**1. INTRODUCTION**

Stack Overflow is the largest, most trusted online community for developers to learn, share their programming knowledge, and build their careers.  
It is something which every programmer uses one way or another. Each month, over 50 million developers come to Stack Overflow to learn, share their knowledge, and build their careers. It features questions and answers on a wide range of topics in computer programming. The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers in a fashion similar to a wiki or Digg. As of April 2014, Stack Overflow has over 4,000,000 registered users, and it exceeded 10,000,000 questions in late August 2015. Based on the type of tags assigned to questions, the top eight most discussed topics on the site are: Java, JavaScript, C#, PHP, Android, jQuery, Python and HTML.

Each question can have at most ﬁve tags, these tags are keywords that serve to distinguish the questions and organize them in topics. These tags are assigned manually by the user (with an auto-ﬁll feature), or by selecting them from a list of most popular tags. This categorization is multi-class and multi-tag, which means, a question can be assigned to different topics and can also have several tags. Stack Overﬂow provided a training set of more than three million questions; each question consists of an identiﬁer, the question title or subject, the question body (which is a free format text ﬁeld) and a list of the tags assigned to that question. They also provided a test set which has around 20 thousand questions in the same format, but without the tags. The question body comes as the user captured it, including HTML tags to format the question. Although the language of the web site is English, some questions contain non-ASCII characters. Also, some questions contain code segments in different programming and mark-up languages, or even alphanumeric tables (with or without format). This is extremely business critical. The more accurately Stack Overflow can predict these tags the better it can create an Ecosystem to send the right question to the right set of people.

* 1. **Objectives**

The objective is to identify tags for Stack Overflow questions by proposing a 3-way classifier technique with SVM, Naïve Bayes and Logistic Regression algorithms. The dataset used is from Kaggle. It consists of 4 columns namely, Id, Title, Body and Tags. The same question with different tags is assigned to each of the 3 different classifiers. In this way a multi-tag classifier is generated. For prediction, the classifier with the most precision and recall is used followed by the remaining classifiers. The dataset chosen contains a maximum of 3 tags for each question.

* 1. **Methodology**

This model uses the following methodology: for each title and question body, the model returns a series of activations for each possible tag, where the tag with the highest activation is the one associated with the evaluated question. The association strength between the words used in the questions and its tags was determined using two thirds of the data set. The model predictions are based on the number of co-occurrences between the words and the tags used in the questions. These tags are clustered using a database of synonym tags

**1.2.1 Dataset**

Proper and large dataset is required for all classiﬁcation research during the training and the testing phase. The dataset used for this project is provided by Kaggle for identifying keywords and tags from millions of text questions. Stack Exchange (and its users) has released the source dataset through its [Creative Commons Data Dumps](http://blog.stackoverflow.com/category/cc-wiki-dump). The questions are randomized and contains a mix of verbose text sites as well as sites related to math and programming. The number of questions from each site may vary, and no filtering has been performed on the questions (such as closed questions).

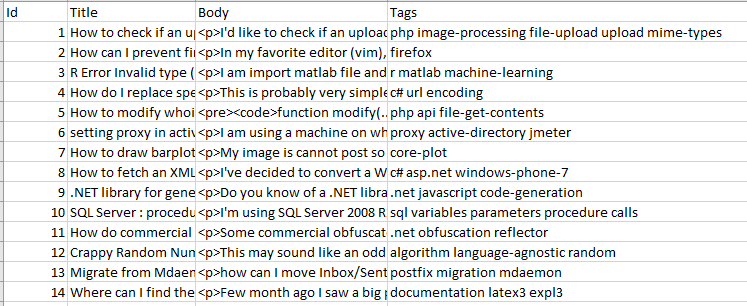
All of the data is in 2 files: Train and Test.

Train.csv contains 4 columns: Id, Title, Body, Tags

* Id - Unique identifier for each question
* Title - The question's title
* Body - The body of the question
* Tags - The tags associated with the question (all lowercase, should not contain tabs '\t' or ampersands '&')

Test.csv contains the same columns but without the Tags, which you are to predict.

The train.csv is of size 2.18 GB and the test.csv is of size 780 MB. The trainset contains nearly 5,40,000 rows.



**Fig 1.2.1.1 Dataset**

**1.2.2 Data cleaning**

Since the questions are in the format provided by the StackOverﬂow database, it was necessary to perform certain preprocessing to convert it to usable data for a classifying

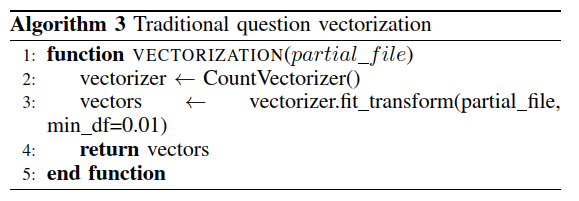
method. Also, because of the length of the questions and the vocabulary used in them which was extensive and variate. Two preprocessing methods were designed for this purpose: one performing a series text cleaning tasks and a “lazy” preprocessing method that makes use of certain parameters of the software library. In both methods the resulting data is vectorized by a Bag of Words (BoW) vectorizer.

**1.2.2.1 Traditional preprocessing**

* With this preprocessing, certain text elements are identiﬁed and reduced or eliminated by querying each question contained in the training set and individually processed for each kind of element to remove.
* Basically, this preprocessing removes: punctuation signs; unusual words i.e. everything that is considered speciﬁc for the question, like variable names, proper names, idioms, numbers, etc; and HTML tags to format the questions.
* Nonetheless, some exceptions were done in the case of unusual words, speciﬁcally words used in computer science. To preserve certain terms, a computer related words lexicon was made to be considered usual words.
* In order to reduce the number of features, a stemmer was incorporated to the preprocessing task, which is a process that removes morphological afﬁxes from words, leaving only the word stem.

**1.2.2.2 Vectorization**

* The new ﬁle with the preprocessed questions is then passed to a vectorizer object which implements the BoW method. This object accepts any iterable data structure, which can be ﬁle handlers or lists.
* Although this series of tasks extract most of the features necessary to perform classiﬁcation, still many features were present in only one question. To remove them, the vectorizer uses a minimum document frequency (i.e. a parameter to control if a feature is present in less of a certain number of documents) to remove every feature that has a document frequency smaller than 0.01.

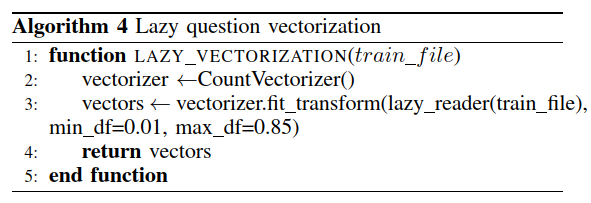


**Fig 1.2.2.2.1 Vectorization**

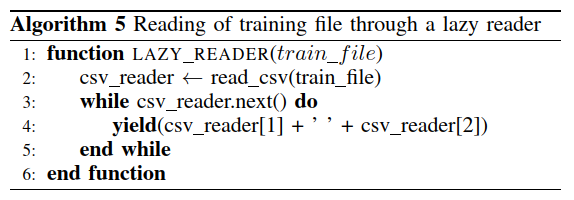
**1.2.2.3 Lazy preprocessing**

* Considering that the vectorizer object can receive any iterable object a simpler method was devised, where by using an intermediate function that retrieves directly from the training set ﬁle the question text and stores it on a temporary list that gets iterated like a ﬁle. This method is described in Algorithm 4.
* The lazy reader function simply reads the selected columns of the csv ﬁle, accumulates them with the yield operator and return it when the end of ﬁle is reached (Algorithm 5).

With the methods described above, it was possible to generate two preprocessing schemes, which we compared later with the classiﬁcation schemes devised.



**Fig 1.2.2.3.1 Lazy Vectorization**



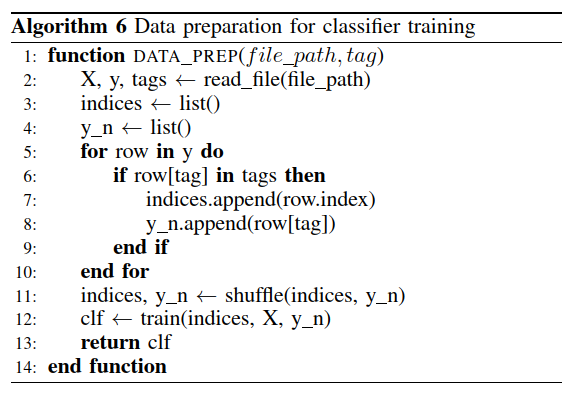
**Fig 1.2.2.3.2 Reading file through lazy reader**

**1.2.3 Classification System: -**

For this work a classiﬁcation system composed of ﬁve multi-class classiﬁers was developed. Each classiﬁer is focused on predicting a single tag of the ﬁve possible tags a question can have from a pool of several classes. To test the viability of this approach, we used three classiﬁcation models: Naive Bayes (NB), Support Vector Machines (SVM) and Logistic Regression (LR).

**1.2.3.1 Data preparation:**

* Certain data preparation is needed to train the classiﬁers. Firstly, we need to indicate the tag position that the classiﬁer is going to be trained, i.e. the ﬁrst, second, third, fourth or ﬁfth tag that a question can have. Once this is determined, the method retrieves the tags for that position for all questions.
* Algorithm 6 does the following actions. It receives file path which contains the questions in the form of feature vectors (variable X), its tags (variable y) and the classes selected to train (variable tags); tag is an integer that will serve to the algorithm to select the tag position that will be used to train the classiﬁer. For every question (line 5) we check if the question has a tag that exist in the list tags (line 6), if so, its index is stored in the indices list (line 7) along with its tag (line 8). These lists are then passed to the training algorithm where the vectors indicated by its indices are the ones used to train the classiﬁer.



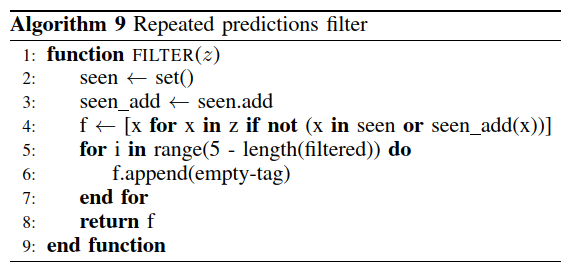
**Fig 1.2.3.1.1 Data preparation**

**1.2.3.2 Classifier training:**

All the 5 classifiers are trained with 3 algorithms namely, Support Vector machine (SVM), Naïve Bayes (NB), and Logistic Regression (LR). For every batch, we obtain the predictions of the ﬁve classiﬁers and arrange them by using the column stack function on the all list. In occasions, the classiﬁers will make the same prediction so we devised a method to deal with these repeats and called it filter (Algorithm 9), the result of the ﬁltering process is stored in a list called z. Then, the list z is rearranged and its contents stored in the y\_pred list. Once all the batches are processed y\_pred is returned to be compared with the true tags.

