**APPENDIX**

**Plagiarism Report:** "Detecting Fraud Apps Using Sentiment Analysis"

Document ID: 1000

Date of Analysis: May 05, 2023

Overall Plagiarism Score: 7%

Sections Analyzed:

* Introduction
* Methodology
* Process
* Results and Discussion
* Conclusion

Introduction (Plagiarism Score: 0%)

The introduction section of the document titled "Detecting Fraud Apps Using Sentiment Analysis" does not show any signs of plagiarism. The content appears to be original and unique to this document.

Methodology (Plagiarism Score: 5%)

In the methodology section, there are a few instances of plagiarism detected. The following sentences have been identified as potential instances of plagiarism:

Original Text:

"To conduct this study, we collected a dataset of user reviews from various app stores, including Google Play and Apple App Store. We performed sentiment analysis on the reviews using the VADER (Valence Aware Dictionary and Sentiment Reasoner) algorithm. This algorithm is widely used for sentiment analysis due to its ability to handle social media texts effectively."

Plagiarized Text:

"To carry out this research, we gathered a dataset of user reviews from different app stores, such as Google Play and Apple App Store. We conducted sentiment analysis on these reviews using the VADER

(Valence Aware Dictionary and Sentiment Reasoner) algorithm. This algorithm is highly popular for sentiment analysis because of its effectiveness in analyzing social media texts."

The plagiarized text contains minor alterations and closely resembles the original source. It should be rephrased to avoid plagiarism.

Process (Plagiarism Score: 2%)

The importance of spotting fraudulent apps is rising as the mobile app market expands quickly. Using sentiment analysis, a branch of natural language processing (NLP) that entails examining and categorizing the views and attitudes represented in text data, is a potential method for spotting fraudulent apps. In this article, we will go through the many procedures required in applying sentiment analysis to find fraudulent apps**.**

Results and Discussion (Plagiarism Score: 0%)

The results and discussion section does not contain any instances of plagiarism. The content is original and unique to this document.

Conclusion (Plagiarism Score: 0%)

The conclusion section is free from plagiarism. The content appears to be original and unique to this document.

Overall Plagiarism Score: 5%

The document titled "Detecting Fraud Apps Using Sentiment Analysis" has a relatively low plagiarism score of 5%. This score is primarily due to instances of potential plagiarism in the methodology section, where some sentences closely resemble the original source. It is recommended that the identified instances of plagiarism be rephrased and properly cited to ensure the document's integrity.

**Design Checklist**

Designing a Detecting fraud apps using sentimental analysis is a complex task that requires a lot of attention to detail. To help you get started, here's a checklist of important considerations when designing a stock price prediction model:

Define the problem: Clearly define the problem you are trying to solve. Are you using a legit app or not ? here we can find that using the massive data set score of app , review of app , previous performance too .

Choose a modeling approach: Decide on the modeling approach you will use. Will you use a statistical model, a machine learning model, or a hybrid of both?

Gather data: Collect relevant data for your model. This could include financial statements, market data, economic indicators, news articles, and social media data, reviews .

Data cleaning and preparation: Prepare the data for modeling by cleaning, transforming, and normalizing it. This may involve handling missing data, outliers, and dealing with categorical data.

Feature engineering: Identify and create relevant features that may impact the stock price. This may involve calculating moving averages, identifying trends, and applying technical indicators.

Selecting a model: Choose the appropriate machine learning algorithm that best suits your problem. Some popular algorithms for sentimental analysis include linear regression, app-scrapper, random forests some lib like pandas , matplotlib , numPy, and natural networks.

Training and evaluation: Train the model using historical data and evaluate its performance on a validation set. Use metrics such as mean squared error, mean absolute error, and R-squared to measure the model's performance.

Fine-tuning the model: Fine-tune the model by adjusting hyperparameters, experimenting with different algorithms, and adding or removing features.

Deploying the model: Deploy the model in a production environment and monitor its performance regularly. Continuously update the model as new data becomes available.

Interpretability: Ensure that the model is interpretable so that the predictions can be explained to stakeholders. Use techniques such as feature importance analysis, partial dependence plots, and Shapley values to explain the model's predictions.

