# Anomaly detection of Log Files Using NLP

## Team Members:

*Group 04*

Dharan Kumar Kunati - 11547742

Venkata Sai Chandrika - 11555636

Bhanu Prasad Krishna Murthy - 11654250

Siddhartha Gurram - 11614580

## Motivation:

The need to better understand and utilize the enormous amounts of textual data generated in numerous sectors, such as software development, customer service, and finance, is what inspired a study on log analytics using NLP. Project logs are a useful source of information on a project's status, risks, and possibilities; yet, manually reviewing this data can be tedious and error prone. Project managers can better understand their projects, pinpoint areas for improvement, and make data-driven choices by automating the extraction of insights from project logs using NLP approaches. This may result in better project outcomes, cost savings, and higher efficiency. Additionally, NLP-based log analytics can offer a scalable and long-lasting solution as the volume of data produced by projects keeps increasing. In log analytics we want to focus on log anomaly detection, there are several reasons for it to focus particularly on anomaly detection. Some particular reasons include automating cybercrime detection mechanisms, fasten incident response teams, and improving efficiency.

## Significance:

Companies often have a large number of servers running different applications which produce lots of data including log data. Logs provide detailed records of activities, error messages, alerts for system administrators. They might also include time and date, IP addresses, Location information etc and can be used to track attackers, customers, monitor security etc. The basic hypothesis is we can use natural language processing capabilities to analyze and detect significant information from these logs. We want to train the model based on different log data and visualize these data in python. In

general log analytics platforms build custom parsers for each type of log data. But this task is quite tedious as the log data types can be ever increasing. In this project we want to make use of traditional machine learning classification models namely Naive Bayes, Decision Tree and Logistic Regression. and fine tune them to detect anomalous log messages and use metrics to figure out which models perform well for anomaly detection. Incident response teams can then make use of these detections to check whether an anomaly is a harmful signal which might be a cyber attack, system downtime or not.

# Objectives and Goals:

Our primary objective is to design and build an NLP anomaly detection model that can parse any log information. We then compare results of different models that are used for log anomaly detections. Some of them include Logistic Regression, Decision Tree and Naive Bayes.

Based on this log data, we then visualize the outputs to gain insights.

We took the dataset from github [A large collection of system log datasets for log analysis research (github.com)]

Tasks Involve

1. **Data Preparation**: For data preparation, we will be using Pandas to clean some log data, we will be using google colab to perform this.
2. **Analysis:** In this step, we analyze the log data based on different attributes. We visualized the most commonly used keywords in our dataset using word cloud and analyzed the dataset to infer some possible anomalies manually.
3. **Tokenization:** Here we will be using prebuilt tokenizers to tokenize the log datasets and build a “bag of words” with which we can build our model.
4. **Perform Keyword Identification:** Some of the keywords include “alerts”, “timestamps”, “location”, “service” etc which can be identified from log messages.
5. **Class Identification:** We want to categorize each log message to a class like whether it is an alert message or warning or error message etc.
6. **Log Anomaly Detection:** To perform log anomaly detection, we have used the same models for class identification and decrease the number of classes to 2{anomaly,not an anomaly}, we need to fine tune the dataset as the dataset is largely imbalance dataset.
7. **Visualization**: Finally we want to visualize all the keywords in the log set.

# Related Work:

With each log message often being a semi-structured text string, system logs are frequently used by large computer systems for troubleshooting. To find anomalous log entries, the ordinary methods precisely employ keywords or regular expressions. These techniques are unable to identify the malicious attacks based on a series of actions where each log entry appears to be normal, but the overall pattern is abnormal. Despite the true fact that the rule-based techniques can achieve good and maximum accuracy, they only can identify the pre-defined anomalous scenarios and need significant manual engineering. Whenever the new attack types are launched by the attackers the rule-based techniques are unable to give and deliver high performance.

Numerous learning-based strategies are suggested as malicious attacks get increasingly complex. Three steps typically make use of the pipeline for these approaches, namely the first step is to convert log messages into log keys, a log parser is used first. The Second step is to create a feature vector to represent a series of log keys in a sliding window using a feature extraction method l. Finally, the third step is an unsupervised technique typically used to find the unusual sequences.

When a physical content differs from a proper set of predicted regular log messages, an anomalous sequence will be picked up on. According to the new study and recent information builds the graph embedding method to create a graph based log sequences in order to find anomalies.

