VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum – 590 018

A project report on

"Analyzing Tweets for Situational Preparedness"

submitted in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

INFORMATION SCIENCE & ENGINEERING

by

Aditya Kumar (1CR16IS007)
Rohit Ranu (1CR16IS084)
Piyush Raj (1CR16IS065)
Priti Singh (1CR16IS076)

Under the guidance of

Ms. Priyadharshini A
Asst. Professor
Dept. of ISE, CMRIT, Bengaluru

CMR INSTITUTE OF TECHNOLOGY
DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING
#132, AECS Layout, IT Park Road, Bengaluru-560037

2018-19

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum – 590 018

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Certificate

This is to certify that the project entitled, **"Analyzing Tweets for Situational Preparedness"**, is a bonafide work carried out by **Aditya Kumar (1CR16IS007), Rohit Ranu (1CR16IS084), Piyush Raj (1CR16IS065), and Priti Singh (1CR16IS076)** in partial fulfillment of the award of the degree of Bachelor of Engineering in Information Science & Engineering of Visvesvaraya Technological University, Belgaum, during the year 2018-19. It is certified that all corrections/suggestions indicated during reviews have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the said Degree.

Name & Signature of Guide	Name & Signature of HOD	Signature of Principal
(Ms. Priyadharshini A)	(Dr. M Farida Begum)	(Dr. Sanjay Jain)

External Viva

Name of the Examiners	**Signature with date**
1.	
2.	

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum – 590 018

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Certificate

We, **Aditya Kumar (1CR16IS007), Rohit Ranu (1CR16IS084), Piyush Raj (1CR16IS065), and Priti Singh (1CR16IS076) ,** bonafide students of CMR Institute of Technology, Bangalore, hereby declare that the report entitled **"Analyzing Tweets for Situational Preparedness"** has been carried out by us under the guidance of **Ms. Priyadharshini A, Assistant Professor,** CMRIT Bangalore, in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in **Information Science Engineering**, of the Visvesvaraya Technological University, Belgaum during the academic year 2018-2019. The work done in this dissertation report is original and it has not been submitted for any other degree in any university.

Aditya Kumar (1CR16IS007)
Rohit Ranu (1CR16IS084)
Piyush Raj (1CR16IS065)
Priti Singh (1CR16IS076)

ABSTRACT

The threat monitor, gathers tweets from a set of Twitter accounts, filters them to target solely the monitored infrastructure, classifies remaining tweets as either relevant or not, aggregates tweets related to same threat using a stream clustering approach, and generates indicators of compromise suitable for threat sharing platforms. The results obtained from the model design experiments revealed that using a single classification model for the complete monitored infrastructure is preferable to using an ensemble of models for different infrastructure parts.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible, success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So, it is with gratitude that we acknowledge all those whose guidance encouragement served as beacon of light and crowned our effort with success.

We would like to thank **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore, for providing an excellent academic environment in the college and his never-ending support for the B.E program.

We would like to express our gratitude towards **Dr. Farida Begaum,** Assoc Professor and HOD, Department of Information Science and Engineering CMRIT, Bangalore, who provided guidance and gave valuable suggestions regarding the project.

We consider it a privilege and honour to express our sincere gratitude to our internal guide **Ms. Priyadharshini A,** Assoc. Professor, Department of Information Science & Engineering and **Mr. Sudhakar K N**, Assoc. Professor, Department of Information Science and Engineering, CMRIT, Bangalore for their valuable guidance throughout the tenure of this project work.

We would also like to thank all the faculty members who have always been very Co-operative and generous. Conclusively, we also thank all the non-teaching staff and all others who have done immense help directly or indirectly during our project.

ADITYA KUMAR

ROHIT RANU

PIYUSH RAJ

PRITI SINGH

TABLE OF CONTENTS

Chapter 1

PREAMBLE

Introduction:

A security analyst must be aware of the latest developments regarding updates, patches, mitigation measures, vulnerabilities, attacks, and exploits to protect an infrastructure adequately. Awareness should raise within the Security Operations Center (SOC) through Security Information and Event Management (SIEM) software, thereby allowing correlating the latest information available with infrastructure events. There are two primary ways of obtaining cybersecurity news. One is to purchase a curated feed from a specialised company such as SenseCy . Another, is to collect Open Source Intelligence (OSINT) publicly available from various sources on the internet (e.g., Threat post). Although there are numerous so-called threat intelligence tools (e.g., SpiderFoot, IntelMQ , and AlienVault OTX), their main focus is on collecting security-related OSINT.

From a wide variety of sources. At most, they apply simple keyword based queries and filters to decrease the big volume of information but do not provide more elaborated processing or analysis. To overcome the limitations of keyword-based methods these tools have to be adapted or extended, configured, and possibly, complemented by the end user. However, recent work demonstrates that different types of useful information and Indicators of Compromise (IoC) can be obtained from OSINT if more sophisticated analysis techniques are applied. These results highlight the gap between the current capabilities of existing OSINT-based tools and the potential that OSINT can raise. To fill this gap, we considered the problem of designing an OSINT-based tool to keep SOC analysts aware of the threat landscape against the infrastructures under their responsibility. Besides choosing sources and collecting OSINT, this requires selecting only the information related to the security of the monitored infrastructure assets and joining identical information published by different sources. Given the volume of information, this is a time-consuming task for which security analysts have a limited time budget, even though the quality of their work depends on this awareness.

To this end, we developed a Twitter-based streaming threat monitor called SYNAPSE. SYNAPSE's pipeline is composed of filtering, feature extraction, binary classification, aggregation, and generation of indicators of compromise. More specifically, an automated tool gathers tweets from security related accounts, a supervised machine learning technique selects those relevant for the specified infrastructure being monitored, and a clustering method avoids presenting repeated or unnecessary information. Twitter was chosen for two main reasons. First, Twitter is well-recognised as a relevant source of short notices (almost in real-time) about web activity and occurring events. Second, the limited size of a tweet makes it simple to process through general-purpose machine learning approaches, which enable low error levels across multiple domains of application. Furthermore, although short, tweets provide enough elements to categorise their content, as well as links for more detailed material.

Chapter 2

LITERATURE SURVEY

A. Data Collection

The data collector module requires a set of accounts; from which it will collect every tweet posted using Twitter's stream API. These accounts can be from security analysts and companies, vendors, hackers, researchers, among others. They are chosen considering the likelihood of users tweeting about the security of elements belonging to the monitored IT infrastructure. Since most security analysts already follow OSINT sources (including Twitter accounts), it is just a matter of including them in the account set to tailor and improve OSINT coverage. We opted to collect tweets from selected accounts instead of using a keyword-based approach since the latter is likely to retrieve large amounts of irrelevant information, an approach already in use in the literature. For instance, tweets with the word "windows" include all Windows-related topics (the OS) and all tweets referring glass windows, besides other non-security related topics. By collecting tweets only from selected security-related accounts, a more substantial fraction of tweet are related to IT security, leaving the focus on filtering tweets not mentioning threats related to the specified IT infrastructure.

B. Filtering

Despite the account-based collection approach, most likely the collected data will include tweets unrelated to the infrastructure under the analyst's care. These have to be dropped by a filter. The filtering approach assumes that a tweet referring a threat to a particular IT infrastructure asset has to metaset properties. Therefore, a second input is required: a set of keywords describing the assets of the monitored IT infrastructure. Only tweets that include at least one of the keywords will pass the filter. Keywords further restrict the scope of the security events, hence decreasing the number of irrelevant tweets beyond the filter. To maximize the effectiveness of the tool, the keywords defining the monitored assets must be as complete as possible .Keywords can, in part, be learned from logs and network traffic within the infrastructure, but have to be complemented with slang and other informal terms The complete details of the proposed system is given below commonly used by IT professionals. Although that work is out of the scope of this article,

this process may be substantially automated. For example, if the analyst is in charge of securing a Linux cluster running virtual machines to serve a web
service with a database, the keyword set could be {linux, ssh, virtualbox, vbox, mysql, apache http, php}. In principle, the more complete and specific the keyword set, the more effective the filtering process will be. For example, instead of using only “Linux”, the analyst could also specify the specific distribution in use.

C. Pre-processing and Feature Extraction

Pre-processing aims to normalise the tweet representation. First, all characters are converted to lower case, and stop words and hyperlinks are removed (these offer little information since they are shortened URLs).Numbers, dots, and hyphens are replaced by their textual representation (e.g., “2” to “two”), as these are relevant to
distinguish software versions (e.g., Mozilla Firefox 4.5.1-2). Finally, all non [a-z] characters are removed. For instance, after pre-processing, the tweet,

#Oracle #Linux6/7: Unbreakable Enterprise kernel (ELSA-2016-3573) https://t.co/vLTel8NodG #Nessus becomes: oracle linux six seven unbreakable enterprise kernel else a hyphen two thousand and sixteen hyphen three thousand five hundred and seventy-three Nessus

Before classification tweets must be converted to a numerical format, thus becoming suitable for supervised learning binary classification techniques. This work uses the well-known Term Frequency - Inverse Document Frequency (TFIDF)method. TF-IDF computes weights to words (features) based on their occurrence frequency in each specific document and group of documents considered. The weight of a word increases with its frequency of occurrence in a single document but is scaled down by the frequency of occurrence in all documents. By mapping each consecutive word token to a corresponding vector position, tweets are converted to a constant size, zero-padded, TF-IDF numeric vector. Finally, to limit the size of the vector we employ the hashing trick technique.

D. Classification

For the classification of tweets according to their security relevance, two classifiers were tested: Support Vector Machines (SVM) and Multi-Layer Perceptron (MLP) \Neural Networks (NN). The SVM is a broadly used classifier achieving good results across a multitude of application domains. In this paper we consider the linear version of SVM. The MLP is a well-established and frequently used NN architecture that has a long track record of good and consistent results over a vast
number of classification tasks.

E. Clustering

An aggregation phase should be included in threat intelligence tools to remove duplicate information and provide a concise summary of the current threat landscape. To perform such summarization, we have proposed a clustering algorithm. Clustering is a data-exploration technique designed to find groups of similar items within a set, and therefore is a natural choice for the problem of finding tweets discussing the same threats. SYNAPSE clustering stage relies on a stream clustering methodology supported by the k-means algorithm. k-means is a a widely used algorithm that has provided good efficiency and empirical success over the last 50 years.F. MISP Compatible IoC Generation After the clustering phase, the clusters of tweets should be transformed into the IoC format to allow their inclusion name keywords TABLE I: The infrastructure designed for tweet collection and altering and its subdivision in four coherent parts.in SIEMs or threat intelligence platforms. There are several standards for sharing IoCs, such as STIX or MISP. The format must be extendible and adaptable as tweets are unstructured and contain unpredictable content. For these reasons the MISP format has been selected to generate IoCs . Moreover, it can be easily converted into other standard formats like STIX. To ease the correlation of the collected threats within the threat intelligence platforms, events should be categorized using tags, added according to a set of threat categories. The categories can be adapted from existing taxonomies, such as those from ENISA and VERIS for cyberthreats.in SIEMs or threat intelligence platforms. There are several standards for sharing IoCs, such as STIX or MISP. The format must be extendible and adaptable as tweets are unstructured and contain unpredictable content. For these reasons the MISP format has been selected to generate IoCs. Moreover, it can be easily converted into other standard formats like STIX. To ease the correlation of the collected threats within the threat intelligence platforms, events should be categorized using tags, added according to a set of threat

categories. The categories can be adapted from existing taxonomies, such as those from ENISA and VERIS for cyber threats.

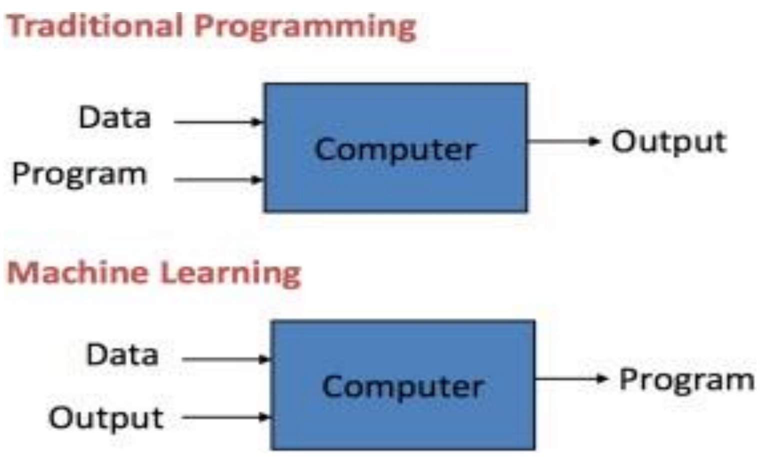
Chapter 3

Theoretical Background

3.1 Overview on Machine Learning

Machine learning is an application of artificial intelligence (AI) that gives systems the ability to automatically learn and evolve from experience without being specially programmed by the programmer. 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 main aim of machine learning is to allow computers to learn automatically and adjust their actions to improve the accuracy and usefulness of the program, without any human intervention or assistance. Traditional writing of programs for a computer can be defined as automating the procedures to be performed on input data in order to create output artefacts. Almost always, they are linear, procedural and logical. A traditional program is written in a programming language to some specification, and it has properties like:

• We know or can control the inputs to the program.
• We can specify how the program will achieve its goal.
• We can map out what decisions the program will make and under what conditions it makes them.