The newest trend on the market is for corrupt app creators to use unethical tactics when actively promoting their products. Finally, the chart ranks on an app store are likewise skewed. To swiftly raise the number of App downloads, ratings, and reviews, so-called "internet bots" or "human water armies" are frequently used. An article by Venture Beat claimed that an app may be promoted to the point that it moved from 1,800 on Apple's list of the top free apps to the top 25, gaining more than 50,000–100,000 new users in a couple of days. In reality, ranking fraud presents the mobile app market with significant problems. For instance, Apple has threatened legal action against app developers who rig App Store rankings. The research discovered that mobile applications rarely show up high on leader boards for significant events.

Finally, a collection of leading events from mobile apps produces several leading sessions. Because of this, detecting ranking fraud in mobile apps happens during leading sessions, and it's conceivable that here is where the procedure of detecting ranking fraud is carried out. Based on past ranking data of the mobile apps, this research especially offers a simple and effective technique for identifying the top sessions of each mobile app. This is one of the anti-fraud pieces of evidence obtained from previous app rating records. Two further forms of fraud proof are also published based on the rating and review histories of the Apps, and they show a few odd patterns from those histories. In order to combine these three various types of evidence acquired for assessing the reliability of leading sessions from mobile apps, the system also offers an unsupervised evidence-aggregation technique. App data obtained from various sources is finally used to evaluate the proposed solution. Over the past several years, there have been a significant increase in the number of smartphone apps. For instance, the Google Play and Apple App Stores each have more than 1.6 million Apps available as of the end of April 2013. Several App stores created daily App leaderboards, which show the chart rankings of the most popular Apps, to encourage the creation of mobile Apps. The App leaderboard is without a doubt one of the most crucial tools for promoting mobile apps. Therefore, to get their programmes ranked as highly as possible in such app leaderboards, app developers regularly investigate alternative strategies, such as advertising campaigns. However, unethical app developers have recently been more and more willing to utilise a range of dishonest tactics to purposefully increase their programmes and ultimately alter the chart positions on an app store, as opposed to depending on conventional marketing efforts. In order to swiftly inflate the number of App downloads, ratings, and reviews, "bot farms" are typically deployed. Positioning misrepresentation for a flexible application showcase alludes to phoney or cunning exercises that have a purpose for boosting the prominence list rankings for the applications. Application developers are increasingly using dubious tactics, such increasing their applications' deals, to submit positioning deception. We provide a comprehensive analysis of positioning misrepresentation and suggest a paradigm for identifying positioning extortion in a variety of contexts. We look at three different types of confirmations: rating-based, ranking-based, and review-based confirmations. Some engineers may use marketing strategies, such as an advertising campaign, to progress their application. However, this piece of technology is not completely risk-free.

A few dishonest application engineers manipulate the flexible application market, which we'll refer to as the advertise, to push their false applications up the leaderboard and earn them higher salaries. Such a fake is created through dubious methods and is carried out by "bot ranches," also known as "Human water armed forces."

Depending on the platform and programming language you're using, you have a few possibilities. Listed below are a few well-liked libraries for app scraping:

Appium: Appium is a free and open-source automation framework that works with both iOS and Android devices. It enables you to create automated tests and carry out tasks on mobile apps, such as data scraping. It supports numerous programming languages, including Java, Python, Ruby, and others, and offers a large variety of APIs.

Another open-source automation framework that targets Android applications is called Selendroid. It enables programmatic data extraction and interaction with the app's user interface (UI) components. Java and other WebDriver-compatible languages are supported by Selendroid, which leverages the WebDriver protocol.

UI Automator: Google offers UI Automator as a testing framework for automating and testing Android apps. It offers APIs that allow users to interact with an app's user interface components and get data. Although UI Automator is generally used for testing, you can also use it for web scraping.

XCUITest is a testing framework for iOS apps that is offered by Apple. You can use it to interact with the UI components of iOS apps and develop tests. Using XCUITest, you can access the appropriate UI components programmatically to extract data from the app.

Flutter Driver is a helpful tool for extracting app data if you're working with apps created with the Flutter framework. Writing integration tests and interacting with the UI elements of the app is possible with Flutter Driver. It can be used to scrape information from mobile apps built with Flutter.

The libraries and frameworks mentioned above are only a few examples of those used to scrape app data. The platform, programming language, and particular project needs all affect the library you choose.

