INTRODUCTION

Stress could be a popular term denoting process believed to contribute to a spread of mental and physical conditions. There are three different perspectives of studying and defining the strain. This three perspectives differ in terms of relative emphasis each places on the environment, the organism, and therefore the interaction between organism and environment over time. Stress activates the sympathetic nervous system(SNS), which regulates pressure level, heart rate, and release of cathecolamines; and therefore the hypothalamic-pituitary-adrenal(HPA) axis, which regulates release of corticosteroids. Immune cells possess receptors for both cathecholamines and glucocorticoids, and there's substantial evidence that these hormones modulate immune function, as noted elsewhere during this volume. These parameters are defer from person to person, their body condition, age, gender etc.

More and more teenagers today are overloaded with adolescent stress from different aspects: academic future, self cognition, inter-personal, and affection. Long-lasting stress may lead to anxiety, withdrawal, aggression, or poor coping skills such as drug and alcohol use, threatening teenagers' health and development. Hence, it is important for both teenagers and their guardians/teachers to be aware of the stress in advance, and manage the stress before it becomes severe and starts causing health problems. The current social media micro-blog offers an open channel for us to timely and unobtrusively sense teenager's stress based on his/her tweeting contents and behaviors. This study describes a framework to further predict teenager's future adolescent stress level from micro-blog, and discusses how we address the challenges (data incompleteness and multifaceted prediction) using machine learning and multi-variant time series prediction techniques. Forthcoming events that may possibly influence teenager's stress levels are also incorporated into our prediction method. Our experimental results demonstrate the effectiveness of considering correlated features and event influence in prediction. To the best of our knowledge, this is the first work on predicting teenager's future stress level via micro-blog. College can be stressful for many freshmen as they cope with a variety of academic, personal, and social pressures . Although not all stress is negative, a certain level of stress can be beneficial to help improve performance. However, too much stress can adversely affect health in the annual survey of the American Freshman; the number of students reported feeling overwhelmed and stressed

has increased steadily in the last decade. Over 50% of college students suffer significant levels of stress during a typical college semester .

Consequently, there is a need to find innovative and costeffective strategies to help identify those students experiencing high levels of stress and negative emotions early on so that they can receive the appropriate treatment in order to prevent future mental illnesses. Social media use, such as Twitter and Facebook, has been rapidly growing, and research has already shown that data from these technologies can be used for novel approaches to public health surveillance. Twitter usage among young adults has increased 16% from 2012 to 2014. Currently, 32% of adults of the ages 18-29 years use Twitter, and the usage is expected to increase steadily in the future. People often have the need to share their emotions and experiences . Researchers have theorized that emotional sharing may fulfill a socio-affective need by eliciting attention, affection, and social support. Consequently, this may help individuals cope with their emotions and provide an immediate relief. Users often share their thoughts, feelings, and opinions on these social media platforms, and as a result, social media data may be used to provide real-time monitoring of stress and emotional state among college students. Previous studies have shown that Twitter data can be used to monitor a wide range of health outcomes, such as detecting human immunodeficiency virus infection outbreaks and predicting an individual's risk of depression. For example, De Choudhury et al conducted one of the first studies that used an individual's tweets to predict the risk of depression. The authors found that certain features extracted from a person's tweets collected over a 1-year period were highly associated with the risk of depression in adults, such as raised negative sentiment in the tweets, frequent mentions of antidepressant medication, and greater expression of religious involvement. Currently, no studies have examined whether Twitter data can be used to monitor stress level and emotional state among college students. Studying this topic is important because the large amount of social media data from college students' frequent use of social media can be used to help university officials and researchers monitor and reduce stress among college students.

LITERATURE SURVEY

Paper 1:

Monika Chauhan, Shivani V. Vora, Dipak Dabhi

In this paper, stress detection is proposed by using physiological parameters like heart rate and blood pressure. The difficulty in this paper is the parameters are very limited and the accuracy of the system depends on more number of parameters used.

Paper 2:

F. Mokhayeri, M-R. Akbarzadeh-T, S.Toosizadeh

This paper proposes the mental stress detection using three signals pupil diameter, Electrocardiogram and Photoplethysmorgam and is analysed by using soft computing techniques. This paper also includes fuzzy technique for uncertainty handling. This system worked on PIPAS dataset.

Paper 3:

Healey, J. A., & Picard, R. W. (2005)

This paper proposes the system of detecting the stress during real-world driving task. This system used the data in three different driving conditions i.e. during the rest, highway driving and city driving and this comes with the 97% of the accuracy. But the difficulty with the system is that only stress is detected only driving and no overall stress can estimated by this system.

Paper 4:

Huijie Lin, Jia Jia, Quan Guo, Yuanyuan Xue, Qi Li, Jie Huang, Lianhong Cai, Ling Feng

et al.

the study of "User-Level Psychological Stress Detection From Social Media Using Deep Neural Network" .The paper employs real online microblog data to investigate the correlations between users' stress and their tweeting content. It also defines two types of stress related attributes: - Low-level content attributes from a single tweet, including text, images and social interactions; and Userscope statistical attributes through their weekly micro-blog postings, mapping information of tweeting time, tweeting types and linguistic styles.

Paper 5:

Li-fang Zhang et al.

proposed the study on titled Occupational stress and teaching approaches among Chinese academics (2009). Researcher suggested that, controlling the self-rating abilities of the participants, the favorable conceptual changes in teaching approach and their role insufficiency predicated that the conceptual change in teaching strategy is negative.

Paper 6:

Another approach for stress analysis is Kavitha et al.

in her research titled -Role of stress among women employees forming majority workforce at IT sector in Chennai and Coimbatore (2012), she has focuses on the organizational role stress for the employees in the IT sector. She found in her research that, women face more stress than men in the organization and she viewed to be more specific married women faces more stress than the unmarried women.

Paper 7:

Amir Shani and Abraham Pizam(2009) et al.

Work-Related Depression among Hotel Employees have conducted a study on the depression of work among hotel employees in Central Florida. They have found that, incidence of depression among workers in the hospitality industry by evaluating the relationship between the occupational stress and work characteristics.

Paper 8:

Kayoko Urakawa and Kazuhito Yokoyam et al.

in their work on Sense of Coherence (SOC) may Reduce the Effects of Occupational Stress on Mental Health Status among Japanese Factory Workers (2009) has found the result i.e. adverse effects on mental health due to the job demand and job stress was positively associated with SOC, the mental health status of males in managerial work was adversely negative, where as it was positive among the female co-workers. Finally they found that, SOC is an important factor determining the coping ability over the job stress for both the genders.

WORKING

The working of the project is divided into parts.

Data Set Selection

After we have selected the dataset. The next step is to clean the data and transform it into the desired format as it is possible the dataset we use may be of different format. It is also possible that we may use multiple datasets from different sources which may be in different file formats. So to use them we need to convert them into the format we want to or the type that type prediction system supports. The reason behind this step is that it is possible that the data set contains the constraints which are not needed by the prediction system and including them makes the system complicated and may extend the processing time. Another reason behind data cleaning is the dataset may contain null value and garbage values too. So the solution to this issue is when the data is transformed the garbage values are replaced. There are many methods to perform that.

Data is the most import part when you work on prediction systems. It plays a very vital role your whole project i.e., you system depends on that data. So selection of data is the first and the critical step which should be performed properly, For our project we got the data from the kaggle website. These datasets were available for all. There are other tons of websites who provide such data. The dataset we choose wad selected based on the various

factors and constraints we were going to take under the consideration for our prediction system.

Data Cleaning and Data Transformation

Data Processing and Algorithm Implementation

After the data is been cleaned and transformed it's ready to process further. After the data has been cleaned and we have taken the required constraints. We divide the whole dataset into the two parts that can be either 70-30 or 80-20. The larger portion of the data is for the processing. The algorithm is applied on that part of data. Which helps the algorithm to learn on its own and make prediction for the future data or the unknown data. The algorithm is executed in which we take only the required constraints from the cleaned data. The output of the algorithm is in 'yes' and 'no'. It gives the error rate and the success rate.

Output and User Side Experience

After the prediction system is ready to use. The Website is developed for the user. The user just has to fill a form which consists of different options they need to select. They are like the State, city, date, time, year, type of causes, the type of vehicle, accident severity and so on. Once the user submits the form the algorithm is triggered and the input given by the user is passed to the prediction system. The user is given how accident prone the road can be in percentage.

SYSTEM ANALYSIS

Functional Requirements:

Requirement analysis is a software engineering task that bridges the gap between system level software allocation & software design.

It enables the system engineer to specify software interface with other system elements & establishes design constraints that the software must meet.

It provides the software design with a representations of information & function that can be translated to date, architecture & procedural design.

Non Functionality Requirements:

Security:

Project level security is set. User needs to login when they start the program option is also provided to create the additional user and level security. Presently user level security is not set but can be implemented with few modification.

Reliability, Availability, Maintainability:

It is very user friendly, software is secure and there is not much maintenance. Project can be upgraded as per the requirement step by step.

Configuration and Compatibility:

Describes requirements such as those connected with individual customization or operations in specific competing environments.

Usability:

Describes items that will ensure the user friendliness of the software.

Eg: Includes error messages that direct the user to a solution, input range checkings soon as entries are made and order of choices and screen corresponding to user performances.

FEASIBILITY SYSTEM

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

1. TECHNICAL FEASIBILITY
2. ECONOMICAL FEASIBILITY

3 OPERATIONAL FEASIBILITY

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

OPERATIONAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

SOFTWARE AND HARDWARE SPECIFICATION

HARDWARE REQUIREMENTS:

Processor	:	Pentium IV 2.4 GHz.
Hard Disk	:	250 GB.
Monitor	:	15 VGA Color.
RAM	:	1 GB
Mouse	:	Optical
Keyboard	:	Multimedia

(These are Minimum Configuration)

SOFTWARE REQUIREMENTS:

Operating system	:	Windows XP Professional / Windows7 or More
Coding Language	:	Python
IDE	:	Jupiter Notebook

SOFTWARE AND LANGUAGES USED

PYTHON

Python is an interpreted, high-level and a general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It is the most used programming language presently. It provides constructs that enable clear programming on both small and large scales. The logistic regression in the system is implemented in jupyter and the algorithm is written in python language.

Eventually, you'll probably want to put your Python programs, or at least your function definitions, in a file you create and edit with a text editor, and then load it into Python later. This saves you having to re-type everything every time you run. The standard Unix implementation of Python provides an integrated development environment called idle, which bundles a Python interpreter window with a Pythonaware text editor. To start up idle, log in to the server from an xterm and type IDLE. You will get a Python shell window, which is an ordinary Python interpreter except that it allows some limited editing capabilities. The real power of idle comes from the use of the integrated editor. To get an editor window for a new file, just choose New Window from the File menu on the Python Shell window. If you want to work with an existing file instead, just choose Open from the File menu, and pick the file you want from the resulting dialog box. You can type text into the editor window, and cut and paste in a fashion that will probably be familiar to most computer users.

JUPYTER:

Jupyter exists to develop open-source software. It is used for open-standards, and services for interactive computing across dozens of programming languages. It is an opensource web application that allows you to create and share documents and code live. Which is a very big advantage of Jupyter. It can be used for data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more. We used Jupyter to run the algorithm.

The Jupyter Notebook is an interactive computing environment that enables users to author notebook documents that include: - Live code - Interactive widgets - Plots - Narrative text - Equations - Images – Video. These documents provide a complete and self-contained record of a computation that can be converted to various formats and shared with others using email, Dropbox, version control systems (like git/GitHub) or nbviewer.jupyter.org.

COMPONENTS

The Jupyter Notebook combines three components:

The notebook web application: An interactive web application for writing and running code interactively and authoring notebook documents.

Kernels: Separate processes started by the notebook web application that runs users' code in a given language and returns output back to the notebook web application. The kernel also handles things like computations for interactive widgets, tab completion and introspection.

Notebook documents: Self-contained documents that contain a representation of all content visible in the notebook web application, including inputs and outputs of the computations, narrative text, equations, images, and rich media representations of objects. Each notebook document has its own kernel.

HTML, CSS, JSCRIPT

These are the most commonly used webpage development languages. The user interface designed for the prediction system is been developed using these languages. The website acts as an interface which takes the input those are the various constraints from the users and **passes to the program to work upon.**

METHODOLOGY

What is Machine Learning? A definition

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Machine learning algorithms are often categorized as supervised or unsupervised.

Supervised machine learning algorithms

can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

Unsupervised machine learning algorithms

are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

Semi-supervised machine learning algorithms

fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiringunlabeled data generally doesn't require additional resources.

Reinforcement machine learning algorithms

is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

ALGORITHMS

Naive Bayes Classifiers

In machine learning, Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on the application of Bayes' theorem with strong (naive) assumptions of independence between characteristics. They are among the simplest Bayesian network models.

Naïve Bayes has been extensively studied since the 1960s. It was introduced (albeit not by that name) into the text retrieval community in the early 1960s and remains a popular (reference) method for categorizing text, the problem of judging documents as belonging to a category or to the other (such as spam or legitimate, sports or political, etc.) with word frequencies as characteristics. With proper preprocessing, it is competitive in this domain with more advanced methods that include supporting vector machines. Naïve Bayes classifiers are highly scalable and require a number of linear parameters in the number of variables (characteristics / predictors) in a learning problem. Maximum likelihood training can be done by evaluating an expression in closed form, which takes linear time, rather than by an expensive iterative approach as is used for many other types of classifiers.

In the statistical and computer literature, naive Bayes models are known by a variety of names, including Simple Bayes and Independence Bayes. All of these names refer to the use of Bayes' theorem in the classifier decision rule, but the naive Bayes is not (necessarily) a Bayesian method. For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning environment. In many practical applications, parameter estimation for naive Bayes models uses the maximum likelihood method; in other words, you can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naïve design and seemingly simplified assumptions, naïve Bayes classifiers performed quite well in many complex real-world situations. However, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes' classification is overtaken by other approaches, such as Fed Trees or Random Forests. A naive advantage of Bayes is that it requires only a small amount of training data to estimate the parameters needed for classification. How does Bayes' naive algorithm work? Naive

Bayes is a probabilistic machine learning algorithm based on Bayes' theorem, used in a wide variety of classification tasks. In this post, you will get a clear and complete understanding of the Naive Bayes algorithm and all the necessary concepts so that there is no room for doubts or gaps in understanding.

Steps to implement naive bayes algorithm

- Load the CSV information into Python and change all strings into floats. Separate all information into preparing information and test information. All things considered, the Naive Bayes approach makes forecasts from the preparation information, while it utilizes the test information to assess the precision of the model. For this situation, we have made 67% preparing information and 33% test information.

- Presently the time has come to examine the information. Every datum will be isolated dependent on the class it has a place with. From that point forward, we have to ascertain the mean and standard deviation for every one of the information inside each trait inside each class.

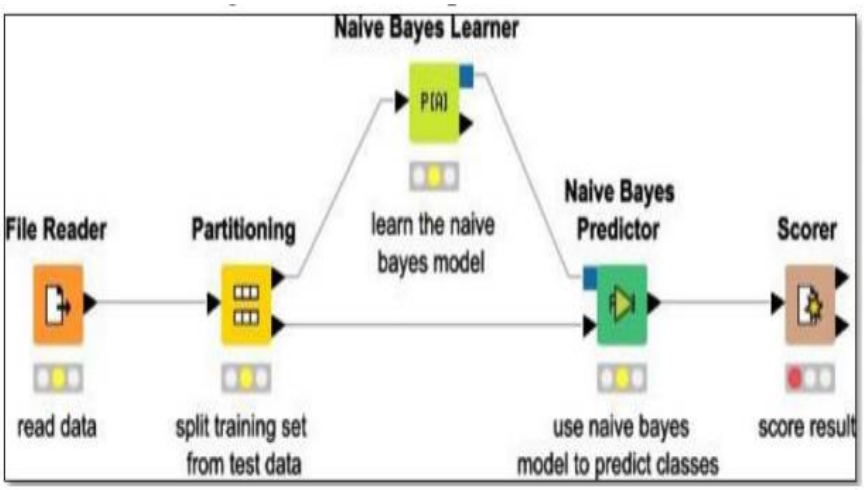
- Presently it an opportunity to utilize the preparation information to make expectations.

- We have to utilize the Gaussian Probability Density Function here.

- In light of the rundown of information for each class, comprising of the mean and change, in view of the info esteem, we process the probability that the esteem will have a place with a specific class (utilizing the Gaussian Probability DensityFunction).

- Our forecast will relate to the class with the biggest likelihood.

- At long last, we have to know how exact our gauge was. It is the rate right out of all expectations made.



Pseudo code for Navie bayes Algorithm:

```
import sklearn

from sklearn.naive_bayes import BernoulliNB

from sklearn import metrics

from sklearn.metrics import accuracy_score

BernNB = BernoulliNB(binarize=.1)

BernNB.fit(X_train,y_train)

print(BernNB)

y_expect = y_test

y_pred = BernNB.predict(X_test)

print(accuracy_score(y_expect,y_pred)*100)

score_nb = accuracy_score(y_expect,y_pred)*100
```

K-Nearest Neighbor(KNN) Algorithm

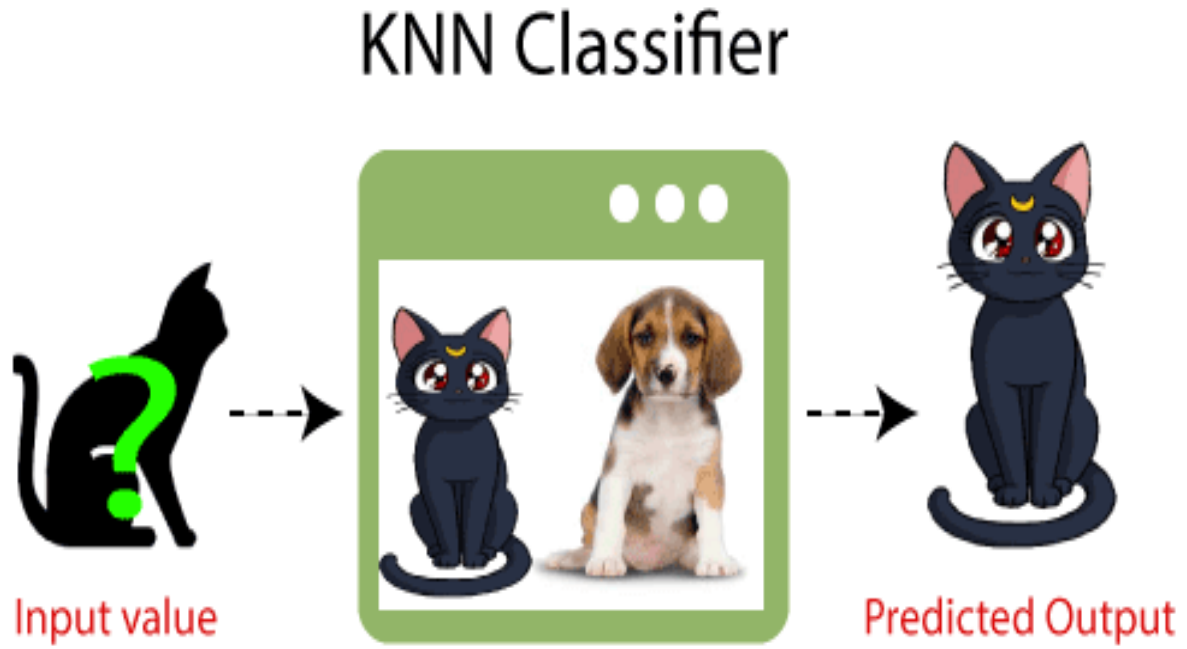
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

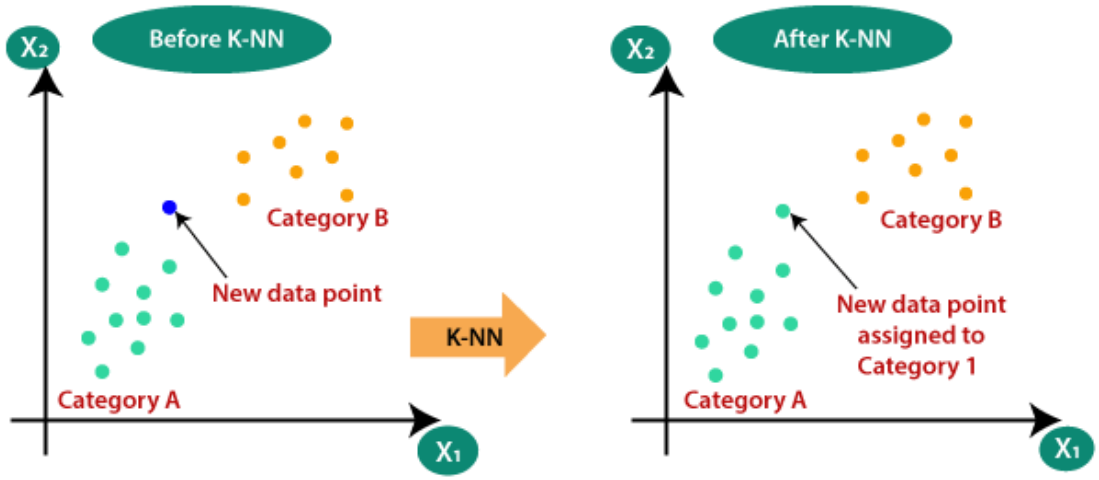
Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset.