**1.2.3.3 Prediction:**

* To make predictions, the questions are passed to the ﬁve classiﬁers. Each classiﬁer predicts a tag for its corresponding position. This will make a ﬁnal prediction in the form of a list of integers, where each integer represents the associated tag. In certain cases, two or more classiﬁers make the same prediction, so a ﬁlter was included to deal with repetitions.
* Algorithm 9 shows the ﬁltering process. Basically, it generates a list f with as many x elements contained in z if they are not stored in the set seen or if they are not have been added to seen (line 4). This way we traverse z in order and skip any repeated element. This process will make f shorter than z if repeats were present, so we ﬁll f with as many empty – tag tags needed (line 6) and return f.



**Fig 1.2.3.3.1 Repeated predictions filter**

With the developed methods it was possible to process the questions and convert them to characteristic vectors, which were used to train the proposed classiﬁer scheme. As we are going to present in the next section, it was possible to attain a series of results that show the capacity of the classiﬁer method.

**2. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT**

**2.1 Requirements Gathering**

**2.1.1 Software Requirements**

Programming Language: Python 3.8

Graphical User Interface: Python Tkinter

Dataset : Stack Overflow Dataset from Kaggle

Packages : NumPy, Pandas, Matplotlib, Scikit-learn, Tkinter, Collections

**2.1.2 Hardware Requirements**

Operating System: Windows 10

Processor : Intel Core i5

Memory : 8 GB (RAM)

**2.2 Technologies Description**

**2.2.1 Python**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Some key advantages of learning Python:

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Characteristics of Python:

Following are important characteristics of Python Programming −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Applications of Python:

As mentioned before, Python is one of the most widely used language over the web.

* Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* Easy-to-read − Python code is more clearly defined and visible to the eyes.
* Easy-to-maintain − Python's source code is fairly easy-to-maintain.
* A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* Extendable − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* Databases − Python provides interfaces to all major commercial databases.
* GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* Scalable − Python provides a better structure and support for large programs than shell scripting.

**2.2.2 Stack Overflow Dataset**

All community-contributed content on Stack Exchange is licensed under the [Creative Commons BY-SA 3.0 license](http://creativecommons.org/licenses/by-sa/3.0/). As part of that commitment, a dump of all user-contributed data (after carefully sanitizing it to protect user private data, of course) is released quarterly. Each site can be downloaded individually, and includes an archive with Posts, Users, Votes, Comments, Badges, PostHistory, and PostLinks (new). Stack Exchange (and its users) has generously released the source dataset through its [Creative Commons Data Dumps](http://blog.stackoverflow.com/category/cc-wiki-dump). It is now available in Kaggle.

**2.2.3 Numpy**

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy:

Using NumPy, a developer can perform the following operations −

* Mathematical and logical operations on arrays.
* Fourier transforms and routines for shape manipulation.
* Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

**2.2.4 Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas:

* Fast and efficient DataFrame object with default and customized indexing.
* Tools for loading data into in-memory data objects from different file formats.
* Data alignment and integrated handling of missing data.
* Reshaping and pivoting of date sets.
* Label-based slicing, indexing and subsetting of large data sets.
* Columns from a data structure can be deleted or inserted.
* Group by data for aggregation and transformations.
* High performance merging and joining of data.
* Time Series functionality.

**2.2.5 Matplotlib**

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.

Matplotlib was originally written by John D. Hunter in 2003. The current stable version is 2.2.0 released in January 2018.

**2.2.6 Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

* **NumPy**: Base n-dimensional array package
* **SciPy**: Fundamental library for scientific computing
* **Matplotlib**: Comprehensive 2D/3D plotting
* **IPython**: Enhanced interactive console
* **Sympy**: Symbolic mathematics
* **Pandas**: Data structures and analysis

Extensions or modules for SciPy care conventionally named [SciKits](http://scikits.appspot.com/scikits). As such, the module provides learning algorithms and is named scikit-learn.

The vision for the library is a level of robustness and support required for use in production systems. This means a deep focus on concerns such as easy of use, code quality, collaboration, documentation and performance.

Although the interface is Python, c-libraries are leverage for performance such as numpy for arrays and matrix operations, [LAPACK](http://www.netlib.org/lapack/), [LibSVM](http://www.csie.ntu.edu.tw/~cjlin/libsvm/) and the careful use of cython.

**2.2.7 Tkinter**

Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

* Tkinter − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
* wxPython − This is an open-source Python interface for wxWindows [http://wxpython.org](http://wxpython.org/).
* JPython − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine [http://www.jython.org](http://www.jython.org/).

There are many other interfaces available, which you can find them on the net.

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the Tkinter module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.
* Enter the main event loop to take action against each event triggered by the user.

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets. There are currently 15 types of widgets in Tkinter.

Geometry Management:

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

* [The pack() Method](https://www.tutorialspoint.com/python/tk_pack.htm) − This geometry manager organizes widgets in blocks before placing them in the parent widget.
* [The grid() Method](https://www.tutorialspoint.com/python/tk_grid.htm) − This geometry manager organizes widgets in a table-like structure in the parent widget.

[The place() Method](https://www.tutorialspoint.com/python/tk_place.htm) − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

**2.2.8 Collections**

Collections in Python are containers that are used to store collections of data, for example, list, dict, set, tuple etc. These are built-in collections. Several modules have been developed that provide additional data structures to store collections of data. One such module is the Python [collections module](https://docs.python.org/2/library/collections.html).

Python collections module was introduced to improve the functionalities of the built-in collection containers. Python collections module was first introduced in its 2.4 release.  Some of the useful data structures present in this module are:

* Counter
* Defaultdict
* OrderedDict
* Deque
* ChainMap
* namedtuple()

**2.3 Algorithms**

**2.3.1 Support Vector machine**

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

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

The disadvantages of support vector machines include:

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

[SVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC), [NuSVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html#sklearn.svm.NuSVC) and [LinearSVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC) are classes capable of performing multi-class classification on a dataset. [SVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC) and [NuSVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html#sklearn.svm.NuSVC) are similar methods, but accept slightly different sets of parameters and have different mathematical formulations (see section [Mathematical formulation](https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation)). On the other hand, [LinearSVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC) is another implementation of Support Vector Classification for the case of a linear kernel. Note that [LinearSVC](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC) does not accept keyword kernel, as this is assumed to be linear.

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

Applications:

SVMs can be used to solve various real-world problems:

* SVMs are helpful in [text and hypertext categorization](https://en.wikipedia.org/wiki/Text_categorization), as their application can significantly reduce the need for labeled training instances in both the standard inductive and [transductive](https://en.wikipedia.org/wiki/Transduction_(machine_learning)) settings. Some methods for [shallow semantic parsing](https://en.wikipedia.org/wiki/Shallow_semantic_parsing) are based on support vector machines.
* [Classification of images](https://en.wikipedia.org/wiki/Image_classification) can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true for [image segmentation](https://en.wikipedia.org/wiki/Image_segmentation) systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik.
* Classification of satellite data like [SAR](https://en.wikipedia.org/wiki/Synthetic-aperture_radar) data using supervised SVM.
* Hand-written characters can be [recognized](https://en.wikipedia.org/wiki/Handwriting_recognition) using SVM.
* The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly.

**2.3.2 Naïve Bayes**

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes’ theorem with the “naive” assumption of conditional independence between every pair of features given the value of the class variable. Bayes’ theorem states the following relationship, given class variable y and dependent feature vector x1 through xn :

P(y∣x1,…,xn)=P(y)P(x1,…xn∣y)P(x1,…,xn)

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. (For theoretical reasons why naive Bayes works well, and on which types of data it does, see the references below.)

Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

On the flip side, although naive Bayes is known as a decent classifier, it is known to be a bad estimator, so the probability outputs from predict\_proba are not to be taken too seriously.

**2.2.3 Logistic Regression**

In [statistics](https://en.wikipedia.org/wiki/Statistics), the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1 and the sum adding to one.

Logistic regression is a [statistical model](https://en.wikipedia.org/wiki/Statistical_model) that in its basic form uses a [logistic function](https://en.wikipedia.org/wiki/Logistic_function) to model a [binary](https://en.wikipedia.org/wiki/Binary_variable) [dependent variable](https://en.wikipedia.org/wiki/Dependent_variable), although many more complex [extensions](https://en.wikipedia.org/wiki/Logistic_regression#Extensions) exist. In [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis), logistic regression (or logit regression) is [estimating](https://en.wikipedia.org/wiki/Estimation_theory) the parameters of a logistic model (a form of [binary regression](https://en.wikipedia.org/wiki/Binary_regression)). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an [indicator variable](https://en.wikipedia.org/wiki/Indicator_variable), where the two values are labeled "0" and "1".

Applications:

Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences.

* For example, the Trauma and Injury Severity Score ([TRISS](https://en.wikipedia.org/wiki/TRISS)), which is widely used to predict mortality in injured patients, was originally developed by Boyd et al. using logistic regression. Many other medical scales used to assess severity of a patient have been developed using logistic regression.
* Logistic regression may be used to predict the risk of developing a given disease (e.g. [diabetes](https://en.wikipedia.org/wiki/Diabetes_mellitus); [coronary heart disease](https://en.wikipedia.org/wiki/Coronary_artery_disease)), based on observed characteristics of the patient (age, gender, [body mass index](https://en.wikipedia.org/wiki/Body_mass_index), results of various [blood tests](https://en.wikipedia.org/wiki/Blood_test), etc.).
* Another example might be to predict whether a Nepalese voter will vote Nepali Congress or Communist Party of Nepal or Any Other Party, based on age, income, gender, race, state of residence, votes in previous elections, etc. The technique can also be used in [engineering](https://en.wikipedia.org/wiki/Engineering), especially for predicting the probability of failure of a given process, system or product.
* It is also used in [marketing](https://en.wikipedia.org/wiki/Marketing) applications such as prediction of a customer's propensity to purchase a product or halt a subscription, etc.
* In [economics](https://en.wikipedia.org/wiki/Economics) it can be used to predict the likelihood of a person's choosing to be in the labor force, and a business application would be to predict the likelihood of a homeowner defaulting on a [mortgage](https://en.wikipedia.org/wiki/Mortgage). [Conditional random fields](https://en.wikipedia.org/wiki/Conditional_random_field), an extension of logistic regression to sequential data, are used in [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing).

