**SEARCH RANK FRAUD AND MALWARE DETECTIONIN GOOGLE PLAY**

|  |  |
| --- | --- |
| **TABLE OF CONTENTS** | |
| **S. NO.** | **TITLE** |
| 1 | Abstract |
| 2 | Introduction |
| 3 | Existing System |
| 4 | Disadvantages |
| 5 | Proposed System |
| 6 | Advantages |
| 7 | Flow Diagram |
| 8 | Use Case Diagram |
| 9 | Activity Diagram |
| 10 | Sequential Diagram |
| 11 | ER Diagram |
| 12 | Architecture Diagram |
| 13 | Modules |
| 14 | Modules Description |
| 15 | Literature Survey |
| 16 | Hardware & Software Requirements |
| 17 | Hardware & Software Description |
| 18 | Testing of Product |
| 19 | Conclusion |
| 20 | Future Enhancement |
| 21 | Coding & Screenshot |
| 22 | References |

**Cloud based malware detection game for mobile devices with offloading**

**ABSTRACT**

Fraudulent behaviors in Google Play, the most popular Android app market, fuel search rank abuse and malware proliferation. To identify malware, previous work has focused on app executable and permission analysis. In this paper, we introduce Fair Play, a novel system that discovers and leverages traces left behind by fraudsters, to detect both malware and apps subjected to search rank fraud. Fair Play correlates review activities and uniquely combines detected review relations with linguistic and behavioral signals gleaned from Google Play app data (87 K apps, 2.9 M reviews, and 2.4M reviewers, collected over half a year), in order to identify suspicious apps.Fair Play achieves over 95 percent accuracy in classifying gold standard datasets of malware, fraudulent and legitimate apps.

**DOMAIN INTRODUCTION**

Data Mining is the Non-trivial extraction od implicit previously unknown and potential useful information from the data. Data mining is the process of sorting through large data sets to identify patterns and establish relationships to solve problems through data analysis. Data mining tools allow enterprises to predict future trends. Data mining is the computing process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. It is an interdisciplinary subfield of computer science.

**INTRODUCTION**

MALWARES such as viruses, worms, Trojans, and spy tools seriously threaten mobile devices such as smartphones and tablets with the privacy leakage, economic loss, power depletion, and network performance degradation. Malware detection systems as proposed in explore the features of the runtime behaviour of thousands of applications (apps) on mobile devices and involve logging data at each application execution. For instance, 117,903 lines of log data have to be scanned to detect malware on an Android smartphone according to Norton Mobile Security. The application traces have to be scanned in real time based on the latest malware signature files downloaded from the security database to avoid the privacy loss due to the zero-day vulnerability. However, that is not always applicable for a mobile device with the limited battery life, computation resources and network bandwidth. By offloading the detection tasks to security servers, cloud-based malware detection can address the zero-day vulnerability and reduce the computation and energy consumption of mobile devices. For instance, the cloud-based security server in hosts a synchronized replica of a mobile device in virtual environments to detect malware simultaneously without overburdening the mobile device. The cloud-based intrusion detection framework in de this research was supported in part by National Natural Science Foundation of China under Grant 61671396 and the CCF-Venustech Hengyang Research Initiative (2016-010). Tests abnormal system calls of smartphone applications to address zero-day attacks with scalability, elasticity and low costs. The detection accuracy of a security server increases with the volume of the samples uploaded from mobile devices, according to [8], [9], [10], and the log data have to be analysed in real time to avoid zero-day attacks. However, the cloud-based detection process that incorporates the radio transmission, wired transmission and the processing of the log data at the cloud sometimes have long delay, if the mobile device experiences severe shadow fading or the serving base station (BS) or access point (AP) cannot provide enough radio bandwidth. In addition, the malware detection that runs background has to avoid interfering with real-time applications such as videos and online games, and the trace offloading cannot occupy all the available radio bandwidth. Therefore, the malware detection speed depends on the network bandwidths, the computation capacity of the security server and the status of the applications that run on the mobile device.

**EXISTING SYSTEM**

* Fraudulent behaviours in Google Play, the most popular Android app market, fuel search rank abuse and malware proliferation. To identify malware, previous work has focused on app executable and permission analysis.
* In this paper, we introduce Fair Play, a novel system that discovers and leverages traces left behind by fraudsters, to detect both malware and apps subjected to search rank fraud. Fair Play correlates review activities and uniquely combines detected review relations with linguistic and behavioural signals gleaned from Google Play app data (87 K apps, 2.9 M reviews, and 2.4M reviewers, collected over half a year), in order to identify suspicious apps.
* Fair Play achieves over 95 present accuracy in classifying gold standard datasets of malware, fraudulent and legitimate apps.

**DISADVANTAGES**

* It does not have a correct accuracy.
* Time delay
* Loss of Information.

**PROPOSED SYSTEM**

* Instead of analyzing app executable, Fair Play employs a relational, linguistic and behavioral approach based on longitudinal app data. Fair Play’s use of app permissions differs from existing work through its focus on the temporal dimension, e.g., changes in the number of requested permissions, in particular the “dangerous” ones. We observe that Fair Play identifies and exploits a new relationship between malware and search rank fraud.
* Performing SVM algorithm we analyze the result of the fraud detect in Google paly.

**ADVANTAGES**

* High Performance.
* No Time Delay
* Improving Security.

**DIAGRAMS**

**FLOW DIAGRAM**

Select Data

Load Data

Preprocessing

Cluster

Prediction

Generate the Data

Data User

Result generate in graph

**USE CASE DIAGRAM**

**BUSINESS ANALYST**

**CLASS DIAGRAM**

**PREPROCESS**

Dataset select()

Dataset Load()

Preprocess()

**USER INTEREST TRACKING**

Calculate Term Frequency()

Term Frequency

**BITERM CONSTRUCTION**

Tokenization()

Stop words Removal()

Stemming()

Word Pairs Generation()

**USER CLUSTERING AND CLASSIFICATION**

Track user(), K- Means Cluster()

Naïve-Bayes Classification()

**REPORT GENERATION**

Analysis Report()

**ACTIVITY DIAGRAM**

RESULT GENERATION

PREDICTION

DATA SELECTION & LOAD

DATA PREPROCESSING

TOKENIZATION

STOP WORDS REMOVAL

CLASSIFICATION

CLUSTERING

TERM FREQUENCY CONSTRUCTION

STEMMING

**SEQUENCE DIAGRAM**

DATA SELECTION & LOAD

DATA PREPROCESSING

BITERM CONSTRUCTION

USER INTEREST TRACKING MODEL

Select Dataset

Preview Data

Data Preprocess

Preview Data

User Interest

CLUSTERING USERS and CLASSIFICATION

Load Data

Tokenization

Clustered users and Classification of users and tweets

ANALYSIS REPORT GENERATION

Postagging

Stopwords Removal

Stemming

Word Pairs Construction

**ER DIAGRAM**

**DATA SELECTION & LOAD**

**DATA PREPROCESS**

**BITERM CONSTRUCTION**

**CLUSTERING USERS AND CLASSIFICATION**

**Analysis Report Generation**

**TERM FREQUENCY CONSTRUCTION**

**ARCHITECTURE DIAGRAM**

**Tweets**

**Tokenization**

**Stop-words removal**

**Stemming**

**Word Pairs Construction**

**Tracking User Interest**

**Clustering Users**

**Classification**

**Result Generation**

**MODULES**

* Select the data
* Load the Data
* Preprocessing the Data
* Clustering the Data
* Predict the Fraud in dataset
* Generate the result in chart

**SELECT THE DATA**

Data selection is the process of selecting the appropriate data set for processing.The dataset which contains the fields of Application name, category, reviews, rating, size, installs, price, current version and android version.

**LOAD THE DATA**

The selected dataset is going to be used for identifying the fraud about the Google play. The dynamically distributed reviews are detected from the Google play dataset. The selected data is loaded into the database.

**PREPROCESSING THE DATA**

The data is pre-processed to remove unwanted data from dataset, since we would like to associate each data from the Google play. Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens. Tokens can be individual words, phrases or even whole sentences.

CLUSTERING THE

Support vector clustering has the following idea: let us transform the points from their space to a higher dimensionality feature space. Find a minimal enclosing sphere in this feature space. In the original space, the sphere becomes a set of disjoin regions. Each region becomes a cluster. Support Vector Machines (SVMs) provide a powerful method for classification (supervised learning). Use of SVMs for clustering (unsupervised learning) is now being considered in a number of different ways.

**PREDICT THE FRAUD**

We will analyze the dataset and cluster the dataset. Cluster means group and divide the dataset. So we will take the group of the particular data. The Naive Bayes algorithm is used to predict the fraud in the dataset. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data.

**GENERATE THE RESULT IN CHART**

The overall clustering report is generated based on their reviews in google play. The report which contain dynamically distributed reviews in google play. To predict the fraud in the dataset and result will be display in chart.

**LITERATURE SURVEY**

**Risk Ranker: Scalable and accurate zero-day Android malware detection**

**M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang**

**2012**

Smartphone sales have recently experienced explosive growth. Their popularity also encourages malware authors to penetrate various mobile marketplaces with malicious applications (or apps). These malicious apps hide in the sheer number of other normal apps, which makes their detection challenging. Existing mobile anti-virus software are inadequate in their reactive nature by relying on known malware samples for signature extraction. In this paper, we propose a proactive scheme to spot zero-day Android malware. Without relying on malware samples and their signatures, our scheme is motivated to assess potential security risks posed by these untrusted apps. Specifically, we have developed an automated system called RiskRanker to scalable analyze whether a particular app exhibits dangerous behavior (e.g., launching a root exploit or sending background SMS messages). The output is then used to produce a prioritized list of reduced apps that merit further investigation. When applied to examine 118, 318 total apps collected from various Android markets over September and October 2011, our system takes less than four days to process all of them and effectively reports 3281 risky apps. Among these reported apps, we successfully uncovered 718 malware samples (in 29 families) and 322 of them are zero-day (in 11 families). These results demonstrate the efficacy and scalability of RiskRanker to police Android markets of all stripes.

