

**A Project on**

**Classification of iris flower species using Machine Learning with ScikitLearn**

**For the fulfillment of internship program**

**On**

**Python**

**At**

**CDAC(Bhubaneswar)**

*SUBMITTED BY: -*

***Priyabrata Mohapatra***

***Guided By:***

***Mr ranjan Kumar das***

***Abstract***

* Machine Learning is defined as an application of artificial intelligence where available information is used through algorithms to process or assist the processing of statistical data.
* While Machine Learning involves concepts of automation, it requires human guidance.
* Machine learning is the set of rules that a computer develops in order to solve problems.
* The basic idea is that the machine learning computer will find out the patterns in data. And predict the outcomes.
* The machine leaning is a critical component to any Artificial Intelligence development.
* Our job is to study the classification of iris flower species by supervised machine learning algorithms like Logistic Regression, Decision Tree, Gaussian Naive, Support Vector Machine(SVM), K-Nearest Neighbour(KNN).

***CONTENTS***

*Introduction*

*1.1 Machine learning*

*1.2 Supervised learning*

*1.3 Unsupervised learning*

*Proposed Work*

*Soft ware Requirements*

*2.1 Python*

*2.2NumPy*

*2.3 Sklearn*

*2.4 Matplotlib*

*2.5 Pandas*

*Machine learning Algorithms*

*3.1 Logistic regression*

*3.2 Svm*

*3.3 Knn*

*3.4 Decisiontree classifier*

*3.5 Gaussian naïve*

*Implementation*

*List Figures*

*Table*

*Conclusion*

*References*

***INTRODUCTION***

**Machine Learning**:

Field of study that gives computers the ability to learn without being explicitly programmed.

In other words, creating an algorithm that learns from data to make a prediction in known as Machine Learning.

**Why do we need machine learning?**

* In a single line, because we cannot do everything, and machines can't not do what they are ordered. They save both money and time by reducing human working hours.
* So elaborating the concept of why do we need machine learning is that, for growing volumes and varieties of data has made it difficult for coders to manually code every program. Here comes Machine learning to ease human effort.
* It recognises and rectifies mistakes without human intervention, and also adapts with experience so that errors or bugs are never repeated in the future. This makes the whole process of coding and compiling faster, smoother and bug free. Moreover machine can update and upgrade algorithms every minute.
* Extension use of **AI** and **Machine learning** proves its importance. From cyber security to virtual assistants all have learning capabilities to serve human better.

**Applications of Machine Learning:**

* We use Machine Learning in our daily lives in a way or the other. Best example –**YouTube**, whichrecommends videos based on our previous watched videos with the help of Machine Learning algorithm
* Other examples include Autonomous helicopter, handwriting recognition, Face recognition, Computer Vision.
* Self-customizing programs E.g., **Amazon, Netflix product recommendations**
* Understanding human learning (brain, real AI).
* Recently, with the help of Machine Learning, AI doctors have designed a model which can predict whether a person suffers from malignant (a type of cancer) or not, whose accuracy is found to be more than the accuracy of doctors.
* Machine learning is a multidisciplinary field,which can be applied in various areas such as:

1. Machine learning application in robotics.
2. Machine learning in data mining
3. Machine learning in application in finance
4. Some more areas like Health care, Marketing personalization, Fraud detection, smart cars

**Advantages of Machine Learning:**

* It is used in so many industries of applications such as banking and financial sector, healthcare, retail, publishing and social media, etc.
* It is used by Google and Facebook to push relevant advertisements based on users search history.
* It allows time cycle reduction and efficient utilization of resources

**Challenges of Machine Learning:**

* Getting relevant data is the major challenge.
* Understanding of results is also a major challenge to determine effectiveness of machine learning algorithms.
* Based on which action to be taken and when to be taken, various machine learning techniques are need to be tried.

**Machine Learning Algorithm:**

**1. Supervised Learning**

**2. Unsupervised Learning**

Other Machine Learning Algorithm includes Reinforcement Learning, Recommender Systems.

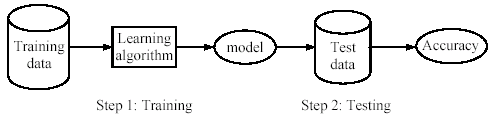
**Supervised Learning:**

* **Supervised learning** is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.
* Supervised learning is one of the methods associated with machine learning which involves allocating labelled data so that a certain pattern or function can be deduced from that data.  
  The bottom line property of supervised learning is that the input data is known and labelled appropriately.  
  It infers a function from labelled*training data* consisting of a set of *trainingexamples*.

It includes two parts:

**1. Learning (training):** Learn a model using the training data

**2. Testing:** Test the model using unseen test data to assess the model accuracy

****

The most commonly used algorithms are:

1. Support Vector Machines
2. Linear Regression
3. Logistic Regression
4. Naive Bayes
5. Decision Trees
6. K-nearest neighbour algorithm
7. Neural Networks

**Unsupervised Learning:**

Unsupervised learning is the second method of machine learning algorithm where inferences are drawn from unlabeled input data.   
One of the defining characters of unsupervised learning is that both the input and output are not known.

**Algorithms used in unsupervised learning vary, including:**

1. Clustering

* K-means Clustering
* Hierarchical Clustering

1. Anomaly Detection
2. Neural Network Auto encoder
3. Self-organizing map

**Proposed Work**

We are going to collect the dataset required for the classification of the iris flower species from the **UCI** repository (<https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>) or load it from the sklearn module.

The iris dataset contains four features namely sepal length, sepal width, petal length, petal width. From these four features, the species(targets) are identified as setosa, versicolor or virginica.

