Name – Vikalp Bharti

CLASS - T.Y CSE IS 2

Roll no – 2193279

Enrolment No - MITU19BTCS0057

ASSIGNMENT 1

Problem Statement-

Installation and Configuration of machine learning environment with Anaconda on windows or Ubuntu (Jupyter notebook)

Objective

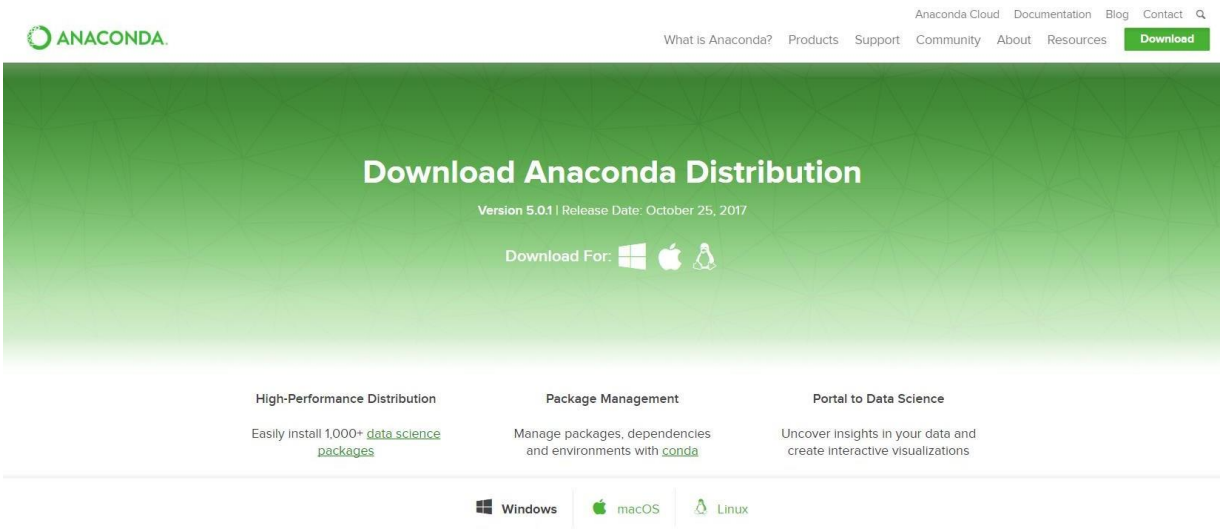
To install anaconda and configure it with Python

Theory

Installation Of Anaconda in Windows

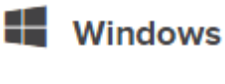
1. Visit the Anaconda downloads page

Go to the following link: Anaconda.com/downloads



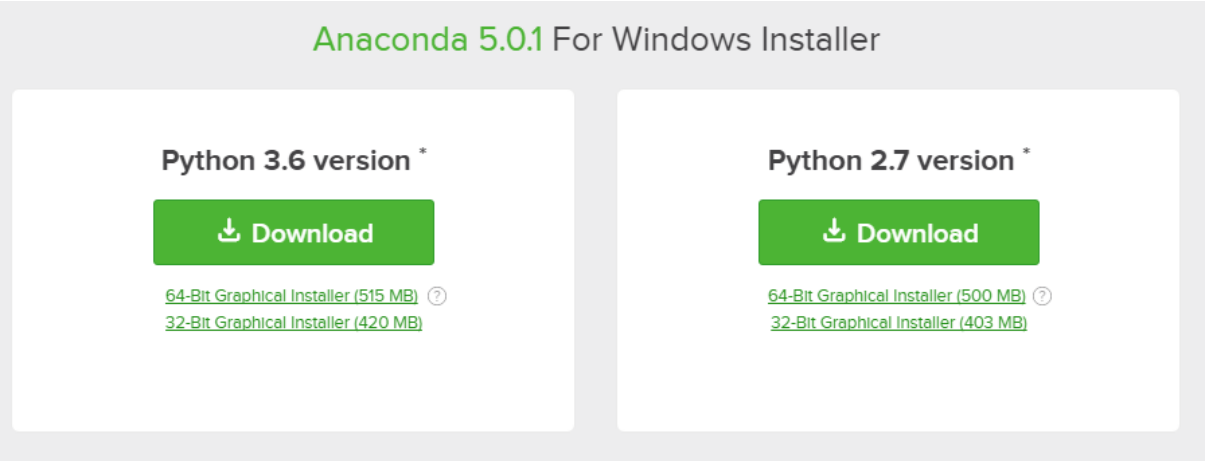
2. Select Windows

Select Windows where the three operating systems are listed.


