

## Network Fault Severity Detection Through Log Data Analysis

Abinaya N<sup>1</sup>, Sangeetha M<sup>2</sup>, Aswin S<sup>3</sup>, Dr. W. Rose Varuna<sup>4</sup>

<sup>1</sup>M.Sc. Information Technology, Department of Information Technology, Bharathiar University, Coimbatore-641046

Email ID: [abinayabscst@gmail.com](mailto:abinayabscst@gmail.com)

<sup>2</sup>M.Sc. Information Technology, Department of Information Technology, Bharathiar University, Coimbatore-641046

Email ID: [Sangeethamoorthy2710@gmail.com](mailto:Sangeethamoorthy2710@gmail.com)

<sup>3</sup>M.Sc. Information Technology, Department of Information Technology, Bharathiar University, Coimbatore-641046

Email ID: [achuaswin58589@gmail.com](mailto:achuaswin58589@gmail.com)

<sup>4</sup>Assistant Professor, Department of Information Technology, Bharathiar University, Coimbatore-641046

Email ID: [rosevaruna@buc.edu.in](mailto:rosevaruna@buc.edu.in)

Cite this paper as: Abinaya N, Sangeetha M, Aswin S, Dr. W. Rose Varuna, (2025) Network Fault Severity Detection Through Log Data Analysis. *Journal of Neonatal Surgery*, 14 (14s), 292-302.

### ABSTRACT

Network reliability is essential for faultless communication in the telecommunication sector. The paper, Network Fault Severity Detection Using Log Data, focuses on predicting fault severity in the network of Telstra based on machine learning. The research establishes a predictive model to predict fault severity into three categories: 1 denotes a few errors, 2 numerous faults in them, and 0 denotes no defects. The approach requires heavy data preprocessing, feature design, and data exploration to establish patterns in logs. Machine learning algorithms like CatBoost, Random Forest, XgBoost, and LightGBM are utilized for accurate prediction. Feature importance analysis is also used to further improve model explainability by isolating major factors of failure. The study highlights the predictive analytics contribution to enhancing network reliability, minimizing downtime, and maximizing customer satisfaction. The method aids Telstra and other telecommunication providers in optimizing service quality, resource efficiency, and maintenance. The scalable approach suggested guarantees proactive fault detection, thereby reducing operational costs and enhancing overall network performance.

**Keywords:** Log Data Analysis, CatBoost, Random Forest, XGBoost, LightGBM, Telecommunications Network.

### 1. INTRODUCTION

Cybersecurity is a rapidly developing field due to the need for advanced tools to detect, prevent, and counteract new cyber threats. One of the main issues in this field is analyzing time-based threats in network traffic, which is inherently complicated. With increasing security controls, cyberattacks become more advanced attacks, and they need robust defense systems. In order to protect vital network infrastructures, a combination of signature detection, anomaly detection, and machine learning or deep learning-based approaches is required [1]. The Internet is part of our daily life with the majority of us relying on it for essential services. But network faults can disrupt access or affect service quality, thus user experience. These defects result from various causes, including Customer On-Premise Equipment (CPE), the core network, and the external plant infrastructure. This research is focused on detecting and diagnosing faults from these three principal sources through log data analysis. Out-of-control network issues that include, e.g., power outage or faults resulting from customers are excluded in the research [2].

About 17% of the over 16 million records in the CSE-CIC-IDS2018 dataset, which is composed of an attacking machine and a malicious device network, are six different forms of malware traffic. However, the dataset is class-imbalanced in binary and multi-class classifications, posing difficulty in detecting fault severity accurately [3]. Electrical power networks depend on an integrated system of segments—generation, transmission, and distribution—to transmit electricity effectively. Power transfer is made necessary by transmission lines, which are susceptible to faults. Fault detection and management require more than manual intervention since this can lead to reliability disruption, outages, and system failures. Correct fault categorization is important for network security, allowing immediate detection and segmentation in order to avoid system failure and ensure energy stability [4]. In this study, we apply the Telstra network log data to investigate and identify network fault severity. The data offers useful information regarding different network problems, such as faults from CPE, core network faults, and infrastructure-related outages. With this information, we will create a fault severity detection model that will effectively identify, classify, and predict network faults to enhance service reliability and reduce downtime.

### 1.1 Problem Statement

In the telecommunications sector, network reliability is a key factor for uninterrupted communication and least service degradation. Yet fault severity identification and prediction are very challenging due to the sheer quantity of log data produced by the network. Existing fault detection solutions are usually slow and reactive and result in unnecessary downtime, escalated operational expenses, and lower customer satisfaction.

