# Foss Lab – Final  Report

## Name : Sameer Kumar Kaushik

## Roll No. : 205319012

## Dataset:

Kerala State 100 Helpline dataset

**Discerption :**

The dataset consists of over 900 observations with 16 features such as call time, event type and district.

**Task:**

classify, visualize and analyze the crime data based on  the crime sheet.

**Methodologies:**

1. Word Clouds
2. NLP using NLTK
3. Classification using Logistic Regression
4. SVM
5. Gradient Boosting Classifier
6. Random Forest Classifier
7. Multi Layer Perceptron Classifier
8. Deep Learning

**Libraries and Requirements:**

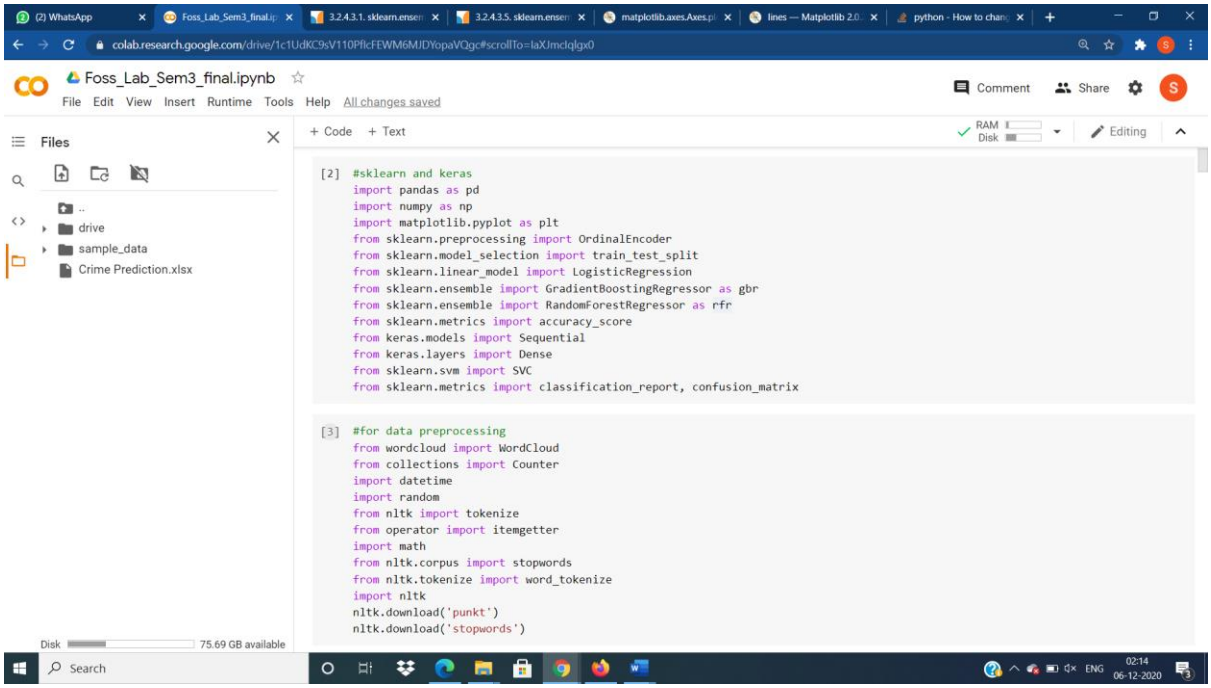
1. Pandas
2. Numpy
3. Matplotlib
4. Wordcloud
5. NLTK
6. Geopandas
7. Descartes
8. Sklearn
9. Keras

Colab link:
https://colab.research.google.com/drive/1c1UdKC9sV110PfIcFEWM6MJDYopaVQgc#scrollTo=L-Gf6BPv9_yk

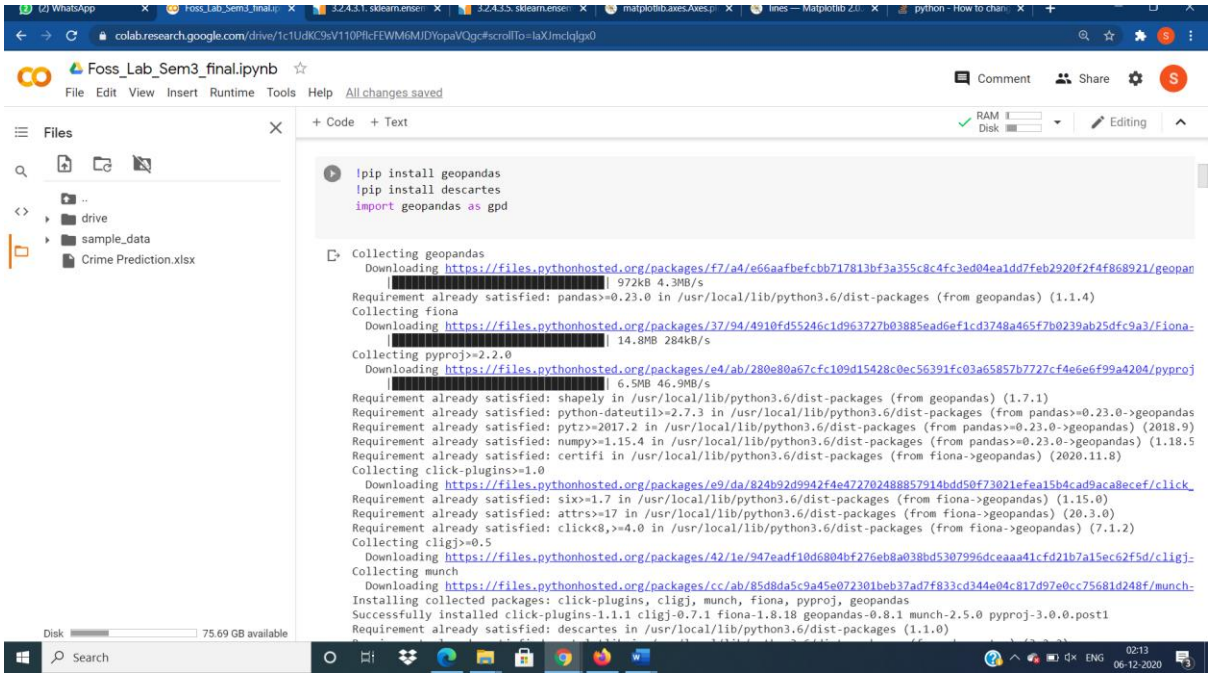
Git Hub Link:

https://github.com/sameerkaushik007/Foss_Lab_Final

## Import different type of libraries




## Install geopandas library for geo map of Kerala region




### Pre-Processing

There is multiple noise data and multiple duplicate row value so. We try to clean up irrelevant data using pre-processing technique.