Consider the below diagram:



How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

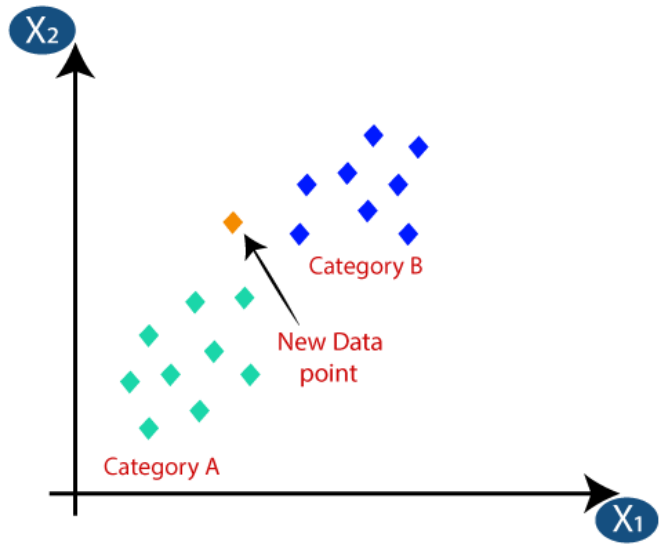
Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

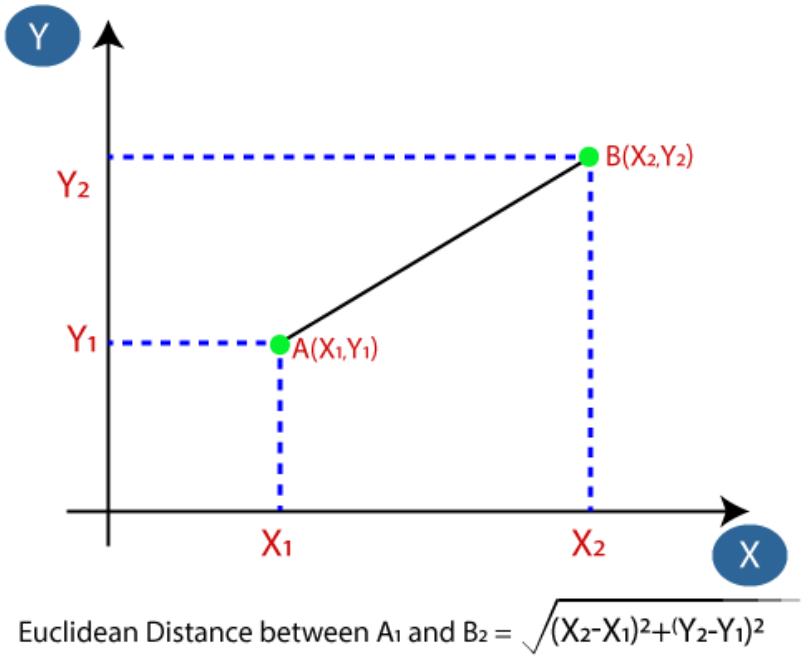
Step-6: Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



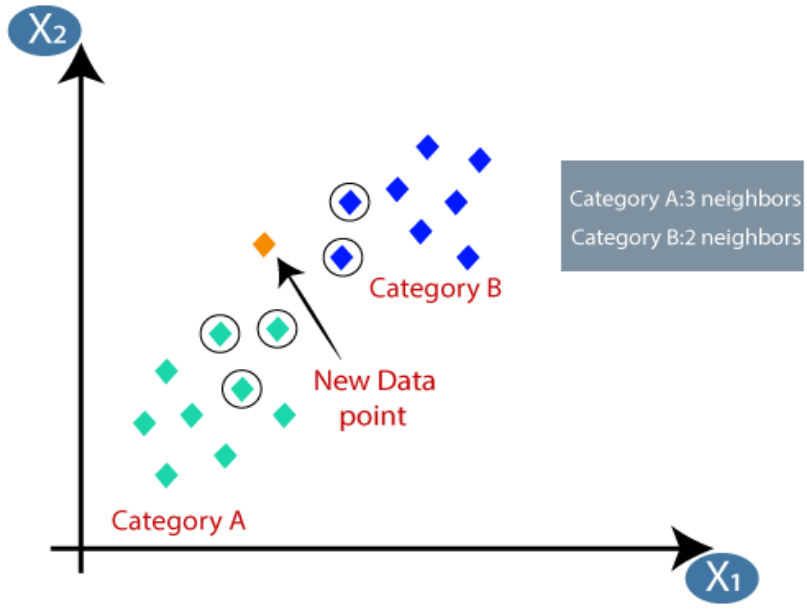
Firstly, we will choose the number of neighbors, so we will choose the k=5.

Next, we will calculate the Euclidean distance between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B.

Consider the below image:



As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Python implementation of the KNN algorithm

To do the Python implementation of the K-NN algorithm, we will use the same problem and dataset which we have used in Logistic Regression. But here we will improve the performance of the model. Below is the problem description:

Problem for K-NN Algorithm: There is a Car manufacturer company that has manufactured a new SUV car. The company wants to give the ads to the users who are interested in buying that SUV. So for this problem, we have a dataset that contains multiple user's information through the social network. The dataset contains lots of information but the Estimated Salary and Age we will consider for the independent variable and the Purchased variable is for the dependent variable.

Below is the dataset:

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

Steps to implement the K-NN algorithm:

- Data Pre-processing step
- Fitting the K-NN algorithm to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

Data Pre-Processing Step:

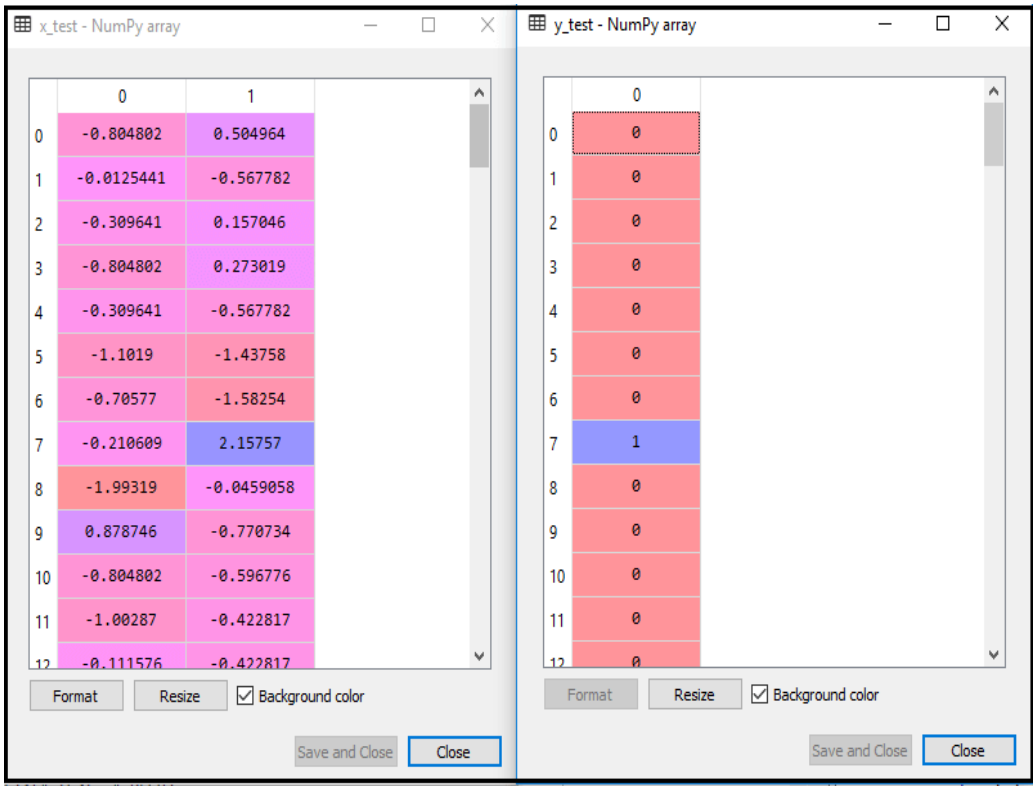
The Data Pre-processing step will remain exactly the same as Logistic Regression.

Below is the code for it:

```
1. # importing libraries
2. import numpy as nm
3. import matplotlib.pyplot as mtp
4. import pandas as pd
```

```
#importing datasets
data_set= pd.read_csv('user_data.csv')
#Extracting Independent and dependent Variable
x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)

#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

By executing the above code, our dataset is imported to our program and well pre-processed. After feature scaling our test dataset will look like:



From the above output image, we can see that our data is successfully scaled.

Fitting K-NN classifier to the Training data:

Now we will fit the K-NN classifier to the training data. To do this we will import the KNeighbors Classifier class of Sklearn Neighbors library. After importing the class, we will create the Classifier object of the class. The Parameter of this class will be

- **n_neighbors:** To define the required neighbors of the algorithm. Usually, it takes 5.
- **metric='minkowski':** This is the default parameter and it decides the distance between the points.
- **p=2:** It is equivalent to the standard Euclidean metric.

And then we will fit the classifier to the training data.

Below is the code for it:

```
1. #Fitting K-NN classifier to the training set
2. from sklearn.neighbors import KNeighborsClassifier
3. classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
4. classifier.fit(x_train, y_train)
```

Output: By executing the above code, we will get the output as:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

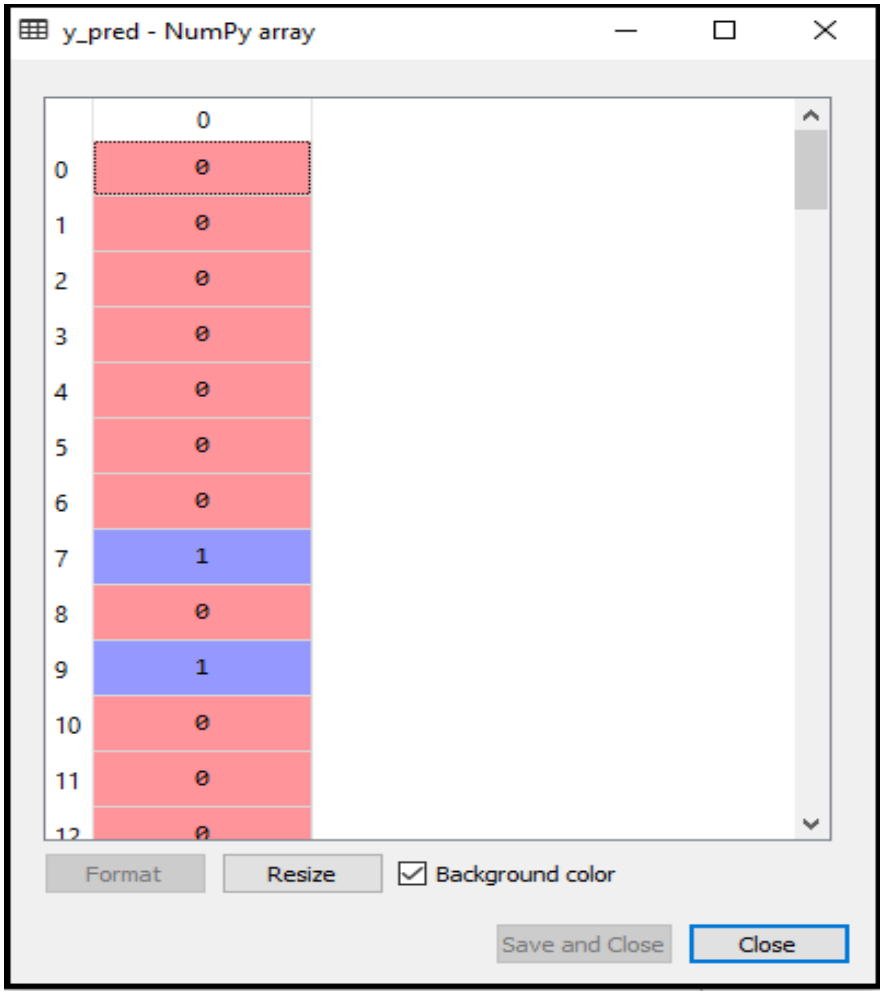
Predicting the Test Result:

To predict the test set result, we will create a y_pred vector as we did in Logistic Regression. Below is the code for it:

```
1. #Predicting the test set result
2. y_pred= classifier.predict(x_test)
```

Output:

The output for the above code will be:



Creating the Confusion Matrix :

Now we will create the Confusion Matrix for our K-NN model to see the accuracy of the classifier.

Below is the code for it:

```
1. #Creating the Confusion matrix
2. from sklearn.metrics import confusion_matrix
3. cm= confusion_matrix(y_test, y_pred)
```

In above code, we have imported the confusion_matrix function and called it using the variable cm.

Output: By executing the above code, we will get the matrix as below:

In the above image, we can see there are 64+29= 93 correct predictions and 3+4= 7 incorrect predictions, whereas, in Logistic Regression, there were 11 incorrect predictions. So we can say that the performance of the model is improved by using the K-NN algorithm.

Visualizing the Training set result:

Now, we will visualize the training set result for K-NN model. The code will remain same as we did in Logistic Regression, except the name of the graph.

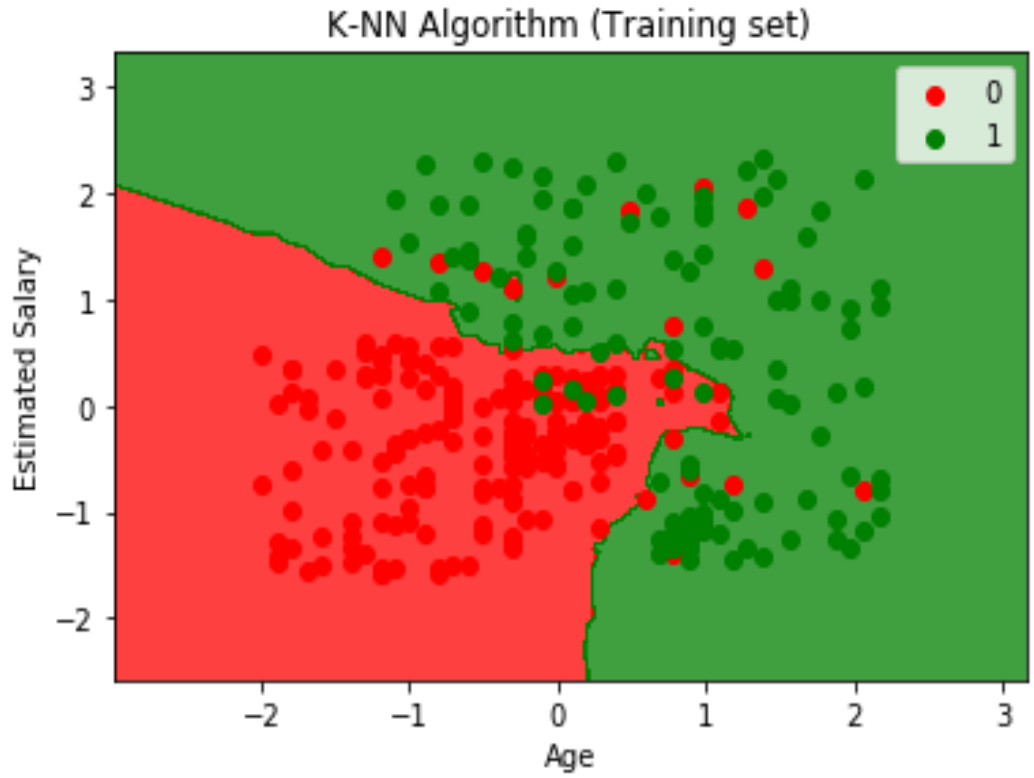
Below is the code for it:

```
1. #Visulaizing the trianing set result
2. from matplotlib.colors import ListedColormap
3. x_set, y_set = x_train, y_train
4. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() -
     1, stop = x_set[:, 0].max() + 1, step  =0.01),
```

```
5. nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1
   .shape),
7. alpha = 0.75, cmap = ListedColormap(('red','green' )))
8. mtp.xlim(x1.min(), x1.max())
9. mtp.ylim(x2.min(), x2.max())
10. for i, j in enumerate(nm.unique(y_set)):
11. mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
12. c = ListedColormap(('red', 'green'))(i), label = j)
13. mtp.title('K-NN Algorithm (Training set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()
```

Output:

By executing the above code, we will get the below graph:



The output graph is different from the graph which we have occurred in Logistic Regression. It can be understood in the below points:

As we can see the graph is showing the red point and green points. The green points are for Purchased(1) and Red Points for not Purchased(0) variable.

The graph is showing an irregular boundary instead of showing any straight line or any curve because it is a K-NN algorithm, i.e., finding the nearest neighbor.

The graph has classified users in the correct categories as most of the users who didn't buy the SUV are in the red region and users who bought the SUV are in the green region.

The graph is showing good result but still, there are some green points in the red region and red points in the green region. But this is no big issue as by doing this model is prevented from overfitting issues.

Hence our model is well trained.

Visualizing the Test set result:

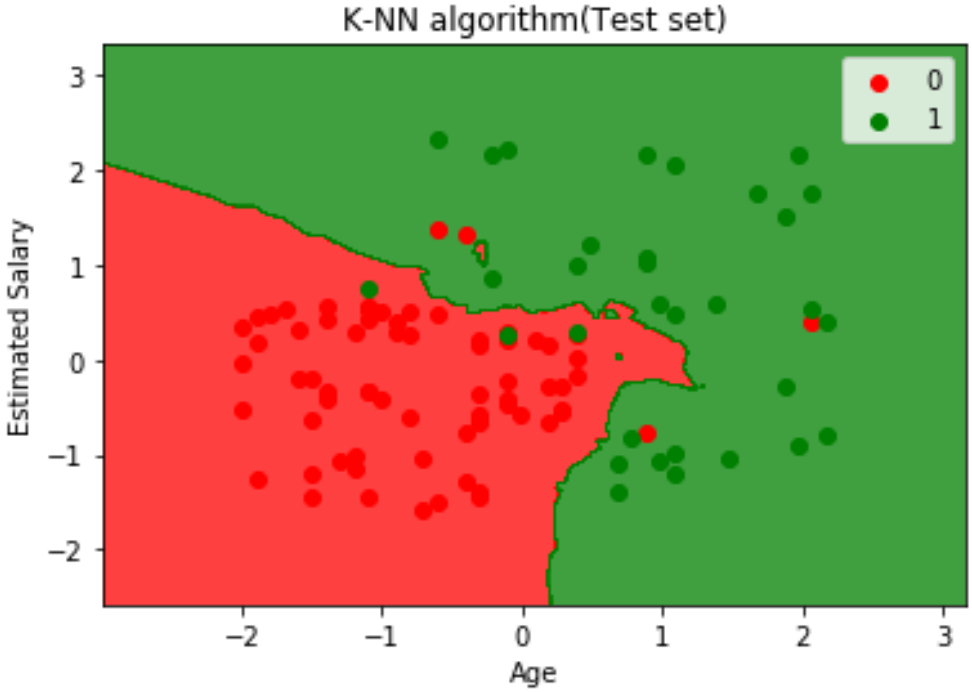
After the training of the model, we will now test the result by putting a new dataset, i.e., Test dataset. Code remains the same except some minor changes: such as x_train and y_train will be replaced by x_test and y_test.

Below is the code for it:

```
1. #Visualizing the test set result
2. from matplotlib.colors import ListedColormap
3. x_set, y_set = x_test, y_test
4. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() -
     1, stop = x_set[:, 0].max() + 1, step  =0.01),
5. nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1
   .shape),
7. alpha = 0.75, cmap = ListedColormap(('red','green' )))
8. mtp.xlim(x1.min(), x1.max())
9. mtp.ylim(x2.min(), x2.max())
10. for i, j in enumerate(nm.unique(y_set)):
```

```
11. mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
12. c = ListedColormap(('red', 'green'))(i), label = j)
13. mtp.title('K-NN algorithm(Test set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()
```

Output:

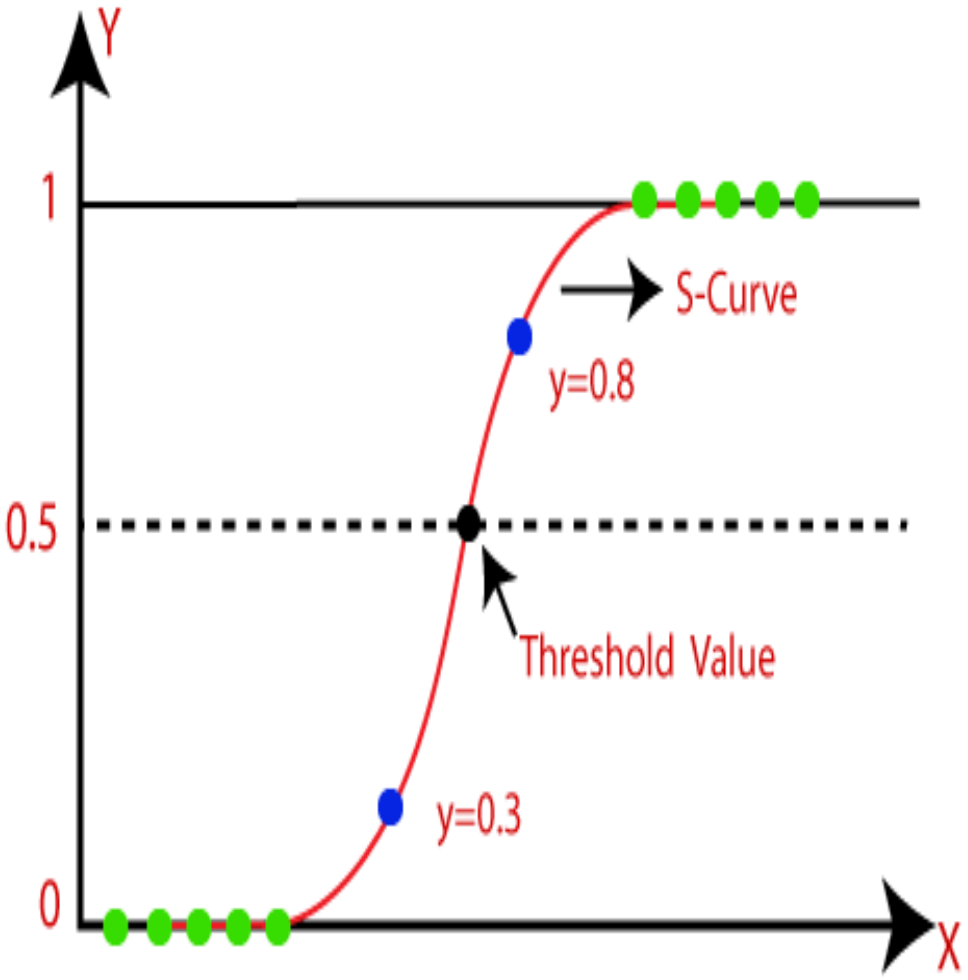


The above graph is showing the output for the test data set. As we can see in the graph, the predicted output is well good as most of the red points are in the red region and most of the green points are in the green region.

However, there are few green points in the red region and a few red points in the green region. So these are the incorrect observations that we have observed in the confusion matrix(7 Incorrect output).

Logistic Regression Algorithm

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Note: Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic regression, but is used to classify samples; Therefore, it falls under the classification algorithm.

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

- Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

Python Implementation of Logistic Regression (Binomial)

To understand the implementation of Logistic Regression in Python, we will use the below example:

14.9M
338
How to find Nth Highest Salary in SQL

Example:

There is a dataset given which contains the information of various users obtained from the social networking sites. There is a car making company that has recently launched a new SUV car. So the company wanted to check how many users from the dataset, wants to purchase the car.

For this problem, we will build a Machine Learning model using the Logistic regression algorithm. The dataset is shown in the below image. In this problem, we will predict the purchased variable (Dependent Variable) by using age and salary (Independent variables).

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

Steps in Logistic Regression: To implement the Logistic Regression using Python, we will use the same steps as we have done in previous topics of Regression. Below are the steps:

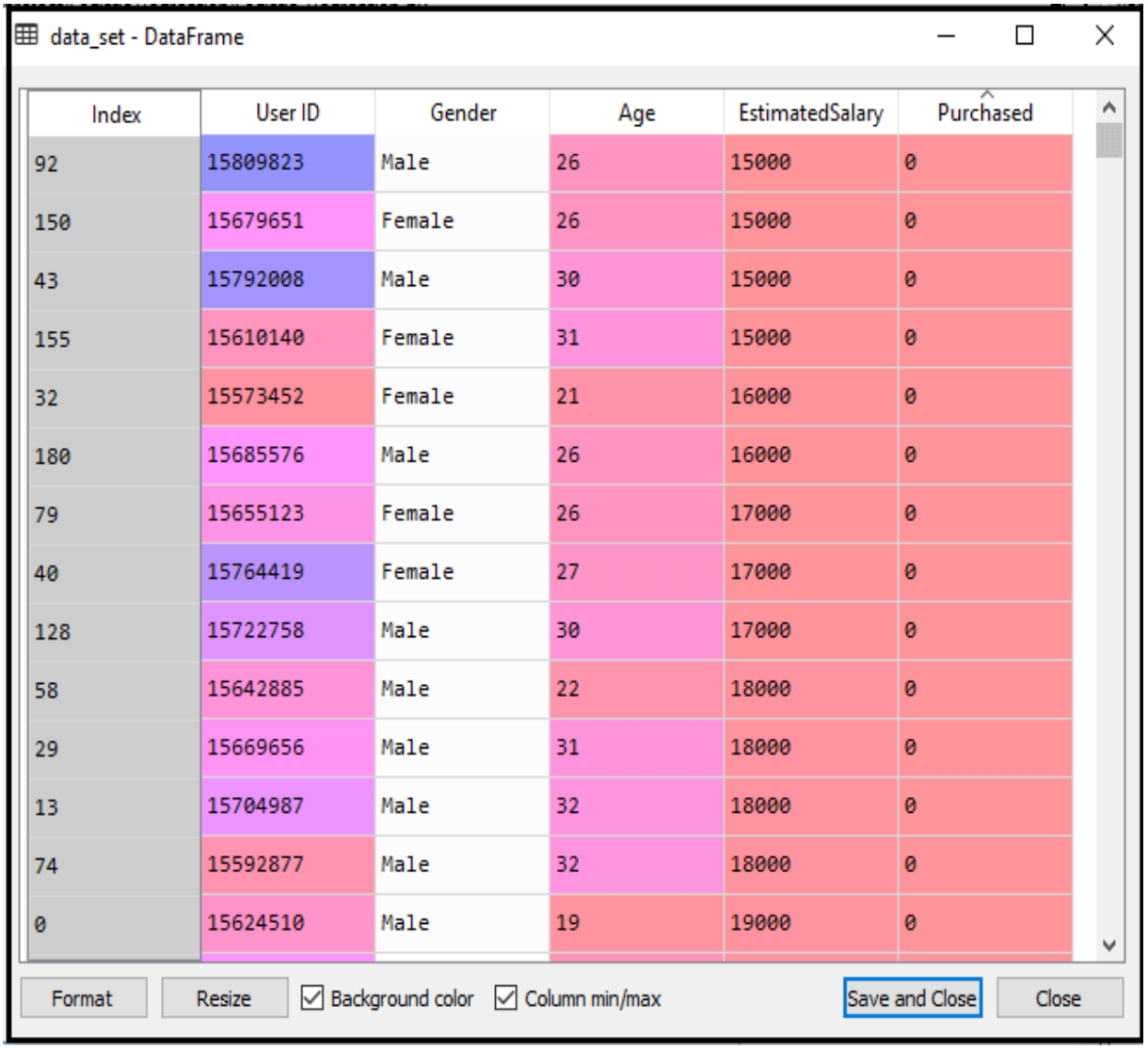
- Data Pre-processing step
- Fitting Logistic Regression to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

1. Data Pre-processing step

In this step, we will pre-process/prepare the data so that we can use it in our code efficiently. It will be the same as we have done in Data pre-processing topic. The code for this is given below:

```
1. #Data Pre-procesing Step
2. # importing libraries
3. import numpy as nm
4. import matplotlib.pyplot as mtp
5. import pandas as pd
6. #importing datasets
7. data_set= pd.read_csv('user_data.csv')
```

By executing the above lines of code, we will get the dataset as the output. Consider the given image:



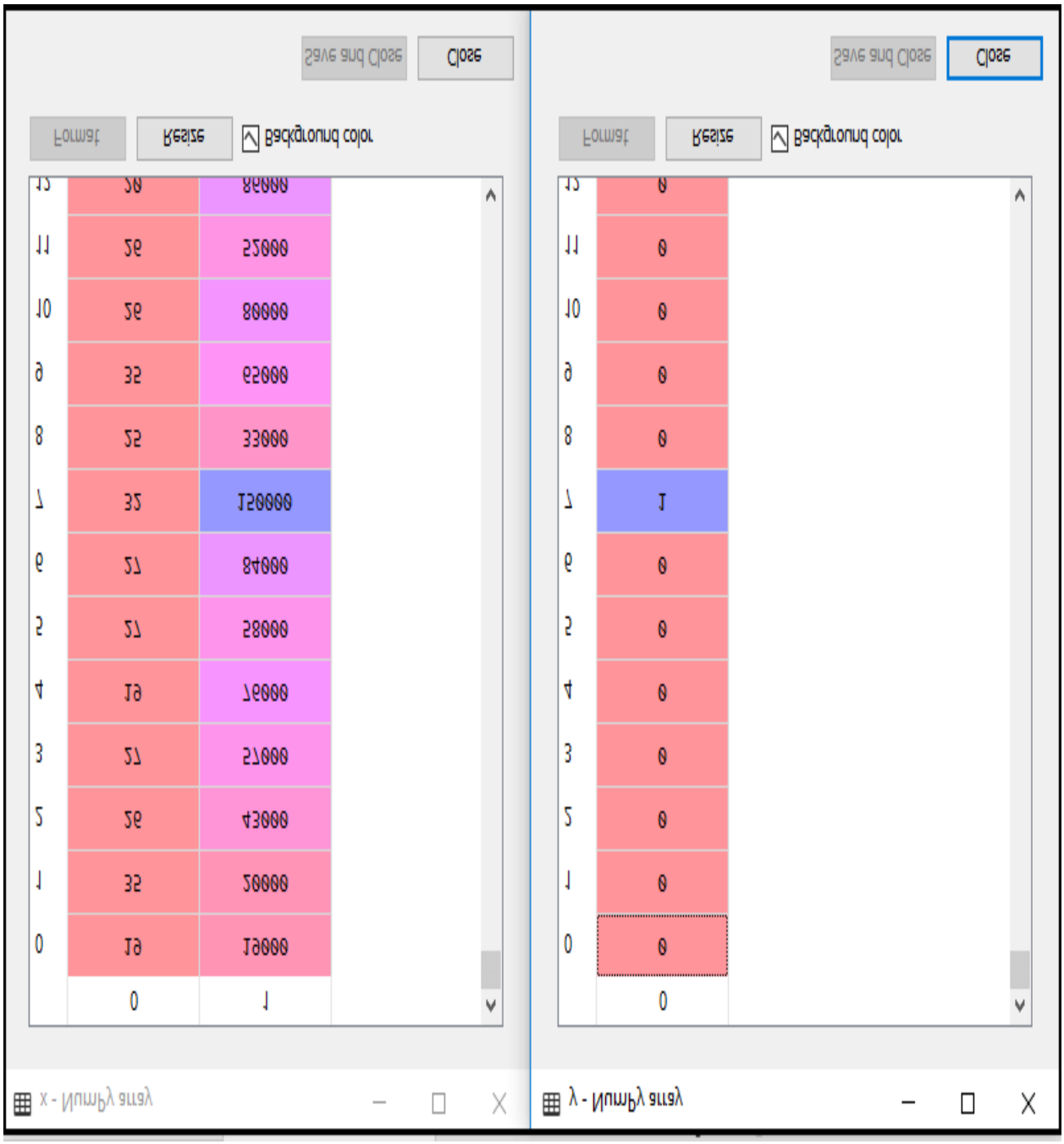
Index	User ID	Gender	Age	EstimatedSalary	Purchased
92	15809823	Male	26	15000	0
150	15679651	Female	26	15000	0
43	15792008	Male	30	15000	0
155	15610140	Female	31	15000	0
32	15573452	Female	21	16000	0
180	15685576	Male	26	16000	0
79	15655123	Female	26	17000	0
40	15764419	Female	27	17000	0
128	15722758	Male	30	17000	0
58	15642885	Male	22	18000	0
29	15669656	Male	31	18000	0
13	15704987	Male	32	18000	0
74	15592877	Male	32	18000	0
0	15624510	Male	19	19000	0

Now, we will extract the dependent and independent variables from the given dataset.

Below is the code for it:

1. #Extracting Independent and dependent Variable
2. x= data_set.iloc[:, [2,3]].values
3. y= data_set.iloc[:, 4].values

In the above code, we have taken [2, 3] for x because our independent variables are age and salary, which are at index 2, 3. And we have taken 4 for y variable because our dependent variable is at index 4. The output will be:

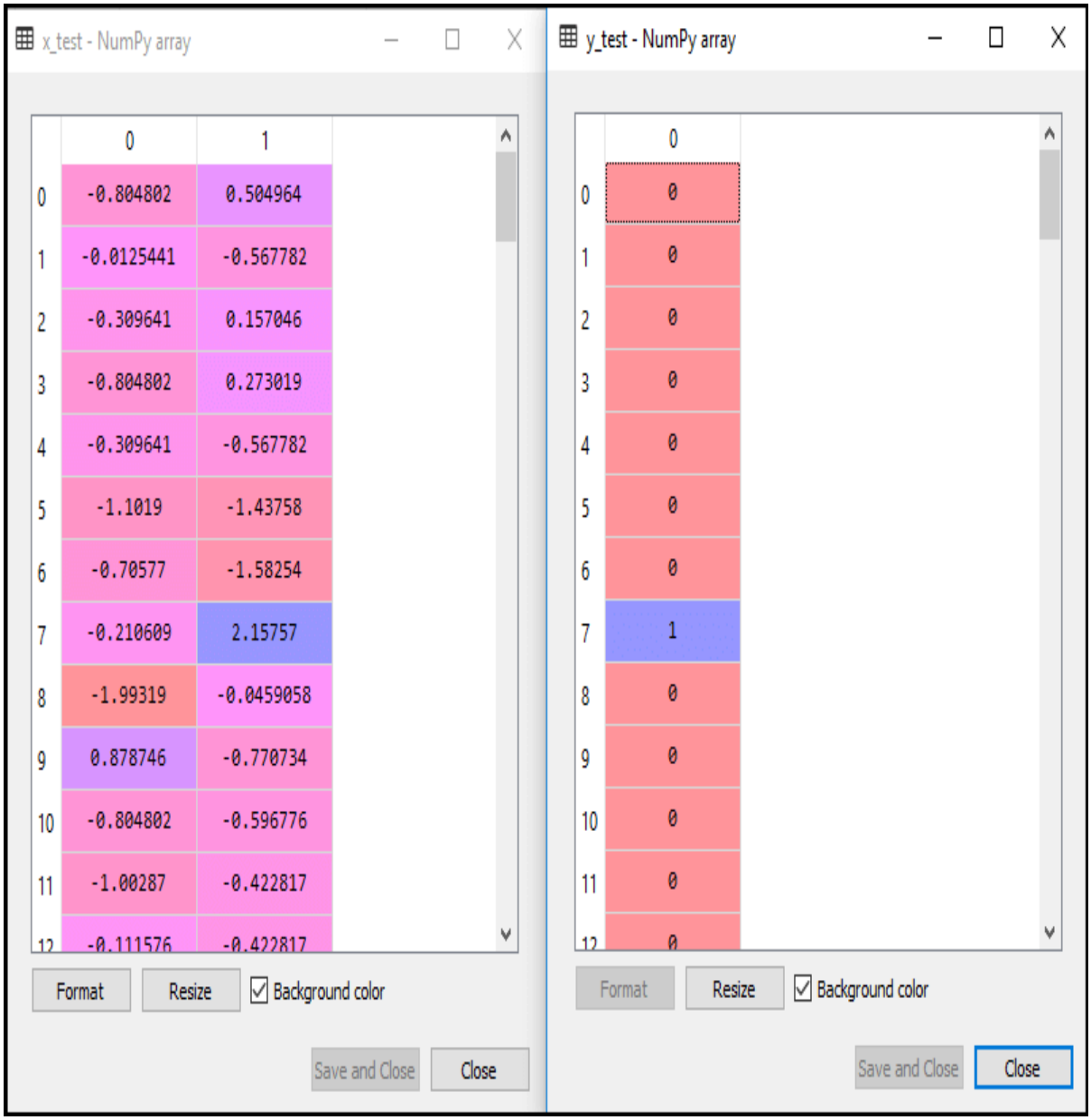


Now we will split the dataset into a training set and test set. Below is the code for it:

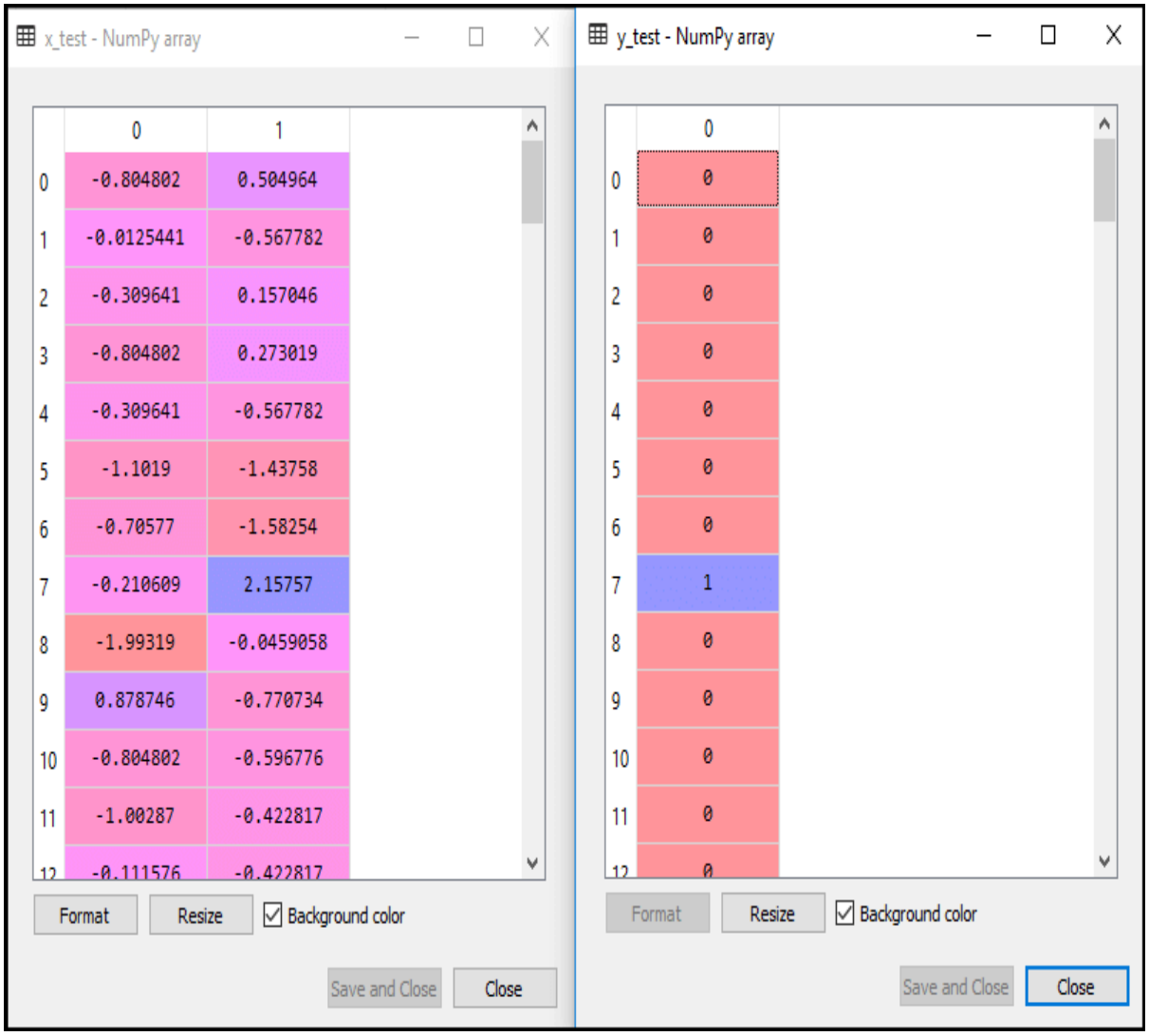
```
1. # Splitting the dataset into training and test set.
2. from sklearn.model_selection import train_test_split
3. x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)
```

The output for this is given below:

For test set



For training set:



In logistic regression, we will do feature scaling because we want accurate result of predictions. Here we will only scale the independent variable because dependent variable have only 0 and 1 values.

Below is the code for it:

```
#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

The scaled output is given below:

2. Fitting Logistic Regression to the Training set

We have well prepared our dataset, and now we will train the dataset using the training set. For providing training or fitting the model to the training set, we will import the LogisticRegression class of the sklearn library.

After importing the class, we will create a classifier object and use it to fit the model to the logistic regression.

Below is the code for it:

```
1. #Fitting Logistic Regression to the training set
2. from sklearn.linear_model import LogisticRegression
3. classifier= LogisticRegression(random_state=0)
4. classifier.fit(x_train, y_train)
```

Output: By executing the above code, we will get the below output:

```
1. LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
2. intercept_scaling=1, l1_ratio=None, max_iter=100,
3. multi_class='warn', n_jobs=None, penalty='l2',
4. random_state=0, solver='warn', tol=0.0001, verbose=0,
5. warm_start=False)
```

Hence our model is well fitted to the training set.

3. Predicting the Test Result

Our model is well trained on the training set, so we will now predict the result by using test set data.

Below is the code for it:

```
1. #Predicting the test set result
2. y_pred= classifier.predict(x_test)
```

In the above code, we have created a y_pred vector to predict the test set result.

Output: By executing the above code, a new vector (y_pred) will be created under the variable explorer option. It can be seen as:

The above output image shows the corresponding predicted users who want to purchase or not purchase the car.

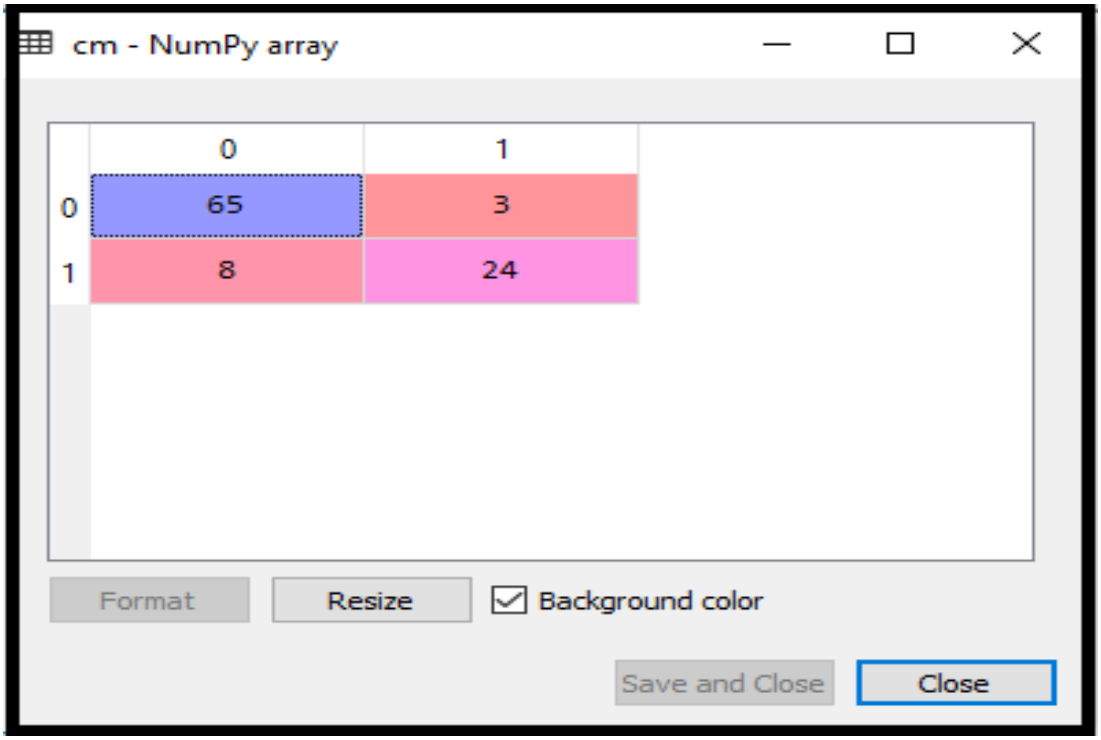
4. Test Accuracy of the result

Now we will create the confusion matrix here to check the accuracy of the classification. To create it, we need to import the confusion_matrix function of the sklearn library. After importing the function, we will call it using a new variable cm. The function takes two parameters, mainly y_true(the actual values) and y_pred (the targeted value return by the classifier).

Below is the code for it:

```
1. #Creating the Confusion matrix
2. from sklearn.metrics import confusion_matrix
3. cm= confusion_matrix()
```

Output: By executing the above code, a new confusion matrix will be created. Consider the below image:



We can find the accuracy of the predicted result by interpreting the confusion matrix. By above output, we can interpret that 65+24= 89 (Correct Output) and 8+3= 11(Incorrect Output).

5. Visualizing the training set result

Finally, we will visualize the training set result. To visualize the result, we will use ListedColormap class of matplotlib library.

Below is the code for it:

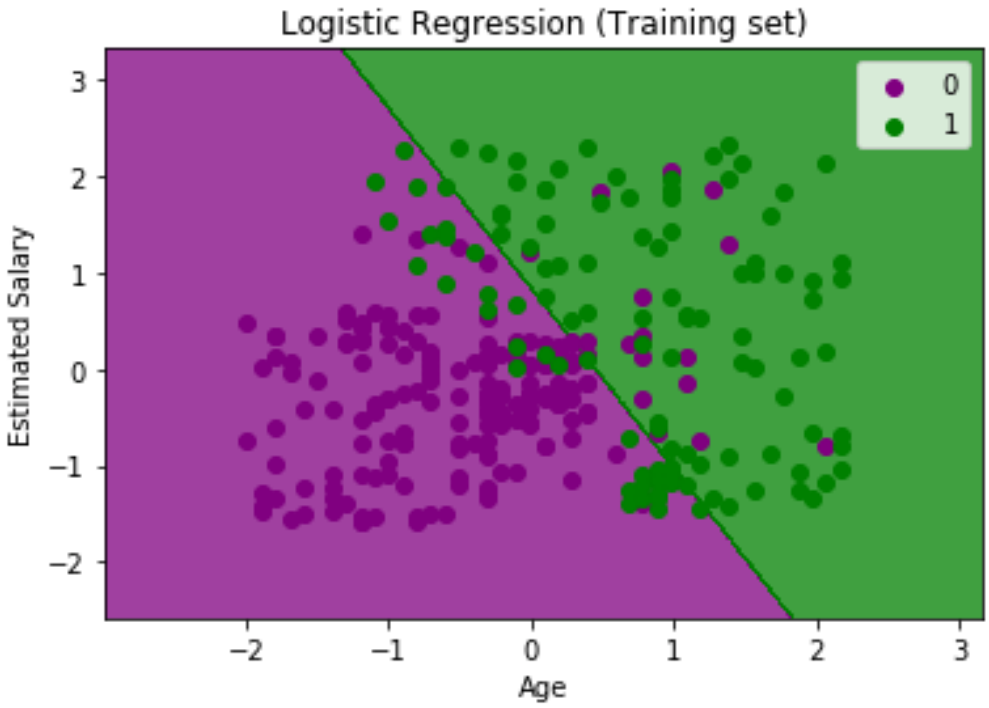
```
1. #Visualizing the training set result
2. from matplotlib.colors import ListedColormap
3. x_set, y_set = x_train, y_train
4. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() -
     1, stop = x_set[:, 0].max() + 1, step  =0.01),
5. nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1
   .shape),
7. alpha = 0.75, cmap = ListedColormap(('purple','green' )))
8. mtp.xlim(x1.min(), x1.max())
9. mtp.ylim(x2.min(), x2.max())
10. for i, j in enumerate(nm.unique(y_set)):
11. mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
12. c = ListedColormap(('purple', 'green'))(i), label = j)
13. mtp.title('Logistic Regression (Training set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()
```

In the above code, we have imported the ListedColormap class of Matplotlib library to create the colormap for visualizing the result. We have created two new

variables x_set and y_set to replace x_train and y_train. After that, we have used the nm.meshgrid command to create a rectangular grid, which has a range of -1(minimum) to 1 (maximum). The pixel points we have taken are of 0.01 resolution.

To create a filled contour, we have used mtp.contourf command, it will create regions of provided colors (purple and green). In this function, we have passed the classifier.predict to show the predicted data points predicted by the classifier.

Output: By executing the above code, we will get the below output:



The graph can be explained in the below points:

- In the above graph, we can see that there are some Green points within the green region and Purple points within the purple region.
- All these data points are the observation points from the training set, which shows the result for purchased variables.
- This graph is made by using two independent variables i.e., Age on the x-axis and Estimated salary on the y-axis.
- The purple point observations are for which purchased (dependent variable) is probably 0, i.e., users who did not purchase the SUV car.

- The green point observations are for which purchased (dependent variable) is probably 1 means user who purchased the SUV car.
- We can also estimate from the graph that the users who are younger with low salary, did not purchase the car, whereas older users with high estimated salary purchased the car.
- But there are some purple points in the green region (Buying the car) and some green points in the purple region(Not buying the car). So we can say that younger users with a high estimated salary purchased the car, whereas an older user with a low estimated salary did not purchase the car.

The goal of the classifier:

We have successfully visualized the training set result for the logistic regression, and our goal for this classification is to divide the users who purchased the SUV car and who did not purchase the car. So from the output graph, we can clearly see the two regions (Purple and Green) with the observation points. The Purple region is for those users who didn't buy the car, and Green Region is for those users who purchased the car.

Linear Classifier:

As we can see from the graph, the classifier is a Straight line or linear in nature as we have used the Linear model for Logistic Regression. In further topics, we will learn for non-linear Classifiers.

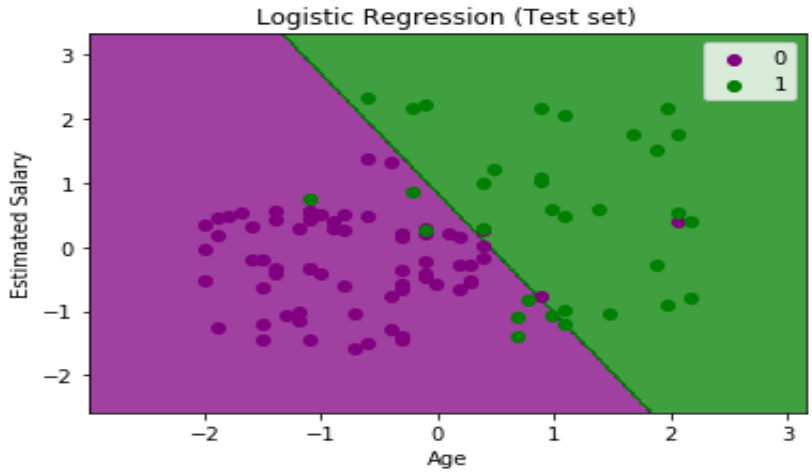
Visualizing the test set result:

Our model is well trained using the training dataset. Now, we will visualize the result for new observations (Test set). The code for the test set will remain same as above except that here we will use x_test and y_test instead of x_train and y_train.

Below is the code for it:

```
1. #Visulaizing the test set result
2. from matplotlib.colors import ListedColormap
3. x_set, y_set = x_test, y_test
4. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() -
   1, stop = x_set[:, 0].max() + 1, step  =0.01),
5. nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1
   .shape),
7. alpha = 0.75, cmap = ListedColormap(('purple','green' )))
8. mtp.xlim(x1.min(), x1.max())
9. mtp.ylim(x2.min(), x2.max())
10. for i, j in enumerate(nm.unique(y_set)):
11. mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
12. c = ListedColormap(('purple', 'green'))(i), label = j)
13. mtp.title('Logistic Regression (Test set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()
```

Output:



The above graph shows the test set result. As we can see, the graph is divided into two regions (Purple and Green). And Green observations are in the green region, and Purple observations are in the purple region. So we can say it is a good prediction and model. Some of the green and purple data points are in different regions, which can be ignored as we have already calculated this error using the confusion matrix (11 Incorrect output).

Hence our model is pretty good and ready to make new predictions for this classification problem.

