PROJECT REPORT

ON

OBSCENE COMMENT CLASSIFIER USING MACHINE LEARNING

Submitted in Partial Fulfilment of the Requirement for the award of Degree of

Bachelor of Technology

(Computer Engineering)

By

[Ravi Sahni]

[17 COB **085**]

[GJ7718]

Under the Guidance of

[Rajendra Prasad]

[Scientist-C, NIC HQ, New Delhi]

Department of Computer Engineering

Zakir Husain College of Engineering & Technology

ALIGARH MUSLIM UNIVERSITY, ALIGARH (U.P.)

CANDIDATE’S DECLARATION CERTIFICATE

I hereby certify that the work which is being presented in the report entitled “**OBSCENE COMMENT CLASSIFIER USING MACHINE LEARNING**” by **RAVI SAHNI** in partial fulfilment of requirements for the award of degree of B. Tech. submitted to Department of COMPUTER ENGINEERING, Z.H. College of Engineering & Technology is an authentic record of my own work carried out during a period from 1st June 2019 to 31st July 2019 under the supervision of Mr **Rajendra Prasad** (Scientist-C, NIC – HQ, New Delhi).

Signature of the Student

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge.

Signature of the Supervisor(s)

ACKNOWLEDGEMENT

It needs more than few words to express my immense gratitude and profound thanks to the people who are responsible for the completion of my project report on “OBSCENE COMMENT CLASSIFIER USING MACHINE LEARNING”.

Nevertheless I wish to express my sincere and heartfelt gratitude for those who have helped me in making this project a success.

Good project reports cannot appear out of thin air. So with gratitude I acknowledge all those valuable comments, suggestions and encouragement given to me, making me cruise smoothly along my path and reward my efforts with success.

I am grateful to ZHCET, Aligarh Muslim University for giving me opportunity to carry out this project. I am fortunate to have benefit of guidance of Mr **Muneeb Hasan Khan**, Assistant Professor, Aligarh Muslim University for his valuable guidance and support through the entire length of the project.

I express my sincere thanks to Mr **Rajendra Prasad**, Scientist-C, Network Security Division, National Informatics Centre, New Delhi to whom I am deeply indebted for valuable insight that he has given to me in the course of time. I am also very thankful to all my guides especially Mr **Vikas Kumar** and Mr **Deepak Kumar** for providing the excellent motivation and valuable guidance throughout the project.

Date: July 31st , 2019

CONTENT

COMPANY'S INTRODUCTION………………………………………..…………………………………8

CHAPTER 1

BRIEF OVERVIEW OF THE PROJECT

1.1 Objective……………………………………………………………………..…………………...13

1.2 Project Description……………………………………..……………..………………….13

1.3 Need to choose the Project…………………………………………..…..………...14

1.4 Applications……………………………………………………………..………………….14

1.5 Technologies Used………………………………………………...............................14

CHAPTER 2

INTRODUCTION TO MACHINE LEARNING

2.1 Definition…………………………………………………………………...……………………….15

2.2 Why we need Machine Learning?………………………………………………….15

2.3 Types of Machine Learning…………………………………………………………….16

CHAPTER 3

SUPERVISED LEARNING

3.1 Supervised Learning & its Types……………………………….......................17

3.2 Regression…………………….…………………………………………..……………………18

3.2.1 Simple Linear Regression

3.2.2 Multiple Linear Regression

3.2.3 Polynomial Regression

3.2.4 Support Vector Regression

3.2.5 Ridge Regression
3.2.6 Lasso Regression
3.2.7 ElasticNet Regression
3.2.8 Bayesian Regression
3.2.9 Decision Tree Regression
3.2.10 Random Forest Regression

3.3 Classification……………………………………………………………………………….22
3.3.1 Logistic Regression/Classification
3.3.2 K-Nearest Neighbours
3.3.3 Support Vector Machines
3.3.4 Kernel Support Vector Machines
3.3.5 Naive Bayes
3.3.6 Decision Tree Classification
3.3.7 Random Forest Classification

CHAPTER 4

UNSUPERVISED LEARNING

4.1 Unsupervised Learning & its Types………………………………………………30
4.2 Clustering……………………………………………………………………………………..31
4.2.1 K-Means Clustering
4.2.2 Hierarchical Clustering
4.3 Dimensionality Reduction…………………………………………………………………..33
4.3.1 Principal Component Analysis
4.3.2 Linear Discriminant Analysis
4.3.3 Kernel Principal Component Analysis

CHAPTER 5

REINFORCEMENT LEARNING

5.1 Reinforcement Learning & its Types……………………………………………….......35
5.2 Q-Learning……………………...................………………………………………………36
5.3 SARSA [State Action Reward State Action]…………………….................36
5.4 Deep Q-Network…………………………………..……………………………………….37
5.5 Markov Decision Processes………………………………….……………………...37
5.6 DDPG[Deep Deterministic Policy Gradient]………………………………….38

CHAPTER 6

SEMI-SUPERVISED LEARNING

6.1 What is Semi-Supervised Learning?……………………..………………………39

CHAPTER 7

MACHINE LEARNING – APPLICATIONS

7.1 Some Applications of ML……………………………………..…......................40
7.2 Examples……………………………………………………………………………………….41

CHAPTER 8

INTRODUCTION TO FLASK

8.1 What is Flask?………..…….43
8.2 A Minimal Application…………………….……………………………………….43
8.3 Routing……………………………………………………………………………………….44
8.3.1 Variable Rules
8.3.2 Unique URLs / Redirection Behaviour
8.3.3 URL Building
8.3.4 HTTP Methods

8.4 Static Files……………………………………………………………………………………48
8.5 Rendering Templates……………………………………………………………….....48
8.6 Accessing Request Data………………………………………..........................49
8.6.1 Context Locals
8.6.2 The Request Object
8.6.3 File Uploads
8.7 Redirects and Errors…………………………………………..............................52

CHAPTER 9

UNDERSTANDING THE PROJECT

9.1 The Algorithm………...…..53
9.2 Developing the Web Application…………………………………………………57

CHAPTER 10

USING THE APPLICATION (USER GUIDE)

10.1 Making the System ready……..…59
10.2 Running the Application……………………………………...........................60

CHAPTER 11

FUTURE ENHANCEMENT……………………………………………………………………………64

BIBLIOGRAPHY……………………...…..65

NATIONAL INFORMATICS CENTRE (NIC)

राष्ट्रीय सूचना विज्ञान केंद्र

ABOUT NIC

The National Informatics Centre (NIC) is the premier science and technology organisation of the Government of India in informatics services and information and communication technology (ICT) applications. It is part of the Indian Ministry of Electronics and Information Technology's Department of Electronics & Information Technology.

It plays a pivotal role in steering e-governance applications in the governmental departments at national, state and district levels, enabling the improvement in, and a wider transparency of, government services. Almost all Indian-government websites are developed and managed by NIC.

ORGANISATION

NIC is a part of the Indian Ministry of Electronics & Information Technology's Department of Electronics & Information Technology and is headquartered in New Delhi. It has offices in all 29 state capitals and 7 union-territory headquarters and almost all districts. At New Delhi Headquarters, Mean Head a large number of Application Divisions exist which provide total Informatics Support to the Ministries and Departments of the Central Government. To cater to the ICT needs at the grassroots level, the NIC has also opened offices in almost all district collectorates. NIC Extends Technical Coordination and IT support to District Administration. NIC computer cells are located in almost all Ministry bhawans (buildings) of the central government and Apex offices including the Indian Prime Minister’s office, the Indian

Presidential Palace (Rashtrapati Bhavan) and India's Parliament House (Sansad Bhavan). It also provides support to grassroot-level administration.

HISTORY

National Informatics Centre (NIC) was established in 1976, and has since emerged as a “prime builder” of e-Government / e-Governance applications up to the grassroots level as well as a promoter of digital opportunities for sustainable development. NIC, through its ICT Network, “NICNET”, has institutional linkages with all the Ministries /Departments of the Central Government, 36 State Governments/ Union Territories, and about 708 District Administrations of India. NIC has been instrumental in steering e-Government/e-Governance applications in government ministries/departments at the Centre, States, Districts and Blocks, facilitating improvement in government services, wider transparency, promoting decentralized planning and Centre, States, Districts and Blocks, facilitating improvement in government services, wider transparency, promoting decentralized planning and management, resulting in better efficiency and accountability to the people of India.

ACTIVITIES UNDERTAKEN:

“Informatics-led-development” programme of the government has been spearheaded by NIC to derive competitive advantage by implementing ICT applications in social & public administration. The following major activities are being undertaken:

- Setting up of ICT Infrastructure
- Implementation of National and State Level e-Governance Projects/Products
- Consultancy to the Government departments
- Research, Development and Capacity Building

FIELD OF WORK

During the last three decades, NIC has implemented many "network centric" application software for Programme implementation in various ministries and departments, using state-of-the-technology software tools. During 1980s and early part of 1990s, the policy thrust was on creating "Management Information System (MIS)" and "Decision Support System (DSS)" for development , planning and responsive administration in governments which led to the genesis of present day "e-Governance" / "e-Government". "Bridging the Digital Divide", "Social and Financial Inclusion through ICT" and "Reaching- the-Unreached" concepts were tried and made operational in the late nineties.

NIC has vast expertise and experience in the design, development and operationalization of various e-Government projects in the areas of Public Administration and Governance like Agriculture & Food, Animal Husbandry, Fisheries, Forestry & Environment, Industry, Health, Education, Budget and Treasury, Fiscal Resources, Transport, Water Resources, Court Management, Rural Development, Land Records and Property registration, Culture & Tourism, Import & Exports facilitation, Social Welfare Services, Micro-level Planning, etc.

With increasing awareness leading to demand and availability of ICT infrastructure with better capacities and programme framework, the governance space in the country witnessed a new round of projects and products, covering the entire spectrum of e-Governance including G2C, G2B, G2G, with emphasis on service delivery.

NIC provides Nationwide Common ICT Infrastructure to support e-Governance services to the citizen, Products and Solutions designed to address e-Governance Initiatives, Major e-Governance Projects, State/UT Informatics Support and district level services rendered.

NIC has set up state-of-the-art ICT infrastructure consisting of National and state Data Centres to manage the information systems and websites of Central Ministries/Departments, Disaster Recovery Centres, Network Operations facility to manage heterogeneous networks spread across Bhawans, States and Districts, Certifying Authority, Video-Conferencing and capacity building across the country. National Knowledge Network (NKN) has been set up to connect institutions/organizations carrying out research and development, Higher Education and Governance with speed of the order of multi Gigabits per second. Further, State Government secretariats are connected to the Central Government by very high speed links on Optical Fibre Cable (OFC). Districts are connected to respective State capitals through leased lines.

Various initiatives like Government eProcurement System(GePNIC), Office Management Software (eOffice), Hospital Management System (eHospital), Government Financial Accounting Information System (eLekha), etc. have been taken up which are replicable in various Government organizations.

As NIC is supporting a majority of the mission mode e-Governance projects, the chapter on National e-Governance Projects lists the of details of these projects namely National Land Records Modernization Programme (NLRMP), Transport and National Registry, Treasury Computerisation, VAT, MG-NREGA, India-Portal, e-Courts, Postal Life Insurance, etc. NIC also lays framework and designs systems for online monitoring of almost all central government schemes like Integrated Watershed Management (IWMP), IAY, SGSY, NSAP, BRGF, Schedule Tribes and other Traditional Forest Dwellers Act etc.

ICT support is also being provided in the States / UTs by NIC. Citizen centric services are also being rendered electronically at the district level, such as Income Certificate, Caste Certificate, and Residence Certificate etc. along with other services like Scholarship portals, permits, passes, licenses to name a few.

Thus, NIC, a small program started by the external stimulus of an UNDP project, in the early 1970s, became fully functional in 1977 and since then has grown with tremendous momentum to become one of India's major S&T; organizations promoting informatics led development. This has helped to usher in the required transformation in government to ably meet the challenges of the new millennium.

1. BRIEF OVERVIEW OF THE PROJECT

1.1 Objective:

1. One area of focus is the study of negative online behaviours, like toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion).

2. Keeping online conversations constructive and inclusive is a crucial task for platform providers.

3. Automatic classification of toxic comments, such as hate speech, threats, and insults, can help in keeping discussions fruitful.

1.2 Project Description:

The pre-built machine model analyses the text and predicts the probability of toxic contents risk in 6 different categories, including "toxic", "severe toxic", "obscene", "threat", "insult", and "identity hate".

The project uses concept of machine learning and has been implemented using Python, Flask, HTML and CSS which trains then tests the machine from a huge data set to predict the probability of various words and hence classify the various statements accordingly.

1.3 Need to choose the Project:

Inspired by the idea of keeping the online environment productive, respectful, and free of profane, vulgar, or offensive languages, I'd like to introduce a toxic comment classifier developed using machine learning that can be used in various online platforms (chat applications, discussion forums, etc.) to eliminate obscenity and promote productive communication and discussions.

1.4 Applications:

1. Chat Applications
2. Group discussion forums
3. Elimination of obscenity from social media

1.5 Technologies Used:

1. Machine Learning (Using Python v3.7.3)
2. Flask (Backend)
3. HTML/CSS/JavaScript (Frontend)

2. INTRODUCTION TO MACHINE LEARNING

2.1 Definition:

Machine learning was defined in 90’s by Arthur Samuel described as the,” it is a field of study that gives the ability to the computer for self-learn without being explicitly programmed”, that means imbuing knowledge to machines without hard-coding it.

> “A computer algorithm/program is said to learn from performance measure P and experience E with some class of tasks T if its performance at tasks in T, as measured by P, improves with experience E.” -*Tom M. Mitchell*.

Machine learning is mainly focused on the development of computer programs which can teach themselves to grow and change when exposed to new data. Machine learning studies algorithms for self-learning to do stuff. It can process massive data faster with the learning algorithm. For instance, it will be interested in learning to complete a task, make accurate predictions, or behave intelligently.

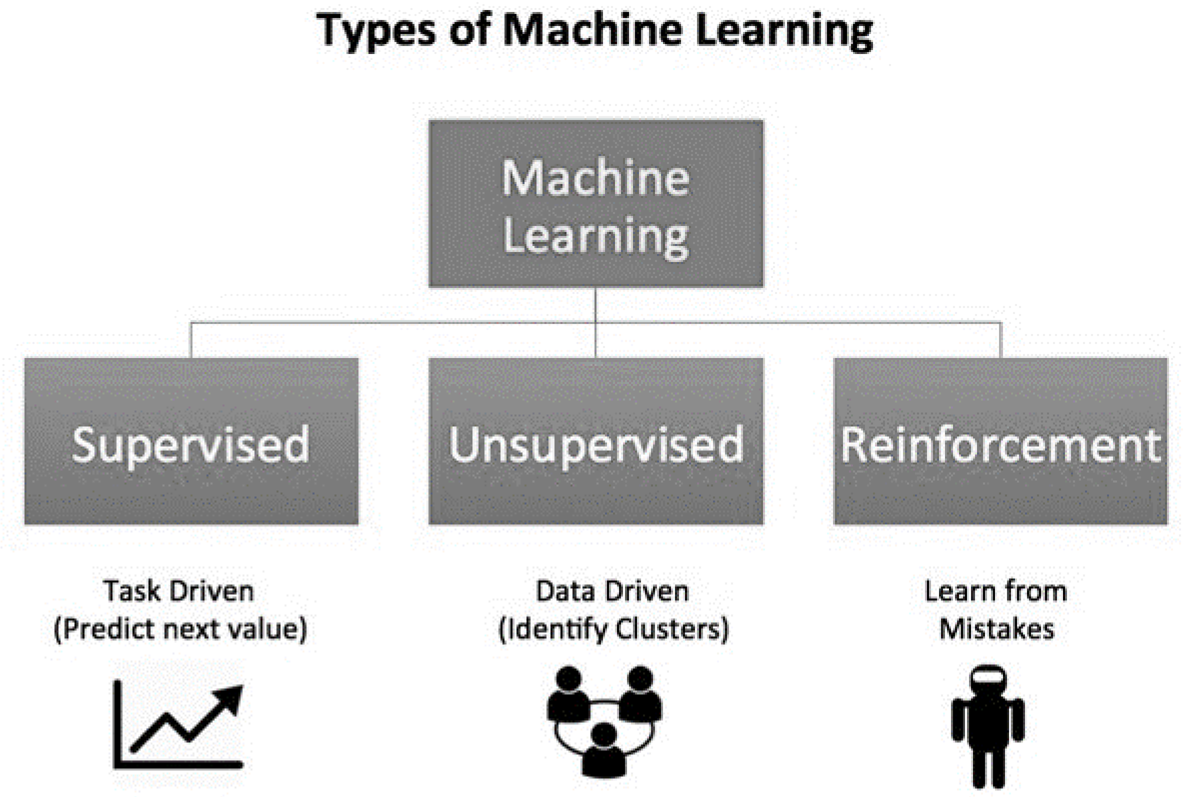
2.2 Why we need Machine Learning? :

Data is growing day by day, and it is impossible to understand all of the data with higher speed and higher accuracy. More than 80% of the data is unstructured that is audios, videos, photos, documents, graphs, etc. Finding patterns in data on planet earth is impossible for human brains. The data has been very massive, the time taken to compute would increase, and this is where Machine Learning comes into action, to help people with significant data in minimum time.

Machine Learning is a sub-field of AI. Applying AI, we wanted to build better and intelligent machines. It sounds similar to a new child learning from itself. So in the machine learning, a new capability for computers was developed. And now machine learning is present in so many segments of technology, that we don’t even realise it while using it.

