PsychePath

A

MINOR PROJECT REPORT

Submitted by

Lakshay Chawla	Mehul Rekhi	Aditya Mehta
C.S.E Department	C.S.E Department	C.S.E Department
MAIT, Rohini	MAIT, Rohini	MAIT, Rohini
40814802718	35814802718	35114802718
lakshaychawla13@gmail.com	rekhi.mehul2000@gmail.com	delhi.adityamehta@gmail.com

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the Guidance

of

Ms. Akanksha Kochhar

Assistant Professor, CSE

Department of Computer Science and Engineering

Maharaja Agrasen Institute ofTechnology,

PSP area, Sector – 22, Rohini, New Delhi – 110085

(Affiliated to Guru Gobind Singh Indraprastha, New Delhi)

(DEC 2021)

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

CERTIFICATE

This is to Certified that this MINOR project report “PsychePAth ” is submitted by “Lakshay Chawla (40814802718) , Aditya Mehta (35114802718) and Mehul Rekhi (35814802718) who carried out the project work under my supervision.

I approve this MINOR project for submission.

Prof. Namita Gupta	Ms. Akanksha Kochhar
(HoD, CSE)	(Assistant Professor , CSE)
	(Project Guide)

ABSTRACT

As we all know, mental health problems in India are not recognized well.
The question to be answered is – What are mental health problems? How many types are there? And how to deal with them?

Firstly, there are many different mental disorders, with different presentations. They are generally characterized by a combination of abnormal thoughts, perceptions, emotions, behaviour and relationships with others.

Mental disorders include depression, bipolar disorder, schizophrenia and other psychoses, dementia, and developmental disorders including autism.

The problem that arrives in India is that first and foremost many people are unaware of the mental health scenario. They don't even consider mental health to be a thing and mental health problems to be real problems. So, the first step needs to be that people should be made aware about such things, which has been growing in recent years because of pandemic and even more so after the Sushant Singh Rajput's case. People have started acknowledging these problems. Even after this many people still consider mental health problems a taboo and are afraid or inconsiderate to talk about them.

Thinking upon the current scenario, the best solution that we could think of was to make some type of tool that is accessible to masses so that one could easily ponder upon their mental health state and get a reality check without facing the social stigma.

So, we set our aim as to create a platform that will be accessible to all, and one does not have to face any criticism for expressing his/her feelings. They can just go to this platform, fill in a couple of forms and get help from professionals.

ACKNOWLEDGEMENT

It gives me immense pleasure to express my deepest sense of gratitude and sincere thanks to my respected guide Ms. Akanksha Kochhar (Assistant Professor , CSE) MAIT Delhi, for their valuable guidance, encouragement and help for completing this work. Their useful suggestions for this whole work and co-operative behavior are sincerely acknowledged.

I also wish to express my indebtedness to my parents as well as my family member whose blessings and support always helped me to face the challenges ahead.

Place: Delhi

Lakshay Chawla (40814802718)
Aditya Mehta (35114802718)
Mehul Rekhi (35814802718)

Date:

Table of Contents

List of Figures

List of Tables

INTRODUCTION

(fig.1)

Mental health is serious topic, which is often being ignored, especially in Indian societies. This topic is in conversation and gathering steam in the recent times. The December 2019 coronavirus disease outbreak has caused a huge number of people getting quarantined because of getting affected. Quarantine, in general, means that you're not allowed to come near vicinity of anyone if you're affected. Often, quarantine is not associated with a true to life experience for the person undergoing that phase. Getting separated form loved ones, losing freedom, not knowing when the phase and illness will end and moreover boredom creates enormous effects. Number of reported suicides have increased1, anger levels have increased, and lawsuits brought2 following the growth in previous outbreaks.

During such hard times, coronavirus being the way it is, quarantine must be the only option. But it has been suggested in various studies that quarantine brings with it negative connotations and weird psychological effects. It has been observed that although all of this is bad firsthand experience, the worrying part is the extension of these effects even after quarantine sometimes even a year later. – although the studies highlighting such effects are not very much. [4,5] This suggests that special care must be taken with such cases.

Considering the advent of technology, the best way to make a good progress in the situation would be to bring the helpline in the hands of everyone. It's a well-known fact that the barrier to smartphone acquirement is Decreasing. So, enabling an individual to know about his/her mental health would be a great boon for the society. In traumatic situations, people don't want to talk to anyone but resort to being lonely and spending time alone which is mostly spent on mobile phones. So, bringing help directly where it is most effective would be a good option.

LITERATURE SURVEY

(fig.2)

Taking upon the difference between the effect of this grave situation on people who were quarantined and not, some studies were made. [3,4,5,6,7]. The most obvious pick was a hospital. Looking upon the staff who was quarantined for 9 days or more showed symptoms of not very good shape. The were diagnosed with acute stress disorder. Expanding on the research, the same staff reported multiple instances of early exhaustion, not willing to socialize, inability to sleep and what not. It was also noticed that concentration levels were on a decline. In another study7 , post traumatic stress symptoms were found in staff even after 3 years. So, the takeaway here is that quarantine indeed takes upon even on the best of us.

Requirement Analysis

User side – To keep the project accessible to the greatest extent, we decided to make a website, that means, anyone on any device can use this with just a web browser being the requirement. And as we know everyone (almost) has access to atleast one of the web browsers in this era of ever-growing technology.

(fig.3)

For development –

1. HTML/CSS –

   (fig.4)

   These contribute to the frontend of the website, that is the look and feel of the webpages. HTML gives structure and CSS gives styling to the website.

2. REACT –

   (fig.5)

   React provides the smartness to the webpage, from making the form entries compulsory to making sure that the required field gets input what we need (for example , the age must be a number not anything else).

3. DJANGO –

   (fig.6)

Django is the main brains of the project joining all the puzzle pieces together. From getting data filled into our databases for machine learning to redirecting webpages and transporting results from one component to another.

4. PYTHON –

(fig.7)

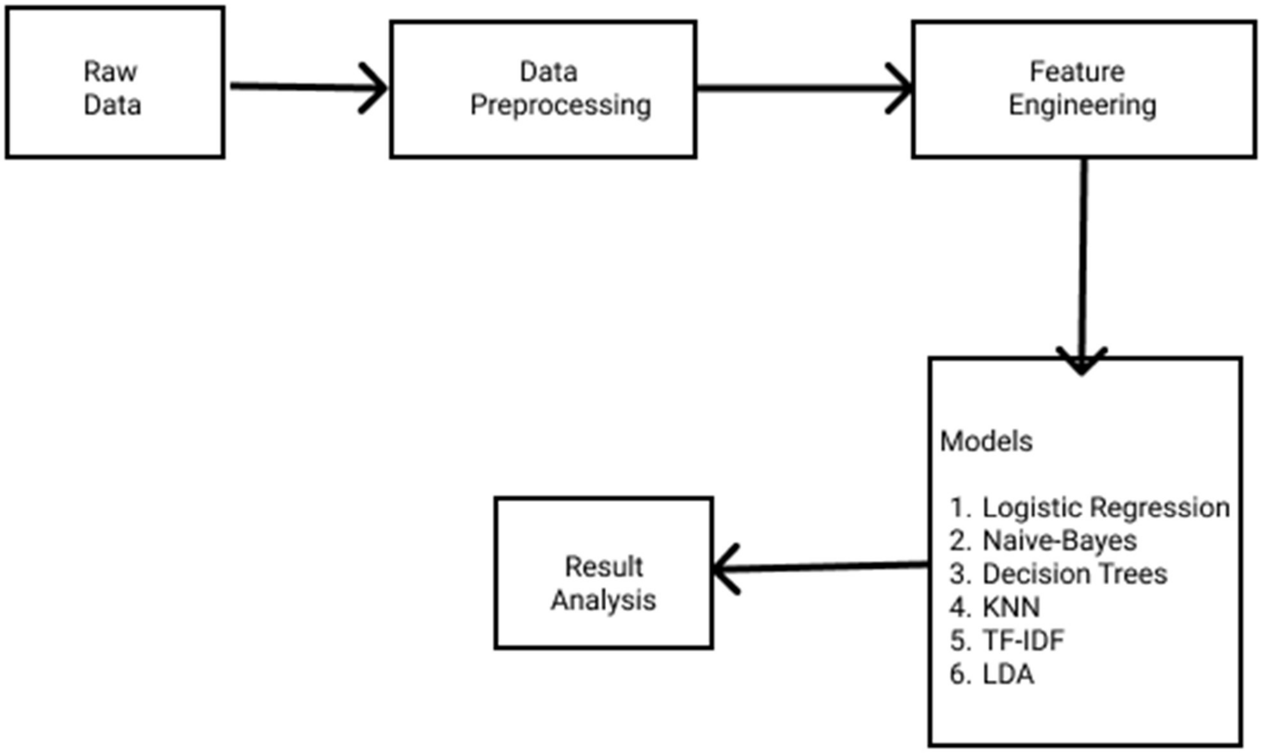
Python is the language used for machine learning and model training. The main functioning of the form is dependent on this part.

5. POSTGRESQL –

(fig.8)

Postgresql is useful to create the database for our project. This is one of the most important aspects of the project, as without data the machine learning model can't be trained and updated.

SYSTEM DESIGN / ARCHITECTURE



(fig.9)

A.Raw data

Considering the topic is really region specific and we can't use existing datasets, we created a survey form with help of psychologists and collected our own data by surveying actual people.

