Query Processing of Distributed Databases using an Improved GraphQL Model and Random Forest Algorithm

Obasi E.C.M[*], Eke B.[**], Egbono F.[**]

[*] Department of Computer Science and Informatics, Federal University Otuoke, Bayelsa State, Nigeria.
[**] Department of Computer Science, University of Port Harcourt Choba, Rivers State, Nigeria.

DOI: 10.29322/IJSRP.12.04.2022.p12461
http://dx.doi.org/10.29322/IJSRP.12.04.2022.p12461

Paper Received Date: 10th April 2022
Paper Acceptance Date: 25th April 2022
Paper Publication Date: 2nd May 2022

Abstract- Query Processing is one of the major issues in a distributed databases. The method used to query a database can influence the response time of the query. Again, the increasing growth of dataset especially related records call for a database that can handle complex relationship. The relational database is an organized method of storing records. But relational database does not perform optimally when it comes to managing multiple relationships. A graph database is also a data storage mechanism. The graph database produces optimal results when handling multiple relationships. Structured Query Language is a standard language for querying a relational database but it is not efficient in querying a graph database. To query a graph, abstraction layer was introduced. A GraphQL, which is an Application Program Interface Query Language can be used to query a graph database. Adding artificial intelligence has been at the top of evolving technology. Software can be programmed to provide both predictive and reactive intelligence, based on the data collected and analyzed. Embedding artificial intelligence in a database application improves the performance of the system. The results of the GraphQL query was fed into a predictive model for classification. This research work was directed towards improving the administration of an organization by adopting an easy query method using GraphQL and a Random Forest Model for Prediction.

Index Terms- *Application Program Interface, Distributed Database, GraphQl, Query processing, Random Forest.*

I. Introduction

There are many distributed systems in operation today. A distributed system is a collection of computers that work together to form a single computer for the end-user. A distributed database represents multiple interconnected databases spread out across sites connected by a network as shown in figure 1

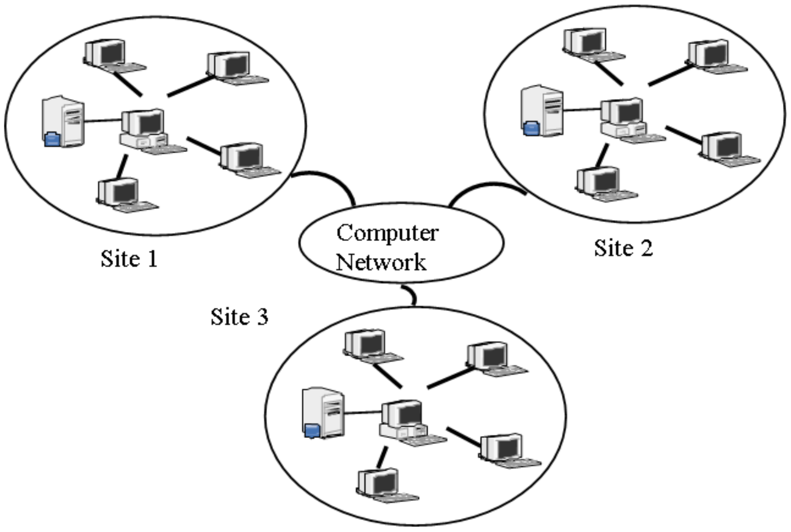


Figure 1: Distributed Database System[1]

Query processing is one of the major problems in distributed systems especially distributed databases. In light of recent technological changes and advancements, distributed systems are becoming more popular. Many top companies have created complex distributed systems to handle billions of request and upgrade without downtime. The methods to regain control in a very big relational databases in a distributed system are fully recognized. The language to access and manipulate data is a relational query language SQL. Again many platforms are open to intercommunication using protocols like REST. GraphQL which is similar to REST affords an integrated platform between the subscriber and the server to fetch and manipulate data. GraphQL is a modern way to build a declarative, efficient and performant API that everyone will like to work with. A GraphQL Server with integrated graph database was used in this research work. A graph database is a database that depends on graph theory. A Graph theory is a branch of Mathematics that is concerned with networks of points connected by lines. A graph database is developed purposely to take care of heavily linked data and the growth in the amount and linkage of recent data portrays great possibilities for maintained aggressive benefits. Not only do graph databases efficaciously store data relationships, they are also pliable when growing a data model or yielding to dynamic business requirements. In the other hand, relational database management systems (RDBMS) do not perform well when dealing with data relationships. Their strict schemas hinders addition to different connections or adoption to recent business needs. In our approach, Neo4j database was used as our graph database and a supervised learning approach, preferably Random Forest was applied to make predictions on the outcome of graphql query.

11. LITERATURE REVIEW

A. GRAPHQL: Application Programming Interfaces Language

Application Program Interfaces can be queried through a language such as GraphQL. The structure of GraphQL explains how data request should be done. Again, it is the program carried out in the server to perform query operations. It works across a specific endpoint. Making use of the HTTP protocol, it works in a particular connection end. Some features of Graphql includes a type system to specify and distinguish the data, using a single request to query multiple data sources, client specification of the exact data that is needed. It conforms to the partnership between objects vertically and the data returned to the client follows a graph structure. A type system makes it easier to determine the validity of query before runtime. When a query is certified valid, the query execution is carried out and the result is displayed, usually as a Java Script Object Notation object. GraphQL contains a contemplation system that enables an app to search the schema definition and decide whether queries are permitted or not. Graphql offers a specific endpoint to query and there is no insufficient fetching or over sufficient fetching of data. It is the duty of the client to specify data requirements such that more than necessary information would not be fetched from the Application Program Interface. This is to prevent making many API calls for a task. Specifying a type system and GraphQL schema assists in understanding what can be passed to and generated from the server. With this, independency can be achieved as Frontend and Backend teams can work without relying on each other. The GraphQL server works as an integrated layer to access and manipulate data. Figure 2 shows how client applications can send requests to the GraphQL server, which checks whether the query is valid and carries out the query execution by picking the information from the different data sources

