**Analysis of Social Issues (Suicide, Road Accidents, Murder)**

****

**Group Members**

|  |  |
| --- | --- |
| Kshitiz Kumar | 16BCE0597 |
| Om Ashish Mishra | 16BCE0789 |
| Shivam Kalra | 16BCE0911 |

**Report submitted for the**

**Final Project Review of**

**Course Code: CSE3021 – Social and Information Networks**

**Slot: A2 + TA2**

**Professor: Dr. W.B.Vasantha**

**1. Abstract:**

The group will make a detailed analysis on social issues like deaths due to road accidents, suicide and murder. Datasets along with several attributes have been selected for the analysis.

The group is making this project with an aim to study the cause and intensity of the deaths and determine through various algorithms which age group faces the maximum risk of deaths (an estimate).

The data that we collected are from various sources in the internet and these are commonly shared by various social media platforms like Twitter, Facebook, etc. The data consists of the well-defined boundaries like gender, timings etc that will help us to do proper analysis of data.

**Keywords:** Datasets, attributes(age,gender,timings) algorithms, death.

**2.** **Introduction**

The today’s world is full of uncertainties, as per as social issues and crimes are concerned. Therefore it is our duty to have a look at the data in the past and analysis the differences now. The social issues that are going on, we keep receiving through multimedia and newspaper. The whole world is getting addicted to social media and is getting mentally ill. Thus keeping that into mind this project’s analysis is very effective.

The basic idea behind the project is to study the criminal and social rates of India. In the project, we have three datasets (Suicides in India, Victims\_Muder Records, Road\_Accidents) and through machine learning algorithms we are going to predictions on the dataset. The analysis of the data for the social issues related to the dataset. The dataset contains the states of India and the Union Territories as the actors and the attributes or actions are being represented with the help of the gender, age, timings etc.

The project aims to have a check on the day-to-day social issue analysis as the crime rate is assumed to be increasing day by day. Every morning we wake up to find in newspaper that the some crime or the other has being taken place. This is a bad sign and we are trying to analysis it. We went through various articles, journals and social agendas discussed across the globe on Facebook, Twitter and other social networking mediums and we felt the need to have a current analysis on this topic and see to it that we are able to perform a good job in predicting or not with the help of statistical tools. This is done with the aim of making India a better place.

**3. Literature Review Summary Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Authors and Year (Reference)* | *Title (Study)* | *Methodology used/ Implementation* | *Dataset details/ Analysis* | *Relevant Finding* | *Limitations/ Future Research/ Gaps identified* |
| ***[1]. B.V.Sreekumar, Dr. V.Sreedevi. (2014)*** | ***Impact of Road Accidents in Kerala During 2001 to 2011 – A Case Study*** | *This study examines the analytical part of covering the table, percentages, Mean, Standard Deviation and Techniques used in Least Square Trend Analysis.* | *collecting secondary sources of data from 2001 to 2011 accordingly year wise and district wise in Kerala state, India* | *Road management is a main source for economic responsibility. Without responsible road activities, a safe and secure society cannot be built and there will be accidents and loss of economic goods* | *Not clear for the solution of the management techniques*  *The points on pedestrian is not clear.* |
| ***[2]. Dinesh Mohan. (2009)*** | ***ROAD ACCIDENTS IN INDIA*** | *This article has considered traffic fatalities per unit population as a rough indicator of risk faced by individuals* | *Gender and Age groups between 0-59 years of age have been analysed for determining which of the given set face the risk of deaths or injuries by road accidents. Time of the day is another dataset because the author had found out different causes for road accidents at different times of the day. Then Cities with population more than 1 million have been taken as a dataset. Rural Highways is another dataset.* | *Road safety policies in India must focus on the following issues to reduce the incidence of road traffic injuries: pedestrians and other non-motorist in urban areas; pedestrians, other non-motorists, and slow vehicles on highways; motorcycles and small cars in urban areas; over-involvement of trucks and buses; night-time driving; and wrong way drivers on divided highways.* | *Details of traffic crashes are not available at the national level. While the official road traffic fatality data may be close to the actual number, the injury data are gross underestimates5. In this report only fatality data are used for analysis as non-fatal data may suffer from many biases.* |
| ***[3].Dominic Merriott. (2016)*** | ***Factors associated with the farmer suicide crisis in India*** | *A literature search was undertaken on Ovid of the Embase (1980–2015 Week 18), Global Health (1973–2015 Week 17) and Ovid MEDLINE(R) inprocess and other non-indexed citations and Ovid MEDLINE(R) (1946 to present) databases. The search term was ‘‘[India and farm\* and (suicide or death)]”. This returned 362 results (301 unique), and all titles and abstracts were read and assessed. A total of 67 were isolated as having potentially some relevance to farmer suicides and read in full. More elementary searches of Google, Google Scholar, and PubMed were done, which identified a few additional reports.* | *The datasets of the study are basically potential factors that could lead to the suicide of the farmers and these datasets were analyzed one by one in reaching the conclusion. The datasets include: Indebtedness, Cash Crops, Bt Cotton, Agrarian Crisis and Neoliberal Reforms, Credit, Irrigation and Subsidiary Occupations.* | *Indebtedness and numerous factors relating to this are clearly identified as the most important risk factors. Further large-scale assessments are required to further understand the situation* | *Trying to present a broad picture of the evidence available in such a small space led to limitations. More databases could be searched, studies presented, and background given to the specific details of each study. This report is also not an authoritative review of every study. Not all available reports were found and of those that were, not all could be included here. Considering the large amount of data, included and not included in this paper, a more authoritative and objective systematic or literature review is warranted. Further research could also compare the situation of farmers in India directly with those of other countries.* |
| ***[4].Anusa Arunachalam Mohandoss, Rooban Thavarajah. (2016)*** | ***A study on suicide among Indians living with cancer during 2001–2014*** | *Data for this study were collected from the Indian National Crime Bureau Record, cancer registry publication in open domain, and published literature. Trends in the number of suicides associated with state, age groups, genders, and years were the only available parameters. Using these data, descriptive statistics of the rate of suicides, linear trend with age groups, gender, and geographical divisions are presented.* | *The year of suicide, age groups (classified as below 14 years, 15–29, 30–44, 45–59, and above 59 years), and gender (male/female). Midyear population of the country, as reported by the Census Authority of India was collected. Depending on geographical positioning, for ease of computation, India was geographically categorized as:*   * *South India (comprising the Indian territories of Andhra Pradesh, Tamil Nadu, Kerala, Karnataka, Pondicherry, Yanam, Karaikal, Mahe, Lakshadweep, Andaman and Nicobar Islands, and newly formed Telangana)* * *North India (comprising the Indian territories of New Delhi, Chandigarh, Haryana, Himachal Pradesh, Jammu and Kashmir, Punjab, and Rajasthan)* * *East India (comprising the Indian territories of Bihar, Jharkhand, Uttarakhand, and Uttar Pradesh)* * *West India (comprising the Indian territories of Goa, Gujarat, Maharashtra, Dadar and Nagar Haveli, and Daman)* * *North-East India (comprising the Indian territories of West Bengal, Sikkim, Assam, Arunachal Pradesh, Meghalaya, Nagaland, Manipur, Mizoram, and Tripura. Note: Cause of suicides in West Bengal in the year 2012 was not listed and hence, it appears missing)* * *Central India (comprising the Indian territories of Madhya Pradesh, Chhattisgarh, and Odisha).* | *The present study, for the first time, presents data on the prevalence of suicide among Indian PLWC. These data will highlight the need for inclusion of mental health workforce in cancer team not only to reduce substance abuse or treat signs/symptoms of mental illness but also to reduce pessimistic or suicidal thoughts and to impart positive feelings and rekindle hope in this highly vulnerable population* | *The authors lacked access to more detailed, individual data, detailed analysis and subsequent comparison to the published literature were not possible. An adequate access to (i) basic demographics (ii) and the type and site of cancer been available would have facilitated a much more detailed analysis. Till such a detailed data are available, the estimates and risk factors of suicide among PLWC could not be easily identified. More correlative details such as those available for previously published studies would have given the authors more insight into this phenomenon* |

**4. Objective of the Project**

The Objective of the Project is to analysis the different Social Issues on Suicide, Victims of Murder and Road Accidents. In order to make the analysis more accurate we adopted three parameters. The parameters were as follows:-

* Google Form Survey and Analysis of the data from it. This helped in giving what the people around us think about the topics by asking them some particular topics who the world of crime is related to Social Media.
* Static data analysis and prediction technique for the better check of the things going around us on the topics like victims of Murder and Road Accident.
* Dynamic data analysis followed by statistical data representation through mining of Twitter and getting to know the reviews of the people about the analysis. This was done on the topic of suicide followed by word cloud to represent the different weights of words in the search through online platform on a particular topic.