This research adopts ML models such as CatBoost, Random Forest, XgBoost, and LightGBM to aptly address Telstra's fault severity prediction issue. Management of log data efficiently, derivation of useful features, and the formation of prediction models that can map fault severity to three classes - 0 for no faults, 1 for few faults, and 2 for several faults are key challenges.

By constructing an effective predictive model, the proposed work hopes to achieve an efficient fault detection method that can ensure optimal scheduling of maintenance, decrease downtime, and enhance resource management. The designed approach is flexible and extensible to other telecommunications companies and can be applied with a data-based approach to offer increased network stability and service performance.

## 2. LITERATURE SURVEY

Network fault detection and severity analysis have attracted substantial interest with the growing dependency on telecommunication infrastructure. Different studies have used machine learning to improve fault detection and classification. Detection of network fault severity is vital for the stability and reliability of telecommunication and IT infrastructure. As the network environment grows complex, predictive fault detection by analyzing log data is now essential. The following is a survey of literature with the pertinent works and the usage of four fundamental algorithms—Random Forest, CatBoost, XGBoost, and LightGBM—within network fault severity prediction.

Shilin He et al. (2021) [5] suggested a systematic review of log analysis automation for reliability engineering with an emphasis on operations such as log compression, parsing, and anomaly detection. The paper discussed the level at which fault detection relies on machine learning, especially boosting algorithms. According to the authors, real-time processing and unstructured log processing are challenging tasks, and ensemble models and deep learning models will improve failure prediction and anomaly detection even more.

Mohamed Saied and his team (2023) [6] took a deep dive into comparing various boosting-based machine learning algorithms for detecting intrusions in IoT networks. They looked at Random Forest, CatBoost, XGBoost, and LightGBM. The results from their study are quite useful for classifying the severity of network faults, as these models excel at sifting through large data log files to spot anomalies. Their research also highlighted that while optimization algorithms significantly boost fault detection accuracy, they do face challenges related to computational load and the intricacies of feature engineering.

Kennedy Okokpujie et al. (2024) [7] had a study in which they had simulated a log-based predictive maintenance model for the purpose of detecting faults in telecommunication networks based on received signal levels (RSL). They used methods such as Random Forest, Gradient Boosting, and K-Nearest Neighbors (KNN) in the simulations. The results showed that Random Forest performed better than the other models in terms of detecting network faults, thus underscoring the significance of log-based predictive maintenance in order to enable minimal downtime.

Ogobuchi Daniel Okey et al. (2022) [8], who suggested BoostedEnML, an ensemble learning-based methodology integrating XGBoost and LightGBM to improve IoT system cyberattack detection. The research showed that applying data balancing strategies such as SMOTE enhances classification performance in faults. Their research indicates that the same ensemble methods could be extended to network fault detection, especially dealing with imbalanced fault data and high classification accuracy.

Shuai Li et al. (2024) [9] proposed an enhanced LightGBM model that utilizes a hybrid optimization approach to enhance industrial fault warning and detection systems. The researchers designed the hybrid strategy by integrating the Arithmetic Optimization Algorithm (AOA), Simulated Annealing (SA), and other high-level search techniques to precisely optimize the parameters of LightGBM. Using their experiments, they showed that optimized LightGBM performed better compared to conventional methods of fault detection in real case studies. When compared with XgBoost as well, their model was efficient in computation but highly accurate too.

## 3. METHODOLOGY

### 3.1 Data Modeling

The approach starts with feature extraction, in which important parameters like voltage features, phase differences, and harmonic contents are sensed to improve model accuracy. Microgrid operation data is then classified using LightGBM classifier to identify possible faults. The authors use a bagging (Bootstrap Aggregating) technique to stabilize classification, training multiple LightGBM classifiers on various subsets of data [10]. The process begins with data gathering, in which the

