A PROJECT REPORT ON

"USER RATINGS AND REVIEWS BASED FRAUD APP DETECTION IN GOOGLE PLAYSTORE"

Submitted in partial fulfilment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

Submitted by

SHAIK SADHULLA

Regd. No. 2251926030

Under the esteemed guidance of

Mr. P. RAMAKRISHNA RAO, M. Tech

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING

Dr. B.R. AMBEDKAR UNIVERSITY, SRIKAKULAM

2022-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING

Dr. B.R. AMBEDKAR UNIVERSITY, SRIKAKULAM

CERTIFICATE

This is to certify that the project entitled "**USER RATINGS AND REVIEWS BASED FRAUD APP DETECTION IN GOOGLE PLAYSTORE**" that is being submitted by **SHAIK SADHULLA (2251926030)** in partial fulfilmentof requirements for the award of the degree in **MASTER OF COMPUTERAPPLICATIONS** during 2022 - 2024, in **Dr. B. R. AMBEDKAR UNIVERSITY, SRIKAKULAM, COLLEGE OF ENGINEERING** is a record of bona fide work carried out by her under my guidance and super vision. The results embodied in this workhave not being submitted to any other university or institute for the award of any degree or diploma.

PROJECT SUPERVISOR
Mr. P.RAMAKRISHNA RAO, M.Tech
Assistant Professor

HEAD OF THE DEPARTMENT
Mr. R. SRIDHAR, M.Tech
Assistant Professor

Signature of the External Examiner

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of the project would be incomplete without the mention of the people who made it possible. I consider it my privilege to express my gratitude and respect to all those who guided and inspired in the successful completion of my project.

It is with great pleasure that, I acknowledgment my sincere thanks and deep sense of gratitude to my guide, **Dr. P. RAMAKRISHNA RAO, M.Tech . Assistant professor, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,** for his valuable guidance throughout the course of the project. No words will be adequate to quantify his support inspiration and coordination. His unflinching help, suggestions, directions and guidance have helped in progress of the project work. His professional attitude and human qualities is a source of my inspiration and model for me to follow.

I acknowledge my sincere gratitude to **Dr.R.SRIDHAR**,M.Tech Head of the, **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING Dr. B.R. AMBEDKAR UNIVERSITY, SRIKAKULAM,** for his encouragement and providing sufficient computational facilities to successfully complete the project work.

Further-more, I would also like to acknowledge my thankfulness with much appreciation the crucial role of our **TEACHING STAFF, NONTEACHING STAFF, PARENTS AND FRIENDS** for their love,

Support, encouragement and cooperation.

Place: SRIKAKULAM
Date:

SHAIK SADHULLA
Reg.No:2251926030

DECLARATION

I hereby declare that the project work entitled **"USER RATINGS AND REVIEWS BASED FRAUD APP DETECTION IN GOOGLE PLAYSTORE"** submitted by me for the award of the degree of **MASTER OF COMPUTER APPLICATIONS (MCA) under the guidance of Mr.P. RAMAKRISHNA RAO, M.Tech, Assistant Professor.** Dr. B.R. AMBEDKAR UNIVERSITY, SRIKAKULAM. Is original and it has not been submitted earlier.

Place: Srikakulam — SHAIK SADHULLA

Date — Regd.No:2251926030

ABSTRACT

The mobile industry is developing rapidly therefore the numbers of mobile applications are increasing day by day in the market. As there are many apps available in market users are in fuzzy state while downloading the apps for their use. Hence it is a need to keep track and develop a system to make sure the apps present are genuine or not. The objective is to develop a system in detecting fraud apps before the user downloads by User Ratings and Reviews analysis with Naive Bayes algorithm. So we are proposing a system which will process the information, ratings and three reviews of the application with Naive Bayes algorithm to give results. So it will be easier to decide fraud application.

INDEX

CHAPTER-4

CHAPTER-5

CHAPTER-6

CHAPTER-7

CHAPTER-8

CHAPTER-9

CHAPTER -10

LIST OF FIGURES

LIST OF SCREENS

CHAPTER-1

INTRODUCTION

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

Google play first releases its app in 2008.Since that it distributes applications to all the Android users. In Google Play Store, it provides services that user can discover the particular application, purchase those applications and install it on their mobile devices. Since Android is open source environment all the detail about the application users can be easily accessed by the application developers through Google play. In Google play 1.8 Million mobile applications are available and that is downloaded by over 25 billion users across the world. This leads to greater chance of installing malware to the applications that could affect user's mobile devices. Google play store uses its own security system known as Bouncer system to remove the malicious apps from its store. However, this method is not effective as testing some apps using virus tools many apps are found as malicious which are not detected by Bouncer system. Fraudulent developers use search ranking algorithm to promote their apps to the top while searching. After downloading mobile applications from Google play users are asked to give the ratings and reviews about that particular downloaded applications. However fraudulent developers give fake ratings and reviews about their application promote their application to the top. There are two typical approaches used for detecting malware in Google Play. Thus are Static and Dynamic. The dynamic approach needs apps to be run in a secure environment to detect its benign. The static approach is not used as the need to give all types of attack in early stage itself but that is impossible as everyday attackers find the new way to inject malware on applications.

1.2 EXISTING SYSTEM:

Many fraud app detection tools are available these days which extract evidences of ratings only to detect the fake apps with different approaches. Generally, ratings are between one to five, in this module we compute the average rating of particular app and compare it with threshold. The rating which are less than or equal to three are considered as negative ratings and rating above three are considered as positive ratings. Finally, the output is in the form of zeros and ones i.e. negative rating gives zero as an output while positive rating gives one as an output.

Disadvantage

- Sometimes the numeric rating has vast difference than the reviews given by the users.
- Low accuracy App developers are using tricky means frequently for increasing their Apps sales

1.3 PROPOSED SYSTEM:

Sometimes the numeric rating has vast difference than the reviews given by the users. To overcome this proposition is based on Naive Bayes algorithm. User Ratings and Reviews analysis is to help in determining the emotional tones behind words which are expressed in online. This method is helps to check for user's User Ratings and Reviews comments on selected applications. Using User Ratings and Reviews analysis the machine is able to learning from training data set and analyzes the sentiments, emotions about reviews and other texts. By using User Ratings and Reviews analysis analyzing reviews and comments can help to determine the fraud or not and determine the correct application for Android.

Advantages

- We build this work on the observation that fraudulent and malicious behaviors leave behind telltale signs on app markets.
- Fair Play achieves over 97% accuracy in classifying fraudulent and benign apps, and over 95% accuracy in classifying malware and benign apps.
- Fair Play significantly outperforms the malware indicators of Furthermore, we show that malware often engages in search rank fraud as well: When trained on fraudulent and benign apps, Fair Play flagged as fraudulent more than 75% of the gold standard malware apps
- Fair Play discovers hundreds of fraudulent apps.
- Fair Play also enabled us to discover a novel, *coercive review campaign* attack type, where app users are harassed into writing a positive review for the app, and install and review other apps

CHAPTER-2
LITERATURE SURVEY
INTRODUCTION

In this work the author proposes the static method to detect the malware in mobile applications. In this system using reverse engineering concept the source code for the suspicious APK files. After that using structured mapping author builds the structure for the classes. Finally using data flow concept several patterns for the different type of threats has been created and use them to detect the malware in applications. Depending upon the number of threading pattern the effectiveness of this method is calculated.

In this work the author proposed a new method to detect malware in mobile applications by examining the runtime behavior of that particular application in the mobile environment. The author proposes that unexpected behavior mobile app can vary from one application to other applications. Also, it varies from the environment of that particular application running on different devices. Using Exposed framework user can change the user and system application without modifying the application package (APK). Depend upon that user can set particular conditions to identify the malware in the mobile applications.

In this work the author proposes some of modern machine learning algorithms to detect malware. For that these algorithms are applied to the metadata collected from the Google Play. While all of the existing methods for detecting algorithm focused on inherent characteristics of the particular mobile app this gives a direct method to detect the applications. For the setup of the experiments the collected 25k data from Google Play. Developers update their applications in particular interval of days whereas fake applications could not be updated since its upload of the Google Play. All of these works focused on only linear models Future work may focus on non-liners models.

In this work the author aims to protect the review spanners or spam reviews. The spammer may target only on the specific protect. After that, they gave fake reviews to that particular mobile app by creating the different account to review that account. The author proposes a novel based scoring method to detect every single review of the

particular product. The author creates highly suspicious as a subset. By using webbased spammer evaluation software the fakeness of the review is calculated. After the completion of the evaluation, the result shows the effective to predict the fake reviews.

In work author proposed novel technique for computing a rank aggregation on the basis of matrix completion to avoid noise and incomplete data. Proposed method

solves a structured matrix-completion problem over the space of skew-symmetric matrices. The author proves a recovery theorem detailing when proposed approach will work. They also perform a detailed evaluation of proposed approach with synthetic data and an anecdotal study with Netflix ratings. To find the solutions, they utilized the sap solver for matrix completion. Rank aggregation is combined with the structure of skewsymmetric matrices. Author applied for latest advances in the theory and algorithms of matrix completion to skew-symmetric matrices. Author enhanced existing algorithm for matrix completion to handle skew symmetric data.

In this work author reported a survey on Web spam detection, which comprehensively introduces the principles and algorithms in the literature. Indeed, the work of Web ranking spam detection is mainly based on the analysis of ranking principles of search engines, such as Page Rank and query term frequency. This is different from ranking fraud detection for mobile Apps. They categorize all existing algorithms into three categories based on the type of information they use: contentbased methods, link-based methods, and methods based on nontraditional data such as user behaviour , clicks, HTTP sessions. In turn, there is a sub categorization of link-based category into five groups based on ideas and principles used: labels propagation, link pruning and reweighting, labels refinement, graph regularization, and feature based.

CHAPTER-3
SYSTEM ANALYSIS

3.1 INTRODUCTION

We propose Fair Play, a system that leverages to efficiently detect Google Play fraud and malware. Our major contributions are: To detect fraud and malware, we propose and generate relational, behavioral and linguistic features that we use to train supervised learning algorithms we formulate the notion of *co-review graphs* to model reviewing relations between users. We develop PCF, an efficient algorithm to identify temporally constrained, co-review pseudo-cliques — formed by reviewers with substantially overlapping co-reviewing activities across short time windows. We use temporal dimensions of review post times to identify suspicious review spikes received by apps; we show that to compensate for a negative review, for an app that has rating R, a fraudster needs to post at least positive reviews. We also identify apps with "unbalanced" review, rating and install counts, as well as apps with permission request ramps. We use linguistic and behavioral information to (i) detect genuine reviews from which we then (ii) extract user-identified fraud and malware indicators.

3.2 FEASIBILITY STUDY:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

3.2.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the Technologies used are freely available. Only the customized products had to be purchased.

3.2.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.2.3 OPERATIONAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the Final user of the system.

