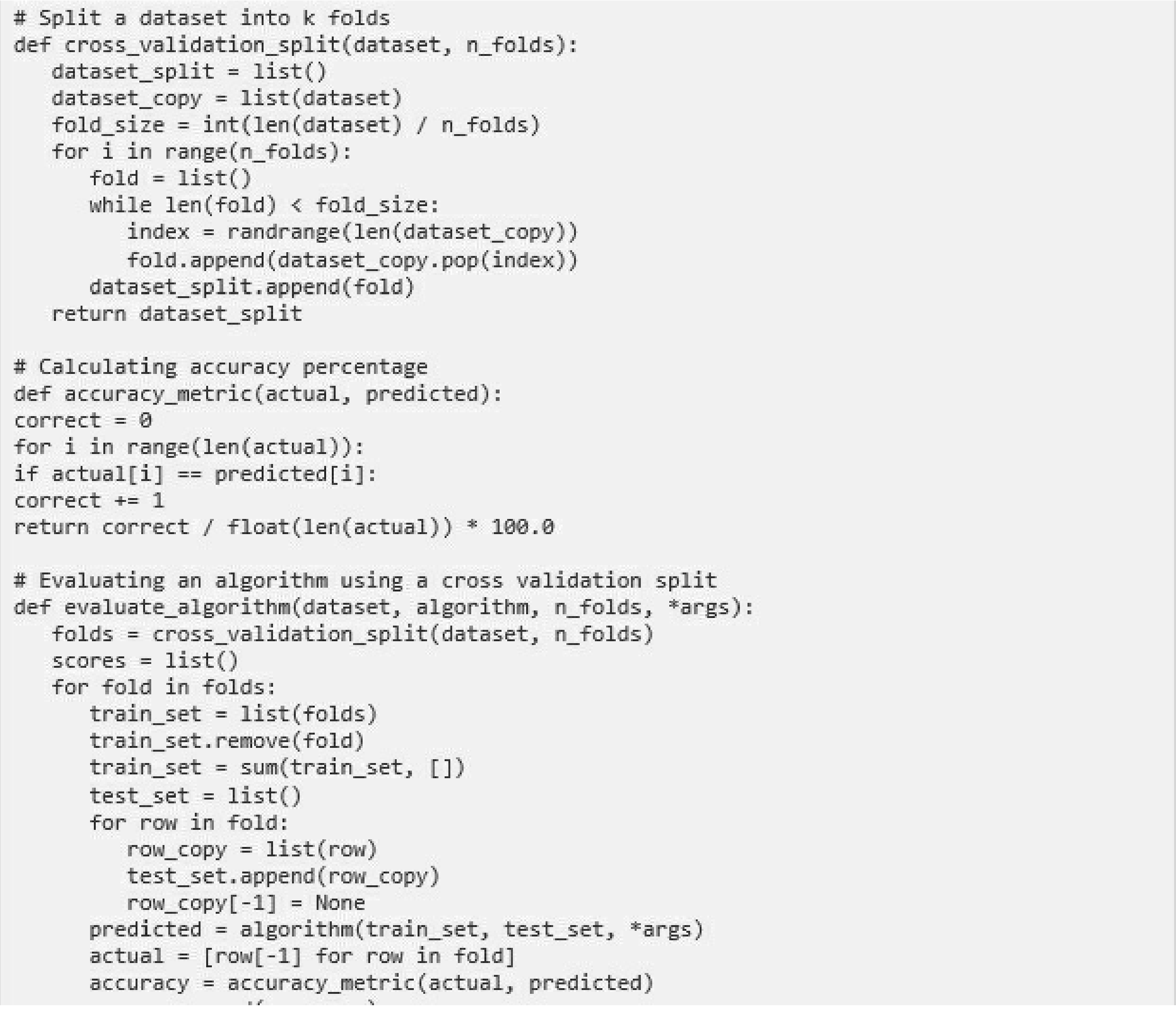
```
Accuracy:
0.955555555556
```

Example

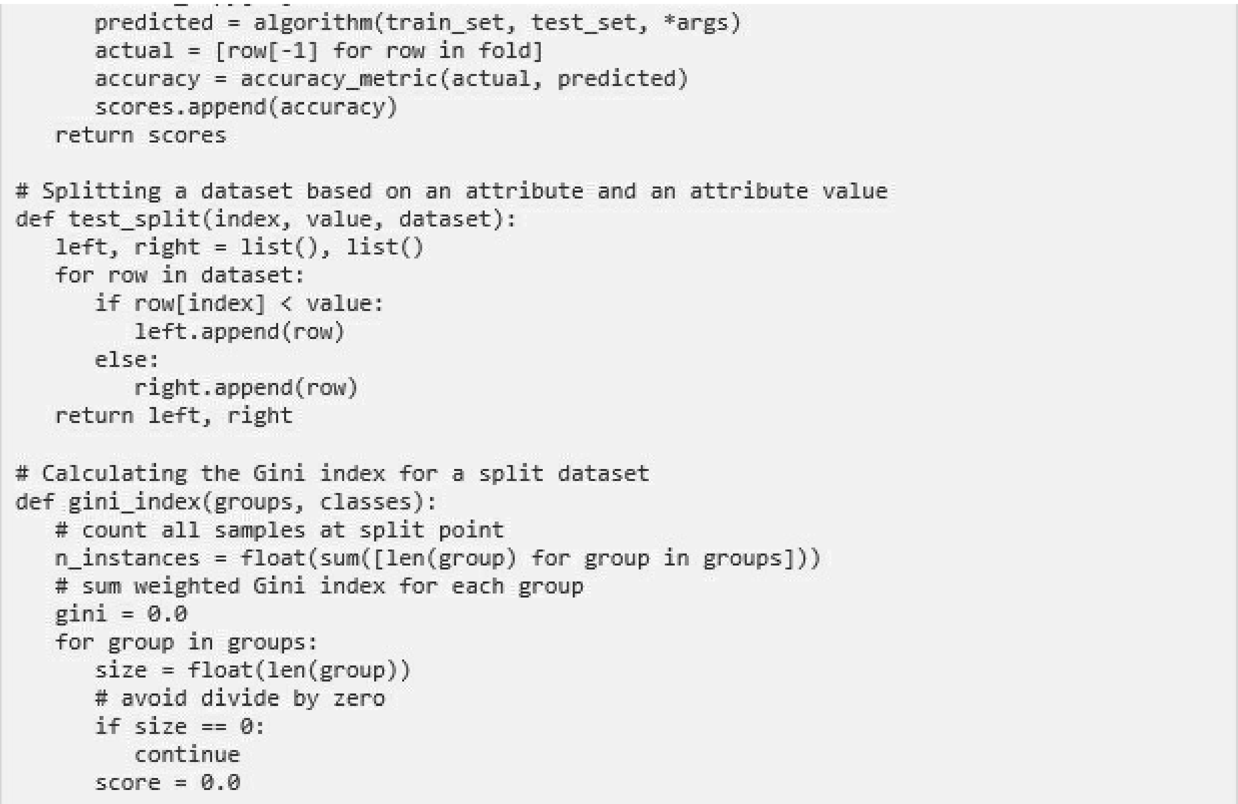
Here we are making use of the banknote authentication dataset to determine the accuracy.



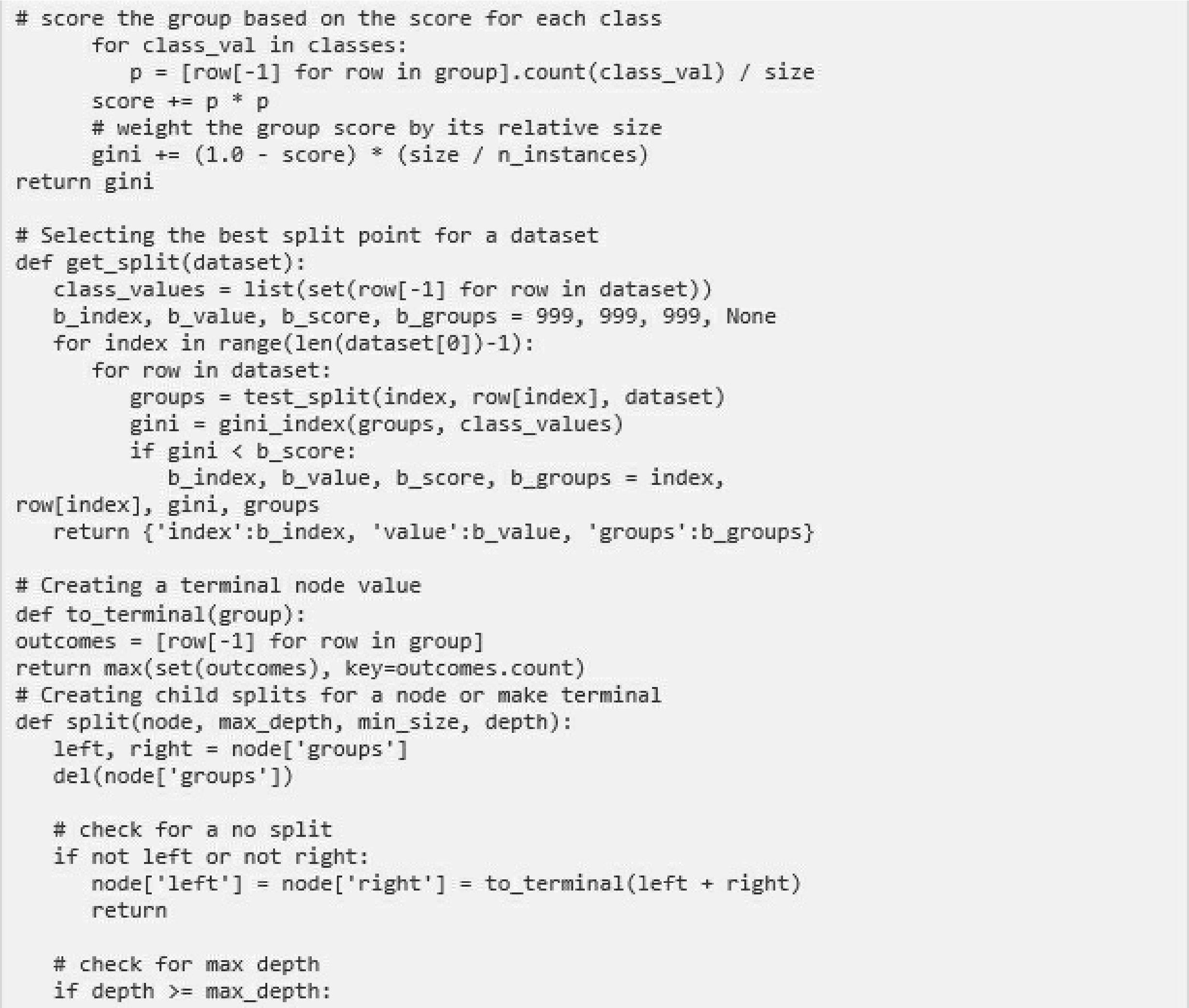
```
# Using the Bank Note dataset
from random import seed
from random import randrange
from csv import reader

# Loading a CSV file
filename ='data_banknote_authentication.csv'
def load_csv(filename):
   file = open(filename, "rb")
   lines = reader(file)
   dataset = list(lines)
   return dataset

# Convert string column to float
def str_column_to_float(dataset, column):
   for row in dataset:
      row[column] = float(row[column].strip())
```



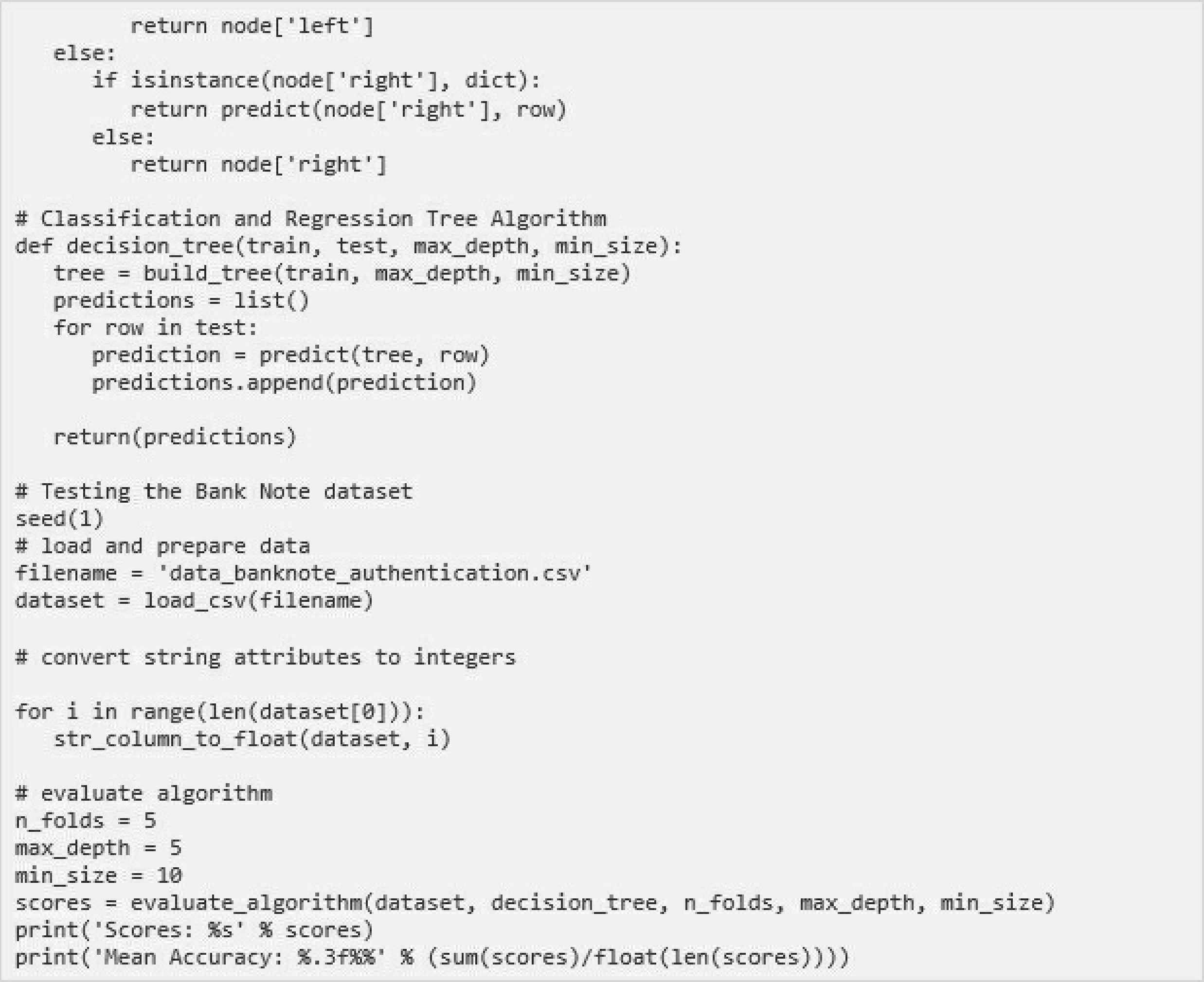
```
# Split a dataset into k folds
def cross_validation_split(dataset, n_folds):
   dataset_split = list()
   dataset_copy = list(dataset)
   fold_size = int(len(dataset) / n_folds)
   for i in range(n_folds):
      fold = list()
      while len(fold) < fold_size:
         index = randrange(len(dataset_copy))
         fold.append(dataset_copy.pop(index))
      dataset_split.append(fold)
   return dataset_split

# Calculating accuracy percentage
def accuracy_metric(actual, predicted):
correct = 0
for i in range(len(actual)):
if actual[i] == predicted[i]:
correct += 1
return correct / float(len(actual)) * 100.0

# Evaluating an algorithm using a cross validation split
def evaluate_algorithm(dataset, algorithm, n_folds, *args):
   folds = cross_validation_split(dataset, n_folds)
   scores = list()
   for fold in folds:
      train_set = list(folds)
      train_set.remove(fold)
      train_set = sum(train_set, [])
      test_set = list()
      for row in fold:
         row_copy = list(row)
         test_set.append(row_copy)
         row_copy[-1] = None
      predicted = algorithm(train_set, test_set, *args)
      actual = [row[-1] for row in fold]
      accuracy = accuracy_metric(actual, predicted)
```



```
      predicted = algorithm(train_set, test_set, *args)
      actual = [row[-1] for row in fold]
      accuracy = accuracy_metric(actual, predicted)
      scores.append(accuracy)
   return scores

# Splitting a dataset based on an attribute and an attribute value
def test_split(index, value, dataset):
   left, right = list(), list()
   for row in dataset:
      if row[index] < value:
         left.append(row)
      else:
         right.append(row)
   return left, right

# Calculating the Gini index for a split dataset
def gini_index(groups, classes):
   # count all samples at split point
   n_instances = float(sum([len(group) for group in groups]))
   # sum weighted Gini index for each group
   gini = 0.0
   for group in groups:
      size = float(len(group))
      # avoid divide by zero
      if size == 0:
         continue
      score = 0.0
```

