Root Cause Analysis Bot using Machine Learning Techniques

Tapan Behera [1,1,1] and Kumud Tripathi [2]

[1]Forrester Research
[2]Affiliation not available

October 31, 2023

Abstract

In this world of quick delivery of quality products, DevOps test automation plays a vital role. This triggers the automation build to test whether the product quality is a good fit for the release or not? When multiple test cases of a test suite got failed, it takes lots of time for the developer

and analyst to analyze the error logs of each test case and do the root cause analysis (RCA). For one test case if RCA takes around 20 minutes then for 100 test cases it will be going to take around 33 hours and it's nothing but 4 Analysts' full-day work, then it's a question on ROI(Return on Investment) ? To solve this time-consuming process, we are introducing Root Cause Analysis Bot (RCA Bot) in this paper. This bot will analyze the failure logs, error message, descriptions, error codes and apply the machine learning (ML) techniques to predicts the root cause of the failure with a percentage of the prediction accuracy.

Highlights

1. Use RCA Bot to determine the root cause of an Enterprise System, Application, or Test Case failure.

2. The RCA Bot was developed using Machine Learning Techniques to predict with maximum accuracy.

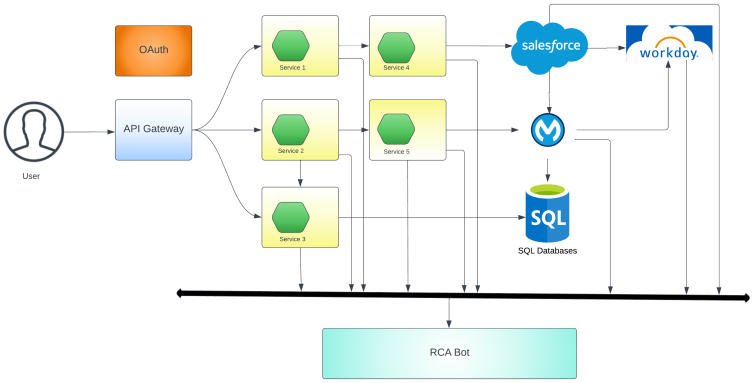
3. It is a generic bot that works with any kind of distributed application, regardless of the environment.