Create transformation function for pre-processing using convert_I and convert_II  function

Avoid the nun value

```
#Transformation Functions for Pre-Processing
def convert_I(lis):
    list1 = []
    for i in lis:
        if i[0]!="event_id":
            if i[0]=='nan' and i[1]=='nan':# Skip nan value
                continue
            else:
                list1.append(i)
    list2 = convert_II(list1)
    return list2

def convert_II(list1):
    i=0
    list2 = []
    while i < len(list1):
        if list1[i][0]!='nan':
            temp = list1[i]
            i+=2
            signal = []
            disconnect = []
            while list1[i][0]=='nan':
                signal.append(list1[i][1])
                disconnect.append(list1[i][2])
                i+=1
                if(i==len(list1)): #safety net
                    break
            temp.append(signal)
            temp.append(disconnect)
            list2.append(temp) #append data frame to list2
```

read crime prediction xl file and in xl sheet there is 16 column

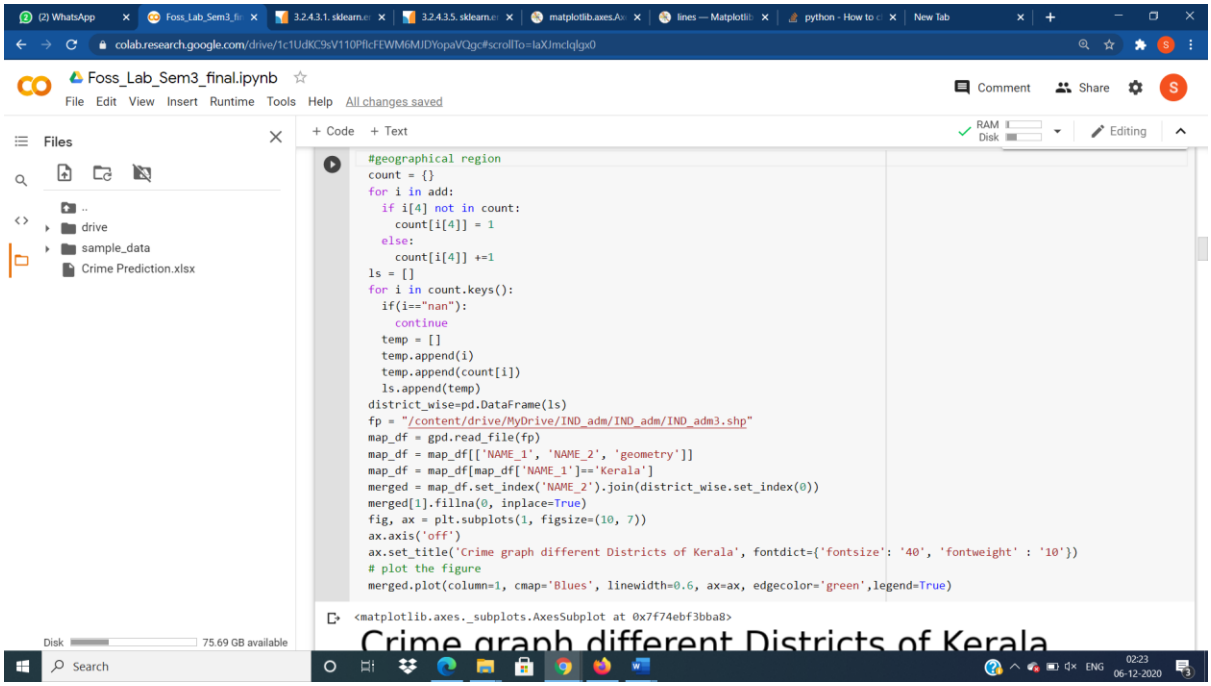
```
[5]         else:
                i+=1
        return list2 #return list2
```

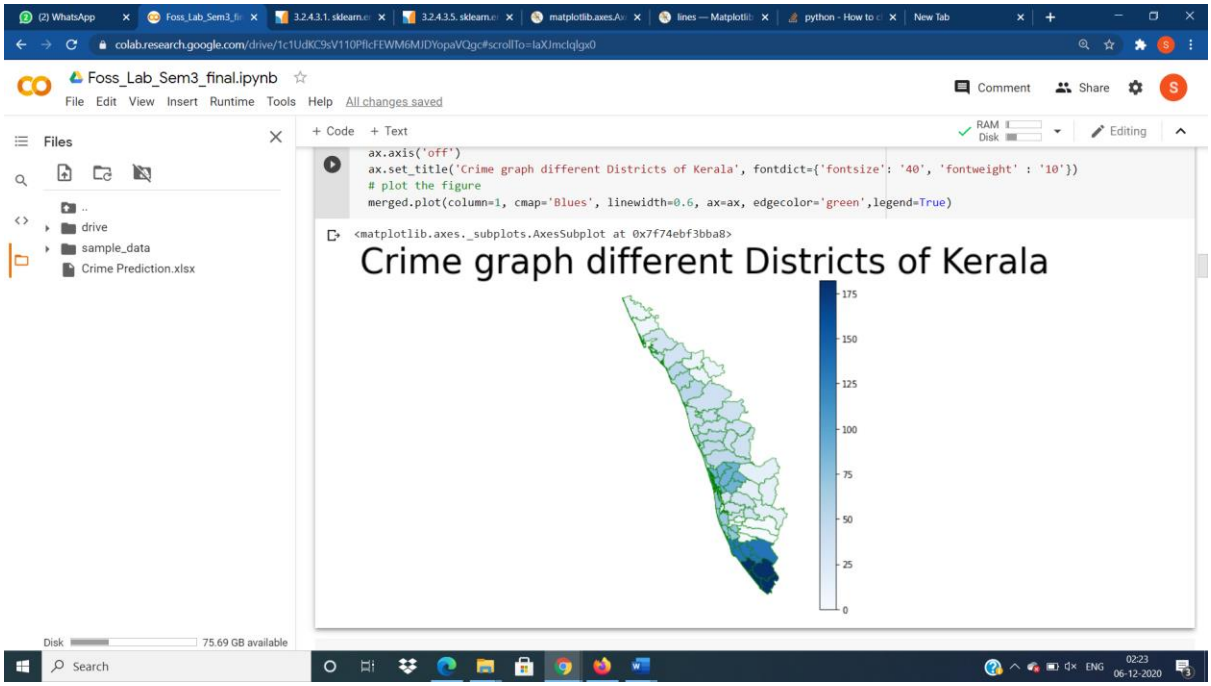
```
[157] #read xlsx
      add = []
      for i in range(1,4):
        df1 = pd.read_excel("/content/Crime Prediction.xlsx", header=0,sheet_name=str(i))
        df = df1.iloc[:,0:16]
        df = df.replace(np.nan,'nan', regex=True)
        lis = df.values.tolist()
        lis1 = convert_I(lis)
        for item in lis1:
          add.append(item)
      print("Number of Value after cleaning data: ",len(add))
```

```
Number of Value after cleaning data:  801
```