One of the the trending research project on log anomaly detection is HelenGuohx/logbert: log anomaly detection via BERT (github.com) techniques summarized in LogBERT: Log Anomaly Detection via BERT | IEEE Conference Publication | IEEE Xplore.
Here the authors discuss their anomaly detection model based on bert(bidirectional encoders) and compare the results with traditional models PCA, isolation Forest. The authors claimed that logbert outperforms state of the art traditional models. Since they have given metrics at the end and we are using the same bgl dataset, we can compare our model results with log bert at the end.

ClusterLog: Clustering Logs for Effective Log-based Anomaly Detection | IEEE Conference Publication | IEEE Xplore
In the IEEE publication "Clusterlog: Clustering Logs for Effective Log-based Anomaly Detection," a unique method for identifying anomalies in log data is presented. The strategy that the study suggests takes advantage of the innate structure and patterns found in logs by grouping them based on semantic similarity enables efficient anomaly detection. In this approach, the authors seek to reduce the number of unique log keys

used to represent log sequences by grouping logs that are semantically and sentimentally similar. This is done through log clustering, where logs with similar patterns or behavior are grouped together, allowing for more efficient and effective representation of log sequences with fewer unique log keys.

# Dataset:

We have collected the data from project LOGPAI (Log-based Performance Analysis and Investigation). It provides a comprehensive suite of tools and techniques for processing, analyzing, and visualizing large-scale log data from various sources, such as system logs, application logs, and network logs. In the Logpai project we have taken our dataset from loghub(a large collection of system log datasets). The log dataset we chose is BGL dataset. BGL is an open dataset of logs collected from a BlueGene/L supercomputer system at Lawrence Livermore National Labs (LLNL) in Livermore, California. The alert category tags in the log are used to distinguish between alert and non-alert messages. "-" denotes non-alert messages in the first column of the log while other characters denote alert messages.

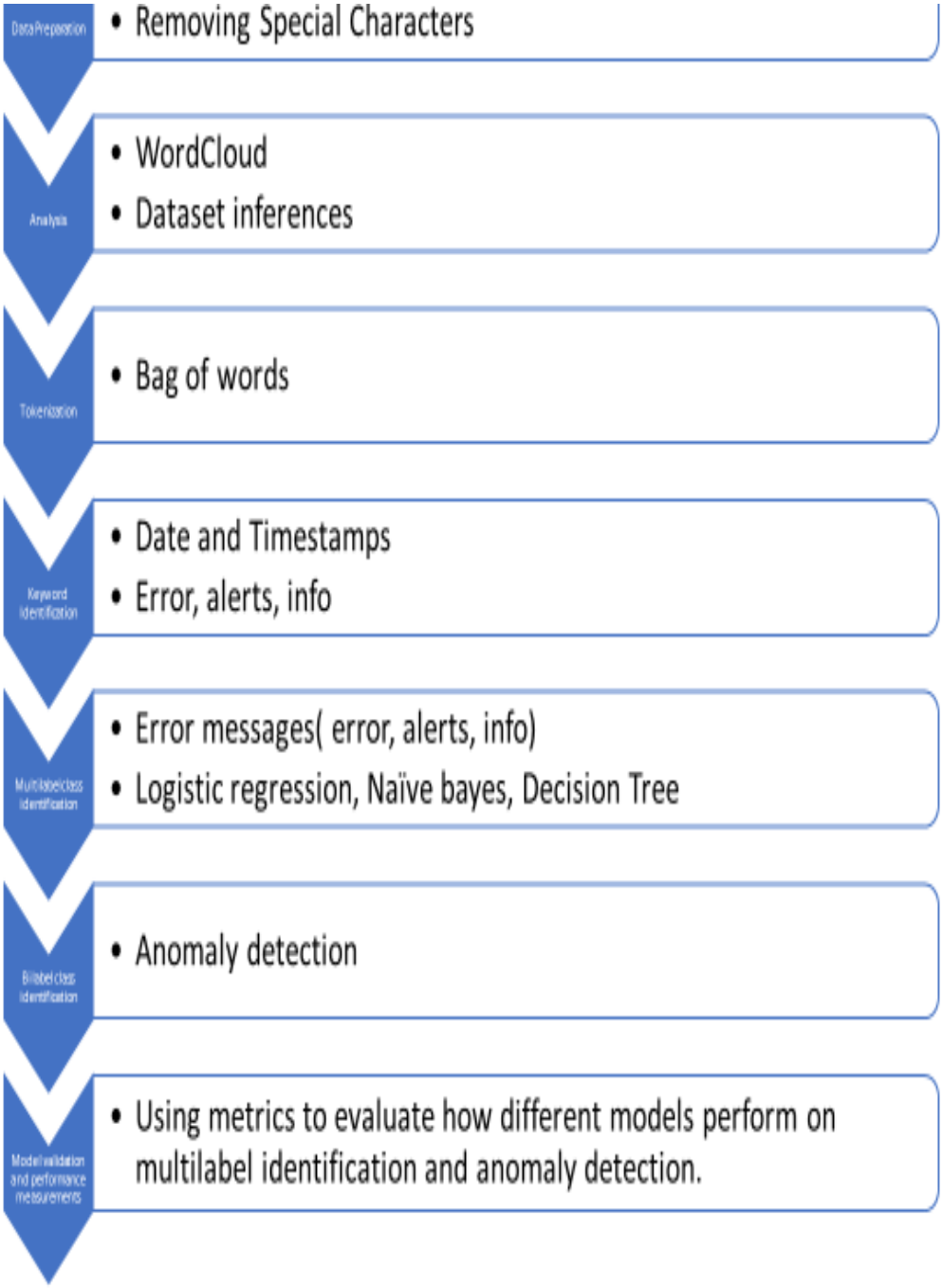
Web page: LOGHUB (github.com)

