

A Project Report on

# Movie Success Prediction

Submitted in partial fulfillment of the requirements for the award  
of the degree of

**Bachelor of Engineering**

in

**Computer Engineering**

by

**Akash Gada(15202021)**  
**Sayali Patil(15102016)**  
**Sumit Tawde(15202012)**  
**Manali Kajari(14102025)**  
**Dewanshi Mishra(15102014)**

Under the Guidance of

**Prof. Sachin B. Takmare**



**Department of Computer Engineering**

A.P. Shah Institute of Technology  
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615  
UNIVERSITY OF MUMBAI

**Academic Year 2018-2019**

## Approval Sheet

This Project Report entitled “*Movie Success Rate Prediction*” Submitted by “*Akash Gada*”(15202021), “*Sayali Patil*”(15102016), “*Sumit Tawde*”(15202012), “*Manali Kajari*”(14102025), “*Dewanshi Mishra*”(15102014) is approved for the partial fulfillment of the requirement for the award of the degree of *Bachelor of Engineering* in *Computer Engineering* from *University of Mumbai*.

Prof. Sachin Takmare  
Guide

Prof. Sachin Malave  
Head Department of Computer Engineering

Place: A.P. Shah Institute of Technology, Thane

Date:

## CERTIFICATE

This is to certify that the project entitled “*Movie Success Rate Prediction*” submitted by “*Akash Gada*” (15202021), “*Sayali Patil*” (15102016), “*Sumit Tawde*” (15202012), “*Manali Kajari*” (14102025), “*Dewanshi Mishra*” (15102014) for the partial fulfillment of the requirement for award of a degree *Bachelor of Engineering* in *Computer Engineering*, to the University of Mumbai, is a bonafide work carried out during academic year 2018-2019.

Prof. Sachin Takmare  
Guide

Prof. Sachin Malave  
Head of Department of Computer Engineering

Dr. Uttam D.Kolekar  
Principal

External Examiner(s)

1.

2.

Place: A.P. Shah Institute of Technology, Thane

Date:

## Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

---

(Signature)

---

(Akash Gada and 15202021)  
(Sayali Patil and 15102016)  
(Sumit Tawde and 15202012)  
(Manali Kajari and 14102025)  
(Dewanshi Mishra and 15102014)

Date:

## **Abstract**

With the advent of Big Data and analytics, the world has changed in ways previously unimaginable. In a rapidly growing and thriving industry such as the motion picture industry, data analytics has opened a number of important new avenues that can be used to analyze past data, make creative marketing decisions and accurately predict the fortunes of impending movie releases. The timing of the movie release is critical to the success of a movie. To facilitate the release date selection, studios decide and pre-announce the targeted release dates on a weekly basis long before the actual release of their forthcoming movies. Their choices of release dates and then the subsequent changes are strategic in nature taking into consideration various factors. Social media analytics can be used to predict the optimal release date for a movie. Using data collected from social media channels, we can gauge expectations of the target audience and the buzz towards the movie. This project implements the elk stack which basically imports data from various sources like twitter, google trends, etc. This data is then visualized through various charts and graphs and data clouds. These visualizations are interactive with the user and filters can be applied spontaneously. This makes the analysis easier to comprehend and the movie's success can be predicted on a time and location basis. This also provides an analyzed data to provide a promotional strategy for a particular movie by providing data as to who is the most talked about from the movie and the location wise data of the songs from the movie.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	How it works? . . . . .	3
1.2	Objective . . . . .	4
1.3	Elasticsearch . . . . .	4
1.4	Logstash . . . . .	5
1.5	Kibana . . . . .	7
<b>2</b>	<b>Literature Review</b>	<b>10</b>
<b>3</b>	<b>Usecase and Class diagram</b>	<b>14</b>
3.1	Usecase . . . . .	14
3.2	Class . . . . .	15
<b>4</b>	<b>Implementation</b>	<b>16</b>
4.1	Installation . . . . .	17
4.1.1	Installing Elasticsearch . . . . .	17
4.1.2	Installing Kibana . . . . .	18
4.1.3	Installing Logstash . . . . .	20
<b>5</b>	<b>Result</b>	<b>21</b>
5.1	Static Data . . . . .	21
5.2	Dynamic Data . . . . .	25
5.3	Final Output . . . . .	28
<b>6</b>	<b>Conclusions and Future Scope</b>	<b>29</b>
6.1	Conclusion . . . . .	29
6.2	Future scope . . . . .	30
	<b>Bibliography</b>	<b>31</b>

# List of Figures

1.1	Working Overview . . . . .	3
4.1	Checks java version. . . . .	16
4.2	Checks elasticsearch service status. . . . .	17
4.3	Changing configuration file. . . . .	17
4.4	Checking elasticsearch using curl. . . . .	18
4.5	Changes in conf file. . . . .	18
4.6	Checks the status of kibana. . . . .	19
4.7	Starting the kibana. . . . .	19
4.8	Changes in conf file. . . . .	20

# List of Abbreviations

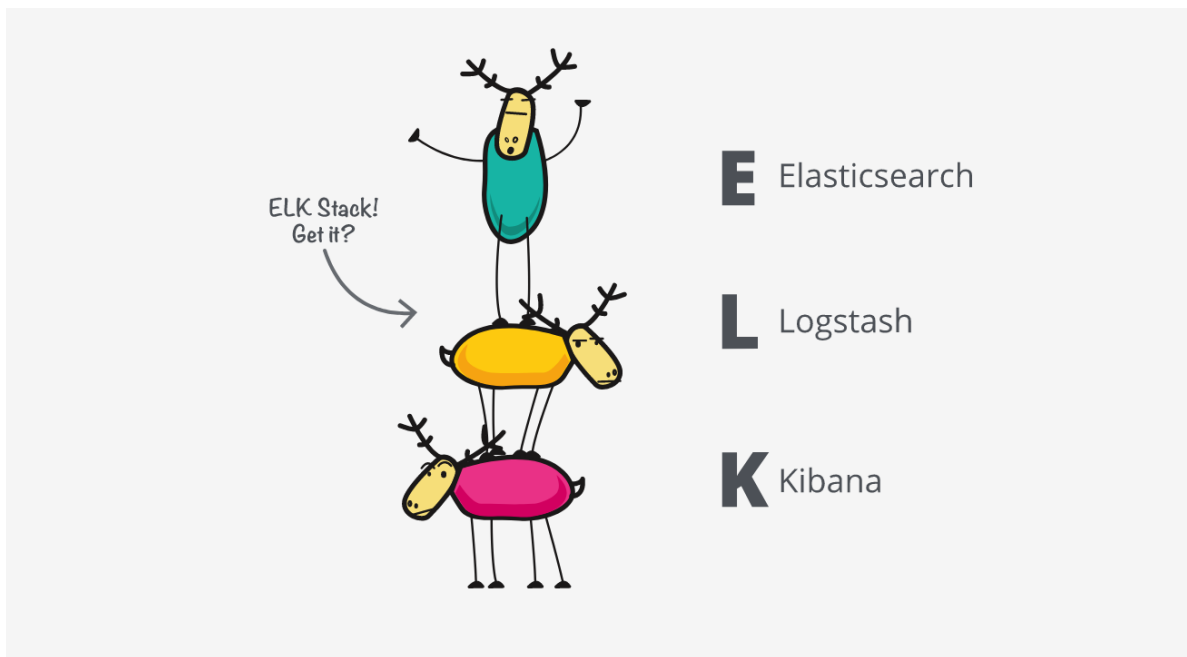
ELK:	Elastic Kibana Logstash
SRA:	Step-wise Regression Analysis
MAPE:	Mean Absolute Percentage Error
SOM:	Self-Organizing Map
CSV:	Comma-Separated Values
SVM:	Support Vector Machine
URL:	Uniform Resource Locator
HTTP:	Hyper Text Transfer Protocol
RFA:	Random Forest Algorithm
SQL:	Structured Query Language



# Chapter 1

## Introduction

The elastic stack framework is computationally effective and can be connected by numerous degrees of opportunity. Intermediary analyzer demonstrates diverse methods for appearing of an intermediary logs , it will likewise indicate distinctive moves made by client Principle work of intermediary analyzer is to screen and dissect each log for making report in comprehensible organization on account of it,It will accommodating for human to make further move



## 1.1 How it works?

Front end: kibana works as a front end tool, which also shows result in visualized formats of data which analyze with the help of logstash and elasticsearch

Elasticsearch provides a REST API over multiple indexes that can be searched and queried. Indexes are automatically created when you post a JSON document to an index scheme. The index scheme is composed of 3 parts:

- Index name
- Index type
- Document ID



Figure 1.1: Working Overview

This is an end-to-end stack that handles everything from data aggregation to data visualization. On a recent project, I needed a database with a schema-less data model for aggregated queries and fast searching. I filtered my options to two choices Elasticsearch and Solr (both based on Apache Lucene). I decided to go with Elasticsearch because of the full stack and AWS support.