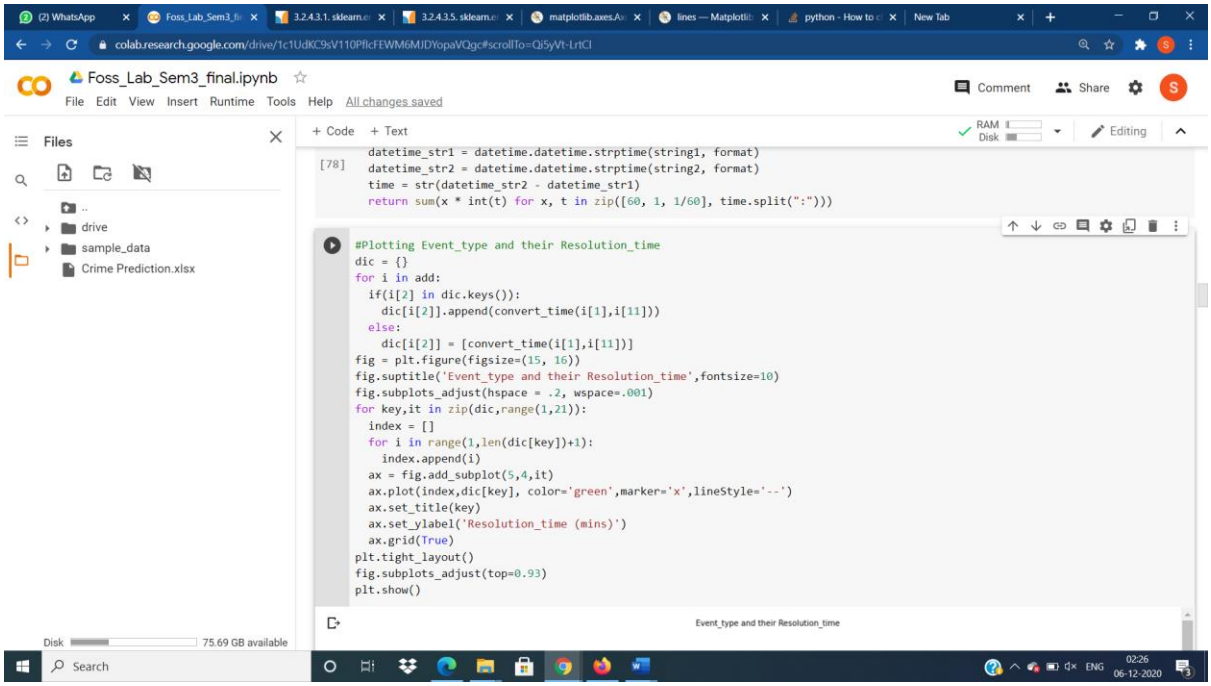
```
#geographical region
count = {}
for i in add:
  if i[4] not in count:
    count[i[4]] = 1
  else:
    count[i[4]] +=1
ls = []
for i in count.keys():
  if(i=="nan"):
    continue
```

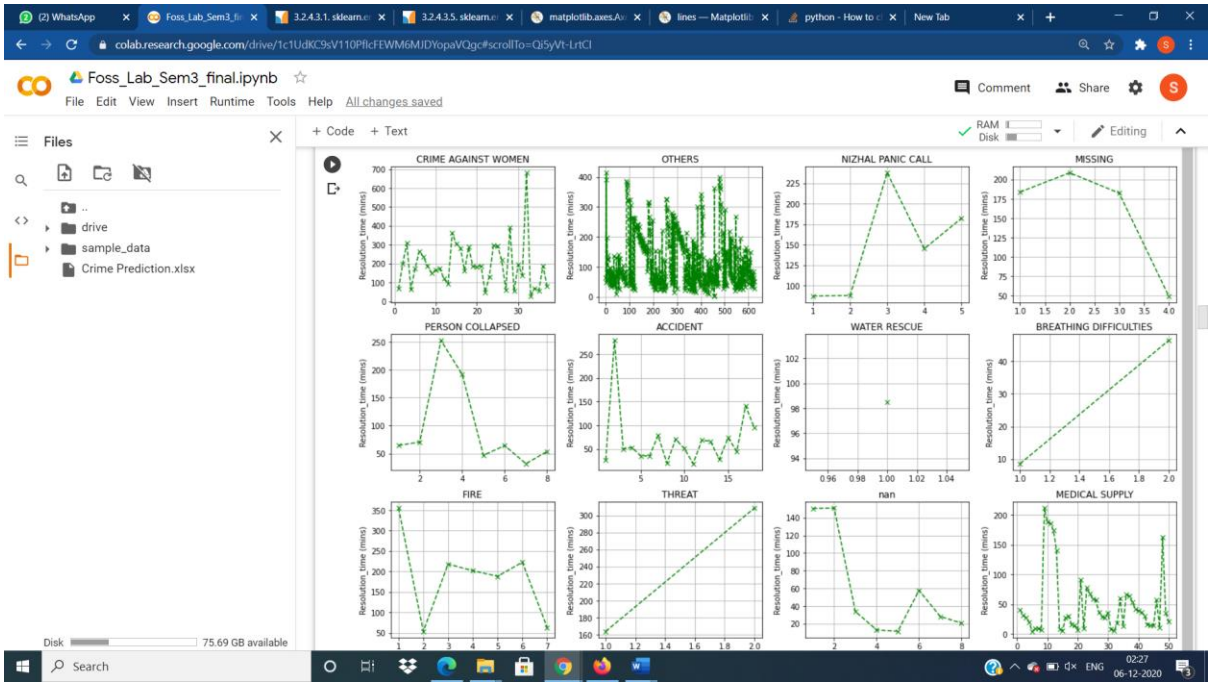
# Create geographical region of Kerala district with crime rate

