

SOCIAL MEDIA CRIME DETECTION USING MACHINE LEARNING ALGORITHMS

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree

In

Computer Science and Engineering

By

PALEPU SAI (Reg. No. 38110378)

PALLA VENKATA NAGA SIVA SAI DURGA HARSHA VARDHAN
(Reg. No. 38110379)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

March - 2022



SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(Established under Section 3 of UGC Act, 1956)

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600119 www.sathyabama.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **PALEPU SAI (Reg. No. 38110378), PALLA VENKATA NAGA SIVA SAI DURGA HARSHA VARDHAN (Reg. No. 38110379)** who carried out the project entitled “**Social Media Crime Detection Using Machine Learning Algorithms**” under our supervision from December 2021 to March 2022.

Internal Guide

Dr.B. ANKAYARKANNI, M.E., PH.D

Head of the Department

Dr. S. VIGNESHWARI M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **PALEPU SAI (Reg.No.38110378), PALLA VENKATA NAGA SIVA SAI DURGA HARSHA VARDHAN (Reg. No. 38110379)** hereby declare that the Project Report entitled “**Social Media Crime Detection Using Machine Learning Algorithms**” done by me under the guidance of **Dr.B. ANKAYARKANNI, M.E., PH.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering / Technology degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D., Dean, School of Computing**, and **DR. S. VIGNESHWARI, M.E., Ph.D., Head of the Department, Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.B.ANKAYARKANNI,M.E.,PH.D** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the department of **COMPUTER SCIENCE AND ENGINEERING** who were helpful in many ways for the completion of the project.

ABSTRACT

Nowadays, the task of preventing suicide is one of the priorities in the health sector. Therefore, it is important to identify people prone to suicide at an early stage. This article discusses the possibility of real-time detection of visited websites containing suicidal statements. The classification of web pages is based on the analysis of the text contained on it. This work can be divided into two parts: creating a browser extension and the server. The extension collects information about the content of the web pages visited by the user and transmits it to the server. The page classification process takes place on the server. In the final part of this work, a comparison of the effectiveness of detecting suicidal websites using various machine learning algorithms is presented.

About 800 000 people commit suicide every year and detecting suicidal people remains a challenging issue as mentioned in a number of suicide studies. With the increased use of social media, we witnessed that people talk about their suicide plans or attempts in public on these networks. This paper addresses the problem of suicide prevention by detecting suicidal profiles in social networks. First, we analyse profiles from social media and extract various features including account features that are related to the profile and features that are related to the social media data. Second, we introduce our method based on machine learning algorithms to detect suicidal profiles using Twitter data. Then, we use a profile data set consisting of people who have already committed suicide. Experimental results verify the effectiveness of our approach in terms of recall and precision to detect suicidal profiles. Finally, we present a Java based prototype of our work that shows the detection of suicidal profiles.

TABLE OF CONTENTS

Chapter no.	Title	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF TABLES	viii
1.	INTRODUCTION	2
	1.1 EXISTING SYSTEM	2
	1.2 PROBLEM STATEMENT	3
	1.3 PROPOSED SYSTEM	3
	1.4 ADVANTAGES	4
2.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	5
	2.1 MACHINE LEARNING ALGORITHMS	5-6
	2.2 EXPERIMENTS	6-9
	2.3 BLOCK DIAGRAM	9
	2.4 FLOW DIAGRAM	10
3.	ALGORITHM USED	11
	3.1 LOGISTIC REGRESSION	11-14
	3.2 TYPES OF LOGISTIC REGRESSION	14-17
	3.3 ADVANTAGES OF LOGISTIC REGRESSION	17-18

	3.4 DISADVANTAGES OF LOGISTIC REGRESSION	18-19
	3.5 APPLICATION OF LOGISTIC REGRESSION	19
	3.6 BEAUTIFUL SOUP	20
	3.7 TKINTER	20-21
4.	MACHINE LEARNING	22
	4.1 INTRODUCTION	22-23
	4.2 GATHERING DATA	23-24
	4.3 DATA PRE-PROCESSING	24
	4.3.1 WHAT IS DATA PRE-PROCESSING	24-25
	4.3.2 HOW CAN DATA PRE-PROCESSING BE PERFORMED	25-26
	4.3.3 RESEARCHING THE MODEL THAT WILL BE BEST FOR THE TYPE OF DATA	26
	4.4 CLASSIFICATION	27-28
	4.5 REGRESSION	28
	4.6 UNSUPERVISED LEARNING	29
	4.6.1 CLUSTERING	29-33
5	5.1 SYSTEM DESIGN	34
	5.1.1 DATA COLLECTION	34
	5.1.2 DATA-CLEANING	34
	5.1.3 PRE-PROCESSING	34-35

	5.1.4 EXPLORATORY DATA ANALYSIS(EDA)	35
	CONCLUSIONS	36
	REFERENCE PAPERS	37-38
	OUTPUTS	39-55

	LIST OF FIGURES	
Figure No.	Name of the Figure	Page No.
3.1.1	Logistic Regression Model	12
4.3.3.1	Researching the model that will be best for the type of data	26
4.4.1	Classification	27
4.5.1	Regression	28
4.6.1.1	Clustering	29
4.6.1.2	Overview Of Models	30
4.6.1.3	Training and testing the model on data	30
4.6.1.4	Validation Set	31
4.6.1.5	prediction table	32

LIST OF TABLES

Table No.	Name of the Table	Page No.

Chapter 1

INTRODUCTION

At the moment, the task of preventing suicide is relevant, because according to statistics of the World Health Organization every year more than 800 000 people commit suicide. According to the same data, the Russian Federation is among the five countries with the highest number of suicides per 100,000 inhabitants. Since the Internet is the easiest way to distribute suicidal content in the form of various web pages dedicated to suicide, a large number of organizations are trying to solve this problem, for example, the popular social network Facebook, which detects suspicious profiles of users prone to suicide, and posts containing suicidal content. In addition to social networks, federal services are trying to solve the problem, for example, in Russia Roskomnadzor in 2016 published recommendations for disseminating information about suicide cases in the media, which probably affects the results of search engines on this topic. In addition, since 2006 there has been a unified register that contains websites blocked in the Russian Federation. However, blocking does not happen immediately. Some people manage to visit dangerous web pages. Manual blocking of suicidal websites can hardly be called an effective measure in the fight against the spread of suicidal content. Since after several years of such blockages, the number of child suicides in Russia has risen sharply again. In addition to the «death groups» in social networks that platform developers are already actively fighting, one of the reasons may be that websites with suicidal content can create their own copies (mirrors). This article discusses the possibility of detecting such web pages by analyzing their content in real time using machine learning algorithms. Detection occurs on the client's side. In this way, with sufficient accuracy to identify dangerous websites visited by the user, it is possible to identify a person who is suicidal at an early stage.

1.1 EXISTING SYSTEM:

- context utilize publicly shared attributes including name, gender, location, and other information to identify user profiles in social networks
- However, due to the privacy settings, user's attributes are not available in many cases and this makes these existing works fragile. In addition, some researchers address the problem of suicide .
- However, even though tweets contain rich information that can identify users, they can miss some significant details that maybe available on the user profile public attributes and may contribute to a higher accuracy of suicide detection.
- Apart from these works, we utilize in our approach both user shared information that we call account features and tweets as an attempt to solve the problem of suicidal profiles detection. First, posted tweets pose important challenges to infer more information about users. The most relevant challenge is semantic features that are difficult to extract directly from user's posted tweets such as stylometry, writing style, sentiments, emoji's, hashtags, n-grams, etc.
- Instead of many existing studies that ignore these features to identify users, we analyses tweets and extract as much as possible of semantic features.
- Second, adding account features to the user's posted tweets can help to improve the suicide detection task since they may reflect the habits and characteristics of users.

1.2 PROBLEM STATEMENT:

- In the present system, the system is actualized to comprehend the network and correspondence attributes of Twitter clients whose duty is to produce the content therefore arranged by a human expert as containing reasonable self-destructive expectation or considering, usually, alluded to as self-destructive ideation.
- Poor performing in data mining.
- Less efficiency.
- Result is not accurate.

1.3 PROPOSED SYSTEM:

- In this work, we propose a method for detecting suicidal profiles. First, we analyse a number of profiles from the social network through exploiting the maximum of available data
- Then, we adopt several features to distinguish between suicidal and not suicidal profiles.
- These features can be explicitly extracted from the user profile or implicitly inferred using different data mining tools and techniques.
- Here, we focus on emotional features and sentiment analysis, which gives indications about the psychological state of suicidal profiles.

- We further use account features to identify users through the shared information on their profiles.
- We finally present each user as a vector that integrates all the used features.

1.4 ADVANTAGES:

- NLP approaches focus on the analysis of acoustic and linguistic aspects of human language derived from speech and text and can be integrated with machine learning approaches to classify depression and its severity .
- Advantages of using these approaches to understand depression symptoms through speech include high ecological validity, low subjectivity, low cost of frequent assessments, and quicker administration of tasks compared to standard assessments.
- An added benefit of speech analysis using NLP is that speech data can be collected remotely, meeting a vital need for remote cognitive and behavioral assessments in the era of the coronavirus disease (COVID-19) pandemic .
- With an aging population globally, the identification and treatment of depression in older adults is critical. NLP approaches are proving to be a promising means to help assess, monitor, and detect depression and other comorbidities in both younger and older individuals.

CHAPTER 2

2.1 MACHINE LEARNING ALGORITHMS:

- Random Forest:

Random forest algorithm, like its name implies, consists of a large number of individual decision trees that operate together. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

- Decision Tree:

A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers to the question; and the leaves represent the actual output or class label. They are used in non-linear decision making with a simple linear decision surface.

- Naïve Bayes:

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

- **Logistic Regression:**

The logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

- **K-nearest Neighbours:**

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

Traditionally, there was no such system to keep an eye on various websites or any suspicious words present online. Cops were unable to track the terrorist related website or any person with suspicious information. The ratio of terrorism is high in today's world. There must be a system to track those suspicious word online and bring down the ratio of terrorism. In various arrangements and have images, videos etc. intermixed on a single web page. So we here propose to use smartly designed web mining algorithms to mine textual information on web pages and detect their relevancy to terrorism. In this way we may judge web pages and check if they may be promoting terrorism. This system proves useful in anti-terrorism sectors and even search engines to classify web pages into the category. Their relevance to the field helps classify and sort them appropriately and flag them for human review.

2.2 EXPERIMENTS:

There are different ways to detect people who are suicidal. In this article, we will discuss a method based on the analysis of text content. At the initial stage of

work, the profiles of people in the popular Russian social network VK were supposed to be used as the analyzed web pages. However, a long manual search for profiles with suicidal content did not yield results. The search was conducted among groups of people in communities associated with suicide. Similar communities were searched for by keywords: "suicide", "death", "self-murder" and others. Most of the viewed profiles were hidden by the privacy settings, which made it impossible to access the profile content for further analysis. More than half of the remaining people did not have any obvious signs in their profiles indicating that they were suicidal, or their profiles were already blocked. Also considered were "dead pages" groups that contain pages of dead people. Next, we looked at the profiles of supposedly dead users whose cause of death was suicide. The profiles of such users also did not show any obvious signs of suicidal tendencies. Most likely, this is due to the fact that since about 2019, the social network VK has created its own algorithms based on neural networks that can detect users who are suicidal. At this stage, it became clear that it would be ineffective to conduct further research based on the VK social network, since we will be working with already filtered data. A client application was developed that reads the source code of the page, as an extension for the Google Chrome browser. The extension extracts text content and sends it over the HTTP Protocol to a web server developed in Python. One of the standard Python libraries, «http.server», was chosen for the server implementation. After receiving the page text, it is processed:

1. The text is divided into sentences.
2. All sentences are translated into the Russian language.
3. All punctuation marks, extraneous characters and numbers are removed from sentences.
4. The words of such parts of speech as conjunctions, particles, numbers, interjections and prepositions are deleted, except for "not" and "neither".
5. Lemmatize sentences (reduction of words to their initial form).

After preprocessing, the sentences are analyzed using a pretrained machine learning algorithm that binary classifies them into potentially dangerous and safe

ones. We then calculate the percentage of the number of sentences determined by the algorithm as related to suicide T to the total number of sentences on web page A.

$$T / A * 100 > P$$

If according to formula the percentage of dangerous offers exceeds a certain threshold P, the user's access to the web page is limited. This solution let us to identify a person prone to suicide even at the stage of visiting suspicious web pages. In addition, the extension lets us to partially implement the "parental control" function, that is, when you visit dangerous web pages, a parent is notified.

Let us consider each of the points in more detail:

1.To simplify the processing of text by the machine learning algorithm, the text should be split into sentences. The decision to split the text into sentences is also related to the format of the available training sample, which will be discussed later. In addition, at this stage, all words are separated by one space.

2.Since the user can visit websites not only in Russian, but also in a foreign language, it is necessary to translate the text content of web pages into Russian. For this, the library «googletrans» was used, which uses Google Translate. It should be noted that this library allows us to automatically detect the language of the text of a web page and translate it into Russian.

3.Numbers, like special characters, cannot significantly affect the meaning of a sentence, so they were removed.

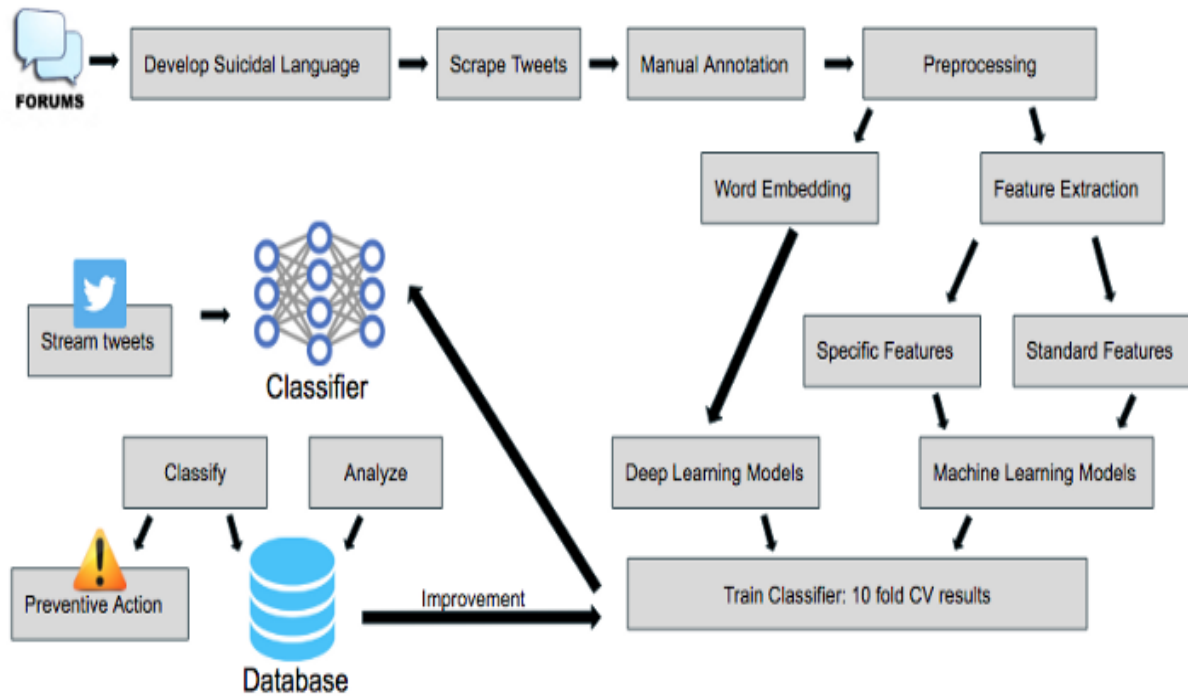
4.In the Russian language, there are parts of speech that that could not affect the attitude of the text to suicide ("I", "a", "ten", "oh"). For this reason, and to increase the effectiveness of recognizing suicidal sentences, the words of these parts of speech were removed from the sentences. However, there are "not" and "nor" particles that can greatly affect meaning of a phrase and sentence. For example, the sentences «I want to live» and «I do not want to live» have exactly the opposite meaning, so the particles «not» and «neither» are saved. To determine the part of speech, the «pymorphy2» library was used [20]. 5. Lemmatization is

the process of reducing a word to its normal form. In Russian, there are many different endings and forms of the same word. In this regard, all the words of the sentences go through this process.