Link to copy of original google form – https://forms.gle/ZpjDJbbn5fPEWtTw6

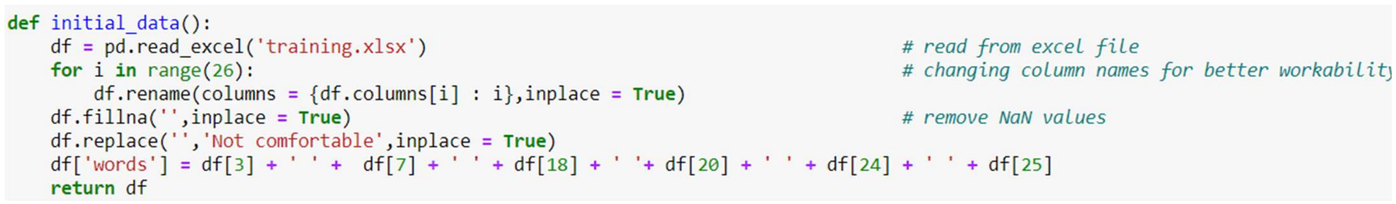
Fields that we made users enter are as following:

i. Name
ii. Age
iii. Gender
iv. pass_time
v. number_of_friends
vi. socialize
vii. personality
viii. hobbies
ix. good_number_of_friends
x. believe_in_humanity
xi. believe_in_god
xii. online_friendships
xiii. no_of_online_friendships
xiv. closest_person
xv. hours_on_phone
xvi. hours_talking_to_people
xvii. productive_time
xviii. covid_effect
xix. good_thing
xx. locality_affected
xxi. bad_thing
xxii. men_life_sphere_affected
xxiii. women_life_sphere_affected
xxiv. sphere_explanation
xxv. affected_explanation

B. Pre-processing

Extracting all the raw data, importing it in the python script, we cleaned the data, removing null values and making it such that we can perform operations on them.

Preprocessing includes treating null values and replacing them with usable values so that the algorithm does not give out unnecessary error codes.



```
def initial_data():
    df = pd.read_excel('training.xlsx')                                            # read from excel file
    for i in range(26):                                                            # changing column names for better workabilit
        df.rename(columns = {df.columns[i] : i},inplace = True)
    df.fillna('',inplace = True)                                                   # remove NaN values
    df.replace('','Not comfortable',inplace = True)
    df['words'] = df[3] + ' ' +  df[7] + ' ' + df[18] + ' '+ df[20] + ' ' + df[24] + ' ' + df[25]
    return df
```

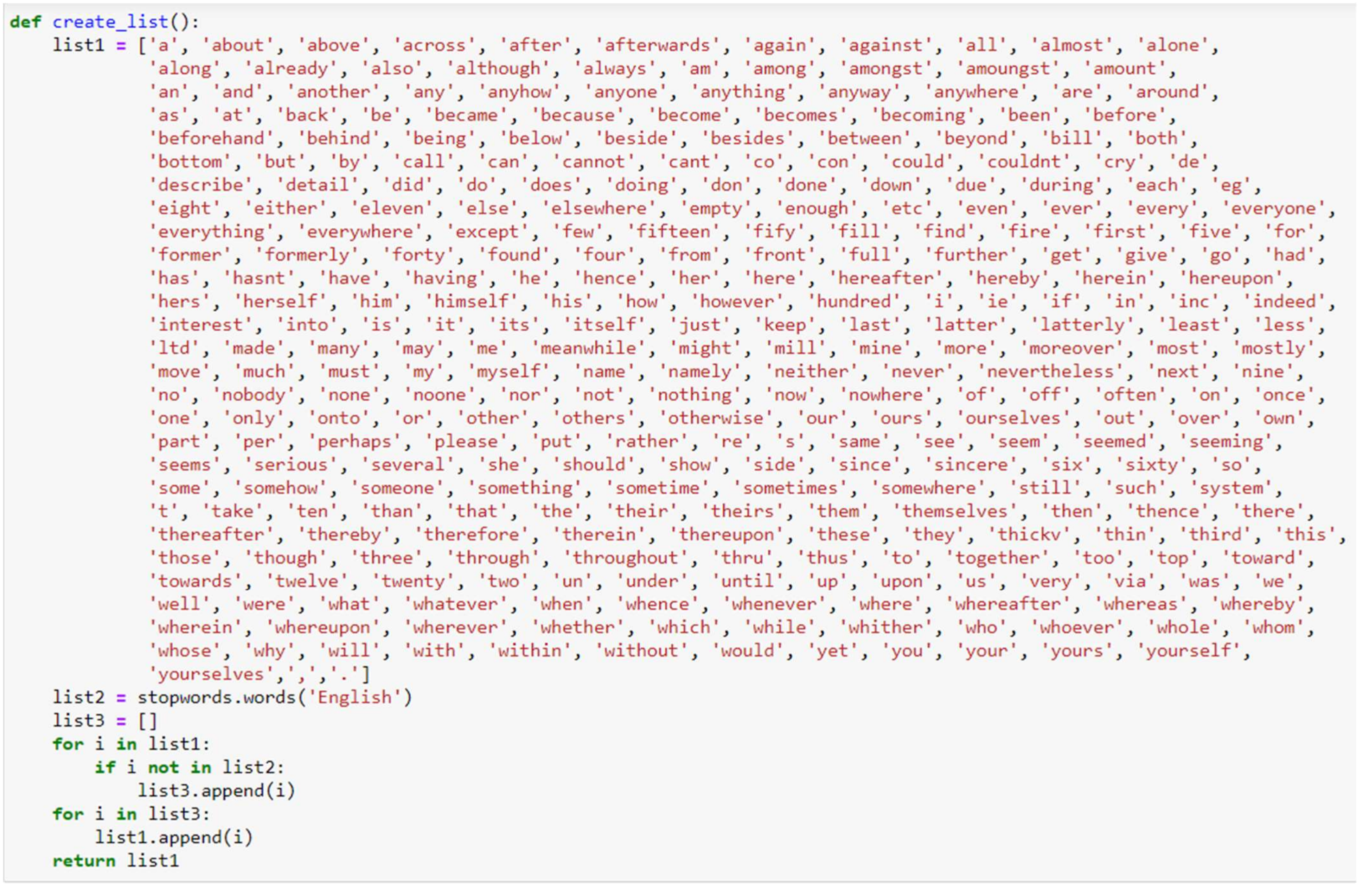
(fig.10)

C. Feature Engineering

Upon analyzing the data, we saw that splitting data in 3 sections would be useful. Applied different operations to make the data useful and easy to use, then went on to implement models.

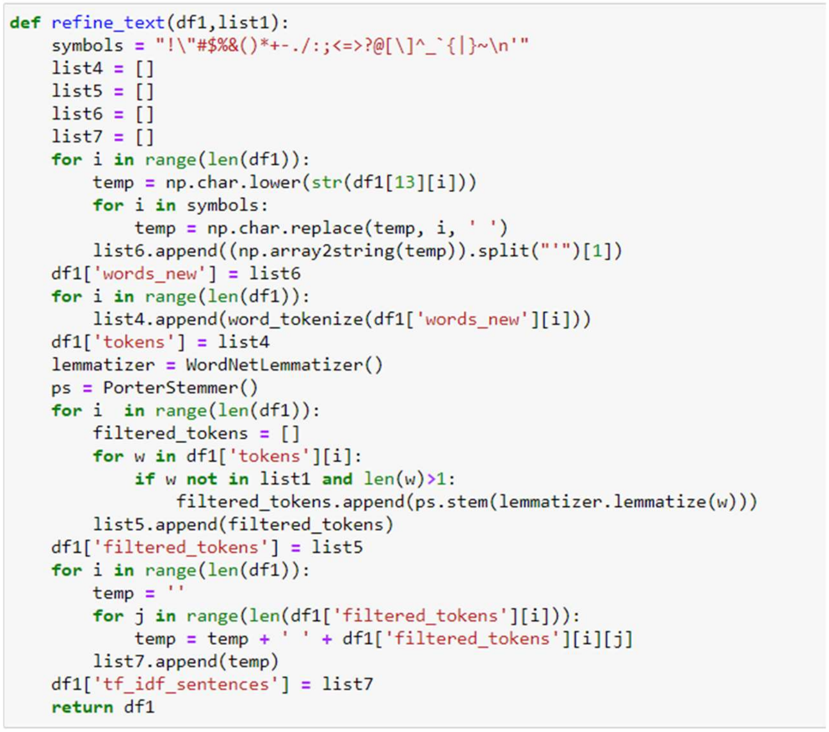
Feature engineering involves various steps. Some of them are

i. Creating a list of stop words.
ii. Lowering the characters of the strings
iii. Removing symbols (to remove punctuation) and apostrophe
iv. Word tokenizing
v. Removing single characters and stop words
vi. Lemmatizing and Stemming
vii. Standardizing the data