IoT-based electrical grid monitoring system's historical sensor records are gathered. The data is preprocessed, including noise filtering, outlier removal, and feature selection, in an effort to optimize the model's performance. LightGBM is then trained using gradient boosting techniques [11]. This project entails the prediction of network fault severity level in Telstra's infrastructure using given log data, identified to particular times and locations. The primary datasets, train.csv and test.csv, document information with a unique "id" linking to other files. Fault severity is classified into three levels: 0 means no fault, 1 for minor fault, and 2 for major faults. Some features are based on independent CSV files—namely event\_type, log\_feature, resource\_type, and severity\_type. The severity\_type in its corresponding file is the warning message type which has been filtered from system logs and is a categorical feature devoid of any numeric order. The main target variable, fault\_severity, found in the training data and shown in Fig 1.1, corresponds to actual network problems reported by users. Additional files supporting the dataset include: event\_type.csv for identifying types of events (Fig 1.2), log\_feature.csv for capturing specific log attributes (Fig 1.3), resource\_type.csv for detailing resources involved (Fig 1.4), sample\_submission.csv as a formatted template for predictions (Fig 1.5), severity\_type.csv for the warning message classifications (Fig 1.6), test.csv containing records for prediction (Fig 1.7), and train.csv, which serves as the base for model training (Fig 1.8).

Name	Type	Compressed size	Password p...	Size	Ratio	Date modified
event_type	Microsoft Excel Comma S...	94 KB	No	591 KB	85%	18-11-2019 02:54 AM
log_feature	Microsoft Excel Comma S...	221 KB	No	1,111 KB	81%	18-11-2019 02:54 AM
resource_type	Microsoft Excel Comma S...	63 KB	No	441 KB	86%	18-11-2019 02:54 AM
sample_submission	Microsoft Excel Comma S...	30 KB	No	125 KB	76%	18-11-2019 02:54 AM
severity_type	Microsoft Excel Comma S...	57 KB	No	388 KB	86%	18-11-2019 02:54 AM
test	Microsoft Excel Comma S...	55 KB	No	202 KB	74%	18-11-2019 02:54 AM
train	Microsoft Excel Comma S...	39 KB	No	148 KB	75%	18-11-2019 02:54 AM

Fig 1.1 File Description

A1						
	A	B	C	D	E	F
1	id	event_type				
2	6597	event_type 11				
3	8011	event_type 15				
4	2597	event_type 15				
5	5022	event_type 15				
6	5022	event_type 11				
7	6852	event_type 11				
8	6852	event_type 15				
9	5611	event_type 15				

Fig 1.2 Sample dataset of event\_type.csv

	A	B	C	D	E	F
1	id	log_featur	volume			
2	6597	feature 68	6			
3	8011	feature 68	7			
4	2597	feature 68	1			
5	5022	feature 17	2			
6	5022	feature 56	1			
7	5022	feature 19	4			
8	5022	feature 71	3			
9	6852	feature 20	2			

Fig 1.3 Sample dataset of log\_feature.csv

	A	B	C	D	E	F
1	id	resource_type				
2	6597	resource_type 8				
3	8011	resource_type 8				
4	2597	resource_type 8				
5	5022	resource_type 8				
6	6852	resource_type 8				
7	5611	resource_type 8				
8	14838	resource_type 8				

**Fig 1.4 Sample dataset of resource\_type.csv**

	A	B	C	D	E	F
1	id	predict_0	predict_1	predict_2		
2	11066	0	1	0		
3	18000	0	1	0		
4	16964	0	1	0		
5	4795	0	1	0		
6	3392	0	1	0		
7	3795	0	1	0		
8	2881	0	1	0		

**Fig 1.5 Sample dataset of sample\_submission.csv**

	A	B	C	D	E	F
1	id	severity_type				
2	6597	severity_type 2				
3	8011	severity_type 2				
4	2597	severity_type 2				
5	5022	severity_type 1				
6	6852	severity_type 1				
7	5611	severity_type 2				

**Fig 1.6 Sample dataset of severity\_type.csv**

	A	B	C	D	E	F
1	id	location				
2	11066	location 481				
3	18000	location 962				
4	16964	location 491				
5	4795	location 532				
6	3392	location 600				
7	3795	location 794				
8	2881	location 375				

**Fig 1.7 Sample dataset of test.csv**

	A	B	C	D	E	F
1	id	location	fault_severity			
2	14121	location 1:	1			
3	9320	location 9:	0			
4	14394	location 1:	1			
5	8218	location 9:	1			
6	14804	location 1:	0			
7	1080	location 6:	0			
8	9731	location 6:	0			