3.3 SYSTEM REQUIREMENTS SPECIFICATION

3.3.1 FUNCTIONAL REQUIREMENTS INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data

input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.
3. Create document, report, or other formats that contain information produced by the system.

3.3.2 NON-FUNCTIONAL REQUIREMENTS:

The major non-functional Requirements of the system are as follows

Deployment: On Android operating system, the system or project should be deployable.

Security: No one should be allowed to tamper with data from others; Enhanced Security Sensitive data.

Backup: There should avoid damaging the record. A simple backup function for the whole student attendance.

Data migration: There must be a simple way to student data from one system to migrate to a new system.

3.4 PERFORMANCE REQUIREMENTS (SDLC):

Structured project management techniques (such as an SDLC) enhance management's control over projects by dividing complex tasks into manageable sections. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management. In the proposed hypothetical model, the concept of user developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three-dimensional model which comprises of the user, owner and the developer. The one size Fits all‖ approach to applying SDLC methodologies is no longer appropriate. We have made an attempt to address the above-mentioned defects by using a new hypothetical model for SDLC described elsewhere. The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process that is, these issues are talked in the project management at the surface level but not at the ground level.

WHAT IS SDLC?

A software cycle deals with various parts and phases from planning to testing and deploying software. All these activities are carried out in different ways, as per the needs. Each way is known as a Software Development Lifecycle Model (SDLC). A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. A descriptive model describes the history of how a particular software system was developed. Descriptive models may be used as the basis for understanding and improving software development processes or for building empirically grounded prescriptive models.

SDLC models

The Linear model (Waterfall) - Separate and distinct phases of specification and development. - All activities in linear fashion. - Next phase starts only when first one is complete.

Evolutionary development - Specification and development are interleaved (Spiral, incremental, prototype based, Rapid Application development) Incremental Model (Waterfall in iteration), RAD (Rapid Application Development) **-** Focus is on developing quality product in less time.

Spiral Model - We start from smaller module and keeps on building it like a spiral. It is also called Component based development.

Formal systems development - A mathematical system model is formally transformed to an implementation.

Agile Methods - Inducing flexibility into development.

Reuse-based development - The system is assembled from existing components.

The General Model

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. There are tons of models, and many companies adopt their own, but all have very similar patterns. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced during implementation that is driven by the design. Testing verifies the deliverable of the implementation phase against requirements.

SDLC METHODOLOGY:

Spiral Model

The spiral model is similar to the incremental model, with more emphases placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. The baseline spiral, starting in the planning phase, requirements is gathered and risk is assessed. Each subsequent spirals builds on the baseline spiral. Requirements are gathered during the planning phase. In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. Software is produced in the engineering phase, along with testing at the end of the phase. The evaluation phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral. In the spiral model, the angular component represents progress, and the radius of the spiral represents cost. Spiral Life Cycle Model. This document plays a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the First model to discuss iterative development, but it was the First model to explain why the iteration models.

The steps for Spiral Model can be generalized as follows:

- A preliminary design is created for the new system.
- A First prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the Final product.
- A second prototype is evolved by a fourfold procedure:
- Evaluating the First prototype in terms of its strengths, weakness, and risks.
- Defining the requirements of the second prototype.
- Planning an designing the second prototype.
- Constructing and testing the second prototype.

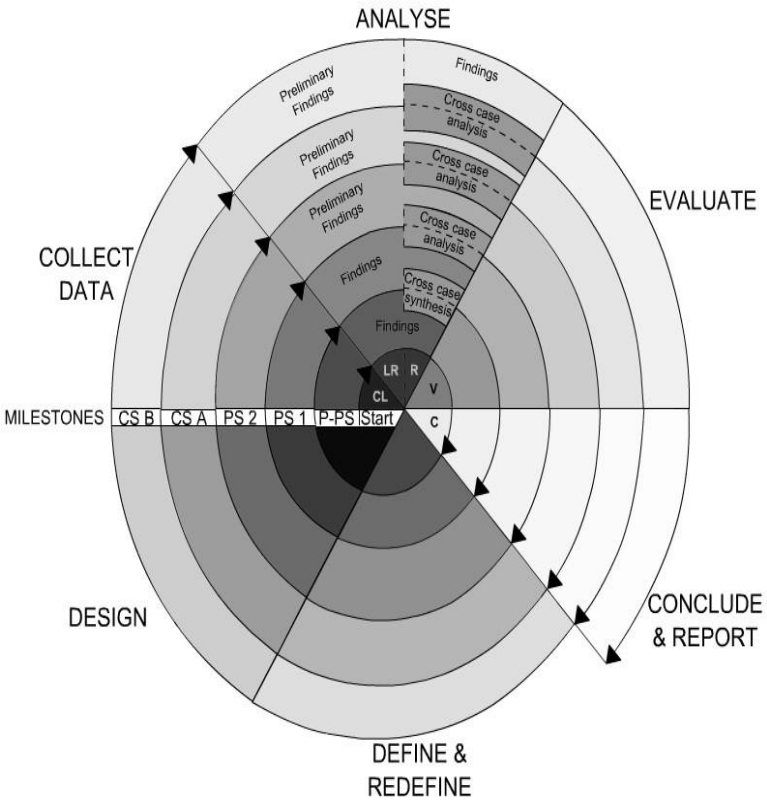


Fig.No:3.4.1 Spiral Model

Advantages

- High amount of risk analysis
- Good for large and mission-critical projects.
- Software is produced early in the software life cycle

Disadvantages

- Does not work well for smaller projects.
- It is not suitable for low risk projects. It is very costly model

3.5 H/S REQUIREMENTS

3.5.1 HARDWARE REQUIREMENTS

- System : 2.4 Hz.
- Hard Disk : 120 GB.
- RAM : 4 GB.

3.5.2 SOFTWARE REQUIREMENTS

- Operating System : Windows 11
- Language : Python 3.6.0
- Front end : HTML/CSS/XHTML

CHAPTER-4
SYSTEM DESIGN

4.1 INTRODUCTION

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software. The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. The purpose of the design phase is to plan a solution of the problem specified by the requirement document.

This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design. System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided. During, Detailed Design, the internal logic of each of the modules specified in system design is decided. During

will eventually be implemented. In system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules. In other works, in system design the attention is on what components are needed, while in detailed design how the components can be implemented in software is the issue. Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified.

4.2. MODULE DESCRIPTION

SYSTEM MODEL:

In the first module, of the project we develop the System environment model to evaluate the performance of our system for Search Rank Fraud. We focus on the Android app market ecosystem of Google Play. The participants, consisting of users and developers, ha e Google accounts. Developers create and upload apps that consist of executables (i.e., apks"), a set of required permissions, and a description. The app market publishes this information, along with the app's received reviews, ratings, aggregate rating (over both reviews and ratings), install count range, size, version number, price, time of last update, and a list of "similar" apps. Each review consists of a star rating ranging between 1-5 stars, and some text. The text is optional and consists of a title and a description. Google Play limits the number of reviews displayed for an app. In this module, we illustrate the participants in Google Play and their relations.

ADVERSARIAL MODEL:

In the second module, we develop the Adversarial model for considering the malicious users. We consider not only malicious developers, who upload malware, but also rational fraudulent developers. Fraudulent developers attempt to tamper with the search rank of their apps, e.g., by recruiting fraud experts in crowdsourcing sites to write reviews, post ratings, and create bogus installs. While Google keeps secret the criteria used to rank apps, the reviews, ratings and install counts are known to play a fundamental part.

To review or rate an app, a user needs to have a Google account, register a mobile device with that account, and install the app on the device. This process complicates the job of fraudsters, who are thus more likely to reuse accounts across jobs. The reason for search rank fraud attacks is impact. Apps that rank higher in search results, tend to

receive more installs. This is beneficial both for fraudulent developers, who increase their revenue, and malicious developers, who increase the impact of their malware.

THE CO-REVIEW GRAPH (COREG) MODULE

This module exploits the observation that fraudsters who control many accounts will re-use them across multiple jobs. Its goal is then to detect sub-sets of an app's reviewers that have performed significant common review activities in the past. In the following, we describe the co-review graph concept, formally present the weighted maximal clique enumeration problem, then introduce an efficient heuristic that leverages natural limitations in the behaviors of fraudsters. Let the co-review graph of an app, be a graph where nodes correspond to user accounts who reviewed the app, and undirected edges have a weight that indicates the number of apps reviewed in common by the edge's endpoint users. The co-review graph concept naturally identifies user accounts with significant past review activities.

REVIEWER FEEDBACK (RF) MODULE

Reviews written by genuine users of malware and fraudulent apps may describe negative experiences. The RF module exploits this observation through a two step approach: (i) detect and filter out fraudulent reviews, then (ii) identify malware and fraud indicative feedback from the remaining reviews

4.3 DATA FLOW DIAGRAMS

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. The Basic Notation used to create a DFD's are as follows:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

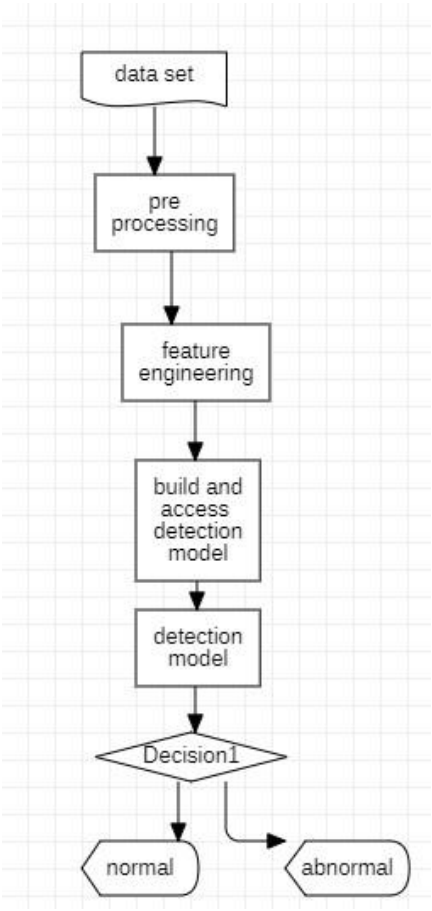


Fig.No:4.3.1 Context level Data flow Diagram

4.4. UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized generalpurpose modeling language in the Field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a

Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. 0.The unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

4.4.1 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. Use case diagrams model the functionality of a system using actors and use cases. Use cases are services or functions provided by the system to its users. Basic Use Case Diagram Symbols and Notations System Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.

