B. TECH (CSE(AI)) (2021)

AMRITA VISHWA VIDYAPEETHAM, COIMBATORE

19MAT204 MATHEMATICS FOR INTELLIGENT SYSTEMS-3

PROJECT REPORT

GROUP-06 TEAM MEMBERS:

- M.Visweswaran [CB.EN. U4AIE20075]
- Vishnu Radhakrishnan [CB.EN. U4AIE20074]
- Thushit Kumar R [CB.EN. U4AIE20072]
- Menta Sai Akshay [CB.EN. U4AIE20040]
- Krishnan K M [CB.EN. U4AIE20031]

PROJECT TOPIC: Emoji Predictor using Least Square SVM

PROJECT REPORT SUBMITTED FOR THE END SEMESTER EXAMINATION OF 19MAT204

EXTERNAL EXAMINER INTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to thank all those who have helped us in completing this project of "EMOJI DETECTOR USING LEAST SQUARE SVM" under the subject "MATHEMATICS FOR INTELLIGENT SYSTEMS-3".

We would like to show our sincere gratitude to our professor NEETHU MOHAN without whom the project would not have been initiated, who taught us the basics to start and visualize the project and enlightened us with the ideas regarding the project, and helped us by clarifying all the doubts whenever being asked.

We would like to thank ourselves. We helped each other and taught each other about various concepts regarding the project which helped in increasing our knowledge.

TABLE OF CONTENTS:

Introduction:

Communication is a key component for developing a positive relationship among others. It includes the verbal sounds to nonverbal actions and facial reactions. It is a process of exchanging ideas and information. It also opens a whole new dimension of understanding a person’s character and emotional state. Yet, there may be a lot of mediums for communication, but all serve the same purpose. Recently, there has occurred a big revolution in digital space. Text-based communication has become a common medium for communication, but it deprived us of the opportunity to read facial expressions. Due to this, there was an emptiness in all digital communication. Eventually, we address the issue by introducing the new concept called “Emoji”, which was introduced in the 1990s which became popular thereafter in the era of social media.

Abstract:

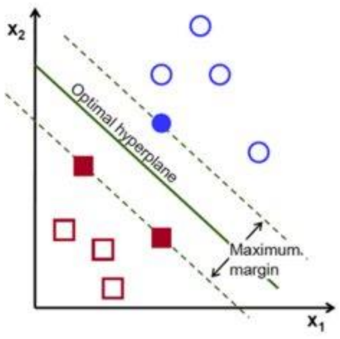
We have taken up a small project based on the emoji predictor that assigns the predicted emoji in a conversation. This project takes in the particular conversation sentence as input and predicts the emoji-based upon the given content. (In our case the content is the sentence). We will be using SVM classifier for achieving our goal. We will compare different SVM procedures like L1, L2. Based upon the given conditions we will be selecting the optimal kernel function and also the hyperparameters.

Theory:

The SVM is a popular model for solving pattern recognition problems. In this method, we will be constructing a hyperplane that separates two classes. In the case of nonlinearity, we will also be introducing a concept that maps the given dataset into a high-dimensional input space.

A **Support Vector Machine (SVM)** is a discriminative classifier that intakes training data and the algorithm outputs an optimal hyperplane that categorizes the data.

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.



Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

L1 SVM:

Support vector machines with the linear sum of slack variables, which are commonly used are called L1-SVM's.

Let training data be x_i (i = 1, ..., M) and its label be y_i =1, if xi belongs to Class 1, and y_i = −1 if Class 2.To obtain the optimal separating hyperplane of the L1-SVM in the feature space, we consider the following optimization problem:

$$\begin{array}{rc} \text{minimize} & \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C\sum_{i=1}^{M} \xi_i, \\ \text{subject to} & y_i(\mathbf{w}^t \mathbf{g}(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ \text{for} & i = 1, ..., M, \end{array}$$

Where,

w is a weight vector**,**

C is the margin parameter that determines the trade-off between the maximization of the margin and the minimization of the classification error,

ξ_i (i = 1, ..., M) are the non-negative slack variables **,**

b is the bias term.

On Introducing the Lagrange multipliers α_i, we obtain the following dual problem,

$$\text{maximize}$$

$$Q(\alpha) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j \mathbf{g}(\mathbf{x}_i)^t \mathbf{g}(\mathbf{x}_j),$$

$$\text{subject to} \quad \sum_{i=1}^{M} y_i \alpha_i = 0, \quad 0 \le \alpha_i \le C.$$

We use the mapping function that satisfies

$$H(\mathbf{x}, \mathbf{x}') = \mathbf{g}(\mathbf{x})^t \mathbf{g}(\mathbf{x}')$$

$H(\mathbf{x}, \mathbf{x}') \Longrightarrow$ Kernel Function

Solving the above dual problem, we obtain the decision function:

$$D(\mathbf{x}) = \sum_{i=1}^{M} \alpha_i^* y_i H(\mathbf{x}_i, \mathbf{x}) + b^*,$$

L2 SVM:

L2-SVMs use the square sum of the slack variables ξ_i in the objective function instead of the linear sum of the slack variables.we consider the optimization problem as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \| \mathbf{w} \|^2 + \frac{C}{2} \sum_{i=1}^{M} \xi_i^2, \\ \text{subject to} \quad & y_i(\mathbf{w}^t \mathbf{x}_i + b) \ge 1 - \xi_i, \\ \text{for} \quad & i = 1, \dots, M. \end{aligned}$$

Introducing the Lagrange multipliers α_i, we obtain the dual problem:

$$\text{maximize}$$

$$Q(\alpha) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} y_i y_j \alpha_i \alpha_j \left(H(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{ij}}{C} \right)$$

$$\text{subject to}$$

$$\sum_{i=1}^{M} y_i \alpha_i = 0, \quad \alpha_i \ge 0 \quad \text{for} \quad i = 1, \dots, M,$$

δ_{ij} is Kronecker’s delta function, in which $\delta_{ij} = 1$ for i = j and 0, otherwise.

DIFFERENCES:

L1 SVM:

- Support vector machines with the linear sum of slack variables, which are commonly used, are called L1-SVMs.
- Objective Function:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\parallel \mathbf{w} \parallel^2 + C\sum_{i=1}^{M}\xi_i, \\ &\text{subject to} && y_i(\mathbf{w}^t\mathbf{g}(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ &\text{for} && i = 1, \ldots, M, \end{aligned}$$

- KKT CONDITIONS

$$\begin{aligned} \frac{\partial Q}{\partial \alpha_i} = (F_i + \beta)y_i + \delta_i - \mu_i = 0, \\ \delta_i \alpha_i = 0, \quad \delta_i \geq 0, \\ \mu_i(\alpha_i - C) = 0, \quad \mu_i \geq 0, \quad \text{for} \quad i = 1, \ldots, M, \end{aligned}$$

L2 SVM:

- Support Vector Machines with the squaresum of slack variables are called L2-SVMs.
- Objective Function:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\parallel \mathbf{w} \parallel^2 + \frac{C}{2}\sum_{i=1}^{M}\xi_i^2, \\ &\text{subject to} && y_i(\mathbf{w}^t\mathbf{x}_i + b) \geq 1 - \xi_i, \\ &\text{for} && i = 1, \ldots, M. \end{aligned}$$

- KKT CONDITIONS:

$$y_i\left(\sum_{j=1}^{M}\alpha_j^* y_j\left(H(\mathbf{x}_j, \mathbf{x}_i) + \frac{\delta_{ij}}{C}\right) + b^*\right) - 1 = 0.$$

Least Square SVM:

The least Square Support vector machines (LS-SVM) are the least square versions of support vector machines (SVM).

$$L(W,b) = \sum_{i=1}^{N} \| f(x_i) - y_i \|_2^2 + C \| W \|_2^2$$

The main objective of LS-SVM is to apply minimization of the sum of squared errors to the objective function.LS-SVM is used for classification and regression analysis and is also a class of Kernel-based learning methods.

Least squares version to the SVM classifier by formulating the classification problem as

$$\min_{\omega,b,e} J_3(\omega, b, e) = \frac{1}{2}\omega^T\omega + \gamma\frac{1}{2}\sum_{k=1}^{N} e_k^2$$

Subjected to the equality constraints

$$y_k[w^T\phi(x_k) + b] = 1 - e_k, k = 1, \ldots, N.$$

Lagrangian is defined as

$$L_3(\omega, b, e, \alpha) = J_3(\omega, b, e) - \sum_{k=1}^{N} \alpha_k\{y_k[\omega^T\phi(x_k) + b] - 1 + e_k\}$$

Optimality Conditions:

$$\begin{cases} \frac{\partial \mathcal{L}_3}{\partial w} = 0 \rightarrow w = \sum_{k=1}^{N} \alpha_k y_k \varphi(x_k) \\ \frac{\partial \mathcal{L}_3}{\partial b} = 0 \rightarrow \sum_{k=1}^{N} \alpha_k y_k = 0 \\ \frac{\partial \mathcal{L}_3}{\partial e_k} = 0 \rightarrow \alpha_k = \gamma e_k, \qquad k = 1, \ldots, N \\ \frac{\partial \mathcal{L}_3}{\partial \alpha_k} = 0 \rightarrow y_k[w^T\varphi(x_k) + b] - 1 + e_k = 0, \quad k = 1, \ldots, N \end{cases}$$

The above equations can be written as the solution for the following set of Linear Equations

$$\left[\begin{array}{ccc|c} I & 0 & 0 & -Z^T \\ 0 & 0 & 0 & -Y^T \\ 0 & 0 & \gamma I & -I \\ \hline Z & Y & I & 0 \end{array}\right] \left[\begin{array}{c} w \\ b \\ e \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline \vec{1} \end{array}\right] \quad \begin{array}{l} Z = [\phi(x_1)^T y_1; \ldots \phi(x_N)^T y_N] \\ Y = [y_1; \ldots; y_N] \\ \vec{1} = [1; \ldots 1] \\ e = [e_1, \ldots e_N], \\ \alpha = [\alpha_1, \ldots \alpha_N] \end{array}$$

The solution is given by,

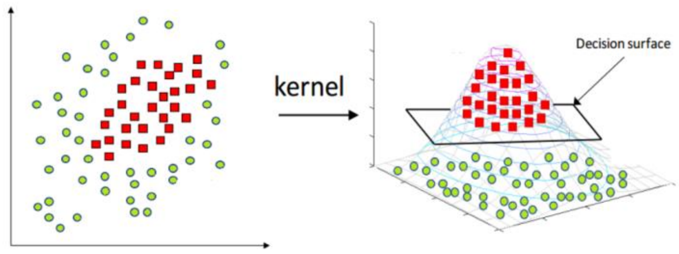
$$\left[\begin{array}{c|c} 0 & -Y^T \\ \hline Y & ZZ^T + \gamma^{-1} I \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \vec{1} \end{array}\right].$$

After applying Mercer's Solution to the matrix Ω = ZZ^T

$$\begin{aligned} \Omega_{kl} &= y_k y_l \varphi(x_k)^T \varphi(x_l) \\ &= y_k y_l \psi(x_k, x_l). \end{aligned}$$

Hence the classifier is found by solving the linear set of equations instead of quadratic programming and the support values αk are proportional to the errors at the data points.

Kernel Trick



The major part of SVM, Kernel trick. A kernel is a way of computing the dot product of two vectors x and y in higher dimensional feature space. Applying Kernel means replacing the dot product of two vectors with the Kernel function.

Types of kernel functions

- Linear

$$K(x, x') = \langle \phi(x), \phi(x') \rangle$$

- Polynomial

Commonly uses degree 2, Larger than 2 causes overfitting
The polynomial kernel suffers from numerical instability

$$K(x,y) = (x^\mathsf{T} y + c)^d$$

- RBF

Radial basis function Kernel is a popular Kernel method.
Here the value depends on the distance from the origin or some point.
Using the distance in the original space we calculate the dot product.

$$K(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$$

Kernel Function

SVM can map the input vector into a high-dimensional feature space. This can be done when a linear boundary is inappropriate. Hereby selecting the non-linear mapping as a priori, SVM constructs an optimal separating hyperplane in this higher dimension.
So basically, the kernel function helps to enable operations to be done in input space rather than the higher dimensional space.

So here some theory suggests that the inner product in feature space has an equivalent Kernel in input space provided certain conditions in the hold.

$$K(x,x') = \langle \phi(x), \phi(x') \rangle$$

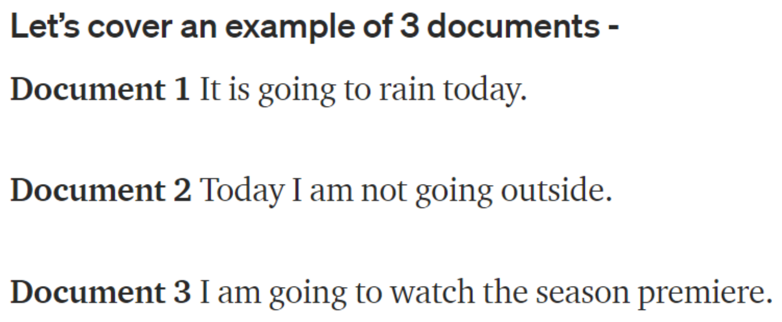
From this Gaussian radial basis function (G-RBF) has received significant attention and it is given by

$$K(x,x') = \mathrm{e}^{-\|x-x'\|^2/2\sigma^2}$$

Here generally, using RBF helps in some method of determining a subset of centers, where usually we use a method of clustering to select a subset of centers.
So, the process of SVM in this implicit selection process is its main focusing feature were each support vector contributes to local Gaussian functions and it is centered at that data point and also the principle of SRM can be used to select global basis function width.