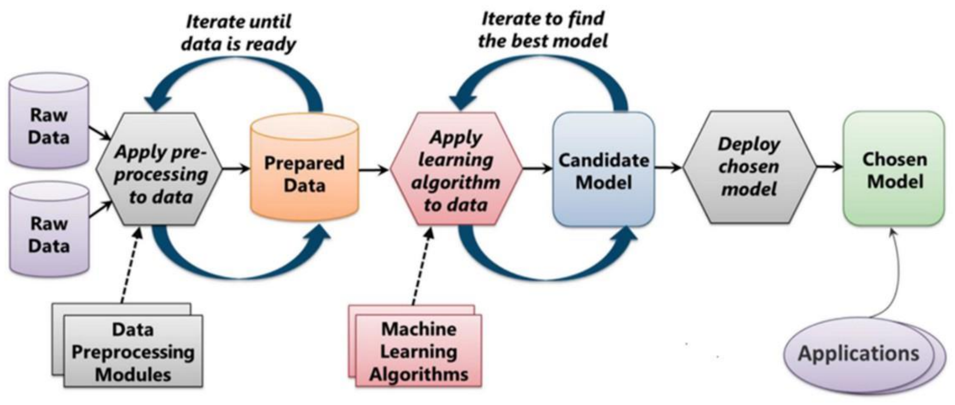
• Since we know the inputs as well as the expected outputs, we can be confident that the program will achieve its goal.

Traditional programming works on the premise that, as long as we can define what a program needs to do, we are confident we can define how a program can achieve that goal. This is not always the case as sometimes, however, there are problems that you can represent in a computer that you cannot write a traditional program to solve. Such problems resist a procedural and logical solution. They have properties such as:

• The scope of all possible inputs is not known beforehand.
• You cannot specify how to achieve the goal of the program, only what that goal is.
• You cannot map out all the decisions the program will need to make to achieve its goal.
• You can collect only sample input data but not all possible input data for the program.

Problems like this resist traditional programmed solutions because manually specifying a solution would require a disproportionate amount of resources. Furthermore, when new inputs arise, the rules may change, thereby necessitating changes to the program. In such cases as these, machine learning might be the optimum approach to use in deriving a solution to the way the problem is represented on the computer. Machine learning techniques can be broadly categorized into the following types: Supervised learning takes a set of feature/label pairs, called the training set. From this training set the system creates a generalised model of the relationship between the set of descriptive features and the target features in the form of a program that contains a set of rules. The objective is to use the output program produced to predict the label for a previously unseen, unlabelled input set of features, i.e. to predict the outcome for some new data. Data with known labels, which have not been included in the training set, are classified by the generated model and the results are compared to the known labels. This dataset is called the test set. The accuracy of the predictive model can then be calculated as the proportion of the correct predictions the model labelled out of the total number of instances in the test set. Unsupervised learning takes a dataset of descriptive features without labels as a training set. In unsupervised learning, the algorithms are left to themselves to discover interesting structures in the data. The goal now is to create a model that finds some hidden structure in the dataset, such as natural clusters or associations. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabelled data. The system does not figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabelled data. Unsupervised learning can be used for clustering, which is used to discover any inherent grouping that are already present in the data. It can also be used for association problems, by creating rules based on the data and finding relationships or associations between them. Semi-supervised machine learning falls somewhere in between supervised and unsupervised learning, since they use both labelled and unlabelled data for training typically a small amount of labelled data and a large amount of unlabelled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi supervised learning is chosen when the acquired labelled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring labelled data generally does not 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 behaviour 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

The Machine Learning Process


Iterate until data is ready
Iterate to find the best model
Raw Data
Raw Data
Apply pre-processing to data
Prepared Data
Apply learning algorithm to data
Candidate Model
Deploy chosen model
Chosen Model
Data Preprocessing Modules
Machine Learning Algorithms
Applications

From "Introduction to Microsoft Azure" by David Chappell

.

3.2 Machine Learning Tools

There are many different software tools available to build machine learning models and to apply these models to new, unseen data. There are also a large number of well defined machine learning algorithms available. These tools typically contain libraries implementing some of the most popular machine learning algorithms. They can be categorized as follows:

• Pre-built application-based solutions.

• Programming languages which have specialized libraries for machine learning.

The first option includes well-known frameworks such as WEKA, Knime Analytics Platform, RapidMiner Studio and SAS Enterprise Miner. WEKA is open source and very popular among researchers and developers. It is used and referred to in many of the research papers found in the literature. Using WEKA it is possible to train and evaluate model very quickly. Despite its ease of use however, the WEKA and other frameworks do not show how they work or how they obtained the given outcome. In other words, most of these frameworks are black boxes which limit or outright stop user experimentation or improvements. This also means that it is difficult to analyze and understand the results. Using programming languages to develop and implement models is more flexible and

gave us better control of the parameters to the algorithms. It also allows us to have a better understanding of the output models produced. Some of the popular \programming languages used in the Fifield of machine learning are:

• **Python**: Python is an extremely popular choice in the Fifield of machine learning and AI development. Its short and simple syntax make it extremely easy to learn and use. It supports both object oriented and functional oriented styles of programming as well. In addition to this, Python has a number of libraries which have been developed for machine learning, such as Numpy, Pandas and SciKit-learn.

• **R**: R is one of the most effective and efficient languages for analysing and manipulating data in statistics. Using R, we can easily produce well-designed publication-quality plot, including mathematical symbols and formulae where needed. Apart from being a general purpose language, R has numerous of packages like RODBC, Gmodels, Class and Tm which are used in the field of machine learning. These packages make the implementation of machine learning algoeasy, for cracking the business associated problems.

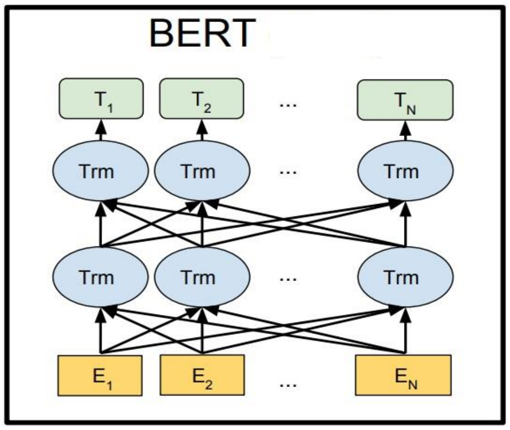
• **MATLAB**: MATLAB is a programming language developed by MathWorks. Created primarily for numerical computing, MATLAB is also extremely popular among machine learning programmers. It is heavily used in statistical analysis and complex systems. MATLAB excels at handling matrices, making it especially useful in image recognition. It is an extremely versatile language which allows matrix manipulations, implementation of algorithms and creation of user interfaces. It is also able to interface with other programming languages like C, Fortran and Java.

.

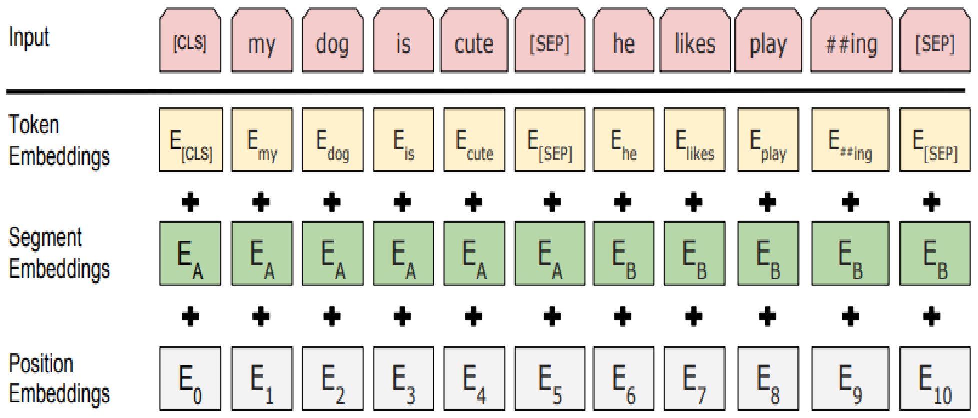
• **Scala**: Scala is a general purpose programming language which supports functional programming and strong static typing. It is easy to build scalable big data applications in the path provided by Scala with regard to complexity and data size. There are many well-designed libraries in Scala that are suitable for linear algebra, random number generation, and scientific computing. The standard scientific library contains special functions such as numerical algebra, non-uniform random generation, and many others. The data library supported by Scala, Saddle provides a solid foundation for manipulation of data through array-backed support, robustness to missing values, automatic data alignment, and 2D data structures.

• **BERT**: BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models, BERT is meant to pretrain deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context layers. As a result, the pre-trained BERT model is finetuned with only one additional output layer to make state-of-the-art models for a good range of tasks, like question answering and language inference, without substantial task specific architecture modifications.

BERT has dramatically accelerated NLU like anything and Google's move to open source BERT has probably changed natural language processing forever. The machine learning ML and NLP communities are very excited about BERT as it takes a huge amount of heavy lifting out of their hand being able to carry out research in natural language. It has been pre-trained on tons of words – and on the entire of English Wikipedia 2,500 million words.



BERT is a technology to generate "contextualized" word embeddings/vectors, which is its biggest advantage but also it's biggest disadvantage as it is very compute-intensive at inference time, meaning that if you want to use it in production at scale, it can become costly.


Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
E[CLS] Emy Edog Eis Ecute E[SEP] Ehe Elikes Eplay E##ing E[SEP]
Segment Embeddings
EA EA EA EA EA EA EB EB EB EB EB
Position Embeddings
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10

BERT models to both ranking and featured snippets in Search, we're able to do a much better job helping you find useful information. In fact, when it comes to ranking results, BERT will help Search better understand one in 10 searches in the U.S. in English.

BERT is pre-trained on two NLP tasks:

1.Masked Language Modeling

2.Next Sentence Prediction

Masked LM (MLM)

Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. In technical terms, the prediction of the output words requires:

1. Adding a classification layer on top of the encoder output.
2. Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension.
3. Calculating the probability of each word in the vocabulary with softmax.

Next Sentence Prediction (NSP): BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence. To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model:

1. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
2. A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.
3. A positional embedding is added to each token to indicate its position in the sequence. The concept and implementation of positional embedding are presented in the Transformer paper.

To predict if the second sentence is indeed connected to the first, the following steps are performed:

- The entire input sequence goes through the Transformer model.

- The output of the [CLS] token is transformed into a 2×1 shaped vector, using a simple classification layer (learned matrices of weights and biases).
- Calculating the probability of IsNextSequence with softmax.

When training the BERT model, Masked LM and Next Sentence Prediction are trained together, with the goal of minimizing the combined loss function of the two strategies.

1. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
2. A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.
3. A positional embedding is added to each token to indicate its position in the sequence. The concept and implementation of positional embedding are presented in the Transformer paper.

To predict if the second sentence is indeed connected to the first, the following steps are performed:

- The entire input sequence goes through the Transformer model.
- The output of the [CLS] token is transformed into a 2×1 shaped vector, using a simple classification layer (learned matrices of weights and biases).
- Calculating the probability of IsNextSequence with softmax.

Chapter 4

SYSTEM REQUIREMENT

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. **Why SRS?**In order to fully understand ones project, it is very important that they come up with a SRS listing out their requirements, how are they going to meet it and how will they complete the project. It helps the team to save upon their time as they are able to comprehend how are going to go about the project. Doing this also enables the team to find out about the limitations and risks early on.SRS also functions as a blueprint for completing a project with as little cost growth as possible. SRS is often referred to as the parent document because all subsequent project management documents, such as design specif`ications, statements of work, software architecture specification, testing and validation plans, and documentation plans, are related to it.

4.1 Functional Requirement

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. Following are the functional requirements on the system:

1. All the data must be in the same format as a structured data.
2. The data collected will be vectorised and sent across to the classifier.
3. Based on the given data, the classifier given us a classification and provides us with accuracy.

4.2 Non Functional Requirements

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviours. They may relate to emergent system properties such as reliability, response time and store. Non-functional requirements arise through the user needs, because of budget constraints, organizational policies and the need for interoperability with other software and hardware systems.

Chapter 5

SYSTEM ANALYSIS

Analysis is the process of fifinding the best solution to the problem. System analysis is the process by which we learn about the existing problems, defifine objects and requirements and evaluates the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

5.1 Feasibility Study

All systems are feasible when provided with unlimited resource and infifinite time. It is a formally documented output that summarizes results of the analysis and evaluations conducted to review the proposed solution and investigate project alternatives for the purpose of identifying if the project is really feasible, cost-effffective and profifitable. It describes and supports the most feasible solution applicable to the project. So it is both necessary and prudent to evaluate the feasibility of the system at the earliest possible time.If project risk is great, the feasibility of producing quality software is reduced.In this case there are three key considerations involved in the feasibility analysis.

5.1.1 Economical Feasibility

As we all know hospitals are expensive and going to it cannot be ignored if you are sick. Going to a hospital and getting tested regularly is very costly. This code helps us to early identify the disease within fifirst few visits. The money required for multiple testing is saved.

5.1.2 Technical Feasibility

The algorithm takes data from multiple sources and brings it down to a few specific categories. This gives accurate information of the condition of the patient and the accurate output if he/she has the disease or not.

5.1.3 Social Feasibility

As the machine does most of the analysing part of the data generated. The doctors and staff is free to handle other more necessary tasks in hand and it helps in betterment of the health care society.

5.2 Analysis

5.2.1 Technical Analysis

The performance of the system can be increased if the technical analysis is done well. The systems hardware requirements must be taken into consideration. The software must go hand in hand with the hard ware else the efficiency of the system deteriorates.

1. **Changes to bring in the system:** All changes should be in positive direction, there will be increased level of efficiency and better customer service.
2. **Required skills:** Platforms tools used this project are widely used.
3. **Acceptability:** The structure of the system is kept feasible enough so that there should not be any problem from the users point of view.