Thus in this way we are trying to predict the best ways how does Social Issues are being affected by internet.

**5. Innovation Component of the Project**

The project revolves around social agenda like Suicide, Road Accident and Victims of Murder. Most of the time we come to know about all this but we don’t take any action. Thus through our analysis we are not only doing analysis on static but also on dynamic data. The dynamic data does checking on the online scraped dataset and gives sentimental analysis to the work and in our project we are also constructing a word cloud which helps us to see the words used in analysis are having how much weight age (bigger the display more the weight age) This feature of the keywords search also help us to highlight the words most frequently used in a search and how these words are co-related to the social issue we are working upon. Suicide is a topic which is not predictable but this dynamic research that e made on it has increased the level checking and this is a good way of text mining.

Our project analysis can help NGOs and Government of India to look into the matters of Suicide, Road Accident and Victims of Murder for better analysis. The Static prediction of the data also shows different forms of Machine Learning Algorithm and Clustering features which makes our project stand out of crowd.

The Google form survey done by us helps in checking the mentality of the people towards the social issues the people are going through and also getting feedback from them about the different problems’ solution and for better understanding of the situation of how to tackle with them. This is an initiative taken by our group to reach out to the people who are in pain and how to get rid of such dilemma and be happy.

**6. Work Done and Implementation**

Methodology adapted:

**Algorithms of machine learning and Text mining for Dynamic data analysis**

The Machine Learning Algorithms we used are

* Clustering – To represent the grouping of data
* Regression – To predict the intensity of deaths
* Classification – To classify the changes that are taking place

The Text Mining For Dynamic data analysis we used are:

* APIs of Twitter – In order to scrap data
* Uses of Libraries like Twipy and WordCloud – For represent the weights of different words and to do sentimental analysis.

The Google Form Survey Analysis for getting to know the inset of the current generation about the following topic and to how far the social media is part of Social Issues.

Hardware and software requirements:

The software requirement is Anaconda version 3.0.0 and in it specifically Spider3.3(Python Tool for machine Learning)

Dataset used / Tools used:

For machine learning and visualization of the data set, the group is using Anaconda (Python 3.6).

Anaconda is a universal platform for doing machine learning projects and in our project, we are going to divide the project into two sets: training set and data set. Through this we will train the machine to analyze the data and to do visual representation

**There are 3 Data Sets( 2 Static and 1 Dynamic):**

**Static Data**

* Road Accidents in India

(from 2001 – 2010 for all Indian states and U.T.’s)

* Victim of Murder in India

**Dynamic Data**

* Suicide Analysis though Dynamic data by word scraping.

1. Where are you taking your dataset from?

The dataset is taken from a Machine Learning website named Kaggle.com. The link to the datasets is given below:-

<https://www.kaggle.com/manugupta/road-accidents-in-india>

<https://www.kaggle.com/gautamuoa/victim-of-murder-india>

Kaggle.com is a website famous for ML coding and has repository for dataset.

The Dynamic data is taken from Twitter APIs by web scraping for analysis of the data.

1. Is your project based on any other reference project (Stanford Univ. or MIT)?

The project Idea is self generated since the subject is all about actors and action and needs to see for social influence networking. Thus the social issues like road-accident due to use of mobile during driving or suicide due to not be able to mix with the people in the society thus being isolated, murder out of lack of possessions on interacting in ego-centric networks.

Thus we came up with this topic to analysis socially affected people as the rate of depression is increasing in the present world.

There are still some references that we went through are:

<https://www.technologyreview.com/s/609142/andrew-ng-has-a-chatbot-that-can-help-with-depression/> (from MIT)

<https://med.stanford.edu/phind/research/dreamteam-projects/multidimensional-predictors.html> (from Standford)

<https://med.stanford.edu/phind/research/individual-projects/sleep-health.html> (from Standford)

These are some of the esteemed projects MIT and Standford are working upon. We are also trying to contribute in this field through our analysis on Indian datasets.

1. How does your project differ from the reference project?\*

The projects that are represented above are based on chat bots, checking depression through sleep cycle and checking the chemical changes in the brain doing analysis.

In our project we are anglicising the death rate and other minor details that we get from the datasets and through comparative analysis we are going to check the various ways in which they are ranked and we are also trying to find what the NGOs of India have done to decrease all this and to how much extend.

Screenshot and Demo:

**a) The Static data Analysis:-**

(i)Road Accident

Classification

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.cross\_validation import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix

from matplotlib.colors import ListedColormap

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.naive\_bayes import GaussianNB

# Importing the dataset

dataset = pd.read\_csv('only\_road\_accidents\_data\_month2.csv')

X = dataset.iloc[:, [2, 13]].values

y = dataset.iloc[:, 1].values

# Splitting the dataset into the Training set and Test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

# Feature Scaling

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting Logistic Regression to the Training set

classifier = LogisticRegression(random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

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

# Making the Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Visualising the Training set results

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Logistic Regression (Training set)')

plt.xlabel('Months')

plt.ylabel('Estimated Suicide Ratio')

plt.legend()

plt.show()

# Visualising the Test set results

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Logistic Regression (Test set)')

plt.xlabel('months')

plt.ylabel('Estimated suicide ratio')

plt.legend()

plt.show()

# Fitting Decision Tree Classification to the Training set

classifier = DecisionTreeClassifier(criterion = 'entropy', random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

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

# Making the Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Visualising the Training set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Decision Tree Classification (Training set)')

plt.xlabel('Month')

plt.ylabel('Estimated Suicide')

plt.legend()

plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Decision Tree Classification (Test set)')

plt.xlabel('month')

plt.ylabel('Estimated suicide')

plt.legend()

plt.show()

# Fitting SVM to the Training set

classifier = SVC(kernel = 'linear', random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

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

# Making the Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Visualising the Training set results

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('SVM (Training set)')

plt.xlabel('Month')

plt.ylabel('Estimated Suicide')

plt.legend()

plt.show()

# Visualising the Test set results

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('SVM (Test set)')

plt.xlabel('Month')

plt.ylabel('Estimated Suicide')

plt.legend()

plt.show()

# Fitting Naive Bayes to the Training set

classifier = GaussianNB()

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

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

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Visualising the Training set results

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Naive Bayes (Training set)')

plt.xlabel('Month')

plt.ylabel('Estimated Suicide')

plt.legend()

plt.show()

# Visualising the Test set results

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

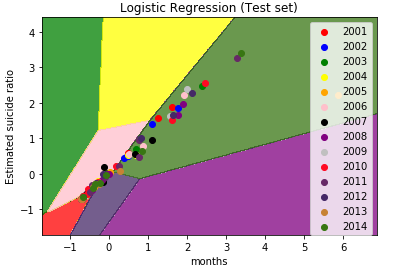
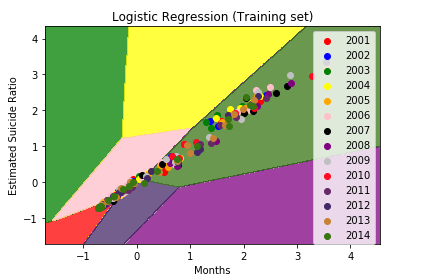
plt.title('Naive Bayes (Test set)')

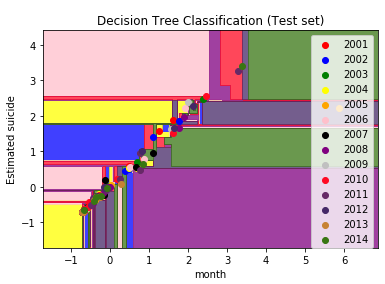
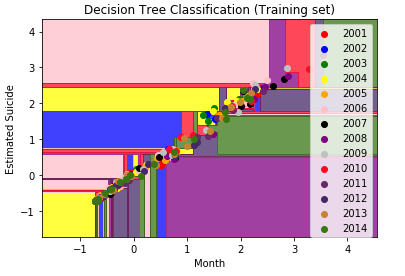
plt.xlabel('Month')

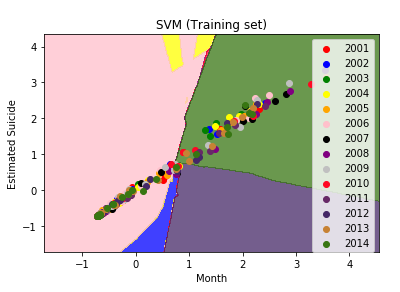
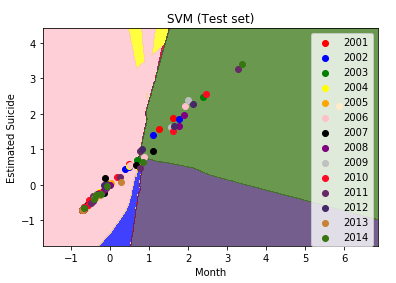
plt.ylabel('Estimated Suicide')

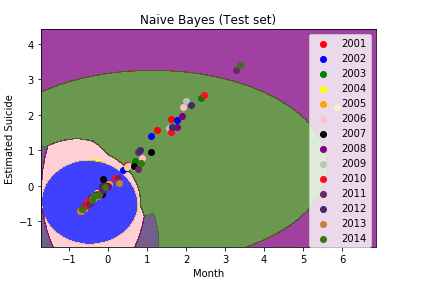
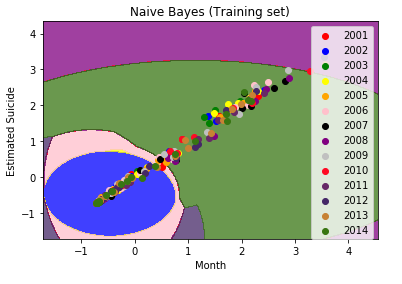
plt.legend()

plt.show()







Explanation of the Code:

First the header finals are initialized and then the data set is inputted. The dataset is distributed in 2 parts X and y where X is the independent dataset and the y is the dependent dataset. The dataset is distributed into training and test set and the scaling operation is done to scale the data down to the same level of weight age. Then following classification algorithms like Logistic Regression, Decision Tree, Support Vector Classification and Naive Bayesian has been done.