4. A time-saver for the manual process of finding the root cause, find the cause in just a few seconds.

5. Highlight error categories with custom labels.

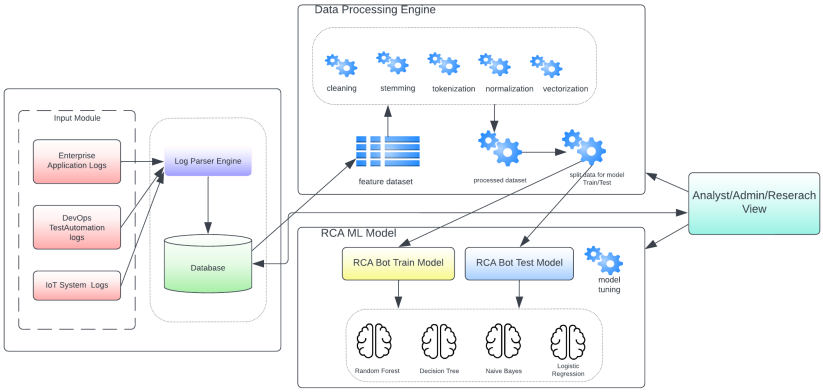
RCA Bot In Eneterprise System

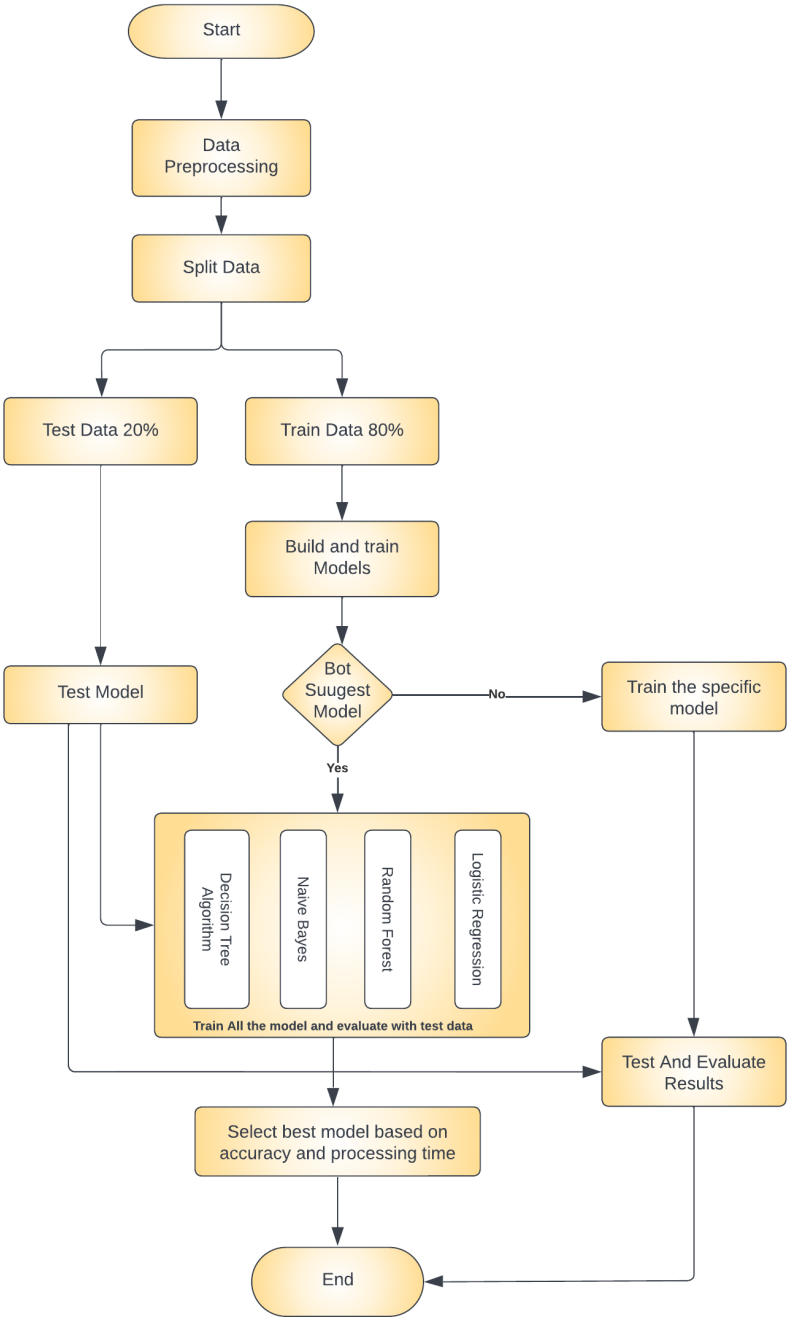
Tapan Behera | Nov 5, 2022

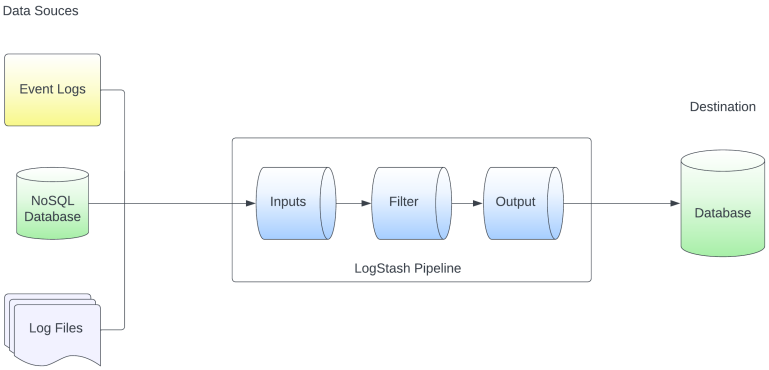


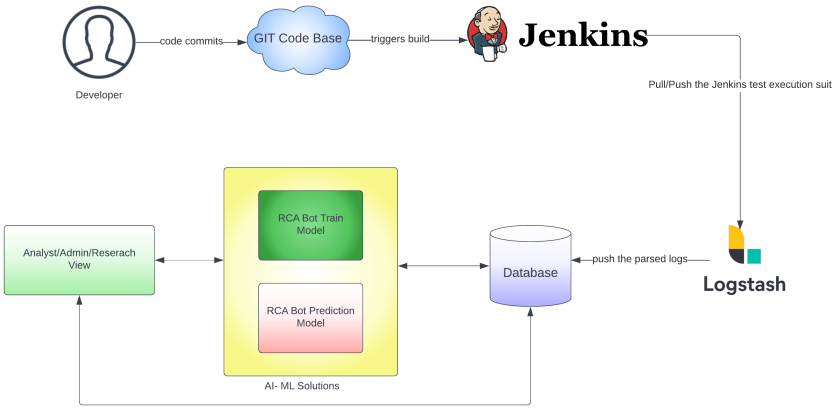
RCA BoT Architecture Diagram

Tapan Behera









Root Cause Analysis Bot using Machine Learning Techniques

Tapan Kumar Behera[1] and Kumud Tripathi[2]

[1]Technical Architect, Cambridge, MA, USA
[2]Department of CSE, IIIT Lucknow, India
Corresponding author: Tapan Kumar Behera, E-mail: tkbehera.usa@gmail.com

March 2023

Abstract

In this world of quick delivery of quality products, DevOps test automation plays a vital role. This triggers the automation build to test whether the product quality is a good fit for the release or not? When multiple test cases of a test suite got failed, it takes lots of time for the developer and analyst to analyze the error logs of each test case and do the root cause analysis (RCA). Second, Enterprises with complex distributed applications designed with multi-tier architecture systems and with a lot of third-party applications communicating with each other, cannot prevent failures from happening. As a result, finding the cause of failures becomes a real challenge, as it requires cross-team cooperation, log analysis, and many more. As a result, a lot of investigation was required in order to perform a Root Cause Analysis and becomes part of the daily routine, hindering the organization's productivity. To solve this time-consuming process, we are introducing the Root Cause Analysis Bot (RCA Bot) in this paper. This bot will analyze the failure logs, error messages, errorClasses, descriptions, and error codes and apply machine learning (ML) techniques to predict the root cause of the failure with a percentage of the prediction accuracy.

Index terms— Machine learning, Artificial Intelligence, Natural Language Processing, Enterprise Application, Microservice, DevOps, AutoML, IoT, Bots

1 Introduction

Since a product's complexity and a short time to market make it difficult to maintain its quality, it is also important to keep the cost low to remain competitive in the market. Software developers must consider many aspects when developing any product, which makes it essential to take this into consideration. Despite following all guidelines, software defects still occur. Many factors can lead to defects and failure of the product, such as a human factor, lack of communication, poor design logic, unrealistic development deadlines, untrained testers, and poor coding practices (TechArcis, 2021). Furthermore, the recurrence of the same defects during the development cycle of new or enhanced versions of the product makes the process more challenging. Corrective actions must be taken to prevent the recurrence of the defects by conducting a root cause analysis.

Second, With complex distributed applications and multiple third-party applications communicating with each other, enterprises cannot prevent failures, such as in Master Data Management System, there are lots of applications communicating with each other for the golden record of the table from the Master Database (Behera, Tapan, 2023). Finding the cause of failures becomes a real challenge, as it requires cross-team collaboration, log analysis, and many more, and becomes part of the daily routine, hampering productivity.

It is the Internet of Things that turns a city into a smart city. IoT is a system of connected devices, such as sensors, vehicles, and home appliances. Using IoT devices, data can be transferred over the internet without the need for human intervention. As part of an IoT smart city, edge computing is also utilized, creating an edge computing smart city. With edge computing, IoT devices can save bandwidth and reduce latency by bringing compute power closer to them. Despite all the benefits, IoT still faces many challenges, such as security, connectivity failures, problems registering and discovering devices, problems attaching a device to a gateway through the cloud, and failures during message exchanges. With the help of RCA Bot, these numerous issues can be traced and immediately resolved. It is possible for this RCA Bot to trace and suggest possible causes for failures across all IoT System logs.

As a solution to this problem, we have developed the Root Cause Analysis (RCA) Bot, a time-saving tool that automates the process for you. With Artificial Intelligence (AI) and Machine Learning (ML) techniques, Root Cause Analysis Bot identifies the root cause of problems and predicts the reason for failure with a high degree of accuracy (prediction accuracy).

Root Cause Analysis bot is based on a multi-class classification model, with custom labels Table 1 for broad categorization of the issues. As it's a custom label if there is any new category of issues coming then it can be added to this list and retrain the Machine Learning model.

There are many research on Root Cause Analysis in different fields like Health Care, the Mechanical or Production Industry, and Software. These studies have addressed about what, how and why something happened; trying to find and solve the data quality problems (Qiuping Ma and Hongyan Li and Anders Thorstenson, 2021). In early bot detection in the Internet of Things (Alex, Anderson and Michele,2022) aims to identify network-infected devices to prevent Cyber attacks. Literature addresses botnet detection by modelling the behaviour of malware spread, the classification of malicious traffic, and the analysis of traffic anomalies (Alex, Anderson and Michele,2022).

It was found in the article (Urli and Zhongxing and Seinturier and Monperrus, 2018) that the authors had found the issues resulting from the failures of the Travis CI builds for the test cases. Input to a pipeline contained three stages was provided by the CI builds: A first stage called CI Build Analysis collected and analyzed the CI builds from GitHub projects; a second stage referred to as Bug Reproduction, which is intended to reproduce the build failures that have occurred on the Continuous Integration (CI) platform; a third stage, referred to as Patch Synthesis, that searches for patches based on the failure reproduction information. In the Repairnator bot, bugs are reproduced and repair tools are run against each reproduced bug so that they can be fixed. A patch will be reported to the

Table 1: Root Cause Custom Labels

S. No.	Root Cause Category
1	Data Issue
2	Product Bug
3	Environment Issue
4	Automation Script Error
5	Services Time out

developers if a Repairnator bot finds it. Here, it uses Java programs to compare a number of possible failures and errors and suggest patches that can be used to fix those failures (Urli and Zhongxing and Seinturier and Monperrus, 2018).

In this paper, we will examine how the RCA Bot performs a Root Cause Analysis. Currently, we are evaluating RCA Bot in DevOps Test Automation Testing. The Test Automation suite contains the Tescases which are a) High Risk – Business Critical test cases b) Test cases that are repeatedly executed c) Test Cases that are very tedious or difficult to perform manually d) Test Cases which are time-consuming. As the Test Automation suite runs, all the test cases are executed, resulting in a validation of the functionality and the generation of test case logs. We use the DevOps Test Automation failed test cases in the Root Cause Failure Analysis in conjunction with error logs, error codes, and error messages. Moreover, finding the failure causes using the Supervised Machine Learning technique. We will be able to use RCA Bot in the future to solve problems related to blockchain supply chains (Behera and Panda, 2022).

In the current work, our aim is to develop a root cause analysis bot to accurately analyse the failure logs, error messages, ExceptionClass, error descriptions and error codes. We have implemented various text processing steps such as stemming, lemmatization, normalization, and vectorization. In vectorization, the text data got transformed into a vector. This work includes the Term Frequency-Inverse Document Frequency (TF-IDF) for text-to-vector conversion. We have explored the traditional supervised machine learning algorithms for building the RCA bot. Based on labelled data, supervised ML recommends how a new record should be classified. We will consider the following four supervised machine learning algorithms: logistic regression (LoR), naive Bayes (NB), decision trees (DT), and random forests (RF) among others.