5.2.2 Economical Analysis

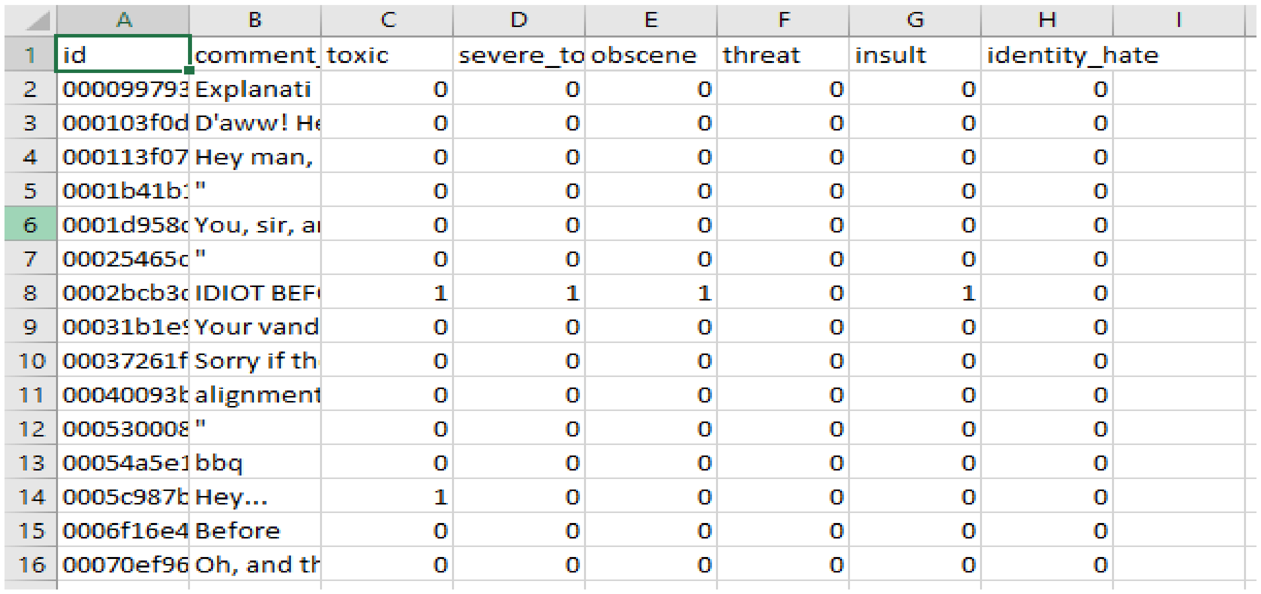
there are about 44 million people worldwide suffering from Alzheimer’s disease. Only one out of four diseased gets diagnosed and gets treatment on time. This could hinge with such techniques coming up and helping the needful. We will have a huge economic surge in the medicine industry as many patients will be finding out in early stages.

Chapter 6

IMPLEMENTATION

The following snapshots define the implementation of our project

Dataset downloaded from Kaggle website

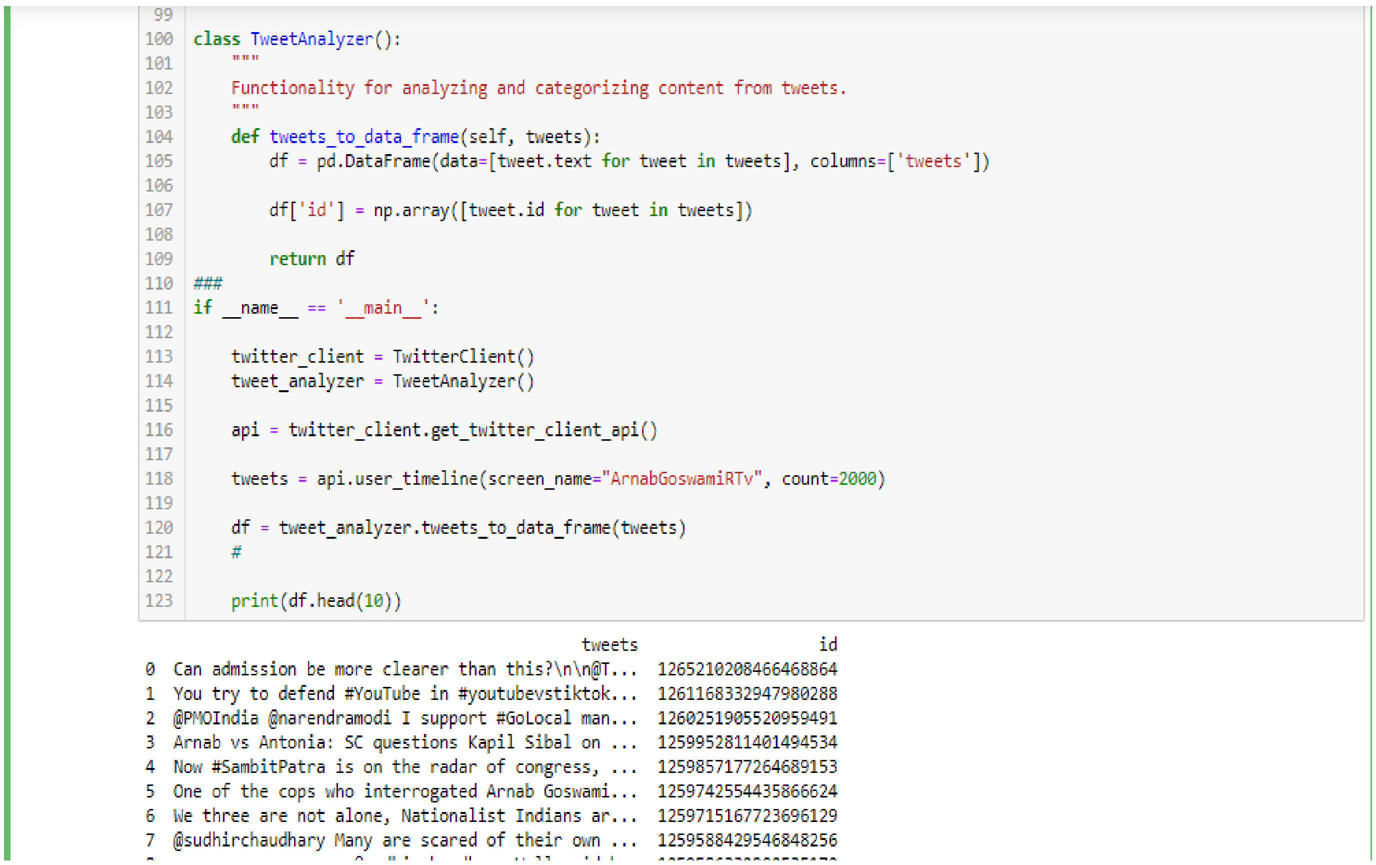


	A	B	C	D	E	F	G	H	I
1	id	comment_	toxic	severe_to	obscene	threat	insult	identity_hate	
2	000099793	Explanati	0	0	0	0	0	0	
3	000103f0d	D'aww! He	0	0	0	0	0	0	
4	000113f07	Hey man,	0	0	0	0	0	0	
5	0001b41b1	"	0	0	0	0	0	0	
6	0001d958c	You, sir, ar	0	0	0	0	0	0	
7	00025465c	"	0	0	0	0	0	0	
8	0002bcb3c	IDIOT BEF	1	1	1	0	1	0	
9	00031b1e9	Your vand	0	0	0	0	0	0	
10	00037261f	Sorry if th	0	0	0	0	0	0	
11	00040093b	alignment	0	0	0	0	0	0	
12	000530008	"	0	0	0	0	0	0	
13	00054a5e1	bbq	0	0	0	0	0	0	
14	0005c987b	Hey...	1	0	0	0	0	0	
15	0006f16e4	Before	0	0	0	0	0	0	
16	00070ef96	Oh, and th	0	0	0	0	0	0	

Libraries used:

- **Pandas** It is a data manipulation tool. It is built on the Numpy package and its key data structure is called the DataFrame.
- **DataFrames** allows us to store and manipulate tabular data in rows and columns.
- **NumPy**: It is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy mathematical and logical operations on arrays can be performed.

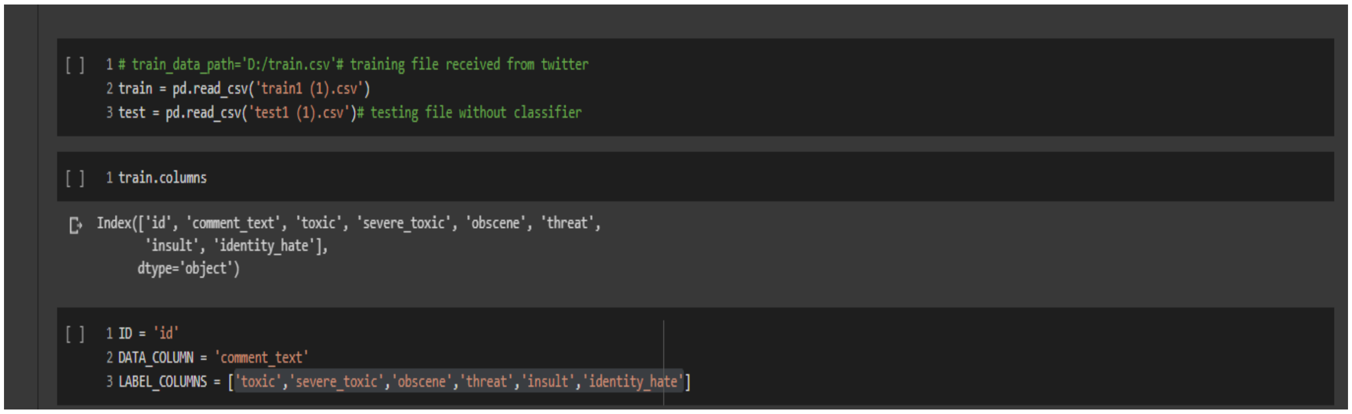
We downloaded the tweets using tweepy and created a DataFrame from it.



```
class TweetAnalyzer():
    """
    Functionality for analyzing and categorizing content from tweets.
    """
    def tweets_to_data_frame(self, tweets):
        df = pd.DataFrame(data=[tweet.text for tweet in tweets], columns=['tweets'])

        df['id'] = np.array([tweet.id for tweet in tweets])

        return df
###
if __name__ == '__main__':

    twitter_client = TwitterClient()
    tweet_analyzer = TweetAnalyzer()

    api = twitter_client.get_twitter_client_api()

    tweets = api.user_timeline(screen_name="ArnabGoswamiRTv", count=2000)

    df = tweet_analyzer.tweets_to_data_frame(tweets)
    #

    print(df.head(10))
```

```
                                              tweets                   id
0  Can admission be more clearer than this?\n\n@T...  1265210208466468864
1  You try to defend #YouTube in #youtubevstiktok...  1261168332947980288
2  @PMOIndia @narendramodi I support #GoLocal man...  1260251905520959491
3  Arnab vs Antonia: SC questions Kapil Sibal on ...  1259952811401494534
4  Now #SambitPatra is on the radar of congress, ...  1259857177264689153
5  One of the cops who interrogated Arnab Goswami...  1259742554435866624
6  We three are not alone, Nationalist Indians ar...  1259715167723696129
7  @sudhirchaudhary Many are scared of their own ...  1259588429546848256
```

Intializing the DataSets



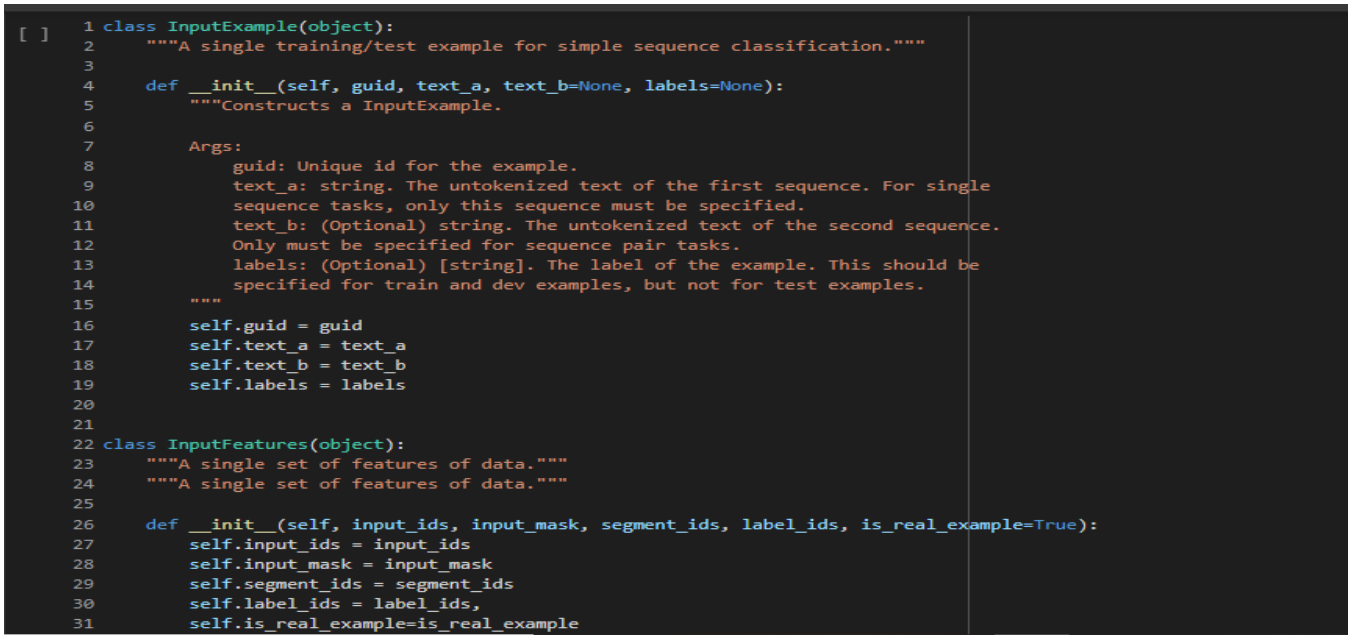
```
# train_data_path='D:/train.csv'# training file received from twitter
train = pd.read_csv('train1 (1).csv')
test = pd.read_csv('test1 (1).csv')# testing file without classifier
```

```
train.columns
```

```
Index(['id', 'comment_text', 'toxic', 'severe_toxic', 'obscene', 'threat',
       'insult', 'identity_hate'],
      dtype='object')
```