From the plots we can see that **Logistic Regression** best suited for the classification as the gradual increase of the domain of the classification is clearly visible.

The Actors are the years and the edges are being played by the months.

Clustering Analysis

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.cluster import KMeans

import scipy.cluster.hierarchy as sch

from sklearn.cluster import AgglomerativeClustering

# Importing the dataset

dataset = pd.read\_csv('only\_road\_accidents\_data\_month2.csv')

X = dataset.iloc[:, [2, 14]].values

# Using the elbow method to find the optimal number of clusters

wcss = []

for i in range(1, 15):

kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)

kmeans.fit(X)

wcss.append(kmeans.inertia\_)

plt.plot(range(1, 15), wcss)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

# Fitting K-Means to the dataset

kmeans = KMeans(n\_clusters = 14, init = 'k-means++', random\_state = 42)

y\_kmeans = kmeans.fit\_predict(X)

# Visualising the clusters

plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1 - 2001')

plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2 - 2002')

plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3 - 2003')

plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4 - 2004')

plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5 - 2005')

plt.scatter(X[y\_kmeans == 5, 0], X[y\_kmeans == 5, 1], s = 100, c = 'pink', label = 'Cluster 6 - 2006')

plt.scatter(X[y\_kmeans == 6, 0], X[y\_kmeans == 6, 1], s = 100, c = 'orange', label = 'Cluster 7 - 2007')

plt.scatter(X[y\_kmeans == 7, 0], X[y\_kmeans == 7, 1], s = 100, c = 'purple', label = 'Cluster 8 - 2008')

plt.scatter(X[y\_kmeans == 8, 0], X[y\_kmeans == 8, 1], s = 100, c = 'black', label = 'Cluster 9 - 2009')

plt.scatter(X[y\_kmeans == 9, 0], X[y\_kmeans == 9, 1], s = 100, c = '#FD4567', label = 'Cluster 10 - 2010')

plt.scatter(X[y\_kmeans == 10, 0], X[y\_kmeans == 10, 1], s = 100, c = 'violet', label = 'Cluster 11 - 2011')

plt.scatter(X[y\_kmeans == 11, 0], X[y\_kmeans == 11, 1], s = 100, c = 'grey', label = 'Cluster 12 - 2012')

plt.scatter(X[y\_kmeans == 12, 0], X[y\_kmeans == 12, 1], s = 100, c = 'brown', label = 'Cluster 13 - 2013')

plt.scatter(X[y\_kmeans == 13, 0], X[y\_kmeans == 13, 1], s = 100, c = 'gold', label = 'Cluster 14 - 2014')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 50, c = 'yellow', label = 'Centroids')

plt.title('Clusters of Road Accidents')

plt.xlabel('Number of poeple')

plt.ylabel('Number of Accicidents')

plt.legend()

plt.show()

dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))

plt.title('Dendrogram')

plt.xlabel('People')

plt.ylabel('Euclidean distances')

plt.show()

# Fitting Hierarchical Clustering to the dataset

hc = AgglomerativeClustering(n\_clusters = 14, affinity = 'euclidean', linkage = 'ward')

y\_hc = hc.fit\_predict(X)

# Visualising the clusters

plt.scatter(X[y\_hc == 0, 0], X[y\_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1- 2001')

plt.scatter(X[y\_hc == 1, 0], X[y\_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2 - 2002')

plt.scatter(X[y\_hc == 2, 0], X[y\_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3 - 2003')

plt.scatter(X[y\_hc == 3, 0], X[y\_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4 - 2004')

plt.scatter(X[y\_hc == 4, 0], X[y\_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5 - 2005')

plt.scatter(X[y\_hc == 0, 0], X[y\_hc == 0, 1], s = 100, c = 'yellow', label = 'Cluster 6- 2006')

plt.scatter(X[y\_hc == 1, 0], X[y\_hc == 1, 1], s = 100, c = 'orange', label = 'Cluster 7 - 2007')

plt.scatter(X[y\_hc == 2, 0], X[y\_hc == 2, 1], s = 100, c = 'pink', label = 'Cluster 8 - 2008')

plt.scatter(X[y\_hc == 3, 0], X[y\_hc == 3, 1], s = 100, c = 'gold', label = 'Cluster 9 - 2009')

plt.scatter(X[y\_hc == 4, 0], X[y\_hc == 4, 1], s = 100, c = 'violet', label = 'Cluster 10 - 2010')

plt.scatter(X[y\_hc == 0, 0], X[y\_hc == 0, 1], s = 100, c = 'purple', label = 'Cluster 11 - 2011')

plt.scatter(X[y\_hc == 1, 0], X[y\_hc == 1, 1], s = 100, c = 'grey', label = 'Cluster 12 - 2012')

plt.scatter(X[y\_hc == 2, 0], X[y\_hc == 2, 1], s = 100, c = 'black', label = 'Cluster 13 - 2013')

plt.scatter(X[y\_hc == 3, 0], X[y\_hc == 3, 1], s = 100, c = 'brown', label = 'Cluster 14 - 2014')

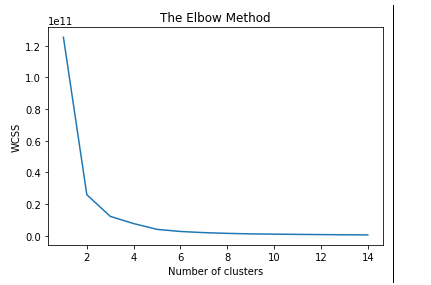
plt.title('Clusters of Accidents')

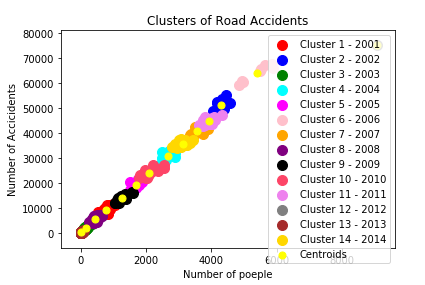
plt.xlabel('Number of accidents')

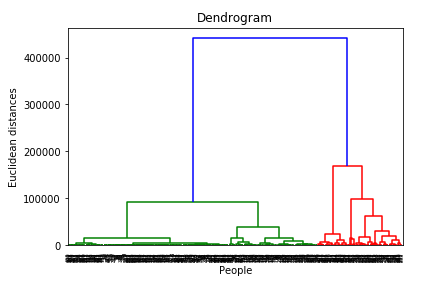
plt.ylabel('No of People')

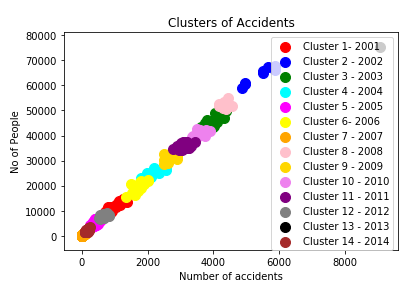
plt.legend()

plt.show()









Explanation of the code:

There are two clustering algorithm used in this case.

Kmeans:

Importing the libraries like matplotlib, numpy, and pandas for the plotting, counting and pandas to handle arrays. Then we read the dataset and store in dataset and since it is a unsupervised learning we take of of it.

Then we call kmeans library for forming clusters. We then make a search of the number of clusters through elbow function check. Then we decide to make clusters depending on the critical point of the elbow graph and plot the points with the centroids.

Hierarchical Clustering:

Importing the libraries like matplotlib, numpy, and pandas for the plotting, counting and pandas to handle arrays. Then we read the dataset and store in dataset and since it is a unsupervised learning we take of of it.