The workflow of the paper is as follows. Section 2 describes the overview of RCA Bot. The dataset used in this work is discussed in Section 3. The RCA bot workflow is described in Section 4. The detailed discussion about machine learning techniques is given in Section 5. Experimental results and performance analysis of the developed bot is discussed in Section 6. The conclusion of this paper as well as the future directions of this work have been presented in Section 7.

2 Overview of RCA Bot

Root Cause Analysis Bot has 6 major components, the first component is Input Data Sources, where the data is feed to the RCA Bot. The second component is the Log Parser Engine, which reads all the input data sources and parses the server log unstructured data into a structured format data and stores it in the database. The third component is the database which stores training and testing datasets, and prediction results, model configuration information. The fourth component is the Data Processing Engine to pre-process the data and feeds it to the ML component. The fifth component is the ML Model, which does the training and testing of the model on the processed vector dataset. Training model which learns about the types of errors, reads the logs and understands the root cause of the failure and has been tested with a sample prediction. The Testing model is the prediction model which predicts the real failures of the system by going through the error descriptions and error logs. And predicts the probable root cause of the failure with a confidence level. The sixth component is the View to show the prediction results and selects the model to deploy and all analyst-related activity can perform through the UI module. Will discuss in detail more on each component in the Architecture section.

2.1 Architecture of RCA Bot

Figure 1 illustrates the architecture of the RCA Bot, and we will discuss each module in more detail.

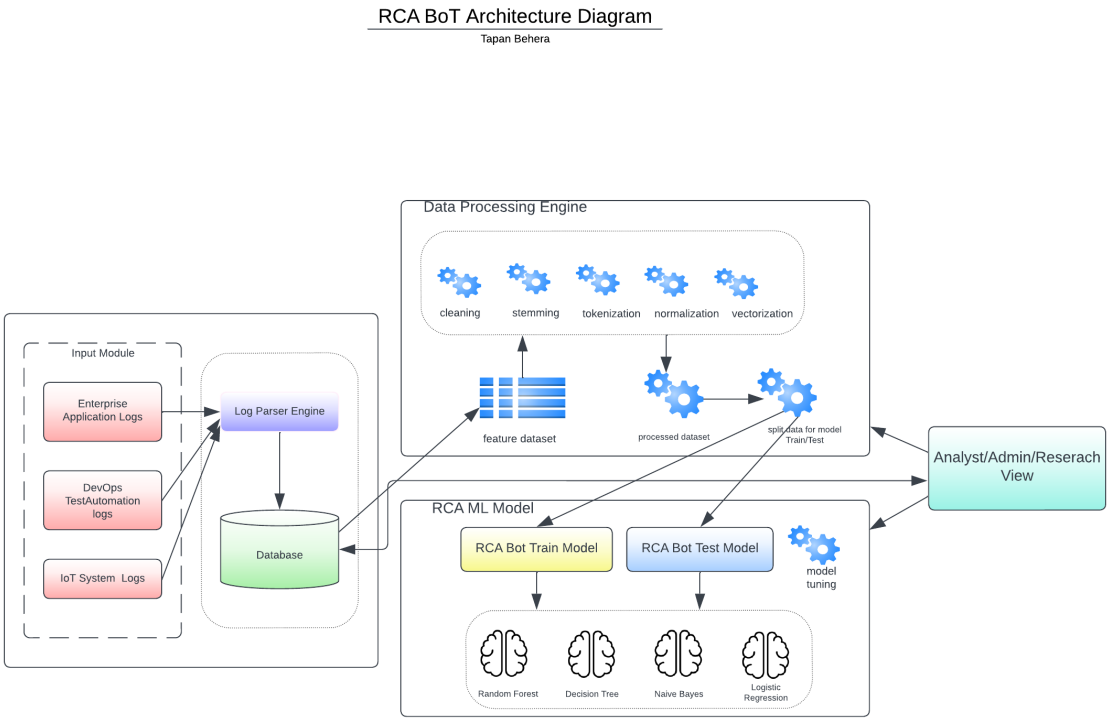


Figure 1: Architecture of RCA bot.

1. **Input Module:** As RCA Bot is a generic bot which can work on any type of application server logs to find the root cause of the failures in that system. It is a collection of input data source, log parser engine and database. Each of them are explained below.

   - **Input Data Sources:** As shown in the Figure 1 the input sources are

     a) Enterprise Logs: The Enterprise System contains a number of applications and microservices that communicate with one another. These applications perform their operations and generate logs on application/web servers.

     b) DevOps Test Automation Logs: During the DevOps build, the test automation runs the test cases to check the quality of the products, which generates the test cases logs.

     c) IoT System Logs: The Internet of Things (IoT) refers to a network of several sensors, cars, appliances, and other devices that are connected to each other. During the operation of every IoT device, the logs are generated, and when there is a failure in one device, it will cause a ripple effect in all the systems. This RCA bot helps to find the root cause of the failure in the IoT System.

   - **Log Parser Engine:** The Log Parser Engine reads all the multiple data sources and parses them. There are many log parsers available in the market; in this RCA Bot we are using the Logstash parser. This is an open-source data collection engine that includes the capability of real-time pipeline generation (Logstash Parser, 2022). As shown in the Figure 2 a pipeline in Logstash has two required elements, input and output, and one optional element, filter. Data is gathered from a source by the input plugins, modified by the filter plugins according to your specifications, and written by the output plugins according to your specifications. Among the most important features of Logstash is its ability to collect and aggregate data from a variety of sources. The Logstash service is capable of gathering data from a wide variety of platforms and services through over 50 plugins. Inputs include files, beats, Syslog, stdin, UDP, TCP, HTTP, heartbeats, as well as specific services like Azure event hub, Apache Kafka, AWS Kinesis, Salesforce, and SQLite.

   - **Database:** In the RCA Bot the data storage we used the Mongo database. It has the capability of storing Structured and UN-Structure data (Lucia de Espona, Ela, 2023). After the Logstash parsing is completed the filtered data is transferred to Mongo Db. The Machine learning Model reads and writes the data from Mongo. This database can be present on any remote server and communicate with other components; it does not necessarily have to be on the same machine as the ML model.

2. **Data Processing Engine:** Data Processing Engines fetch and process raw data from datasets using text processing pipelines. Datasets of server logs contain error descriptions, error classes, error codes, and exceptions. It is necessary to remove a lot of irrelevant information from the data, including stop words, lowercase words, white spaces, and punctuation. As shown in Figure 6, the steps involved in text processing are as follows.

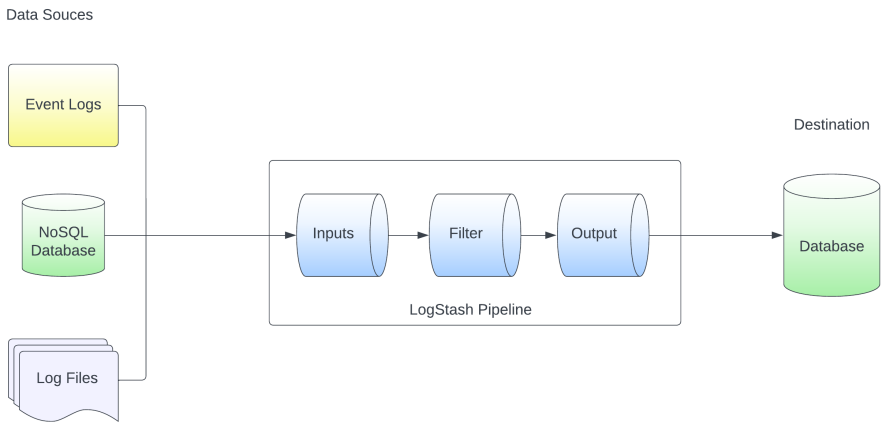


Figure 2: Log Parser Engine.