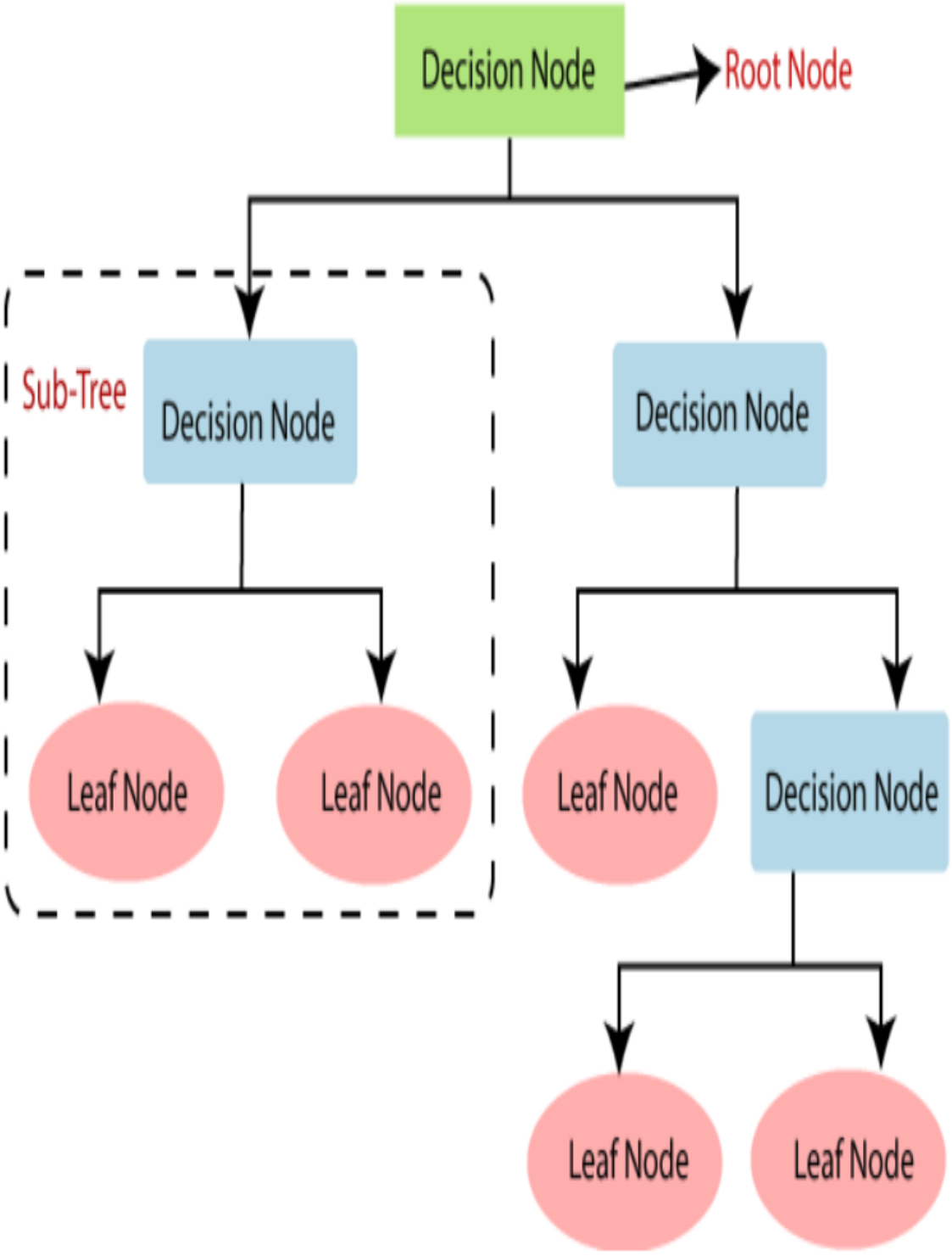
Decision Tree Classification Algorithm

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

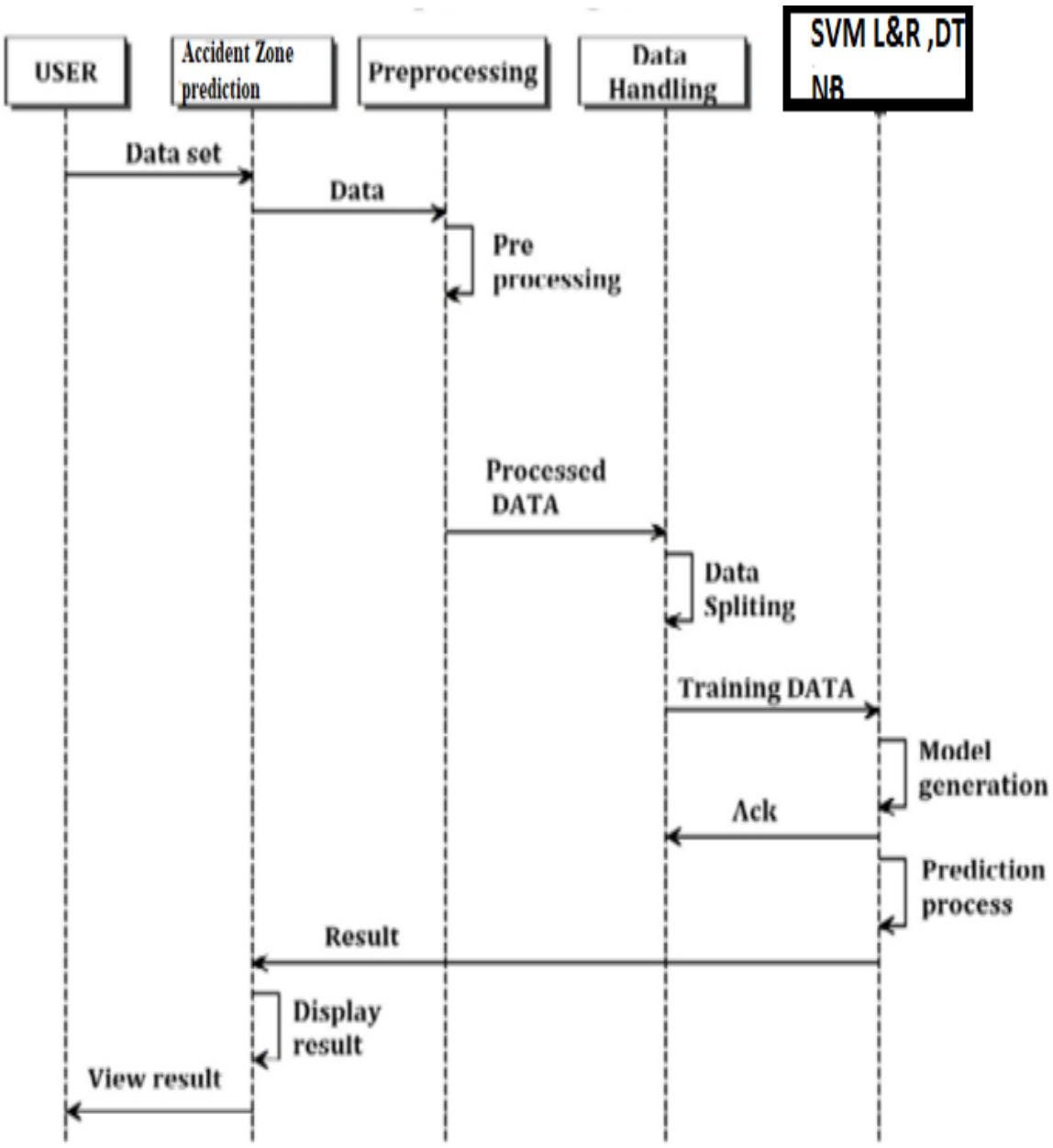
In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node**.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions**.**
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm**,** which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

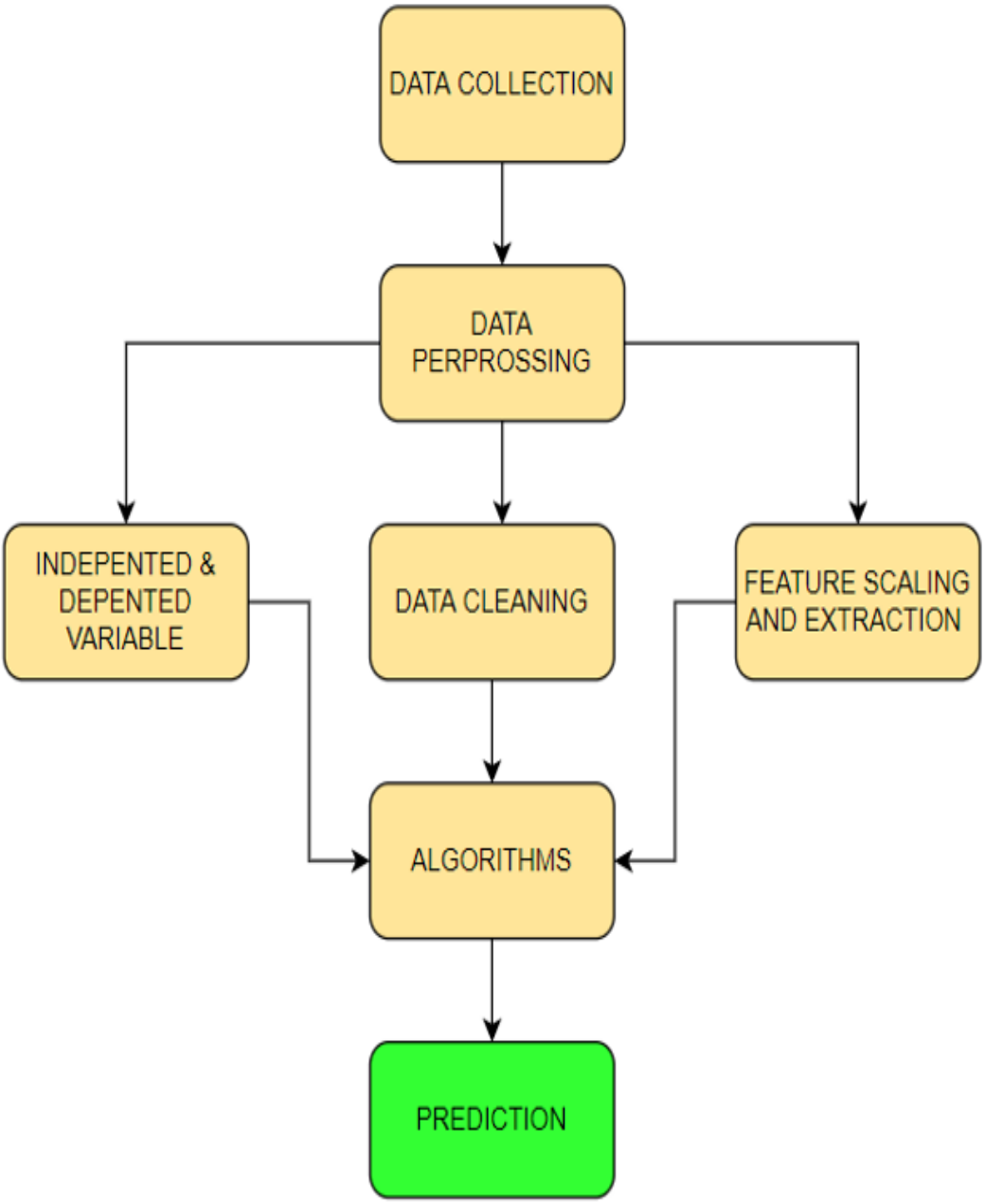
Below diagram explains the general structure of a decision tree:


Decision Node
Root Node
Sub-Tree
Decision Node
Decision Node
Leaf Node
Leaf Node
Leaf Node
Decision Node
Leaf Node
Leaf Node