**Advantages**

* Low latency

**Disadvantages**

* Good Accuracy

**Android Permissions: A Perspective Combining Risks and Benefits**

**B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy**

**2012**

The phenomenal growth of the Android platform in the past few years has made it a lucrative target of malicious application (app) developers. There are numerous instances of malware apps that send premium rate SMS messages, track users' private data, or apps that, even if not characterized as malware, conduct questionable actions affecting the user's privacy or costing them money. In this paper, we investigate the feasibility of using both the permissions an app requests, the category of the app, and what permissions are requested by other apps in the same category to better inform users whether the risks of installing an app is commensurate with its expected benefit. Existing approaches consider only the risks of the permissions requested by an app and ignore both the benefits and what permissions are requested by other apps, thus having a limited effect. We propose several risk signals that and evaluate them using two datasets, one consists of 158,062 Android apps from the Android Market, and another consists of 121 malicious apps. We demonstrate the effectiveness of our proposal through extensive data analysis

**Advantages**

* The results of our analysis thousands of spamming accounts were shut down

**Disadvantages**

* We believe that these techniques can help social networks to improve their security and detect malicious users

**Using probabilistic generative models for ranking risks of Android AppsZ. H. Peng, et al**

**2012**

One of Android's main defence mechanisms against malicious apps is a risk communication mechanism which, before a user installs an app, warns the user about the permissions the app requires, trusting that the user will make the right decision. This approach has been shown to be ineffective as it presents the risk information of each app in a “stand-alone” fashion and in a way that requires too much technical knowledge and time to distil useful information. We introduce the notion of risk scoring and risk ranking for Android apps, to improve risk communication for Android apps, and identify three desiderata for an effective risk scoring scheme. We propose to use probabilistic generative models for risk scoring schemes, and identify several such models, ranging from the simple Naive Bayes, to advanced hierarchical mixture models. Experimental results conducted using real-world datasets show that probabilistic generative models significantly outperform existing approaches, and that Naive Bayes models give a promising risk scoring approach.

**Advantages:**

* We use real experiments to show that it is both scalable and robust

**Disadvantages:**

Data Loss

**Android Malware detection using parallel machine learning classifiers**

**S. Yerima, S. Sezer, and I. Muttik**

2014

Mobile malware has continued to grow at an alarming rate despite on-going mitigation efforts. This has been much more prevalent on Android due to being an open platform that is rapidly overtaking other competing platforms in the mobile smart devices market. Recently, a new generation of Android malware families has emerged with advanced evasion capabilities which make them much more difficult to detect using conventional methods. This paper proposes and investigates a parallel machine learning based classification approach for early detection of Android malware. Using real malware samples and benign applications, a composite classification model is developed from parallel combination of heterogeneous classifiers. The empirical evaluation of the model under different combination schemes demonstrates its efficacy and potential to improve detection accuracy. More importantly, by utilizing several classifiers with diverse characteristics, their strengths can be harnessed not only for enhanced Android malware detection but also quicker white box analysis by means of the more interpretable constituent classifiers.

**Advantages:**

We use real experiments to show that it is both scalable and robust

**Disadvantages:**

Does not have the real accuracy

**Opinion Fraud Detection in Online Reviews by Network Effects**

**L. Akoglu, R. Chandy, and C. Faloutsos**

**2013**

User-generated online reviews can play a significant role in the success of retail products, hotels, restaurants, etc. However, review systems are often targeted by opinion spammers who seek to distort the perceived quality of a product by creating fraudulent reviews. We propose a fast and effective framework, FRAUDEAGLE, for spotting fraudsters and fake reviews in online review datasets. Our method has several advantages: (1) it exploits the network effect among reviewers and products, unlike the vast majority of existing methods that focus on review text or behavioural analysis, (2) it consists of two complementary steps; scoring users and reviews for fraud detection, and grouping for visualization and sense making, (3) it operates in a completely unsupervised fashion requiring no labelled data, while still incorporating side information if available, and (4) it is scalable to large datasets as its run time grows linearly with network size. We demonstrate the effectiveness of our framework on synthetic and real datasets; where FRAUDEAGLE successfully reveals fraud-bots in a large online app review database. Copyright © 2013, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

**Advantages:**

Such a phenomenon motivates us to design new and more robust features to detect fraud

**Disadvantages:**

While graph-based features such as local clustering coefficient

**SYSTEM REQUIREMENTS**

**Software Requirements**

* O/S : Windows 7.
* Language : Java.
* IDE : NetBeans 8.2
* Data Base : MySQL

**Hardware Requirements**

* System : Pentium IV 2.4 GHz
* Hard Disk : 160 GB
* Monitor : 15 VGA color
* Mouse : Logitech.
* Keyboard : 110 keys enhanced
* Ram : 2GB

**SOFTWARE DESCRIPTION**

**Java**

Java is a programming language originally developed by James Gosling at Sun Microsystems (now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere." Java is currently one of the most popular programming languages in use, particularly for client-server web applications.

**Java Platform**

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java byte code, instead of directly to platform-specific machine code. Java byte code instructions are analogous to machine code, but are intended to be interpreted by a virtual machine (VM) written specifically for the host hardware.

End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java applets. Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

A major benefit of using byte code is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executables would. Just-in-Time compilers were introduced from an early stage that compiles byte codes to machine code during runtime.

Just as application servers such as Glass Fish provide lifecycle services to web applications, the Net Beans runtime container provides them to Swing applications. All new shortcuts should be registered in "Key maps/Net Beans" folder. Shortcuts installed INS Shortcuts folder will be added to all key maps, if there is no conflict. It means that if the same shortcut is mapped to different actions in Shortcut folder and current key map folder (like Key map/Net Beans), the Shortcuts folder mapping will be ignored.

* + Database Explorer Layer API in Database Explorer
  + Loaders-text-dB schema-Actions in Database Explorer
  + Loaders-text-sq.-Actions in Database Explorer
  + Plug-in Registration in Java EE Server Registry

The keyword public denotes that a method can be called from code in other classes, or that a class may be used by classes outside the class hierarchy. The class hierarchy is related to the name of the directory in which the .java file is located.

The keyword static in front of a method indicates a static method, which is associated only with the class and not with any specific instance of that class. Only static methods can be invoked without a reference to an object. Static methods cannot access any class members that are not also static. The keyword void indicates that the main method does not return any value to the caller. If a Java program is to exit with an error code, it must call System. Exit () explicitly.

The method name "main" is not a keyword in the Java language. It is simply the name of the method the Java launcher calls to pass control to the program. Java classes that run in managed environments such as applets and Enterprise JavaBeans do not use or need a main () method. A Java program may contain multiple classes that have main methods, which means that the VM needs to be explicitly told which class to launch from.

The Java launcher launches Java by loading a given class (specified on the command line or as an attribute in a JAR) and starting its public static void main(String[]) method. Stand-alone programs must declare this method explicitly. The String [] rags parameter is an array of String objects containing any arguments passed to the class. The parameters to main are often passed by means of a command line.

**Java a High-level Language**

A high-level programming language developed by Sun Microsystems. Java was originally called OAK, and was designed for handheld devices and set-top boxes. Oak was unsuccessful so in 1995 Sun changed the name to Java and modified the language to take advantage of the burgeoning World Wide Web.

Java source code files (files with a .java extension) are compiled into a format called byte code (files with a .class extension), which can then be executed by a Java interpreter. Compiled Java code can run on most computers because Java interpreters and runtime environments, known as Java Virtual Machines (VMs). Byte code can also be converted directly into machine language instructions by a just-in-time compiler (JIT).

Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Small Java applications are called Java applets and can be downloaded from a Web server and run on your computer by a Java-compatible Web browser, such as Netscape Navigator or Microsoft Internet Explorer.

Object-Oriented Software Development using Java: Principles, Patterns, and Frameworks contain a much applied focus that develops skills in designing software-particularly in writing well-designed, medium-sized object-oriented programs. It provides a broad and coherent coverage of object-oriented technology, including object-oriented modeling using the Unified Modeling Language (UML) object-oriented design using Design Patterns, and object-oriented programming using Java.

**NetBeans**

The **Net Beans Platform** is a reusable framework for simplifying the development of Java Swing desktop applications. The Net Beans IDE bundle for Java SE contains what is needed to start developing Net Beans plug-in and Net Beans Platform based applications; no additional SDK is required.

Applications can install modules dynamically. Any application can include the Update Center module to allow users of the application to download digitally-signed upgrades and new features directly into the running application.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are:

* User interface management (e.g. menus and toolbars)
* User settings management
* Storage management (saving and loading any kind of data)
* Window management
* Wizard framework (supports step-by-step dialogs)
* Net Beans Visual Library
* Integrated Development Tools

**J2EE**

A **Java EE application** or a **Java Platform, Enterprise Edition application** is any deployable unit of Java EE functionality. This can be a single Java EE module or a group of modules packaged into an EAR file along with a Java EE application deployment descriptor.

Enterprise applications can consist of the following:

* EJB modules (packaged in JAR files)
* Web modules (packaged in WAR files)
* connector modules or resource adapters (packaged in RAR files)
* Session Initiation Protocol (SIP) modules (packaged in SAR files)
* application client modules
* Additional JAR files containing dependent classes or other components required by the application

**Wamp Server**

**WAMP**s are packages of independently-created programs installed on computers that use a Microsoft Windows operating system.

Apache is a web server. MySQL is an open-source database. PHP is a scripting language that can manipulate information held in a database and generate web pages dynamically each time content is requested by a browser. Other programs may also be included in a package, such as phpMyAdmin which provides a graphical user interface for the MySQL database manager, or the alternative scripting languages Python or Perl.

**MySQL**

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

Free-software-open source projects that require a full-featured database management system often use MySQL. Applications which use MySQL databases include: TYPO3, Joomla, WordPress, hob, Drupal and other software built on the LAMP software stack.

**Platforms and interfaces**

Many programming languages with language-specific APIs include libraries for accessing MySQL databases. These include MySQL Connector/Net for integration with Microsoft's Visual Studio (languages such as C# and VB are most commonly used) and the JDBC driver for Java. In addition, an ODBC interface called Modoc allows additional programming languages that support the ODBC interface to communicate with a MySQL database, such as ASP or ColdFusion. The MySQL server and official libraries are mostly implemented in ANSI C/ANSI C++.

**FEASIBILITY STUDY**

The feasibility study is carried out to test whether the proposed system is worth being implemented. The proposed system will be selected if it is best enough in meeting the performance requirements.

The feasibility carried out mainly in three sections namely.

**•** Economic Feasibility

• Technical Feasibility

• Behavioural Feasibility

**Economic Feasibility**

Economic analysis is the most frequently used method for evaluating effectiveness of the proposed system. More commonly known as cost benefit analysis. This procedure determines the benefits and saving that are expected from the system of the proposed system. The hardware in system department if sufficient for system development.