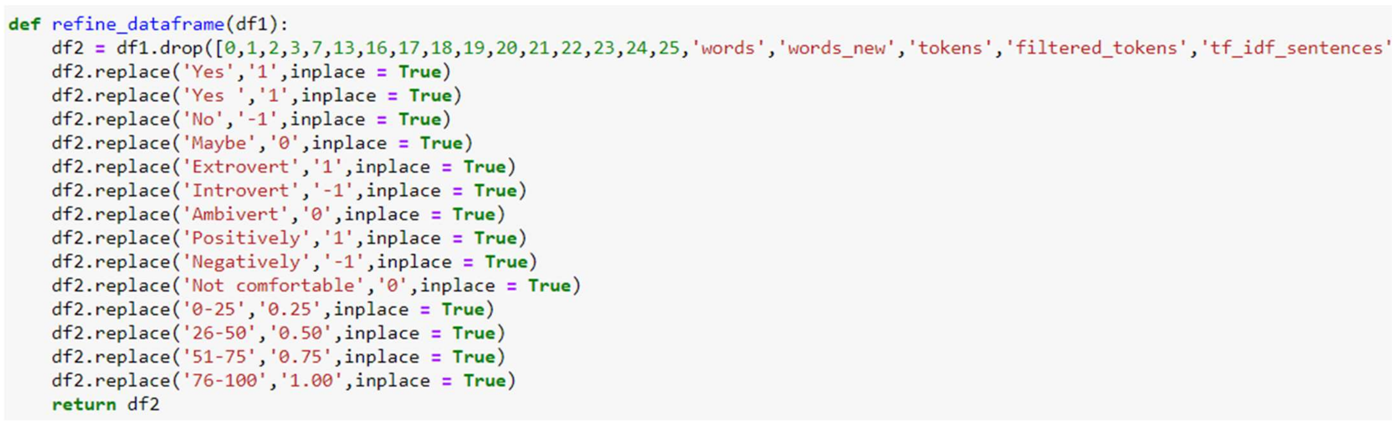
```python
def create_list():
    list1 = ['a', 'about', 'above', 'across', 'after', 'afterwards', 'again', 'against', 'all', 'almost', 'alone',
             'along', 'already', 'also', 'although', 'always', 'am', 'among', 'amongst', 'amoungst', 'amount',
             'an', 'and', 'another', 'any', 'anyhow', 'anyone', 'anything', 'anyway', 'anywhere', 'are', 'around',
             'as', 'at', 'back', 'be', 'became', 'because', 'become', 'becomes', 'becoming', 'been', 'before',
             'beforehand', 'behind', 'being', 'below', 'beside', 'besides', 'between', 'beyond', 'bill', 'both',
             'bottom', 'but', 'by', 'call', 'can', 'cannot', 'cant', 'co', 'con', 'could', 'couldnt', 'cry', 'de',
             'describe', 'detail', 'did', 'do', 'does', 'doing', 'don', 'done', 'down', 'due', 'during', 'each', 'eg',
             'eight', 'either', 'eleven', 'else', 'elsewhere', 'empty', 'enough', 'etc', 'even', 'ever', 'every', 'everyone',
             'everything', 'everywhere', 'except', 'few', 'fifteen', 'fify', 'fill', 'find', 'fire', 'first', 'five', 'for',
             'former', 'formerly', 'forty', 'found', 'four', 'from', 'front', 'full', 'further', 'get', 'give', 'go', 'had',
             'has', 'hasnt', 'have', 'having', 'he', 'hence', 'her', 'here', 'hereafter', 'hereby', 'herein', 'hereupon',
             'hers', 'herself', 'him', 'himself', 'his', 'how', 'however', 'hundred', 'i', 'ie', 'if', 'in', 'inc', 'indeed',
             'interest', 'into', 'is', 'it', 'its', 'itself', 'just', 'keep', 'last', 'latter', 'latterly', 'least', 'less',
             'ltd', 'made', 'many', 'may', 'me', 'meanwhile', 'might', 'mill', 'mine', 'more', 'moreover', 'most', 'mostly',
             'move', 'much', 'must', 'my', 'myself', 'name', 'namely', 'neither', 'never', 'nevertheless', 'next', 'nine',
             'no', 'nobody', 'none', 'noone', 'nor', 'not', 'nothing', 'now', 'nowhere', 'of', 'off', 'often', 'on', 'once',
             'one', 'only', 'onto', 'or', 'other', 'others', 'otherwise', 'our', 'ours', 'ourselves', 'out', 'over', 'own',
             'part', 'per', 'perhaps', 'please', 'put', 'rather', 're', 's', 'same', 'see', 'seem', 'seemed', 'seeming',
             'seems', 'serious', 'several', 'she', 'should', 'show', 'side', 'since', 'sincere', 'six', 'sixty', 'so',
             'some', 'somehow', 'someone', 'something', 'sometime', 'sometimes', 'somewhere', 'still', 'such', 'system',
             't', 'take', 'ten', 'than', 'that', 'the', 'their', 'theirs', 'them', 'themselves', 'then', 'thence', 'there',
             'thereafter', 'thereby', 'therefore', 'therein', 'thereupon', 'these', 'they', 'thickv', 'thin', 'third', 'this',
             'those', 'though', 'three', 'through', 'throughout', 'thru', 'thus', 'to', 'together', 'too', 'top', 'toward',
             'towards', 'twelve', 'twenty', 'two', 'un', 'under', 'until', 'up', 'upon', 'us', 'very', 'via', 'was', 'we',
             'well', 'were', 'what', 'whatever', 'when', 'whence', 'whenever', 'where', 'whereafter', 'whereas', 'whereby',
             'wherein', 'whereupon', 'wherever', 'whether', 'which', 'while', 'whither', 'who', 'whoever', 'whole', 'whom',
             'whose', 'why', 'will', 'with', 'within', 'without', 'would', 'yet', 'you', 'your', 'yours', 'yourself',
             'yourselves',',','.']
    list2 = stopwords.words('English')
    list3 = []
    for i in list1:
        if i not in list2:
            list3.append(i)
    for i in list3:
        list1.append(i)
    return list1
```

(fig.11)



```python
def refine_text(df1,list1):
    symbols = "!\"#$%&()*+-./:;<=>?@[\]^_`{|}~\n'"
    list4 = []
    list5 = []
    list6 = []
    list7 = []
    for i in range(len(df1)):
        temp = np.char.lower(str(df1[13][i]))
        for i in symbols:
            temp = np.char.replace(temp, i, ' ')
        list6.append((np.array2string(temp)).split("'")[1])
    df1['words_new'] = list6
    for i in range(len(df1)):
        list4.append(word_tokenize(df1['words_new'][i]))
    df1['tokens'] = list4
    lemmatizer = WordNetLemmatizer()
    ps = PorterStemmer()
    for i  in range(len(df1)):
        filtered_tokens = []
        for w in df1['tokens'][i]:
            if w not in list1 and len(w)>1:
                filtered_tokens.append(ps.stem(lemmatizer.lemmatize(w)))
        list5.append(filtered_tokens)
    df1['filtered_tokens'] = list5
    for i in range(len(df1)):
        temp = ''
        for j in range(len(df1['filtered_tokens'][i])):
            temp = temp + ' ' + df1['filtered_tokens'][i][j]
        list7.append(temp)
    df1['tf_idf_sentences'] = list7
    return df1
```

(fig.12)



```python
def refine_dataframe(df1):
    df2 = df1.drop([0,1,2,3,7,13,16,17,18,19,20,21,22,23,24,25,'words','words_new','tokens','filtered_tokens','tf_idf_sentences'
    df2.replace('Yes','1',inplace = True)
    df2.replace('Yes ','1',inplace = True)
    df2.replace('No','-1',inplace = True)
    df2.replace('Maybe','0',inplace = True)
    df2.replace('Extrovert','1',inplace = True)
    df2.replace('Introvert','-1',inplace = True)
    df2.replace('Ambivert','0',inplace = True)
    df2.replace('Positively','1',inplace = True)
    df2.replace('Negatively','-1',inplace = True)
    df2.replace('Not comfortable','0',inplace = True)
    df2.replace('0-25','0.25',inplace = True)
    df2.replace('26-50','0.50',inplace = True)
    df2.replace('51-75','0.75',inplace = True)
    df2.replace('76-100','1.00',inplace = True)
    return df2
```

(fig.13)

D.Machine Learning algorithms (Models)

Trying to learn new machine learning models, we stumbled upon many of them, at the end deciding to give a chance to these models, as they seemed apt for our data.

i. Logistic Regression
ii. Naïve Bayes
iii. KNN
iv. Decision Trees
v. Tf – Idf
vi. LDA

Logistic Regression

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Types of Logistic Regression

Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on those number of categories, Logistic regression can be divided into following types –

Binary or Binomial

In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.

Multinomial

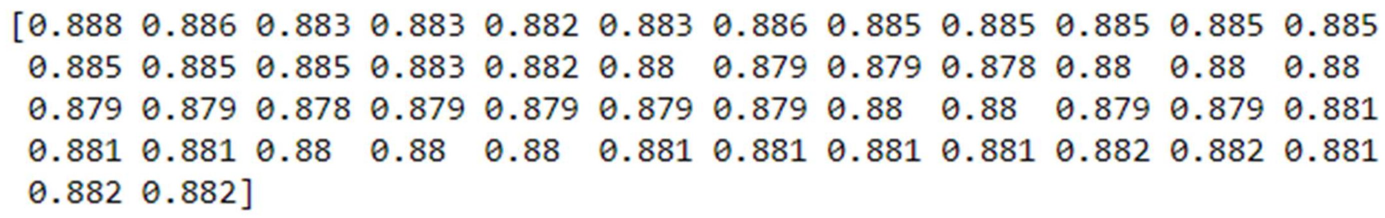
In such a kind of classification, dependent variable can have 3 or more possible ***unordered*** types or the types having no quantitative significance. For example, these variables may represent "Type A" or "Type B" or "Type C".

Ordinal

In such a kind of classification, dependent variable can have 3 or more possible ***ordered*** types or the types having a quantitative significance. For example, these variables may represent "poor" or "good", "very good", "Excellent" and each category can have the scores like 0,1,2,3.

In our algorithm , we used binary logistic regression as the part one of our custom algorithm. On testing with the initial data , we got to know that C = 1.0 gives the best results in out case, so we stuck with C = 1.0.

Following image shows the results for C = 1.0 to C = 50.0



```
[0.888 0.886 0.883 0.883 0.882 0.883 0.886 0.885 0.885 0.885 0.885 0.885
 0.885 0.885 0.885 0.883 0.882 0.88  0.879 0.879 0.878 0.88  0.88  0.88
 0.879 0.879 0.878 0.879 0.879 0.879 0.879 0.88  0.88  0.879 0.879 0.881
 0.881 0.881 0.88  0.88  0.88  0.881 0.881 0.881 0.881 0.882 0.882 0.881
 0.882 0.882]
```

(fig.14)

Naïve Bayes

Naïve Bayes algorithms is a classification technique based on applying Bayes' theorem with a strong assumption that all the predictors are independent to each other. In simple words, the assumption is that the presence of a feature in a class is independent to the presence of any other feature in the same class. For example, a phone may be considered as smart if it is having touch screen, internet facility, good camera etc. Though all these features are dependent on each other, they contribute independently to the probability of that the phone is a smart phone.