# Features:

Some of the project features include

1. **Log Parsing**: The system should be able to extract import information in the logs. Logs can be of windows log files, Apache server log files, Hadoop log files.
2. **Keyword Extraction:** From the logs, the system should be able to extract the most crucial terms that may be used to spot patterns and trends.
3. **Multilabel Class Identification:** Assign label to each log message. Common logs can be “error”,”warning”,”anomaly”.
4. **Log Anomaly Detection**: The dataset needs to be fine-tuned because it is generally imbalanced. To perform log anomaly detection, we used the same models for class identification and reduced the number of classes to two: anomaly and not an anomaly.
5. **Visualization:** The system must be able to display log data in an understandable manner, such as graphs, tables, and charts, which may be used to spot patterns and trends.

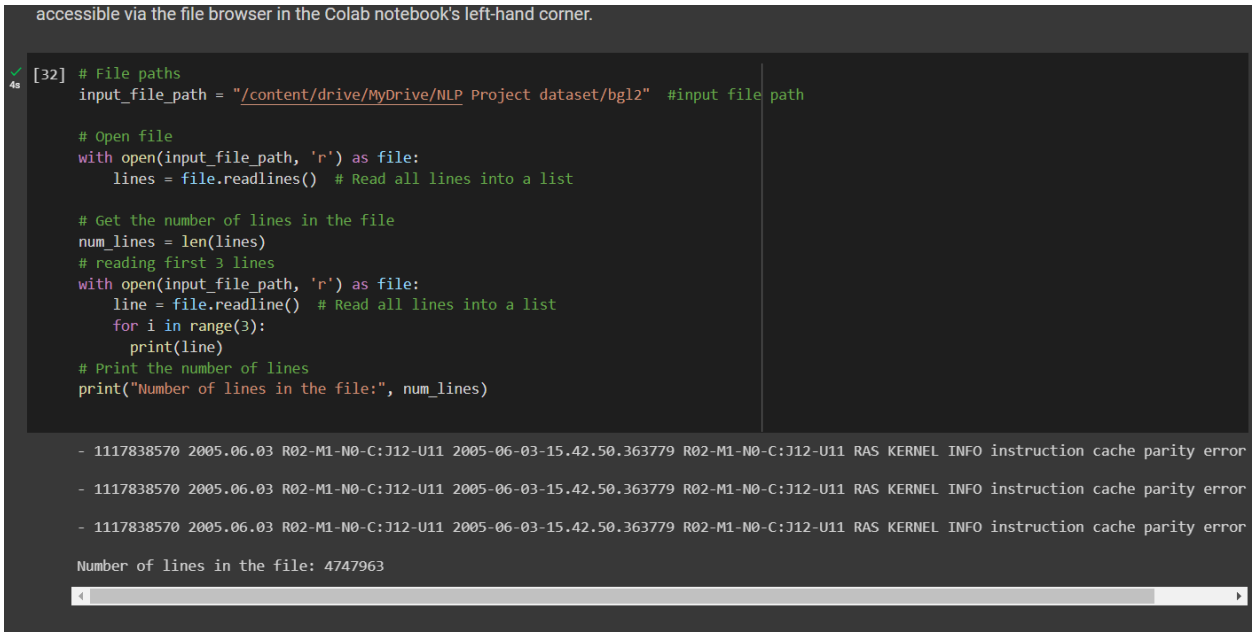
**Steps Involved:**



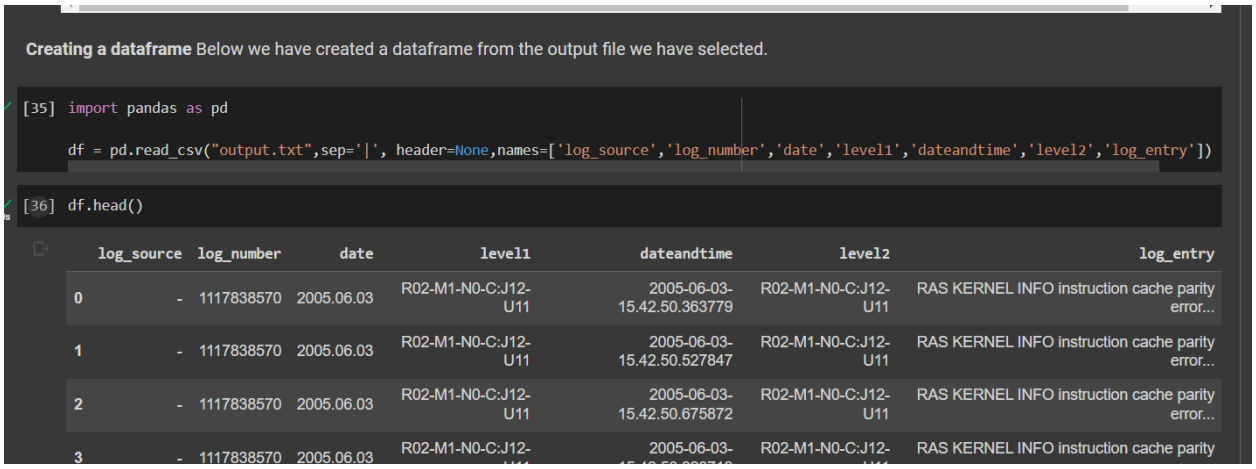

# MODEL IMPLEMENTATION:

**Data Preprocessing:**

In this step, we modified the dataset to use pipes instead of spaces so that it would be easier to load it as a pandas dataframe.