2.3 Types of Machine Learning:

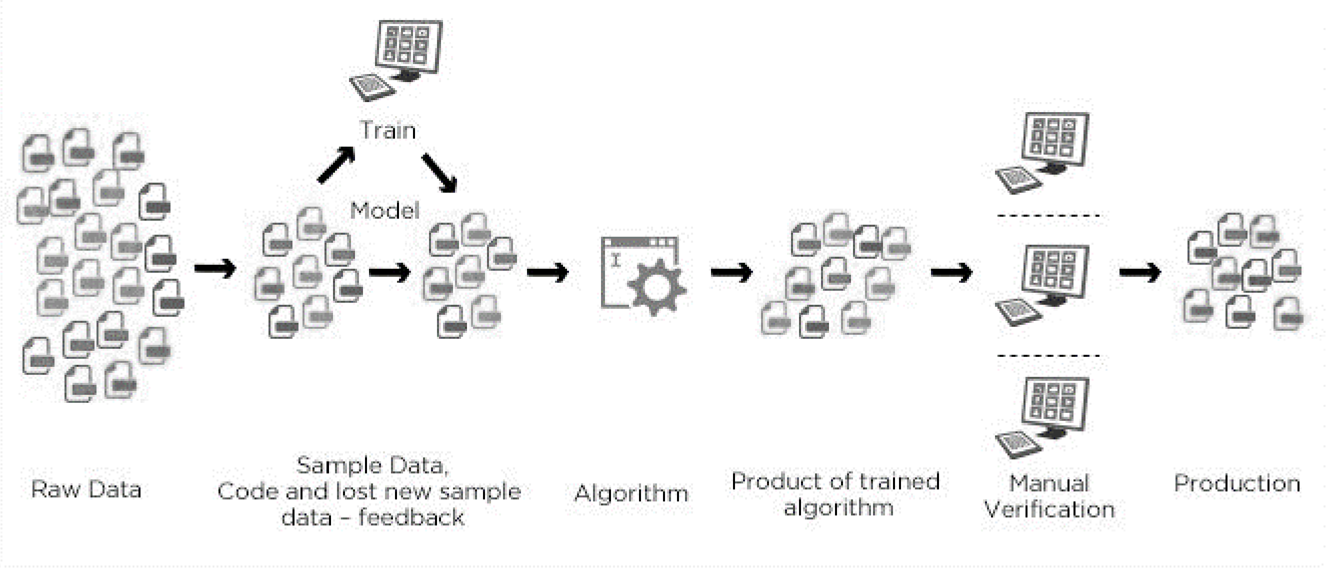
Machine Learning mainly divided into three categories, which are as follows-


Types of Machine Learning
Machine Learning
Supervised
Unsupervised
Reinforcement
Task Driven (Predict next value)
Data Driven (Identify Clusters)
Learn from Mistakes

3. SUPERVISED LEARNING

3.1 Supervised Learning & its Types:

Supervised Learning is the first type of machine learning, in which labelled data used to train the algorithms. In supervised learning, algorithms are trained using marked data, where the input and the output are known. We input the data in the learning algorithm as a set of inputs, which is called as Features, denoted by X along with the corresponding outputs, which is indicated by Y, and the algorithm learns by comparing its actual production with correct outputs to find errors. It then modifies the model accordingly. The raw data divided into two parts. The first part is for training the algorithm, and the other region used for test the trained algorithm.

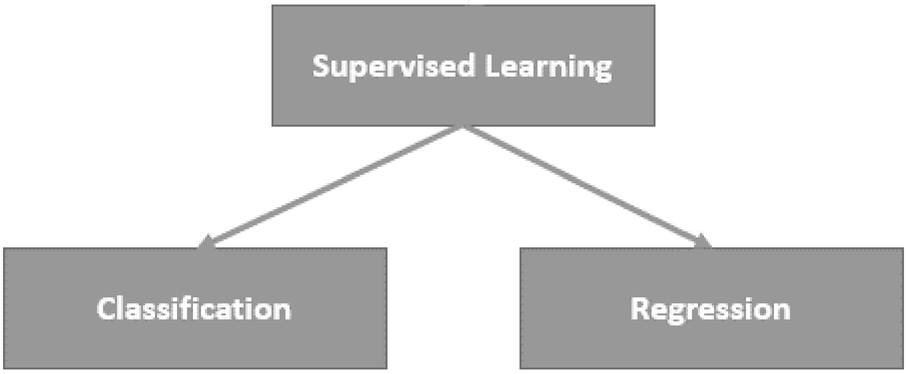

Train
Model
Raw Data
Sample Data, Code and lost new sample data – feedback
Algorithm
Product of trained algorithm
Manual Verification
Production

Supervised Learning

Supervised learning uses the data patterns to predict the values of additional data for the labels. This method will commonly use in applications where historical data predict likely upcoming events. Ex: - It can anticipate when transactions are likely to be fraudulent or which insurance customer is expected to file a claim.

Types of Supervised Learning:-

The Supervised Learning mainly divided into two parts which are as follows-



3.2 Regression:

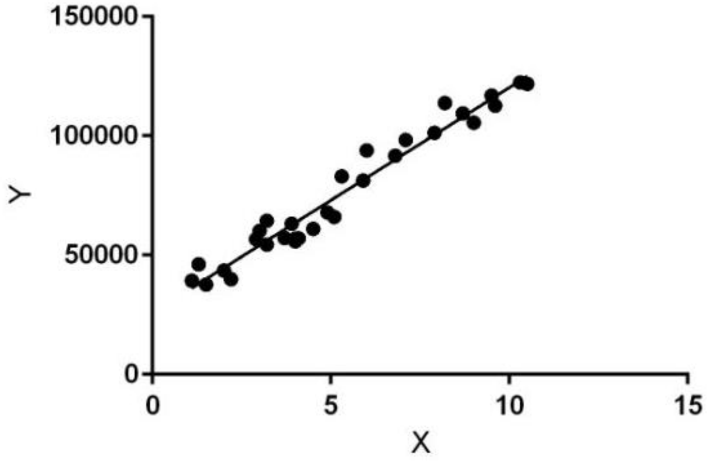
Regression is the type of Supervised Learning in which labelled data used, and this data is used to make predictions in a continuous form. The output of the input is always ongoing, and the graph is linear. Regression is a form of predictive modelling technique which investigates the relationship between a dependent variable [*Outputs*] and independent variable [*Inputs*]. This technique used for forecasting the weather, time series modelling, process optimisation. Ex: - One of the examples of the regression technique is House Price Prediction, where the price of the house will predict from the inputs such as No of rooms, Locality, Ease of transport, Age of house, Area of a home.

Types of Regression Algorithms: -

There are many Regression algorithms are present in machine learning, which will use for different regression applications. Some of the main regression algorithms are as follows-

3.2.1 **Simple Linear Regression:-**

In simple linear regression, we predict scores on one variable from the ratings on a second variable. The variable we are forecasting is called the criterion variable and referred to as Y. The variable we are basing our predictions on is called the predictor variable and denoted to as X. e.g.: In the figure below, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.



Hypothesis function for Linear Regression:

$$y = \theta_1 + \theta_2.x$$

While training the model we are given:

x: input training data (univariate – one input variable(parameter))

y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.

θ1: intercept

θ2: coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

Cost Function (J): By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ1 and θ2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y).

$$minimize\frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

$$J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

Cost function(J) of Linear Regression is the Root Mean Squared Error (RMSE) between predicted y value (pred) and true y value (y).

3.2.2 **Multiple Linear Regression:-**

Multiple linear regression is one of the algorithms of regression technique, and it is the most common form of linear regression analysis. As a predictive analysis, the multiple linear regression is used to explain the relationship between one dependent variable with two or more than two independent variables. The independent variables can be continuous or categorical.

3.2.3 **Polynomial Regression:-**

Polynomial regression is another form of regression in which the maximum power of the independent variable is more than 1. In this regression technique, the best fit line is not a straight line instead it is in the form of a curve.

3.2.4 **Support Vector Regression:-**

Support Vector Regression can be applied not only to regression problems, but it also used in the case of classification. It contains all the features that characterise maximum margin algorithm. Linear learning machine mapping leans a non-linear function into high dimensional kernel-induced feature space. The system capacity was controlled by parameters that do not depend on the dimensionality of feature space.

3.2.5 **Ridge Regression:-**

Ridge Regression is one of the algorithms used in Regression technique. It is a technique for analysing multiple regression data that suffer from multicollinearity. By the addition of a degree of bias to the regression calculates, it reduces the standard errors. The net effect will be to give calculations that are more reliable.

3.2.6 **Lasso Regression:-**

Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

3.2.7 **ElasticNet Regression:-**

Elastic net regression combined L1 norms (LASSO) and L2 norms (ridge regression) into a penalised model for generalised linear regression, and it gives it sparsity (L1) and robustness (L2) properties.

3.2.8 **Bayesian Regression:-**

Bayesian regression allows a reasonably natural mechanism to survive insufficient data or poorly distributed data. It will enable you to put coefficients on the prior and the noise so that the priors can take over in the absence of data. More importantly, you can ask Bayesian regression which parts (if any) of its fit to the data are it confident about, and which parts are very uncertain.

3.2.9 **Decision Tree Regression:-**

Decision tree builds a form like a tree structure from regression models. It breaks down the data into smaller subsets and while an associated decision tree developed incrementally at the same time. The result is a tree with decision nodes and leaf nodes.

3.2.10 **Random Forest Regression:-**

Random Forest is also one of the algorithms used in regression technique, and it is very flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning. Also, this algorithm widely used because of its simplicity and the fact that it can use for both regression and classification tasks. The forest it builds, is an ensemble of Decision Trees, most of the time trained with the “bagging” method.

3.3 Classification:

Classification is the type of Supervised Learning in which labelled data can use, and this data is used to make predictions in a non-continuous form. The output of the information is not always continuous, and the graph is non-linear. In the classification technique, the algorithm learns from the data input given to it and then uses this learning to classify new observation. This data set may merely be bi-class, or it may be multi-class too. Ex: - One of the examples of classification problems is to check whether the email is spam or not spam by train the algorithm for different spam words or emails.

Types of Classification Algorithms:-

There are many Classification algorithms are present in machine learning, which used for different classification applications. Some of the main classification algorithms are as follows-

3.3.1 **Logistic Regression/Classification:-**

Logistic regression falls under the category of supervised learning; it measures the relationship between the dependent variable which is categorical with one or more than one independent variables by estimating probabilities using a logistic/sigmoid function. Logistic regression can generally use where the dependent variable is Binary or Dichotomous. It means that the dependent variable can take only two possible values like “Yes or No”, “Living or dead”.

e.g.: Consider an example dataset which maps the number of hours of study with the result of an exam. The result can take only two values, namely passed (1) or failed (0):

HOURS(X)	0.50	0.75	1.00	1.25	1.50	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	3.75	4.00	4.25	4.50	4
PASS(Y)	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**1**	**1**

$$y = \begin{cases} 0, iffail \\ 1, ifpass \end{cases}$$

So, we have,

i.e. y is a categorical target variable which can take only two possible type:“0” or “1”. In order to generalize our model, we assume that:

- The dataset has ‘p’ feature variables and ‘n’ observations.
- The feature matrix is represented as:

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}$$

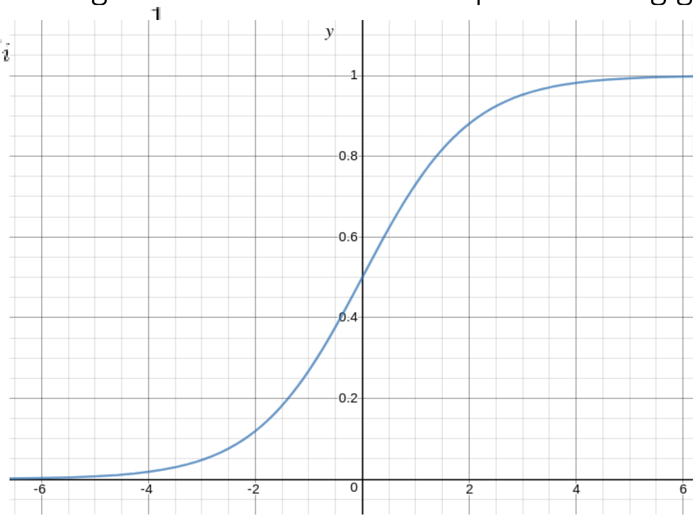
  Here, x_{ij} denotes the values of j^{th} feature for i^{th} observation. Also, we are keeping the convention of letting x_{i0}= 1.

$$x_i = \begin{bmatrix} 1 \\ x_{i1} \\ x_{i2} \\ . \\ . \\ x_{ip} \end{bmatrix}$$

- The i^{th} observation, x_i can be represented as:
- $h(x_i)$ represents the predicted response for i^{th} observation, i.e. x_i. The formula we use for calculating $h(x_i)$ is called **hypothesis** and is given by:

where, $g(z) = \frac{1}{1+e^{-z}}$ is called logistic function or the sigmoid function. Here is a plot showing g(z):

$$h(x_i) = g(\beta^T x_i$$



We can infer from above graph that:

- g(z) tends towards 1 as z→∞
- g(z) tends towards 0 as z→ - ∞
- g(z) is always bounded between 0 and 1

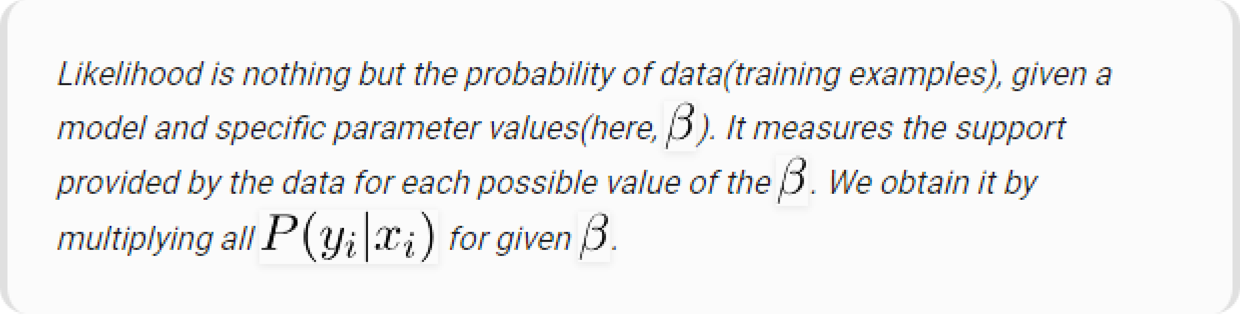
So, now, we can define conditional probabilities for 2 labels (0 and 1) for ith observation as:

$$P(y_i = 1|x_i;\beta) = h(x_i)$$
$$P(y_i = 0|x_i;\beta) = 1 - h(x_i)$$

We can write it more compactly as: $P(y_i|x_i;\beta) = (h(x_i))^{y_i}(1 - h(x_i))^{1-y_i}$

Now, we define another term, **likelihood of parameters** as:

$$L(\beta) = \prod_{i=1}^{n} P(y_i|x_i;\beta)$$
$$or$$
$$L(\beta) = \prod^{n}(h(x_i))^{y_i}(1 - h(x_i))^{1-y_i}$$



Likelihood is nothing but the probability of data(training examples), given a model and specific parameter values(here, β). It measures the support provided by the data for each possible value of the β. We obtain it by multiplying all $P(y_i|x_i)$ for given β.

And for easier calculations, we take log likelihood:

$$l(\beta) = log(L(\beta))$$
$$or$$
$$l(\beta) = \sum_{i=1}^{n} y_i log(h(x_i)) + (1 - y_i)log(1 - h(x_i))$$

The cost function for logistic regression is proportional to inverse of likelihood of parameters. Hence, we can obtain an expression for cost function, J using log likelihood equation as:

$$J(\beta) = \sum_{i=1}^{n} -y_i log(h(x_i)) - (1 - y_i)log(1 - h(x_i))$$

and our aim is to estimate β so that cost function is minimized.

Using Gradient descent algorithm: Firstly, we take partial derivatives of J(β) w.r.t each β_j € β to derive the stochastic gradient descent rule:

$$\frac{\partial J(\beta)}{\partial \beta_j} = (h(x) - y)x_j$$

Here, y and h(x) represent the response vector and predicted response vector (respectively). Also, x_j is the vector representing the observation values for j^{th} feature. Now, in order to get min J(β),

$$\begin{array}{l} Repeat\{ \\ \beta_j := \beta_j - \alpha \sum_{i=1}^{n} (h(x_i) - y_i)x_{ij} \\ (Simultaneously\ update\ all\ \beta_j) \\ \} \end{array}$$

where α is called **learning rate** and needs to be set explicitly.

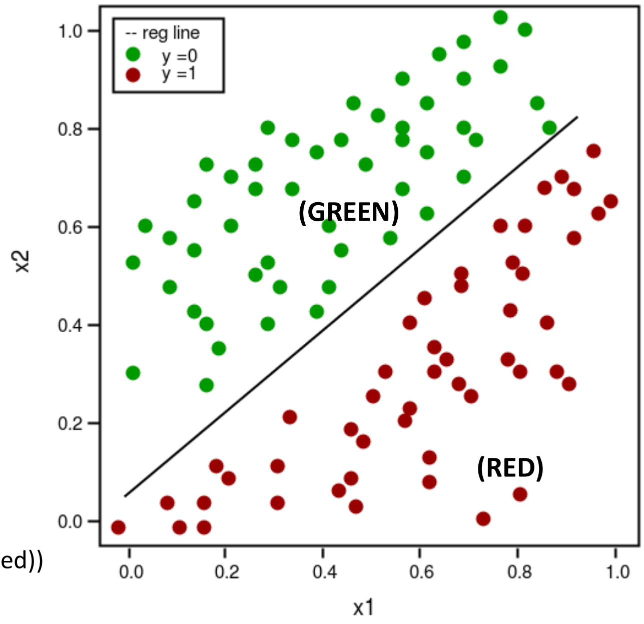
Let us see the python implementation of above technique on a sample dataset

(it can be downloaded from https://github.com/nikhilkumarsingh/Machine-Learning-Samples/blob/master/Logistic_Regression/dataset1.csv)

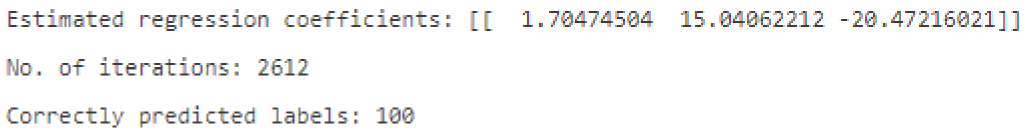
```
import csv
import numpy as np
import matplotlib.pyplot as plt
def loadCSV(filename):
        with open(filename,"r") as csvfile:
                lines = csv.reader(csvfile)
                dataset = list(lines)
                for i in range(len(dataset)):
                        dataset[i] = [float(x) for x in dataset[i]]
        return np.array(dataset)
def normalize(X):
        mins = np.min(X, axis = 0)
        maxs = np.max(X, axis = 0)
        rng = maxs - mins
        norm_X = 1 - ((maxs - X)/rng)
        return norm_X
def logistic_func(beta, X):
        return 1.0/(1 + np.exp(-np.dot(X, beta.T)))
```

```
def log_gradient(beta, X, y):
    first_calc = logistic_func(beta, X) - y.reshape(X.shape[0], -1)
    final_calc = np.dot(first_calc.T, X)
    return final_calc
def cost_func(beta, X, y):
    log_func_v = logistic_func(beta, X)
    y = np.squeeze(y)
    step1 = y * np.log(log_func_v)
    step2 = (1 - y) * np.log(1 - log_func_v)
    final = -step1 - step2
    return np.mean(final)
def grad_desc(X, y, beta, lr=.01, converge_change=.001):
    cost = cost_func(beta, X, y)
    change_cost = 1
    num_iter = 1
    while(change_cost > converge_change):
        old_cost = cost
        beta = beta - (lr * log_gradient(beta, X, y))
        cost = cost_func(beta, X, y)
        change_cost = old_cost - cost
        num_iter += 1
    return beta, num_iter
def pred_values(beta, X):
    pred_prob = logistic_func(beta, X)
    pred_value = np.where(pred_prob >= .5, 1, 0)
    return np.squeeze(pred_value)
def plot_reg(X, y, beta):
    x_0 = X[np.where(y == 0.0)]
    x_1 = X[np.where(y == 1.0)]
    plt.scatter([x_0[:, 1]], [x_0[:, 2]], c='b', label='y = 0')
    plt.scatter([x_1[:, 1]], [x_1[:, 2]], c='r', label='y = 1')
    x1 = np.arange(0, 1, 0.1)
    x2 = -(beta[0,0] + beta[0,1]*x1)/beta[0,2]
    plt.plot(x1, x2, c='k', label='reg line')


    plt.xlabel('x1')
```

```
        plt.ylabel('x2')
        plt.legend()
        plt.show()


if __name__ == "__main__":
        dataset = loadCSV('dataset1.csv')
        X = normalize(dataset[:, :-1])
        X = np.hstack((np.matrix(np.ones(X.shape[0])).T, X))
        y = dataset[:, -1]
        beta = np.matrix(np.zeros(X.shape[1]))
        beta, num_iter = grad_desc(X, y, beta)
        print("Estimated regression coefficients:", beta)
        print("No. of iterations:", num_iter)
        y_pred = pred_values(beta, X)
        print("Correctly predicted labels:", np.sum(y == y_pred))
        plot_reg(X, y, beta)
```



Output:



```
Estimated regression coefficients: [[  1.70474504  15.04062212 -20.47216021]]
No. of iterations: 2612
Correctly predicted labels: 100
```

3.3.2 **K-Nearest Neighbours:-**

KNN algorithm is one of the most straightforward algorithms in classification, and it is one of the most used learning algorithms. A majority vote of an object is classified by its neighbours, with the purpose being assigned to the class most common among its k nearest neighbours. It can also use for regression — output is the value of the object (predicts continuous values). This value is the average (or median) of the benefits of its k nearest neighbours.