TFIDF Vectorizer

It is one of the text-based data feature extraction methods. When compared to the basic token count vectorization method, TFIDF focuses on meaningful tokens rather than the overall count-based tokens. (Token = Unique word in the document). It emphasizes the frequency of the tokens and also on the uniqueness of tokens on comparison between documents. In layman's terms, more common terms in a document have lesser importance whereas less frequent occurring tokens are given more weightage.



Let's cover an example of 3 documents -

Document 1 It is going to rain today.

Document 2 Today I am not going outside.

Document 3 I am going to watch the season premiere.

First, we can start finding the frequency of occurrence of each token in each of the documents.

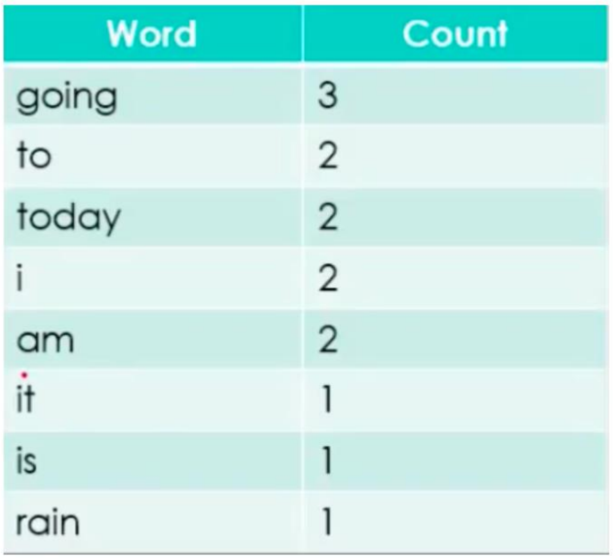
For finding the TF we can apply the following formula

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

n_{ij}= No of ith token in jth document

$\sum_k nij$ = Total number of ith token in the jth document

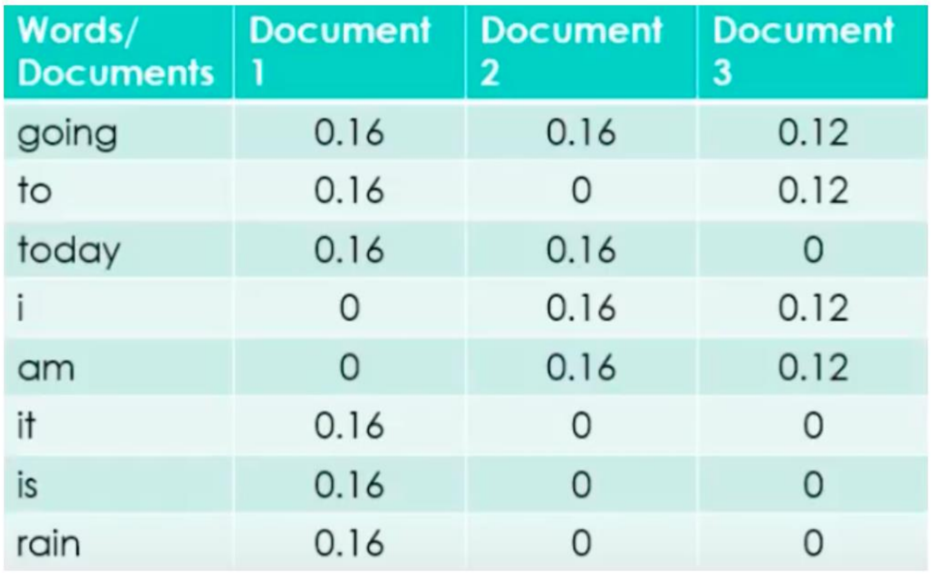
The below table represents the total number of each token in all documents.



Word	Count
going	3
to	2
today	2
i	2
am	2
it	1
is	1
rain	1

Vocab of document

On applying the above equation we get the following TF table

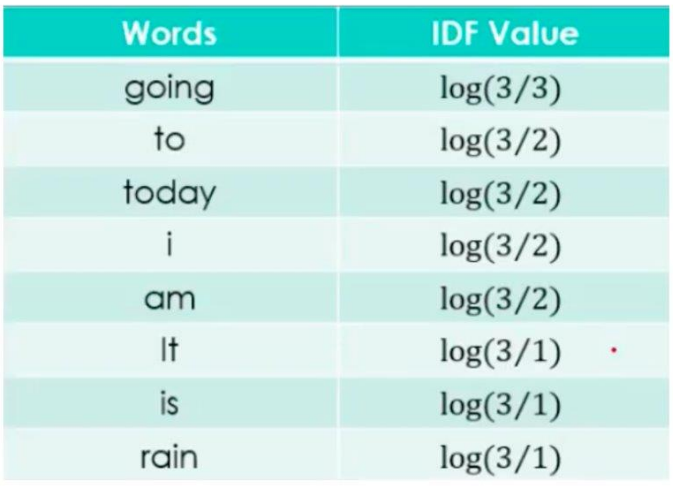


Words/ Documents	Document 1	Document 2	Document 3
going	0.16	0.16	0.12
to	0.16	0	0.12
today	0.16	0.16	0
i	0	0.16	0.12
am	0	0.16	0.12
it	0.16	0	0
is	0.16	0	0
rain	0.16	0	0

Now, we have to find the inverse document frequency, it tells us about all the documents that consist of a given particular token.

$$idf(w) = log(\frac{N}{df_t})$$

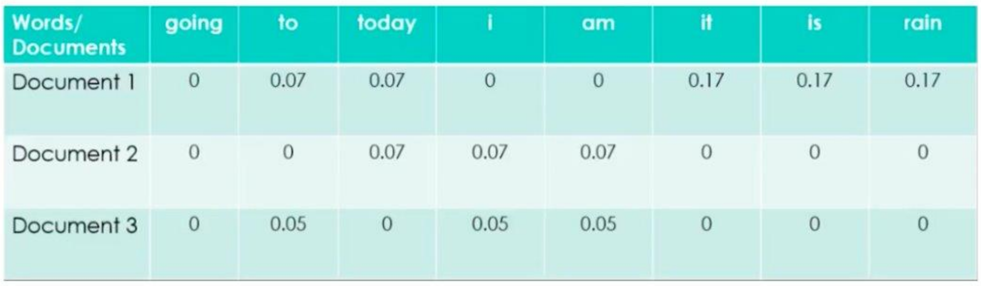
On applying the following formula we get this table



Words	IDF Value
going	log(3/3)
to	log(3/2)
today	log(3/2)
i	log(3/2)
am	log(3/2)
It	log(3/1)
is	log(3/1)
rain	log(3/1)

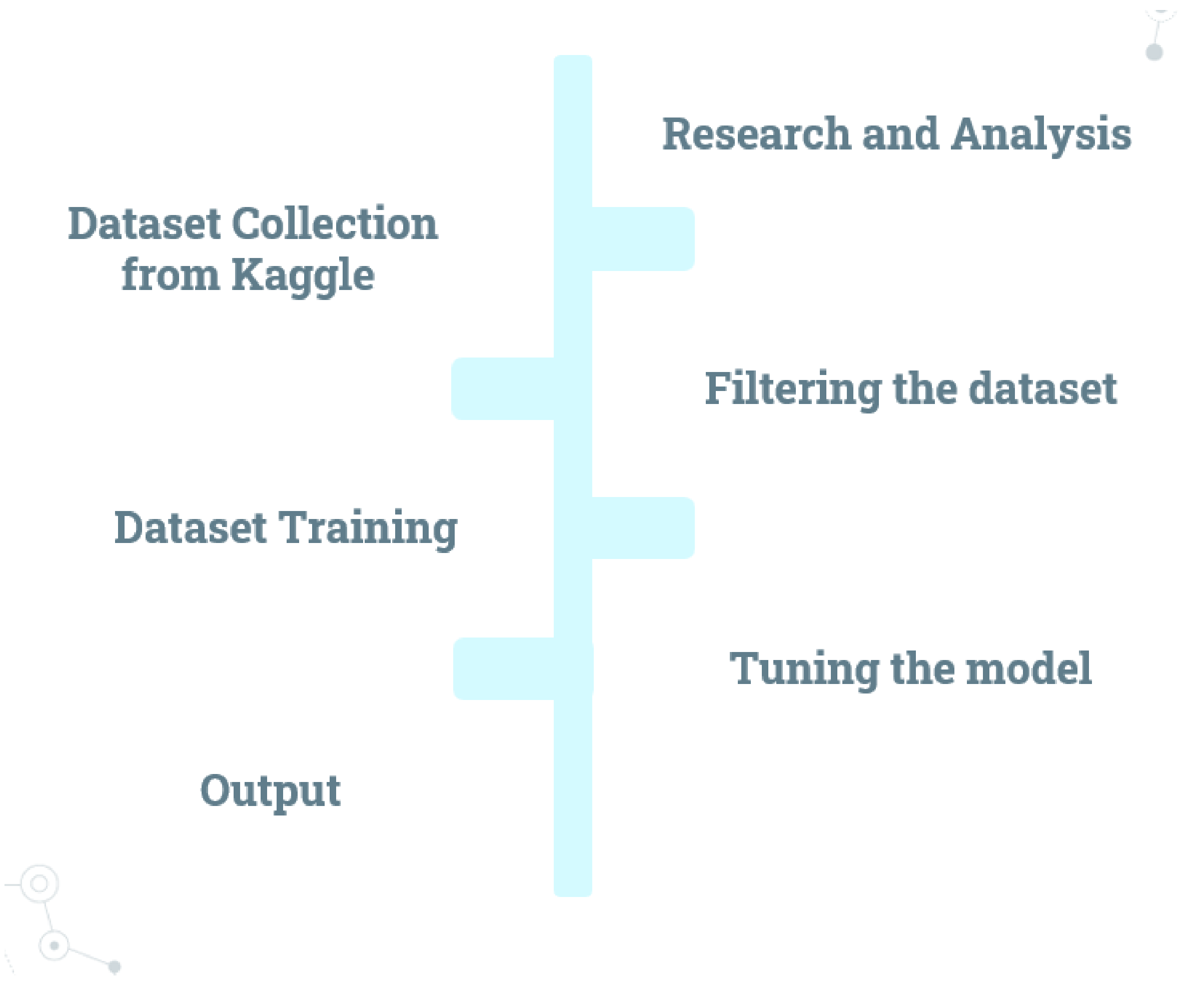
For calculating the TF, IDF weight we will be multiplying the values based on the formula

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$



Words/ Documents	going	to	today	i	am	it	is	rain
Document 1	0	0.07	0.07	0	0	0.17	0.17	0.17
Document 2	0	0	0.07	0.07	0.07	0	0	0
Document 3	0	0.05	0	0.05	0.05	0	0	0

Project Timeline

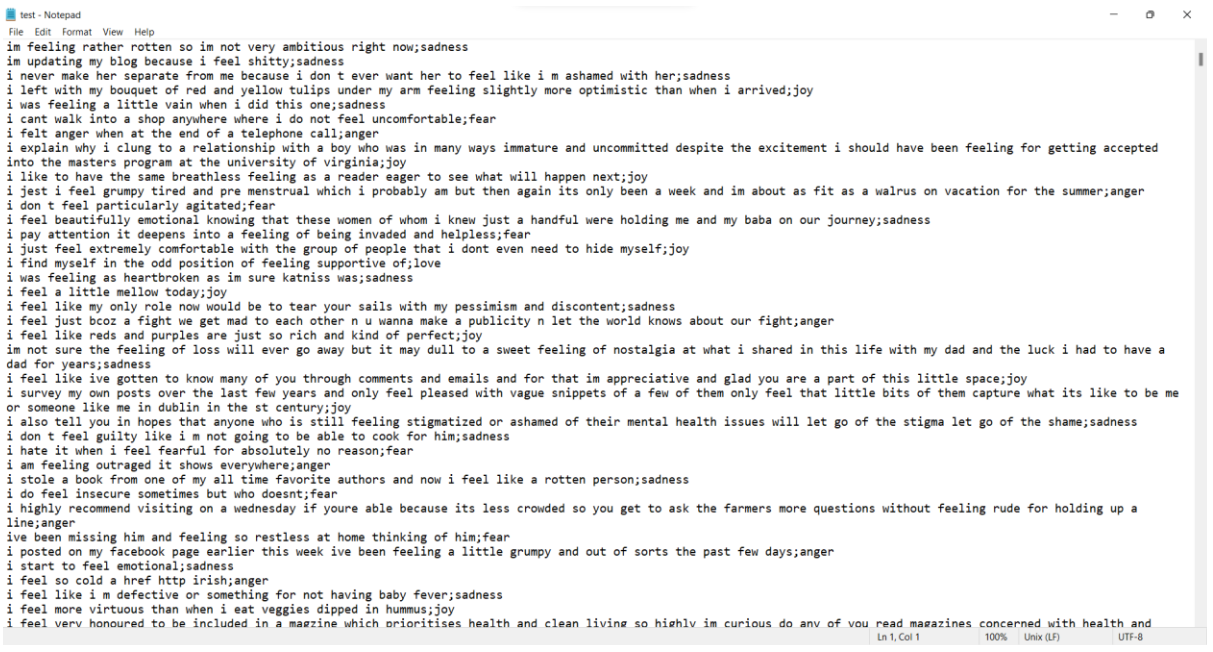


Dataset Collection from Kaggle:

To collect the dataset of conversion to train our SVM model, we have taken the public dataset from Kaggle to understand the emotions of our general conversation in social media applications.