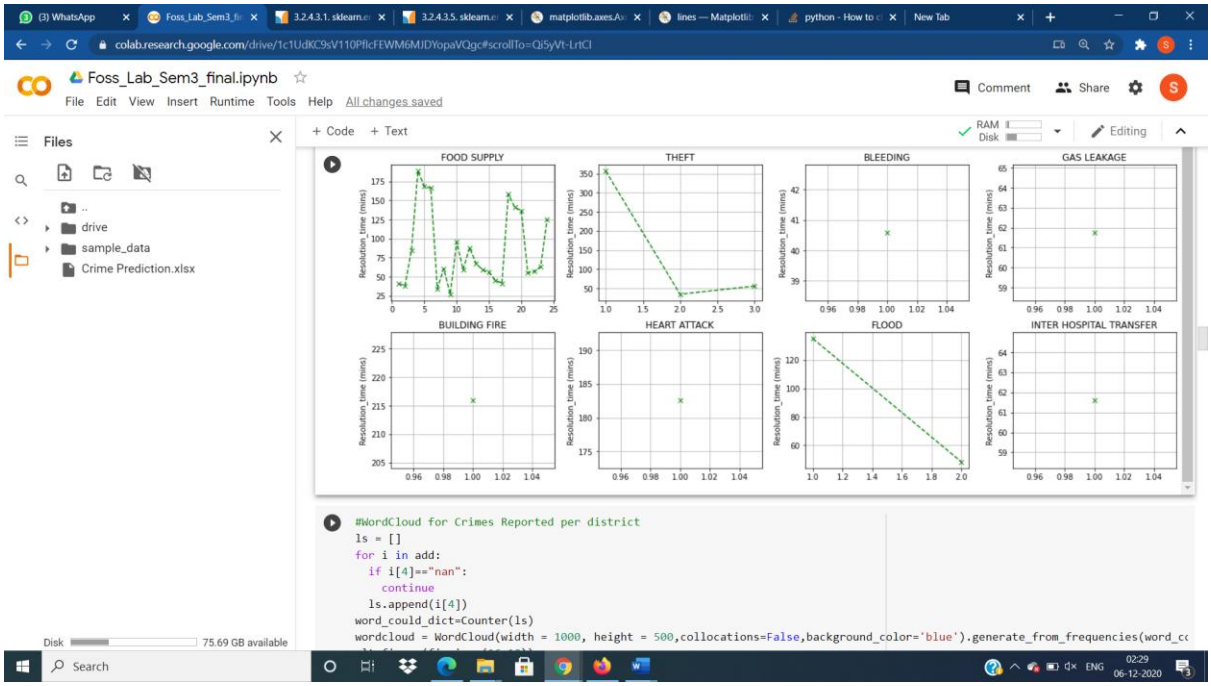


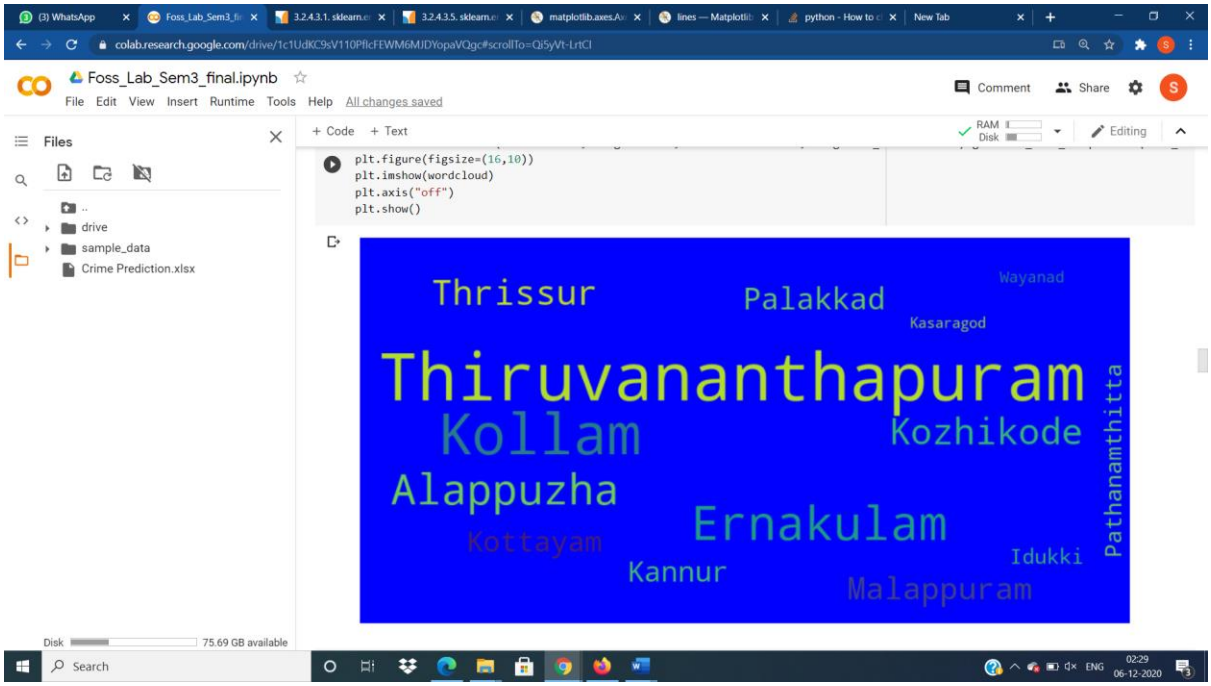
# Plot event  Vs Time graph



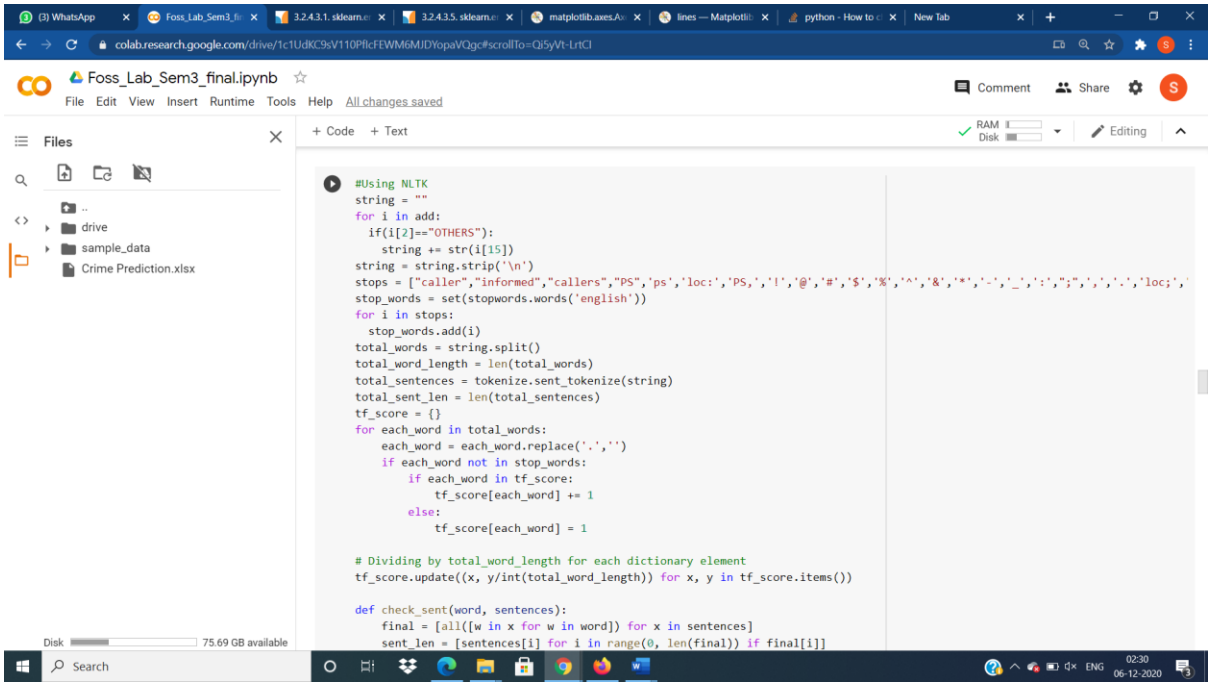





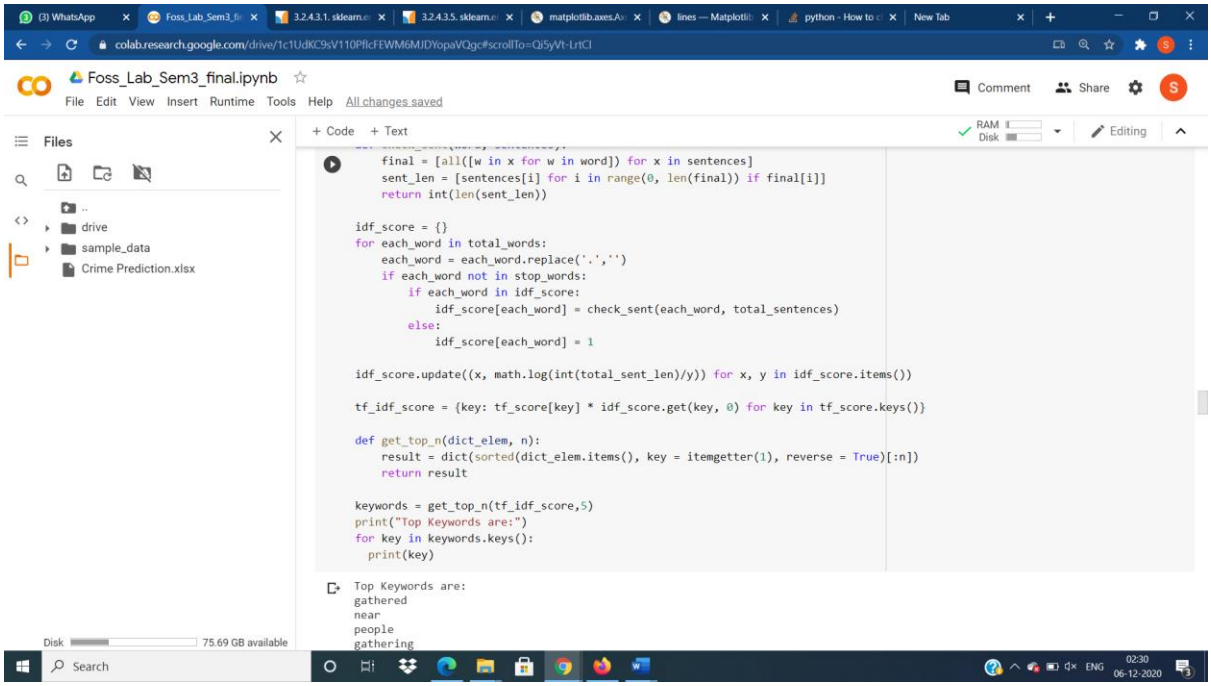

# Crime rate in Different District in Word Cloud



## Using NLTK we try to remove stope word



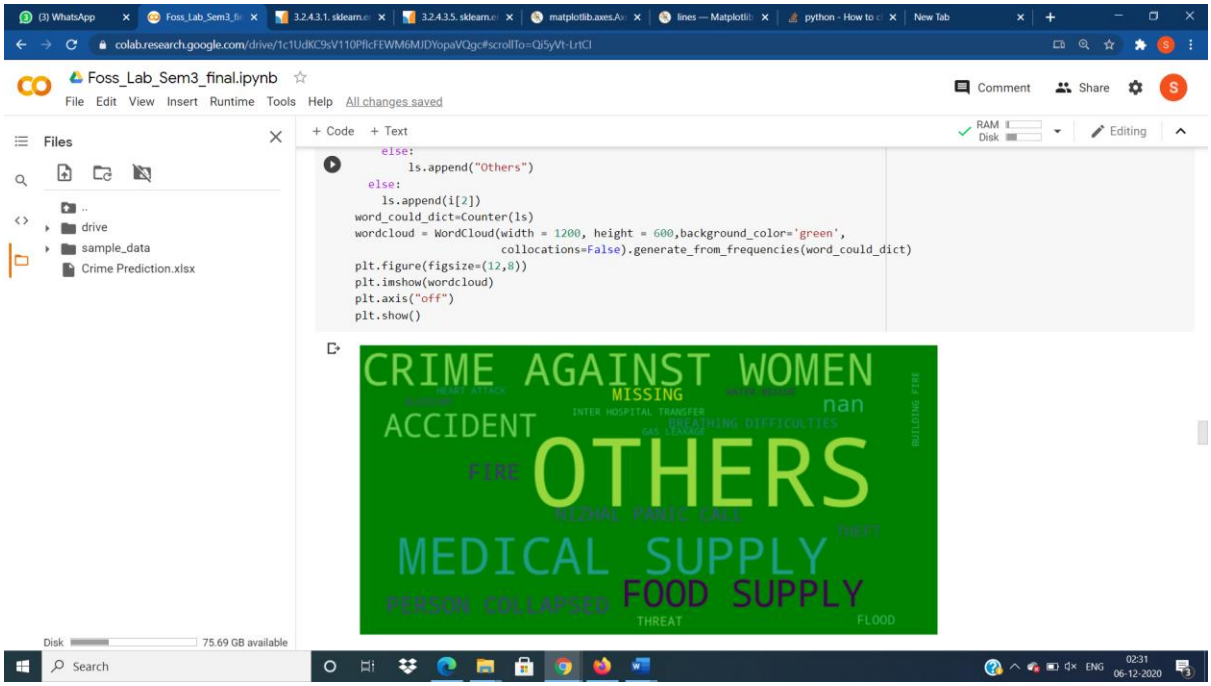
```
#Using NLTK
string = ""
for i in add:
  if(i[2]=="OTHERS"):
    string += str(i[15])
string = string.strip('\n')
stops = ["caller","informed","callers","PS","ps','loc:','PS,','!','@','#','$','%','^','&','*','-','_',':',';',',','.','loc;','
stop_words = set(stopwords.words('english'))
for i in stops:
  stop_words.add(i)
total_words = string.split()
total_word_length = len(total_words)
total_sentences = tokenize.sent_tokenize(string)
total_sent_len = len(total_sentences)
tf_score = {}
for each_word in total_words:
    each_word = each_word.replace('.','')
    if each_word not in stop_words:
        if each_word in tf_score:
            tf_score[each_word] += 1
        else:
            tf_score[each_word] = 1

# Dividing by total_word_length for each dictionary element
tf_score.update((x, y/int(total_word_length)) for x, y in tf_score.items())

def check_sent(word, sentences):
    final = [all([w in x for w in word]) for x in sentences]
    sent_len = [sentences[i] for i in range(0, len(final)) if final[i]]
```



```
    final = [all([w in x for w in word]) for x in sentences]
    sent_len = [sentences[i] for i in range(0, len(final)) if final[i]]
    return int(len(sent_len))

idf_score = {}
for each_word in total_words:
    each_word = each_word.replace('.','')
    if each_word not in stop_words:
        if each_word in idf_score:
            idf_score[each_word] = check_sent(each_word, total_sentences)
        else:
            idf_score[each_word] = 1

idf_score.update((x, math.log(int(total_sent_len)/y)) for x, y in idf_score.items())

tf_idf_score = {key: tf_score[key] * idf_score.get(key, 0) for key in tf_score.keys()}

def get_top_n(dict_elem, n):
    result = dict(sorted(dict_elem.items(), key = itemgetter(1), reverse = True)[:n])
    return result

keywords = get_top_n(tf_idf_score,5)
print("Top Keywords are:")
for key in keywords.keys():
  print(key)
```