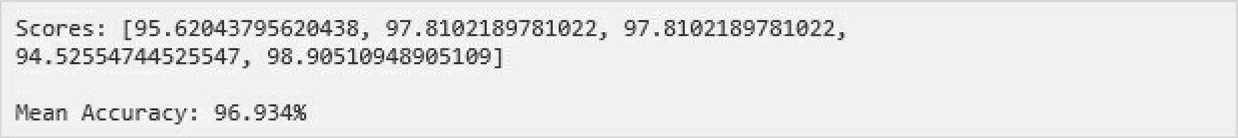


```
# score the group based on the score for each class
      for class_val in classes:
         p = [row[-1] for row in group].count(class_val) / size
      score += p * p
      # weight the group score by its relative size
      gini += (1.0 - score) * (size / n_instances)
return gini

# Selecting the best split point for a dataset
def get_split(dataset):
   class_values = list(set(row[-1] for row in dataset))
   b_index, b_value, b_score, b_groups = 999, 999, 999, None
   for index in range(len(dataset[0])-1):
      for row in dataset:
         groups = test_split(index, row[index], dataset)
         gini = gini_index(groups, class_values)
         if gini < b_score:
            b_index, b_value, b_score, b_groups = index,
row[index], gini, groups
   return {'index':b_index, 'value':b_value, 'groups':b_groups}

# Creating a terminal node value
def to_terminal(group):
outcomes = [row[-1] for row in group]
return max(set(outcomes), key=outcomes.count)
# Creating child splits for a node or make terminal
def split(node, max_depth, min_size, depth):
   left, right = node['groups']
   del(node['groups'])

   # check for a no split
   if not left or not right:
      node['left'] = node['right'] = to_terminal(left + right)
      return

   # check for max depth
   if depth >= max_depth:
      node['left'], node['right'] = to_terminal(left), to_terminal(right)
      return

   # process left child
   if len(left) <= min_size:
      node['left'] = to_terminal(left)
   else:
      node['left'] = get_split(left)
      split(node['left'], max_depth, min_size, depth+1)
   # process right child
   if len(right) <= min_size:
      node['right'] = to_terminal(right)
   else:
      node['right'] = get_split(right)
      split(node['right'], max_depth, min_size, depth+1)

# Building a decision tree
def build_tree(train, max_depth, min_size):
   root = get_split(train)
   split(root, max_depth, min_size, 1)
   return root

# Making a prediction with a decision tree
   def predict(node, row):
   if row[node['index']] < node['value']:
      if isinstance(node['left'], dict):
         return predict(node['left'], row)
      else:
```



```
         return node['left']
   else:
      if isinstance(node['right'], dict):
         return predict(node['right'], row)
      else:
         return node['right']

# Classification and Regression Tree Algorithm
def decision_tree(train, test, max_depth, min_size):
   tree = build_tree(train, max_depth, min_size)
   predictions = list()
   for row in test:
      prediction = predict(tree, row)
      predictions.append(prediction)

   return(predictions)

# Testing the Bank Note dataset
seed(1)
# load and prepare data
filename = 'data_banknote_authentication.csv'
dataset = load_csv(filename)

# convert string attributes to integers

for i in range(len(dataset[0])):
   str_column_to_float(dataset, i)

# evaluate algorithm
n_folds = 5
max_depth = 5
min_size = 10
scores = evaluate_algorithm(dataset, decision_tree, n_folds, max_depth, min_size)
print('Scores: %s' % scores)
print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))
```

When the code given above is executed, you can observe the output as follows –



```
Scores: [95.62043795620438, 97.8102189781022, 97.8102189781022,
94.52554744525547, 98.90510948905109]

Mean Accuracy: 96.934%
```

Support vector machines (SVM)

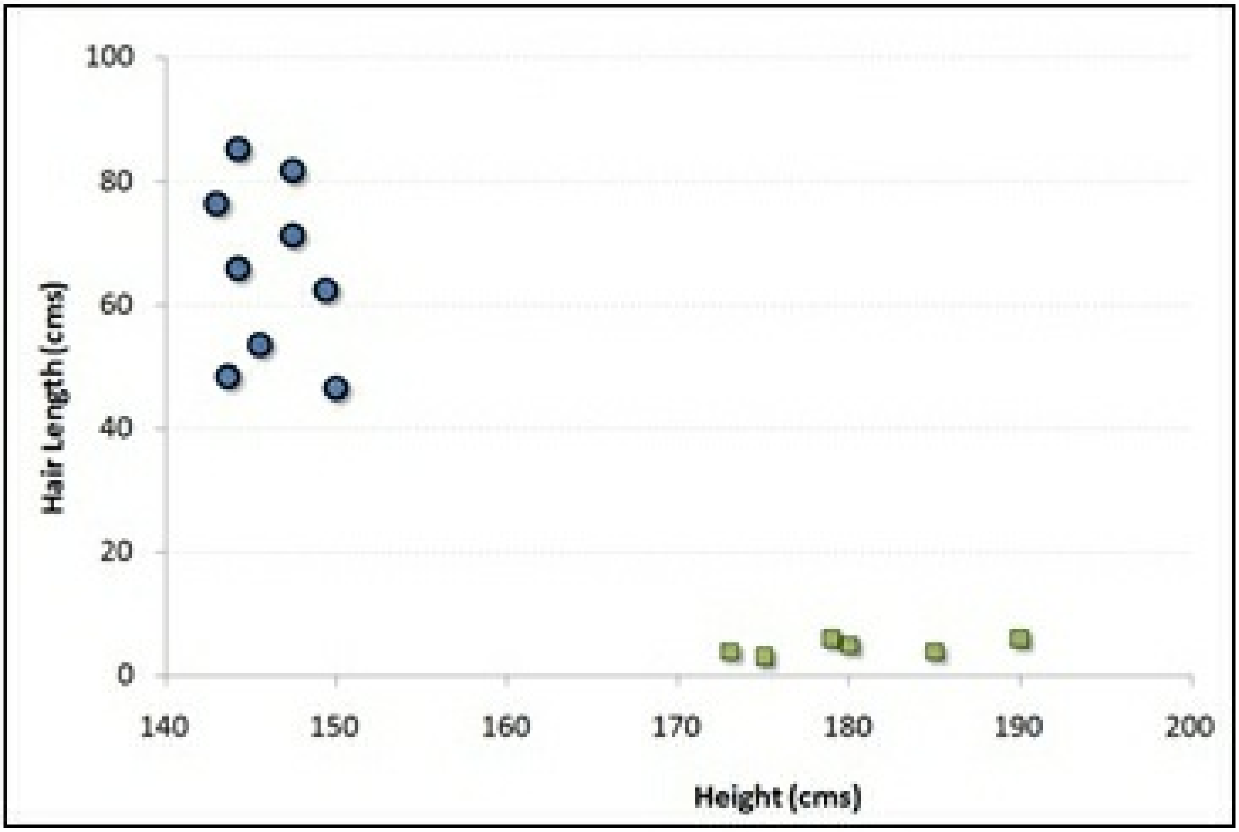
Support vector machines, also known as SVMs, are well-known supervised classification algorithms which distinguish different categories of data.

These vectors are also classified by optimizing the line to make sure that the closest point in each group is farthest from each other.

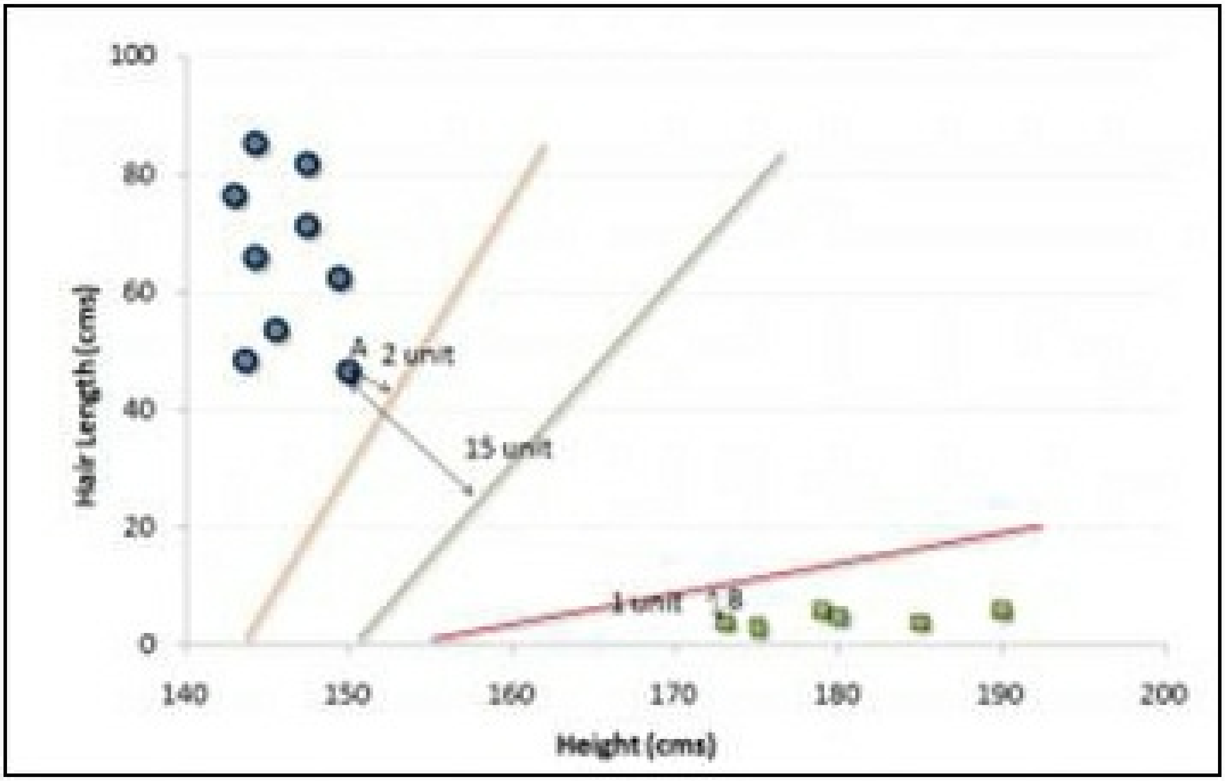
This vector is linear by default and is often considered linear. However, the vector may also have a nonlinear shape if the kernel type is changed from the "Gaussian" or standard linear type.

It is a classification method, in which each point data is plotted in n-dimensional space (where n is the number of entities), the value of each entity is the value of a coordinate given.