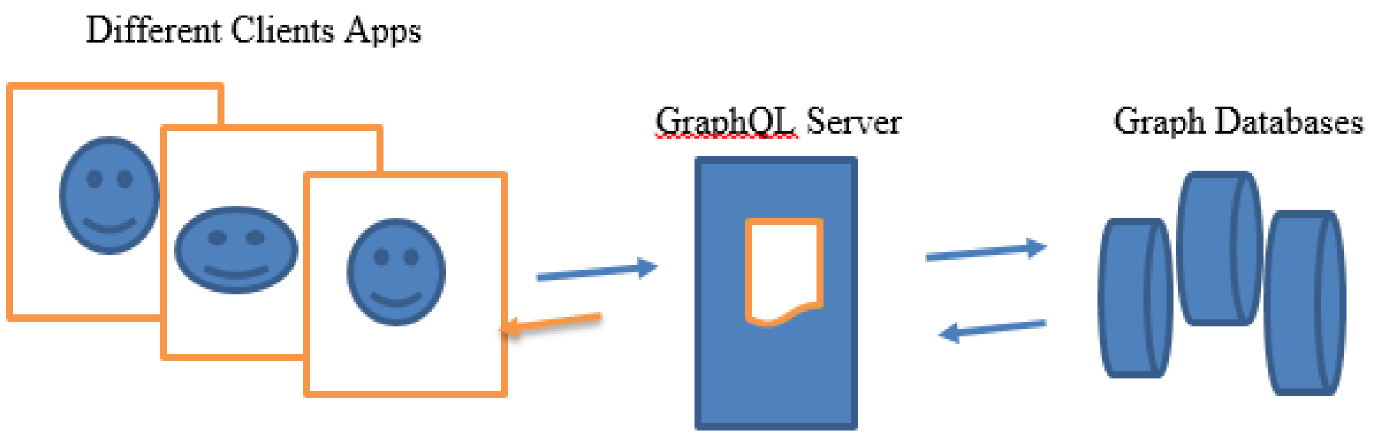


Figure 2. The Characteristic of the GraphQL Server in a distributed system

This protocol is a standard request–response message change by reversal practice, which has the next two messages a request, which should be made in a distinct query language and delivered as plain string to a unique GraphQL endpoint, and – a response of the GraphQL server defined as a Java Script Object Notation document. In many programs, composite, organized data is demanded from the Application Program Interface. To display a single entity with all its dependents from a unique API endpoint, a GraphQL query is organized in a vertical manner. The result of the query will be in the form similar to the request. The output document is a set of entities with their dependents declared in the type system. This elucidates the name GraphQL, since the entities and relations can be opined as a graph.

B. Random Forest Algorithm

1n Supervised learning, an AI system is presented with data which is labeled, which means that each data is tagged with the correct label. Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. Recently, the Random Forest (RF) algorithm [2] for Machine. Learning has achieved broad popularity. It is the most flexible and easy to use algorithm. A forest is composed of trees. . The number of trees is denoted by the parameter n_{tree} [3]. It is said that the more trees it has, the more robust a forest is. Random forest creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting as shown in figure 3. The Random forest algorithm combines multiple algorithms of the same type that is multiple decision trees, resulting in a forest of trees, hence the name “Random Forest”. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. . It also provides a pretty good indicator of the feature importance. It performs better results for classification problems.

The following Steps are involved in random forest algorithm**:**
Step 1: In Random forest n number of random records are taken from the data set having k number of records.
Step 2: Individual decision trees are built for every illustration.
Step 3: Each decision tree will generate an output.
Step 4: Final result is obtained based on ***Majority Voting or Averaging*** for Classification and regression respectively.
The followings are the advantages of Random Forest algorithm –

1. Random Forest algorithm is not biased, since there are multiple trees and each tree is trained on a subset of data. Basically, the Random Forest algorithm relies on the power of the “the crowd”. Therefore the overall biasedness of the algorithm is reduced. It aggregates the prediction outcome of multiple decision trees and creates a final outcome based on averaging mechanism (majority voting)
2. It overcomes the problem of overfitting by averaging or combining the results of different decision trees.
3. Random forests work well for a large range of data items than a single decision tree does.
4. Random forest has less deviation than single decision tree.
5. Random forests are very pliable and possess very high accuracy.

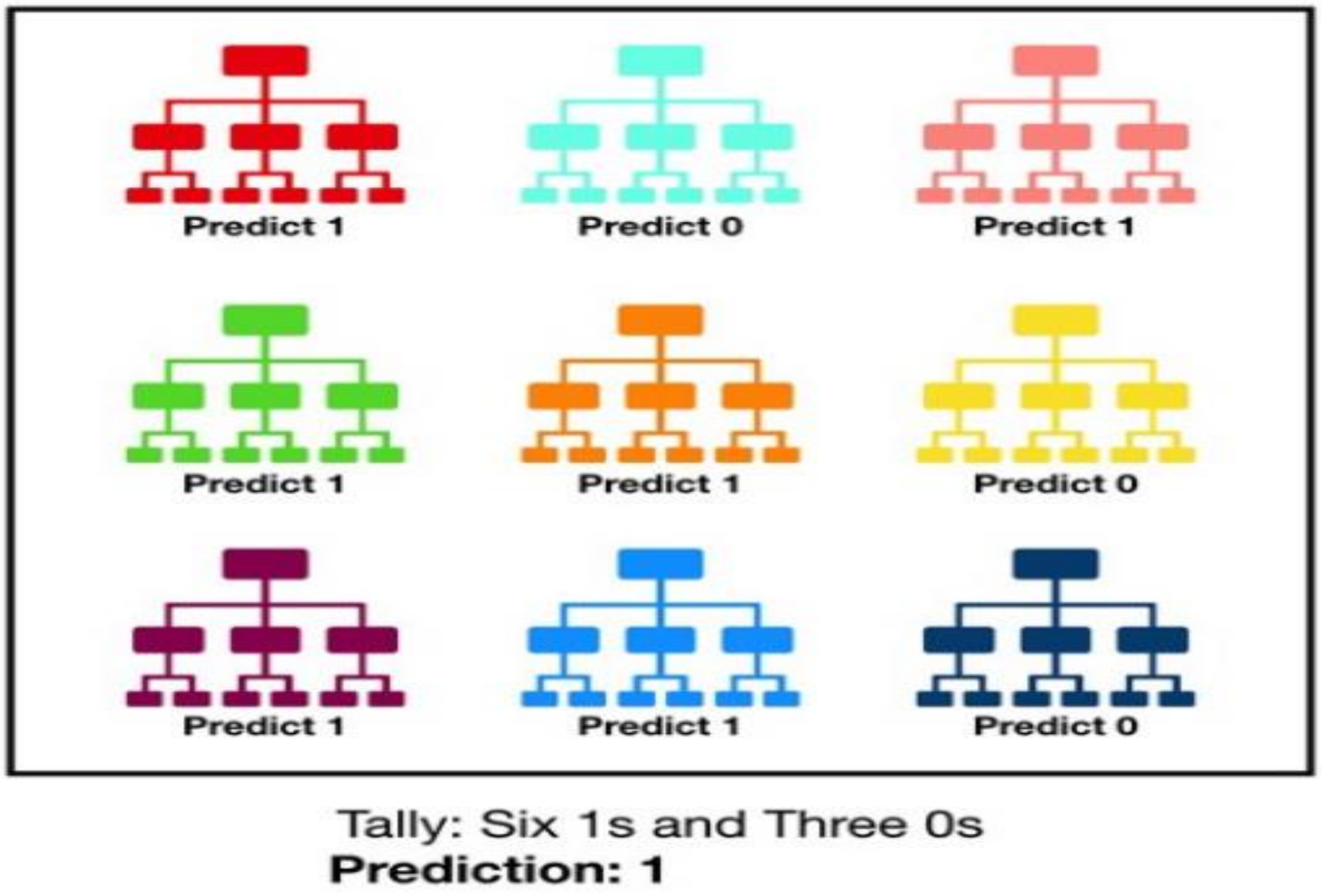


Figure 3: Visualization of Random Forest Model Making a Prediction [4].