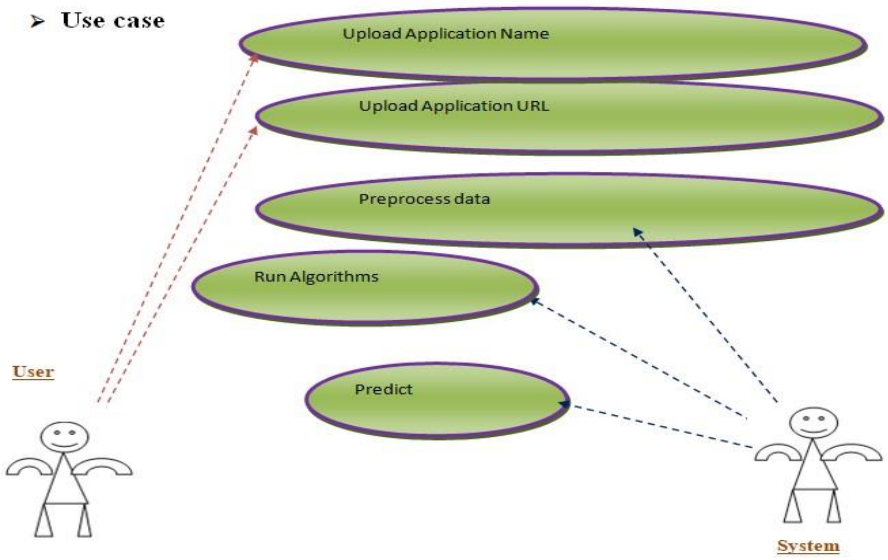


Fig.No:4.4.1 Use Case Diagram for User

4.4.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information. Class diagrams are the backbone of almost every object-oriented method including UML. They describe the static structure of a system. Basic Class Diagram Symbols and Notations Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition, and write operations into the third.

Class Name
attribute:Type = initialValue
operation(arg list):return type

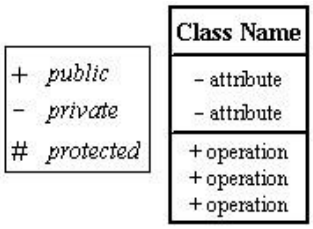
Active Class

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.

Visibility

Use visibility markers to signify who can access the information contained within a class. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class.

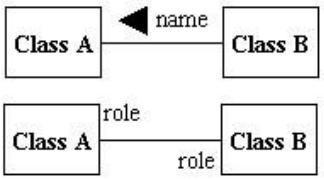
.



Associations

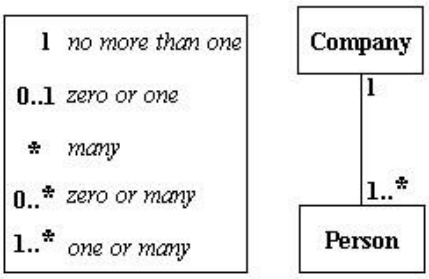
Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.

Note: It's uncommon to name both the association and the class roles.

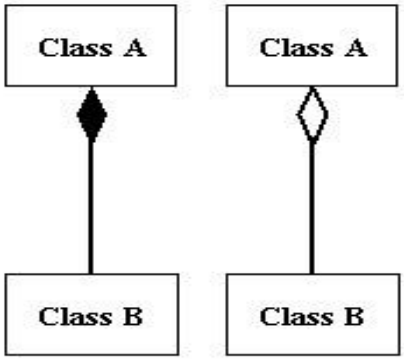


Multiplicity (Cardinality)

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.

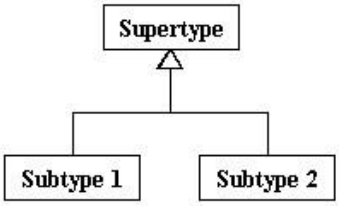


Constraint ship between Class A, the whole, and Class B, its part. Illustrate composition with a filled diamond. Use a hollow diamond to represent a simple aggregation relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other. The diamond end in both a composition and aggregation relationship points toward the "whole" class or the aggregate



Generalization

Generalization is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another. For example, Honda is a type of car. So the class Honda would have a generalization relationship with the class car.



In real life coding examples, the difference between inheritance and aggregation can be confusing. If you have an aggregation relationship, the aggregate (the whole) can access only the PUBLIC functions of the part class. On the other hand, inheritance allows the inheriting class to access both the PUBLIC and PROTECTED functions of the super class.

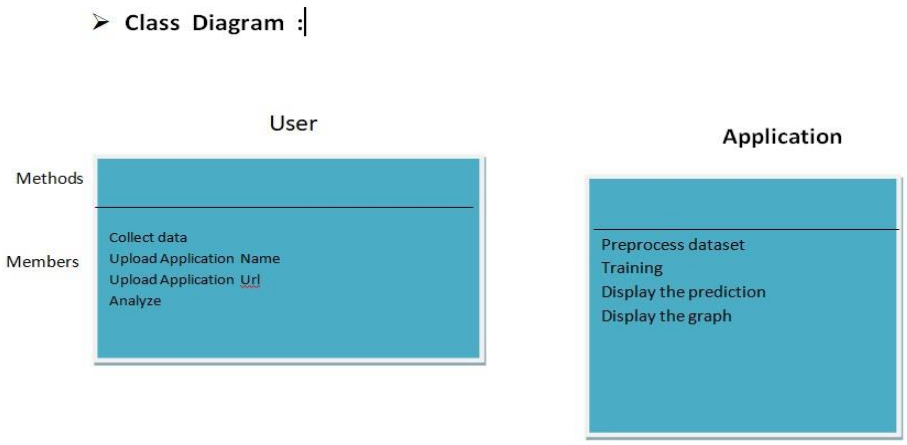


Fig.No:4.4.2 Class Diagram for USER

4.4.3 SEQUECE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It I s a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

Actors – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the

Lifelines –A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram. The standard in UML for naming a lifeline follows the following format – Instance Name: Class Name. Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

Basic Sequence Diagram Symbols and Notations Class roles Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Loops A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].

➢ **Sequence Diagram**

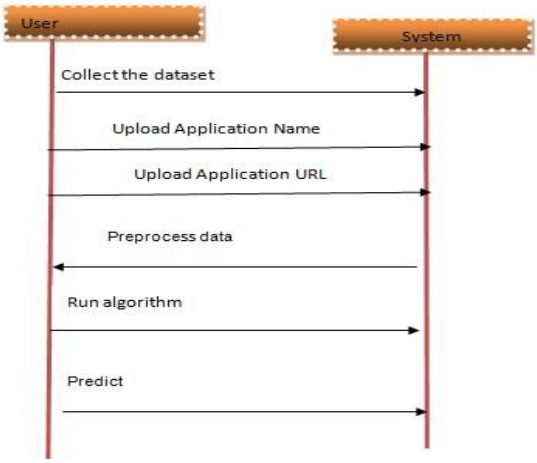


Fig.No:4.4.3 Sequence Diagram for User and System

4.4.4 ACTIVITY DIAGRAM FOR TPAU

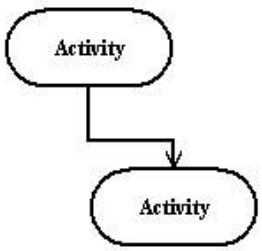
Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step work flows of components in a system. An activity diagram shows the overall flow of control.

Initial State – The starting state before an activity takes place is depicted using the initial state. for initial state or start state A process can have only one initial state unless we are depicting nested activities. We use a black Filled circle to depict the initial state of a system. For objects, this is the state when they are instantiated. The Initial State from the UML Activity Diagram marks the entry point and the initial Activity State. For example – Here the initial state is

Action states Action states represent the non-interruptible actions of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.

Action Flow

Action flow arrows illustrate the relationships among action states.



Object Flow

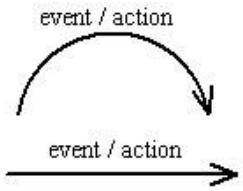
Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.

A state chart diagram shows the behavior of classes in response to external stimuli. This diagram models the dynamic flow of control from state to state within a system. Basic State chart Diagram Symbols and Notations

States: States represent situations during the life of an object. You can easily illustrate a state in Smart Draw by using a rectangle with rounded corners.

Transition

A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it.

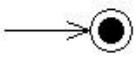


Initial State

A filled circle followed by an arrow represents the object's initial state.

Final State

An arrow pointing to a filled circle nested inside another circle represents the object's final state.



Synchronization and Splitting of Control

A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.

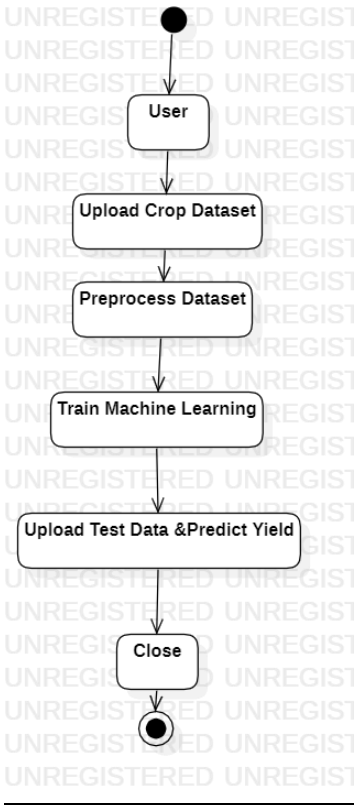


Fig.No:4.4.4 Activity Diagram for USER

CHAPTER-5
IMPLEMENTATION

5.1 FRAME WORK

In this paper, I propose a neural embedding-based apps similarity detection framework that allows to identify counterfeit apps from a large corpus of apps represented by their icons and text descriptions. We leverage the recent advances in Convolutional Neural Networks (CNNs) to generate feature embeddings from given images using pretrained models such as Alex Net, Vignette, and Reset. However, in contrast to commonly used content embeddings generated from fully connected layers before the last soft-max layer, we show that combining content embeddings with style embeddings generated from the Gram matrix of convolutional layer feature maps of the same pre-trained models achieve better results in detecting visually similar app icons.

- We show that the novel method of using combined style and content embeddings generated from pretrained CNNs outperforms many baseline image retrieval methods including hashing, feature-based, and structural similarity-based methods such as SIFT, SURF, and SSIM, for the task of detecting visually similar app icons. We also validate this method using two standard image retrieval datasets; Holidays dataset and Bench, and show that neural embeddings also perform better than baseline hashing and feature-based methods. Using a large dataset of over 1.2 million app icons, we show that combined content and style embeddings achieve percent higher precision and 14–18 percent higher recall when k 2 f5; 10g.
- We show that adding text embeddings generated using the app description as an additional modality for similarity search, further increases the performance by 3–5 percent and 6–7 percent in terms of precision and recalk, respectively when k 2 f5; 10g.
- We identify a set of 7,246 potential counterfeits (that are similar both visually as well as functionality wise) to the top-10,000 popular apps in Google Play and show that under a conservative assumption, 2,040 of them contain malware. We further show that 1,565 potential counterfeits ask for at least five additional dangerous permissions than the original app and 1,407 potential counterfeits have at least five extra third party advertisement libraries. To the best of our knowledge this is the first large-scale study that investigates the depth of app counterfeit problem in app stores.