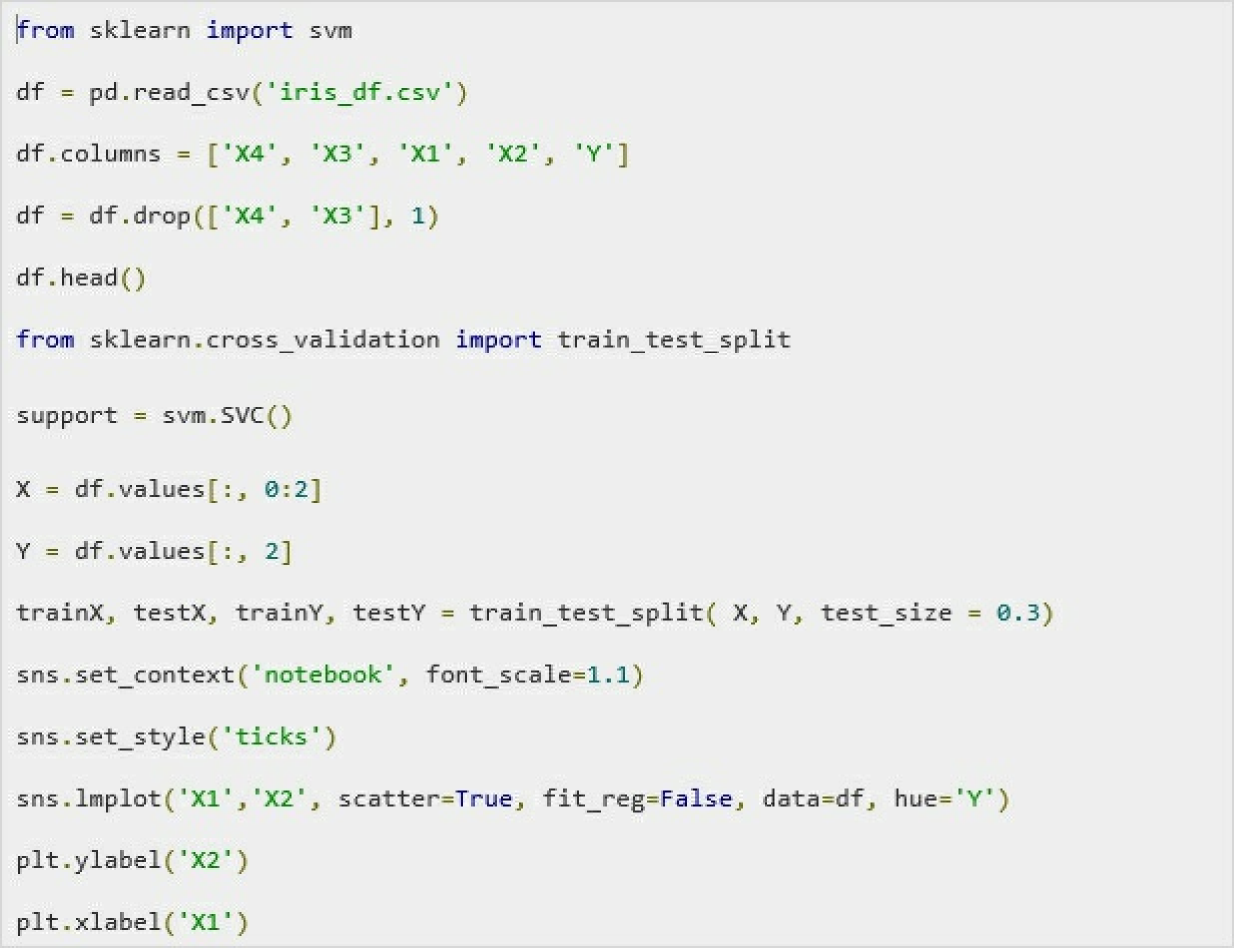
For example, if we only have two characteristics, such as the height and length of an individual's hair, we must first draw these two variables in the two-dimensional space, where each point has two coordinates called support vectors. Observe the following diagram for a better understanding -



Now find a line that divides the data between the two groups of data sorted differently. This line can be seen such that the distances from the nearest point in each of the two groups will be farther apart.

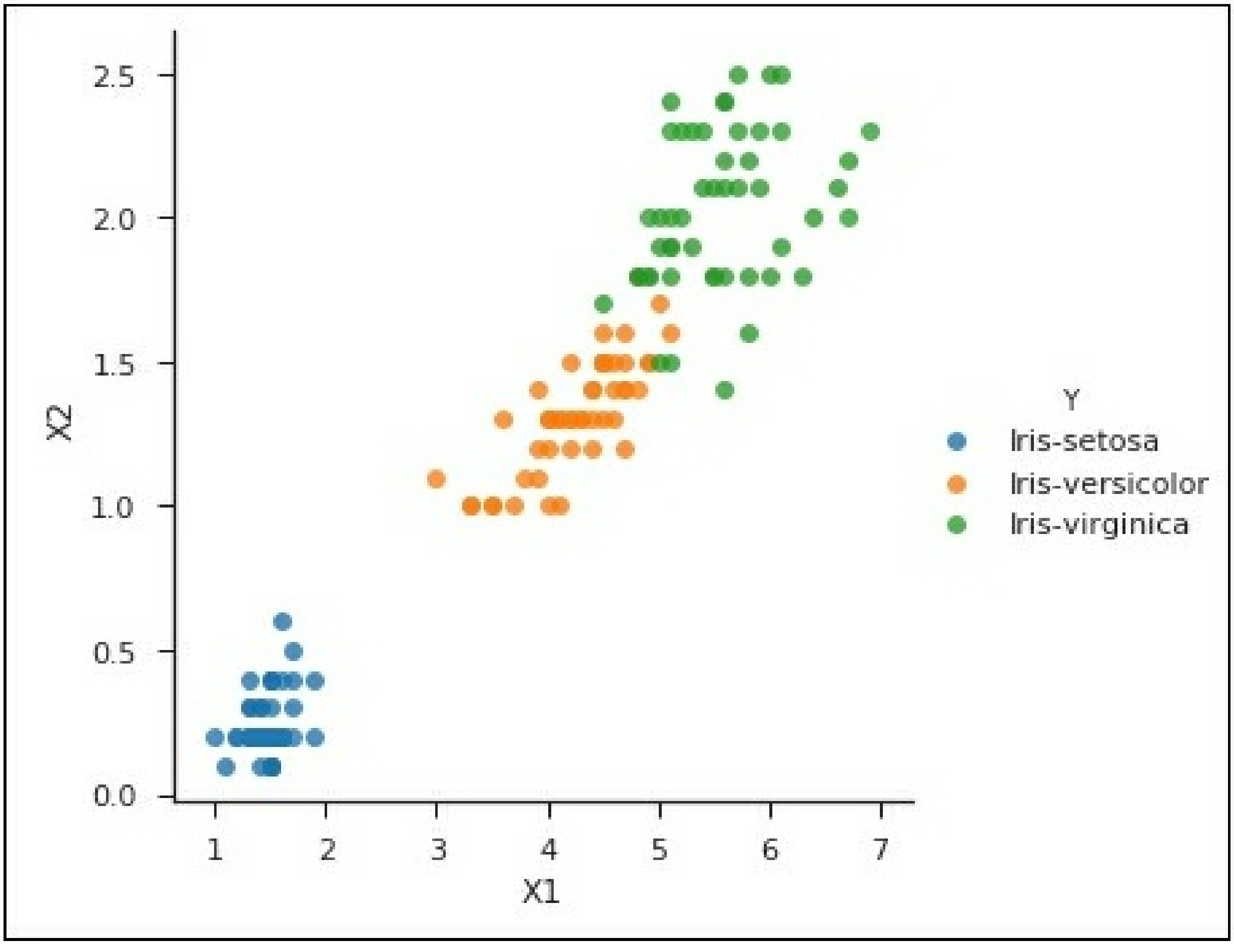


In the above example, the line dividing the data into two differently sorted groups is the black line, because the two closest points are furthest from the line. This line is our classifier. So, depending on the arrival of the test data on both sides of the line, we can sort the new data.



```
from sklearn import svm

df = pd.read_csv('iris_df.csv')

df.columns = ['X4', 'X3', 'X1', 'X2', 'Y']

df = df.drop(['X4', 'X3'], 1)

df.head()

from sklearn.cross_validation import train_test_split

support = svm.SVC()

X = df.values[:, 0:2]

Y = df.values[:, 2]

trainX, testX, trainY, testY = train_test_split( X, Y, test_size = 0.3)

sns.set_context('notebook', font_scale=1.1)

sns.set_style('ticks')

sns.lmplot('X1','X2', scatter=True, fit_reg=False, data=df, hue='Y')

plt.ylabel('X2')

plt.xlabel('X1')
```

You can take note of the following result and plot after running the code shown above –



Naïve Bayes Algorithm

It is a classification technique based on the Bayes theorem with an assumption of independence of the predictor variables. Succinctly put, a Naive Bayes classifier assumes that the presence of a particular characteristic in one class is dissimilar to the existence of another characteristic.

For example, fruit can be considered an orange if it is orange, round and about 3 inches in diameter. Even if these characteristics depend on each other or the existence of other characteristics, a Bayes naive classifier would consider that all these characteristics contribute independently to the probability that this fruit is an orange.

The naive Bayes model is easy to perform and particularly useful for very large datasets. In addition to being simple, Naive Bayes is known for its ability to overcome even the most advanced classification methods.

Bayes' theorem provides a means of calculating the posterior probability P (c | x) of P (c), P (x) and P (x | c). Observe the equation given here: P (c / x) = P (x / c) P (c) / P (x)

Or,

P (c | x) stands for the posterior probability of the given predictor (attribute) of class (target).

P (c) is the previous class probability.

P (x | c) is the probability which is the probability of a given predictor class.

P (x) is the prior probability of the predictor.

Consider the example below for a better understanding

Suppose a set of Time training data and the corresponding Play goal variable. We must now determine whether players will play or not depending on the weather conditions. For this, you will have to follow the steps below -

Step 1 - Convert the dataset into the frequency table.

Step 2 - Create the probability table, looking for probabilities, such as overcast probability = 0.29 and gambling probability of 0.64.

Step 3 - Use the naive Bayesian equation to calculate the posterior probability of each class. The result of the forecast will be the class with the highest probability.

Problem - Players will play if the weather is nice, is this statement correct?

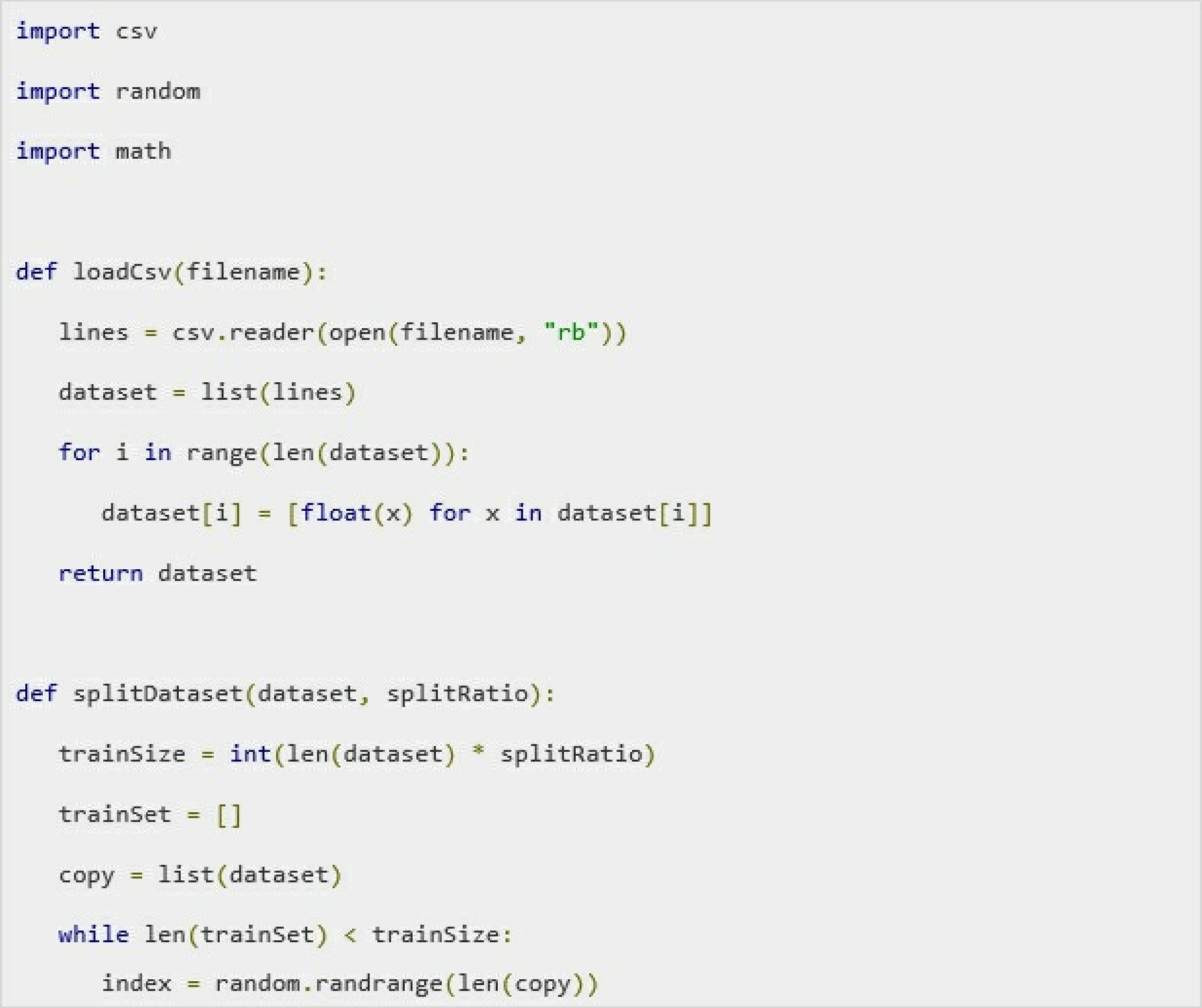
Solution - We can solve it using the method described above, then P (Yes | Sun) = P (Sun | Yes) * P (Yes) / P (Sun)

Here we have, P (Sun | Yes) = 3/9 = 0.333, P (Sun) = 5/14 = 0.357, P (Yes) = 9/14 = 0.64

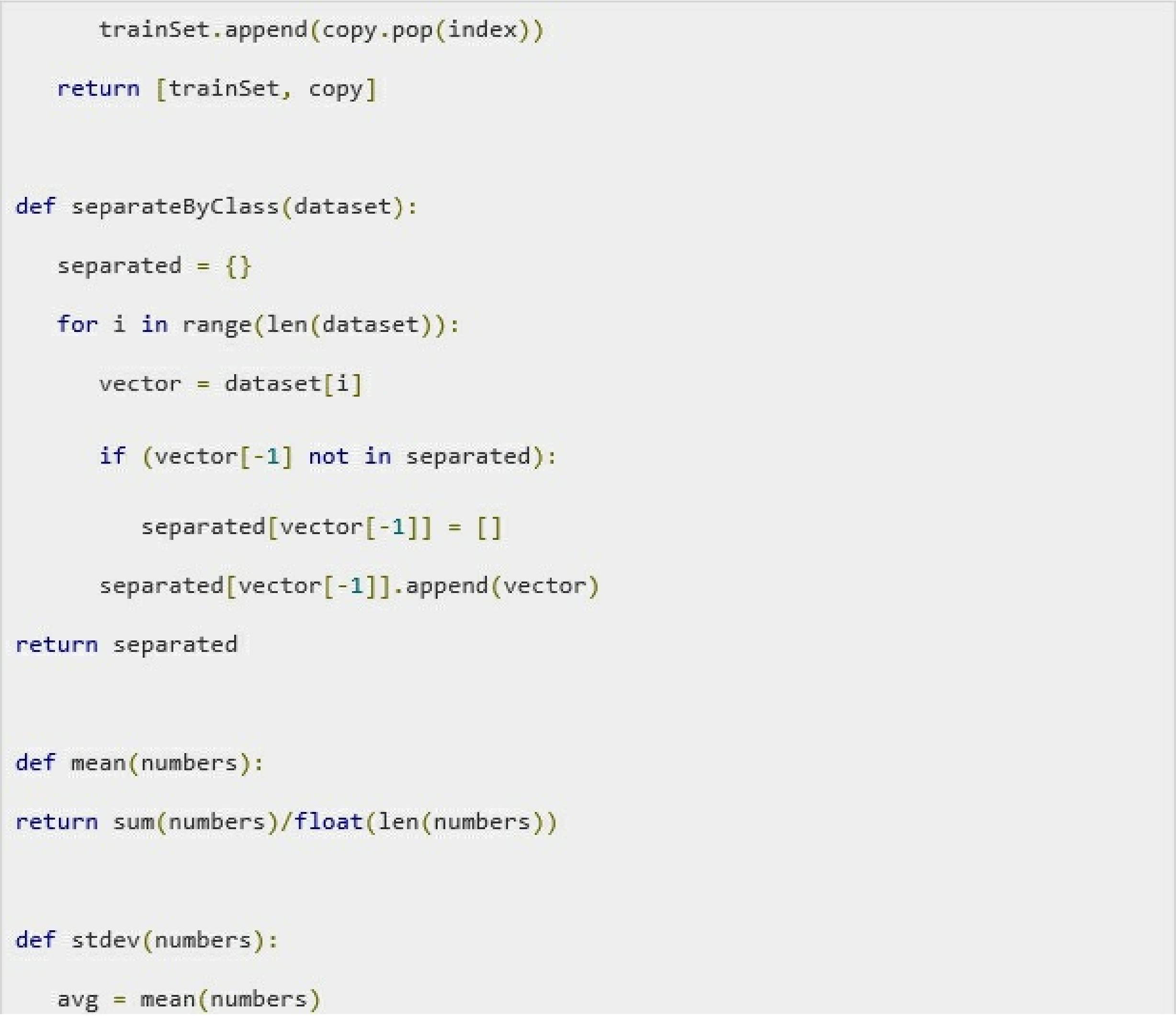
Now, P (Yes | Sun) = 0.333 * 0.64 / 0.357 = 0.60, which has a higher probability.

