**Pandas**

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with 'relationa' or 'labeled' data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

**Pandas is well suited for many different kinds of data:**

* Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
* Ordered and unordered (not necessarily fixed-frequency) time series data.
* Arbitrary matrix data with row and column labels
* Any other form of observational / statistical data sets.

**Pandas Basic commands:**

import pandas

**Numpy**

“A NumPy tutorial for beginners in which you'll learn how to create a NumPy array, use broadcasting, access values, manipulate arrays, and much more.”

* NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can’t miss when you’re learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in

reading and writing items, being more convenient and more efficient.

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

* 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. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Getting Started**

To install NumPy, we strongly recommend using a scientific Python distribution. See [Installing the SciPy Stack](https://www.scipy.org/install.html) for details.

Many high quality online tutorials, courses, and books are available to get started with NumPy. For a quick introduction to NumPy we provide the [NumPy Tutorial](https://www.numpy.org/devdocs/user/quickstart.html). We also recommend the [SciPy Lecture Notes](https://scipy-lectures.org/) for a broader introduction to the scientific Python ecosystem.

## Documentation[¶](https://www.numpy.org/#documentation)

The most up-to-date NumPy documentation can be found at [Latest (development) version](https://www.numpy.org/devdocs). It includes a user guide, full reference **documentation, a developer guide, meta information, and “NumPy** Enhancement Proposals” (which include the NumPy Roadmap and detailed plans for major new features).

## Support NumPy

Institutional Partners are organizations that support the project by employing NumPy contributors, with contributing to the project as part of their official duties. Current Institutional Partners include:







# **Matplotlib**

Matplotlib is a python library used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc. It supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc. It is used along with NumPy to provide an environment that is an effective open source alternative for MatLab. It can also be used with graphics toolkits like PyQt and wxPython.

## Matplotlib Example:-

import numpy as np

import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on a sine curve

x = np.arange(0, 3 \* np.pi, 0.1)

y = np.sin(x)

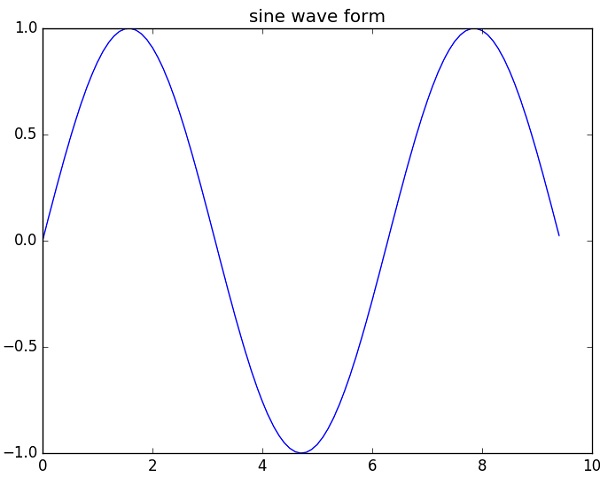
plt.title("sine wave form")

# Plot the points using matplotlib

plt.plot(x, y)

plt.show()

## Its output is as follows –



# **Scikit-learn**

Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

To do this, we’ll be using the [Sales\_Win\_Loss data](https://www.ibm.com/communities/analytics/watson-analytics-blog/sales-win-loss-sample-dataset/) set from IBM’s Watson repository. We will import the data set using pandas, explore the data using pandas methods like head(), tail(), dtypes(), and then try our hand at using plotting techniques from Seaborn to visualize our data.

**Sklearn\_Feature Selection**

Feature selection is a process where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested.

Having irrelevant features in your data can decrease the accuracy of many models, especially linear algorithms like linear and logistic regression.

Three benefits of performing feature selection before modeling your data are:

* **Reduces Overfitting**: Less redundant data means less opportunity to make decisions based on noise.
* **Improves Accuracy**: Less misleading data means modeling accuracy improves.
* **Reduces Training Time**: Less data means that algorithms train faster.

[**sklearn.model\_selection**](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection)**.train\_test\_split**

Split arrays or matrices into random train and test subsets.Quick utility that wraps input validation and next(ShuffleSplit().split(X, y)) and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

sklearn.model\_selection.**train\_test\_split**

# **Random Forests Classifiers**

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forests has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

**Sklearn metrics**

The module **[sklearn.metrics](https://scikit-learn.org/stable/modules/classes.html" \l "module-sklearn.metrics" \o "sklearn.metrics)** also exposes a set of simple functions measuring a prediction error given ground truth and prediction:

* functions ending with \_score return a value to maximize, the higher the better.
* functions ending with \_error or \_loss return a value to minimize, the lower the better. When converting into a scorer object using **[make\_scorer](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.make_scorer.html" \l "sklearn.metrics.make_scorer" \o "sklearn.metrics.make_scorer)**, set the greater\_is\_better parameter to False.

Many metrics are not given names to be used as scoring values, sometimes because they require additional parameters, such as **[fbeta\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.fbeta_score.html" \l "sklearn.metrics.fbeta_score" \o "sklearn.metrics.fbeta_score)**. In such cases, you need to generate an appropriate scoring object. The simplest way to generate a callable object for scoring is by using **[make\_scorer](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.make_scorer.html" \l "sklearn.metrics.make_scorer" \o "sklearn.metrics.make_scorer)**. That function converts metrics into callables that can be used for model evaluation.

**Sklearn.metrics accuracy\_score**

The **[accuracy\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html" \l "sklearn.metrics.accuracy_score" \o "sklearn.metrics.accuracy_score)** function computes the [accuracy](https://en.wikipedia.org/wiki/Accuracy_and_precision), either the fraction (default) or the count (normalize=False) of correct predictions.

In multilabel classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0.

If y^ is the predicted value of the i-th sample and yi is the corresponding true value, then the fraction of correct predictions over nsamples is defined as

**SVC**

The most applicable machine learning algorithm for our problem is [Linear SVC](http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html). Before hopping into Linear SVC with our data, we're going to show a very simple example that should help solidify your understanding of working with Linear SVC.

The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is. This makes this specific algorithm rather suitable for our uses, though you can use this for many situations.

**SVM**

**Support vector machines (SVM)** are a set of supervised learning methods used for [classification](https://scikit-learn.org/stable/modules/svm.html#svm-classification), [regression](https://scikit-learn.org/stable/modules/svm.html#svm-regression) and [outliers detection](https://scikit-learn.org/stable/modules/svm.html#svm-outlier-detection).

The advantages of support vector machines are:

* Effective in high dimensional spaces.
* Still effective in cases where number of dimensions is greater than the number of samples.
* Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
* Versatile: different [Kernel functions](https://scikit-learn.org/stable/modules/svm.html#svm-kernels) can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

* If the number of features is much greater than the number of samples, avoid over-fitting in choosing [Kernel functions](https://scikit-learn.org/stable/modules/svm.html#svm-kernels) and regularization term is crucial.
* SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see [Scores and probabilities](https://scikit-learn.org/stable/modules/svm.html#scores-probabilities), below).