Then we call the hierarchy for dataset evaluation and make the dendograms.

We did on Agglomerative Clustering with the help of Euclidian distance and fitting it into the prediction for the output. Thus the output is shown.

Again here also the actors are years and edges are represented by other factors.

(ii)Victims of Murder

Classification

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.cross\_validation import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix

from matplotlib.colors import ListedColormap

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.naive\_bayes import GaussianNB

# Importing the dataset

dataset = pd.read\_csv('VICTIM\_OF\_MURDER\_0.csv')

X = dataset.iloc[:, [3,9]].values

y = dataset.iloc[:, 1].values

# Splitting the dataset into the Training set and Test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

# Feature Scaling

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting Logistic Regression to the Training set

classifier = LogisticRegression(random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

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

# Making the Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Visualising the Training set results

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Logistic Regression (Training set)')

plt.xlabel('Months')

plt.ylabel('Estimated Suicide Ratio')

plt.legend()

plt.show()

# Visualising the Test set results

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Logistic Regression (Test set)')

plt.xlabel('months')

plt.ylabel('Estimated suicide ratio')

plt.legend()

plt.show()

# Fitting Decision Tree Classification to the Training set

classifier = DecisionTreeClassifier(criterion = 'entropy', random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

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

# Making the Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Visualising the Training set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Decision Tree Classification (Training set)')

plt.xlabel('Month')

plt.ylabel('Estimated Suicide')

plt.legend()

plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Decision Tree Classification (Test set)')

plt.xlabel('month')

plt.ylabel('Estimated suicide')

plt.legend()

plt.show()

# Fitting SVM to the Training set

classifier = SVC(kernel = 'linear', random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

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

# Making the Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Visualising the Training set results

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('SVM (Training set)')

plt.xlabel('Month')

plt.ylabel('Estimated Suicide')

plt.legend()

plt.show()

# Visualising the Test set results

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('SVM (Test set)')

plt.xlabel('Month')

plt.ylabel('Estimated Suicide')

plt.legend()

plt.show()

# Fitting Naive Bayes to the Training set

classifier = GaussianNB()

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

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

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Visualising the Training set results

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Naive Bayes (Training set)')

plt.xlabel('Month')

plt.ylabel('Estimated Suicide')

plt.legend()

plt.show()

# Visualising the Test set results

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

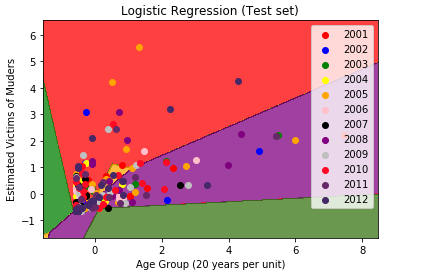
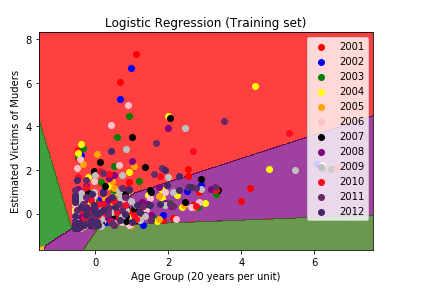
plt.title('Naive Bayes (Test set)')

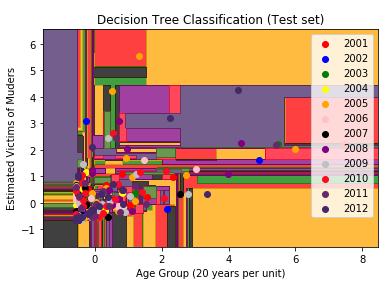
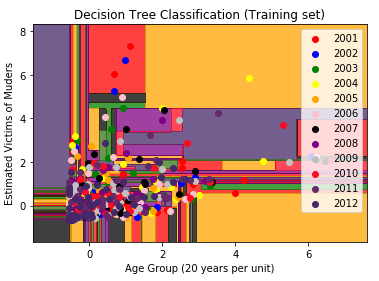
plt.xlabel('Month')

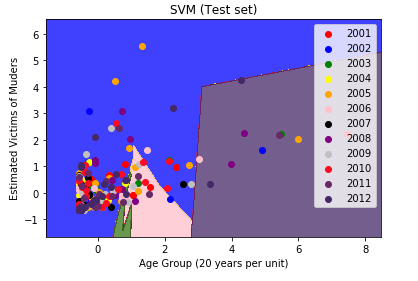
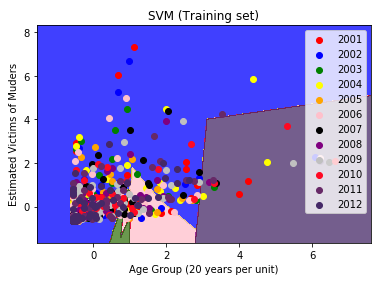
plt.ylabel('Estimated Suicide')

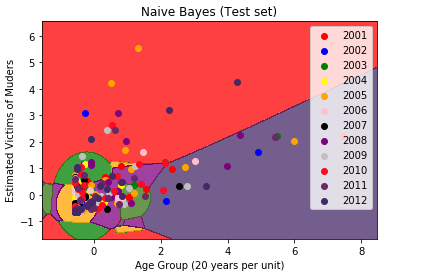
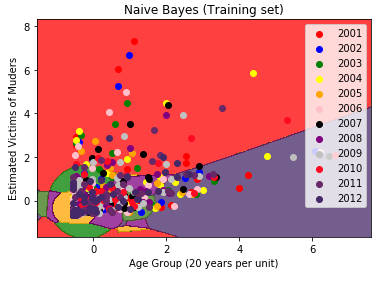
plt.legend()

plt.show()









First the header finals are initialized and then the data set is inputted. The dataset is distributed in 2 parts X and y where X is the independent dataset and the y is the dependent dataset. The dataset is distributed into training and test set and the scaling operation is done to scale the data down to the same level of weight age. Then following classification algorithms like Logistic Regression, Decision Tree, Support Vector Classification and Naive Bayesian has been done.

From the plots we can see that **Logistic Regression** best suited for the classification as the gradual increase of the domain of the classification is clearly visible.

The Actors are the years and the edges are being played by the age groups.

Clustering:

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.cluster import KMeans

import scipy.cluster.hierarchy as sch

from sklearn.cluster import AgglomerativeClustering

# Importing the dataset

dataset = pd.read\_csv('VICTIM\_OF\_MURDER\_0.csv')

X = dataset.iloc[:, [3,9]].values

# Using the elbow method to find the optimal number of clusters

wcss = []

for i in range(1, 15):

kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)

kmeans.fit(X)

wcss.append(kmeans.inertia\_)

plt.plot(range(1, 15), wcss)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

# Fitting K-Means to the dataset

kmeans = KMeans(n\_clusters = 14, init = 'k-means++', random\_state = 42)

y\_kmeans = kmeans.fit\_predict(X)

# Visualising the clusters

plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1 - 2001')

plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2 - 2002')

plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3 - 2003')

plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4 - 2004')

plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5 - 2005')

plt.scatter(X[y\_kmeans == 5, 0], X[y\_kmeans == 5, 1], s = 100, c = 'pink', label = 'Cluster 6 - 2006')

plt.scatter(X[y\_kmeans == 6, 0], X[y\_kmeans == 6, 1], s = 100, c = 'orange', label = 'Cluster 7 - 2007')

plt.scatter(X[y\_kmeans == 7, 0], X[y\_kmeans == 7, 1], s = 100, c = 'purple', label = 'Cluster 8 - 2008')

plt.scatter(X[y\_kmeans == 8, 0], X[y\_kmeans == 8, 1], s = 100, c = 'black', label = 'Cluster 9 - 2009')

plt.scatter(X[y\_kmeans == 9, 0], X[y\_kmeans == 9, 1], s = 100, c = '#FD4567', label = 'Cluster 10 - 2010')

plt.scatter(X[y\_kmeans == 10, 0], X[y\_kmeans == 10, 1], s = 100, c = 'violet', label = 'Cluster 11 - 2011')

plt.scatter(X[y\_kmeans == 11, 0], X[y\_kmeans == 11, 1], s = 100, c = 'grey', label = 'Cluster 12 - 2012')

plt.scatter(X[y\_kmeans == 12, 0], X[y\_kmeans == 12, 1], s = 100, c = 'brown', label = 'Cluster 13 - 2013')

plt.scatter(X[y\_kmeans == 13, 0], X[y\_kmeans == 13, 1], s = 100, c = 'gold', label = 'Cluster 14 - 2014')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 50, c = 'yellow', label = 'Centroids')

plt.title('Clusters of Victims of Murder')

plt.xlabel('Number of Age')

plt.ylabel('Number of People')

plt.legend()

plt.show()

dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))

plt.title('Dendrogram')

plt.xlabel('People')

plt.ylabel('Euclidean distances')

plt.show()