Loaded the dataset into dataframe

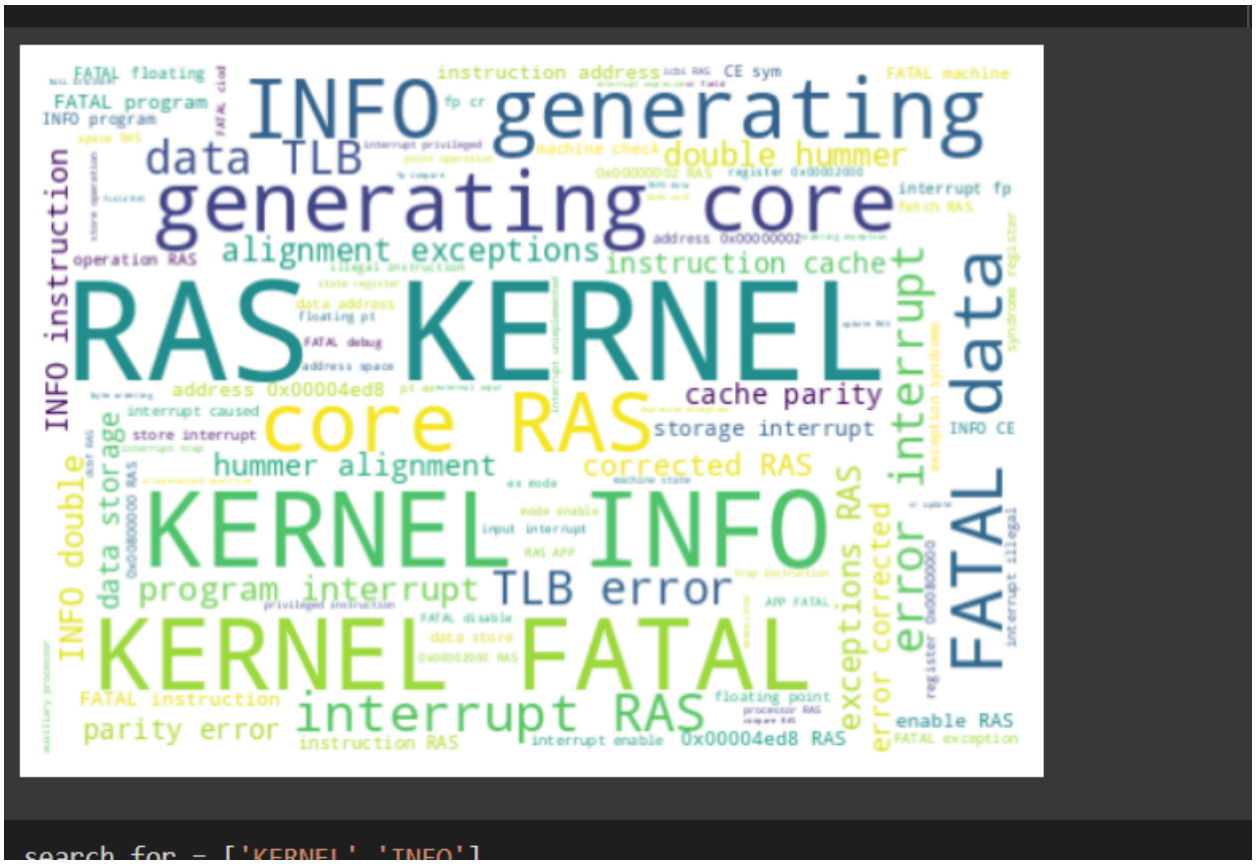



Total number of rows in our dataset are 1138779

Some analysis include

The length of possible lines with words containing Alert, ERR, Warn are : 4972

Word cloud images of log data:




With our data in a DataFrame, a WordCloud is an intriguing visualization to highlight frequent word sequences in the data. By adjusting the font size according to the frequency of each word sequence, a word cloud displays all frequent word sequences. Font sizes would be higher for word groups that occur more frequently. We need to import the WordCloud module in order to generate it. The following line of code defines a function that takes a corpus parameter, which is a set of texts. Concatenating all of the text in column log_entry creates this corpus. To ensure that common English terms like "a," "is," "the," etc. are not taken into account in this analysis, a collection of stopwords is employed in the formation of the WordCloud instance.

We can see the most common words in log_entry are KERNEL, RAS, INFO. Later we can see whether we see these words in our entity recognition or not.

We have also performed trigram frequency analysis. N-grams are a crucial text processing concept in NLP. A sequence of n words known as a "n-gram" is used as a point of reference; examples include the single-worded "unigram," the two-word "bigram," and the three-word "trigram." Using the TfidfVectorizer from scikit-learn, we extract n-grams from our log data.  The general technique of converting a group of text documents into numerical feature vectors is known as vectorization. The Bag of Words