**3. DESIGN**

**3.1 Introduction**

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement has been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

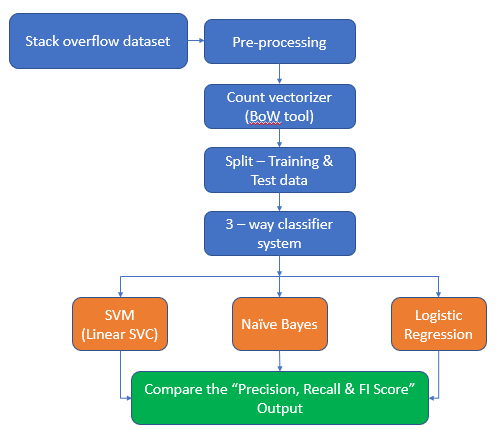
The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

**3.2 Architecture Diagram**

Web applications are by nature distributed applications, meaning that they are programs that run on more than one computer and communicate through a network or server. Specifically, web applications are accessed with a web browser and are popular because of the ease of using the browser as a user client. For the enterprise, software on potentially thousands of client computers is a key reason for their popularity. Web applications are used for web mail, online retail sales, discussion boards, weblogs, online banking, and more. One web application can be accessed and used by millions of people.

Like desktop applications, web applications are made up of many parts and often contain mini programs and some of which have user interfaces. In addition, web applications frequently require an additional markup or scripting language, such as HTML, CSS, or JavaScript programming language. Also, many applications use only the Python programming language, which is ideal because of its versatility.



**Fig 3.2.1 Architecture diagram**

**3.2 UML Diagrams**

**3.2.1 Use Case Diagram**

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

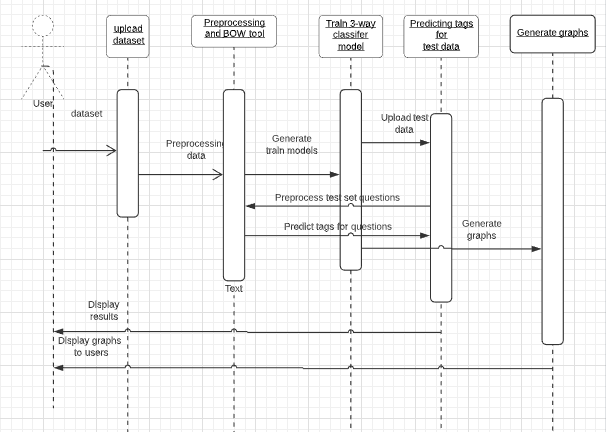
Hence to model the entire system, a number of use case diagrams are used.



**Fig 3.2.1.1 Use case diagram**

**3.2.2 Sequence Diagram**

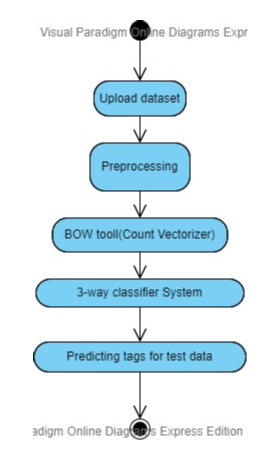
Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.



**Fig 3.2.2.1 Sequence diagram**

**3.2.3 Activity Diagram**

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency.In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**Fig 3.2.3.1 Activity diagram**

**4. IMPLEMENTATION**

**4.1 Coding**

**Multiclass3.py**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

import matplotlib.pyplot as plt

from tkinter.filedialog import askopenfilename

import numpy as np

import pandas as pd

import collections

import re

from sklearn.feature\_extraction.text import CountVectorizer

from nltk.corpus import stopwords

from sklearn import svm

from sklearn.naive\_bayes import MultinomialNB

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score

from sklearn.model\_selection import train\_test\_split

stop\_words = set(stopwords.words('english'))

main = tkinter.Tk()

main.title("Multi-class Multi-tag Classifier System for StackOverflow Questions")

main.geometry("1300x1200")

global filename

global cls\_svm1,cls\_svm2,cls\_svm3,cls\_svm4,cls\_svm5

global cls\_nb1,cls\_nb2,cls\_nb3,cls\_nb4,cls\_nb5

global cls\_lr1,cls\_lr2,cls\_lr3,cls\_lr4,cls\_lr5

global cv1,cv2,cv3,cv4,cv5

global X\_train1, X\_test1, y\_train1, y\_test1

global X\_train2, X\_test2, y\_train2, y\_test2

global X\_train3, X\_test3, y\_train3, y\_test3

global X\_train4, X\_test4, y\_train4, y\_test4

global X\_train5, X\_test5, y\_train5, y\_test5

**#Get the index of the tags stored in the tags list**

def getTagID(name):

tid = -1

for i in range(len(tag\_names)):

if tag\_names[i] == name:

tid = i

break;

return tid

#**Remove the HTML tags for the questions**

def rem\_html\_tags(question):

regex = re.compile('<.\*?>')

return re.sub(regex, '', question)

#**Remove the punctuation marks from the questions**

def removePunct(question):

question = re.sub('\W+',' ', question)

question = question.strip()

return question

#**Upload dataset for training**

def uploadTrain():

global filename

filename = filedialog.askopenfilename(initialdir="dataset")

#pathlabel.config(text=filename)

textarea.delete('1.0', END)

textarea.insert(END,"Dataset loaded\n");

**#Preprocess the questions in dataset and segregate**

def preprocess():

global tag\_names

ques = pd.read\_csv(filename,encoding='iso-8859-1')

count = 0

size = 0

data1 = 'Question,Tag\n'

data2 = 'Question,Tag\n'

data3 = 'Question,Tag\n'

data4 = 'Question,Tag\n'

data5 = 'Question,Tag\n'

tag\_names = []

for i in range(len(ques)):

size = size + 1

question = ques.get\_value(i, 'Body')

question = rem\_html\_tags(question)

question = removePunct(question)

tag = ques.get\_value(i, 'Tags')

tag = tag.replace("#", "sharp")

tag = tag.replace("++","pp")

tag = removePunct(tag)

tag = tag.split(" ")

for k in range(len(tag)):

if k == 0 and len(tag) > 0:

data1+=question+","+tag[0]+"\n"

if k == 1 and len(tag) > 1:

data2+=question+","+tag[1]+"\n"

if k == 2 and len(tag) > 2:

data3+=question+","+tag[2]+"\n"

tag\_names.append(str(tag[k]))

tag\_names = list(set(tag\_names))

print(tag\_names)

if data1 != 'Question,Tag\n':

f = open("set/set1.csv", "w")

f.write(data1)

f.close()

if data2 != 'Question,Tag\n':

f = open("set/set2.csv", "w")

f.write(data2)

f.close()

if data3 != 'Question,Tag\n':

f = open("set/set3.csv", "w")

f.write(data3)

f.close()

textarea.delete('1.0', END)

textarea.insert(END,"Total processed questions are : "+str(size)+"\n")

#textarea.insert(END,"All Set files saved inside Set folder\n")

#**Convert the preprocessed questions into vectors**

def vector(file):

X = []

Y = []

train = pd.read\_csv(file,encoding='iso-8859-1')

count = 0

for i in range(len(train)):

question = train.get\_value(i,0,takeable = True)

tag = train.get\_value(i,1,takeable = True)

question = question.lower()

arr = question.split(" ")

msg = ''

for k in range(len(arr)):

word = arr[k].strip()

if len(word) > 2 and word not in stop\_words:

msg+=word+" "

text = msg.strip()

X.append(text)

Y.append(getTagID(tag))

X = np.asarray(X)

Y = np.asarray(Y)

return X,Y

#**Apply the BoW tool (Count Vectorizer) for the preprocessed questions**

def countVector():

textarea.delete('1.0', END)

global cv1,cv2,cv3

global X\_train1, X\_test1, y\_train1, y\_test1

global X\_train2, X\_test2, y\_train2, y\_test2

global X\_train3, X\_test3, y\_train3, y\_test3

X1,Y1 = vector('set/set1.csv')

X2,Y2 = vector('set/set2.csv')

X3,Y3 = vector('set/set3.csv')

cv1 = CountVectorizer(analyzer='word',stop\_words = stop\_words, lowercase = True)

X1 = cv1.fit\_transform(X1).toarray()

cv2 = CountVectorizer(analyzer='word',stop\_words = stop\_words, lowercase = True)

X2 = cv2.fit\_transform(X2).toarray()

cv3 = CountVectorizer(analyzer='word',stop\_words = stop\_words, lowercase = True)

X3 = cv3.fit\_transform(X3).toarray()

textarea.insert(END,"Set 1 total records : "+str(X1.shape)+"\n")

textarea.insert(END,"Set 2 total records : "+str(X2.shape)+"\n")

textarea.insert(END,"Set 3 total records : "+str(X3.shape)+"\n")

X\_train1, X\_test1, y\_train1, y\_test1 = train\_test\_split(X1, Y1, test\_size = 0.2, random\_state = 0)

X\_train2, X\_test2, y\_train2, y\_test2 = train\_test\_split(X2, Y2, test\_size = 0.2, random\_state = 0)

X\_train3, X\_test3, y\_train3, y\_test3 = train\_test\_split(X3, Y3, test\_size = 0.2, random\_state = 0)

def dataPreparation():

textarea.delete('1.0', END)

textarea.insert(END,"Set 1 train records : "+str(X\_train1.shape)+" Set 1 test records : "+str(X\_test1.shape)+"\n")

textarea.insert(END,"Set 2 train records : "+str(X\_train2.shape)+" Set 2 test records : "+str(X\_test2.shape)+"\n")

textarea.insert(END,"Set 3 train records : "+str(X\_train3.shape)+" Set 3 test records : "+str(X\_test3.shape)+"\n")