3. **RCA ML Model:** This is the core component of RCA Bot, it has two major components, one is the RCA Train model and the second one is the RCA prediction model. This bot is currently developed in the Apache Spark MLib, Scala, and running in the Spark Job Server(Karau and Rachel, 2017). It has machine learning capabilities and used classification algorithms such as Naıve Bayes, Random Forest, Decision Tree, and Logistic Regression. The intelligent component of this bot is, it auto-suggests the best algorithm to use to predict the root cause of the issue with accuracy and processing time. Moreover, it also has manual configuration and selection features to override the Bot Suggestion model.

4. **Analyst View:** The view is the UI component, where an Analyst, Admin uses it to see the results of the prediction, configure the model, and select the model to deploy into production. This is the optional component in the RCA Bot, the end customer can use their own UI module to fetch the prediction results from the database and provide the necessary configuration input to model through the database.

2.2 Use of RCA Bot in Test Automation

The Root Cause Analysis Bot is a generic bot to predict failure causes and can be used across any domain and N-tier enterprise applications. This paper will evaluate the RCA Bot in DevOps test automation logs, where it receives the failed automation test cases and logs and will find out the Root Cause of the Failures, using the RCA Bot. The components in the DevOps Test Automation are as follows.

1. **GIT Hub:** GIT Hub is a source code repository, a version control using Git. A number of features are supported by it, including access control, bug tracking, task management, and continuous integration. And its easily integrated with any build pipeline. The step-by-step integration of GitHub with Jenkins pipeline described in this blog(Anton Angelov, 2022).

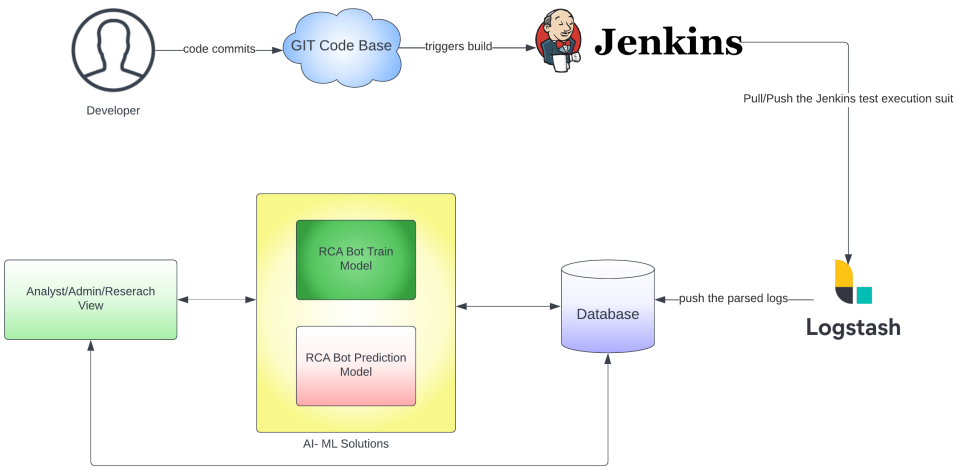


Figure 3: RCA Bot in DevOps Test-Automation

2. **Jenkins Pipeline:** This is an open-source application server that is used to build software artefacts, test them, and deploy them to servers as part of Continuous Integration and Continuous Delivery (CI/CD). The Jenkins pipeline auto triggers the build and runs the test automation whenever there is a code commit or a scheduled event triggers. This pulls the latest code from the GIT hub repository and performs the build. The developer needs to configure the Jenkins pipeline for all the steps of pre and post-build execution.

3. **Logstash Parser:** Logstash parser reads the Automation test suit server logs, which is an Unstructured log data and to parse the log, in the Logstash "filters" needs to define. Which contains the rules, patterns of an error log, error code, test case name, description, etc. The logstash pipeline needs to define the input source and the target destination location where wants to send parsed data. It has the capability of remote connection and writing the output data into the cloud or on-prem database servers.

4. **Database:** After the Logstash parsing is completed the filtered data is transferred into MongoDB database. The Machine learning Model reads and writes the training and testing dataset from the MongoDB database. And also used to store the model configuration and Root Cause prediction results.

5. **RCA AI-ML Component:** The RCA Bot machine learning component trained with the test automation datasets and cross-validated all models to select the most accurate model with the AutoML bot suggested model feature. Whenever a new test automation build occurs through Jenkins, the selected model begins to predict the root cause of the failure.

3 Dataset

Currently, the dataset is derived from the log files of DevOps Test Automation test cases. The dataset contains the testCaseName, ReleaseName, buildId, Error Description, Exception. As part of the RCA Bot training and testing, we are using 5 test automation builds and 400 datasets in total. The test automation dataset was randomly divided into 80 and 20 percent for training and testing machine learning models. For training the machine learning models, we used 80% of the input data (320 data points) and tested 20% of the input data (80 data points) to evaluate the performance of these machine learning models.

As shown in the Figure 4, the error description from the server log provides very detailed information about the failures. For example, this error message provides information about what flow got failed, and the line number of the flow causing this failure. As a result, this error occurred when validating the data before it is going to be inserted into the table. Based on the test case, functionality, and validation, it is classified as a data issue in the actual root cause.