**Technical Feasibility**

This study centre around the system’s department hardware, software and to what extend it can support the proposed system department is having the required hardware and software there is no question of increasing the cost of implementing the proposed system. The criteria, the proposed system is technically feasible and the proposed system can be developed with the existing facility.

**Behavioural Feasibility**

People are inherently resistant to change and need sufficient amount of training, which would result in lot of expenditure for the organization. The proposed system can generate reports with day-to-day information immediately at the user’s request, instead of getting a report, which doesn’t contain much detail.

**TESTING OF PRODUCT**

**Testing of Product**

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system.  System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved.  The candidate system is subject to variety of tests-on-line response, Volume Street, recovery and security and usability test.  A series of tests are performed before the system is ready for the user acceptance testing.  Any engineered product can be tested in one of the following ways.  Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational.  Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

**UNIT TESTING**

Unit testing is the testing of each module and the integration of the overall system is done.  Unit testing becomes verification efforts on the smallest unit of software design in the module.  This is also known as ‘module testing’.  The modules of the system are tested separately.  This testing is carried out during the programming itself.  In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module.  There are some validation checks for the fields.  For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included.  It is very easy to find error and debug the system.

**INTEGRATION TESTING**

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function.  Integrated testing is systematic testing that can be done with sample data.  The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

1. Top-down integration testing.
2. Bottom-up integration testing.

**WHITE BOX TESTING**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases.  Using the white box testing methods, we derived test cases that guarantee that all independent paths within a module have been exercised at least once.

**BLACK BOX TESTING**

* Black box testing is done to find incorrect or missing function
* Interface error
* Errors in external database access
* Performance errors
* Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’.  It tests the external behaviour of the system.  Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

**VALIDATION TESTING**

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many, but a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer.

# **USER ACCEPTANCE TESTING**

User acceptance of the system is the key factor for the success of the system.  The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

# **OUTPUT TESTING**

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format.  The output displayed or generated by the system under consideration.  Here the output format is considered in two ways.  One is screen and the other is printed format.  The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs.  For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

**System Implementation**

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended users and the operation of the system. The people are not sure that the software is meant to make their job easier.

* The active user must be aware of the benefits of using the system
* Their confidence in the software built up
* Proper guidance is impaired to the user so that he is comfortable in using the application

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not running on the server, the actual processes will not take place.

**User Training**

To achieve the objectives and benefits expected from the proposed system it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for education and training is more and more important. Education is complementary to training. It brings life to formal training by explaining the background to the resources for them. Education involves creating the right atmosphere and motivating user staff. Education information can make training more interesting and more understandable.

**Training on the Application Software**

After providing the necessary basic training on the computer awareness, the users will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design, type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the data entered. This training may be different across different user groups and across different levels of hierarchy.

**Operational Documentation**

Once the implementation plan is decided, it is essential that the user of the system is made familiar and comfortable with the environment. A documentation providing the whole operations of the system is being developed. Useful tips and guidance is given inside the application itself to the user. The system is developed user friendly so that the user can work the system from the tips given in the application itself.

**System Maintenance**

The maintenance phase of the software cycle is the time in which software performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is to make adaptable to the changes in the system environment. There may be social, technical and other environmental changes, which affect a system which is being implemented. Software product enhancements may involve providing new functional capabilities, improving user displays and mode of interaction, upgrading the performance characteristics of the system. So only thru proper system maintenance procedures, the system can be adapted to cope up with these changes. Software maintenance is of course, far more than “finding mistakes”.

**Corrective Maintenance**

The first maintenance activity occurs because it is unreasonable to assume that software testing will uncover all latent errors in a large software system. During the use of any large program, errors will occur and be reported to the developer. The process that includes the diagnosis and correction of one or more errors is called Corrective Maintenance.

**Adaptive Maintenance**

The second activity that contributes to a definition of maintenance occurs because of the rapid change that is encountered in every aspect of computing. Therefore Adaptive maintenance termed as an activity that modifies software to properly interfere with a changing environment is both necessary and commonplace.

**Perceptive Maintenance**

The third activity that may be applied to a definition of maintenance occurs when a software package is successful. As the software is used, recommendations for new capabilities, modifications to existing functions, and general enhancement are received from users. To satisfy requests in this category, Perceptive maintenance is performed. This activity accounts for the majority of all efforts expended on software maintenance.

**Preventive Maintenance**

The fourth maintenance activity occurs when software is changed to improve future maintainability or reliability, or to provide a better basis for future enhancements. Often called preventive maintenance, this activity is characterized by reverse engineering and re-engineering techniques.

**Types of Software Testing**

**Ad-hoc testing**

This type of software testing is very informal and unstructured and can be performed by any stakeholder with no reference to any test case or test design documents. The person performing Ad-hoc testing has a good understanding of the domain and workflows of the application to try to find defects and break the software. Ad-hoc testing is intended to find defects that were not found by existing test cases.

**Acceptance Testing**

Acceptance testing is a formal type of software testing that is performed by end user when the features have been delivered by developers. The aim of this testing is to check if the software confirms to their business needs and to the requirements provided earlier. Acceptance tests are normally documented at the beginning of the sprint (in agile) and is a means for testers and developers to work towards a common understanding and shared business domain knowledge.

**Accessibility Testing**

In accessibility testing, the aim of the testing is to determine if the contents of the website can be easily accessed by disable people. Various checks such as color and contrast (for color blind people), font size for visually impaired, clear and concise text that is easy to read and understand.

**Agile Testing**

Agile Testing is a type of software testing that accommodates agile software development approach and practices. In an Agile development environment, testing is an integral part of software development and is done along with coding. Agile testing allows incremental and iterative coding and testing.

**API Testing**

API testing is a type of testing that is similar to unit testing. Each of the Software APIs are tested as per API specification. API testing is mostly done by testing team unless APIs to be tested or complex and needs extensive coding. API testing requires understanding both API functionality and possessing good coding skills.

**Automated testing**

This is a testing approach that makes use of testing tools and/or programming to run the test cases using software or custom developed test utilities. Most of the automated tools provided capture and playback facility, however there are tools that require writing extensive scripting or programming to automate test cases.

**All Pairs testing**

Also known as Pair wise testing, is a black box testing approach and a testing method where in for each input is tested in pairs of inputs, which helps to test software works as expected with all possible input combinations.

**Beta Testing**

This is a formal type of software testing that is carried out by end customers before releasing or handing over software to end users. Successful completion of Beta testing means customer acceptance of the software.

**Black Box testing**

Black box testing is a software testing method where in testers are not required to know coding or internal structure of the software. Black box testing method relies on testing software with various inputs and validating results against expected output.

**Backward Compatibility Testing**

Type of software testing performed to check newer version of the software can work successfully installed over previous version of the software and newer version of the software works as fine with table structure, data structures, files that were created by previous version of the software.

**Boundary Value Testing (BVT)**

Boundary Value Testing is a testing technique that is based on concept “error aggregates at boundaries”. In this testing technique, testing is done extensively to check for defects at boundary conditions. If a field accepts value 1 to 100 then testing is done for values 0, 1, 2, 99, 100 and 101.

**Big Bang Integration testing**

This is one of the integration testing approaches, in Big Bang integration testing all or all most all of the modules are developed and then coupled together.

**Bottom up Integration testing**

Bottom up integration testing is an integration testing approach where in testing starts with smaller pieces or sub systems of the software till all the way up covering entire software system. Bottom up integration testing begins with smaller portion of the software and eventually scale up in terms of size, complexity and completeness.

**Branch Testing**

Is a white box testing method for designing test cases to test code for every branching condition? Branch testing method is applied during unit testing.

**Browser compatibility Testing**

It is one of the sub types of testing of compatibility testing performed by testing team. Browser compatibility testing is performed for web applications with combination of different browsers and operating systems.

**Compatibility testing**

Compatibility testing is one of the test types performed by testing team. Compatibility testing checks if the software can be run on different hardware, operating system, bandwidth, databases, web servers, application servers, hardware peripherals, emulators, different configuration, processor, different browsers and different versions of the browsers etc.

**Component Testing**

This type of software testing is performed by developers. Component testing is carried out after completing unit testing. Component testing involves testing a group of units as code together as a whole rather than testing individual functions, methods.

**Condition Coverage Testing**

Condition coverage testing is a testing technique used during unit testing, where in developer tests for all the condition statements like if, if else, case etc., in the code being unit tested.

**Dynamic Testing**

Testing can be performed as Static Testing and Dynamic testing, Dynamic testing is a testing approach where-in testing can be done only by executing code or software are classified as Dynamic Testing. Unit testing, Functional testing, regression testing, performance testing etc.

**Decision Coverage Testing**

Is a testing technique that is used in Unit testing, objective of decision coverage testing is to expertise and validate each and every decisions made in the code e.g. if, if else, case statements.

**End-to-end Testing**

End to end testing is performed by testing team, focus of end to end testing is to test end to end flows e.g. right from order creation till reporting or order creation till item return etc. and checking. End to end testing is usually focused mimicking real life scenarios and usage. End to end testing involves testing information flow across applications.

**Exploratory Testing**

Exploratory testing is an informal type of testing conducted to learn the software at the same time looking for errors or application behaviour that seems non-obvious. Exploratory testing is usually done by testers but can be done by other stake holders as well like Business Analysts, developers, end users etc. who are interested in learning functions of the software and at the same time looking for errors or behaviour is seems non-obvious.

**Equivalence Partitioning**

Equivalence partitioning is also known as Equivalence Class Partitioning is a software testing technique and not a type of testing by itself. Equivalence partitioning technique is used in black box and grey box testing types. Equivalence partitioning classifies test data into Equivalence classes as positive Equivalence classes and negative Equivalence classes, such classification ensures both positive and negative conditions are tested.

**Functional Testing**

Functional testing is a formal type of testing performed by testers. Functional testing focuses on testing software against design document, Use cases and requirements document. Functional testing is a black box type of testing and does not require internal working of the software unlike white box testing.

**Fuzz Testing**

Fuzz testing or fuzzing is a software testing technique that involves testing with unexpected or random inputs. Software is monitored for failures or error messages that are presented due to the input errors.

**GUI (Graphical User Interface) testing**

This type of software testing is aimed at testing the software GUI (Graphical User Interface) of the software meets the requirements as mentioned in the GUI mock-ups and Detailed designed documents. For e.g. checking the length and capacity of the input fields provided on the form, type of input field provided, e.g. some of the form fields can be displayed as dropdown box or a set of radio buttons. So GUI testing ensures GUI elements of the software are as per approved GUI mock-ups, detailed design documents and functional requirements. Most of the functional test automation tools work on GUI capture and playback capabilities. This makes script recording faster at the same time increases the effort on script maintenance.