**Fig 1.8 Sample dataset of train.csv**

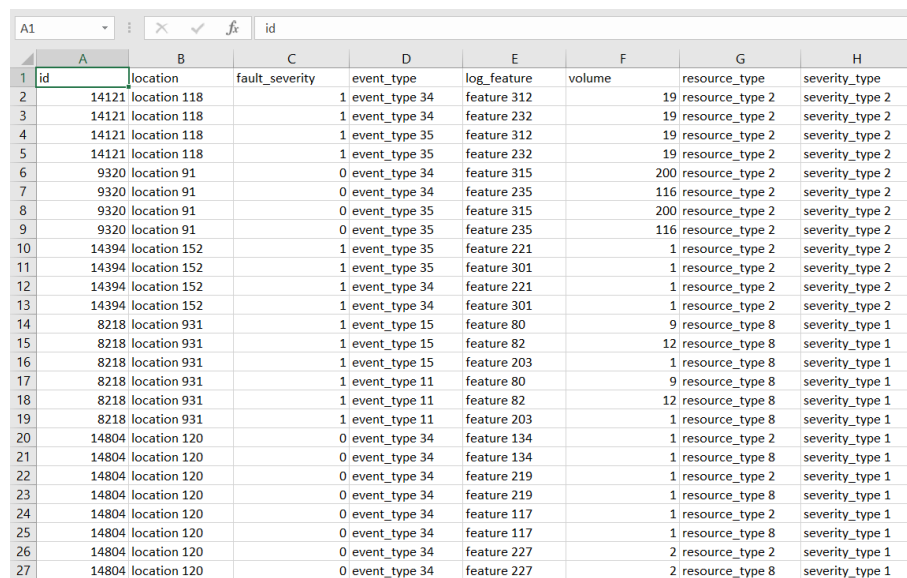
The dataset consists of multiple sub-datasets, which are merged to form a final structured dataset. Each dataset file provides

specific attributes that contribute to the predictive model. After merging the sub-datasets, the final dataset (Table 1) follows this schema:

**Table 1 Final Dataset**

Column name	Data Type	Description
Id	Integer	Unique identifier for each fault record.
location	Integer	Network location where the fault occurred.
severity_type	Categorical	Additional severity classification for the fault.
event_type	Categorical	Type of event linked to the fault occurrence.
log_feature	Categorical	Log-based feature associated with the fault.
volume	Integer	Count of occurrences for a log feature.
resource_type	Categorical	Type of affected network resource (e.g., server, router).
fault_severity	Categorical (0,1,2)	<b>Target variable:</b> Severity level (0: No fault, 1: Few faults, 2: Many faults) - Only in train.csv.

The dataset for "Network Fault Severity Detection Through Log Data Analysis" is constructed by merging multiple sub-datasets into a single dataset (Fig 1.9). This combined dataset ensures a comprehensive feature set for accurate fault severity prediction. Below is the example of the dataset:



	A	B	C	D	E	F	G	H
1	id	location	fault_severity	event_type	log_feature	volume	resource_type	severity_type
2	14121	location 118		1 event_type 34	feature 312	19	resource_type 2	severity_type 2
3	14121	location 118		1 event_type 34	feature 232	19	resource_type 2	severity_type 2
4	14121	location 118		1 event_type 35	feature 312	19	resource_type 2	severity_type 2
5	14121	location 118		1 event_type 35	feature 232	19	resource_type 2	severity_type 2
6	9320	location 91		0 event_type 34	feature 315	200	resource_type 2	severity_type 2
7	9320	location 91		0 event_type 34	feature 235	116	resource_type 2	severity_type 2
8	9320	location 91		0 event_type 35	feature 315	200	resource_type 2	severity_type 2
9	9320	location 91		0 event_type 35	feature 235	116	resource_type 2	severity_type 2
10	14394	location 152		1 event_type 35	feature 221	1	resource_type 2	severity_type 2
11	14394	location 152		1 event_type 35	feature 301	1	resource_type 2	severity_type 2
12	14394	location 152		1 event_type 34	feature 221	1	resource_type 2	severity_type 2
13	14394	location 152		1 event_type 34	feature 301	1	resource_type 2	severity_type 2
14	8218	location 931		1 event_type 15	feature 80	9	resource_type 8	severity_type 1
15	8218	location 931		1 event_type 15	feature 82	12	resource_type 8	severity_type 1
16	8218	location 931		1 event_type 15	feature 203	1	resource_type 8	severity_type 1
17	8218	location 931		1 event_type 11	feature 80	9	resource_type 8	severity_type 1
18	8218	location 931		1 event_type 11	feature 82	12	resource_type 8	severity_type 1
19	8218	location 931		1 event_type 11	feature 203	1	resource_type 8	severity_type 1
20	14804	location 120		0 event_type 34	feature 134	1	resource_type 2	severity_type 1
21	14804	location 120		0 event_type 34	feature 134	1	resource_type 8	severity_type 1
22	14804	location 120		0 event_type 34	feature 219	1	resource_type 2	severity_type 1
23	14804	location 120		0 event_type 34	feature 219	1	resource_type 8	severity_type 1
24	14804	location 120		0 event_type 34	feature 117	1	resource_type 2	severity_type 1
25	14804	location 120		0 event_type 34	feature 117	1	resource_type 8	severity_type 1
26	14804	location 120		0 event_type 34	feature 227	2	resource_type 2	severity_type 1
27	14804	location 120		0 event_type 34	feature 227	2	resource_type 8	severity_type 1