**#Function to predict**

def prediction(X\_test, cls):

y\_pred = cls.predict(X\_test)

return y\_pred

**# Function to calculate accuracy**

def cal\_accuracy(y\_test, y\_pred):

accuracy = accuracy\_score(y\_test,y\_pred)\*100

return accuracy

**# Training with the SVM classifier**

def trainSVMClassifier():

textarea.delete('1.0', END)

global cls\_svm1,cls\_svm2,cls\_svm3

global acc\_svm

global recall\_svm

global pres\_svm

global f1\_svm

acc\_svm = []

pres\_svm = []

recall\_svm = []

f1\_svm = []

cls\_svm1 = svm.LinearSVC(penalty='l1', intercept\_scaling=1, dual=False)

cls\_svm1.fit(X\_train1, y\_train1)

y\_pred1 = cls\_svm1.predict(X\_test1)

acc\_svm.append(accuracy\_score(y\_test1,y\_pred1)\*100)

pres\_svm.append(precision\_score(y\_test1, y\_pred1, average='macro'))

recall\_svm.append(recall\_score(y\_test1, y\_pred1, average='macro'))

print(recall\_svm[0])

f1\_svm.append(f1\_score(y\_test1, y\_pred1, average='macro'))

cls\_svm2 = svm.LinearSVC(penalty='l1', intercept\_scaling=1, dual=False)

cls\_svm2.fit(X\_train2, y\_train2)

y\_pred2 = cls\_svm2.predict(X\_test2)

acc\_svm.append(accuracy\_score(y\_test2,y\_pred2)\*100)

pres\_svm.append(precision\_score(y\_test2, y\_pred2, average='macro'))

recall\_svm.append(recall\_score(y\_test2, y\_pred2, average='macro'))

#print(recall\_svm[1])

f1\_svm.append(f1\_score(y\_test2, y\_pred2, average='macro'))

cls\_svm3 = svm.LinearSVC(penalty='l1', intercept\_scaling=1, dual=False)

cls\_svm3.fit(X\_train3, y\_train3)

y\_pred3 = cls\_svm3.predict(X\_test3)

acc\_svm.append(accuracy\_score(y\_test3,y\_pred3)\*100)

pres\_svm.append(precision\_score(y\_test3, y\_pred3, average='macro'))

recall\_svm.append(recall\_score(y\_test3, y\_pred3, average='macro'))

#print(recall\_svm[2])

f1\_svm.append(f1\_score(y\_test3, y\_pred3, average='macro'))

textarea.insert(END," SVM Classifier1 Accuracy : {:.4f}".format(acc\_svm[0])+"\n")

textarea.insert(END," SVM Classifier2 Accuracy : {:.4f}".format(acc\_svm[1])+"\n")

textarea.insert(END," SVM Classifier3 Accuracy : {:.4f}".format(acc\_svm[2])+"\n")

**# Training with the NB classifier**

def trainNBClassifier():

textarea.delete('1.0', END)

global cls\_nb1,cls\_nb2,cls\_nb3

global acc\_nb

global recall\_nb

global pres\_nb

global f1\_nb

acc\_nb = []

pres\_nb = []

recall\_nb = []

f1\_nb = []

cls\_nb1 = MultinomialNB(fit\_prior=True, class\_prior=None)

cls\_nb1.fit(X\_train1, y\_train1)

y\_pred1 = cls\_nb1.predict(X\_test1)

acc\_nb.append(accuracy\_score(y\_test1,y\_pred1)\*100)

pres\_nb.append(precision\_score(y\_test1, y\_pred1, average='macro'))

recall\_nb.append(recall\_score(y\_test1, y\_pred1, average='macro'))

f1\_nb.append(f1\_score(y\_test1, y\_pred1, average='macro'))

cls\_nb2 = MultinomialNB(fit\_prior=True, class\_prior=None)

cls\_nb2.fit(X\_train2, y\_train2)

y\_pred2 = cls\_nb2.predict(X\_test2)

acc\_nb.append(accuracy\_score(y\_test2,y\_pred2)\*100)

pres\_nb.append(precision\_score(y\_test2, y\_pred2, average='macro'))

recall\_nb.append(recall\_score(y\_test2, y\_pred2, average='macro'))

f1\_nb.append(f1\_score(y\_test2, y\_pred2, average='macro'))

cls\_nb3 = MultinomialNB(fit\_prior=True, class\_prior=None)

cls\_nb3.fit(X\_train3, y\_train3)

y\_pred3 = cls\_nb3.predict(X\_test3)

acc\_nb.append(accuracy\_score(y\_test3,y\_pred3)\*100)

pres\_nb.append(precision\_score(y\_test3, y\_pred3, average='macro'))

recall\_nb.append(recall\_score(y\_test3, y\_pred3, average='macro'))

f1\_nb.append(f1\_score(y\_test3, y\_pred3, average='macro'))

f1\_nb.append(f1\_score(y\_test5, y\_pred5, average='macro'))

textarea.insert(END," NB Classifier1 Accuracy : {:.4f}".format(acc\_nb[0])+"\n")

textarea.insert(END," NB Classifier2 Accuracy : {:.4f}".format(acc\_nb[1])+"\n")

textarea.insert(END," NB Classifier3 Accuracy : {:.4f}".format(acc\_nb[2])+"\n")

**# Training with the LR classifier**

def trainLRClassifier():

textarea.delete('1.0', END)

global cls\_lr1,cls\_lr2,cls\_lr3

global acc\_lr

global recall\_lr

global pres\_lr

global f1\_lr

acc\_lr = []

pres\_lr = []

recall\_lr = []

f1\_lr = []

cls\_lr1 = LogisticRegression()

cls\_lr1.fit(X\_train1, y\_train1)

y\_pred1 = cls\_lr1.predict(X\_test1)

acc\_lr.append(accuracy\_score(y\_test1,y\_pred1)\*100)

pres\_lr.append(precision\_score(y\_test1, y\_pred1, average='macro'))

recall\_lr.append(recall\_score(y\_test1, y\_pred1, average='macro'))

f1\_lr.append(f1\_score(y\_test1, y\_pred1, average='macro'))

cls\_lr2 = LogisticRegression()

cls\_lr2.fit(X\_train2, y\_train2)

y\_pred2 = cls\_lr2.predict(X\_test2)

acc\_lr.append(accuracy\_score(y\_test2,y\_pred2)\*100)

pres\_lr.append(precision\_score(y\_test2, y\_pred2, average='macro'))

recall\_lr.append(recall\_score(y\_test2, y\_pred2, average='macro'))

f1\_lr.append(f1\_score(y\_test2, y\_pred2, average='macro'))

cls\_lr3 = LogisticRegression()

cls\_lr3.fit(X\_train3, y\_train3)

y\_pred3 = cls\_lr3.predict(X\_test3)

acc\_lr.append(accuracy\_score(y\_test3,y\_pred3)\*100)

pres\_lr.append(precision\_score(y\_test3, y\_pred3, average='macro'))

recall\_lr.append(recall\_score(y\_test3, y\_pred3, average='macro'))

f1\_lr.append(f1\_score(y\_test3, y\_pred3, average='macro'))

textarea.insert(END," LR Classifier1 Accuracy : {:.4f}".format(acc\_lr[0])+"\n")

textarea.insert(END," LR Classifier2 Accuracy : {:.4f}".format(acc\_lr[1])+"\n")

textarea.insert(END," LR Classifier3 Accuracy : {:.4f}".format(acc\_lr[2])+"\n")

**# Function to preprocess the test data questions**

def processLine(line):

line = rem\_html\_tags(line)

line = removePunct(line)

msg = ''

arr = line.split(' ')

for i in range(len(arr)):

arr[i] = arr[i].strip()

if len(arr[i]) > 2 and arr[i] not in stop\_words:

msg+=arr[i]+" "

msg = msg.strip();

print(msg)

return msg

**# Function to predict the tags for test data**

def predict():

textarea.delete('1.0', END)

filename = filedialog.askopenfilename(initialdir="dataset")

index = 1

with open(filename, "r") as file: #reading emotion word

for line in file:

ques = line

line = line.strip('\n')

line = line.strip()

temp = line

line = line.lower()

line = processLine(line)

cv = CountVectorizer(vocabulary=cv1.get\_feature\_names(),stop\_words = "english", lowercase = True)

test1 = cv.fit\_transform([line])

cv = CountVectorizer(vocabulary=cv2.get\_feature\_names(),stop\_words = "english", lowercase = True)

test2 = cv.fit\_transform([line])

cv = CountVectorizer(vocabulary=cv3.get\_feature\_names(),stop\_words = "english", lowercase = True)

test3 = cv.fit\_transform([line])

c1 = cls\_lr1.predict(test1.toarray())[0]

c2 = cls\_lr2.predict(test2.toarray())[0]

c3 = cls\_lr3.predict(test3.toarray())[0]

final\_tags = [tag\_names[c1], tag\_names[c2], tag\_names[c3]]

final\_tags = list(dict.fromkeys(final\_tags))

textarea.insert(END, 'Question- '+ques+ "\n TAG Predicted AS : ")

textarea.insert(END, str(final\_tags)+"\n\n")

index = index + 1

**# Function to generate precision graph**

def Pres\_graph():

#bars = ('Cls1 Precision','Cls2 Precision','Cls3 Precision')

#y\_pos = np.arange(len(bars))

#plt.bar(y\_pos, pres\_lr)

#plt.xticks(y\_pos, bars)

N = 3

ind = np.arange(N) # the x locations for the groups

width = 0.27 # the width of the bars

fig = plt.figure()

ax = fig.add\_subplot(111)

rects1 = ax.bar(ind, pres\_svm, width, color='indianred')

rects2 = ax.bar(ind+width, pres\_nb, width, color='goldenrod')

rects3 = ax.bar(ind+width\*2, pres\_lr, width, color='lightseagreen')

ax.set\_ylabel('Precision Values')

ax.set\_xticks(ind+width)

ax.set\_xticklabels( ('Classifer1', 'Classifer2', 'Classifer3') )

ax.legend( (rects1[0], rects2[0], rects3[0]), ('SVM', 'NB', 'LR') )

def autolabel(rects):

for rect in rects:

h = rect.get\_height()

ax.text(rect.get\_x() - 0.05, 1.02\*h, "{:.2}".format(h))

autolabel(rects1)

autolabel(rects2)

autolabel(rects3)

plt.show()

**# Function to generate Recall graph**

def Rec\_graph():

#bars = ('Cls1 Precision','Cls2 Precision','Cls3 Precision')

#y\_pos = np.arange(len(bars))

#plt.bar(y\_pos, pres\_lr)

#plt.xticks(y\_pos, bars)

N = 3

ind = np.arange(N) # the x locations for the groups

width = 0.27 # the width of the bars

fig = plt.figure()

ax = fig.add\_subplot(111)

rects1 = ax.bar(ind, recall\_svm, width, color='indianred')

rects2 = ax.bar(ind+width, recall\_nb, width, color='goldenrod')

rects3 = ax.bar(ind+width\*2, recall\_lr, width, color='lightseagreen')

ax.set\_ylabel('Recall Values')

ax.set\_xticks(ind+width)

ax.set\_xticklabels( ('Classifer1', 'Classifer2', 'Classifer3') )

ax.legend( (rects1[0], rects2[0], rects3[0]), ('SVM', 'NB', 'LR') )

def autolabel(rects):

for rect in rects:

h = rect.get\_height()

ax.text(rect.get\_x() - 0.05, 1.02\*h, "{:.2}".format(h))

autolabel(rects1)

autolabel(rects2)

autolabel(rects3)

plt.show()