Naive Bayes makes use of a similar technique to predict the probability of different classes based on various attributes. This algorithm is mainly used in the classification of text and during problems with several classes.

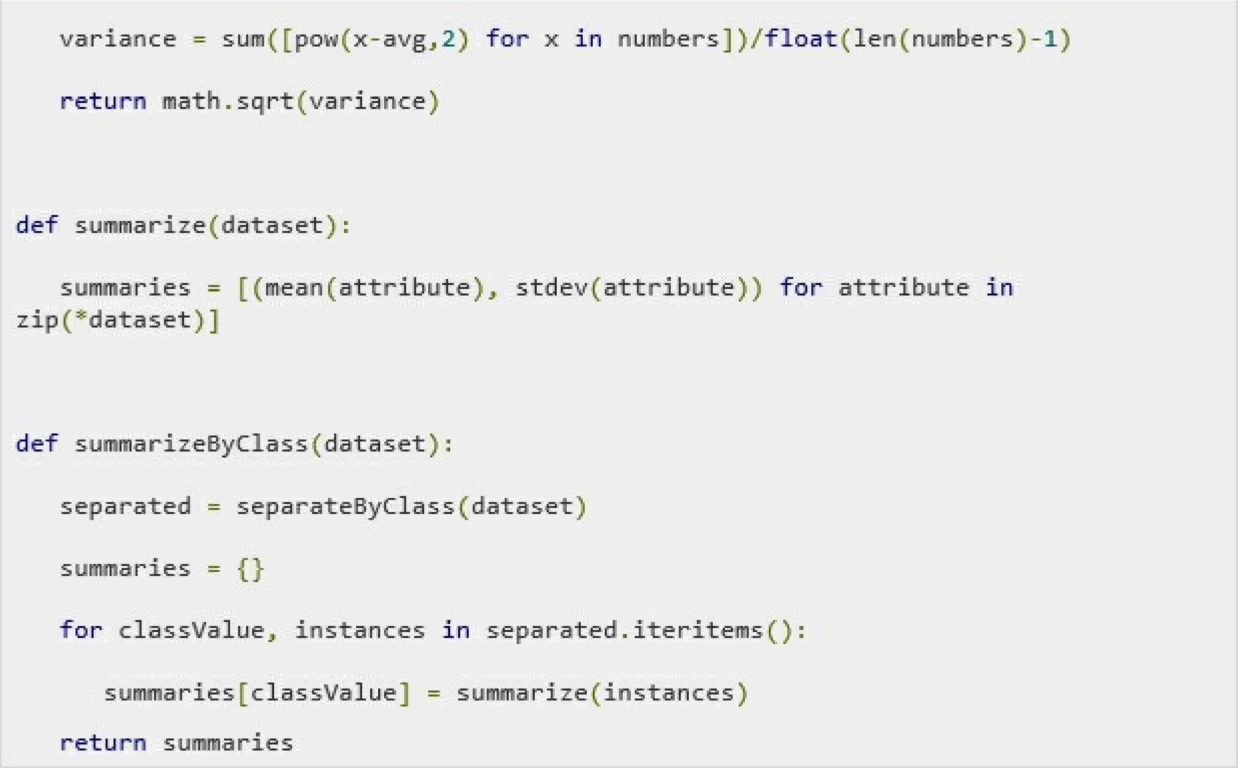
The code below snippet shows an example of Naive Bayes implementation –



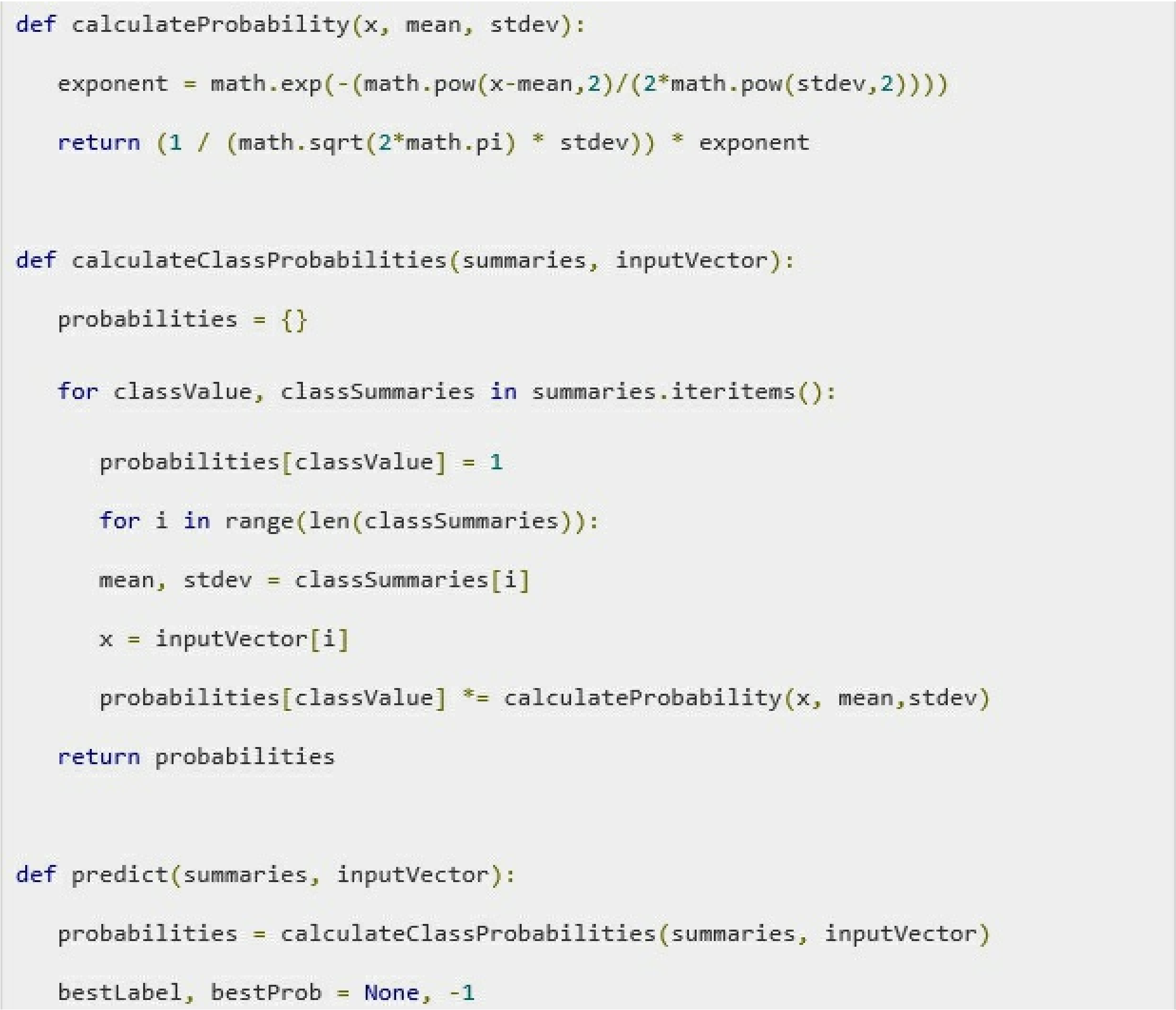
```
import csv

import random

import math


def loadCsv(filename):

   lines = csv.reader(open(filename, "rb"))

   dataset = list(lines)

   for i in range(len(dataset)):

      dataset[i] = [float(x) for x in dataset[i]]

   return dataset


def splitDataset(dataset, splitRatio):

   trainSize = int(len(dataset) * splitRatio)

   trainSet = []

   copy = list(dataset)

   while len(trainSet) < trainSize:
      index = random.randrange(len(copy))
```



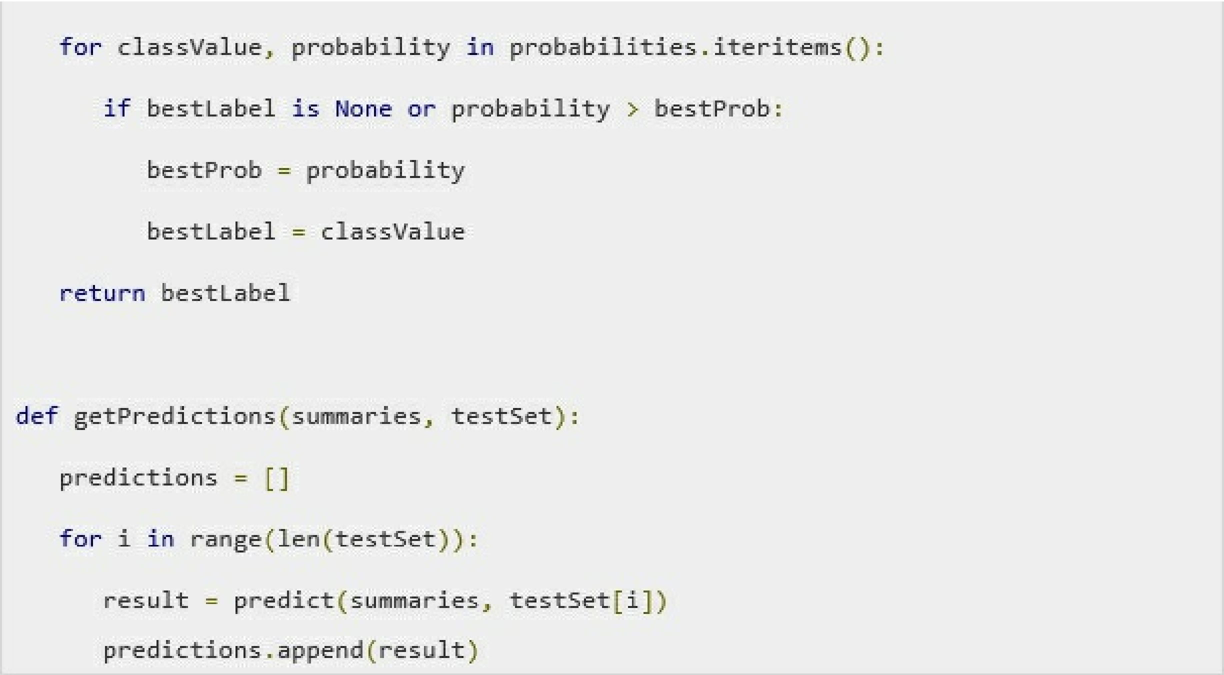
```
      trainSet.append(copy.pop(index))
   return [trainSet, copy]


def separateByClass(dataset):
   separated = {}
   for i in range(len(dataset)):
      vector = dataset[i]
      if (vector[-1] not in separated):
         separated[vector[-1]] = []
      separated[vector[-1]].append(vector)
return separated


def mean(numbers):
return sum(numbers)/float(len(numbers))


def stdev(numbers):
   avg = mean(numbers)
```