C. Related Works

In [5] G-SQL: Fast Query Processing via Graph Exploration was proposed. The researchers developed G-SQL, which was a coordinated means to speed up the execution of costly SQL-graph hybrid queries. G-SQL makes use of an in-memory graph engine placed on upper side of a relational database. G-SQL greatly takes great advantage of full development of a relational database technology and the quick careful systematic search control of the in-memory graph engine. The researchers presented an integrated cost model to organize the subjacent SQL execution engine and the native graph query processing engine. The impressive achievement advantage presented by the graph-empowered G-SQL was made firm by their broad investigations. Many real-life data are in form of graph. Notwithstanding, it is just lately that organizations start utilizing the interconnectedness of data for organizational breakthrough. From another point of view, RDBMS are a fully developed technology to manage data but they cannot be used to process graph. For example, graph traversal which is a simple operation that can be done on a graph depends greatly on a graph primitives to retrieve information from the neighborhood of a node. Joining of tables is needed taking into consideration foreign keys to reach the nodes in the vicinity assuming management of a graph data is done with RDBMS. Graph search is a basic part of major graph algorithms. Though this operation is easy. It is expensive because of the great amount of I/O caused by the excessive amount of joins in tables. They proposed G-SQL in their research. G-SQL impinges on the quick graph search strength given by graph engine to respond to multi-way join queries. In the mean time, it made use of RDBMSs to render reliable data management practicality, like dependable storage of data and other more methods to access data. To be specific, G-SQL was a SQL dialect supported with graph search practicalities and it sent query request to the in-memory graph engine and its subjacent RDMBS. The management of the double query processors through an integrated cost model to make sure that the entire query was executed expeditiously was done by G-SQL runtime. The outcome of the experiment revealed that their method increased the functionalities of RDBMs and generated extraordinary operations for SQL-graph hybrid queries.

In [6] G-CORE: A Core for Future Graph Query Languages was proposed. The study reported on a group venture between industry and academia to direct the future of graph query languages. Furthermore, the authors argued that existing graph database management systems should give consideration to encouraging a query language with two important features. The authors did a good job. However, performance evaluation on the G-CORE showed that there was deficiency in time complexity, benchmarking and cost benefit analysis.

In [7], a new dynamic query optimization algorithm based on the greedy algorithm that uses the randomized strategies was developed. The execution cost of queries and system resources requirements were reduced significantly, but only applicable to centralized database systems.

Effective Learning and Classification using Random Forest Algorithm was proposed [8]. Their research work looked at means for making better the performance of Random Forest Classifier in areas of accuracy, and time for learning and distribution into classes. In the area of improving accuracy, investigation was done using diverse feature check metrics and mix procedures. Suggestion made was a complex decision tree model in addition to weighted voting and this improved the accuracy. Learning and classification was made faster by focusing improvement in learning on slashing the amount of base decision trees in Random Forest. A new parallel approach was proposed in which both, individual tress as well as entire forest was generated in parallel. These new methods led to efficacious learning and classification using Random Forest Algorithm.

[9] researched on Machine Learning Framework depended on Random Forest Algorithm. Targeting at the limits that already existed in random forest algorithm, their research critically examined the machine learning framework based on random forest algorithm, designs and implemented a set of machine learning framework, optimized the existing limit in random forest algorithm. It also provided a machine learning algorithm platform and important information for concealed data preprocessing, choice of parameter and optimization algorithm for programmers that lack requisite knowledge of step by step method of developing machine learning software, decreasing the boundary for the use of machine learning framework.

11I. ARCHITECTURE DESIGN

A. Research Methodology

The research methodology used for the proposed system was Object-Oriented Analysis and Design (OOAD) Methodology. Object-oriented programming (OOP) is a programming paradigm that uses “objects” and their interactions to design applications and computer programs. Object Oriented Analysis introduces new concepts to investigate a problem. It is based in the set of basic principles, which are

i. The information domain is modelled
ii. Behavior is represented.
iii. Function is described.
iv. Every models represent the essence of the problem, while later ones provide implementation details.

B. Analysis of Existing System

[10] proposed an Improved GraphQL Model Query Processing of Distributed Databases as shown in figure 4. The research work critically examined query operations done on a relational database management system. Relational database management system is an organized database management mechanism that stores records on tables. Structured Query Language is a standard language for querying relational database. But as the number of tables increase, the performance of relational database gets reduced since query processing involves complex operations. The researchers adopted graph database preferably neo4j databases as a data management system for increased dataset. The distributed databases were stored in neo4j databases. The joins operations used in relational databases are not needed in graph databases and this makes graph database cheaper. The neo4j database were queried using GraphQL which is an Application Program Interface Query Language, developed by facebook in 2012, open-sourced in 2015.The neo4j databases were implemented using Django neomodel. Django made it possible to create the entire site and database backend, along with the plugins used. The neo4j inference engine applied some reasoning on neo4j database to return the results of the data queried for. The neo4j database demonstrated excellent performance as it was able to handle high volume of connected data. Faster query results were achieved through GraphQL as it provided simple and precise query and returned the results of a query through a single endpoint. Also the performance of the graph database did not decline as dataset increased.