We are going to use the Pandas library to read the datasets from the repository by using read\_csv (“file path”) of pandas library.

Then the dataset will be splitted into two parts:

1. Training
2. Testing

The dataset will be splitted in such a way to prevent overfitting of the model (70% for training and 30% for testing). This can be done by importing train\_test\_split module from sklearn.cross\_validation module.

The various models for classification which will be used to train and test the model are:

1. Logistic Regression (from sklearn.linear\_model module)
2. Decision Tree Classifier (from sklearn. tree module)
3. K Nearest Neighbor (from sklearn.neighbors module)
4. Gaussian Naive Bayes (from sklearn.naive\_bayes module)
5. Support Vector Classification(SVC) (from sklearn.svm module)

First the model will be trained using fit method. And then it will be tested using predict method.

Then the accuracy would be found out by accuracy\_score method which can be done by importing from sklearn. metrics module.

Then we have to plot a graph between the Machine Learning Algorithms Used Vs Accuracy in % from which the algorithm having the best accuracy can be found.

**REQUIREMENTS**

### **Our project is based on the study of classification of iris flower species by using machine learning. Here in our project we use python language.**

**PYTHON**

* Python is a simple,powerful, object oriented and a very high-level language.
* It supports both pop and oop.
* It is an open source language and supports dynamic typing.
* This language is platform independent.

**Python supports following packages:**

# **NUMPY** NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

**Installation:**

* **Mac**and **Linux**users can install NumPy via pip command:

pip install NumPy

**1.Arrays in NumPy:** NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers.

* In NumPy dimensions are called axes. The number of axes is rank.
* NumPy’s array class is called **ndarray**. It is also known by the alias **array**

**2.Array creation:**There are various ways to create arrays in NumPy.

* For example, you can create an array from a regular Python [**list**](http://quiz.geeksforgeeks.org/python-set-3-strings-lists-tuples-iterations/) or **tuple** using the **array** function. The type of the resulting array is deduced from the type of the elements in the sequences.
* Often, the elements of an array are originally unknown, but its size is known. Hence, NumPy offers several functions to create arrays with **initial placeholder content**. These minimize the necessity of growing arrays, an expensive operation.  
  **For example:** np. zeros, np. ones, np.full, np.empty, etc.
* **arrange:** returns evenly spaced values within a given interval. **step** size is specified.
* **linespace:** returns evenly spaced values within a given interval. **num** no. of elements are returned.
* **Reshaping array:** We can use **reshape** method to reshape an array

**Flatten array:** We can use **flatten** method to get a copy of array collapsed into **one dimension**.

**3.Array Indexing:** Knowing the basics of array indexing is important for analysing and manipulating the array object. NumPy offers many ways to do array indexing.

* **Slicing:** Just like lists in python, NumPy arrays can be sliced. As arrays can be multidimensional, you need to specify a slice for each dimension of the array.
* **Integer array indexing:** In this method, lists are passed for indexing for eac dimension.
* **Boolean array indexing:** This method is used when we want to pick elements from array which satisfy some condition.

**4. Basic operations:** Plethora of built-in arithmetic functions are provided in NumPy.

* **Operations on single array:** We can use overloaded arithmetic operators to do element-wise operation on array to create a new array. In case of +=, -=, \*= operators, the existing array is modified.

**Unary operators:** Many unary operations are provided as a method of **ndarray** class. This includes sum, min, max, etc. These functions can also be applied row-wise or column-wise by setting an axis parameter

**Binary operators:** These operations apply on array elementwise and a new array is created. You can use all basic arithmetic operators like +, -, /, etc. In case of +=, -=,= operators, the existing array is modified

* **Universal functions (ufunc):** NumPy provides familiar mathematical functions such as sin, cos, exp, etc. These functions also operate elementwise on an array, producing an array as output.

**Note:** All the operations we did above using overloaded operators can be done using ufuncs like np.add, np.subtract, np.multiply, np.divide, np.sum etc.

**5.Sorting array:**There is a simple **np.sort** method for sorting NumPy arrays

**Scikit-Learn**

**Scikit-learn** is a [free software](https://en.wikipedia.org/wiki/Free_software) [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [library](https://en.wikipedia.org/wiki/Library_(computing)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language. It features various [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and [clustering](https://en.wikipedia.org/wiki/Cluster_analysis) algorithms including [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine), [random forests](https://en.wikipedia.org/wiki/Random_forests), [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting), [*k*-means](https://en.wikipedia.org/wiki/K-means_clustering) and [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN), and is designed to interoperate with the Python numerical and scientific libraries [NumPy](https://en.wikipedia.org/wiki/NumPy) and [SciPy](https://en.wikipedia.org/wiki/SciPy).

Scikit learn is:

* Simple and efficient tools for data mining and data analysis
* Accessible to everybody, and reusable in various contexts
* Built on NumPy , SciPy , and matplotlib
* Open source, commercially usable - BSD license

It is used in:

**1.**[**Classification**](http://scikit-learn.org/stable/supervised_learning.html#supervised-learning)

Identifying to which category an object belongs to.

**Applications**: Spam detection, Image recognition.  
**Algorithms**:  [SVM](http://scikit-learn.org/stable/modules/svm.html#svm), [nearest neighbors](http://scikit-learn.org/stable/modules/neighbors.html#classification) ,[random forest](http://scikit-learn.org/stable/modules/ensemble.html#forest)

**2.**[**Regression**](http://scikit-learn.org/stable/supervised_learning.html#supervised-learning)

Predicting a continuous-valued attribute associated with an object.