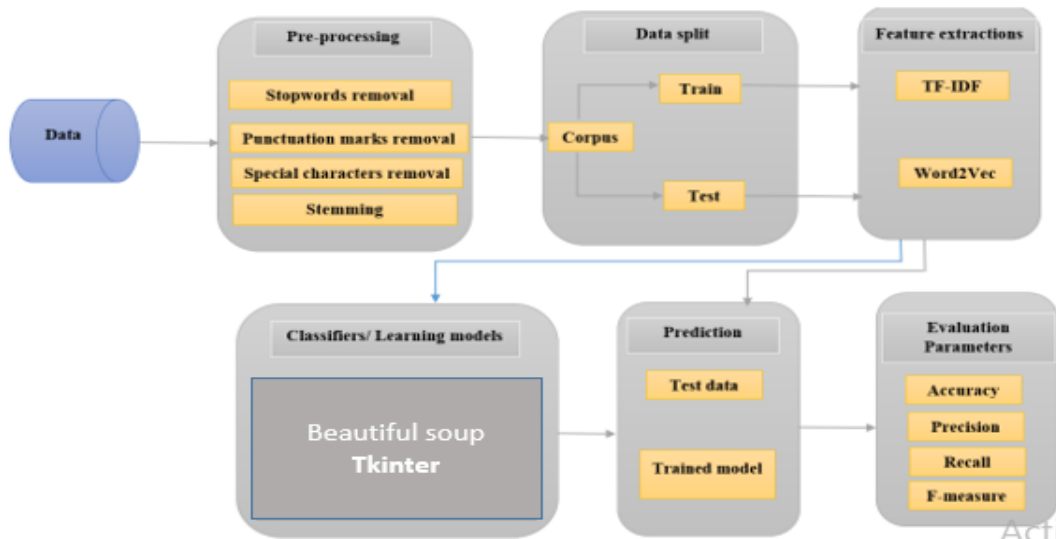
This process is to improve the accuracy of text classification by machine learning algorithms. The «pymorphy2» library was also used for this purpose. For training and checking of accuracy, a pre-prepared dataset was used, which was collected based on other articles in this area and divided into training and test samples. The training sample, which was used to train the models, contained approximately 700.000 sentences, and the test sample, on which the accuracy of the algorithms was checked, contained about 2.000. The data is binary marked: 1 - the sentence contains a suicidal statement, 0 - the sentence does not contain a suicidal statement. The data format of this set is like the format to which the content of the web pages was converted.

Fragment from the training sample: «I go for a walk with my dog» - 0, «I really want to die» - 1. Then the text is vectorized using the built-in «scikit-learn» module «CountVectorizer». Then the converted text is submitted for training in different algorithms. After that, a control group of web pages was assembled to test the algorithms in close-to-reality conditions. The control group of websites consisted of 50 web pages, half of which contained text content with suicidal statements. The other half contained completely ordinary random sites of different topics that are not related to suicide. For example, we considered a threshold of 10% and 50%. The results of detecting dangerous web pages using different algorithms are shown below. For each threshold and algorithm, the percentage of correctly detected web pages was presented.

2.3 BLOCK DIAGRAM:



2.4 FLOW DIAGRAM:



CHAPTER 3

ALGORITHM USED :

3.1 LOGISTIC REGRESSION :

What is Logistic Regression?

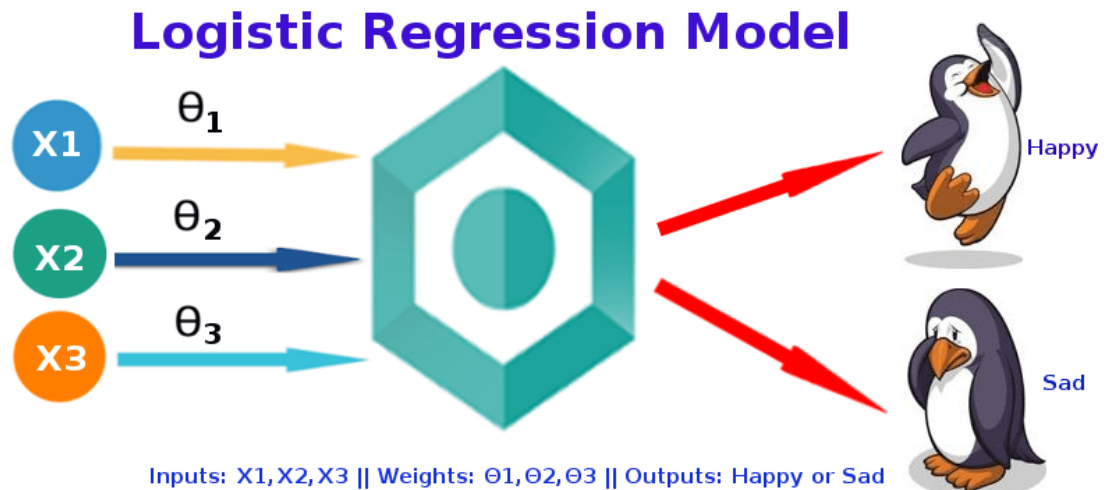
I hope the preceding discussion has provided us with a better understanding of machine learning and its various types. Logistic Regression is a Machine Learning method that is used to solve classification issues. It is a predictive analytic technique that is based on the probability idea. The classification algorithm Logistic Regression is used to predict the likelihood of a categorical dependent variable. The dependant variable in logistic regression is a binary variable with data coded as 1 (yes, True, normal, success, etc.) or 0 (no, False, abnormal, failure, etc.).

The goal of Logistic Regression is to discover a link between characteristics and the likelihood of a specific outcome. For example, when predicting whether a student passes or fails an exam based on the number of hours spent studying, the response variable has two values: pass and fail.

A Logistic Regression model is similar to a Linear Regression model, except that the Logistic Regression utilizes a more sophisticated cost function, which is known as the “Sigmoid function” or “logistic function” instead of a linear function.

Many people may have a question, whether Logistic Regression is a classification or regression category. The logistic regression hypothesis suggests that the cost function be limited to a value between 0 and 1. As a result, linear functions fail to describe it since it might have a value larger than 1 or less than 0, which is impossible according to the logistic regression hypothesis.

Behind every great leader, there was an even greater logistician



3.1.1 -- Fig

To answer this question, Logistic Regression is considered a regression model also. This model creates a regression model to predict the likelihood that a given data entry belongs to the category labeled “1.” Logistic regression models the data using the sigmoid function, much as linear regression assumes that the data follows a linear distribution.

Why the name Logistic Regression?

It's called 'Logistic Regression' since the technique behind it is quite similar to Linear Regression. The name “Logistic” comes from the Logit function, which is utilized in this categorization approach.

Why we can't use Linear Regression instead of Logistic Regression?

Before answering this question, we will explain from Linear Regression concept, from the scratch then only we can understand it better. Although logistic regression is a sibling of linear regression, it is a classification technique, despite its name. Mathematically linear regression can be explained by,

$$y = mx + c$$

y – predicted value

m – slope of the line

x – input data

c- Y-intercept or slope

We can forecast y values such as using these values. Now observe the below diagram for a better understanding, The x values are represented by the blue dots (the input data). We can now compute slope and y coordinate using the input data to ensure that our projected line (red line) covers most of the locations. We can now forecast any value of y given its x values using this line.

One thing to keep in mind about linear regression is that it only works with continuous data. If we want to include linear regression in our classification methods, we'll have to adjust our algorithm a little more. First, we must choose a threshold so that if our projected value is less than the threshold, it belongs to class 1; otherwise, it belongs to class 2. Now, if you're thinking, "Oh, that's simple, just create linear regression with a threshold, and hurray!, classification method," there's a catch. We must specify the threshold value manually, and calculating the threshold for huge datasets will be impossible. Furthermore, even if our anticipated values vary, the threshold value will remain the same. A logistic regression, on the other hand, yields a logistic curve with values confined to 0 and 1. The curve in logistic regression is generated using the natural logarithm of the target variable's "odds," rather than the probability, as in linear regression. Furthermore, the predictors need not be regularly distributed or have the same variance in each group.

And now the question?

This famous title question was explained by our beloved person Andrew Ng, assume we have information about tumor size and malignancy. Because this is a classification issue, we can see that all the values will fall between 0 and 1. And, by fitting the best-found regression line and assuming a threshold of 0.5, we can do a very good job with the line. Even if we fit the best-found regression line, we won't be able to determine any point where we can distinguish classes. It will insert some instances from the positive class into the negative class. The green dotted line (Decision Boundary) separates malignant and benign tumors,

however, it should have been a yellow line that clearly separates the positive and negative cases. As a result, even a single outlier can throw the linear regression estimates off. And it's here that logistic regression comes into play.

Logit function to Sigmoid Function – Logistic Regression:

where $p(x)/(1-p(x))$ is termed odds, and the left-hand side is called the logit or log-odds function. The odds are the ratio of the chances of success to the chances of failure. As a result, in Logistic Regression, a linear combination of inputs is translated to $\log(\text{odds})$, with an output of 1. The following is the inverse of the aforementioned function This is the Sigmoid function, which produces an S-shaped curve. It always returns a probability value between 0 and 1. The Sigmoid function is used to convert expected values to probabilities. The function converts any real number into a number between 0 and 1. We utilize sigmoid to translate predictions to probabilities in machine learning. The mathematically sigmoid function can be,

3.2 Types of Logistic Regression:

In general, it can be classified into,

1. Binary Logistic Regression – two or binary outcomes like yes or no
2. Multinomial Logistic Regression – three or more outcomes like first, second, and third class or no class
degree
3. Ordinal Logistic Regression – three or more like multinomial logistic regression but here with the order like customer rating in the supermarket from 1 to 5

Requirements for Logistic Regression to work well

This model can work for all the datasets, but still, if you need good performance, then there will be some assumptions to consider,

1. The dependant variable in binary logistic regression must be binary.
2. Only the variables that are relevant should be included.

3. The independent variables must be unrelated to one another. That is, there should be minimal or no multicollinearity in the model.

4. The log chances are proportional to the independent variables. 5. Large sample sizes are required for logistic regression.

A threshold can be established to forecast which class a data belongs to. The derived estimated probability is categorized into classes based on this threshold. If the predicted value is less than 0.5, categorize the particular student as a pass; otherwise, label it as a fail. There are two types of decision boundaries: linear and non-linear. To provide a complicated decision boundary, the polynomial order can be raised.

Why can't the cost function that was used for linearity be used for logistics?

The cost function for linear regression is mean squared error. If this is utilized for logistic regression, the function of parameters will be non-convex. Only if the function is convex will gradient descent lead to a global minimum.

Cost function – Linear Regression Vs Logistic Regression

Linear regression employs the Least Squared Error as the loss function, which results in a convex network, which we can then optimize by identifying the vertex as the global minimum. For logistic regression, however, it is no longer a possibility. Because the hypothesis has been modified, calculating Least Squared Error using the sigmoid function on raw model output will result in a non-convex graph with local minimums. What is cost function? Cost functions are used in machine learning to estimate how poorly models perform. Simply put, a cost function is a measure of how inaccurate the model is in estimating the connection between X and y . This is usually stated as a difference or separation between the expected and actual values. A machine learning model's goal is to discover parameters, weights, or a structure that minimizes the cost function. A convex function indicates there will be no intersection between any two points on the curve, but a non-convex function will have at least one intersection. In terms of

cost functions, a convex type always guarantees a global minimum, whereas a non-convex type only guarantees local minima.

How to reduce cost function? – Gradient Descent

The challenge now is: how can we lower the cost value? Gradient Descent can be used to accomplish this. Gradient descent's main objective is to reduce the cost value.

Regularization

Let's also discuss Regularization quickly for reducing the cost function to match the parameters to training data. L1 (Lasso) and L2 (Lasso) are the two most frequent regularization types (Ridge). Instead of simply maximizing the aforementioned cost function, regularization imposes a limit on the size of the coefficients in order to avoid overfitting. L1 and L2 use distinct approaches to defining upper limits for coefficients, allowing L1 to conduct feature selection by setting coefficients to 0 for less relevant characteristics and reducing multi-collinearity, whereas L2 penalizes extremely large coefficients but does not set any to 0. There's also a parameter that regulates the constraint's weight, λ , to ensure that coefficients aren't penalized too harshly, resulting in underfitting.

It's a fascinating topic to investigate why L1 and L2 have different capacities owing to the 'squared' and 'absolute' values, and how λ affects the weight of regularized and original fit terms. We won't go into everything here, but it's well worth your time and effort to learn about. The steps below demonstrate how to convert an original cost function to a regularized cost function.

How Logistic Regression links with Neural Network?

We all know that Neural Networks are the foundation for Deep Learning. The best part is that Logistic Regression is intimately linked to Neural networks. Each neuron in the network may be thought of as a Logistic Regression; it contains input, weights, and bias, and you conduct a dot product on all of that before

applying any non-linear function. Furthermore, a neural network's last layer is a basic linear model (most of the time). That can be understood by visualization as shown below,

Take a deeper look at the “output layer,” and you'll notice that it's a basic linear (or logistic) regression: we have the input (hidden layer 2), the weights, a dot product, and finally a non-linear function, depends on the task. A helpful approach to thinking about neural networks is to divide them into two parts: representation and classification/regression. The first section (on the left) aims to develop a decent data representation that will aid the second section (on the right) is doing a linear classification/regression.

Hyperparameter Fine-tuning – Logistic Regression

There are no essential hyperparameters to adjust in logistic regression. Even though it has many parameters, the following three parameters might be helpful in fine-tuning for some better results,

Regularization (penalty) might be beneficial at times. Penalty – {'l1', 'l2', 'elasticnet', 'none'}, default='l2' The penalty strength is controlled by the **C parameter, which might be useful. C – float, default=1.0**

With different solvers, you might sometimes observe useful variations in performance or convergence.

Solver – {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default='lbfgs' **Note:**

The algorithm to use is determined by the penalty: Solver-supported penalties:

1. 'newton-cg' – ['l2', 'none']
2. 'lbfgs' – ['l2', 'none']
3. 'liblinear' – ['l1', 'l2']
4. 'sag' – ['l2', 'none']
5. 'saga' – ['elasticnet', 'l1', 'l2', 'none']

3.3 Advantages of Logistic Regression

1. Overfitting is less likely with logistic regression, although it can happen in high-dimensional datasets. In these circumstances, regularization (L1 and L2) techniques may be used to minimize over-fitting.
2. It works well when the dataset is linearly separable and has good accuracy for many basic data sets.
3. It is more straightforward to apply, understand, and train.
4. The inferences regarding the relevance of each characteristic are based on the anticipated parameters (trained weights). The association's orientation, positive or negative, is also specified. As a result, logistic regression may be used to determine the connection between the characteristics.
5. Unlike decision trees or support vector machines, this technique allows models to be readily changed to incorporate new data. Stochastic gradient descent can be used to update the data.
6. It is less prone to over-fitting in a low-dimensional dataset with enough training instances.
7. When the dataset includes linearly separable characteristics, Logistic Regression shows to be highly efficient.
8. It has a strong resemblance to neural networks. A neural network representation may be thought of as a collection of small logistic regression classifiers stacked together.
9. The training time of the logistic regression method is considerably smaller than that of most sophisticated algorithms, such as an Artificial Neural Network, due to its simple probabilistic interpretation.
10. Multinomial Logistic Regression is the name given to an approach that may easily be expanded to multi-class classification using a softmax classifier.

3.4 Disadvantages of Logistic Regression

1. Logistic Regression should not be used if the number of observations is fewer than the number of features; otherwise, it may result in overfitting.
2. Because it creates linear boundaries, we won't obtain better results when dealing with complex or non-linear data.
3. It's only good for predicting discrete functions. As a result, the Logistic Regression dependent variable is restricted to the discrete number set.
4. The average or no multicollinearity between independent variables is required for logistic regression.
5. Logistic regression needs a big dataset and enough training samples to identify all of the categories.
6. Because this method is sensitive to outliers, the presence of data values in the dataset that differs from the anticipated range may cause erroneous results.
7. Only significant and relevant features should be utilized to construct a model; otherwise, the model's probabilistic predictions may be inaccurate, and its predictive value may suffer.
8. Complex connections are difficult to represent with logistic regression. This technique is readily outperformed by more powerful and sophisticated algorithms such as Neural Networks.
9. Because logistic regression has a linear decision surface, it cannot address nonlinear issues. In real-world settings, linearly separable data is uncommon. As a result, non-linear features must be transformed, which may be done by increasing the number of features such that the data can be separated linearly in higher dimensions.
10. Based on independent variables, a statistical analysis model seeks to predict accurate probability outcomes. On high-dimensional datasets, this may cause the model to be over-fit on the training set, overstating the accuracy of predictions on the training set, and so preventing the model from accurately predicting outcomes on the test set. This is most common when the model is trained on a little amount of training data with many features. Regularization strategies should be explored on high-dimensional datasets to minimize over-fitting (but this makes

the model complex). The model may be under-fit on the training data if the regularization parameters are too high.