Following is our dataset [Emotions dataset for NLP | Kaggle]

An Example of the training data is given below



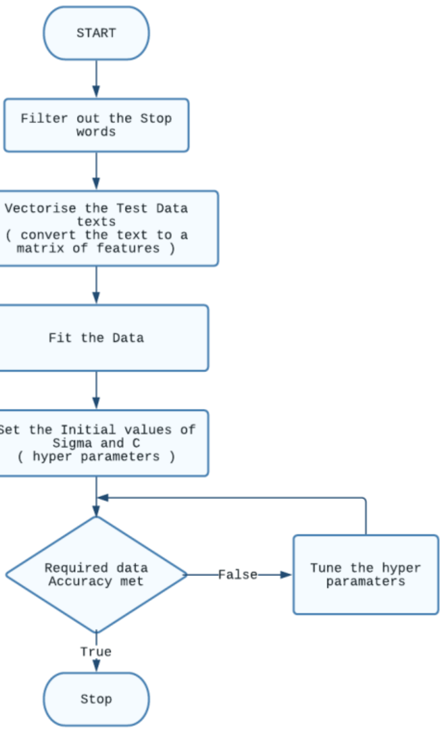
```
im feeling rather rotten so im not very ambitious right now;sadness
im updating my blog because i feel shitty;sadness
i never make her separate from me because i don t ever want her to feel like i m ashamed with her;sadness
i left with my bouquet of red and yellow tulips under my arm feeling slightly more optimistic than when i arrived;joy
i was feeling a little vain when i did this one;sadness
i cant walk into a shop anywhere where i do not feel uncomfortable;fear
i felt anger when at the end of a telephone call;anger
i explain why i clung to a relationship with a boy who was in many ways immature and uncommitted despite the excitement i should have been feeling for getting accepted
into the masters program at the university of virginia;joy
i like to have the same breathless feeling as a reader eager to see what will happen next;joy
i jest i feel grumpy tired and pre menstrual which i probably am but then again its only been a week and im about as fit as a walrus on vacation for the summer;anger
i don t feel particularly agitated;fear
i feel beautifully emotional knowing that these women of whom i knew just a handful were holding me and my baba on our journey;sadness
i pay attention it deepens into a feeling of being invaded and helpless;fear
i just feel extremely comfortable with the group of people that i dont even need to hide myself;joy
i find myself in the odd position of feeling supportive of;love
i was feeling as heartbroken as im sure katniss was;sadness
i feel a little mellow today;joy
i feel like my only role now would be to tear your sails with my pessimism and discontent;sadness
i feel just bcoz a fight we get mad to each other n u wanna make a publicity n let the world knows about our fight;anger
i feel like reds and purples are just so rich and kind of perfect;joy
im not sure the feeling of loss will ever go away but it may dull to a sweet feeling of nostalgia at what i shared in this life with my dad and the luck i had to have a
dad for years;sadness
i feel like ive gotten to know many of you through comments and emails and for that im appreciative and glad you are a part of this little space;joy
i survey my own posts over the last few years and only feel pleased with vague snippets of a few of them only feel that little bits of them capture what its like to be me
or someone like me in dublin in the st century;joy
i also tell you in hopes that anyone who is still feeling stigmatized or ashamed of their mental health issues will let go of the stigma let go of the shame;sadness
i don t feel guilty like i m not going to be able to cook for him;sadness
i hate it when i feel fearful for absolutely no reason;fear
i am feeling outraged it shows everywhere;anger
i stole a book from one of my all time favorite authors and now i feel like a rotten person;sadness
i do feel insecure sometimes but who doesnt;fear
i highly recommend visiting on a wednesday if youre able because its less crowded so you get to ask the farmers more questions without feeling rude for holding up a
line;anger
ive been missing him and feeling so restless at home thinking of him;fear
i posted on my facebook page earlier this week ive been feeling a little grumpy and out of sorts the past few days;anger
i start to feel emotional;sadness
i feel so cold a href http irish;anger
i feel like i m defective or something for not having baby fever;sadness
i feel more virtuous than when i eat veggies dipped in hummus;joy
i feel very honoured to be included in a magzine which prioritises health and clean living so highly im curious do any of you read magazines concerned with health and
```

Filtering the dataset

To refine the dataset by extracting the important emotion conveying feature, we filter the dataset to remove the stopwords and perform vectorization.

Stopwords are highly occurring words like ‘and’, ‘or’,’ but’, ’I’…etc. that do not convey any emotion, these can be considered as unwanted noises in our dataset and can be removed.

Dataset Training


START
Filter out the Stop words
Vectorise the Test Data texts (convert the text to a matrix of features)
Fit the Data
Set the Initial values of Sigma and C (hyper parameters)
Required data Accuracy met
False
Tune the hyper paramaters
True
Stop

Tuning the model

We have to perform parameter tuning to get the optimum hyperparameters to get maximum efficiency such that it suits real-world situations well.

Initially, a set of C and sigma values were chosen and according to it the accuracy was calculated and plotted as shown in the graphs below.

According to the range in which the maximum accuracy was obtained after one iteration of accuracy calculation, the C and Sigma values were again changed (nearer to the values yielding maximum accuracy) and the set corresponding to maximum accuracy was then determined.

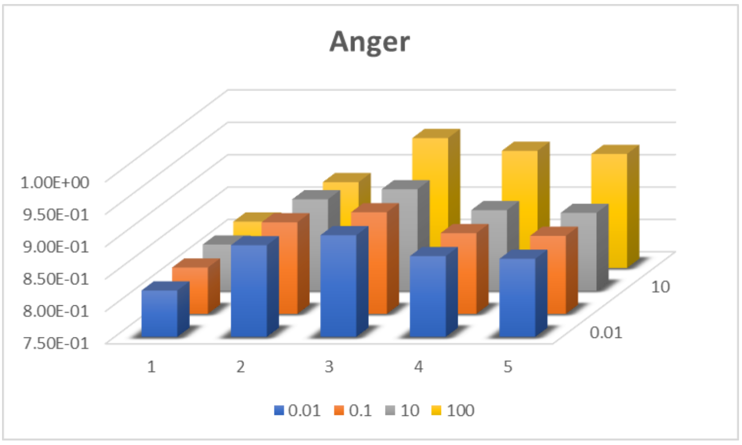
This will be performed for all the 6 emotions we have taken.

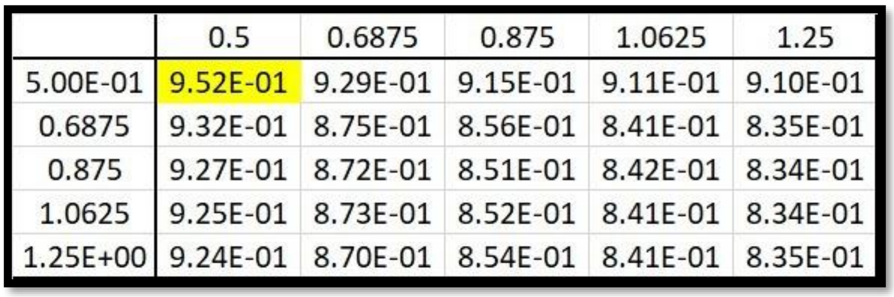
Following is the observation

Here the yellow box corresponds to the maximum value of accuracy and the first column represents the C values and the first row represents the Sigma values.

Anger

The accuracy of the model was plotted for corresponding C and Sigma values pair





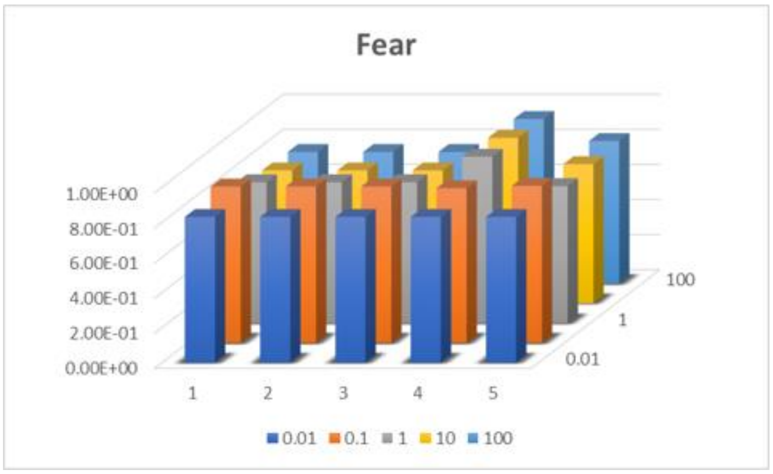
	0.5	0.6875	0.875	1.0625	1.25
5.00E-01	9.52E-01	9.29E-01	9.15E-01	9.11E-01	9.10E-01
0.6875	9.32E-01	8.75E-01	8.56E-01	8.41E-01	8.35E-01
0.875	9.27E-01	8.72E-01	8.51E-01	8.42E-01	8.34E-01
1.0625	9.25E-01	8.73E-01	8.52E-01	8.41E-01	8.34E-01
1.25E+00	9.24E-01	8.70E-01	8.54E-01	8.41E-01	8.35E-01

X-axis: Sigma value

Y series: Corresponding to the given C values in the bottom

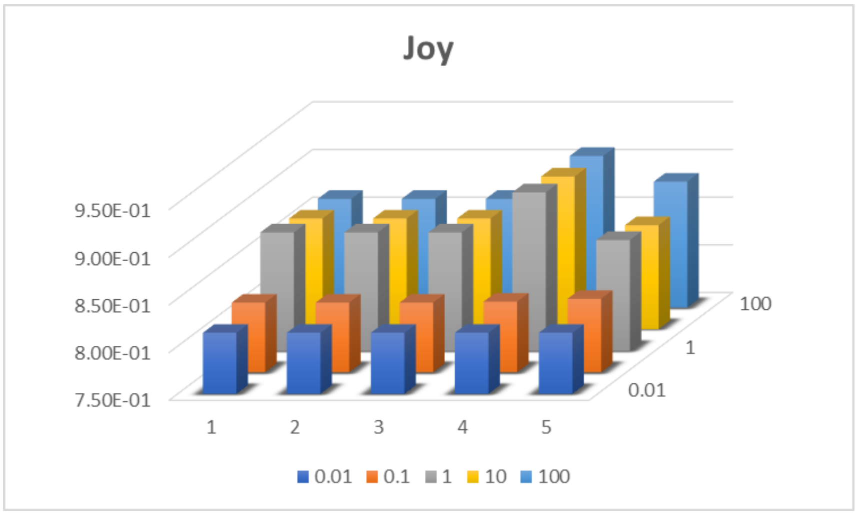
Z-axis: accuracy of the Model

Fear



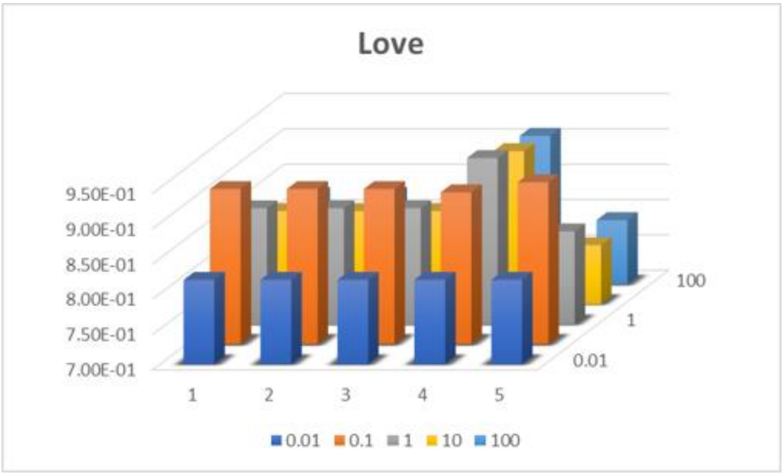
	0.5	0.6875	0.875	1.0625	1.25
5.00E-01	9.30E-01	9.44E-01	9.44E-01	9.44E-01	9.44E-01
0.6875	9.28E-01	9.47E-01	9.50E-01	9.47E-01	9.46E-01
0.875	9.27E-01	9.46E-01	9.48E-01	9.49E-01	9.46E-01
1.0625	9.25E-01	9.46E-01	9.50E-01	9.49E-01	9.44E-01
1.25E+00	9.25E-01	9.46E-01	9.48E-01	9.47E-01	9.44E-01

Joy



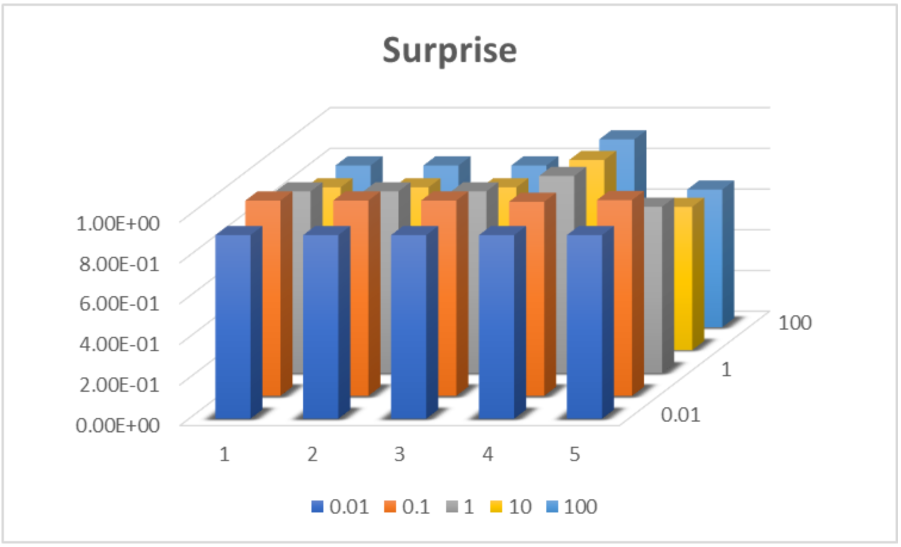
	0.8	0.85	0.9	0.95	1
8.00E-01	9.13E-01	9.14E-01	9.14E-01	9.17E-01	9.18E-01
0.85	9.15E-01	9.15E-01	9.18E-01	9.18E-01	9.16E-01
0.9	9.16E-01	9.17E-01	9.18E-01	9.19E-01	9.18E-01
0.95	9.18E-01	9.18E-01	9.18E-01	9.18E-01	9.16E-01
1.00E+00	9.18E-01	9.19E-01	9.18E-01	9.18E-01	9.16E-01

