Steps:

# Phase 0 — Data Preparation

We will use the UCI Machine Learning Repository for breast cancer [dataset](http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29).

[*http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29*](http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29)

The dataset used in this story is publicly available and was created by Dr. William H. Wolberg, physician at the University Of Wisconsin Hospital at Madison, Wisconsin, USA. To create the dataset Dr. Wolberg used fluid samples, taken from patients with solid breast masses and an easy-to-use graphical computer program called Xcyt, which is capable of perform the analysis of cytological features based on a digital scan. The program uses a curve-fitting algorithm, to compute ten features from each one of the cells in the sample, than it calculates the mean value, extreme value and standard error of each feature for the image, returning a 30 real-valuated vector

Attribute Information:

1. ID number 2) Diagnosis (M = malignant, B = benign) 3–32)

Ten real-valued features are computed for each cell nucleus:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness (perimeter² / area — 1.0)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension (“coastline approximation” — 1)

The mean, standard error and “worst” or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

## Objectives

This analysis aims to observe which features are most helpful in predicting malignant or benign cancer and to see general trends that may aid us in model selection and hyper parameter selection. The goal is to classify whether the breast cancer is benign or malignant. To achieve this i have used machine learning classification methods to fit a function that can predict the discrete class of new input.

# Phase 1 — Data Exploration

We will be using **Spyder** to work on this dataset. We will first go with importing the necessary libraries and import our dataset to Spyder :

#importing the libraries  
import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd#importing our cancer dataset  
dataset = pd.read\_csv(‘cancer.csv')  
X = dataset.iloc[:, 1:31].values  
Y = dataset.iloc[:, 31].values



Fig : Dataset and X set after importing the dataset

We can examine the data set using the pandas’ **head()** method.

dataset.head()

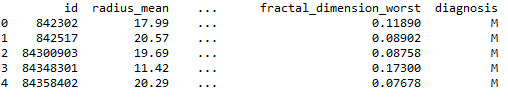


Fig : top 5 data of our dataset

We can find the dimensions of the data set using the panda dataset ‘shape’ attribute.

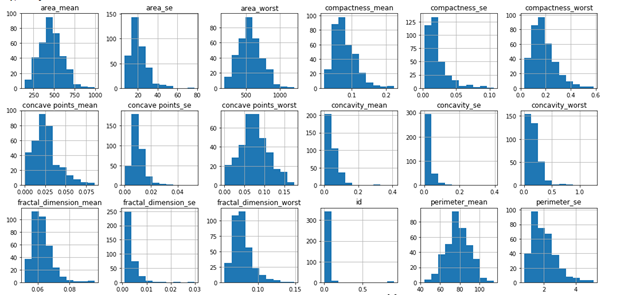
print("Cancer data set dimensions : {}".format(dataset.shape))Cancer data set dimensions : (569, 32)

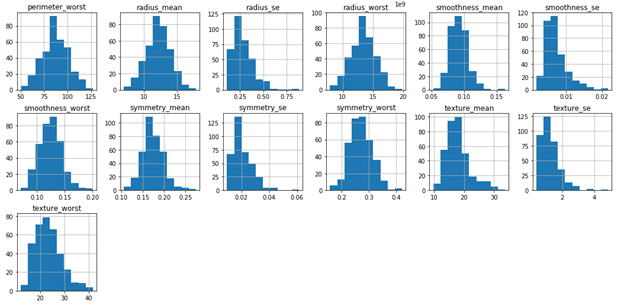
We can observe that the data set contain 569 rows and 32 columns. ‘Diagnosis’ is the column which we are going to predict , which says if the cancer is M = malignant or B = benign. 1 means the cancer is malignant and 0 means benign. We can identify that out of the 569 persons, 357 are labeled as B (benign) and 212 as M (malignant).

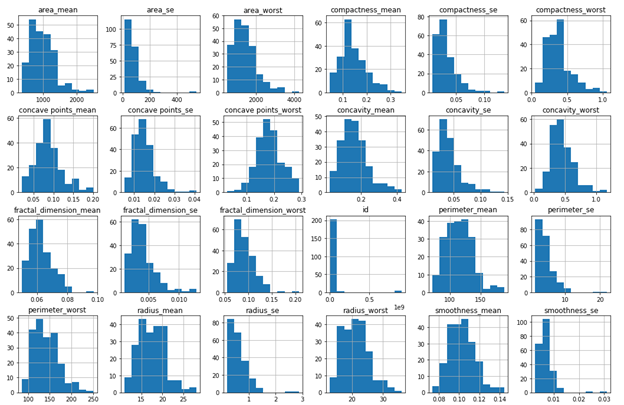


Visualization of data is an imperative aspect of data science. It helps to understand data and also to explain the data to another person. Python has several interesting visualization libraries such as Matplotlib, Seaborn etc.

In this tutorial we will use pandas’ visualization which is built on top of matplotlib, to find the data distribution of the features.







