National College of Ireland

Twitter Rumour Detection using Temporal Property of Tweets

MSc Research Project
Data Analytics

Nikita Nitin Parab
Student ID: x17166136

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Nikita Nitin Parab
Student ID:	x17166136
Programme:	Data Analytics
Year:	2019
Module:	MSc Research Project
Supervisor:	Dr. Muhammad Iqbal
Submission Due Date:	12/08/2019
Project Title:	Twitter Rumour Detection using Temporal Property of Tweets
Word Count:	7416
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	10th August 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Contents

1 Introduction **1**
- 1.1 Domain Overview . . . 1
- 1.2 Motivation . . . 2
- 1.3 Research Objective . . . 3

2 Related Work **3**
- 2.1 Twitter Analysis . . . 3
- 2.2 Rumour Detection based on Feature of Comments . . . 4
- 2.3 Rumour Detection based on Conversation Structure . . . 5
- 2.4 Rumour Detection based on Network Patterns . . . 5
- 2.5 Rumour Detection based on Temporal Properties . . . 6
- 2.6 Conclusion . . . 6

3 Methodology **7**
- 3.1 Data Selection . . . 8
- 3.2 Pre-Processing . . . 8
- 3.3 Transformation . . . 9
- 3.4 Data Mining . . . 9
- 3.5 Evaluation . . . 10

4 Implementation **11**
- 4.1 Design Specification . . . 11
- 4.2 Data Analysis . . . 12
- 4.3 Data Preparation . . . 12
- 4.4 Implementation . . . 13

5 Evaluation **15**
- 5.1 Improved Support Vector Machine . . . 15
- 5.2 Evaluating Classifiers . . . 15
  - 5.2.1 Experiment 1: Support Vector Machine . . . 15
  - 5.2.2 Experiment 2: Random Forest . . . 15
  - 5.2.3 Experiment 3: Gaussian Naive Bayes . . . 16
  - 5.2.4 Experiment 4: Classification and Regression Trees . . . 16
  - 5.2.5 Experiment 5: K Nearest Means . . . 17
  - 5.2.6 Experiment 6: Deep Learning . . . 17

6 Discussion **18**

7 Conclusion and Future Work **20**

Twitter Rumour Detection using Temporal Property of Tweets

Nikita Nitin Parab
x17166136

Abstract

There are mainly two type of rumours on social media, it can be disinformation or misinformation. While disinformation is intentional and spread to divert people from truth, misinformation is unintentional news which turns out to be false. Either ways, it causes panic and needs to be identified as quickly as possible to avoid further chaos. This research paper presents a method for detecting rumour in the first hour the tweet was posted. The social media platform used for this project is twitter due to its popularity. This research is implemented on 7 events which have different propagation patterns. The feature used for this project is timestamp of the tweets and its reactions. The models implemented include Gaussian Naive Bayes, Support Vector Machine, Random Forest, Classification and Regression Trees, K Nearest Neighbours and Deep Learning. All these models are implemented to study how they perform for different propagation patterns. Overall, it can be seen that most of the models work best with events that have well defined pattern and work poor if the events have less samples or do not have defined patterns. The highest accuracy recorded of 78% was for Charlie Hebdo event for both CART and SVM models. The lowest accuracy of 19% was for Gurlitt event after implementing Deep Learning Model.

1 Introduction

1.1 Domain Overview

Internet has gained popularity over the past few decades. New technologies were developed along with the advancement of internet. From just being a search engine and sending work emails to calling and video chatting, internet has opened a whole new horizon of possibilities. It has also become an aid for those who wish to stay in touch with each other with the help of social media applications and messengers. It has become very easy to keep in touch with people who stay far away because of social media. Calling someone millions of miles away is just a matter of minutes due to the advancement in technology. Not only this, social media platforms like Twitter also let people follow famous celebrities which keeps them updated about their well being. Now a days, some people use twitter as means to raise flags against something or support some cause. Famous people like celebrities, political parties etc. use this as a medium to promote various causes. According to statista[1] the number of twitter users have been increasing in a continuous trend over the past few yearly quarters.

[1] https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

This social media platform also acts like a informing people about the current happenings around the world, especially during big events. This events attracts a lot of attention of all the people. It is therefore very important to know the credibility of the tweets. Some people post false tweets without realizing the repercussions caused by it. For example, a tweet about two bomb explosions at the White House and the President of U.S being injured was shared by an official press account on twitter in 2013 (Sicilia et al.; 2018). It later turned out to be a hoax as the account was hacked. As this tweet was generated by a trustable source people posted and commented on it without confirming its reality. It hence becomes very important to detect these rumours as soon as possible and take necessary action against them. Madhav Kotteti et al. (2019) detected that rumours and non rumours have different propagation patterns. This feature was used to distinguish between the both. This research paper presents an advancement on this feature.

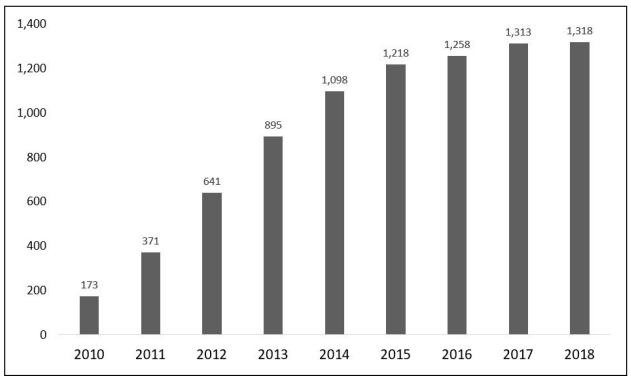

Figure 1: Twitter Users

1.2 Motivation

This research project first extracts the temporal properties to analyze the propagation patterns of both rumours and non-rumours for all the data available. The data used for this project is a public dataset by PHEME dataset (2016).

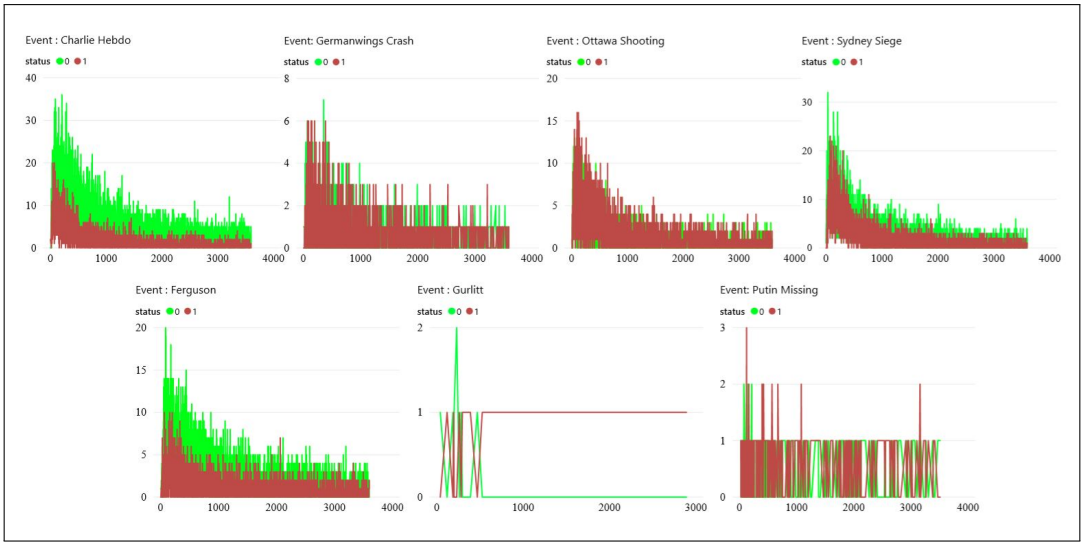

Figure 2: Tweets propagation patterns

The graphs in figure 2 show the propagation of rumours and non-rumours. The x-axis denotes the time interval. It is in second format as this research project considers the tweets posted within an hour of the source tweet. The y-axis denotes the number of interactions that take place at each second. As it can be seen for most of the events the number of interaction taking place for rumours is less than the interactions taking place for non-rumours. It is difficult to differentiate between rumours and non-rumours for

the event of Ottawa Shooting and Germanwings Crash. The events Putin Missing and Gurlitt have very few tweets and hence the propagation patterns cannot be seen properly like other events. This research project will focus on developing machine learning models which will learn from these propagation patterns and will be able to detect rumours and non-rumours within the first hour of the tweet.