In Bayesian classification, the main interest is to find the posterior probabilities i.e. the probability of a label given some observed features, $P(L \mid features)$. With the help of Bayes theorem, we can express this in quantitative form as follows –

P(L|features)=P(L) * P(features|L) / P(features)

Here, $P(L \mid features)$ is the posterior probability of class.

$P(L)$ is the prior probability of class.

$P(features \mid L)$ is the likelihood which is the probability of predictor given class.

$P(features)$ is the prior probability of predictor.

Python library, Scikit learn is the most useful library that helps us to build a Naïve Bayes model in Python. We have the following three types of Naïve Bayes model under Scikit learn Python library –

Gaussian Naïve Bayes

It is the simplest Naïve Bayes classifier having the assumption that the data from each label is drawn from a simple Gaussian distribution.

Multinomial Naïve Bayes

Another useful Naïve Bayes classifier is Multinomial Naïve Bayes in which the features are assumed to be drawn from a simple Multinomial distribution. Such kind of Naïve Bayes are most appropriate for the features that represents discrete counts.

Bernoulli Naïve Bayes

Another important model is Bernoulli Naïve Bayes in which features are assumed to be binary (0s and 1s). Text classification with 'bag of words' model can be an application of Bernoulli Naïve Bayes.

In our testing we used Multinomial Naïve Bayes algorithm, but the results were not upto our satisfaction. We got only 25.3% accuracy during our testing, so we decided not to use this algorithm in the final website deployment.

KNN

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well –

- **Lazy learning algorithm** – KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
- **Non-parametric learning algorithm** – KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps –

- **Step 1** – For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.
- **Step 2** – Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.
- **Step 3** – For each point in the test data do the following –

3.1 – Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.

3.2 – Now, based on the distance value, sort them in ascending order.

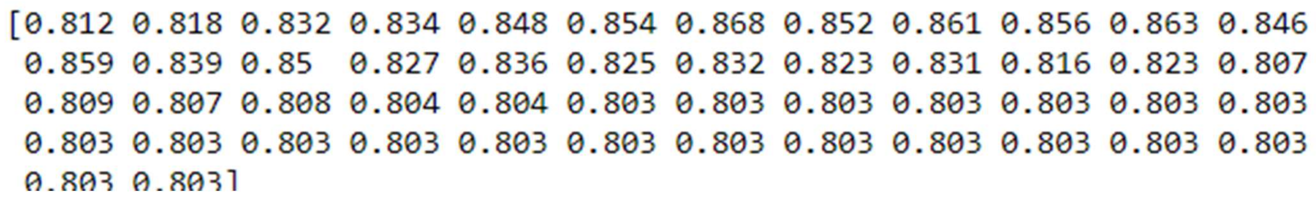
3.3 – Next, it will choose the top K rows from the sorted array.

3.4 – Now, it will assign a class to the test point based on most frequent class of these rows.

- **Step 4** – End

Again, ransom repeat the same process of training another model and testing, this time on KNN , that is K- Nearest Neighbors.

The following image shows the results for K = 1 to K = 50.



```
[0.812 0.818 0.832 0.834 0.848 0.854 0.868 0.852 0.861 0.856 0.863 0.846
 0.859 0.839 0.85  0.827 0.836 0.825 0.832 0.823 0.831 0.816 0.823 0.807
 0.809 0.807 0.808 0.804 0.804 0.803 0.803 0.803 0.803 0.803 0.803 0.803
 0.803 0.803 0.803 0.803 0.803 0.803 0.803 0.803 0.803 0.803 0.803 0.803
 0.803 0.803]
```

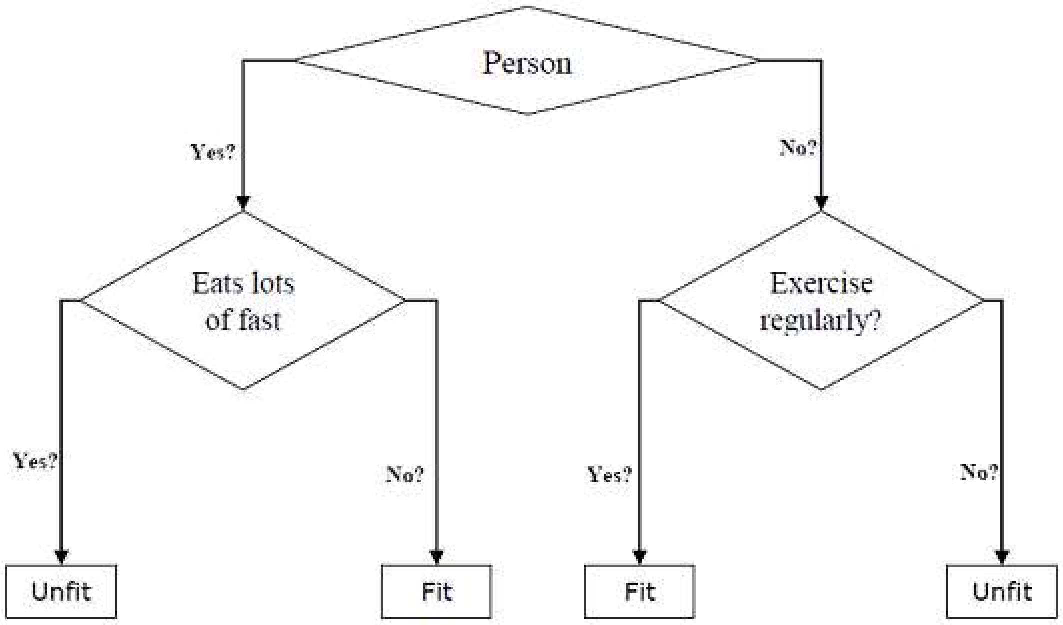
(fig.15)

As we can see that the best result that we're getting is from K = 7 , that is 86.8%. The results were good , but they were still less than logistic regression that agve accuracy of 88.8%. So, we decided to stick with logistic regression for now.

Decision Trees

In general, Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions tress are the most powerful algorithms that falls under the category of supervised algorithms.

They can be used for both classification and regression tasks. The two main entities of a tree are decision nodes, where the data is split and leaves, where we got outcome. The example of a binary tree for predicting whether a person is fit or unfit providing various information like age, eating habits and exercise habits, is given below –



(fig.16)

Implementing Decision Tree Algorithm

Gini Index

It is the name of the cost function that is used to evaluate the binary splits in the dataset and works with the categorial target variable "Success" or "Failure".

Higher the value of Gini index, higher the homogeneity. A perfect Gini index value is 0 and worst is 0.5 (for 2 class problem). Gini index for a split can be calculated with the help of following steps –

- First, calculate Gini index for sub-nodes by using the formula p^2+q^2 , which is the sum of the square of probability for success and failure.
- Next, calculate Gini index for split using weighted Gini score of each node of that split.

Classification and Regression Tree (CART) algorithm uses Gini method to generate binary splits.

Split Creation

A split is basically including an attribute in the dataset and a value. We can create a split in dataset with the help of following three parts –

- **Part1: Calculating Gini Score** – We have just discussed this part in the previous section.
- **Part2: Splitting a dataset** – It may be defined as separating a dataset into two lists of rows having index of an attribute and a split value of that attribute. After getting the two groups – right and left, from the dataset, we can calculate the value of split by using Gini score calculated in first part. Split value will decide in which group the attribute will reside.
- **Part3: Evaluating all splits** – Next part after finding Gini score and splitting dataset is the evaluation of all splits. For this purpose, first, we must check every value associated with each attribute as a candidate split. Then we need to find the best possible split by evaluating the cost of the split. The best split will be used as a node in the decision tree.

Building a Tree

As we know that a tree has root node and terminal nodes. After creating the root node, we can build the tree by following two parts –

Part1: Terminal node creation

While creating terminal nodes of decision tree, one important point is to decide when to stop growing tree or creating further terminal nodes. It can be done by using two criteria namely maximum tree depth and minimum node records as follows –

- **Maximum Tree Depth** – As name suggests, this is the maximum number of the nodes in a tree after root node. We must stop adding terminal nodes once a tree reached at maximum depth i.e. once a tree got maximum number of terminal nodes.
- **Minimum Node Records** – It may be defined as the minimum number of training patterns that a given node is responsible for. We must stop adding terminal nodes once tree reached at these minimum node records or below this minimum.

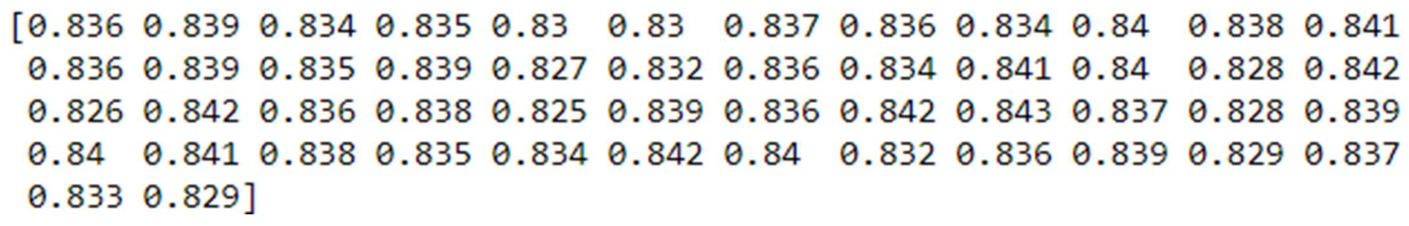
Terminal node is used to make a final prediction.

Part2: Recursive Splitting

As we understood about when to create terminal nodes, now we can start building our tree. Recursive splitting is a method to build the tree. In this method, once a node is created, we can create the child nodes (nodes added to an existing node) recursively on each group of data, generated by splitting the dataset, by calling the same function again and again.