# Fitting Hierarchical Clustering to the dataset

hc = AgglomerativeClustering(n\_clusters = 14, affinity = 'euclidean', linkage = 'ward')

y\_hc = hc.fit\_predict(X)

# Visualising the clusters

plt.scatter(X[y\_hc == 0, 0], X[y\_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1- 2001')

plt.scatter(X[y\_hc == 1, 0], X[y\_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2 - 2002')

plt.scatter(X[y\_hc == 2, 0], X[y\_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3 - 2003')

plt.scatter(X[y\_hc == 3, 0], X[y\_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4 - 2004')

plt.scatter(X[y\_hc == 4, 0], X[y\_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5 - 2005')

plt.scatter(X[y\_hc == 5, 0], X[y\_hc == 5, 1], s = 100, c = 'yellow', label = 'Cluster 6- 2006')

plt.scatter(X[y\_hc == 6, 0], X[y\_hc == 6, 1], s = 100, c = 'orange', label = 'Cluster 7 - 2007')

plt.scatter(X[y\_hc == 7, 0], X[y\_hc == 7, 1], s = 100, c = 'pink', label = 'Cluster 8 - 2008')

plt.scatter(X[y\_hc == 8, 0], X[y\_hc == 8, 1], s = 100, c = 'gold', label = 'Cluster 9 - 2009')

plt.scatter(X[y\_hc == 9, 0], X[y\_hc == 9, 1], s = 100, c = 'violet', label = 'Cluster 10 - 2010')

plt.scatter(X[y\_hc == 10, 0], X[y\_hc == 10, 1], s = 100, c = 'purple', label = 'Cluster 11 - 2011')

plt.scatter(X[y\_hc == 11, 0], X[y\_hc == 11, 1], s = 100, c = 'grey', label = 'Cluster 12 - 2012')

plt.scatter(X[y\_hc == 12, 0], X[y\_hc == 12, 1], s = 100, c = 'black', label = 'Cluster 13 - 2013')

plt.scatter(X[y\_hc == 13, 0], X[y\_hc == 13, 1], s = 100, c = 'brown', label = 'Cluster 14 - 2014')

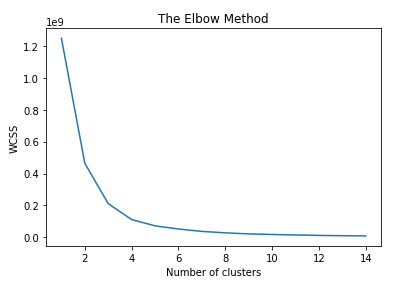
plt.title('Clusters of Victims of Murder')

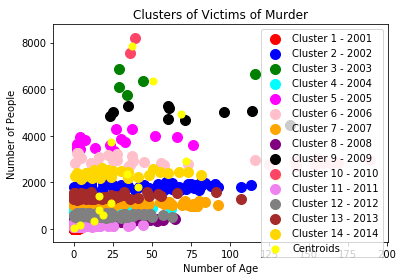
plt.xlabel('Number of Age')

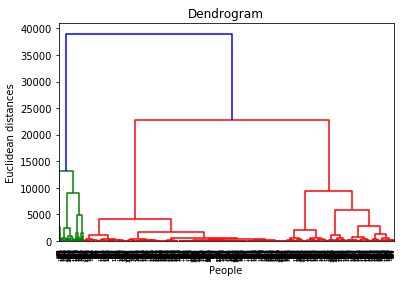
plt.ylabel('Number of People')

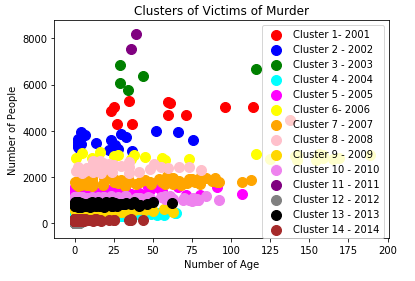
plt.legend()

plt.show()









Explanation of the code:

There are two clustering algorithm used in this case.

Kmeans:

Importing the libraries like matplotlib, numpy, and pandas for the plotting, counting and pandas to handle arrays. Then we read the dataset and store in dataset and since it is a unsupervised learning we take of of it.

Then we call kmeans library for forming clusters. We then make a search of the number of clusters through elbow function check. Then we decide to make clusters depending on the critical point of the elbow graph and plot the points with the centroids.

Hierarchical Clustering:

Importing the libraries like matplotlib, numpy, and pandas for the plotting, counting and pandas to handle arrays. Then we read the dataset and store in dataset and since it is a unsupervised learning we take of of it.

Then we call the hierarchy for dataset evaluation and make the dendograms.

We did on Agglomerative Clustering with the help of Euclidian distance and fitting it into the prediction for the output. Thus the output is shown.

Again here also the actors are years and edges are represented by other factors.

**b) The Dynamic data Analysis:-**

Sentimental Analysis

import tweepy

from textblob import TextBlob

import csv

import re

import sys

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from IPython.display import display

#Does plotting as well as analysis

consumer\_key='hzNYHN9xRlXeKu7g2aj7AI'

consumer\_secret='Xm3ScyKHRL5EBXdr08n1IHuJjO3YLv1ea65rVCVo56SsYNq'

access\_token\_key='78845728-kLsebe0WCdRMxISdaIRbx2pNzgUzsrNSKSbYDy'

access\_token\_secret='jd7cOjgDph8kznb4pqofoA0TqFtlUs1RmCJBCPsf'

auth=tweepy.OAuthHandler(consumer\_key,consumer\_secret)

auth.set\_access\_token(access\_token\_key,access\_token\_secret)

api=tweepy.API(auth)

topicname=input("Enter the topic you want to search about\n")

number=int(input("Enter the number of tweets to search\n"))

public\_tweets=api.search(

lang="en",

q=topicname + " -rt",

count=number, result\_type='mixed')

unwanted\_words=['@','RT',':','https','http']

symbols=['@','#']

data=[]

data = pd.DataFrame(data=[tweet.text for tweet in public\_tweets], columns=['Tweets'])

display(data.head(number))

data['len'] = np.array([len(tweet.text) for tweet in public\_tweets])

data['ID'] = np.array([tweet.id for tweet in public\_tweets])

data['Date'] = np.array([tweet.created\_at for tweet in public\_tweets])

data['Source'] = np.array([tweet.source for tweet in public\_tweets])

data['Likes'] = np.array([tweet.favorite\_count for tweet in public\_tweets])

data['RTs'] = np.array([tweet.retweet\_count for tweet in public\_tweets])

display(data.head(number))

fav\_max = np.max(data['Likes'])

rt\_max = np.max(data['RTs'])

fav = data[data.Likes == fav\_max].index[0]

rt = data[data.RTs == rt\_max].index[0]

# Max FAVs:

print("The tweet with most likes is: \n{}".format(data['Tweets'][fav]))

print("Number of likes: {}".format(fav\_max))

print("{} characters.\n".format(data['len'][fav]))

# Max RTs:

print("The tweet with most retweets is: \n{}".format(data['Tweets'][rt]))

print("Number of retweets: {}".format(rt\_max))

print("{} characters.\n".format(data['len'][rt]))

tlen = pd.Series(data=data['len'].values, index=data['Date'])

tfav = pd.Series(data=data['Likes'].values, index=data['Date'])

tret = pd.Series(data=data['RTs'].values, index=data['Date'])

tlen.plot(figsize=(16,4), color='r');

tfav.plot(figsize=(16,4), label="Likes", legend=True)

tret.plot(figsize=(16,4), label="Retweets", legend=True);

plt.show()

countpos=countneg=countneut=0

arr=[]

a1=[]

for tweet in public\_tweets:

tweeta = tweet.text

tidy\_tweet = (tweeta.strip().encode('ascii', 'ignore')).decode("utf-8")

analysis= TextBlob(tidy\_tweet)