1.3 Research Objective

It is important to detect the rumours at an early stage to avoid causing any chaos. This research project aims at achieving the same. The use of only one feature which is the temporal property of the tweets decreases the computational power and further helps in detecting rumours more efficiently. The research question of this project is **How expeditiously can temporal property be used for detecting rumours on twitter for events with different propagation patterns?** The aim of this project is as follows:

- Convert all data into time series format.
- Extract data that belongs to the first hour of tweet.
- Apply machine learning models which include Support Vector Machine (SVM), Gaussian Naive Bayes (GNB), K Nearest Neighbour (KNN), Random Forest (RF), Classification and Regression Trees (CART) and Deep Learning.
- Evaluate the models to perceive how they work for events with different kind of propagation patterns.

The rest of the paper of this report is organized as follows, Section 2 discusses the contribution of other researchers to this field. Methodology followed for this project is explained in Section 3. Implementation stages are briefly discussed in Section 4. The results are evaluated in Section 5 followed by conclusion in Section 7.

2 Related Work

Rumour detection is a very important factor to be considered in this era. It is essential to keep everything in balance and avoid unnecessary chaos. This section discusses the various work done by researchers in this fields. Many researches have been conducted to detect rumour using various properties of the tweets or blogs. The sub sections below shed light on this researches.

2.1 Twitter Analysis

Twitter is one of the popular websites of social media. It has a lot of information for many years. This information can be used to gain insights about various topics. This section discuss some researches undertaken using data which is available from twitter. Data available from twitter was used to analyze and organize traffic services by Ounsrimuang and Nootyaskool (2019). The research aimed at classifying tweets to retrieve the status of traffic, level of accident, type of accident and GPS location for taking appropriate action. The project was based on the country of Thailand, hence only Thai tweets were considered. Tweets were collected from a page named FM91 Trafficpro which posted tweets regarding the traffic conditions. This research project was successfully implemented and could classify the tweets appropriately.

Another purpose where twitter was used is for detecting the credibility of tweets. This was done by Hassan et al. (2018) who used supervised learning approaches for the same. For example if someone is posting a tweet regarding an accident, what is the proof that the content is credible, as anyone can post on twitter. This is why this research project was important. The author has used source based and content based features for detecting credibility as the single parameters and also together. This has caused to understand the improvement by using both features together and also to understand which feature is more reliable for detecting credibility. The five supervised learning classifiers used included Naive Bayes, Logistic Regression, Random Forests, Support Vector Machines and K-Nearest Neighbor. After implementing the project the author stated that Random Forests performed the best compared to others.

Twitter is used as a platform for sharing thoughts, but some people take undue advantage of this platform. Online public shaming has been on the rise since the time social media started existing. It has now become very important to detect such posts and mitigate it. Basak et al. (2019) implemented a project that could detect, analyze and mitigate such tweets. The tweets were divided into six types which included comparison, passing judgment, sarcasm, whataboutery, religious and abusive. The author also observed a pattern wherein the number of followers of the shamers increases much faster than the non shamers. Overall the author was able to successfully implement the project. Another con of using social media are the rumours. Rumours spread very quickly and these were analyzed by Hashimoto et al. (2011). The author has developed a rumour analysis framework for all social media platforms. This platform classifies the topic structures and the time variations to detect rumours by visualizing them. The framework was successful in detecting rumours and analyzing them.

2.2 Rumour Detection based on Feature of Comments

Users behavior was used to classify rumours and non-rumours by Chen et al. (2018), who treated rumours as an anomaly. Efficient results were acquired due the use of Neural Networks. A model was built using the user behaviour on recent microblogs. Comments of users on a certain post called crowd wisdom was also taken into consideration for this model, which helped in making it more efficient. Rumours were classified with the help of Recurrent Neural Network. To improve the results, autoencoder was used for the first time. Using unsupervised models removed the need of training it with labelled data. Weibo microblog was used for this project which consisted of around 167,700 posts and 1,501,500 comments. All this methods resulted in high accuracy of 92.41%. Xu et al. (2018) used a similar way for detecting rumours. The author used a Merged Neural Network Model for an increase in accuracy. The original post and the re-tweets were treated separately as they play different roles. Different models were built for both features. One single merged model was then built using the two models for an attention based neural network. Diffusion process was used to extract important re-tweets and model was built based on these re-tweets and the original post. User information used for this model included features like location, age, mutual followers, gender, number of followers etc.

Another author to use microblogging website for detecting rumours is Ma et al. (2015). The author however used temporal property along with feature changes over time to detect rumours. The author built a Dynamic Series-Time Structure (DSTS) for the same. DSTS was responsible for saving the spectrum fluctuations of social media. The models used for this project included SVM, Decision Trees, Random Forest, Extended Random

Forest and SVM with DSTS. The proposed model worked better than the traditional methods. SVM was also used by Sicilia et al. (2018) for detecting rumours in the health domain on twitter. The factors considered for doing it were personal interests, network characteristics and influence potential. Prediction about a tweet being retweeted was also achieved by the author. Along with SVM other models used for this project included Multiclass Adaboost, Random Forest, Multi-layer Perceptron. Random forest performed the best with around 90% accuracy. Unlike this project, 22 features were considered for detecting rumours by Mahmoodabad et al. (2018) on twitter. The project was implemented for detecting Persian rumours. All 22 features were majorly divided into three groups namely; the demographic features, the content features and the structural features. Models used for this project included Multi Layer Perceptron, KNN, Decision tree, Naive Bayes and Random Forest. Random Forest performed the best in all this models.

2.3 Rumour Detection based on Conversation Structure

Conversation structure deals with how people interact with the tweet. This feature was used by Poddar et al. (2018) to predict the veracity of rumours on twitter with neural networks. The two steps involved for achieving the end result involved detecting conversation tree of the tweet with its timestamp and structure, and predicting the veracity of tweets using the extracted features. The labels used for veracity were support, deny, query and comment. These labels helped in further tagging them if it was rumour by true, false and unverified. Recurrent Neural Network was used for saving the conversation tree as it is efficient in dealing with sequential data. Text of the tweets was encoded using Convolution Neural Networks. This made a two level attention model which outperformed old models. The aggregate of prediction was used to analyze the veracity of tweets. Akhtar et al. (2018) further enhanced this project by predicting the veracity of tweet with the aim of checking the authenticity of the post. Same labels were used for this project too. However this project used a different dataset named SemEval-2017.