We can get varying degrees of success in decision trees by varying the depth of the tree. For our data , we tested on various depths ranging from 1 to 50.

The following image shows all the results.



```
[0.836 0.839 0.834 0.835 0.83  0.83  0.837 0.836 0.834 0.84  0.838 0.841
 0.836 0.839 0.835 0.839 0.827 0.832 0.836 0.834 0.841 0.84  0.828 0.842
 0.826 0.842 0.836 0.838 0.825 0.839 0.836 0.842 0.843 0.837 0.828 0.839
 0.84  0.841 0.838 0.835 0.834 0.842 0.84  0.832 0.836 0.839 0.829 0.837
 0.833 0.829]
```

(fig.17)

As we can see that the best result for our data comes at a tree depth of 33 , with 84.3% accuracy.

Amazing results but again not as good as Logistic regression, so we will be sticking with logistic regression for the classification model.

TF – IDF

TF-IDF stands for **“Term Frequency — Inverse Document Frequency”**. This is a technique to quantify words in a set of documents. We generally compute a score for each word to signify its importance in the document and corpus. This method is a widely used technique in Information Retrieval and Text Mining.

If I give you a sentence for example “This building is so tall”. It’s easy for us to understand the sentence as we know the semantics of the words and the sentence. But how can any program (eg: python) interpret this sentence? It is easier for any programming language to understand textual data in the form of numerical value. So, for this reason, we need to vectorize all of the text so that it is better represented.

By vectorizing the documents we can further perform multiple tasks such as finding the relevant documents, ranking, clustering, etc. This exact technique is used when you perform a google search (now they are updated to newer transformer techniques). The web pages are called documents and the search text with which you search is called a query. The search engine maintains a fixed representation of all the documents. When you search with a query, the search engine will find the relevance of the query with all of the documents, ranks them in the order of relevance and shows you the top k documents. All of this process is done using the vectorized form of query and documents.

Now coming back to our TF-IDF,

TF-IDF = Term Frequency (TF) * Inverse Document Frequency (IDF)

Terminology

- t — term (word)
- d — document (set of words)
- N — count of corpus
- corpus — the total document set

Term Frequency

This measures the frequency of a word in a document. This highly depends on the length of the document and the generality of the word, for example, a very common word such as “was” can appear multiple times in a document. But if we take two documents with 100 words and 10,000 words respectively, there is a high probability that the common word “was” is present more in the 10,000 worded document. But we cannot say that the longer document is more important than the shorter document. For this exact reason, we perform normalization on the frequency value, we divide the frequency with the total number of words in the document.

Recall that we need to finally vectorize the document. When we plan to vectorize documents, we cannot just consider the words that are present in that particular document. If we do that, then the vector length will be different for both the documents, and it will not be feasible to compute the similarity. So, what we do is that we vectorize the documents on the **vocab**. Vocab is the list of all possible worlds in the corpus.

We need the word counts of all the vocab words and the length of the document to compute TF. In case the term doesn’t exist in a particular document, that particular

TF value will be 0 for that particular document. In an extreme case, if all the words in the document are the same, then TF will be 1. The final value of the normalised TF value will be in the range of [0 to 1]. 0, 1 inclusive.

TF is individual to each document and word, hence we can formulate TF as follows:

tf(t,d) = count of t in d / number of words in d

If we already computed the TF value and if this produces a vectorized form of the document, why not use just TF to find the relevance between documents? Why do we need IDF?

Let me explain, words which are most common such as 'is', 'are' will have very high values, giving those words very high importance. But using these words to compute the relevance produces bad results. These kinds of common words are called stop-words. Although we will remove the stop words later in the preprocessing step, finding the presence of the word across the documents and somehow reduce their weightage is more ideal.

Document Frequency

This measures the importance of documents in a whole set of the corpus. This is very similar to TF but the only difference is that TF is the frequency counter for a term t in document d, whereas DF is the count of **occurrences** of term t in the document set N. In other words, DF is the number of documents in which the word is present. We consider one occurrence if the term is present in the document at least once, we do not need to know the number of times the term is present.

Df(t) = occurrence of t in N documents

To keep this also in a range, we normalize by dividing by the total number of documents. Our main goal is to know the informativeness of a term, and DF is the exact inverse of it. That is why we inverse the DF

Inverse Document Frequency

IDF is the inverse of the document frequency which measures the informativeness of term t. When we calculate IDF, it will be very low for the most occurring words such as stop words (because they are present in almost all of the documents, and N/df will give a very low value to that word). This finally gives what we want, a relative weightage.

Idf(t) = N/df

Now there are few other problems with the IDF, when we have a large corpus size say N=10000, the IDF value explodes. So to dampen the effect we take the log of IDF.

At query time, when the word is not present in is not in the vocab, it will simply be ignored. But in few cases, we use a fixed vocab and few words of the vocab might be absent in the document, in such cases, the df will be 0. As we cannot divide by 0, we smoothen the value by adding 1 to the denominator.

Idf(t) = log(N/(df + 1))

Finally, by taking a multiplicative value of TF and IDF, we get the TF-IDF score. There are many different variations of TF-IDF but for now, let us concentrate on this basic version.

Tf-idf(t, d) = tf(t, d) * log(N/(df + 1))

LDA

It is one of the most popular topic modeling methods. Each document is made up of various words, and each topic also has various words belonging to it. The aim of LDA is to find topics a document belongs to, based on the words in it. Confused much?

What we want to figure out are the words in different topics, as shown in the table below. Each row in the table represents a different topic and each column a different word in the corpus. Each cell contains the probability that the word(column) belongs to the topic(row).

Each topic contains a score for all the words in the corpus.

Finding Representative Words for a Topic

- We can **sort the words** with respect to their probability score.
  The top *x* words are chosen from each topic to represent the topic. If *x* = 10, we'll sort all the words in topic1 based on their score and take the top 10 words to represent the topic.
  This step may not always be necessary because if the corpus is small we can store all the words in sorted by their score.

- Alternatively, we can **set a threshold** on the score. All the words in a topic having a score above the threshold can be stored as its representative, in order of their scores.

Let's say we have 2 topics that can be classified as *CAT_related* and *DOG_related*. A topic has probabilities for each word, so words such as *milk*, *meow*, and *kitten*, will have a higher probability in the *CAT_related* topic than in the *DOG_related* one. The *DOG_related* topic, likewise, will have high probabilities for words such as *puppy*, *bark*, and *bone*.

If we have a document containing the following sentences:

"*Dogs* like to *chew* on *bones* and fetch sticks".
"*Puppies* drink *milk*."
"Both like to *bark*."

We can easily say it belongs to topic *DOG_related* because it contains words *such as Dogs*, *bones, puppies*, and *bark*. Even though it contains the word *milk* which belongs to the topic *CAT_related*, the document belongs to *DOG_related* as more words match with it.

Assumptions:

- Each document is just a collection of words or a "bag of words". Thus, the **order of the words** and the **grammatical role** of the words (subject, object, verbs, ...) are **not considered** in the model.

- Words like am/is/are/of/a/the/but/… don’t carry any information about the “topics” and therefore can be eliminated from the documents as a preprocessing step. In fact, **we can eliminate words that occur in at least %80 ~ %90 of the documents,** without losing any information. For example, if our corpus contains only medical documents, words like human, body, health, etc might be present in most of the documents and hence can be removed as they don’t add any specific information which would make the document stand out.
- We **know beforehand how many topics** we want. ‘***k***’ is pre-decided.
- **All topic assignments except for the current word in question are correct**, and then updating the assignment of the current word using our model of how documents are generated

How does LDA work?

There are 2 parts in LDA:

- The ***words that belong to a document***, that we already know.
- The ***words that belong to a topic*** or the probability of words belonging into a topic, that we need to calculate.

The Algorithm to find the latter

- Go through each document and randomly assign each word in the document to one of ***k*** topics (*k* is chosen beforehand).
- For each document ***d***, go through each word ***w*** and compute :

1. **p(topic *t* | document *d*)**: the **proportion of words in document *d* that are assigned to topic *t***. Tries to capture how many words belong to the topic *t* for a given document *d*. Excluding the current word.
   If a lot of words from *d* belongs to *t*, it is more probable that word *w* belongs to *t*.
   (#words in *d* with *t* +*alpha*/ #words in *d* with any topic+ *k***alpha*)

2. **p(word *w*| topic *t*)**: the proportion of assignments to topic *t* over all documents that come from this word *w*. Tries to capture how many documents are in topic *t* because of word *w*.
   LDA represents documents as a mixture of topics. Similarly, a topic is a mixture of words. If a word has high probability of being in a topic, all the documents having *w* will be more strongly associated with *t* as well. Similarly, if *w* is not very probable to be in *t*, the documents which contain the *w* will be having very low probability of being in *t*, because rest of the words in *d* will belong to some other topic and hence *d* will have a higher probability for those topic. So even if *w* gets added to *t*, it won't be bringing many such documents to *t*.

- Update the probability for the word *w* belonging to topic *t*, as

```
p(word w with topic t) = p(topic t | document d) * p(word w |
topic t)
```

A layman's example

Suppose you have various photographs(*documents*) with captions(*words*). You want to display them in a gallery so you decide to categorize the photographs on various themes(*topics*) based on which you will create different sections in your gallery.

You decide to create *k=2* sections in your album — nature and city. Naturally, the classification isn't so clear as some photographs with city have trees and flowers while the nature ones might have some buildings in it. You, as a start, decide to assign the photographs which only have nature or city elements in them into their respective categories while you randomly assigned the rest.

You notice that a lot of photographs in nature have the word *tree* in their captions. So you concluded that the word *tree* and topic *nature* must be closely related.

Next, you pick the word *building* and check how many photographs are in *nature* because they have the word *building* in their caption. You don't find many and now are less sure about *building* belonging to the topic *nature* and associate it more strongly with the topic *city*.