**Fig 1.9 Example of Dataset**

## 3.2 Model Training

### 3.2.1 CatBoost Model Training

The algorithm specially designed to be used with categorical variables without deep preprocessing. Contrary to the classical gradient boosting models, CatBoost utilizes ordered boosting, which avoids target leakage because leaf values are calculated based on previous observations only. It also utilizes symmetric decision trees, which means that every layer in the tree is equilibrated, resulting in more efficient training and better generalization. The other important characteristic of CatBoost is that it can carry out target encoding for categorical features, making manual one-hot encoding less necessary. In training (Fig 1.10), CatBoost tries to reduce log loss or cross-entropy loss when it is used in a classification problem. Hyperparameters

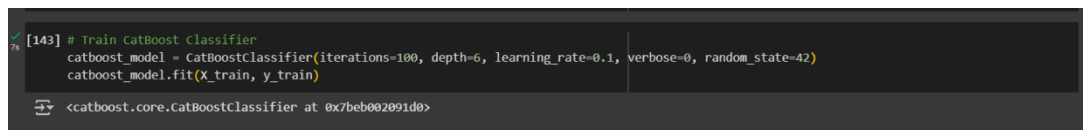
such as iterations, learning rate, depth, and L2 regularization are tuned to enhance performance. Feature importance analysis is also implemented within the algorithm, making it easier to identify the most influential factors contributing to network fault severity. The power of CatBoost lies in its ability to handle large-sized categorical data with minimal preprocessing, making it highly suitable to carry out analysis for network log data. The loss function and training process are as follows:

#### Objective Function (Loss Function)

CatBoost optimizes the loss function using gradient descent as shown in Equation (1).

$$L = \sum_{i=1}^N \ell(y_i, f(x_i)) \quad \rightarrow \text{Equation (1)}$$

In this above equation (1),  $\ell(y_i, f(x_i))$  represents the loss function—such as Log Loss for classification tasks or Mean Squared Error (MSE) for regression—where  $f(x_i)$  is the model's predicted value and  $y_i$  denotes the actual target label.



```
[143] # Train CatBoost Classifier
catboost_model = CatBoostClassifier(iterations=100, depth=6, learning_rate=0.1, verbose=0, random_state=42)
catboost_model.fit(X_train, y_train)

<catboost.core.CatBoostClassifier at 0x7beb002091d0>
```

**Fig 1.10 Training CatBoost Model**

### 3.2.2 Random Forest Model Training

It is an ensemble technique with numerous decision trees aiming to predict better and prevent overfitting. The model works by creating several subsets of training data through bootstrap sampling (bagging), which then trains a single decision tree on each subset. Unlike one decision tree, whose high variance can make it invalid, Random Forest reduces variance by averaging predictions from multiple trees. All trees in the Random Forest are trained (Fig 1.11) on a randomly selected subset of features in order to create diversity between models and prevent reliance on a single feature. Prediction is made final with majority voting, where the class label voted by all the trees is taken. Computation time and performance are struck in equilibrium during the hyperparameters like lowest data division, maximum length, and number of trees ( $n\_estimators$ ) being optimized. Random Forest is extremely tolerant to noisy data and missing values and thus may be an optimal choice for telecommunication networks' fault severity classification.

#### Prediction Function

The ultimate output of a Random Forest is the average (for regression) or majority vote (for classification) of multiple decision trees prediction function equation as shown in Equation (2).

$$\hat{y} = \frac{1}{M} \sum_{m=1}^M f_m(x) \quad \text{Equation (2)}$$

In this above equation (2),  $f_m(x)$  is the prediction from the  $m$ -th single tree,  $M$  is the number of decision trees, and  $\hat{y}$  is the final prediction.



```
[135] # Train Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

RandomForestClassifier
RandomForestClassifier(random_state=42)
```