3.3.3 **Support Vector Machines:-**

A Support Vector Machine is a type of Classifier, in which a discriminative classifier formally defined by a separating hyperplane. The algorithm outputs an optimal hyperplane which categorises new examples. In two dimensional space, this hyperplane is a line dividing a plane into two parts wherein each class lay on either side.

3.3.4 **Kernel Support Vector Machines:-**

Kernel-SVM algorithm is one the algorithms used in classification technique, and it is mathematical functions set that defined as the kernel. The purpose of the core is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types. For example linear and nonlinear functions, polynomial functions, radial basis function, and sigmoid functions.

3.3.5 **Naive Bayes:-**

Naive Bayes is a type of Classification technique, which based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other function. Naive Bayes model is accessible to build and particularly useful for extensive datasets.

3.3.6 **Decision Tree Classification:-**

Decision tree makes classification models in the form of a tree structure. An associated decision tree incrementally developed and at the same time It breaks down a large data-set into smaller subsets. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Root) has two or more branches. Leaf node represents a classification or decision. The first decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

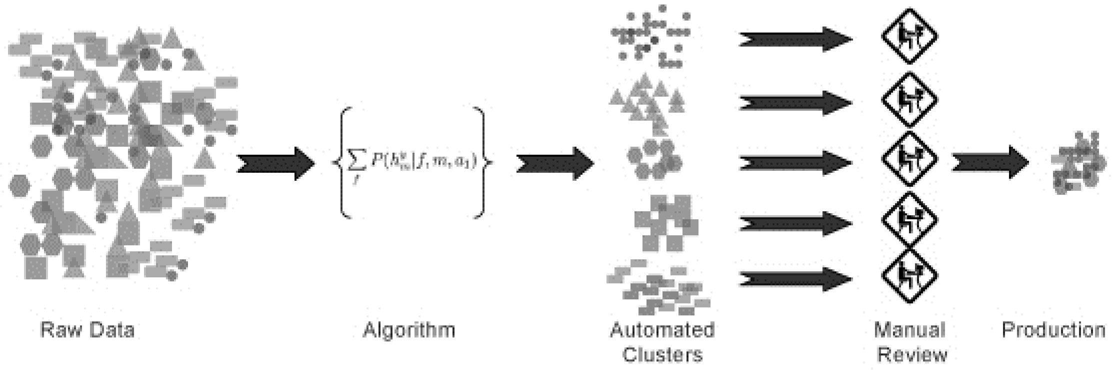
3.3.7 **Random Forest Classification:-**

Random Forest is a supervised learning algorithm. It creates a forest and makes it somehow casual. The wood it builds is an ensemble of Decision Trees, it most of the time the decision tree algorithm trained with the “bagging” method, which is a combination of learning models increases the overall result.

4. UNSUPERVISED LEARNING

4.1 Unsupervised Learning & its Types:

Unsupervised Learning is the second type of machine learning, in which unlabelled data are used to train the algorithm, which means it used against data that has no historical labels. What is being showing must figure out by the algorithm. The purpose is to explore the data and find some structure within. In unsupervised learning the data is unlabelled, and the input of raw information directly to the algorithm without pre-processing of the data and without knowing the output of the data and the data cannot divide into a train or test data. The algorithm figures out the data and according to the data segments, it makes clusters of data with new labels.



Unsupervised Machine Learning

This learning technique works well on transactional data. For example, it can identify segments of customers with similar attributes who can then be treated similarly in marketing campaigns. Or it can find the primary qualities that separate customer segments from each other. These algorithms are also used to segment text topics, recommend items and identify data outliers.

Types of Unsupervised Learning:-

The Unsupervised Learning mainly divided into two parts which are as follows-

- Clustering
- Dimensionality Reduction

4.2 Clustering:

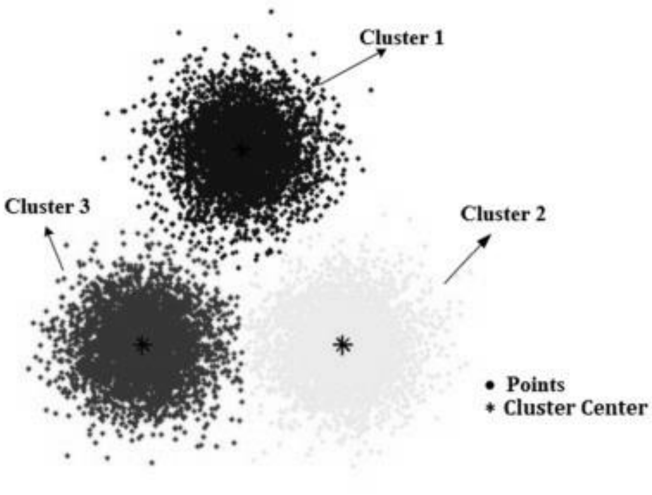
Clustering is the type of Unsupervised Learning in which unlabeled data used, and it is the process of grouping similar entities together, and then the grouped data is used to make clusters. The goal of this unsupervised machine learning technique is to find similarities in the data point and group similar data points together and to figures out that new data should belong to which cluster.

Types of Clustering Algorithms:-

There are many Clustering algorithms are present in machine learning, which is used for different clustering applications. Some of the main clustering algorithms are as follows-

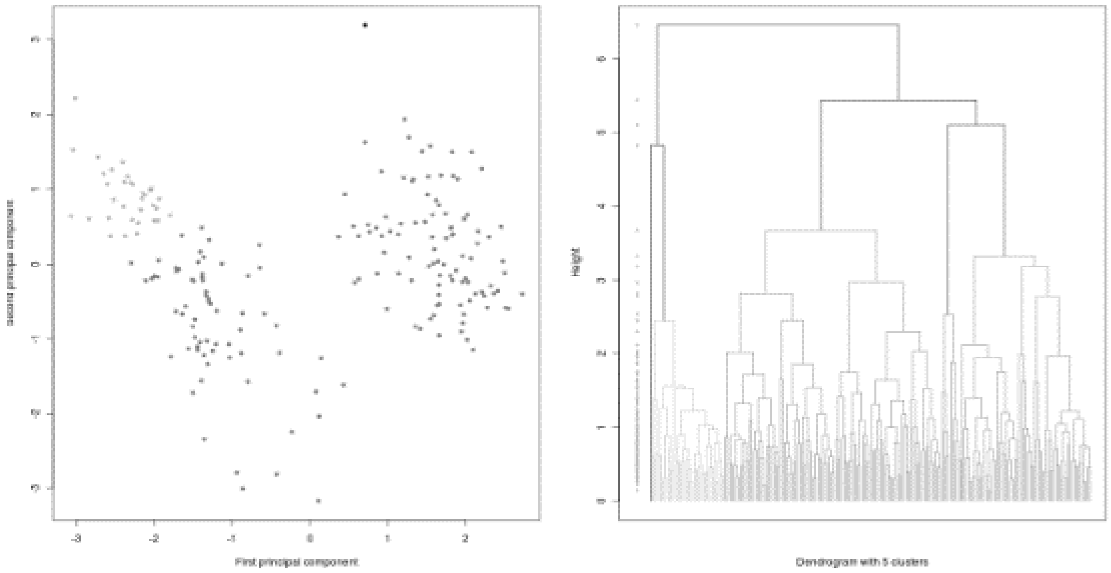
4.2.1 **K-Means Clustering:-**

K-Means clustering is one of the algorithms of Clustering technique, in which similar data grouped in a cluster. K-means is an iterative clustering algorithm that aims to find local maxima in each iteration. It starts with K as the input which is how many groups you want to see. Input k centroids in random locations in your space. Now, with the use of the Euclidean distance method calculate the distance between data points and centroids, and assign data point to the cluster which is close to it. Recalculate the cluster centres as a mean of data points attached to it. Repeat until no further changes occur.



4.2.2 **Hierarchical Clustering:-**

Hierarchical clustering is one of the algorithms of Clustering technique, in which similar data grouped in a cluster. It is an algorithm that builds the hierarchy of clusters. This algorithm starts with all the data points assigned to a bunch of their own. Then two nearest groups are merged into the same cluster. In the end, this algorithm terminates when there is only a single cluster left. Start by assign each data point to its bunch. Now find the closest pair of the group using Euclidean distance and merge them into the single cluster. Then calculate the distance between two nearest clusters and combine until all items clustered into a single cluster.



4.3 Dimensionality Reduction:

Dimensionality Reduction is the type of Unsupervised Learning, in which the dimensions of the data is reduced to remove the unwanted data from the input. This technique is used to remove the undesirable features of the data. It relates to the process of converting a set of data having large dimensions into data with carries same data and small sizes. These techniques used while solving machine learning problems to obtain better features.

Types of Dimensionality Reduction Algorithms:-

There are many Dimensionality reduction algorithms are present in machine learning, which applied for different dimensionality reduction applications. Some of the main dimensionality reduction algorithms are as follows-

4.3.1 **Principal Component Analysis:-**

Principal Component Analysis is one of the algorithms of Dimensionality Reduction, in this technique, it transformed into a new set of variables from old variables, which are the linear combination of real variables. Specific new set of variables are known as principal components. As a result of the transformation, the first primary component has the most significant possible variance, and each following element has the highest potential difference under the constraint that it is orthogonal to the above ingredients. Keeping only the first $m < n$ components reduces the data dimensionality while retaining most of the data information.

4.3.2 **Linear Discriminant Analysis:-**

The linear discriminant analysis is one of the algorithms of Dimensionality Reduction in which it also creates linear combinations of your original features. However, unlike PCA, LDA doesn’t maximise explained variance. Instead, it optimises the separability between classes. LDA can improve the predictive performance of the extracted features. Furthermore, LDA offers variations to tackle specific roadblocks.

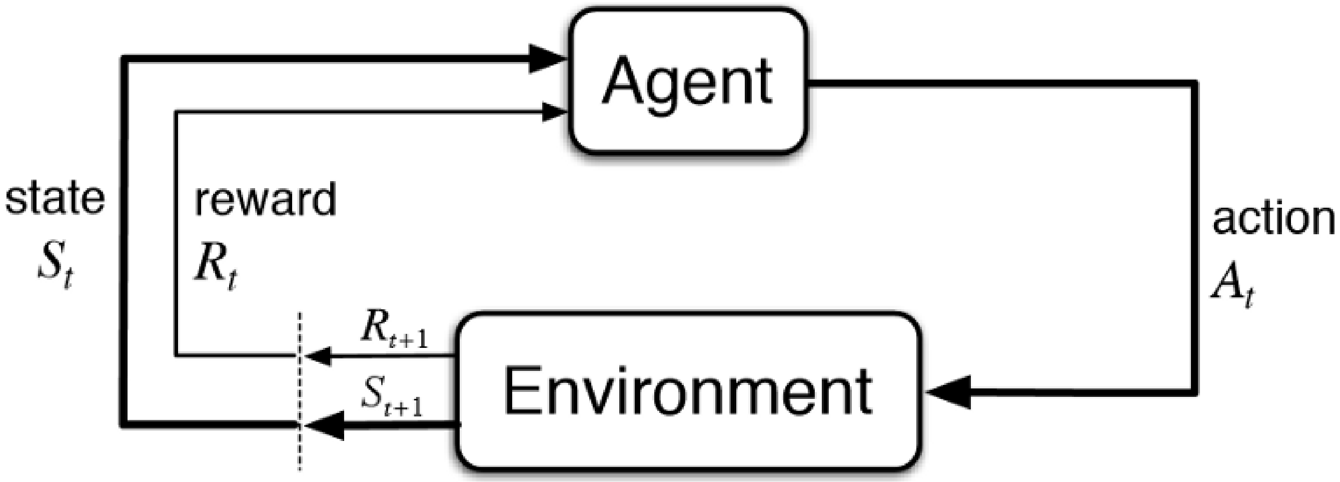
4.3.3 **Kernel Principal Component Analysis:-**

Kernel Principal Component Analysis is one of the algorithms of Dimensionality Reduction, and the variables which are transformed into variables of the new set, which are the non-linear combination of original variables means the nonlinear version of PCA, called as Kernel Principal Component Analysis (KPCA). It is capable of capturing part of the high order statistics, thus provides more information from the original dataset.

5. REINFORCEMENT LEARNING

5.1 Reinforcement Learning & its Types:

Reinforcement Learning is the third type of machine learning in which no raw data is given as input instead reinforcement learning algorithm have to figures out the situation on their own. The reinforcement learning frequently used for robotics, gaming, and navigation. With reinforcement learning, the algorithm discovers through trial and error which actions yield the most significant rewards. This type of training has three main components which are the agent which can describe as the learner or decision maker, the environment which described as everything the agent interacts with and actions which represented as what the agent can do.



The objective is for the agent to take actions that maximise the expected reward over a given measure of time. The agent will reach the goal much quicker by following a good policy. So the purpose of reinforcement learning is to learn the best plan.

Types of Reinforcement Learning Algorithms:-

There are many Reinforcement Learning algorithms are present in machine learning, which applied for different reinforcement learning applications. Some of the main algorithms are Q-Learning, SARSA, Deep Q-Network, Markov Decision Processes, and DDPG.

5.2 Q-Learning:

Q-learning is one of the algorithms of Reinforcement Learning, in which an agent attempts to learn the optimal strategy from its history of communication with the environment. A record of an agent is a sequence of state-action-rewards. Q-learning learns an optimal policy no matter which procedure the agent is following as long as there is no restriction on the plenty of times it tries an action in any state. Because it learns an optimal policy no matter which strategy it is carrying out, it is called an off-policy method.

5.3 SARSA [State Action Reward State Action]:

SARSA is one of the algorithms of Reinforcement Learning, in which it determines it refreshed to the action values. It’s a minor difference between the SARSA and Q-learning implementations, but it causes a profound effect. The SARSA method takes another parameter, action2, which is the action that was made by the agent from the second state. It allows the agent to find the future reward value explicitly. Next, that followed, rather than assuming that the optimal action will use and that the most significant reward.

5.4 Deep Q-Network:

Deep Q-Network is one of the algorithms of Reinforcement Learning, although Q-learning is a very robust algorithm, its main flaw is lack of generality. If you view Q-learning as renewing numbers in a two-dimensional array (Action Space * State Space), it, in fact, follows the dynamic programming. It indicates that for states that the Q-learning agent has not seen before, it has no clue which action to take. In other words, a Q-learning agent cannot estimate value for unseen states. To deal with this problem, DQN gets rid of the two-dimensional array by introducing Neural Network.

5.5 Markov Decision Processes:

Markov Decision Process is one of the algorithms of Reinforcement Learning, in which it contains *A set of possible world states S. *A set of Models. *A set of possible actions A. *A real-valued reward function R(s, a). *A policy the solution of Markov Decision Process. To achieve a goal, the Markov Decision Process is used it is a straightforward framing of the problem of learning from interaction. The agent was selecting actions and the environment responding to these actions, and the agent and the environment interact continually and presenting new situations to the agent.

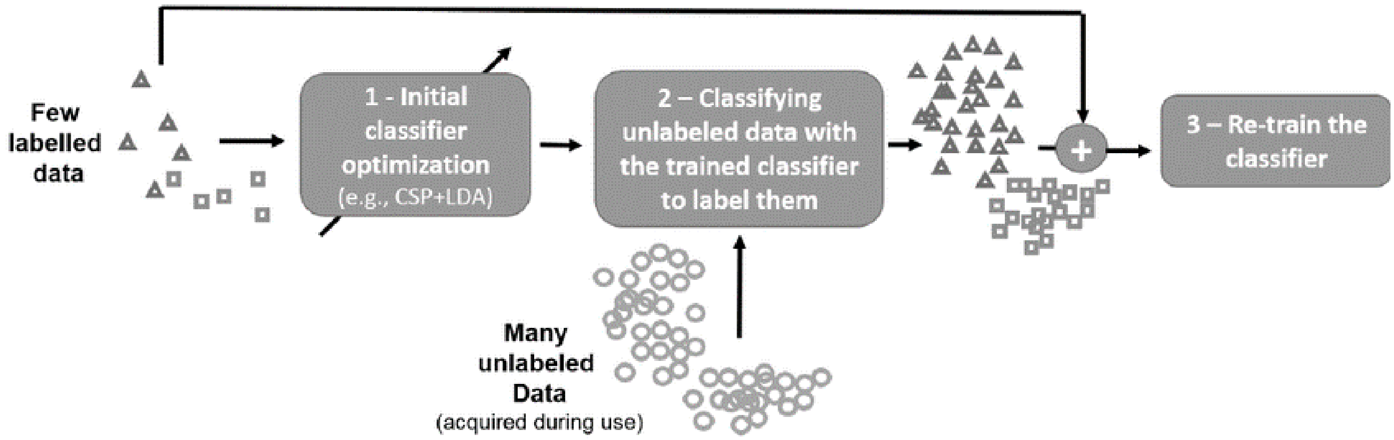
5.6 DDPG [Deep Deterministic Policy Gradient]:

Deep Deterministic Policy Gradient is one of the algorithms of Reinforcement Learning, in which it relies on the actor-critic design with two eponymous components, actor, and critic. An actor is utilised to tune the parameter $\boldsymbol{\theta}$ for the policy function, i.e. decide the best action for a specific state. The ideas of separate target network and experience replay are also borrowed from DQN. The seldom performs exploration for operations is another issue for DDPG. A solution for this is adding noise to the parameter space or the action space.

6. SEMI-SUPERVISED LEARNING

6.1 What is Semi-Supervised Learning? :

Semi-Supervised Learning is the fourth type of Machine Learning, in which both types of raw data used. Semi-supervised learning is a hybrid of supervised and unsupervised machine learning. The Semi-supervised learning used for the same purposes as supervised learning, where it employs both labelled and unlabeled data for training typically a small amount of labelled data with a significant amount of unlabelled data. This type of learning can use with methods such as classification, regression, and prediction.