**# Function to generate F1 score graph**

def F1\_graph():

#bars = ('Cls1 Precision','Cls2 Precision','Cls3 Precision')

#y\_pos = np.arange(len(bars))

#plt.bar(y\_pos, pres\_lr)

#plt.xticks(y\_pos, bars)

N = 3

ind = np.arange(N) # the x locations for the groups

width = 0.27 # the width of the bars

fig = plt.figure()

ax = fig.add\_subplot(111)

rects1 = ax.bar(ind, f1\_svm, width, color='indianred')

rects2 = ax.bar(ind+width, f1\_nb, width, color='goldenrod')

rects3 = ax.bar(ind+width\*2, f1\_lr, width, color='lightseagreen')

ax.set\_ylabel('F1 Scores')

ax.set\_xticks(ind+width)

ax.set\_xticklabels( ('Classifer1', 'Classifer2', 'Classifer3') )

ax.legend( (rects1[0], rects2[0], rects3[0]), ('SVM', 'NB', 'LR') )

def autolabel(rects):

for rect in rects:

h = rect.get\_height()

ax.text(rect.get\_x() - 0.05, 1.02\*h, "{:.2}".format(h))

autolabel(rects1)

autolabel(rects2)

autolabel(rects3)

plt.show()

def close():

global main

main.destroy()

font = ('times', 16, 'bold')

title = Label(main, text='Multi-class Multi-tag Classifier System for StackOverflow Questions')

title.config(bg='mint cream', fg='olive drab')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 14, 'bold')

uploadButton = Button(main, text="Upload Stackoverflow Dataset", command=uploadTrain)

uploadButton.place(x=50,y=100)

uploadButton.config(font=font1)

#pathlabel = Label(main)

#pathlabel.config(bg='mint cream', fg='olive drab')

#pathlabel.config(font=font1)

#pathlabel.place(x=460,y=100)

processButton = Button(main, text="Preprocess Questions", command=preprocess)

processButton.place(x=450,y=100)

processButton.config(font=font1)

countButton = Button(main, text="Count Vectorization", command=countVector)

countButton.place(x=700,y=100)

countButton.config(font=font1)

dataButton = Button(main, text="Data Preparation For Classifier", command=dataPreparation)

dataButton.place(x=950,y=100)

dataButton.config(font=font1)

trainButton = Button(main, text="Train SVM Classifier", command=trainSVMClassifier)

trainButton.place(x=100,y=170)

trainButton.config(font=font1)

trainButton = Button(main, text="Train NB Classifier", command=trainNBClassifier)

trainButton.place(x=330,y=170)

trainButton.config(font=font1)

trainButton = Button(main, text="Train LR Classifier", command=trainLRClassifier)

trainButton.place(x=560,y=170)

trainButton.config(font=font1)

predictButton = Button(main, text="Upload Test data", command=predict)

predictButton.place(x=890,y=170)

predictButton.config(font=font1)

graphButton = Button(main, text="Precision Graph", command=Pres\_graph)

graphButton.place(x=350,y=240)

graphButton.config(font=font1)

graphButton = Button(main, text="Recall Graph", command=Rec\_graph)

graphButton.place(x=580,y=240)

graphButton.config(font=font1)

graphButton = Button(main, text="F1-score Graph", command=F1\_graph)

graphButton.place(x=810,y=240)

graphButton.config(font=font1)

font1 = ('times', 12, 'bold')

textarea=Text(main,height=20,width=158)

scroll=Scrollbar(textarea)

textarea.configure(yscrollcommand=scroll.set)

textarea.place(x=20,y=300)

textarea.config(font=font1)

main.config(bg='gainsboro')

main.mainloop()

**4.2 Testing**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant costs associated with a software failure are motivating factors for we planned, through testing. Testing is the process of executing a program with the intent of finding an error. The design of tests for software and other engineered products can be as challenging as the initial design of the product itself.

There are basically two types of testing approaches.

One is Black-Box testing – the specified function that a product has been designed to   perform, tests can be conducted that demonstrate each  function is fully operated.

The other is White-Box testing – knowing the internal workings  of the product ,tests can  be conducted to ensure that the  internal operation of the product performs according  to specifications and all internal components have been  adequately exercised.

White box   and Black box   testing methods have been used to test this package.    The entire loop constructs have been tested for their boundary and intermediate conditions. The   test data was designed with a view to check for all the conditions and logical decisions.  Error handling has been taken care of by the use of exception handlers.

**4.2.1 Testing Strategies**

Testing is a set of activities that can be planned in advance and conducted systematically.  A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

Software testing is one element of verification and validation.   Verification refers to the set of activities that ensure that software correctly implements a specific function.  Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

The main objective of software is testing to uncover errors.  To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed.  Each test step is accomplished through a series of systematic test techniques that assist in the design of test cases. With each testing step, the level of abstraction with which software is considered is broadened.

Testing is the only way to assure the quality of software and it is an umbrella activity rather than a separate phase. This is an activity to be performed in parallel with the software effort and one that consists of its own phases of analysis, design, implementation, execution and maintenance.

**Unit Testing:**

This testing method considers a module as a single unit and checks the unit at interfaces and communicates with other modules rather than getting into details at statement level. Here the module will be treated as a black box, which will take some input and generate output.  Outputs for a given set of input combinations are pre-calculated and are generated by the module.

**System Testing:**

Here all the pre tested individual modules will be assembled to create the larger system and tests are carried out at system level to make sure that all modules are working in synchronous with each other.  This testing methodology helps in making sure that all modules which are running perfectly when checked individually are also running in cohesion with other modules. For this testing we create test cases to check all modules once and then generate test combinations of test paths throughout the system to make sure that no path is making its way into chaos.

**Integrated Testing**

Testing is a major quality control measure employed during software development.  Its basic function is to detect errors. Sub functions when combined may not produce what is desired.  Global data structures can represent the problems. Integrated testing is a systematic technique for constructing the program structure while conducting the tests. To uncover errors that are associated with interfacing the objective is to make unit test modules and build a program structure that has been detected by design.  In a non - incremental integration all the modules are combined in advance and the program is tested as a whole. Here errors will appear in an endless loop function. In incremental testing the program is constructed and tested in small segments where the errors are isolated and corrected.

Different incremental integration strategies are top – down integration, bottom – up integration, regression testing.

**Regression Testing**

Each time a new module is added as a part of integration as the software changes. Regression testing is an act that helps to ensure changes that do not introduce unintended behavior as additional errors.

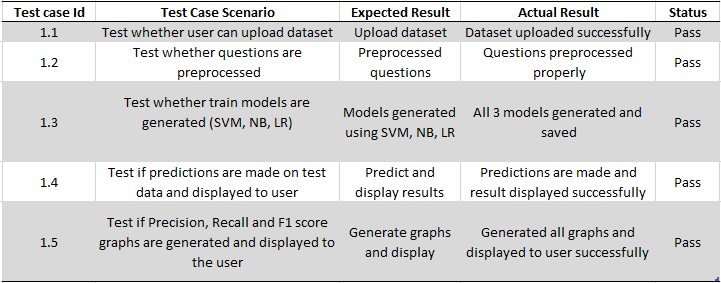
Regression testing may be conducted manually by executing a subset of all test cases or using automated capture playback tools enables the software engineer to capture the test case and results for subsequent playback and compression.  The regression suit contains different classes of test cases.

A representative sample to tests that will exercise all software functions.

Additional tests that focus on software functions that are likely to be affected by the change.

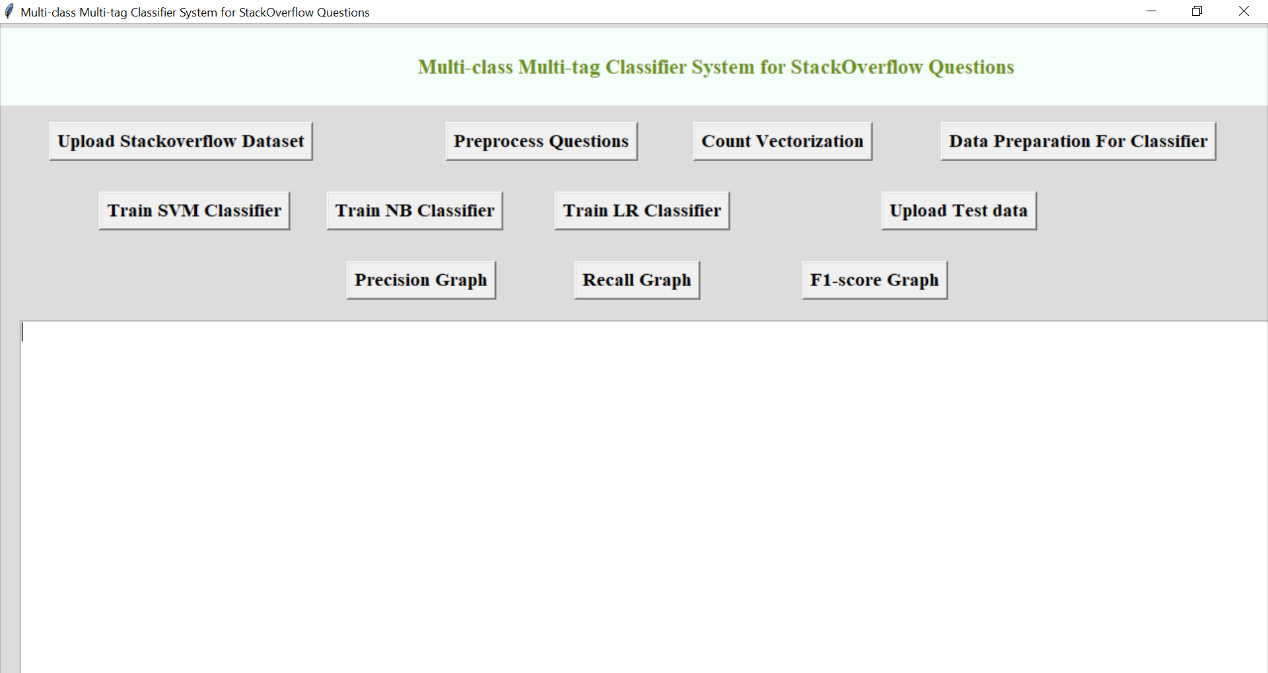
**4.3 Test Cases**

Integrated and regression testing strategies are used in this application for testing.