Windows
macOS
Linux

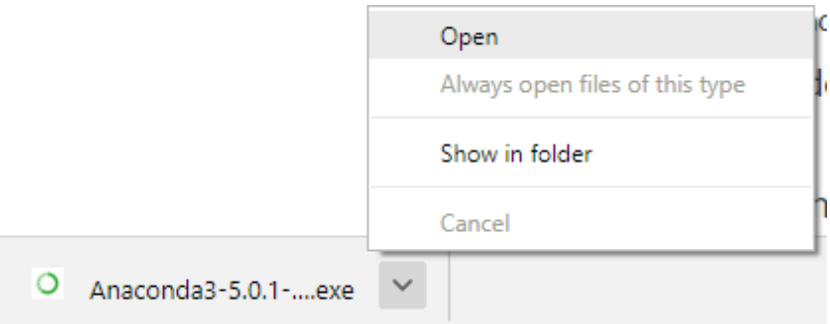
3. Download

Download the most recent Python 3 release. At the time of writing, the most recent release was the Python 3.6 Version. Python 2.7 is legacy Python. For problem solvers, select the Python 3.6 version. If you are unsure if your computer is running a 64-bit or 32-bit version of Windows, select 64-bit as 64- bit Windows is most common.

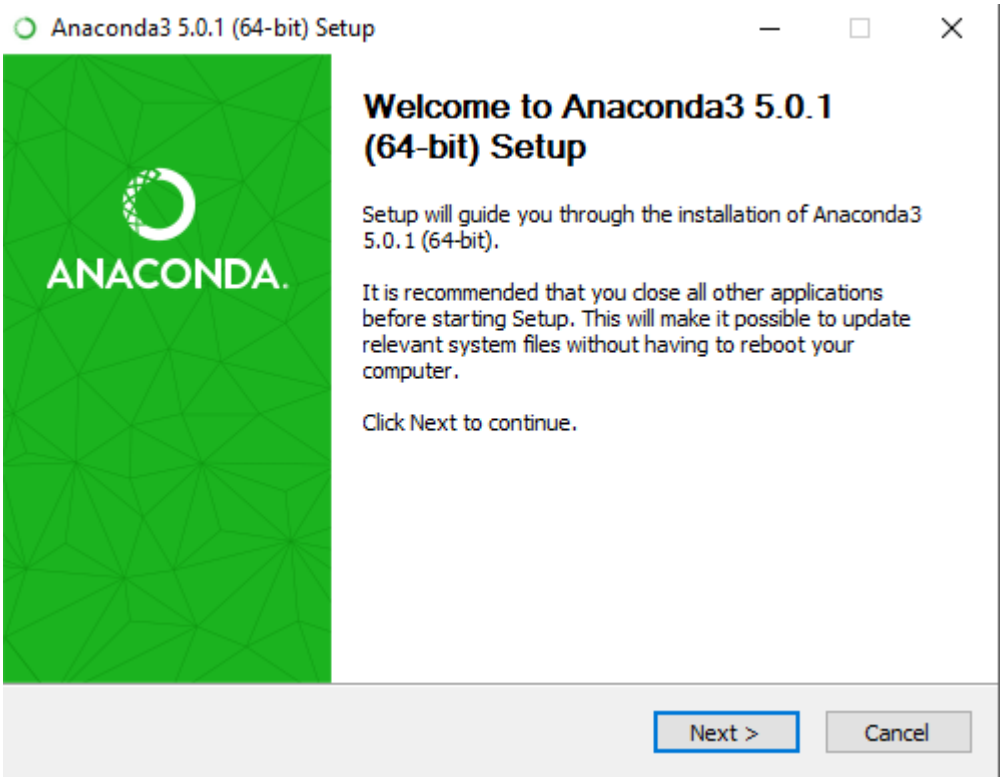


4. Open and run the installer

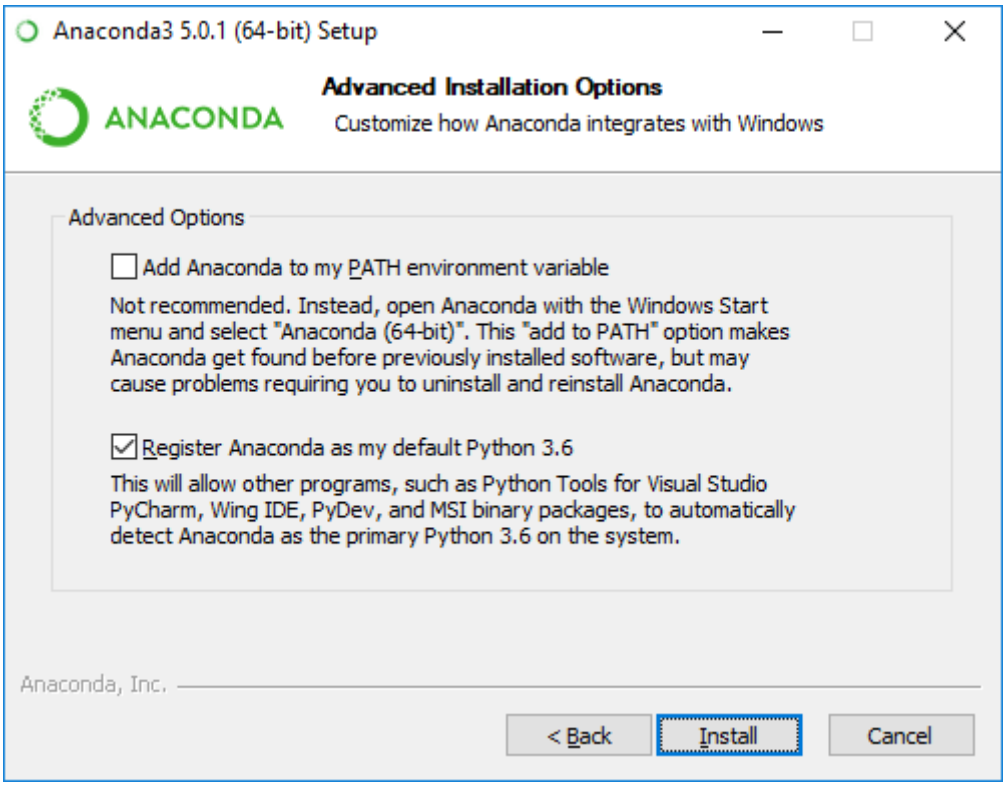
Once the download completes, open and run the ***.exe*** installer



At the beginning of the install, you need to click **Next** to confirm the installation.



At the Advanced Installation Options screen, I recommend that you **do not check** "Add Anaconda to my PATH environment variable"