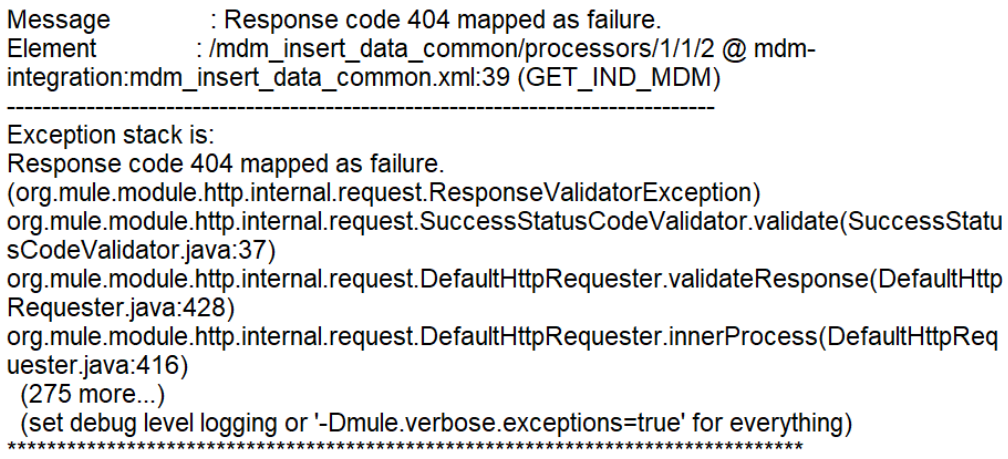


Figure 4: Error Description of failure testcase

LogSlno	buildId	suiteName	projectId	releaseName	testCaseName	environment	startTime	endTime	durationInMS	exceptionClass	errorDesc	actualRootCause
1	101	MuleMDMRegressio	MDMProd01	AdamRelease	testRestCall	Prod	2023-02-14: 1	2023-02-14:	400MS	INTERNAL_SERVER_ERROR	Could not dispatch soap message using the [HTTP_Request_configuration] HTTP configuration Caused by: java.util.concurrent.ExecutionException:	Environment Issue
2	101	MuleMDMRegressio	MDMProd01	AdamRelease	testUpdateIndvPref	Prod	2023-02-14: 1	2023-02-14:	600MS	ResponseValidatorExceptio	Message : Response code 404 mapped as failure. Element : /mdm_insert_data_common/processors/1/1/2 @ mdm-integration:mdm_insert_data_common.xml:39 (GET_IND_MDM)	Data Issue
3	101	MuleMDMRegressio	MDMProd01	AdamRelease	testUpdateUserData	Prod	2023-02-14: 1	2023-02-14:	700MS	MessagingException	Exception stack is: Failed to invoke sendMessage. (org.mule.api.MessagingException)	Event Failure
4	101	MuleMDMRegressio	MDMProd01	AdamRelease	testInsertProfileLogs	Prod	2023-02-14: 1	2023-02-14:	500MS	SQLException	Caused by: java.sql.SQLException: Cannot get connection for URL jdbc:postgresql://us-wset-1.rds.amazonaws.com:1111/production?password=password&user=tapan : The	Environment Issue
5	101	MuleMDMRegressio	MDMProd01	AdamRelease	testSendMsgtoSF	Prod	2023-02-14: 1	2023-02-14:	500MS	NumberFormatException	Message : Error sending HTTP request. Message payload is of type: byte[] Code : MULE_ERROR-29999 Exception stack is: 1. null (java.lang.NumberFormatException) org.glassfish.grizzly.http.util.Ascii:271 (null)	Product Bug
6	101	MuleMDMRegressio	MDMProd01	AdamRelease	testGetUserDetails	Prod	2023-02-14: 1	2023-02-14:	300MS	WebDriverException	<![CDATA[org.openqa.selenium.WebDriverException: unknown error: cannot create temp dir for user data dir\n (Driver info: chromedriver=2.40.565498 (ea082db3280dd6843ebfb08a625e3eb905c4f5ab),platform=Windows NT 10.0.14393 x86_64) (WARNING: The server did not provide any stacktrace information)\nCommand duration or	Automation Script Error

Figure 5: Sample of dataset to train the ML models

4 RCA Bot Workflow

The Figure 3 illustrates how the RCA bot determines the root cause of a failure of a test automation script. This RCA Bot uses continuous-integration build tests as its primary input. The build is triggered whenever the developer commits code to the GitHub repository or in accordance with a schedule or manual trigger. The Jenkins Pipeline generates automation test results once the build has been completed, which include both success and failure test case logs.

There are two options for fetching the Jenkins logs post-build, either Jenkins can push the build results to the target location through the FTP connector or Jenkins provides services to pull the results from the Jenkins Hub. After the results are pulled to the location where Logstash parser is running, the Logstash parser begins reading those test cases, goes through the filter setup, and retrieves the data. As shown in the Figure 5, we are retrieving the name of the test case, the release name, the build ID, the error description, the exception class, the execution date, and the status of the test case.

As of now, we have discussed the process of importing the data into the database. The Machine Learning components will be introduced at this point. We will discuss the data that the bot will read, the "error description" contains the complete error log for the failed test case, which includes numerous stop words, special symbols, package details, and error information about the error. To begin with, the data will be processed through text processing pipelines and then cleaned, stop words will be removed, lowercased, white spaces will be omitted, and punctuation will be removed. As shown in Figure 6, the text processing steps are described below.

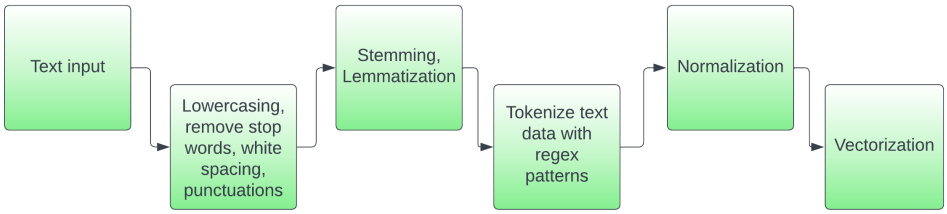


Figure 6: Text processing pipelines.

- **Stemming:** Consolidating a word's variant forms into a single stem is the process of stemming. As an example, the words: "Connect", "Connection", and "Connected" can all be represented by the word "Connect". As a result of posing a query with the word Connect and Connected, the procedure relies on the assumption that documents containing the words Connect and Connected are of interest(Mardiana, Adji and Hidayah, 2016).

- **Lemmatization:**For linear text classification systems, lemmatization has traditionally been a standard preprocessing technique used to prepare the text for classification. Due to the rare occurrence of different inflected forms of the same lemma, lemmatization is primarily intended to reduce sparsity(Jose, Mohammad, 2018). Therefore, a lemmatization algorithm determines that better is derived from good, thereby indicating the lemme is good.

- **Tokenization:** The process of tokenization involves breaking down the text into individual terms usually words (Karau and Rachel, 2017). Using RegexTokenizer, you can perform more advanced tokenization using regular expressions (regex). A delimiter called "pattern" (regex, default: "//s+") is used to split the input text. Instead of splitting gaps, users can set the parameter "gaps" to false and find all matching occurrences as the tokenization result, rather than splitting gaps.

- **Vectorization**: It transforms the text data into vector indices, applying TF-IDF techniques. The TF-IDF stands for Term Frequency- Inverse Document Frequency is a text processing step. Using this technique it tries to find the importance or relevance of the word in the document, how often it appears.

5 Machine Learning techniques

Machine learning evaluates data-driven approaches for processing massive datasets and captures the underlying pattern. Precise data-intensive analyses are provided by ML, which concentrates on automatically optimising a model for forecasting and making decisions. ML can be broadly divided into three categories: supervised, unsupervised, and reinforcement learning. The current data analysis uses supervised machine learning (ML) methods. As a result, 4 well-known ML approaches are used in this

study, such as Decision Tree (DT), Random Forest (RF), Naive Bayes (NB), and Logistic Regression (LoR) for classifying failure in 5 classes as shown in Table 1.

Decision Tree A decision tree is used to create tree-like models for classification or regression. As a dataset is segmented into smaller and smaller sections, an associated decision tree is incrementally developed while a decision tree is incrementally developed (Charbuty, Abdulazeez, 2021). It is the result of this process that we get a tree containing leaf nodes and decision nodes. A decision node is made up of two or more nodes, each of which represents a value for the characteristic that is being considered by the decision maker. In this case, a leaf represents the decision on the numerical target that has been made. The root node is the topmost decision node in a tree and corresponds to the best predictor. As a result of the various factors contributing to the predictions, we can evaluate the contributions of the factors via DT results, which can also be viewed visually.

Random forest Random Forest is a collection of decision trees that are used to make decisions. Basically, a Random Forest is a set of trees that were built in a specific "random" manner that consists of several different types of trees. As a result, each tree is constructed from a unique sample of rows (Aurelien Géron,2017)(Svetnik, Liaw, Tong, Culberson, Christopher, Sheridan and Feuston, Bradley, 2003)(Belgiu, Dragut, 2016). At each node, a separate sample of characteristics is selected to be split. In each of the trees, there is a unique prediction that each tree makes. The average of these predicted values is then used to produce a single result by combining all the predictions. The averaged data of a Random Forest outperforms the data from a single Decision Tree since it uses more variables. As a result, the accuracy of the model will be improved and overfitting will be reduced. In the Random Forest Regression, the average of the estimates provided by the trees in the forest constitutes a prediction from the random forest regression model.

Naive Bayes NBC, also known as Naive Bayes Classifier (NBC), is a simple and efficient classification algorithm, which is one of the most effective and straightforward algorithms in machine learning, which can make accurate predictions in a short period of time. A Nave Bayes algorithm is composed of two words, Naive and Bayes, which are the two words that make up the Nave Bayes algorithm.

Naïve: This is a text mining method that assumes that the presence of one feature has no relationship to the absence of other features, hence it has been referred to as a naive method. Consequently, each feature of the system works independently of the others in order to achieve the desired results.

*Bayes:*Since this principle relies on the Bayes' Theorem concept, it is called the Bayes principle as it is based upon the Bayes' Theorem concept.

It is well known that the naive Bayes classifier is optimal when attributes within a given class are independent from one another. The absence of missing values does not affect the robustness and classification of the NBC algorithm (Aurelien Géron, 2017) (Svetnik, Liaw, Tong, Culberson, Christopher, Sheridan and Feuston, Bradley, 2003) (Tripathi, Rao,2018).