#print (analysis.sentiment)

if(analysis.sentiment.polarity > 0.2):

polarity = 'Positive'

countpos=countpos+1

pol=1

elif(0<=analysis.sentiment.polarity <=0.2):

polarity = 'Neutral'

countneut=countneut+1

pol=0

elif(analysis.sentiment.polarity < 0):

polarity = 'Negative'

countneg=countneg+1

pol=-1

arr.append(polarity)

a1.append(pol)

se=pd.Series(arr)

df=pd.DataFrame(data)

df['Sentiment']=se.values

ss=pd.Series(a1)

df['Senti']=ss.values

#print(df)

df.to\_csv('analysedfile.csv')

positive = countpos

negative = countneg

neutral = countneut

colors = ['blue', 'red', 'yellow']

sizes = [positive, negative, neutral]

labels = 'Positive', 'Negative', 'Neutral'

plt.pie(

x=sizes,

shadow=True,

colors=colors,

labels=labels,

startangle=90

)

plt.title("Sentiment of {} Tweets about {}".format(number, topicname))

plt.show()

percpos=(countpos/int(number))\*100

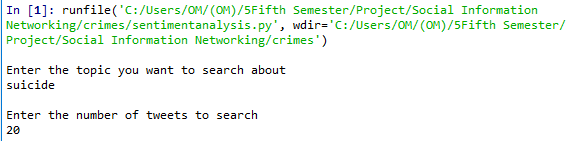
percneg=(countneg/int(number))\*100

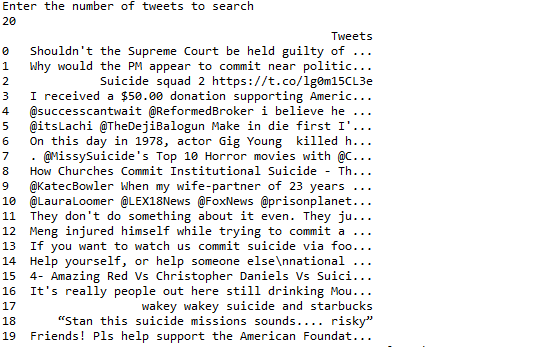
percneut=(countneut/int(number))\*100

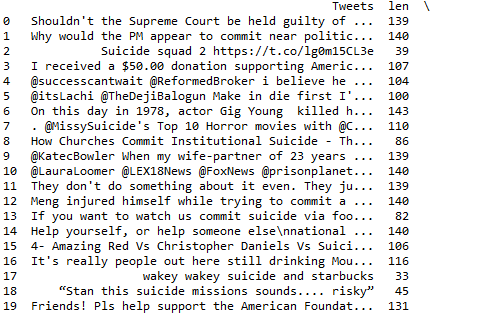
print("\nPercentage of positive tweets:",percpos)

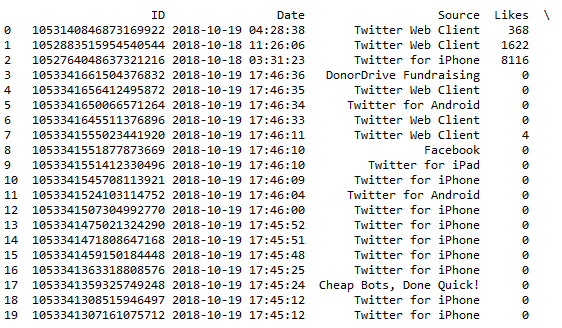
print("\nPercentage of negative tweets:",percneg)

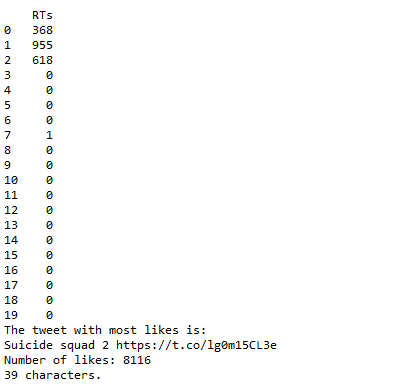
print("\nPercentage of neutral tweets:",percneut)

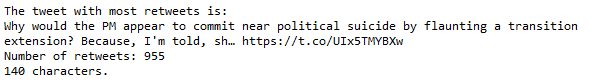


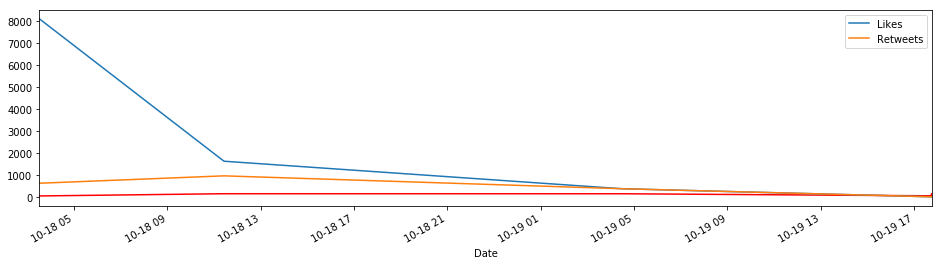


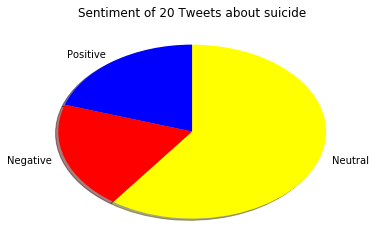












Percentage of positive tweets: 20.0

Percentage of negative tweets: 20.0

Percentage of neutral tweets: 60.0

The Explanation of the code:

Importing of the libraries and collecting of the consumer key and secret key from the Twitter API and checking of the access token key and access token secret key. Then checking the authority through tweepy library. Thus api call is done to get the data from the online.

Then user inputs the tweet to be searched and number of tweets and thus th search takes place. Then mixed English queries are searched. We remove the unwanted words from online. And check the popular symbols like @ and # get the data. The output shows likes, retweets, source from where it is taken, ID and Date of the tweets.

Then we represent the maximum tweeted data.

Then we check the popularity of a tweet and see for the positive, neutral and negative.Then plotting is done followed by percentage calculation.

The Word Cloud

import numpy as np

import matplotlib.pyplot as plt

import re

from twython import Twython

from PIL import Image

from wordcloud import WordCloud, STOPWORDS

from IPython.display import Image as im

consumer\_key = 'hzNYHN9xRlXeKu7g2aj7nAI'

consumer\_secret = 'Xm3ScyKHRL5EBXdr08n1IHuJjOv1ea68Td5rVCVo56SsYNq'

twitter = Twython(consumer\_key, consumer\_secret)

access\_key = '78845728-kLsebXB9e0WMxISdaIRbx2pNzgUzsrNSKSbYDy'

access\_secret = 'jd7cOjgDi0rysph8k4pqofoA0TqFtlUs1RmCJBCPsf'

#Get timeline

user\_timeline=twitter.get\_user\_timeline(screen\_name='depression',count=1)

#get most recent id

last\_id = user\_timeline[0]['id']-1

for i in range(16):

batch = twitter.get\_user\_timeline(screen\_name='depression',count=200, max\_id=last\_id)

user\_timeline.extend(batch)

last\_id = user\_timeline[-1]['id'] - 1

#Extract textfields from tweets

raw\_tweets = []

for tweets in user\_timeline:

raw\_tweets.append(tweets['text'])

#Create a string form of our list of text

raw\_string = ''.join(raw\_tweets)

no\_links = re.sub(r'http\S+', '', raw\_string)

no\_unicode = re.sub(r"\\[a-z][a-z]?[0-9]+", '', no\_links)

no\_special\_characters = re.sub('[^A-Za-z ]+', '', no\_unicode)

words = no\_special\_characters.split(" ")

words = [w for w in words if len(w) > 2] # ignore a, an, be, ...

words = [w.lower() for w in words]

words = [w for w in words if w not in STOPWORDS]

wc = WordCloud(background\_color="white", max\_words=2000, mask=None)

clean\_string = ','.join(words)

wc.generate(clean\_string)

f = plt.figure(figsize=(30,30))

f.add\_subplot(1,2, 1)

#plt.imshow(mask, cmap=plt.cm.gray, interpolation='bilinear')

plt.title('Original Stencil', size=40)

#plt.axis("off")

#f.add\_subplot(1,2, 2)

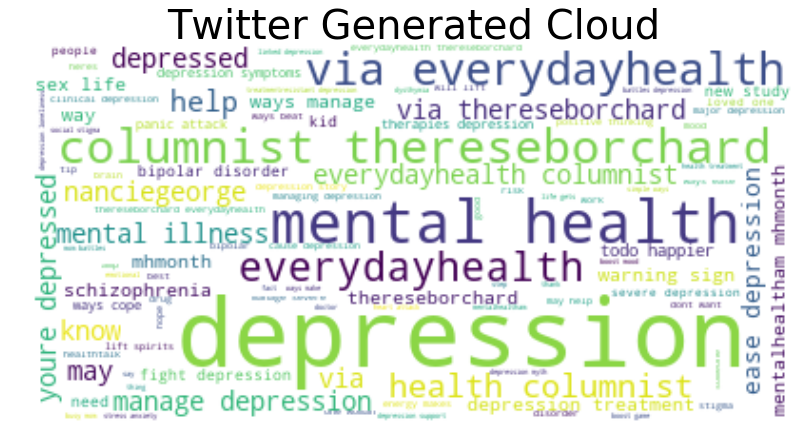
plt.imshow(wc, interpolation='bilinear')

plt.title('Twitter Generated Cloud', size=40)

plt.axis("off")

plt.show()

**The Output:**



The Explanation of Code:

First we initialize the libraries and handle the access the key and token from twitter. Then we use user\_timeline function see the depression tweets. We do the checking of the data by removing the none needed data followed by the highlight of keywords.