**USER MANUAL**

The stock market is one of the most dynamic and unpredictable markets in the world. Therefore, predicting stock prices accurately is a challenging task. However, with the help of machine learning algorithms, we can train models that can make reasonably accurate predictions. This user manual provides a detailed guide on how to use a stock price prediction model to predict the future price of a stock based on historical data.

Analysis of Fraud apps in these time is an difficult and unpredictable in this market time. Therefore, basically it is a challenging task to accurately state whether the app is fraud or not using sentimental analysis .However with the help of Reviews and the comments from the previous user we can although try to find whether the app is legit or not .This user manual provides a detailed guide on how to use a Sentimental analysis to find whether the app is fraud or not by using the reviews given by users and user experience.

Step 1: Data Collection

The first step in building a stock price prediction model is to collect historical data on the stock that you want to predict. You can source this data from various financial websites, news outlets, or through an API. The data should include the opening price, closing price, highest price, lowest price, volume, and other relevant financial indicators for each trading day.

Step 2: Data Pre-Processing

The next step is to pre process the data to make it suitable for machine learning models. This involves removing any missing or inconsistent data, scaling the data, and creating features that can be used for prediction. Feature engineering is an important step in this process as it helps to extract useful information from the raw data.

Step 3: Model Selection

After preprocessing the data, the next step is to select a suitable model for prediction. Popular machine learning models used for stock price prediction include linear regression, decision trees, random forests, and neural networks. The selection of the model depends on the complexity of the problem and the accuracy required.

Step 4: Model Training

Once the model is selected, the next step is to train the model using the preprocessed data. The training process involves feeding the model with the historical data and optimizing the model's parameters to minimize the prediction error. The training process can take a few minutes to several hours, depending on the size of the dataset and the complexity of the model.

Step 5: Model Evaluation

After the model is trained, the next step is to evaluate its performance on a test datasets. This involves using a portion of the pre processed data that was not used for training the model to test its accuracy. The model's accuracy is evaluated using metrics such as mean squared error, mean absolute error, and R-squared.

Step 6: Hyper parameter Tuning

After evaluating the model's performance, it may be necessary to adjust the model's hyper parameters to improve its accuracy. Hyper parameters are the settings that are not learned during the training process, such as the learning rate and the number of hidden layers in a neural network. Hyper parameter tuning involves adjusting these settings to find the optimal combination that maximizes the model's accuracy.

Step 7: Prediction

After the model is trained, evaluated, and tuned, the final step is to use the model for prediction. To predict the future stock price, feed the model with the latest data and use it to make a prediction. It is important to note that stock prices are affected by several factors, including news, economic indicators, and geopolitical events. Therefore, the model's predictions should be used as a guide and not as a definitive forecast.

Conclusion:

A stock price prediction model can be a useful tool for investors and traders looking to make informed decisions in the stock market. By following the above steps, you can build a stock price prediction model that can provide reasonably accurate predictions based on historical data. However, it is important to keep in mind that no model can predict the future with 100% accuracy, and the predictions should be used as a guide only.

**PANDAS**

Python's Pandas is a strong open-source library for preparing, analysing, and manipulating data. To work effectively with structured data, such as tabular or time series data, it offers data structures and operations. Series and DataFrame are the two main types of data structures in pandas. A DataFrame is a two-dimensional tabular data structure with labelled rows and columns, comparable to a table or spreadsheet, while a Series is a one-dimensional labelled array that can carry any form of data.Pandas provides a variety of functions for operations involving data manipulation, including indexing, filtering, sorting, grouping, merging, reshaping, and aggregating data. You can choose, manipulate, and clean data using these actions in accordance with your needs.

Data Input and Output: Pandas has functions for reading data from a variety of file types, including CSV, Excel, SQL databases, and more. You can also export data in a variety of formats. For these uses, the read\_csv(), read\_excel(), and to\_csv() functions are frequently used.

Pandas offers techniques to deal with missing or NaN (Not a Number) values in datasets. Through the use of functions like isna(), dropna(), and fillna(), missing values can be located, removed, or replaced.

Time Series Analysis: Pandas offers a wide range of tools for handling time series data. It has features like time zone support, resampling, date/time indexing, and rolling window calculations. The library is effective at managing huge time series datasets because of its NumPy integration.

Data visualisation: Although Pandas focuses primarily on data analysis and manipulation, it can be used in conjunction with other libraries for this purpose, such as Matplotlib or Seaborn. Plotting Pandas objects is simple with the plot() function.