**Applications**: Drug response, Stock prices.  
**Algorithms**:  [SVR](http://scikit-learn.org/stable/modules/svm.html#svm), [ridge regression](http://scikit-learn.org/stable/modules/linear_model.html#ridge-regression), [Lasso](http://scikit-learn.org/stable/modules/linear_model.html#lasso)

**3.**[**Clustering**](http://scikit-learn.org/stable/modules/clustering.html#clustering)

Automatic grouping of similar objects into sets.

**Applications**: Customer segmentation, Grouping experiment outcomes  
**Algorithms**:  [k-Means](http://scikit-learn.org/stable/modules/clustering.html#k-means), [spectral clustering](http://scikit-learn.org/stable/modules/clustering.html#spectral-clustering),[mean-shift](http://scikit-learn.org/stable/modules/clustering.html#mean-shift)

**4.**[**Dimensionality reduction**](http://scikit-learn.org/stable/modules/decomposition.html#decompositions)

Reducing the number of random variables to consider.

**Applications**: Visualization, Increased efficiency  
**Algorithms**: [PCA](http://scikit-learn.org/stable/modules/decomposition.html#pca), [feature selection](http://scikit-learn.org/stable/modules/feature_selection.html#feature-selection), [non-negative matrix factorization](http://scikit-learn.org/stable/modules/decomposition.html#nmf).

5.[**Model selection**](http://scikit-learn.org/stable/model_selection.html#model-selection)

Comparing, validating and choosing parameters and models.

**Goal**: Improved accuracy via parameter tuning  
**Modules**:  [grid search](http://scikit-learn.org/stable/modules/grid_search.html#grid-search), validation, metrics.

**6.** [**Pre-processing**](http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing)

Feature extraction and normalization.

**Application**: Transforming input data such as text for use with machine learning algorithms.  
**Modules**:  [pre-processing](http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing), feature extraction.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

## **General Concepts:**

matplotlib has an extensive codebase that can be daunting to many new users. However, most of matplotlib can be understood with a fairly simple conceptual framework and knowledge of a few important points.

## 

## **Usage of plotting:**

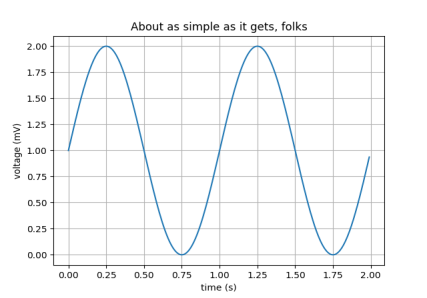
Plotting requires action on a range of levels, from the most general (e.g., ‘contour this 2-D array’) to the most specific (e.g., ‘color this screen pixel red’). The purpose of a plotting package is to assist you in visualizing your data as easily as possible, with all the necessary control – that is, by using relatively high-level commands most of the time, and still have the ability to use the low-level commands when needed.

Therefore, everything in matplotlib is organized in a hierarchy. At the top of the hierarchy is the matplotlib “state-machine environment” which is provided by the[matplotlib.pyplot](https://matplotlib.org/api/pyplot_api.html#module-matplotlib.pyplot) module

## **Types of plots in matplotlib**

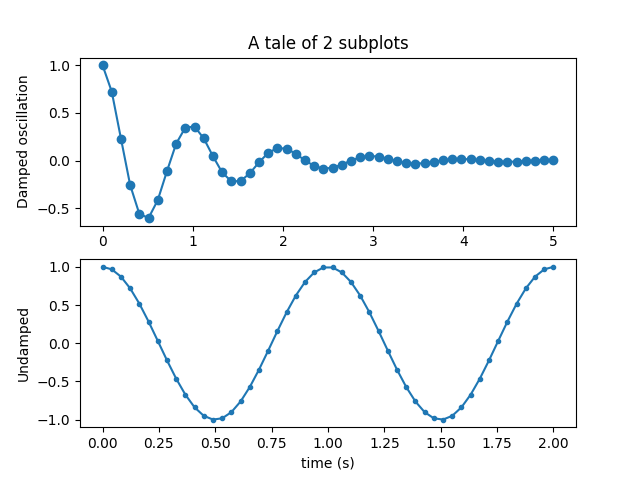
## **Line Plot**

Here’s how to create a line plot with text labels using [plot()](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot).



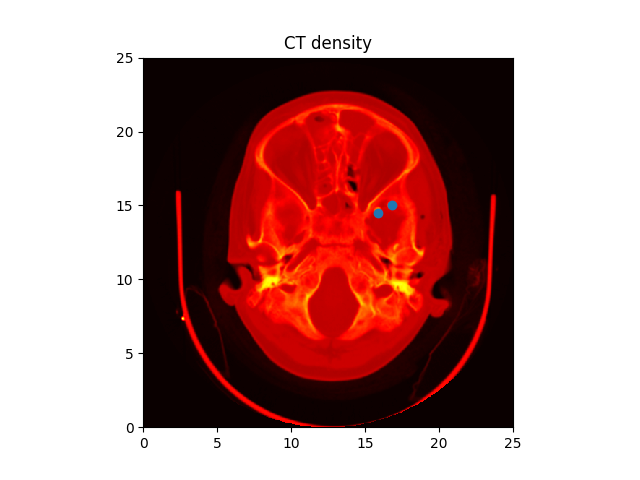
## **Multiple subplots in one figure**

Multiple axes (i.e. subplots) are created with the[subplot()](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.subplot.html#matplotlib.pyplot.subplot) function:



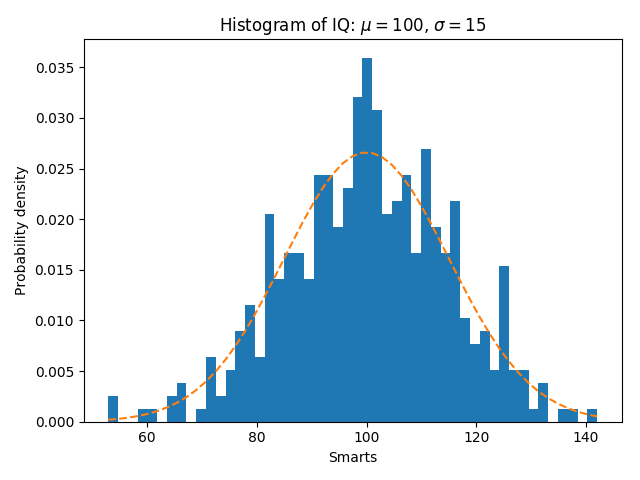
## **Images**

Matplotlib can display images (assuming equally spaced horizontal dimensions) using the[imshow()](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.imshow.html#matplotlib.pyplot.imshow) function.



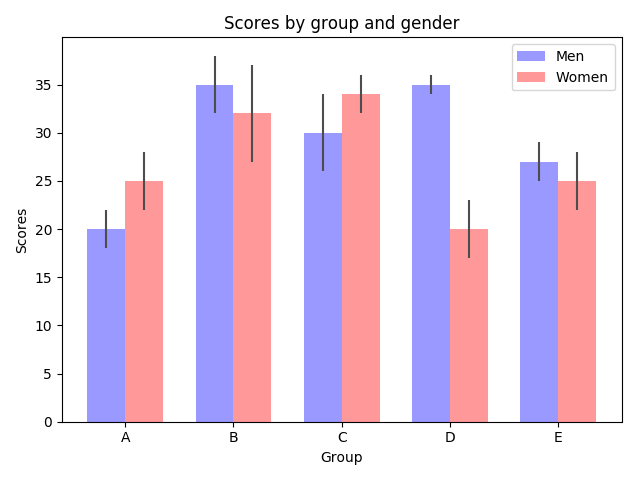
## **Histograms**

The[hist()](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.hist.html#matplotlib.pyplot.hist) function automatically generates histograms and returns the bin counts or probabilities:



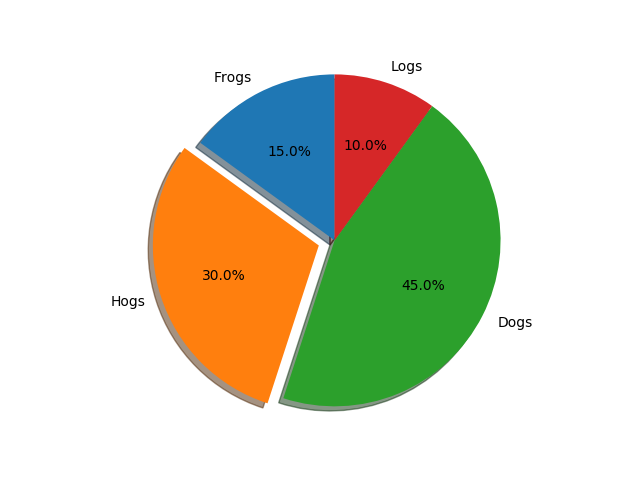
## **Bar charts**

Use the[bar()](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.bar.html#matplotlib.pyplot.bar) function to make bar charts, which includes customizations such as error bars:



## **Pie charts**

The[pie()](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.pie.html#matplotlib.pyplot.pie) function allows you to create pie charts. Optional features include auto-labeling the percentage of area, exploding one or more wedges from the center of the pie, and a shadow effect.



With the help of all different matplotlib module we can plot many different types of plots. There are numerous types of plotting techniques in matplotlib which can be used in different areas. Other plotting techniques are like Filled curves, GUIwidgets,Scatterplot,Datehandling

**PANDAS**

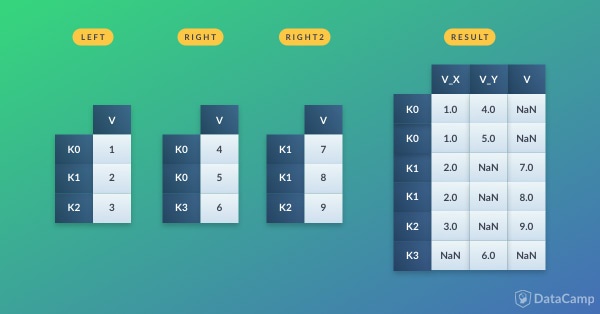
It is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.  
  
**Application:**  
Pandas is a library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. It will be specifically useful for people working with data cleansing and analysis. Pandas library uses most of the functionalities of NumPy. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

**Key Features of Pandas**:  
  
1.Fast and efficient Data Frame object with default and customized indexing.  
2.Tools for loading data into in-memory data objects from different file formats.  
 3.Data alignment and integrated handling of missing data.  
 4.Reshaping and pivoting of date sets.  
 5.Label-based slicing, indexing and sub setting of large data sets.  
 6.Columns from a data structure can be deleted or inserted.  
7.Group by data for aggregation and transformations.  
8.High performance merging and joining of data.  
9.Time Series functionality.  
  
Pandas deals with the following three data structures –

* Series
* Dataframe
* panel

These data structures are built on top of NumPy array, which means they are fast.

**Data Frames in python**:  
Pandas Data Frames make manipulating your data easy, from selecting or replacing columns and indices to reshaping your data.  
  
  
  
Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The Data Frame is one of these structures.  
  