Logistic Regression The logistic regression method (LoR) contains a regression line that is fitted to the data instead of fitting a function, which is shaped like an S, that anticipates two maximum values (0 and 1). Another name for the S-form curve is the logistic function, which is also known as

the sigmoid function. There are several independent variables used in the LoR model that are used to predict a categorical dependent variable (Svetnik, Liaw, Tong, Culberson, Christopher, Sheridan and Feuston, Bradley, 2003)(Breiman, Friedman, Olshen, Stone,1984) (Menard,2002). Logistic regression is one of the most powerful and versatile machine learning tools available since it is capable of assigning probabilities and classifying new data based on continuous as well as discrete datasets.

5.1 Selection of features

We use feature selection (FS) to eliminate non-attributes from datasets by searching through all of their attributes. To maximize the value of the subset of attributes that are informative or redundant, select the subset of attributes that maximizes the value of those attributes. The feature selection reduces noise-driven decisions as well as reduces overfitting. By removing misleading variables, accuracy can be improved and training time can be reduced. We are selecting error description as the feature that contains the error codes and error classes in the log. In order to analyze the reasons for failure, this is a more relevant feature.

The attribute evaluator and the filter ranker search method (Hossein Nematzadeh, 2022) were applied when building the algorithm. With the filter ranker method, each feature in the dataset was assigned a relevance score by an attribute evaluator. A descending order of relevance scores was then applied to the attributes. Modeling was performed to select a high-scoring feature and eliminate a low-scoring feature. Several attributes were selected and tabulated as a result.

5.2 Modelling approach

Using the SPARK ML and Scala programming, 4 ML algorithms were implemented in this work. The experimental datasets were collected and randomly divided into training and testing datasets, with 80% of the input data being trained using 4 ML models and the remaining 20% being used to test the performance of these ML models as shown in Figure add figure.

In this case, k-fold cross-validation is used in the training phase to reduce the overfitting, with k being set at 10. Dynamic validation subsets are used in cross-validation to objectively determine the best algorithmic parameters. As illustrated in Figure 5, the initial 80% training dataset is split into k subsets, and the selected 4 ML model is trained k times separately. By choosing one subset for validation and the remaining k – 1 subsets for training, the ideal values for the hyper parameters are discovered over those k–folds. Cross-validation score comparison is used to analyse the results. Due to the fact that the test set was not presented to the algorithms during the training phase, it is used for external validation. Grid search is used to tune the hyper-parameters of the ML algorithms during the training phase (Andrea Ialenti, 2020). A detailed investigation over a manually defined search subspace was carried out to determine the best set of hyper-parameters for each algorithm using cross-validation metrics and the estimator's score technique. The optimised values of the studied hyper-parameter space for each algorithm are summarized in Table 2. In the training phase, the

Table 2: Hyper-parameter tuning using Grid Search for optimal performance

Serial No.	ML Model	Hyper-parameter	Optimal
1	Logistic Regression - LoR	C Penalty	1.0 l1
2	Random Forest - RF	max_depth bootstrap max_features min_samples_split n_estimators criterion	5 True auto 3 25 gini
3	Naive Bayes - NB	alpha class_prior fit_prior	10 None True
4	Decision Tree - DT	min_samples_split max_depth criterion	5 7 entropy

optimal hyper-parameters were used, and the trained ML models were then used to predict the output for the test set.

5.3 Training and Testing of RCA ML Model

The RCA Bot is an auto-ML based multi-class classification model that predicts the category of issues. A description of the category of issues in RCA Bot can be found in Table 1. There are a few steps that need to be followed to train and test the RCA ML models, as shown in Figure 7 of the Data flow diagram below.

Steps followed for training the model:

1. From the database table, read the error description, error logs, and test case name of the failed test cases.

2. As a next step, we process the data through pre-processing pipelines, which remove the stop words, lowercase, remove white spaces, and punctuation based on the parameters provided Figure 6.

3. After the data is ready, the next step is to train the model, so before training, we must split the data for testing also. In this case, we will use 80% percent of the data for training and 20% percent for testing.

4. As mentioned in the flow diagram, we are considering one parameter from the Analysts input as "BotSuggestModel". If the parameter is opt-in and the value is true, then the train data will be run through all of the machine learning algorithms, including Logistic Regression, Random Forest, Naive Bayes, and Decision Tree Algorithms. Additionally, this train model should be saved to the physical drive so that real-time predictions can be made.

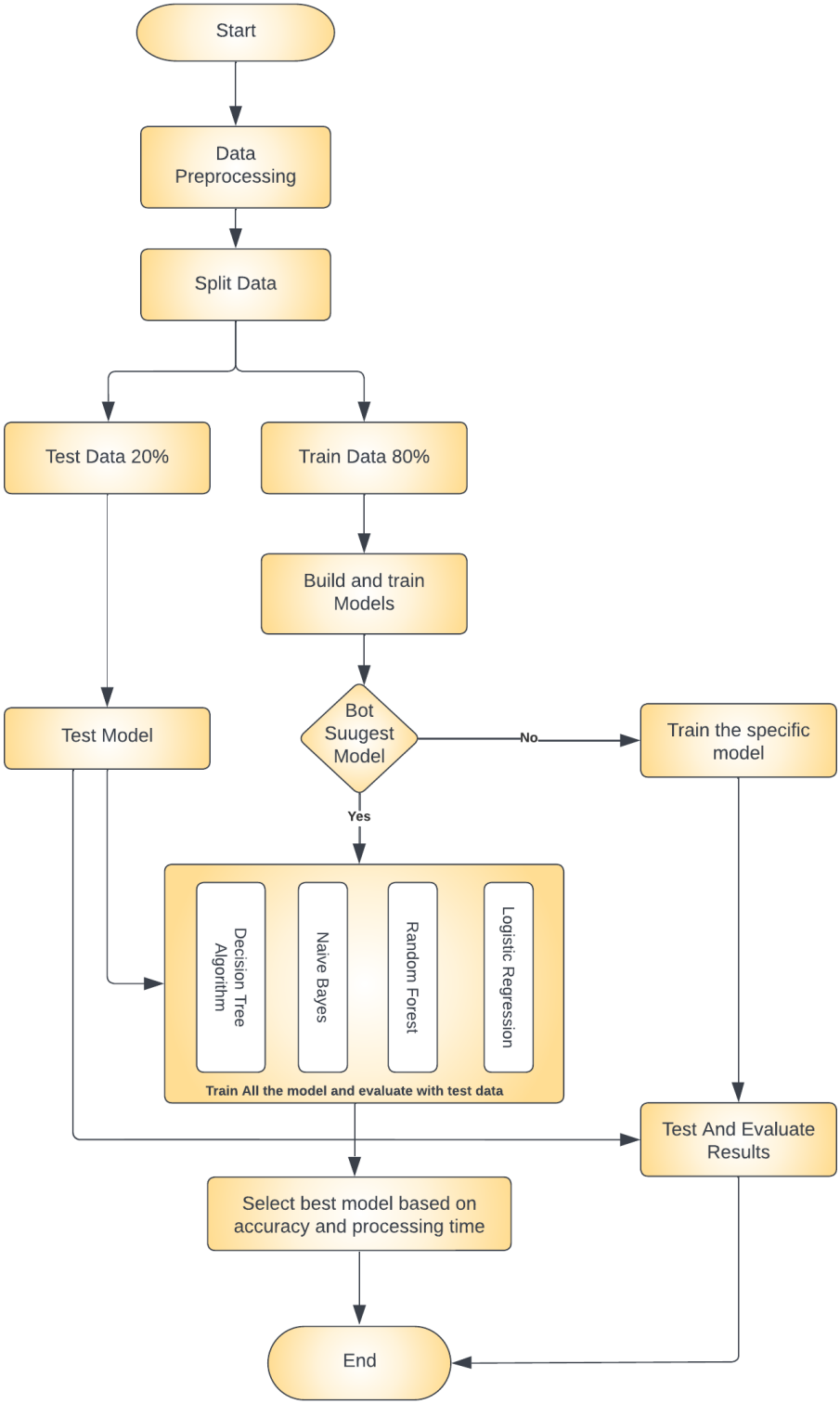


Figure 7: Flow chart shows the RCA machine learning model.