**Glass box Testing**

Glass box testing is another name for White box testing. Glass box testing is a testing method that involves testing individual statements, functions etc., Unit testing is one of the Glass box testing methods.

**Gorilla Testing**

This type of software testing is done by software testing team, has a scary name though? Objective of Gorilla Testing is to exercise one or few functionality thoroughly or exhaustively by having multiple people test the same functionality.

**Happy Path Testing**

Also known as Golden path testing, this type of testing focuses on selective execution of tests that do not exercise the software for negative or error conditions.

**Integration Testing**

Integration testing also known as met in short, in one of the important types of software testing. Once the individual units or components are tested by developers as working then testing team will run tests that will test the connectivity among these units/component or multiple units/components. There are different approaches for Integration testing namely, Top-down integration testing, Bottom-up integration testing and a combination of these two known as Sand witch testing.

**Interface Testing**

Software provides support for one or more interfaces like “Graphical user interface”, “Command Line Interface” or “Application programming interface” to interact with its users or other software. Interfaces serves as medium for software to accept input from user and provide result. Approach for interface testing depends on the type of the interface being testing like GUI or API or CLI.

**Internationalization Testing**

Internationalization testing is a type of testing that is performed by software testing team to check the extent to which software can support Internationalization i.e., usage of different languages, different character sets, double byte characters etc., For e.g.: Gmail, is a web application that is used by people all over work with different languages, single by or multi byte character sets.

**Keyword-driven Testing**

Keyword driver testing is more of an automated software testing approach than a type of testing itself. Keyword driven testing is known as action driven testing or table driven testing.

**Load Testing**

Load testing is a type of non-functional testing; load testing is done to check the behaviour of the software under normal and over peak load conditions. Load testing is usually performed using automated testing tools. Load testing intends to find bottlenecks or issues that prevent software from performing as intended at its peak workloads.

**Localization Testing**

Localization testing a type of software testing performed by software testers, in this type of testing, software is expected to adapt to a particular locale, it should support a particular locale/language in terms of display, accepting input in that particular locale, display, font, date time, currency etc., related to a particular locale. For e.g. many web applications allow choice of locale like English, French, German or Japanese. So once locale is defined or set in the configuration of software, software is expected to work as expected with a set language/locale.

**Negative Testing**

This type of software testing approach, which calls out the “attitude to break”, these are functional and non-functional tests that are intended to break the software by entering incorrect data like incorrect date, time or string or upload binary file when text files supposed to be upload or enter huge text string for input fields etc. It is also a positive test for an error condition.

**Non-functional testing**

Software are built to fulfil functional and non-functional requirements, non-functional requirements like performance, usability, localization etc., There are many types of testing like compatibility testing, compliance testing, localization testing, usability testing, volume testing etc., that are carried out for checking non-functional requirements.

**Pair Testing**

**It** is a software testing technique that can be done by software testers, developers or Business analysts (BA). As the name suggests, two people are paired together, one to test and other to monitor and record test results. Pair testing can also be performed in combination of tester-developer, tester-business analyst or developer-business analyst combination. Combining testers and developers in pair testing helps to detect defects faster, identify root cause, fix and test the fix.

**Performance Testing**

**It** is a type of software testing and part of performance engineering that is performed to check some of the quality attributes of software like Stability, reliability, availability. Performance testing is carried out by performance engineering team. Unlike Functional testing, Performance testing is done to check non-functional requirements. Performance testing checks how well software works in anticipated and peak workloads. There are different variations or sub types of performance like load testing, stress testing, volume testing, soak testing and configuration testing.

**Penetration Testing**

**It** is a type of security testing, also known as pen test in short. Penetration testing is done to tests how secure software and its environments (Hardware, Operating system and network) are when subject to attack by an external or internal intruder. Intruder can be a human/hacker or malicious programs. Pen test uses methods to forcibly intrude (by brute force attack) or by using a weakness (vulnerability) to gain access to a software or data or hardware with an intent to expose ways to steal, manipulate or corrupt data, software files or configuration. Penetration Testing is a way of ethical hacking, an experienced Penetration tester will use the same methods and tools that a hacker would use but the intention of Penetration tester is to identify vulnerability and get them fixed before a real hacker or malicious program exploits it.

**Regression Testing**

**It** is a type of software testing that is carried out by software testers as functional regression tests and developers as Unit regression tests. Objective of regression tests are to find defects that got introduced to defect fix (is) or introduction of new feature(s). Regression tests are ideal candidate for automation.

**Retesting**

**It** is a type of retesting that is carried out by software testers as a part of defect fix verification. For e.g. a tester is verifying a defect fix and let us say that there are 3 test cases failed due to this defect. Once tester verifies defect fix as resolved, test will retest or test the same functionality again by executing the test cases that were failed earlier.

**Risk based Testing**

**It** is a type of software testing and a different approach towards testing a software. In Risk based testing, requirements and functionality of software to be tested are prioritized as Critical, High, Medium and low. In this approach, all critical and high priority tests are tested and them followed by Medium. Low priority or low risk functionality are tested at the end or may not base on the time available for testing.

**Smoke testing**

**It** is a type of testing that is carried out by software testers to check if the new build provided by development team is stable enough i.e., major functionality is working as expected in order to carry out further or detailed testing. Smoke testing is intended to find “show stopper” defects that can prevent testers from testing the application in detail. Smoke testing carried out for a build is also known as build verification test.

**Security Testing**

**It** is a type of software testing carried out by specialized team of software testers. Objective of security testing is to secure the software is to external or internal threats from humans and malicious programs. Security testing basically checks, how good is software’s authorization mechanism, how strong is authentication, how software maintains confidentiality of the data, how does the software maintain integrity of the data, what is the availability of the software in an event of an attack on the software by hackers and malicious programs is for Security testing requires good knowledge of application, technology, networking, security testing tools. With increasing number of web applications necessarily of security testing has increased to a greater extent.

**Sanity Testing**

**It** is a type of testing that is carried out mostly by testers and in some projects by developers as well. Sanity testing is a quick evaluation of the software, environment, network, external systems are up & running, software environment as a whole is stable enough to proceed with extensive testing. Sanity tests are narrow and most of the time sanity tests are not documented.

**Scalability Testing**

**It** is a non-functional test intended to test one of the software quality attributes i.e. “Scalability”. Scalability test is not focused on just one or few functionality of the software instead performance of software as a whole. Scalability testing is usually done by performance engineering team. Objective of scalability testing is to test the ability of the software to scale up with increased users, increased transactions, increase in database size etc., It is not necessary that software’s performance increases with increase in hardware configuration, scalability tests helps to find out how much more workload the software can support with expanding user base, transactions, data storage etc.,

**Stability Testing**

**It** is a non-functional test intended to test one of the software quality attributes i.e. “Stability”. Stability testing focuses on testing how stable software is when it is subject to loads at acceptable levels, peak loads, loads generated in spikes, with more volumes of data to be processed. Scalability testing will involve performing different types of performance tests like load testing, stress testing, spike testing, soak testing, spike testing etc…

**Static Testing** is a form of testing where in approaches like reviews, walkthroughs are employed to evaluate the correctness of the deliverable. In static testing software code is not executed instead it is reviewed for syntax, commenting, naming convention, size of the functions and methods etc. Static testing usually has check lists against which deliverables are evaluated. Static testing can be applied for requirements, designs, and test cases by using approaches like reviews or walkthroughs.

**Stress Testing** is a type of performance testing, in which software is subjected to peak loads and even to a break point to observe how the software would behave at breakpoint. Stress testing also tests the behavior of the software with insufficient resources like CPU, Memory, Network bandwidth, Disk space etc. Stress testing enables to check some of the quality attributes like robustness and reliability.

**CONCLUSION**

Since the variation of formats is high, it is appropriate to use machine learning based model rather than rule-based model. We closely compare our ML model with the rule-based model and we compare it with the external ML model.In inner test the performance of rule-based model is perfect as we expected because it has dedicated patterns in the same news sources whereas our ML model also gets over 97% on the basis of F1 score.

**CODING & SCREENSHOTS**

**CODING**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package repeated\_activities\_twitter;

import java.awt.HeadlessException;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Scanner;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

import net.proteanit.sql.DbUtils;

/\*\*

\*

\* @author egc

\*/

public class classification extends javax.swing.JFrame {

/\*\*

\* Creates new form classification

\*/

public static String a1,a2,a3,a4,a5;

public classification() {

initComponents();

this.setResizable(false);

this.setLocationRelativeTo(null);

}

public static float pfind(String a1,String a2,String a3,String a5 )

{

float ans = 0;

try{

Scanner sc = new Scanner(System.in);

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/twitter","root","");

Statement s = con.createStatement();

String query = null;

ResultSet rs = null;

int a=0 , b=0 , c=0 , d=0, total=0;

/\* Queries below are constructed using parameter values of Glucose,Blood\_Pressure,Skin\_Thickness,

Insulin,Diabetes\_Pedigree\_Function,Age

passed to function. Function finds probability for every individual parameter of provided class value

and using naive baye's theorem it calculates total probability \*/

query ="SELECT COUNT(\*) AS Expr1 FROM tweet\_label WHERE (Tweet\_ID = '"+ a1 + "' ) AND (label = '" +a5 +"') ";

s.execute(query);

rs= s.getResultSet();

if(rs.next())

a=Integer.parseInt(rs.getString(1));

// a = count of values in training set having Glucose , Outcome same as passed in argument

query ="SELECT COUNT(\*) AS Expr1 FROM tweet\_label WHERE ( User\_Name = '"+ a2 + "' ) AND (label = '" +a5 +"') ";

s.execute(query);

rs= s.getResultSet();

if(rs.next())

b=Integer.parseInt(rs.getString(1));

// b = count of values in training set having Blood\_Pressure , Outcome same as passed in argument

query ="SELECT COUNT(\*) AS Expr1 FROM tweet\_label WHERE ( Tweets = '"+ a3 + "' ) AND (label = '" +a5 +"') ";

s.execute(query);

rs= s.getResultSet();

if(rs.next())

c=Integer.parseInt(rs.getString(1));

// c = count of values in training set having Skin\_Thickness , Outcome as passed in argument

query ="SELECT COUNT(\*) AS Expr1 FROM tweet\_label WHERE (label = '" + a5 +"') ";

s.execute(query);

rs= s.getResultSet();

if(rs.next())

total=Integer.parseInt(rs.getString(1)); //total no resuults

ans = (float)a / (float)total \* (float)b /(float)total \* (float)c /(float)total ;