## 1.2 Objective

- To establish a friendly interaction with human
- Easy to analyze big data with low efforts
- High performance other than another competitors

## 1.3 Elasticsearch

Elasticsearch is the living heart of what is the present the most well-known log investigation stage the ELK Stack (Elasticsearch, Logstash, and Kibana). The pretended by Elasticsearch is central to the point that it has turned out to be synonymous with the name of the stack itself. Utilized basically for inquiry and log examination, Elasticsearch is today a standout amongst the most famous database frameworks accessible today.

At first discharged in 2010, Elasticsearch is a cutting edge look and investigation motor which depends on Apache Lucene. Totally open source and worked with Java, Elasticsearch is ordered as a NoSQL database. Elasticsearch stores information in an unstructured manner, and as of not long ago you couldn't question the information utilizing SQL. Notwithstanding, the new Elasticsearch SQL undertaking will permit utilizing SQL proclamations to communicate with the information. More on that in future posts.

With regards to information examination, Elasticsearch is utilized together with different segments in the ELK Stack, Logstash and Kibana, and assumes the job of information ordering and capacity.

### Basic Elasticsearch Concepts

Elasticsearch is a component rich and complex structure. Specifying and drilling down into all of its stray pieces is unfathomable. Regardless, there are some central thoughts and terms that any Elasticsearch customers should learn and get settled with. Coming up next are the five "must-know" thoughts in any case.

- Documents

Archives are JSON objects that are put away inside an Elasticsearch file and are viewed as the base unit of capacity. In the realm of social databases, archives can be contrasted with a column in a table.

For instance, we should accept that you are running an online business application. You could have one archive for each item or one record for each request. There is no restriction to what number of reports you can store in a specific record

- Types

Elasticsearch types are utilized inside records to subdivide comparative kinds of information wherein each sort speaks to an extraordinary class of archives. Types comprise of a name and a mapping (see underneath) and are utilized by including the type field.

- Mapping

Like an outline in the realm of social databases, mapping characterizes the diverse kinds that dwell inside a record. It characterizes the fields for reports of a particular kind the information type, (for example, string and whole number) and how the fields ought to be filed and put away in Elasticsearch.

- Index

Elasticsearch Indices are legitimate parcels of archives and can be contrasted with a database in the realm of social databases. Proceeding with our internet business application precedent, you could have one list containing the majority of the information identified with the items and another with the majority of the information identified with the clients.

You can have the same number of records characterized in Elasticsearch as you need however this can influence execution. These, thus, will hold archives that are special to each list.

- Shards

Record measure is a typical reason for Elasticsearch crashes. Since there is no restriction to what number of reports you can store on each record, a list may take up a measure of plate space that surpasses the points of confinement of the facilitating server. When a list approaches this farthest point, ordering will start to come up short. One approach to counter this issue is to part up records on a level plane into pieces called shards. This enables you to disperse tasks crosswise over shards and hubs to improve execution.

## 1.4 Logstash

In the ELK Stack (Elasticsearch, Logstash and Kibana), the urgent errand of parsing information is given to the "L" in the stack Logstash.

Logstash began as an open source instrument created to deal with the gushing of a lot of log information from different sources. In the wake of being consolidated into the ELK Stack, it formed into the stack's workhorse, accountable for likewise preparing the log messages, improving them and rubbing them and afterward dispatching them to a characterized goal for capacity (reserving).

Because of a substantial biological system of modules, Logstash can be utilized to gather, enhance and change a wide cluster of various information types. There are more than 200

distinctive modules for Logstash, with a huge network utilizing its extensible highlights. It has not generally been smooth cruising for Logstash. Because of some inborn execution issues and configuration imperfections, Logstash has gotten a not too bad measure of protests from clients throughout the years. Side undertakings were created to lighten a portion of these issues (for example Logger, Logstash-Forwarder, Beats), and elective log aggregators started contending with Logstash.

However in spite of these blemishes, Logstash still remains a vital part of the stack. Enormous advances have been made to attempt and reduce these torments by acquainting Beats and enhancements with Logstash itself, eventually make logging with ELK substantially more solid than what it used to be.

## **Logstash Configuration**

Occasions totaled and prepared by Logstash experience three phases: accumulation, handling, and dispatching. Which information is gathered, how it is prepared and where it is sent to, is characterized in a Logstash arrangement record that characterizes the pipeline. Every one of these stages is characterized in the Logstash setup document with what are called modules "Info" modules for the information gathering stage, "Channel" modules for the handling stage, and "Yield" modules for the dispatching stage. Both the info and yield modules support codecs that enable you to encode or decipher your information (for example json, multiline, plain).

### **Input plugins**

One of the things that makes Logstash so powerful is its ability to aggregate logs and events from various sources. Using more than 50 input plugins for different platforms, databases and applications, Logstash can be defined to collect and process data from these sources and send them to other systems for storage and analysis. The most common inputs used are: file, beats, syslog, http, tcp, udp, stdin, but you can ingest data from plenty of other sources.

### **Filter plugins**

Logstash underpins various incredibly ground-breaking channel modules that empower you to advance, control, and procedure logs. It's the intensity of these channels that makes Logstash an adaptable and significant apparatus for parsing log information. Channels can be joined with restrictive explanations to play out an activity if a particular basis is met.

### **Output plugins**

Likewise with the sources of info, Logstash bolsters various yield modules that empower you to push your information to different areas, administrations, and innovations. You can store occasions utilizing yields, for example, File, CSV, and S3, convert them into messages with RabbitMQ and SQS, or send them to different administrations like HipChat, Pager-Duty, or IRC. The quantity of blends of sources of info and yields in Logstash makes it an extremely adaptable occasion transformer.

Logstash occasions can emerge out of various sources, so it's critical to check whether an

occasion ought to be handled by a specific yield. On the off chance that you don't characterize a yield, Logstash will consequently make a stdout yield. An occasion can go through various yield modules.

## Logstash Codecs

Codecs can be utilized in the two data sources and yields. Info codecs give an advantageous method to translate your information before it enters the information. Yield codecs give a helpful method to encode your information before it leaves the yield. Some common codecs:

- The default "plain" codec is for plain content with no delimitation between occasions
- The "json" codec is for encoding JSON occasions in sources of info and interpreting json messages in yields note that it will return to plain content if the got payloads are not in a substantial JSON position
- The "json-lines" codec permits you either to get and encode json occasions delimited by 'n' or to interpret JSON messages delimited by 'n' in yields
- The "rubydebug," which is valuable in investigating, enables you to yield Logstash occasions as information Ruby items

## 1.5 Kibana

Totally open source, Kibana is a program based UI that can be utilized to seek, investigate and picture the information put away in Elasticsearch lists (Kibana can't be utilized related to different databases). Kibana is particularly eminent and mainstream because of its rich graphical and perception capacities that enable clients to investigate substantial volumes of information.

Kibana can be introduced on Linux, Windows and Mac utilizing .zip or tar.gz, vaults or on Docker. Kibana keeps running on node.js, and the establishment bundles come worked in with the required parallels. Peruse progressively about setting up Kibana in our Kibana instructional exercise.

