**Feature Extraction and Analysis of Natural Language Processing for Deep Learning English Language**

**Objective:**

This aims to classify textual content into non-hate or hate speech, in which case the method may also identify the targeting characteristics (i.e., types of hate, such as race, and religion) in the hate speech.

**Abstract:**

In this work, we argue for a focus on the latter problem for practical reasons. We show that it is a much more challenging task, as our analysis of the language in the typical datasets shows that hate speech lacks unique, discriminative features and therefore is found in the ‘long tail’ in a dataset that is difficult to discover. We then propose Deep Neural Network structures serving as feature extractors that are particularly effective for capturing the semantics of hate speech. Our methods are evaluated on the largest collection of hate speech datasets based on Twitter, and are shown to be able to outperform state of the art by up to 6 percentage points in macro-average F1, or 9 percentage points in the more challenging case of identifying hateful content.

**Introduction:**

The exponential growth of social media such as Twitter and community forums has revolutionised communication and content publishing, but is also increasingly exploited for the propagation of hate speech and the organisation of hate-based activities [1, 3]. The anonymity and mobility afforded by such media has made the breeding and spread of hate speech – eventually leading to hate crime – effortless in a virtual land scape beyond the realms of traditional law enforcement. The term ‘hate speech’ was formally defined as ‘any communication that disparages a person or a group on the basis of some characteristics (to be referred to as types of hate or hate classes) such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics’. In the UK, there has been significant increase of hate speech towards the migrant and Muslim communities following recent events including leaving the EU, the Manchester and the London attacks. In the EU, surveys and reports focusing on young people in the EEA (European Economic Area) region show rising hate speech. And related crimes based on religious beliefs, ethnicity, sexual orientation or gender, as 80% of respondents have encountered hate speech online and 40% felt attacked or threatened. Statistics also show that in the US, hate speech and crime is on the rise since the Trump election. The urgency of this matter has been increasingly recognised, as a range of international initiatives have been launched towards the qualification of the problems and the development of counter-measures.

**Motivation:**

We have chosen to work with twitter since we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for twitter, as compared to traditional blogging sites. Moreover the response on twitter is more prompt and also more general (since the number of users who tweet is substantially more than those who write web blogs on a daily basis). Sentiment analysis of public is highly critical in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm. This could be done by analysing overall public sentiment towards that firm with respect to time and using economics tools for finding the correlation between public sentiment and the firm’s stock market value. Firms can also estimate how well their product is responding in the market, which areas of the market is it having a favourable response and in which a negative response (since twitter allows us to download stream of geo-tagged tweets for particular locations. If firms can get this information they can analyze the reasons behind geographically differentiated response, and so they can market their product in a more optimized manner by looking for appropriate solutions like creating suitable market segments. Predicting the results of popular political elections and polls is also an emerging application to sentiment analysis. One such study was conducted by Tumasjan et al. in Germany for predicting the outcome of federal elections in which concluded that twitter is a good reflection of offline sentiment .

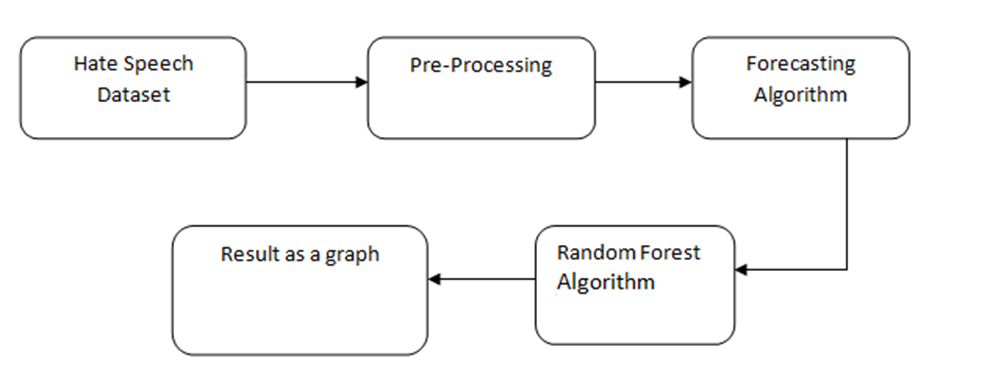
**Problem Statement:**

Twitter is a popular social networking website where members create and interact with messages known as “tweets”. This serves as a mean for individuals to express their thoughts or feelings about different subjects. Various different parties such as consumers and marketers have done sentiment analysis on such tweets to gather insights into products or to conduct market analysis. Furthermore, with the recent advancements in machine learning algorithms, we are able improve the accuracy of our sentiment analysis predictions.In this report, we will attempt to conduct sentiment analysis on “tweets” using various different machine learning algorithms. We attempt to classify the polarity of the tweet where it is either positive or negative. If the tweet has both positive and negative elements, the more dominant sentiment should be picked as the final label.We use the dataset from [Kaggle](https://www.kaggle.com/c/cs5228-project-2)which was crawled and labeled positive/negative. The data pro-vided comes with emoticons, usernames and hashtags which are required to be processed and converted into a standard form. We also need to extract useful features from the text such uni-grams and bigrams which is a form of representation of the “tweet”.We use various machine learning algorithms to conduct sentiment analysis using the extracted fea-tures. However, just relying on individual models did not give a high accuracy so we pick the top few models to generate a model ensemble. Ensembling is a form of meta learning algorithm tech-nique where we combine different classifiers in order to improve the prediction accuracy. Finally, we report our experimental results and findings at the end.