//calculating total probability by naive bayes

s.close();

con.close();

}

catch(Exception e)

{

System.out.println("Exception:"+ e);

}

return ans;

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jPanel2 = new javax.swing.JPanel();

jPanel3 = new javax.swing.JPanel();

jLabel30 = new javax.swing.JLabel();

jPanel4 = new javax.swing.JPanel();

jLabel6 = new javax.swing.JLabel();

jLabel7 = new javax.swing.JLabel();

jPanel12 = new javax.swing.JPanel();

jPanel11 = new javax.swing.JPanel();

jLabel24 = new javax.swing.JLabel();

jLabel25 = new javax.swing.JLabel();

jLabel27 = new javax.swing.JLabel();

jLabel28 = new javax.swing.JLabel();

jLabel29 = new javax.swing.JLabel();

jLabel31 = new javax.swing.JLabel();

jLabel32 = new javax.swing.JLabel();

jLabel33 = new javax.swing.JLabel();

jLabel34 = new javax.swing.JLabel();

jLabel35 = new javax.swing.JLabel();

jLabel36 = new javax.swing.JLabel();

jLabel37 = new javax.swing.JLabel();

jPanel6 = new javax.swing.JPanel();

jPanel8 = new javax.swing.JPanel();

jLabel20 = new javax.swing.JLabel();

jSeparator1 = new javax.swing.JSeparator();

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

jTable1 = new javax.swing.JTable();

jScrollPane2 = new javax.swing.JScrollPane();

jTable2 = new javax.swing.JTable();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jPanel1.setBackground(new java.awt.Color(204, 204, 204));

jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder());

jPanel2.setBorder(javax.swing.BorderFactory.createEtchedBorder());

jPanel3.setBackground(new java.awt.Color(0, 204, 255));

jLabel30.setBackground(new java.awt.Color(0, 204, 255));

jLabel30.setFont(new java.awt.Font("Cambria", 1, 14)); // NOI18N

jLabel30.setText("-> Next");

jLabel30.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel30.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel30MouseClicked(evt);

}

});

javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);

jPanel3.setLayout(jPanel3Layout);

jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel3Layout.createSequentialGroup()

.addGap(865, 865, 865)

.addComponent(jLabel30, javax.swing.GroupLayout.PREFERRED\_SIZE, 54, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel3Layout.createSequentialGroup()

.addGap(16, 16, 16)

.addComponent(jLabel30, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGap(15, 15, 15))

);

jPanel4.setBackground(new java.awt.Color(0, 204, 255));

jLabel6.setFont(new java.awt.Font("Cambria", 1, 22)); // NOI18N

jLabel6.setText("The contagion effects of repeated activation in social networks");

jLabel7.setFont(new java.awt.Font("Cambria", 1, 24)); // NOI18N

javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);

jPanel4.setLayout(jPanel4Layout);

jPanel4Layout.setHorizontalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel4Layout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel6)

.addGap(123, 123, 123)

.addComponent(jLabel7)

.addGap(29, 29, 29))

);

jPanel4Layout.setVerticalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel4Layout.createSequentialGroup()

.addGap(17, 17, 17)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel4Layout.createSequentialGroup()

.addGap(0, 11, Short.MAX\_VALUE)

.addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED\_SIZE, 37, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel4Layout.createSequentialGroup()

.addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED\_SIZE, 34, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, Short.MAX\_VALUE)))

.addContainerGap())

);

jPanel12.setBackground(new java.awt.Color(0, 204, 255));

jLabel24.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jLabel24.setText("Load Dataset");

jLabel24.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel24.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel24jLabel8MouseClicked(evt);

}

});

jLabel25.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jLabel25.setText("Data Preprocessing");

jLabel25.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel25.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel25jLabel9MouseClicked(evt);

}

});

jLabel27.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jLabel27.setText("Tokenization");

jLabel27.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel27.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel27jLabel10MouseClicked(evt);

}

});

jLabel28.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jLabel28.setText("Stop words Removal");

jLabel28.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel28.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel28jLabel11MouseClicked(evt);

}

});

jLabel29.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jLabel29.setText("Stemming");

jLabel29.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel29.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel29jLabel13MouseClicked(evt);

}

});

jLabel31.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jLabel31.setText("TF - IDF");

jLabel31.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel31.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel31jLabel14MouseClicked(evt);

}

});

jLabel32.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jLabel32.setText("Biterm Construction");

jLabel32.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel32.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel32jLabel15MouseClicked(evt);

}

});

jLabel33.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jLabel33.setText("Select Dataset");

jLabel33.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel33.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel33jLabel16MouseClicked(evt);

}

});

jLabel34.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jLabel34.setText("Clustering");

jLabel34.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel34.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel34jLabel17MouseClicked(evt);

}

});

jLabel35.setFont(new java.awt.Font("Cambria", 1, 14)); // NOI18N

jLabel35.setText("Classification");

jLabel35.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel35.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel35jLabel18MouseClicked(evt);

}

});

jLabel36.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jLabel36.setText("Analysis Report");

jLabel36.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel36.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel36jLabel19MouseClicked(evt);

}

});

jLabel37.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jLabel37.setText("POSTagging");

jLabel37.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel37.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel37jLabel12MouseClicked(evt);

}

});

javax.swing.GroupLayout jPanel11Layout = new javax.swing.GroupLayout(jPanel11);

jPanel11.setLayout(jPanel11Layout);

jPanel11Layout.setHorizontalGroup(

jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel11Layout.createSequentialGroup()

.addGap(22, 22, 22)

.addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel37, javax.swing.GroupLayout.PREFERRED\_SIZE, 168, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel36, javax.swing.GroupLayout.PREFERRED\_SIZE, 168, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel35, javax.swing.GroupLayout.PREFERRED\_SIZE, 168, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel33, javax.swing.GroupLayout.PREFERRED\_SIZE, 168, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel32, javax.swing.GroupLayout.PREFERRED\_SIZE, 168, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel29, javax.swing.GroupLayout.PREFERRED\_SIZE, 168, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel28, javax.swing.GroupLayout.PREFERRED\_SIZE, 168, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel27, javax.swing.GroupLayout.PREFERRED\_SIZE, 168, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel25, javax.swing.GroupLayout.PREFERRED\_SIZE, 168, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel24, javax.swing.GroupLayout.PREFERRED\_SIZE, 168, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel31)

.addComponent(jLabel34))

.addContainerGap(13, Short.MAX\_VALUE))

);

jPanel11Layout.setVerticalGroup(

jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel11Layout.createSequentialGroup()

.addGap(17, 17, 17)

.addComponent(jLabel33, javax.swing.GroupLayout.DEFAULT\_SIZE, 20, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jLabel24, javax.swing.GroupLayout.DEFAULT\_SIZE, 20, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jLabel25, javax.swing.GroupLayout.DEFAULT\_SIZE, 20, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jLabel27, javax.swing.GroupLayout.DEFAULT\_SIZE, 20, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jLabel37, javax.swing.GroupLayout.DEFAULT\_SIZE, 20, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jLabel28, javax.swing.GroupLayout.DEFAULT\_SIZE, 20, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jLabel29, javax.swing.GroupLayout.DEFAULT\_SIZE, 20, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jLabel32, javax.swing.GroupLayout.DEFAULT\_SIZE, 20, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jLabel31, javax.swing.GroupLayout.DEFAULT\_SIZE, 20, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jLabel34, javax.swing.GroupLayout.DEFAULT\_SIZE, 20, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jLabel35, javax.swing.GroupLayout.DEFAULT\_SIZE, 25, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jLabel36, javax.swing.GroupLayout.DEFAULT\_SIZE, 22, Short.MAX\_VALUE)

.addGap(23, 23, 23))

);

javax.swing.GroupLayout jPanel12Layout = new javax.swing.GroupLayout(jPanel12);

jPanel12.setLayout(jPanel12Layout);

jPanel12Layout.setHorizontalGroup(

jPanel12Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel12Layout.createSequentialGroup()

.addGap(20, 20, 20)

.addComponent(jPanel11, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(22, Short.MAX\_VALUE))

);

jPanel12Layout.setVerticalGroup(

jPanel12Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel12Layout.createSequentialGroup()

.addGap(21, 21, 21)

.addComponent(jPanel11, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(22, Short.MAX\_VALUE))

);

jPanel6.setBackground(new java.awt.Color(0, 204, 255));

jLabel20.setFont(new java.awt.Font("Cambria", 1, 18)); // NOI18N

jLabel20.setText("Classification");

jLabel20.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jLabel20.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel20MouseClicked(evt);

}

});

jButton1.setBackground(new java.awt.Color(0, 204, 255));

jButton1.setFont(new java.awt.Font("Cambria", 1, 14)); // NOI18N

jButton1.setText("Classification");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jButton2.setBackground(new java.awt.Color(0, 204, 255));

jButton2.setFont(new java.awt.Font("Cambria", 1, 14)); // NOI18N

jButton2.setText("View Classified Data");

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

jTable1.setBackground(new java.awt.Color(0, 204, 255));

jTable1.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jTable1.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

{null, null, null, null},

{null, null, null, null},

{null, null, null, null},

{null, null, null, null}

},

new String [] {

"Title 1", "Title 2", "Title 3", "Title 4"

}

));

jTable1.setRowHeight(20);

jScrollPane1.setViewportView(jTable1);

jTable2.setBackground(new java.awt.Color(0, 204, 255));

jTable2.setFont(new java.awt.Font("Cambria", 0, 14)); // NOI18N

jTable2.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

{null, null, null, null},

{null, null, null, null},

{null, null, null, null},

{null, null, null, null}

},

new String [] {

"Title 1", "Title 2", "Title 3", "Title 4"

}

));

jTable2.setRowHeight(20);

jScrollPane2.setViewportView(jTable2);

javax.swing.GroupLayout jPanel8Layout = new javax.swing.GroupLayout(jPanel8);

jPanel8.setLayout(jPanel8Layout);

jPanel8Layout.setHorizontalGroup(

jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel8Layout.createSequentialGroup()

.addGap(22, 22, 22)

.addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jLabel20, javax.swing.GroupLayout.PREFERRED\_SIZE, 168, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jSeparator1, javax.swing.GroupLayout.DEFAULT\_SIZE, 525, Short.MAX\_VALUE)