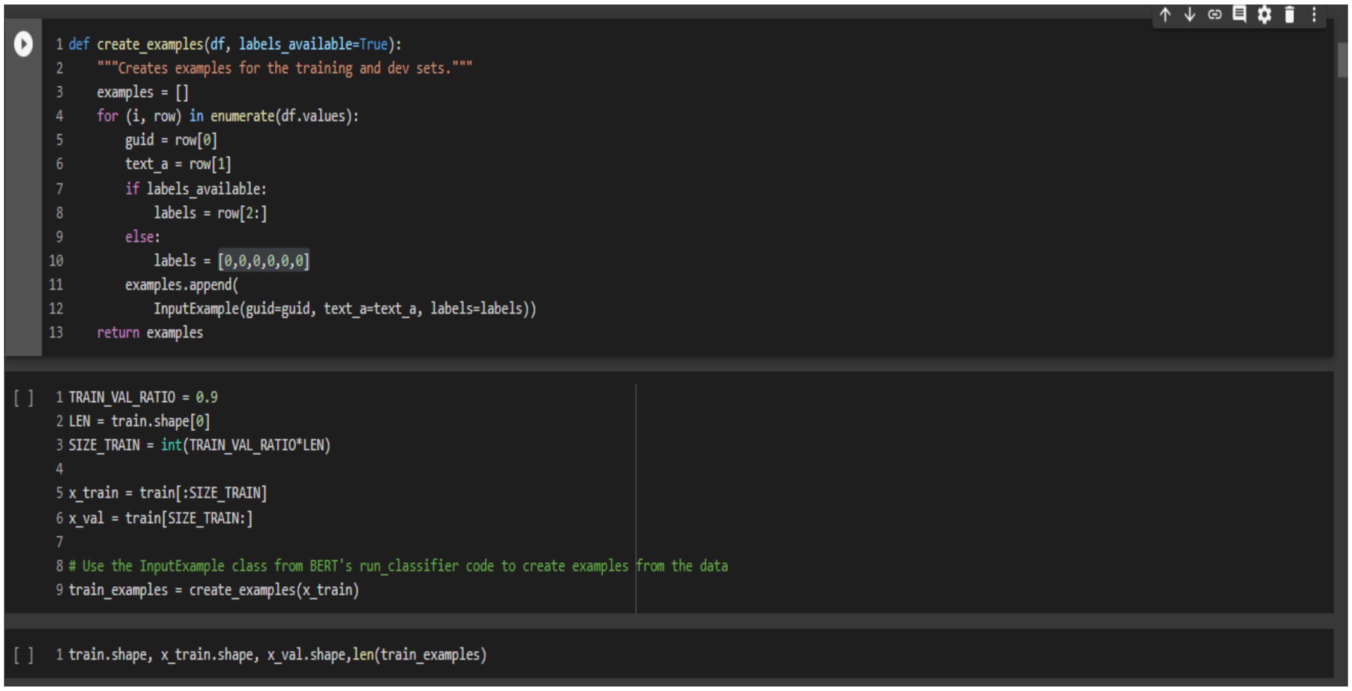
```
ID = 'id'
DATA_COLUMN = 'comment_text'
LABEL_COLUMNS = ['toxic','severe_toxic','obscene','threat','insult','identity_hate']
```

Input to the Model:



```
[ ] class InputExample(object):
        """A single training/test example for simple sequence classification."""

        def __init__(self, guid, text_a, text_b=None, labels=None):
            """Constructs a InputExample.

            Args:
                guid: Unique id for the example.
                text_a: string. The untokenized text of the first sequence. For single
                sequence tasks, only this sequence must be specified.
                text_b: (Optional) string. The untokenized text of the second sequence.
                Only must be specified for sequence pair tasks.
                labels: (Optional) [string]. The label of the example. This should be
                specified for train and dev examples, but not for test examples.
            """
            self.guid = guid
            self.text_a = text_a
            self.text_b = text_b
            self.labels = labels


    class InputFeatures(object):
        """A single set of features of data."""
        """A single set of features of data."""

        def __init__(self, input_ids, input_mask, segment_ids, label_ids, is_real_example=True):
            self.input_ids = input_ids
            self.input_mask = input_mask
            self.segment_ids = segment_ids
            self.label_ids = label_ids,
            self.is_real_example=is_real_example
```

Data Cleaning and Handling Null Values:

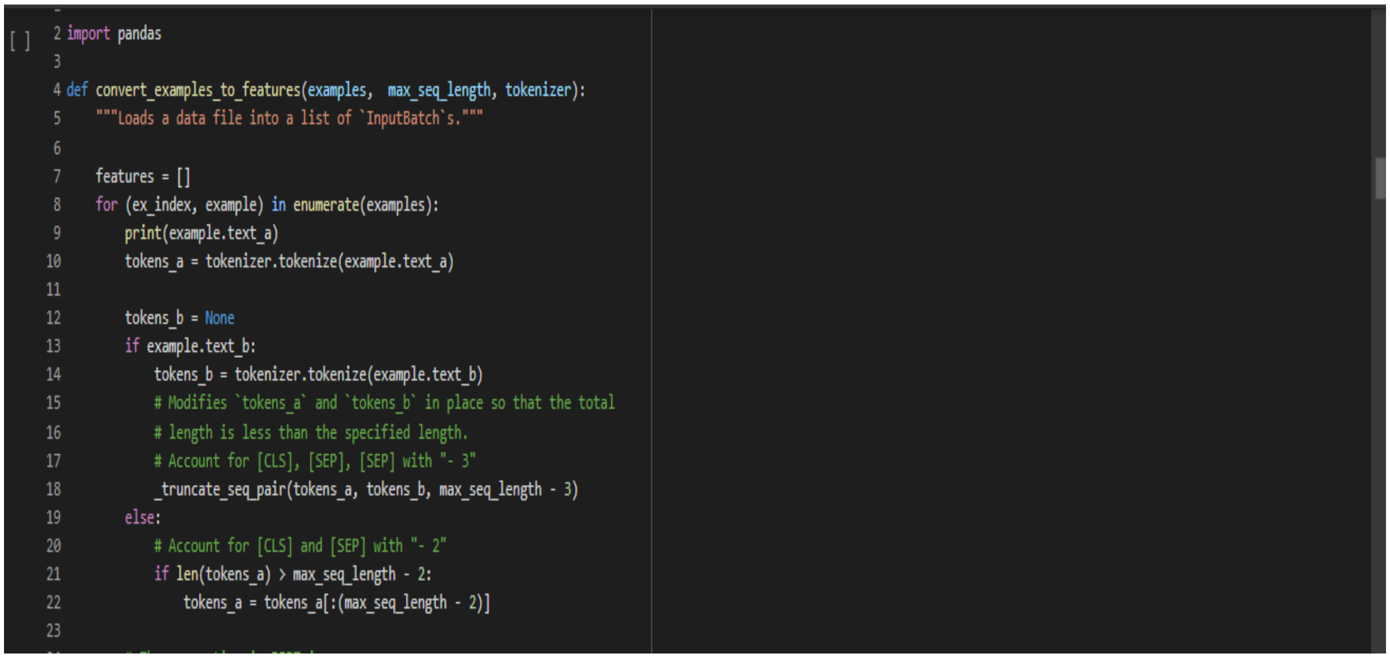


```
def create_examples(df, labels_available=True):
    """Creates examples for the training and dev sets."""
    examples = []
    for (i, row) in enumerate(df.values):
        guid = row[0]
        text_a = row[1]
        if labels_available:
            labels = row[2:]
        else:
            labels = [0,0,0,0,0,0]
        examples.append(
            InputExample(guid=guid, text_a=text_a, labels=labels))
    return examples
```

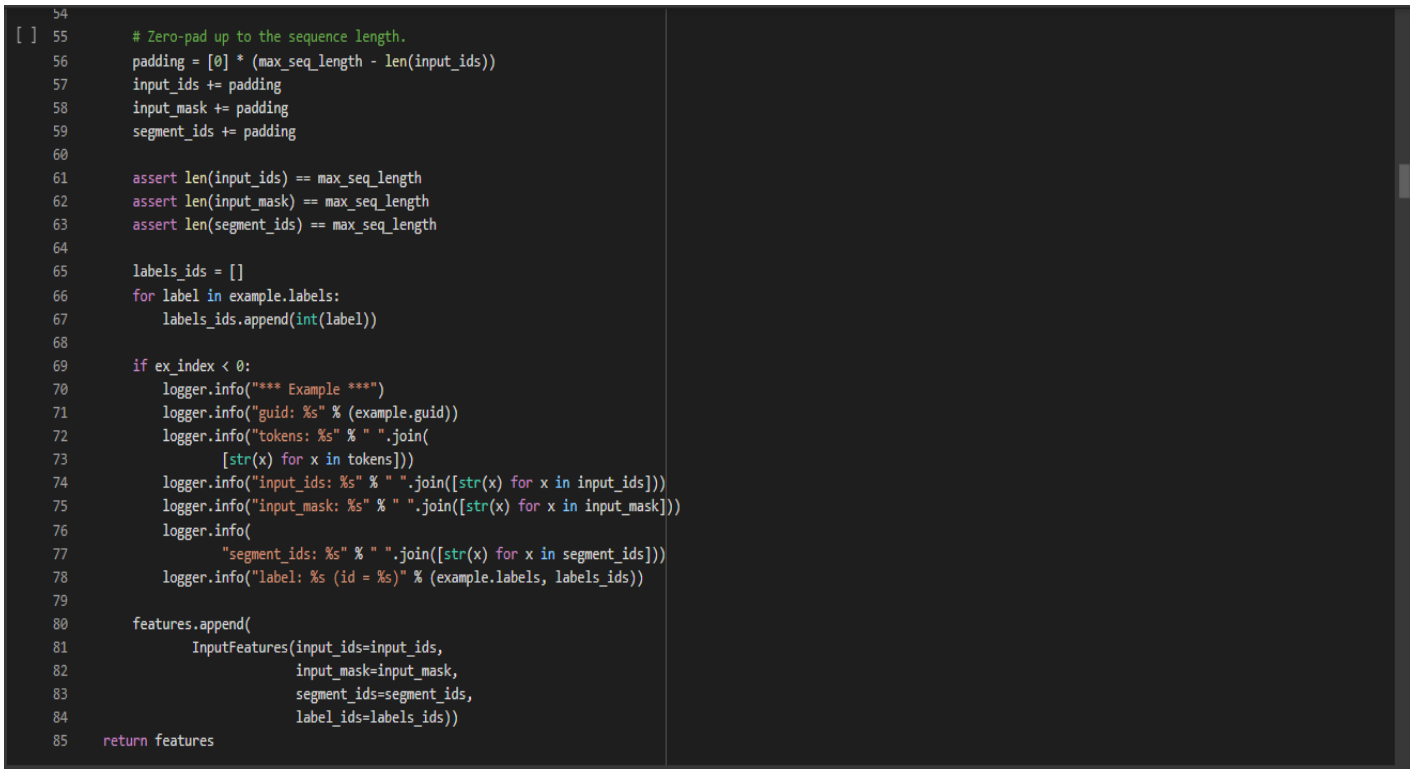
```
[ ] TRAIN_VAL_RATIO = 0.9
    LEN = train.shape[0]
    SIZE_TRAIN = int(TRAIN_VAL_RATIO*LEN)

    x_train = train[:SIZE_TRAIN]
    x_val = train[SIZE_TRAIN:]

    # Use the InputExample class from BERT's run_classifier code to create examples from the data
    train_examples = create_examples(x_train)
```

```
[ ] train.shape, x_train.shape, x_val.shape,len(train_examples)
```

Converting examples to features:

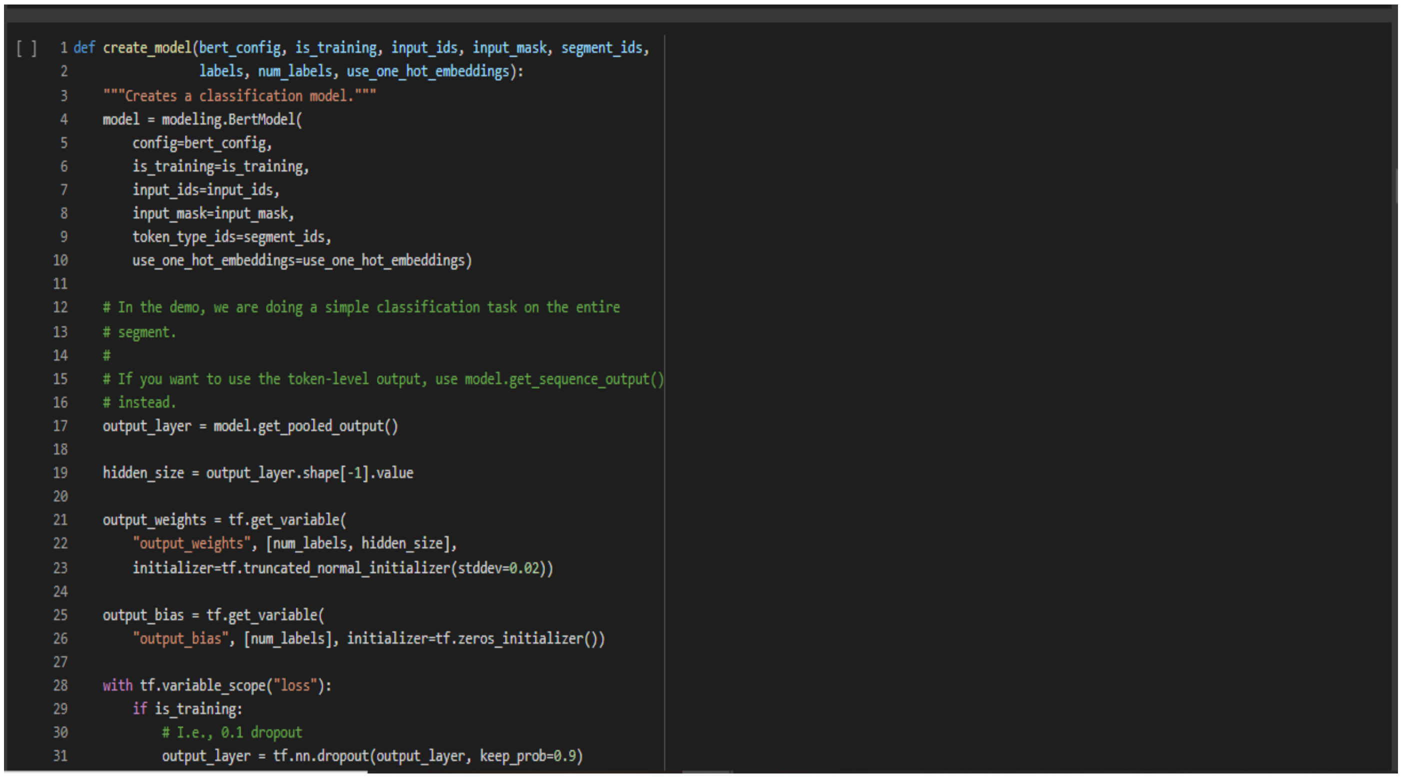


```
import pandas

def convert_examples_to_features(examples,  max_seq_length, tokenizer):
    """Loads a data file into a list of `InputBatch`s."""

    features = []
    for (ex_index, example) in enumerate(examples):
        print(example.text_a)
        tokens_a = tokenizer.tokenize(example.text_a)

        tokens_b = None
        if example.text_b:
            tokens_b = tokenizer.tokenize(example.text_b)
            # Modifies `tokens_a` and `tokens_b` in place so that the total
            # length is less than the specified length.
            # Account for [CLS], [SEP], [SEP] with "- 3"
            _truncate_seq_pair(tokens_a, tokens_b, max_seq_length - 3)
        else:
            # Account for [CLS] and [SEP] with "- 2"
            if len(tokens_a) > max_seq_length - 2:
                tokens_a = tokens_a[:(max_seq_length - 2)]

```

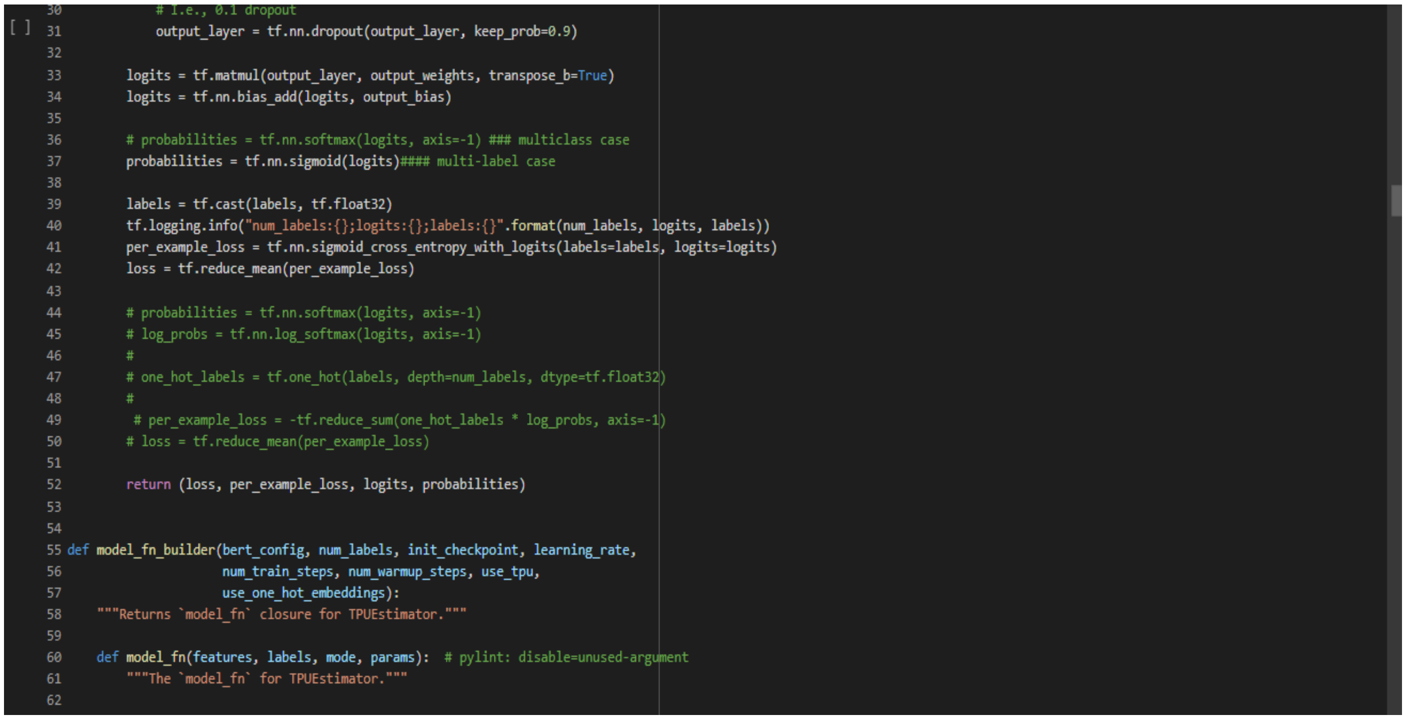


```
        # Zero-pad up to the sequence length.
        padding = [0] * (max_seq_length - len(input_ids))
        input_ids += padding
        input_mask += padding
        segment_ids += padding

        assert len(input_ids) == max_seq_length
        assert len(input_mask) == max_seq_length
        assert len(segment_ids) == max_seq_length

        labels_ids = []
        for label in example.labels:
            labels_ids.append(int(label))

        if ex_index < 0:
            logger.info("*** Example ***")
            logger.info("guid: %s" % (example.guid))
            logger.info("tokens: %s" % " ".join(
                    [str(x) for x in tokens]))
            logger.info("input_ids: %s" % " ".join([str(x) for x in input_ids]))
            logger.info("input_mask: %s" % " ".join([str(x) for x in input_mask]))
            logger.info(
                    "segment_ids: %s" % " ".join([str(x) for x in segment_ids]))
            logger.info("label: %s (id = %s)" % (example.labels, labels_ids))

        features.append(
                InputFeatures(input_ids=input_ids,
                              input_mask=input_mask,
                              segment_ids=segment_ids,
                              label_ids=labels_ids))
    return features
```

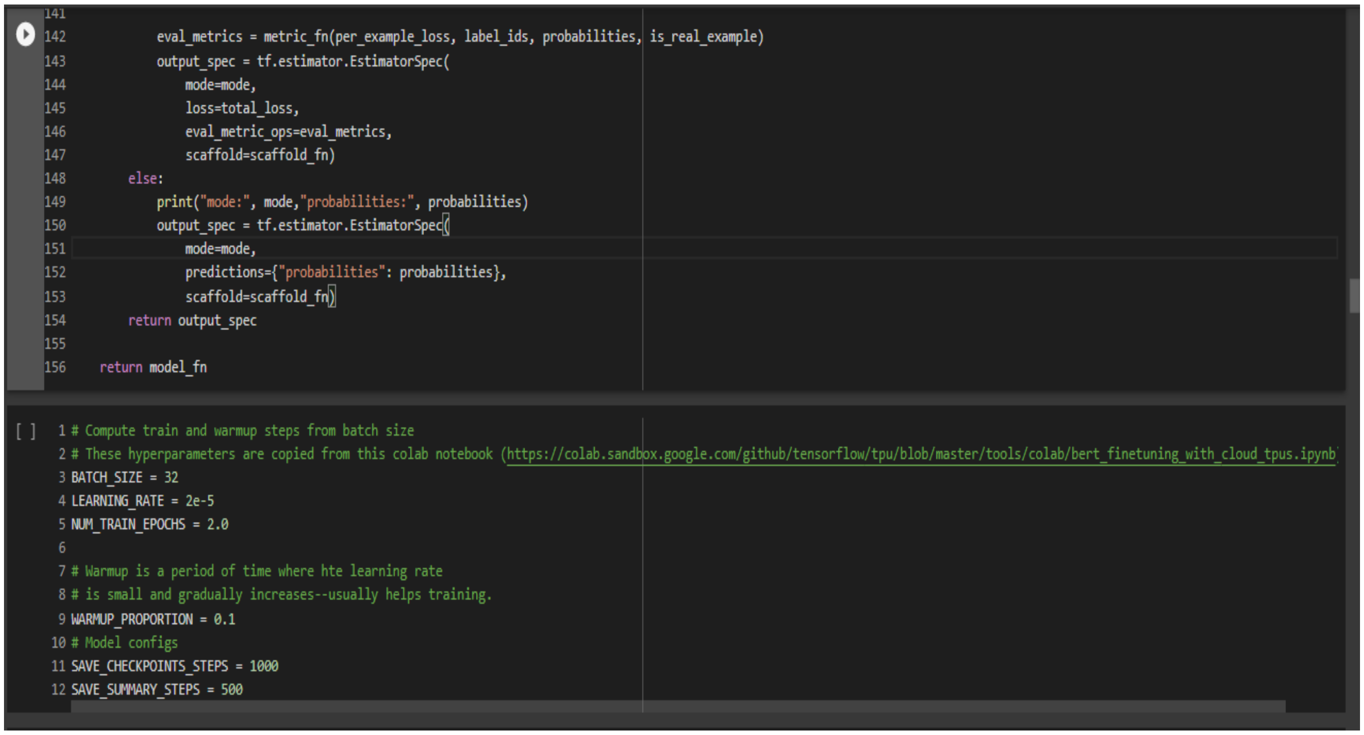
Create a model:



```
[ ] def create_model(bert_config, is_training, input_ids, input_mask, segment_ids,
                     labels, num_labels, use_one_hot_embeddings):
      """Creates a classification model."""
      model = modeling.BertModel(
          config=bert_config,
          is_training=is_training,
          input_ids=input_ids,
          input_mask=input_mask,
          token_type_ids=segment_ids,
          use_one_hot_embeddings=use_one_hot_embeddings)

      # In the demo, we are doing a simple classification task on the entire
      # segment.
      #
      # If you want to use the token-level output, use model.get_sequence_output()
      # instead.
      output_layer = model.get_pooled_output()

      hidden_size = output_layer.shape[-1].value

      output_weights = tf.get_variable(
          "output_weights", [num_labels, hidden_size],
          initializer=tf.truncated_normal_initializer(stddev=0.02))

      output_bias = tf.get_variable(
          "output_bias", [num_labels], initializer=tf.zeros_initializer())

      with tf.variable_scope("loss"):
          if is_training:
              # I.e., 0.1 dropout
              output_layer = tf.nn.dropout(output_layer, keep_prob=0.9)
```