**Fig 1.11 Training Random Forest Model**

### 3.2.3 XGBoost Model Training

Extreme Gradient Boosting (XGBoost) is a tweaked boosting algorithm for building trees step by step with each tree tending



to counter the mistakes done by the prior one. Differently, Random Forest randomly grows all the trees separately; XGBoost applies gradient boosting, wherein additional trees are created to minimize previous trees' residual errors. It utilizes shrinkage (learning rate tuning) so that it prevents overfitting by fine-tuning the model step by step. XGBoost introduces regularization methods like L1 (Lasso) and L2 (Ridge) penalties, which limit overfitting risks without sacrificing accuracy. XGBoost further employs tree pruning and early stopping to enhance efficiency in computation. One of the key advancements with XGBoost is the weighted quantile sketch algorithm, by which the model can handle sparse datasets optimally, a common characteristic of network log data. Hyperparameters such as number of boosting iterations, maximum depth of trees, learning rate, and column sampling ratio are optimized to achieve maximum model performance (Fig 1.12). Due to the fact that it is computationally light and can handle big data, xgboost has been extensively used for network failure prediction and fault severity classification.

#### Objective Function

XGBoost minimizes the sum of the loss function and a regularization formula as shown in Equation (3):

$$L(\theta) = \sum_{i=1}^N \ell(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k) \quad \text{Equation (3)}$$

Here,  $\ell(y_i, \hat{y}_i)$  denotes the loss function—such as Mean Squared Error for regression or Log Loss for classification—while  $\Omega(f_k)$  represents the regularization component used to control model complexity and reduce overfitting, as illustrated in Equation (4).

$$\Omega(f_k) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2 \quad \text{Equation (4)}$$

Here,  $T$  is the number of leaves in the tree,  $\gamma$  is the regularization term that manages the complexity by imposing a penalty on the number of leaves,  $\lambda$  is the L2 regularization parameter that is used to avoid overfitting, and  $w_i$  is the weight given to every individual leaf.

```
[174]: # Train XGBoost Classifier
xgb_model = XGBClassifier(n_estimators=100, learning_rate=0.1, max_depth=6, random_state=42, use_label_encoder=False, eval_metric='mlogloss')
xgb_model.fit(X_train, y_train)

/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [14:21:31] WARNING: /workspace/src/learner.cc:740:
Parameters: { 'use_label_encoder' } are not used.

warnings.warn(msg, UserWarning)

XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric='mlogloss',
               feature_types=None, gamma=None, grow_policy=None,
               importance_type=None, interaction_constraints=None,
               learning_rate=0.1, max_bin=None, max_cat_threshold=None,
               max_cat_to_onehot=None, max_delta_step=None, max_depth=6,
               max_leaves=None, min_child_weight=None, missing=nan,
               monotone_constraints=None, multi_strategy=None, n_estimators=100,
               n_jobs=None, num_parallel_tree=None, objective='multi:softprob', ...)
```

Fig 1.12 Training XGBoost Model

### 3.2.4 LightGBM Model Training

It is another gradient boosting framework that is performance and speed optimized for processing large data. While xgboost expands trees depth-wise, LightGBM expands trees leaf-wise, and this enables it to concentrate on the locations of maximum error reduction. This significantly increases precision with a small computational overhead. Histogram-based learning is one of the striking aspects of LightGBM where continuous value is bucketed into discrete bins conserving memory as well as training time. Sparse feature handling support is also a part of this algorithm, and it makes this algorithm highly efficient for data sets with missing values. LightGBM is trained (Fig 1.13, Fig 1.14) to reduce loss functions such as log loss when dealing with classification and mean squared error when dealing with regression. Its hyperparameters, including number of leaves, learning rate, maximum depth, and min data in leaf, are tuned for better performance. LightGBM is particularly suitable for

real-time fault analysis and predictive analytics in network administration due to its fast training time and good accuracy.

#### Objective Function

Similar to XGBoost, LightGBM minimizes and the formula as shown in Equation (5).

$$L(\theta) = \sum_{i=1}^N \ell(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k) \quad \text{Equation (5)}$$

Here,  $\ell(y_i, \hat{y}_i)$  is a loss function measuring observed and predicted value differences and  $\Omega(f_k)$  is a regularization term making sure model complexity is controlled and overfitting is prevented.

```
[181] # Train LightGBM Classifier
lgb_model = lgb.LGBMClassifier(n_estimators=100, learning_rate=0.1, max_depth=6, random_state=42)
lgb_model.fit(X_train, y_train)
```

Fig 1.13 Training the LightGBM Classifier

```
[LightGBM] [warning] No further splits with positive gain, best gain is 0.
LGBMClassifier
LGBMClassifier(max_depth=6, random_state=42)
```