You then pick a photograph which has the caption **"The tree is in front of the building and behind a car"** and see that it is in the category nature currently. You then chose the word ***tree***, and calculate the first probability **p(topic *t* | document *d*):** other words in the caption are *building* and *car*, most photographs having captions with *building* or *car* in it are in *city*, so you get a low probability. Now the second probability **p(word *w*| topic *t*)**: we know that a lot of photographs in nature have the word *trees* in it. So you get a high score here.
You update the probability of *tree* belonging in *nature* by multiplying the two. You

get a lower value than before for *tree* in *topic* nature because now you have seen that *tree* and words such as *building/car* in the same caption, implying that trees can also be found in cities.
For the same reason, when you update the probability for *tree* belonging in topic *city*, you will notice that it will be greater than what it was before.

After multiple iterations over all the photographs and for each topic, you will have accurate scores for each word with respect to each topic. Your guesses will keep getting better and better because you'll conclude from the context that words such as buildings, sidewalk, subway appear together and hence must belong to the same topic, which we can easily guess is *city*.
Words such as mountains, fields, beach which might not appear together in a lot of captions but they do appear often without city words and hence will have higher scores for nature.
While words such as trees, flowers, dogs, sky will have almost the same probability of being in either as they occur in both topics.

As for the photograph, you see that it has 1 word (with average probability) from category nature and 2 words (with high probability) from city, you conclude, it belongs to city more strongly than it does to nature and hence you decide to add it in city.

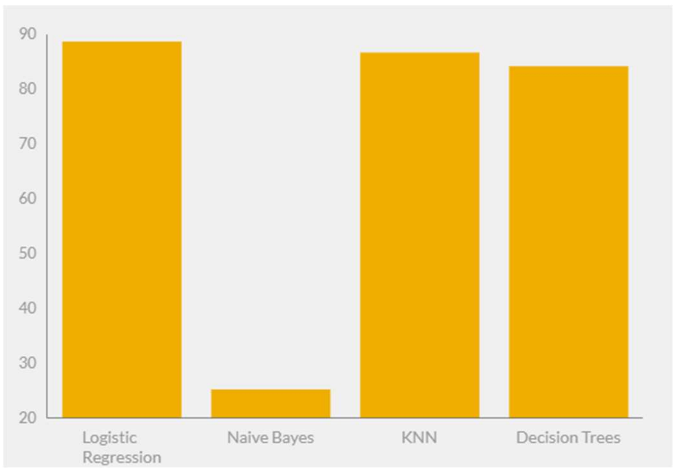
For our model, we had to distribute the word set in two parts, one that are positive sounding and one that are negative sounding, so we set the value of N-components for our model to be equal to 2 and then trained it on our data.

E. **Result Analysis**

Now, running our data through all the algorithms and checking the results various times, that is changing training and testing data multiple times, we got the following results.

Model Name	Accuracy
Logistic Regression	88.8%
Naïve Bayes	25.3%
KNN	86.8%
Decision Trees	84.3%

(Table 1)

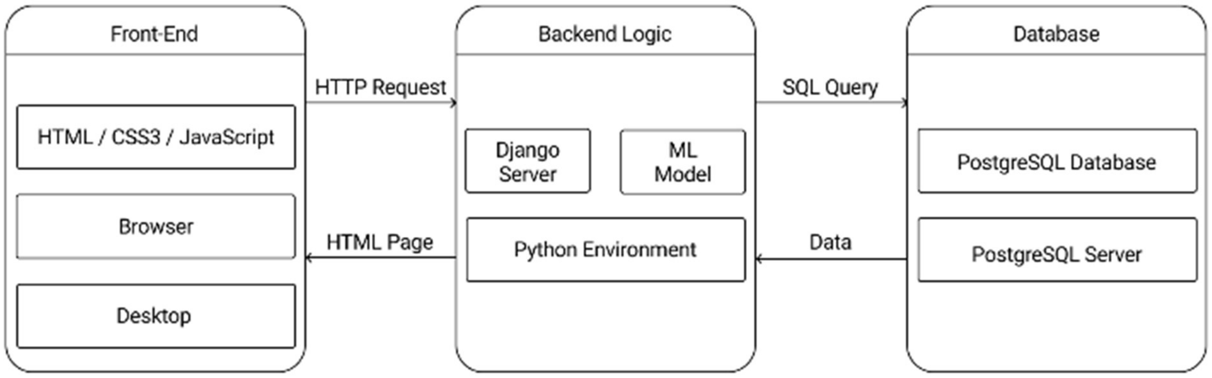


(fig.18)

F. Final Deployment

Of all the algorithms, Logistic regression came out on top with a blistering 88.8% accuracy. But this also has accuracy not satisfactory to us, to improve upon that we decided to make our custom model that was a combination of a custom function, logistic regression model and Tf-Idf /LDA model.

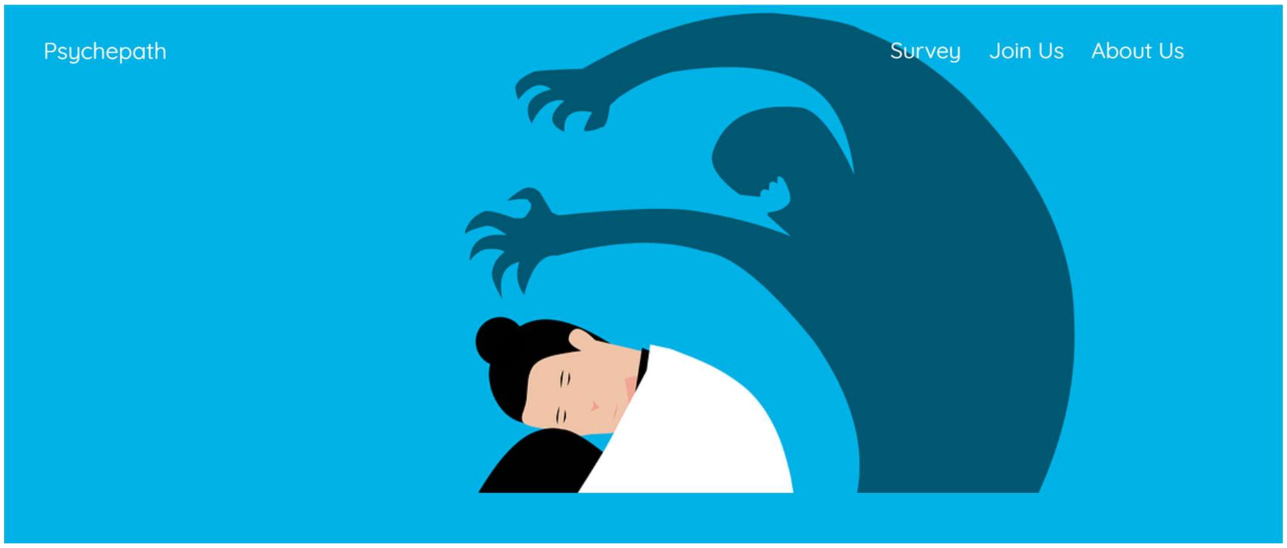
Finally, after we were satisfied with the model, we deployed all our work on a website – www.psychepath.xyz



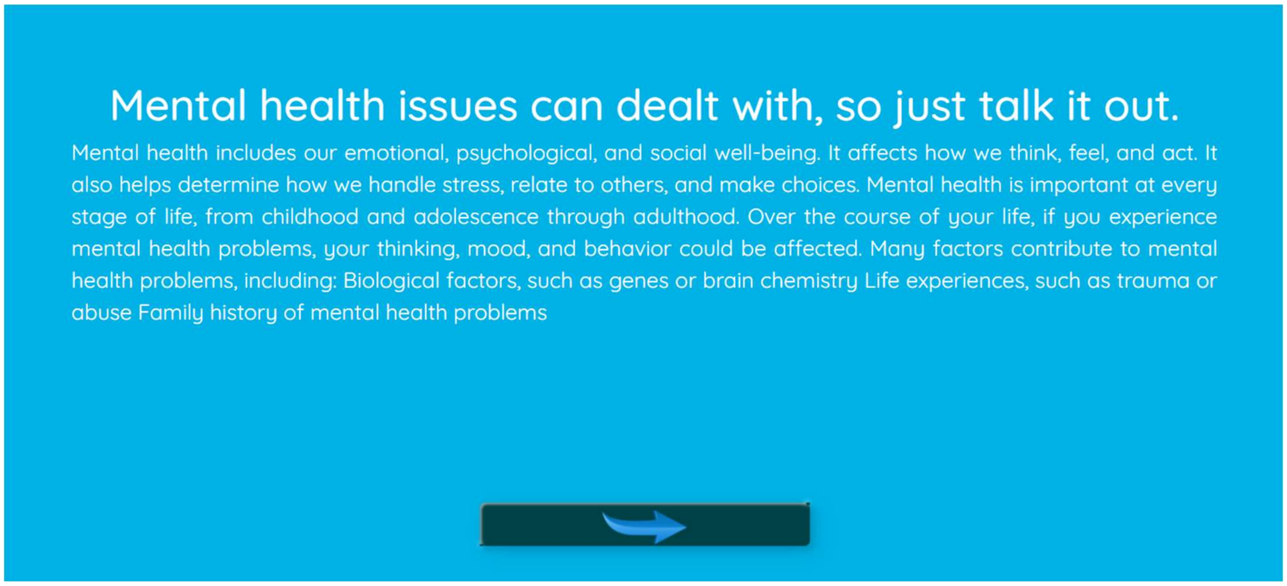
(fig.19)

In the website as you can see in the architecture diagram, the frontend webpage takes the required inputs from the user, transfers it to the backend Django server using react JS which in turn calls the custom machine learning model that we built and gives the output. If the user is tagged for potential mental risk, he/she is given a certified medical test PHQ-9 to confirm the mental state. On the result, the user is directed to a professional practitioner if required.