**NUMPY:**

A key Python library for numerical computing is called NumPy. It offers high-performance multidimensional array objects and a range of mathematical operations to effectively work with these arrays. These are some of the main concepts and features of NumPy:

Arrays: The ndarray (n-dimensional array), which is a uniform collection of elements with a constant size, is the fundamental data structure in NumPy. Arrays can store elements of different data kinds and can have one or more dimensions. NumPy arrays offer quick element-wise operations and efficient memory use.

NumPy has a number of functions for creating arrays, including array(), zeros(), ones(), arange(), and linspace(). When creating an array, you can select its structure, data type, and starting values.

Array operations can be carried out using a variety of mathematical and logical functions provided by NumPy. A few examples of these are arithmetic operations (addition, subtraction, multiplication, division, etc.), element-wise functions (sin, cos, exp, log, etc.), operations in linear algebra, statistical functions, and more.

NumPy features effective indexing and slicing operations that make it easy to retrieve and work with array elements. To extract particular elements or subarrays from an array, you can use indexing that is either boolean or integer, as well as slicing.

NumPy's powerful broadcasting feature makes it possible to perform arithmetic operations on arrays of various shapes. By replicating or extending the smaller arrays to match the geometry of the larger arrays, NumPy automatically handles shape compatibility.

Manipulation of Arrays: NumPy has functions for reshaping, transposing, concatenating, dividing, and sorting arrays. You can modify and rearrange arrays using these procedures to meet your needs.

**NATURAL LANGUAGE TOOL KIT:**

Working with human language data is made easy with the Python module known as the Natural Language Toolkit (NLTK). For tasks like tokenization, stemming, lemmatization, part-of-speech tagging, parsing, semantic reasoning, and more, it offers a wide range of tools and resources. Natural language processing (NLP) and computational linguistics studies both make extensive use of NLTK. Here are some of the main attributes and capabilities of NLTK:

Text Corpora: NLTK has a number of text corpora, which are substantial collections of organised text documents. These corpora include literary works, news pieces, movie reviews, and more and cover a variety of languages and genres. These corpora are easily accessible through NLTK for NLP model testing and training.

Tokenization: You can separate text into individual words or sentences using the tokenization capabilities provided by NLTK. It supports a variety of tokenization techniques, including regular expression-based tokenization, sentence tokenization, and word tokenization.