### Kibana searches

Looking Elasticsearch for explicit log message or strings inside these messages is the bread and butter of Kibana. In Kibana's inquiry bar, you can enter Lucene question sentence structure or ventures dependent on Elasticsearch Query DSL. Once entered, the principle show territory will channel the information showed in like manner, appearing backward sequential request.

Kibana questioning is a craftsmanship unto itself, and there are different strategies for performing seeks on your information. Here are the most well-known inquiry types:

- Free content ventures utilized for rapidly scanning for a particular string.
- Field-level inquiries utilized for hunting down a string inside a particular field.
- Logical articulations used to join seeks into a coherent proclamation.
- Proximity seeks utilized for looking terms inside a particular character vicinity.

### **Kibana visualizations**

As referenced above, Kibana is eminent for perception abilities. Utilizing a wide range of outlines and diagrams, you can cut up your information any way you need. You will find that you can do practically anything you desire with your information.

Making representations, be that as it may, is presently constantly direct and can require some serious energy. Key to making this procedure effortless is knowing your information. The more you are familiar with the distinctive alcoves and crevices in your information, the simpler it is.

Kibana visualization are based over Elasticsearch inquiries. Utilizing Elasticsearch conglomerations (for example entirety, normal, min, macintosh, and so forth.), you can perform different handling activities to influence your perceptions to portray drifts in the information.

### **Visualization types**

Visualizations in Kibana are sorted into five unique kinds of perceptions:

- Basic Charts (Area, Heat Map, Horizontal Bar, Line, Pie, Vertical bar)
- Data (Date Table, Gauge, Goal, Metric)
- Maps (Coordinate Map, Region Map)
- Time arrangement (Timelion, Visual Builder)
- Other (Controls, Markdown, Tag Cloud)

## **Kibana dashboards**

When you have an accumulation of perceptions prepared, you can include them all into one complete representation called a dashboard. Dashboards enable you to screen a framework or condition from a high vantage point for simpler occasion relationship and pattern investigation.

Dashboards are exceptionally powerful they can be altered, shared, played around with, opened in various showcase modes, and that's only the tip of the iceberg. Tapping on one field in a particular representation inside a dashboard, channels the whole dashboard appropriately (you will see a channel included at the highest point of the page).

## **Kibana Elasticsearch index**

The hunts, representations, and dashboards spared in Kibana are called objects. These articles are put away in a committed Elasticsearch record (.kibana) for investigating, sharing, rehashed use and reinforcement.

The list is made when Kibana begins. You can change its name in the Kibana setup document. The list contains the accompanying records, each containing their own arrangement of fields:

- Saved record designs
- Saved looks
- Saved representations



# Chapter 2

## Literature Review

### **Abstract:**

In real world prediction models and mechanisms can be used to predict the success of a movie. The proposed work aims to develop a system based upon data mining techniques that may help in predicting the success of a movie in advance thereby reducing certain level of uncertainty. An attempt is made to predict the past as well as the future of movie for the purpose of business certainty or simply a theoretical condition in which decision making [the success of the movie] is without risk, because the decision maker [movie makers and stake holders] has all the information about the exact outcome of the decision, before he or she makes the decision [release of the movie]. With over two million spectators a day and films exported to over 100 countries, the impact of Bollywood film industry is formidable. We gather a series of interesting facts and relationships using a variety of data mining techniques.

In particular, we concentrate on attributes relevant to the success prediction of movies, such as whether any particular actors or actresses are likely to help a movie to succeed. Given the low success rate of movies, models and mechanisms can be used to predict the success of a movie. It will help the business significantly. Various stakeholders such as actors, producers, directors etc. can use these predictions to make more informed decisions. They can make the decision before the movie release. Historical data of each component such as actor, actress, and director, composer that influences the success or failure of a movie is given due to its weightage.. This proposed work aims to develop a model based upon the data mining techniques that may help in predicting the success of a movie in advance thereby reducing certain level of uncertainty.

### **Related Works:**

With over two million Audience a day and films exported to over 100 countries, the impact of Bollywood film industry is formidable. From the first Indian film Raja Harishchandra by Dhundhiraj Govind (Dadasaheb) Phalke in 1913 to 1981, India produced over 15000 feature films[1]. Since then it has produced, at least another 15000 at a rate of more than 1000 films a year (1091 in 2006, 1146 in 2007 and 1325 in 2008) in 26 languages. Literature survey has revealed only two studies which have attempted to predict the success of movies. While one study uses Bayesian belief network to predict the success, the other one uses neural network for the same. Lee and Change in their study using. Bayesian Belief Network for predicting box office performance concluded that Bayesian Belief Networks were better in predicting the success as compared to neural networks [2]. Machine learning has also been used for predicting movie success by using algorithms like RF and SVM . Although the use

of RF and SVM within the movie domain seems to be fairly limited, the two algorithms have been applied and evaluated in many applications for the purpose of regression as well as classification. Within recent study Verikas et al. (2011) have surveyed a number of large as well as small scale comparisons on data mining and machine learning, all of which include the RF algorithm, specifically issuing its prediction performance in comparison to other algorithms as well as the use of the variable importance estimates available from RF. Among the previous applications and algorithm comparisons included by Verikas et al. (2011) are several large-scale studies such as Meyer et al. (2003) and Statnikov et al. (2008), evaluating RF and SVM among other algorithms over a number of 33 and 22 datasets respectively. Another is implementing neural networks which have been extensively used in forecasting and prediction studies, and so it has been also employed for predicting the success and failure of the movies also. However, Hadavandi et al have used integration of genetic fuzzy systems and artificial neural networks for stock price forecasting.

Stock market prediction is regarded as a challenging task in financial time-series forecasting. The central idea to successful stock market prediction is achieving best results using minimum required input data and the least complex stock market model. They used stepwise regression analysis (SRA) to determine factors which have most influence on stock prices. In the next stage divided raw data into  $k$  clusters by means of self-organizing map (SOM) neural networks. Finally, all clusters were fed into independent GFS models with the ability of rule base extraction and data base tuning. They evaluated capability of the proposed approach by applying it on stock price data gathered from IT and Airlines sectors, and compare the outcomes with previous stock price forecasting methods using mean absolute percentage error (MAPE).

Results show that the proposed approach outperforms all previous methods, so it can be considered as a suitable tool for stock price forecasting problems. An organization has to make the right decisions in time depending on demand information to enhance the commercial competitive advantage in a constantly fluctuating business environment. Therefore, estimating the demand quantity for the next period most likely appears to be crucial. Efendigil et al have suggested a decision support system for demand forecasting with artificial neural network and Neuro fuzzy logic. Neural networks have been used in predicting the traffic flow. Chan et al have deployed neural networks based upon exponential smoothing method for predicting the traffic flow.

Reuter and Muller have developed artificial neural network for forecasting of fuzzy time series. Forecasting box office revenue of a movie before its theatrical release is a difficult and challenging problem. In a study conducted by Zhang et al, [8] a multi-layer BP neural network (MLBP) with multi-input and multi output was employed to build the prediction model. All the movies were divided into six categories ranged from blob to bomb according to their box office incomes, and the purpose is to predict a film into the right class.

## **Disadvantages of previous systems:**

Indian Hindi Cinema industry popularly known as Bollywood has reached staggering proportions in terms of volume of business (184.3 billion), manpower employment (over 6 million workers), movies produced (more than 100 in a year) and its reach (exported to more than 100 countries worldwide). With so much at stake and highly uncertain nature of returns, it is of commercial interest to develop a model which can predict success of a movie. This however, is not an easy work, since movies have been described as experience goods with very less shelf life; it is difficult to forecast demand for a movie. There are number of parameters that may influence success of a movie like time of its release, marketing gimmicks, lead actor, lead actress, director, producer, writer, music director being some of the factors.