**Pandas Data Frames:**Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the NumPy package and its key data structure is called the Data Frame. Data Frames allow you to store and manipulate tabular data in rows of observations and columns of variables.  
  
There are several ways to create a Data Frame.  
Pandas can be imported into Python using:

>>> import pandas as pd

**DATA STRUCTURES:**

Some commonly used data structures in pandas are:

Series objects: 1D array, similar to a column in a spreadsheet

DataFrame objects: 2D table, similar to a spreadsheet

Panel objects: Dictionary of DataFrames, similar to sheet in MS Excel

Pandas Series object is created using pd.Series function. Each row is provided with an index and by defaults is assigned numerical values starting from 0. Like NumPy, Pandas also provide the basic mathematical functionalities like addition, subtraction and conditional operations and broadcasting.

Pandas dataframe object represents a spreadsheet with cell values, column names, and row index labels. Dataframe can be visualized as dictionaries of Series. Dataframe rows and columns are simple and intuitive to access. Pandas also provide SQL-like functionality to filter, sort rows based on conditions

New columns and rows can be easily added to the dataframe. In addition to the basic functionalities, pandas dataframe can be sorted by a particular column.

Dataframes can also be easily exported and imported from CSV, Excel, JSON, HTML and SQL database. Some other essential methods that are present in data frames are:

**head():** returns the top 5 rows in the dataframe object

**tail():** returns the bottom 5 rows in the dataframe

**info():** prints the summary of the dataframe

**describe():** gives a nice overview of the main aggregated values over each column

**SciPy:**

**SciPy**  is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source) [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) library used for [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing) and technical computing.

SciPy contains modules for [optimization](https://en.wikipedia.org/wiki/Optimization_(mathematics)), [linear algebra](https://en.wikipedia.org/wiki/Linear_algebra), [integration](https://en.wikipedia.org/wiki/Integral), [interpolation](https://en.wikipedia.org/wiki/Interpolation), [special functions](https://en.wikipedia.org/wiki/Special_functions), [FFT](https://en.wikipedia.org/wiki/Fast_Fourier_transform), [signal](https://en.wikipedia.org/wiki/Signal_processing) and [image processing](https://en.wikipedia.org/wiki/Image_processing), [ODE](https://en.wikipedia.org/wiki/Ordinary_differential_equation) solvers and other tasks common in science and engineering.

SciPy builds on the [NumPy](https://en.wikipedia.org/wiki/NumPy) array object and is part of the NumPy stack which includes tools like [Matplotlib](https://en.wikipedia.org/wiki/Matplotlib), [pandas](https://en.wikipedia.org/wiki/Pandas_(software)) and [SymPy](https://en.wikipedia.org/wiki/SymPy), and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as [MATLAB](https://en.wikipedia.org/wiki/MATLAB), [GNU Octave](https://en.wikipedia.org/wiki/GNU_Octave), and [Scilab](https://en.wikipedia.org/wiki/Scilab).The NumPy stack is also sometimes referred to as the SciPy stack.

The SciPy package of key algorithms and functions core to Python's scientific computing capabilities. Available sub-packages include:

* **constants**: physical constants and conversion factors (since version 0.7.0[[5]](https://en.wikipedia.org/wiki/SciPy#cite_note-5))
* **cluster**: hierarchical clustering, vector quantization, K-means
* **fftpack**: Discrete Fourier Transform algorithms
* **integrate**: numerical integration routines
* **interpolate**: interpolation tools
* **io**: data input and output
* **lib**: Python wrappers to external libraries
* **linalg**: linear algebra routines
* **misc**: miscellaneous utilities (e.g. image reading/writing)
* **ndimage**: various functions for multi-dimensional image processing
* **optimize**: optimization algorithms including linear programming
* **signal**: signal processing tools
* **sparse**: sparse matrix and related algorithms
* **spatial**: KD-trees, nearest neighbors, distance functions
* **special**: special functions
* **stats**: statistical functions
* **weave**: tool for writing C/C++ code as Python multiline string

***Machine Learning Algorithms Used***

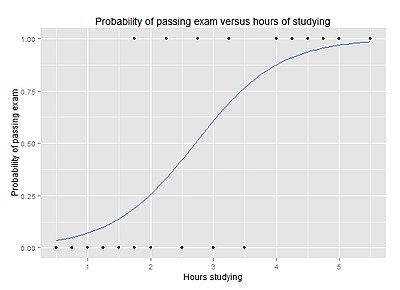
**Logistic Regression:**

Logistic regression was developed by statistician David Cox in 1958.

In statistics, the **logistic model** (or **logit model**) is a statistical model that is usually taken to apply to a binary dependent variable. In regression analysis, **logistic regression** or **logit regression** is estimating the parameters of a logistic model.

More formally, a logistic model is one where the log-odds of the probability of an event is a linear combination of independent or predictor variables.

Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences. For example, the Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patients, was originally developed by Boyd et al. using logistic regression.[[3]](https://en.wikipedia.org/wiki/Logistic_regression#cite_note-3) Many other medical scales used to assess severity of a patient have been developed using logistic regression. Logistic regression may be used to predict the risk of developing a given disease (e.g. diabetes; coronary heart disease), based on observed characteristics of the patient (age, sex, body mass index, results of various blood tests, etc.). Another example might be to predict whether an Indian voter will vote BJP or Trinamool Congress or Left Front or Congress, based on age, income, sex, race, state of residence, votes in previous elections, etc.[[10]](https://en.wikipedia.org/wiki/Logistic_regression#cite_note-Harrell-10) The technique can also be used in engineering, especially for predicting the probability of failure of a given process, system or product. It is also used in marketing applications such as prediction of a customer's propensity to purchase a product or halt a subscription, etc.[]](https://en.wikipedia.org/wiki/Logistic_regression#cite_note-13) In economics it can be used to predict the likelihood of a person's choosing to be in the labor force, and a business application would be to predict the likelihood of a homeowner defaulting on a mortgage. Conditional random fields, an extension of logistic regression to sequential data, are used in natural language processing.