3.5 Application of Logistic Regression

All use cases where data must be categorized into multiple groups are covered by Logistic Regression.

Consider the following illustration:

1. Fraud detection in Credit card
2. Email spam or ham
3. Sentiment Analysis in Twitter analysis
4. Image segmentation, recognition, and classification – X-rays, Scans
5. Object detection through video ½ . Handwriting recognition
7. Disease prediction – Diabetes, Cancer, Parkinson etc...

3.6 BEAUTIFUL SOUP:

- Beautiful Soup is a Python library that is used for web scraping purposes to pull the data out of HTML and XML files. It creates a parse tree from page source code that can be used to extract data in a hierarchical and more readable manner.
- It is a complete framework for web-scraping or crawling. Beautiful Soup is a parsing library which also does a pretty good job of fetching contents from URL and allows you to parse certain parts of them without any hassle. It only fetches the contents of the URL that you give and then stops.
- Web scraping or Web data extraction is a method to collect data from websites. ... Most of this data is unstructured data in an HTML format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications.

- Comparing selenium vs BeautifulSoup allows you to see that BeautifulSoup is more user-friendly and allows you to learn faster and begin web scraping smaller tasks easier. Selenium on the other hand is important when the target website has a lot of java elements in its code.

3.7 TKINTER:

- Tkinter is a library written in Python that is widely used to create GUI applications. It is very easy to build GUI using Tkinter and the process is even faster. Tkinter has several widgets that can be used while developing GUI. These include buttons, radio buttons, checkboxes, etc
- ML requires continuous data processing, and Python's libraries let you access, handle and transform data. These are some of the most widespread libraries you can use for ML and AI: Scikit-learn for handling basic ML algorithms like clustering, linear and logistic regressions, regression, classification, and others.
- tkinter offers access to geometric configuration of the widgets which organize the widgets in parent windows in python for data science. Mainly three geometry manager classes class are there.
- 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. ... Add one or more of the above-mentioned widgets to the GUI application.
- If your goal is to learn how to create GUIs, tkinter is arguably one of the best toolkits there is to reach that goal. It's simple and easy to learn, and can provide a fantastic introduction to concepts you must master in order to create graphical desktop applications.

CHAPTER 4

MACHINE LEARNING

What are the 7 steps of machine learning?

7 STEPS OF MACHINE LEARNING

- Step 1: Gathering Data. ...
- Step 2: Preparing that Data. ...
- Step 3: Choosing a Model. ...
- Step 4: Training. ...

- Step 5: Evaluation. ...
- Step 6: Hyper parameter Tuning. ...
- Step 7: Prediction.

4.1 INTRODUCTION:

In this blog, we will discuss the workflow of a Machine learning project this includes all the steps required to build the proper machine learning project from scratch.

We will also go over data pre-processing, data cleaning, feature exploration and feature engineering and show the impact that it has on Machine Learning Model Performance. We will also cover a couple of the pre-modelling steps that can help to improve the model performance.

Python Libraries that would be need to achieve the task:

- 1.Numpy
- 2.Pandas
- 3.Sci-kitLearn
4. Matplotlib

UNDERSTANDING THE MACHINE LEARNING WORKFLOW

We can define the machine learning workflow in 3 stages.

1. Gathering data
2. Data pre-processing
3. Researching the model that will be best for the type of data
4. Training and testing the model

5. Evaluation

Okay but first let's start from the basics

What is the machine learning Model?

The machine learning model is nothing but a piece of code; an engineer or data scientist makes it smart through training with data. So, if you give garbage to the model, you will get garbage in return, i.e. the trained model will provide false or wrong prediction

4.2 Gathering Data

The process of gathering data depends on the type of project we desire to make, if we want to make an ML project that uses real-time data, then we can build an IoT system that using different sensors data. The data set can be collected from various sources such as a file, database, sensor and many other such sources but the collected data cannot be used directly for performing the analysis process as there might be a lot of missing data, extremely large values, unorganized text data or noisy data. Therefore, to solve this problem Data Preparation is done.

We can also use some free data sets which are present on the internet. Kaggle and UCI Machine learning Repository are the repositories that are used the most for making Machine learning models. Kaggle is one of the most visited websites that is used for practicing machine learning algorithms, they also host competitions in which people can participate and get to test their knowledge of machine learning.

4.3 Data pre-processing

Data pre-processing is one of the most important steps in machine learning. It is the most important step that helps in building machine learning models more accurately. In machine learning, there is an 80/20 rule. Every data scientist

should spend 80% time for data pre-processing and 20% time to actually perform the analysis.

4.3.1 What is data pre-processing?

Data pre-processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn't feasible for the analysis.

Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data pre-processing.

Why do we need it?

As we know that data pre-processing is a process of cleaning the raw data into clean data, so that can be used to train the model. So, we definitely need data pre-processing to achieve good results from the applied model in machine learning and deep learning projects. Most of the real-world data is messy, some of these types of data are:

1. **Missing data:** Missing data can be found when it is not continuously created or due to technical issues in the application (IOT system).
2. **Noisy data:** This type of data is also called outliers, this can occur due to human errors (human manually gathering the data) or some technical problem of the device at the time of collection of data.
3. **Inconsistent data:** This type of data might be collected due to human errors (mistakes with the name or values) or duplication of data.

Three Types of Data

1. Numeric e.g. income, age

2. Categorical e.g. gender, nationality

3. Ordinal e.g. low/medium/high

4.3.2 How can data pre-processing be performed?

These are some of the basic pre — processing techniques that can be used to convert raw data.

1. **Conversion of data:** As we know that Machine Learning models can only handle numeric features, hence categorical and ordinal data must be somehow converted into numeric features.

2. **Ignoring the missing values:** Whenever we encounter missing data in the data set then we can remove the row or column of data depending on our need. This method is known to be efficient but it shouldn't be performed if there are a lot of missing values in the dataset.

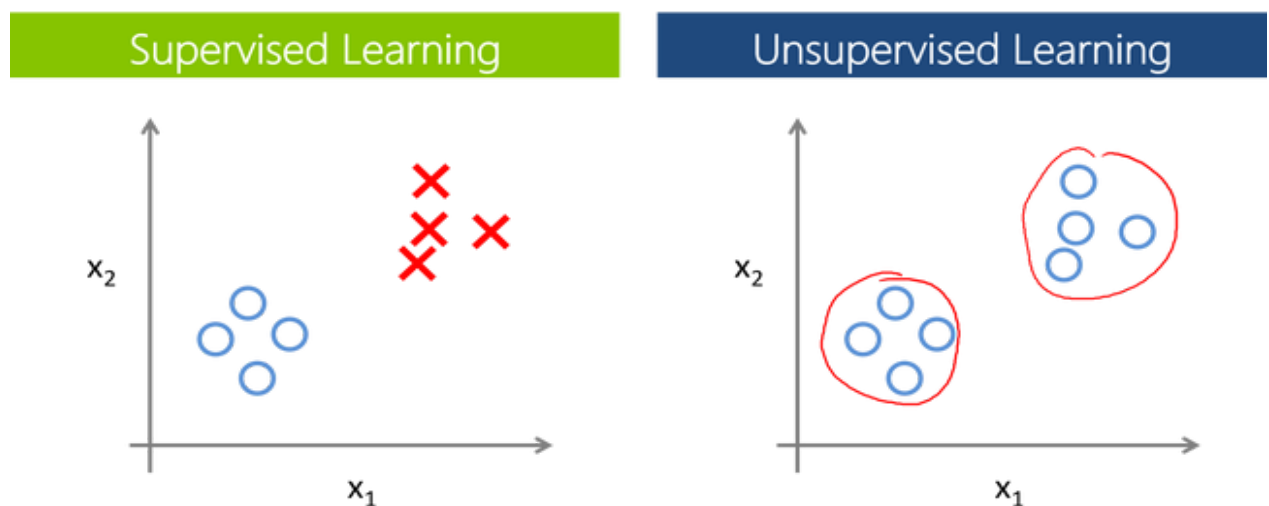
3. **Filling the missing values:** Whenever we encounter missing data in the data set then we can fill the missing data manually, most commonly the mean, median or highest frequency value is used.

4. **Machine learning:** If we have some missing data then we can predict what data shall be present at the empty position by using the existing data.

5. **Outliers detection:** There are some error data that might be present in our data set that deviates drastically from other observations in a data set. [Example: human weight = 800 Kg; due to mistyping of extra 0]

4.3.3 Researching the model that will be best for the type of data

Our main goal is to train the best performing model possible, using the pre-processed data.



4.3.3.1----Fig

SUPERVISED LEARNING:

In Supervised learning, an AI system is presented with data which is labelled, which means that each data tagged with the correct label.

The supervised learning is categorized into 2 other categories which are “**Classification**” and “**Regression**”.

4.4 CLASSIFICATION:

Classification problem is when the target variable is **categorical** (i.e. the output could be classified into classes — it belongs to either Class A or B or something else).

A classification problem is when the output variable is a category, such as “red” or “blue” , “disease” or “no disease” or “spam” or “not spam”.



4.4.1-- Fig

As shown in the above representation, we have 2 classes which are plotted on the graph i.e. red and blue which can be represented as ‘setosa flower’ and ‘versicolor flower’, we can image the X-axis as there ‘Sepal Width’ and the Y-axis as the ‘Sepal Length’, so we try to create the best fit line that separates both classes of flowers.

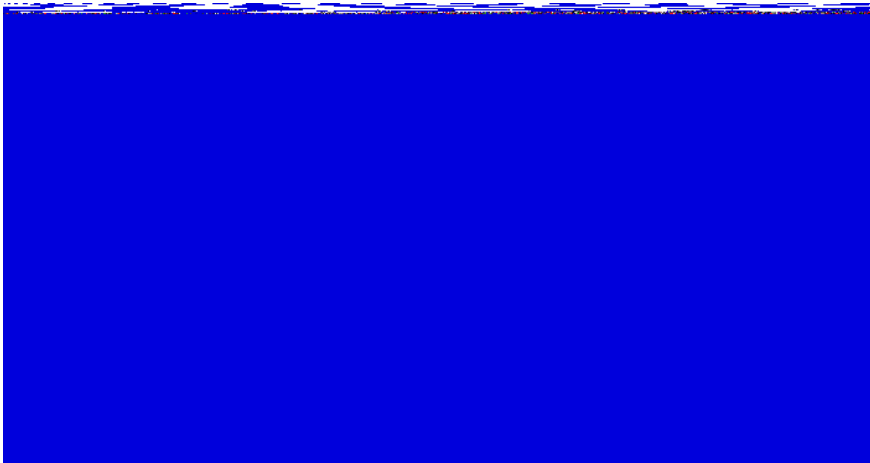
These some most used classification algorithms.

- **K-Nearest Neighbor**
- **Naive Bayes**

- **Decision Trees/Random Forest**
- **Support Vector Machine**
- **Logistic Regression**

4.5 REGRESSION:

While a **Regression** problem is when the target variable is **continuous** (i.e. the output is numeric).



4.5.1---Fig

As shown in the above representation, we can imagine that the graph's X-axis is the 'Test scores' and the Y-axis represents 'IQ'. So we try to create the **best fit** line in the given graph so that we can use that line to predict any approximate IQ that isn't present in the given data.

These some most used regression algorithms.

- **Linear Regression**
- **Support Vector Regression**

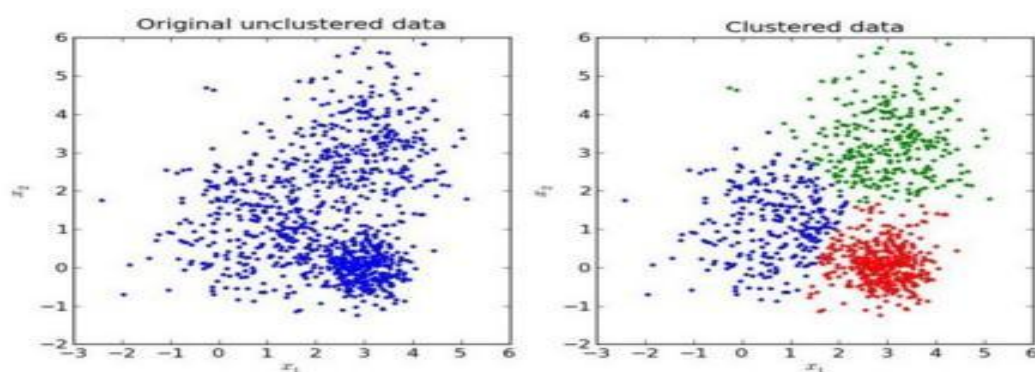
- Decision Tress/Random Forest
- Gaussian Progresses Regression
- Ensemble Methods

4.6 UNSUPERVISED LEARNING:

The unsupervised learning is categorized into 2 other categories which are “**Clustering**” and “**Association**”.

4.6.1 CLUSTERING:

A set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

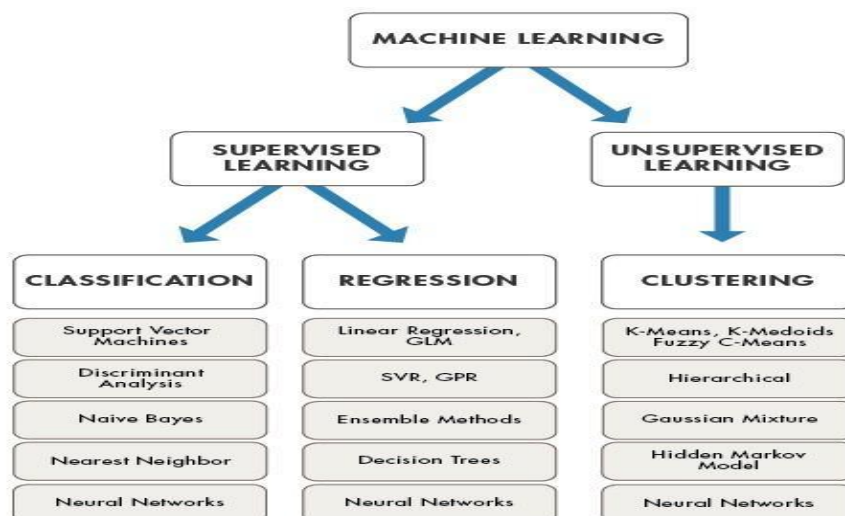


4.6.1.1---Fig

Methods used for clustering are:

- Gaussian mixtures
- K-Means Clustering
- Boosting
- Hierarchical Clustering
- K-Means Clustering
- Spectral Clustering

Overview of models under categories:



4.6.1.2---Fig

Training and testing the model on data:

For training a model we initially split the model into 3 three sections which are '**Training data**' , '**Validation data**' and '**Testing data**'. You train the classifier using '**training data set**', tune the parameters using '**validation set**' and then test the performance of your classifier on unseen '**test data set**'. An important point to note is that during training the classifier only the training and/or validation set is

available. The test data set must not be used during training the classifier. The test set will only be available during testing the classifier.