5.2 ALGORITHM

In this Algorithm the first step, we represent the apps with three types of neural embeddings and their combinations; content, style, and text. In the second step, given an app icon, we retrieve nearest neighbors calculated based on distances between different embedding's with the expectation that if there are any counterfeits to the query app, those will be retrieved in the nearest neighbor search. For the rest of this section, we discuss details of the app embedding generation and how we do the nearest neighbor search for various scenarios At We encode the original app icon image of size 300 300 3 to a lower dimension for efficient search as well as to avoid false positives happening at large dimensions. We create two types of low dimensional neural representations of the images. For this we use a pre-trained VGG19. a state-of-the-art CNN that is trained on ImageNet dataset. VGG19 consists of five convolutional blocks followed by two fully connected layers. All convolutional layers use 3x3 kernels as the receptive field and convolution stride is fixed to 1 to preserve spatial resolution after convolution. The numbers of filters used in the five blocks are 64, 128, 256, 512, and 512 respectively. Each block has multiple convolutional layers followed by a max pooling layer which reduces the dimensions of the input.

The two fully connected layers consist of 4,096 neurons and followed by the prediction layer with 1,000 classes. All layers use Rectified Linear Unit (ReLU) activations while the prediction layer uses SoftMax activation. i) Content Embeddings. To extract the content representations, we fed all 1.2M app icons to the VGG19, and used the content embeddings, C 2 R4096, generated at the last fully connected layer of VGG19 that have shown good results in the past

Style Embeddings. As mentioned in Section 1, content similarity itself is not sufficient for counterfeit detection since sometimes developers keep the visual similarity and change the content. For example, if a developer is trying to create a fake app for a popular game that has birds as characters, she can create a game that has the same visual "look and feel" and replace birds with cats. Therefore, we require an embedding that represents the style of an image. Several work demonstrated that the filter responses (also known as feature maps) of convolutional neural networks can be used to represent the style of an image For example, Gaits et al. used pre-trained convolutional neural networks to transfer the style characteristics of an arbitrary source image to an arbitrary target image. This was done by defining an objective function which captures the content loss and the style loss. To represent the style of an image,

authors used the Gram matrix of the filter responses of the convolution layers. We followed a similar approach and used the fifth convolution layer (specifically conv5 1) of the VGG19 to obtain the style representation of the image, as previous comparable work indicated that conv5 1 provides better performance in classifying artistic styles [44]. In the process of getting the embeddings for icons, each icon is passed through the VGGNet, and at conv5 1 the icon is convolved with pre-trained filters and activated through ReLU.

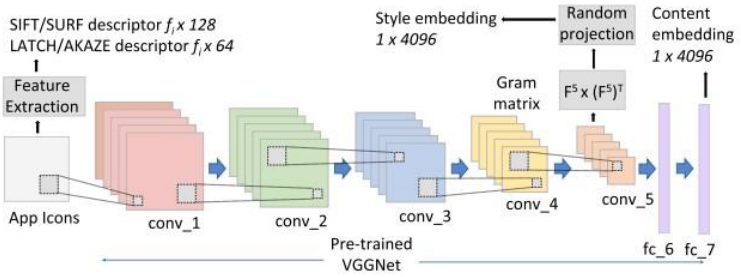


Fig.No: 5.2. Summary of the image encoding/embeddings generation methodology

5.3 PROCEDURE

Using a dataset of over 1.2 million app icons, text descriptions, and their executables, we presented the problem of counterfeits in mobile app markets. We proposed an icon encoding method that allows to efficiently search potential counterfeits to a given app, using neural embeddings generated by a state-of-the-art convolutional neural network. More specifically, for app counterfeit detection problem, we showed that content and style neural embeddings generated from pre-trained VGG-Net significantly outperforms hashing and feature-based image retrieval methods. We also showed that adding the text embeddings generated from the app descriptions further increase counterfeit detection rates. To investigate the depth of the app counterfeit problem in Google Play Store, we used our multi-modal embedding methodology to retrieve potential counterfeits for the top10,000 popular apps and investigated the possible inclusion of malware, permission usage, and embedded third party advertisement libraries. We found that 2,040 potential counterfeits we retrieved were marked by at least five commercial antivirus tools as malware, 1,565 asked for at least five additional dangerous permissions, and 1,407 had at least five additional embedded third party advertisement libraries.

5.4 MATHEMATICAL PROOF

Several works demonstrated that the filter responses (also known as feature maps) of convolutional neural networks can be used to represent the style of an image. For example, Gaits et al. used pre-trained convolutional neural networks to transfer the style characteristics of an arbitrary source image to an arbitrary target image. This was done by defining an objective function which captures the content loss and the style loss. To represent the style of an image, authors used the Gram matrix of the filter

responses of the convolution layers. We followed a similar approach and used the fifth convolution layer (specifically conv5 1) of the VGG19 to obtain the style representation of the image, as previous comparable work indicated that conv5 1 provides better performance in classifying artistic styles. In the process of getting the embeddings for icons, each icon is passed through the VGGNet, and at conv5 1 the icon is convolved with pre-Trai pre-trained filters and activated through ReLU activation function. More specifically, for an image I, let Fl 2 RNlMl be the filter response of layer l, where Nl denotes the number of filters in layer l and Ml is the height times width of the feature map. Fl ij is the activation of it filter at position j in the layer. Similar to Gayts et al. [39], to capture style information dot product. That is, for a given image I, let Gl 2 RNlNl be the dot product Gram matrix at layer l, i.e.,

$$G^l_{ij} = \sum_{k=1}^{M_l} F^l_{ik} F^l_{jk},$$

where Fl 2 RNlMl is the activation of I. Then, Gl is used as the style representation of an image to retrieve similar images. The conv5 1 layer of the VGGNet we use has 512 filters, thus the resulting Gram matrix is of size G5 2 R512512.

More specifically, let $\mathbf{A} \in \mathbb{R}^{n \times D}$ be our style matrix that contains the style embeddings of a mini batch of n (in the experiments we used n=20,000) icons stacked vertically, and D is the dimension of the style vector, which in this case is 131,328. Then, we create a sparse random matrix $\mathbf{R} \in \mathbb{R}^{D \times k}$ and multiply it with A. The elements r_{ij} of $\mathbf{R}$ are drawn according to the below distribution

$$r_{ij} = \sqrt[4]{D} \begin{cases} 1, & \text{with prob. } \frac{1}{2\sqrt{D}} \\ 0, & \text{with prob. } 1 - \frac{1}{\sqrt{D}} \\ -1, & \text{with prob. } \frac{1}{2\sqrt{D}} \end{cases}. \quad (2)$$

$$\mathbf{B} = \frac{1}{\sqrt{k}} \mathbf{AR} \in \mathbb{R}^{n \times k}.$$

5.5 SCHEME

Counterfeit apps impersonate existing popular apps in attempts to misguide users to install them for various reasons such as collecting personal information, spreading malware, or simply to increase their advertisement revenue. Many counterfeits can be identified once installed, however even a tech-savvy user may struggle to detect them before installation as app icons and descriptions can be quite similar to the original app. To this end, this paper proposes to leverage the recent advances in deep learning methods to create image and text embeddings so that

counterfeit apps can be efficiently identified when they are submitted to be published in app markets. We show that for the problem of counterfeit detection, a novel approach of combining content embeddings and style embeddings (given by the Gram matrix of CNN feature maps) outperforms the baseline methods for image similarity such as SIFT, SURF, LATCH, and various image hashing methods. We first evaluate the performance of the proposed method on two well-known datasets for evaluating image similarity methods and show that, content, style, and combined embeddings increase precision and recall by 10- 15 percent and 12-25 percent, respectively when retrieving five nearest neighbors. Second specifically for the app counterfeit detection problem, combined content and style embeddings achieve 12 and 14 percent increase in precision and recall, respectively compared to the baseline methods. We also show that adding text embeddings further increases the performance by 5 and 6 percent in terms of precision and recall, respectively when k is five. Third, we present an analysis of approximately 1.2 million apps from Google Play Store and identify a set of potential counterfeits for top-10,000 popular apps. Under a conservative assumption, we were able to find 2,040 potential counterfeits that contain malware in a set of 49,608 apps that showed high similarity to one of the top-10,000 popular apps in Google Play Store. We also find 1,565 potential counterfeits asking for at least five additional dangerous permissions than the original app and 1,407 potential counterfeits having at least five extra third party advertisement libraries.

5.6 SOFTWARE ENVIRONMENT

Python is a general-purpose interpreted, interactive, object-oriented, and highlevel programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. Python, the reference implementation of Python, is open source software and has a communitybased development model, as do nearly all of its variant implementations. Python is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple

programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library **It is used for:**

- WEB DEVELOPMENT (SERVER-SIDE),
- SOFTWARE DEVELOPMENT,
- MATHEMATICS,
- SYSTEM SCRIPTING.

What can Python do

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics. Why Python
- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- **Python Syntax compared to other programming languages**

Python was designed for readability, and has some similarities to the English language with influence from mathematics. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses. Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

C:\Users\Your Name>python --version

To check if you have python installed on a Linux or Mac, then on Linux open the command line or on Mac open the Terminal and type: python --version

If you find that you do not have python installed on your computer, then you can download it for free from the following website: https://www.python.org/

Python QuickStart

Python is an interpreted programming language, this means that as a developer you write Python (.Py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

C:\Users\Your Name>python helloworld.py

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py print

("Hello, World!")

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

C:\Users\Your Name>python helloworld.py The

output should read:

Hello, World!

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

C:\Users\Your Name>python

Or, if the "python" command did not work, you can try "py":

C:\Users\Your Name>py

From there you can write any python, including our hello world example from earlier in the tutorial:

C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>print ("Hello, World!")

Which will write "Hello, World!" in the command line:

C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>print ("Hello, World!") Hello,

World!

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface: exit ()

Virtual Environments and Packages

Introduction

Python applications will often use packages and modules that don't come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library's interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages. Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A's environment.

Creating Virtual Environments

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the Ven module as a script with the directory path: python3 -m venv tutorial-env This will create the tutorial-env directory if it doesn't exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you've created a virtual environment, you may activate it.

On Windows, run: tutorial-

env\Scripts\activate.bat On Unix

or MacOS, run:

source tutorial-env/bin/activate

(This script is written for the bash shell. If you use the cash or fish shells, there are alternate *activates* and activate. Fish scripts you should use instead.)