Love



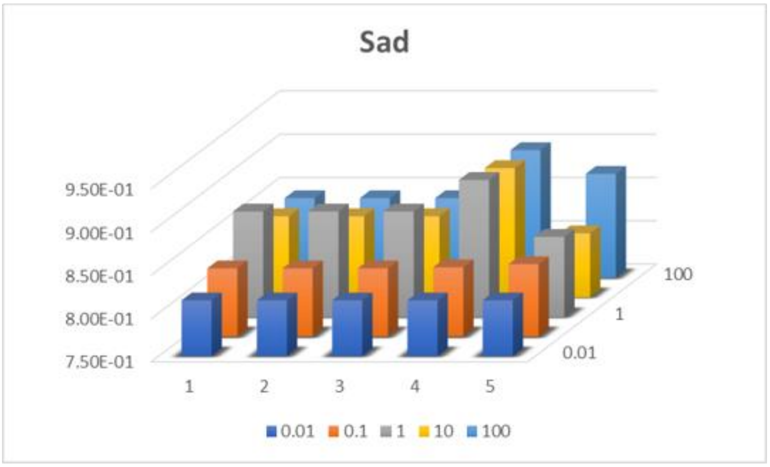
	0.1	0.45	0.8	0.15	1.5
5.00E-01	9.15E-01	9.13E-01	9.13E-01	9.16E-01	9.17E-01
0.6875	9.33E-01	9.50E-01	9.52E-01	9.47E-01	9.43E-01
0.875	9.20E-01	9.44E-01	9.42E-01	9.36E-01	9.27E-01
1.0625	9.13E-01	9.39E-01	9.37E-01	9.29E-01	9.25E-01
1.25E+00	9.10E-01	9.35E-01	9.34E-01	9.30E-01	9.23E-01

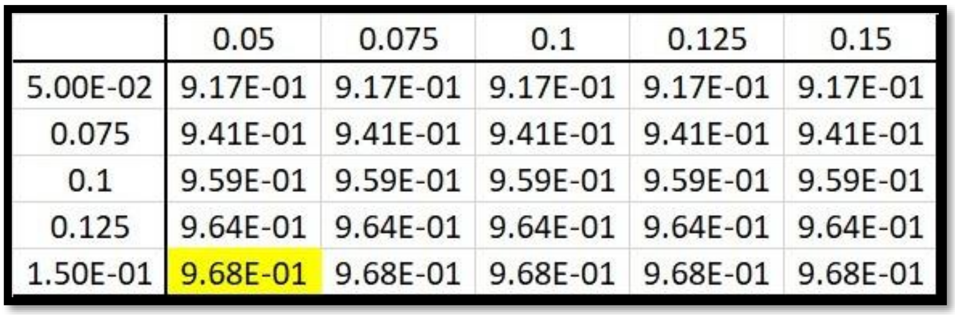
Surprise



	0.5	0.6875	0.875	1.0625	1.5
5.00E-01	8.99E-01	9.08E-01	9.09E-01	9.09E-01	9.08E-01
0.75	9.06E-01	9.12E-01	9.10E-01	9.09E-01	9.06E-01
1	9.03E-01	9.11E-01	9.08E-01	9.03E-01	8.98E-01
1.25	9.04E-01	9.09E-01	9.06E-01	8.99E-01	8.95E-01
1.50E+00	9.03E-01	9.09E-01	9.03E-01	8.96E-01	8.94E-01

SAD

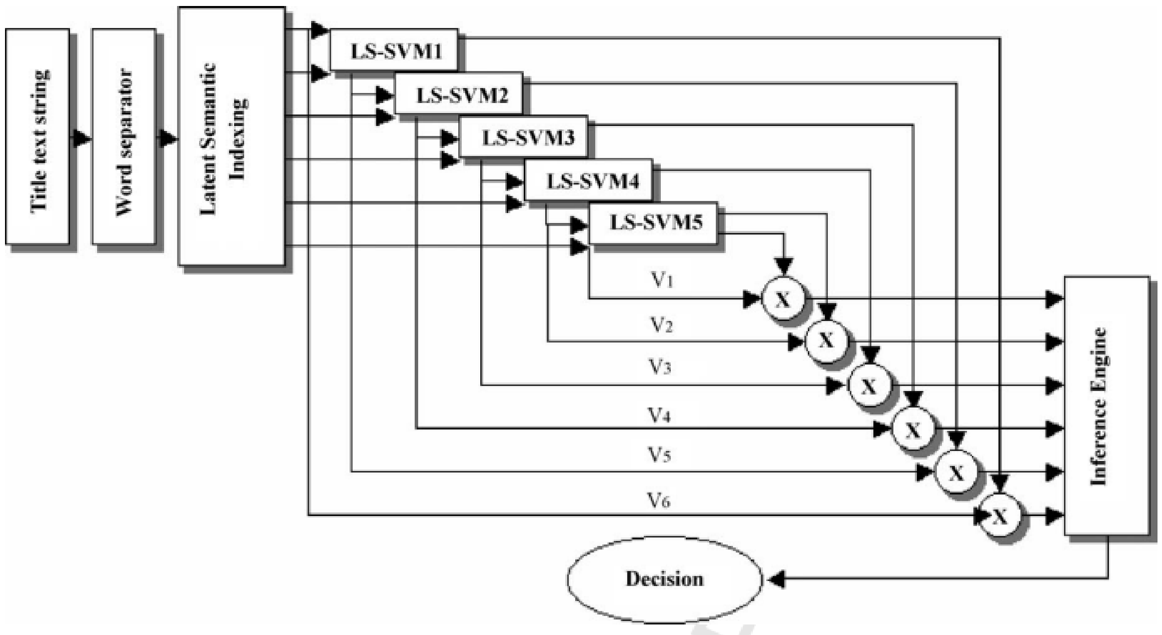




	0.05	0.075	0.1	0.125	0.15
5.00E-02	9.17E-01	9.17E-01	9.17E-01	9.17E-01	9.17E-01
0.075	9.41E-01	9.41E-01	9.41E-01	9.41E-01	9.41E-01
0.1	9.59E-01	9.59E-01	9.59E-01	9.59E-01	9.59E-01
0.125	9.64E-01	9.64E-01	9.64E-01	9.64E-01	9.64E-01
1.50E-01	9.68E-01	9.68E-01	9.68E-01	9.68E-01	9.68E-01

BASIC FLOWCHART

The basic idea that will be implemented is as shown in the flowchart.

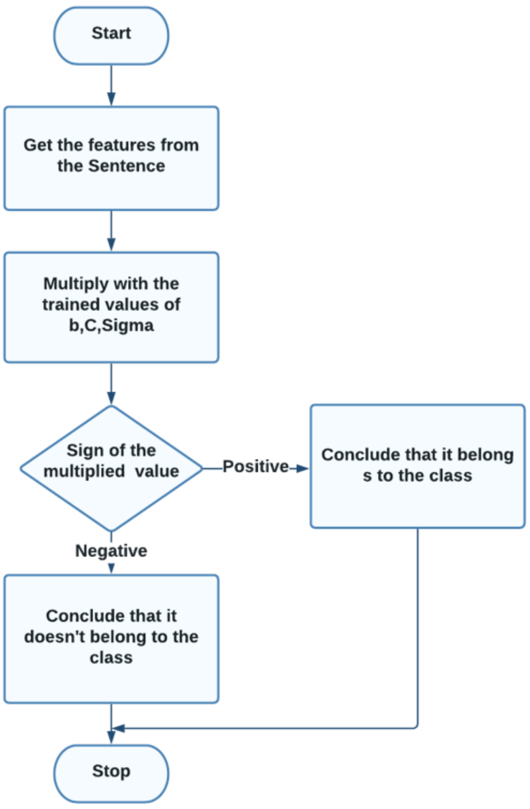


On analysing when given input (the given sentence), the input sentence is converted to feature using feature extraction methods (namely TFIDFVectorizer)

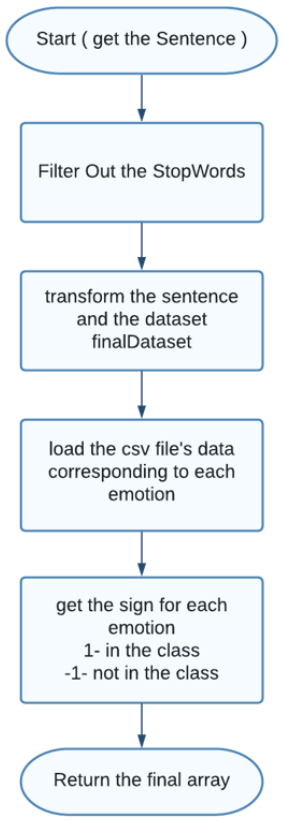
From the given Vector it is passed to each SVM model and the outcomes of these models are analysed. Each SVM models are tuned and trained based on each emotion. (Here we will be using 6 emotion models). These models give 1/-1 as outcomes, where 1 represents the particular emotion and -1 represents not an emotion.

Based upon the outcomes (+1/-1) we can predict the final respective emotion and pass it on to the respective environment/GUI.

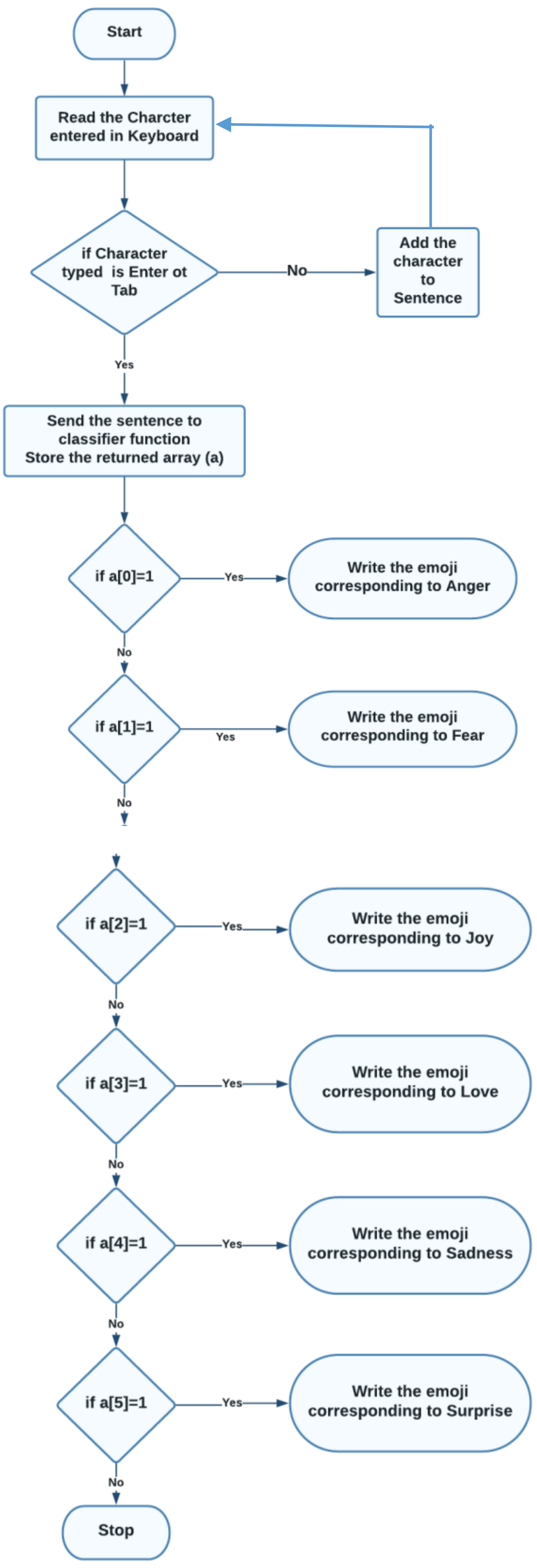
Thus the idea is refined furthermore and is given in the below flowchart.