.addGroup(jPanel8Layout.createSequentialGroup()

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 247, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 247, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(jScrollPane1)

.addComponent(jScrollPane2, javax.swing.GroupLayout.Alignment.TRAILING))

.addContainerGap(22, Short.MAX\_VALUE))

);

jPanel8Layout.setVerticalGroup(

jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel8Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel20, javax.swing.GroupLayout.DEFAULT\_SIZE, 32, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED\_SIZE, 10, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(21, 21, 21)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 108, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 108, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(27, 27, 27))

);

javax.swing.GroupLayout jPanel6Layout = new javax.swing.GroupLayout(jPanel6);

jPanel6.setLayout(jPanel6Layout);

jPanel6Layout.setHorizontalGroup(

jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel6Layout.createSequentialGroup()

.addGap(19, 19, 19)

.addComponent(jPanel8, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(19, Short.MAX\_VALUE))

);

jPanel6Layout.setVerticalGroup(

jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel6Layout.createSequentialGroup()

.addGap(21, 21, 21)

.addComponent(jPanel8, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 916, Short.MAX\_VALUE)

.addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(21, 21, 21)

.addComponent(jPanel12, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(21, 21, 21)

.addComponent(jPanel6, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(22, Short.MAX\_VALUE))

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()

.addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(28, 28, 28)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jPanel12, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanel6, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 27, Short.MAX\_VALUE)

.addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

);

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(21, 21, 21)

.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(16, 16, 16)

.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(18, Short.MAX\_VALUE))

);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

private void jLabel30MouseClicked(java.awt.event.MouseEvent evt) {

try {

new classification().setVisible(false);

new analysis\_report().setVisible(true);

} catch (IOException ex) {

Logger.getLogger(clustering\_users.class.getName()).log(Level.SEVERE, null, ex);

}}

private void jLabel24jLabel8MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jLabel25jLabel9MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jLabel27jLabel10MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jLabel28jLabel11MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jLabel29jLabel13MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jLabel31jLabel14MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jLabel32jLabel15MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jLabel33jLabel16MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jLabel34jLabel17MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jLabel35jLabel18MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jLabel36jLabel19MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jLabel37jLabel12MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jLabel20MouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

BufferedReader br1=null;

File f=new File("./tweet.CSV");

if(f.exists())

f.delete();

try {

//Set a database connection

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/twitter","root","");

Statement stmt=con.createStatement();

// Delete all the records in a table load\_dataset

stmt.executeUpdate("TRUNCATE TABLE tweet\_label");

String s3=null;

int aa=1;

br1 = new BufferedReader(new FileReader("./Interesting\_Topics/"+"Top\_Interesting\_Topics"+".txt"));

while((s3=br1.readLine())!=null) //Compare each word with all the word that are presented int the form

{

System.out.println(""+s3);

BufferedReader br2=new BufferedReader(new FileReader("./Dataset/tweet.CSV"));

String s33=null;

while((s33=br2.readLine())!=null)

{

if(s33.contains(s3))

{

FileWriter fw=new FileWriter("./tweet.CSV",true);

fw.write(s33+","+"1");

fw.write("\r\n");

fw.close();

String values[]=s33.split(",");

String sql;

sql = "INSERT INTO tweet\_label(Tweet\_ID,User\_Name,Tweets,Tweet\_Created,label)VALUES('"+values[0]+"','"+values[1]+"','"+values[2]+"','"+values[3]+"','"+"1"+"')";

//Execute the query and update the table load\_dataset

stmt.executeUpdate(sql);

//Print the data

System.out.println(values[2]);

}

else

{

FileWriter fw=new FileWriter("./tweet.CSV",true);

fw.write(s33+","+"0");

fw.write("\r\n");

fw.close();

String values[]=s33.split(",");

String sql;

sql = "INSERT INTO tweet\_label(Tweet\_ID,User\_Name,Tweets,Tweet\_Created,label)VALUES('"+values[0]+"', '"+values[1]+"', '"+values[2]+"','"+values[3]+"','"+"0"+"')";

//Execute the query and update the table load\_dataset

stmt.executeUpdate(sql);

//Print the data

System.out.println(values[2]);

}

}

br2.close();

aa++;

}

br1.close();

} catch (FileNotFoundException ex) {

Logger.getLogger(classification.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(classification.class.getName()).log(Level.SEVERE, null, ex);

} catch (ClassNotFoundException ex) {

Logger.getLogger(classification.class.getName()).log(Level.SEVERE, null, ex);

} catch (SQLException ex) {

Logger.getLogger(classification.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

br1.close();

} catch (IOException ex) {

Logger.getLogger(classification.class.getName()).log(Level.SEVERE, null, ex);

}

}

try {

// TODO add your handling code here:

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/twitter","root","");

String sql = "select \* from tweet\_label";

PreparedStatement ps = con.prepareStatement(sql);

ResultSet rs = ps.executeQuery();

Statement stmt1=con.createStatement();

stmt1.executeUpdate("TRUNCATE TABLE classify1");

stmt1.executeUpdate("TRUNCATE TABLE classify2");

int a=0;

while(rs.next())

{

a1 = rs.getString("Tweet\_ID");

a2 = rs.getString("User\_Name");

a3 = rs.getString("Tweets");

a++;

if(a>=1000)

{

break;

}

/\*

Naive Bayes Classifier Program

Problem :- Classifying whether the person having the interest or not

Probability is calculated for interesting and and not interesting topics and accordingly prediction is made.

Test set has following fields on which prediction is done:-

1) Tweet ID

2) User Name

3) Tweets

Test set with above columns and values of column from options in bracket to be stored in database

class tells whether the patient have the diabetes or not.

When first 3 field values are supplied as input program makes prediction for as yes(Interesting Topics)

or no(Not an Interesting Topics).

\*/

try

{

/\*

c1 and c2 denote class 1 & class 2.

class1 :- Interesting Topics

class2 :- Not an Interesting Topics

\*/

Statement s = con.createStatement();

String query = null;

ResultSet rs1 = null;

int c1=0 ,c2=0 ,n=0;

query ="SELECT COUNT(\*) AS Expr1 FROM tweet\_label WHERE (label = '1') ";

s.execute(query);

rs1= s.getResultSet();

if(rs1.next())

//Count of cases when Person is interested

c1=Integer.parseInt(rs1.getString(1));

query ="SELECT COUNT(\*) AS Expr1 FROM tweet\_label WHERE (label= '0') ";

s.execute(query);

rs1= s.getResultSet();

if(rs1.next())

//Count of cases when person not interested

c2=Integer.parseInt(rs1.getString(1));

query = "SELECT COUNT(\*) AS Expr1 FROM tweet\_label ";

s.execute(query);

rs1= s.getResultSet();

if(rs1.next())

//Count of total cases in training set

n = Integer.parseInt(rs1.getString(1));

float pc1 = (float)c1/n; //General probability for class c1

float pc2 = (float)c2/n; //General probability for class c2

System.out.println("c1= " +c1 +"\nc2="+c2+"\ntotal="+n);

System.out.println("p(c1)="+pc1);

System.out.println("p(c2)="+pc2);

Scanner sc = new Scanner(System.in);

String a1,a2,a3,a4,a5;

// Accept the parameter values for which class is to be predicted

a1 = rs.getString("Tweet\_ID");

a2 = rs.getString("User\_Name");

a3 = rs.getString("Tweets");

float pinc1=0,pinc2=0;

//pinc1 = probability of prediction to be class1 (Interesting Topics)

//pinc2 = probability of prediction to be class2 (Not an Interesting Topics)

pinc1 = pfind(a1,a2,a3,"1");

pinc2 = pfind(a1,a2,a3,"0");

pinc1 = pinc1 \* pc1;

pinc2 = pinc2 \* pc2;

// compare pinc1 & pinc2 and predict the class that user will or not having the interest

if(pinc1 > pinc2){

System.out.println("Interested Topics");

sql = "INSERT INTO classify1(Tweet\_ID,User\_Name,Tweets,label)VALUES('"+a1+"', '"+a2+"', '"+a3+"','"+"1"+"')";

s.executeUpdate(sql);

}

else{

System.out.println("Not-Interested Topics");

sql = "INSERT INTO classify2(Tweet\_ID,User\_Name,Tweets,label)VALUES('"+a1+"', '"+a2+"', '"+a3+"','"+"0"+"')";

s.executeUpdate(sql);

}

//s.close();

//con.close();

}

catch(Exception e)

{

System.out.println(""+e);

}

}

} catch (SQLException ex) {

System.out.println(""+ex);

}

JOptionPane.showMessageDialog(null, "Classified Successfully");

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

try{

Class.forName("com.mysql.jdbc.Driver");

try (Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/twitter","root","")) {

String sql;

sql = "SELECT User\_Name,Tweets FROM classify1";

PreparedStatement ps=con.prepareStatement(sql);

ResultSet rs=ps.executeQuery();

jTable1.setModel(DbUtils.resultSetToTableModel(rs));

}

}

catch( ClassNotFoundException | SQLException e)

{

System.out.println(e);

}

try{

Class.forName("com.mysql.jdbc.Driver");

try (Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/twitter","root","")) {

String sql;

sql = "SELECT User\_Name,Tweets FROM classify2";

PreparedStatement ps=con.prepareStatement(sql);

ResultSet rs=ps.executeQuery();

jTable2.setModel(DbUtils.resultSetToTableModel(rs));

}

}

catch( ClassNotFoundException | SQLException e)

{

System.out.println(e);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(classification.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(classification.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(classification.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(classification.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new classification().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JLabel jLabel20;

private javax.swing.JLabel jLabel24;

private javax.swing.JLabel jLabel25;

private javax.swing.JLabel jLabel27;

private javax.swing.JLabel jLabel28;

private javax.swing.JLabel jLabel29;

private javax.swing.JLabel jLabel30;

private javax.swing.JLabel jLabel31;

private javax.swing.JLabel jLabel32;

private javax.swing.JLabel jLabel33;

private javax.swing.JLabel jLabel34;

private javax.swing.JLabel jLabel35;

private javax.swing.JLabel jLabel36;

private javax.swing.JLabel jLabel37;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel7;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel11;

private javax.swing.JPanel jPanel12;

private javax.swing.JPanel jPanel2;

private javax.swing.JPanel jPanel3;

private javax.swing.JPanel jPanel4;

private javax.swing.JPanel jPanel6;

private javax.swing.JPanel jPanel8;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane2;

private javax.swing.JSeparator jSeparator1;

private javax.swing.JTable jTable1;