## **Abstract for system based on EKL:**

Elastic Search is a way to organize data and make it effortlessly accessible. In particular Elastic Search-a distributed full-text search engine-explicitly addresses issues of scalability, big data search and performance that relational databases were simply never designed to support. Elastic Search is a Server Based search developed in Java that is published as open source under the terms of the Apache Lucene. Elastic Search includes all advances in speed, security, scalability and hardware efficiency. It takes data and optimizes according to the language-based searches and stores it in a sophisticated format. We plan to use Elastic Search's analytics tool to help improve the queried data. And also pre-process the query to make it faster before sending it to the server so as to reduce the time latency while being used over slower data connections. Whether it is searching a database of trade products by description or finding similar text in a body of crawled web pages, Elastic search is imaginary excellent.

## **Introduction:**

A developer who uses relational databases, needed to search tables that had millions of records, resulting in exceedingly complex database views/stored-procedures and adding full text search on relational database fields. It made the database double the size and the speed was not optimum either. Relational databases are simply not built for such operations. With Elastic search we can achieve the speed we would like, as it leases us index millions of documents. Elastic search can perform queries across all those millions of documents and return accurate results in fraction of a second. Elastic search is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.

Elastic search is developed in Java and is released as open source under the terms of the Apache Lucene. Elastic search is the most popular enterprise search engine followed by Apache Solr, also based on Lucene. Shay Banon formed the antecedent to Elastic search, named Compass, in 2004. While thinking about the third version of Compass he realized that it would be necessary to rewrite big parts of Compass to "create a scalable search solution. So he created "a solution built from the ground up to be distributed" and used a common interface, JSON over HTTP, suitable for programming languages other than Java as well. Shay Banon released the first version of Elastic search in February 2010. Elastic search BV was founded in 2012 to provide commercial services and products around Elastic

search and related software.

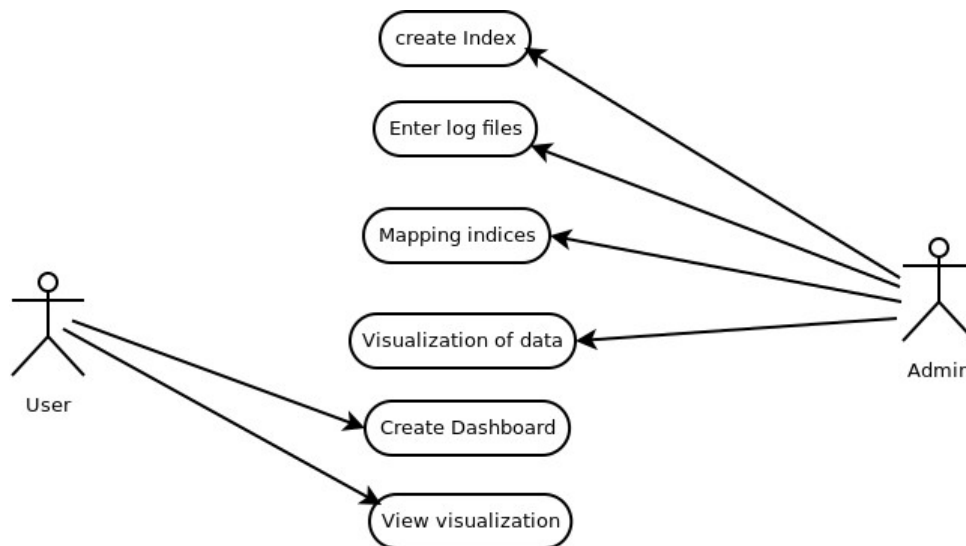
**Advantages:**

The approach to viewer rating prediction is very straightforward and can be easily applied by anyone. The movie prediction requires more experience, because of its dependency on the user to be able to interpret the different visualizations of the data in the right way and infer the right conclusions from it. Visual Analytics will become more and more integrated in our lives for the great possibilities it enables. The human mind is very intelligent for extracting information from visualizations and thus can analyze data quicker and in a more complex way. This system fetches data from different sites like twitter, google trends, etc, various visualisations can thus be created and all these visualisations can then be put up on one, single interactive dashboard so that all the data can be visualised in one place.

# Chapter 3

## Usecase and Class diagram

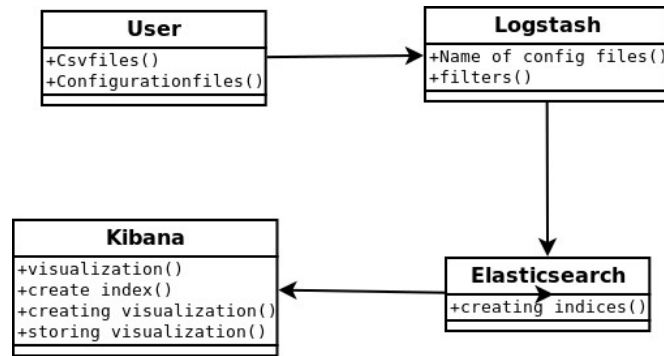
### 3.1 Usecase



- User/Actor: give its Information in the logs form
- Area log: where different types of logs are store
- Logstash:Logstash collects logs and It transforms the data and sends to the Elasticsearch database.
- Apply Filter:These activity done with logstash in which logs are filter by applying various filters and these filters apply with help of configuration file which provide by admin.
- kibana: Kibana is an opensource visualization tool which provides a beautiful web interface to visualize the Elasticsearch data.it fetch data from elasticsearch and elasticsearch get data from logstash.
- Visualization: Apply various visualization so that output will give user in various visualize format which help big data analyses in less period of time.

- Admin: In these Admin work for creating config file which use in applying filters so that it process logs which will enter in kibana for visualization.

## 3.2 Class



# Chapter 4

## Implementation

The ELK Stack can be installed using a variety of methods and on a wide array of different operating systems and environments. ELK can be installed locally, on the cloud, using Docker and configuration management systems like Ansible, Puppet, and Chef. The stack can be installed using a tarball or .zip packages or from repositories.

Many of the installation steps are similar from environment to environment and since we cannot cover all the different scenarios, we will provide an example for installing all the components of the stack Elasticsearch, Logstash, Kibana, and Beats on Linux.

Following are steps to install ELK (Elasticsearch, Kibana , Logstash)  
You want to see your JDK version, Bit version (32 and 64 bit), storage of RAM, ROM  
we will install Elastic + Logstash + Kibana on ubuntu system 14.04 LTS spec minimum requirement to running ELK my recommendation is :

- Ram: 4GB
- CPU :2core
- Disk 40GB

1. Confirm JDK 8 + version available on the machine.

Use command : `java -version`

```
sumit@sumit-HP-Notebook:~$ java -version
openjdk version "1.8.0_191"
OpenJDK Runtime Environment (build 1.8.0_191-8u191-b12-0ubuntu0.16.04.1-b12)
OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode)
```

Figure 4.1: Checks java version.

## 4.1 Installation

### 4.1.1 Installing Elasticsearch

**Step 1 :** Use following command to installing elasticsearch. command :  
`dpkg -i elasticsearch-7.0.0.deb`

**Step 2:** Check elasticsearch service status using command

```
root@sumit-HP-Notebook:/home/sumit# service elasticsearch status
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; disabled; vend
   Active: active (running) since Sun 2019-04-21 20:25:40 IST; 2 days ago
     Docs: http://www.elastic.co
   Main PID: 2884 (java)
    Tasks: 89 (limit: 512)
   Memory: 474.1M
      CPU: 5min 18.927s
   CGroup: /system.slice/elasticsearch.service
           └─2884 /usr/share/elasticsearch/jdk/bin/java -Xms1g -Xmx1g -XX:+UseCo
             2969 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_

Apr 21 20:25:40 sumit-HP-Notebook systemd[1]: Started Elasticsearch.
Apr 21 20:25:43 sumit-HP-Notebook elasticsearch[2884]: OpenJDK 64-Bit Server VM
Apr 21 20:26:24 sumit-HP-Notebook systemd[1]: Started Elasticsearch.
```

Figure 4.2: Checks elasticsearch service status.