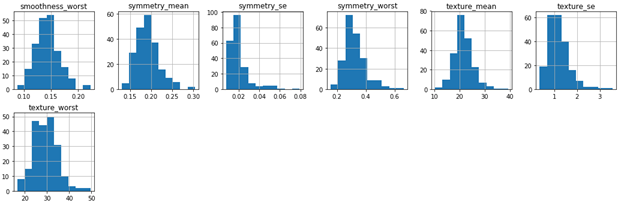


Fig : Visualization of Dataset

**Missing or Null Data points**

We can find any missing or null data points of the data set (if there is any) using the following pandas function.

dataset.isnull().sum()  
dataset.isna().sum()

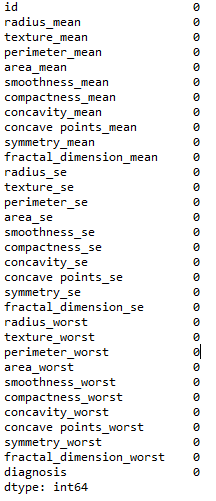


Fig : Observe missing data

# Phase 2 — Categorical Data

Categorical data are variables that contain label values rather than numeric values.The number of possible values is often limited to a fixed set.

For example, users are typically described by country, gender, age group etc.

We will use Label Encoder to label the categorical data. Label Encoder is the part of SciKit Learn library in Python and used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

#Encoding categorical data values  
from sklearn.preprocessing import LabelEncoder  
labelencoder\_Y = LabelEncoder()  
Y = labelencoder\_Y.fit\_transform(Y)

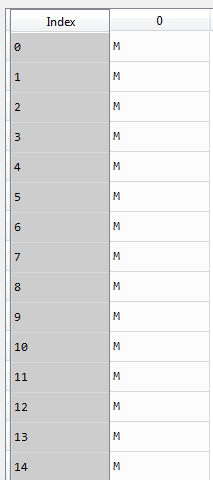


Fig: Diagnosis Data without Encoding

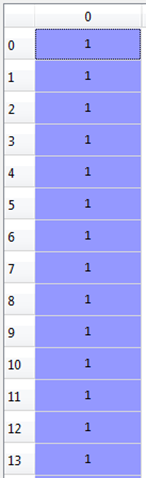


Fig: Diagnosis Data after Encoding

## **Splitting the dataset**

The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset (or subset) in order to test our model’s prediction on this subset.

We will do this using SciKit-Learn library in Python using the train\_test\_split method.

# Splitting the dataset into the Training set and Test setfrom sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.25, random\_state = 0)

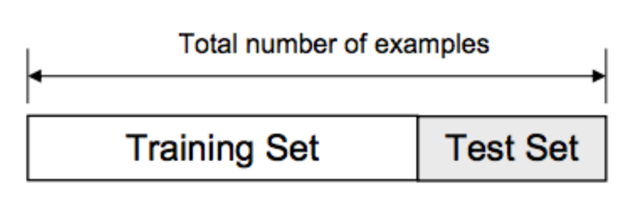


Fig: Training and test set

# Phase 3 — Feature Scaling

Most of the times, your dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Eucledian distance between two data points in their computations. We need to bring all features to the same level of magnitudes. This can be achieved by scaling. This means that you’re transforming your data so that it fits within a specific scale, like 0–100 or 0–1.

We will use StandardScaler method from SciKit-Learn library.

#Feature Scalingfrom sklearn.preprocessing import StandardScaler  
sc = StandardScaler()  
X\_train = sc.fit\_transform(X\_train)  
X\_test = sc.transform(X\_test)

# Phase 4 — Model Selection

This is the most exciting phase in Applying Machine Learning to any Dataset. It is also known as Algorithm selection for Predicting the best results.

Usually Data Scientists use different kinds of Machine Learning algorithms to the large data sets. But, at high level all those different algorithms can be classified in two groups : supervised learning and unsupervised learning.

Without wasting much time, I would just give a brief overview about these two types of learnings.

Supervised learning : Supervised learning is a type of system in which both input and desired output data are provided. Input and output data are labelled for classification to provide a learning basis for future data processing. Supervised learning problems can be further grouped into **Regression** and **Classification** problems.

A **regression** problem is when the output variable is a real or continuous value, such as “salary” or “weight”.

A **classification** problem is when the output variable is a category like filtering emails “spam” or “not spam”

Unsupervised Learning : Unsupervised learning is the [algorithm](https://whatis.techtarget.com/definition/algorithm) using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance.

In our dataset we have the outcome variable or Dependent variable i.e Y having only two set of values, either M (Malign) or B(Benign). So we will use Classification algorithm of supervised learning.

We have different types of classification algorithms in Machine Learning :-

1. Logistic Regression

2. Nearest Neighbor

3. Support Vector Machines

4. Kernel SVM

5. Naïve Bayes

6. Decision Tree Algorithm

7. Random Forest Classification

Lets start applying the algorithms :

We will use sklearn library to import all the methods of classification algorithms.