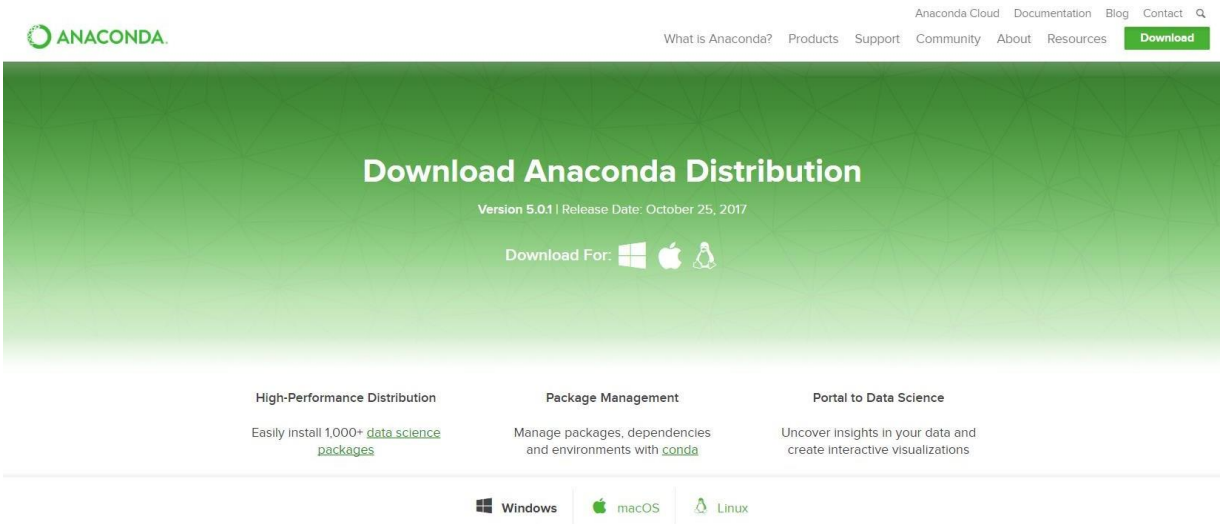
Installation Of Anaconda in Ubuntu

This section details the installation of the Anaconda distribution of Python on Linux, specifically Ubuntu 18.04, but the instructions should work for other Debian-based Linux distributions as well.

Ubuntu 18.04 comes pre-installed with Python (Version 3.6) and legacy Python (Version 2.7). You can confirm the legacy version of Python is installed by opening up a terminal.

1. Visit the Anaconda downloads page

Go to the following link: Anaconda.com/downloads

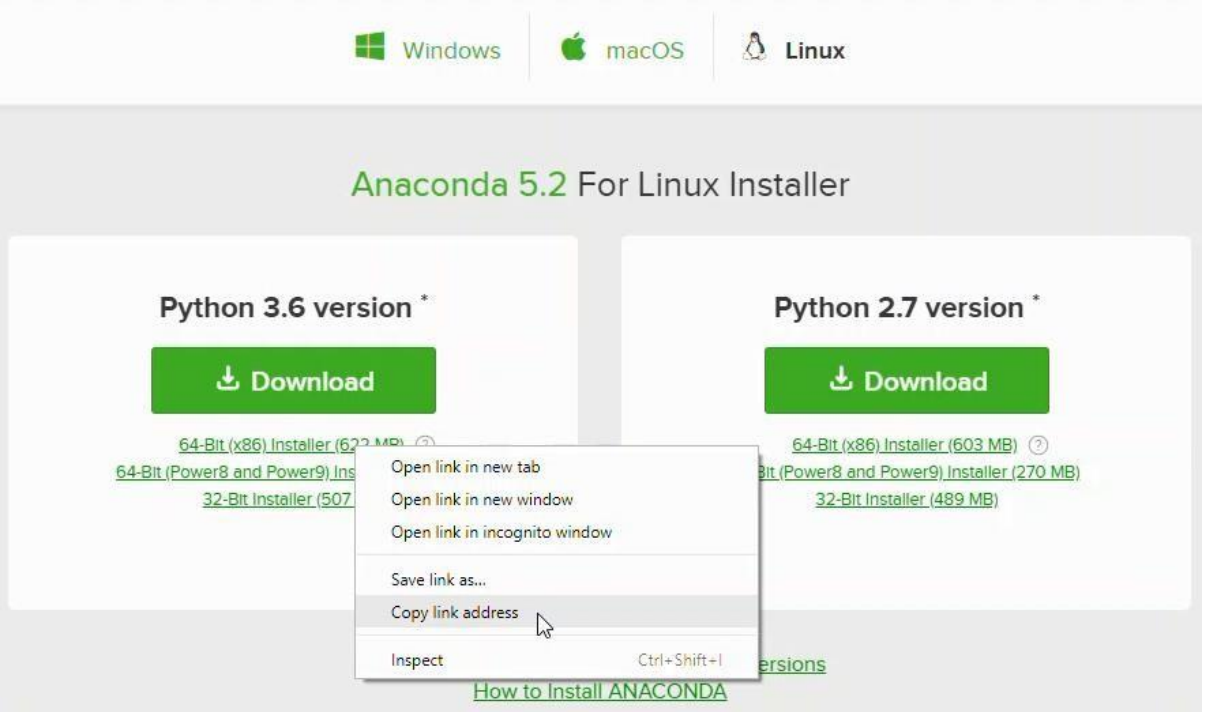


2. Select Linux

On the downloads page, select the Linux operating system.

3. Copy the bash (.sh file) installer link

In the **Python 3.6 Version*** box, right-click on the [64-Bit(x86) Installer] link. Select [copy link address].