Few labelled data
1 - Initial classifier optimization (e.g., CSP+LDA)
2 – Classifying unlabeled data with the trained classifier to label them
+
3 – Re-train the classifier
Many unlabeled Data (acquired during use)

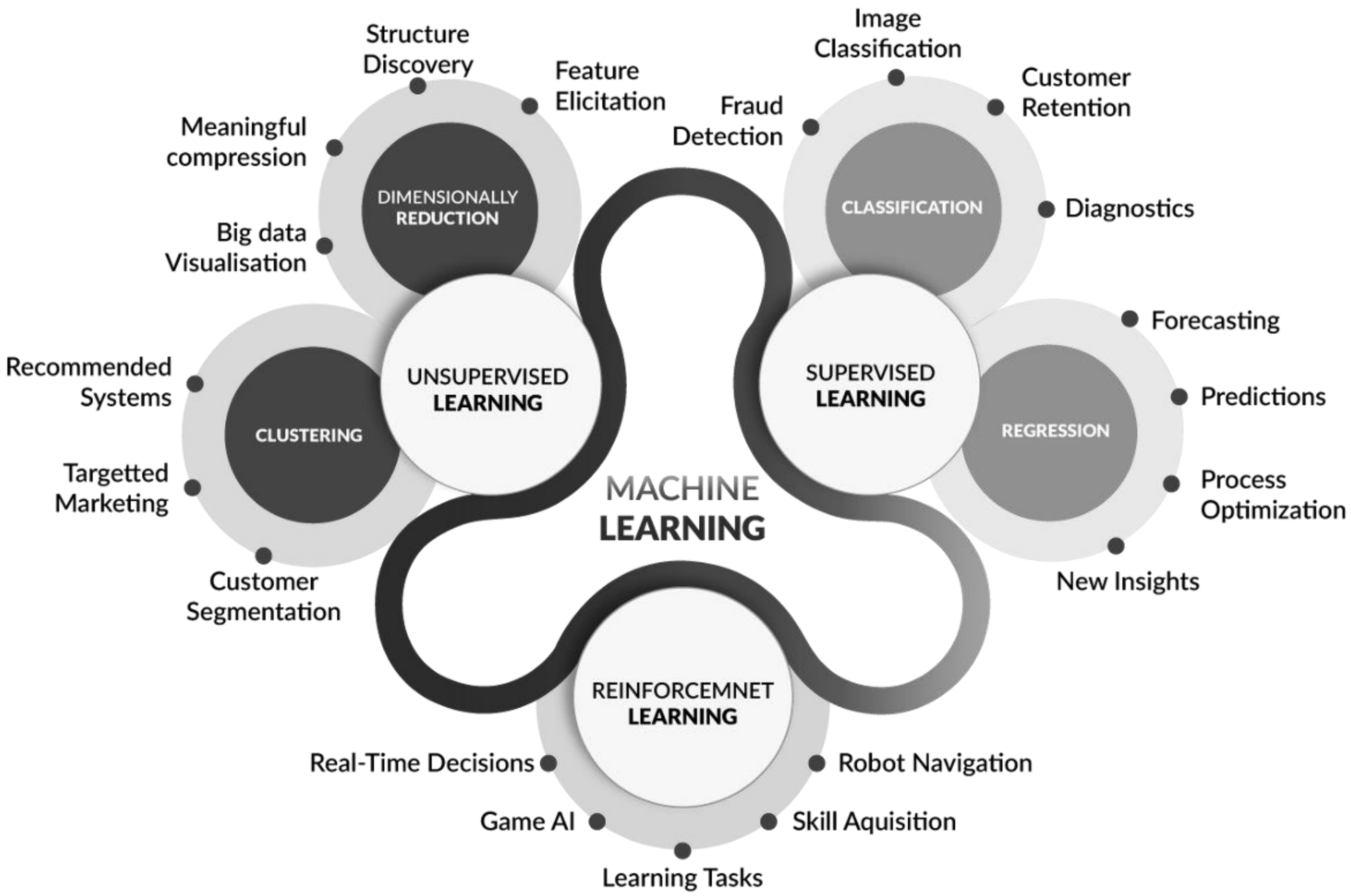
This technique is useful for a few reasons. First, the process of labelling massive amounts of data for supervised learning is often prohibitively time-consuming and expensive. What's more, too much labelling can impose human biases on the model. That means including lots of unlabelled data during the training process tends to improve the accuracy of the final model while reducing the time and cost spent building it.

7. MACHINE LEARNING – APPLICATIONS

7.1 Some Applications of ML:

Machine learning is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that which makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect. We probably use a learning algorithm dozens of time without even knowing it. Applications of Machine Learning include:

- **Web Search Engine**: One of the reasons why search engines like Google, Bing etc. work so well is because the system has learnt how to rank pages through a complex learning algorithm.
- **Photo tagging Applications**: Be it Facebook or any other photo tagging application, the ability to tag friends makes it even more happening. It is all possible because of a face recognition algorithm that runs behind the application.
- **Spam Detector**: Our mail agent like Gmail or Hotmail does a lot of hard work for us in classifying the mails and moving the spam mails to spam folder. This is again achieved by a spam classifier running in the back end of mail application.



7.2 Examples:

Today, companies are using Machine Learning to improve business decisions, increase productivity, detect disease, forecast weather, and do many more things. With the exponential growth of technology, we not only need better tools to understand the data we currently have, but we also need to prepare ourselves for the data we will have. To achieve this goal we need to build intelligent machines. We can write a program to do simple things. But for most of times Hardwiring Intelligence in it is difficult. Best way to do it is to have some way for machines to learn things themselves. A mechanism for learning – if a machine can learn from input then it does the hard work for us. This is where Machine Learning comes in action. Some examples of machine learning are:

Database Mining for growth of automation: Typical applications include Web-click data for better UX (User eXperience), Medical records for better automation in healthcare, biological data and many more.

Applications that cannot be programmed: There are some tasks that cannot be programmed as the computers we use are not modelled that way. Examples include Autonomous Driving, Recognition tasks from unordered data (Face Recognition/ Handwriting Recognition), Natural language Processing, computer Vision etc.

Understanding Human Learning: This is the closest we have understood and mimicked the human brain. It is the start of a new revolution, The real AI. Now, After a brief insight lets come to a more formal definition of Machine Learning

Arthur Samuel (1959): "Machine Learning is a field of study that gives computers, the ability to learn without explicitly being programmed."Samuel wrote a Checker playing program which could learn over time. At first it could be easily won. But over time, it learnt all the board position that would eventually lead him to victory or loss and thus became a better chess player than Samuel itself. This was one of the most early attempts of defining Machine Learning and is somewhat less formal.

Tom Michel (1999): "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." This is a more formal and mathematical definition. For the previous Chess program:

- E is number of games.
- T is playing chess against computer.
- P is win/loss by computer.

8. INTRODUCTION TO FLASK

8.1 What is flask? :

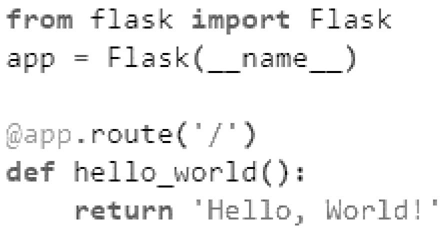
Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, and wiki or go as big as a web-based calendar application or a commercial website.

Flask is part of the categories of the micro-framework. Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins. In the case of Flask, its dependencies are:

- Werkzeug a WSGI utility library
- jinja2 which is its template engine

8.2 A Minimal Application:

A minimal Flask application looks something like this:



```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

So what did that code do?

First we imported the Flask class. An instance of this class will be our WSGI application.

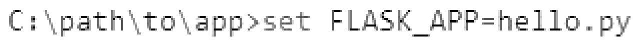
Next we create an instance of this class. The first argument is the name of the application’s module or package. If you are using a single module (as in this example), you should use __name__ because depending on if it’s started as application or imported as module the name will be different ('__main__' versus the actual import name). This is needed so that Flask knows where to look for templates, static files, and so on.

We then use the route() decorator to tell Flask what URL should trigger our function.

The function is given a name which is also used to generate URLs for that particular function, and returns the message we want to display in the user’s browser.

Just save it as hello.py or something similar. Make sure to not call your application flask.py because this would conflict with Flask itself.

To run the application you can either use the flask command or python’s -m switch with Flask. On Command Prompt:



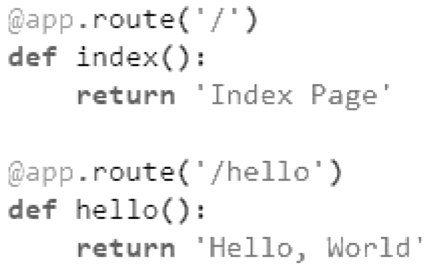
```
C:\path\to\app>set FLASK_APP=hello.py
```

Now head over to http://127.0.0.1:5000/, and you should see your hello world greeting.

8.3 Routing:

Modern web applications use meaningful URLs to help users. Users are more likely to like a page and come back if the page uses a meaningful URL they can remember and use to directly visit a page.

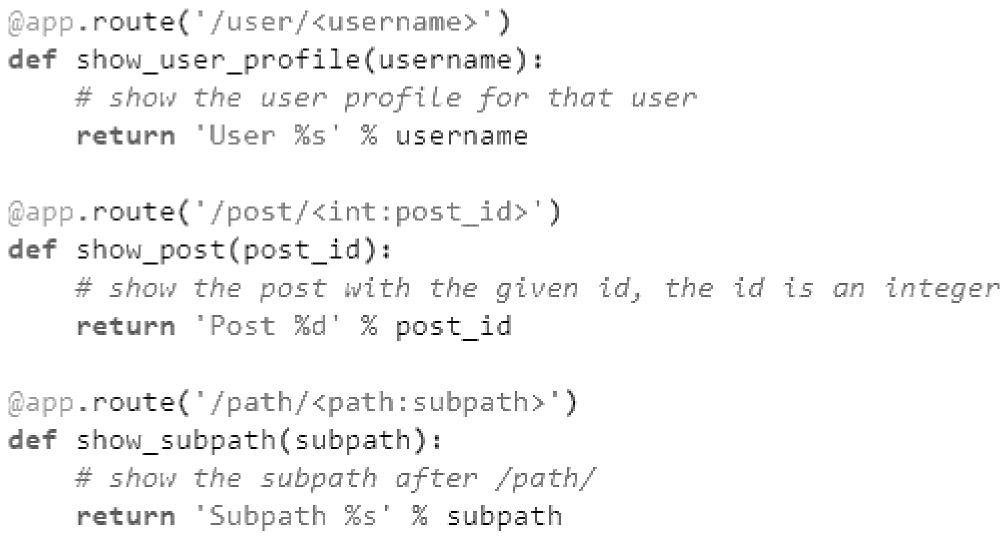
Use the route() decorator to bind a function to a URL.



```
@app.route('/')
def index():
    return 'Index Page'

@app.route('/hello')
def hello():
    return 'Hello, World'
```

8.3.1 **Variable Rules:**

You can add variable sections to a URL by marking sections with <variable_name>. Your function then receives the <variable_name> as a keyword argument. Optionally, you can use a converter to specify the type of the argument like <converter:variable_name>.



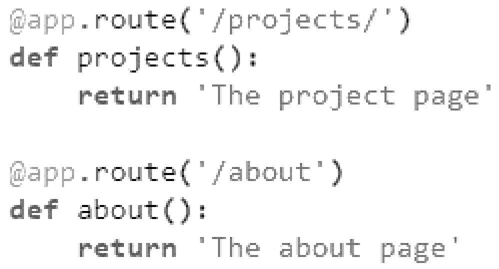
```
@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return 'User %s' % username

@app.route('/post/<int:post_id>')
def show_post(post_id):
    # show the post with the given id, the id is an integer
    return 'Post %d' % post_id

@app.route('/path/<path:subpath>')
def show_subpath(subpath):
    # show the subpath after /path/
    return 'Subpath %s' % subpath
```

Converter types:

string	(default) accepts any text without a slash
int	accepts positive integers
float	accepts positive floating point values
path	like string but also accepts slashes
uuid	accepts UUID strings

8.3.2 **Unique URLs / Redirection Behaviour:**

The following two rules differ in their use of a trailing slash.



```
@app.route('/projects/')
def projects():
    return 'The project page'

@app.route('/about')
def about():
    return 'The about page'
```

The canonical URL for the projects endpoint has a trailing slash. It’s similar to a folder in a file system. If you access the URL without a trailing slash, Flask redirects you to the canonical URL with the trailing slash.

The canonical URL for the about endpoint does not have a trailing slash. It’s similar to the pathname of a file. Accessing the URL with a trailing slash produces a 404 “Not Found” error. This helps keep URLs unique for these resources, which helps search engines avoid indexing the same page twice.

8.3.3 **URL Building:**

To build a URL to a specific function, use the url_for() function. It accepts the name of the function as its first argument and any number of keyword arguments, each corresponding to a variable part of the URL rule. Unknown variable parts are appended to the URL as query parameters.

Why would you want to build URLs using the URL reversing function url_for() instead of hard-coding them into your templates?

Reversing is often more descriptive than hard-coding the URLs.

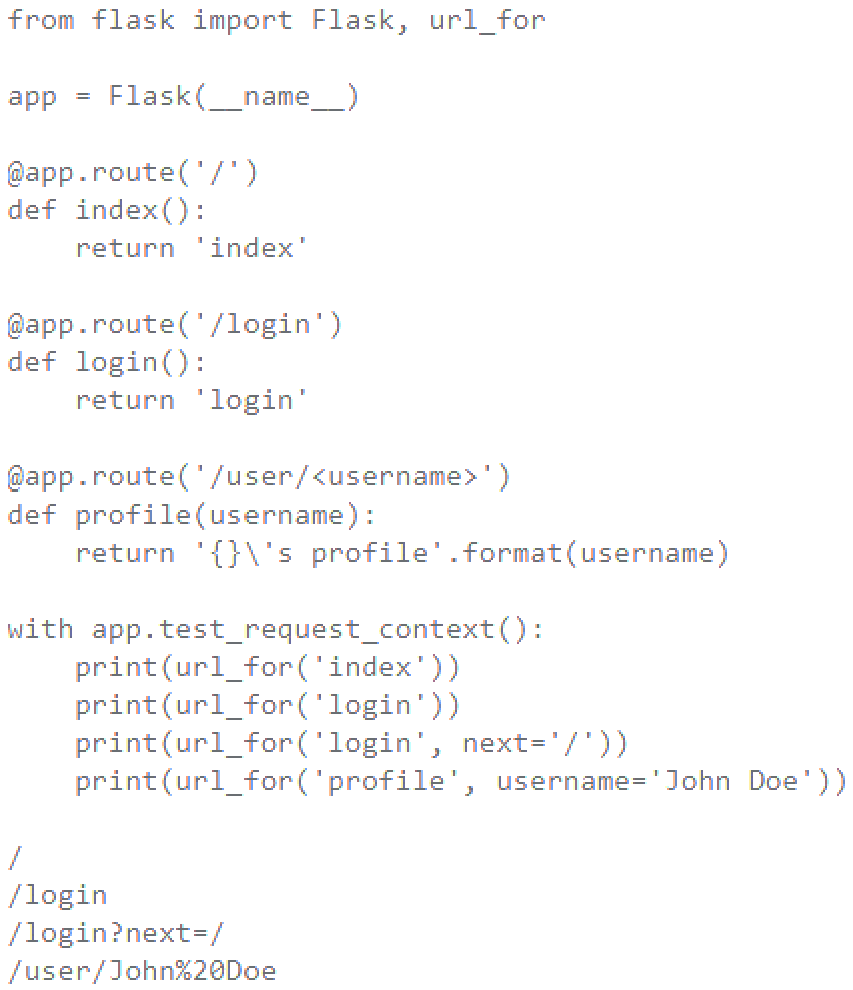
You can change your URLs in one go instead of needing to remember to manually change hard-coded URLs.

URL building handles escaping of special characters and Unicode data transparently. The generated paths are always absolute, avoiding unexpected behavior of relative paths in browsers.

If your application is placed outside the URL root, for example, in

/myapplication instead of /, url_for() properly handles that for you.

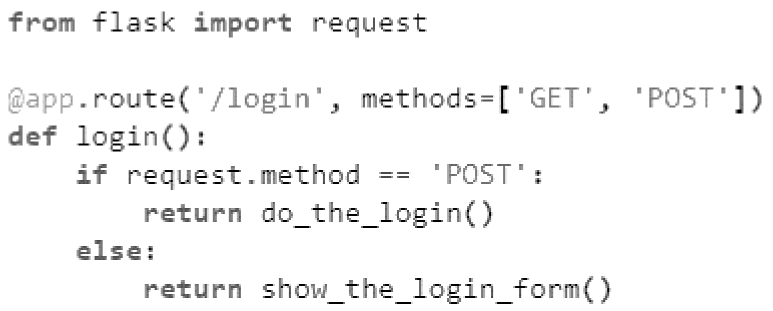
For example, here we use the test_request_context() method to try out url_for(). test_request_context() tells Flask to behave as though it’s handling a request even while we use a Python shell.