**Step 3:** Change the configuration file open file using command:  
`gedit /etc/elasticsearch/elasticsearch.yml` and Uncomment the `network.host` and `http.port`

```
# ----- Network -----
#
# Set the bind address to a specific IP (IPv4 or IPv6):
#
network.host: localhost
#
# Set a custom port for HTTP:
#
http.port: 9200
#
# For more information, consult the network module documentation.
..
```

Figure 4.3: Changing configuration file.

**Step 4:** Save and close the file after this changes and Start the elasticsearch service using command:

```
service elasticsearch start
service elasticsearch status
```



**Step 5 :** Check elasticsearch running using curl  
curl http://localhost:9200

```
root@sumit-HP-Notebook:/home/sumit# curl http://localhost:9200
{
  "name" : "sumit-HP-Notebook",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "S87HZEX1Qg0Ghy2enBiKhW",
  "version" : {
    "number" : "7.0.0",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "b7e28a7",
    "build_date" : "2019-04-05T22:55:32.697037Z",
    "build_snapshot" : false,
    "lucene_version" : "8.0.0",
    "minimum_wire_compatibility_version" : "6.7.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Figure 4.4: Checking elasticsearch using curl.

## 4.1.2 Installing Kibana

**Step 1 :** Use following command to installing kibana. `dpkg -i kibana-7.0.0-amd64.deb`

**Step 2.** Configure Kibana: open configuration file using gedit `/etc/kibana/kibana.yml` and Make required changes

**Changes:**

1. `server.port:` 5601
2. `server.host:` "localhost"
3. `elasticsearch.url:` "http://localhost:9200"

```
# Kibana is served by a back end server. This setting specifies the port to use.
server.port: 5601

# Specifies the address to which the Kibana server will bind. IP addresses and host names are both
# valid values.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: "localhost"

# The URLs of the Elasticsearch instances to use for all your queries.
elasticsearch.hosts: ["http://localhost:9200"]
```

Figure 4.5: Changes in conf file.

**Step 3 :** After this changes save and close the file. Start the kibana service using command and check the status: `service kibana start` `service kibana status`

```
root@sumit-HP-Notebook:/home/sumit# service kibana status
● kibana.service - Kibana
   Loaded: loaded (/etc/systemd/system/kibana.service; disabled; vendor preset:
   Active: active (running) since Sun 2019-04-21 20:25:59 IST; 2 days ago
     Main PID: 3012 (node)
       Tasks: 11 (limit: 512)
      Memory: 152.7M
         CPU: 2min 56.189s
        CGroup: /system.slice/kibana.service
               └─3012 /usr/share/kibana/bin/./node/bin/node --no-warnings --max-htt
```

Figure 4.6: Checks the status of kibana.

**Step 4 :** Once kibana is running wait for few seconds and then open url : `http://localhost:5601` in browser.

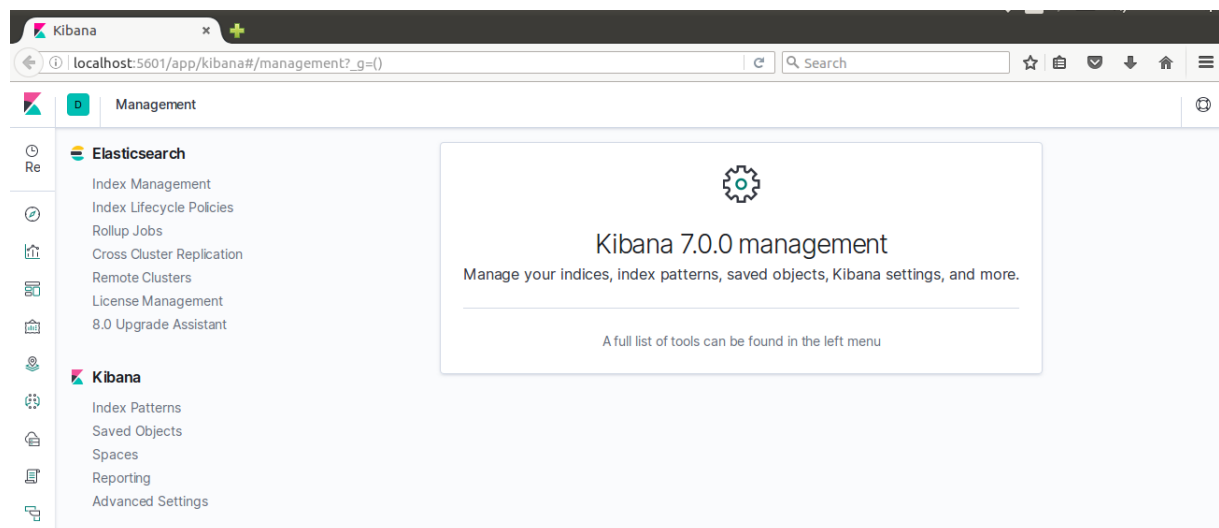


Figure 4.7: Starting the kibana.

### 4.1.3 Installing Logstash

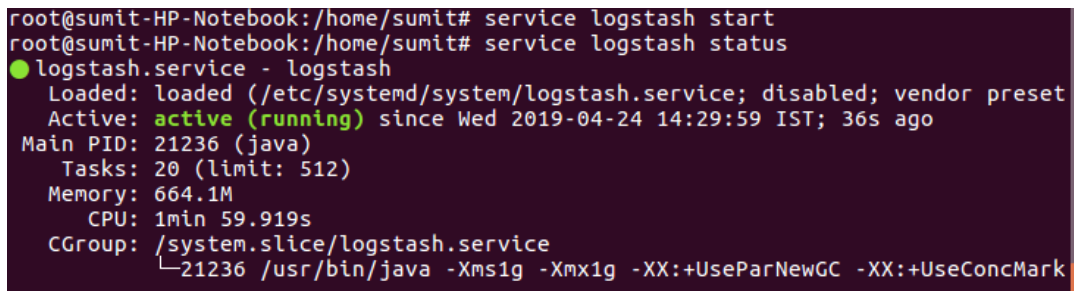
**Step 1 :** Use following command to installing logstash.

command : `dpkg -i logstash-6.3.0.deb`

**Step 2 :** Start logstash service using command:

command : `service logstash start`

**Step 3 :** Check service running : `service logstash status`

A terminal window screenshot showing the command 'service logstash status' and its output. The output indicates that the logstash.service is loaded and active (running). It provides details such as the main PID (21236), tasks (20), memory usage (664.1M), CPU usage (1min 59.919s), and the CGroup path. The command line for the service is also displayed: '21236 /usr/bin/java -Xms1g -Xmx1g -XX:+UseParNewGC -XX:+UseConcMark'.

```
root@sumit-HP-Notebook:/home/sumit# service logstash start
root@sumit-HP-Notebook:/home/sumit# service logstash status
● logstash.service - logstash
   Loaded: loaded (/etc/systemd/system/logstash.service; disabled; vendor preset
   Active: active (running) since Wed 2019-04-24 14:29:59 IST; 36s ago
     Main PID: 21236 (java)
        Tasks: 20 (limit: 512)
      Memory: 664.1M
         CPU: 1min 59.919s
       CGroup: /system.slice/logstash.service
              └─21236 /usr/bin/java -Xms1g -Xmx1g -XX:+UseParNewGC -XX:+UseConcMark
```

Figure 4.8: Changes in conf file.

# Chapter 5

## Result

### 5.1 Static Data

#### Results for Static Data:

The data has the following structure. The headings for columns and the actual columns always stay in this position:

```
Region,total_buzz,first_class,ghar_more_perdesiya,Tile_track
Andaman and Nicobar Islands,100,100,100,67
Maharashtra,55,82,80,64
Delhi,51,73,78,59
Gujarat,51,64,67,54
Chhattisgarh,51,63,60,54
Goa,46,62,58,53
Daman and Diu,44,60,55,46
Uttarakhand,43,59,52,45
Jharkhand,41,48,51,38
Arunachal Pradesh,41,42,50,36
```

See in which location your term was most popular during the specified time frame. Values are calculated on a scale from 0 to 100, where 100 is the location with the most popularity as a fraction of total searches in that location, a value of 50 indicates a location which is half as popular. A value of 0 indicates a location where there was not enough data for this term.

Elasticsearch was originally created to store time-based data, thus it helps if there is a timestamp or date in your CSV. We usually say that Elasticsearch was made to store and search log files and you can think of log files as just files with data and a timestamp associated with the data. This will allow us to make visualizations of different aspects of the data over time. This is what our config file looks like:

## Movie.conf