The machine learning models used for classification included Decision Tree, Naive Bayes, SVM, Multi Layer Perceptron. Vanilla LSTM was later used over this to predict the veracity of tweets. This project gave an accuracy of 78.61%. Conversational trees give out a lot of information . This information was also used by Yavary and Sajedi (2018) for detecting rumours. The dataset used was the same as previous project for analyzing the conversation trees to detect rumours. The parameter used for detecting rumour was the feedback a tweet received. A single layer feedforward neural network named ELM was used for the deployment of this project. The project had an overall good performance. Structural aspect of tweets was used by Tai et al. (2018) to detect rumours using Recurrent Neural Network (RNN). The author first classified the tweet text according to the pattern and then used it to build a RNN model which generated review. Semantic and Syntactic integration helped increase the quality of reviews in social network. The data used for this collected tweets based on politics over the period of 2012 to 2017. The author stated that the increase in training sample can improve the accuracy much further.

2.4 Rumour Detection based on Network Patterns

European fox rabies SIR model was used to build a rumour spreading model by SuyalatuDong et al. (2018). The model combined the changes in the number of users and

population dynamics for spreading rumours. The main goal of the model was to study the network for understanding the propagation of rumours. According to the author the rumours lasted for a limited amount of time as it was assumed to not occur in an open system. This project considered this possibility by taking into account the changing number of users. Other parameters which were considered included registration of new users, inactive users, network growth for a month. The project was carried out on Facebook data with 15 users and their friend list. The results showed that once a certain threshold of environmental capacity is passed rumours spread or they do not. Wang et al. (2017) also used network patterns for understanding the propagation of rumours. This enabled the author to detect rumours at very early stages. Methodologies used for implementing the project included Sliding Window and Window based Patterns.

Multiple datasets were used by the author such as KAIST which consisted of 109 trending topics from 2006 to 2009. Another data was collected related to the topic Zika Virus. The model development phase had two steps which included identification of tweets to model and cluster them accordingly; the second step was to verify the topics with trusted sources. The tweets were tagged as alarm, normal or detected according to the content. The author claimed that this method could identify rumours around 24 days and 38 hours prior to the peak window.

2.5 Rumour Detection based on Temporal Properties

The research project is based on the temporal property of tweets. This section will discuss some researches of a similar fashion. Temporal property was used by Kwon et al. (2017) to classify rumours. But the author also took into account other factors which included user and linguistic features of tweets. A comparison of all features was done on the basis of which features could detect rumours the fastest and earliest. As per the analysis timestamp could successfully detect rumours in long term windows. Rumours were detected in their initial phases with the use of user and linguistic features. Combining these two observations, a new model was built which could identify rumours in short as well as long term windows. This project followed a similar approach like SuyalatuDong et al. (2018) who considered the change in users. The data used ranged for around 3 years which consisted of data from the initial 3 days to 56 days of circulation. The parameters included for rumour detection where the user information, linguistic characteristics, structural properties and temporal property. Out of all these, the structural characteristic performed the worst.

According to this author rumour cannot be detected in the initial phases with the temporal property. However, Madhav Kotteti et al. (2019) detected rumours within the first hour by using just the temporal property. The author divided the data into various time intervals. These time interval data was fed into various models to create sub models. A majority vote was taken to get the final result if the tweet is a rumour or not. The data consisted of 5 events which had 6,000 tweets out of which, 1,972 were rumours. A total of 8 models were implemented which included Gaussian Naive Bayes, SVM, MLP, Decision Tree, KMeans, Birch, Random Forest and Logistic Regression. SVM gave the worst performance while Gaussian Naive Bayes performed the best.

2.6 Conclusion

Data extracted from twitter can be used for many researches. Some of them mentioned are implemented by Ounsrimuang and Nootyaskool (2019), Hassan et al. (2018) and Basak

et al. (2019). This research is focused on rumour detection which was implemented by Hashimoto et al. (2011) but using topic structure and time variations. Further Chen et al. (2018), Xu et al. (2018), Ma et al. (2015) and Sicilia et al. (2018) used comments on twitter and other microblogging websites to detect rumours. Some authors used comments along with other features while some used just the comments for understanding how well it can be used for detecting rumours. While this approaches gave very good results, multiple features were used for implementing it.

In order to use less features and give better results some authors like Poddar et al. (2018), Akhtar et al. (2018), Yavary and Sajedi (2018) and Tai et al. (2018) used conversation structure also known as stances to identify rumours and non-rumours. Multiple features of conversation structure of tweets were used to detect rumours. While these approaches used less features than the previous researches, they still used multiple features. Researchers like SuyalatuDong et al. (2018) and Wang et al. (2017) used network patterns to identify rumours. This feature gave very good results and was able to detect rumours at very early stage, but the data used for this was hard to collect as it involved the propagation of network patterns. Finally, temporal properties along with other properties were considered by Kwon et al. (2017) and SuyalatuDong et al. (2018) to detect rumours. Using timestamps gave very good results. This was noted by Madhav Kotteti et al. (2019) who used only the timestamps to detect rumours. The author divided the timestamps into 5 time intervals. This created complexity in detecting rumours. This research project uses timestamps without dividing them into any groups.

3 Methodology

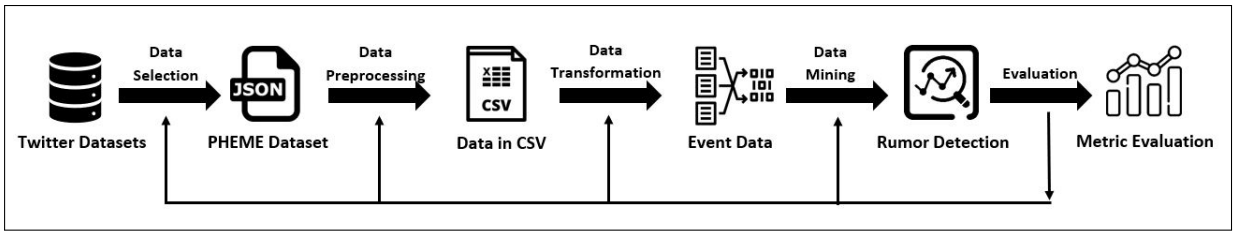

Figure 3: KDD for rumour detection [2]

Data Mining is huge process which involves many steps right from data creation. To simplify the process of data mining and knowledge gained with that many methodologies have been developed. Some of them are Knowledge Discovery in Databases (KDD), CRISP-DM and SEMMA. The most popular methodology is KDD. This project too follows the KDD methodology.

KDD is a method of acquiring useful knowledge from data. It is a method for developing methods and techniques to make sense of the data by gaining insights from it. It consists of 6 steps as shown in figure 3 which starts right from data collection or selection to gaining insights or knowledge. These steps for this research project have been discussed in the points below.

[2]https://icon-library.net/icon

3.1 Data Selection

This is the most important step in any project as selecting wrong data can result in giving unreliable results. It is important to check the credibility of data before using it. This can be done by using data from credible resources.

This research project uses data by Zubiaga et al. (2016) which is available online as a public dataset. The data consists of events some of which were likely to spark rumours while some rumours were known *a priori*. Tweets which are known *a priori* are searched using specific keywords. The dataset consists of 9 events out of which 5 events were breaking news and likely to consist many rumours while the remaining 4 events were collected as *a priori*.