**Decision Tree Algorithm:**

**Decision Tree algorithm belongs to the family of**[**supervised learning algorithms**](https://dataaspirant.com/2014/09/19/supervised-and-unsupervised-learning/)**. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving**[**regression and classification**](https://dataaspirant.com/2014/09/27/classification-and-prediction/)**problems too.**

**The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data(training data).**

**The understanding level of Decision Trees algorithm is so easy compared with other classification algorithms. The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.**

**The below are the some of the assumptions we make while using Decision tree:**

* **At the beginning, the whole training set is considered as the root.**
* **Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.**
* **Records are distributed recursively on the basis of attribute values.**
* **Order to placing attributes as root or internal node of the tree is done by using some statistical approach.**

**K Nearest Neighbors:**

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

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

*k*-NN is a type of instance-based learning , or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all machine learning  algorithms.

Both for classification and regression, a useful technique can be to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/*d*, where *d* is the distance to the neighbor.[[2]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-2)

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

A peculiarity of the *k*-NN algorithm is that it is sensitive to the local structure of the data. The algorithm is not to be confused with [*k*-means](https://en.wikipedia.org/wiki/K-means), another popular machine learning  technique.

# **Naive Bayes for Machine Learning:**

Naive Bayes is a simple but surprisingly powerful algorithm for predictive modelling.

**N**aïve Bayes classifier is one of the most effective machine learning algorithms implemented in machine learning projects and distributed MapReduce implementations leveraging Apache Spark.

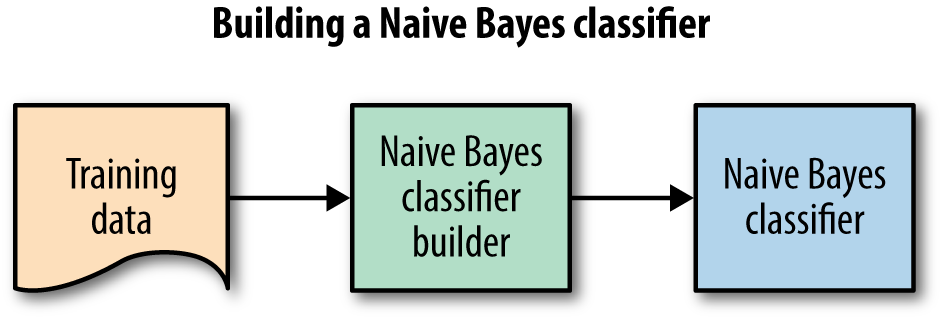
Primarily Naïve Bayes is a linear classifier, which is a supervised machine learning method and works as a probabilistic classifier as well. Most of the time, for the numeric implementations K-Nearest Neighbors and K-Means clustering algorithms can be implemented. Naïve Bayes classifier works effectively for classifying e+

mails, texts, symbols, and names.

It’s not unusual Naïve Bayes classifier is used for numeric data as well in some instances. Naïve Bayes classifier can be implemented on high-dimensional datasets effectively as well. Naïve Bayes classifier predicts the probability of each class based on the feature vector for text classification for continuous big data with a prior distribution of the probability, tackling the challenges of the curse of the dimensionality.

There are three types of Naïve Bayes classifiers. When handling real-time data with continuous distribution, Naïve Bayes classifier considers that the big data is generated through a Gaussian process with normal distribution. Multinomial Naïve Bayes classifier can be applied when handling event models where the events are modelled through a multinomial distribution. In this situation, the features are frequencies.

In the third scenario, when the features are Boolean or independent, the features are generated through a Bernoulli an process. In this scenario, a Bernoulli Naïve Bayes classifier can be applied.



**Support Vector Machine:**

In machine learning, **support vector machines** (**SVMs**, also **support vector networks**[[1]](https://en.wikipedia.org/wiki/Support_vector_machine" \l "cite_note-CorinnaCortes-1)) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are not labeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups.

**Implementation**

The data set required for the classification of the iris flower species was collected from the **UCI** repository (<https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>).

The iris dataset contains four features namely sepal length, sepal width, petal length, petal width. From these four features, the species(targets) are identified as setosa, versicolor or virginica.

Pandas library was used to read the datasets from the repository by using read\_csv (“file path”) of pandas library.

The dataset was then splitted into two parts:

* Training
* Testing

The dataset was splitted in such a way to prevent overfitting of the model (70% for training and 30% for testing). This was done by importing train\_test\_split module from sklearn.cross\_validation module.

Then the various models for classification were used to train and test the model by importing from sklearn library. The models used were:

* LogisticRegression (imported from sklearn. linear\_model module)
* Decision Tree Classifier (imported from sklearn.tree module)
* K Nearest Neighbor(imported from sklearn.neighbors module)
* Gaussian Naive Bayes (imported from sklearn.naive\_bayes module)
* Support Vector Classification (SVC) (imported from sklearn.svm module)

First the model was trained using fit method. And then it was tested using predict method.

Then the accuracy was found out by accuracy score method which was imported from sklearn.metrics module.

Finally, all the accuracies of the machine learning were stored in a list and a graph showing the machine learning model vs accuracy was plotted using matplotlib. pyplot module.