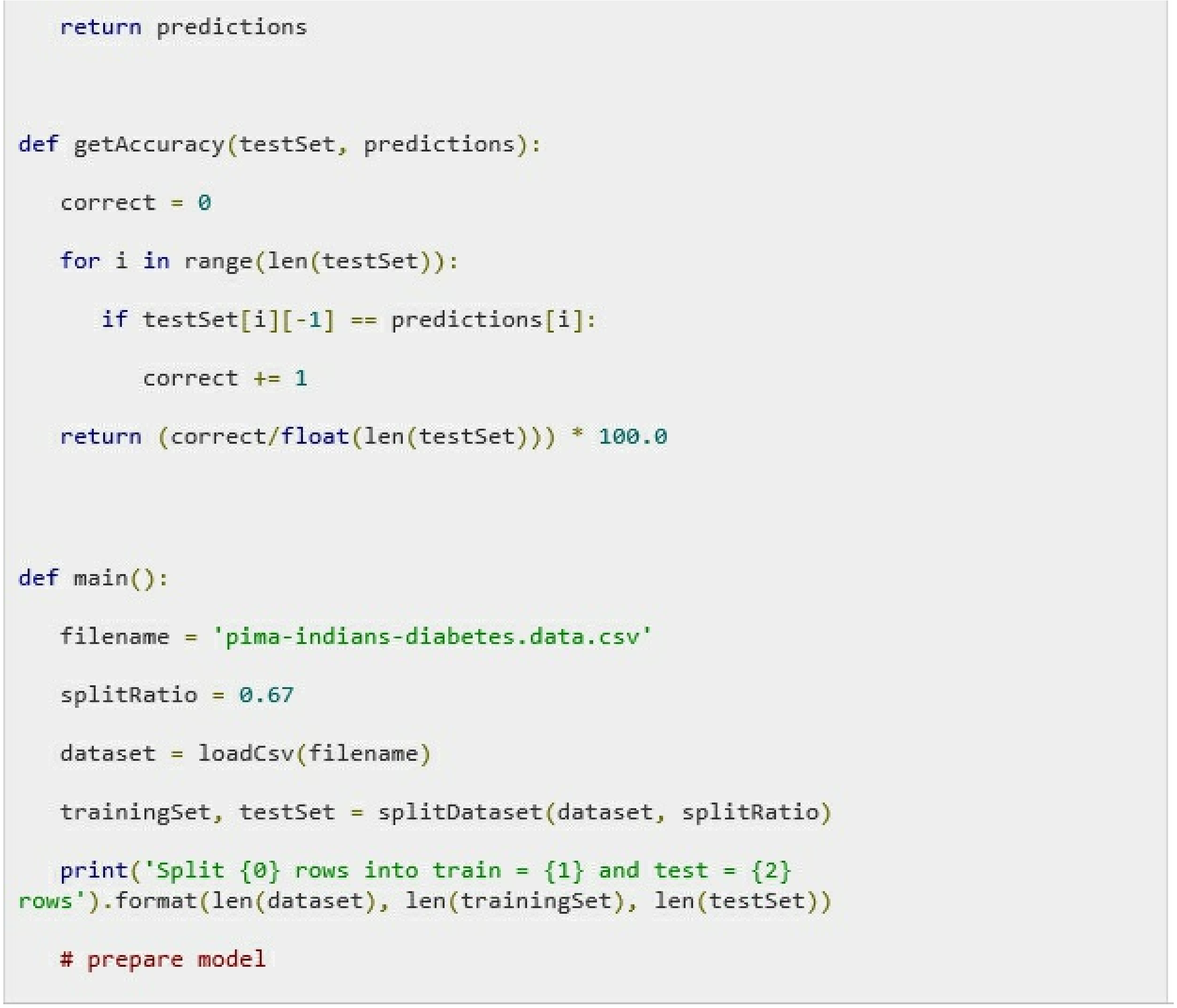
```
   variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers)-1)
   return math.sqrt(variance)


def summarize(dataset):
   summaries = [(mean(attribute), stdev(attribute)) for attribute in
zip(*dataset)]


def summarizeByClass(dataset):
   separated = separateByClass(dataset)
   summaries = {}
   for classValue, instances in separated.iteritems():
      summaries[classValue] = summarize(instances)
   return summaries
```



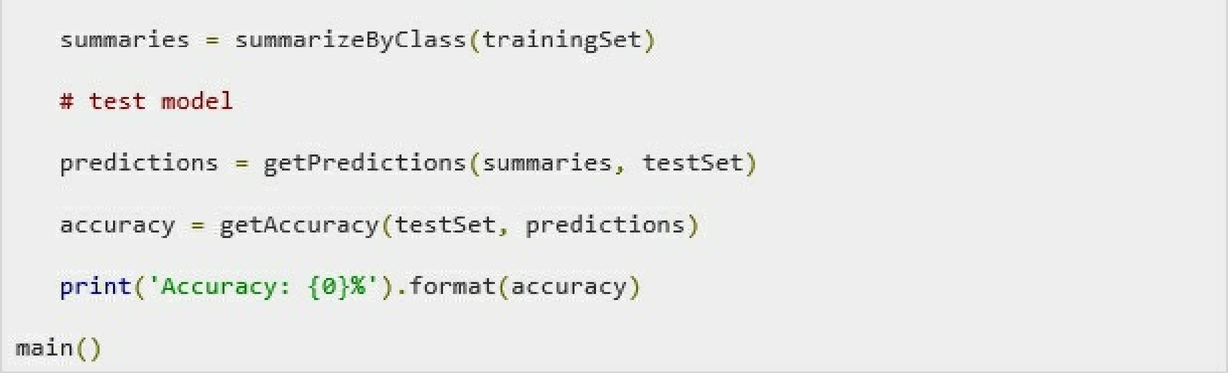
```
def calculateProbability(x, mean, stdev):

   exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))

   return (1 / (math.sqrt(2*math.pi) * stdev)) * exponent


def calculateClassProbabilities(summaries, inputVector):

   probabilities = {}

   for classValue, classSummaries in summaries.iteritems():

      probabilities[classValue] = 1

      for i in range(len(classSummaries)):

      mean, stdev = classSummaries[i]

      x = inputVector[i]

      probabilities[classValue] *= calculateProbability(x, mean,stdev)

   return probabilities


def predict(summaries, inputVector):

   probabilities = calculateClassProbabilities(summaries, inputVector)

   bestLabel, bestProb = None, -1
```



```
   for classValue, probability in probabilities.iteritems():
      if bestLabel is None or probability > bestProb:
         bestProb = probability
         bestLabel = classValue
   return bestLabel


def getPredictions(summaries, testSet):
   predictions = []
   for i in range(len(testSet)):
      result = predict(summaries, testSet[i])
      predictions.append(result)
```

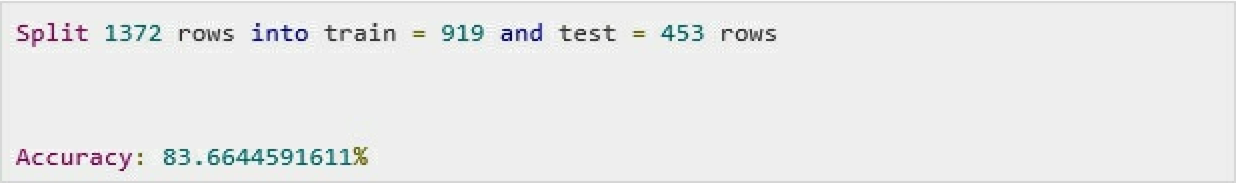


```
   return predictions


def getAccuracy(testSet, predictions):

   correct = 0

   for i in range(len(testSet)):

      if testSet[i][-1] == predictions[i]:

         correct += 1

   return (correct/float(len(testSet))) * 100.0



def main():

   filename = 'pima-indians-diabetes.data.csv'

   splitRatio = 0.67

   dataset = loadCsv(filename)

   trainingSet, testSet = splitDataset(dataset, splitRatio)

   print('Split {0} rows into train = {1} and test = {2}
rows').format(len(dataset), len(trainingSet), len(testSet))

   # prepare model
```



```
   summaries = summarizeByClass(trainingSet)

   # test model

   predictions = getPredictions(summaries, testSet)

   accuracy = getAccuracy(testSet, predictions)

   print('Accuracy: {0}%').format(accuracy)

main()
```

After running the code given above, you can observe the following output –



```
Split 1372 rows into train = 919 and test = 453 rows



Accuracy: 83.6644591611%
```

KNN (K-nearest neighbors)

K-Nearest Neighbors, in short KNN, is a supervised learning algorithm that specialized in classification. KNN is a simple algorithm which keeps all available cases and classifies new cases by the vote of most of its neighbors k. The case allocated to the class is the most common case among its nearest neighbors, measured by a distance function. These distance functions can be Euclidean distance, Manhattan, Minkowski, and Hamming. The first three functions are used for the continuous function and the fourth one (Hamming) for the categorical variables. If K = 1, the case is assigned to the class of its nearest neighbor. Sometimes the choice of K is difficult during KNN modeling.

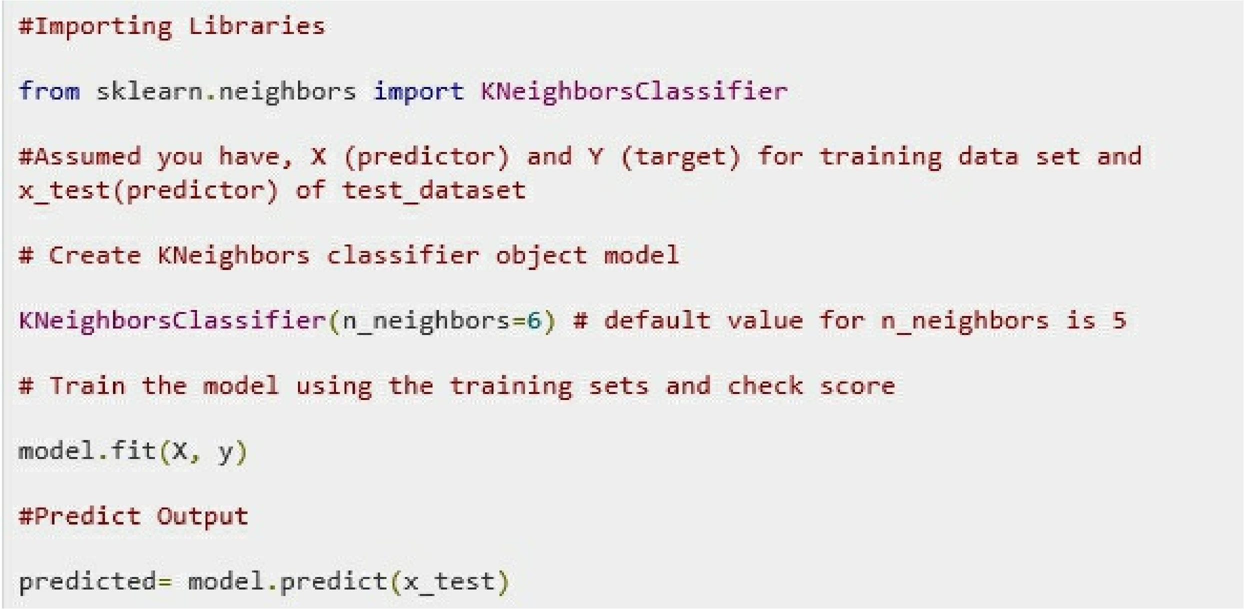
The algorithm examines different centroids and compares the distance using a kind of (usually Euclidean) function, analyzes these results and assigns each point to the group so that it is optimized to be placed with all the nearest points.

You can use KNN for classification and regression issues. However, it is more popularly used in classification issues in the industry. KNN can be easily mapped in our real lives.

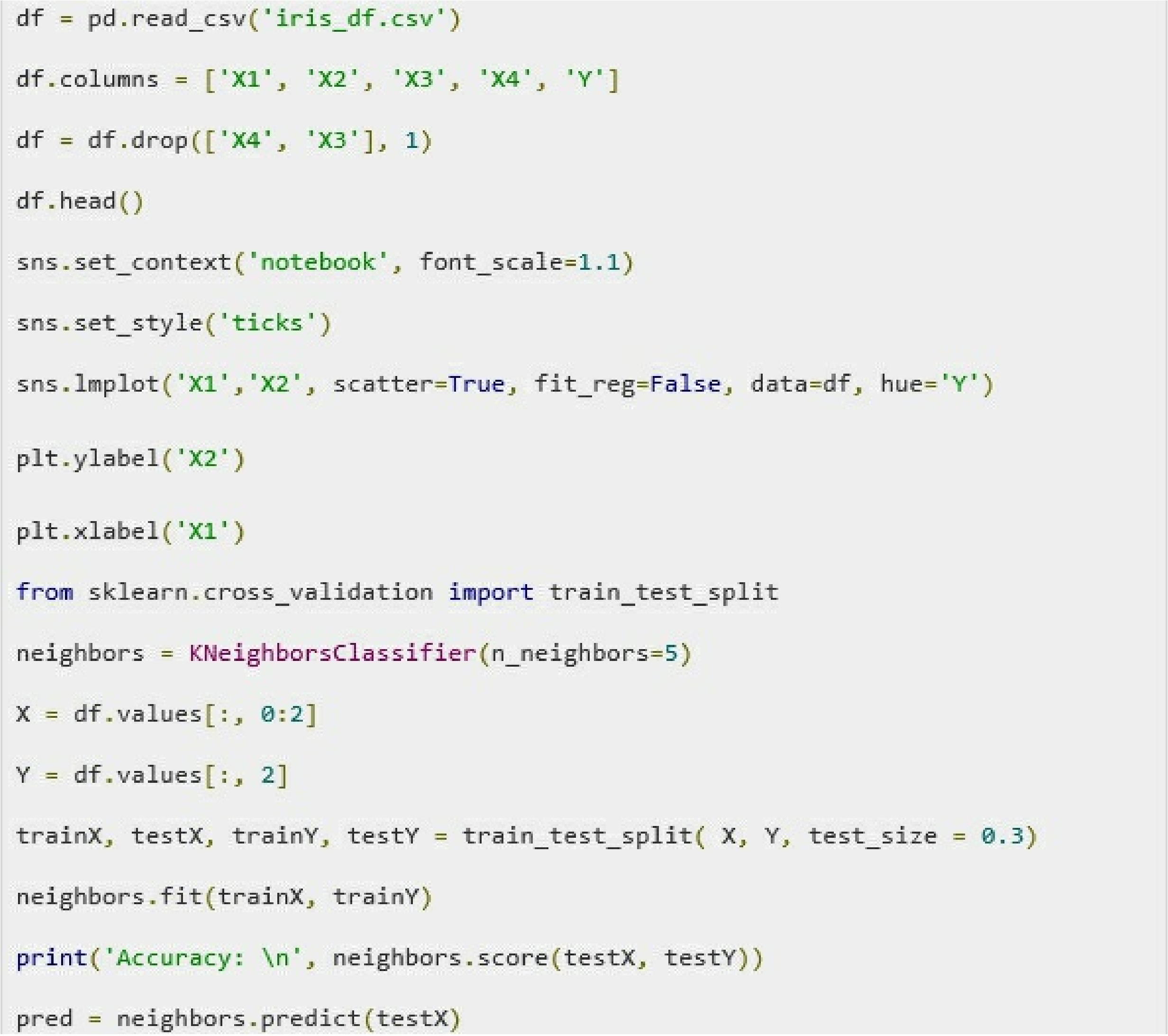
You will have to write the following points before selecting KNN

- KNN is expensive in the calculation.
- Variables must be normalized, otherwise higher interval variables may influence it.
- Works more at the preprocessing stage before going to KNN as an outlier

Observe the following code to understand KNN better –

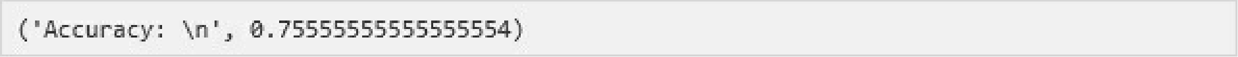


```
#Importing Libraries

from sklearn.neighbors import KNeighborsClassifier

#Assumed you have, X (predictor) and Y (target) for training data set and
x_test(predictor) of test_dataset

# Create KNeighbors classifier object model

KNeighborsClassifier(n_neighbors=6) # default value for n_neighbors is 5

# Train the model using the training sets and check score

model.fit(X, y)

#Predict Output

predicted= model.predict(x_test)
```



```
df = pd.read_csv('iris_df.csv')

df.columns = ['X1', 'X2', 'X3', 'X4', 'Y']

df = df.drop(['X4', 'X3'], 1)

df.head()

sns.set_context('notebook', font_scale=1.1)

sns.set_style('ticks')

sns.lmplot('X1','X2', scatter=True, fit_reg=False, data=df, hue='Y')

plt.ylabel('X2')

plt.xlabel('X1')

from sklearn.cross_validation import train_test_split

neighbors = KNeighborsClassifier(n_neighbors=5)

X = df.values[:, 0:2]

Y = df.values[:, 2]

trainX, testX, trainY, testY = train_test_split( X, Y, test_size = 0.3)

neighbors.fit(trainX, trainY)

print('Accuracy: \n', neighbors.score(testX, testY))

pred = neighbors.predict(testX)
```

The code given above will give the output below –



```
('Accuracy: \n', 0.75555555555555554)
```

K-Means

This is a kind of unsupervised algorithm that handles clustering issues. Its procedure follows an easy and simple way to classify a given dataset using several clusters (suppose k clusters). The data points of a cluster are homogeneous and heterogeneous for matched groups.