**Data Description:**

The data given is in the form of a comma-separated values files with tweets and their correspond-ing sentiments. The training dataset is a csv file of type tweet\_id,sentiment,tweet where the tweet\_id is a unique integer identifying the tweet, sentiment is either 1 (positive) or 0 (neg-ative), and tweet is the tweet enclosed in "". Similarly, the test dataset is a csv file of type tweet\_id,tweet.

**System Architecture:**

****

**Existing System:**

Existing methods primarily cast the problem as a supervised document classification task [33]. These can be divided into two categories: one relies on manual feature engineering that are then consumed by algorithms such as SVM, Naive Bayes, and Logistic Regression [3, 9, 11, 15, 19, 23, 35–39] (classic methods); the other represents the more recent deep learning paradigm that employs neural networks to automatically learn multi-layers of abstract features from raw data [13, 26, 30, 34] (deep learning methods).

**Disadvantage:**

* Existing studies on hate speech detection have primarily reported their results using micro-average Precision, Recall and F1 [1, 13, 30, 36, 37, 40].
* The problem with this is that in an unbalanced dataset where instances of one class (to be called the ‘dominant class’) significantly out-number others (to be called ‘minority classes’), micro-averaging can mask the real performance on minority classes.

**Proposed System:**

All datasets are significantly biased towards non-hate, as hate Tweets account between only 5.8% (DT) and 31.6% (WZ). When we inspect specific types of hate, some can be even scarcer, such as ‘racism’ and as mentioned before, the extreme case of ‘both’. This has two implications. First, an evaluation measure such as the micro F1 that looks at a system’s performance on the entire dataset regardless of class difference can be biased to the system’s ability of detecting ‘non-hate’. In other words, a hypothetical system that achieves almost perfect F1 in identifying ‘racism’ Tweets can still be overshadowed by its poor F1 in identifying ‘non-hate’, and vice versa. Second, compared to non-hate, the training data for hate Tweets are very scarce. This may not be an issue that is easy to address as it seems, since the datasets are collected from Twitter and reflect the real nature of data imbalance in this domain. Thus to annotate more training data for hateful content we will almost certainly have to spend significantly more effort annotating non-hate.

**Advantage:**

* Also, as we shall show in the following, this problem may not be easily mitigated by conventional methods of over- or under-sampling.
* Because the real challenge is the lack of unique, discriminative linguistic characteristics in hate Tweets compared to non-hate.
* As a proxy to quantify and compare the linguistic characteristics of hate and non-hate Tweets, we propose to study the ‘uniqueness’ of the vocabulary for each class.

**Conclusion:**

As hate speech continues to be a societal problem, the need for automatic hate speech detection systems becomes more apparent. In this report, we proposed a solution to the detection of hate speech and offensive language on Twitter through machine learning using Bag of Words and TF IDF values. We performed comparative analysis of Logistic Regression, Naive Bayes, Decision Tree, Random Forest and Gradient Boosting on various sets of feature values and model parameters. The results showed that **Logistic Regression performs comparatively better with the TF IDF approach.**We presented the current problems for this task and our system that achieves reasonable accuracy (89%) as well as recall (84%). Given all the challenges that remain, there is a need for more research on this problem, including both technical and practical matters.

**SYSTEM REQUIREMENTS:**

**HARDWARE REQUIREMENTS:**

System : i3core

Hard Disk:; 120 GB or above

Ram:4 GB(min) or above

**SOFTWARE REQUIREMENTS:**

Operating system : Windows7. Or above

Coding Language : Python

Tool : Anaconda Navigator

Software Description:

**Domain Specification:**

**MACHINE LEARNING**

Machine Learning is a system that can learn from example through self-improvement and without being explicitly coded by programmer. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results.

Machine learning combines data with statistical tools to predict an output. This output is then used by corporate to makes actionable insights. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input, use an algorithm to formulate answers.

A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation.

Machine learning is also used for a variety of task like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on.

**Machine Learning vs. Traditional Programming**

Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.

**COMPUTER**

**DATA RULES**

**OUTPUT**

**Machine Learning**

## How does Machine learning work?

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the **learning** and **inference**. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the **data**. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a **feature vector.** You can think of a feature vector as a subset of data that is used to tackle a problem.

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a **model**. Therefore, the learning stage is used to describe the data and summarize it into a model.