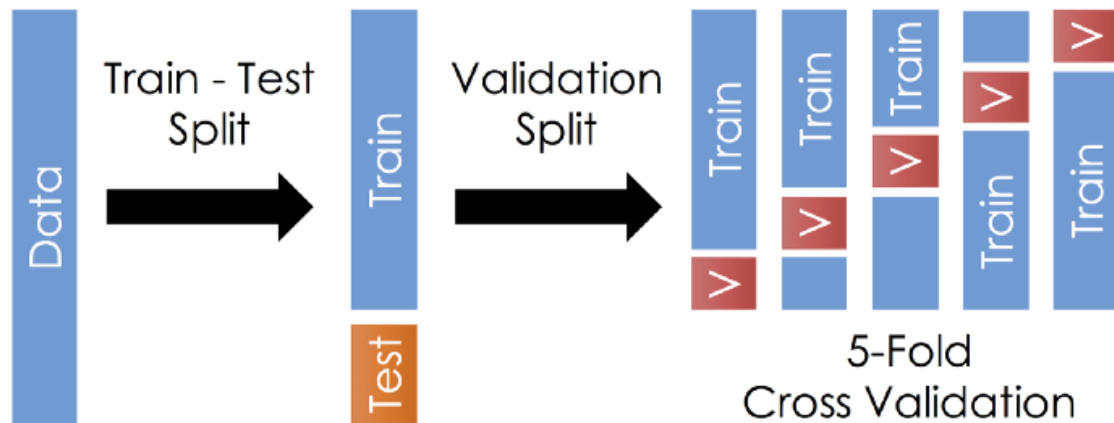
4.6.1.3---Fig

Training set:

The training set is the material through which the computer learns how to process information. Machine learning uses algorithms to perform the training part. A set of data used for learning, that is to fit the parameters of the classifier.

Validation set:

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. A set of unseen data is used from the training data to tune the parameters of a classifier.



4.6.1.4---Fig

Once the data is divided into the 3 given segments we can start the training process.

In a data set, a training set is implemented to build up a model, while a test (or validation) set is to validate the model built. Data points in the training set are excluded from the test (validation) set. Usually, a data set is divided into a training set, a validation set (some people use 'test set' instead) in each iteration, or divided into a training set, a validation set and a test set in each iteration.

The model uses any one of the models that we had chosen in step 3/ point 3. Once the model is trained we can use the same trained model to predict using the testing data i.e. the unseen data. Once this is done we can develop a confusion matrix, this tells us how well our model is trained. A confusion matrix has 4 parameters, which are '**True positives**', '**True Negatives**', '**False Positives**' and '**False Negative**'. We prefer that we get more values in the True negatives and true positives to get a more accurate model. The size of the Confusion matrix completely depends upon the number of classes.

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

4.6.1.5--Fig

True positives : These are cases in which we predicted TRUE and our predicted output is correct.

True negatives : We predicted FALSE and our predicted output is correct.

False positives : We predicted TRUE, but the actual predicted output is FALSE.

False negatives : We predicted FALSE, but the actual predicted output is TRUE.

We can also find out the accuracy of the model using the confusion matrix.

Accuracy = (True Positives + True Negatives) / (Total number of classes)

i.e. for the above example:

Accuracy = $(100 + 50) / 165 = 0.9090$ (90.9% accuracy)

Evaluation:

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future.

To improve the model we might tune the hyper-parameters of the model and try to improve the accuracy and also looking at the confusion matrix to try to increase the number of true positives and true negatives.

Conclusion

In this blog, we have discussed the workflow a Machine learning project and gives us a basic idea of how a should the problem be tackled.

CHAPTER 5

5.1 SYSTEM DESIGN:

Modules:

- Data Collection
- Data Cleaning
- Pre-processing
- Exploratory Data Analysis(EDA)
- Auditioning Models

5.1.1 Data collection:

Collecting data allows you to capture a record of past events so that we can use data analysis to find recurring patterns. From those patterns, you build predictive models using machine learning algorithms that look for trends and predict future changes. Predictive models are only as good as the data from which they are built, so good data collection practices are crucial to developing high-performing models. The data need to be error-free (garbage in, garbage out) and contain relevant information for the task at hand. For example, a loan default model would not benefit from tiger population sizes but could benefit from gas prices over time.

In this module, we collect the data from kaggle dataset archives. This dataset contains the information of divorce in previous years.

5.1.2 Data-Cleaning:

Data cleaning is a critically important step in any machine learning project. In this module data cleaning is done to prepare the data for analysis by removing or modifying the data that may be incorrect, incomplete, duplicated or improperly formatted. In tabular data, there are many different statistical analysis and data visualization techniques you can use to explore your data in order to identify data cleaning operations you may want to perform.

5.1.3 Pre-processing:

We will build a pre-processing function that helped change the text to lowercase, remove punctuation's, lemmatise related words down to a common base word.

5.1.4 Exploratory Data Analysis(EDA):

Using a Count Vectorizer, we analysed the most-used words from each subreddit to understand our data better (The words used were similar with some subtle differences).

CHAPTER 6

CONCLUSIONS:

- Cyberbullying has become a severe problem in modern societies. This paper proposed a cyber-bully detection model whereby several classifiers based on TF-IDF and Word2Vec feature extraction have been used. Furthermore, various methods of text classification based on machine learning were investigated.
- The experiments were conducted on a global Twitter dataset. The experimental results indicate that LR achieved the best accuracy and F1 score in our dataset, where the classification accuracy and F1 score are 90.57% and 0.9280, respectively.
- Meanwhile, there is a slight difference between LR, SGD, and LGBM classifier performance, where SGD achieved an accuracy of 90.6%, but the F1 score was lower than LR. However, the LGBM classifier achieved an accuracy of 90.55%, and the F1 score was 0.9271. This means that LR performs better than other classifiers. Moreover, during the experiments, it was observed that LR performs better as data size increases and obtains the best prediction time compared to other classifiers used in this study.
- Therefore, SGD performs almost as LR, but the error is not minimal as in LR. The feature extraction is a critical aspect in machine learning to enhance the detection accuracy. In this paper, we did not investigate many feature extraction techniques. Thus, one of the improvements is to incorporate and test different feature extractions to improve the detection rate of both classifiers LR and SGD. Another limitation that we are working on is building a real-time cyberbully detection platform, which will be useful to instantly detect and prevent the cyberbully. Another

research direction is working on cyberbully detection in various languages, mainly in an Arabic context.

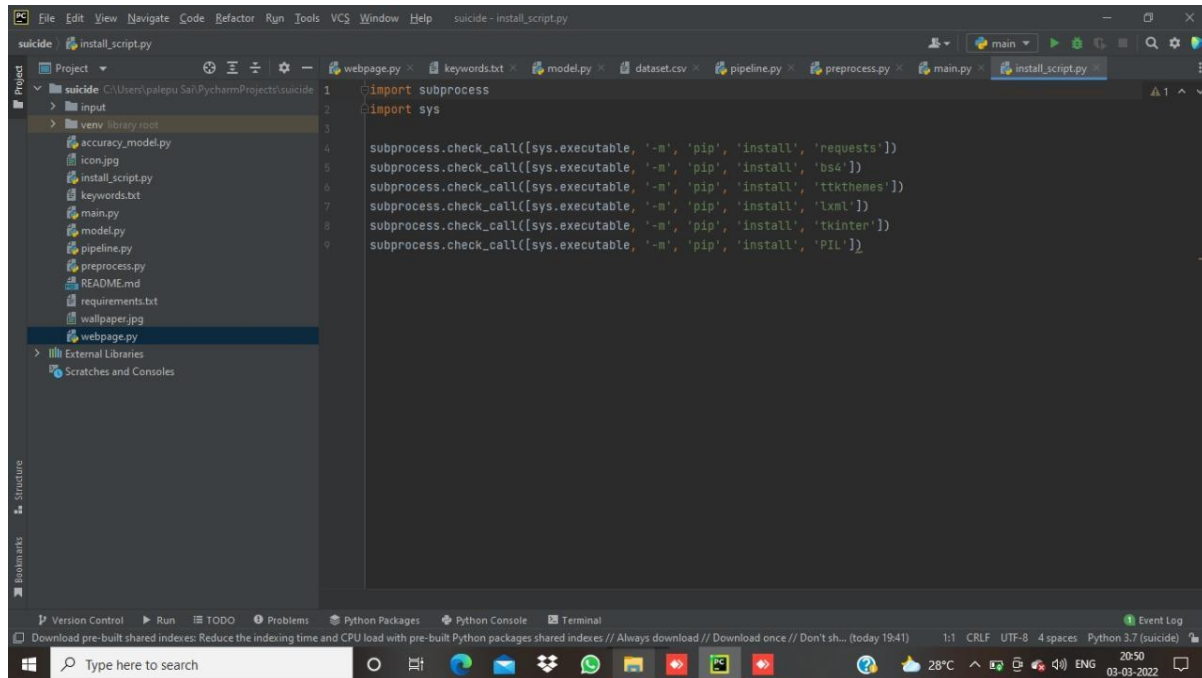
REFERENCE PAPERS:

1. James SL, Abate D, Abate KH, Abay SM, Abbafati C, Abbasi N, et al. Global, regional, and national incidence, prevalence, and years lived with disability for 354 diseases and injuries for 195 countries and territories, 1990–2017: a systematic analysis for the Global Burden of Disease Study 2017. *Lancet*. (2018) 392:1789–858. doi: 10.1016/S0140-6736(18)32279-7
2. Kessler RC, Birnbaum H, Bromet E, Hwang I, Sampson N, Shahly V. Age differences in major depression: results from the National Comorbidity Survey Replication (NCS-R). *Psychol Med*. (2010) 40:225–37. doi: 10.1017/S0033291709990213
3. Hodgetts S, Gallagher P, Stow D, Ferrier IN, O'Brien JT. The impact and measurement of social dysfunction in late-life depression: an evaluation of current methods with a focus on wearable technology. *Int J Geriatr Psychiatry*. (2017) 32:247–55. doi: 10.1002/gps.4632
4. Fiske A, Wetherell JL, Gatz M. Depression in older adults. *Annu Rev Clin Psychol*. (2009) 5:363–89. doi: 10.1146/annurev.clinpsy.032408.153621
5. Rodda J, Walker Z, Carter J. Depression in older adults. *BMJ*. (2011) 343:d5219–d5219. doi: 10.1136/bmj.d5219
6. “Suicide.” <https://www.who.int/news-room/fact-sheets/detail/suicide> (accessed Nov. 24, 2020).
7. “GHO | World Health Statistics data visualizations dashboard | Suicide,” WHO.
8. “Facebook artificial intelligence spots suicidal users - BBC News.” <https://www.bbc.com/news/technology-39126027> (accessed Nov. 24, 2020).

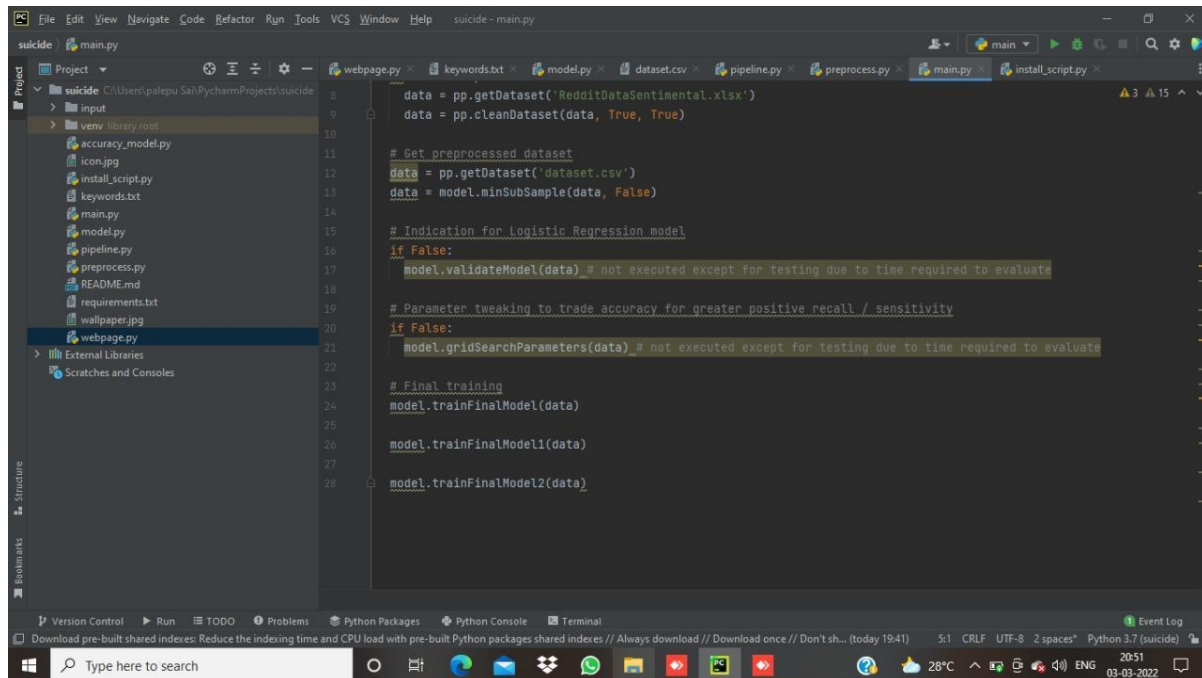
9. «Rekomendacii po rasprostraneniyu v SMI informacii o sluchayah samoubijstva».
https://www.rospotrebnadzor.ru/documents/details.php?ELEMENT_ID=6735 (accessed Nov. 24, 2020).
10. «O deyatel'nosti Rospotrebnadzora po predotvrashcheniyu samoubijstv sredi detej i podrostkov - RSS - Oficial'nyj sajt Rospotrebnadzora».
http://11.rospotrebnadzor.ru/rss_all/-/asset_publisher/Kq6J/content/id/382348 (accessed Nov. 24, 2020).
11. 11. «Chislo detskih suicidov v Rossii v 2016 godu vyroslo pochtu na 60%».
<https://www.interfax.ru/russia/554375> (accessed Nov. 24, 2020).
12. V. Chandler, «Google and suicides: what can we learn about the use of internet to prevent suicides?», *Public Health*, vol. 154, pp. 144–150, Jan. 2018, doi: 10.1016/j.puhe.2017.10.016.
13. L. Biddle et al., «Suicide and the Internet: Changes in the accessibility of suicide-related information between 2007 and 2014», *J. Affect. Disord.*, vol. 190, pp. 370–375, Jan. 2016, doi: 10.1016/j.jad.2015.10.028.
14. «Tor Project | Anonymity Online.» <https://www.torproject.org/> (accessed Nov. 27, 2020).
15. C. M. Mörch, L. P. Côté, L. Corthésy-Blondin, L. Plourde-Léveillé, L. Dargis, and B. L. Mishara, «The Darknet and suicide», *J. Affect. Disord.*, vol. 241, pp. 127–132, Dec. 2018, doi: 10.1016/j.jad.2018.08.028.
16. K. P. Kumar, N. Jaisankar, and N. Mythili, «An efficient technique for detection of suspicious malicious web site», *Journal of Advances in Information Technology*, vol. 2, no. 4, 2011.
17. B. Feinstein, D. Peck, and I. SecureWorks, «Caffeine monkey: Automated collection, detection and analysis of malicious javascript», *Black Hat USA*, vol. 2007, 2007.
18. C. N. Dos Santos and M. Gatti, «Deep convolutional neural networks for sentiment analysis of short texts.», in *COLING*, pp. 69–78, 2014.
19. X. Zhang, J. Zhao, and Y. LeCun, «Character-level convolutional networks for text classification», in *Advances in neural information processing systems*, pp. 649–657, 2015.

20.J. Saxe and K. Berlin, “Deep neural network based malware detection using two dimensional binary program features,” in *Malicious and Unwanted Software (MALWARE)*, 2015 10th International Conference on, pp. 11–20, IEEE, 2015.