Start
Get the features from the Sentence
Multiply with the trained values of b,C,Sigma
Sign of the multiplied value
Positive
Conclude that it belong s to the class
Negative
Conclude that it doesn't belong to the class
Stop

Classifier

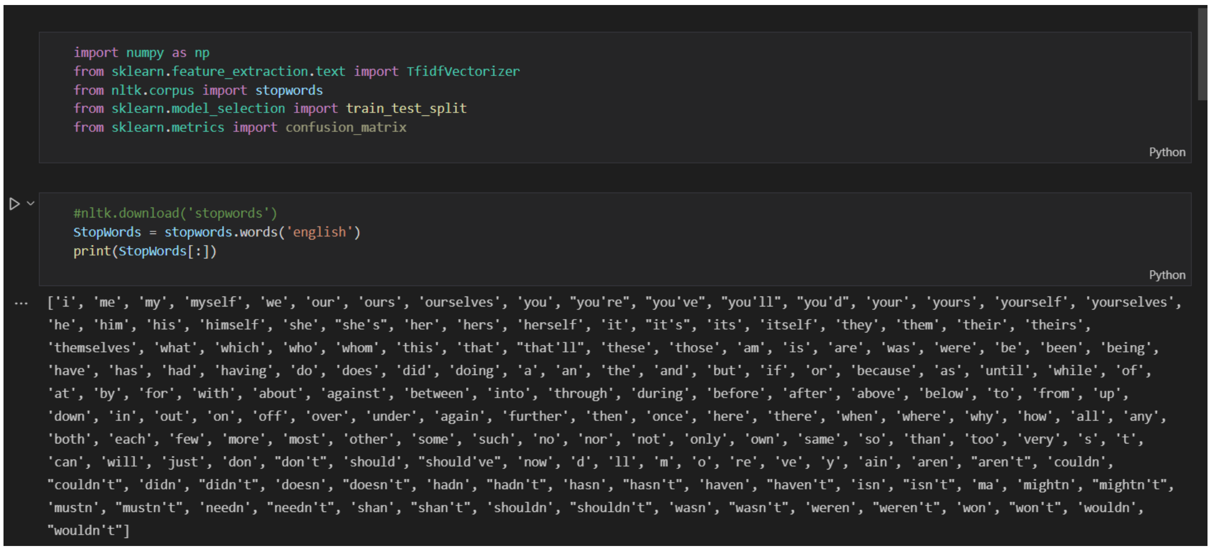

Start (get the Sentence)
Filter Out the StopWords
transform the sentence and the dataset finalDataset
load the csv file's data corresponding to each emotion
get the sign for each emotion
1- in the class
-1- not in the class
Return the final array

Backend


Start
Read the Charcter entered in Keyboard
if Character typed is Enter ot Tab
No
Add the character to Sentence
Yes
Send the sentence to classifier function Store the returned array (a)
if a[0]=1
Yes
Write the emoji corresponding to Anger
No
if a[1]=1
Yes
Write the emoji corresponding to Fear
No
if a[2]=1
Yes
Write the emoji corresponding to Joy
No
if a[3]=1
Yes
Write the emoji corresponding to Love
No
if a[4]=1
Yes
Write the emoji corresponding to Sadness
No
if a[5]=1
Yes
Write the emoji corresponding to Surprise
No
Stop

Codes

TainingModel.ipynb



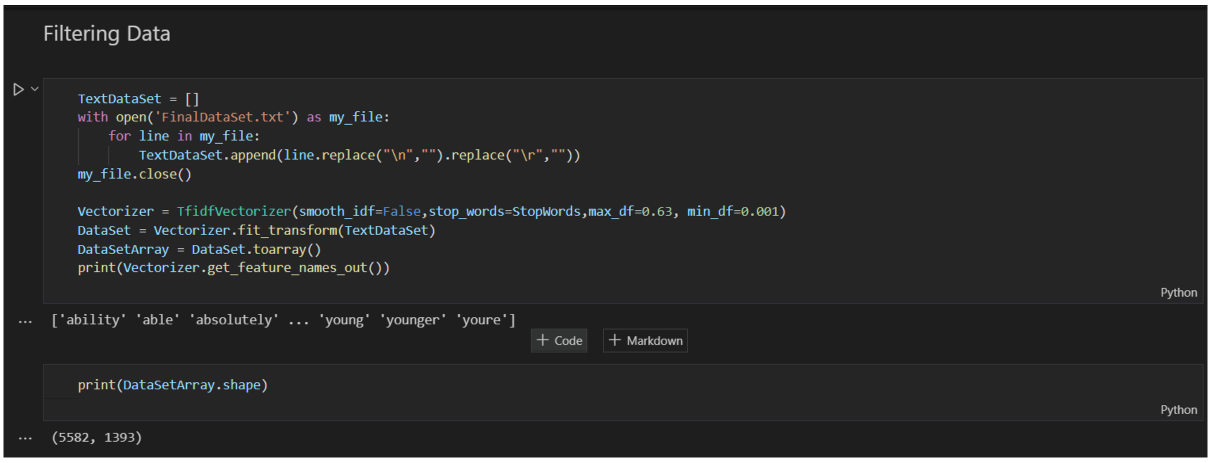
```
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

Python

```
#nltk.download('stopwords')
StopWords = stopwords.words('english')
print(StopWords[:])
```

Python

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs',
'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being',
'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of',
'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up',
'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any',
'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't',
'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn',
"couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't",
'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
"wouldn't"]
```



Filtering Data

```
TextDataSet = []
with open('FinalDataSet.txt') as my_file:
    for line in my_file:
        TextDataSet.append(line.replace("\n","").replace("\r",""))
my_file.close()

Vectorizer = TfidfVectorizer(smooth_idf=False,stop_words=StopWords,max_df=0.63, min_df=0.001)
DataSet = Vectorizer.fit_transform(TextDataSet)
DataSetArray = DataSet.toarray()
print(Vectorizer.get_feature_names_out())
```

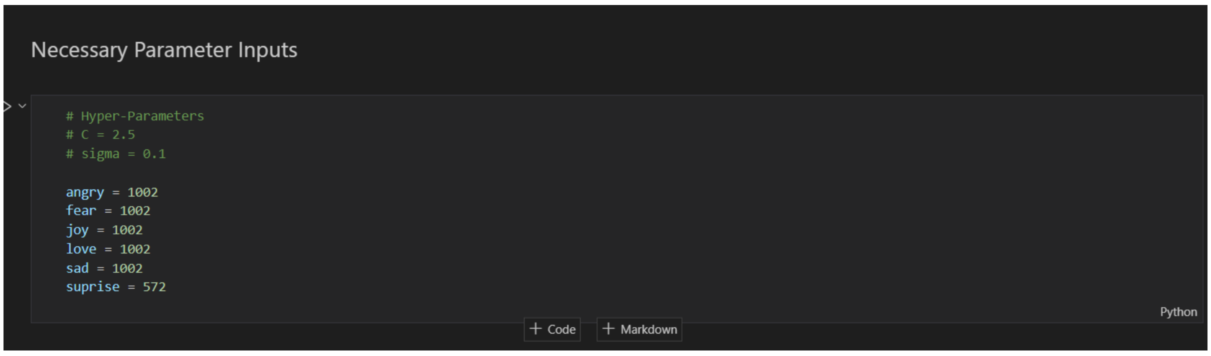
Python

```
['ability' 'able' 'absolutely' ... 'young' 'younger' 'youre']
```

```
print(DataSetArray.shape)
```

Python

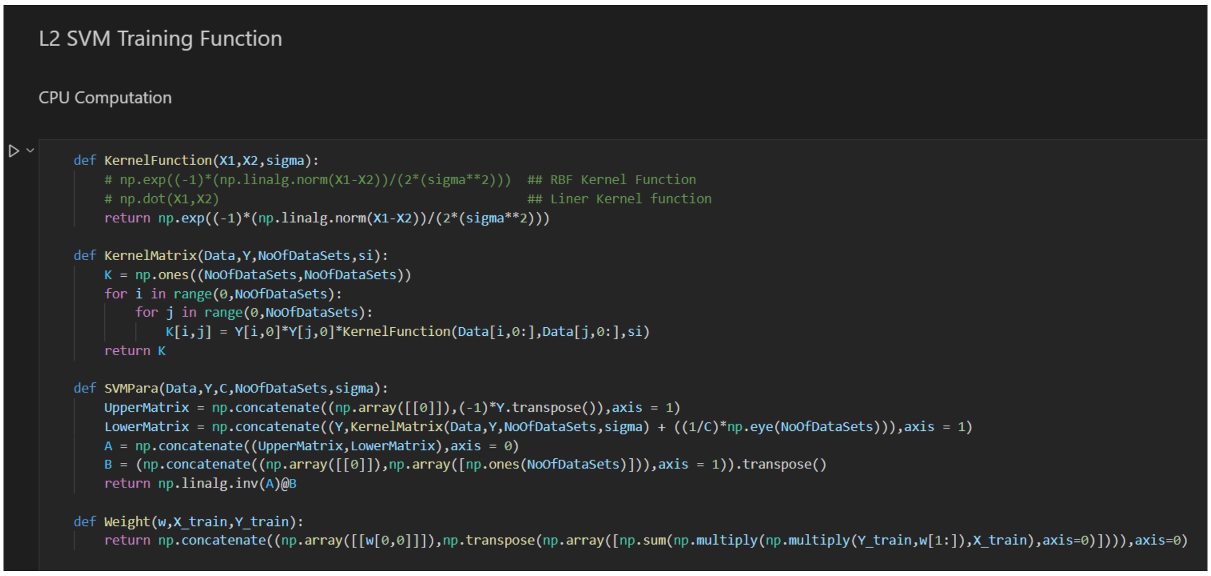
```
(5582, 1393)
```



Necessary Parameter Inputs

```
# Hyper-Parameters
# C = 2.5
# sigma = 0.1

angry = 1002
fear = 1002
joy = 1002
love = 1002
sad = 1002
suprise = 572
```

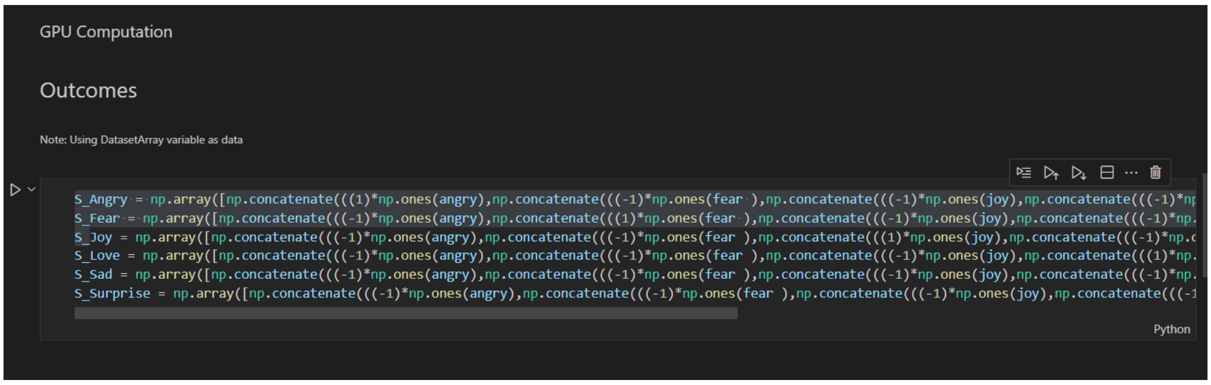
Python



L2 SVM Training Function

CPU Computation

```python
def KernelFunction(X1,X2,sigma):
    # np.exp((-1)*(np.linalg.norm(X1-X2))/(2*(sigma**2)))  ## RBF Kernel Function
    # np.dot(X1,X2)                                        ## Liner Kernel function
    return np.exp((-1)*(np.linalg.norm(X1-X2))/(2*(sigma**2)))

def KernelMatrix(Data,Y,NoOfDataSets,si):
    K = np.ones((NoOfDataSets,NoOfDataSets))
    for i in range(0,NoOfDataSets):
        for j in range(0,NoOfDataSets):
            K[i,j] = Y[i,0]*Y[j,0]*KernelFunction(Data[i,0:],Data[j,0:],si)
    return K

def SVMPara(Data,Y,C,NoOfDataSets,sigma):
    UpperMatrix = np.concatenate((np.array([[0]]),(-1)*Y.transpose()),axis = 1)
    LowerMatrix = np.concatenate((Y,KernelMatrix(Data,Y,NoOfDataSets,sigma) + ((1/C)*np.eye(NoOfDataSets))),axis = 1)
    A = np.concatenate((UpperMatrix,LowerMatrix),axis = 0)
    B = (np.concatenate((np.array([[0]]),np.array([np.ones(NoOfDataSets)])),axis = 1)).transpose()
    return np.linalg.inv(A)@B

def Weight(w,X_train,Y_train):
    return np.concatenate((np.array([[w[0,0]]]),np.transpose(np.array([np.sum(np.multiply(np.multiply(Y_train,w[1:]),X_train),axis=0)]))),axis=0)
```



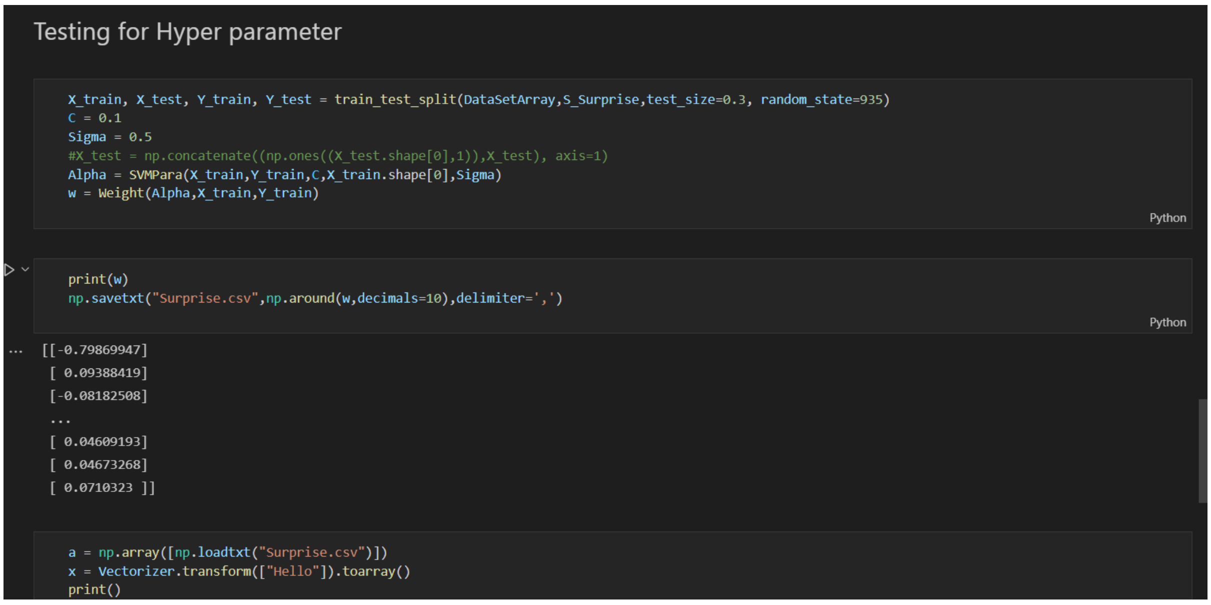
GPU Computation

Outcomes

Note: Using DatasetArray variable as data

```python
S_Angry = np.array([np.concatenate(((1)*np.ones(angry),np.concatenate(((-1)*np.ones(fear ),np.concatenate(((-1)*np.ones(joy),np.concatenate(((-1)*np
S_Fear = np.array([np.concatenate(((-1)*np.ones(angry),np.concatenate(((1)*np.ones(fear ),np.concatenate(((-1)*np.ones(joy),np.concatenate(((-1)*np.
S_Joy = np.array([np.concatenate(((-1)*np.ones(angry),np.concatenate(((-1)*np.ones(fear ),np.concatenate(((1)*np.ones(joy),np.concatenate(((-1)*np.c
S_Love = np.array([np.concatenate(((-1)*np.ones(angry),np.concatenate(((-1)*np.ones(fear ),np.concatenate(((-1)*np.ones(joy),np.concatenate(((1)*np.
S_Sad = np.array([np.concatenate(((-1)*np.ones(angry),np.concatenate(((-1)*np.ones(fear ),np.concatenate(((-1)*np.ones(joy),np.concatenate(((-1)*np.
S_Surprise = np.array([np.concatenate(((-1)*np.ones(angry),np.concatenate(((-1)*np.ones(fear ),np.concatenate(((-1)*np.ones(joy),np.concatenate(((-1
```