[](https://www.guru99.com/images/tensorflow/082918_1102_WhatisMachi3.png)

For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

#### Inferring

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

[](https://www.guru99.com/images/tensorflow/082918_1102_WhatisMachi4.png)

The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4-7 until the results are satisfying
9. Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

## Machine learning Algorithms and where they are used?

[](https://www.guru99.com/images/tensorflow/082918_1102_WhatisMachi5.png)

Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms

#### Supervised learning

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

* Classification task
* Regression task

#### Classification

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

#### Regression

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macroeconomics index. The system will be trained to estimate the price of the stocks with the lowest possible error.

|  |  |  |
| --- | --- | --- |
| **Algorithm Name** | **Description** | **Type** |
| **Linear regression** | Finds a way to correlate each feature to the output to help predict future values. | Regression |
| **Logistic regression** | Extension of linear regression that's used for classification tasks. The output variable 3is binary (e.g., only black or white) rather than continuous (e.g., an infinite list of potential colors) | Classification |
| **Decision tree** | Highly interpretable classification or regression model that splits data-feature values into branches at decision nodes (e.g., if a feature is a color, each possible color becomes a new branch) until a final decision output is made | Regression Classification |
| **Naive Bayes** | The Bayesian method is a classification method that makes use of the Bayesian theorem. The theorem updates the prior knowledge of an event with the independent probability of each feature that can affect the event. | Regression Classification |
| **Support vector machine** | Support Vector Machine, or SVM, is typically used for the classification task. SVM algorithm finds a hyperplane that optimally divided the classes. It is best used with a non-linear solver. | Regression (not very common) Classification |
| **Random forest** | The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many times simple decision trees and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most vote; while for the regression task, the average prediction of all the trees is the final prediction. | Regression Classification |
| **AdaBoost** | Classification or regression technique that uses a multitude of models to come up with a decision but weighs them based on their accuracy in predicting the outcome | Regression Classification |
| **Gradient-boosting trees** | Gradient-boosting trees is a state-of-the-art classification/regression technique. It is focusing on the error committed by the previous trees and tries to correct it. | Regression Classification |

#### Unsupervised learning

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns)

You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Description** | **Type** |
| **K-means clustering** | Puts data into some groups (k) that each contains data with similar characteristics (as determined by the model, not in advance by humans) | Clustering |
| **Gaussian mixture model** | A generalization of k-means clustering that provides more flexibility in the size and shape of groups (clusters | Clustering |
| **Hierarchical clustering** | Splits clusters along a hierarchical tree to form a classification system.  Can be used for Cluster loyalty-card customer | Clustering |
| **Recommender system** | Help to define the relevant data for making a recommendation. | Clustering |
| **PCA/T-SNE** | Mostly used to decrease the dimensionality of the data. The algorithms reduce the number of features to 3 or 4 vectors with the highest variances. | Dimension Reduction |

**Application of Machine learning**

**Augmentation**:

* Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

**Automation**:

* Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

**Finance Industry**

* Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.

**Government organization**

* The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.

**Healthcare industry**

* Healthcare was one of the first industry to use machine learning with image detection.

**Marketing**

* Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign.

**Example of application of Machine Learning in Supply Chain**

Machine learning gives terrific results for visual pattern recognition, opening up many potential applications in physical inspection and maintenance across the entire supply chain network.

Unsupervised learning can quickly search for comparable patterns in the diverse dataset. In turn, the machine can perform quality inspection throughout the logistics hub, shipment with damage and wear.

For instance, IBM's Watson platform can determine shipping container damage. Watson combines visual and systems-based data to track, report and make recommendations in real-time.

In past year stock manager relies extensively on the primary method to evaluate and forecast the inventory. When combining big data and machine learning, better forecasting techniques have been implemented (an improvement of 20 to 30 % over traditional forecasting tools). In term of sales, it means an increase of 2 to 3 % due to the potential reduction in inventory costs.

**Example of Machine Learning Google Car**

For example, everybody knows the Google car. The car is full of lasers on the roof which are telling it where it is regarding the surrounding area. It has radar in the front, which is informing the car of the speed and motion of all the cars around it. It uses all of that data to figure out not only how to drive the car but also to figure out and predict what potential drivers around the car are going to do. What's impressive is that the car is processing almost a gigabyte a second of data.

Deep Learning

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks. The machine uses different layers to learn from the data. The depth of the model is represented by the number of layers in the model. Deep learning is the new state of the art in term of AI. In deep learning, the learning phase is done through a neural network.

**Reinforcement Learning**

Reinforcement learningis a subfield of machine learning in which systems are trained by receiving virtual "rewards" or "punishments," essentially learning by trial and error. Google's DeepMind has used reinforcement learning to beat a human champion in the Go games. Reinforcement learning is also used in video games to improve the gaming experience by providing smarter bot.

One of the most famous algorithms are:

* Q-learning
* Deep Q network
* State-Action-Reward-State-Action (SARSA)
* Deep Deterministic Policy Gradient (DDPG)

**Applications/ Examples of deep learning applications**

**AI in Finance:**The financial technology sector has already started using AI to save time, reduce costs, and add value. Deep learning is changing the lending industry by using more robust credit scoring. Credit decision-makers can use AI for robust credit lending applications to achieve faster, more accurate risk assessment, using machine intelligence to factor in the character and capacity of applicants.