```
from flask import Flask, url_for

app = Flask(__name__)

@app.route('/')
def index():
    return 'index'

@app.route('/login')
def login():
    return 'login'

@app.route('/user/<username>')
def profile(username):
    return '{}\'s profile'.format(username)

with app.test_request_context():
    print(url_for('index'))
    print(url_for('login'))
    print(url_for('login', next='/'))
    print(url_for('profile', username='John Doe'))

/
/login
/login?next=/
/user/John%20Doe
```

8.3.4 **HTTP Methods:**

Web applications use different HTTP methods when accessing URLs. You should familiarize yourself with the HTTP methods as you work with Flask. By default, a route only answers to GET requests. You can use the methods argument of the route() decorator to handle different HTTP methods.



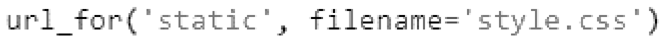
```
from flask import request

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return do_the_login()
    else:
        return show_the_login_form()
```

If GET is present, Flask automatically adds support for the HEAD method and handles HEAD requests according to the HTTP RFC. Likewise, OPTIONS is automatically implemented for you.

8.4 Static Files:

Dynamic web applications also need static files. That's usually where the CSS and JavaScript files are coming from. Ideally your web server is configured to serve them for you, but during development Flask can do that as well. Just create a folder called static in your package or next to your module and it will be available at /static on the application.

To generate URLs for static files, use the special 'static' endpoint name:

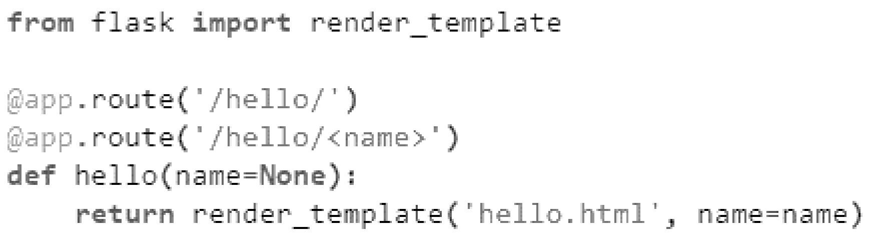


```
url_for('static', filename='style.css')
```

The file has to be stored on the file system as static/style.css.

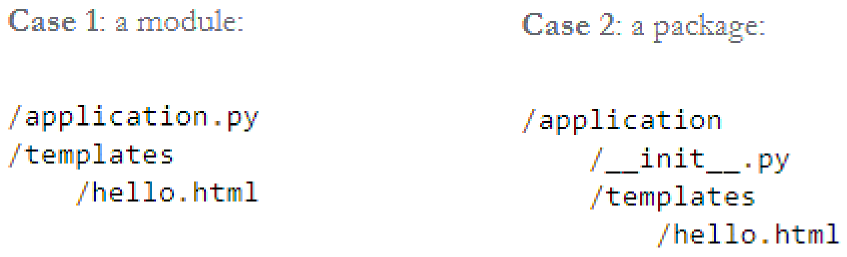
8.5 Rendering Templates:

Generating HTML from within Python is not fun, and actually pretty cumbersome because you have to do the HTML escaping on your own to keep the application secure. Because of that Flask configures the Jinja2 template engine for you automatically. To render a template you can use the render_template() method. All you have to do is provide the name of the template and the variables you want to pass to the template engine as keyword arguments. Here's a simple example of how to render a template:



```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', name=name)
```

Flask will look for templates in the templates folder. So if your application is a module, this folder is next to that module, if it's a package it's actually inside your package:



Case 1: a module:

```
/application.py
/templates
    /hello.html
```

Case 2: a package:

```
/application
    /__init__.py
    /templates
        /hello.html
```

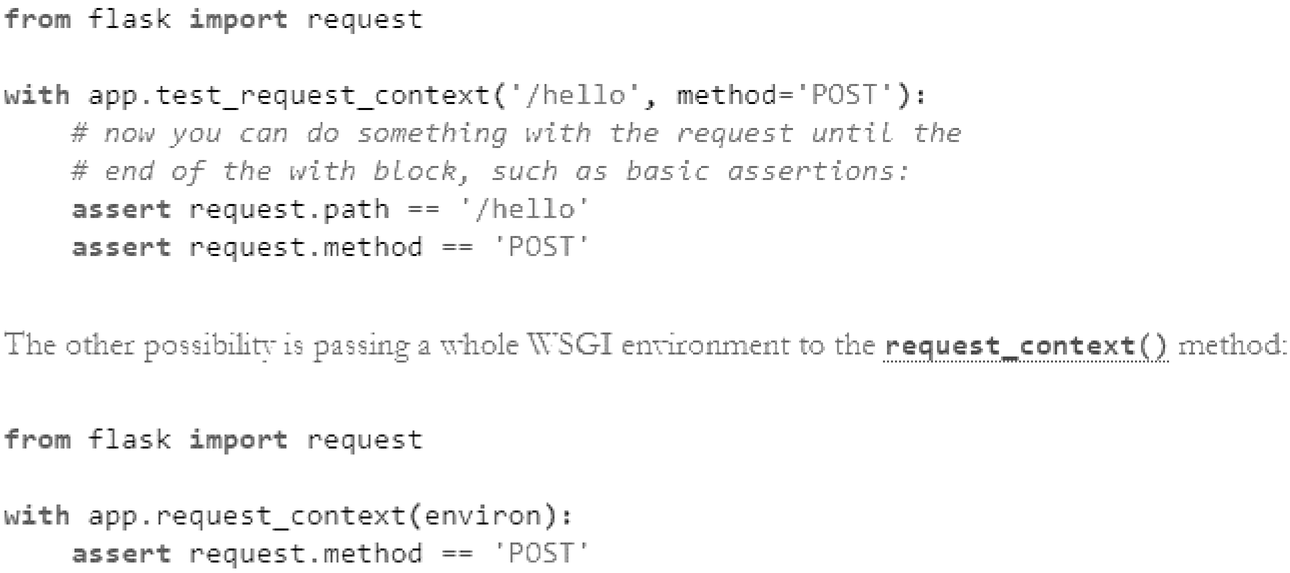
8.6 Accessing Request Data:

For web applications it's crucial to react to the data a client sends to the server. In Flask this information is provided by the global request object. If you have some experience with Python you might be wondering how that object can be global and how Flask manages to still be threadsafe. The answer is context locals:

8.6.1 **Context Locals:**

Certain objects in Flask are global objects, but not of the usual kind. These objects are actually proxies to objects that are local to a specific context. What a mouthful. But that is actually quite easy to understand. Imagine the context being the handling thread. A request comes in and the web server decides to spawn a new thread (or something else, the underlying object is capable of dealing with concurrency systems other than threads). When Flask starts its internal request handling it figures out that the current thread is the active context and binds the current application and the WSGI environments to that context (thread). It does that in an intelligent way so that one application can invoke another application without breaking.

So what does this mean to you? Basically you can completely ignore that this is the case unless you are doing something like unit testing. You will notice that code which depends on a request object will suddenly break because there is no request object. The solution is creating a request object yourself and binding it to the context. The easiest solution for unit testing is to use the test_request_context() context manager. In combination with the with statement it will bind a test request so that you can interact with it. Here is an example:



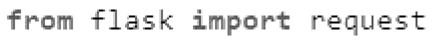
```
from flask import request

with app.test_request_context('/hello', method='POST'):
    # now you can do something with the request until the
    # end of the with block, such as basic assertions:
    assert request.path == '/hello'
    assert request.method == 'POST'
```

The other possibility is passing a whole WSGI environment to the `request_context()` method:

```
from flask import request

with app.request_context(environ):
    assert request.method == 'POST'
```

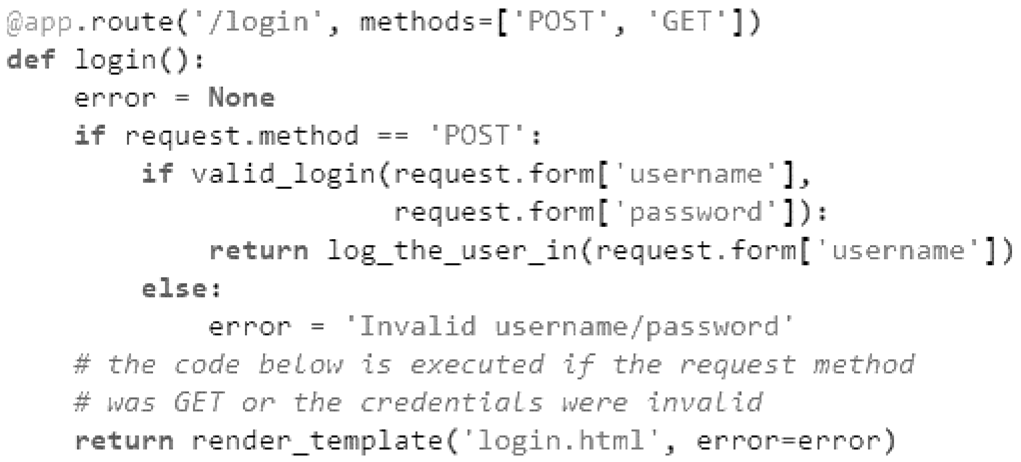
8.6.2 **The Request Object:**

The request object is documented in the API section and we will not cover it here in detail (see Request). Here is a broad overview of some of the most common operations. First of all you have to import it from the flask module:



```
from flask import request
```

The current request method is available by using the method attribute. To access form data (data transmitted in a POST or PUT request) you can use the form attribute. Here is a full example of the two attributes mentioned above:

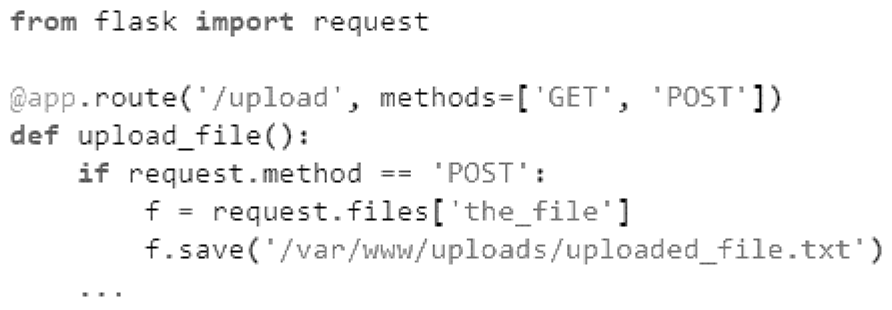


```
@app.route('/login', methods=['POST', 'GET'])
def login():
    error = None
    if request.method == 'POST':
        if valid_login(request.form['username'],
                       request.form['password']):
            return log_the_user_in(request.form['username'])
        else:
            error = 'Invalid username/password'
    # the code below is executed if the request method
    # was GET or the credentials were invalid
    return render_template('login.html', error=error)
```

8.6.3 **File Uploads**:

You can handle uploaded files with Flask easily. Just make sure not to forget to set the enctype="multipart/form-data" attribute on your HTML form, otherwise the browser will not transmit your files at all.

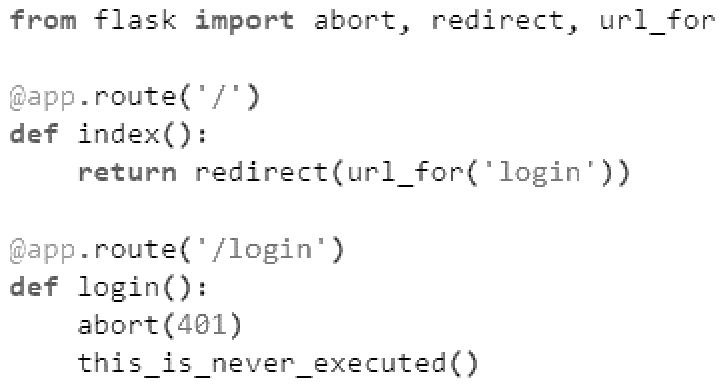
Uploaded files are stored in memory or at a temporary location on the filesystem. You can access those files by looking at the files attribute on the request object. Each uploaded file is stored in that dictionary. It behaves just like a standard Python file object, but it also has a save() method that allows you to store that file on the filesystem of the server. Here is a simple example showing how that works:



```
from flask import request

@app.route('/upload', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        f = request.files['the_file']
        f.save('/var/www/uploads/uploaded_file.txt')
    ...
```

8.7 Redirects and Errors:

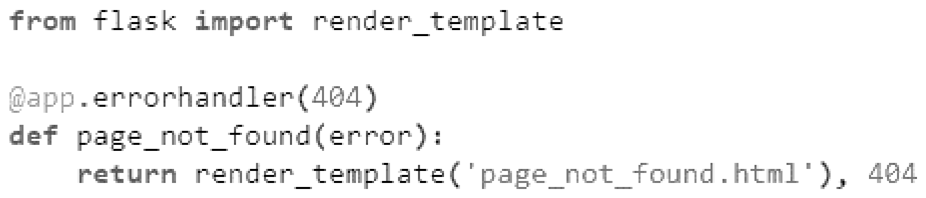
To redirect a user to another endpoint, use the redirect() function; to abort a request early with an error code, use the abort() function:



```
from flask import abort, redirect, url_for

@app.route('/')
def index():
    return redirect(url_for('login'))

@app.route('/login')
def login():
    abort(401)
    this_is_never_executed()
```

This is a rather pointless example because a user will be redirected from the index to a page they cannot access (401 means access denied) but it shows how that works.

By default a black and white error page is shown for each error code. If you want to customize the error page, you can use the errorhandler() decorator:



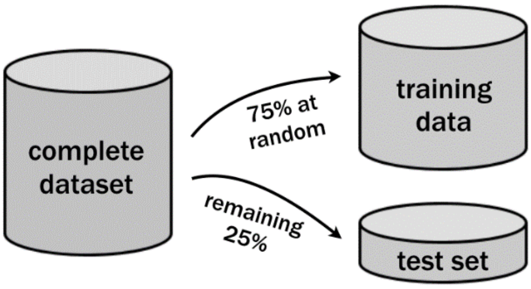
```
from flask import render_template

@app.errorhandler(404)
def page_not_found(error):
    return render_template('page_not_found.html'), 404
```

Note the 404 after the render_template() call. This tells Flask that the status code of that page should be 404 which means not found. By default 200 is assumed which translates to: all went well

9. UNDERSTANDING THE PROJECT

9.1 The Algorithm:

1. Reading & Dividing the Dataset:



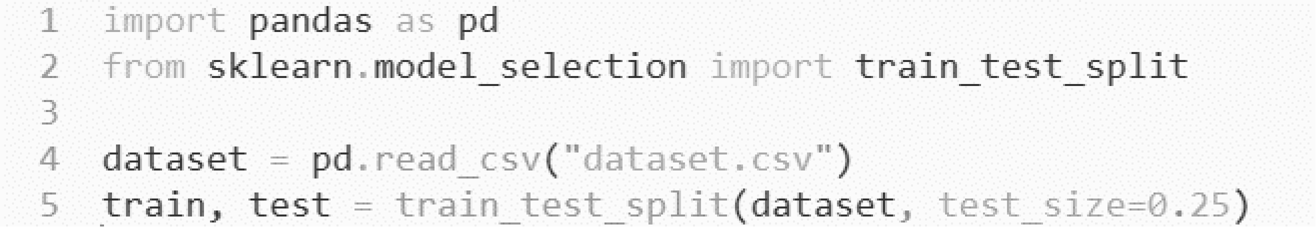
We divided the dataset into 2 parts:

A. Training Data
B. Testing Data

Training Data: The part of data we use to train our model. This is the data which our model actually sees (both input and output) and learn from.

Testing Data: Once our model is completely trained, testing data provides the unbiased evaluation. When we feed in the inputs of testing data, our model will predict some values (without seeing actual output). After prediction, we evaluate our model by comparing it with actual output present in the testing data. This is how we evaluate and see how much our model has learned from the experiences feed in as training data, set at the time of training.

>>> Code for Step 1:



```
import pandas as pd
from sklearn.model_selection import train_test_split

dataset = pd.read_csv("dataset.csv")
train, test = train_test_split(dataset, test_size=0.25)
```

2. Converting the text in Dataset to numbers:

Since we cannot work with text directly when using machine learning algorithms, we need to convert the text to numbers. The text must be parsed to remove words, called tokenization. Then the words need to be encoded as integers or floating point values for use as input to a machine learning algorithm, called feature extraction (or vectorization). The scikit-learn library offers easy-to-use tools to perform both tokenization and vectorization of the text data. It provides 3 different schemes for this that we can use: a)CountVectorizer, b)TfidfVectorizer, c)HashingVectorizer.

We have used Tfidf Vectorizer for better accuracy.

TFIDF Vectorizer: TFIDF is an acronym than stands for "Term Frequency – Inverse Document Frequency" which are the components of the resulting scores assigned to each word.

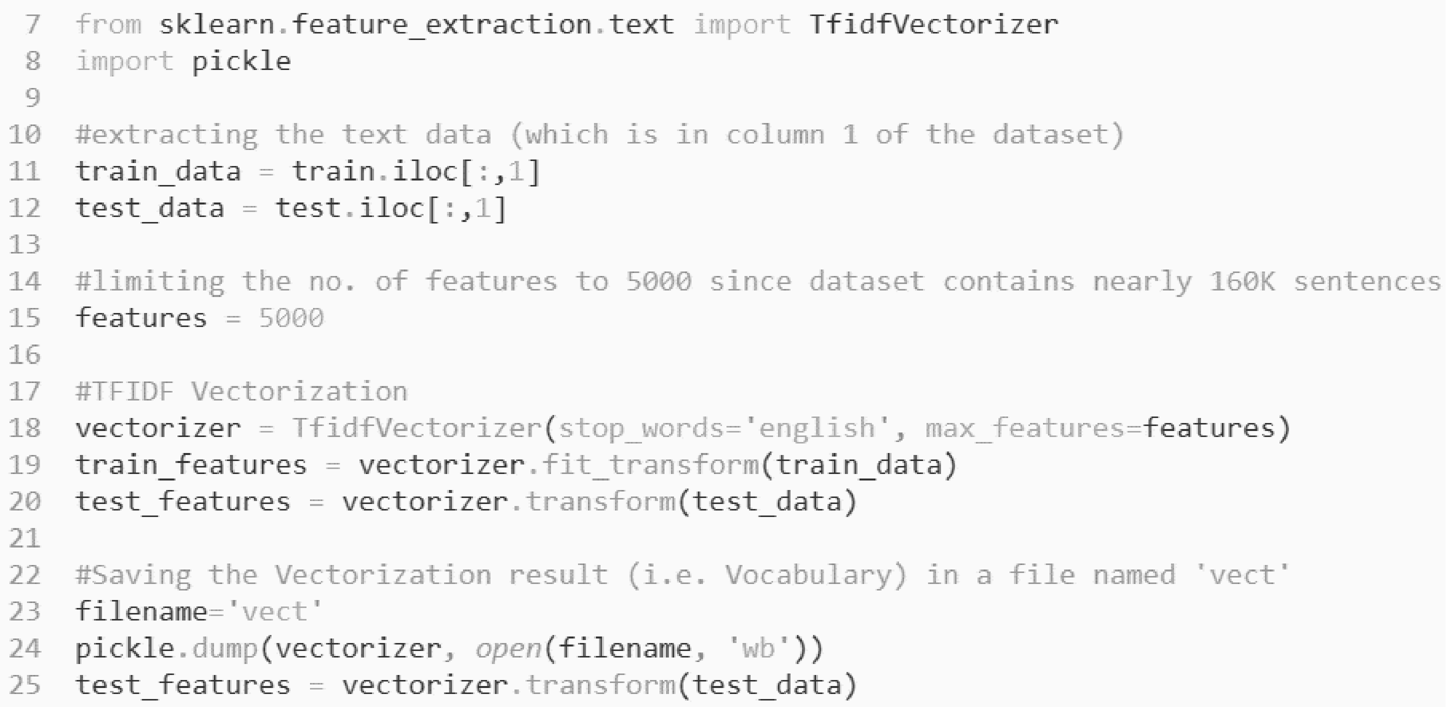
- Term Frequency: This summarizes how often a given word appears within a document.
- Inverse Document Frequency: This downscales words that appear a lot across documents.

Without going into the math, TF-IDF are word frequency scores that try to highlight words that are more interesting, e.g. frequent in a document but not across documents.

The TfidfVectorizer will tokenize documents, learn the vocabulary and inverse document frequency weightings, and allow you to encode new documents. It is used as follows:

1. *Create an instance of the TfidfVectorizer class.*
2. *Call the fit() function in order to learn a vocabulary from one or more documents.*
3. *Call the transform() function on one or more documents as needed to encode each as a vector.*

>>> Code for Step 2:

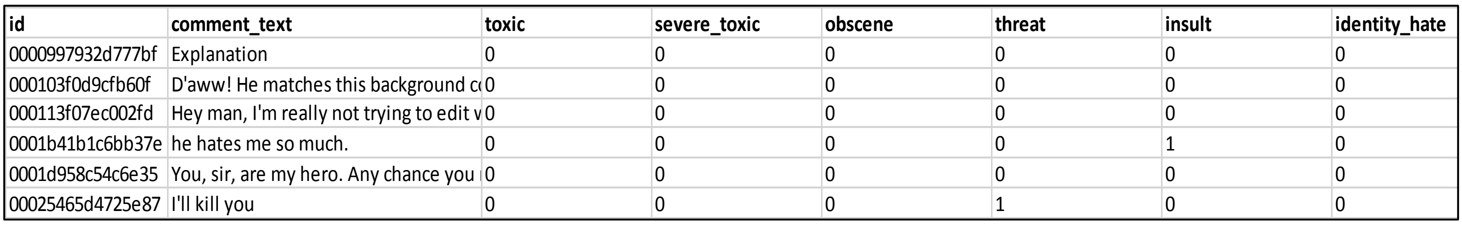