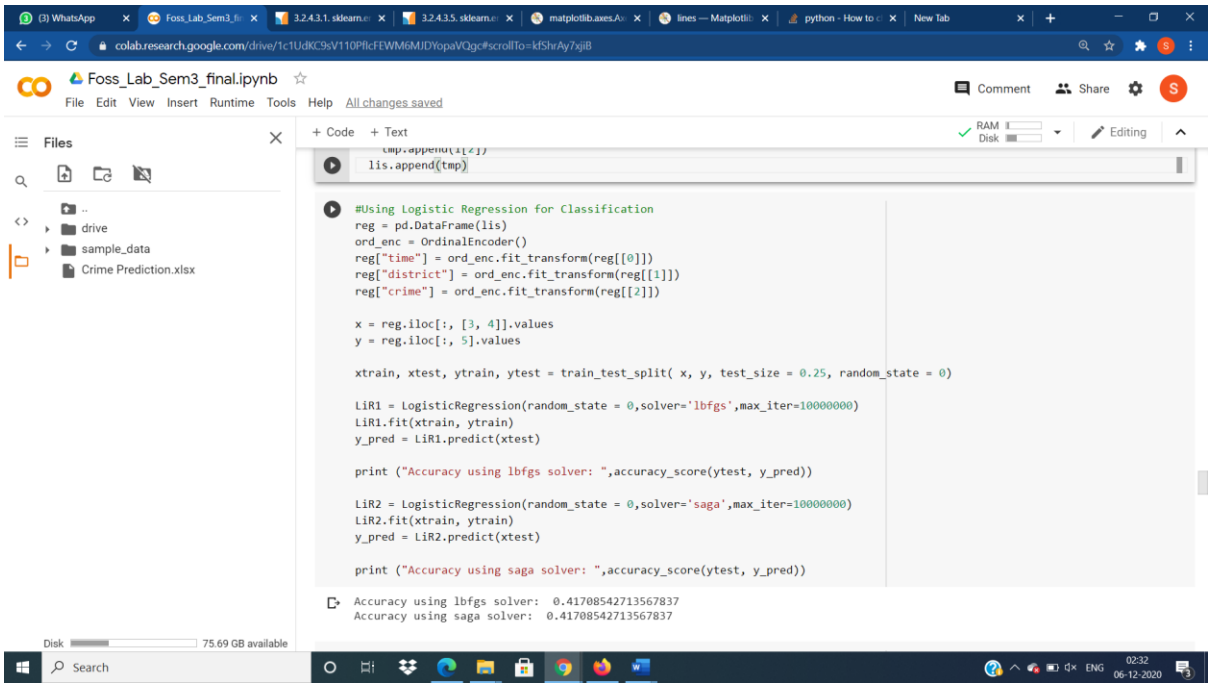
```
Top Keywords are:
gathered
near
people
gathering
```

Word Cloud Result of different event after using NLTk




## Logistic Regression

Using Logistic regression classifier and split the data set into 75 for training purpose 25 for test purpose and random state 0, create two classifier and use solver function 'lbfgs' & 'saga' for accuracy purpose.