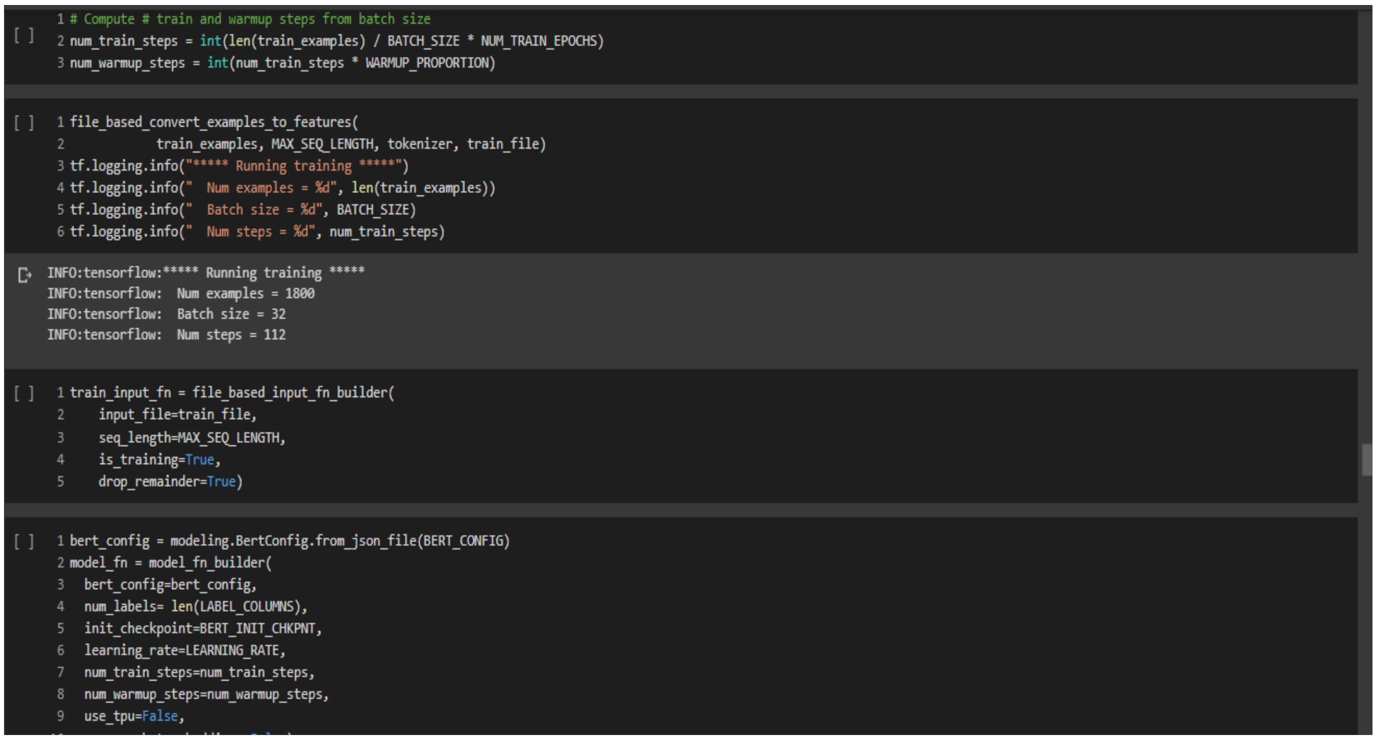


```
              # I.e., 0.1 dropout
[ ]           output_layer = tf.nn.dropout(output_layer, keep_prob=0.9)

          logits = tf.matmul(output_layer, output_weights, transpose_b=True)
          logits = tf.nn.bias_add(logits, output_bias)

          # probabilities = tf.nn.softmax(logits, axis=-1) ### multiclass case
          probabilities = tf.nn.sigmoid(logits)#### multi-label case

          labels = tf.cast(labels, tf.float32)
          tf.logging.info("num_labels:{};logits:{};labels:{}".format(num_labels, logits, labels))
          per_example_loss = tf.nn.sigmoid_cross_entropy_with_logits(labels=labels, logits=logits)
          loss = tf.reduce_mean(per_example_loss)

          # probabilities = tf.nn.softmax(logits, axis=-1)
          # log_probs = tf.nn.log_softmax(logits, axis=-1)
          #
          # one_hot_labels = tf.one_hot(labels, depth=num_labels, dtype=tf.float32)
          #
           # per_example_loss = -tf.reduce_sum(one_hot_labels * log_probs, axis=-1)
          # loss = tf.reduce_mean(per_example_loss)

          return (loss, per_example_loss, logits, probabilities)


def model_fn_builder(bert_config, num_labels, init_checkpoint, learning_rate,
                     num_train_steps, num_warmup_steps, use_tpu,
                     use_one_hot_embeddings):
    """Returns `model_fn` closure for TPUEstimator."""

    def model_fn(features, labels, mode, params):  # pylint: disable=unused-argument
        """The `model_fn` for TPUEstimator."""

```



```

            eval_metrics = metric_fn(per_example_loss, label_ids, probabilities, is_real_example)
            output_spec = tf.estimator.EstimatorSpec(
                mode=mode,
                loss=total_loss,
                eval_metric_ops=eval_metrics,
                scaffold=scaffold_fn)
        else:
            print("mode:", mode,"probabilities:", probabilities)
            output_spec = tf.estimator.EstimatorSpec(
                mode=mode,
                predictions={"probabilities": probabilities},
                scaffold=scaffold_fn)
        return output_spec

    return model_fn
```

```
[ ]
# Compute train and warmup steps from batch size
# These hyperparameters are copied from this colab notebook (https://colab.sandbox.google.com/github/tensorflow/tpu/blob/master/tools/colab/bert_finetuning_with_cloud_tpus.ipynb
BATCH_SIZE = 32
LEARNING_RATE = 2e-5
NUM_TRAIN_EPOCHS = 2.0

# Warmup is a period of time where hte learning rate
# is small and gradually increases--usually helps training.
WARMUP_PROPORTION = 0.1
# Model configs
SAVE_CHECKPOINTS_STEPS = 1000
SAVE_SUMMARY_STEPS = 500
```



```
[ ]
# Compute # train and warmup steps from batch size
num_train_steps = int(len(train_examples) / BATCH_SIZE * NUM_TRAIN_EPOCHS)
num_warmup_steps = int(num_train_steps * WARMUP_PROPORTION)
```

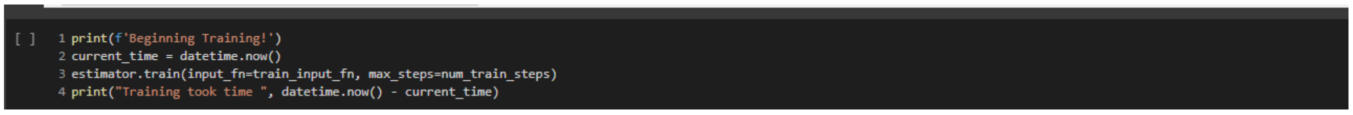
```
[ ]
file_based_convert_examples_to_features(
            train_examples, MAX_SEQ_LENGTH, tokenizer, train_file)
tf.logging.info("***** Running training *****")
tf.logging.info("  Num examples = %d", len(train_examples))
tf.logging.info("  Batch size = %d", BATCH_SIZE)
tf.logging.info("  Num steps = %d", num_train_steps)
```

```
INFO:tensorflow:***** Running training *****
INFO:tensorflow:  Num examples = 1800
INFO:tensorflow:  Batch size = 32
INFO:tensorflow:  Num steps = 112
```

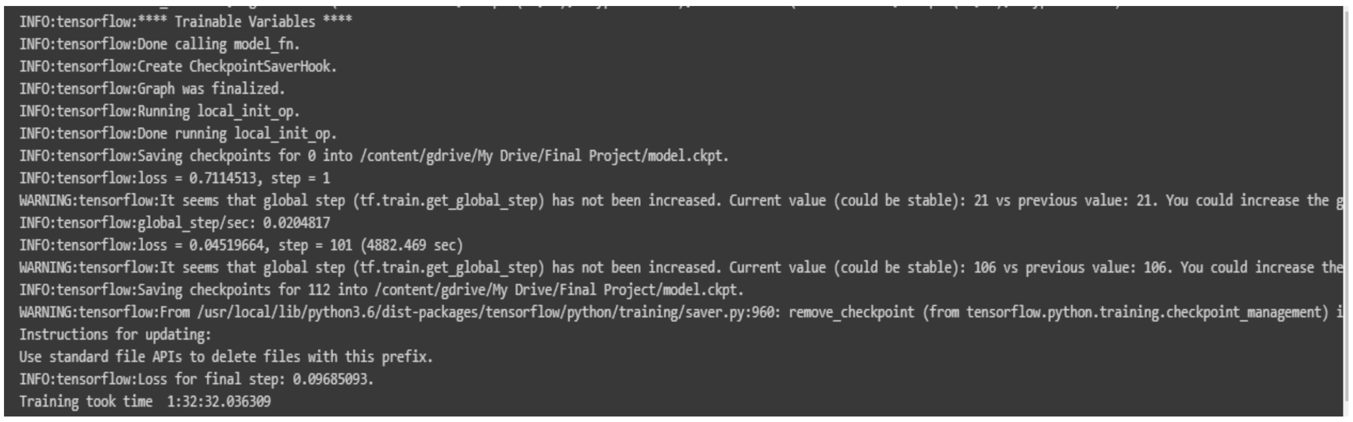
```
[ ]
train_input_fn = file_based_input_fn_builder(
    input_file=train_file,
    seq_length=MAX_SEQ_LENGTH,
    is_training=True,
    drop_remainder=True)
```

```
[ ]
bert_config = modeling.BertConfig.from_json_file(BERT_CONFIG)
model_fn = model_fn_builder(
  bert_config=bert_config,
  num_labels= len(LABEL_COLUMNS),
  init_checkpoint=BERT_INIT_CHKPNT,
  learning_rate=LEARNING_RATE,
  num_train_steps=num_train_steps,
  num_warmup_steps=num_warmup_steps,
  use_tpu=False,
```

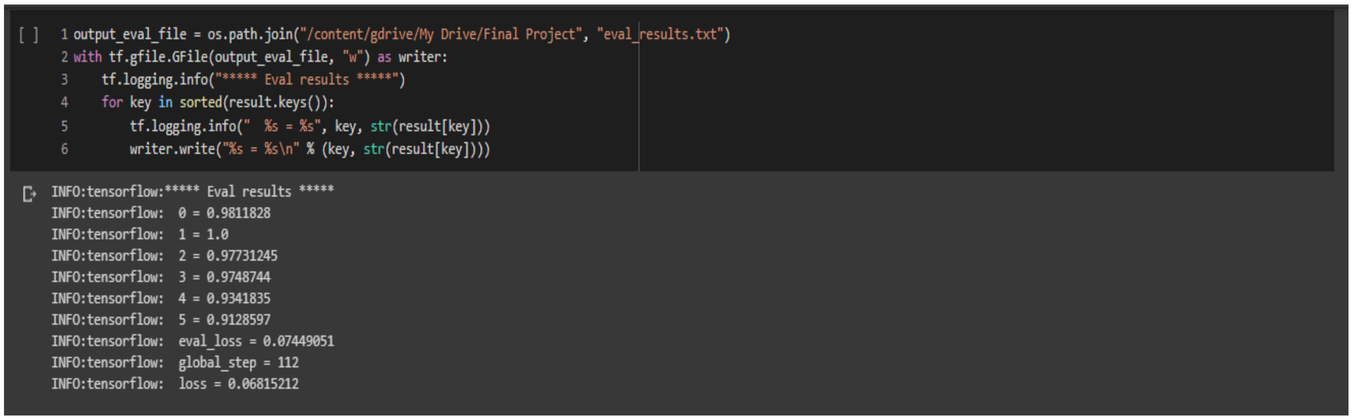
Training the Model:



```
[ ]  print(f'Beginning Training!')
     current_time = datetime.now()
     estimator.train(input_fn=train_input_fn, max_steps=num_train_steps)
     print("Training took time ", datetime.now() - current_time)
```



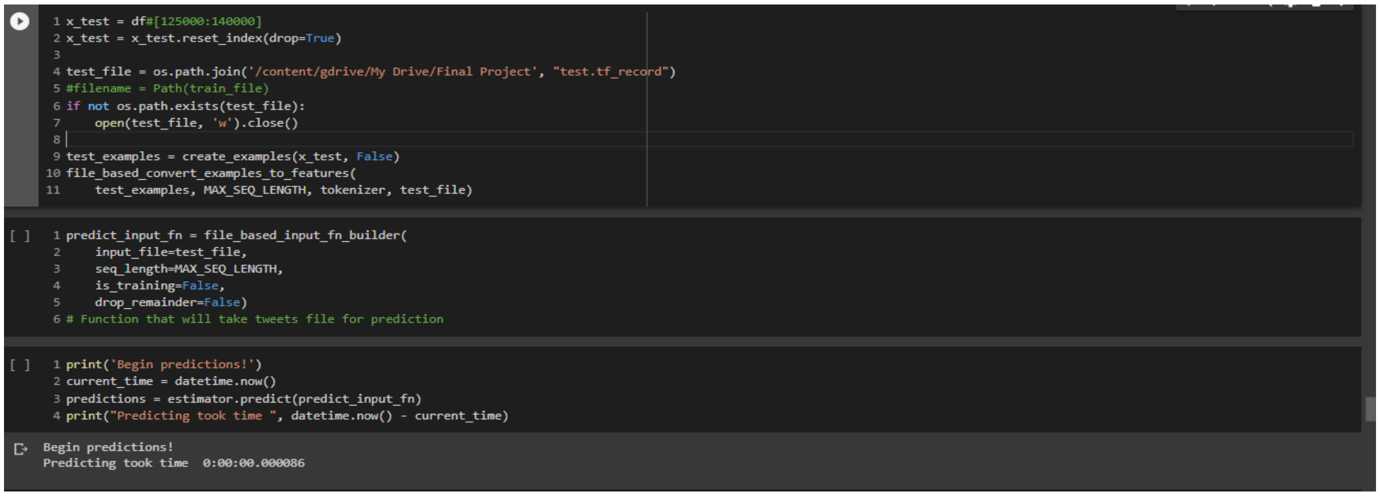
```
INFO:tensorflow:**** Trainable Variables ****
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 0 into /content/gdrive/My Drive/Final Project/model.ckpt.
INFO:tensorflow:loss = 0.7114513, step = 1
WARNING:tensorflow:It seems that global step (tf.train.get_global_step) has not been increased. Current value (could be stable): 21 vs previous value: 21. You could increase the g
INFO:tensorflow:global_step/sec: 0.0204817
INFO:tensorflow:loss = 0.04519664, step = 101 (4882.469 sec)
WARNING:tensorflow:It seems that global step (tf.train.get_global_step) has not been increased. Current value (could be stable): 106 vs previous value: 106. You could increase the
INFO:tensorflow:Saving checkpoints for 112 into /content/gdrive/My Drive/Final Project/model.ckpt.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/saver.py:960: remove_checkpoint (from tensorflow.python.training.checkpoint_management) i
Instructions for updating:
Use standard file APIs to delete files with this prefix.
INFO:tensorflow:Loss for final step: 0.09685093.
Training took time  1:32:32.036309
```



```
[ ]  output_eval_file = os.path.join("/content/gdrive/My Drive/Final Project", "eval_results.txt")
     with tf.gfile.GFile(output_eval_file, "w") as writer:
         tf.logging.info("***** Eval results *****")
         for key in sorted(result.keys()):
             tf.logging.info("  %s = %s", key, str(result[key]))
             writer.write("%s = %s\n" % (key, str(result[key])))
```

```
INFO:tensorflow:***** Eval results *****
INFO:tensorflow:  0 = 0.9811828
INFO:tensorflow:  1 = 1.0
INFO:tensorflow:  2 = 0.97731245
INFO:tensorflow:  3 = 0.9748744
INFO:tensorflow:  4 = 0.9341835
INFO:tensorflow:  5 = 0.9128597
INFO:tensorflow:  eval_loss = 0.07449051
INFO:tensorflow:  global_step = 112
INFO:tensorflow:  loss = 0.06815212
```