We will use LogisticRegression method of model selection to use Logistic Regression Algorithm,

#Using Logistic Regression Algorithm to the Training Setfrom sklearn.linear\_model import LogisticRegression  
classifier = LogisticRegression(random\_state = 0)  
classifier.fit(X\_train, Y\_train)#Using KNeighborsClassifier Method of neighbors class to use Nearest Neighbor algorithmfrom sklearn.neighbors import KNeighborsClassifierclassifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)classifier.fit(X\_train, Y\_train)  
#Using SVC method of svm class to use Support Vector Machine Algorithm  
from sklearn.svm import SVCclassifier = SVC(kernel = 'linear', random\_state = 0)classifier.fit(X\_train, Y\_train)  
#Using SVC method of svm class to use Kernel SVM Algorithm  
from sklearn.svm import SVCclassifier = SVC(kernel = 'rbf', random\_state = 0)classifier.fit(X\_train, Y\_train)  
#Using GaussianNB method of naïve\_bayes class to use Naïve Bayes Algorithm  
from sklearn.naive\_bayes import GaussianNBclassifier = GaussianNB()classifier.fit(X\_train, Y\_train)  
#Using DecisionTreeClassifier of tree class to use Decision Tree Algorithm  
from sklearn.tree import DecisionTreeClassifierclassifier = DecisionTreeClassifier(criterion = 'entropy', random\_state = 0)classifier.fit(X\_train, Y\_train)  
#Using RandomForestClassifier method of ensemble class to use Random Forest Classification algorithm  
from sklearn.ensemble import RandomForestClassifierclassifier = RandomForestClassifier(n\_estimators = 10, criterion = 'entropy', random\_state = 0)classifier.fit(X\_train, Y\_train)

We will now predict the test set results and check the accuracy with each of our model:

Y\_pred = classifier.predict(X\_test)

To check the accuracy we need to import confusion\_matrix method of metrics class. The confusion matrix is a way of tabulating the number of mis-classifications, i.e., the number of predicted classes which ended up in a wrong classification bin based on the true classes.

from sklearn.metrics import confusion\_matrix  
cm = confusion\_matrix(Y\_test, Y\_pred)

We will use Classification Accuracy method to find the accuracy of our models. Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.



Fig: Accuracy

To check the correct prediction we have to check confusion matrix object and add the predicted results diagonally which will be number of correct prediction and then divide by total number of predictions.

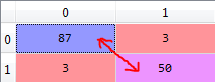


Fig: Confusion Matrix

After applying the different classification models, we have got below accuracies with different models:

1. Logistic Regression — 95.8%

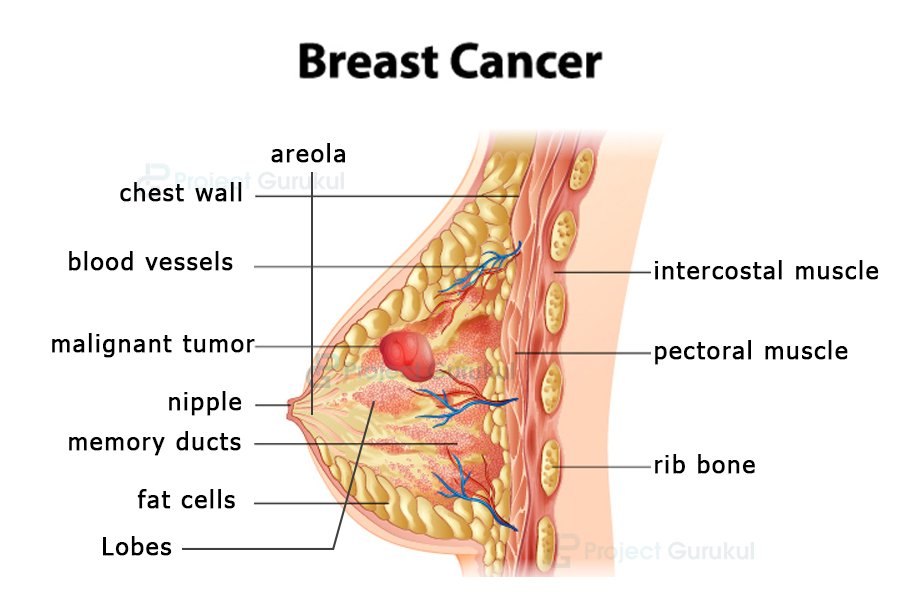
2. Nearest Neighbor — 95.1%

3. Support Vector Machines — 97.2%

4. Kernel SVM — 96.5%

5. Naive Bayes — 91.6%

6. Decision Tree Algorithm — 95.8%



7. Random Forest Classification — 98.6%

So finally we have built our classification model and we can see that Random Forest Classification algorithm gives the best results for our dataset. Well its not always applicable to every dataset. To choose our model we always need to analyze our dataset and then apply our machine learning model.