Underwrite is a Fintech company providing an AI solution for credit makers company. underwrite.ai uses AI to detect which applicant is more likely to pay back a loan. Their approach radically outperforms traditional methods.

**AI in HR:**Under Armour, a sportswear company revolutionizes hiring and modernizes the candidate experience with the help of AI. In fact, Under Armour Reduces hiring time for its retail stores by 35%. Under Armour faced a growing popularity interest back in 2012. They had, on average, 30000 resumes a month. Reading all of those applications and begin to start the screening and interview process was taking too long. The lengthy process to get people hired and on-boarded impacted Under Armour's ability to have their retail stores fully staffed, ramped and ready to operate.

At that time, Under Armour had all of the 'must have' HR technology in place such as transactional solutions for sourcing, applying, tracking and onboarding but those tools weren't useful enough. Under armour choose **HireVue**, an AI provider for HR solution, for both on-demand and live interviews. The results were bluffing; they managed to decrease by 35% the time to fill. In return, the hired higher quality staffs.

**AI in Marketing:**AI is a valuable tool for customer service management and personalization challenges. Improved speech recognition in call-center management and call routing as a result of the application of AI techniques allows a more seamless experience for customers.

For example, deep-learning analysis of audio allows systems to assess a customer's emotional tone. If the customer is responding poorly to the AI chatbot, the system can be rerouted the conversation to real, human operators that take over the issue.

Apart from the three examples above, AI is widely used in other sectors/industries.

**Artificial Intelligence**

**Deep Learning**

**Machine Learning**

ML

## Difference between Machine Learning and Deep Learning

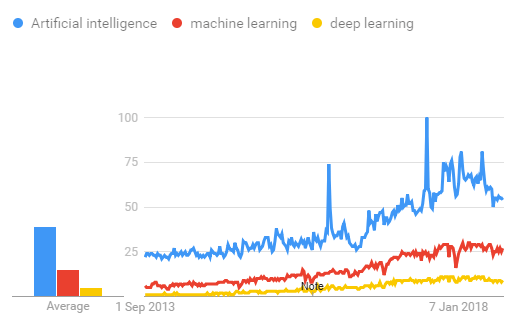
|  |  |  |
| --- | --- | --- |
|  | **Machine Learning** | **Deep Learning** |
| **Data Dependencies** | Excellent performances on a small/medium dataset | Excellent performance on a big dataset |
| **Hardware dependencies** | Work on a low-end machine. | Requires powerful machine, preferably with GPU: DL performs a significant amount of matrix multiplication |
| **Feature engineering** | Need to understand the features that represent the data | No need to understand the best feature that represents the data |
| **Execution time** | From few minutes to hours | Up to weeks. Neural Network needs to compute a significant number of weights |
| **Interpretability** | Some algorithms are easy to interpret (logistic, decision tree), some are almost impossible (SVM, XGBoost) | Difficult to impossible |

## When to use ML or DL?

In the table below, we summarize the difference between machine learning and deep learning.

|  |  |  |
| --- | --- | --- |
|  | **Machine learning** | **Deep learning** |
| **Training dataset** | Small | Large |
| **Choose features** | Yes | No |
| **Number of algorithms** | Many | Few |
| **Training time** | Short | Long |

With machine learning, you need fewer data to train the algorithm than deep learning. Deep learning requires an extensive and diverse set of data to identify the underlying structure. Besides, machine learning provides a faster-trained model. Most advanced deep learning architecture can take days to a week to train. The advantage of deep learning over machine learning is it is highly accurate. You do not need to understand what features are the best representation of the data; the neural network learned how to select critical features. In machine learning, you need to choose for yourself what features to include in the model.

[](https://www.guru99.com/images/tensorflow/083018_0454_MachineLear6.png)

## TensorFlow

the most famous deep learning library in the world is Google's TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations.

To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword a the search bar, Google provides a recommendation about what could be the next word.

Google wants to use machine learning to take advantage of their massive datasets to give users the best experience. Three different groups use machine learning:

* Researchers
* Data scientists
* Programmers.

They can all use the same toolset to collaborate with each other and improve their efficiency.

Google does not just have any data; they have the world's most massive computer, so TensorFlow was built to scale. TensorFlow is a library developed by the Google Brain Team to accelerate machine learning and deep neural network research.

It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, C++ or Java.

In this tutorial, you will learn

**TensorFlow Architecture**

Tensorflow architecture works in three parts:

* Preprocessing the data
* Build the model
* Train and estimate the model

It is called Tensorflow because it takes input as a multi-dimensional array, also known as **tensors**. You can construct a sort of **flowchart** of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

**Where can Tensorflow run?**

TensorFlow can hardware, and software requirements can be classified into

Development Phase: This is when you train the mode. Training is usually done on your Desktop or laptop.