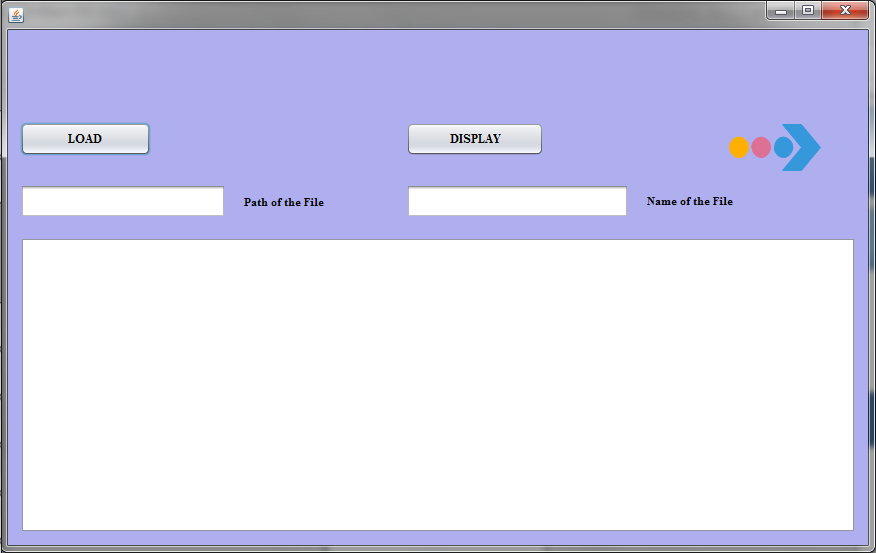
private javax.swing.JTable jTable2;

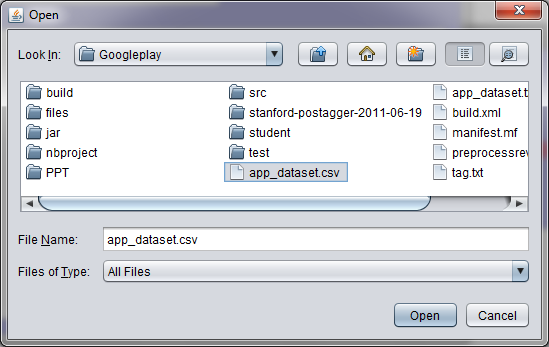
// End of variables declaration

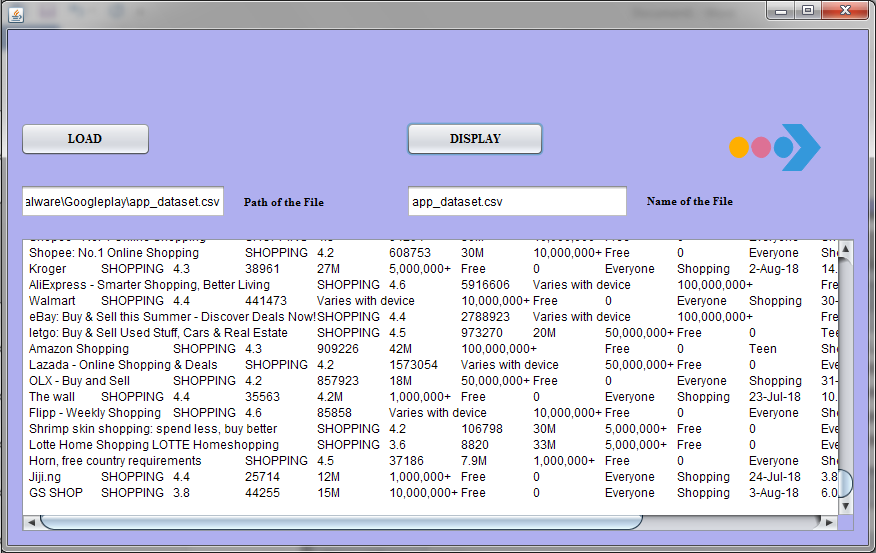
}

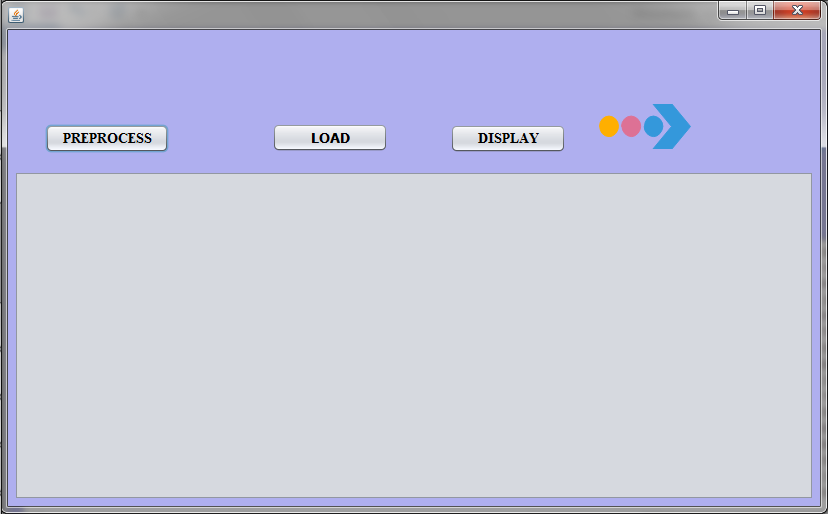
**SCREENSHOTS**

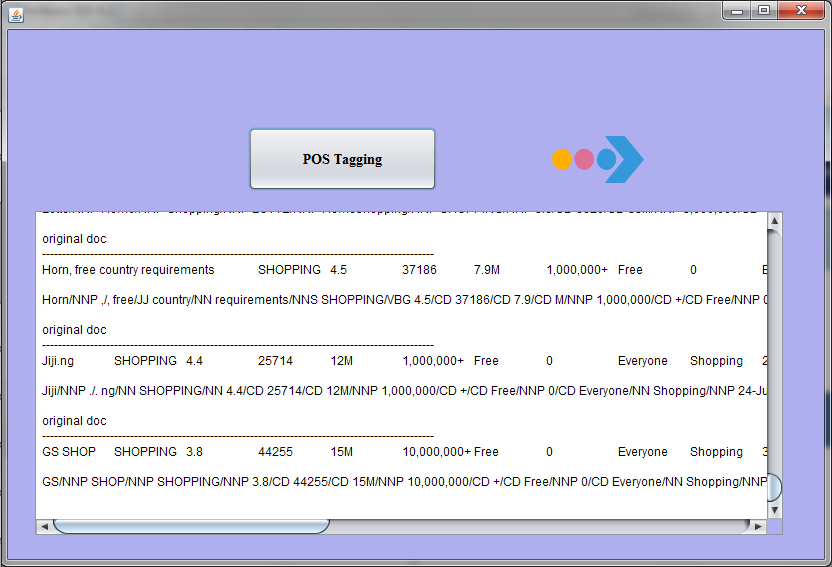
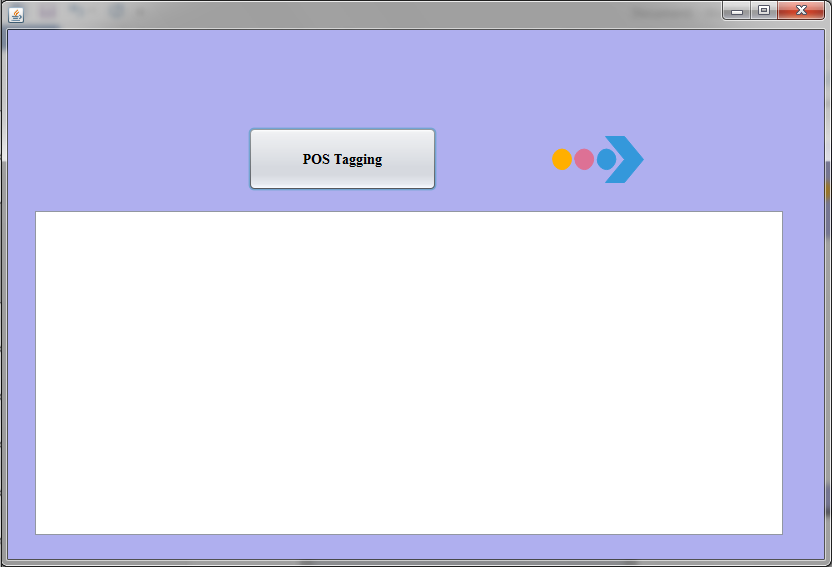
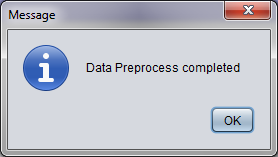
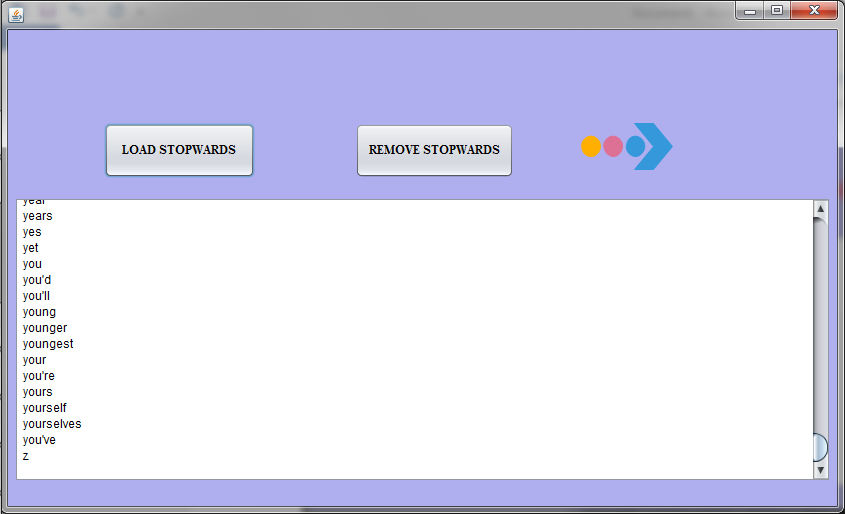
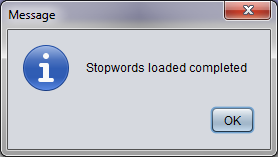
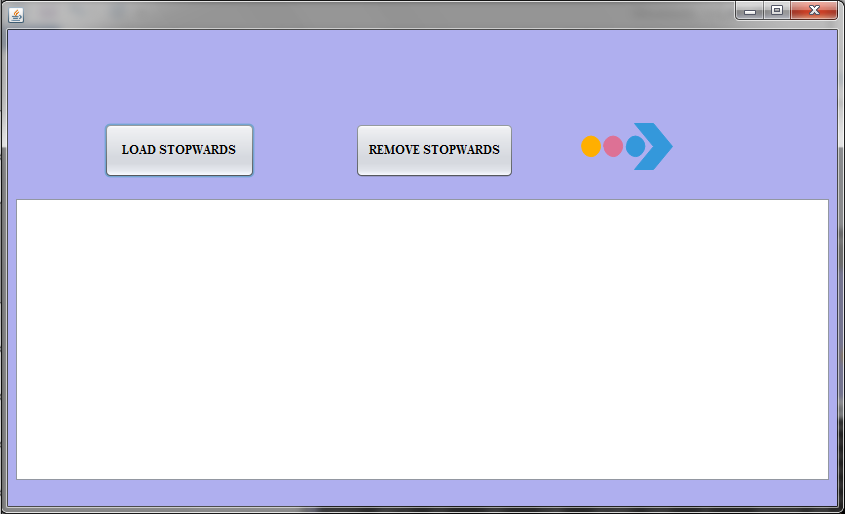
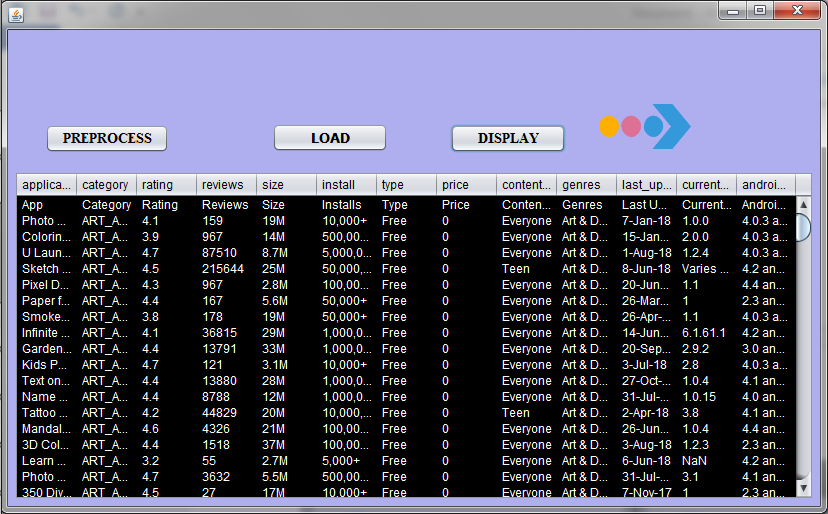
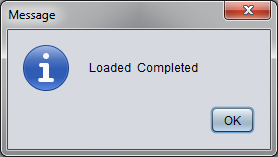
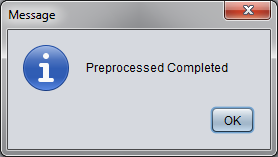