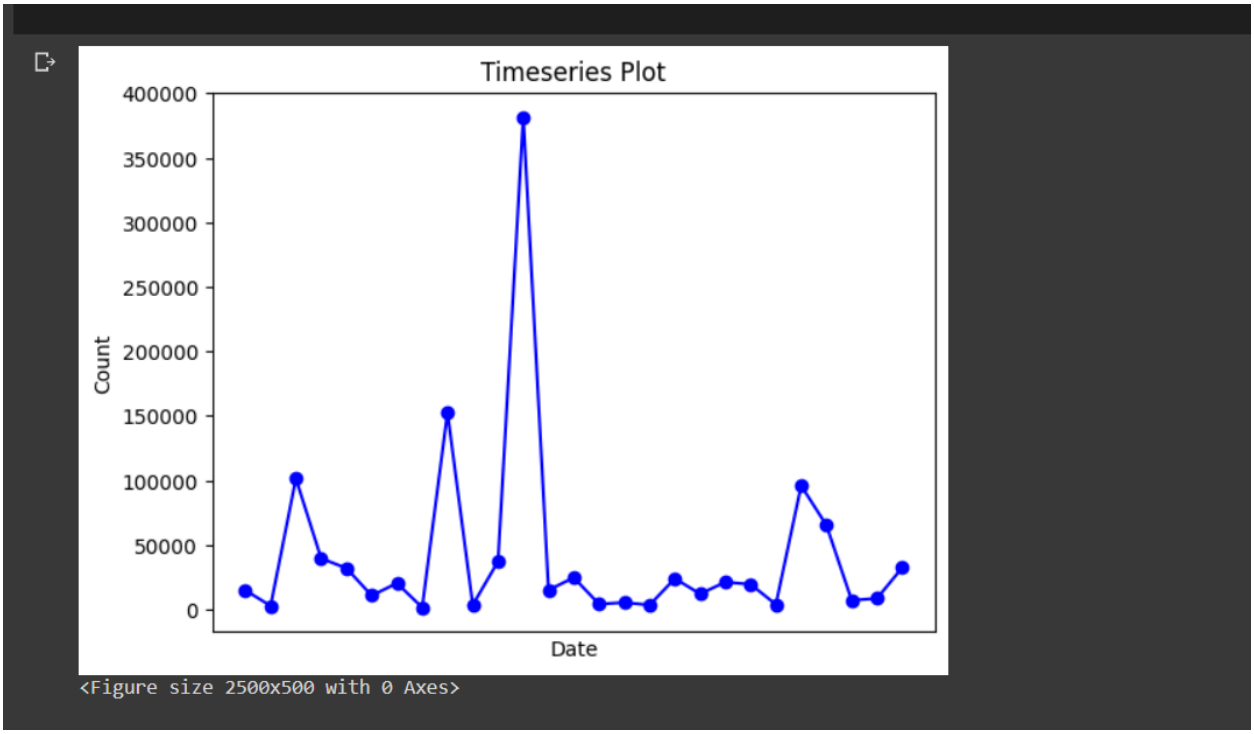
representation, often known as the "Bag of n-grams," is a particular technique that entails tokenization, counting, and normalization. Below, we can see some results

```
    print("{0:35}{1:}".format(word,freq))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
ras kernel fatal                    117423.1014498484
kernel fatal data                   98155.42853753311
ras kernel info                     78625.40254027644
fatal data tlb                      75935.4544566271
data tlb error                      75935.4544566271
tlb error interrupt                 75935.4544566271
kernel info generating              61126.54520074478
info generating core                61126.54520074478
data storage interrupt              39189.99035502865
fatal data storage                  39189.82359739589
kernel info instruction             29468.18991361574
kernel fatal instruction            29103.17509904902
fatal instruction address           29101.326228790058
cache parity error                  28590.698064421747
info instruction cache              28588.482073965395
instruction cache parity            28588.482073965395
parity error corrected              28588.482073965395
instruction address 0x00004ed8      26941.960924278344
kernel fatal program                21116.405408688872
fatal program interrupt             21116.405408688872
```

Time series plots:

The general understanding is more logs will be generated at the time of an attack or an anomaly or in the case of system downtime because applications try to generate a general set of logs for normal operations and some set of different logs for dowtimes and system defects. Based on that inference, we generated a time series graph with xtitle dates and ytitle number of logs generated on each day




We can see it is not uniform and on some days there were a lot more logs being generated.