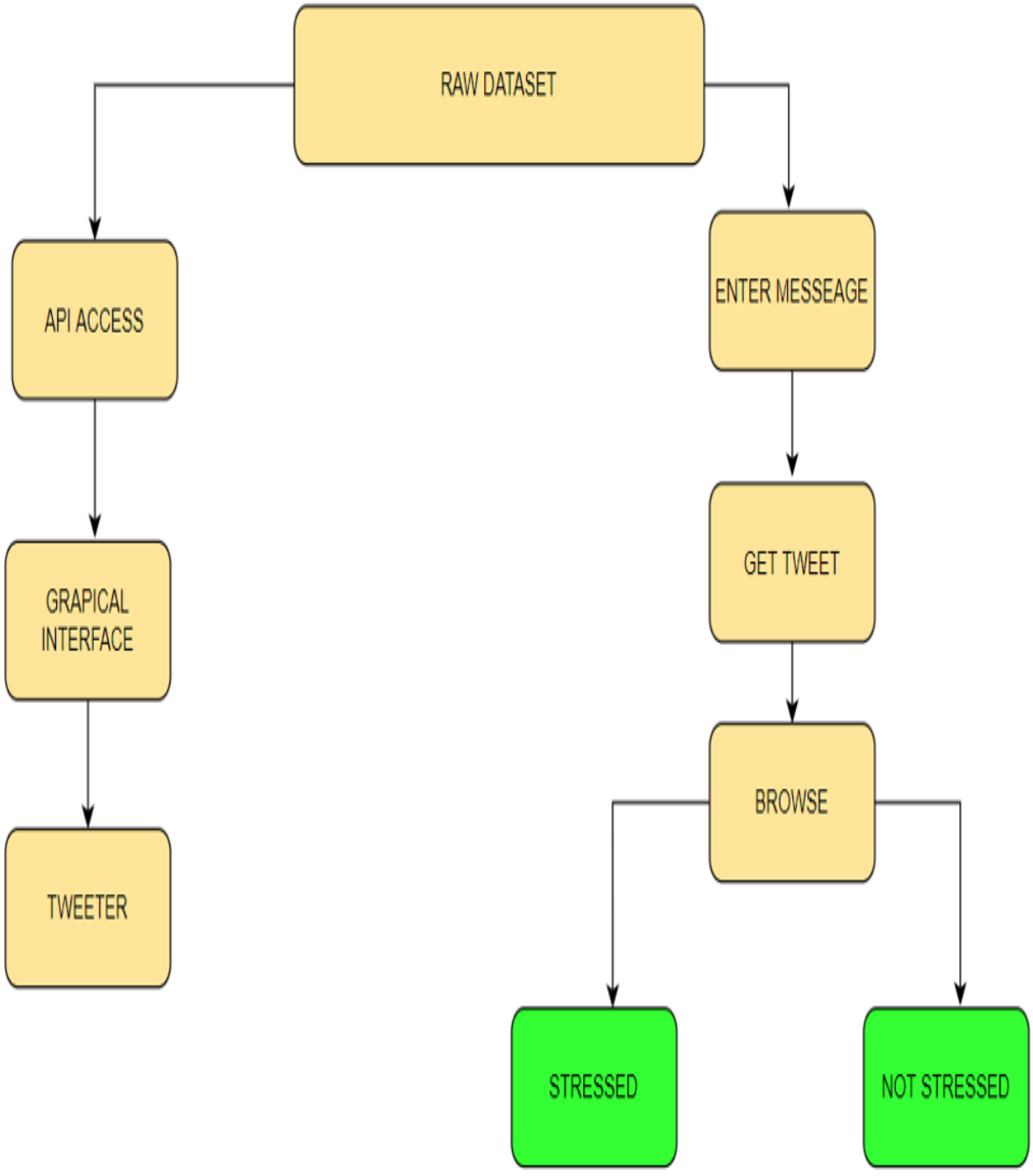
Sequence Diagram



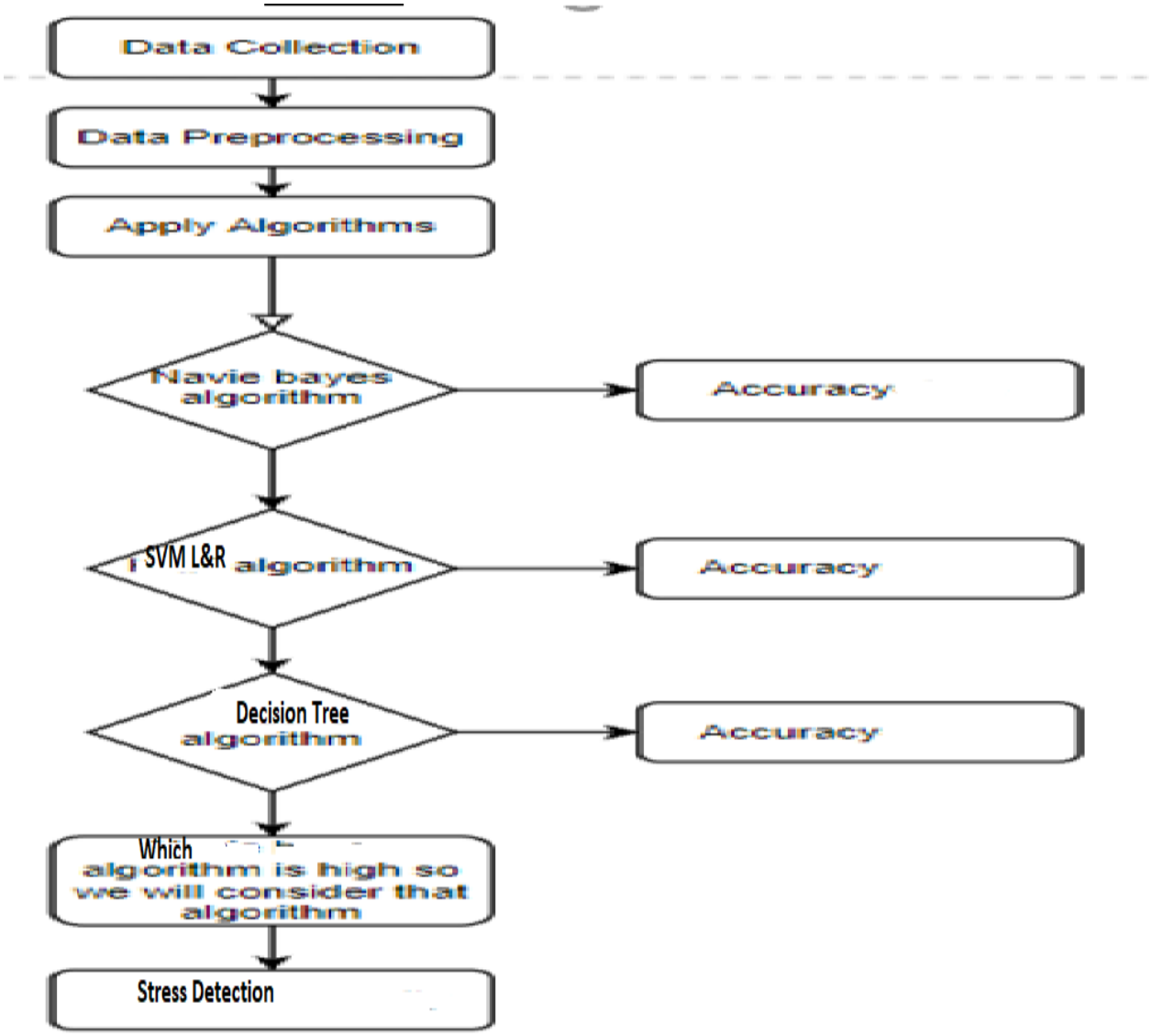
Data Flow Diagram

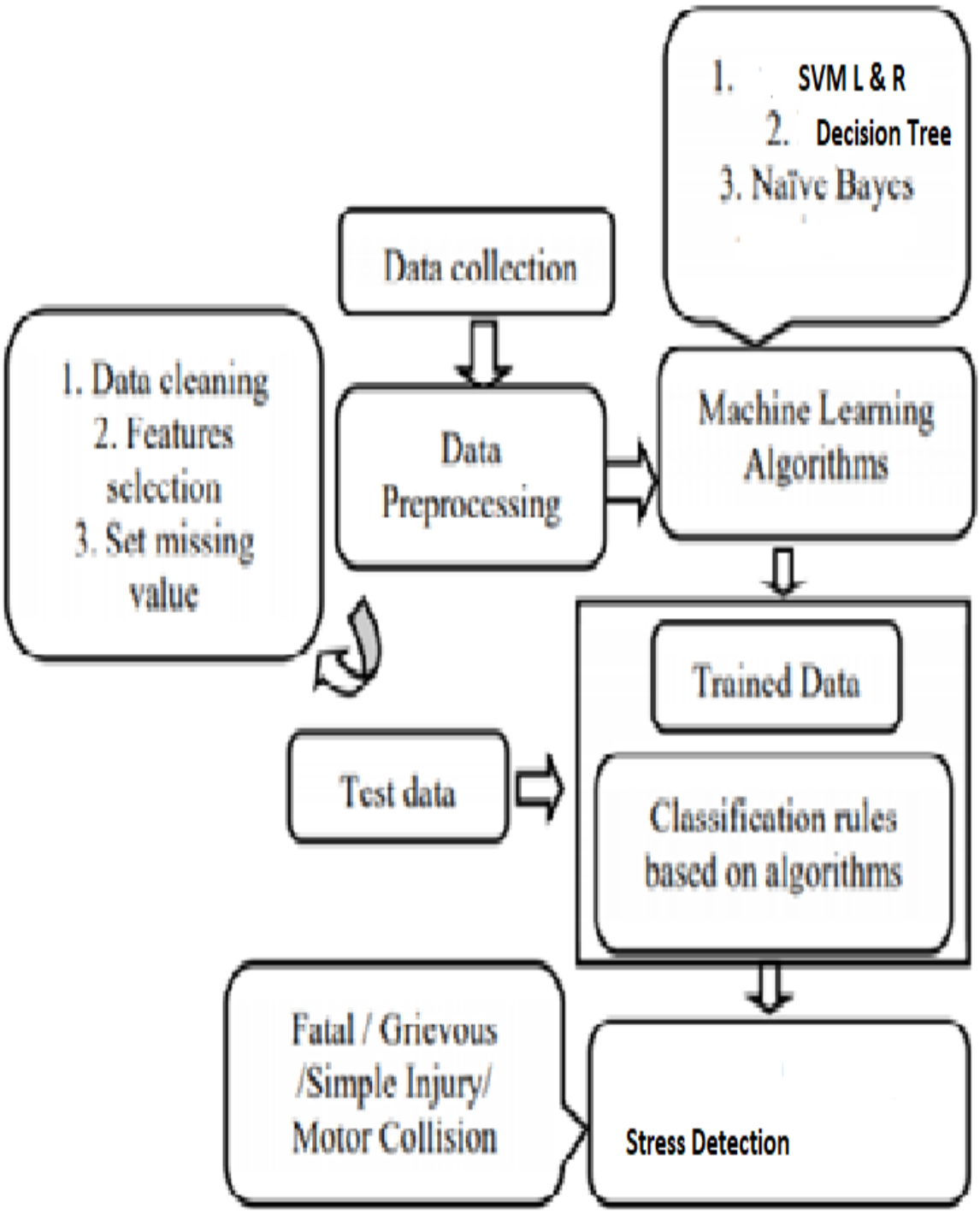


System Diagram

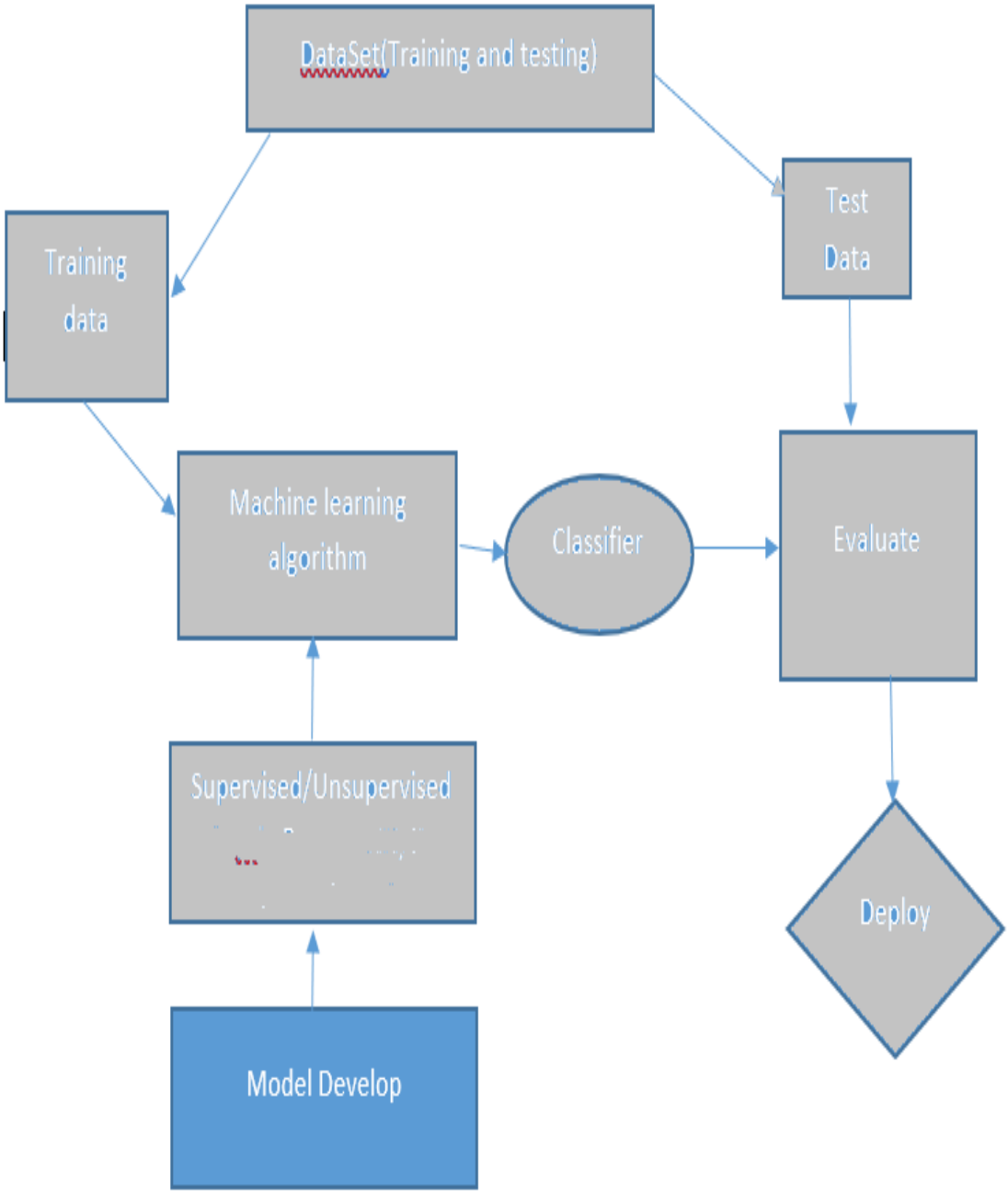


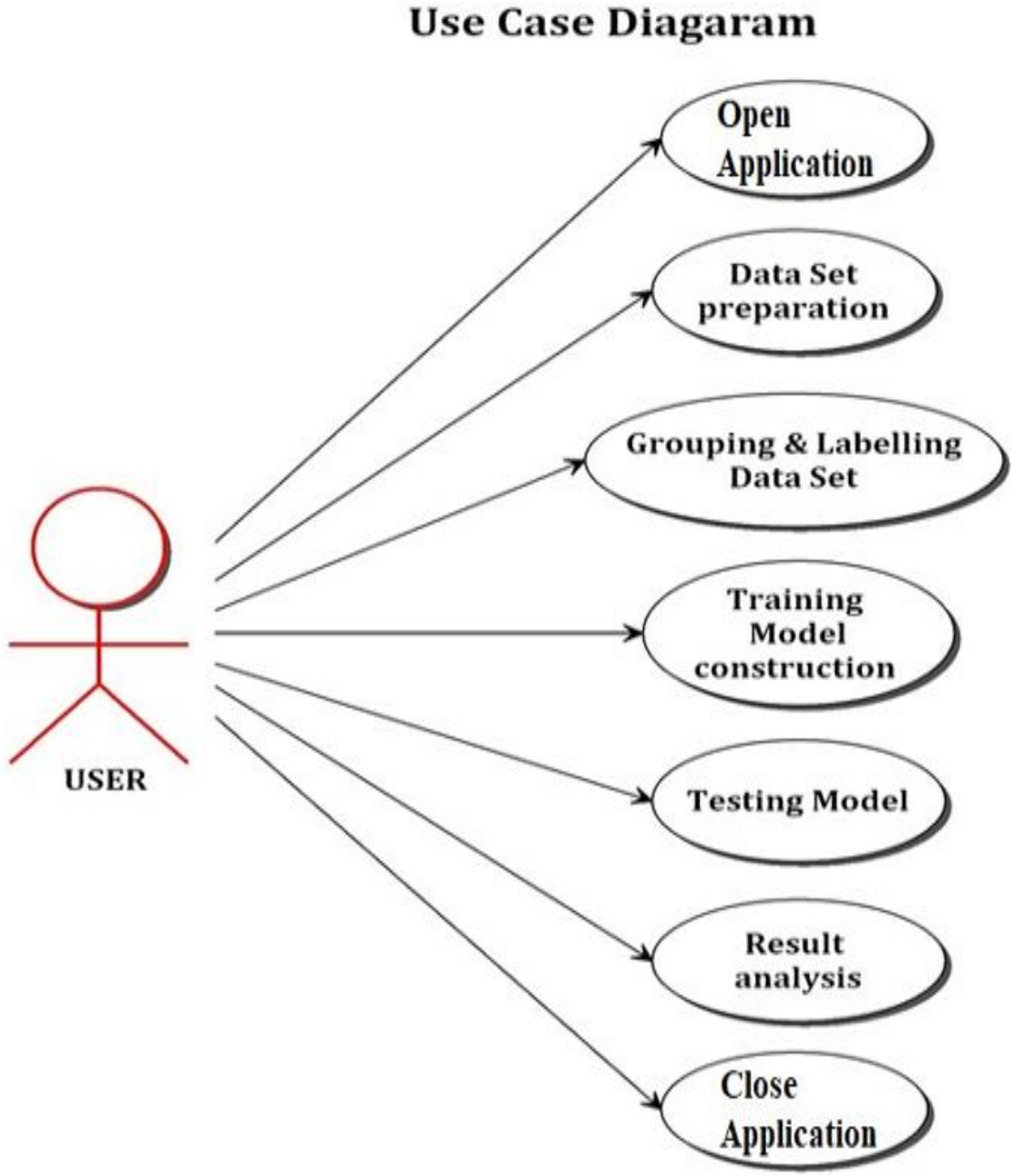
Flow Chart


Data Collection
Data Preprocessing
Apply Algorithms
Navie bayes algorithm
Accuracy
SVM L&R algorithm
Accuracy
Decision Tree algorithm
Accuracy
Which algorithm is high so we will consider that algorithm
Stress Detection


1. SVM L & R
2. Decision Tree
3. Naïve Bayes
Data collection
1. Data cleaning
2. Features selection
3. Set missing value
Data Preprocessing
Machine Learning Algorithms
Trained Data
Test data
Classification rules based on algorithms
Fatal / Grievous /Simple Injury/ Motor Collision
Stress Detection

Training and deploying a classifier:




Use Case Diagaram
Open Application
Data Set preparation
Grouping & Labelling Data Set
Training Model construction
Testing Model
Result analysis
Close Application
USER

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

All field entries must work properly.
Pages must be activated from the identified link.
The entry screen, messages and responses must not be delayed.

Features to be tested

Verify that the entries are of the correct format
No duplicate entries should be allowed
All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

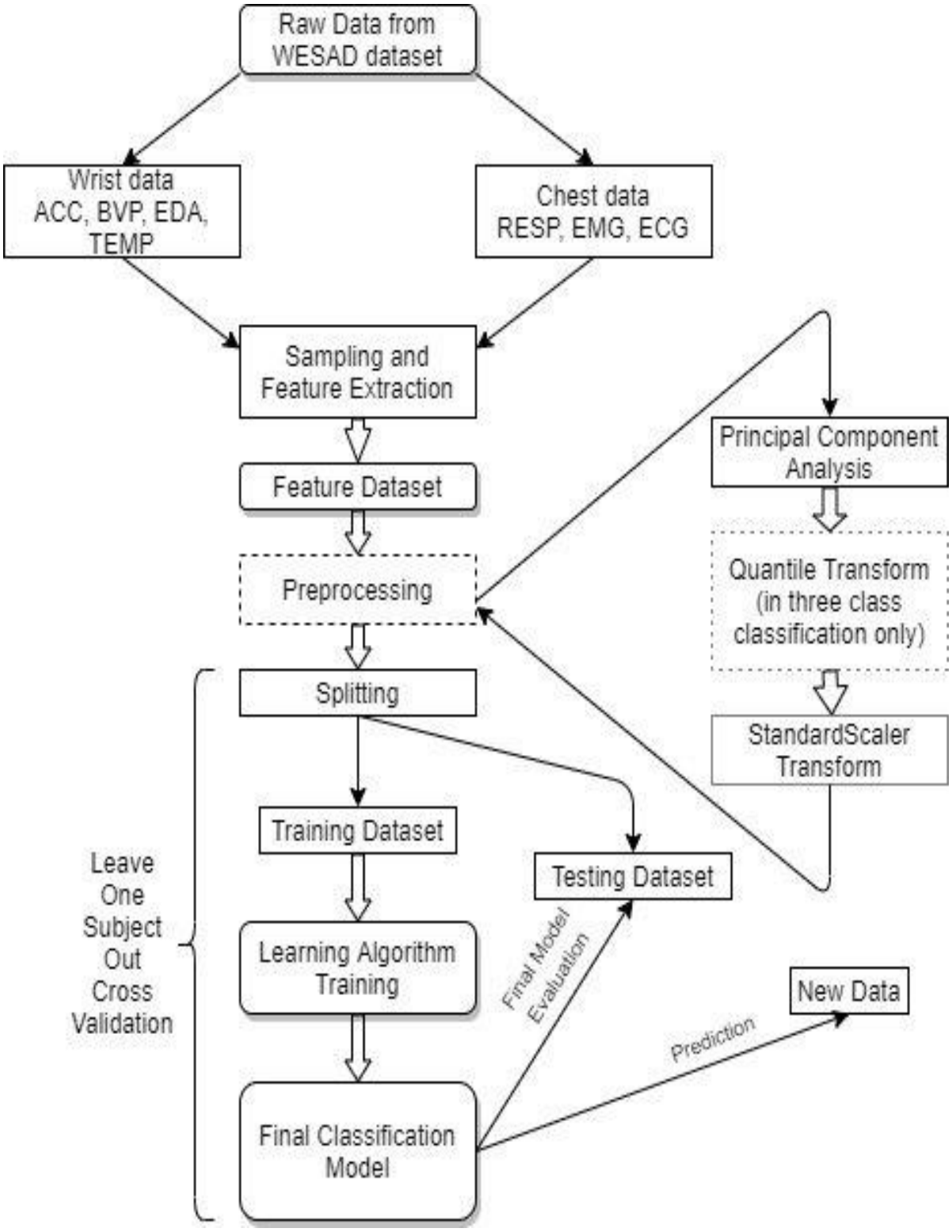
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

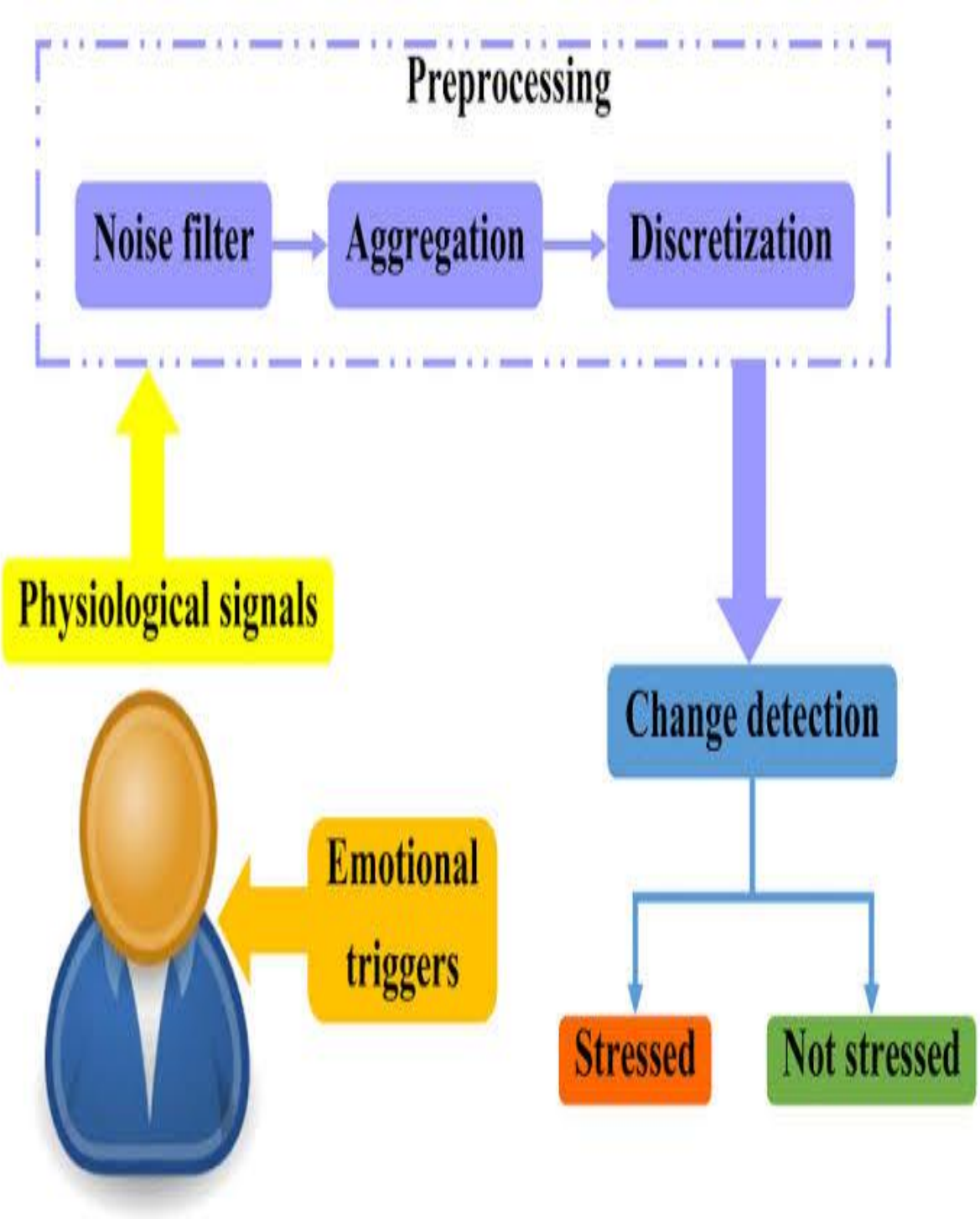
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

FLOW DIAGRAM



TEST CASE


Preprocessing
Noise filter
Aggregation
Discretization
Physiological signals
Change detection
Emotional triggers
Stressed
Not stressed

PROGRAM

```
#importing the required libraries


import pandas as pd


# this is used to plot the graph

import matplotlib.pyplot as plt

import seaborn as sns



#importing dataset to train and test

data = pd.read_csv('dev.csv')

data



#deleting the unwanted features

data = data.drop('qtext',axis=1)


data.head()


data.tail()


data.sample(10)
```

```
data.info()


len(data['label'].unique())


data['label'].unique()


data.dropna(inplace=True)


data.info()
data['label'] = data['label'].map({'sentiment': 3, 'empty': 5, 'sadness': 5,
                                   'enthusiasm': 3, 'neutral': 5,
                                   'worry': 0, 'surprise': 3,
                                   'love': 6, 'fun': 3, 'hate': 4,
                                   'happiness': 3, 'relief':1,
                                   'boredom': 5,'anger': 4})
data.dropna(inplace=True)


data.info()


# Identify dependent variable and independent variable
X = data.atext
Y = data.label


x
```

```
y


#converting string to numerical array
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
X = cv.fit_transform(X)


from sklearn.model_selection import train_test_split
X_train , X_test , Y_train , Y_test = train_test_split(X,Y,test_size=.2,random_state=100)


# data preprocessing completed
#exploring algorithems
#KNN
from sklearn.neighbors import KNeighborsClassifier
clf_WKNN = KNeighborsClassifier(n_neighbors=13,weights='distance')


clf_WKNN.fit(X_train, Y_train)
y_pred_WKNN = clf_WKNN.predict(X_test)


from sklearn.metrics import accuracy_score
print(accuracy_score(y_pred_WKNN,Y_test))
score_wknn = accuracy_score(y_pred_WKNN,Y_test)*100


Decision Tree
```

```
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score

from sklearn import tree


clf_entropy =
DecisionTreeClassifier(criterion="entropy",random_state=100,max_depth=11,min_samples
_leaf=3)

clf_entropy.fit(X_train,Y_train)

Y_pred_en=clf_entropy.predict(X_test)

Y_pred_en


print('Accuracy is',accuracy_score(Y_test,Y_pred_en)*100)

score_dt = accuracy_score(Y_test,Y_pred_en)*100

from sklearn.metrics import confusion_matrix

print(confusion_matrix(Y_test, Y_pred_en))
```