```
1 input
2 {
3   file
4   {
5     path => "/home/sumit/Desktop/dataset/movie.csv" start_position => "beginning" sincecb_path => "/dev/null"
6   }
7 }
8 filter
9 { csv {
10   separator => ","
11   columns => ['Region', 'total_buzz', 'first_class', 'ghar_more_perdesiya', 'Tile_track']
12 } } output { elasticsearch{
13   hosts => "http://localhost:9200" index => "movie_buzz12"
14 } stdout{} }
15
```

## Sincedb

The first part is the part in the file input section that refers to "sincedb". Logstash has an interesting component or feature called sincedb. Logstash keeps track of where it was last reading a file before it crashed or stopped. This is useful for reading files with Logstash when realizing you need to change something in your configuration file.

Sincedb is a file that is created by Logstash by default, although you can configure Logstash to create it at the location of your choice. Sincedb follows a specific format. To make use of sincedb, you need to tell it where to write itself to. This comes as part as the file input plugin.

You can tell it to write to a location by specifying a value here:

```
1 {
2   path => "/home/sumit/Desktop/dataset/movie.csv"
3   start_position => "beginning"
4   sincecb_path => "/dev/null"
5 }
6
```

We need to be running Elasticsearch, Kibana and Logstash. Now run configuration file on terminal.

Output after running .conf file.

```
[INFO ] 2019-04-25 15:09:11.677 [Api Webserver] agent - Successfully started Logstash API endpoint {:port=>9601}
{
  "@timestamp" => 2019-04-25T09:39:11.580Z,
  "message" => "Maharashtra,55,82,80,64",
  "@version" => "1",
  "Tile_track" => "64",
  "first_class" => "82",
  "total_buzz" => "55",
  "host" => "sumit-HP-Notebook",
  "path" => "/home/sumit/Desktop/dataset/movie.csv",
  "Region" => "Maharashtra",
  "ghar_more_perdesiya" => "80"
}
{
  "@timestamp" => 2019-04-25T09:39:11.585Z,
  "message" => "Goa,46,62,58,53",
  "@version" => "1",
  "Tile_track" => "53",
  "first_class" => "62",
  "total_buzz" => "46",
  "host" => "sumit-HP-Notebook",
  "path" => "/home/sumit/Desktop/dataset/movie.csv",
  "Region" => "Goa",
  "ghar_more_perdesiya" => "58"
}
{
  "@timestamp" => 2019-04-25T09:39:11.588Z,
  "message" => "Arunachal Pradesh,41,42,50,36",
  "@version" => "1",
  "Tile_track" => "36",
  "first_class" => "42",
  "total_buzz" => "41",
  "host" => "sumit-HP-Notebook",
  "path" => "/home/sumit/Desktop/dataset/movie.csv",
  "Region" => "Arunachal Pradesh",
}
```

Now open up <http://localhost:5601/app/kibana> in your browser. Go to settings index - Add new.

Now choose your index name and click "Date", which was one of the fields of the CSV. This allows us to map the data based on the timestamps provided in the CSV file.

Now make sure your mapping survived after you indexing your data. Once you've configured the index pattern you should now see the mapping for the index. The fields that don't have a tick next to them in the analyzed column are the fields that you set as not analyzed in your mapping.

/ movie\_buzz12\*

Create index pat...

- ★ d3
- d2
- data1
- data1\*
- dynamic\*
- kibana\_sample\_data\_e...
- movie2
- movie21\*
- movie\_buzz\*
- movie\_buzz12\***
- movie\_intrest\*
- sagar

## movie\_buzz12\*

Time Filter field name: @timestamp

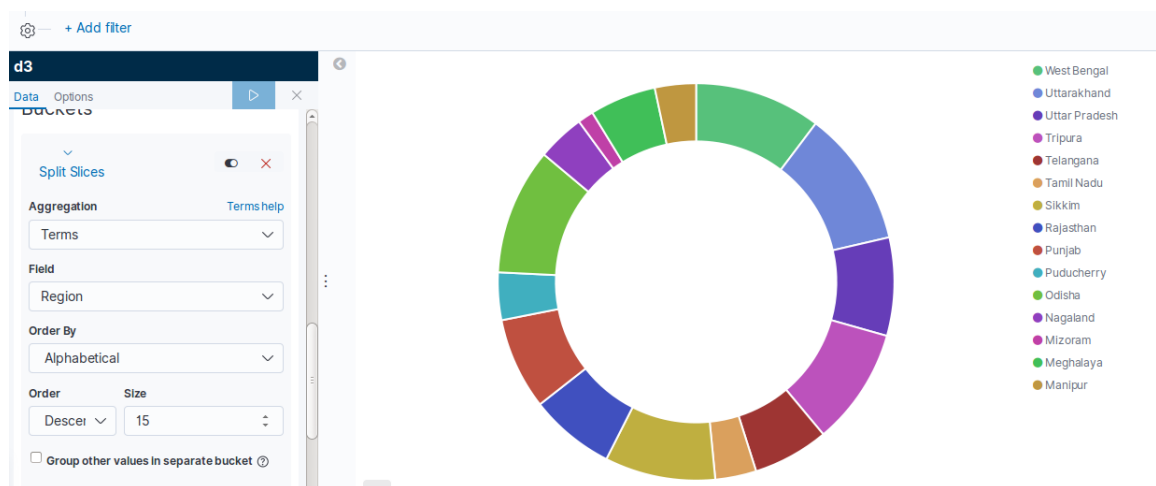
This page lists every field in the **movie\_buzz12\*** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch [Mapping API](#)

Fields (26)
Scripted fields (0)
Source filters (0)

Filter
All field types

Name	Type	Format	Searchable	Aggregat...	Excluded
@timestamp	date		•	•	
@version	string		•		
@version.keyword	string		•	•	
Region	string		•		
Region.keyword	string		•	•	
Tile_track	string		•		
Tile_track.keyword	string		•	•	

Now that we are satisfied with the mapping of the indexed we just loaded data into, we can create a graph from the data.



## 5.2 Dynamic Data

### Results for dynamic data

To establish a connection with Twitter and extract data, we will need Twitter API keys. To get your hands on these keys, you will first need to create a Twitter app.

Go to the Twitter apps page, and create a new app. You will need to enter a name, description, and website URL for the app. Dont worry about the particulars, your entries here will not affect how the data is shipped into Elasticsearch.

Once created, open the apps Keys and Access Tokens tab, and click the button at the bottom of the page to generate a new access token.

Lets create a configuration file called twitter.conf and copy the below text. Update the consumer-key, consumer-secret, oath-token and oath-token-secret values with the values from your Twitter Stream App created earlier.

#### Twitter.conf