Predicting from Model:

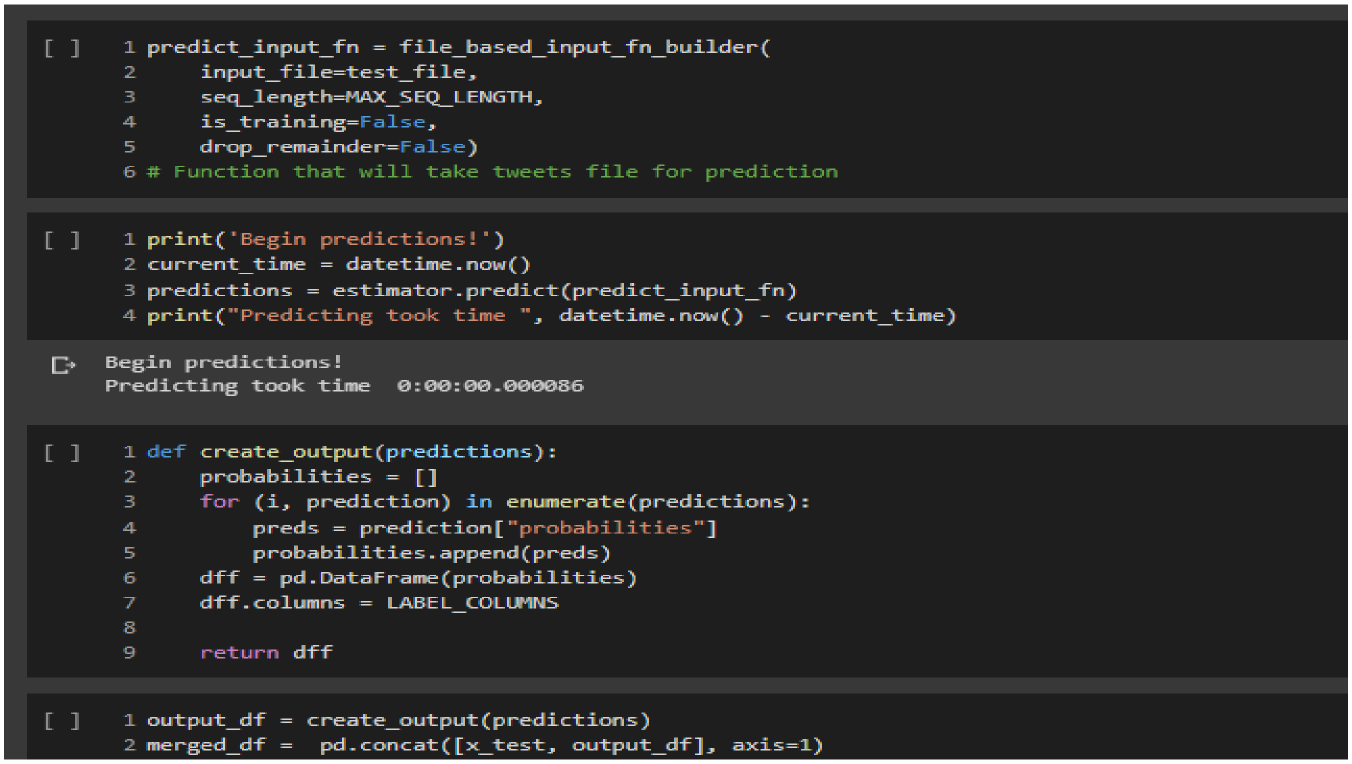


```
x_test = df#[125000:140000]
x_test = x_test.reset_index(drop=True)

test_file = os.path.join('/content/gdrive/My Drive/Final Project', "test.tf_record")
#filename = Path(train_file)
if not os.path.exists(test_file):
    open(test_file, 'w').close()

test_examples = create_examples(x_test, False)
file_based_convert_examples_to_features(
    test_examples, MAX_SEQ_LENGTH, tokenizer, test_file)
```

```
[ ]  predict_input_fn = file_based_input_fn_builder(
         input_file=test_file,
         seq_length=MAX_SEQ_LENGTH,
         is_training=False,
         drop_remainder=False)
     # Function that will take tweets file for prediction
```

```
[ ]  print('Begin predictions!')
     current_time = datetime.now()
     predictions = estimator.predict(predict_input_fn)
     print("Predicting took time ", datetime.now() - current_time)
```

```
Begin predictions!
Predicting took time  0:00:00.000086
```



```
predict_input_fn = file_based_input_fn_builder(
    input_file=test_file,
    seq_length=MAX_SEQ_LENGTH,
    is_training=False,
    drop_remainder=False)
# Function that will take tweets file for prediction
```

```
print('Begin predictions!')
current_time = datetime.now()
predictions = estimator.predict(predict_input_fn)
print("Predicting took time ", datetime.now() - current_time)
```

```
Begin predictions!
Predicting took time  0:00:00.000086
```

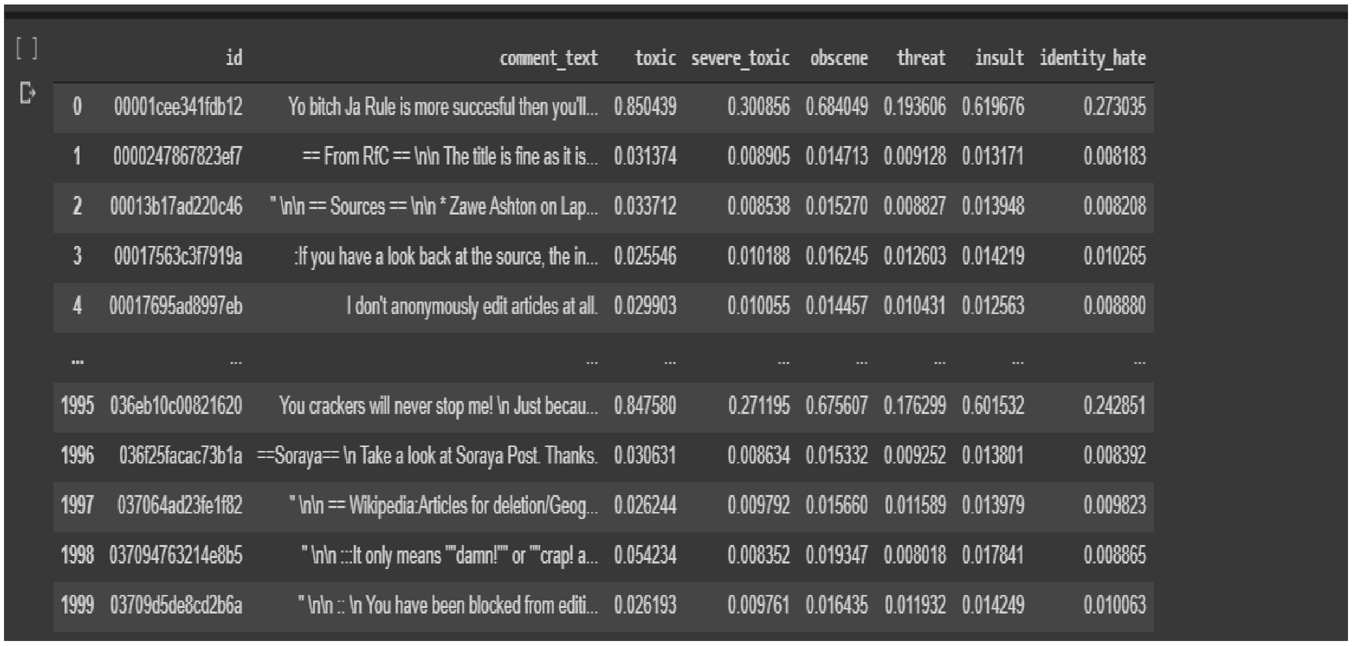
```
def create_output(predictions):
    probabilities = []
    for (i, prediction) in enumerate(predictions):
        preds = prediction["probabilities"]
        probabilities.append(preds)
    dff = pd.DataFrame(probabilities)
    dff.columns = LABEL_COLUMNS

    return dff
```

```
output_df = create_output(predictions)
merged_df =  pd.concat([x_test, output_df], axis=1)
```

Chapter 7

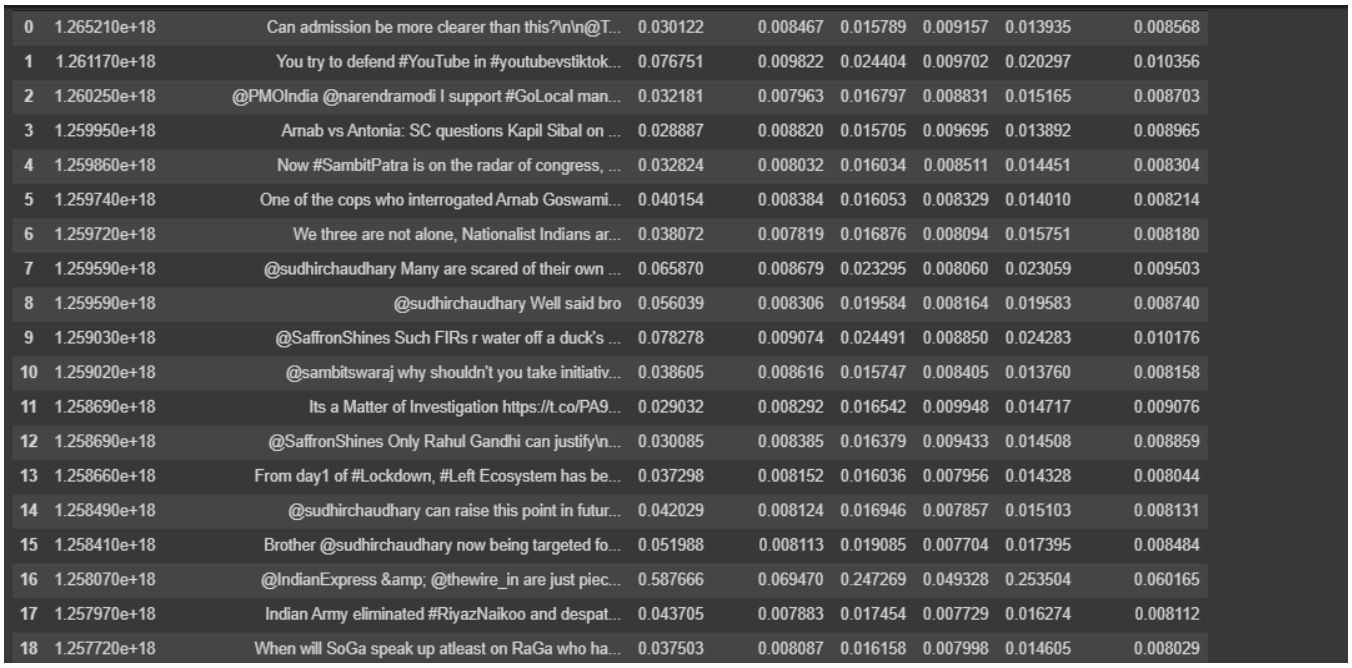
RESULTS

For the training Dataset:

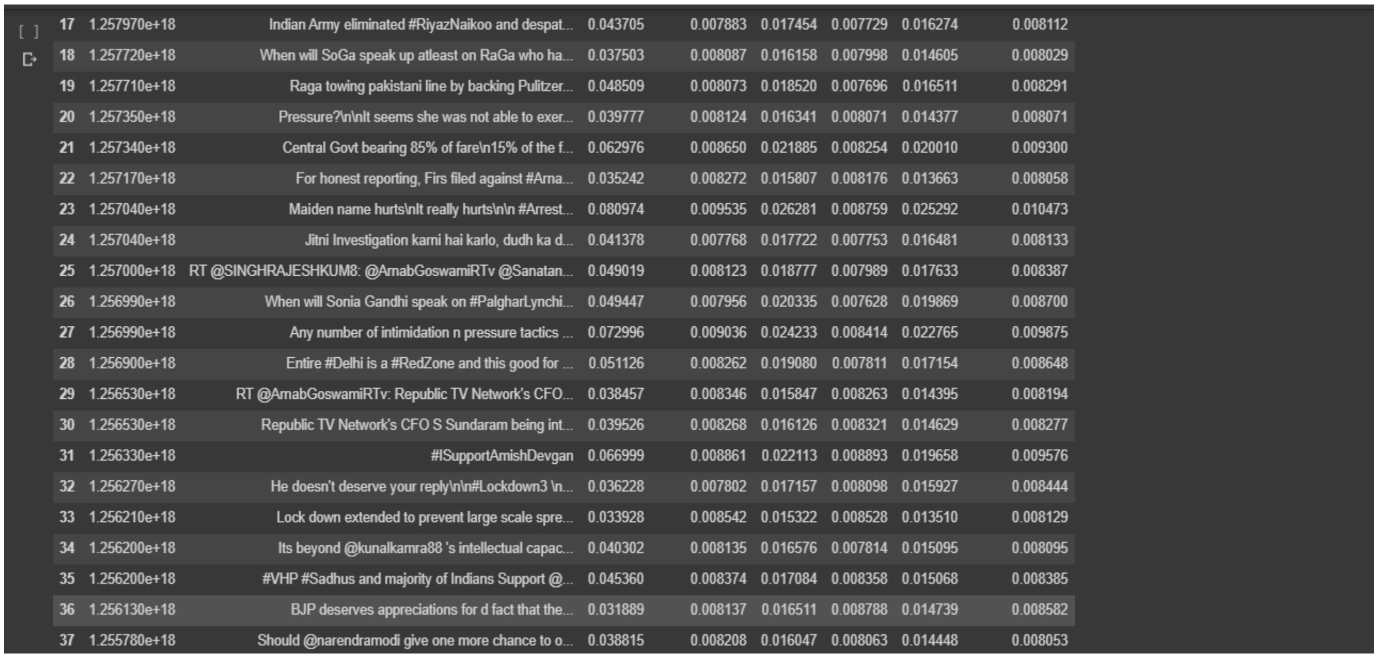


	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...	0.850439	0.300856	0.684049	0.193606	0.619676	0.273035
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...	0.031374	0.008905	0.014713	0.009128	0.013171	0.008183
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	0.033712	0.008538	0.015270	0.008827	0.013948	0.008208
3	00017563c3f7919a	:If you have a look back at the source, the in...	0.025546	0.010188	0.016245	0.012603	0.014219	0.010265
4	00017695ad8997eb	I don't anonymously edit articles at all.	0.029903	0.010055	0.014457	0.010431	0.012563	0.008880
...	...	...	...	...	...	...	...	...
1995	036eb10c00821620	You crackers will never stop me! \n Just becau...	0.847580	0.271195	0.675607	0.176299	0.601532	0.242851
1996	036f25facac73b1a	==Soraya== \n Take a look at Soraya Post. Thanks.	0.030631	0.008634	0.015332	0.009252	0.013801	0.008392
1997	037064ad23fe1f82	" \n\n == Wikipedia:Articles for deletion/Geog...	0.026244	0.009792	0.015660	0.011589	0.013979	0.009823
1998	037094763214e8b5	" \n\n :::It only means ""damn!"" or ""crap! a...	0.054234	0.008352	0.019347	0.008018	0.017841	0.008865
1999	03709d5de8cd2b6a	" \n\n :: \n You have been blocked from editi...	0.026193	0.009761	0.016435	0.011932	0.014249	0.010063

For the Tweets:



0	1.265210e+18	Can admission be more clearer than this?\n\n@T...	0.030122	0.008467	0.015789	0.009157	0.013935	0.008568
1	1.261170e+18	You try to defend #YouTube in #youtubevstiktok...	0.076751	0.009822	0.024404	0.009702	0.020297	0.010356
2	1.260250e+18	@PMOIndia @narendramodi I support #GoLocal man...	0.032181	0.007963	0.016797	0.008831	0.015165	0.008703
3	1.259950e+18	Arnab vs Antonia: SC questions Kapil Sibal on ...	0.028887	0.008820	0.015705	0.009695	0.013892	0.008965
4	1.259860e+18	Now #SambitPatra is on the radar of congress, ...	0.032824	0.008032	0.016034	0.008511	0.014451	0.008304
5	1.259740e+18	One of the cops who interrogated Arnab Goswami...	0.040154	0.008384	0.016053	0.008329	0.014010	0.008214
6	1.259720e+18	We three are not alone, Nationalist Indians ar...	0.038072	0.007819	0.016876	0.008094	0.015751	0.008180
7	1.259590e+18	@sudhirchaudhary Many are scared of their own ...	0.065870	0.008679	0.023295	0.008060	0.023059	0.009503
8	1.259590e+18	@sudhirchaudhary Well said bro	0.056039	0.008306	0.019584	0.008164	0.019583	0.008740
9	1.259030e+18	@SaffronShines Such FIRs r water off a duck's ...	0.078278	0.009074	0.024491	0.008850	0.024283	0.010176
10	1.259020e+18	@sambitswaraj why shouldn't you take initiativ...	0.038605	0.008616	0.015747	0.008405	0.013760	0.008158
11	1.258690e+18	Its a Matter of Investigation https://t.co/PA9...	0.029032	0.008292	0.016542	0.009948	0.014717	0.009076
12	1.258690e+18	@SaffronShines Only Rahul Gandhi can justify\n...	0.030085	0.008385	0.016379	0.009433	0.014508	0.008859
13	1.258660e+18	From day1 of #Lockdown, #Left Ecosystem has be...	0.037298	0.008152	0.016036	0.007956	0.014328	0.008044
14	1.258490e+18	@sudhirchaudhary can raise this point in futur...	0.042029	0.008124	0.016946	0.007857	0.015103	0.008131
15	1.258410e+18	Brother @sudhirchaudhary now being targeted fo...	0.051988	0.008113	0.019085	0.007704	0.017395	0.008484
16	1.258070e+18	@IndianExpress &amp; @thewire_in are just piec...	0.587666	0.069470	0.247269	0.049328	0.253504	0.060165
17	1.257970e+18	Indian Army eliminated #RiyazNaikoo and despat...	0.043705	0.007883	0.017454	0.007729	0.016274	0.008112
18	1.257720e+18	When will SoGa speak up atleast on RaGa who ha...	0.037503	0.008087	0.016158	0.007998	0.014605	0.008029



17	1.257970e+18	Indian Army eliminated #RiyazNaikoo and despat...	0.043705	0.007883	0.017454	0.007729	0.016274	0.008112
18	1.257720e+18	When will SoGa speak up atleast on RaGa who ha...	0.037503	0.008087	0.016158	0.007998	0.014605	0.008029
19	1.257710e+18	Raga towing pakistani line by backing Pulitzer...	0.048509	0.008073	0.018520	0.007696	0.016511	0.008291
20	1.257350e+18	Pressure?\n\nIt seems she was not able to exer...	0.039777	0.008124	0.016341	0.008071	0.014377	0.008071
21	1.257340e+18	Central Govt bearing 85% of fare\n15% of the f...	0.062976	0.008650	0.021885	0.008254	0.020010	0.009300
22	1.257170e+18	For honest reporting, Firs filed against #Arna...	0.035242	0.008272	0.015807	0.008176	0.013663	0.008058
23	1.257040e+18	Maiden name hurts\nIt really hurts\n\n #Arrest...	0.080974	0.009535	0.026281	0.008759	0.025292	0.010473
24	1.257040e+18	Jitni Investigation karni hai karlo, dudh ka d...	0.041378	0.007768	0.017722	0.007753	0.016481	0.008133
25	1.257000e+18	RT @SINGHRAJESHKUM8: @ArnabGoswamiRTv @Sanatan...	0.049019	0.008123	0.018777	0.007989	0.017633	0.008387
26	1.256990e+18	When will Sonia Gandhi speak on #PalgharLynchi...	0.049447	0.007956	0.020335	0.007628	0.019869	0.008700
27	1.256990e+18	Any number of intimidation n pressure tactics ...	0.072996	0.009036	0.024233	0.008414	0.022765	0.009875
28	1.256900e+18	Entire #Delhi is a #RedZone and this good for ...	0.051126	0.008262	0.019080	0.007811	0.017154	0.008648
29	1.256530e+18	RT @ArnabGoswamiRTv: Republic TV Network's CFO...	0.038457	0.008346	0.015847	0.008263	0.014395	0.008194
30	1.256530e+18	Republic TV Network's CFO S Sundaram being int...	0.039526	0.008268	0.016126	0.008321	0.014629	0.008277
31	1.256330e+18	#ISupportAmishDevgan	0.066999	0.008861	0.022113	0.008893	0.019658	0.009576
32	1.256270e+18	He doesn't deserve your reply\n\n#Lockdown3 \n...	0.036228	0.007802	0.017157	0.008098	0.015927	0.008444
33	1.256210e+18	Lock down extended to prevent large scale spre...	0.033928	0.008542	0.015322	0.008528	0.013510	0.008129
34	1.256200e+18	Its beyond @kunalkamra88 's intellectual capac...	0.040302	0.008135	0.016576	0.007814	0.015095	0.008095
35	1.256200e+18	#VHP #Sadhus and majority of Indians Support @...	0.045360	0.008374	0.017084	0.008358	0.015068	0.008385
36	1.256130e+18	BJP deserves appreciations for d fact that the...	0.031889	0.008137	0.016511	0.008788	0.014739	0.008582
37	1.255780e+18	Should @narendramodi give one more chance to o...	0.038815	0.008208	0.016047	0.008063	0.014448	0.008053

Chapter 8

CONCLUSION

The threat monitor, named SYNAPSE, gathers tweets from a set of Twitter accounts, filters them to target solely the monitored infrastructure, classifies remaining tweets as either relevant or not, aggregates tweets related to same threat using a stream clustering approach, and generates indicators of compromise suitable for threat sharing platforms. The results obtained from the model design experiments revealed that using a single classification model for the complete monitored infrastructure is preferable to using an ensemble of models for different infrastructure parts.

Chapter 9

FUTURE SCOPE

- This model can be used for the analysis of rating of the product.
- The future of analyzing tweets is going to continue to dig deeper, far past the surface of the number of likes, comments and shares, and aim to reach, and truly understand, the significance of social media interactions and what they tell us about the consumers behind the screens.
- We believe that tweet analysis will only increase in importance as more and more people use online channels to communicate, both directly and indirectly, with corporations.

References

[1] Marshland, S. (2009) Machine Learning an Algorithmic Perspective. CRC Press, New Zealand, 6-7. 4

[2] Sharma, P. and Kaur, M. (2013) Classification in Pattern Recognition: A Review. International Journal of Advanced Research in Computer Science and Software E ngineering , 3, 298.

[3] Rambhajani, M., Deepanker, W. and Pathak, N. (2015) A Survey on Implementation of Machine Learning Techniques for Dermatology Diseases Classification. I nternational Journal of Advances in Engineering & Technology , 8, 194-195.

[4] Kononenko, I. (2001) Machine Learning for Medical Diagnosis: History, State of the Art and Perspective. Journal of Artificial Intelligence in Medicine , 1, 89-109.

[5] Otoom, A.F., Abdallah, E.E., Kilani, Y., Kefaye, A. and Ashour, M. (2015) Effective Diagnosis and Monitoring of Heart Disease. International Journal of Software En gineering and Its Applications . 9, 143-156.

[6] Vembandasamy, K., Sasipriya, R. and Deepa, E. (2015) Heart Diseases Detection Using Naive Bayes Algorithm. IJISET - International Journal of Innovative Science , Engineering & Technology , 2, 441-444.

[7] Chaurasia, V. and Pal, S. (2013) Data Mining Approach to Detect Heart Disease. International Journal of Advanced Computer Science and Information Technology (IJACSIT), 2, 56-66.

[8] Parthiban, G. and Srivatsa, S.K. (2012) Applying Machine Learning Methods in Diagnosing Heart Disease for Diabetic Patients. International Journal of Applied I nformation Systems (IJAIS), 3, 25-30.

[9] Tan, K.C., Teoh, E.J., Yu, Q. and Goh, K.C. (2009) A Hybrid Evolutionary Algorithm for Attribute Selection in Data Mining. Journal of Expert System with Appl ications , 36, 8616-8630.

https://doi.org/10.1016/j.eswa.2008.10.013

[10] Karamizadeh, S., Abdullah, S.M., Halimi, M., Shayan, J. and Rajabi, M.J. (2014) Advantage and Drawback of Support Vector Machine Functionality. 2014 IEEE I nternational Conference on Computer , Communication and Control Technology (I 4 CT), Langkawi, 2-4 September 2014, 64-65. https://doi.org/10.1109/i4ct.2014.6914146

[11] Iyer, A., Jeyalatha, S. and Sumbaly, R. (2015) Diagnosis of Diabetes Using Classification Mining Techniques. International Journal of Data Mining & Knowledge Ma nagement Process (IJDKP), 5, 1-14. https://doi.org/10.5121/ijdkp.2015.5101

[12] Sen, S.K. and Dash, S. (2014) Application of Meta Learning Algorithms for the Prediction of Diabetes Disease. International Journal of Advance Research in Compu ter Science and Management Studies , 2, 396-401.

[13] Kumari, V.A. and Chitra, R. (2013) Classification of Diabetes Disease Using Support Vector Machine. International Journal of Engineering Research and Applic ations (IJERA), 3, 1797-1801.

[14] Sarwar, A. and Sharma, V. (2012) Intelligent Naïve Bayes Approach to Diagnose Diabetes Type-2. Special Issue of International Journal of Computer Applications (0975-8887) on Issues and Challenges in Networking , Intelligence and Computing Technologies - ICNICT 2012, 3, 14-16.

[15] Ephzibah, E.P. (2011) Cost Effective Approach on Feature Selection using Genetic Algorithms and Fuzzy Logic for Diabetes Diagnosis. International Journa l on Soft Computing (IJSC), 2, 1-10. https://doi.org/10.5121/ijsc.2011.210.