Navie Bayes

```
import sklearn

from sklearn.naive_bayes import BernoulliNB

from sklearn import metrics

from sklearn.metrics import accuracy_score
```

```
BernNB = BernoulliNB(binarize=.1)

BernNB.fit(X_train,Y_train)

print(BernNB)


Y_expect = Y_test

Y_pred = BernNB.predict(X_test)

print(accuracy_score(Y_expect,Y_pred)*100)

score_nb = accuracy_score(Y_expect,Y_pred)*100



from sklearn.metrics import confusion_matrix

print(confusion_matrix(Y_test, Y_pred))

from sklearn.linear_model import LogisticRegression


# instantiate the model (using the default parameters)

logreg = LogisticRegression()


# fit the model with data

logreg.fit(X_train,Y_train)



y_pred=logreg.predict(X_test)

score_lr = accuracy_score(Y_test,y_pred)*100

score_lr
```

```
# Random forest

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(max_depth=2, random_state=0)

rf.fit(X_train, Y_train)


y_pred = rf.predict(X_test)


score_rf = accuracy_score(Y_test,y_pred)*100

score_rf


scores = [score_dt,score_nb,score_lr,score_wknn,score_rf]

algorithms = ["DecisionTreeClassifier","naive_bayes","Logistic Regression","KNN","Random
forest"]


for i in range(len(algorithms)):

    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")


sns.set(rc={'figure.figsize':(15,8)})

plt.xlabel("Algorithms")

plt.ylabel("Accuracy score")


sns.barplot(algorithms,scores)
```

```
data = input('Enter your Message:')

vect = cv.transform([data]).toarray()

my_prediction = logreg.predict(vect)

print('my_prediction',my_prediction[0])


import tweepy

import pandas as pd

import csv

from tkinter import *

from tkinter import messagebox

from tkinter.filedialog import askopenfilename

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from PIL import ImageTk,Image

import seaborn as sns

root = Tk()

title_name = 'Psychological stress prediction'

root.title(title_name)

# bg_image = PhotoImage(file ="agri1 (2).ppm")

# x = Label (image = bg_image)

# x.grid(row = 0, column = 0)

root.geometry('1050x650')
```

```
#root.configure(background="white")


var = StringVar()
label = Label( root, textvariable = var,font=('arial',20,'bold'),bd=20,bg="Steel Blue2")
var.set(title_name)
label.grid(row=0,columnspan=6,column=3)


def rule_file():
   root1=Tk()
   root1.title("login page")
   root1.geometry('600x500')
   #root1.configure(background="SkyBlue2")
   def login():
      user = E.get()
      password = E1.get()
      admin_login(user,password)
   L=Label(root1, text =
"Username",bd=8,background="aliceblue",height=1,padx=16,pady=16,font=('arial',16,'bold')
,width=10,).grid(row = 0,column=0)
   E=Entry(root1)
   E.grid(row = 0, column = 1)
   L1=Label(root1, text =
"Password",bd=8,background="aliceblue",height=1,padx=16,pady=16,font=('arial',16,'bold'),
width=10,).grid(row = 1,column=0)
```

```
    E1=Entry(root1,show="*")
    E1.grid(row = 1, column = 1)

B1=Button(root1,text="Login",width=4,height=1,command=login,bd=8,background="lawngr
een")
    B1.grid(row = 2, column = 1)
    root1.mainloop()

def admin_login(user,password):
    if user == "admin" and password == "admin":
        plot_graph()



def plot_graph():
    root10 = Tk()
    root10.title('GRAPHS')
    root10.geometry('900x800')
    #root10.configure(background="moonwhite")

    #Get your Twitter API credentials and enter them here
    consumer_key = "mDDxXp6AD7nDlW1iccfu4QCBy"
    consumer_secret = "5ejGS33wUtRej3ClYWniw9ztK7rAbsaUjO0nsMTtrNPcTZPZjK"
    access_key = "3220527972-lEs0rHzBEUPcTiNxcl4M06V1BUrkYEqyxVx87FA"
    access_secret = "VLUa4SHApjz7DYXPUgfsr75wM4wqZ8CmhcjZNoKb0rqPU"
```

```
def get_tweets():
    #global E1
    username = E1.get()
    #http://tweepy.readthedocs.org/en/v3.1.0/getting_started.html#api
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key, access_secret)
    api = tweepy.API(auth)
    #set count to however many tweets you want
    number_of_tweets = 100
    #get tweets
    tweets_for_csv = []
    # Open/Create a file to append data
    csvFile = open(username[1:]+'.csv', 'a')
    #Use csv Writer
    csvWriter = csv.writer(csvFile)

    for tweet in tweepy.Cursor(api.search,q=username,count=100,lang="en",since="2017-
04-03").items(number_of_tweets):
        print (tweet.created_at, tweet.text)
        csvWriter.writerow([tweet.created_at, tweet.text.encode('utf-8')])

def OpenFile_train():
    global data,data1,name
```

```
    name =
askopenfilename(initialdir="C:/Users/Batman/Documents/Programming/tkinter/",
                    filetypes =(("csv File", "*.csv"),("All Files","*.*")),
                    title = "Choose a file.")
    try:
        with open(name,'r') as UseFile:
            data = UseFile.readlines()
            index = data[0].index('b')
            print(data[0][index+2:])
            e3.delete(0,END)
            e3.insert(0,data[0][index+2:])
    except FileNotFoundError:
        print("No file exists")


  def psychologicalstress_via_tweet():
    msg = E1.get()
    vect = cv.transform([msg]).toarray()
    my_prediction = logreg.predict (vect)
    print('my_prediction',my_prediction[0])
    if my_prediction[0] == 0:
        print("depressd")
        e4.delete(0,END)
        e4.insert(0,'depressd')
    else:
```

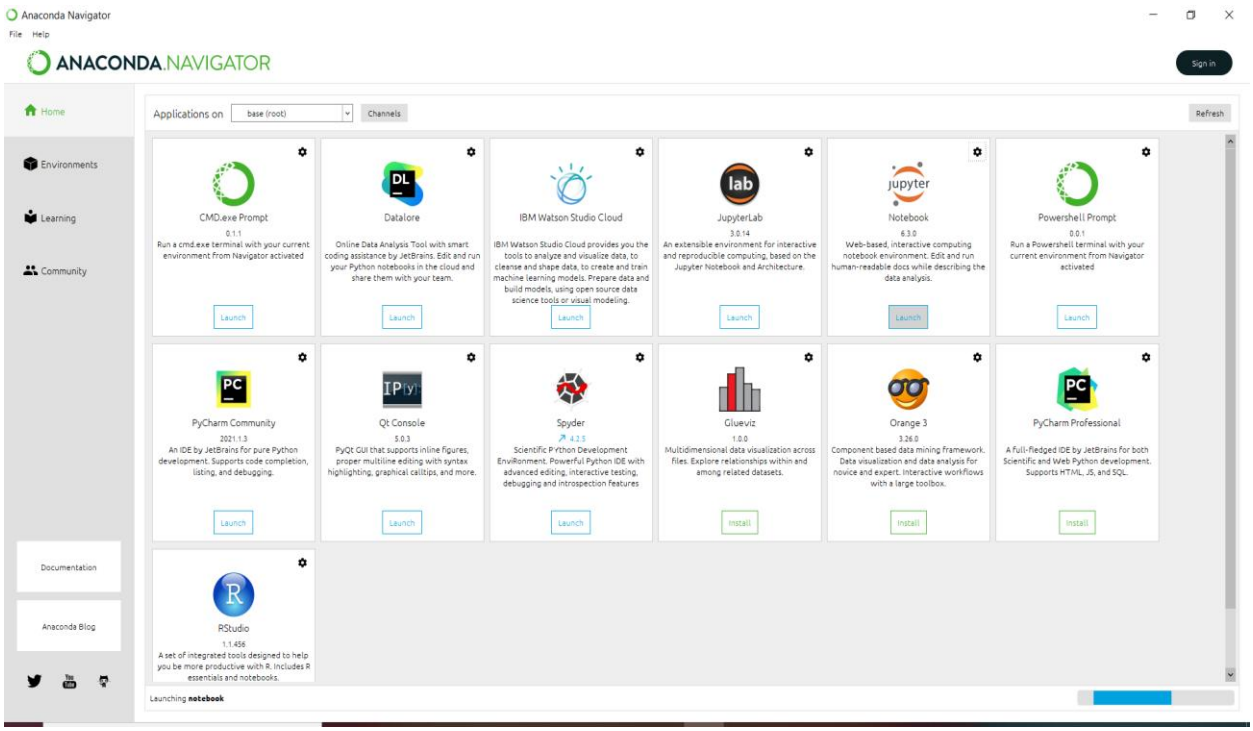
```
        print("Not depressed")
        e4.delete(0,END)
        e4.insert(0,'Not depressed')


def detect_psychologicalstress():
    msg = e3.get()
    vect = cv.transform([msg]).toarray()
    my_prediction = logreg.predict(vect)
    print('my_prediction',my_prediction[0])

    if my_prediction[0] == 0:
        print("depressd")
        e4.delete(0,END)
        e4.insert(0,'depressd')
    elif my_prediction[0] == 2:
        print("Blocked")
        e4.delete(0,END)
        e4.insert(0,'Blocked')
    elif my_prediction[0] == 3:
        print("Happy")
        e4.delete(0,END)
        e4.insert(0,'Happy')
```

```
    elif my_prediction[0] == 4:
        print("Hate")
        e4.delete(0,END)
        e4.insert(0,'Hate')
    elif my_prediction[0] == 5:
        print("Sad")
        e4.delete(0,END)
        e4.insert(0,'Sad')

    elif my_prediction[0] == 6:
        print("Love")
        e4.delete(0,END)
        e4.insert(0,'Love')
    else:
        print("Not depressed")
        e4.delete(0,END)
        e4.insert(0,'Not depressed')



  L  = Label(root10, text = "Enter
Name",bd=8,height=1,padx=16,pady=16,font=('arial',16,'bold'),width=10,)
  L.grid(row = 1,column=0)

  E1  = Entry(root10)
```

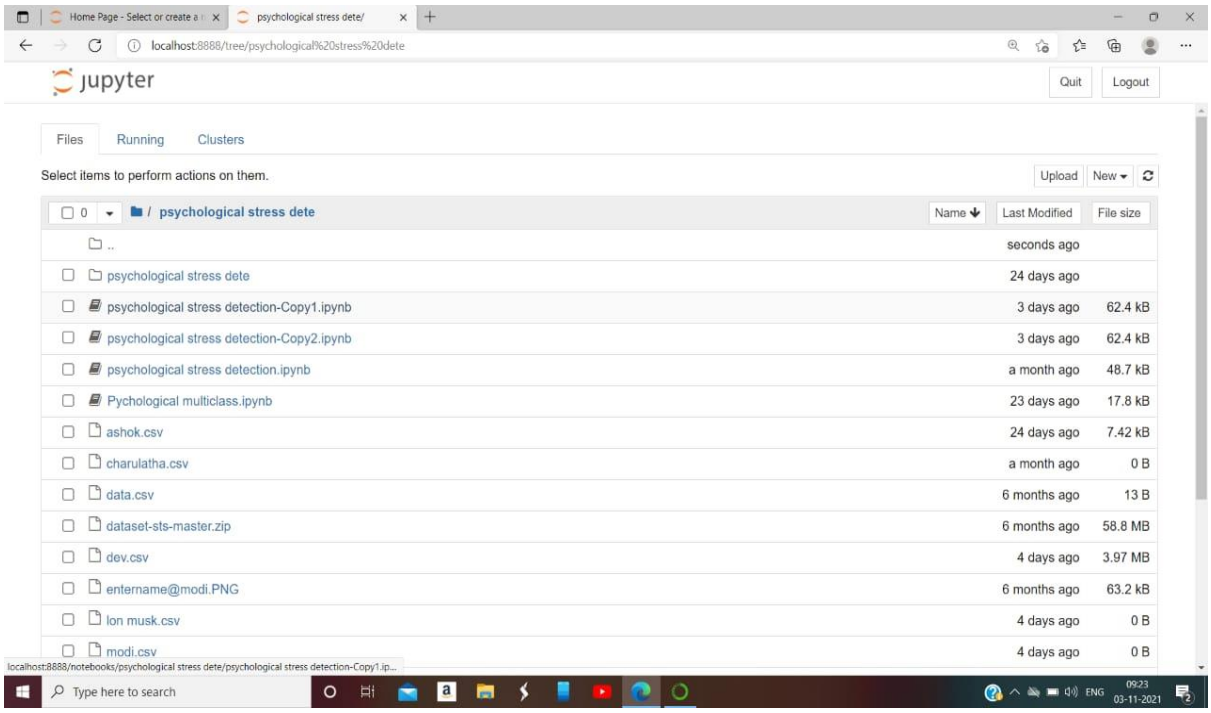
```
    E1.grid(row=1,column=5)


    B1 = Button(root10, text = "get
tweets",height=1,padx=16,pady=16,bd=8,font=('arial',16,'bold'),width=10,bg="white",comm
and = get_tweets)
    B1.grid(row=2,column=3)


    B2 = Button(root10, text = "Browse
file",height=1,padx=16,pady=16,bd=8,font=('arial',16,'bold'),width=10,bg="white",command
= OpenFile_train)
    B2.grid(row=2,column=4)


    B3 = Button(root10, text = "Tweet
Predict",height=1,padx=16,pady=16,bd=8,bg="SteelBlue1",font=('arial',16,'bold'),\
            width=10,command = psychologicalstress_via_tweet)
    B3.grid(row=2,column=5)



    L2  = Label(root10, text = "Entry
Message",bd=8,height=1,padx=16,pady=16,font=('arial',16,'bold'),width=10,).grid(row =
3,column=0)


    e3 = Entry(root10)
    e3.grid(row=3,column=3)
```

```
   B2 = Button(root10, text = "Text
predict",height=1,padx=16,pady=16,bd=8,font=('arial',16,'bold'),\
            width=10,bg="lemon chiffon",command = detect_psychologicalstress)
   B2.grid(row=4,column=3)

   e4 = Entry(root10)
   e4.grid(row=4,column=4)

   root10.mainloop()

# B3= Button(root, text = "Plot
psychological_stress",height=1,padx=16,pady=16,bd=8,font=('arial',16,'bold'),width=10,bg="
white",command = plot_psychologicalstress)
# B3.grid(row=5,column=3)

B_rule = Button(root, text = "User",height=1,padx=16,pady=16,bd=8,font=('arial',16,'bold'),\
            width=10,command=rule_file)
B_rule.grid(row=4,column=1)

output = Entry(root)

root.mainloop()

!pip install tweepy
```

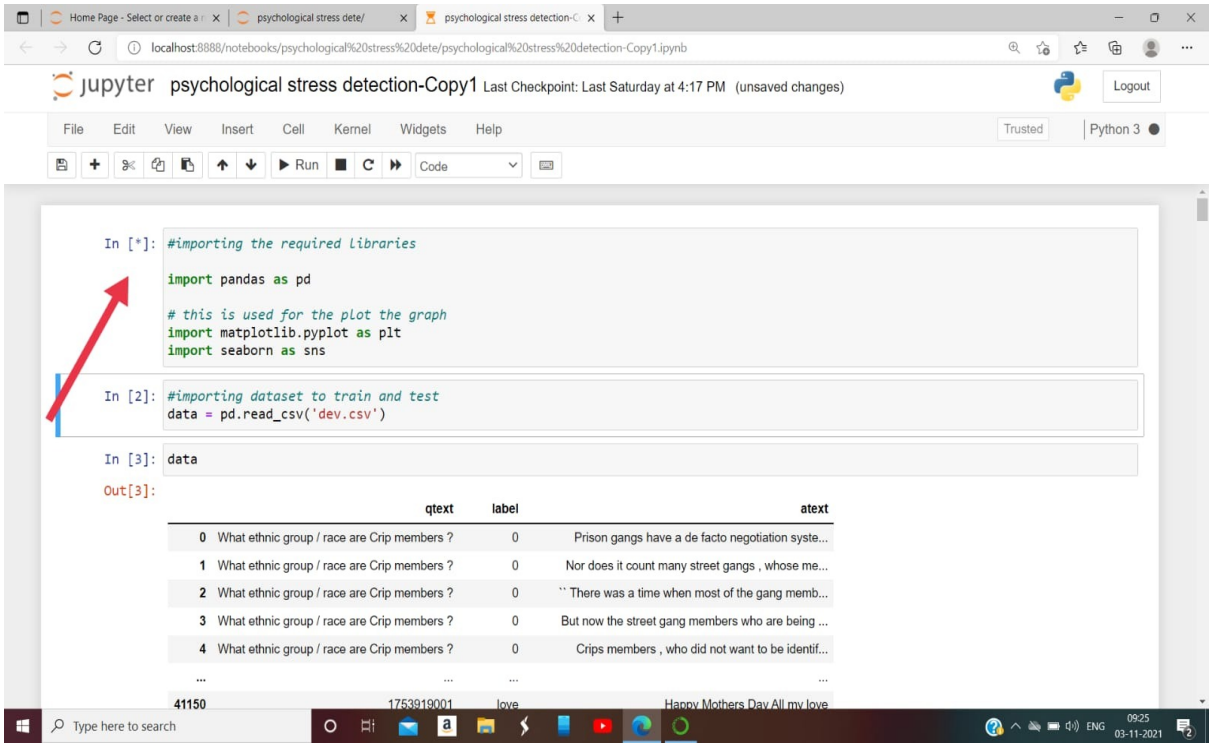
OUT PUT



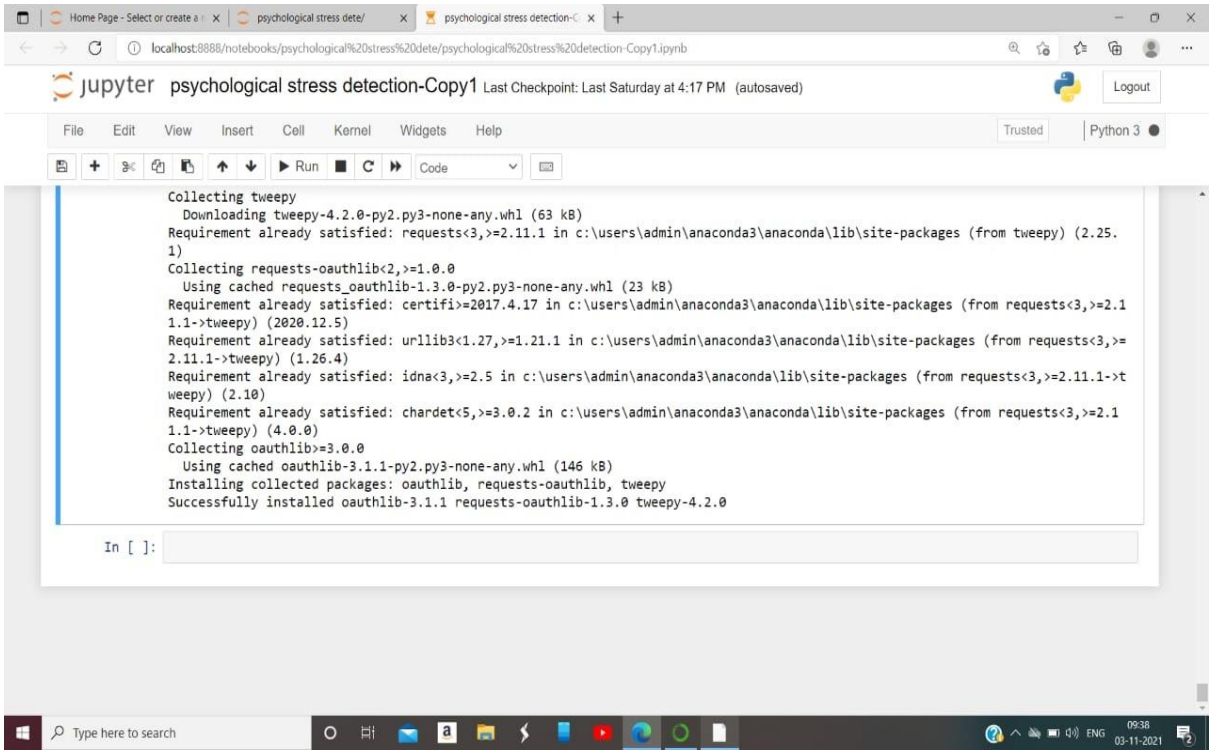
First , open anaconda navigator ,click on Jupiter notebook.



Jupiter notebook page is displayed, here click on psychological stress detection, Jupiter notebook will run the code .