```
from sklearn.feature_extraction.text import TfidfVectorizer
import pickle

#extracting the text data (which is in column 1 of the dataset)
train_data = train.iloc[:,1]
test_data = test.iloc[:,1]

#limiting the no. of features to 5000 since dataset contains nearly 160K sentences
features = 5000

#TFIDF Vectorization
vectorizer = TfidfVectorizer(stop_words='english', max_features=features)
train_features = vectorizer.fit_transform(train_data)
test_features = vectorizer.transform(test_data)

#Saving the Vectorization result (i.e. Vocabulary) in a file named 'vect'
filename='vect'
pickle.dump(vectorizer, open(filename, 'wb'))
test_features = vectorizer.transform(test_data)
```

3. Training, Testing & Creation of Models:

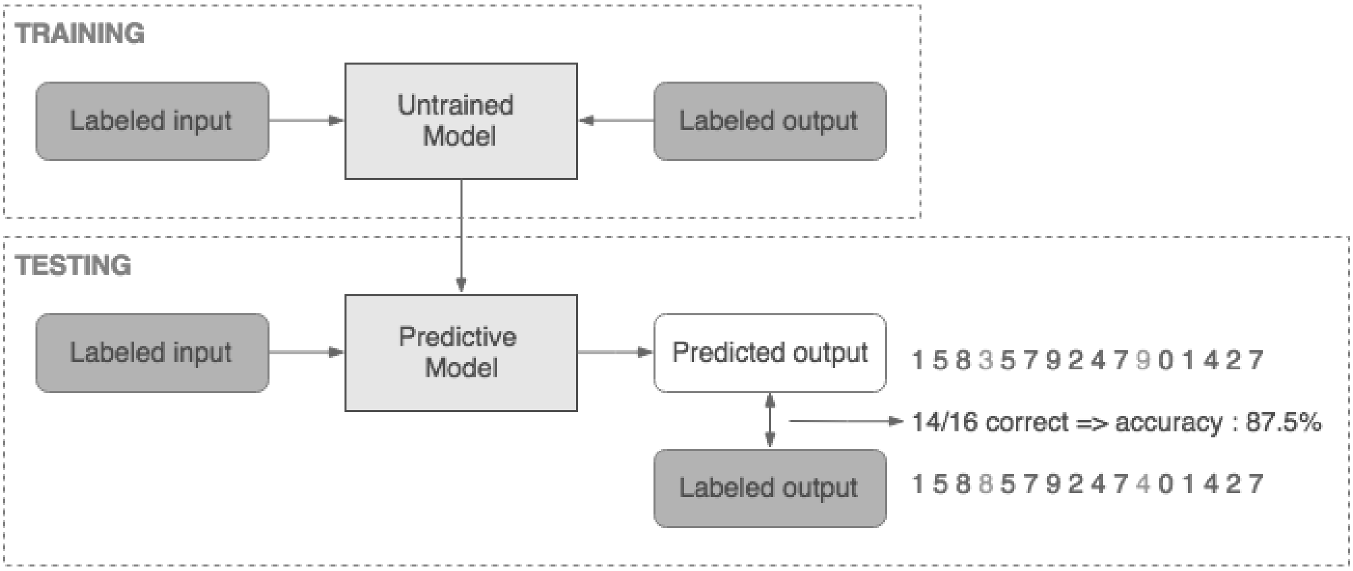
Since in the dataset we used, comments are classified into 6 different categories (as shown below), we need to train and create 6 different models with the help of **Logistic Regression** **(explained in section 3.3.1)**. This is done with the using LogisticRegression class of sklearn.linear_model library

Fig: Format of dataset used.

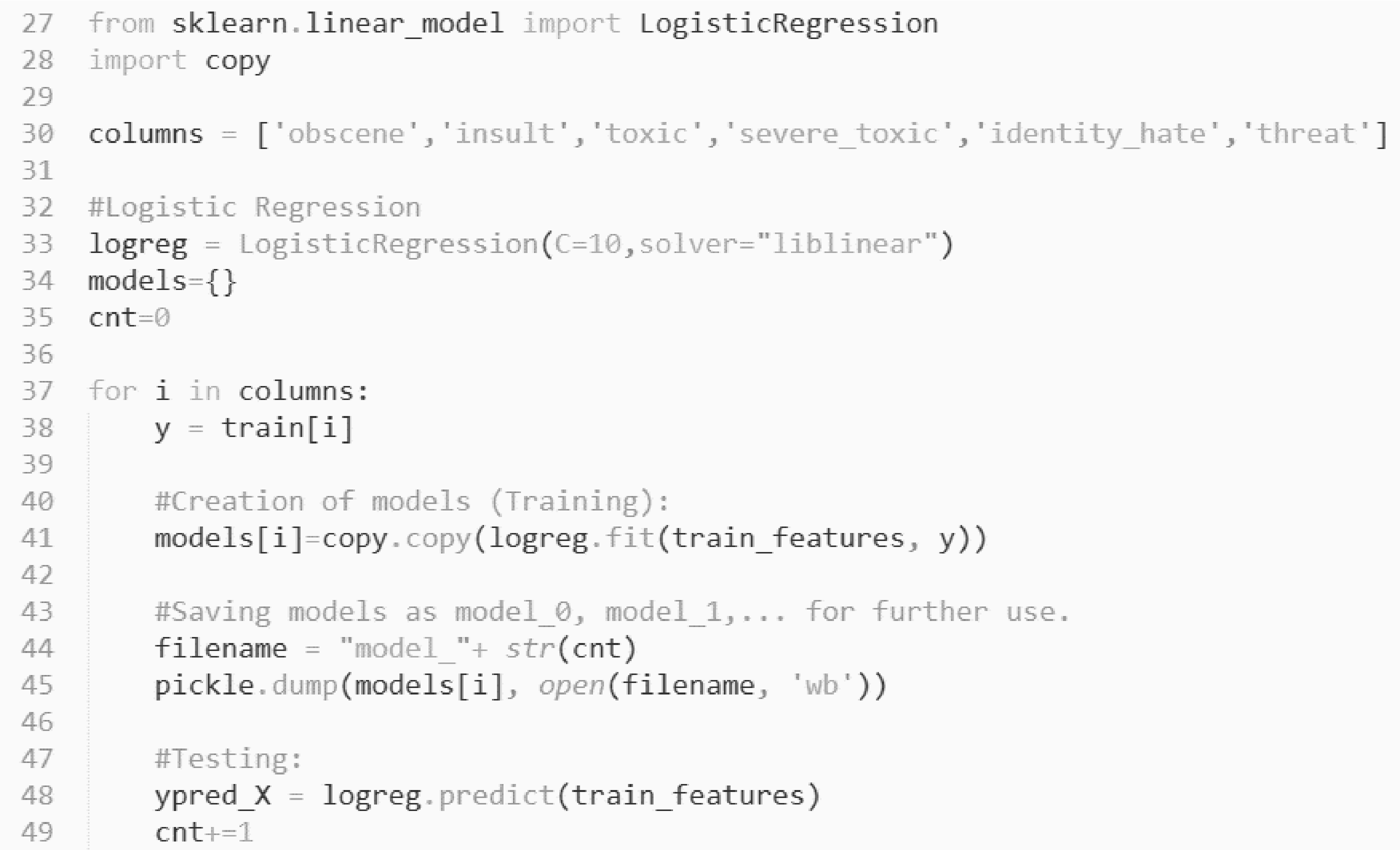


id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0000997932d777bf	Explanation	0	0	0	0	0	0
000103f0d9cfb60f	D'aww! He matches this background c	0	0	0	0	0	0
000113f07ec002fd	Hey man, I'm really not trying to edit v	0	0	0	0	0	0
0001b41b1c6bb37e	he hates me so much.	0	0	0	0	1	0
0001d958c54c6e35	You, sir, are my hero. Any chance you	0	0	0	0	0	0
00025465d4725e87	I'll kill you	0	0	0	1	0	0

Fig: Process of Training & Testing of Model.

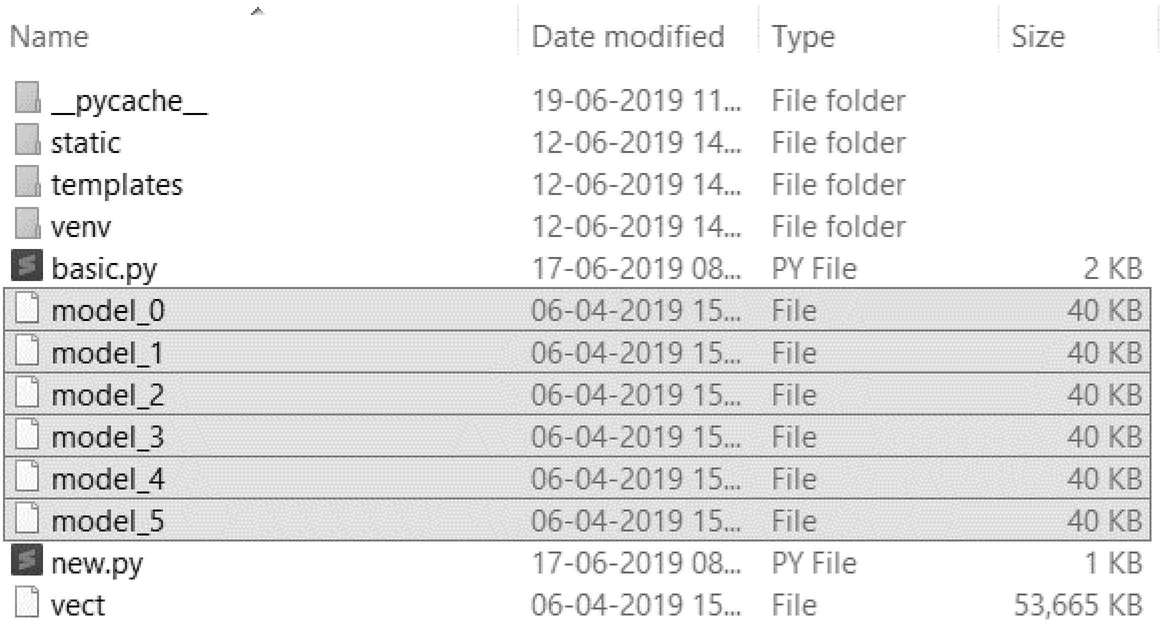


>>> Code for Step 3:



```python
from sklearn.linear_model import LogisticRegression
import copy

columns = ['obscene','insult','toxic','severe_toxic','identity_hate','threat']

#Logistic Regression
logreg = LogisticRegression(C=10,solver="liblinear")
models={}
cnt=0

for i in columns:
    y = train[i]

    #Creation of models (Training):
    models[i]=copy.copy(logreg.fit(train_features, y))

    #Saving models as model_0, model_1,... for further use.
    filename = "model_"+ str(cnt)
    pickle.dump(models[i], open(filename, 'wb'))

    #Testing:
    ypred_X = logreg.predict(train_features)
    cnt+=1
```

The above step will create 6 files in the project directory (corresponding to 6 different models created):



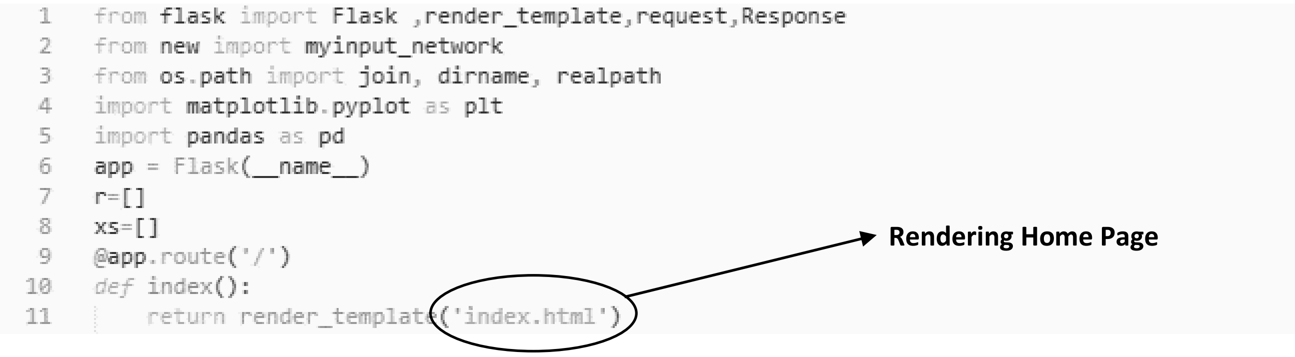
Name	Date modified	Type	Size
__pycache__	19-06-2019 11...	File folder	
static	12-06-2019 14...	File folder	
templates	12-06-2019 14...	File folder	
venv	12-06-2019 14...	File folder	
basic.py	17-06-2019 08...	PY File	2 KB
model_0	06-04-2019 15...	File	40 KB
model_1	06-04-2019 15...	File	40 KB
model_2	06-04-2019 15...	File	40 KB
model_3	06-04-2019 15...	File	40 KB
model_4	06-04-2019 15...	File	40 KB
model_5	06-04-2019 15...	File	40 KB
new.py	17-06-2019 08...	PY File	1 KB
vect	06-04-2019 15...	File	53,665 KB

These models are further used in the web app we developed, to predict whether the entered text contains elements of Obscenity, Insult, Toxicity, Spitefulness, Identity Hate, Threat or not.

9.2 Developing the Web Application:

Web Application for the implementation this project is made with the help of Flask. Various components of Flask used in this project are described in detail in Chapter 8.

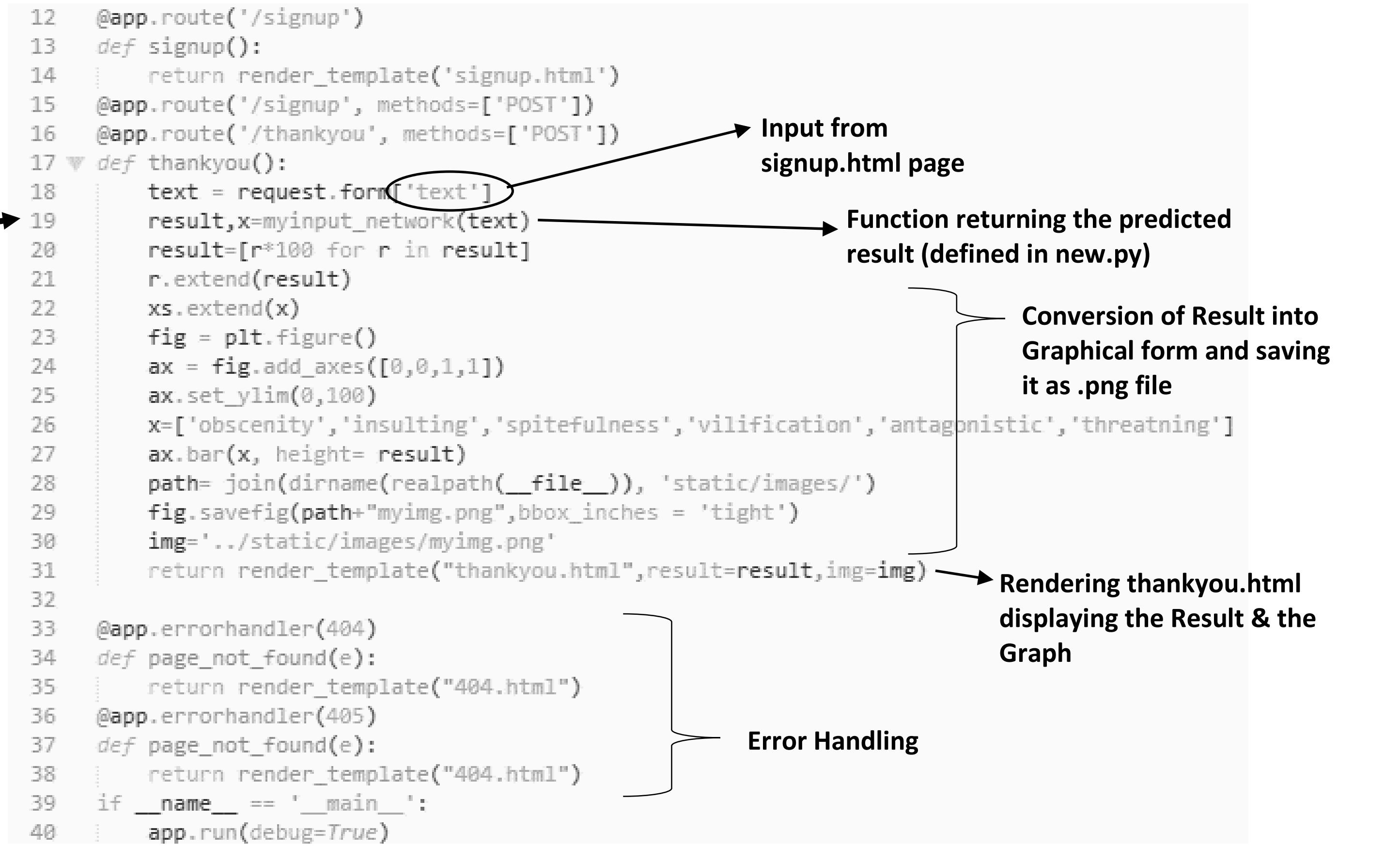
>>>Code: a) basic.py:



```
from flask import Flask ,render_template,request,Response
from new import myinput_network
from os.path import join, dirname, realpath
import matplotlib.pyplot as plt
import pandas as pd
app = Flask(__name__)
r=[]
xs=[]
@app.route('/')
def index():
    return render_template('index.html')
```

Rendering Home Page

cntd.