Run Phase or Inference Phase: Once training is done Tensorflow can be run on many different platforms. You can run it on

* Desktop running Windows, macOS or Linux
* Cloud as a web service
* Mobile devices like iOS and Android

You can train it on multiple machines then you can run it on a different machine, once you have the trained model.

The model can be trained and used on GPUs as well as CPUs. GPUs were initially designed for video games. In late 2010, Stanford researchers found that GPU was also very good at matrix operations and algebra so that it makes them very fast for doing these kinds of calculations. Deep learning relies on a lot of matrix multiplication. TensorFlow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, TensorFlow can be accessed and controlled by other languages mainly, Python.

Finally, a significant feature of TensorFlow is the TensorBoard. The TensorBoard enables to monitor graphically and visually what TensorFlow is doing.

**List of Prominent Algorithms supported by TensorFlow**

* Linear regression: tf.estimator.LinearRegressor
* Classification:tf.estimator.LinearClassifier
* Deep learning classification: tf.estimator.DNNClassifier
* Deep learning wipe and deep: tf.estimator.DNNLinearCombinedClassifier
* Booster tree regression: tf.estimator.BoostedTreesRegressor
* Boosted tree classification: tf.estimator.BoostedTreesClassifier

**PYTHON OVERVIEW**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted:** Python is processed at runtime by the interpreter.You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive:** You can actually sit at a Python prompt and interactwith the interpreter directly to write your programs.
* **Python is Object-Oriented:** Python supports Object-Oriented style ortechnique of programming that encapsulates code within objects.
* **Python is a Beginner's Language:** Python is a great language for thebeginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Python Features**

Python's features include:

* **Easy-to-learn:** Python has few keywords, simple structure, and a clearlydefined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
* **A broad standard library:** Python's bulk of the library is very portable andcross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode:** Python has support for an interactive mode which allowsinteractive testing and debugging of snippets of code.
* **Portable:** Python can run on a wide variety of hardware platforms and has thesame interface on all platforms.
* **Extendable:** You can add low-level modules to the Python interpreter. Thesemodules enable programmers to add to or customize their tools to be more efficient.
* **Databases:** Python provides interfaces to all major commercial databases.
* **GUI Programming:** Python supports GUI applications that can be created andported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable:** Python provides a better structure and support for large programsthan shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

* IT supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**PYTHON ENVIRONMENT**

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

**REQUIREMENTS ANAYLSIS**

**SOFTWARE REQUIREMENTS**

* Python
* Anaconda Navigator
* Python built-in modules
* Numpy
* Pandas
* Matplotlib
* Sklearn
* Seaborm

**ANACONDA NAVIGATOR**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

## Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

## **WHAT APPLICATIONS CAN I ACCESS USING NAVIGATOR**?

The following applications are available by default in Navigator:

* JupyterLab
* Jupyter Notebook
* QTConsole
* Spyder
* VSCode
* Glueviz
* Orange 3 App
* Rodeo
* RStudio

Advanced conda users can also build your own Navigator applications

## How can I run code with Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.

You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

## What’s new in 1.9?

* Add support for **Offline Mode** for all environment related actions.
* Add support for custom configuration of main windows links.
* Numerous bug fixes and performance enhancements.

**PYTHON**

**Python**

Python is a general-purpose, versatile and popular programming language. It's great as a first language because it is concise and easy to read, and it is also a good language to have in any programmer's stack as it can be used for everything from web development to soitware development and scientific applications.   
  
It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time.  **Features of Python**   
  
A simple language which is easier to learn, Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.

* **Free and open source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes to the Python's source code. Python has a large community constantly improving it in each iteration.

* **Portability**  
  You can move Python programs from one platform to another, and run it without any changes.   
  It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.
* **Extensible and Embeddable**   
  Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code. This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.
* **A high-level, interpreted language**

Unlike C/C++, you don’t have to worry about daunting tasks like memory management, garbage collection and so on.   
Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower level operations.

* **Large standard libraries to solve common tasks**   
  Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server You can use MySQLdb library using import MySQL db Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.
* **Object-oriented**

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.   
With OOP, you are able to divide these complex problems into smaller sets by creating object