```
input
{
  twitter
  {
    consumer_key => "98R4oqOrETDRwTXcTEJn1Gkz9"
    consumer_secret => "Cck9Th1Ro6v1c7CwqGnIwNzVuNu67Zh4HfXFDRun4x52we7Ue"
    oauth_token => "1101430533085528064-R5uGfo8TJPsxwnoZgvlTpMg45WGEIE"
    oauth_token_secret => "YP34CQ7ghBY1vyOP2507kHdUxvC0HyMxjw1HssoitJQ2x"
    keywords => ["kalank"]
  }
}
output
{
  elasticsearch
  {
    hosts => ["localhost:9200"]
    index => "dynamicdata"
  }
}
```

This configuration will receive all tweets tagged with kalank and store them to an index called dyanamicdata in elasticsearch. Our data is not yet properly setup to handle time-based events. We'll fix this later. Replace logstash-\* with twitter.

Kibana will now show you the index definition for the dynamic index. You can see all of the fields names, their data types and if the fields are analyzed and indexed. This provides a good high level overview of the data configuration in the index. You can also filter fields in the filter box.



You should see something similar to this:

## dynamic\*



Time Filter field name: @timestamp

This page lists every field in the **dynamic\*** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch [Mapping API](#)

Fields (31)

Scripted fields (0)

Source filters (0)

Filter

All field types

Name	Type	Format	Searchable	Aggregat...	Excluded
@timestamp	date				
@version	string				
@version.keyword	string				
_id	string				
_index	string				
_score	number				

## Discover Twitter Data

Click on the Discover link in the top navigation bar. This opens the Discover view which is a very helpful way to dive into your raw data. You should see something similar to this:

Discover

# In-reply-to

t message

retweeted

t source

t uris

t user

@timestamp

@version

t \_id

# \_score

? hashtags

? symbols

? user\_mentions

> Apr 21, 2019 @ 01:49:55.000

user\_mentions: { user: beenish\_mrs client: <a href="https://mobile.twitter.com" rel="nofollow">Twitter Web App</a> @timestamp: Apr 21, 2019 @ 01:49:55.000 hashtags: { "indices": [ 50, 64 ], "screen\_name": "VivianDsena01", "id": 292208028, "name": "Vivian Dsena" }, { "id\_str": "610642546", "indices": [ 71, 82 ], "screen\_name": "RubiDilaik", "id": 610642546, "name": "Rubina Dilaik" } user: HeenaB23 client: <a href="https://twitter.com/download/android" rel="nofollow">Twitter for Android</a> @timestamp: Apr 21, 2019 @ 01:50:10.000

user\_mentions: { "id\_str": "1004308485687791616", "indices": [ 3, 12 ], "screen\_name": "AvChahat", "id": 1004308485687791600, "name": "Chahat AV" }, { "id\_str": "292208028", "indices": [ 50, 64 ], "screen\_name": "VivianDsena01", "id": 292208028, "name": "Vivian Dsena" }, { "id\_str": "610642546", "indices": [ 71, 82 ], "screen\_name": "RubiDilaik", "id": 610642546, "name": "Rubina Dilaik" } user: HeenaB23 client: <a href="https://twitter.com/download/android" rel="nofollow">Twitter for Android</a> @timestamp: Apr 21, 2019 @ 01:50:10.000

> Apr 21, 2019 @ 01:50:28.000

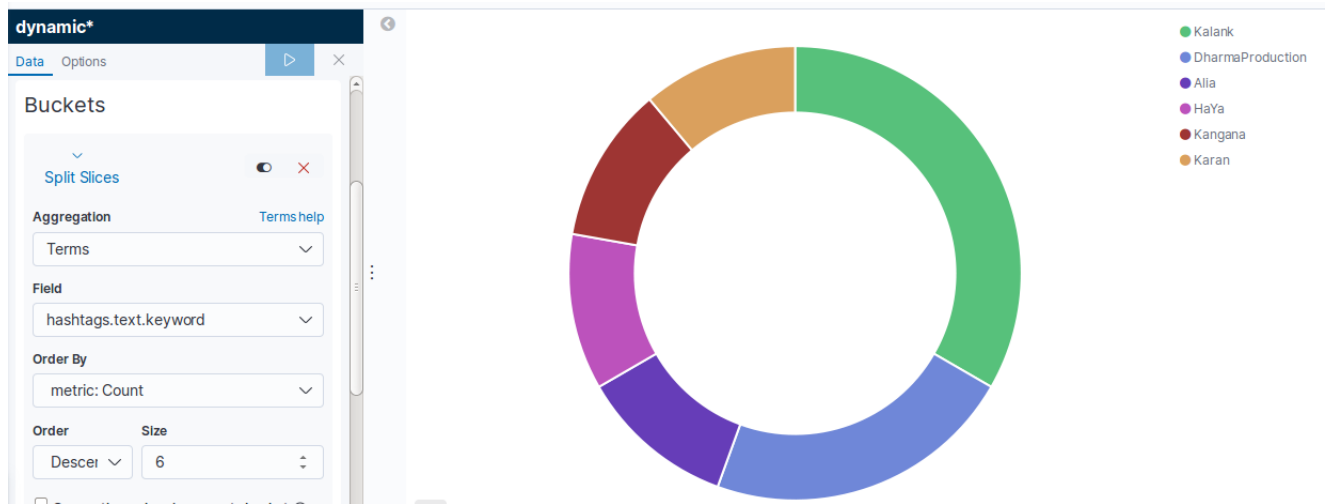
user\_mentions: { "id\_str": "1875769380", "indices": [ 3, 16 ], "screen\_name": "urvi\_singhh", "id": 1875769380, "name": "urvi" } user: Saurabhkh client: <a href="https://mobile.twitter.com" rel="nofollow">Twitter Web App</a> @timestamp: Apr 21, 2019 @ 01:50:28.000 hashtags: { "indices": [ 36, 43 ], "text": "Kalank" } source: http://twitter.com/Saurabhkh/status/1119697360668381184 @version: 1 message: RT @urvi\_singhh: Kalank Dobra #Kalank https://t.co/Cf1olw3lqS symbols: retweeted: false \_id: CkNoPG0BHPJ45Bw7XU-L \_ty

> Apr 21, 2019 @ 01:51:23.000

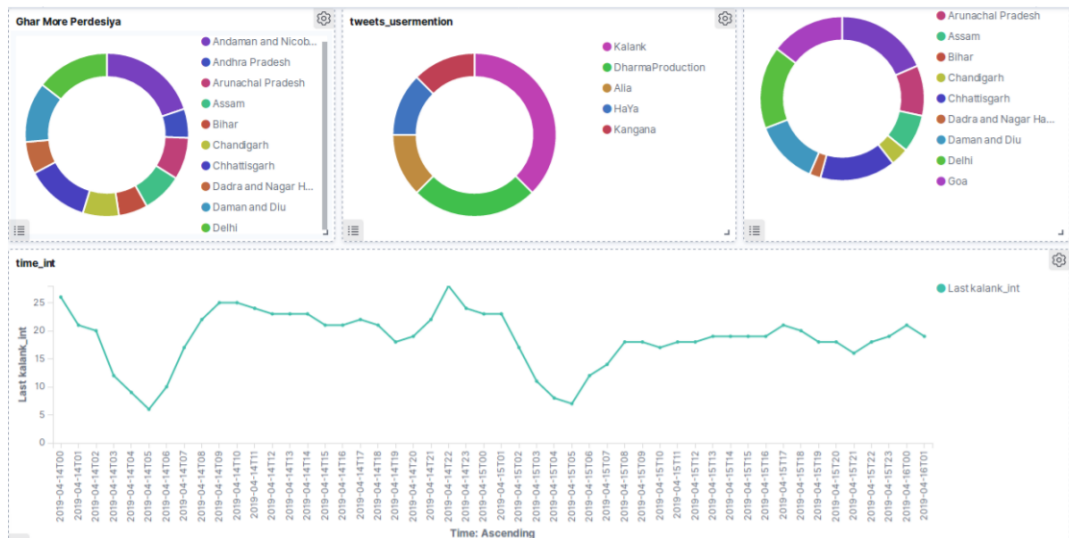
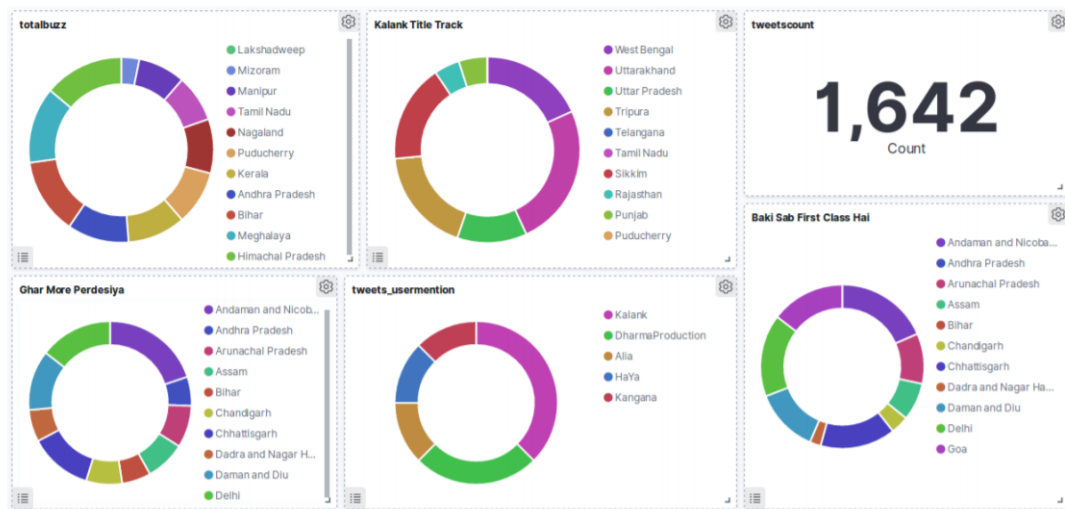
user\_mentions: { "id\_str": "770259414133551104", "indices": [ 3, 18 ], "screen\_name": "RahulVerma4860", "id": 770259414133551100, "name": "Rahul verma" }, { "id\_str": "101311381", "indices": [ 29, 36 ], "screen\_name": "iamsrk", "id": 101311381, "name": "Shah Rukh Khan" } user: SRKFANJUNAEID client: <a href="https://twitter.com/download/android" rel="nofollow">Twitter for Android</a> @timestamp: Apr 21, 2019 @ 01:51:23.000 hashtags: { "indices": [ 44, 61 ], "text": "DharmaProduction" } source: http://twitter.com/SRKFNJUNAEID/status

## Create Kibana Visualizations

Now that our index is properly configured, we can create some visualizations. Click on the Visualize link in the navigation bar. You should see something similar to this:



## 5.3 Final Output



# Chapter 6

## Conclusions and Future Scope

### 6.1 Conclusion

Elastic search can scale out to hundreds or even thousands of servers and handle megabytes of data. Elasticsearch is distributed by nature, and it is designed to hide the complexity that comes with being distributed. The distributed aspect of Elastic search is largely transparent. Elastic search is an open-source, broadly-distributable, readily-scalable, enterprise-grade search engine. Accessible through an extensive and elaborate API, Elastic search can power extremely fast searches that support data discovery applications.

Elastic Search uses standard RESTful APIs and JSON to build and maintain clients in many languages such as Java, Python, .NET, and Groovy. Conventional SQL database managements systems aren't really designed for full-text searches, and they certainly don't perform well against loosely structured raw data that resides outside the database. On the same hardware, queries that would take more than 10 seconds using SQL will return results in under 10 milliseconds in Elastic search.

Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. You use Kibana to search, view, and interact with data stored in Elasticsearch indices. You can easily perform advanced data analysis and visualize your data in a variety of charts, tables, and maps. Kibana makes it easy to understand large volumes of data. Its simple, browser-based interface enables you to quickly create and share dynamic dashboards that display changes to Elasticsearch queries in real time.