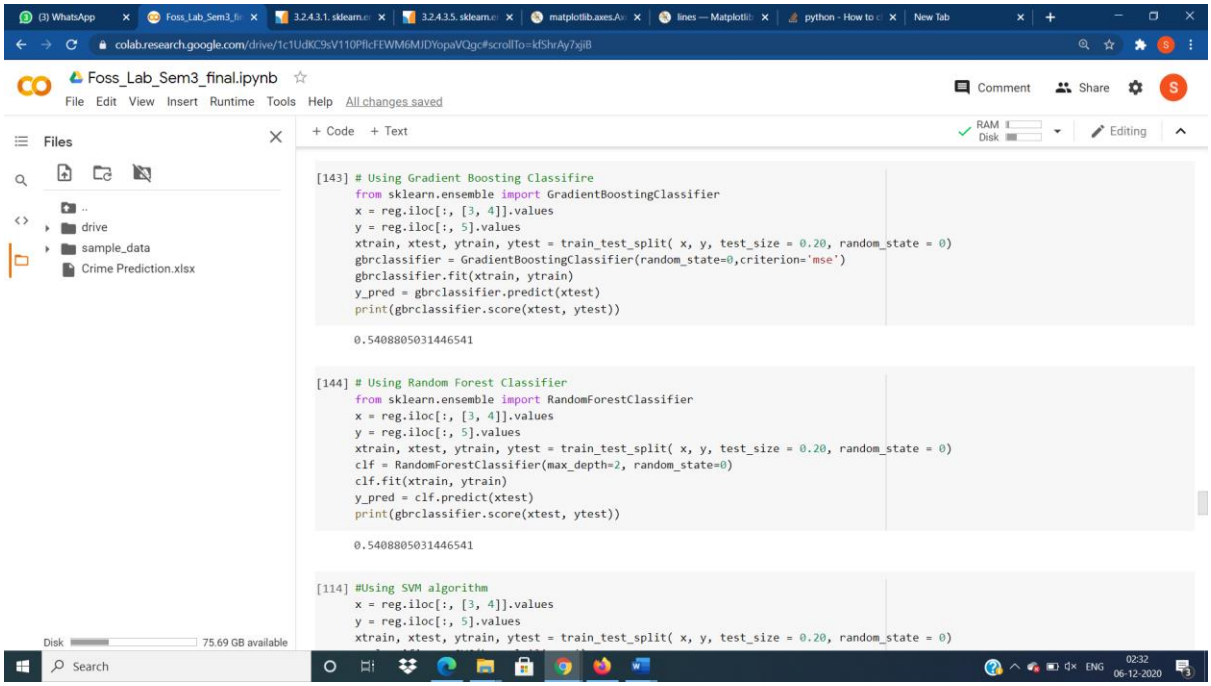

**Accuracy 41%**

## Gradient Boosting Classifier & Random Forest Classifier

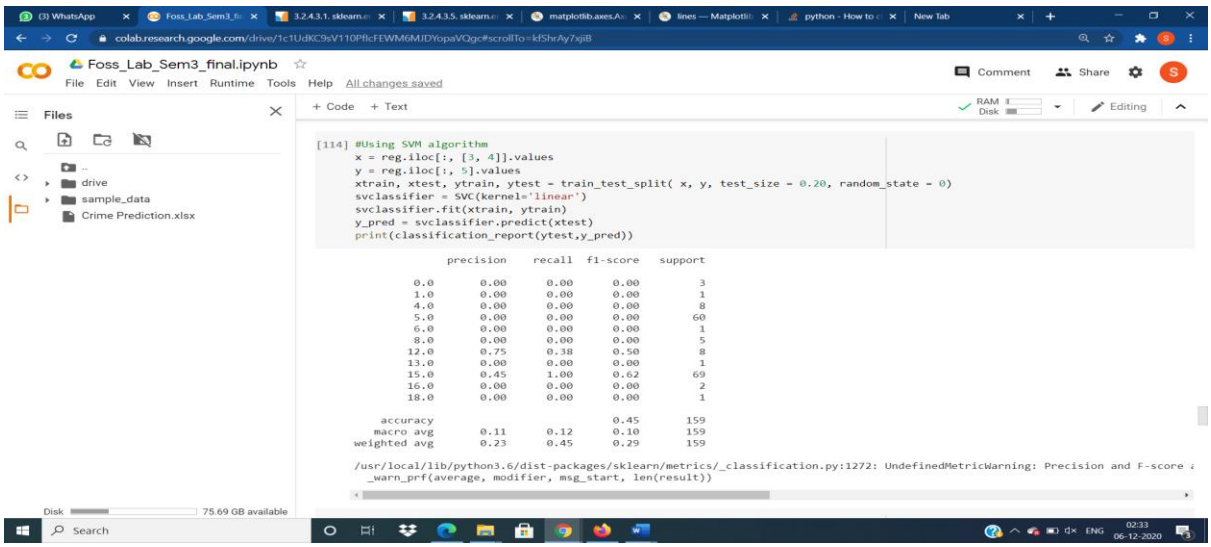
Data set split into 80 : 20 ratio for training and testing purpose and in gradient boosting classifier parameter Random state 0 and criterion ‘mse’ we got accuracy 54%