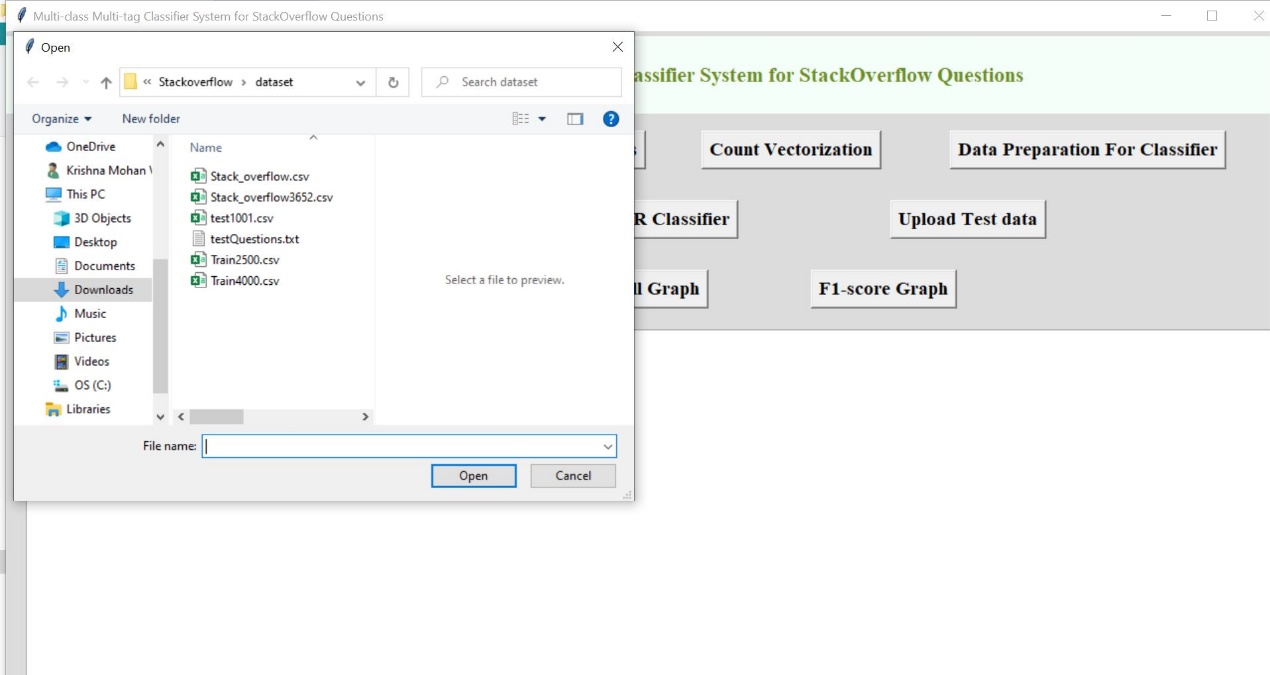


**Fig 4.3.1 Test cases**

**4.4 Screenshots**



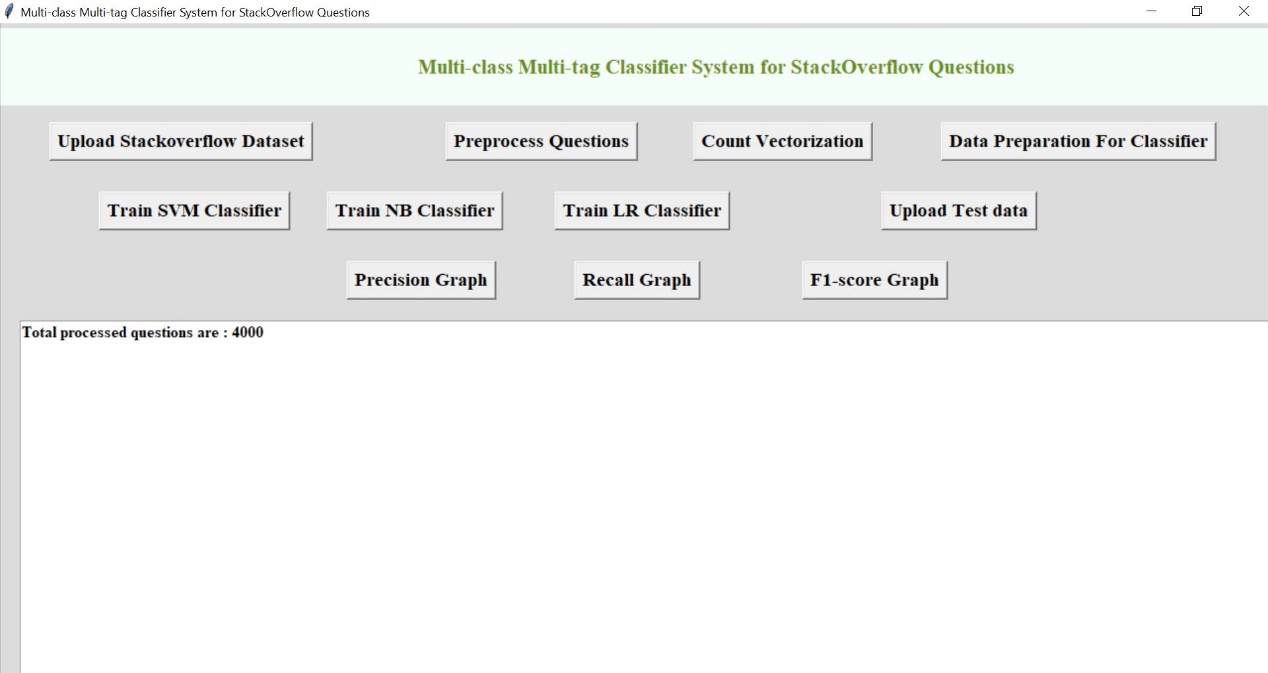
**Fig 4.4.1 Application screen**



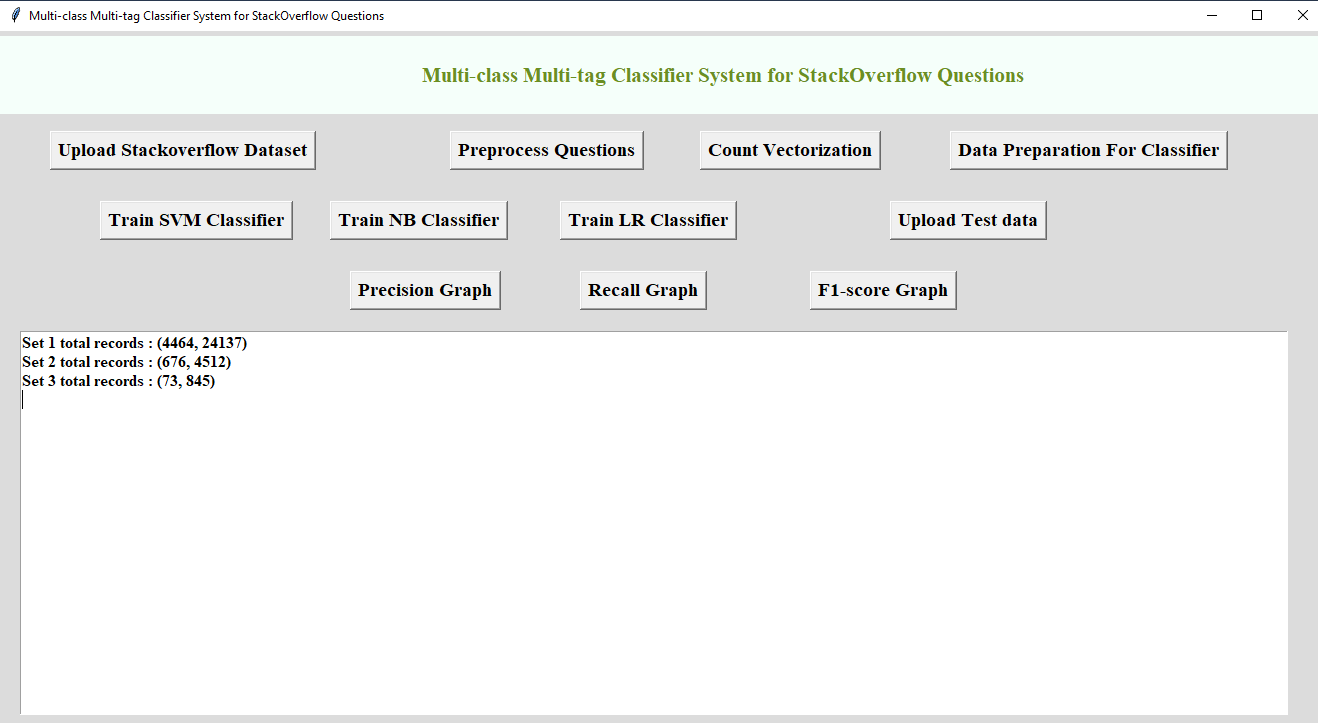
**Fig 4.4.2 Uploading Stack Overflow dataset**