Here are some screenshots from our website –



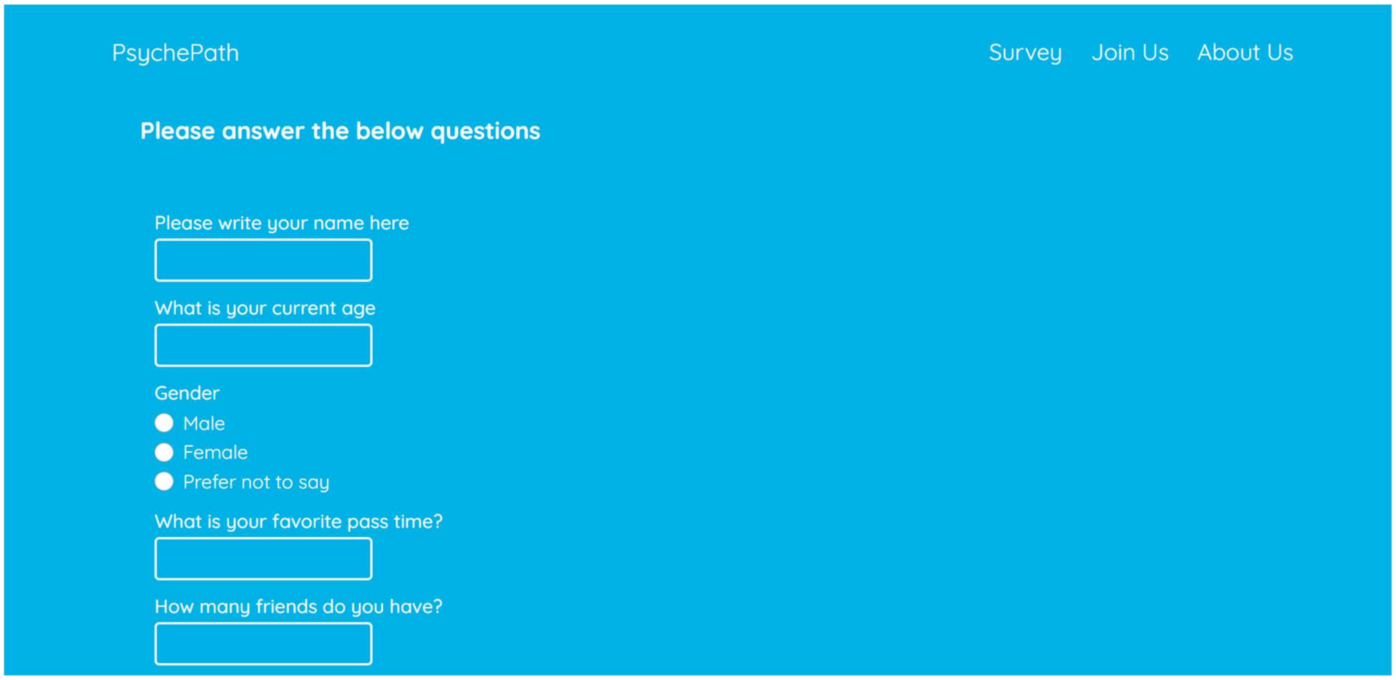
(fig.20)



Mental health issues can dealt with, so just talk it out.

Mental health includes our emotional, psychological, and social well-being. It affects how we think, feel, and act. It also helps determine how we handle stress, relate to others, and make choices. Mental health is important at every stage of life, from childhood and adolescence through adulthood. Over the course of your life, if you experience mental health problems, your thinking, mood, and behavior could be affected. Many factors contribute to mental health problems, including: Biological factors, such as genes or brain chemistry Life experiences, such as trauma or abuse Family history of mental health problems

(fig.21)



Please answer the below questions

Please write your name here

What is your current age

Gender

○ Male

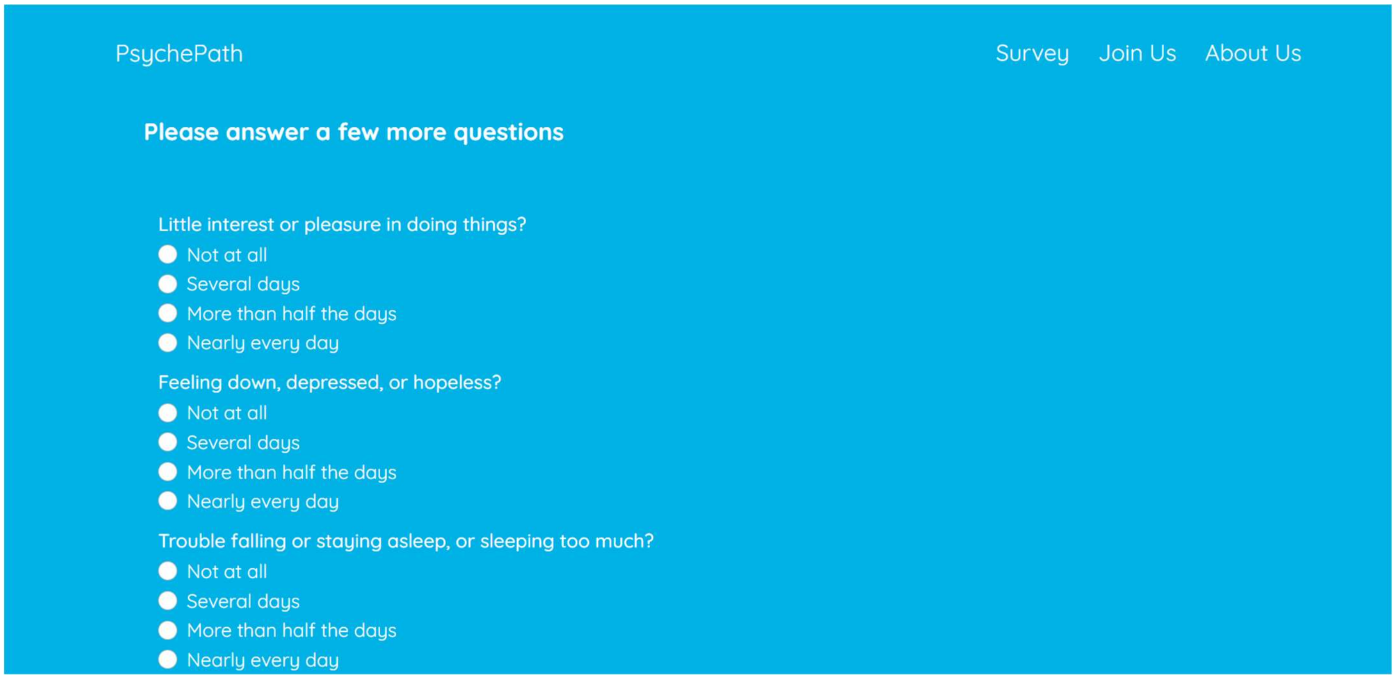
○ Female

○ Prefer not to say

What is your favorite pass time?

How many friends do you have?

(fg.22)



Please answer a few more questions

Little interest or pleasure in doing things?

○ Not at all

○ Several days

○ More than half the days

○ Nearly every day

Feeling down, depressed, or hopeless?

○ Not at all

○ Several days

○ More than half the days

○ Nearly every day

Trouble falling or staying asleep, or sleeping too much?