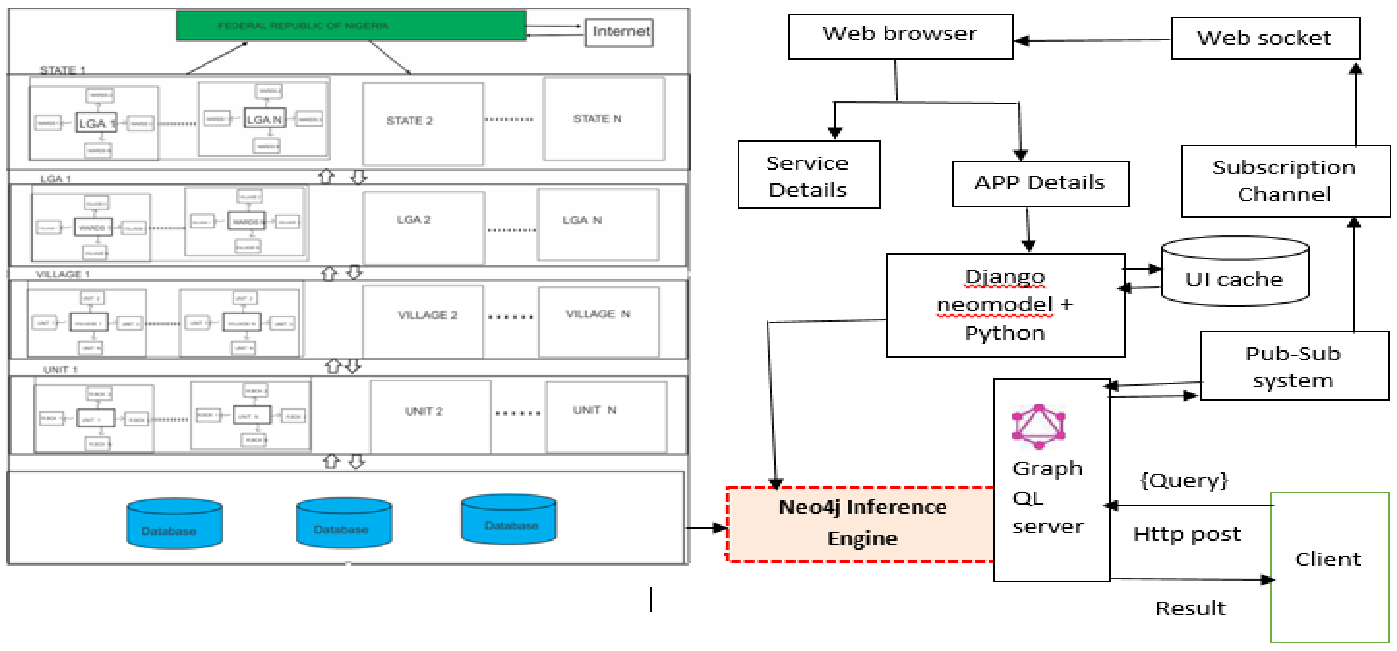


Figure 4: Architecture of an improved Graph Query Language Model Query Processing of Distributed Databases [10].

The Shortfall of the Existing System.

In the existing system, no artificial intelligence was embedded in it. Making a system to be intelligent is one of the evolving ideas in the recent years. The use of machine learning algorithm to classify the output of the query results is highly needed in the existing system as this eases some stress for humans. Hence the need for a new system.

C. The Proposed System

The proposed improvement for the existing system encompassed the application of Random Forest Classifier (figure 5). Random forests is a supervised learning algorithm. Classification and regression problems can be solved using the algorithm. The algorithm is very pliable and simple to use. A forest is made up of trees. The number of trees in a forest determines the robustness of a forest. Random forests generates decision trees on haphazardly picked data samples, gets the prediction a tree makes and picks the optimal solution making use of voting. It can also be used to determine the feature importance values. Random forests can be applied in different areas like recommendation engines, image classification and feature selection. One can use it to determine a patriotic loan requester, discover dishonest activity and predict different diseases. In the proposed system, Random forest was used to classify members of the organization into two categories: Dedicated and Non dedicated members. In the sharing of the benefits from the organization, worthy members should be considered first. Scholarship could be given, appointments and prediction made, contracts awarded based on the prediction of the Random Forest Classifier. Graph database and GraphQl were still used in the proposed system. As GraphQl returned the results of the query, these results were fed into the Random Forest Classifier for prediction. Having trained the dataset using Random Forest Algorithm, accurate prediction was made. There are other machine learning algorithms for classification like Support Vector Machine, Decision Trees, etc but Random forest was chosen because of its ability to make good predictions even with small datasets. Based on these predictions, the problem that arises when members gather together either for campaigning or sharing of benefits was address as sharing was based on meritocracy and not nepotism. Disagreements, violence, killings and the greedy accumulation of wealth was reduced to the barest minimum since privy communication was channelled through the network.