OUTPUTS:



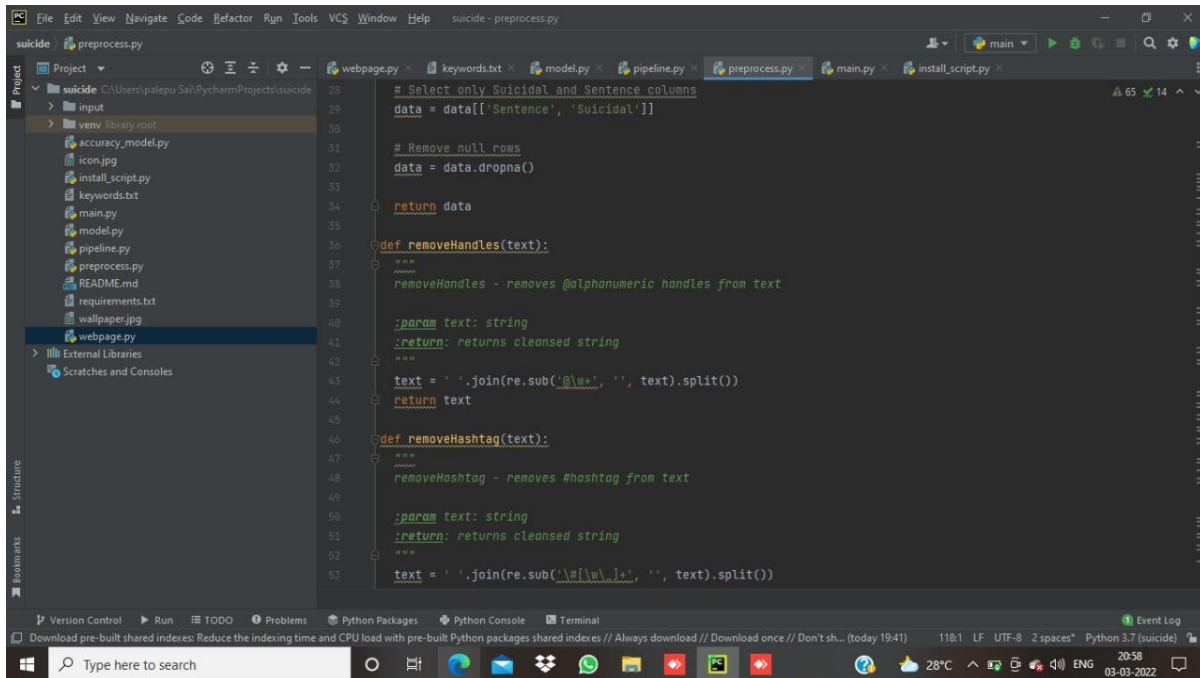
```
1 import subprocess
2 import sys
3
4 subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'requests'])
5 subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'bs4'])
6 subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'ttkthemes'])
7 subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'lxml'])
8 subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'tkinter'])
9 subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'PIL'])
```



```
8 data = pp.getDataset('RedditDataSentimental.xlsx')
9 data = pp.cleanDataset(data, True, True)
10
11 # Set preprocessed dataset
12 data = pp.getDataset('dataset.csv')
13 data = model.minSubSample(data, False)
14
15 # Indication for Logistic Regression model
16 if False:
17     model.validateModel(data) # not executed except for testing due to time required to evaluate
18
19 # Parameter tweaking to trade accuracy for greater positive recall / sensitivity
20 if False:
21     model.gridSearchParameters(data) # not executed except for testing due to time required to evaluate
22
23 # Final training
24 model.trainFinalModel(data)
25
26 model.trainFinalModel1(data)
27
28 model.trainFinalModel2(data)
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help suicide - main.py
suicide main.py
1 import model
2 import preprocess as pp
3
4 shouldPreprocess = False
5
6 if __name__ == '__main__':
7     if shouldPreprocess:
8         data = pp.getDataset('RedditDataSentimental.xlsx')
9         data = pp.cleanDataset(data, True, True)
10
11     # Get preprocessed dataset
12     data = pp.getDataset('dataset.csv')
13     data = model.minSubSample(data, False)
14
15     # Indication for Logistic Regression model
16     if False:
17         model.validateModel(data) # not executed except for testing due to time required to evaluate
18
19     # Parameter tweaking to trade accuracy for greater positive recall / sensitivity
20     if False:
21         model.gridSearchParameters(data) # not executed except for testing due to time required to evaluate
22
23     # Final training
24     model.trainFinalModel(data)
25
26     model.trainFinalModel1(data)
27
28 Version Control Run TODO Problems Python Packages Python Console Terminal
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't sh... (today 1941) 5:1 CRLF UTF-8 2 spaces Python 3.7 (suicide)
Type here to search 28°C 03-03-2022
```

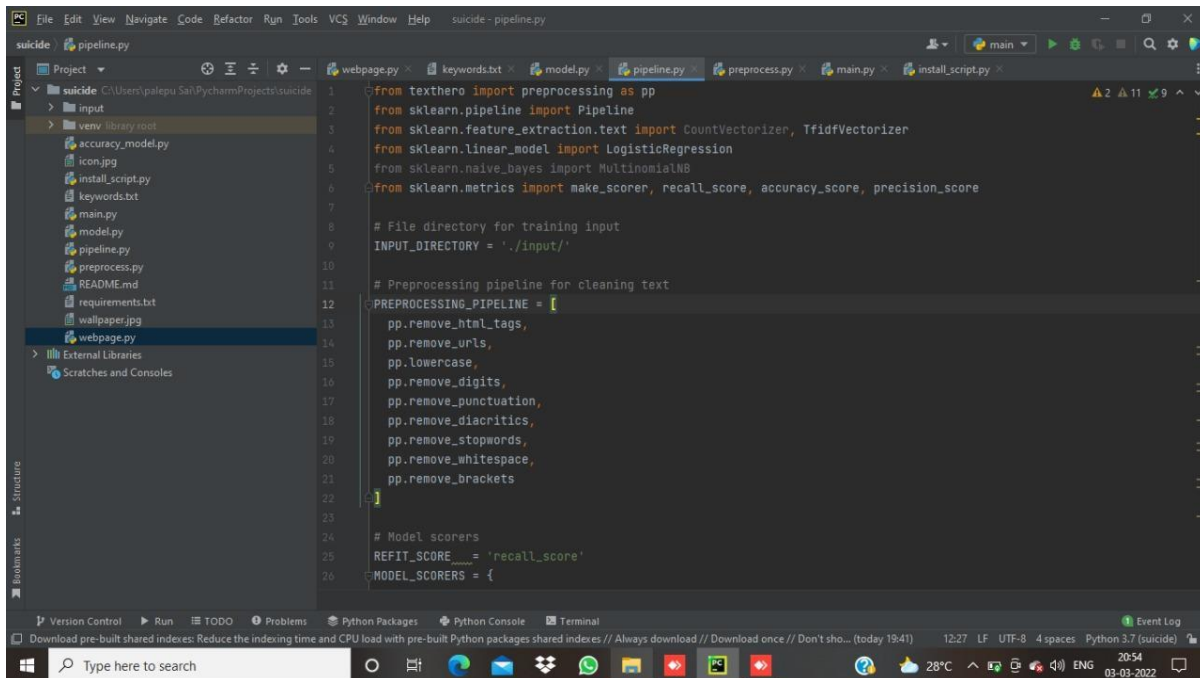
```
File Edit View Navigate Code Refactor Run Tools VCS Window Help suicide - pipeline.py
suicide pipeline.py
Project
  suicide
    input
    venv
    accuracy_model.py
    icon.jpg
    instal_script.py
    keywords.txt
    main.py
    model.py
    pipeline.py
    preprocess.py
    README.md
    requirements.txt
    wallpaper.jpg
    webpage.py
    External Libraries
    Scratches and Consoles
41 'tfidf_ngram_range': [(1, 2)],
42 'clf_penalty': ['l2'],
43 'clf_max_iter': [1000],
44 'clf_C': [0.25, 0.5, 1, 2, 4]
45 }
46 },
47 'fe=tfidf+trigram+default_c=LR': {
48     'Pipeline': Pipeline([
49         ('tfidf', TfidfVectorizer()),
50         ('clf', LogisticRegression())
51     ]),
52     'Parameters': {
53         'tfidf_norm': ['l1', 'l2'],
54         'tfidf_ngram_range': [(1, 3)],
55         'clf_penalty': ['l2'],
56         'clf_max_iter': [1000],
57         'clf_C': [0.25, 0.5, 1, 2, 4]
58     }
59 }
60 }
61
Version Control Run TODO Problems Python Packages Python Console Terminal
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't sh... (today 1941) 12:27 LF UTF-8 4 spaces Python 3.7 (suicide)
Type here to search 28°C 03-03-2022
```



The screenshot shows the PyCharm IDE with the 'suicide' project open. The 'preprocess.py' file is selected in the Project view on the left. The main editor displays the following Python code:

```
28 # Select only Suicidal and Sentence columns
29 data = data[['Sentence', 'Suicidal']]
30
31 # Remove null rows
32 data = data.dropna()
33
34 return data
35
36 def removeHandles(text):
37     """
38     removeHandles - removes @alphanumeric handles from text
39
40     :param text: string
41     :return: returns cleansed string
42     """
43     text = ' '.join(re.sub('@[a-zA-Z0-9_]+', '', text).split())
44     return text
45
46 def removeHashtag(text):
47     """
48     removeHashtag - removes #hashtag from text
49
50     :param text: string
51     :return: returns cleansed string
52     """
53     text = ' '.join(re.sub('#[a-zA-Z0-9_]+', '', text).split())
```

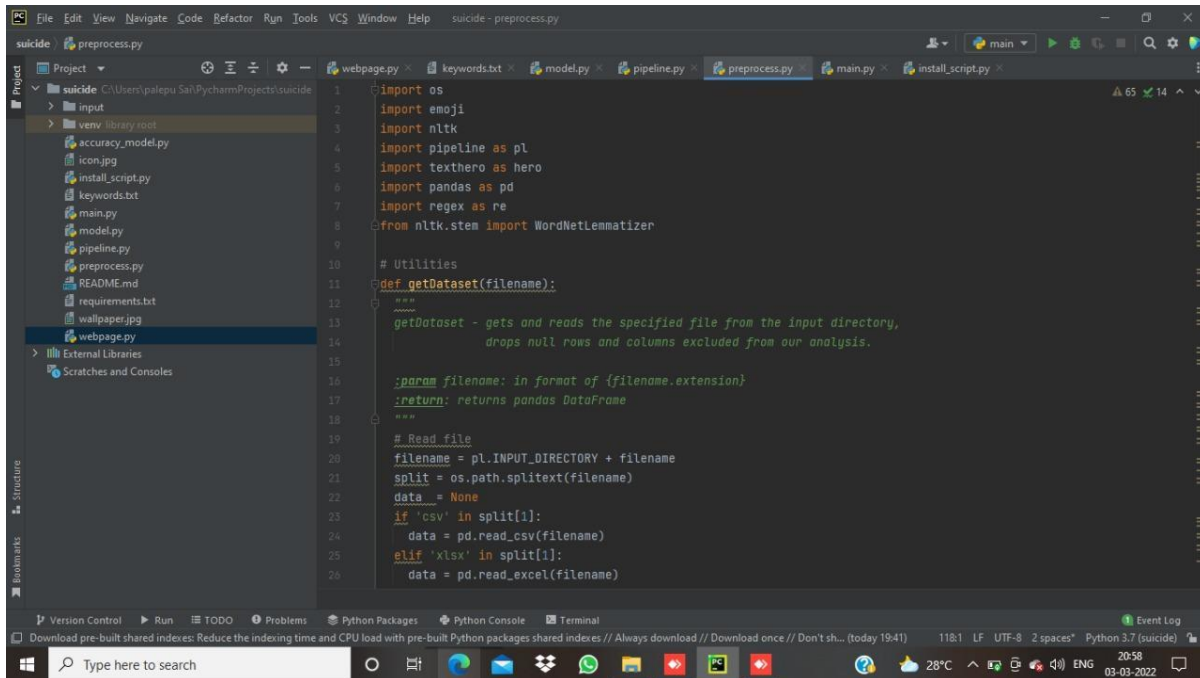
The bottom status bar shows the file encoding as UTF-8, 2 spaces, and Python 3.7 (suicide). The system tray at the bottom indicates a temperature of 28°C and the date 03-03-2022.



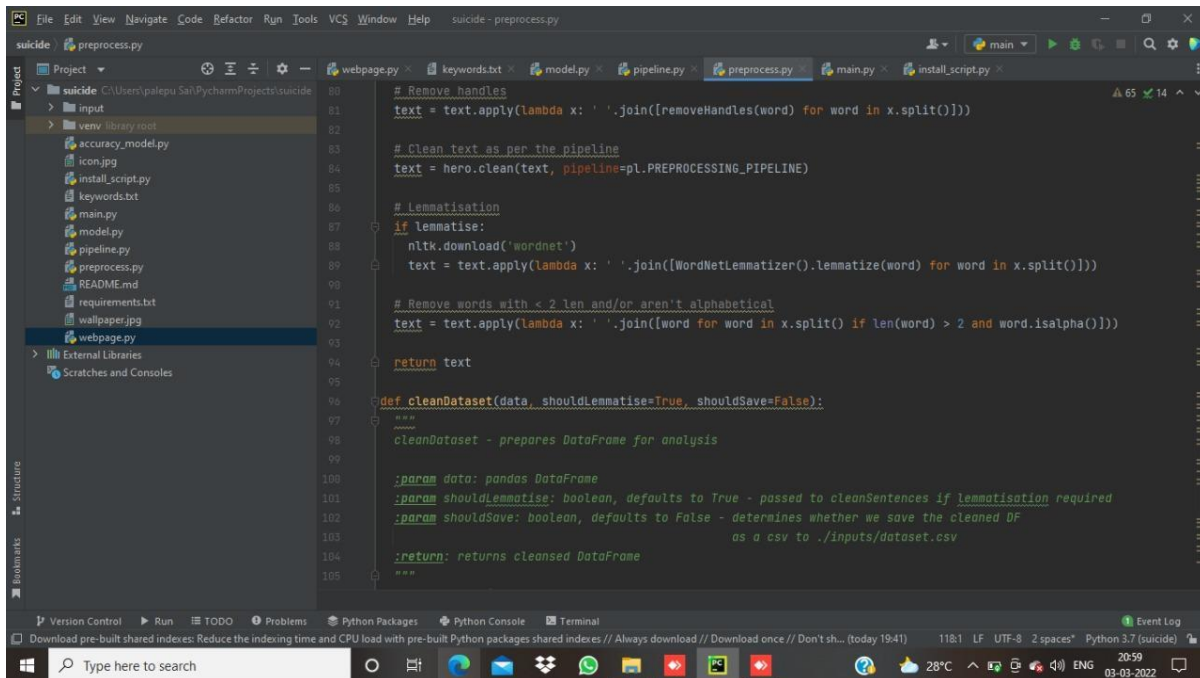
The screenshot shows the PyCharm IDE with the 'suicide' project open. The 'pipeline.py' file is selected in the Project view on the left. The main editor displays the following Python code:

```
1 from textthero import preprocessing as pp
2 from sklearn.pipeline import Pipeline
3 from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
4 from sklearn.linear_model import LogisticRegression
5 from sklearn.naive_bayes import MultinomialNB
6 from sklearn.metrics import make_scorer, recall_score, accuracy_score, precision_score
7
8 # File directory for training input
9 INPUT_DIRECTORY = './input/'
10
11 # Preprocessing pipeline for cleaning text
12 PREPROCESSING_PIPELINE = [
13     pp.remove_html_tags,
14     pp.remove_urls,
15     pp.lowercase,
16     pp.remove_digits,
17     pp.remove_punctuation,
18     pp.remove_diacritics,
19     pp.remove_stopwords,
20     pp.remove_whitespace,
21     pp.remove_brackets
22 ]
23
24 # Model scorers
25 REFIT_SCORE__ = 'recall_score'
26 MODEL_SCORERS = {
```

The bottom status bar shows the file encoding as UTF-8, 4 spaces, and Python 3.7 (suicide). The system tray at the bottom indicates a temperature of 28°C and the date 03-03-2022.



```
1 import os
2 import emoji
3 import nltk
4 import pipeline as pl
5 import textthero as hero
6 import pandas as pd
7 import regex as re
8 from nltk.stem import WordNetLemmatizer
9
10 # Utilities
11 def getDataset(filename):
12     """
13     getDataset - gets and reads the specified file from the input directory,
14     drops null rows and columns excluded from our analysis.
15
16     :param filename: in format of {filename.extension}
17     :return: returns pandas DataFrame
18     """
19     # Read file
20     filename = pl.INPUT_DIRECTORY + filename
21     split = os.path.splitext(filename)
22     data = None
23     if 'csv' in split[1]:
24         data = pd.read_csv(filename)
25     elif 'xlsx' in split[1]:
26         data = pd.read_excel(filename)
```