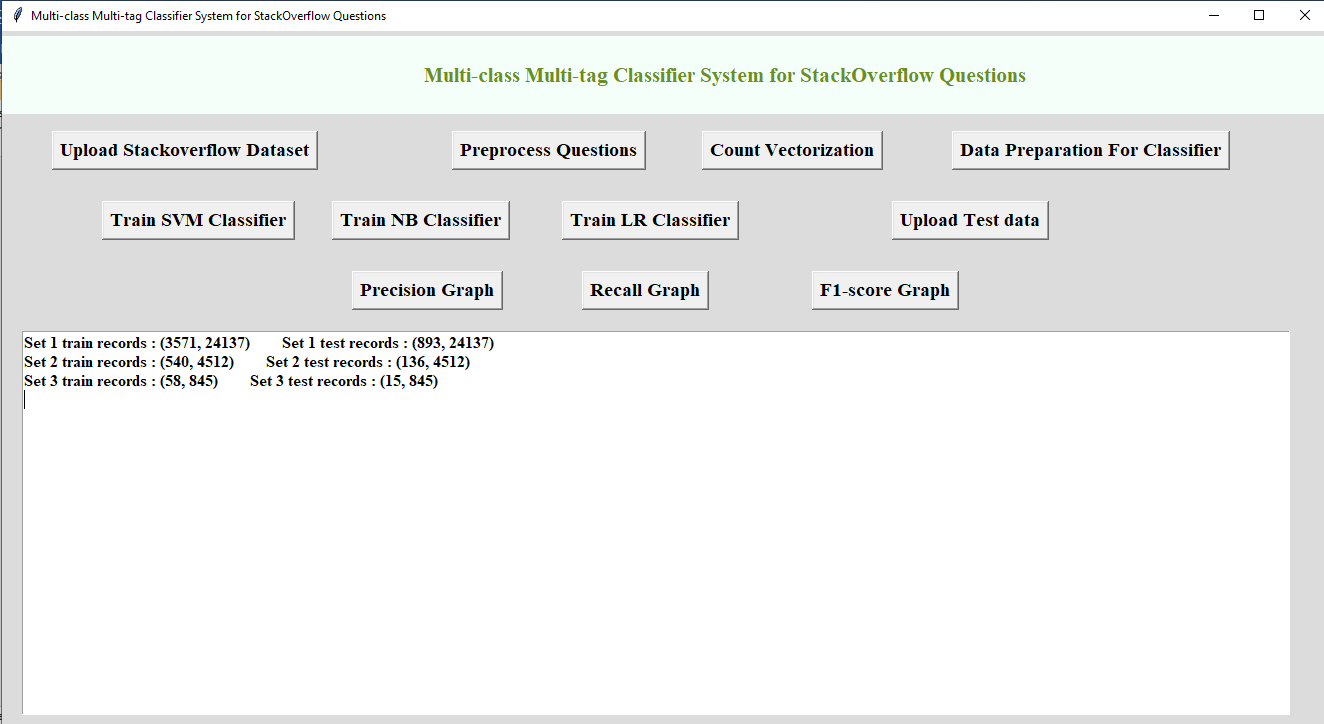
**Fig 4.4.3 Dataset uploaded**



**Fig 4.4.4 Preprocessed all the questions in dataset**

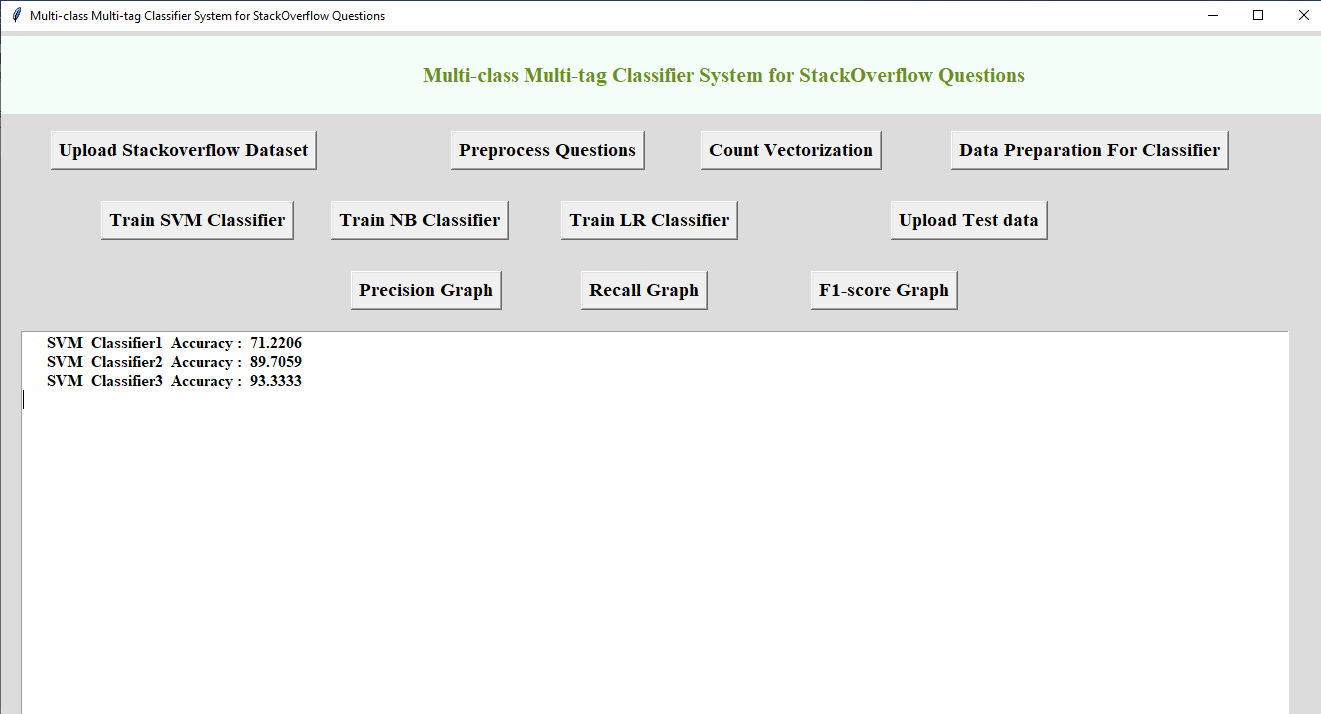


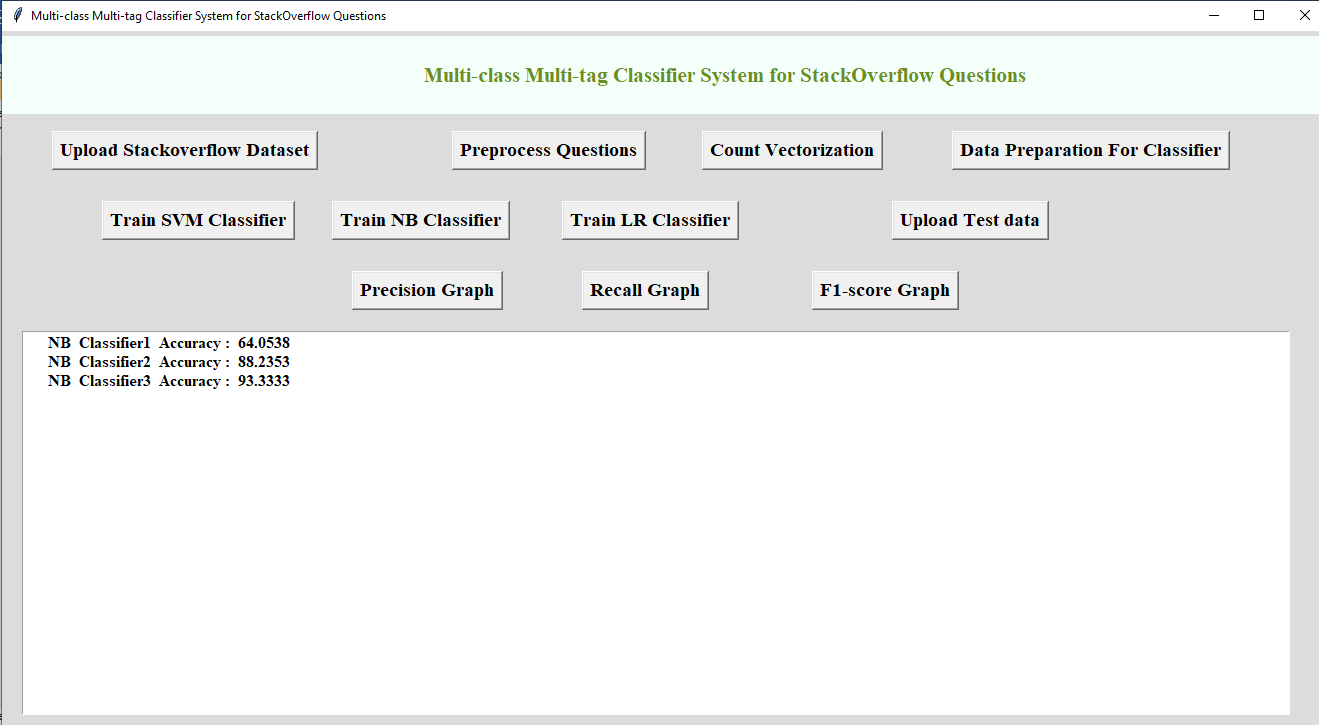
**Fig 4.4.5 Applying Count Vectorizer (BoW) tool**



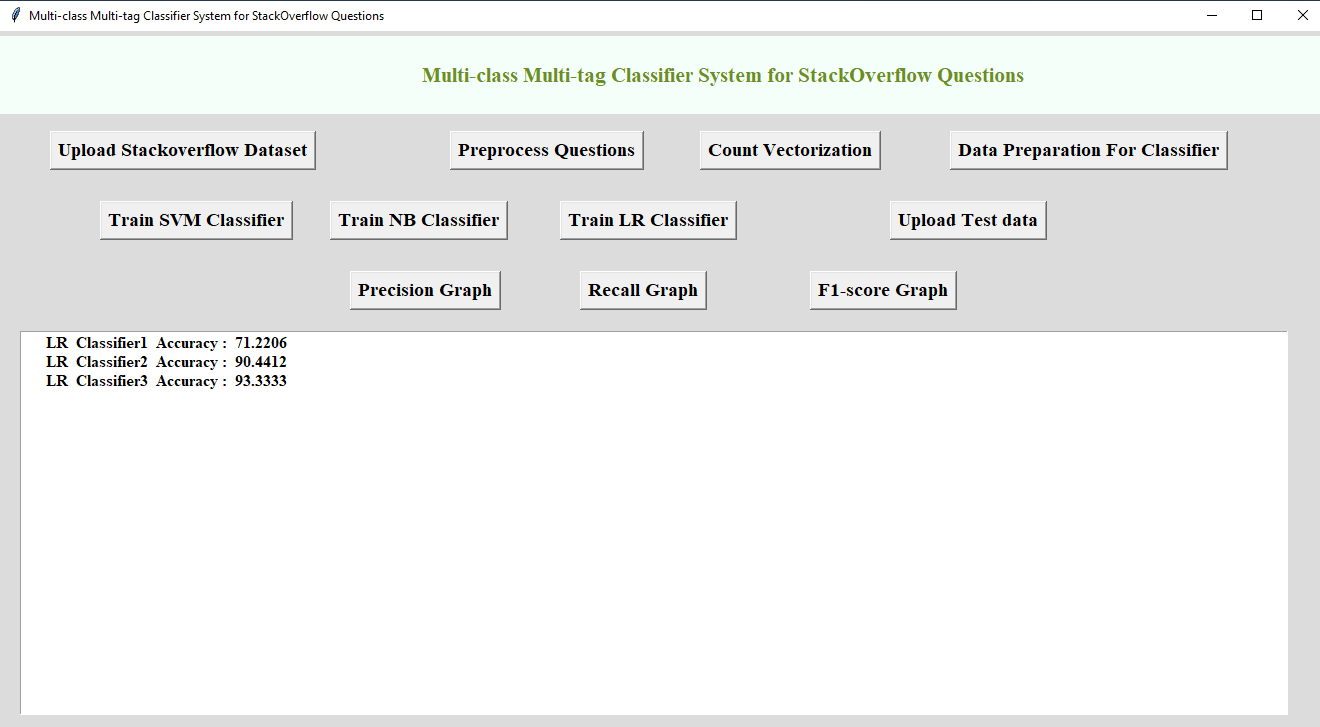
**Fig 4.4.6 Preparing the data for training all the 3 algorithms**