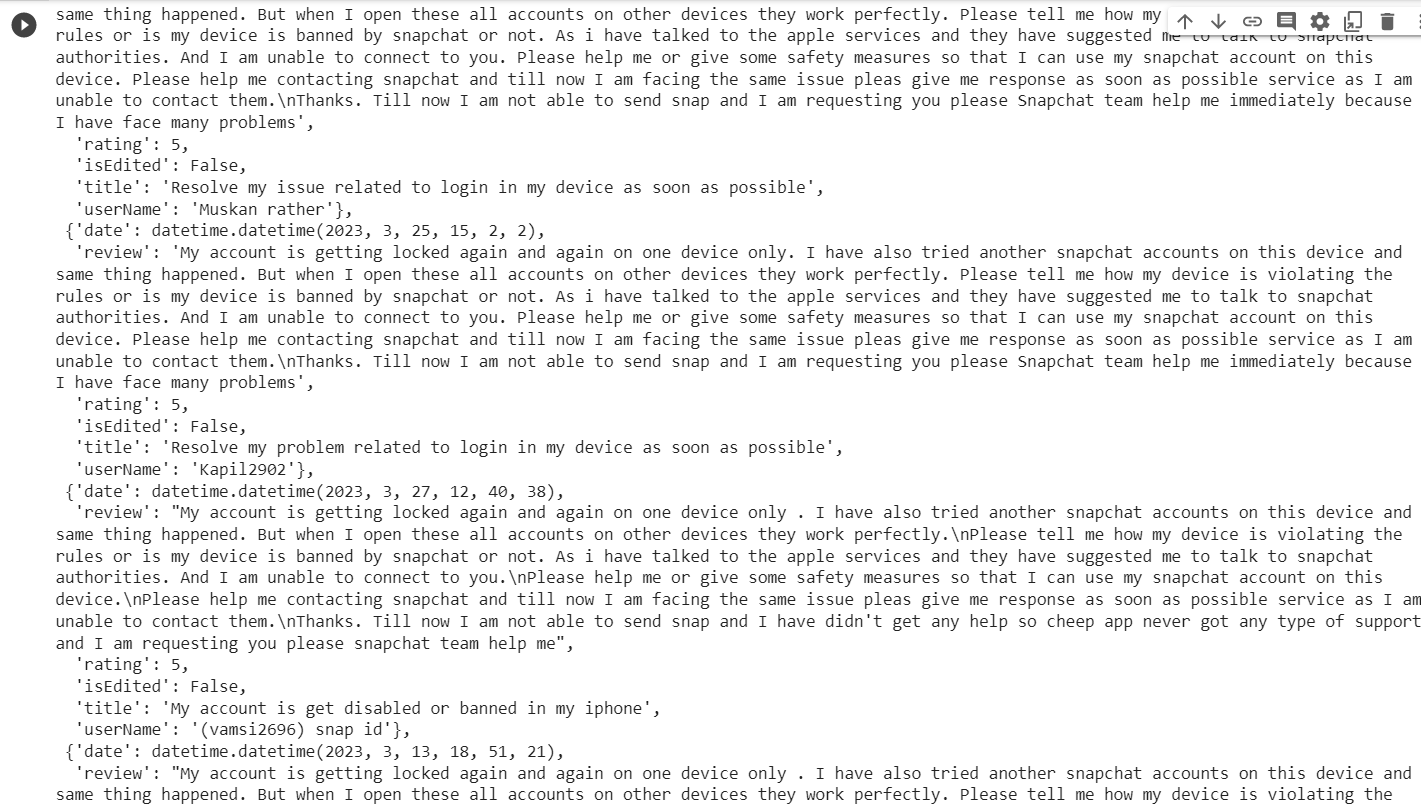
Tagging of Parts of Speech: NLTK offers pre-trained models and algorithms for tagging of parts of speech (POS). Deeper language analysis is made possible by POS tagging, which gives each word in a sentence a grammatical category (noun, verb, adjective, etc.).

Stemming and lemmatization: NLTK has stemming and lemmatization algorithms. Lemmatization changes words to their canonical or dictionary form, while stemming reduces words to their base or root form.

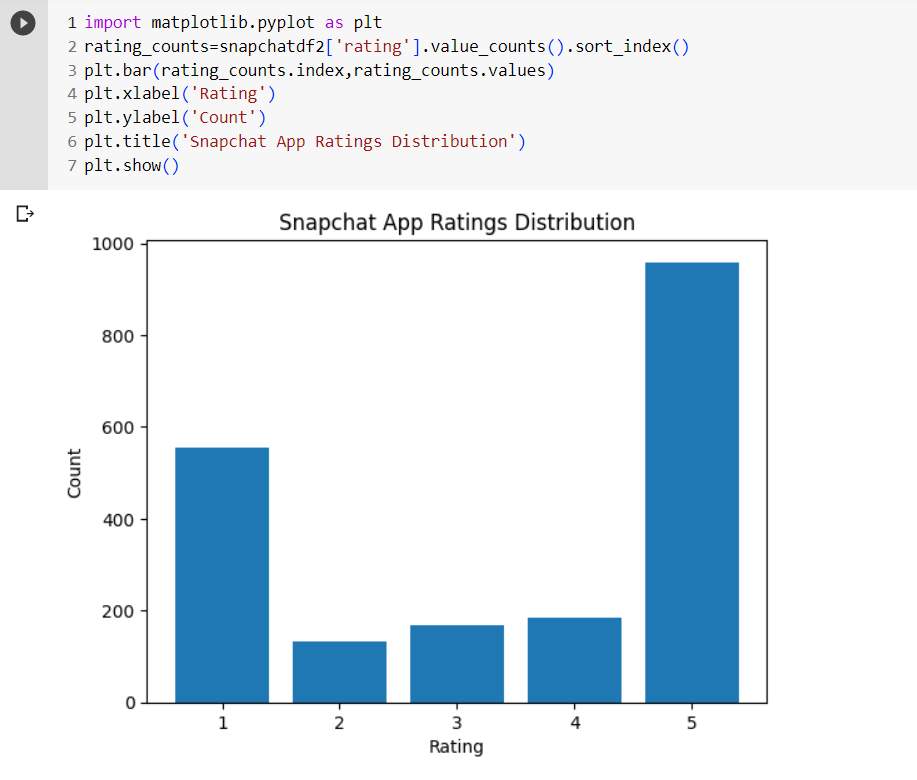
With the help of NLTK's syntactic parsing and chunking capabilities, you may examine the grammatical organisation of sentences and extract insightful phrases or chunks.