Python



Testing for Hyper parameter

```python
X_train, X_test, Y_train, Y_test = train_test_split(DataSetArray,S_Surprise,test_size=0.3, random_state=935)
C = 0.1
Sigma = 0.5
#X_test = np.concatenate((np.ones((X_test.shape[0],1)),X_test), axis=1)
Alpha = SVMPara(X_train,Y_train,C,X_train.shape[0],Sigma)
w = Weight(Alpha,X_train,Y_train)
```

Python

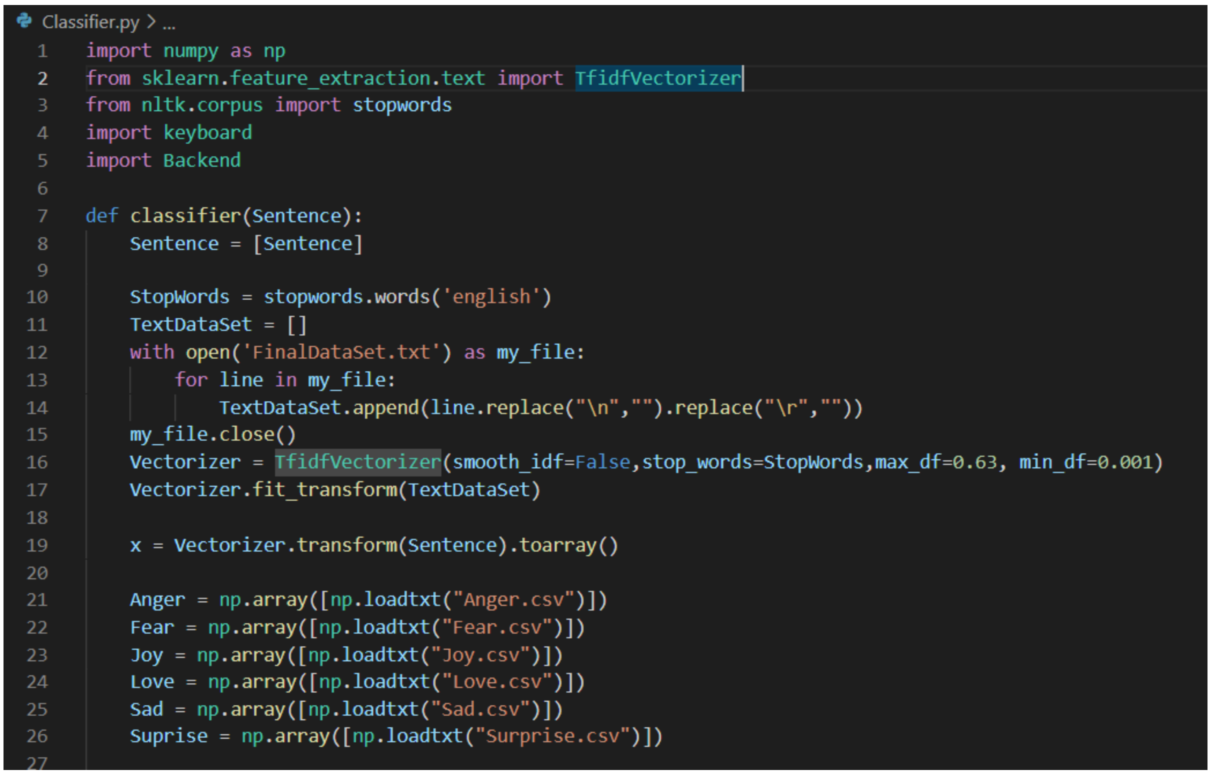
```python
print(w)
np.savetxt("Surprise.csv",np.around(w,decimals=10),delimiter=',')
```

Python

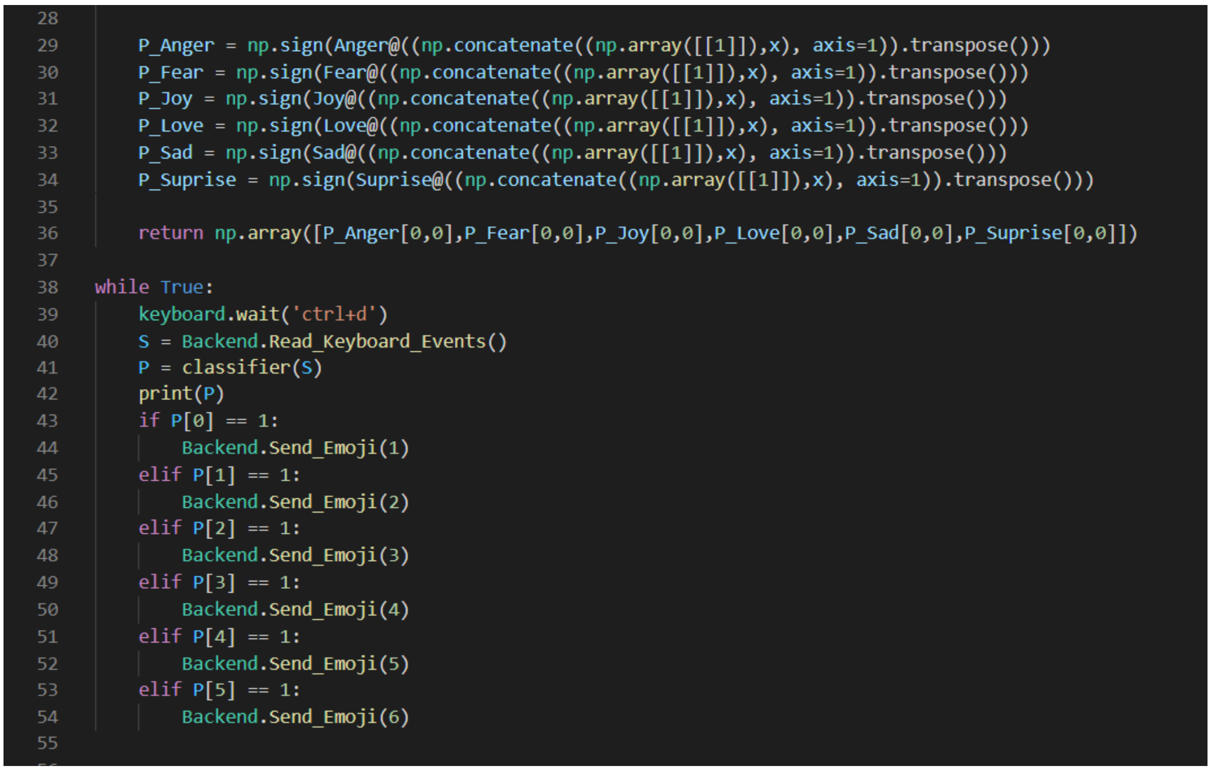
```
[[-0.79869947]
 [ 0.09388419]
 [-0.08182508]
 ...
 [ 0.04609193]
 [ 0.04673268]
 [ 0.0710323 ]]
```

```python
a = np.array([np.loadtxt("Surprise.csv")])
x = Vectorizer.transform(["Hello"]).toarray()
print()
```

Classifier.py

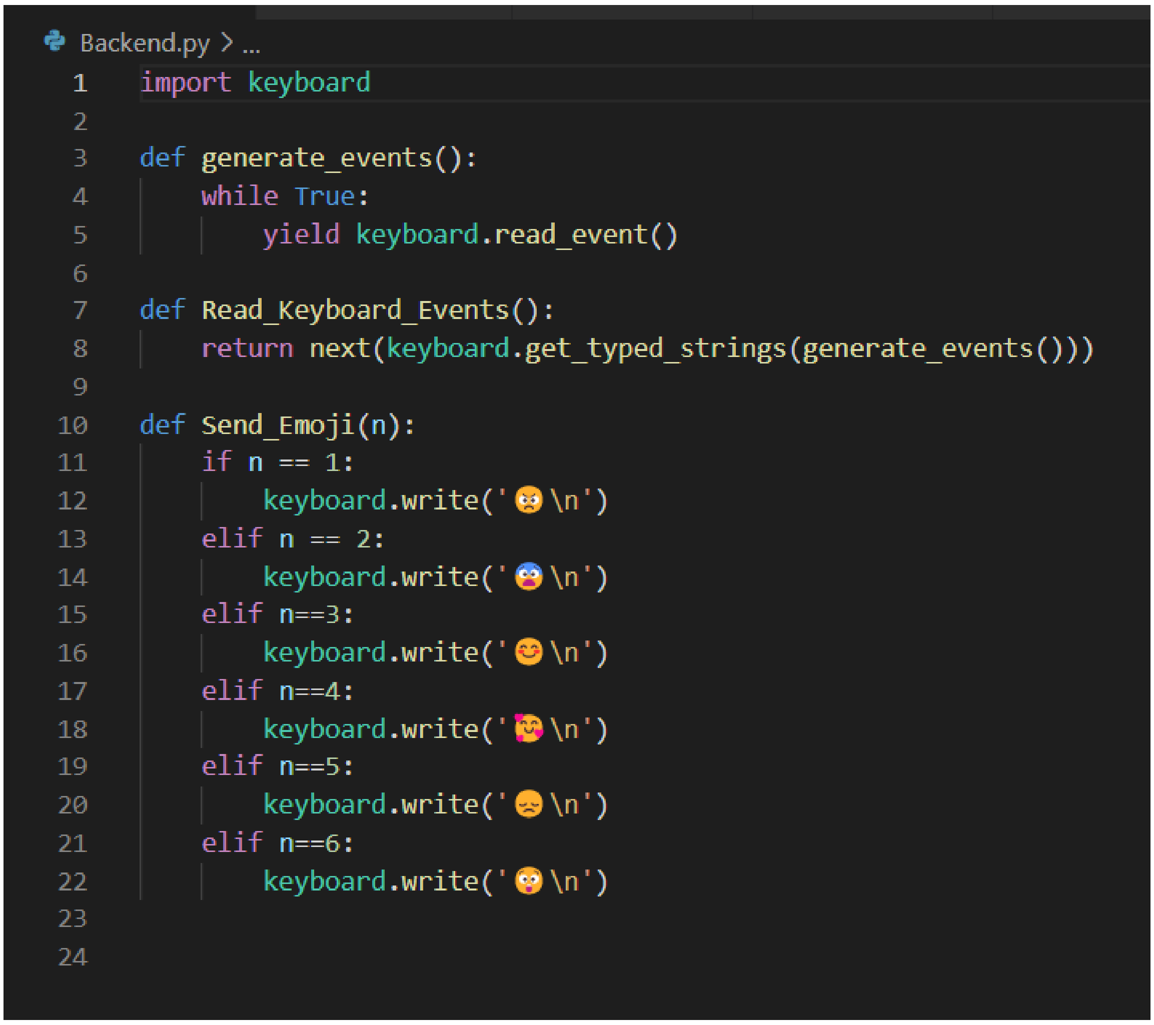


```python
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
import keyboard
import Backend

def classifier(Sentence):
    Sentence = [Sentence]

    StopWords = stopwords.words('english')
    TextDataSet = []
    with open('FinalDataSet.txt') as my_file:
        for line in my_file:
            TextDataSet.append(line.replace("\n","").replace("\r",""))
    my_file.close()
    Vectorizer = TfidfVectorizer(smooth_idf=False,stop_words=StopWords,max_df=0.63, min_df=0.001)
    Vectorizer.fit_transform(TextDataSet)

    x = Vectorizer.transform(Sentence).toarray()

    Anger = np.array([np.loadtxt("Anger.csv")])
    Fear = np.array([np.loadtxt("Fear.csv")])
    Joy = np.array([np.loadtxt("Joy.csv")])
    Love = np.array([np.loadtxt("Love.csv")])
    Sad = np.array([np.loadtxt("Sad.csv")])
    Suprise = np.array([np.loadtxt("Surprise.csv")])
```



```python

    P_Anger = np.sign(Anger@((np.concatenate((np.array([[1]]),x), axis=1)).transpose()))
    P_Fear = np.sign(Fear@((np.concatenate((np.array([[1]]),x), axis=1)).transpose()))
    P_Joy = np.sign(Joy@((np.concatenate((np.array([[1]]),x), axis=1)).transpose()))
    P_Love = np.sign(Love@((np.concatenate((np.array([[1]]),x), axis=1)).transpose()))
    P_Sad = np.sign(Sad@((np.concatenate((np.array([[1]]),x), axis=1)).transpose()))
    P_Suprise = np.sign(Suprise@((np.concatenate((np.array([[1]]),x), axis=1)).transpose()))

    return np.array([P_Anger[0,0],P_Fear[0,0],P_Joy[0,0],P_Love[0,0],P_Sad[0,0],P_Suprise[0,0]])

while True:
    keyboard.wait('ctrl+d')
    S = Backend.Read_Keyboard_Events()
    P = classifier(S)
    print(P)
    if P[0] == 1:
        Backend.Send_Emoji(1)
    elif P[1] == 1:
        Backend.Send_Emoji(2)
    elif P[2] == 1:
        Backend.Send_Emoji(3)
    elif P[3] == 1:
        Backend.Send_Emoji(4)
    elif P[4] == 1:
        Backend.Send_Emoji(5)
    elif P[5] == 1:
        Backend.Send_Emoji(6)
```