```python
@app.route('/signup')
def signup():
    return render_template('signup.html')
@app.route('/signup', methods=['POST'])
@app.route('/thankyou', methods=['POST'])
def thankyou():
    text = request.form['text']
    result,x=myinput_network(text)
    result=[r*100 for r in result]
    r.extend(result)
    xs.extend(x)
    fig = plt.figure()
    ax = fig.add_axes([0,0,1,1])
    ax.set_ylim(0,100)
    x=['obscenity','insulting','spitefulness','vilification','antagonistic','threatning']
    ax.bar(x, height= result)
    path= join(dirname(realpath(__file__)), 'static/images/')
    fig.savefig(path+"myimg.png",bbox_inches = 'tight')
    img='../static/images/myimg.png'
    return render_template("thankyou.html",result=result,img=img)

@app.errorhandler(404)
def page_not_found(e):
    return render_template("404.html")
@app.errorhandler(405)
def page_not_found(e):
    return render_template("404.html")
if __name__ == '__main__':
    app.run(debug=True)
```

Input from signup.html page

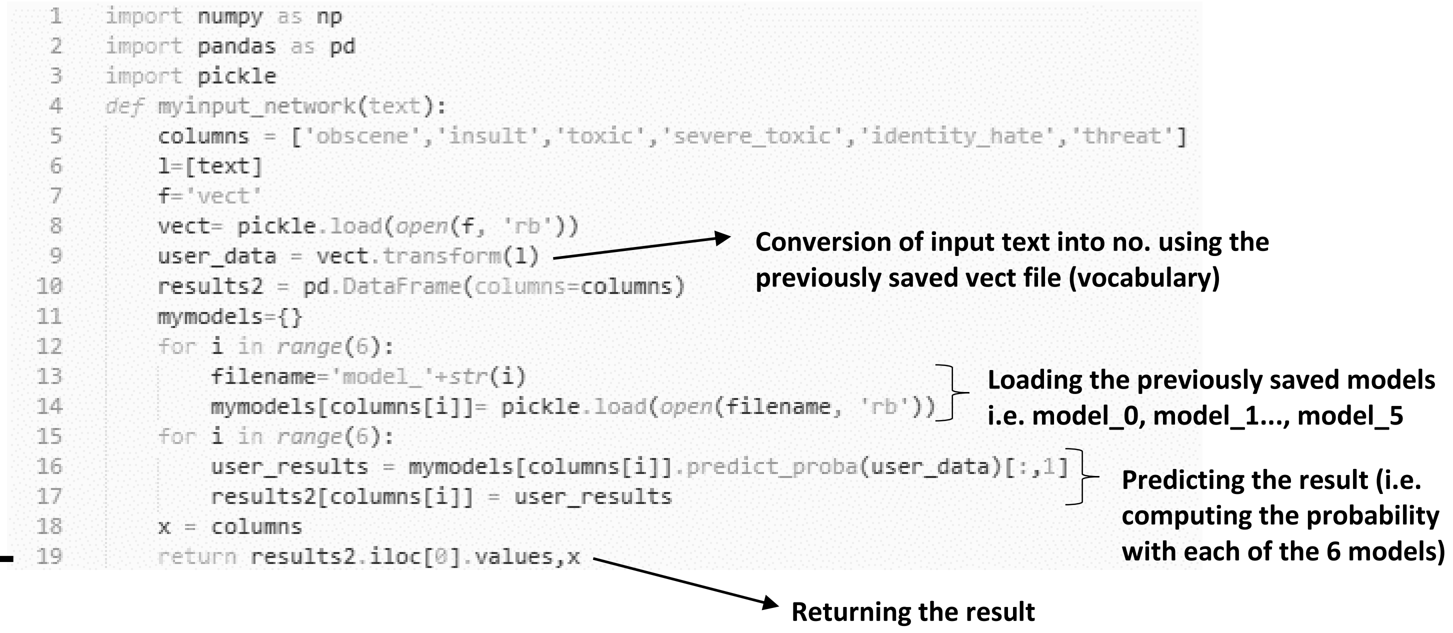
Function returning the predicted result (defined in new.py)

Conversion of Result into Graphical form and saving it as .png file

Rendering thankyou.html displaying the Result & the Graph

Error Handling

b) new.py:



```python
import numpy as np
import pandas as pd
import pickle
def myinput_network(text):
    columns = ['obscene','insult','toxic','severe_toxic','identity_hate','threat']
    l=[text]
    f='vect'
    vect= pickle.load(open(f, 'rb'))
    user_data = vect.transform(l)
    results2 = pd.DataFrame(columns=columns)
    mymodels={}
    for i in range(6):
        filename='model_'+str(i)
        mymodels[columns[i]]= pickle.load(open(filename, 'rb'))
    for i in range(6):
        user_results = mymodels[columns[i]].predict_proba(user_data)[:,1]
        results2[columns[i]] = user_results
    x = columns
    return results2.iloc[0].values,x
```

Conversion of input text into no. using the previously saved vect file (vocabulary)

Loading the previously saved models i.e. model_0, model_1..., model_5

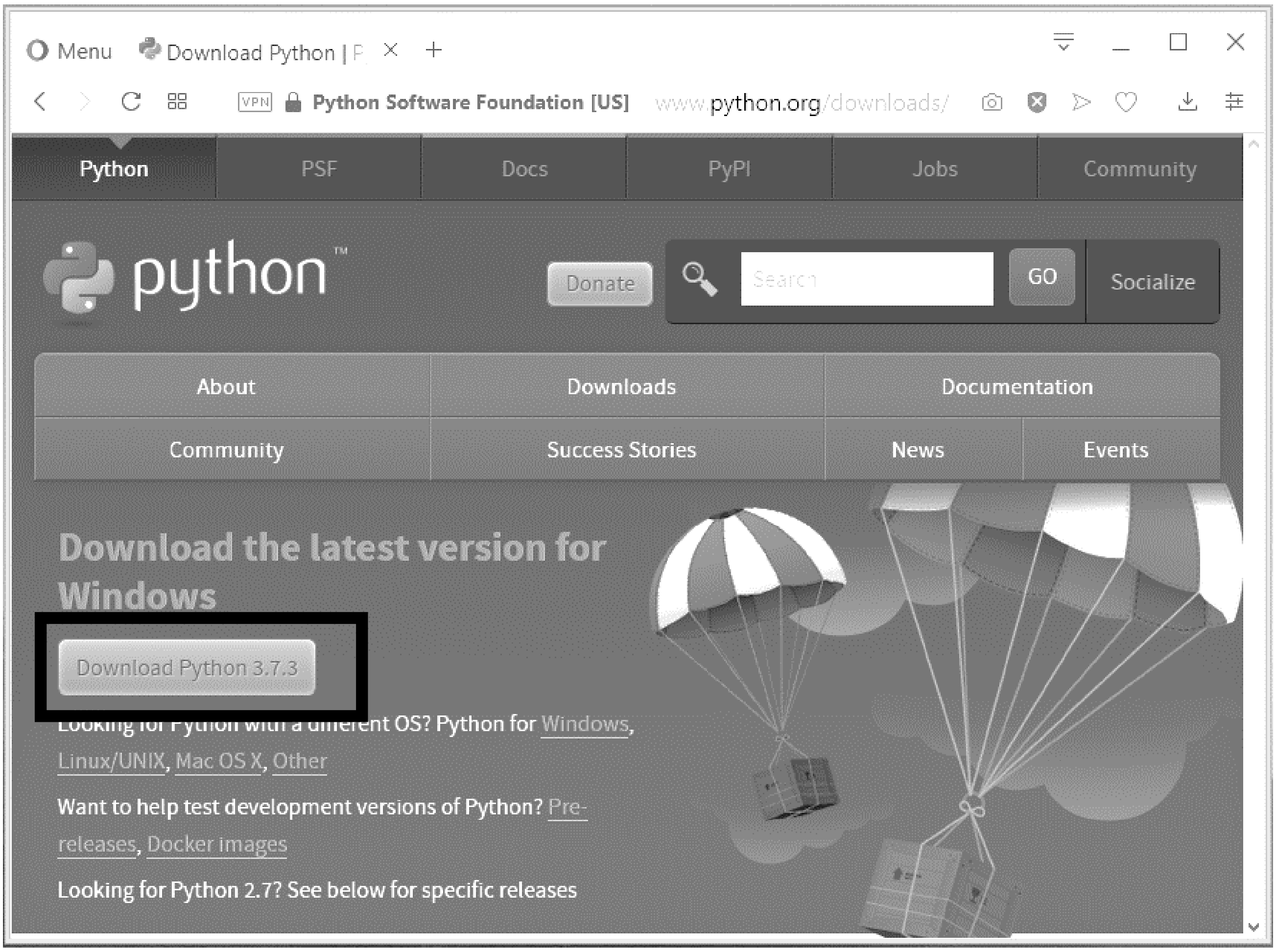
Predicting the result (i.e. computing the probability with each of the 6 models)

Returning the result

10. USING THE APPLICATION (USER GUIDE)

10.1 Making the System Ready:

1. **Installing Python**: Make sure the newest version of Python is installed in your system. If not, download it from here: https://www.python.org/downloads/



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

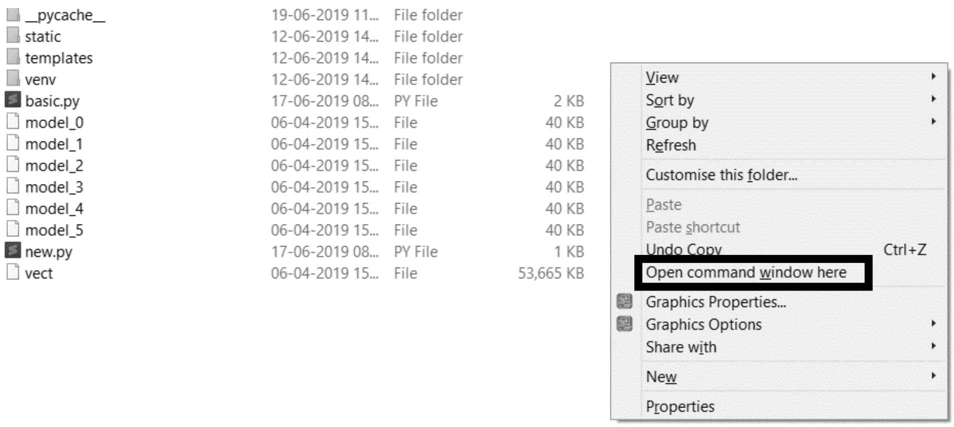
Looking for Python 2.7? See below for specific releases

2. **Installing the required python libraries**: Install the required libraries using the following Command Prompt (cmd) instructions:

```
>>>pip install numpy            >>>pip install pandas
>>>pip install flask            >>>pip install matplotlib.pyplot
>>>pip install os.path          >>>pip install pickle
```

3. **Creating a Virtual Environment**: Create a virtual environment for the project by opening the Command Prompt in the Project Directory (Shift + Right Click -> Open command window here). In the command window install the virtual environment using:

```
>>>pip install virtualenv

>>>virtualenv  venv
```



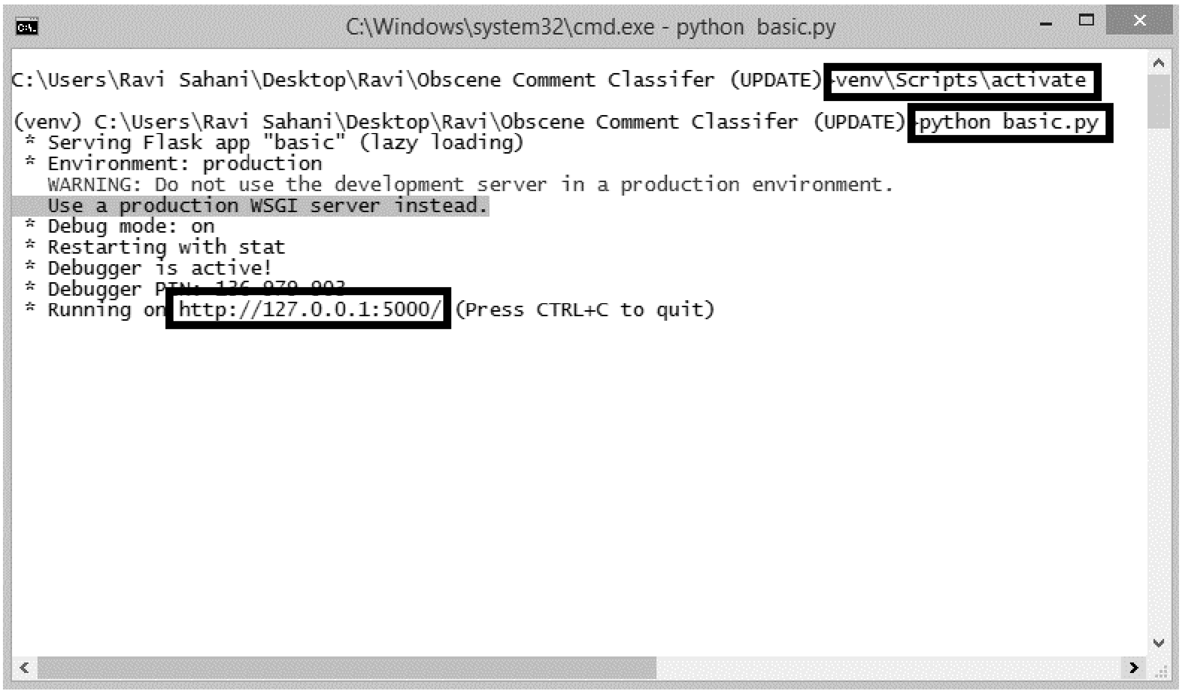
__pycache__	19-06-2019 11...	File folder	
static	12-06-2019 14...	File folder	
templates	12-06-2019 14...	File folder	
venv	12-06-2019 14...	File folder	
basic.py	17-06-2019 08...	PY File	2 KB
model_0	06-04-2019 15...	File	40 KB
model_1	06-04-2019 15...	File	40 KB
model_2	06-04-2019 15...	File	40 KB
model_3	06-04-2019 15...	File	40 KB
model_4	06-04-2019 15...	File	40 KB
model_5	06-04-2019 15...	File	40 KB
new.py	17-06-2019 08...	PY File	1 KB
vect	06-04-2019 15...	File	53,665 KB

10.2 Running the Application:

1. **Activating the Virtual Environment**: Open the command window in the project directory the activate the virtual environment using:

```
>>>venv\Scripts\activate
```

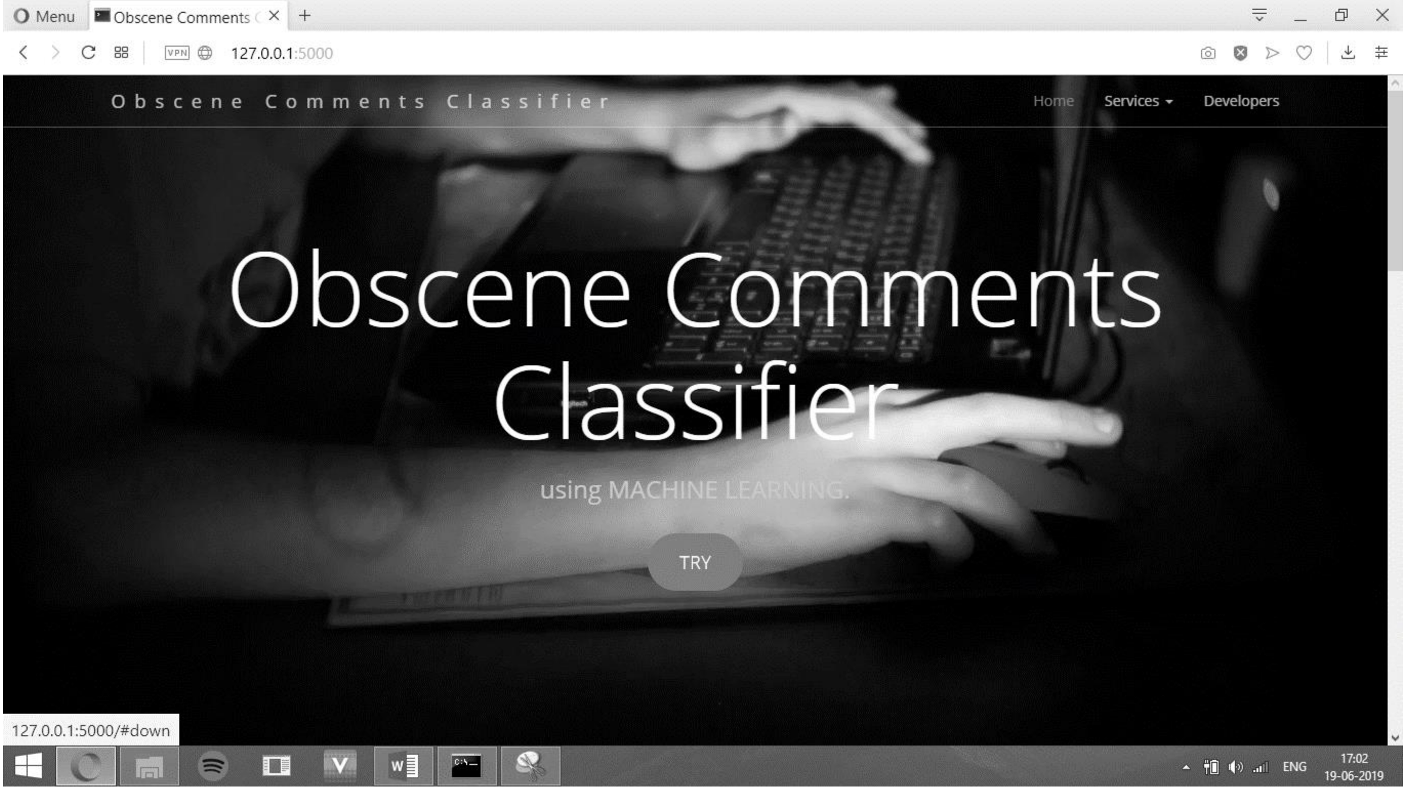
2. **Running basic.py**: *>>>python basic.py*



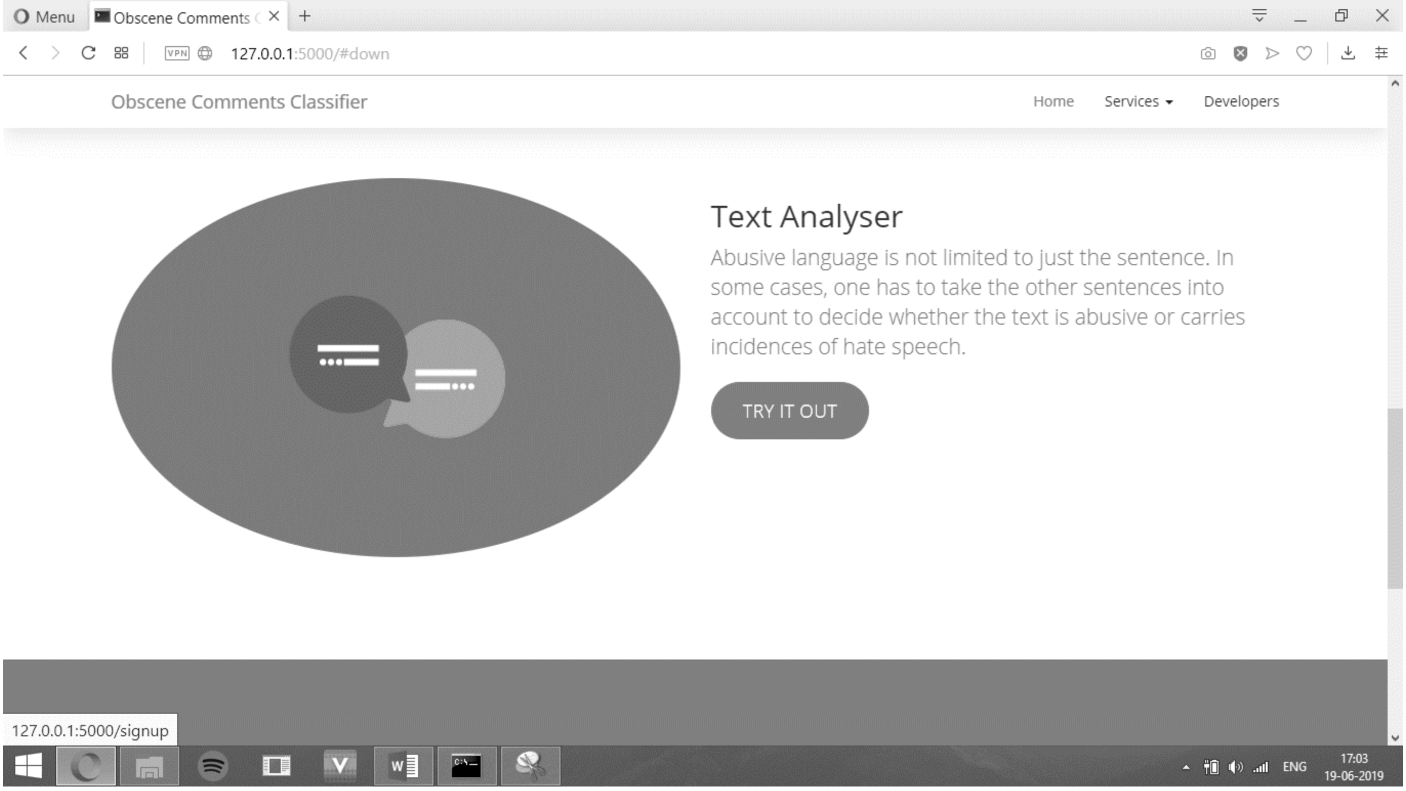
C:\Windows\system32\cmd.exe - python basic.py