5. If the Analyst/User wishes to select a specific model to train with custom parameters and check its accuracy in predicting failures in comparison to the bot-suggested model, then the "BotSuggestModel" flag is false.

Steps followed for testing the model:

1. Next, we will evaluate the accuracy and performance of the models based on the remaining 20% of data.

2. In order to test the model against the data, the saved model will be loaded from the physical location.

3. Afterwards, we will start recording the processing time of each ML algorithm and measure the model's accuracy in predicting test data.

   Now it is time for Bot to suggest the most accurate and efficient algorithm based on accuracy and processing time. This capability is referred to as Auto-ML in this Root Cause Analysis task.

4. The Bot suggests the model with the highest accuracy by default. The processing time is considered when there are two or more models with the same accuracy, and the model with the lowest processing time is chosen.

6 Results and Discussion

In order to evaluate the performance of the model, k-fold cross-validation was used. Out of total dataset, 80% are used to train the model, and remaining 20% are used to test the ML models efficiency. Therefore, cross-validation technique is applied on 80% of the data which is used for training the model. Using cross-validation, a predictive model is validated on new data sets to see if it performs well. To perform cross-validation, the data is further divided into two sets, training data and validation data (Stojancho, 2020). The training set is used to build the model, and the validation set is used to validate the model by computing prediction errors.

The repeated k-fold cross-validation method was implemented by randomly splitting the data. The dataset is divided into k = 10 equal sets. Afterwards, we evaluated the developed model using the test dataset (20%) to make sure that it was correct. As a result of the unseen observations, the validity and accuracy of the model can be evaluated. Using the results as a guide, we quantified the prediction error as a mean squared difference between the observed and the predicted outcome values calculated on the basis of the data.

To develop the RCA bot the considered ML models such as decision tree, random forest, naive Bayes, and logistic regression. A comparison of the classification accuracy of the RCA Bot developed using different machine learning models can be seen in Table 3. The first column "ML Model" of the table represents the different machine learning models that were used to build the RCA bot. The second column represents the model ID, and the third column shows the classification accuracy

Table 3: The classification performance of RCA Bot developed using Machine Learning models.

ML Model	Model ID	Accuracy (in %)	Execution Time (in sec)
Logistic Regression	LoR	90	27.211
Random Forest	RF	88	23.632
Naive Bayes	NB	66	17.314
Decision Tree	DT	81	19.935

achieved based on the model ID. From the Table 3, it is observed that the classification accuracy of the Logistic Regression model is better than the other ML models. It is important to note that the Logistic Regression can model feature interactions more effectively than Naive Bayes, which assumes that the features are conditionally independent given the class label. Decision Tree and Random Forest models can model feature interactions, but they may require more data and computational resources to do so effectively. Moreover, Decision Tree and Random Forest models are prone to overfitting, especially when the data has many features or there are many trees in the forest. Overfitting occurs when the model learns to fit the noise in the training data, rather than the underlying patterns. This can lead to poor generalization performance on the test data. Logistic Regression, on the other hand, is less prone to overfitting and can generalize better to unseen data. Therefore, the Logistic Regression model is recommended by the bot for the predictions. The last column of the table represents the execution time for each ML model. When more than one model is able to achieve the same level of accuracy, then the bot will recommend the model with the shortest processing time.

6.1 RCA Bot in Enterprise System

RCA Bot is also capable of being deployed in enterprise applications, as illustrated in Figure 8 . The interactions between Microservices, Salesforce, Mule Services, Workday, and SQL database are shown in this enterprise system. When a Microservice calls another service internally to process the customer's request, this results in a wide range of communication between the services and a high risk of failure of the entire system. The root cause of the failure is challenging to track down, and to find the source of the problem requires cross-team cooperation. As a result, it will be difficult to trace the failure and find the root cause. A robot like RCA comes into play here and saves the cross-team a lot of time and effort by automating these processes for them. In this case, you only need to stream the server log files of all services and components to the RCA Bot, and the rest will be handled automatically by the RCA system and the Root Causes will be predicted correctly by the system.

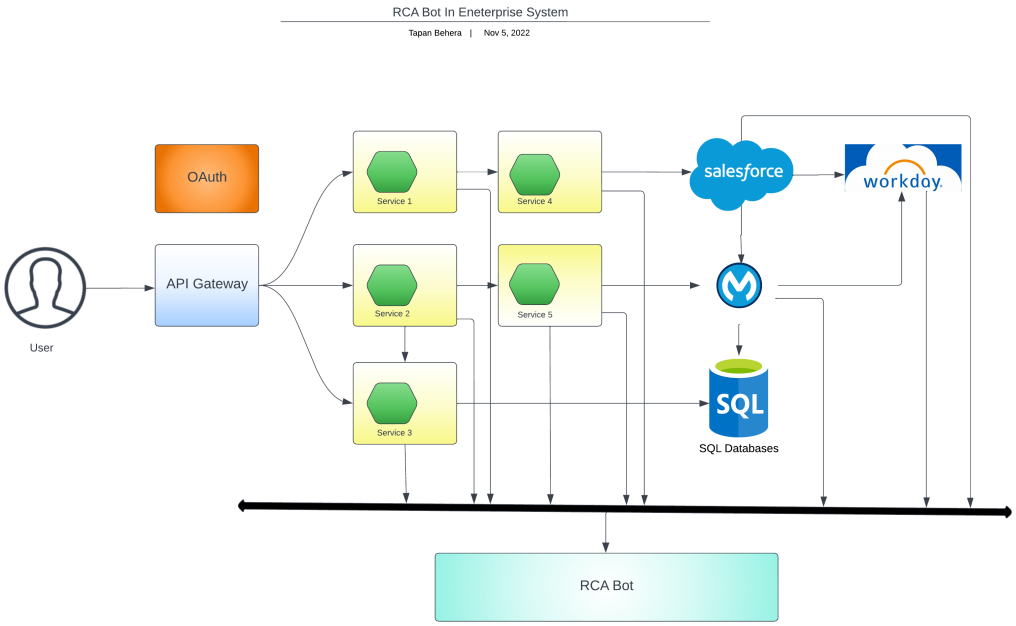


Figure 8: RCA Bot in Enterprise Application System.

7 Conclusion & Future Scope

In an era where technological growth in the field of software development is accelerating rapidly, bots can play an increasingly important role in accelerating an organization's growth. This RCA bot can be very helpful for complex enterprise applications, and it bears weight in the cross-team work when dealing with complex enterprise applications. There will be a tremendous amount of help from this bot when it comes to releasing the product early. As well as being helpful to the Operations team, who are responsible for managing legacy applications and production support for applications in production.

The RCA bot is designed in a generic manner so that it can be applied to any type of application to predict the cause of the failure by identifying its root cause. In the Healthcare Industry, patients were reminded to refill their prescriptions before the due date, but many patients did not receive the push notifications because the user restored a device from a backup, installed the app on a new device, or reinstalled the operating system, and the device token from where the notification was sent had changed. In any of these circumstances, the patient will not receive a reminder regarding the refill of their prescription. This RCA bot will be very helpful to find out the Root Cause of these issues.

IoT is the key to turning a city into a smart city. IoT is a system of several connected devices, such as sensors, vehicles, and home appliances. IoT devices can communicate and transfer data via

the internet without any human intervention. An IoT smart city also utilizes edge computing, creating an edge computing smart city. Edge computing brings compute power closer to an IoT device to save bandwidth and reduce latency. With all the benefits still there are lot of challenges in IoT system, security, failure in the devices connectivity, failure in registering and discovery of devices, problem on attaching a device to a gateway through cloud, communication events failure during message exchange. This lots of issues can be traced and immediately solved with the help of RCA Bot. Across all the IoT System logs this RCA Bot can trace and suggest the failure cause and solutions.