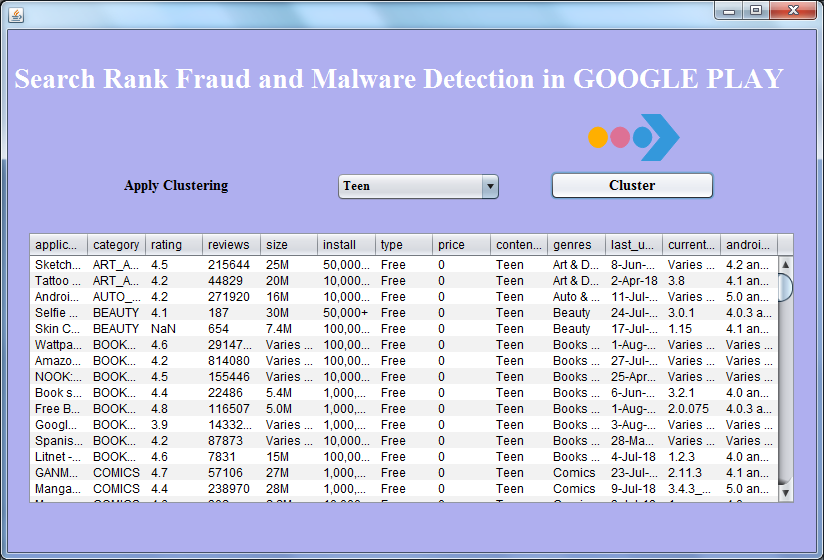


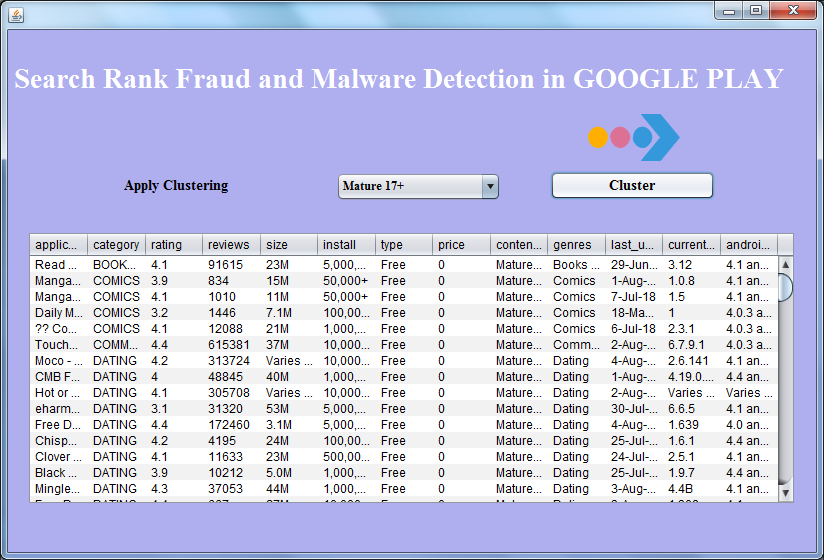
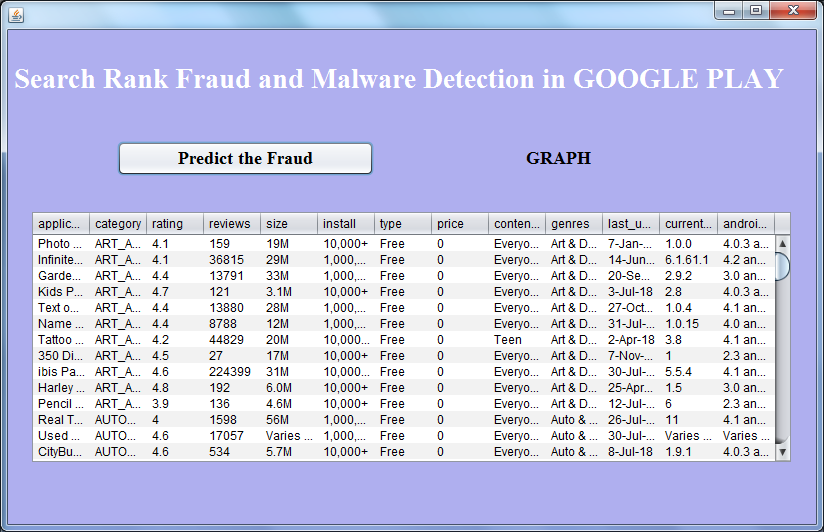
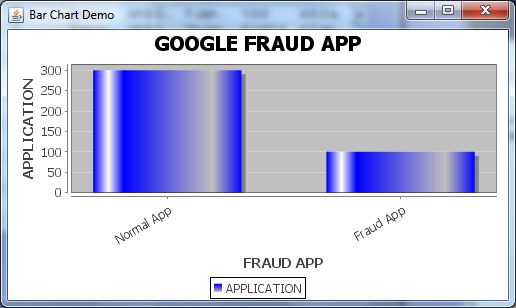










**REFERENCES**

* [1] Google Play. [Online]. Available: https://play.google.com/
* [2] E. Siegel, “Fake reviews in Google Play and Apple App Store,” Appentive, Seattle, WA, USA, 2014.
* [3] Z. Miners. (2014, Feb. 19). “Report: Malware-infected Android apps spike in the Google Play store,” PC World. Available: http:// www.pcworld.com/article/2099421/report-malwareinfectedandroid-apps-spike-in-the-google-play-store.html
* [4] S. Mlot. (2014, Apr. 8). “Top Android App a Scam, Pulled From Google Play,” PCMag. Available: http://www.pcmag.com/ article2/0,2817,2456165,00.asp
* [5] D. Roberts. (2015, Jul. 8). “How to spot fake apps on the Google Play store,” Fortune. Available: http://fortune.com/2015/07/08/ google-play-fake-app/
* [6] J. Sahs and L. Khan, “A machine learning approach to Android malware detection,” in Proc. Eur. Intell. Secur. Inf. Conf., 2012,pp. 141–147.
* [7] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, and G. Alvarez, “Puma: Permission usage to detect malware in android,” in Proc. Int. Joint Conf. CISIS12-ICEUTE’ 12-SOCO’Special Sessions, 2013, pp. 289–298.
* [8] J. Ye and L. Akoglu, “Discovering opinion spammer groups by network footprints,” in Machine Learning and Knowledge Discoveryin Databases. Berlin, Germany: Springer, 2015, pp. 267–282.
* [9] L. Akoglu, R. Chandy, and C. Faloutsos, “Opinion Fraud Detection in Online Reviews by Network Effects,” in Proc. 7th Int. AAAIConf. Web logs Soc. Media, 2013, pp. 2–11.
* [10] G. Wang, et al., “Serf and turf: Crowd turfing for fun and profit, ”in Proc. ACM WWW, 2012. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187928
* [11] J. Oberheide and C. Miller, “Dissecting the Android Bouncer, "presented at the SummerCon2012, New York, NY, USA, 2012.
* [12] Virus Total - free online virus, Malware and URL scanner.[Online]. Available: https://www.virustotal.com/, Last accessed on: May 2015.
* [13] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani , “Crow droid: Behavior-based Malware detection system for Android,” in Proc.ACM SPSM, 2011, pp. 15–26.
* [14] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss,“Andromaly: A behavioral malware detection framework for Android devices,” Intell. Inform. Syst., vol. 38, no. 1, pp. 161–190, 2012.
* [15] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, “Risk Ranker: Scalable and accurate zero-day Android malware detection,” inProc. ACM MobiSys, 2012, pp. 281–294.