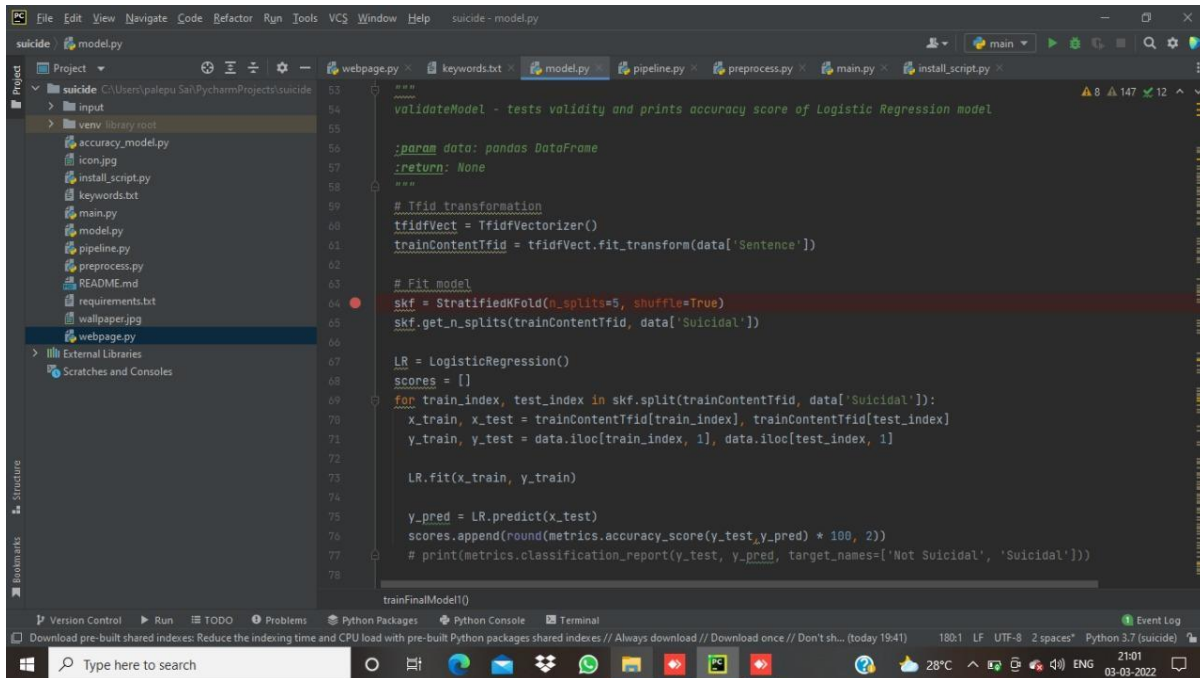
```
88 # Remove handles
89 text = text.apply(lambda x: ''.join([removeHandles(word) for word in x.split()]))
90
91 # Clean text as per the pipeline
92 text = hero.clean(text, pipeline=pl.PREPROCESSING_PIPELINE)
93
94 # Lemmatization
95 if lemmatise:
96     nltk.download('wordnet')
97     text = text.apply(lambda x: ''.join([WordNetLemmatizer().lemmatize(word) for word in x.split()]))
98
99 # Remove words with < 2 len and/or aren't alphabetical
100 text = text.apply(lambda x: ''.join([word for word in x.split() if len(word) > 2 and word.isalpha()]))
101
102 return text
103
104 def cleanDataset(data, shouldLemmatise=True, shouldSave=False):
105     """
106     cleanDataset - prepares DataFrame for analysis
107
108     :param data: pandas DataFrame
109     :param shouldLemmatise: boolean, defaults to True - passed to cleanSentences if lemmatization required
110     :param shouldSave: boolean, defaults to False - determines whether we save the cleaned DF
111     as a csv to ./inputs/dataset.csv
112     :return: returns cleansed DataFrame
113     """
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help suicide - preprocess.py
suicide preprocess.py
Project
  suicide C:\Users\palepu Sai\PycharmProjects\suicide
  > input
  > venv library root
    accuracy_model.py
    icon.jpg
    install_script.py
    keywords.txt
    main.py
    model.py
    pipeline.py
    preprocess.py
    README.md
    requirements.txt
    wallpaper.jpg
    webpage.py
  External Libraries
  Scratches and Consoles
Structure
  suicide
  > input
  > venv library root
    accuracy_model.py
    icon.jpg
    install_script.py
    keywords.txt
    main.py
    model.py
    pipeline.py
    preprocess.py
    README.md
    requirements.txt
    wallpaper.jpg
    webpage.py
  External Libraries
  Scratches and Consoles
54 return text
55
56 def removeEmoji(text):
57     """
58     removeEmoji - removes all instances of emoji in the string passed.
59
60     :param text: string
61     :return: returns cleansed string
62     """
63     text = emoji.get_emoji_regexp().sub(u'', text)
64     return text
65
66 def cleanSentences(text, lemmatise=True):
67     """
68     cleanSentences - pre-processes text for analysis, optional lemmatisation (defaults to True)
69
70     :param text: array of strings
71     :param lemmatise: boolean, defaults True - determines whether function performs lemmatisation
72     :return: returns array of cleaned strings
73     """
74     # Remove emoji
75     text = text.apply(lambda x: ' '.join([removeEmoji(word) for word in x.split()]))
76
77     # Remove hashtag
78     text = text.apply(lambda x: ' '.join([removeHashtag(word) for word in x.split()]))
79
80 Version Control Run TODO Problems Python Packages Python Console Terminal
81 Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't sh... (today 1941) 118:1 LF UTF-8 2 spaces* Python 3.7 (suicide) 20:58 03-03-2022
```

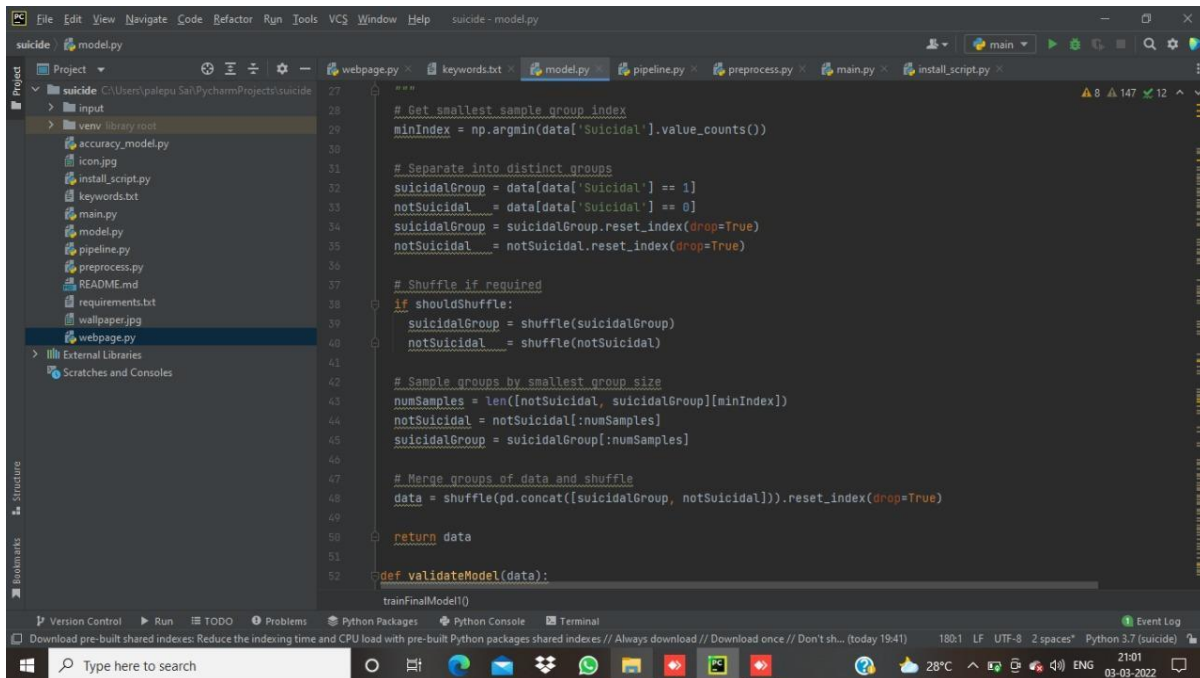
```
File Edit View Navigate Code Refactor Run Tools VCS Window Help suicide - preprocess.py
suicide preprocess.py
Project
  suicide C:\Users\palepu Sai\PycharmProjects\suicide
  > input
  > venv library root
    accuracy_model.py
    icon.jpg
    install_script.py
    keywords.txt
    main.py
    model.py
    pipeline.py
    preprocess.py
    README.md
    requirements.txt
    wallpaper.jpg
    webpage.py
  External Libraries
  Scratches and Consoles
Structure
  suicide
  > input
  > venv library root
    accuracy_model.py
    icon.jpg
    install_script.py
    keywords.txt
    main.py
    model.py
    pipeline.py
    preprocess.py
    README.md
    requirements.txt
    wallpaper.jpg
    webpage.py
  External Libraries
  Scratches and Consoles
98 cleanDataset - prepares DataFrame for analysis
99
100 :param data: pandas DataFrame
101 :param shouldLemmatise: boolean, defaults to True - passed to cleanSentences if lemmatisation required
102 :param shouldSave: boolean, defaults to False - determines whether we save the cleaned DF
103 as a csv to ./inputs/dataset.csv
104 :return: returns cleansed DataFrame
105
106 # Replace yes/no string datapoint as binary representation
107 data.loc[data['Suicidal'] == 'Yes', 'Suicidal'] = 1
108 data.loc[data['Suicidal'] == 'No', 'Suicidal'] = 0
109
110 # Preprocess string data
111 data['Sentence'] = cleanSentences(data['Sentence'], shouldLemmatise)
112
113 # Save input if required
114 if shouldSave:
115     data.to_csv(pl.INPUT_DIRECTORY + 'dataset.csv')
116
117 return data
118
119 Version Control Run TODO Problems Python Packages Python Console Terminal
120Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't sh... (today 1941) 118:1 LF UTF-8 2 spaces* Python 3.7 (suicide) 20:59 03-03-2022
```

```
1 import math
2
3 from sklearn.ensemble import RandomForestClassifier
4 from sklearn.metrics import confusion_matrix, roc_curve
5 import pipeline as pl
6 import preprocess as pp
7 import numpy as np
8 import pandas as pd
9 import seaborn as sns
10 from matplotlib import pyplot as plt
11 from pprint import pprint
12 from sklearn.utils import shuffle
13 from sklearn.feature_extraction.text import TfidfVectorizer
14 from sklearn.linear_model import LogisticRegression
15 from sklearn.model_selection import StratifiedKFold
16 from sklearn.model_selection import GridSearchCV
17 from sklearn.model_selection import train_test_split
18 from sklearn import metrics
19
20 def minSubSample(data, shouldShuffle=False):
21     """
22     minSubSample - prepares DataFrame by taking ?shuffled subsample from data, based on minimum size of groups, to bol
23
24     :param data: pandas DataFrame
25     :param shouldShuffle: boolean, defaults False - determines whether groups are shuffled
26     :return: returns DataFrame
27
28     trainFinalModel()
```

```
105 cv=StratifiedKFold(n_splits=10, random_state=None),
106 n_jobs=-1
107 )
108 gs_eval = gs_eval.fit(train_data, train_labels)
109
110 print(f'Best params for {pl.REFIT_SCORE}')
111 print(gs_eval.best_params_)
112
113 return gs_eval
114
115 def evaluateModels(train_data, train_labels):
116     """
117     evaluateModels - evaluate all models and parameters within pl.MODEL_PIPELINES
118
119     :param train_data: x_train data
120     :param train_labels: y_train data
121     :return: array (names of pipelines), pd DataFrame (cv_results_), array (best_params_)
122     """
123     names = []; results = []; params = []
124     for curPipeline in pl.MODEL_PIPELINES:
125         gs = evaluateModel(curPipeline, train_data, train_labels)
126         names.append(curPipeline)
127         results.append(pd.DataFrame(gs.cv_results_))
128         params.append(gs.best_params_)
129
130     return names, results, params
131
132 trainFinalModel()
```



```
53 """
54 validateModel - tests validity and prints accuracy score of Logistic Regression model
55
56 :param data: pandas DataFrame
57 :return: None
58 """
59 # TfIdf transformation
60 tfidfVect = TfidfVectorizer()
61 trainContentTfidf = tfidfVect.fit_transform(data['Sentence'])
62
63 # Fit model
64 skf = StratifiedKFold(n_splits=5, shuffle=True)
65 skf.get_n_splits(trainContentTfidf, data['Suicidal'])
66
67 LR = LogisticRegression()
68 scores = []
69 for train_index, test_index in skf.split(trainContentTfidf, data['Suicidal']):
70     x_train, x_test = trainContentTfidf[train_index], trainContentTfidf[test_index]
71     y_train, y_test = data.iloc[train_index, 1], data.iloc[test_index, 1]
72
73     LR.fit(x_train, y_train)
74
75     y_pred = LR.predict(x_test)
76     scores.append(round(metrics.accuracy_score(y_test, y_pred) * 100, 2))
77     # print(metrics.classification_report(y_test, y_pred, target_names=['Not Suicidal', 'Suicidal']))
78
79 trainFinalModel()
```



```
27 """
28 # Get smallest sample group index
29 minIndex = np.argmin(data['Suicidal'].value_counts())
30
31 # Separate into distinct groups
32 suicidalGroup = data[data['Suicidal'] == 1]
33 notSuicidal = data[data['Suicidal'] == 0]
34 suicidalGroup = suicidalGroup.reset_index(drop=True)
35 notSuicidal = notSuicidal.reset_index(drop=True)
36
37 # Shuffle if required
38 if shouldShuffle:
39     suicidalGroup = shuffle(suicidalGroup)
40     notSuicidal = shuffle(notSuicidal)
41
42 # Sample groups by smallest group size
43 numSamples = len([notSuicidal, suicidalGroup][minIndex])
44 notSuicidal = notSuicidal[:numSamples]
45 suicidalGroup = suicidalGroup[:numSamples]
46
47 # Merge groups of data and shuffle
48 data = shuffle(pd.concat([suicidalGroup, notSuicidal])).reset_index(drop=True)
49
50 return data
51
52 def validateModel(data):
53     trainFinalModel()
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help suicide - model.py
suicide - model.py
Project
  suicide
    input
    venv
    accuracy_model.py
    icon.jpg
    install_script.py
    keywords.txt
    main.py
    model.py
    pipeline.py
    preprocess.py
    README.md
    requirements.txt
    wallpaper.jpg
    webpage.py
  External Libraries
  Scratches and Consoles
Structure
  suicide
    input
    venv
    accuracy_model.py
    icon.jpg
    install_script.py
    keywords.txt
    main.py
    model.py
    pipeline.py
    preprocess.py
    README.md
    requirements.txt
    wallpaper.jpg
    webpage.py
  External Libraries
  Scratches and Consoles
132 def gridSearchParameters(data):
133     """
134     gridSearchParameters - hyper parameter tweaking as per pl model pipelines to trade accuracy for higher sensitivity
135
136     :param data: pandas DataFrame
137     :return: None
138     """
139     names, results, params = evaluateModels(data.iloc[:, 0], data.iloc[:, 1])
140
141     for i, x in enumerate(names):
142         print('Test -->' + x)
143         print('Results -->')
144         print(results[i].head())
145
146 def trainFinalModel(data):
147     """
148     trainFinalModel - final training model with best parameters
149
150     :param data: pandas DataFrame
151     :return: None
152     """
153     # Transform
154     tfidfVect = TfidfVectorizer(ngram_range=(1, 3), norm='l1')
155     trainContentTfidf = tfidfVect.fit_transform(data['Sentence'])
156
157     x_train, x_test, y_train, y_test = train_test_split(trainContentTfidf, data.iloc[:, 1], test_size=0.2)
158     trainFinalModel()
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help suicide - model.py
suicide - model.py
Project
  suicide
    input
    venv
    accuracy_model.py
    icon.jpg
    install_script.py
    keywords.txt
    main.py
    model.py
    pipeline.py
    preprocess.py
    README.md
    requirements.txt
    wallpaper.jpg
    webpage.py
  External Libraries
  Scratches and Consoles
Structure
  suicide
    input
    venv
    accuracy_model.py
    icon.jpg
    install_script.py
    keywords.txt
    main.py
    model.py
    pipeline.py
    preprocess.py
    README.md
    requirements.txt
    wallpaper.jpg
    webpage.py
  External Libraries
  Scratches and Consoles
79 # Print scores
80 scores = [{'Split': i, 'Accuracy': x} for i, x in enumerate(scores)]
81 print('____LR Indication____\n')
82 pprint(scores)
83 print('-----')
84
85 def evaluateModel(pipeline, train_data, train_labels):
86     """
87     evaluateModel - evaluate pipeline model from pl
88
89     :param pipeline: pipeline referring to MODEL_PIPELINES index
90     :param train_data: x_train data
91     :param train_labels: y_train data
92     :return: return GridSearchCV dataframe
93     """
94     # Get the current pipeline and its parameters
95     curPipeline = pl.MODEL_PIPELINES[pipeline]['Pipeline']
96     curPipelineParams = pl.MODEL_PIPELINES[pipeline]['Parameters']
97
98     # Grid search cv to evaluate the models for scorers
99     gs_eval = GridSearchCV(
100         curPipeline,
101         curPipelineParams,
102         scoring=pl.MODEL_SCORERS,
103         refit=pl.REFIT_SCORE,
104         error_score='raise',
105     )
106     trainFinalModel()
```