How K forms a cluster

K-means forms cluster in the steps below -

- K-means selects the number of points for each cluster called centroids.
- A cluster is formed by each data point with the nearest centroids, k k.
- Finds the centroid of all clusters based on existing cluster members. Here we have new centroids.

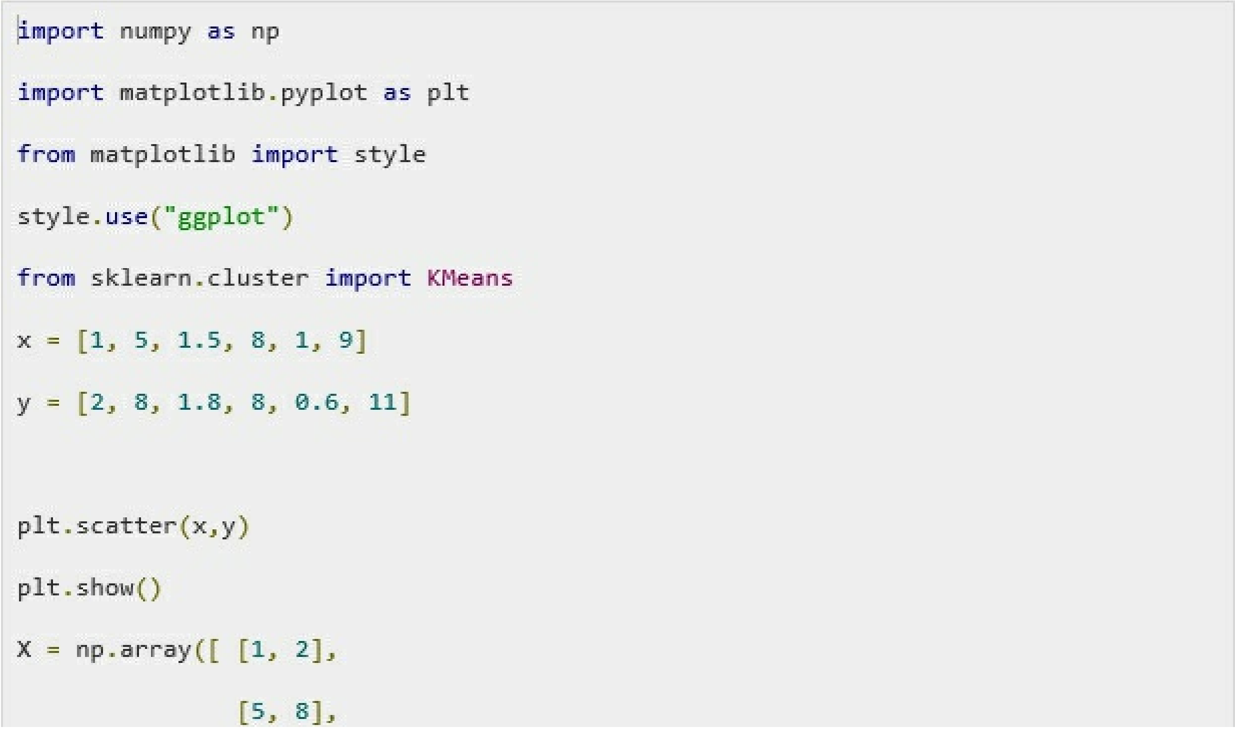
Since we have new centroids, repeat steps 2 and 3. Find the nearest distance from each data point from the new centroids and join the new k-clusters. Keep repeating this process until convergence occurs, that is, until the centroids do not change.

Determination of the value of K

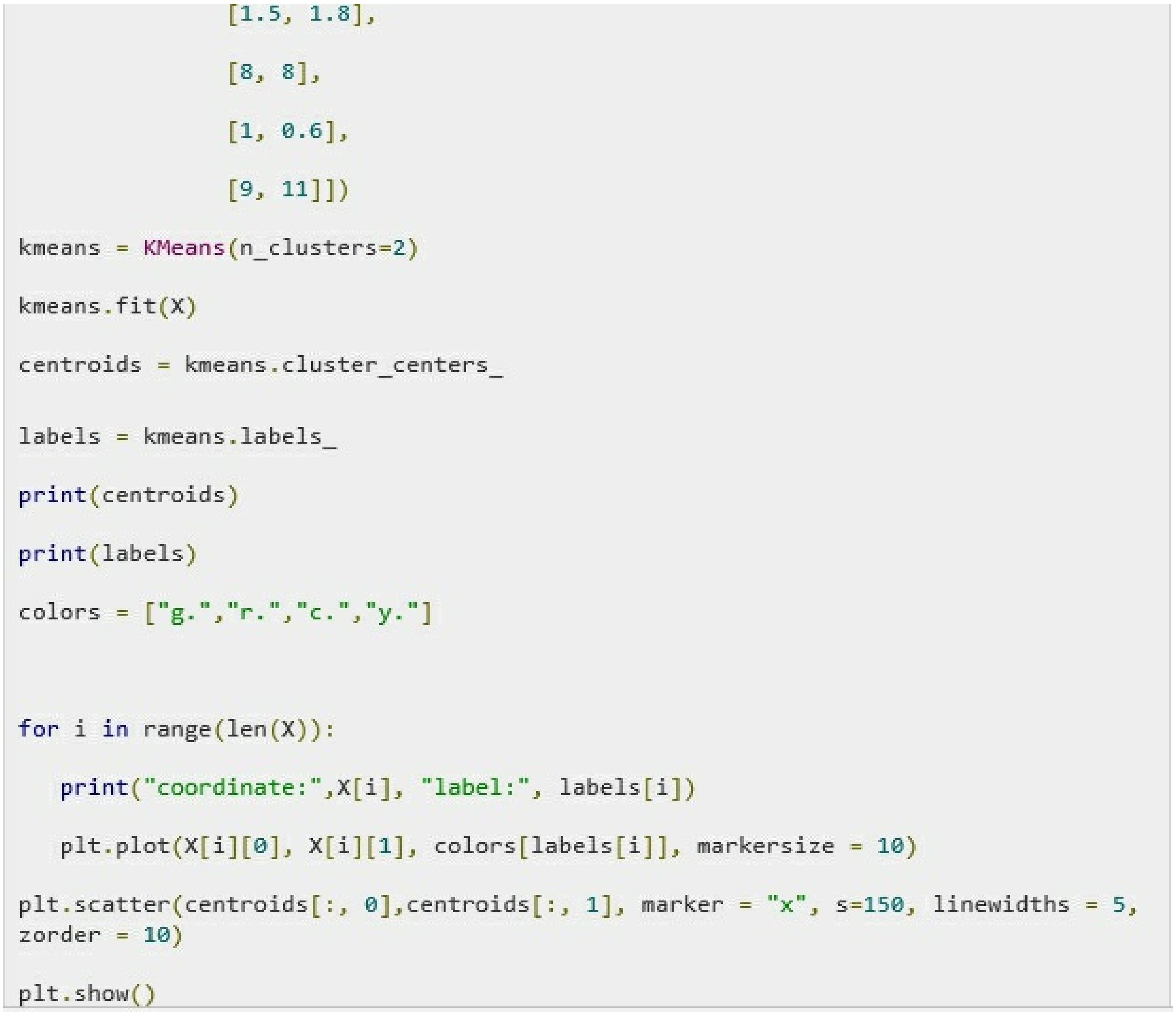
In K-means, there are clusters, and every cluster has its centroid. The addition of the square of the difference between the center of gravity and the data points of a cluster is the sum of the square value of that cluster. Also, when the sum of the square values of all clusters is added, it becomes total within the sum of the square value of the cluster solution.

We know that the value continues to decrease as the number of clusters increases, but if you plot the result, you will see that the sum of the square distance decreases abruptly to a value of ke, then much more slowly after that. Here we can find the optimal cluster number.

Note the following code –

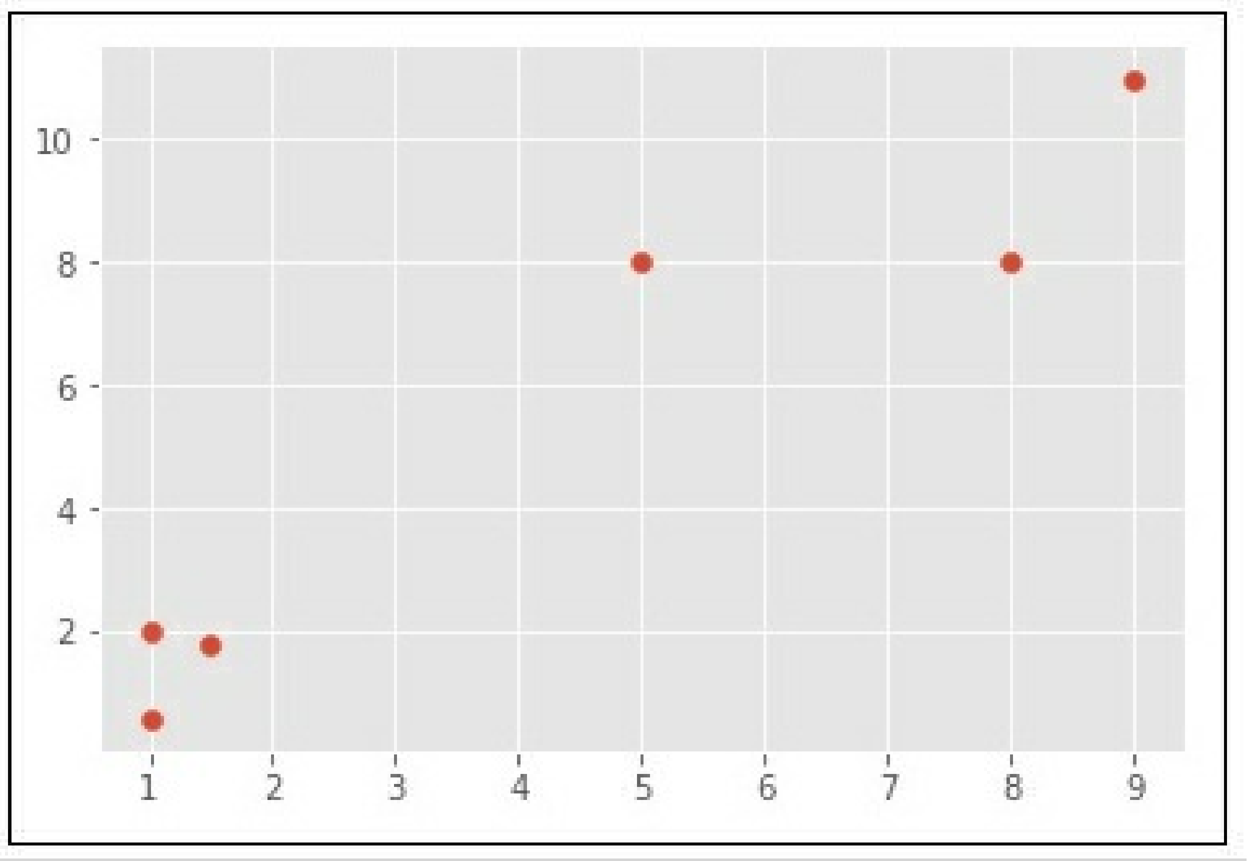


```
import numpy as np

import matplotlib.pyplot as plt

from matplotlib import style

style.use("ggplot")

from sklearn.cluster import KMeans

x = [1, 5, 1.5, 8, 1, 9]

y = [2, 8, 1.8, 8, 0.6, 11]


plt.scatter(x,y)

plt.show()

X = np.array([ [1, 2],

               [5, 8],
```



```
               [1.5, 1.8],

               [8, 8],

               [1, 0.6],

               [9, 11]])

kmeans = KMeans(n_clusters=2)

kmeans.fit(X)

centroids = kmeans.cluster_centers_

labels = kmeans.labels_

print(centroids)

print(labels)

colors = ["g.","r.","c.","y."]



for i in range(len(X)):

   print("coordinate:",X[i], "label:", labels[i])

   plt.plot(X[i][0], X[i][1], colors[labels[i]], markersize = 10)

plt.scatter(centroids[:, 0],centroids[:, 1], marker = "x", s=150, linewidths = 5,
zorder = 10)

plt.show()
```