**Logistic Regression**

Import numpy as np

import pandas as pd

from sklearn import datasets

*#reading the data from uci repository*

data=pd.read\_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data")

X=data.iloc[:, :-1].values

y=data.iloc[:, 4].values

from sklearn.cross\_validation import train\_test\_split

*# Splitting the data for training (70%) and testing (30%)*

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3)

print ("The training data is\n",X\_train)

print ("The testing data is\n",X\_test)

print ("The expected result is\n",y\_test)

*# By using LogisticRegression*

from sklearn. linear\_model import LogisticRegression

clf=LogisticRegression ()

*# Training the dataset using Logistic Regression Model*

clf.fit(X\_train, y\_train)

*# Prediction*

prediction=clf. predict(X\_test)

print ("The prediction by the machine is\n”, prediction)

from sklearn. metrics import accuracy\_score

*# Finding the accuracy of the model*

a=accuracy\_score(y\_test, prediction)

a1=a\*100

print("The accuracy of the model is:",a1)

**Decision Tree Classifier**

Import numpy as np

import pandas as pd

from sklearn import datasets

import matplotlib. pyplot as plt

*#reading the data from uci repository*

data=pd.read\_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data")

X=data.iloc[:, :-1].values

y=data. iloc[:, 4].values

from sklearn. cross-validation import train\_test\_split

*# Splitting the data for training (70%) and testing(30%)*

X\_train, X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3)

print ("The training data is\n",X\_train)

print ("The testing data is\n",X\_test)

print ("the expected result is\n",y\_test)

*# By using Decision Tree Clasifier*

from sklearn. tree import DecisionTreeClassifier

clf=DecisionTreeClassifier ()

*# Training the dataset using Decision Tree Classifier Model*

clf=clf.fit(X\_train, y\_train)

*# Prediction*

prediction=clf.predict(X\_test)

print("The prediction by the machine learning model is\n",prediction)

from sklearn.metrics import accuracy\_score

*# Finding the accuracy of the model*

a=accuracy\_score(y\_test,prediction)

a2=a\*100

print("The accuracy of this model is: ", a2)

**Gaussian Naive Bayes**

import numpy as np

import pandas as pd

from sklearn import datasets

import matplotlib.pyplot as plt

*#reading the data from uci repository*

data=pd.read\_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data")

X=data.iloc[:, :-1].values

y=data.iloc[:, 4].values

from sklearn.cross\_validation import train\_test\_split

p=[]

*# Splitting the data for training(70%) and testing(30%)*

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3)

print("The training data is\n",X\_train)

print("The testing data is\n",X\_test)

print("the expected result is\n",y\_test)

*# By using Naive Bayes*

from sklearn.naive\_bayes import GaussianNB

clf=GaussianNB()

*# Training the dataset using Naive Bayes Model*

clf=clf.fit(X\_train,y\_train)

*# Prediction*

prediction=clf.predict(X\_test)

print("The prediction by the machine learning model is\n",prediction)

from sklearn.metrics import accuracy\_score

*# Finding the accuracy of the model*

a=accuracy\_score(y\_test,prediction)

a3=a\*100

print("The accuracy of this model is:- ", a3)

**K Nearest Neighbors**

import pandas as pd

from sklearn import datasets

import matplotlib.pyplot as plt

*#reading the data from uci repository*

data=pd.read\_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data")

X=data.iloc[:, :-1].values

y=data.iloc[:, 4].values

from sklearn.cross\_validation import train\_test\_split

*# Splitting the data for training (70%) and testing(30%)*

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3)

print("The training data is\n",X\_train)

print("The testing data is\n",X\_test)

print("the expected result is\n",y\_test)

*# By using K Nearest Neighbor*

from sklearn.neighbors import KNeighborsClassifier

clf=KNeighborsClassifier()

*# Training the dataset using K Nearest Neighbor Model*

clf=clf.fit(X\_train,y\_train)

*# Prediction*

prediction=clf.predict(X\_test)

print("The prediction by the machine is\n",prediction)

from sklearn.metrics import accuracy\_score

*# Finding the accuracy of the model*

a=accuracy\_score(y\_test,prediction)

a4=a\*100

print("The accuracy of this model is:- ", a4)

**Support Vector Classification**

import numpy as np

import pandas as pd

from sklearn import datasets

import matplotlib.pyplot as plt

*#reading the data from uci repository*

data=pd.read\_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data")

X=data.iloc[:, :-1].values

y=data.iloc[:, 4].values

from sklearn.cross\_validation import train\_test\_split

*# Splitting the data for training(70%) and testing(30%)*

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3)

print("The training data is\n",X\_train)

print("The testing data is\n",X\_test)

print("the expected result is\n",y\_test)

*# By using Support Vector Machine*

from sklearn.svm import SVC

clf=SVC()

*# Training the dataset using Support Vector Machines Model*

clf=clf.fit(X\_train,y\_train)

*# Prediction*

prediction=clf.predict(X\_test)

print("The prediction by the machine learning model is\n",prediction)

from sklearn.metrics import accuracy\_score

*# Finding the accuracy of the model*

a=accuracy\_score(y\_test,prediction)

a5=a\*100

print("The accuracy of this model is: ", a5)

*# Plotting the line graph*

l=[a1,a2,a3,a4,a5]

m=['Linear Regression','DecisionTree','NaiveBayes','KNN','SVM']