Fig 1.14 Training LightGBM Model

## 4. MODEL EVALUATION

Valuation of the precision of machine learning models is the most crucial part of ensuring the reliability and effectiveness of telecommunication network fault severity detection. The models that are tuned using CatBoost, Random Forest, XgBoost, and LightGBM must be tested (as depicted in Fig 1.15, Fig 1.16, Fig 1.17, Fig 1.18) using an appropriate metric so that their strength, predictability, and accuracy can be quantified. Evaluation techniques, evaluation measures, and comparison analysis are the most trending ones discussed here to find which model performs the best. Trained model value values evaluated in this are mentioned in Table1.

```
[145] # Evaluation Metrics
cat_accuracy = accuracy_score(y_test, y_pred_cat)
cat_precision = precision_score(y_test, y_pred_cat, average='weighted')
cat_recall = recall_score(y_test, y_pred_cat, average='weighted')
cat_f1 = f1_score(y_test, y_pred_cat, average='weighted')
```

Fig 1.15 Evaluation of CatBoost model

```
[137] # Evaluation Metrics
rf_accuracy = accuracy_score(y_test, y_pred_rf)
rf_precision = precision_score(y_test, y_pred_rf, average='weighted')
rf_recall = recall_score(y_test, y_pred_rf, average='weighted')
rf_f1 = f1_score(y_test, y_pred_rf, average='weighted')
```

Fig 1.16 Evaluation of Random Forest model

```
[176] # Evaluation Metrics
xgb_accuracy = accuracy_score(y_test, y_pred_xgb)
xgb_precision = precision_score(y_test, y_pred_xgb, average='weighted')
xgb_recall = recall_score(y_test, y_pred_xgb, average='weighted')
xgb_f1 = f1_score(y_test, y_pred_xgb, average='weighted')
```

Fig 1.17 Evaluation of XGBoost model



```

[183] # Evaluation Metrics
lgbm_accuracy = accuracy_score(y_test, y_pred_lgbm)
lgbm_precision = precision_score(y_test, y_pred_lgbm, average='weighted')
lgbm_recall = recall_score(y_test, y_pred_lgbm, average='weighted')
lgbm_f1 = f1_score(y_test, y_pred_lgbm, average='weighted')

```

Fig 1.18 Evaluation of LightGBM model

Table 1: Accuracy Scores of the Trained Models

Trained model	Accuracy score
CatBoost Classifier	70.24%
Random Forest Classifier	77.66%
XgBoost Classifier	76.18%
LightGBM Classifier	77.09%

#### 4.1 Model Comparison Table

The following Table 2 provides the comparison of four machine learning models—CatBoost, Random Forest, XgBoost, and LightGBM— on their F1 Score, Accuracy, Precision, and Recall to determine the significance of the network issues. The models have been trained and tested on Telstra network log data as shown in Fig 1.19.

MODEL COMPARISON TABLE					
	Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
0	Random Forest	77.66	77.36	77.66	77.48
1	CatBoost	70.24	68.93	70.24	68.68
2	XGBoost	76.18	75.47	76.18	75.41
3	LightGBM	77.09	76.46	77.09	76.42

Fig 1.19 Model Comparison Table

Table 2: Model Results for the algorithms

Model	Accuracy	Precision	Recall	F1 score
Random Forest Classifier	77.66%	77.36%	77.66%	77.48%
CatBoost Classifier	70.24%	68.93%	70.24%	68.68%
XgBoost classifier	76.18%	75.47%	76.18%	75.41%
LightGBM Classifier	77.09%	76.46%	77.09%	76.42%

## 5. CONCLUSION

Based on performance parameters, Random Forest Classifier's optimal performance is when its accuracy stands at 77.66%. It has the maximum best recall (77.66%) and F1 score (77.48%) of all classifiers and hence is most dependable for the given classification task. The model represents a harmonized trade-off among precision and recall, ensuring hence a solid capacity

to rightly class positive instances as such without significant false positives. To the side of Random Forest is LightGBM Classifier that has a accuracy of 77.09% and an F1 measure of 76.42%. While it is slower than Random Forest, LightGBM is computationally fast and therefore an extremely viable second choice, especially when training speed and scalability are the highest concerns. The XgBoost Classifier performs well too, at 76.18%, but not nearly as well as Random Forest and LightGBM. Accuracy and F1 score both indicate a good but not great performance, that it would need to be tuned again to match the best models. In contrast, the poorest performance on all the metrics is achieved using the CatBoost Classifier with 70.24% accuracy, 68.93% precision, and 68.68% F1 score. This indicates that CatBoost is not as accurate at class separation and can use extra hyperparameter tuning or another data set to better perform. Generally, the best model for this project is Random Forest, with LightGBM being a close second. XgBoost would also be a good option, and CatBoost is behind and perhaps not the best option unless it is properly tuned.