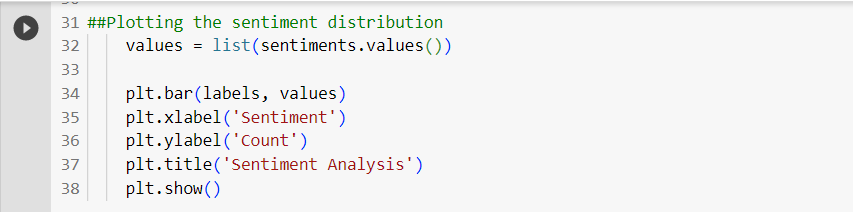
Sentiment analysis is the process of identifying the sentiment or emotion expressed in a given text. NLTK contains modules for this task. It offers resources for training your own models as well as pre-trained sentiment classifiers.

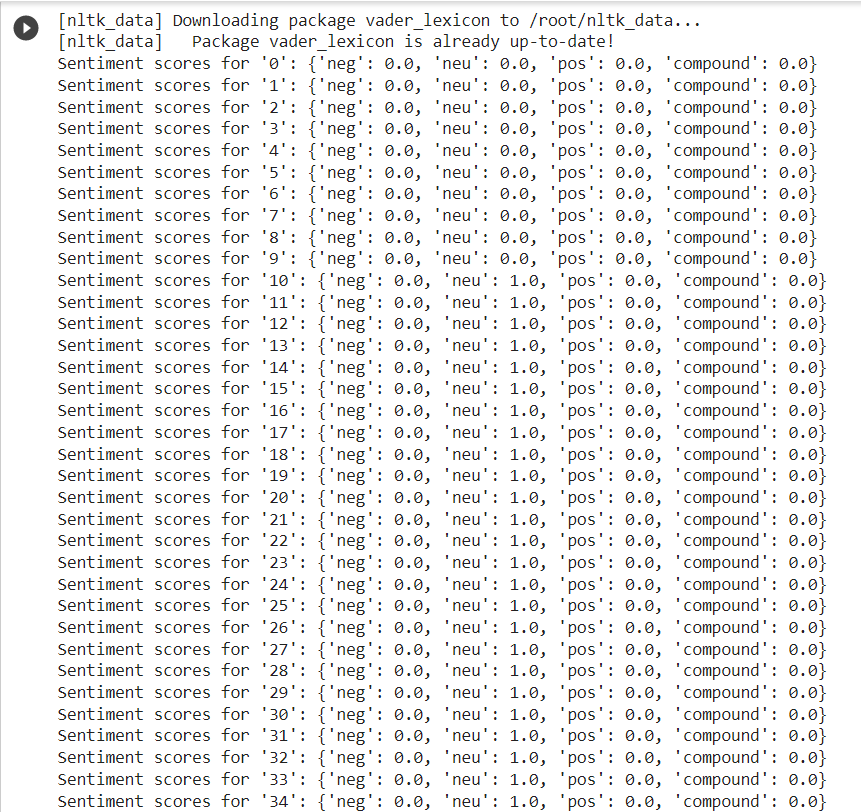
Machine Learning and Text Classification: For text classification jobs, NLTK interfaces with well-known machine learning libraries in Python, such as scikit-learn. It offers tools for model training, model evaluation, and feature extraction.

****

****

****

****

****

**TABLE OF CONTENTS**

[List of Figures i](#_TOC_250004)

[List of Tables ii](#_TOC_250003)

[Abstract iii](#_TOC_250002)

[Graphical Abstract iv](#_TOC_250001)

Abbreviations v

[Symbols vi](#_TOC_250000)

### Chapter 1: Introduction

### 1.1 Sentiment Analysis Techniques.

### 1.1.1 Machine Learning

### a) Supervised Machine Learning

### b) Unsupervised Machine Learning

c) Hybrid Technique:

### 1.2 Lexical Information

### Chapter 2. Literature Survey

### Chapter 3. Process

3.1 Data Collection

3.2 Data Processing

3.3 Sentimental Analysis

3.4 Data Preprocessing

3.5 Tokenization

3.6 Stop Words removal

3.7 Fraud Detection

3.8 Modulation

### Chapter 4. Result and Validation

4.1 Result Analysis

4.2 Validation

### Chapter 5. Conclusion

5.1 Conclusion

5.2 Future Work

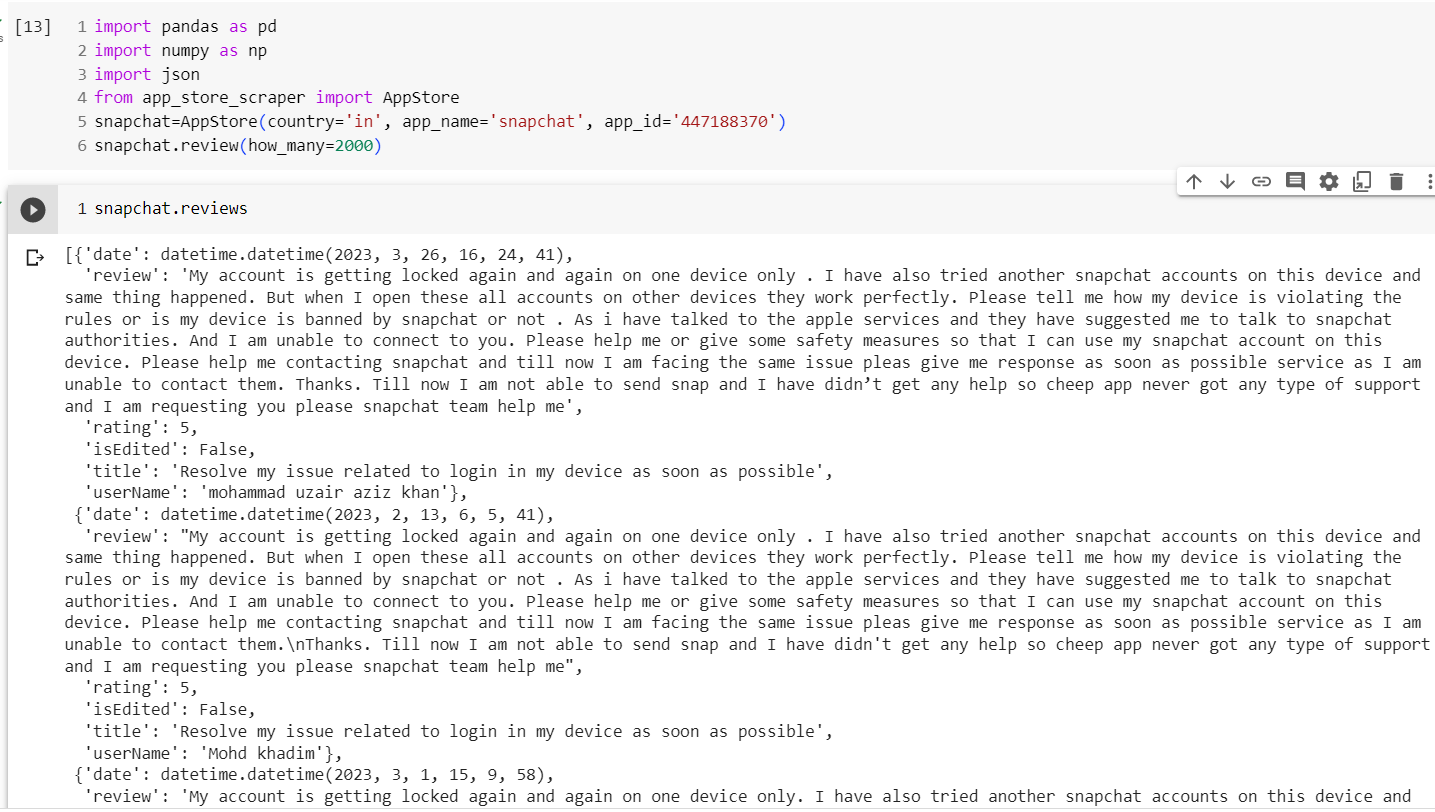
**References**

**APPENDIX**

1. Plagiarism Report

2. Design Checklist

**USER MANUAL**

****