The events Ferguson unrest, Ottawa shooting, Sydney siege, Charlie Hebdo shooting, germanwings plane crash actually happened but sparked many rumours about it. Hence it has a combination of rumours and non-rumours. The *a priori* events Prince Toronto, Gurlitt collection, Putin Missing, Michael Essien contracted Ebola were partially true while others were totally false. Hence this events have many rumours and very less or no non-rumours.

All tweets were collected using the streaming API and tracked at the same time. These tweets were then annotated using an annotation tool by journalists (Zubiaga et al.; 2016). This makes the data legitimate as it has been annotated by a group of experience journalists.

3.2 Pre-Processing

The base of the whole implementation phase in any project is formed by the pre-processing stage. This step consists of cleaning, pre-processing and outlier removal etc. After downloading data it was extracted from tar extension using WinRAR software [3]. After extracting the data, RStudio[4] was used for understanding and pre-processing the data. The data was converted from JSON to dataframe for further processing. The data consisted of multiple files in JSON format which contained the whole information of a single tweet. The information included the username, location, content of tweet, date, time etc. Out of all these variables only the time variable was extracted.

After creating a dataframe of each event which consisted the timestamp of the source tweet and reaction tweets, difference between the two was calculated using the following formula:

$$D(c_{mn}) = R_{mn} - S_{mn}$$

where,

- m represents the rumours while n represent non-rumours.
- R_{mn} is the time-stamp of reaction for the source tweet S_{mn}.
- $D(c_{mn})$ is the difference between source tweet and reaction tweet for a conversation c_{mn}

This step was repeated for both rumours and non-rumours for each event.

[3] `https://www.rarlab.com/`

[4] `https://www.rstudio.com/products/rstudio/download/`

3.3 Transformation

The time stamps were in HH:MM:SS format and were converted to seconds. To understand the propagation of rumours and non-rumours it was important to know how frequent were the tweets at any given time. To see this a frequency table was created which counted the number of tweets an event had at any given point of time.

A new column was added which showed the status of the tweet time. It could be 0 or 1 where 0 denotes non-rumour while 1 denotes rumour. This helped in distinguishing between the both. Before training the data for models the status column of each event was converted to factor data type as it was a categorical dependent variable. Graphs were made to analyze the propagation of rumours and non-rumours for each event. This data was then trained using different models for classification.

3.4 Data Mining

This research project classifies tweets either as rumours or non-rumours. Hence classification algorithms are used for this research project. In total this research project implements 6 classifiers which use different methods. The justification for using specific models is given below.

- Gaussian Naive Bayes
  Gaussian Naive Bayes is an extended version of the classic Naive Bayes. Naive Bayes requires all variables to categorical. But the data used for this research project contains numeric data. Hence, Gaussian Naive Bayes which considers the data to be Gaussian. This classifier is also selected as Naive Bayes is a known as one of the best classifier models. This classifier was also used by Madhav Kotteti et al. (2019) who has done a similar project and acquired the best results with this classifier. This classifier uses partial_fit method which is useful in large datasets as it considers chunks of data while training the classifier.

- Support Vector Machine
  Support Vector Machine is a popular supervised learning algorithm used for classification of data. It is known to be highly accurate while using less computation power. It works on the principle of hyperplane. This algorithm divides the by inserting hyperplanes in N-dimensional space till the data is distinctly classified. There are various kernels who have different functions for for classification. Madhav Kotteti et al. (2019) used 'rbf' kernel and acquired negligible results. This was overcome with the use of 'radial' kernel. Using this kernel improved the results greatly. The results are stated in Section 5.1.

- Random Forest
  Random Forest can be imagined as a combination trees working as an ensemble model. This classifier builds many decision trees and takes majority vote to get the final prediction. This works better than decision trees as in this the decision trees work as group and overcome each others errors. In random forest each tree is trained using different chunks using bagging of data as well as features. This makes the trees uncorrelated to each other.

- Classification and Regression Trees
  Classification and Regression Trees (CART) can also be called decision tree. This model builds a tree by splitting the training data. When new data is given to this

trained classifier, it traverses the previously formed tree to see which leaf does the data land. CART is the modern name of decision tree. The deeper the tree the complex are the rules of decision making. The main advantage of this classifier is that both numerical and categorical data can be used for building this model.

- K Nearest Neighbour
  K Nearest Neighbour or KNN is an algorithm that uses the principle of clustering. It is a supervised learning algorithm that assumes that similar things are near to each other. The distance between two points of data can be stated as the slope between that two points. When a new data point is added to the model, it calculates the distance between the points and the clusters. It classifies based on the distance parameter. It is a simple non linear algorithm.

- Deep Learning
  Deep learning algorithms use Neural Networks to learn about data and classify them. A simple neural network has a connection with each weight that transforms it and forms the input of neuron. This neuron has a bias term and an activation function. This project consists of 2 layers and uses the sigmoid activation function. This approach has never been used by any other research project. This is used to see if neural networks work better than supervised learning models.

3.5 Evaluation

After training the models using the training data created, next step is to test them using testing data. The classifiers predict the status of tweet if it is a rumour or not by using the rules learned during training. But it is important to understand how well was the classifier able to correctly classify data. The metrics below give the values in percentages to show how many tweets was the model able to classify correctly.

	Prediction		
Actual		Rumour	Non-Rumour
	Rumour	True Positive(TP)	False Negative(FN)
	Non-Rumour	False Positive(FP)	True Negative(TN)

- Accuracy
  This metric is the most commonly used metric used for classification projects. It can be defined as the ratio of predictions that were correct to the total number of outcomes. The formula can stated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision
  Using only accuracy as a metric for evaluate the performance can be misleading sometimes. Hence, other metrics are also used, one of them is Precision. It can be stated as the ratio of correctly predicted outcomes to the total predicted outcomes detected by the classifier. The formula can be stated as follows:

$$Precision = \frac{TP}{TP + FP}$$

- Recall
  Another metric is Recall which is also known as the sensitivity of predictions. It can be stated as the ratio of accurately predicted positive outcomes to all the outcomes detected by the classifier. The formula be stated as follows:

$$Recall = \frac{TP}{TP + FN}$$

- F1 score
  F1 score is the weighted mean of Recall and Precision. This score takes both false positives and false negatives into consideration. This score is useful when dealing with an uneven class. The formula be stated as follows:

$$F1 = 2X\frac{RecallXPrecision}{Recall + Precision}$$

4 Implementation

This section discusses each step and decision taken right from selection of data to training and testing the models. The implementation phase starts from selecting the right type of data. The next step is to extract data and pre-processing the data to convert it into time series data that can be used to train the classifiers. Last step involves evaluating results to see which classifiers worked the best.

4.1 Design Specification

This section describes the architecture used for implementation of this project. It is important to understand the flow of project before implementing as it helps in understanding the type of data required for classifiers. Architecture for this research project has been shown in diagram 4, each element has been explained in section 3. The next sections will discuss the implementation of this architecture in detail.