Later we performed some more cleansing and data preparation. In next iteration, we perform model building and model evaluation.

# PROJECT MANAGEMENT

| Name | Tasks Completed | Tasks Pending(Will be done by next iteration) |
|---|---|---|
| Dharan Kumar Kunati | Detailed analysis of dataset, preprocessing, Time series plots, report development | Logistic regression model for both multilabel class identification and anomaly detection. |
| Chandrika Gudipati | Literature Survey, Related work, word cloud module, report generation. | Decision Tree model for both multilabel class identification and anomaly detection |
| Bhanu Prasad Krishna Murthy | Literature Survey, Code review, n-gram analysis, report generation | NaiveBayes model for both multilabel class identification and anomaly detection |

| | | |
|---|---|---|
| Siddhartha Gurram | Inferences on the dataset, Model evaluation schemes, report generation. | Model evaluation metrics and final summary analysis. |

# COLAB Notebook, Files, Github links:

Below is link to our colab notebook
Colab Notebook
Below is our github code link
DHARAN656/NLP-Loganomalydetection: Log anomaly detection based on NLP models (github.com)
We have pasted the video link in video link branch
DHARAN656/NLP-Loganomalydetection at videolink (github.com)

# References:


Log Parsing With AI: Faster and With Greater Accuracy - Database Trends and Applications (dbta.com)

(22) Log analysis with Natural Language Processing leads to interesting insights | LinkedIn

https://jiemingzhu.github.io/pub/slhe_issre2016.pdf

1. Guo, Haixuan. “logBERT github repo.” https://github.com/HelenGuohx/logbert.

2. Guo, Haixuan, et al. “LogBERT: Log Anomaly Detection via BERT.” 2021,

https://arxiv.org/abs/2103.04475.

3.Yinglung Liang, et al, . “Filtering failure logs for a BlueGene/l”. In 2005 International

Conference on Dependable Systems and Networks

(DSN’05). IEEE, 476–485.

4. USENIX, BGL Dataset. https://www.usenix.org/cfdr-data#hpc4