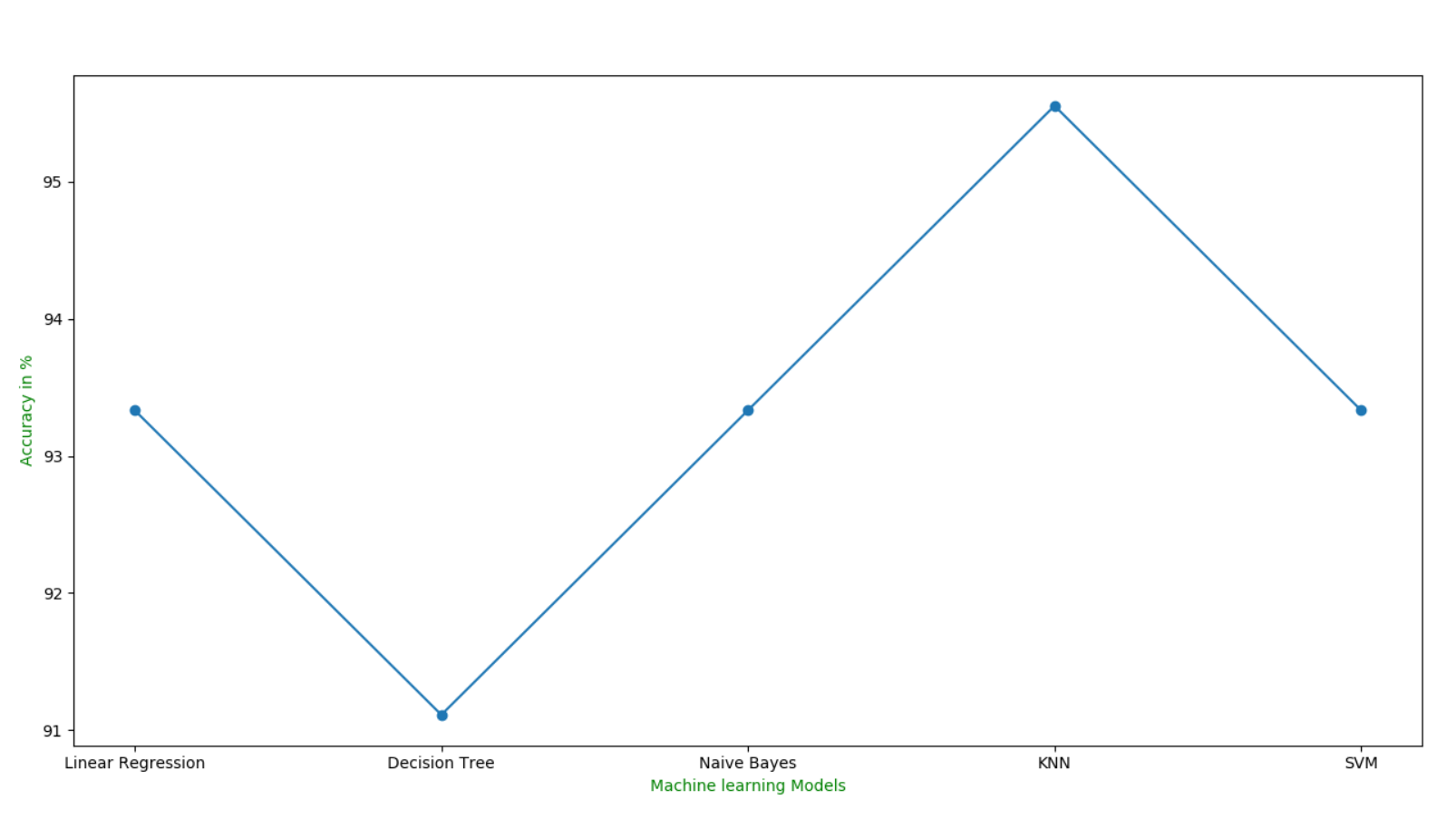
import matplotlib.pyplot as plt

plt.plot(m,l,marker='o')

plt. xlabel("Machine learning Models",color="green")

plt. ylabel ("Accuracy in %”, color="green")

plt. show ()



**Graph Showing Machine learning models Vs Accuracy in %**

**Table:**

This table shows the accuracy of the model by using different types of algorithms:

|  |  |
| --- | --- |
| **Algorithms used** | **Accuracy in %** |
| 1.Linear Regression (LR) | 93.33% |
| 2.Decision Tree | 91.11% |
| 3.Gaussian Naive Bayes (NB) | 93.33% |
| 4.K-Nearest Neighbors (KNN) | 95.55% |
| 5.Support Vector Machines (SVM) | 93.33% |

**Conclusion**

The purpose of this project was to explore data visualization techniques utilizing the Iris dataset. This kernel follows the “Python Data Visualization”. The data set contains 150 observations of iris flowers.

We installed 5 Key libraries like SciPy, numpy,matplot,pandas,sklearn.We load the data directly from the UCI Machine Learning repository. It will load all the data regarding iris.

We need to know that the model we created is any good. So, we did some visualization. We used statistical methods to estimate the accuracy of the models.

We got an idea from the plots that some of the classes are partially linearly separable and some dimension. So, we used different algorithms to check the accuracy of the models. We used models like Linear Regression(LR),Decision tree, Gaussian Naive Bayes,KNN and SVM for checking accuracy.

Comparing these models, we got an idea of the best algorithm which has got the best accuracy. The K-Nearest Neighbors is the best model with an accuracy of 95.55%. So, this was an overall idea of using different types of libraries, algorithms and testing and training for the iris dataset to know the best accuracy for the proposed model.

**References:**

* <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>
* <https://machinelearningmastery.com/machine-learning-in-python-step-by-step/>
* <https://en.wikipedia.org/wiki/NumPy>
* <https://en.wikipedia.org/wiki/SciPy>
* <https://en.wikipedia.org/wiki/Matplotlib>
* <https://en.wikipedia.org/wiki/Scikit-learn>
* <https://www.coursera.org/learn/machine-learning>