There is a lot of potential for further enhancement of this bot, and useful for any domain and industry, the immediate capability is to suggest the steps to be taken in order to resolve the issue which is discovered during the root cause analysis of the problem. In addition, this model can also be trained with other upcoming algorithms of classification in order to make it even better in the future.

References

[1] Alex Medeiros Araujo and Anderson Bergamini de Neira and Michele Nogueira, Autonomous machine learning for early bot detection in the internet of things, DOI: https://doi.org/10.1016/j.dcan.2022.05.011

[2] Apache Spark: https://spark.apache.org/docs/latest/ml-guide.html

[3] Andrea Ialenti: Effortless Hyperparameters Tuning with Apache Spark, 2020

[4] Anton Angelov, Automatetheplanet: Integrate Jenkins and GitHub https://www.automatetheplanet.com/integrate-jenkins-github/

[5] Balachandran. Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. Proceedings - International Conference on Software Engineering, pages 931–940, 2013.

[6] B. Cornu, T. Durieux, L. Seinturier, and M. Monperrus. NPEFix: Automatic Runtime Repair of Null Pointer Exceptions in Java. 2015.

[7] Behera, Tapan, Dr BS, Panda; How Blockchain Solves the Supply Chain Problems Using RFID Techniques (November 27, 2022). DOI: http://dx.doi.org/10.2139/ssrn.4287240

[8] Belgiu, M., Dragut, L. (2016). Random forest in remote sensing: A review of applications and future directions. ISPRS journal of photogrammetry and remote sensing, 114, 24-31.

[9] Beller, G. Gousios, and A. Zaidman. TravisTorrent : Synthesizing Travis CI and GitHub for Full-Stack Research on Continuous Integration.

[10] Behera, Tapan; Panda, BS (2023): Master Data Management using Machine Learning Techniques: MDM Bot. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.21818040.v1

[11] Breiman L, Friedman JH, Olshen RA, Stone CJ: Classification and Regression Trees (2nd Ed.). Pacific Grove, CA: Wadsworth, 1984.

[12] Charbuty, B., Abdulazeez, A. (2021). Classi

cation based on decision tree algorithm for machine learning. Journal of Applied Science and Technology Trends, 2(01), 20-28.

[13] C. L. Goues, T. Nguyen, S. Forrest, and W. Weimer. Genprog: A generic method for automatic software repair. IEEE Trans. Software Eng., 38(1):54–72, 2012.

[14] FileZilla FTP Connector: https://filezilla-project.org/

[15] GitHub. octoverse.github.com, 2017.

[16] Gyorodi, C., Gyorodi, R., Pecherle, G., Olah, A. A comparative study: MongoDB vs. MySQL. In 2015 13th International Conference on Engineering of

[17] Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, 2nd Edition, by Aurelien Géron

[18] Hossein Nematzadeh, José García-Nieto, Ismael Navas-Delgado, José F. Aldana-Montes, Automatic frequency-based feature selection using discrete weighted evolution strategy, 2022 DOI: https://doi.org/10.1016/j.asoc.2022.109699.

[19] I. Beschastnikh, M. F. Lungu, and Y. Zhuang. Accelerating software engineering research adoption with analysis bots. Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Results Track, ICSE-NIER 2017, pages 35–38, 2017.

[20] IBM Cloud Learn Hub: https://www.ibm.com/cloud/learn/supervised-learning

[21] IntelliJ, I. D. E. A. (2011). the most intelligent Java IDE. JetBrains https://www. jetbrains. com/idea/ chooseYourEdition.

[22] Jose Camacho-Collados, Mohammad Taher Pilehvar, On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis, DOI: https://doi.org/10.48550/arXiv.1707.01780

[23] Katherine B. Percarpio and B. Vince Watts and William B. Weeks, The Effectiveness of Root Cause Analysis: What Does the Literature Tell Us? doi: https://doi.org/10.1016/S1553-7250(08)34049-5

[24] Karau, Holden, and Rachel Warren. High performance Spark: best practices for scaling and optimizing Apache Spark. " O'Reilly Media, Inc.", 2017.

[25] Kim, Gene, et al. The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations. IT Revolution, 2021.

[26] Logstash Parser : https://www.elastic.co/guide/en/logstash/current/advanced-pipeline.html

[27] de Espona, Lucia, and Ela Pustulka. "MongoDB Data Versioning Performance: local versus Atlas." (2023).

[28] Mardiana, Tari, Teguh Bharata Adji, and Indriana Hidayah. "Stemming influence on similarity detection of abstract written in Indonesia." TELKOMNIKA (Telecommunication Computing Electronics and Control) 14.1 (2016): 219-227.

[29] Menard, S. (2002). Applied logistic regression analysis (No. 106). Sage.

[30] Qiuping Ma and Hongyan Li and Anders Thorstenson(2021) , A big data-driven root cause analysis system: Application of Machine Learning in quality problem solving

[31] Stojancho Tudjarski, Feature Selection in Details, 2020

[32] S. A. Carr, F. Logozzo, and M. Payer. Automatic contract insertion with ccbot. IEEE Transactions on Software Engineering, 43(8):701–714, 2017.

[33] Scikit-learn: https://scikit-learn.org/stable/modules/tree.html

[34] Svetnik, Vladimir and Liaw, Andy and Tong, Christopher and Culberson, J. Christopher and Sheridan, Robert P. and Feuston, Bradley P. Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling

[35] TechArcis : https://www.techarcis.com/why-is-root-cause-analysis-extremely-important/

[36] T. Durieux and M. Monperrus. DynaMoth: Dynamic Code Synthesis for Automatic Program Repair. In 11th International Workshop in Automation of Software Test (AST 2016), 2016.

[37] Tripathi, K., Rao, K. S. (2018). Improvement of phone recognition accuracy using speech mode classification. International Journal of Speech Technology, 21(3), 489-500. Modern Electric Systems (EMES) (pp. 1-6). IEEE.

[38] Urli, Simon and Yu, Zhongxing and Seinturier, Lionel and Monperrus, Martin, How to Design a Program Repair Bot? Insights from the Repairnator Project DOI: https://doi.org/10.1145/3183519.3183540

[39] V. Balachandran. Fix-it: An extensible code auto-fix component in review bot. IEEE 13th International Working Conference on Source Code Analysis and Manipulation, SCAM 2013, pages 167–172, 2013.

[40] Wikipedia. DevOps: https://en.wikipedia.org/wiki/DevOps

Authors Information

Tapan Kumar Behera is a Software Architect and specializes in Robotic Process Automation, who has over the years channelled his expertise within the healthcare, market research space. Tapan advises business and technology leaders in the transformation of both their organization and technology platform. He has worked with several Fortune 500 healthcare, market research and product-based companies such as Forrester Research, Walgreens, Cognizant, Hewlett-Packard, DXC and HCL. His experience cuts across a wide range of projects in Robotic Process Automation, RCA Bot, Master Data Management, Walgreens Mobile Application Service, Integration of Systems, Application Modernization, and Search engine tuning using AI/ML and so on. He is also the Author of a Book chapter "Architecture Principles for Enterprise Software and Mobile Application Development". His email Id: tkbehera.usa@gmail.com

Kumud Tripathi has completed her PhD from the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. There are several papers she has been published on Machine Learning and Neural Networks on Speech Recognition.

Corresponding Author

Correspondence to Tapan Kumar Behera, email: tkbehera.usa@gmail.com

Declarations

- Ethics approval and consent to participate: Not applicable.
- Consent for publication: The Author giving consents to publish this Article.
- Competing interests: The authors declare that they have no competing interests.