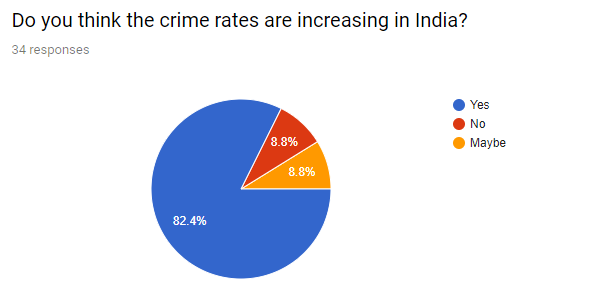
Thus plotting them in the output.

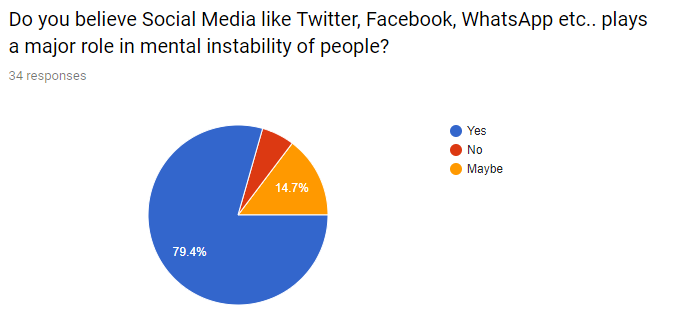
**7. Results and Discussions**

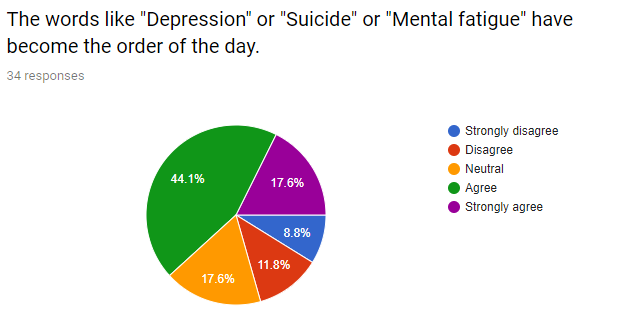
**The Expected TimeLine, We had:-**

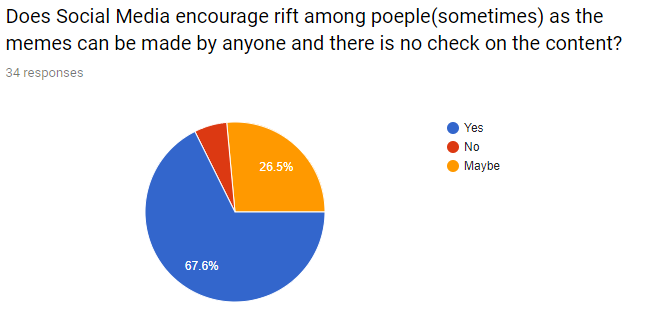
* Review 1
  + Idea and how the data analysis is useful for our project
* Review 2
  + 50% of implementation of code and discussion and continuation of the project
* Review 3
  + Total implementation with graph analysis in detail.