Activating the virtual environment will change your shell's prompt to show what virtual environment you're using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

$ source ~/envy/tutorial-env/bin/activate

You can install, upgrade, and remove packages using a program called pip. By default pip will install packages from the Python Package Index, <https://pypi.org>. You can browse the Python Package Index by going to it in your web browser, or you can use pip's limited search feature: (tutorial-env) $ pip search astronomy pip has a number of subcommands: "search", "install", "uninstall", "freeze", etc. (Consult the Installing

Python Modules guide for complete documentation for pip.) You can install the latest version of a package by specifying a package's name:

```
(Tutorial-env) $ pip install Novas
Collecting Novas
  Downloading novas-3.1.1.3.tar.gz (136kB)
Installing collected packages: Novas
  Running setup.py install for Novas
Successfully installed novas-3.1.1.3
```

You can also install a specific version of a package by giving the package name followed by == and the version number:

```
(Tutorial-env) $ pip install requests==2.6.0
Collecting requests==2.6.0
  Using cached requests-2.6.0-py2.py3-none-any.whl
Installing collected packages: requests
Successfully installed requests-2.6.0
```

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install --upgrade to upgrade the package to the latest version:

```
(Tutorial-env) $ pip install --upgrade requests
Collecting requests
Installing collected packages: requests
Found existing installation: requests 2.6.0
Uninstalling requests-2.6.0:
    Successfully uninstalled requests-2.6.0 Successfully
installed requests-2.7.0
```

pip uninstall followed by one or more package names will remove the packages from the virtual environment.

pip show will display information about a particular package:

```
(Tutorial-env) $ pip show requests
---
```

Metadata-Version: 2.0

Name: requests

Version: 2.7.0

Summary: Python HTTP for Humans.

Home-page: http://python-requests.org

Author: Kenneth Reitz

Author-email: me@kennethreitz.com
License: Apache 2.0

Location: /Users/aching/envy/tutorial-env/lib/python3.4/site-packages Requires:

pip list will display all of the packages installed in the virtual environment:

(Tutorial-env) $ pip list

Novas (3.1.1.3)

NumPy (1.9.2) pip

(7.0.3) requests

(2.7.0) setup tools

(16.0)

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

(Tutorial-env) $ pip freeze > requirements.txt

(Tutorial-env) $ cat requirements.txt

Novas==3.1.1.3 NumPy==1.9.2

requests==2.7.0

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

(Tutorial-env) $ pip install -r requirements.txt

Collecting Novas==3.1.1.3 (from -r requirements.txt (line 1))

Collecting NumPy==1.9.2 (from -r requirements.txt (line 2))

Collecting requests==2.7.0 (from -r requirements.txt (line 3))

Installing collected packages: Novas, NumPy, requests

Running setup.py install for Novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you've written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.

Cross Platform

Platform. Architecture (executable=sys.executable, bits='', linkage='') Queries the given executable (defaults to the Python interpreter binary) for various architecture information. Returns a tuple (bits, linkage) which contain information about the bit architecture and the linkage format used for the executable. Both values are returned as strings.

Values that cannot be determined are returned as given by the parameter presets. If bits is given as '', the size of(pointer) (or size of(long) on Python version < 1.5.2) is used as indicator for the supported pointer size.

The function relies on the system's file command to do the actual work. This is available on most if not all Unix platforms and some non-Unix platforms and then only if the executable points to the Python interpreter. Reasonable defaults are used when the above needs are not met.

Note On Mac OS X (and perhaps other platforms), executable files may be universal files containing multiple architectures.

To get at the "64-bitness" of the current interpreter, it is more reliable to query the sys.maxsize attribute: is_64bits = sys.maxsize> 2**32 platform. Machine () Returns the machine type, e.g., 'i386'. An empty string is returned if the value cannot be determined. platform. Node () Returns the computer's network name (may not be fully qualified!). An empty string is returned if the value cannot be determined. platform. Platform (aliased=0, terse=0) Returns a single string identifying the underlying platform with as much useful information as possible. The output is intended to be human readable rather than machine parse able. It may look different on different platforms and this is intended.

If aliased is true, the function will use aliases for various platforms that report system names which differ from their common names, for example SunOS will be reported as Solaris. The system alias () function is used to implement this. Setting terse to true causes the function to return only the absolute minimum information needed to identify the platform. platform. Processor ()

Returns the (real) processor name, e.g. 'amdk6'.

An empty string is returned if the value cannot be determined. Note that many platforms do not provide this information or simply return the same value as for machine ().

NetBSD does this.

platform. python build ()

Returns a tuple (build no, build date) stating the Python build number and date as strings. platform. python_ compiler ()

Returns a string identifying the compiler used for compiling Python.

platform. python_ branch ()

Returns a string identifying the Python implementation SCM branch.

New in version 2.6. platform.

python_ implementation ()

Returns a string identifying the Python implementation. Possible return values are:

‘Python’, ‘Iron Python’, ‘Jython’, ‘PyPy’.

New in version 2.6.

Platform. python_ revision ()

Returns a string identifying the Python implementation SCM revision.

New in version 2.6.

Platform. python_ version ()

Returns the Python version as string 'major. minor. Patch level'.

Note that unlike the Python Severson, the returned value will always include the patch level (it defaults to 0).

platform. python_ version_ tuple ()

Returns the Python version as tuple (major, minor, patch level) of strings.

Note that unlike the Python Severson, the returned value will always include the patch level (it defaults to '0'). platform. Release ()

Returns the system's release, e.g. '2.2.0' or 'NT' An empty string is returned if the value cannot be determined. platform. System ()

Returns the system/OS name, e.g. 'Linux', 'Windows', or 'Java'. An empty string is returned if the value cannot be determined.

platform. system_ alias (system, release, version)

Returns (system, release, version) aliased to common marketing names used for some systems. It also does some reordering of the information in some cases where it would otherwise cause confusion. platform. version ()

Note This function works best with Mark Hammond's win32all package installed, but also on Python 2.3 and later (support for this was added in Python 2.6). It obviously only runs on Win32 compatible platforms. Win95/98 specific platform. popes(cmd, mode='r', bufsize=None) Portable popen() interface. Find a working popen implementation preferring win32pipe.popen(). On Windows NT, win32pipe.popen() should work; on Windows 9x it hangs due to bugs in the MS C library.

Using the Python Interpreter

Invoking the Interpreter

The Python interpreter is usually installed as /user/local/bin/python3.8 on those machines where it is available; putting /user/local/bin in your Unix shell's search path makes it possible to start it by typing the command:

python3.8

to the shell. 1 Since the choice of the directory where the interpreter lives is an installation option, other places are possible; check with your local Python guru or system administrator. (E.g., /user/local/python is a popular alternative location.)

On Windows machines where you have installed Python from the Microsoft Store, the python3.8 command will be available. If you have the py.exe launcher installed, you can use the Py command. See Excursus: Setting environment variables for other ways to launch Python.

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero-exit status. If that doesn't work, you can exit the interpreter by typing the following command: quit ().

The interpreter's line-editing features include interactive editing, history substitution and code completion on systems that support the GNU ReadiLine library. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. If nothing appears to happen, or if ^P is echoed, command line editing isn't available; you'll only be able to use backspace to remove characters from the current line.

The interpreter operates somewhat like the Unix shell: when called with standard input connected to a try device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in command, analogous to the shell's -c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote command in its entirety with single quotes.

Some Python modules are also useful as scripts. These can be invoked using python m module [arg] ..., which executes the source file for module as if you had spelled out its full name on the command line. When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

All command line options are described in Command line and environment.

Argument Passing

When known to the interpreter, the script name and additional arguments thereafter are turned into a list of strings and assigned to the argv variable in the sys module. You can access this list by executing import sys. The length of the list is at least one; when no script and no arguments are given, sys. rag is an empty string. When the script name is given as '-' (meaning standard input), sys. rag is set to '-'. When -c command is used, sys. rag is set to '-c'. When -m module is used, sys. rag is set to the full name of the located module. Options found after -c command or -m module are not consumed by the Python interpreter's option processing but left in sys. argv for the command or module to handle.

Interactive Mode

When commands are read from a try, the interpreter is said to be in interactive mode. In this mode it prompts for the next command with the primary prompt, usually three greater-than signs (>>>); for continuation lines it prompts with the secondary prompt, by default three dots (...). The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

$ python3.8

Python 3.8 (default, Sep 16 2015, 09:25:04)

[GCC 4.8.2] on Linux

Type "help", "copyright", "credits" or "license" for more information.

>>>

Continuation lines are needed when entering a multi-line construct. As an example, take a look at this if statement:

>>>

>>>the_world_is_flat = True >>>ifthe_world_is_flat:

... print ("Be careful not to fall off!") ...Be

careful not to fall off!

For more on interactive mode, see Interactive Mode.

The Interpreter and Its Environment

Source Code Encoding

By default, Python source files are treated as encoded in UTF-8. In that encoding, characters of most languages in the world can be used simultaneously in string literals, identifiers and comments — although the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, your editor must recognize that the file is UTF-8, and it must use a font that supports all the characters in the file.

To declare an encoding other than the default one, a special comment line should be added as the first line of the file. The syntax is as follows:

-*- coding: encoding -*- where encoding is one of the valid

codecs supported by Python.

For example, to declare that Windows-1252 encoding is to be used, the first line of your source code file should be:

-*- coding: cp1252 -*-

One exception to the first line rule is when the source code starts with a UNIX "shebang" line. In this case, the encoding declaration should be added as the second line of the file. For example:

Introduction to Artificial Intelligence

"The science and engineering of making intelligent machines, especially intelligent computer programs". -John McCarthy- Artificial Intelligence is an approach to make a computer, a robot, or a product to think how smart human think. AI is a study of how human brain think, learn, decide and work, when it tries to solve problems. And finally this study outputs intelligent software systems. The aim of AI is to improve computer functions which are related to human knowledge, for example, reasoning, learning, and problem-solving. · Gaming − AI plays important role for machine to think of large number of possible positions based on deep knowledge in strategic games. for example, chess, river crossing, N-queens' problems and etc.

Natural Language Processing − Interact with the computer that understands natural language spoken by humans. · Expert Systems − Machine or software provide explanation and advice to the users. · Vision Systems − Systems understand, explain, and describe visual input on the computer. · Speech Recognition − There are some AI based speech recognition systems have ability to hear and express as sentences and understand their meanings while a person talks to it. For example Siri and Google assistant. · Handwriting Recognition − The handwriting recognition software reads the text written on paper and recognize the shapes of the letters and convert it into editable text. · Intelligent Robots − Robots are able to perform the instructions given by a human. **Major Goals**

- Knowledge reasoning
- Planning
- Machine Learning
- Natural Language Processing
- Computer Vision
- Robotics

"Watson" is an IBM supercomputer that combines Artificial Intelligence (AI) and complex inquisitive programming for ideal execution as a "question answering" machine. The supercomputer is named for IBM's founder, Thomas J. Watson.