**5. RESULTS AND EVALUATION**

**5.1 Evaluation metrics**

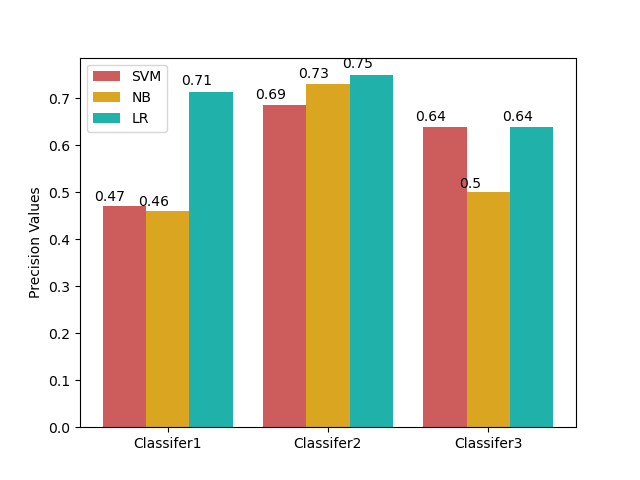
**Fig 5.1.1 SVM 3-way classifier accuracies**

**Fig 5.1.2 NB 3-way classifier accuracies**

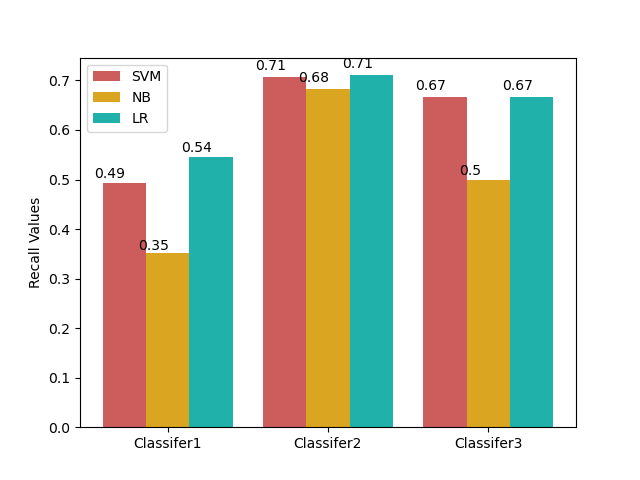


**Fig 5.1.3 LR 3-way classifier accuracies**

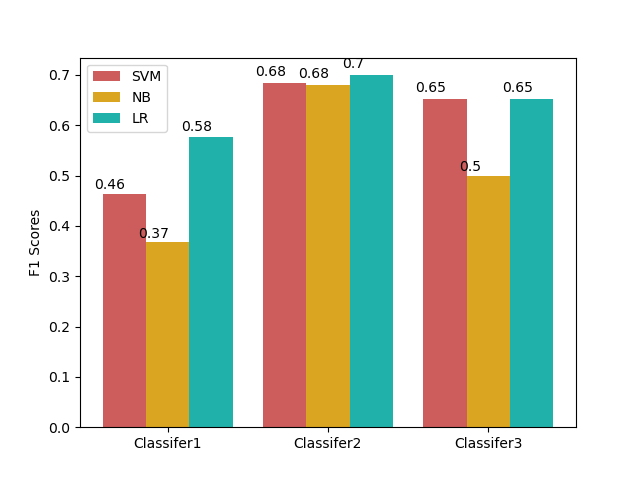
**5.2 Graphs**



**Fig 5.2.1 Precision graph**



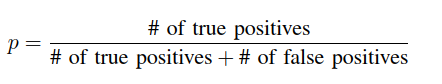
**Fig 5.2.2 Recall graph**



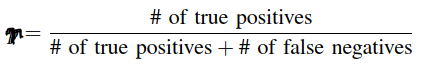
**Fig 5.2.3 F1 score graph**

The metrics employed to measure the effectiveness of the classiﬁer systems are the precision, recall and F1 value for multi-class and multi-tag classiﬁcation. We chose to not use ROC (Receiver Operating Characteristic) curves because the evaluation metric employed for the competition was not ROC curves, and the representation of ROC curves for multi-class classiﬁers is not as intuitive as their binary classiﬁcation

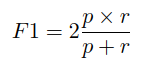
counterpart. Precision is deﬁned as the proportion of positive results that are truly positive. It is expressed as:



Recall measures the proportion of existing positives that are actually identiﬁed as positives.

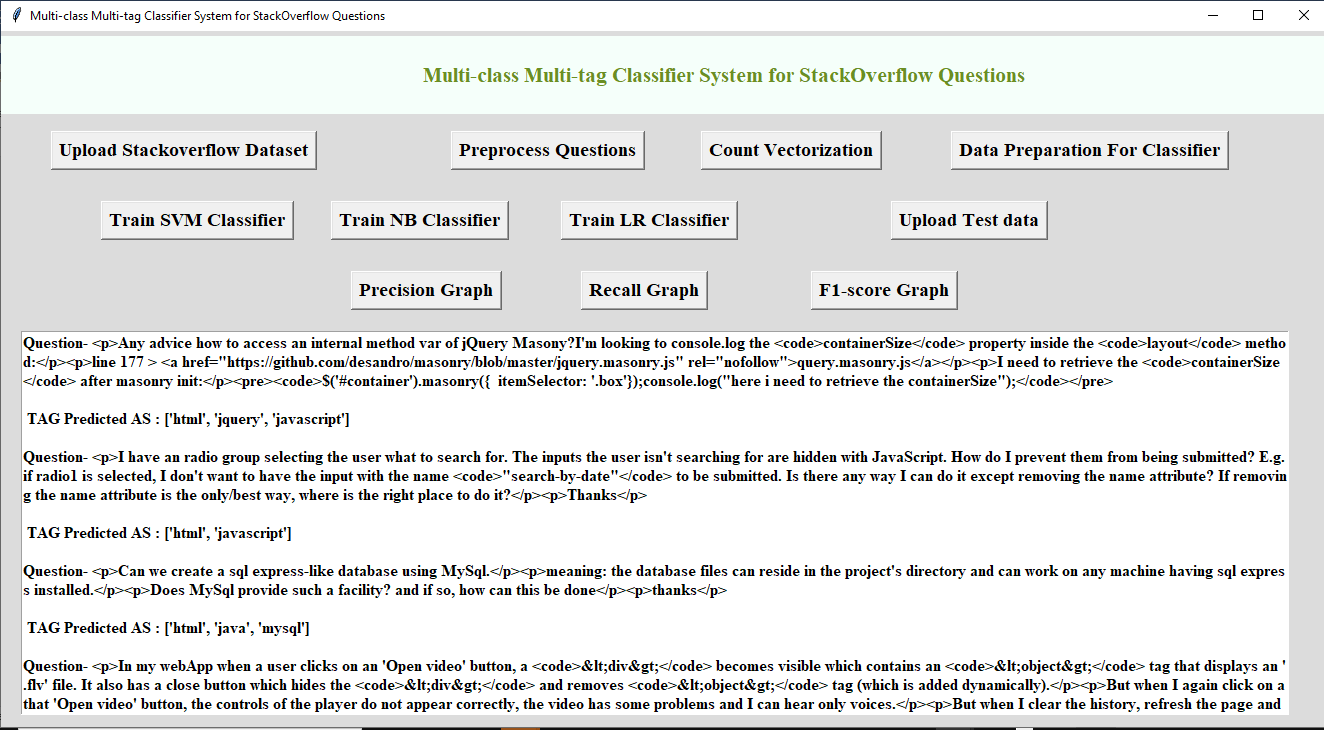


The F1 value is given by below Equation.



where p is the precision and r is the recall. The F1 value can be interpreted as the harmonic mean of the precision and recall where the best value is when F1 = 1 and worse when tends to 0. This metric is employed in the classiﬁcation problem because it weights equally both the precision and recall, which a good classiﬁer will try to maximize simultaneously, which means that the classiﬁer is, besides of being accurate, it also limits accordingly its predictions.

**5.3 Output Screenshot**



**Fig 5.3.1 Output**

**6. CONCLUSION AND FUTURE SCOPE**

This work shows that we can achieve good results using a simple classifying system, based on publicly available scientiﬁc libraries. This proposal presents certain advantages that are not available in other schemes. One of these advantages is that classiﬁers can be tuned respect to the tag they try to predict. That is, if it is known that there is high probability of a tag to appear in a position, it is possible to train the classiﬁer with certain bias, which is not possible to attain with the existing developments.

Also, most notably, the classifying systems presented are quite simple compared to other works for the same problem, where more computer power was needed to make and prepare their predictions; and with this approach it did not required more than easily available statistical tools.

**7. REFERENCES**

* [1] A. Goldbloom, “Kaggle,” <http://www.kaggle.com/c/> facebook-recruiting-iii-keyword-extraction, 2013, [Online; accessed 13-November-2013].
* [2] J. Attwood, “StackOverﬂow,” http://www.stackoverﬂow.com, 2013, [Online; accessed 13-November-2013].
* [3] J. Hong and M. Fang, “Keyword extraction and semantic tag predic- tion,” 2013.
* [4] A. Rajaraman and J. D. Ullman, Mining of massive datasets. Cam- bridge University Press, 2011.
* [5] C. Stanley and M. D. Byrne, “Predicting tags for stackoverﬂow posts,” in Proceedings of ICCM, 2013.
* [6] C. Parikh, “Identifying tags from millions of text question,” 2013.
* [7] Z. S. Harris, “Distributional structure.” Word, 1954.
* [8] J. B. Lovins, Development of a stemming algorithm. MIT Information Processing Group, Electronic Systems Laboratory, 1968.
* [9] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” The Journal of Machine Learning Research, vol. 3, pp. 1157– 1182, 2003.
* [10] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, “1-norm support vector machines,” Advances in neural information processing systems, vol. 16, no. 1, pp. 49–56, 2004