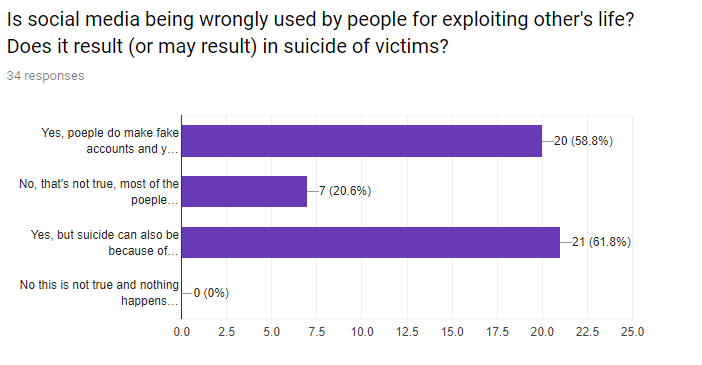
**The Google Form Analysis:-**

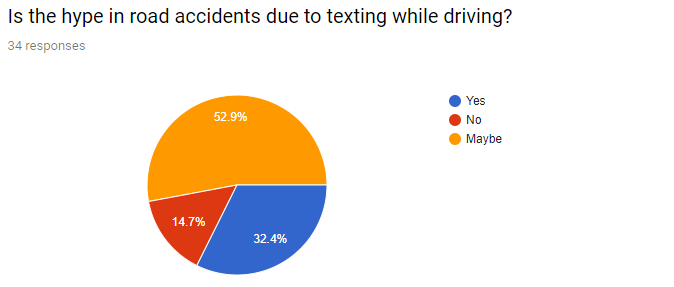


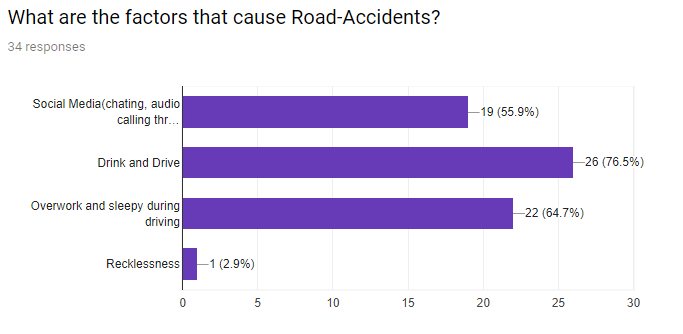


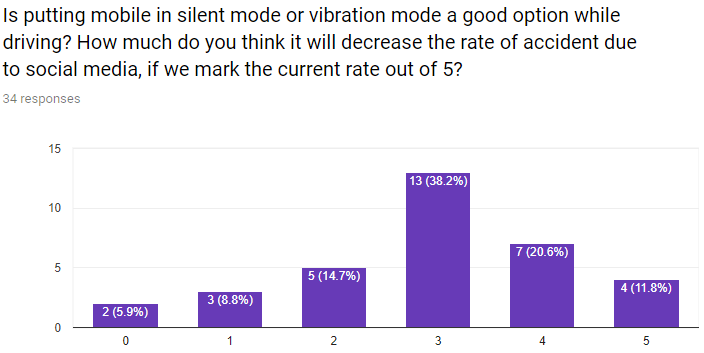


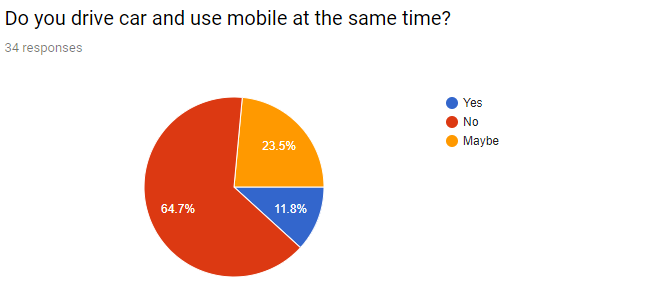


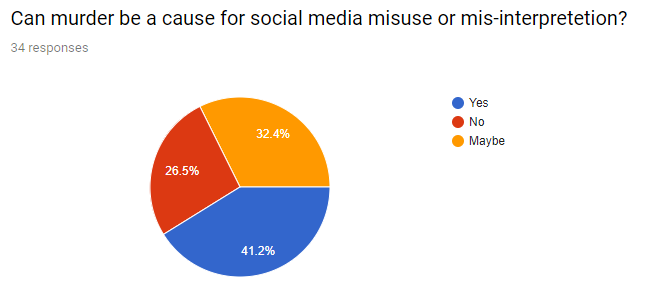


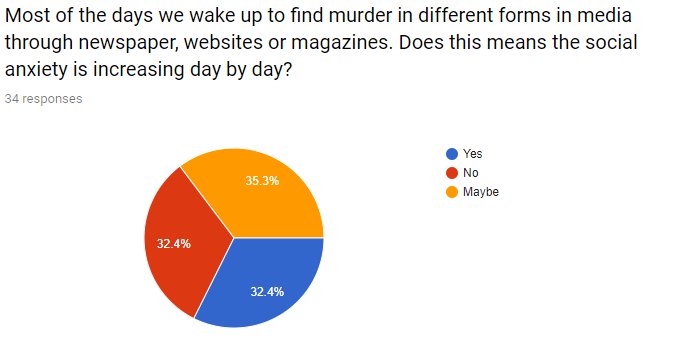


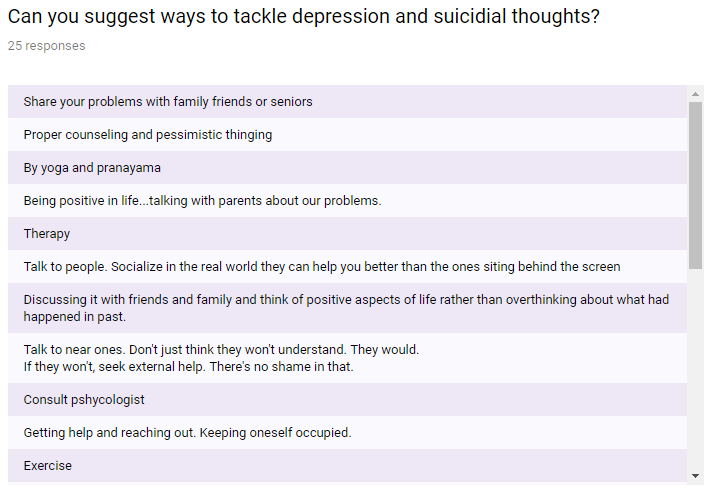


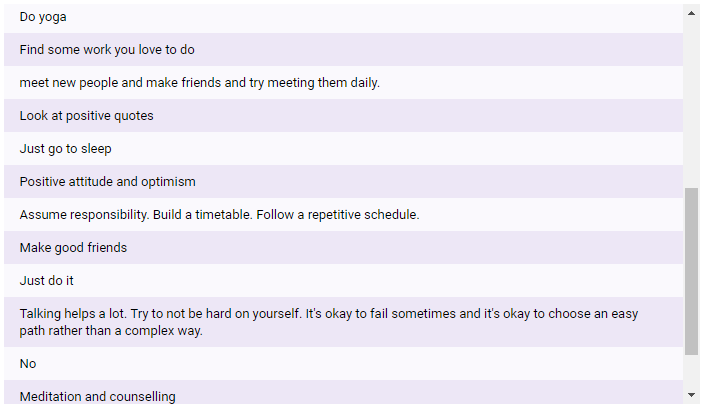


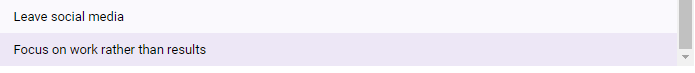


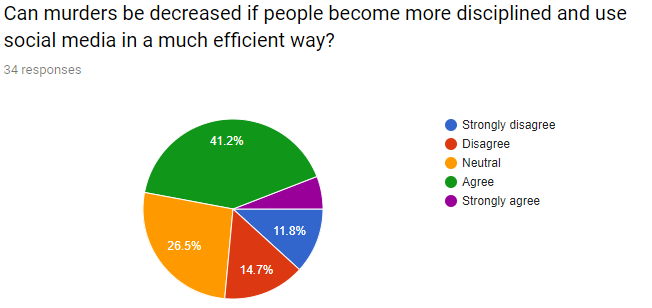


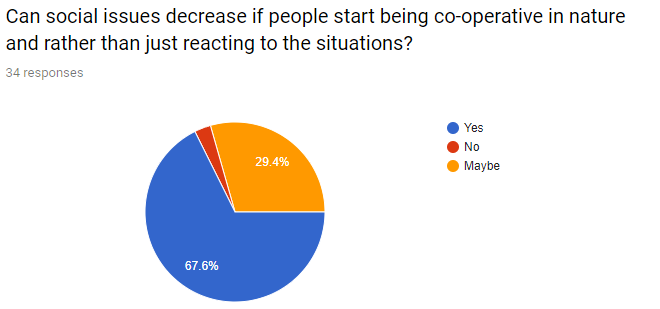




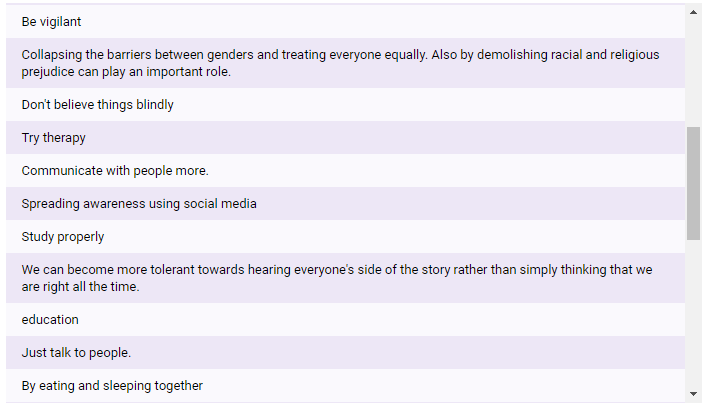


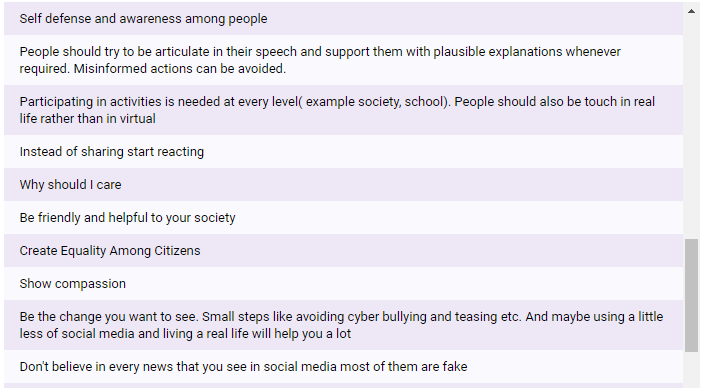


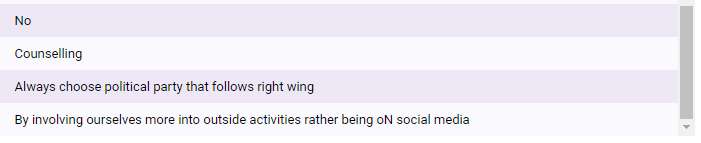












**Analysis of the Google Form Understanding**

The group did a survey using Google form to find out what perception the general people have about various kinds of crimes and their causes and some measures which normal citizens can take in their day to day lives.

Some of the results that were obtained through the survey are:

1. Crime rates are increasing in India and Social Media like Facebook, Twitter, Whatsapp etc. play a role in instigating such behaviour amongst the masses.
2. Nowadays, people are often depressed or contemplate suicide and due to a hectic lifestyle, people are “mentally fatigued”.
3. Social Media is a very powerful tool. But unfortunately, instead of using such a powerful tool for the betterment of society, some people are using it to spread hatred and to exploit somebody else’s private life. Even people who are not very educated are being brainwashed by the influence of Social Media.
4. Before, only drunk driving used to be a major cause for road accidents. But nowadays, using Social Media while driving for example texting while driving or clicking selfies while driving is also becoming a major reason for road accidents.
5. As the saying goes, an ocean is made by adding water drop by drop. Similarly, taking small steps one at a time can lead to recovery and reduction of all these dreadful events. People can use Social Media in a more responsible and disciplined manner. Also, if people just think and respond to a situation rather than reacting furiously to anything that can also be a misinterpretation or a misunderstanding can lead to lives being saved.

**The Outcomes:-**

* The Static data analysis shows the various prediction and the technique to predict on accident of roads and victims of murder.
* Dynamic data analysis shows the various sentimental analysis like how most of the people remain aloof to the suicide and keywords are being represented in wordcloud.
* Overall view of the data by use of prediction and machine learning algorithms to predict ups and downs in the rate of each above topic.
* The Google form shows the thought process of current mentality of people for the things going around related to social issues.
* Moreover we are seeing how the road accident is increasing day by day and the data analysis also show that Logistic Regression and SVM suits the best for the prediction by data visualization. The clustering algorithm also shows how the data analysis getting a rise in deaths.
* Thus by supervised and unsupervised learning we see the death rate are increasing.
* The Neutrality is more towards the suicide and depression so from google survey we could see had the people generally see feel of suicide.
* Years are taken as actors as the states were clumsy and other attributes are taken as the edges to work upon.

Thus from Google form to static machine learning prediction to dynamic sentimental analysis. The end results show that we should be humble with the people around us and we should find ways of Lessing the social media impact on our generation. Thus reduce use of social media and reduce the impact of social issues.

**8. References**

1.https://www.kaggle.com

2.https://www.google.com

3.https://www.technologyreview.com/s/609142/andrew-ng-has-a-chatbot-that-can-help-with-depression/

4.https://med.stanford.edu/phind/research/dreamteamprojects/multidimensional-predictors.html

5.https://med.stanford.edu/phind/research/individual-projects/sleep-health.html

6.http://shodhganga.inflibnet.ac.in/bitstream/10603/150521/18/18\_article.pdf

7.https://www.researchgate.net/publication/273511031\_Road\_accidents\_in\_India

8.https://www.sciencedirect.com/science/article/pii/S2210600615300277

9. http://www.indianjcancer.com/article.asp?issn=0019-509X;year=2016;volume=53;issue=3;spage=435;epage=440;aulast=Mohandoss