In random forest classifier

Parameter: max_depth=2,random state=0 Accuracy = 54%



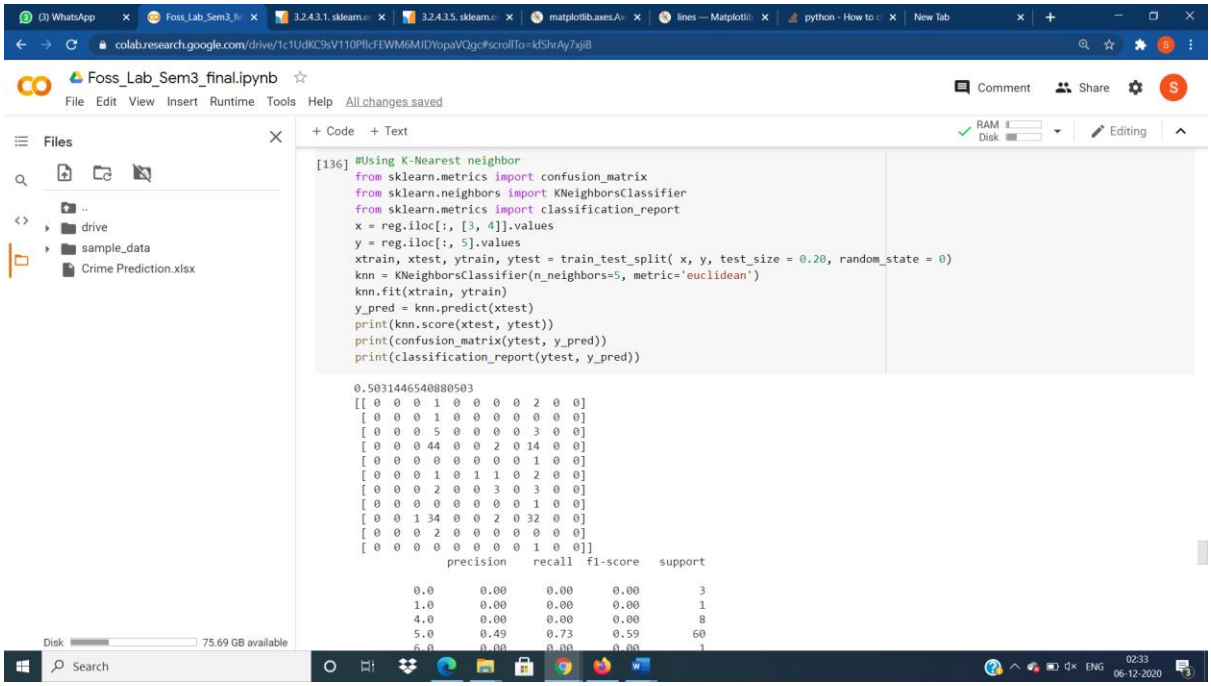

## Using SVM:




**Accuracy 45%**

## Using KNN Classifier

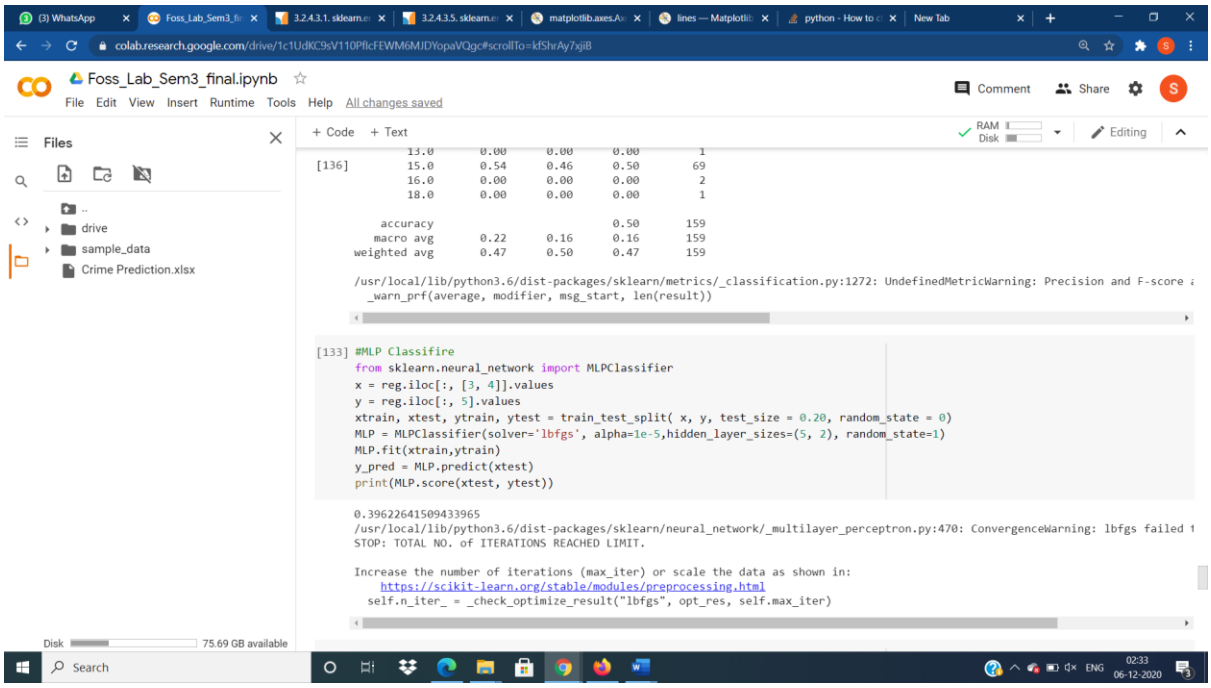
**Accuracy 50%**




## Using MLP Classifier:

In Parameter: solver= ‘lbfgs’ , hidden layer size=(5,2),Random state=0

**Accuracy 39%**

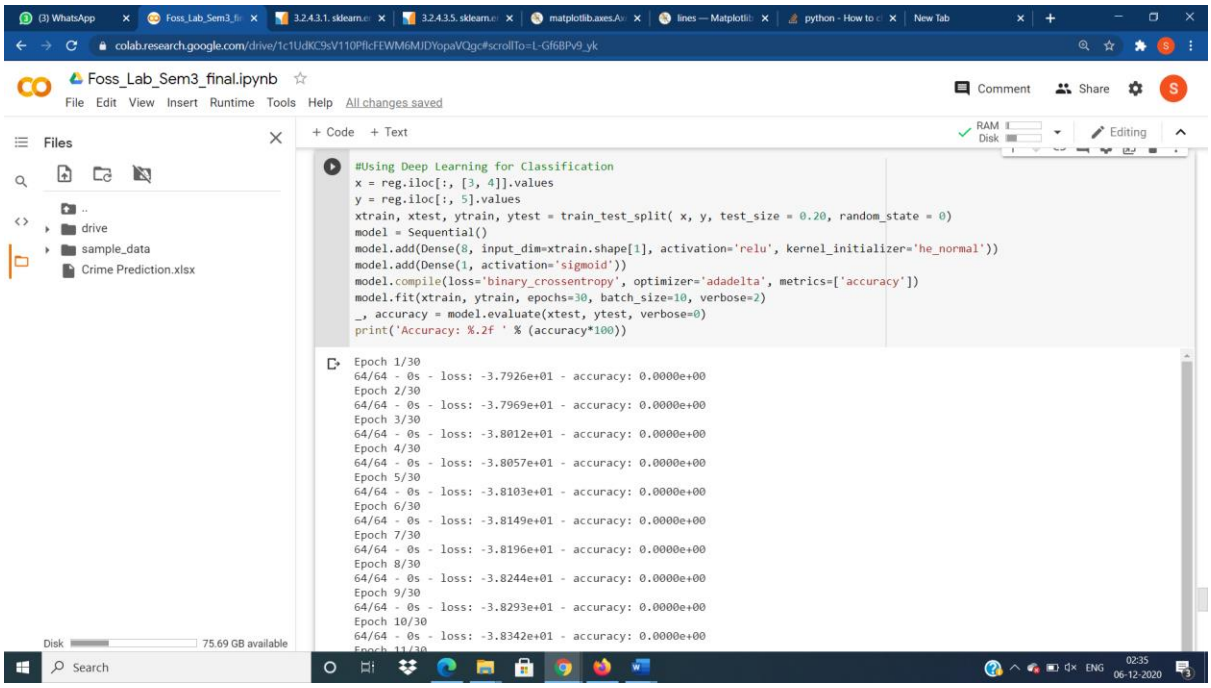


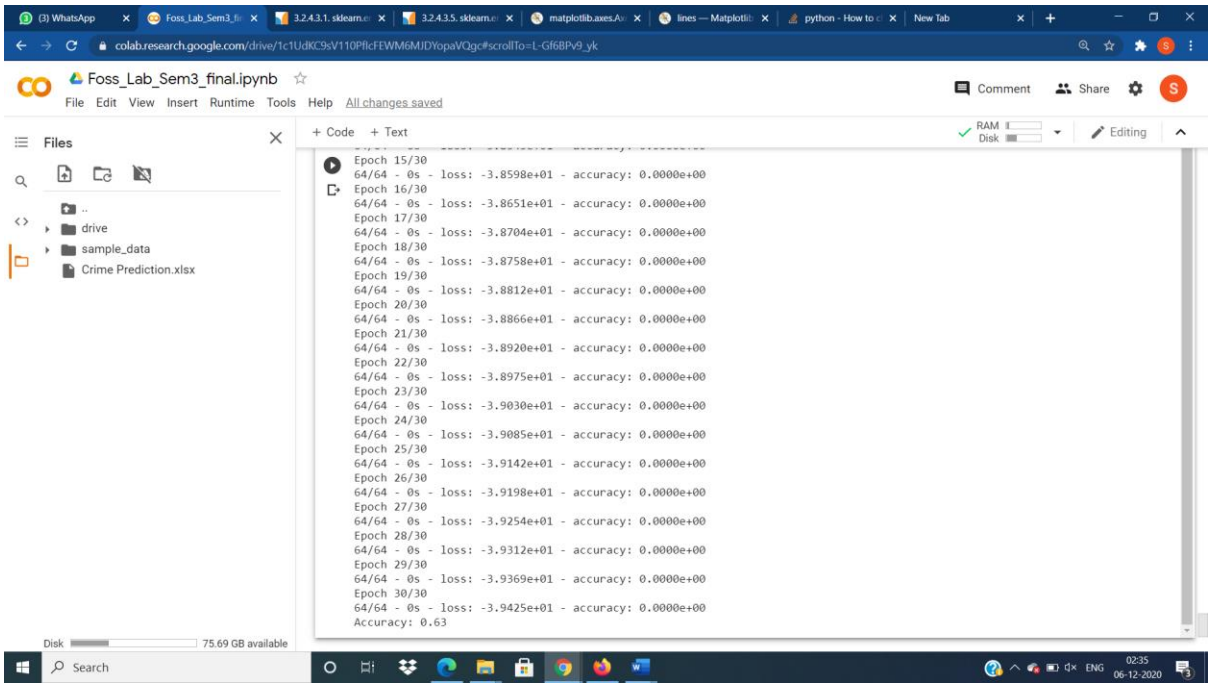

## Using Deep Learning :

Activation= ‘relu’ in dense layer 8 and Activation= ‘sigmoid’

Model.compile loss= ‘binary_crossentopy’  and optimizer= ‘adadelta’

Epoch=30 and batch size =10




**Accuracy 63%**