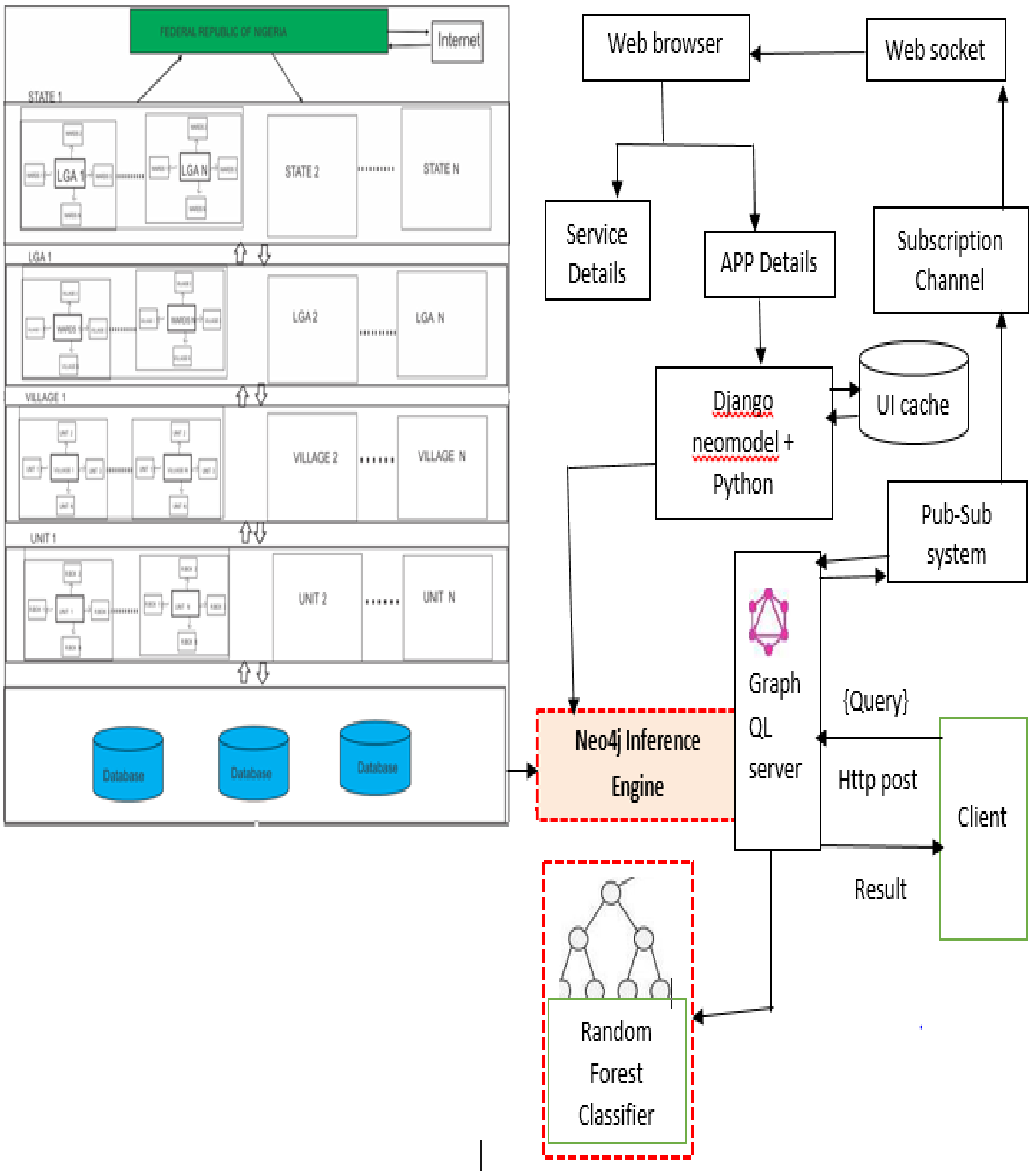


Figure 5: Architecture of the Proposed System

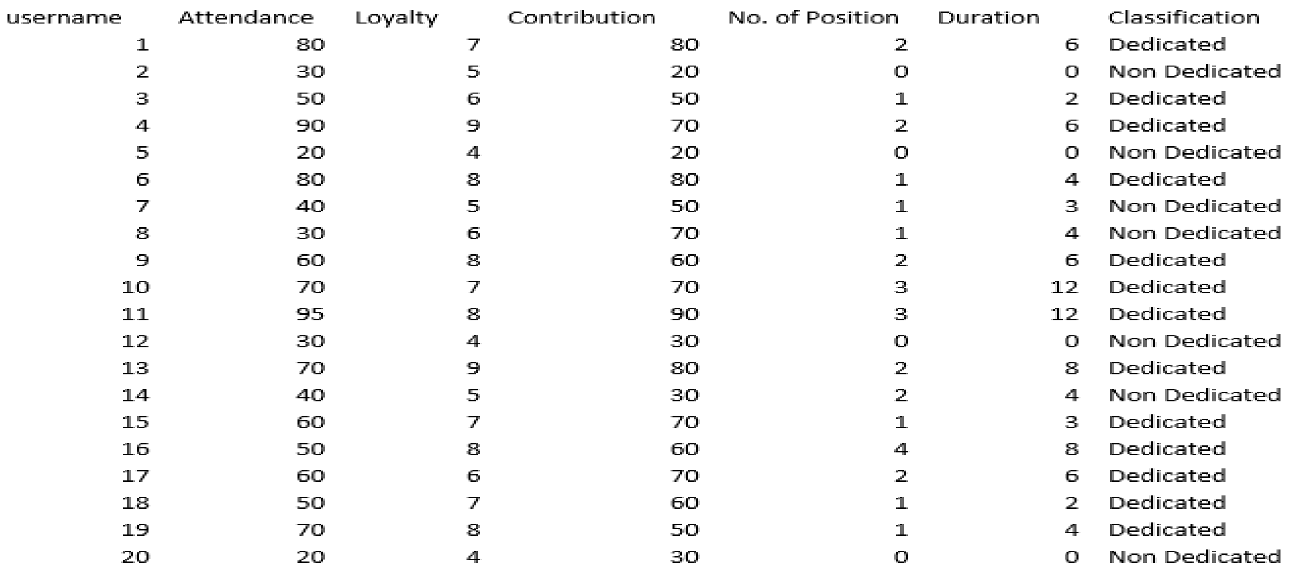
Advantages of the Proposed System

i. Incorporating Random Forest Algorithm into the new system made the system intelligent.
ii. Classification of members of the organization into dedicated and non dedicated members helped in fair sharing of benefits.
iii. Application of machine learning allowed the system to become more accurate in predicting outcomes without being explicitly programmed to do so.
iv. Random Forest algorithm introduced in figure 5 relied on the power of 'the crowd' as each tree is trained on a subset of data to make decision.