4. Use wget to download the bash installer

Now that the bash installer (.sh file) link is stored on the clipboard, use wget to download the installer script. In a terminal, cd into the home directory and make a new directory called tmp. cd into tmp and use wget to download the installer. Although the installer is a bash script, it is still quite large and the download will not be immediate (Note the link below includes <release>. the specific release depends on when you download the installer)

```
$ cd ~
$ mkdir tmp
$ cd tmp
$ https://repo.continuum.io/archive/Anaconda3<release>.sh
```

5. Run the bash script to install **Anaconda3**

With the bash installer script downloaded, run the ***.sh*** script to install **Anaconda3**. Ensure you are in the directory where the installer script downloaded:

$ ls
Anaconda3-5.2.0-Linux-

x86_64.sh Runthe

installerscriptwithbash.

$ bash Anaconda3-5.2.0-Linux-x86_64.sh
Accept the Licence Agreement and allow Anaconda to be added to your PATH. By adding Anaconda to your PATH, the Anaconda distribution of Python will be called when you type $ python in a terminal.

6. source the .bash-rc file to add Anaconda to your PATH

Now that **Anaconda3** is installed and **Anaconda3** is added to ourPATH, source the .bashrc file to load the new PATH environment variable into the current terminalsession.Notethe.bashrcfileisinthehomedirectory. You can see it with $ ls -a.
$ cd ~
$ source .bashrc

Installation Of

NumPy Command –

pip3 install numpy

Installation of Scipy-

stack Command–pip3

installscipy-stack

NUMPY/SCIPY:-
NumPy is a Python library, which stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object, provide tools for integrating C, C++ etc.

- NumPy array can also be used as an efficient multi-dimensional container for generic.data.
- Thendarray(NumPyArray)isamultidimensionalarrayusedtostore values of same datatype. These arrays are indexed just like Sequences, starts with zero.

- The ndarrays are better than regular arrays in terms of faster computation and ease of manipulation.
- In different algorithms of Machine Learning like K-means Clustering, Random Forest etc. we have to store the values in an array. So, instead of using regular array, ndarray helps us to manipulate and execute easily.

Installation of scikit

Command pip3 install -u scikit-learn

SCIKIT:-
The functionality that scikit-learn provides include:

- **Regression**, including Linear and Logistic Regression
- **Classification**, including K-Nearest Neighbors
- **Clustering**, including K-Means and K-Means++
- **Model selection**
- **Preprocessing**, including Min-Max Normalization.

Installation of Pandas
Command- pip3 installpandas

PANDAS:-
- Merging and Joining Data Sets.
- Reshaping & pivoting Data Sets.
- Inserting & deleting columns in Data Structure.
- Aligning data & dealing with missing data.
- Iterating over a Data set.
- Analyzing Time Series.
- Filtering Data around a condition.
- Arranging Data in an ascending & descending.
- Reading from flies with CSV, TXT, XLSX, other formats.
- Manipulating Data using integrated indexing for DataFrame objects.
- Generating Data range, date shifting, lagging, converting frequency, and other other Time Series functionality.
- Subsetttingfancyindexing,&labelbasedslicingDataSetsthatarelarge in size.

- Performingsplitapplycombineon DataSetsusingthegroupbyengine. WithPython Pandas,itiseasiertoclean&wranglewithyourData.featuresof Pandas make it a great choice for Data Science and Analysis.

Installation of Matplotlib

Command- pip3 install

matplotlib

MATPLOTLIB:-

- Matplotlib is a visualization library in Python for 2D plots of arrays. It consists of several plots like line, bar, scatter, histogram etc.
- Matplotlibisamulti-platformdatavisualizationlibrarybuiltonNumPy arrays and designed to work with the broader SciPy stack.It can also be used with graphics toolkits like PyQt and wxPython.
- One of the advantage of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals

Conclusion

Thuswehavesuccessfullyinstalled AnacondaonwindowsandUbuntu;Wealso have described the libraries and the installation of the libraries.

ASSIGNMENT 2 :

Problem Statement: Download the any dataset from UCI or Data.org or from any other data repositories and perform the basic data pre-processing steps using python/R.