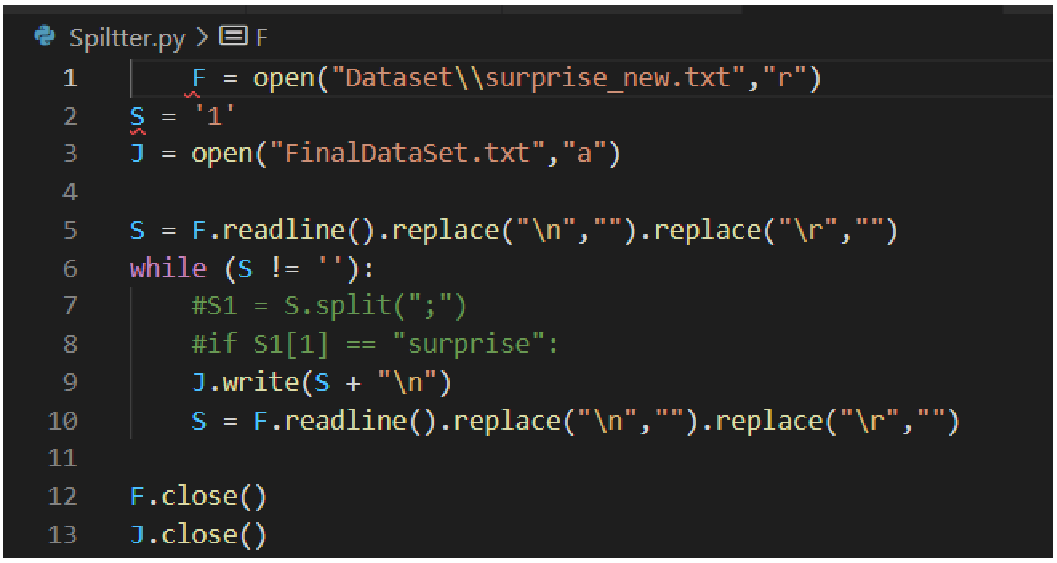
Backend.py



Backend.py > ...

```python
import keyboard

def generate_events():
    while True:
        yield keyboard.read_event()

def Read_Keyboard_Events():
    return next(keyboard.get_typed_strings(generate_events()))

def Send_Emoji(n):
    if n == 1:
        keyboard.write('😠\n')
    elif n == 2:
        keyboard.write('😱\n')
    elif n==3:
        keyboard.write('😊\n')
    elif n==4:
        keyboard.write('🥰\n')
    elif n==5:
        keyboard.write('😞\n')
    elif n==6:
        keyboard.write('😲\n')


```

Splitter.py



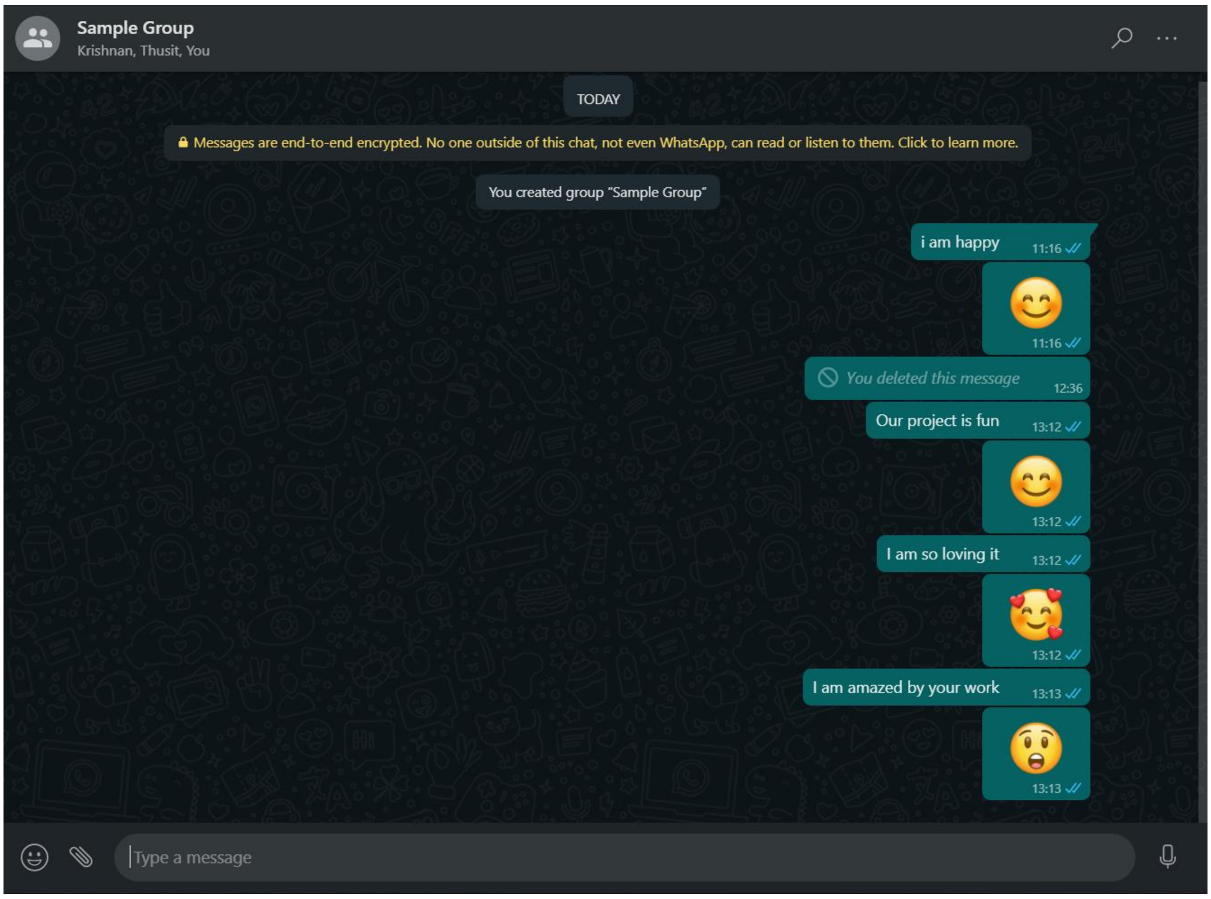
Spiltter.py > F

```python
F = open("Dataset\\surprise_new.txt","r")
S = '1'
J = open("FinalDataSet.txt","a")

S = F.readline().replace("\n","").replace("\r","")
while (S != ''):
    #S1 = S.split(";")
    #if S1[1] == "surprise":
    J.write(S + "\n")
    S = F.readline().replace("\n","").replace("\r","")

F.close()
J.close()
```

IMPLEMENTATION IN WHATSAPP CHAT

Given an input message, the emoticon gets automatically sent to the sender after the input message has been sent.

For every successful prediction the emoticon is sent in WA automatically



Sample Group
Krishnan, Thusit, You

TODAY

Messages are end-to-end encrypted. No one outside of this chat, not even WhatsApp, can read or listen to them. Click to learn more.

You created group "Sample Group"

i am happy 11:16

😊 11:16

You deleted this message 12:36

Our project is fun 13:12

😊 13:12

I am so loving it 13:12

🥰 13:12

I am amazed by your work 13:13

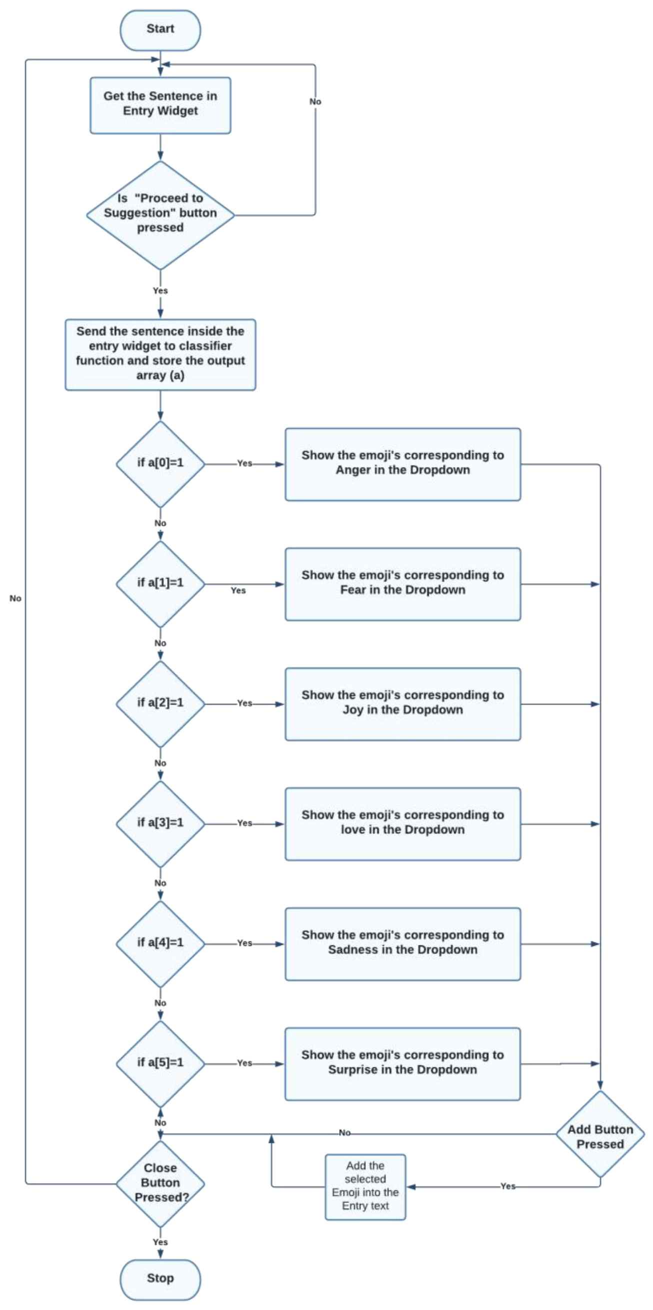
😲 13:13

Type a message

Refer to the video for live simulation example

https://amritavishwavidyapeetham-my.sharepoint.com/:v:/g/personal/cb_en_u4aie20074_cb_students_amrita_edu/Ee3KLnTPWzVIjxD5vi-7vrgBfUMvdVIqJUO4QXS59iuQSQ?e=6cjHPZ

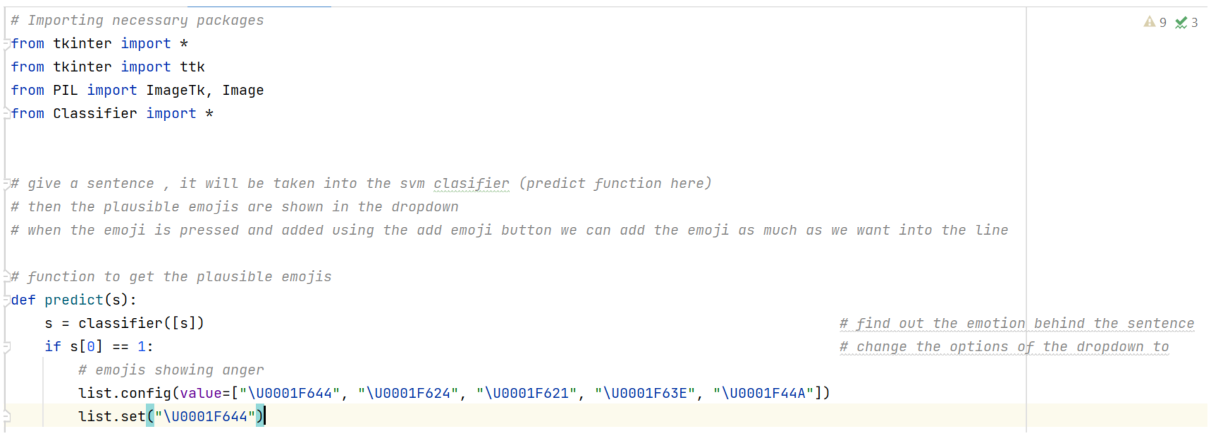
GUI DEMONSTRATION:



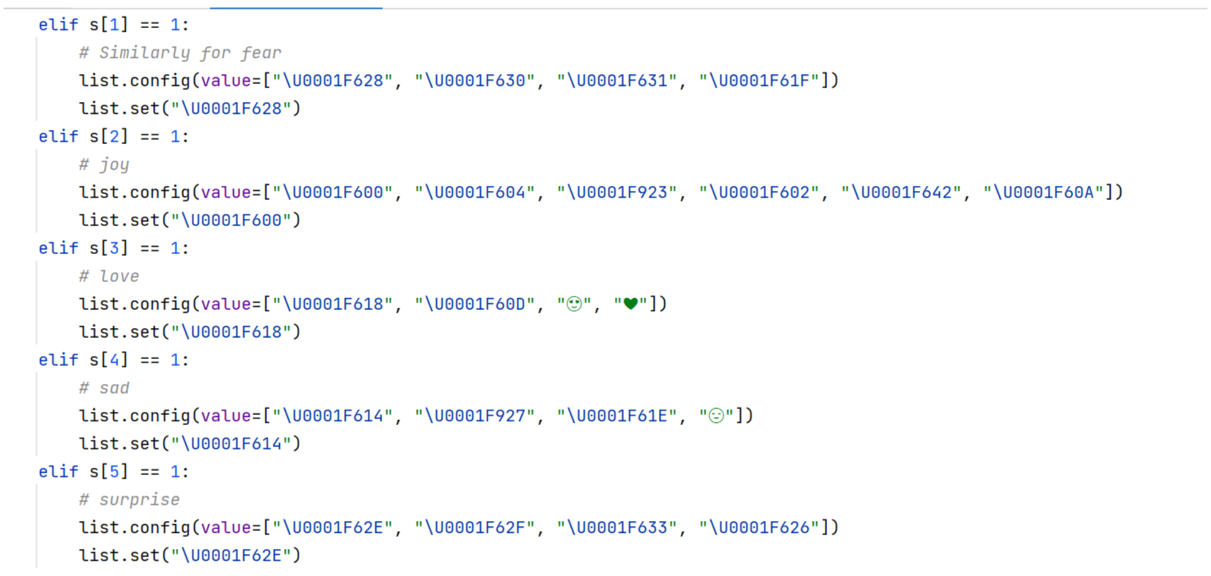
Start
Get the Sentence in Entry Widget
Is "Proceed to Suggestion" button pressed
No
Yes
Send the sentence inside the entry widget to classifier function and store the output array (a)
if a[0]=1
Show the emoji's corresponding to Anger in the Dropdown
if a[1]=1
Show the emoji's corresponding to Fear in the Dropdown
if a[2]=1
Show the emoji's corresponding to Joy in the Dropdown
if a[3]=1
Show the emoji's corresponding to love in the Dropdown
if a[4]=1
Show the emoji's corresponding to Sadness in the Dropdown
if a[5]=1
Show the emoji's corresponding to Surprise in the Dropdown
Add Button Pressed
Add the selected Emoji into the Entry text
Close Button Pressed?
Stop

GUI For Emoji Suggestion

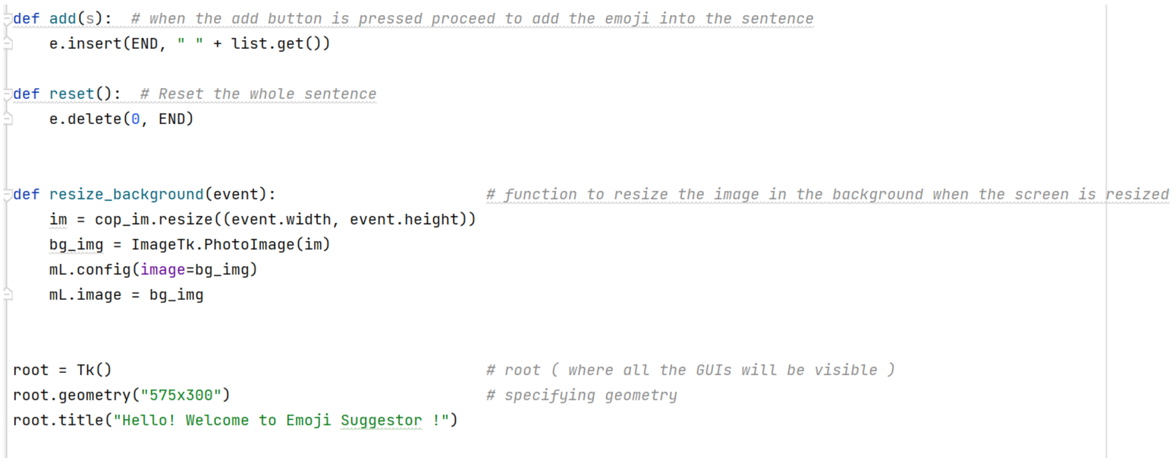
Code:



```python
# Importing necessary packages
from tkinter import *
from tkinter import ttk
from PIL import ImageTk, Image
from Classifier import *


# give a sentence , it will be taken into the svm clasifier (predict function here)
# then the plausible emojis are shown in the dropdown
# when the emoji is pressed and added using the add emoji button we can add the emoji as much as we want into the line

# function to get the plausible emojis
def predict(s):
    s = classifier([s])                                                         # find out the emotion behind the sentence
    if s[0] == 1:                                                               # change the options of the dropdown to
        # emojis showing anger
        list.config(value=["\U0001F644", "\U0001F624", "\U0001F621", "\U0001F63E", "\U0001F44A"])
        list.set("\U0001F644")
```



```python
    elif s[1] == 1:
        # Similarly for fear
        list.config(value=["\U0001F628", "\U0001F630", "\U0001F631", "\U0001F61F"])
        list.set("\U0001F628")
    elif s[2] == 1:
        # joy
        list.config(value=["\U0001F600", "\U0001F604", "\U0001F923", "\U0001F602", "\U0001F642", "\U0001F60A"])
        list.set("\U0001F600")
    elif s[3] == 1:
        # love
        list.config(value=["\U0001F618", "\U0001F60D", "☺", "♥"])
        list.set("\U0001F618")
    elif s[4] == 1:
        # sad
        list.config(value=["\U0001F614", "\U0001F927", "\U0001F61E", "☹"])
        list.set("\U0001F614")
    elif s[5] == 1:
        # surprise
        list.config(value=["\U0001F62E", "\U0001F62F", "\U0001F633", "\U0001F626"])
        list.set("\U0001F62E")
```



```python
def add(s):  # when the add button is pressed proceed to add the emoji into the sentence
    e.insert(END, " " + list.get())

def reset():  # Reset the whole sentence
    e.delete(0, END)


def resize_background(event):                    # function to resize the image in the background when the screen is resized
    im = cop_im.resize((event.width, event.height))
    bg_img = ImageTk.PhotoImage(im)
    mL.config(image=bg_img)
    mL.image = bg_img


root = Tk()                                      # root ( where all the GUIs will be visible )
root.geometry("575x300")                         # specifying geometry
root.title("Hello! Welcome to Emoji Suggestor !")
```

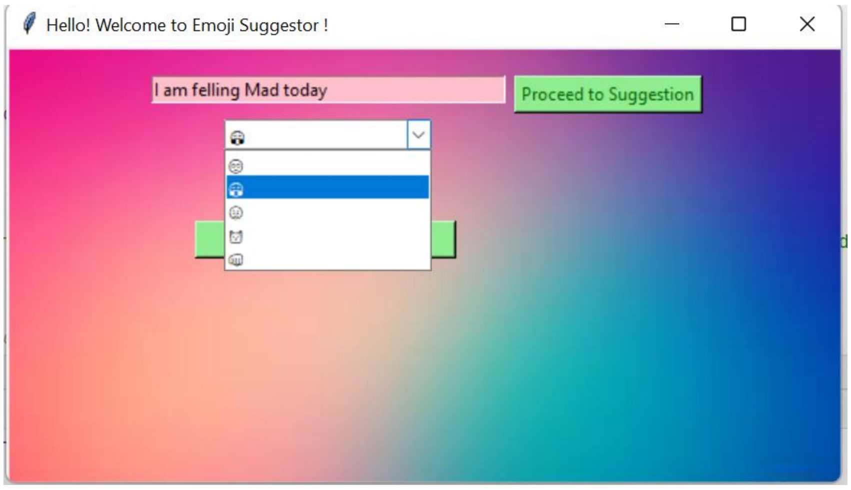
```python
# opening the background image and making it the Screen's background image
im = Image.open("C:/Users/Viswes/Downloads/Bgs/bg1.png")
cop_im = im.copy()
bg_img = ImageTk.PhotoImage(im)
mL = ttk.Label(root, image=bg_img)
mL.bind('<Configure>', resize_background)
mL.pack(fill=BOTH, expand=YES)

# combobox for the dropdown ( emoji options )
list = ttk.Combobox(root, value=[""])
list.place(x=150, y=50)

# Add button
ok2 = Button(root, text="Add", command=lambda: add(list.get()), bg="light green", fg="dark green")
ok2.place(x=200, y=80)

# Suggestion Button
Ok = Button(root, text="Proceed to Suggestion", command=lambda: predict(e.get()), bg="light green", fg="dark green")
Ok.place(x=350, y=20)
```

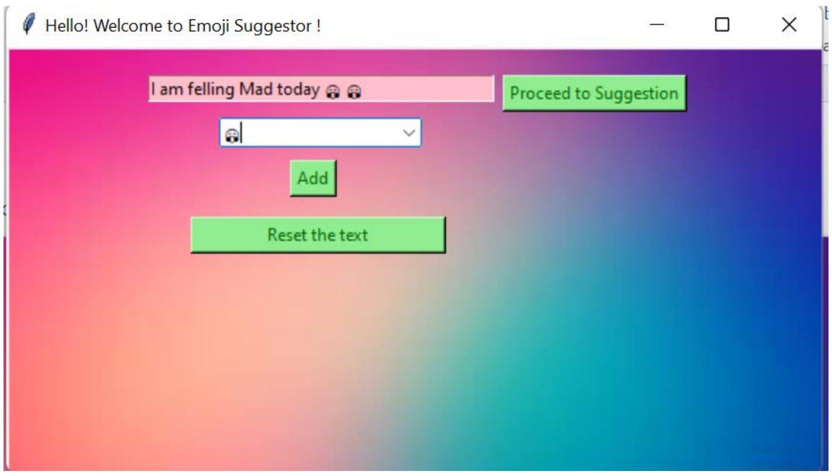
```python
# Entry to get the input sentence
e = Entry(root, width=40, bg="Pink")
e.place(x=100, y=20)
e.insert(0, "Enter your Sentence here")

# Button to reset
Reset = Button(root, text="Reset the text", command=reset, padx=50, bg="light green", fg="dark green")
Reset.place(x=130, y=120)

root.mainloop()
```

OUTPUTS:

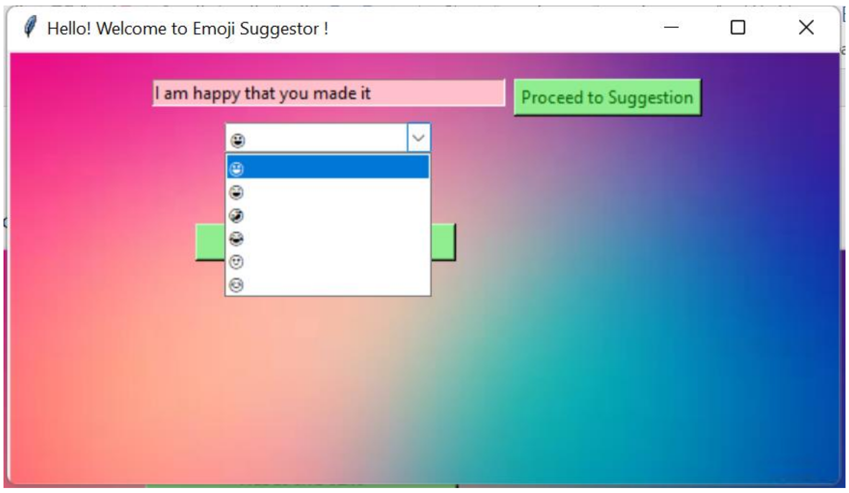
Here we have pasted the output for only two emotions (Angry, Happy).

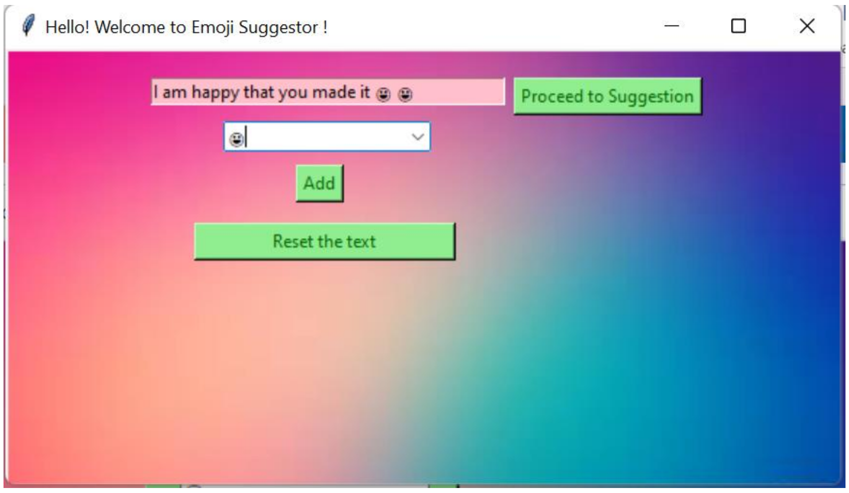
Angry:




Hello! Welcome to Emoji Suggestor !
I am felling Mad today
Proceed to Suggestion
Add
Reset the text

Happy:


Hello! Welcome to Emoji Suggestor !
I am happy that you made it
Proceed to Suggestion


Hello! Welcome to Emoji Suggestor !
I am happy that you made it
Proceed to Suggestion
Add
Reset the text

References:

1. Least Squares Support Vector Machine Classifiers -
https://www.researchgate.net/publication/220578095_Least_Squares_Support_Vector_Machine_Classifiers

2. Comparison of L1 and L2 Support Vector Machines -
https://www.researchgate.net/publication/4030193_Comparison_of_L1_and_L2_Support_Vector_Machines

3. Text classification: A least square support vector machine approach -
https://www.researchgate.net/publication/222420760_Text_classification_A_least_square_support_vector_machine_approach

4. TFIDF Vectorisation –

https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3

THANK YOU

GROUP-6

****