**Python**

**History and Versions**:

Python is predominantly a dynamic typed programming language which was initiated by Guido van Rossum in the year 1989. The major design philosophy that was given more importance was the readability of the code and expressing an idea in fewer lines of code rather than the verbose way of expressing things as in C++ and Java [K-8][K-9]. The other design philosophy that was worth mentioning was that, there should be always a single way and a single obvious way to express a given task which is contradictory to other languages such as C++, Perl etc. [K-10]. Python compiles to an intermediary code and this in turn is interpreted by the Python Runtime Environment to the Native Machine Code. The initial versions of Python were heavily inspired from lisp (for functional programming constructs). Python had heavily borrowed the module system, exception model and also keyword arguments from Modula-3 language [K-10]. Pythons’ developers strive not to entertain premature optimization, even though it might increase the performance by a few basis points [K-9]. During its design, the creators had conceptualized the language as being a very extensible language, and hence they had designed the language to have a small core library which was extended by a huge standard library [K-7]. Thus as a result, python is used as a scripting language as it can be easily embedded into any application, though it can be used to develop a full-fledged application. The reference implementation of python is CPython. There are also other implementations like Jython, Iron Python which can use python syntax as well as can use any class of Java (Jython) or .Net class (Iron Python). Versions: Python has two versions 2.x version and 3.x version. The 3.x version is a backward incompatible release was released to fix many design issues which plagued the 2.x series. The latest in the 2.x series is 2.7.6 and the latest in 3.x series is 3.4.0. 1.5.2 Paradigms: Python supports multi-paradigms such as: Object-Oriented, Imperative, Functional, Procedural, and Reflective. In Object-Oriented Paradigm, Python supports most of the OOPs concepts such as Inheritance (It also has support for Multiple Inheritance), Polymorphism but its lack of support for encapsulation is a blatant omission as Python doesn’t have private, protected members: all class members are public [K-11]. Earlier Python 2.6 versions didn’t support some OOP’s concepts such as Abstraction through Interfaces and Abstract Classes [K-19]. It also supports Concurrent paradigm, but with Python we will not be able to make truly multitasking applications as the inbuilt threading API is limited by GIL (Global Interpreter Lock) and hence applications that use the threading API cannot run on multi-core parallelly [K-12].The only remedy is that, the user has to either use the multi-processing module which would fork processes or use Interpreters that haven’t implemented GIL such as Jython or Iron Python [K-12]. 1.5.3 Compilation, Execution and Memory Management: Compilation, Execution and Memory Management: 21 A Comparative Studies of Programming Languages (Comparative Studies of Six Programming Language) Just like the other Managed Languages, Python compiles to an intermediary code and this in turn is interpreted by the Python Runtime Environment to the Native Machine Code. The reference implementation (i.e. CPython) doesn’t come with a JIT compiler because of which the execution speed is slow compared to native programming languages [K-17]. We can use PyPy interpreter as it includes a JIT compiler rather than using the Python interpreter that comes by default with the python language, if speed of execution is one of the important factors [K-18]. The Python Runtime Environment also takes care of all the allocation and deallocation of memory through the Garbage Collector. When a new object is created, the GC allocates the necessary memory, and once the object goes out of its scope, the GC doesn’t release memory immediately but instead it becomes eligible for Garbage Collection, which would eventually release the memory. Typing Strategies: Python is a strongly dynamic typed language. Python 3 also supports optional static typing [K-20]. There are a few advantages in using a dynamic typed language, the most prominent one would be that the code is more readable as there is less code (in other words has less boiler-plate code). But the main disadvantage in having python as a dynamic programming language is that there would be no way to guarantee that a particular piece of code would run successfully for all the different data-types scenarios simply because it had run successfully with one type. Basically, we don’t have any means to find out an error in the code, till the code has started running. 1.5.4 Strengths and Weaknesses and Application Areas: Python is predominantly used as a scripting language used in developing standalone applications that are being developed with Static-Typed languages, because of the flexibility it provides due to its dynamic typed nature. Python favours rapid application development, which qualifies it to be used for prototyping. To a certain extent, Python is also used in developing websites. Due to its dynamic typing and of the presence of a Virtual Machine, there is a considerable overhead which translates to way less performance when we compare with native programming languages [K-13]. And hence it is not suited

**NUMPY**

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences: • NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original. • The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements. • NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences. • A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today’s scientific/mathematical Python-based software, just knowing how to use Python’s built-in sequence types is insufficient - one also needs to know how to use NumPy arrays. The points about sequence size and speed are particularly important in scientific computing. As a simple example, consider the case of multiplying each element in a 1-D sequence with the corresponding element in another sequence of the same length. If the data are stored in two Python lists, a and b, we could iterate over each element:

The Numeric Python extensions (NumPy henceforth) is a set of extensions to the Python programming language which allows Python programmers to efficiently manipulate large sets of objects organized in grid-like fashion. These sets of objects are called arrays, and they can have any number of dimensions: one dimensional arrays are similar to standard Python sequences, two-dimensional arrays are similar to matrices from linear algebra. Note that one-dimensional arrays are also different from any other Python sequence, and that two-dimensional matrices are also different from the matrices of linear algebra, in ways which we will mention later in this text. Why are these extensions needed? The core reason is a very prosaic one, and that is that manipulating a set of a million numbers in Python with the standard data structures such as lists, tuples or classes is much too slow and uses too much space. Anything which we can do in NumPy we can do in standard Python – we just may not be alive to see the program finish. A more subtle reason for these extensions however is that the kinds of operations that programmers typically want to do on arrays, while sometimes very complex, can often be decomposed into a set of fairly standard operations. This decomposition has been developed similarly in many array languages. In some ways, NumPy is simply the application of this experience to the Python language – thus many of the operations described in NumPy work the way they do because experience has shown that way to be a good one, in a variety of contexts. The languages which were used to guide the development of NumPy include the infamous APL family of languages, Basis, MATLAB, FORTRAN, S and S+, and others. This heritage will be obvious to users of NumPy who already have experience with these other languages. This tutorial, however, does not assume any such background, and all that is expected of the reader is a reasonable working knowledge of the standard Python language. This document is the “official” documentation for NumPy. It is both a tutorial and the most authoritative source of information about NumPy with the exception of the source code. The tutorial material will walk you through a set of manipulations of simple, small, arrays of numbers, as well as image files. This choice was made because: • Aconcrete data set makes explaining the behavior of some functions much easier to motivate than simply talking about abstract operations on abstract data sets; • Every reader will at least an intuition as to the meaning of the data and organization of image files, and • The result of various manipulations can be displayed simply since the data set has a natural graphical representation. All users of NumPy, whether interested in image processing or not, are encouraged to follow the tutorial with a working NumPy installation at their side, testing the examples, and, more importantly, transferring the understanding gained by working on images to their specific domain. The best way to learn is by doing – the aim of this tutorial is to guide you along this “doing.”

**System Implementation:**

**Modules:**

**Exploratory Data Analysis:**

During this step we performed some descriptive analysis and determined the target variable. We also explored how many classes were in the target and a selection of other possibly problematic (high cardinality) variables. I also visualized the target variable in a histogram which is a good technique for understanding the distribution of the data to assist in parameter tuning.

**Data Cleaning**

We dropped those high cardinality variables during this step as a precursor to the pre-processing step.

**Pre-processing & Transformation**

We removed the target variable from the entire data set and transformed the categorical variable into a model matrix with one-hot encoding. This is sometimes the requirements for certain algorithms to process the data in a sparse matrix format. Other statistical software such as R, automates this step when generating models. I imputed the missing values in the data to 0. I scaled the continuous variables using min-max normalization which transforms values onto a scale from 0 to 1 to prevent variables on different scales heavily impacting the coefficients.

**Data Partition**

We partitioned the pre-processed data into a training and test data set. Modelling: We built a k-NN classifier model, using 10 neighbour classes and the Euclidean distance.

**Evaluation**

We scored the classifier on unseen test data and calculated the R squared values for both the training and test data.