Objective

1. Learn to pre-process dataset

2. Learn to use pandas and sklearn

Theory:

Data cleaning:

The main aim of Data Cleaning is to identify and remove errors & duplicate data, in order to create a reliable dataset. This improves the quality of the training data for analytics and enables accurate decision-making.

Normalization:

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Standardization:

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

Basic Steps :

In [4]:
```
data = pd.read_csv("D:\\Desktop\\Downloads\\housing.csv")
```

In [5]:
```
print(data.isna().sum())
```

```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

In [2]:
```
import numpy as np
import pandas as pd
import matpotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
```

In [12]:
```
x = data.iloc[:,:-1]
x
```

Out[12]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0
...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	330.0	1.5603	78100.0
20636	-121.21	39.49	18.0	697.0	150.0	356.0	114.0	2.5568	77100.0
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	433.0	1.7000	92300.0
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	349.0	1.8672	84700.0
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	530.0	2.3886	89400.0

20640 rows × 9 columns

In [10]:
```
label= LabelEncoder()
data['ocean_proximity']=label.fit_transform(data['ocean_proximity'])
```

In [14]:
```
y = data.ocean_proximity
y
```

Out[14]:
```
0        NEAR BAY
1        NEAR BAY
2        NEAR BAY
3        NEAR BAY
4        NEAR BAY
           ...
20635      INLAND
20636      INLAND
20637      INLAND
20638      INLAND
20639      INLAND
Name: ocean_proximity, Length: 20640, dtype: object
```

In [15]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

Conclusion : Thus we have successfully implemented pre-processing operations on a dataset.

ASSIGNMENT 3 :

Problem Statement:

Download the any dataset from UCI or Data.org or from any other data repositories and Implement Single and multilayer perceptron on a dataset.

A. MLP Classifier

B. MLP Regressor

Display the loss graph for the same. Also generate classification report in terms of confusion matrix, precision, recall, and f1 score.

Objective

1. To learn about classification and regression

2. To learn MLP and backpropagation

3. To demonstrate and analyse the results

Theory:

Regression:

A regression problem is when the output variable is a real or continuous value, such as "salary" or "weight". Many different models can be used, the simplest is the linear regression. It tries to fit data with the best hyper-plane which goes through the points.

Classification:

A classification problem is when the output variable is a category, such as "red" or "blue" or "disease" and "no disease". A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. For example, when filtering emails "spam" or "not spam", when looking at transaction data, "fraudulent", or "authorized".

BackPropagation Algorithm:

The algorithm is used to effectively train a neural network through a method called chain rule. In simple terms, after each forward pass through a network, backpropagation performs a backward pass while adjusting the model's parameters (weights and biases). In other words, backpropagation aims to minimize the cost function by adjusting network's weights and biases. The level of adjustment is determined by the gradients of the cost function with respect to those parameters.

Activation function :

Activation functions also known non- linearity, describe the input-output relations in a non-linear way. This gives the model power to be more flexible in describing arbitrary relations. Here are some popular activation functions Sigmoid, Relu, and TanH. I will describe these in my next blog.

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Precision = TP/TP+FP

Recall - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 score - F1 Score is the weighted average of Precision and Recall.

Confusion Matrix :

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This matrix shows that how much data is classified as positive and negative as well as it also indicates that how that positive and negative classification done w.r.t positive and negative classe, i.e. True positive, False Positive, True Negative, False Negative

Code :

In [38]:
```
#Vikalp Bharti 2193279
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

In [7]:
```
data = pd.read_csv("D:\\Desktop\\Downloads\\diabetes.csv")
```

In [8]:
```
print(data.isna().sum())
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [34]:
```
x=data.iloc[:,:-1]
x
```

Out[34]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63
764	2	122	70	27	0	36.8	0.340	27
765	5	121	72	23	112	26.2	0.245	30
766	1	126	60	0	0	30.1	0.349	47
767	1	93	70	31	0	30.4	0.315	23

768 rows × 8 columns

In [35]:
```
y=y=data.Outcome
y
```

Out[35]:
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [36]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

In [37]:
```
mlp = MLPClassifier(hidden_layer_sizes=(100),max_iter=50)
mlp.fit(x_train,y_train)
```

Out[37]:
```
MLPClassifier(hidden_layer_sizes=100, max_iter=50)
```

In [24]:
```
mlp.score(x_train,y_train)
```

Out[24]:
```
0.7133550488599348
```

In [25]:
```
mlp.score(x_test,y_test)
```

Out[25]:
```
0.7337662337662337
```

In [26]:
```
y_pred = mlp.predict(x_test)
```

In [27]:
```
c=confusion_matrix(y_test,y_pred)
print(c)
```