IBM Watson is at the forefront of the new era of computing. At the point when IBM Watson made, IBM communicated that "more than 100 particular techniques are used to inspect perceive sources, find and make theories, find and score affirm, and combination and rank speculations." recently, the Watson limits have been expanded and the way by which Watson works has been changed to abuse new sending models (Watson on IBM Cloud) and propelled machine learning capacities and upgraded hardware open to architects and authorities. It isn't any longer completely a request answering figuring system arranged from Q&A joins yet can now 'see', 'hear', 'read', 'talk', 'taste', 'translate', 'learn' and 'endorse'.

Machine Learning Introduction

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs. Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Selfdriving cars that rely on machine learning to navigate may soon be available to consumers.

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Two of the most widely adopted machine learning methods are supervised learning which trains algorithms based on example input and output data that is labeled by humans, and unsupervised learning which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let's explore these methods in more detail.

Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to "learn" by comparing its actual output with the "taught" outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data.

For example, with supervised learning, an algorithm may be fed data with images of sharks labeled as `fish` and images of oceans labeled as `water`. By being trained on this data, the supervised learning algorithm should be able to later identify unlabeled shark images as `fish` and unlabeled ocean images as `water`.

A common use case of supervised learning is to use historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails. In supervised learning, tagged photos of dogs can be used as input data to classify untagged photos of dogs.

Unsupervised Learning

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable. The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data. Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this

audience in order to increase their number of purchases. Without being told a "correct" answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together.

NumPy: NumPy is a general-purpose array-processing package. It provides a highperformance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities Besides its obvious scientific uses, Numpy can also be used as an efficient multidimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

PANDAS: Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP. Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs. Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something.

5.7 SOURCE CODE

```
import matplotlib.pyplot as plt
from tkinter import * import
threading import os
from google_play_scraper import app, Sort,  reviews import
pandas as pd
from  sklearn.model_selection  import  train_test_split  from
sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

appNameList=[] url=[]

class MyWindow:     def
__init__(self, win):
self.appNameList=[]
        self.url=[]
        self.lbl1=Label(win, text='Application Name')
self.lbl1['fg']="blue"
        self.lbl2=Label(win, text='Application Url')
        self.lbl2['fg']="blue"
self.lbl3=Label(win, text='RESULT')
self.lbl4=Label(win, text='')
self.t1=Entry(bd=3,width=80)
self.t2=Entry(width=80)          self.t3 =
Text(win, height=20, width=60)
self.t3.pack()
        self.btn1 = Button(win, text='Analysis')
self.lbl1.place(x=100, y=50)
self.t1.place(x=200, y=50)
self.lbl2.place(x=100, y=100)
self.t2.place(x=200, y=100)
        self.b1=Button(win, text='Analysis', command=self.processinBack)
        self.b1.place(x=700, y=100)
self.lbl3.place(x=450, y=150)
self.lbl4.place(x=550, y=150)
        self.t3.place(x=180, y=200)

    def processinBack(self):
        if          (str(self.t1.get())!=""          or          str(self.t1.get())!=""):
self.lbl4['text']=""
            self.lbl4['fg']="white"
```

```
            download_thread = threading.Thread(target=self.Process)
            download_thread.start()
else:           print("else")
            self.lbl4['text']="*Please Enter All Fields"
self.lbl4['fg']="red"
             def Process(self):
           self.b1['state'] ="disabled"
          self.t3.delete(0.0, 'end')
self.appNameList.append(str(self.t1.get()))
self.url.append(str(self.t2.get()))
           startingPoint = 0
           self.t3.insert(END,("Started Analysing the Training Model\n"))

                #print(str(self.t2.get()))
appurl=str(self.t2.get())          start =
appurl.index('details?id=')          end =
len(appurl)          #print(len(appurl))
           #print(start)
           appid = appurl[start+len('details?id='):end]
           #print(appid)

           us_reviews, token =  reviews(
appid, # app's ID, found in app's url
lang='en',           # defaults to 'en'
            country='us',          # defaults to 'us'
            sort=Sort.NEWEST,     # defaults to Sort.MOST_RELEVANT
filter_score_with=None,  # defaults to None (get all scores)
            count=40              # defaults to 100
            # , continuation_token=token
           )

           app_reviews_df = pd.DataFrame(us_reviews)
          #print(app_reviews_df["content"])
#print(app_reviews_df["score"])
finReviews=app_reviews_df["content"]
```

```
        finRatings = app_reviews_df["score"].to_string(index=False)

        ##########################################################################
#########################################

        self.t3.insert(END,("\n"))
        self.t3.insert(END,("Completed getting the Reviews and Ratings\n"))


        df = pd.read_csv('training.csv')
df.head()       def
preprocess_data(df):

            # Remove package name as it's not relevant
            df = df.drop('package_name', axis=1)

            # Convert text to lowercase

            df['review']      =      df['review'].str.strip().str.lower()
return df

        df = preprocess_data(df)

        x = df['review']
y = df['polarity']
        x,    x_test,    y,    y_test    =    train_test_split(x,y,    stratify=y,    test_size=0.25,
random_state=42)

        vec = CountVectorizer(stop_words='english')
x = vec.fit_transform(x).toarray()        x_test =
vec.transform(x_test).toarray()
        #Using naive Bayes to train model and classification
```

```
        model          =          MultinomialNB()
model.fit(x, y)
        model.score(x_test, y_test)
fop=model.predict(vec.transform(finReviews))
self.t3.insert(END,("\n"))
        self.t3.insert(END, ("Completed Analysis"))
        self.t3.insert(END,("\n"))
self.t3.insert(END,(fop))
        PosrevCount=0
NegrevCount=0         for
i in fop:
if(i==1):
                PosrevCount+=1
else:
                NegrevCount+=1

        negratCount = finRatings.count('1')
negratCount += finRatings.count('2')
posratCount = finRatings.count('3')         posratCount
+= finRatings.count('4')         posratCount +=
finRatings.count('5')

        self.t3.insert(END,("\n"))
        self.t3.insert(END,("Negitive Reviews count : "+str(NegrevCount)))
self.t3.insert(END,("\n"))        self.t3.insert(END,("Positive Reviews
count : "+str(PosrevCount)))        self.t3.insert(END,("\n"))

       self.t3.insert(END,("\n"))
       self.t3.insert(END,("Negitive       Ratings       count       :       "+str(negratCount)))
self.t3.insert(END,("\n"))
        self.t3.insert(END,("Positive       Ratings       count       :       "+str(posratCount)))
  self.t3.insert(END,("\n"))
        PosrevCount +=posratCount
       NegrevCount +=negratCount
```

```
        PosrevCount=PosrevCount/2          NegrevCount=NegrevCount/2
self.t3.insert(END,("Average Negitive Reviews and Ratings percent :
"+str((NegrevCount/(NegrevCount+PosrevCount)*100))))
self.t3.insert(END,("\n"))          self.t3.insert(END,("Average Positive
Reviews and Ratings percent :
"+str((PosrevCount/(NegrevCount+PosrevCount)*100))))
 self.t3.insert(END,("\n"))
        fig = plt.figure()

        if PosrevCount>=NegrevCount:              fig.suptitle(self.appNameList[starting
Point]+" Verdict: This is a good APP")          else:
            fig.suptitle(self.appNameList[startingPoint]+" Verdict: This is a Fraud/Faulty
APP")

        ax = fig.add_axes([0,0,1,1])
        ax.axis('equal')          langs = ['Positive reviews',
'Negetive reviews']          students =
[PosrevCount,NegrevCount]          ax.pie(students, labels
= langs,autopct='%1.2f%%')          plt.show()
        self.b1.config(state="normal")
window=Tk() mywin=MyWindow(window)
window.title('Fraud App Detector')
window.geometry("800x600+10+10")
window.mainloop()
```

CHAPTER-6
SYSTEM TESTING

6. TYPES OF TESTS

6.1 Unit Testing: Unit testing involves testing individual components of the software program or application. The main purpose behind this is to check that all the individual parts are working as the intended. A unit is known as the smallest possible components of software that can be tested.

- Generally, it has a few inputs and a single output.
- Create the test cases
- Review or rework
- Baseline
- Execute test

Unit test reveal a basic understanding of API units or functions to understand the functionality of the code. It is a way to write test cases for all functions and methods so that whenever a change causes a fault then in that case.

Integration Testing

Integration testing is a type of testing meant to check the combinations of different units, their interactions, as the way sub systems unite into one common system, and code compliance with the requirements. Integration testing designed to test integrated the software components to determine if they actually run as one programs.

Acceptance Testing

User acceptance testing of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in the

touch with the prospective system. The testing is to ensure that the solution by the project meets the functional and non-functional requirements specified in the project.

System Testing

System testing is defined as a testing of a complete and fully integrated software project. This testing falls in block box testing where in knowledge of the inner design of the code is not a pre-requisite and is done by the testing team.it tests a configuration to ensure known as predictable results.

White Box Testing

White box testing is a testing the software in which software tester has knowledge. White box testing is an approach that allows testers to inspects and verify the inner works of a software system, its code infrastructure, and integrations with external system.

Block Box Testing

Block box testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Block box testing refers to any type of software test that examines. Block box testing can be performed.

6.2 TEST CASES

To Validate and verify system functionality, the organization must take a multi-faceted approach that evaluates the product's front and back end. There are different ways to categorize the various type of test cases. One way to start is these two categories

- Formal test cases
- Informal test cases

Formal Test Cases

Formal test cases. With this type of test case, the tester writes a test in which the input is known and detailed, such as the preconditions and test data. Formal tests have

predefined input, which means they provide an expected output, which the test attempts to validate.

Informal Test Cases

Conversely, Informal test cases do not have known inputs or outputs. Tester executes these types of test cases to discover and record the outcomes, which can reveal interesting finding about digital quality. Most types of test cases are formal-planned in advance according to software requirements. Let's explore some more test case types

- Functionality
- UI
- Performance

Test Case ID	Test Objective ID	Category	Condition	Expected Results	Actual Results	Requirement ID
0048	0011	ServerSide	Make (previous version will used)	Passes	Failed	00132

CHAPTER-7
CONCLUSIONAND FUTURE ENHANCEMENT

In this project, we developed a fraud detection system for mobile Apps. Specifically, we first showed that fraud happened in leading sessions and provided a method for mining leading sessions for each App from its historical ranking records. We identified that for the detection of the rank ranking, rating, review based evidence are considered. Moreover, we proposed an optimization based aggregation method to integrate all the evidence for evaluating the credibility of leading sessions from mobile Apps. A unique perspective of this approach is that all the evidence can be modeled by statistical hypothesis tests, thus it is easy to be extended with other evidence from domain knowledge to detect ranking fraud. Finally, we validate the proposed system with extensive experiments on real-world App data collected from the Apple's App Store. Experimental results showed the effectiveness of the proposed approach. In the future, we plan to study more effective fraud evidence and analyze the latent relationship among rating, review, and rankings. Moreover, we will extend our ranking fraud detection approach with other mobile App related services, such as mobile Apps recommendation, for enhancing user experience.