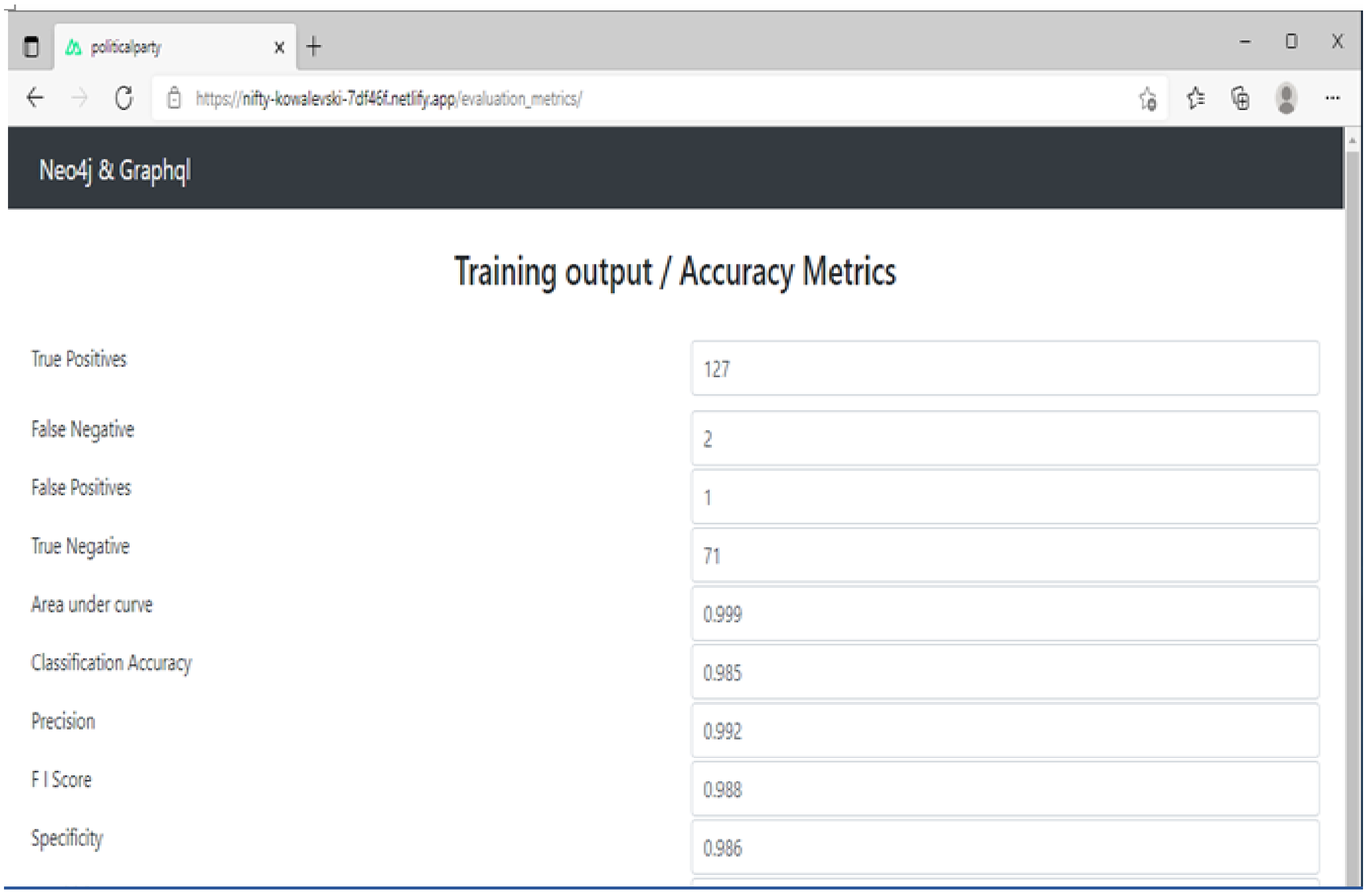
1V. IMPLEMENTATION AND SAMPLE RESULTS

To implement the GraphQL with Random Forest Algorithm effectively, some tools were used. The programming languages used for the proposed system were JavaScript and Python respectively. GraphQL plays nicely with familiar languages and frameworks like javascript. The web page was developed using hypertext markup language(HTML), Cascading Style Sheet and javascript. Javascript was also used as a front end to construct query and perform mutations. Apollo platform was used as an implementation of GraphQL that could transfer data between the cloud (server) to the user interface of the application. Apollo platform contained a Javascript GraphQL server that defined the schema and a set of resolvers that implemented each part of that schema. On the other hand, the Random forest classification algorithm was implemented in Python 3.6 platform using the Science Kit Learn Library available on the Anaconda 5.2 data science package for windows. Python library Pandas was used for data manipulation relying on the structure known as a dataframe, which in essence is an excel spreadsheet as shown in Table 1. Pandas read the csv files for organization's records. NumPy which stands for Numerical Python is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. NumPy was responsible for converting the excel file into an array. For the distributed databases of organization's administration, Neo4j database was used. Neo4j is a non- relational graph database that is optimized for managing relationships. To add neo4j-based functionality to the proposed system, Django-neomodel plugin was used. Django neomodel is an Object Graph Mapper (OGM) for the neo4j graph database built on the awesome neo4j_driver. This module allowed us to use the neo4j graph database with Django using neomodel. Random Forest classification algorithm was embedded into the system to classify the performance of members. Table 1 showed the sample of the dataset for organization's administration and features used for training. The dataset was prepared in excel format. 70% of the dataset was used for training while 30% was used for testing. Python programming language was used for the training. Python has a lot of in-built libraries that makes programming easier. It read the csv files. The data was separated into the features and targets. The target, also known as the label was the value we wanted to predict and the features were all the columns the model used to make a prediction. We converted the Pandas dataframe to Numpy arrays because that was the way the algorithm worked. Numpy was responsible for converting the excel file into an array and could perform mathematical operations / functions on that array. Another library Matplotlib is a python 2D plotting library used to plot the graphs and produce figures in a numerous hard copies. Figure 6 showed the evaluation metrics of the Random Forest Training .Classification accuracy was 0.985. This shows the high predictive accuracy of the model. Accuracy measures how many observations, both positive and negative, were correctly classified. Random Forest also showed the feature that was contributing most to the classification. Figure 7 showed the feature importance of the model and we could depict that 'attendance' contributed most to the classification. Figure 8 and Figure 9 showed the Lift Curve for the dedicated and non dedicated members of the organization. The lift charts compared the ratio between the results obtained with and without a model. It is a chart between the lift on the vertical axis and the corresponding docile on the horizontal axis .The greater the area between the Lift, the better the model. Figure 10 showed the Pythagorean Forest of the model. Pythagorean forest is a visualization of Random Forest algorithm. Random Forest takes N samples from a data set with N instances, but with replacement. Then a tree is grown for each sample, which alleviates the Classification Tree's tendency to overfit the data.

Table 1: Sample of Dataset used for Random Forest Training.



username	Attendance	Loyalty	Contribution	No. of Position	Duration	Classification
1	80	7	80	2	6	Dedicated
2	30	5	20	0	0	Non Dedicated
3	50	6	50	1	2	Dedicated
4	90	9	70	2	6	Dedicated
5	20	4	20	0	0	Non Dedicated
6	80	8	80	1	4	Dedicated
7	40	5	50	1	3	Non Dedicated
8	30	6	70	1	4	Non Dedicated
9	60	8	60	2	6	Dedicated
10	70	7	70	3	12	Dedicated
11	95	8	90	3	12	Dedicated
12	30	4	30	0	0	Non Dedicated
13	70	9	80	2	8	Dedicated
14	40	5	30	2	4	Non Dedicated
15	60	7	70	1	3	Dedicated
16	50	8	60	4	8	Dedicated
17	60	6	70	2	6	Dedicated
18	50	7	60	1	2	Dedicated
19	70	8	50	1	4	Dedicated
20	20	4	30	0	0	Non Dedicated


politicalparty
https://nifty-kowalevski-7df46f.netlify.app/evaluation_metrics/
Neo4j & Graphql

Training output / Accuracy Metrics

True Positives	127
False Negative	2
False Positives	1
True Negative	71
Area under curve	0.999
Classification Accuracy	0.985
Precision	0.992
F I Score	0.988
Specificity	0.986