The screenshot shows the PyCharm IDE with the 'model.py' file open. The code implements a Logistic Regression model using the 'LogisticRegression' class from sklearn. It includes functions for fitting the model, predicting, and visualizing the results. The 'trainFinalModel1()' function is also defined.

```
# Fit model
LR = LogisticRegression(C=4, max_iter=1000, penalty='l2')
LR.fit(x_train, y_train)

# Predict
y_pred = LR.predict(x_test)
print('Logistic regression Accuracy Score : {round(metrics.accuracy_score(y_test, y_pred) * 100, 2)}%')
print(metrics.classification_report(y_test, y_pred))

# Visualise model prediction vs truth
sns.heatmap(metrics.confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='vlgmBu')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.show()

def trainFinalModel1(data):
    tfidfVect = TfidfVectorizer(ngram_range=(1, 3), norm='l1')
    trainContentTfidf = tfidfVect.fit_transform(data['Sentence'])

    x_train, x_test, y_train, y_test = train_test_split(trainContentTfidf, data.iloc[:, 1], test_size=0.2)
    model_Fashion = RandomForestClassifier(n_estimators=20)

    from sklearn.neighbors import KNeighborsClassifier
    knn = KNeighborsClassifier(n_neighbors=1) # n_neighbors means k
    knn.fit(x_train, y_train)
    prediction = knn.predict(x_test)

    trainFinalModel1()
```

The screenshot shows the PyCharm IDE with the 'model.py' file open. The code implements a Support Vector Machine (SVM) model using the 'SVC' class from sklearn. It includes functions for fitting the model, predicting, and visualizing the results. The 'trainFinalModel2()' function is also defined.

```
knn = KNeighborsClassifier(n_neighbors=1) # n_neighbors means k
knn.fit(x_train, y_train)
prediction = knn.predict(x_test)
print("{:.2f}% KNN accuracy score: {:.2f}%".format(2, knn.score(x_test, y_test) * 100))

print("Confusion Matrix of KNN:\n", confusion_matrix(prediction, y_test))

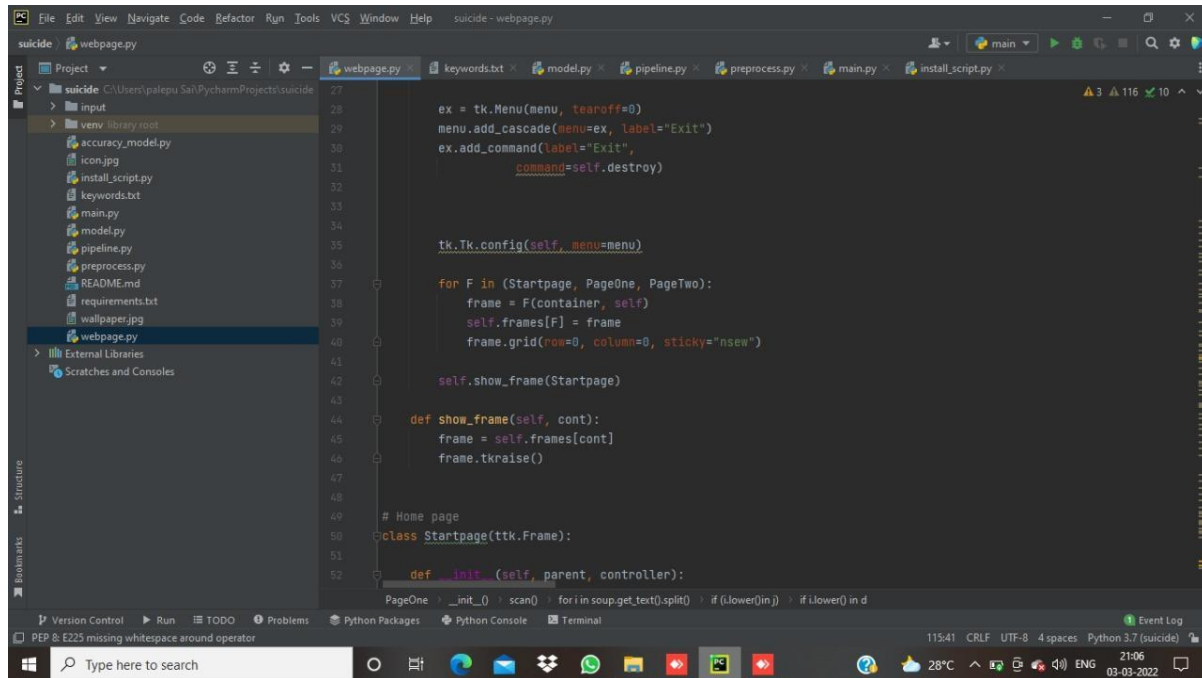
def trainFinalModel2(data):
    tfidfVect = TfidfVectorizer(ngram_range=(1, 3), norm='l1')
    trainContentTfidf = tfidfVect.fit_transform(data['Sentence'])

    x_train, x_test, y_train, y_test = train_test_split(trainContentTfidf, data.iloc[:, 1], test_size=0.2)

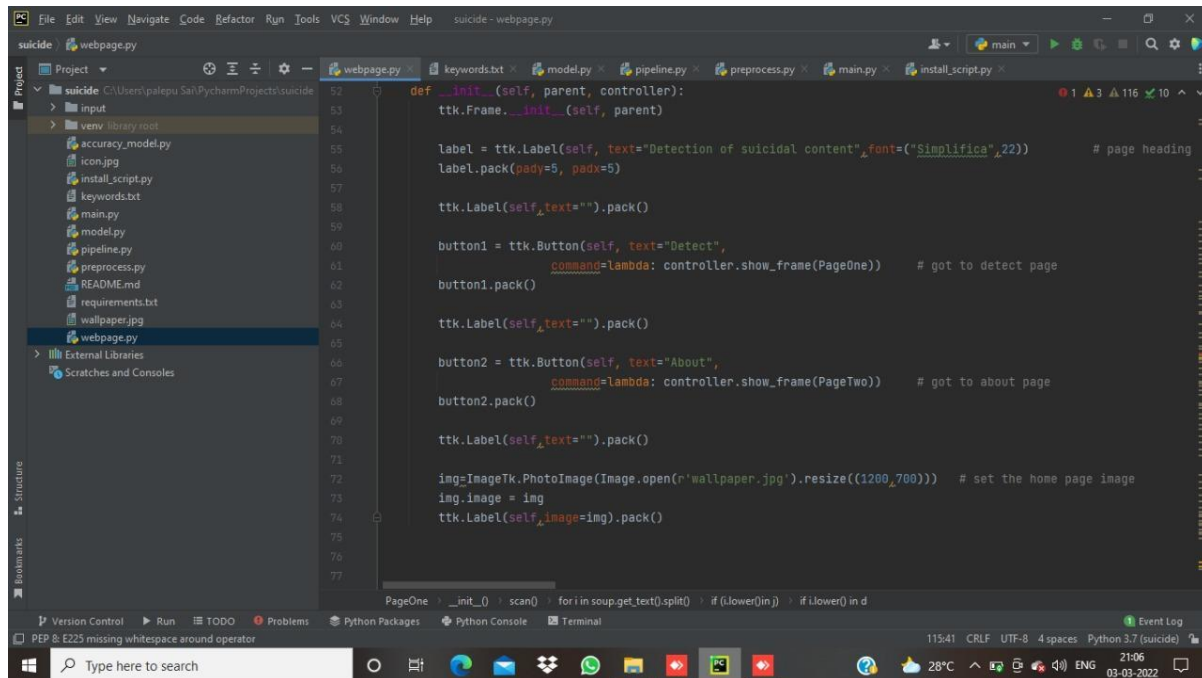
    from sklearn import svm

    clf_3 = svm.SVC(random_state=0).fit(x_train, y_train)
    pred = clf_3.predict(x_test)
    SVM_score = metrics.accuracy_score(pred, y_test) * 100
    metrics.accuracy_score(pred, y_test)
    print("Accuracy of Support Vector Machine: {:.2f}%".format(SVM_score))
    print("Confusion Matrix of SVM :\n", confusion_matrix(pred, y_test))

    trainFinalModel2()
```



```
27 ex = tk.Menu(menu, tearoff=0)
28 menu.add_cascade(menu=ex, label="Exit")
29 ex.add_command(label="Exit",
30               command=self.destroy)
31
32
33
34
35 tk.Tk.config(self, menu=menu)
36
37 for F in (Startpage, PageOne, PageTwo):
38     frame = F(container, self)
39     self.frames[F] = frame
40     frame.grid(row=0, column=0, sticky="nsew")
41
42 self.show_frame(Startpage)
43
44 def show_frame(self, cont):
45     frame = self.frames[cont]
46     frame.tkraise()
47
48
49 # Home page
50 class Startpage(tk.Frame):
51
52     def __init__(self, parent, controller):
53         PageOne.__init__(self, parent)
54         for i in soup.get_text().split():
55             if (i.lower() in j):
56                 if (i.lower() in d)
```