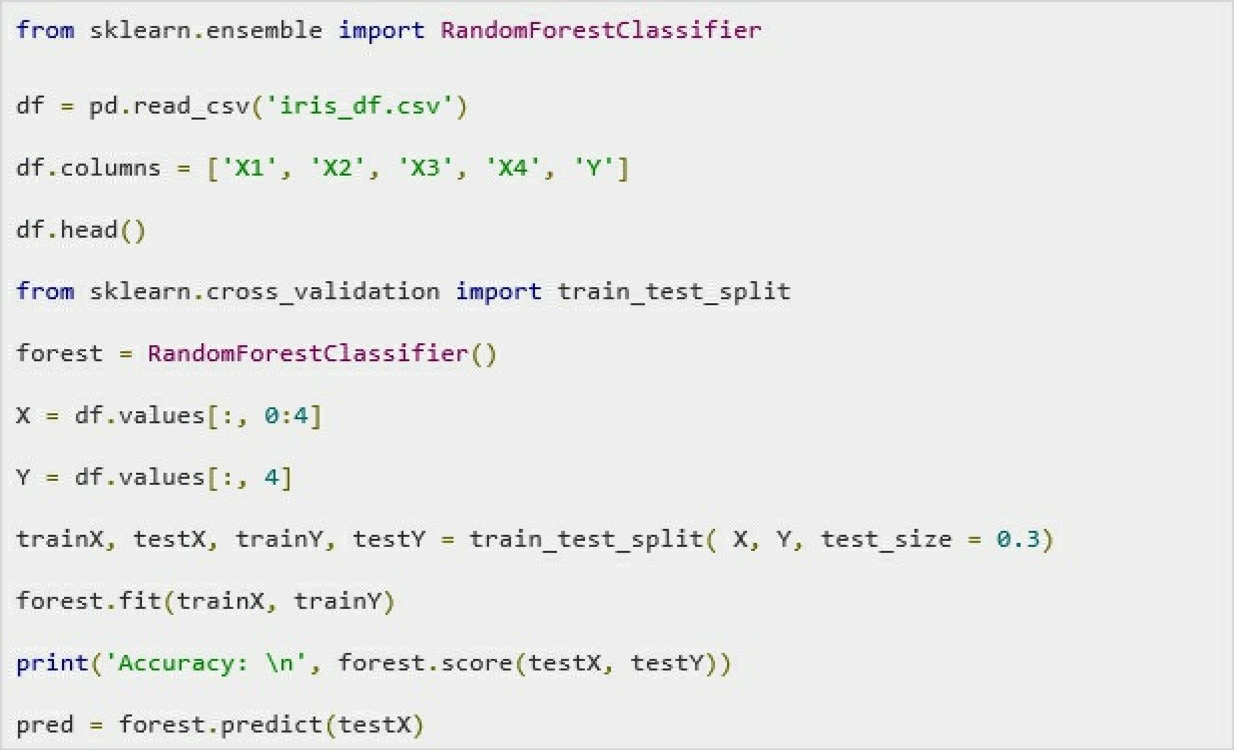
You can see the following output after running the code given above -



Random forest

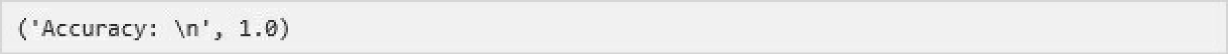
Random Forest is a popular ensemble supervised learning algorithm. "Together" means that it takes a lot of "weak students" and that they work together to form a powerful predictor. In this case, weak students are all randomly implemented decision trees that are pooled to form the strongest predictor - a random forest.

Note the following code –



```
from sklearn.ensemble import RandomForestClassifier

df = pd.read_csv('iris_df.csv')

df.columns = ['X1', 'X2', 'X3', 'X4', 'Y']

df.head()

from sklearn.cross_validation import train_test_split

forest = RandomForestClassifier()

X = df.values[:, 0:4]

Y = df.values[:, 4]

trainX, testX, trainY, testY = train_test_split( X, Y, test_size = 0.3)

forest.fit(trainX, trainY)

print('Accuracy: \n', forest.score(testX, testY))

pred = forest.predict(testX)
```

Below is the output for the code –

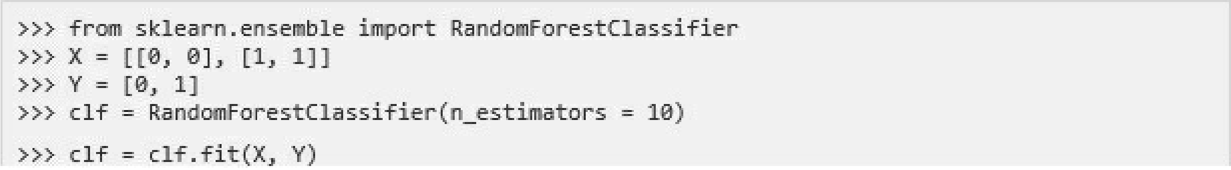


```
('Accuracy: \n', 1.0)
```

The overall methods are intended to combine the predictions of several basic estimators constructed with a given learning algorithm to improve the generalization/robustness of a single estimator.

The sklearn.ensemble module consists of two averaging algorithms depending on randomized decision trees - the RandomForest algorithm and the Extra-Trees method. Both algorithms are perturbation and combination techniques [B1998] designed specifically for trees. This means that a diverse set of classifiers is created by introducing randomness into the construction of the classifier. The prediction of the set is given as the average prediction of the individual classifiers.

Forest classifiers must have two arrays: a fragmented or dense X-sized array [n_samples, n_features] containing the training samples and an array of size n [n_samples] containing the target values (class labels) of the samples as shown in the code below –



```
>>> from sklearn.ensemble import RandomForestClassifier
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = RandomForestClassifier(n_estimators = 10)
>>> clf = clf.fit(X, Y)
```

Like decision trees, trees also extend to multiple output problems (if Y is an array of size [n_samples, n_outputs]).

Unlike the original publication [B2001], scikit-learn implementation combines classifiers by averaging their probabilistic prediction rather than allowing each classifier to vote in a single class.

Random Forest is a trademarked term for a set of decision trees. In Random Forest, there are a collection of decision trees, called "Forest." To sort a new object based on attributes, each tree provides a sort, and the "vote" tree for that class is said to be sorted. The forest chooses the ranking with the most votes (on all trees in the forest).

Each tree is planted and grown as follows -

- If the number of cases in the learning set is N, the sample of N cases will be taken at random but with replacement. This sample will be the training required for tree growth.
- If there are input variables M, a number m

<< M is specified so that at each node, m variables are randomly selected outside M and that the best division of these values m is used for divide the knot. The value of m is kept constant during the growth of the forest.

- Each tree is developed to the greatest extent possible. There is no pruning.

Dimensionality Reduction Algorithm

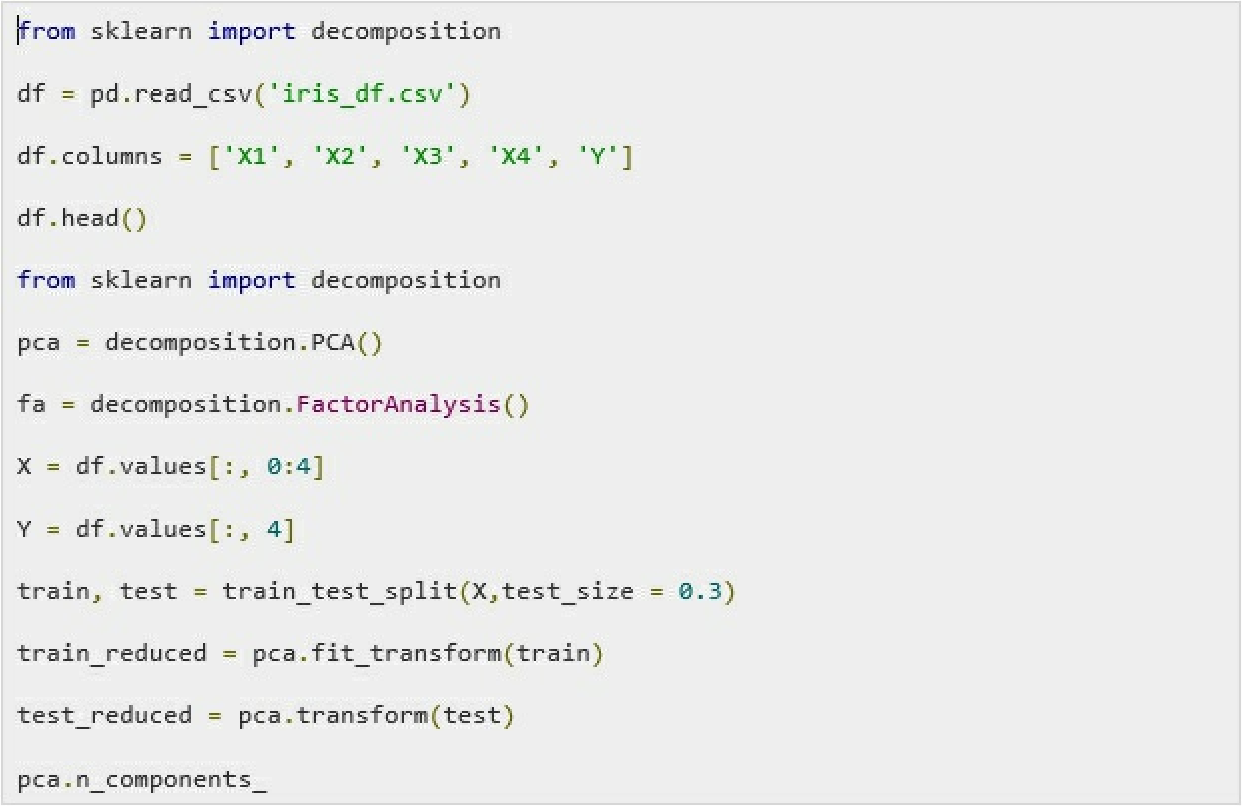
Reducing dimensionality is another common task of unsupervised learning. Some problems may contain tens of millions of input or explanatory variables, which can be expensive in calculations and calculations. Also, the capacity of the generalization program may be reduced if some of the input variables capture the noise or are not relevant to the underlying relationship.

Dimensionality reduction is the process of finding input variables that are responsible for the most important changes in the output variable or response. The reduction in dimensionality is often used to visualize data. A regression problem is easier to visualize, such as predicting the property price from its size, where the property size can be plotted along the x-axis of the chart, and the price of the property. The property can be traced along the y-axis. Similarly, it is easy to see the problem of the regression of the price of real estate when a second explanatory variable is added. The number of pieces of the property can be plotted on the z-axis, for example. A problem which consists of thousands of input variables, however, becomes impossible to visualize.

The reduced dimensionality reduces a very large number of explanatory variable inputs to a smaller set of the input variables which retain as much data as possible.

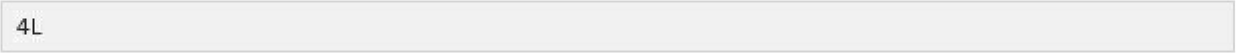
An example of a dimensionality reduction algorithm which can carry out useful activities for data analysis is PCA. More importantly, it can dramatically reduce the number of calculations involved in a model when hundreds, if not thousands, of different input variables, are required. As an unsupervised learning task, you should always analyze the results and verify that you are managing at least 95% of the behavior of the original dataset.

Observe the following code for better understanding –



```
from sklearn import decomposition

df = pd.read_csv('iris_df.csv')

df.columns = ['X1', 'X2', 'X3', 'X4', 'Y']

df.head()

from sklearn import decomposition

pca = decomposition.PCA()

fa = decomposition.FactorAnalysis()

X = df.values[:, 0:4]

Y = df.values[:, 4]

train, test = train_test_split(X,test_size = 0.3)

train_reduced = pca.fit_transform(train)

test_reduced = pca.transform(test)

pca.n_components_
```

The code below is the output for the code given above –



```
4L
```

Over the last six years, there has been an exponential increase in data capture at all possible levels and points. Government agencies/research organizations/companies not only offer new sources of data; they also capture very detailed data at different times and different stages.

For example, e-commerce companies capture more details about customers, such as demographics, browsing history, likes or dislikes, shopping history, reviews, and other details for personalized attention. Currently, available data can have thousands of features. It is difficult to reduce them while keeping as much information as possible. In such situations, reducing dimensionality helps a lot.

Boosting Algorithms

The term "Boosting" denotes a family of algorithms that convert weak students into strong learners. Let's understand this definition by solving a spam problem, as shown below -

What procedure should be followed to sort an email as spam or not? In the initial approach, we identify spam and non-spam using the following criteria if:

- The e-mail has only one image file (advertising image); it is a spam.
- The email has only one link, is a spam
- The body of the e-mail consists of a sentence of the type "You have won a cash prize of xxxxxx $." It's a spam
- Emails from our official domain "Tutorialspoint.com" is not spam
- Email from a known source, not spam

Above we have defined several rules for classifying an email as spam or non-spam. However, these rules are not perfect enough to classify spam mail successfully or not. Therefore, these rules are called weak apprentices.

To convert a weak student into a strong student, we combine each low student's prediction using methods such as:

- Use of the weighted average / weighted average
- Given the highest voting forecasts

For example, suppose we have defined seven weak students. Of these seven people, five are elected "SPAM," and two are elected "Not a SPAM." In this case, by default, we will consider an email as spam because we have more than (5) votes for "SPAM."

How it works

Boosting combines the weak learner or the basic learner to form a strong rule. This section explains how reinforcement identifies weak rules.

To find a weak rule, we apply basic learning algorithms (ML) with a different distribution. Each time-based learning algorithm is applied; it generates a new weak forecasting rule. This uses the iteration process many times. After many iterations, the optimization algorithm combines these weak rules into one strong prediction rule.

To choose the right distribution for each round, follow the steps -

Step 1 - The basic student takes all distributions and assigns equal weight to each one.

Step 2 - If there is a prediction error caused by the learning algorithm of the first base, we will have a larger weight for the observations with a prediction error. Then we apply the next-base-learning algorithm.

We go through step 2 until we reach the threshold of the basic learning algorithm or higher accuracy.

Finally, it combines the results of the weak student and allows a strong learner to improve the predictive power of the model. The incentive puts more emphasis on examples that are misclassified or have more serious errors due to loose rules.