Figure 6: Evaluation Metrics of the Random Forest Training

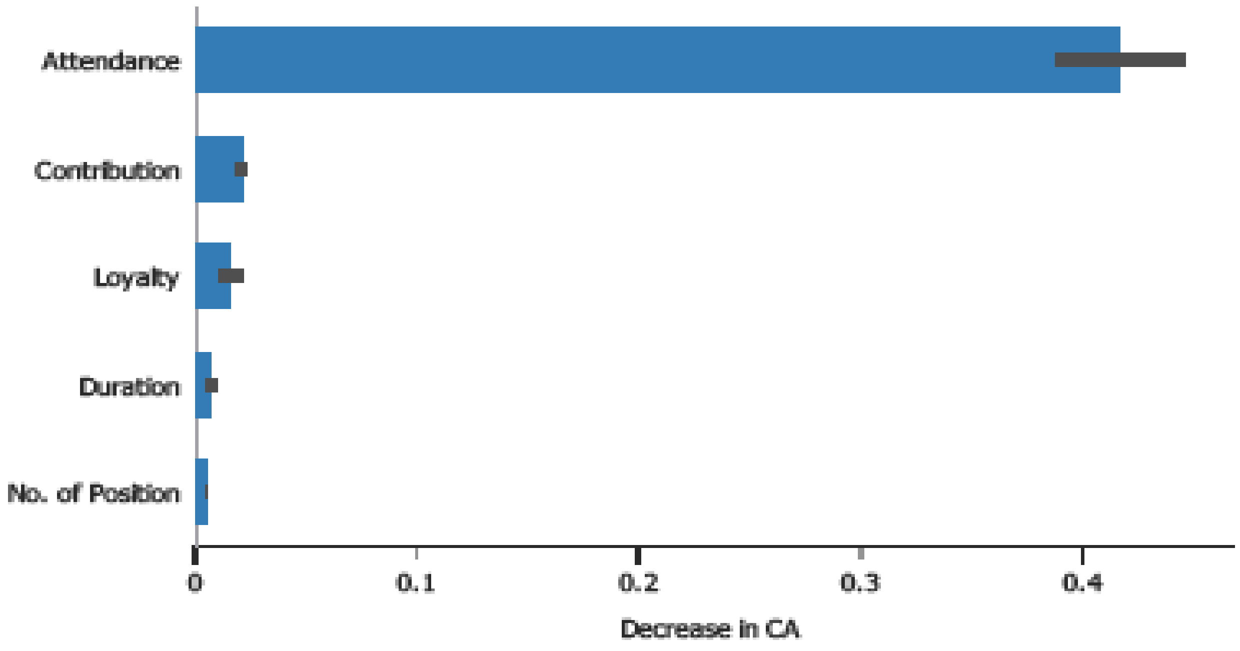


Figure 7: Future Importance Classification Accuracy

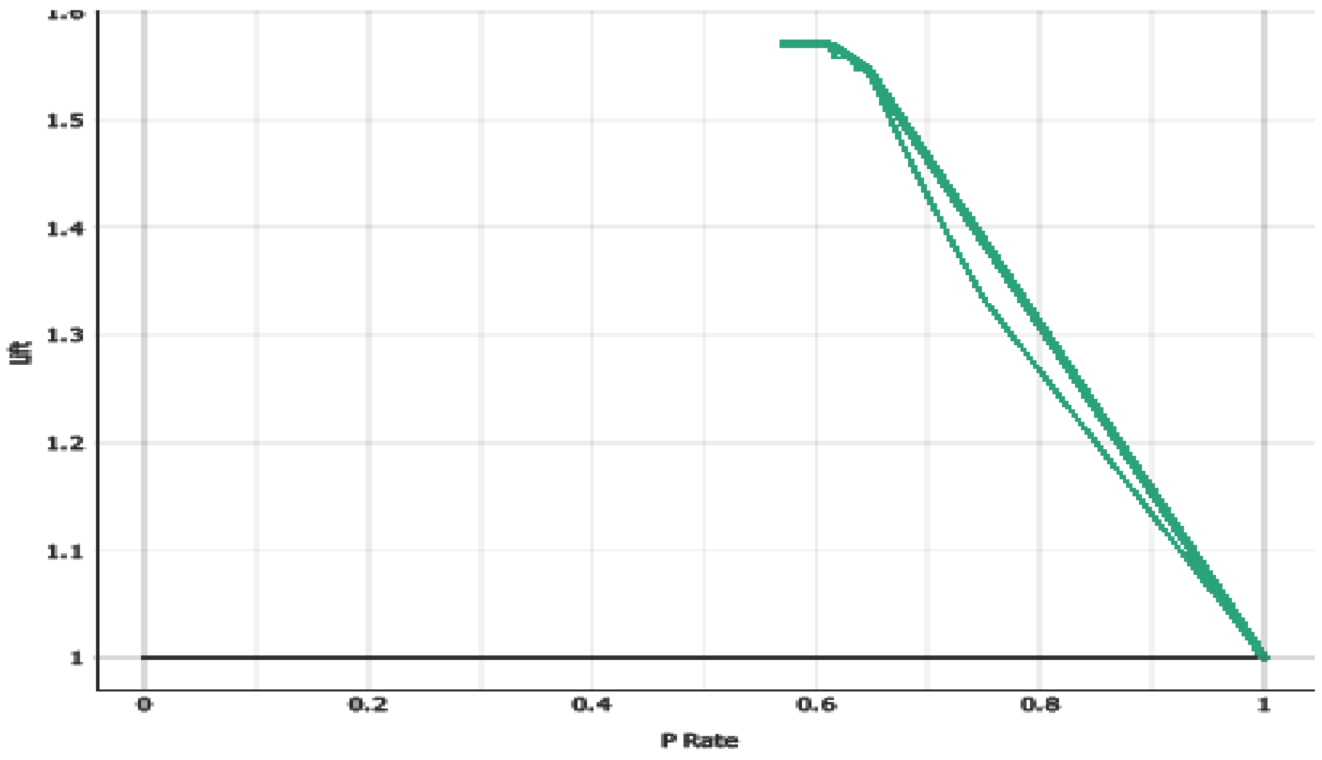


Figure 8: LIFT Curve for Dedicated

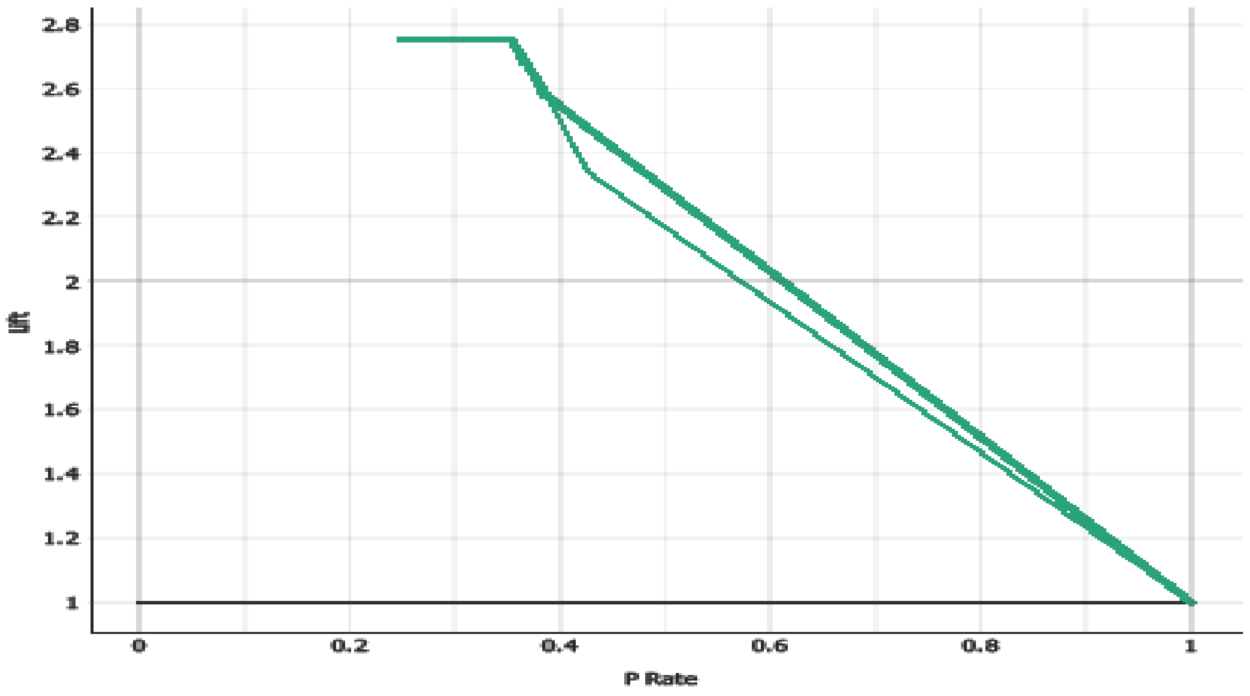


Figure 9: LIFT Curve for non dedicated

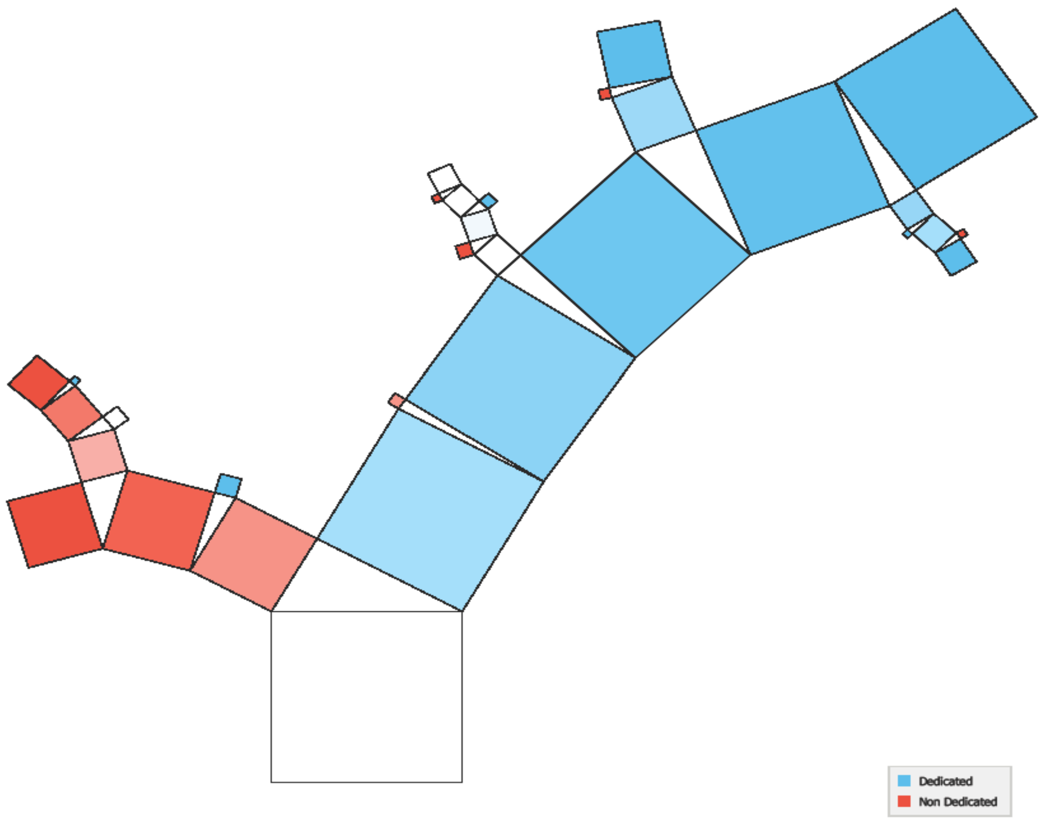


Figure 10: Pythagorean Forest

1V. CONCLUSIONS

Query Processing of Distributed Databases using Improved GraphQL Model and Random Forest Algorithm has been developed. Based on the findings of the research work, the new system generated better results. The inclusion of Random Forest Machine Algorithm simplified the system and eased the work of the user. Graph database was used as a data storage mechanism and query was done on graph database through GraphQl. The results of the query were passed through Random Forest (RF)Machine Algorithm for classification. The accuracy of the RF model was recommendable. The Random Forest model was used to solve a binary classification problem. Members of an organization were classified into Dedicated and non-dedicated members. This has solved a lot of problems that arose when members gathered to share benefits, be it financial, contracts, scholarships and other rewards. The new system has demonstrated excellent performance and could do a lot of operations with less human interventions.

ACKNOWLEDGMENT

Our profound gratitude goes to the Almighty God for His sustenance, providence and unflinching firmness obtained during the course of this research. We also acknowledge the efforts of other lecturers in the department of Computer Science, University of Port Harcourt towards the successful completion of this research work.

REFERENCES