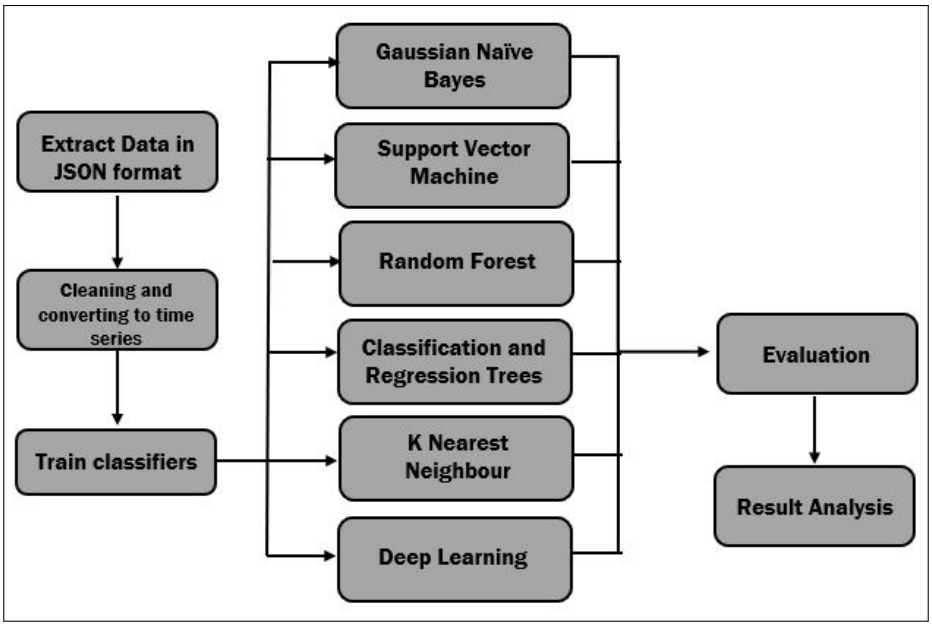

Figure 4: Architecture for rumour detection

4.2 Data Analysis

The data used for this project is called the PHEME dataset and is available online by PHEME dataset (2016). The description of dataset is given in section 3.1. The following table 1 shows the distribution of rumours and non-rumours in the original dataset.

Event	Rumours	Non-Rumours	Total Count
Charlie Hebdo	458 (22.0%)	1,621 (78.0%)	2,079
Ebola	14 (100%)	0 (0%)	14
Ferguson	284 (24.8%)	859 (75.2%)	1,143
Germanwings Crash	238 (50.7%)	231 (49.3%)	469
Gurlitt	61 (44.2%)	77 (55.8%)	138
Ottawa Shooting	470 (52.8%)	420 (47.2%)	890
Prince Toronto	229 (98.2%)	4 (0.2%)	233
Putin Missing	126 (52.9%)	112 (47.0%)	238
Sydney Siege	522 (42.8%)	699 (57.2%)	1,221
Total	**2,402 (37.4%)**	**4,023 (62.6%)**	**6,425**

Table 1: PHEME Dataset

As it can be seen from the table, the events Ebola and Prince Toronto have a very imbalanced distribution of samples. This is the reason that events are not considered for this project. Using these events may cause unreliable results as they may influence the models and give a wrong classification result. To avoid this from happening the data used for this project does not include the highlighted events in table 1. The data of events used for this project is given below in table 2.

Event	Rumours	Non-Rumours	Total Count
Charlie Hebdo	458 (22.0%)	1,621 (78.0%)	2,079
Ferguson	284 (24.8%)	859 (75.2%)	1,143
Germanwings Crash	238 (50.7%)	231 (49.3%)	469
Gurlitt	61 (44.2%)	77 (55.8%)	138
Ottawa Shooting	470 (52.8%)	420 (47.2%)	890
Putin Missing	126 (52.9%)	112 (47.0%)	238
Sydney Siege	522 (42.8%)	699 (57.2%)	1,221
Total	**2,159 (34.9%)**	**4,019 (65.1%)**	**6,178**

Table 2: Data used for project

The data used is visualized in figure 2. As seen from the table, the events Charlie Hebdo, Ferguson, Gurlitt and Sydney Siege have more non-rumours than rumours while remaining events have more rumours than non-rumours. Also, events Gurlitt and Putin Missing have less number of tweets in total compared to other events. This variation in data will help identify appropriate classifiers that work with each type of data. Overall there are 4,019 non-rumours and 2,159 rumours in the whole dataset of 6,178 tweets for 7 events. The next section will discuss further processing of data.

4.3 Data Preparation

The project is implemented using RStudio. Various libraries are used for the implementation and pre-processing of this research project. The data has a collection of tweets which

is organized in the following format. Each event consists of two folders, namely rumours and non-rumours. Each of these folders have multiple folders. Each folder belongs to one twitter conversation. One folder of twitter conversation consists of two folders, named source tweet and reactions. The source tweet folder has one JSON file which contains the source tweet and its details. The reaction folder has many JSON files with reactions for the source tweet mentioned in the last point. To be able to use this data it was important prepare it in a format appropriate for classifiers. This section will describe the pre-processing of data. The following are the steps:

1. Directory address of source tweets were stored in a list.
2. The following steps were iterated throughout the length of list (n) created in Step 1 :
   - i A dataframe was created for storing the timestamp of the n^{th} source tweet.
   - ii A new dataframe was created to store timestamps of all reactions for the n^{th} source tweet. There were some tweets with no reactions, these tweets were not considered.
   - iii Both dataframes are combined by keeping the source tweet as the first dataframe.
   - iv To extract just the time from the whole timestamp, each timestamp is separated to finest granularity.
   - v Date and time columns are created by combining appropriate columns created in last step. Datatype of time column is changes to time format for further calculations.
3. A new list which do not have any null values is created.
4. To calculate the time interval between the source and reaction tweets, a for loop is used. This loop calculates the difference between the first and i^{th} reaction row, while recursively saving the result in a dataframe.
5. All dataframes are stored in a list after converting them to time datatype.
6. The whole list is combined as a single dataframe.
7. Finally a new column is added names status which contains 0 for non-rumours and 1 for rumours.

The whole process mentioned above is done for both rumours and non-rumours. Reactions which were posted within an hour of source tweet are considered for this research project. The final dataframe contains data for both rumours and non-rumours. A table is created which has the frequency of reaction tweet posted for each unit of time. This is then converted to seconds by subtracting the time with 00:00:00 and keeping the unit as seconds. This is then stored as a CSV file for using it in machine learning models.

4.4 Implementation

After pre-processing the data, the final CSV for each event consist the number of rows mentioned in table 3. Each CSV had the unit of time in seconds, frequency of rumour or non-rumour and status which stated if the frequency is for rumour or non-rumour.

Event	Count
Charlie Hebdo	6,806
Ferguson	5,978
Germanwings Shooting	3,184
Gurlitt	62
Ottawa Shooting	4,840
Putin Missing	638
Sydney Siege	6,222
Total	**27,730**

Table 3: Data Count

Event	Training	Testing
Charlie Hebdo	20,924	6,806
Ferguson	21,752	5,978
Germanwings Shooting	24,546	3,184
Gurlitt	27,668	62
Ottawa Shooting	22,890	4,840
Putin Missing	27,092	638
Sydney Siege	21,508	6,222

Table 4: Training and Testing data

A total of 7 datasets were created from raw data. This data was used to create a 7-fold cross validation technique. This means that for each case one event is used as testing data while others were used for training. This method helps creating real-time scenarios by detecting rumours in data which is completely unknown to the classifier. This will also help in understanding the way model predicts rumours based on the various propagation patterns. Table 4 shows the number of samples used for testing and training.

In all 6 models are implemented to detect rumours. The selection of each classifier is given in section 3.4. Each classifier is implemented using RStudio. Most of the models belong to the caret package. To improve performance of SVM model by Madhav Kotteti et al. (2019), SVM model is implemented first with the same event dtaa. The author has implemented this model using the rbf kernel which gave very poor performance. However, with the use of radial kernel the model performed much better. Evaluation of this model is mentioned in section 5.1.