```
[[99 10]
 [31 14]]
```

ASSIGNMENT 4 :

Problem Statement:

Aim:

Develop a Bayesian classifier IRIS dataset dataset

Objective :

1. To learn bayes theorem

2. To implement Bayesian classifier

Theory :

Bayes' Theorem is a way of finding a probability when we know certain other probabilities. The formula is: P(A|B) = P(A) P(B|A)P(B) Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.

Code and Output:

In [22]:
```
#Vikalp Bharti
import pandas as pd
```

In [23]:
```
data = pd.read_csv("D:\\Desktop\\Downloads\\diabetes.csv")
```

In [24]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [25]:
```
x=data.iloc[:,:-1]
x
```

Out[25]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63
764	2	122	70	27	0	36.8	0.340	27
765	5	121	72	23	112	26.2	0.245	30
766	1	126	60	0	0	30.1	0.349	47
767	1	93	70	31	0	30.4	0.315	23

768 rows × 8 columns

In [26]:
```
y=data.Outcome
y
```

Out[26]:
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [27]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=3)
```

In [28]:
```
baye= GaussianNB()
baye.fit(x_train,y_train)
```

Out[28]:
```
GaussianNB()
```

In [31]:
```
baye.score(x_test,y_test)
```

Out[31]:
```
0.7272727272727273
```

In [32]:
```
baye.score(x_train,y_train)
```

Out[32]:
```
0.7746741154562383
```

In [ ]:
```

```

ASSIGNMENT 5 :

Problem Statement:

Implement SVM Classifier or Regression for given dataset

Objective

1. To learn SVM and kernel functions
2. To implement SVM classifier

Theory:

SVM:

support-vector machines (SVMs, also support-vector networks[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,

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.

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

In machine learning, kernel methods are a class of algorithms for pattern analysis, whose best known member is the support vector machine (SVM). The general task of pattern analysis is to find and study general types of relations (for example clusters, rankings, principal components, correlations, classifications) in datasets.

Significance of SVM

It is capable of doing both classification and regression. In this post I'll focus on using SVM for classification. In particular I'll be focusing on non-linear SVM, or SVM using a non-linear kernel. Non-linear SVM means that the boundary that the

algorithm calculates doesn't have to be a straight line. The benefit is that you can capture much more complex relationships between your datapoints without having to perform difficult transformations on your own. The downside is that the training time is much longer as it's much more computationally intensive.

what is the purpose of support vector in SVM?

A support vector machine attempts to find the line that "best" separates two classes of points. By "best", we mean the line that results in the largest margin between the two classes. The points that lie on this margin are the support vectors

Code and Output :

In [33]:
```
#Vikalp Bharti
import pandas as pd
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
```

In [34]:
```
data = pd.read_csv("D:\\Desktop\\Downloads\\breast-cancer.csv")
```

In [35]:
```
print(data.isna().sum())
```

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
```

```
fractal_dimension_worst    0
dtype: int64
```

In [36]:
```
x=data.iloc[:,3:]
x
```

Out[36]:

	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean
0	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.0787
1	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.0566
2	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.0599
3	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.0974
4	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.0588
...	...	...	...	...	...	...	...	...	...
564	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.0562
565	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.0553
566	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.0564
567	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.0701
568	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.0588

569 rows × 29 columns

In [37]:
```
y=data.diagnosis
y
```

Out[37]:
```
0      M
1      M
2      M
3      M
4      M
      ..
564    M
565    M
566    M
567    M
568    B
Name: diagnosis, Length: 569, dtype: object
```

In [38]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=1)
```

In [39]:
```
sv.score(x_test,y_test)
```

Out[39]:
```
0.951048951048951
```

In [40]:
```
sv= SVC(C=3,kernel="linear")
sv.fit(x_train,y_train)
```

Out[40]:
```
SVC(C=3, kernel='linear')
```

In [32]:
```
sv.score(x_train,y_train)
```

Out[32]:
```
0.9694835680751174
```