FUTURE ENHANCEMENT

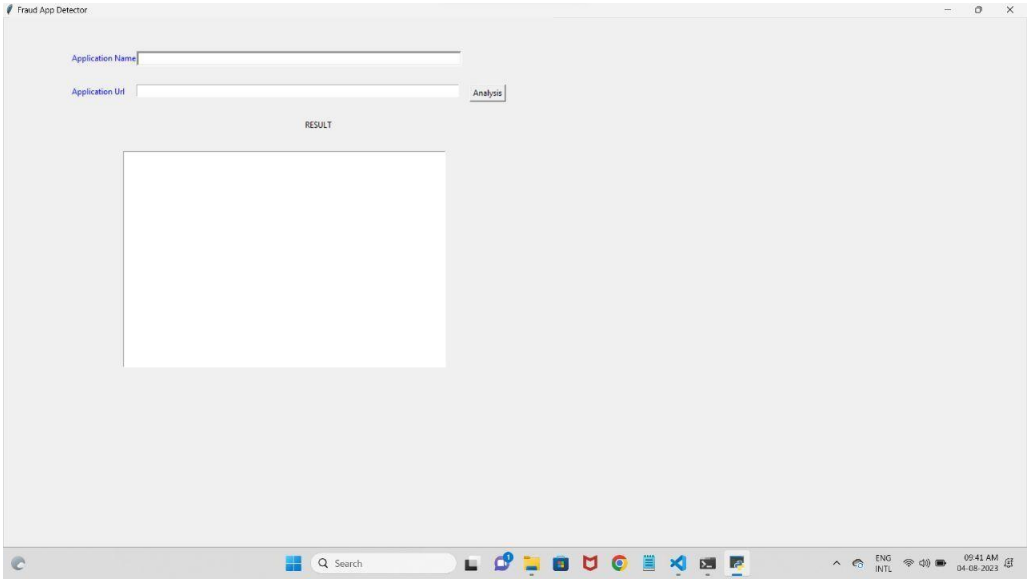
Enhancing user ratings and reviews fraud detection in the Google Play Store is crucial for maintaining trust and fairness. Here are some feature enhancement ideas: -Analyze user behavior patterns, such as the frequency and timing of reviews. Unusual review patterns may indicate fraudulent activity Train machine learning models to detect fake reviews based on historical data. Continuously update these models to adapt to evolving fraud techniques. Consider the context in which reviews are posted. Are they reviewing the same app multiple times with similar language or from the same IP address? Aggregate reviews from multiple sources to cross-verify authenticity. This could include reviews from trusted third-party websites or social media Provide users with information on how the review system works and how fraud detection is performed. Transparency can build trust - Collaborate with app developers to take legal action against individuals or entities engaged in fraudulent activities These enhancements should be implemented carefully to balance fraud prevention with user experience and fairness. Regularly update and adapt these features to stay ahead of new fraud techniques.

CHAPTER-8
REFERENCES

1. L. Azzopardi, M. Girolami, and K. V. Risjbergen, Investigating the relationship between language model perplexity and if precision- recall measures, in Proc. 26th Int. Conf. Res. Develop. Inform. Retrieval, 2003, pp. 369370.
2. D. M. Blei, A. Y. Ng, and M. I. Jordan, Latent Dirichlet allocation, J. Mach. Learn. Res., pp. 9931022, 2003.
3. Y. Ge, H. Xiong, C. Liu, and Z.-H. Zhou, A taxi driving fraud detection system, in Proc. IEEE 11th Int. Conf. Data Mining, 2011, pp. 181190.
4. D. F. Gleich and L.-h. Lim, Rank aggregation via nuclear norm minimization, in Proc. 17th ACM SIGKDD Int. Conf. Know. Discovery Data Mining, 2011, pp. 6068.
5. T. L. Griffiths and M. Stivers, Finding scientific topics, Proc. Nat. Acad. Sci. USA, vol. 101, pp. 52285235, 2004.
6. G. Heinrich, Parameter estimation for text analysis, Univ. Leipzig, Leipzig, Germany, Tech. Rep., http://faculty.cs.byu.edu/~ringer/ CS601R/papers/HeinrichGibbsLDA.pdf, 2008.
7. [N. Jindal and B. Liu, Opinion spam and analysis, in Proc. Int. Conf. Web Search Data Mining, 2008, pp. 219230.
8. J. Kivinen and M. K. Warmuth, Additive versus exponentiated gradient updates for linear prediction, in Proc. 27th Annu. ACM Symp. Theory Compute., 1995, pp. 209218.
9. A. Klementiev, D. Roth, and K. Small, An unsupervised learning algorithm for rank aggregation, in Proc. 18th Eur. Conf. Mach. Learn., 2007, pp. 616623.
10. A. Klementiev, D. Roth, and K. Small, Unsupervised rank aggregation with distance-based models, in Proc. 25th Int. Conf. Mach. Learn., 2008, pp. 472479.
11. A. Klementiev, D. Roth, K. Small, and I. Titov, Unsupervised rank aggregation with domain-specific expertise, in Proc. 21st Int. Joint Conf. Arif. Intel., 2009, pp. 11011106.
12. E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw, Detecting product review spammers using rating behaviors, in Proc. 19thACMInt. Conf. Inform. Know. Manage., 2010, pp. 939 948.
13. Y.-T. Liu, T.-Y. Liu, T. Qin, Z.-M. Ma, and H. Li, Supervised rank aggregation, in Proc. 16th Int. Conf. World Wide Web, 2007, pp. 481490.

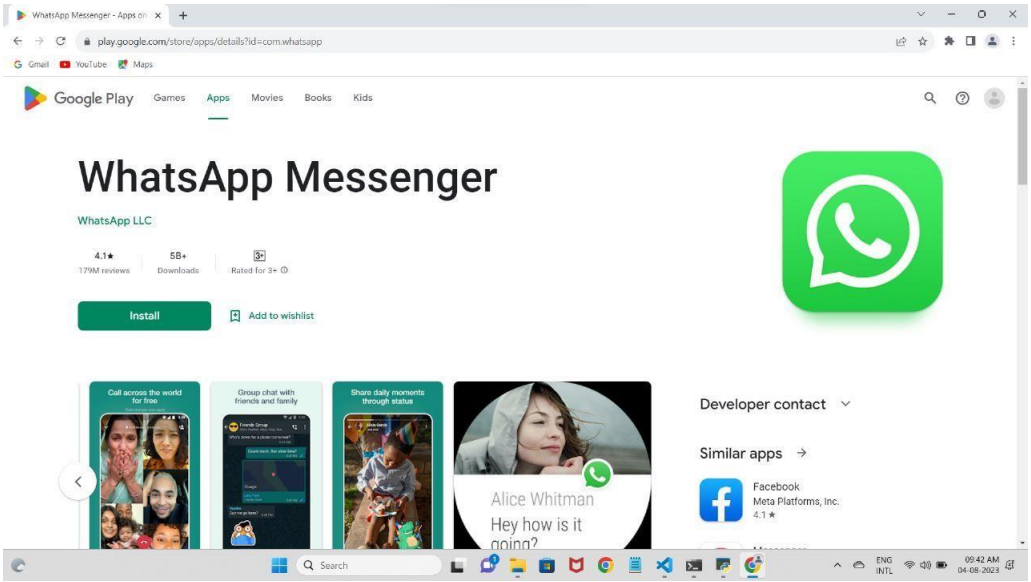
14. A. Mukherjee, A. Kumar, B. Liu, J. Wang, M. Hsu, M. Castellanos, and R. Ghosh, Spotting opinion spammers using behavioral footprints, in Proc. 19th ACM SIGKDD Int. Conf. Know. Discovery Data Mining, 2013, pp. 632640.
15. . Notaulus, M. Major, M. Manasse, and D. Fetterly, Detecting spam web pages through content analysis, in Proc. 15th Int. Conf. World Wide Web, 2006, pp. 8392.
16. G. Shafer, A Mathematical Theory of Evidence. Princeton, NJ, USA: Princeton Univ. Press, 1976.
17. K. Shi and K. Ali, get jar mobile application recommendations with very sparse datasets, in Proc. 18th ACM SIGKDD Int. Conf. Know. Discovery Data Mining, 2012, pp. 204212.
18. [8] L. Azzopardi, M. Girolami, and K. V. Risjbergen, Investigating the relationship between language model perplexity and ir precision- recall measures, in Proc. 26th Int. Conf. Res. Develop. Inform. Retrieval, 2003, pp. 369370.
19. D. M. Blei, A. Y. Ng, and M. I. Jordan, Latent Dirichlet allocation, J. Mach. Learn. Res., pp. 9931022, 2003.

CHAPTER-9
APPENDIX



Screen.No:9.1 User Interface

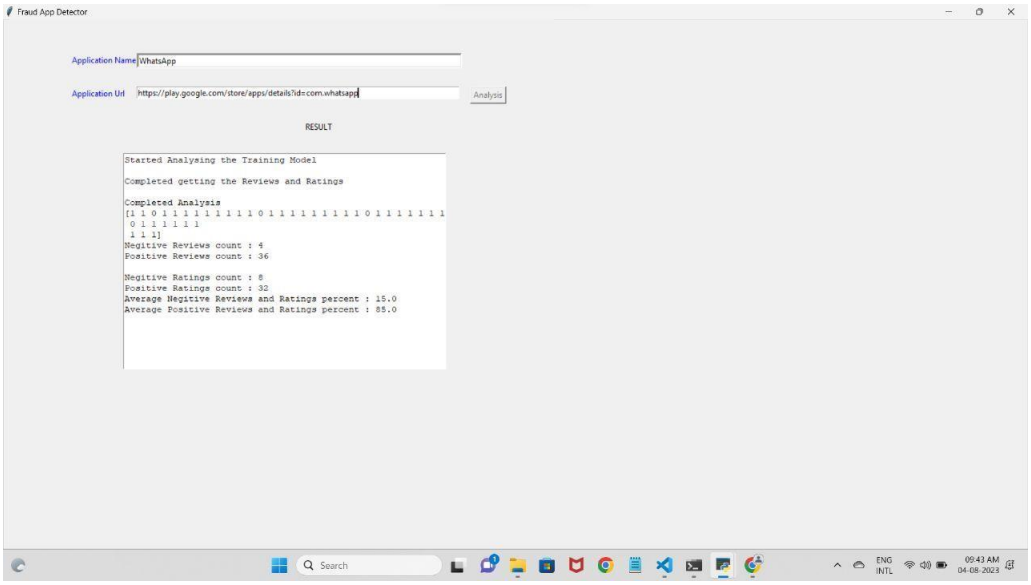
In this page we provide App name & App link for Verification