The next model used is Gaussian Naive Bayes. For this model x variable is created which contains the independent variables of training set in the form of a matrix. The y variable contains independent variables of testing set. This model belongs to a package named Rfast. Variables passed include x variable created and the dependent variable. The classifier learns from this data and uses it to predict the testing data. SVM is again implemented, but with the inclusions of new datasets too. Radial kernel is used for the same. Random Forest, KNN classifiers belongs to a package named caret. While implementing KNN model, parameters were pre-processed using center and scale to normalize them. Further, tuneLength was set to 7 which means the value of k is 7. All these models are trained and tested.

The last model is implemented using Keras. This deep learning model x and y variables for both training and testing data. The x variable consists of all variable except the dependent variables. The variables are scaled too. The y variable consists of the dependent variable after converting it in a categorical type. A sequential model is first initialised with dense layers using the famous 'relu' activation fuction. Dropout layers are also added to avoid overfitting the model. The last layer uses 'sigmoid' activation function. Next, model is compiled using adam optimizer and binary crossentropy as loss, this is because this is a binary classification model. The model is then fit using epochs and batch size. The model predicts the testing data after evaluating it. Epochs, batch size, units passed to each layer are tuned to get the best results. The next section will evaluated this models based on various metrics.

5 Evaluation

5.1 Improved Support Vector Machine

Event	Accuracy	Precision	Recall	F1 Score
Charlie Hebdo	0.78	0.87	0.66	0.75

Table 5: Improved SVM Results

In a previous research by Madhav Kotteti et al. (2019), the results achieved for SVM classifier were very poor. In order to improve that result, this classifier is built using a different kernel. The author acquired 0 precision, recall and F1 score using 'rbf' kernel. Using 'radial' kernel gave results mentioned in table 5 which is much better than the results achieved by Madhav Kotteti et al. (2019).

5.2 Evaluating Classifiers

The propagation patterns of rumours and non-rumours are mentioned in figure 2 which show that the events Charlie Hebdo, Sydney Siege and Ferguson have very distinct patterns while Germanwings Crash and Ottawa Shooting have almost non distinguishable patterns. Patterns for Gurlitt and Putin Missing are confusing as they had very few number of samples. This section will discuss how each classifier works for each of these cases.

5.2.1 Experiment 1: Support Vector Machine

Event	Accuracy	Precision	Recall	F1 Score
Charlie Hebdo	0.78	0.86	0.67	0.75
Ferguson	0.66	0.63	0.79	0.70
Germanwings Shooting	0.49	0.49	0.85	0.62
Gurlitt	0.45	0.47	0.9	0.62
Ottawa Shooting	0.46	0.47	0.61	0.53
Putin Missing	0.48	0.49	0.90	0.63
Sydney Siege	0.64	0.64	0.62	0.63

Table 6: Support Vector Machine (SVM) Results

SVM works on the principle of support vectors and hyperplanes. It is one of the most popular algorithms used for classifying data. Table 6 displays the results of this classifier. As it can be seen, SVM works best with data which has distinct patterns. The metrics for events Charlie Hebdo, Ferguson and Sydney Siege are much better than other events. Charlie Hebdo has the most distinct pattern, hence has the best metric. It has a precision of 86% and recall of 67% with 75% F1 score and 78% accuracy. Less samples and indistinct patterns resulted in very poor results.

5.2.2 Experiment 2: Random Forest

Performance of Random Forest classifier is given in table 7. The event that were classified more accurately by this classifier were Charlie Hebdo, Ferguson and Sydney Siege with a precision of above 55%. Even though Gurlitt and Putin Missing had less samples, random

Event	Accuracy	Precision	Recall	F1 Score
Charlie Hebdo	0.65	0.67	0.60	0.64
Ferguson	0.58	0.57	0.65	0.61
Germanwings Shooting	0.50	0.50	0.70	0.58
Gurlitt	0.52	0.51	0.65	0.57
Ottawa Shooting	0.48	0.48	0.62	0.54
Putin Missing	0.47	0.48	0.66	0.56
Sydney Siege	0.57	0.57	0.59	0.58

Table 7: Random Forest Results

forest performed quite good for that data as well. Overall, Random Forest performed very good for events which had distinct propagation patterns than the ones which had less samples or indistinct patterns. Overall, it can be said that random forest classifier did not perform very good and could not classify rumours and non-rumours effectively within an hour on the event.

5.2.3 Experiment 3: Gaussian Naive Bayes

Event	Accuracy	Precision	Recall	F1 Score
Charlie Hebdo	0.62	0.57	0.94	0.71
Ferguson	0.60	0.55	0.94	0.70
Germanwings Shooting	0.50	0.50	1.00	0.67
Gurlitt	0.50	0.50	1.00	0.67
Ottawa Shooting	0.48	0.49	0.92	0.64
Putin Missing	0.50	0.50	1.00	0.67
Sydney Siege	0.54	0.52	0.91	0.66

Table 8: Gaussian Naive Bayes Results

Gaussian Naive Bayes is said to perform good with small data samples (Madhav Kotteti et al.; 2019). However, it gave an accuracy of 50% and recall of 100% for events like Gurlitt and Putin Missing which can be seen in table 8. This is because the classifier considered all data samples as the positive outcome, which was Rumours in this case. Hence, it classified 50% of the samples correctly. The same phenomenon also happened with Germanwings Crash event. This classifier could however classify events more accurately for all other propagation patterns. The model performed best in terms with a recall, it is above 90% for all events. Ottawa Shooting had poorest performance among all other events, which maybe due pattern which are almost similar for both rumours and non-rumours.

5.2.4 Experiment 4: Classification and Regression Trees

CART is a new name for decision tree classifier. Decision Trees are created by splitting data into groups based on various features. While CART is just like random forest, the results are quite different. It can be seen that CART performs better than random forest in most cases. This algorithm was successfully able to classify rumours and non-rumours with a precision of 78% for event of Charlie Hebdo which can be seen in table 9. However, it performed poorly for other events. The classifier was not able to detect rumours and

Event	Accuracy	Precision	Recall	F1 Score
Charlie Hebdo	0.78	0.78	0.77	0.77
Ferguson	0.64	0.60	0.86	0.71
Germanwings Shooting	0.50	0.50	0.96	0.66
Gurlitt	0.50	0.50	1.00	0.67
Ottawa Shooting	0.46	0.48	0.76	0.58
Putin Missing	0.50	0.50	0.97	0.66
Sydney Siege	0.62	0.65	0.53	0.59

Table 9: Classification and Regression Trees (CART) Results

non-rumours for events with very less samples like Putin Missing and Gurlitt. The highest accuracy of 78% was obtained for Charlie Hebdo with Precision, Recall and F1 score as 78%, 77% and 77% respectively.

5.2.5 Experiment 5: K Nearest Means

Event	Accuracy	Precision	Recall	F1 Score
Charlie Hebdo	0.70	0.75	0.61	0.67
Ferguson	0.62	0.61	0.68	0.64
Germanwings Shooting	0.49	0.50	0.81	0.61
Gurlitt	0.52	0.51	0.71	0.59
Ottawa Shooting	0.46	0.47	0.63	0.54
Putin Missing	0.48	0.49	0.76	0.60
Sydney Siege	0.61	0.60	0.61	0.61

Table 10: K Nearest Neighbours (KNN) Results

KNN works on the principle of clustering. The parameters were tuned for this algorithm. The value of k was set to 7 before implementing this model. The model was the most successful for Charlie Hebdo event with accuracy of 70% and worked the poorest for Ottawa Shooting with accuracy of just 46% which can be seen in table 10. Similar result can be seen for Germanwings Shooting and Putin Missing who also have indistinct patterns like Ottawa Shooting.