Logstash is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favorite stash. Data is often scattered or siloed across many systems in many formats. Logstash supports a variety of inputs that pull in events from a multitude of common sources, all at the same time. Easily ingest from your logs, metrics, web applications, data stores, and various AWS services, all in continuous, streaming fashion.

## 6.2 Future scope

Movie data analytics can be unbelievably powerful and can make educated guesses. Analytics gives businesses the quantitative data necessary to make better, more informed decisions and improve the services they provide to their audience. Like most industries today, Hollywood has access to more supportive data for decision-making than ever before. Ingesting and interpreting all of that information has until very recently, been an intimidating task. Thankfully, the right visualization of that data is making analysis easier and faster. Instead of looking for a needle in a haystack, producers can use data analytics to interpret viewing histories, web searches, or user engagement, to identify the kinds of content people actively seek.

Each element of a script can be measured and compared to external trends like social media engagement to gauge topic relevance and audience size through methods like sentiment analysis. From all of this data, studios can build custom models to cherry pick or highlight scripts to review and fast-track through the greenlighting process. Data analytics can help marketers tailor campaigns based on geographic data; there is even research suggesting a correlation between the number of Facebook likes a given film gets in a location and its chances of selling out at a nearby movie theater. Marketers can also use social listening to optimize release date based on geography, subject matter, or topic. At the movie and television post-production stage, data analytics can improve elements of editing and budgeting, turning eye-crossing spreadsheets into useful information and actionable insight.

One of the most significant strengths data analytics gives decision-makers is the ability to take a comprehensive but detailed view of the past and apply the lessons found to the present and future. A script could test positively, the story and topics correspond with trends observed in social analytics, and the movie can still fail. Furthermore, the buzz about songs or trailers of the movie can also be analysed by implementing Youtube API in elk stack

# Bibliography

- [1] Subhani shaikh, Nalamothu Naga Malleswara Rao 'A Conceptual Review of Elastic Search Survey Paper' International Journal for Research in Applied Science Engineering Technology (IJRASET)ISSN: 2321-9653 Volume 5 Issue XI November 2017
- [2] Pragya Gupta, Sreeja Nair 'Elasticsearch, future scope- Survey Paper' International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Index Copernicus Value (2013): 6.14 Impact Factor (2014):5.611
- [3] [https://mentormate.com/blog/what-is-elasticsearch/?utmmedium=Socialutm\\_source=Quorautm\\_term=2-1-17+Fulltext+Elasticsearch+Blog+Promo+JC](https://mentormate.com/blog/what-is-elasticsearch/?utmmedium=Socialutm_source=Quorautm_term=2-1-17+Fulltext+Elasticsearch+Blog+Promo+JC)
- [4] <https://www.latentview.com/blog/using-analytics-to-predict-movie-success>
- [5] <http://engineerom.trackmaven.com/blog/first-monthly-challenge-elasticsearch>
- [6] <https://logz.io/learn/complete-guide-elk-stack>
- [7] <https://storyfit.com/using-analytics-to-predict-movie-success>
- [8] <https://dennisdell.com/blog/hashtag-instagram/>
- [9] <https://www.elastic.co/>
- [10] <https://www.tutorialspoint.com/elasticsearch/elasticsearchbasicconcepts.html>

## Acknowledgement

We have great pleasure in presenting the report on **Movie Success Prediction**. We take this opportunity to express our sincere thanks towards our guide **Sachin B. Takmare** Department of IT, APSIT Thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Prof. Sachin Malave** Head of Department, COMPS, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Prof. Amol Kalugade** BE project co-ordinator, Department of IT, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

**Akash Gada**  
**15202021**

**Sayali Patil**  
**15102016**

**Sumit Tawde**  
**14102012**

**Manali Kajari**  
**14102025**

**Dewanshi Mishra**  
**15102014**

## Plagiarism Report



# Plagiarism Checker X Originality Report

**Similarity Found: 22%**

Date: Saturday, April 27, 2019

Statistics: 175 words Plagiarized / 784 Total words

Remarks: Medium Plagiarism Detected - Your Document needs Selective Improvement.

---

/ A Project Report on Movie Success Rate Prediction

Submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Engineering by Akash Gada(15202021)

Sayali Patil(15102016) Sumit Tawde(14102012) Manali Kajari(14102025)

Dewanshi Mishra(Student ID) Under the Guidance of Prof. Sachin Takmare

Department of Branch Name A.P. Shah Institute of Technology

G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615 UNIVERSITY OF MUMBAI

Academic Year 2017-2018 Approval Sheet

This Project Report entitled "Movie Success Rate Prediction" Submitted by "Akash Gada"(15202021),"Sayali Patil"(15102016),"Sumit Tawde"(14102012),"Manali Kajari"(14102025),"Dewanshi Mishra"(xxxxxxx)is approved for the partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Computer Engineering from University of Mumbai . Prof. Sachin Takmare Guide

Prof. Sachin Malave Head Department of Computer Engineering

Place:A.P.Shah Institute of Technology, Thane Date: CERTIFICATE

This is to certify that the project entitled "Movie Success Rate Prediction" sub-