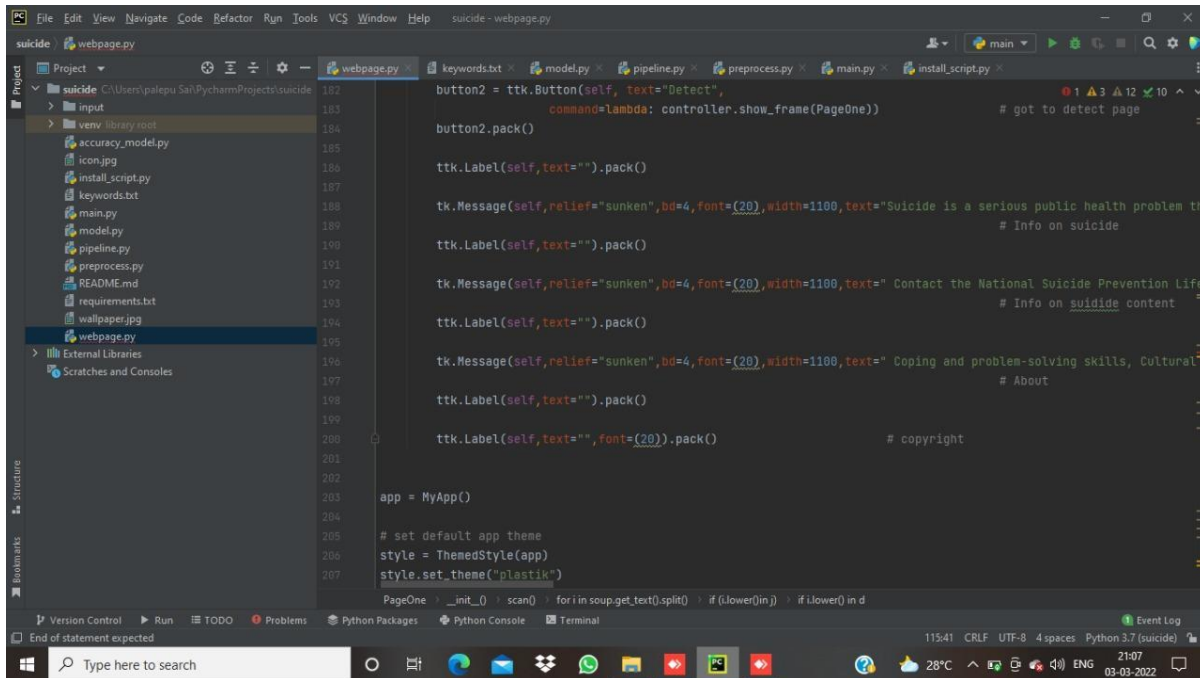
```
52 def __init__(self, parent, controller):
53     ttk.Frame.__init__(self, parent)
54
55     label = ttk.Label(self, text="Detection of suicidal content", font=("Simplifica", 22)) # page heading
56     label.pack(pady=5, padx=5)
57
58     ttk.Label(self, text="").pack()
59
60     button1 = ttk.Button(self, text="Detect",
61                         command=lambda: controller.show_frame(PageOne)) # got to detect page
62     button1.pack()
63
64     ttk.Label(self, text="").pack()
65
66     button2 = ttk.Button(self, text="About",
67                         command=lambda: controller.show_frame(PageTwo)) # got to about page
68     button2.pack()
69
70     ttk.Label(self, text="").pack()
71
72     img = ImageTk.PhotoImage(Image.open(r'wallpaper.jpg').resize((1200, 700))) # set the home page image
73     img_image = img
74     ttk.Label(self, image=img).pack()
75
76
77
```

```
78 ***** PAGES *****
79 # Detect page
80 class PageOne(tk.Frame):
81
82     def __init__(self, parent, controller):
83         tk.Frame.__init__(self, parent)
84         label = ttk.Label(self, text="Detect", font=("Simplifica", 22)) # page heading
85         label.pack(pady=5, padx=5)
86
87         ttk.Label(self, text="\n").pack()
88
89         ttk.Label(self, text="Enter a webpage", font=(18)).pack()
90         text = tk.Entry(self, font=(26), width=70, bg="lightgray") # textbox to enter a website
91         text.pack()
92
93         ttk.Label(self, text="").pack()
94
95
96         # load the file containing fixed keywords
97         j=[]
98         f = open(r'keywords.txt')
99         for line in f:
100             j.append(line.strip())
101         f.close()
102         d = dict.fromkeys(j, 0)
103
```

```
104 # code to scan the website given in textbox
105 def scan():
106     count=0
107     url = text.get()
108     text.delete(0, "end")
109     result = requests.get(url.strip())
110     soup = BeautifulSoup(result.content, 'lxml')
111     for i in soup.get_text().split():
112         if i.lower() in j:
113             count+=1
114             if i.lower() in d:
115                 d[i.lower()] +=1
116                 l3.config(state=tk.NORMAL)
117                 l3.delete('1.0', "end")
118                 d1 = dict(sorted(d.items(), reverse=True, key=lambda item: item[1]))
119                 lis = [(k,v) for k,v in d1.items() if v >= 1]
120                 l3.insert(tk.END, url.strip()+" "+str(count)+"\n\nKeywords matched: \n"+json.dumps(lis))
121                 l3.config(state=tk.DISABLED)
122
123     b2=ttk.Button(self, text="Scan", command= scan)
124     b2.pack()
125
126     ttk.Label(self, text="").pack()
127
128 # code to open and scan the list of websites given in a text file
129 def open_n_scan():
```

```
1 import json
2 import tkinter as tk
3 import tkinter.scrolledtext as scrolledtext
4 import tkinter.ttk as ttk
5 from tkinter.filedialog import askopenfile
6
7 import requests
8 from bs4 import BeautifulSoup
9 from PIL import Image, ImageTk
10 from ttkthemes import ThemedStyle
11
12 # class to get all frames together
13 class MyApp(tk.Tk):
14
15     def __init__(self, *args, **kwargs, ):
16         tk.Tk.__init__(self, *args, **kwargs)
17         container = tk.Frame(self)
18         container.pack(side="top", fill="both", expand=True)
19
20
21         container.grid_rowconfigure(0, weight=1)
22         container.grid_columnconfigure(0, weight=1)
23
24         self.frames = {}
25
26         menu = tk.Menu(container)
```

```
155 button1 = ttk.Button(self, text="Back to Home",
156                      command=lambda: controller.show_frame(Startpage))
157 button1.pack()
158
159 ttk.Label(self, text="").pack()
160
161 button2 = ttk.Button(self, text="About",
162                      command=lambda: controller.show_frame(PageTwo))
163 button2.pack()
164
165 # About page
166 class PageTwo(ttk.Frame):
167
168     def __init__(self, parent, controller):
169         ttk.Frame.__init__(self, parent)
170         label = ttk.Label(self, text="About", font=("Simplifica", 22))
171         label.pack(pady=5, padx=5)
172
173         ttk.Label(self, text="").pack()
174
175         button1 = ttk.Button(self, text="Back to Home",
176                             command=lambda: controller.show_frame(Startpage))
177         button1.pack()
178
179         ttk.Label(self, text="").pack()
```



```
button2 = ttk.Button(self, text="Detect",
                      command=Lambda: controller.show_frame(PageOne))
button2.pack()

ttk.Label(self, text="").pack()

tk.Message(self, relief="sunken", bd=4, font=(20), width=1100, text="Suicide is a serious public health problem th
# Info on suicide

ttk.Label(self, text="").pack()

tk.Message(self, relief="sunken", bd=4, font=(20), width=1100, text=" Contact the National Suicide Prevention Life
# Info on suicide content

ttk.Label(self, text="").pack()

tk.Message(self, relief="sunken", bd=4, font=(20), width=1100, text=" Coping and problem-solving skills, Cultural
# About

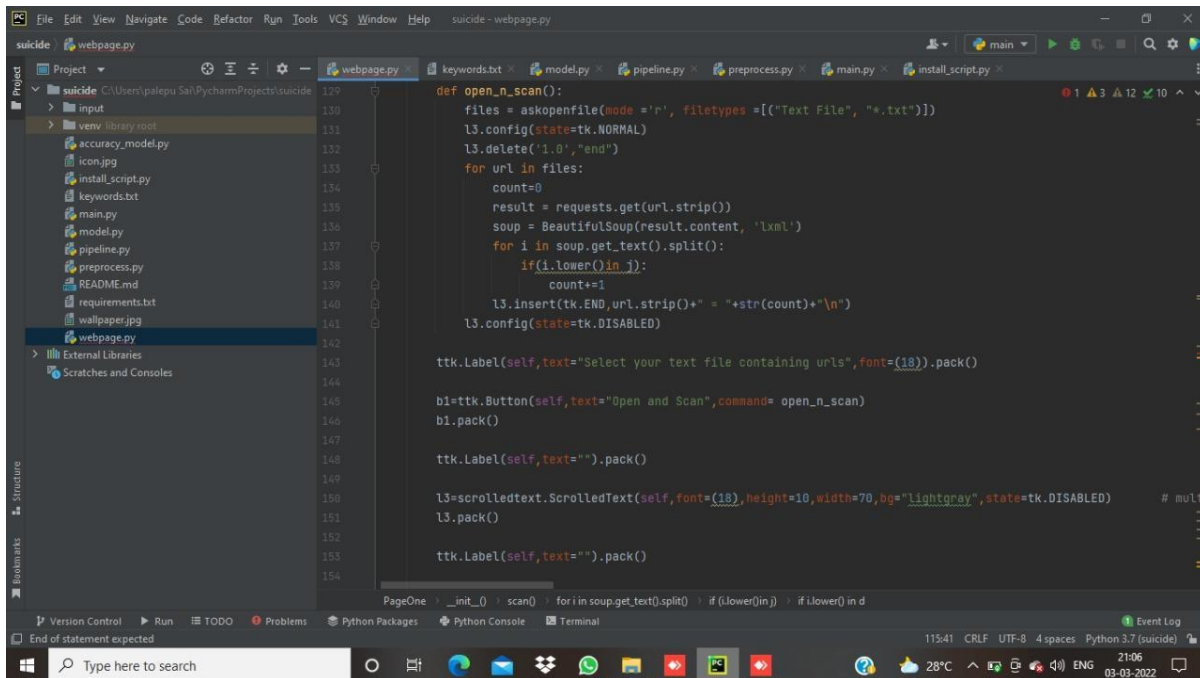
ttk.Label(self, text="").pack()

ttk.Label(self, text="", font=(20)).pack() # copyright

app = MyApp()

# set default app theme
style = ThemedStyle(app)
style.set_theme("plastik")

PageOne > __init__() > scan() > for i in soup.get_text().split() > if (i.lower() in j) > if (i.lower() in d
```



```
def open_n_scan():
    files = askopenfile(mode='r', filetypes=[("Text File", "*.txt")])
    l3.config(state=tk.NORMAL)
    l3.delete('1.0', "end")
    for url in files:
        count=0
        result = requests.get(url.strip())
        soup = BeautifulSoup(result.content, 'lxml')
        for i in soup.get_text().split():
            if (i.lower() in j):
                count+=1
        l3.insert(tk.END, url.strip()+" "+str(count)+"\n")
    l3.config(state=tk.DISABLED)

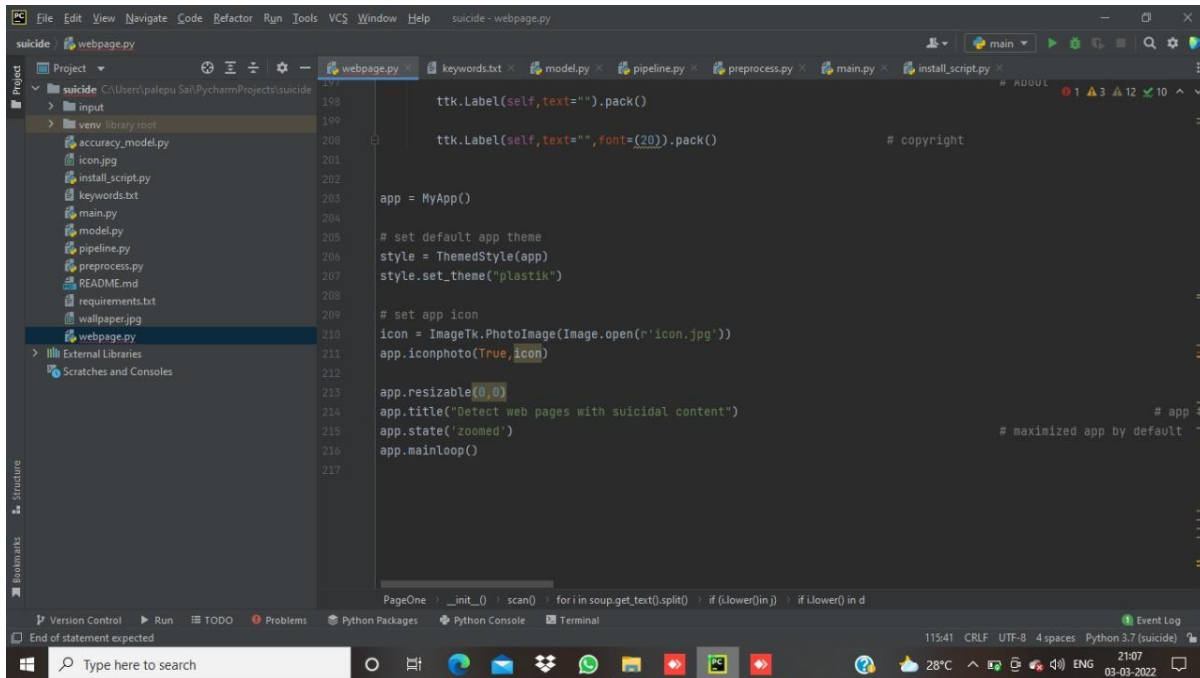
ttk.Label(self, text="Select your text file containing urls", font=(18)).pack()

b1=ttk.Button(self, text="Open and Scan", command= open_n_scan)
b1.pack()

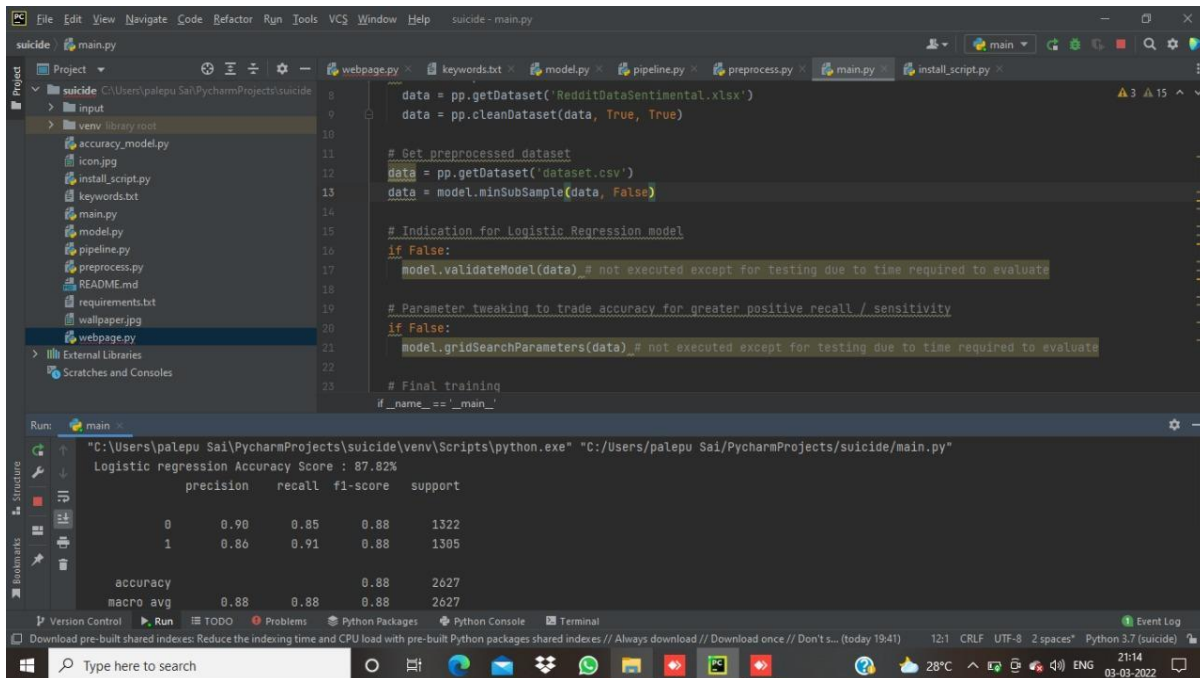
ttk.Label(self, text="").pack()

l3=scrolledtext.ScrolledText(self, font=(18), height=10, width=70, bg="lightgray", state=tk.DISABLED) # multi
l3.pack()

ttk.Label(self, text="").pack()
```



```
197 ttk.Label(self, text="").pack()
198
199
200 ttk.Label(self, text="", font=(20)).pack()
201
202 # copyright
203
204 app = MyApp()
205
206 # set default app theme
207 style = ThemedStyle(app)
208 style.set_theme("plastik")
209
210 # set app icon
211 icon = ImageTk.PhotoImage(Image.open(r'icon.jpg'))
212 app.iconphoto(True, icon)
213
214 app.resizable(0, 0)
215 app.title("Detect web pages with suicidal content")
216 app.state('zoomed')
217 # maximized app by default
218 app.mainloop()
```



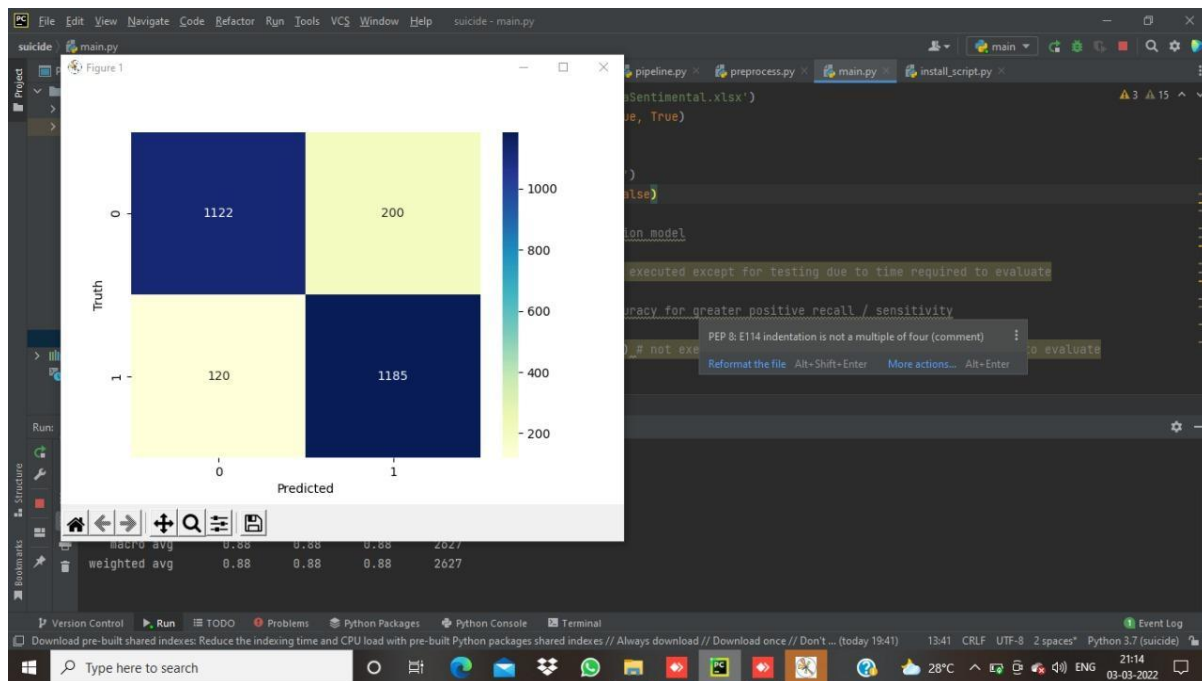
```
8 data = pp.getDataset('RedditDataSentimental.xlsx')
9 data = pp.cleanDataset(data, True, True)
10
11 # Set preprocessed dataset
12 data = pp.getDataset('dataset.csv')
13 data = model.minSubSample(data, False)
14
15 # Indication for Logistic Regression model
16 if False:
17     model.validateModel(data) # not executed except for testing due to time required to evaluate
18
19 # Parameter tweaking to trade accuracy for greater positive recall / sensitivity
20 if False:
21     model.gridSearchParameters(data) # not executed except for testing due to time required to evaluate
22
23 # Final training
24 if __name__ == '__main__':
```

Run: main

"C:\Users\palepu Sai\PycharmProjects\suicide\venv\Scripts\python.exe" "C:\Users\palepu Sai\PycharmProjects\suicide\main.py"

Logistic regression Accuracy Score : 87.82%

	precision	recall	f1-score	support
0	0.90	0.85	0.88	1322
1	0.86	0.91	0.88	1305
accuracy			0.88	2627
macro avg	0.88	0.88	0.88	2627



```
data = pp.getDataset('RedditDataSentimental.xlsx')
data = pp.cleanDataset(data, True, True)

# Get preprocessed dataset
data = pp.getDataset('dataset.csv')
data = model.minSubSample(data, False)

# Indication for Logistic Regression model
if False:
    model.validateModel(data) # not executed except for testing due to time required to evaluate

# Parameter tweaking to trade accuracy for greater positive recall / sensitivity
if False:
    model.gridSearchParameters(data) # not executed except for testing due to time required to evaluate

# Final training
if __name__ == '__main__':
```

Run: main

"C:\Users\palepu Sai\PycharmProjects\suicide\venv\Scripts\python.exe" "C:\Users\palepu Sai\PycharmProjects\suicide\main.py"

Logistic regression Accuracy Score : 87.82%

	precision	recall	f1-score	support
0	0.90	0.85	0.88	1322
1	0.86	0.91	0.88	1305
accuracy			0.88	2627
macro avg	0.88	0.88	0.88	2627

```
data = pp.getDataset('RedditDataSentimental.xlsx')
data = pp.cleanDataset(data, True, True)

# Get preprocessed dataset
data = pp.getDataset('dataset.csv')
data = model.minSubSample(data, False)

# Indication for Logistic Regression model
if False:
    model.validateModel(data) # not executed except for testing due to time required to evaluate

# Parameter tweaking to trade accuracy for greater positive recall / sensitivity
if False:
    model.gridSearchParameters(data) # not executed except for testing due to time required to evaluate

# Final training
if __name__ == '__main__':
```

Run: main

[[55 23]
[1230 1319]]

Accuracy of Support Vector Machine: 82.76%

Confusion Matrix OF SVM :

[[1088 244]
[209 1086]]

Process finished with exit code 0