5.2.6 Experiment 6: Deep Learning

Event	Accuracy	Precision	Recall	F1 Score
Charlie Hebdo	0.71	0.78	0.59	0.67
Ferguson	0.52	0.54	0.19	0.28
Germanwings Shooting	0.46	0.46	0.41	0.43
Gurlitt	0.19	0.19	0.19	0.19
Ottawa Shooting	0.46	0.47	0.61	0.53
Putin Missing	0.41	0.40	0.38	0.39
Sydney Siege	0.62	0.59	0.74	0.66

Table 11: Deep Learning Results

Deep Learning uses neurons to learn about the samples provided and keeps learning from mistakes. The metrics of this model are showed in table 11. Epochs and batch size have to be tuned to get good results for all events. The model worked best for Charlie Hebdo with an accuracy of 71% while it gave poor accuracy of 19% for Gurlitt. Deep learning model was implemented in an attempt to see if it learns from the different propagation patterns and gives a better result than the other models implemented. But, as it can be seen, it has a pretty average result overall.

6 Discussion

This section discuss the performances of different models according to different various events. This will give an insight of which model performs best and worst according to the propagation patterns of events. In the event of Charlie Hebdo, SVM and CART gave

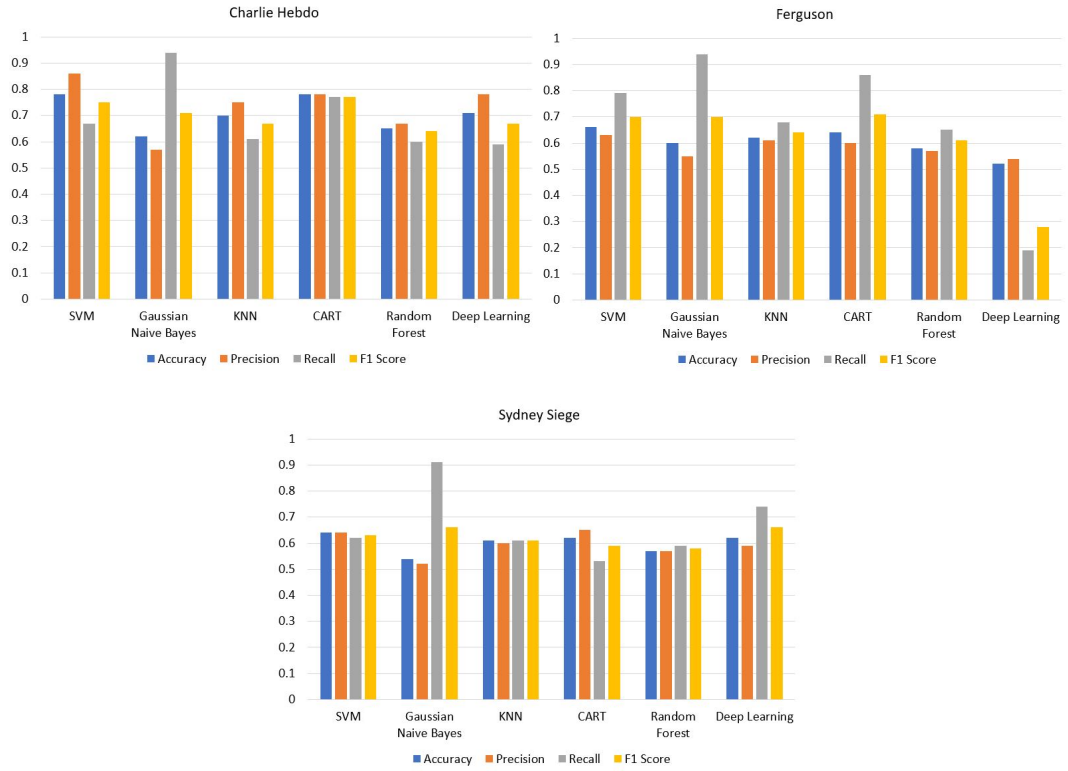

Figure 5: Events with distinct propagation patterns

the highest accuracy of 78% which can be seen in diagram 5. The highest recall rate was 94% with Gaussian Naive Bayes and precision and F1 Score was 77% by CART. Overall CART performed the best for this data. The lowest accuracy achieved was 62% with Gaussian Naive Bayes. A similar propagation pattern was observed for the event named Ferguson. For this event, the highest accuracy achieved was 66% by SVM. SVM also had the highest precision of 63%, but, the highest recall rate of 94% was achieved with Gaussian Naive Bayes. Sydney Siege had somewhat similar pattern and also had the highest accuracy of 64% by SVM model. The event had the highest precision score of 65% with CART model and highest recall of 91% by Gaussian Naive Bayes. Overall, it can be said that Gaussian Naive Bayes was able to detect maximum rumours correctly while SVM had the maximum accuracy.

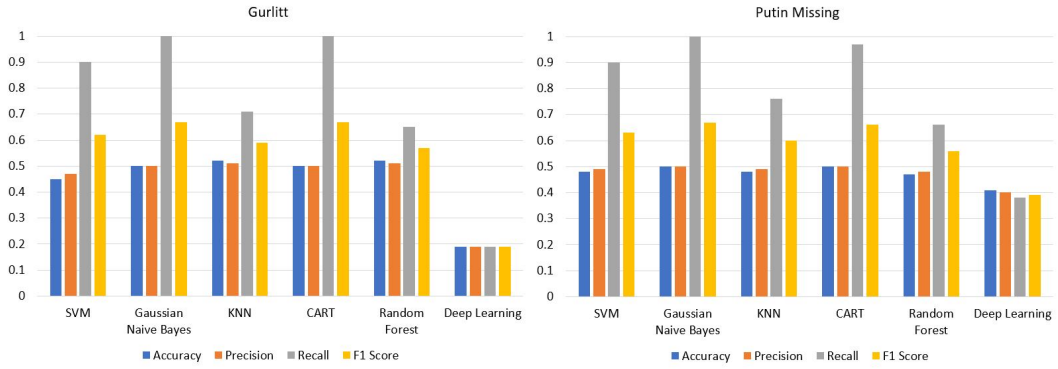

Figure 6: Events with less samples

Gurlitt had less number of samples which resulted in a very indistinct propagation pattern. Out of all the models implemented in this project, KNN and Random Forest performed the best with an accuracy of 52% as shown in diagram 6. KNN and Random Forest also had the maximum precision of 51% but CART and Gaussian Naive Bayes had the maximum Recall rate. This was because Gaussian Naive Bayes classified 50 percent of data as positive class which is rumour and the other 50 percent as negative class which is non-rumour. This resulted in a high recall rate. The event Putin Missing also had few samples but they were more than Gurlitt. The highest accuracy of 50% was achieved by Gaussian Naive Bayes and CART. Both these models also had a high precision score of 50% but the recall of Gaussian Naive Bayes was higher than CART. Overall, it can be said that CART and Gaussian Naive Bayes are good at detecting rumours when the sample size is small.