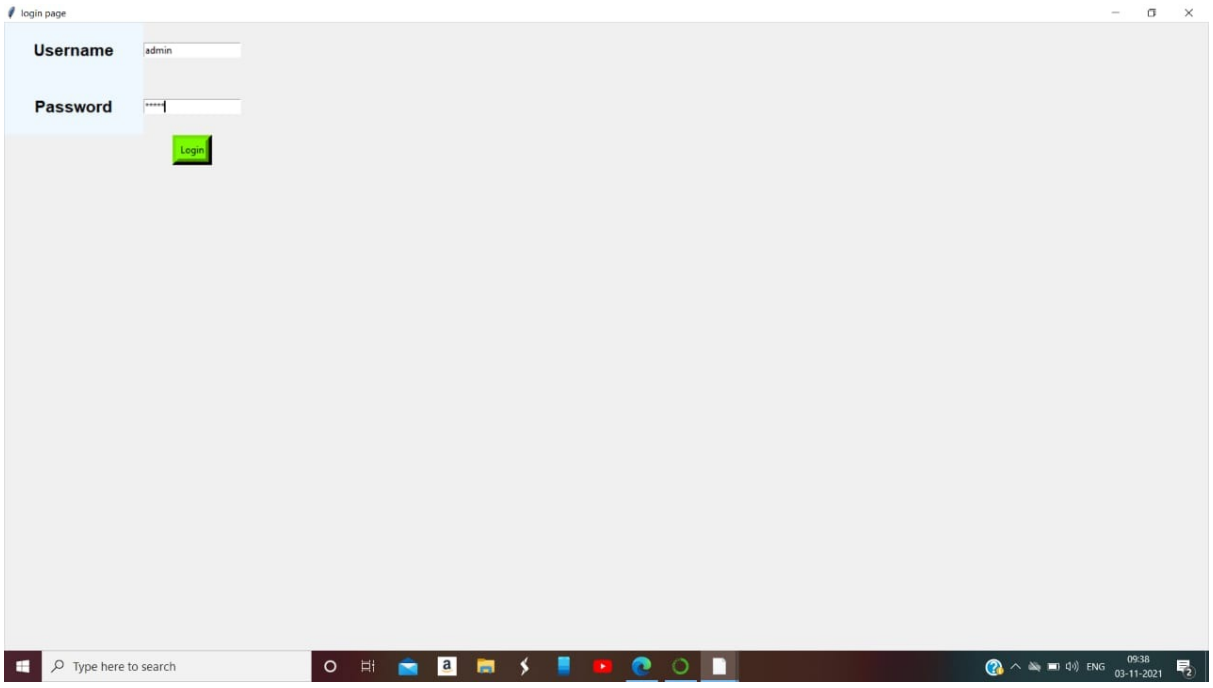


Run the code, the star button is replaced by a number .,run the entire code till all the star buttons are replaced by numbers ..this completes the process ..

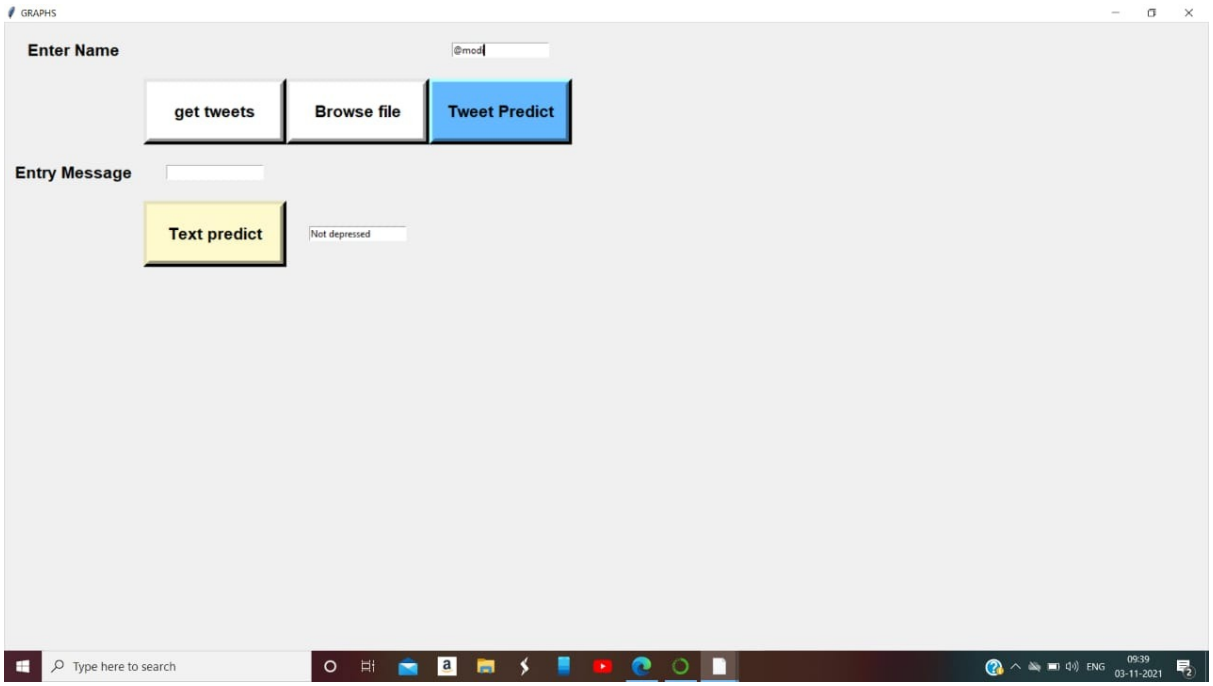


The document sheet is displayed .

The login screen is displayed as shown, click on user



Enter username and password, And click on login.

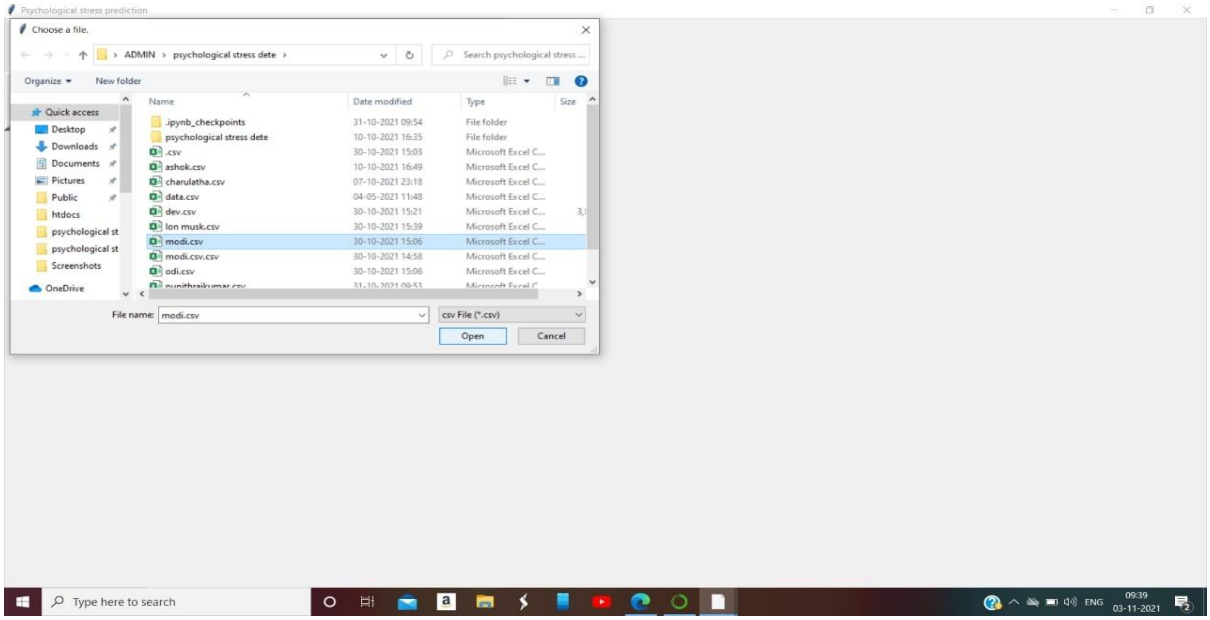


Enter the tweet name.

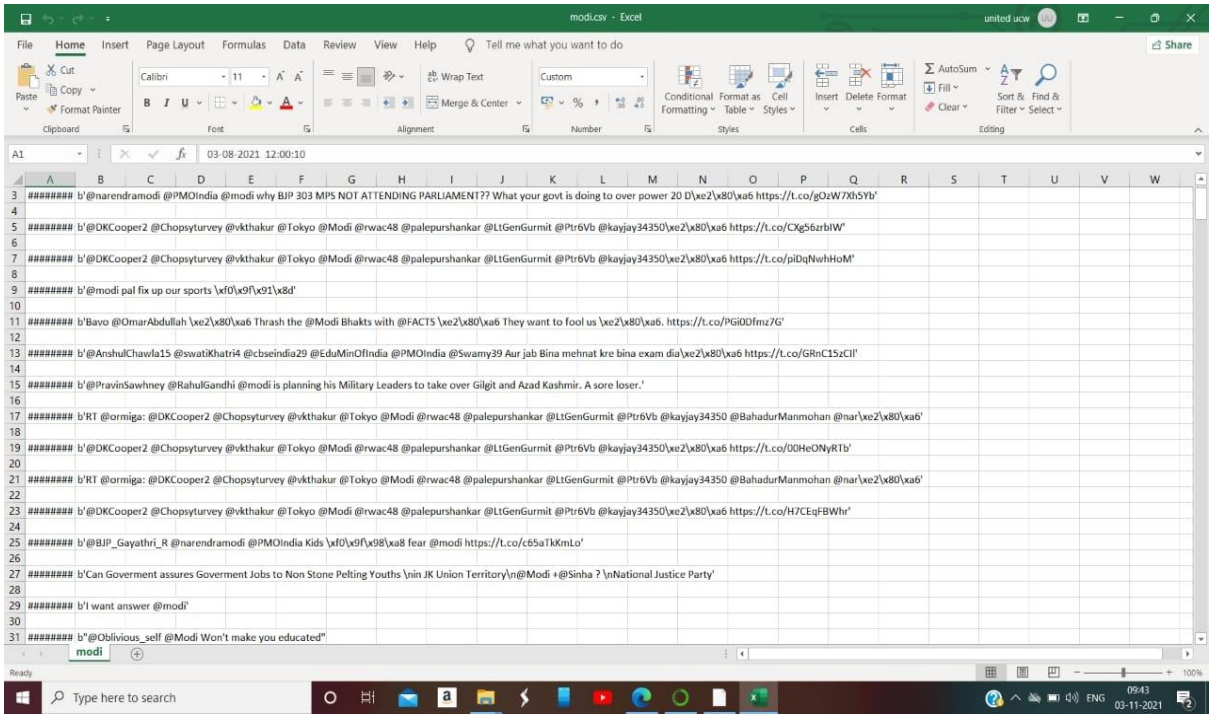
Click on browse file .

Click on get tweets .

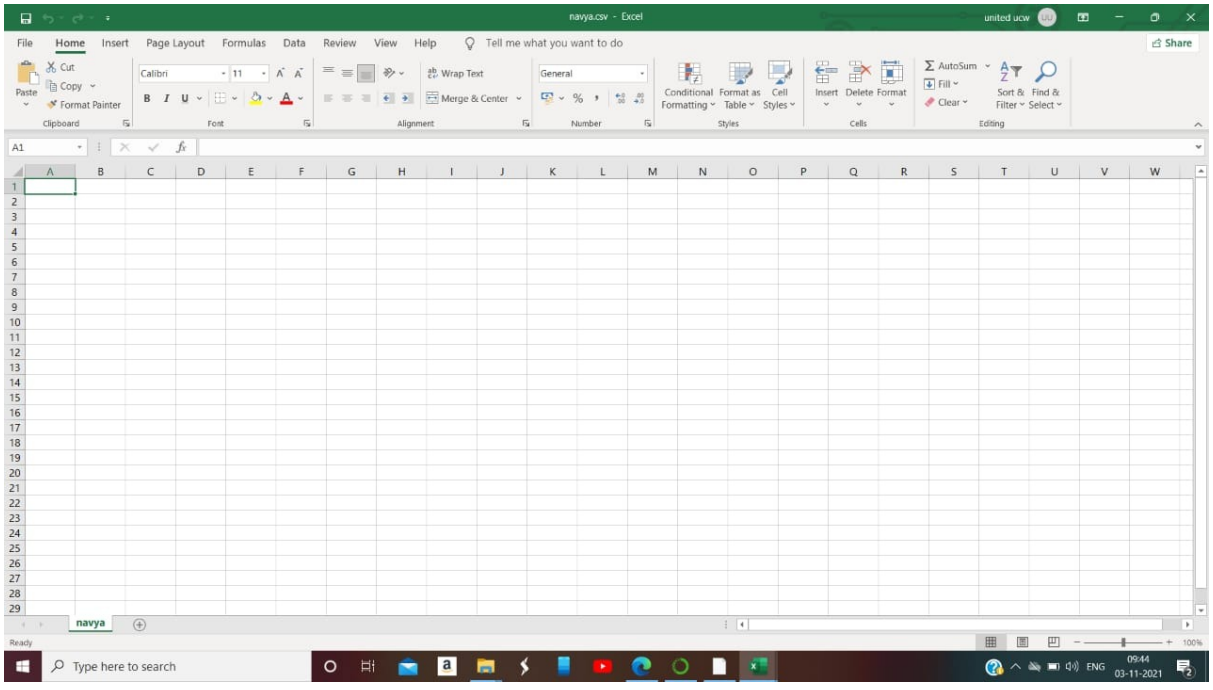
The file is browsed and is predicted if it is depressed or not depressed



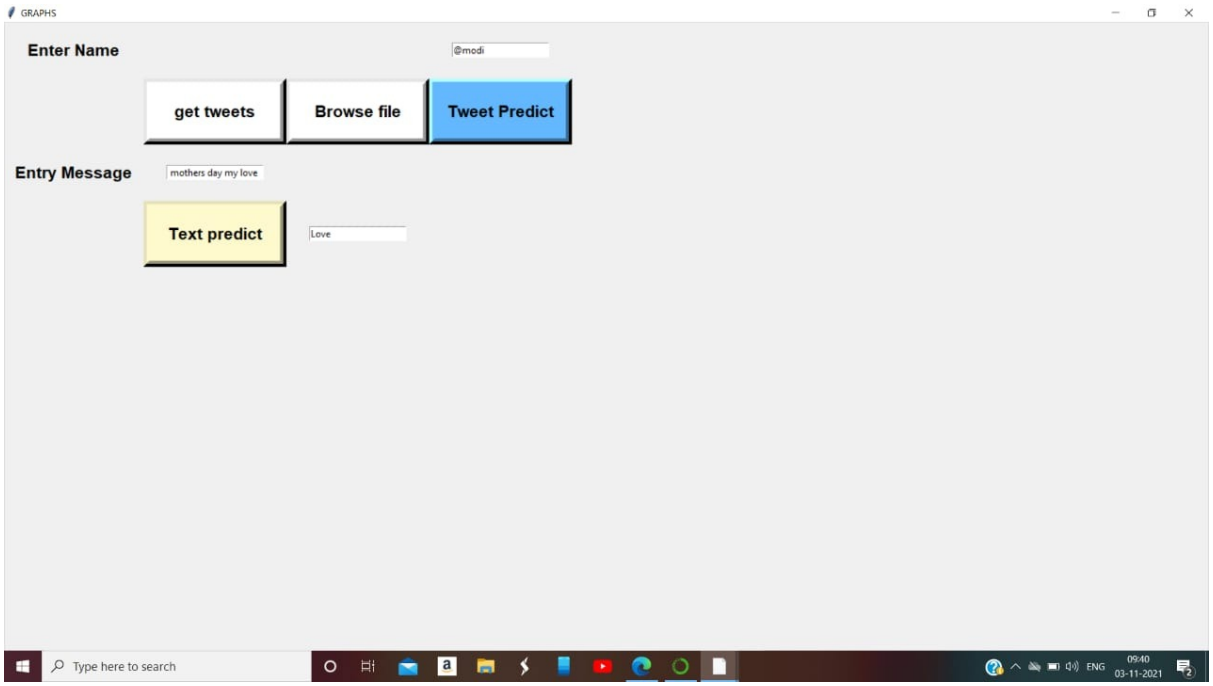
As followed by the above step if the tweeted name is in the xl sheet, CSV file is created, cl;ick on file.



CSV file of the tweed name is displayed and it prediction is done if it is depressed or not depressed.



If no person has tweeted by that name, in xl sheet the data will not be displayed.



Enter the text message in the box ...it predicts the emotion if its is love , happy , sad , angry etc…,

RESULTS

In the road accident prediction project use the dataset is in terms of values and some data is plain English word so, the numerical values data is easily predicted and also the calculation are easily done but, the normal word are display as it is or the non predicted data are drop in the table. So, This dataset are many columns and rows and all numbers of null values will be fulfill in forward fill method and also use the classification algorithm entire dataset.

CONCLUSION

Tweeter can be best used to detect stress level. Current existing work identifies the stressed person and gives the output depending on the parameters and user input. By adding more parameters we can also identify some of the status existing in person, the rise or decrease in any of the parameter may help us to detect whether the person is only in stress or is facing any sort of health issue too. Comparing various algorithm but depending on the accuracy only fast classification algorithm is used.

REFERENCES

[1] Huijie Lin, Jia Jia, Jiezhon Qiu, Yongfeng Zhang, Lexing Xie, Jie Tang, Ling Feng, and Tat-Seng Chua, "Detecting Stress Based on Social Interactions inSocial Networks", IEEE Transactions on Knowledge and Data Engineering, 2017.

[2] Aasia Manzoor, Hadia Awan & Sabita Mariam,- INVESTIGATING THE IMPACT OF WORK STRESS ON JOB PERFORMANCE: A Study on Textile Sector of Faisalabad, Asian Journal of Business and Management Sciences ISSN: 2047-2528 Vol. 2 No. 1 [20-28]

[3] Dr. Jyotsna Codaty, - Key to Stress Free Living V&S publications,2013,New Delhi,pp14,15,45,46.

[4] P. Kavitha, Role of stress among women employees forming majority workforce at IT sector in Chennai and Coimbatore, Tier-I & Tier-II centers, SONA- Global Management Review , Volume 6, Issues 3, May 2012.

[5] Kayoko Urakawa and Kazuhito Yokoyam, ―Sense of Coherence (SOC) may reduce the Effects of Occupational Stress on Mental Health Status among Japanese Factory Workers, Journal ofIndustrial Health, Vol. 47 , No. 5 pp.503- 508

[6] Amir Shani and Abraham Pizam, ―Work-Related Depression among Hotel Employees, Cornell Hospitality Quarterly, Vol. 50, No. 4, 446-459 (2009)

[7] Li-fang Zhang, -Occupational stress teaching approach among Chinese academics, Educational Psychology, Volume 29, Issue 2, March 2009, pages 203 – 219.

[8] Andrey Bogomolov, Bruno Lepri, Michela Ferron, Fabio Pianesi,and Alex Pentland. Daily stress recognition from mobile phone data,weather.

[9] Chris Buckley and EllenM Voorhees. Retrieval evaluation with incomplete information. In Proceedings of the 27th annual international ACM SIGIR conference on and development in informationretrieval, pages 25–32, 2004.

[10] Xiaojun Chang, Yi Yang, Alexander G Hauptmann, Eric P Xing,and Yao-Liang Yu. Semantic concept discovery for largescale zero-shot event detection. In Proceedings of International Joint Conference on Artificial Intelligence, pages 2234–2240, 2015.

[11] Wanxiang Che, Zhenghua Li, and Ting Liu. Ltp: A Chinese language technology platform. In Proceedings of International Conferenceon Computational Linguistics, pages 13–16, 2010.