```
C:\Users\Ravi Sahani\Desktop\Ravi\Obscene Comment Classifer (UPDATE)>venv\Scripts\activate

(venv) C:\Users\Ravi Sahani\Desktop\Ravi\Obscene Comment Classifer (UPDATE)>python basic.py
 * Serving Flask app "basic" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 136-079-993
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

3. **Open Browser, Visit http://127.0.0.1:5000 and click TRY.**



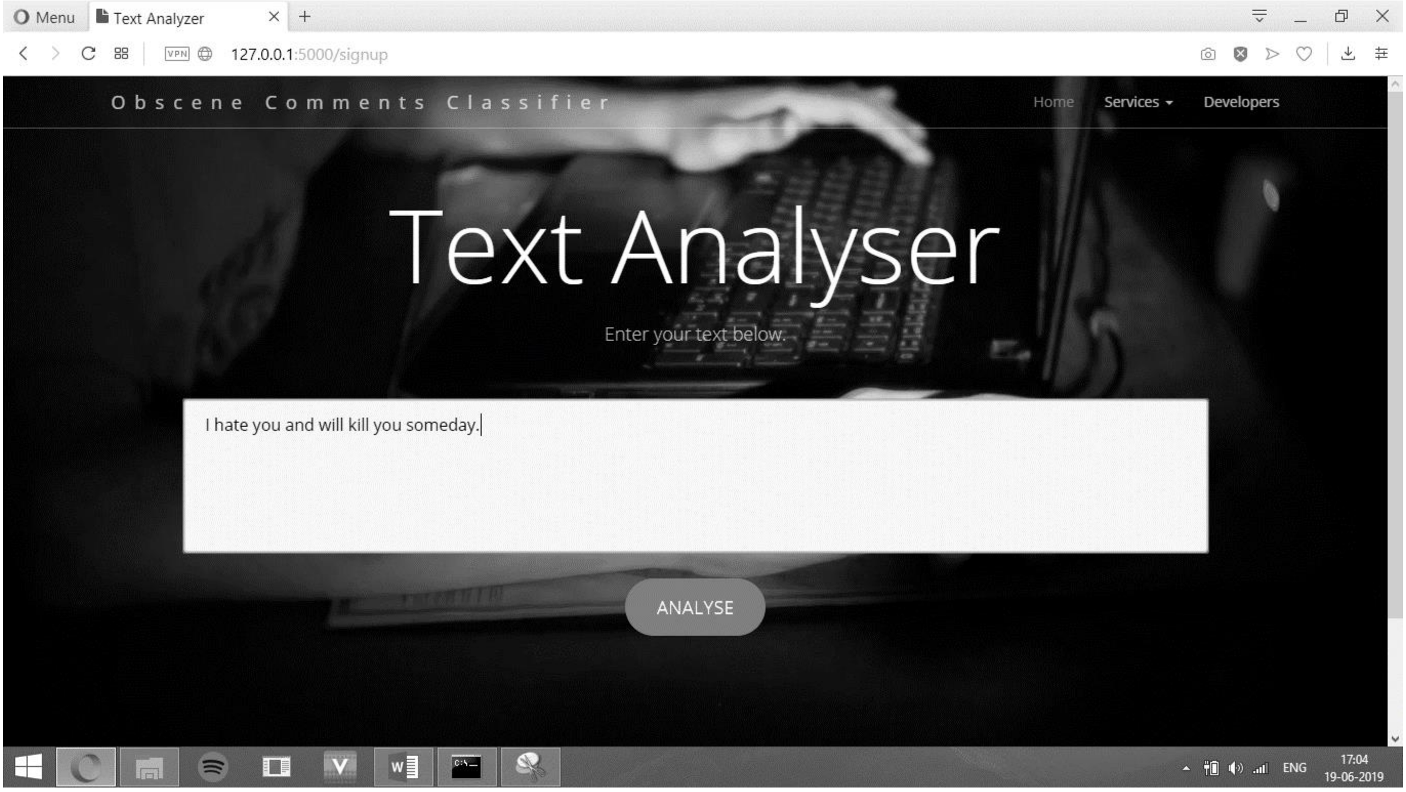
4. **Click on TRY IT OUT under Text Analyser Section**.



Text Analyser

Abusive language is not limited to just the sentence. In some cases, one has to take the other sentences into account to decide whether the text is abusive or carries incidences of hate speech.

5. **Enter the text in the area provided and click on ANALYSE**.

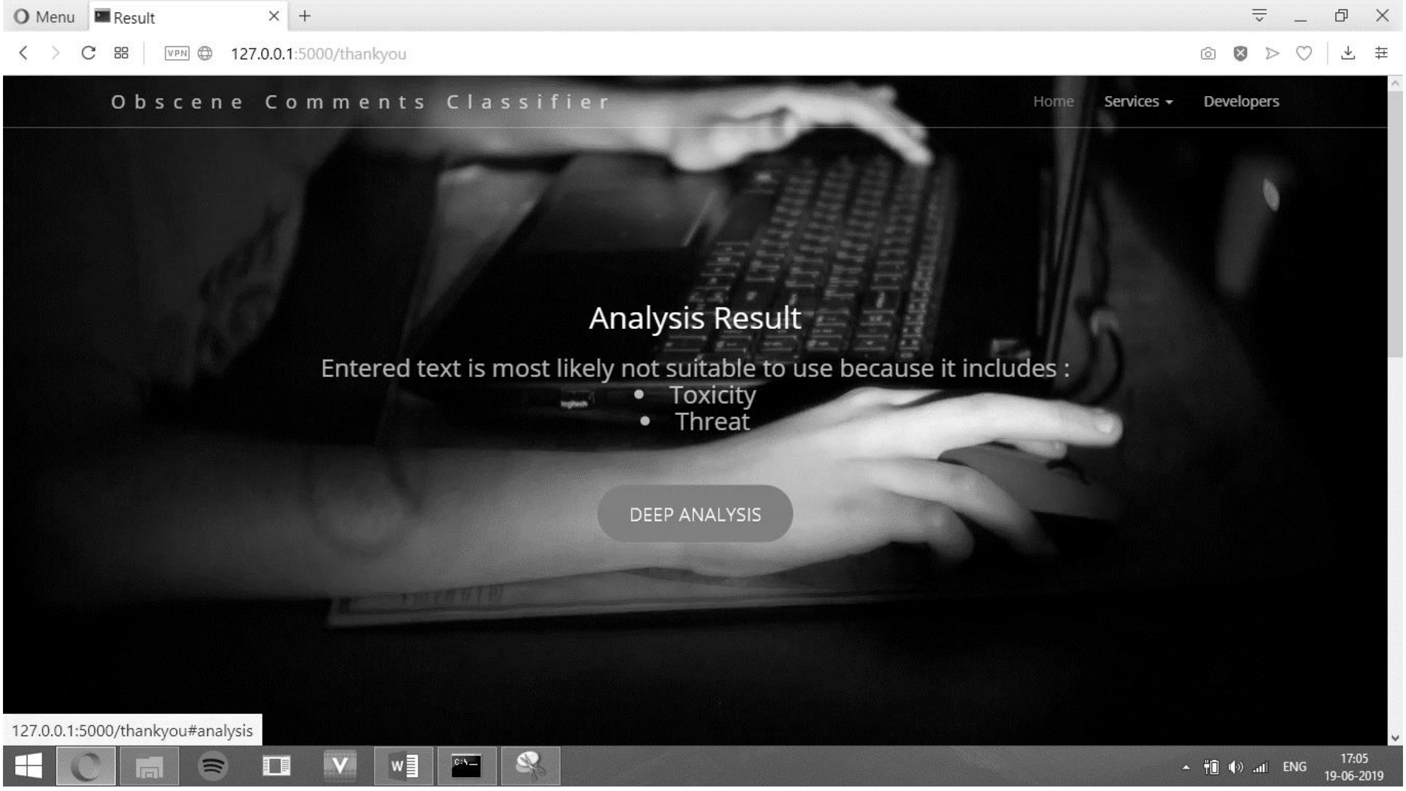


Text Analyser

Enter your text below.

I hate you and will kill you someday.

6. **Click on DEEP ANALYSES for Graphical analysis of the entered text**.

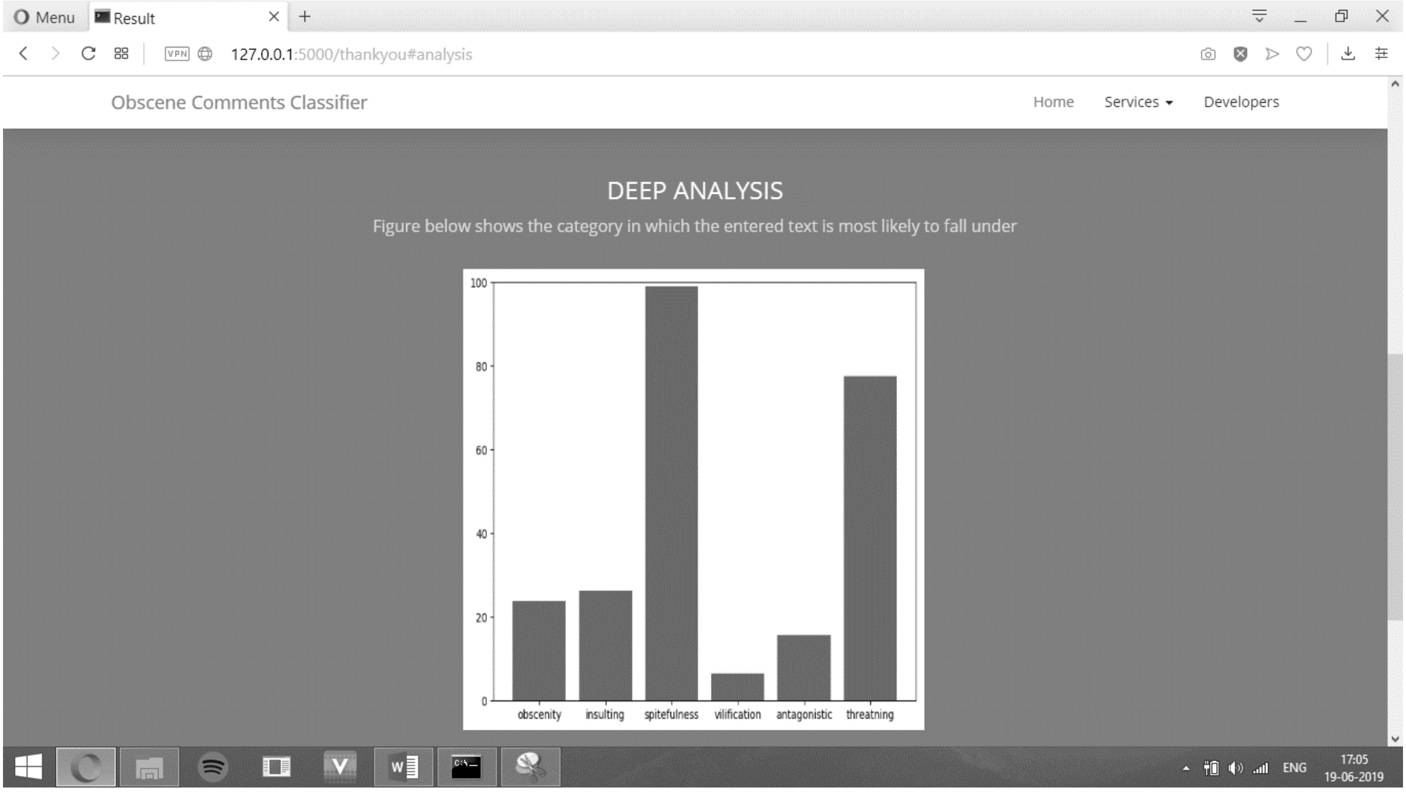


Analysis Result

Entered text is most likely not suitable to use because it includes :

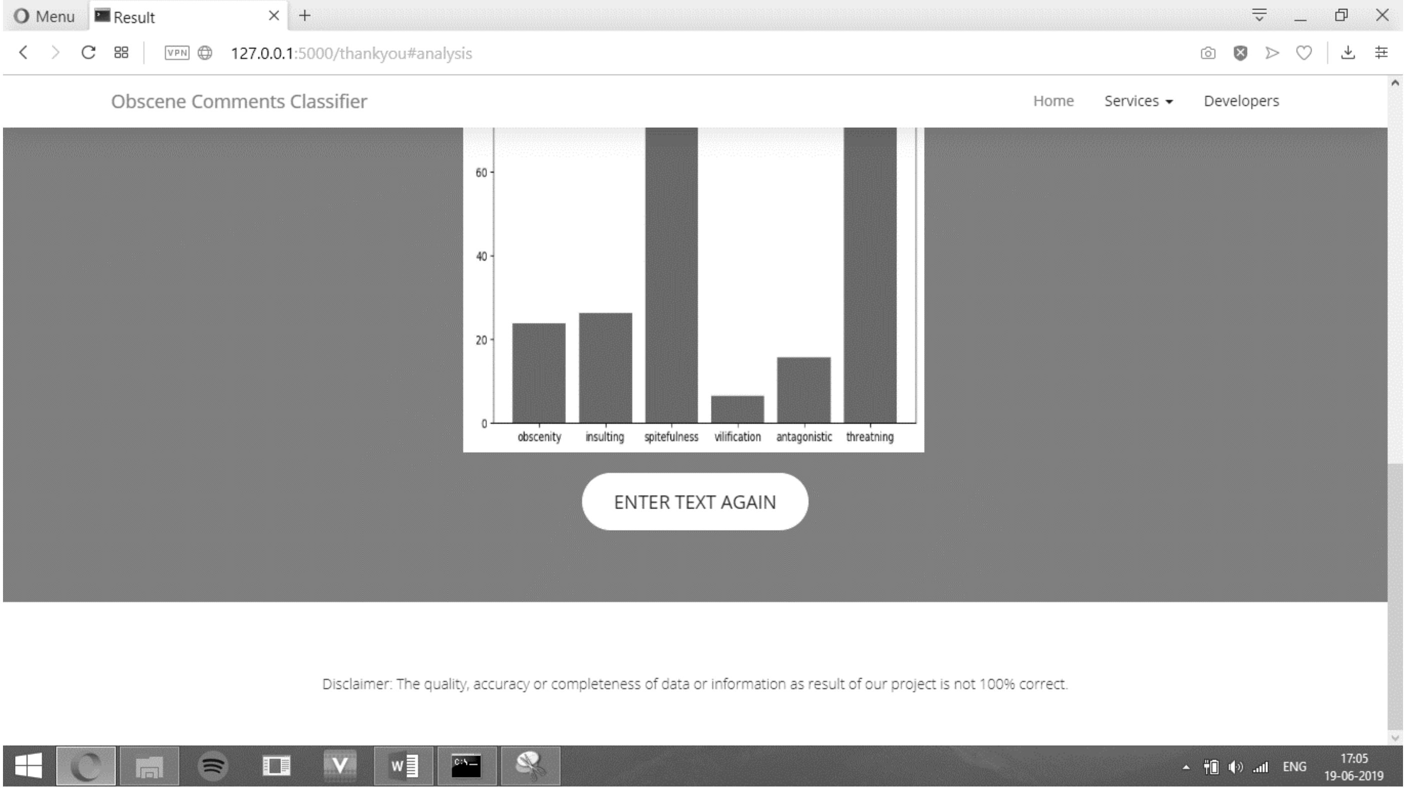
- Toxicity
- Threat

7. **Click on ENTER TEXT AGAIN to input again**.



DEEP ANALYSIS

Figure below shows the category in which the entered text is most likely to fall under



Disclaimer: The quality, accuracy or completeness of data or information as result of our project is not 100% correct.

11. FUTURE SCOPE

Some of the future scopes for this project includes:

1. Incorporation of toxic comment classifier with chat applications can help in creating obscenity free chat environment.

2. Toxic comment classifier can hamper people from spreading hatred and toxicity on various online platforms if used as filter.

3. Can be used in various discussion forums to maintain the decorum of the forum and hence resulting more fruitful discussions.

4. Can be used in identification of negative online behaviour of different groups and individuals and consequentially, restricting their scope.

5. Can be used in classifying tweets.

6. Can be used as a filter in doubt section of live seminars.

BIBLIOGRAPHY

1. Article on ‘**Machine Learning for Beginners’**

   By: Divyansh Dwivedi

   https://towardsdatascience.com/machine-learning-for-beginners-d247a9420dab

2. Article on ‘**Introduction to Machine Learning’**

   https://www.geeksforgeeks.org/machine-learning/

3. Book ‘**Introduction to Machine Learning with Python: A Guide for Data Scientists**’

   By: Andreas C. Müller and Sarah Guido

   Published by: O'Reilly Media, Inc.

4. Website ‘**Flask: Web Development, One Drop at the Time**’

   http://flask.pocoo.org

5. Article on ‘**How to Prepare Text Data for Machine Learning with scikit-learn**’

   By: Jason Brownlee

   https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/

6. Article on ‘**Building a Logistic Regression in Python, Step by Step**’

   By: Susan Li

   https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

7. Dataset: **Toxic Comment Classification**

   https://www.kaggle.com