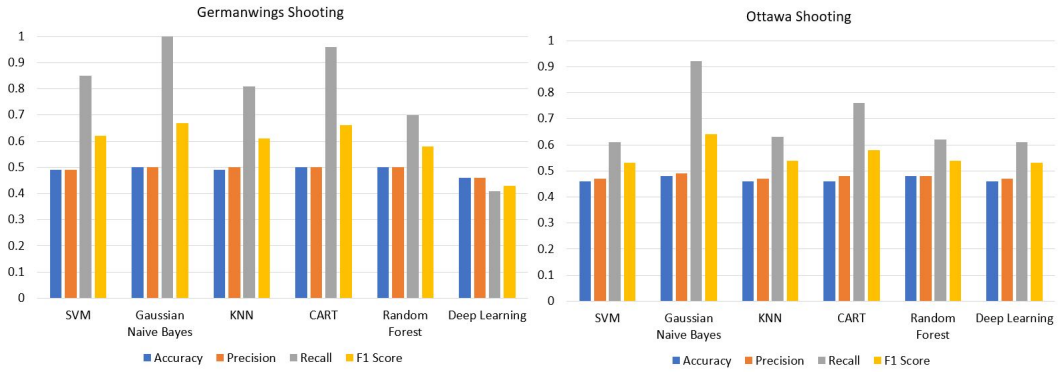

Figure 7: Events with indistinct propagation patterns

Indistinct patterns were a real challenge for the model as there was a very small difference in the propagation patterns of rumours and non-rumours. One such event was Germanwings Crash for which CART and Random Forest performed the best with an accuracy of 50% which can be seen in diagram 7. While all these also had a precision score of 50% along with KNN and Gaussian Naive Bayes, the recall rate of Gaussian Naive Bayes was the highest with 100%. Another such event was Ottawa Shooting in which Gaussian Naive Bayes and Random Forest performed the best among others with an accuracy of 48%. The highest precision was given by Gaussian Naive Bayes and Random Forest model which was 49%. Although, Gaussian Naive Bayes had the highest Recall rate of 92%. Overall, it can be stated that Gaussian Naive Bayes performed better than

other in this cases.

Although deep learning model was used to see if it performs better than the other models, it failed to do so. There are some instances where it performed almost equal to the model with highest results, but most of the times it was not that good. This may be due to the insufficient data used for training the model. Further, maximum computation time was required by SVM followed by Random Forest. SVM took time but gave better results in most cases. Overall, Gaussian Naive Bayes could predict the highest number of rumours out of all the predicted values.

7 Conclusion and Future Work

Rumours are a huge problem and social media websites like twitter are helping them rise. It is important to detect rumours to reduce chaos created by hoax tweets. This research project is implemented with the aim of detecting rumours in the first hour it was posted. The timestamps of tweets and their reactions are considered. There are 7 events which have different propagation patterns. Most of the models were better are detecting rumours for events who had distinct propagation patterns. Some models like CART and Gaussian Naive Bayes were also able to identify rumours for events with indistinct propagation patterns. But, all classifiers performed poorly for events with less number of patterns.

The models may perform better in future by training them with data that has more events with less number of samples. This will train the models to understand the propagation patterns for events with less samples along with patterns for many samples. To further improve the performance of models, they can be tuned even further and unsupervised classifiers like LSTM can be implemented. This research project was implemented with the aim of understanding the model performances for different type of propagation pattern of rumours and non-rumours.

GitHub

The following is the GitHub link of the project repository:
`https://github.com/Nikita-Parab/Master_Thesis`

Acknowledgement

I would like to thank my supervisor, Dr. Muhammad Iqbal, for guiding me throughout the project. He has helped me by answering my queries and guiding me whenever I was stuck. I would like to thank my family who have been always very supportive and understanding. Last but not the least, my friends who have always been of help at any hour of the day and also encouraged me throughout the module.

References

Akhtar, M. S., Ekbal, A., Narayan, S. and Singh, V. (2018). No, that never happened!! investigating rumors on twitter, *IEEE Intelligent Systems* **33**(5): 8–15.

Basak, R., Sural, S., Ganguly, N. and Ghosh, S. K. (2019). Online public shaming on twitter: Detection, analysis, and mitigation, *IEEE Transactions on Computational Social Systems* **6**(2): 208–220.

Chen, W., Zhang, Y., Yeo, C. K., Lau, C. T. and Lee, B. S. (2018). Unsupervised rumor detection based on users behaviors using neural networks, *Pattern Recognition Letters* **105**: 226 – 233. Machine Learning and Applications in Artificial Intelligence.

Hashimoto, T., Kuboyama, T. and Shirota, Y. (2011). Rumor analysis framework in social media, *TENCON 2011 - 2011 IEEE Region 10 Conference*, pp. 133–137.

Hassan, N. Y., Gomaa, W. H., Khoriba, G. A. and Haggag, M. H. (2018). Supervised learning approach for twitter credibility detection, *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, pp. 196–201.

Kwon, S., Cha, M. and Jung, K. (2017). Rumor detection over varying time windows., *PLoS ONE* **12**(1): 1 – 19.

Ma, J., Gao, W., Wei, Z., Lu, Y. and Wong, K.-F. (2015). Detect rumors using time series of social context information on microblogging websites, *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, ACM, New York, NY, USA, pp. 1751–1754.
URL: *http://doi.acm.org/10.1145/2806416.2806607*

Madhav Kotteti, C. M., Dong, X. and Qian, L. (2019). Multiple time-series data analysis for rumor detection on social media, pp. 4413–4419.

Mahmoodabad, S. D., Farzi, S. and Bakhtiarvand, D. B. (2018). Persian rumor detection on twitter, *2018 9th International Symposium on Telecommunications (IST)*, pp. 597–602.

Ounsrimuang, P. and Nootyaskool, S. (2019). Classifying vehicle traffic messages from twitter to organize traffic services, *2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA)*, pp. 705–708.

PHEME dataset (2016). PHEME rumour scheme dataset: journalism use case.
URL: *https://figshare.com/articles/PHEME_rumour_scheme_dataset_journalism_use_case/2068650*

Poddar, L., Hsu, W., Lee, M. L. and Subramaniyam, S. (2018). Predicting stances in twitter conversations for detecting veracity of rumors: A neural approach, *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 65–72.

Sicilia, R., Giudice, S. L., Pei, Y., Pechenizkiy, M. and Soda, P. (2018). Twitter rumour detection in the health domain, *Expert Systems with Applications* **110**: 33 – 40.

SuyalatuDong, Feng-HuaFan and Yong-ChangHuang (2018). Studies on the population dynamics of a rumor-spreading model in online social networks, *Physica A: Statistical Mechanics and its Applications* **492**: 10 – 20.

Tai, Y., He, H., Zhang, W. and Jia, Y. (2018). Automatic generation of review content in specific domain of social network based on rnn, *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 601–608.

Wang, S., Moise, I., Helbing, D. and Terano, T. (2017). Early signals of trending rumor event in streaming social media, **2**: 654–659.

Xu, N., Chen, G. and Mao, W. (2018). Early signals of trending rumor event in streaming social media, pp. 1–7.

Yavary, A. and Sajedi, H. (2018). Rumor detection on twitter using extracted patterns from conversational tree, *2018 4th International Conference on Web Research (ICWR)*, pp. 78–85.

Zubiaga, A., Liakata, M., Procter, R., Wong Sak Hoi, G. and Tolmie, P. (2016). Analysing how people orient to and spread rumours in social media by looking at conversational threads, *PLOS ONE* **11**(3): 1–29.
URL: *https://doi.org/10.1371/journal.pone.0150989*