## 6. FUTURE ENHANCEMENT

Future developments for Network Fault Severity Detection can use deep learning architectures such as LSTMs, transformers, and mixed models to increase precision through the understanding of sequential log patterns. Automated hyperparameter tuning through Bayesian Optimization and Genetic Algorithms can be used to make models more efficient. Deployment of a real-time fault detection solution with streaming analytics (Kafka, Flink) and edge computing will support proactive problem solving. Enhancing model explainability using SHAP, LIME, and visualization dashboards (Grafana, Kibana) will enhance network engineers' decision-making. Running the system on cloud platforms (AWS, Azure) with containerization (Docker, Kubernetes) will maximize scalability, and federated learning can facilitate decentralized model training. Transfer learning and domain adaptation can make models applicable across various telecommunication providers with a minimum need for re-training. These developments will make the system an intelligent, scalable, and real-time fault prediction system for future telecom networks.

## REFERENCES

- [1] Saleem, Moeed, et al. "Machine learning for improved threat detection: lightGBM vs. CATBoost." *Journal of Computing & Biomedical Informatics* 7.01 (2024): 571-580.
- [2] [2] Tan, Ji Sheng, et al. "Predicting network faults using random forest and C5.0." *International Journal of Engineering & Technology* 7.2.14 (2018): 93-96.
- [3] [3] Boopathi, E., and V. Thiagarasu. "Edge detection in color images using RGB color model." *Int J Comput Appl* 180.9 (2018): 6-11.
- [4] [4] Ogar, Vincent Nsed, Sajjad Hussain, and Kelum AA Gamage. "Transmission line fault classification of multi-dataset using catboost classifier." *Signals* 3.3 (2022): 468-482.
- [5] [5] He, Shilin, et al. "A survey on automated log analysis for reliability engineering." *ACM Computing Surveys (CSUR)* 54.6 (2021): 1-37.
- [6] [6] Saied, Mohamed, Shawkat Guirguis, and Magda Madbouly. "A comparative study of using boosting-based machine learning algorithms for IoT network intrusion detection." *International Journal of Computational Intelligence Systems* 16.1 (2023): 177.
- [7] [7] Okokpujie, Kennedy, et al. "Development of a Machine Learning Based Fault Detection Model for Received Signal Level in Telecommunication Enterprise Infrastructure." *International Journal of Safety & Security Engineering* 14.3 (2024).
- [8] [8] Okey, Ogobuchi Daniel, et al. "BoostedEnML: Efficient technique for detecting cyberattacks in IoT systems using boosted ensemble machine learning." *Sensors* 22.19 (2022): 7409.
- [9] [9] Li, Shuai, et al. "Enhancing LightGBM for industrial fault warning: An innovative hybrid algorithm." *Processes* 12.1 (2024): 221.
- [10] [10] Lu, Zhiye, Lishu Wang, and Panbao Wang. "Microgrid Fault Detection Method Based on Lightweight Gradient Boosting Machine-Neural Network Combined Modeling." *Energies* 17.11 (2024): 2699.
- [11] [11] Rajalakshmi, D., et al. "Advancing Fault Detection Efficiency in Wireless Power Transmission with Light GBM for Real-Time Detection Enhancement." *International Research Journal of Multidisciplinary Technovation* 6.4 (2024): 54-68.
- [12] [12] Kumar, E. Boopathi, and M. Sundaresan. "Edge detection using trapezoidal membership function based on fuzzy's mamdani inference system." *2014 International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2014.
- [13] [13] Prakash, G., P. Logapriya, and A. Sowmiya. "Smart Parking System Using Arduino and

Sensors." NATURALISTA CAMPANO 28 (2024): 2903-2911.

- [14] [14] Kumar, E. Boopathi, and V. Thiagarasu. "Comparison and Evaluation of Edge Detection using Fuzzy Membership Functions." International Journal on Future Revolution in Computer Science & Communication Engineering (IJFRCSCE), ISSN (2017): 2454-4248.
- [15] [15] Leevy, Joffrey L., et al. "Detecting cybersecurity attacks using different network features with lightgbm and xgboost learners." 2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI). IEEE, 2020.
-