Types of Boosting algorithms

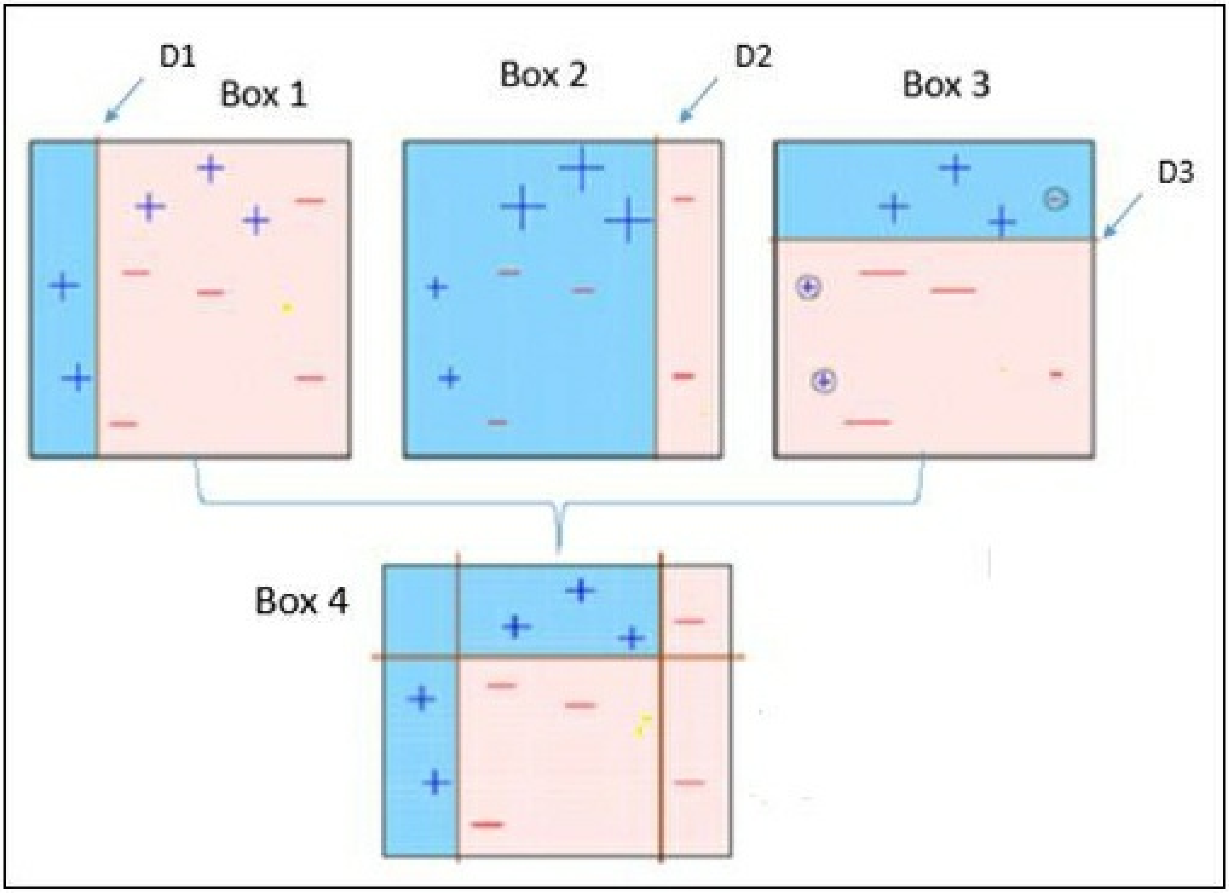
There are several types of mechanisms used to drive algorithms - decision segment, margin maximization classification algorithm, and so on. Different boost algorithms are listed here -

- AdaBoost (adaptive improvement)
- Gradient tree boosting
- XGBoost

This section focuses on AdaBoost and gradient enhancement, followed by their respective enhancement algorithms.

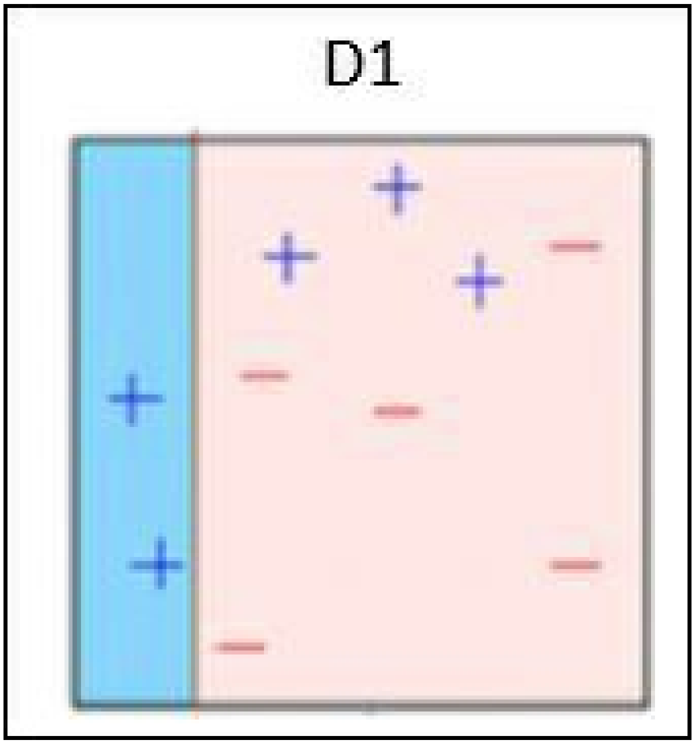
AdaBoost

Observe the following figure that explains the Ada-boost algorithm.

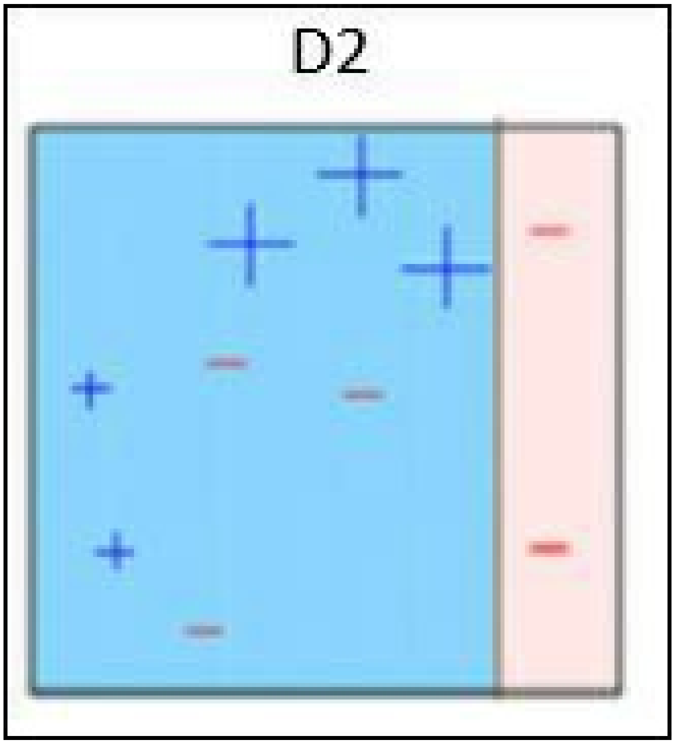


This is explained below –

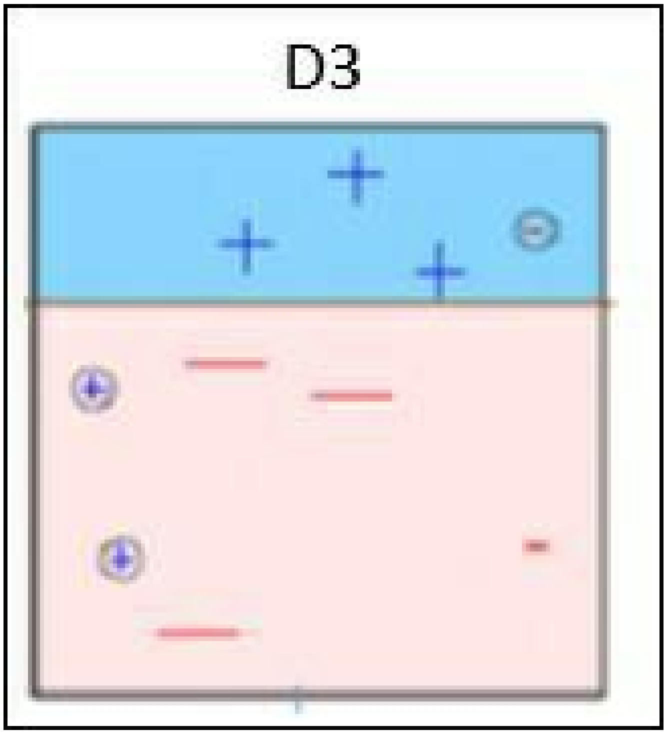
Box 1 - You can see that we assign equal weights to each data point and apply a decision stub to classify them as + (plus) or - (minus). The decision stub (D1) created a vertical line on the left side to sort the data points. This vertical line incorrectly predicts three + (plus) as - (minus). Then, we will assign a higher weight to these three + (plus) and apply another decision segment.



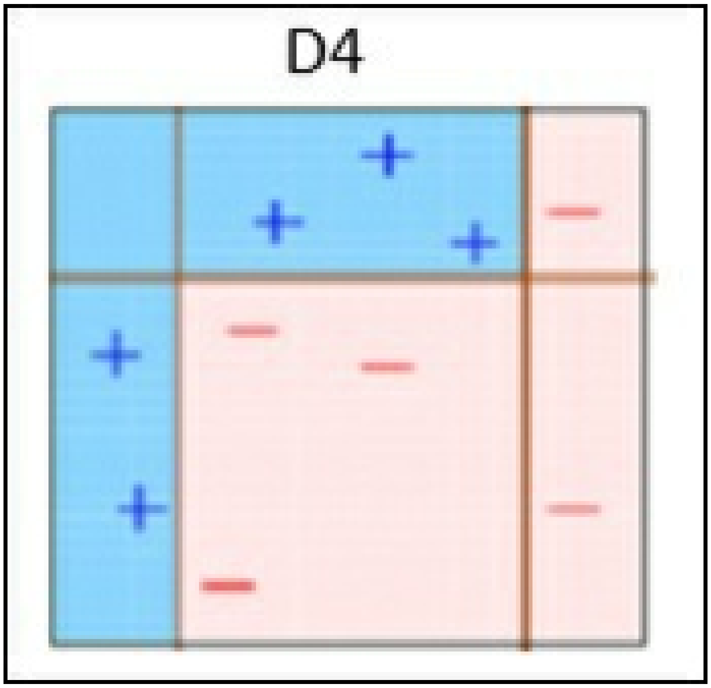
Box 2 - Here, you can see that the size of three data points (predicted by mistake) + (plus) is larger than the rest of the data points. In this case, the second decision point (D2) will attempt to predict correctly. Now, a vertical line (D2) on the right side of this box has ranked three misclassified + (plus) correctly. But again, he made incorrect sorting mistakes. This time with three - (less) data points. Again, higher weights will be assigned to the three fewer data points and apply another decision point.



Box 3 - Here, three (least) data points receive larger weights. A decision stub (D3) is applied to predict these misclassified observations properly. This time, a horizontal line is produced to rank the + (plus) and - (minus) data points based on the larger weight of the misclassified observations.



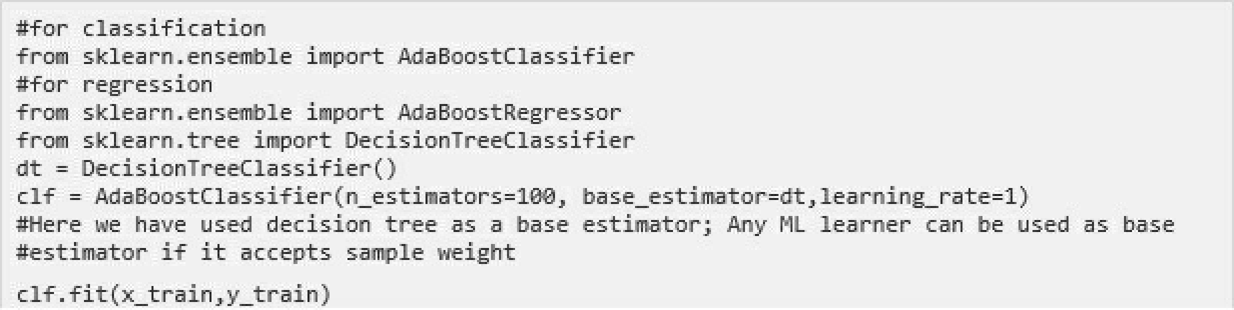
Box 4 - We reach here D1, D2, and D3 to form a strong prediction with complex rules compared to weak individual students. It can be seen that this algorithm ranked these observations highly relative to any of the weak individual students.



AdaBoost or Adaptive Boosting - Works according to the similar method described above. It is part of a sequence of weak students in different weighted training data. It begins by predicting the original dataset and assigns equal weight to each observation. If the prediction is incorrect using the first apprentice, this will give more weight to the observations that were incorrectly predicted. As an iterative process, it continues to add students until a threshold is reached for the number of models or accuracy.

We mainly use decision stamps with AdaBoost. However, we can use any machine learning algorithm as basic learning if it accepts the weight in the training data set. We can use AdaBoost algorithms for classification and regression issues.

The Python code below can be used for this purpose –



```
#for classification
from sklearn.ensemble import AdaBoostClassifier
#for regression
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
clf = AdaBoostClassifier(n_estimators=100, base_estimator=dt,learning_rate=1)
#Here we have used decision tree as a base estimator; Any ML learner can be used as base
#estimator if it accepts sample weight

clf.fit(x_train,y_train)
```

Parameters for Tuning

The parameters can be adjusted to optimize the performance of the algorithm. The main adjustment parameters are:

n_estimators - They control the number of weak students.

learning_rate - Controls the contribution of weak students in the final combination. There comes a trade-off between learning_rate and n_estimators.

base_estimators - This help to specify the different ML algorithms.

Students' basic settings can also be adjusted to optimize their performance.

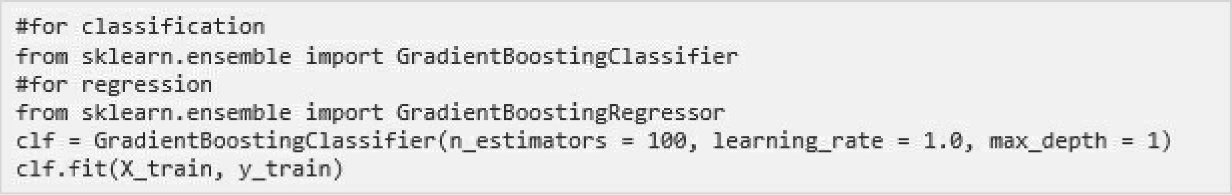
Gradient Boosting

As the gradient increases, many patterns are formed sequentially. Each new model progressively minimizes the loss function (y = ax + b + e, where "e" is the error term) of the whole system using the gradient descent method. The learning method sequentially adjusts the new models to provide a more precise evaluation of the response variable.

The key impression behind this algorithm is to create new basic learners that can be optimally correlated with the loss function negative gradient, and relevant for the whole.

In the Sklearn Python library, we use Gradient Tree Boosting or GBRT, which is a generalization of the stimulus for arbitrarily differentiable loss functions. Can be used for classification and regression issues.

You can make use of the code below for this purpose –



```
#for classification
from sklearn.ensemble import GradientBoostingClassifier
#for regression
from sklearn.ensemble import GradientBoostingRegressor
clf = GradientBoostingClassifier(n_estimators = 100, learning_rate = 1.0, max_depth = 1)
clf.fit(X_train, y_train)
```

Here are the terms used in the code above -

n_estimators - They control the number of weak students.

learning_rate - Controls the contribution of weak students in the final combination. There is a trade-off between n_estimators and learning_rate.

max_depth - This is the maximum depth of individual regression estimators limiting the number of nodes in the tree. This parameter is adjusted to optimize performance, and the best value depends on the interaction of the input variables.

The loss function can be adjusted for better performance.