**DATA FLOW DIAGRAM**

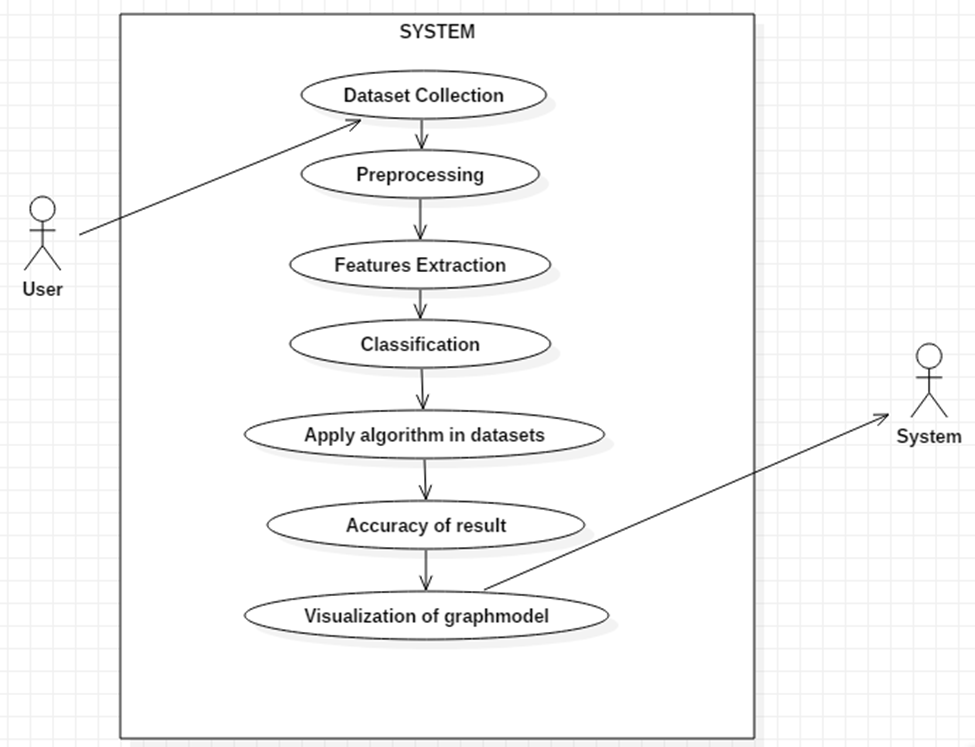
LEVEL 0

LEVEL 1

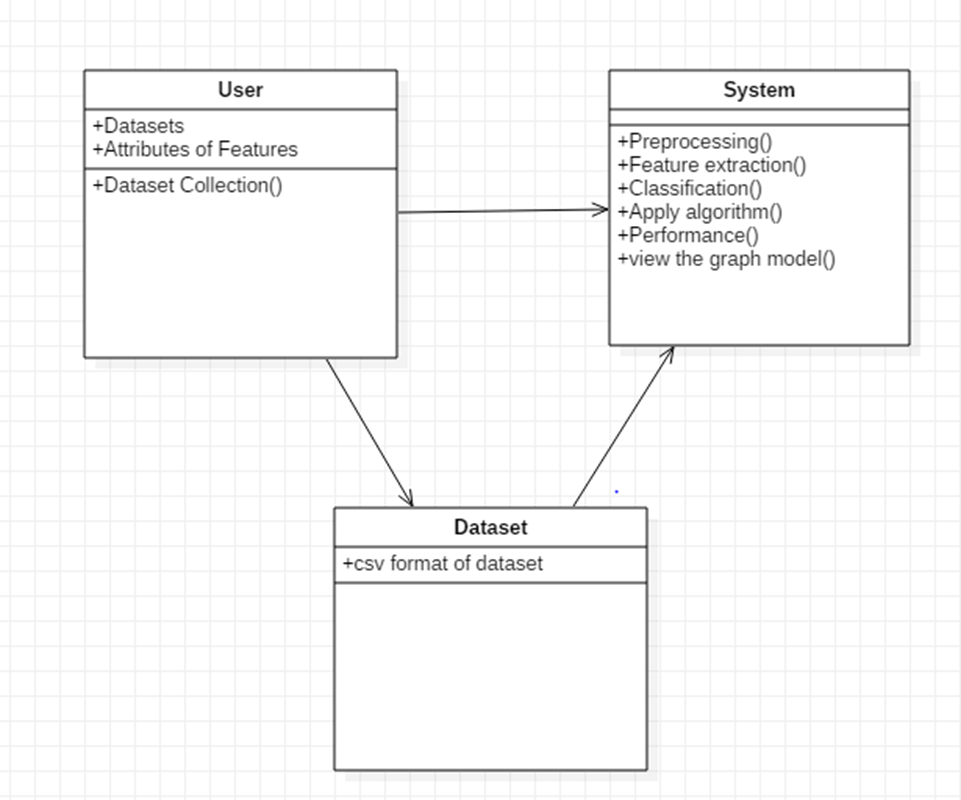
LEVEL 2

**UML DIAGRAMS**

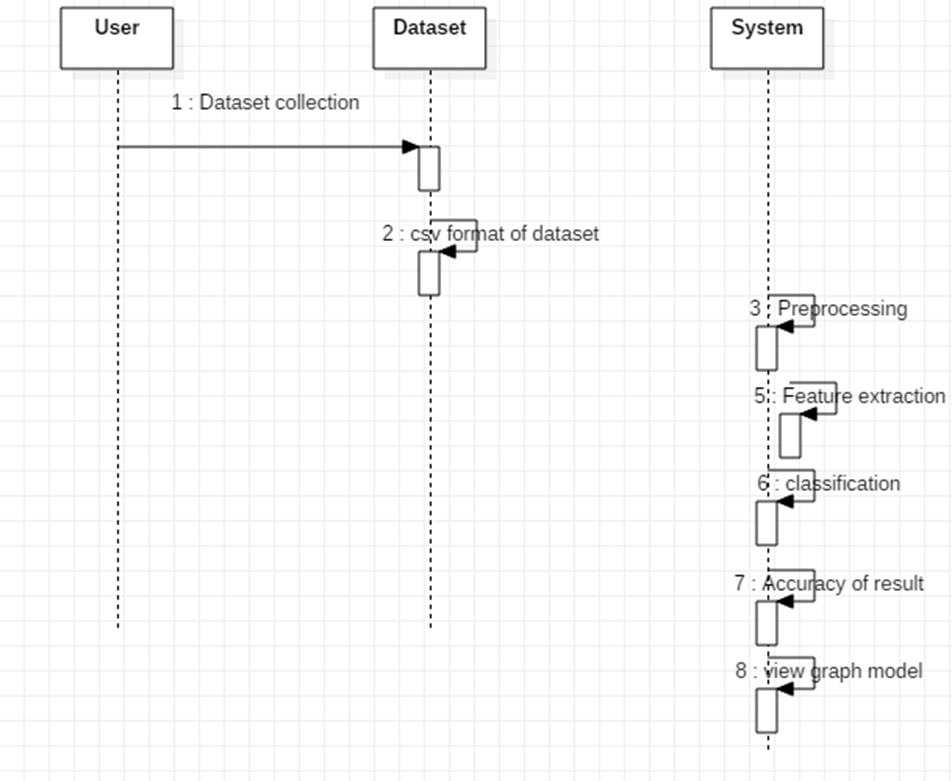
**Use case diagram:**



**Class diagram:**



**Sequence diagram:**



**ACTIVITY DIAGRAM:**