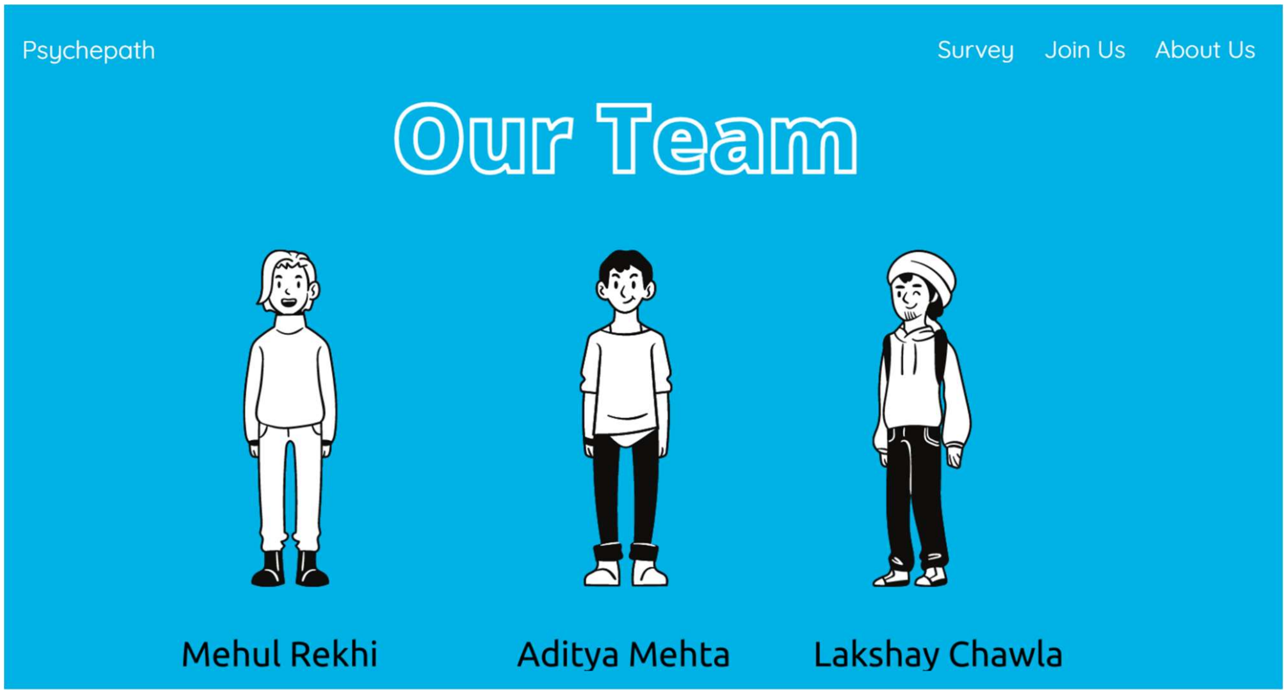
○ Not at all

○ Several days

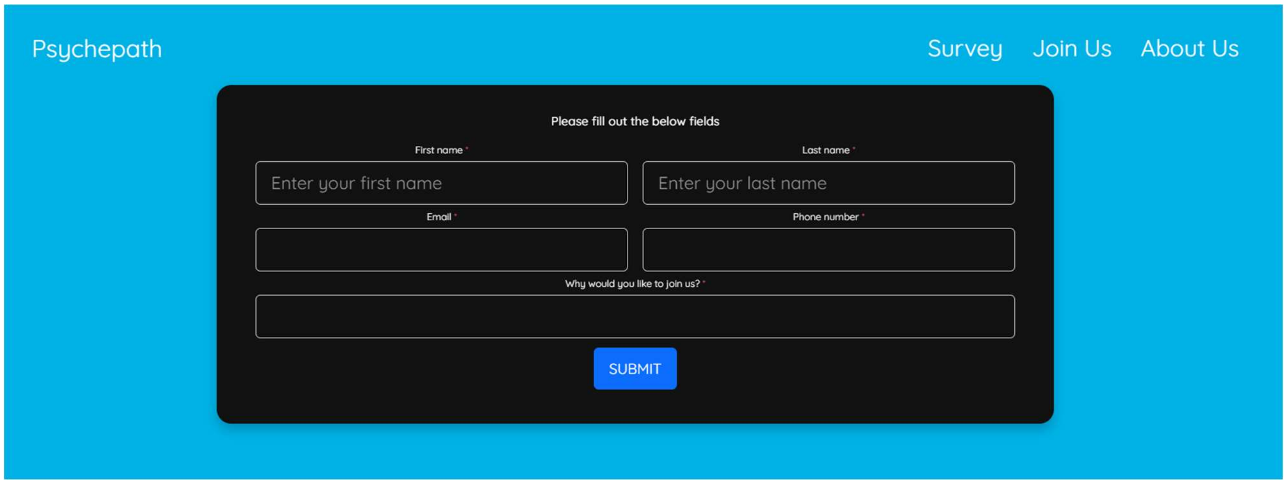
○ More than half the days

○ Nearly every day

(fig.23)



(fig.24)



Please fill out the below fields

First name *

Enter your first name

Last name *

Enter your last name

Email *

Phone number *

Why would you like to join us? *

SUBMIT

(fig.25)

Conclusion

We noticed from the readings and observations, that indeed something must be done to improve the situation regarding mental health issues. Building upon that we got in touch with some psychologists and developed a form for tagging. The form had been set such that the person filling the form wouldn't feel filling a mental health checkup, giving him/her some friendliness. On the other hand, it had enough information for us to process. We built a custom model to tag the person, following on to a certified medical test which would come up only if the model tags you. After getting the form submitted, you can contact professionals from the site only, without needing to face societies stigma. We hope for a better society and a Earth being a better place to live for people facing mental health issues.

References

1- Barbisch D, Koenig KL, Shih FY.
Is there a case for quarantine? Perspectives from SARS to Ebola. Disaster Med Public Health Prep 2015; 9: 547–53.

2- Miles SH. Kaci Hickox
Public health and the politics of fear.
Am J Bioeth. 2015;
15(4):17-9

3- Bai Y, Lin C-C, Lin C-Y, Chen J-Y, Chue C-M, Chou P.
Survey of stress reactions among health care workers involved with the SARS outbreak.
Psychiatr Serv 2004;
55: 1055–57.

4- Jeong H, Yim HW, Song Y-J, et al.

5- Mental health status of people isolated due to Middle East respiratory syndrome. Epidemiol Health 2016;
38: e2016048.

6- Liu X, Kakade M, Fuller CJ, et al.
Depression after exposure to stressful events: lessons learned from the severe acute respiratory syndrome epidemic. Compr Psychiatry 2012;
53: 15–23.

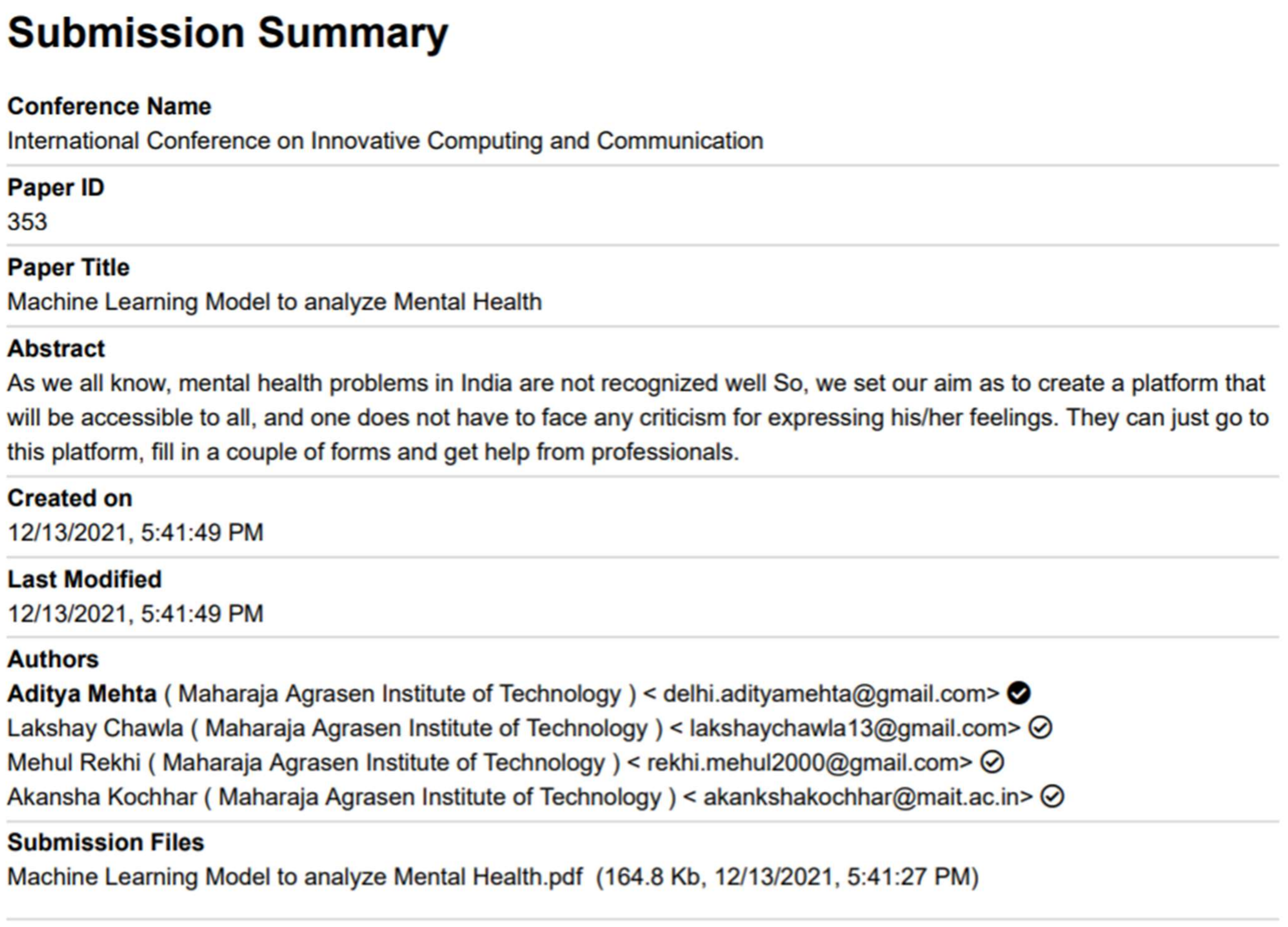
7- Taylor MR, Agho KE, Stevens GJ, Raphael B.
Factors influencing psychological distress during a disease epidemic: data from Australia's first outbreak of equine influenza.
BMC Public Health 2008;
8: 347.

8- Wu P, Fang Y, Guan Z, et al.
The psychological impact of the SARS epidemic on hospital employees in China: exposure, risk perception, and altruistic acceptance of risk.
Can J Psychiatry 2009;
54: 302–11.

9- Twende J, Martin G.
Gender differences in associations between digital media use and psychological well-being: evidence from three large datasets.
J Adolesc. 2020;
79:91-102

10- Iannotti RJ, Janssen I, Haug E, et al.
Interrelationships of adolescent physical activity, screen based sedentary behavior, and social and psychological health.
Int J Public Health. 2009;
54:191–198.

Proof of Research Paper Publishing



Submission Summary

Conference Name
International Conference on Innovative Computing and Communication

Paper ID
353

Paper Title
Machine Learning Model to analyze Mental Health

Abstract
As we all know, mental health problems in India are not recognized well So, we set our aim as to create a platform that will be accessible to all, and one does not have to face any criticism for expressing his/her feelings. They can just go to this platform, fill in a couple of forms and get help from professionals.

Created on
12/13/2021, 5:41:49 PM

Last Modified
12/13/2021, 5:41:49 PM

Authors
Aditya Mehta (Maharaja Agrasen Institute of Technology) < delhi.adityamehta@gmail.com> ✓
Lakshay Chawla (Maharaja Agrasen Institute of Technology) < lakshaychawla13@gmail.com> ✓
Mehul Rekhi (Maharaja Agrasen Institute of Technology) < rekhi.mehul2000@gmail.com> ✓
Akansha Kochhar (Maharaja Agrasen Institute of Technology) < akankshakochhar@mait.ac.in> ✓

Submission Files
Machine Learning Model to analyze Mental Health.pdf (164.8 Kb, 12/13/2021, 5:41:27 PM)

(fig.26)