Screen No:9.2 Data Link Page

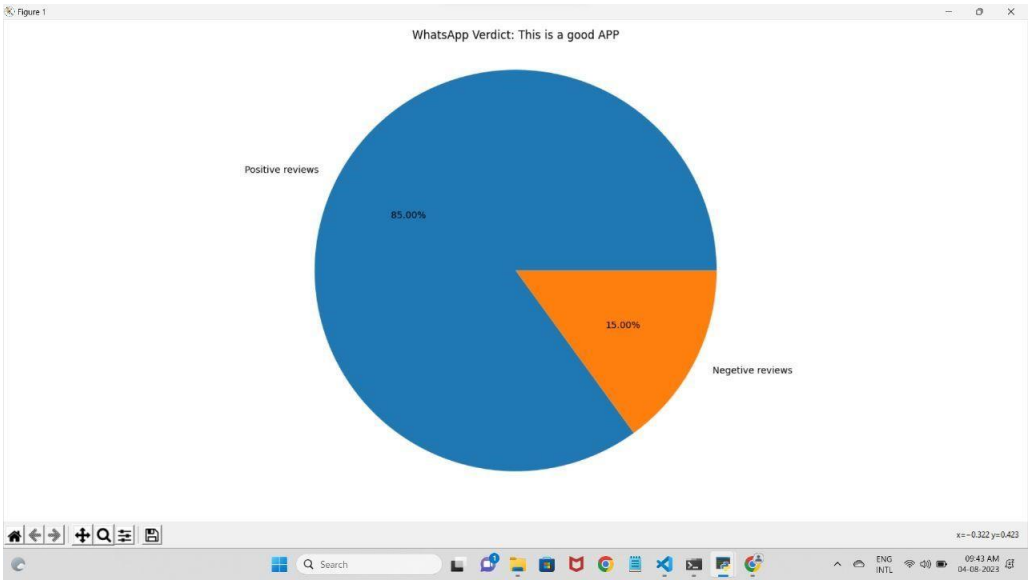
In this it shows the App which is given in the above page

Appendix



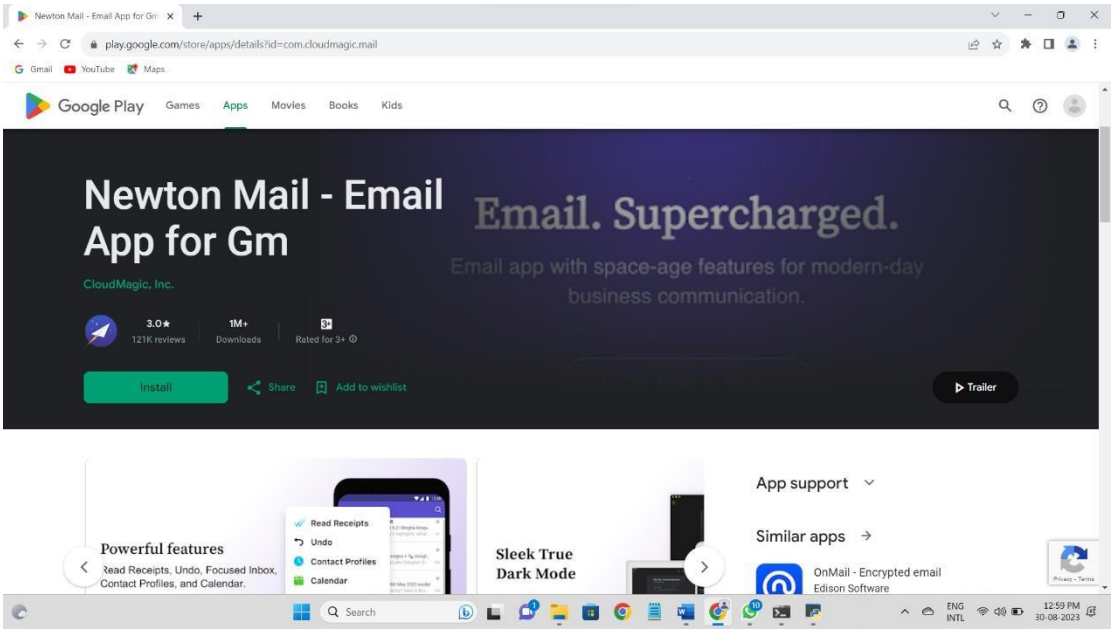
Screen No:9.3 Normal Result Page

It shows the result of verified app with detailed



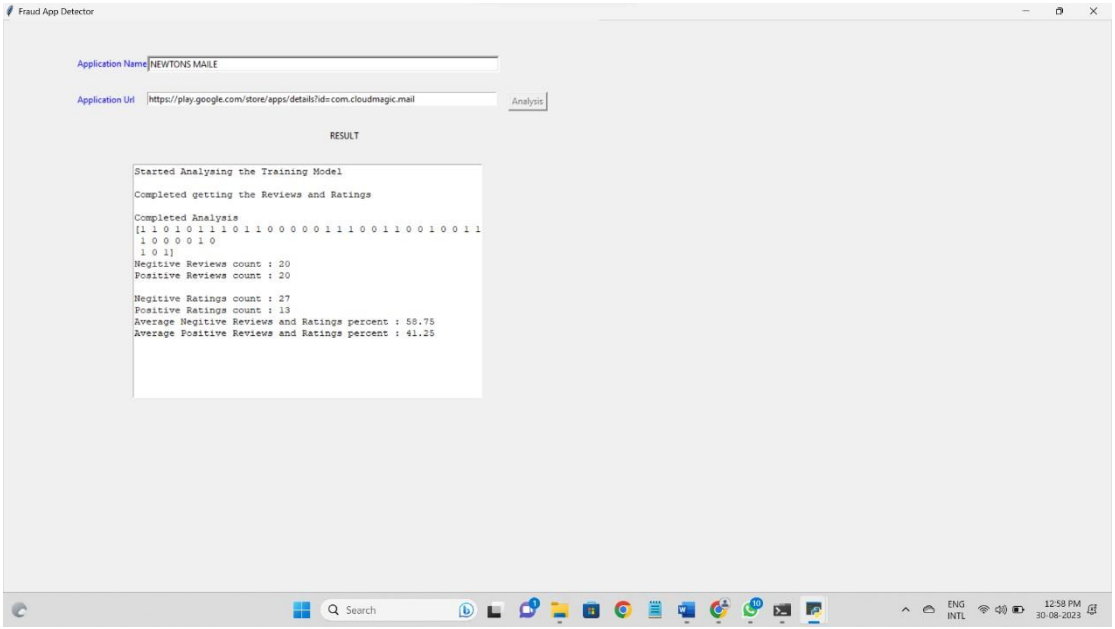
Screen No:9.4 Result in the form pie chart

The Results have shown in the form of PIE Chart



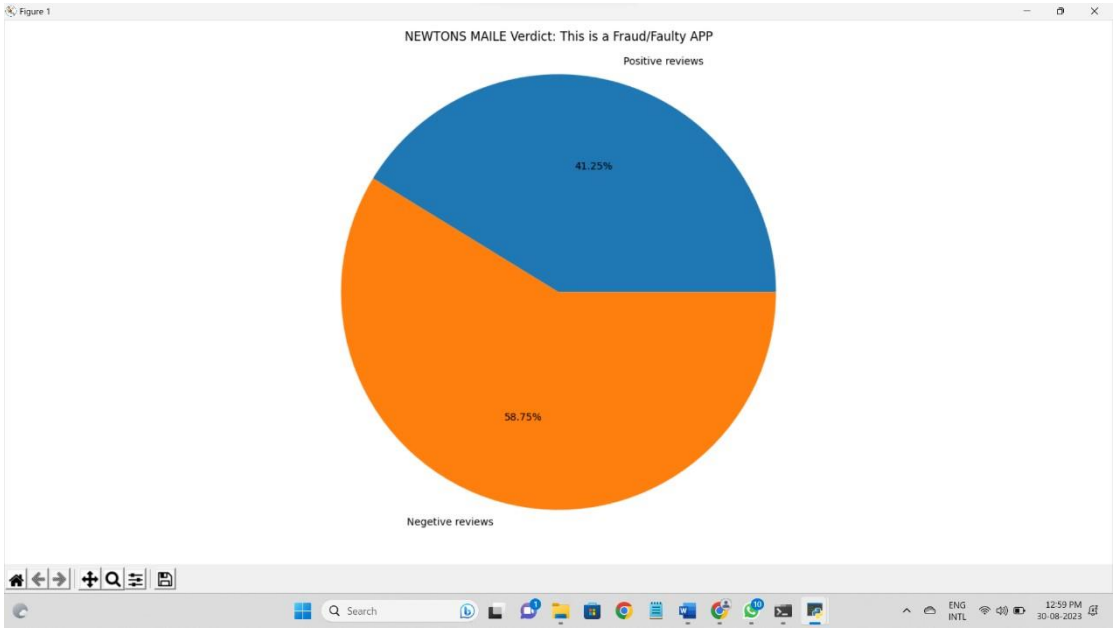
Screen No:9.5 Email App for GM

In this it shows the App which is given in the above user interface page



Screen.No:9.6 User Interface for verification

It shows the result of verified app with detailed



Screen No:9.7 Result in the form pie chart

The Results have shown in the form of PIE Chart

CHAPTER-10
EXECUTION PROCEDURE

How to Install Python on Windows

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3.Download the Python Cheat sheet here.The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link:

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Execution Procedure

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to

their version. Here, we are downloading the most recent python version for windows 3.7.4

Step 4: Scroll down the page until you find the Files option.

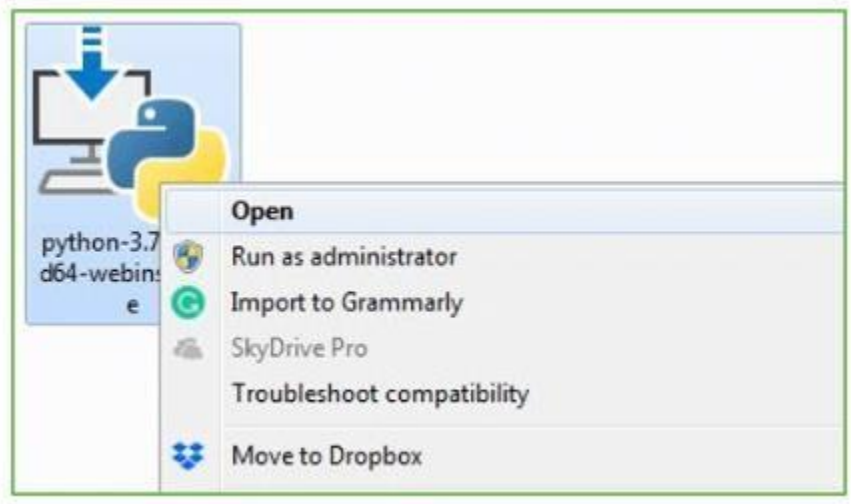
Step 5: Here you see a different version of python along with the operating system.

To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer. To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer. Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Execution Procedure

Step 3: Click on Install NOW After the installation is successful. Click on Close.

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.