[1] Mohamed Firdous, *Implementation of Security in Distributed Systems-A Comparative Study*, International Journal of Computer Information Systems, 2011, 2(2)

[2] Breiman L., *Random Forest*, Mach Learn, 2001, 45, 5-32

[3] Liaw A Wiener M.,*Classification and Regression by Random Forest*, R News. 2002, 2, 18-22

[4] Tony Yiu, Understanding Random Forest, 2019.

[5] Hongbin M., Bin S., Yanghua X., Liang J. and Haixun W., *G-SQL:Fast Query Processing via Graph Exploration,* Proceedings of VLDB Endowment, 2016, 9(12)

[6] Renzo A., A.Marcelo, B.Pablo, *G-Gore:A Core for Future Graph Query Languages*, SIGMOD 18.2018, Houston, TX, USA, 1-12

[7] Satyanarayana, SK.Sharfuddin, SK.JanBhasha, *New Dynamic Query Optimization Technique in Relational Database Management Systems*, International Journal of Communication Network Security.2013, 2, 2231-1882

[8] Vrushali Y Kulkarni and Pradeep K Sinha , *Effective Learning and Classification using Random Forest Algorithm*, International Journal of Engineering and Innovative Technology, 2014, 3(11).

[9] Qiong Ren, Hui Cheng and Hai Han, Research on Machine Learning Framework Based on Random Forest Algorithm, AIP Conference Preceedings 1820, 080020, 2017.

[10] Obasi E.C.M, Eke B., Egbono F., *Improved GraphQL Model Query Processing of Distributed Databases*, Global Scientific Journal,2022, 10(3), 877-883

AUTHORS

First Author – Obasi Emmanuela Chinonye Mary, B.Tech (FUTO), M.Sc (UniLag), Federal University Otuoke, Bayelsa State, Nigeria. anchinos@yahoo.co.uk.
Second Author – Eke Bathlomew.O., BSc.,M.SC.,Ph.D.(UPH), University of Port Harcourt, Choba, Nigeria. bathoyol@gmail.com.
Third Author – Egbono Fubara., B.Sc. (USSR), M.SC.(USSR), Ph.D(ESU), University of Port Harcourt, Choba, Nigeria. fubaraegbono@gmail.com

Correspondence Author – Obasi Emmanuela Chinonye Mary, anchinos@yahoo.co.uk, obasiec@fuoke.edu.ng , 07036673665.