Original Article

Comparison of machine learning algorithms used to catalog Google Appstore

Priyadarshini Pattanaik[1]^, Dimple Nagpal[2]

[1]Faculty of Computer Science and Informatics, Berlin School of Business & Innovation (BSBI), Berlin, Germany; [2]Lovely Professional University, Phagwara, India

Contributions: (I) Conception and design: Both authors; (II) Administrative support: Both authors; (III) Provision of study materials or patients: Both authors; (IV) Collection and assembly of data: Both authors; (V) Data analysis and interpretation: Both authors; (VI) Manuscript writing: Both authors; (VII) Final approval of manuscript: Both authors.

Correspondence to: Priyadarshini Pattanaik, PhD. Faculty of Computer Science and Informatics, Berlin School of Business & Innovation (BSBI), Alte Post, Karl-Marx-Straße 97-99, 12043 Berlin, Germany. Email: ppattanaik055@gmail.com.

Background: Google Play Store is a popular Android app store where users' reviews and ratings provide valuable insights. As part of application development, clients and app designers have a significant impact on the market. Accurately predicting market trends is critical to the success of applications, and this is where information mining comes in. By evaluating various factors such as application name, pricing, reviews, and category, we can predict which types of apps are most likely to be successful.

Methods: To do this, we can use machine learning techniques that help us analyze data from diverse metrics and identify relationships. In this work, five different types of existing machine learning computation techniques like decision tree algorithm (DTA), support vector machine (SVM), random forest (RF), stochastic gradient boosting (SGB) and gated recurrent unit (GRU) were used to show a comprehensive comparison between the models to describe and forecast the structure of Android Market applications. Through these methods, we can extract a wealth of information to make wise decisions and present data more effectively.

Results: Our findings have shown that the GRU-based technique performance accuracy of our predictions is high, with an average of 89.4465%, which can be best visualized as an unusual forest picture better than the four baseline algorithms. Among all the five techniques, GRU classifies 5,616 apps and provides the most precise results for both users and developers, whereas the other classifiers only classified 3,744 apps.

Conclusions: According to the result analysis, we can conclude that all five machine learning algorithms were capable of analysing the android market, GRU outperforms in terms of accuracy.

Keywords: Google Appstore; machine learning; gated recurrent unit (GRU); classification; ratings

Received: 05 June 2023; Accepted: 18 September 2023; Published online: 01 November 2023.

doi: 10.21037/jmai-23-58

View this article at: https://dx.doi.org/10.21037/jmai-23-58

^ ORCID: 0000-0001-5058-5471.

Introduction

Artificial intelligence has many different applications and perspectives, and there is tremendous potential for growth in this field. In this study, machine learning models and frameworks are examined in depth to address a variety of difficulties (1,2). One important aspect of artificial intelligence is accurately predicting information based on the available data. In the future, machine learning is likely to improve its ability to make precise predictions, using a variety of unsupervised learning capabilities as more and more data is generated at a global level. Additionally, neural network topologies are expected to become increasingly complex in order to more accurately distinguish between semantically significant items (3). As support for deep learning improves, researchers will be able to use these areas of interest to complete more tasks with greater accuracy.

Mobile phones have become an integral part of people's lives and the mobile application industry is growing rapidly, resulting in increased revenue for the global sector. However, with the growing number of mobile application designers, it is important for them to sustain their income in the market and move in the right direction. Google Play Store (4,5) is one of the most popular app stores, with over 0.675 million Android applications available. With so many options available, online application surveys have created a significant impact compared to paid applications. It is difficult for potential clients/users to sort out all the reviews and ratings, and for developers to improve application performance based only on evaluations. Along with this, various challenges are faced on the Google Play Store due to an increase in the number of applications with no specific features and usage that misleads the users with app quality, security, privacy, app monetization, policy compliance, competition, and user engagement. This study examines the features and accessibility of the Google Play Store and forecasts the success of the app. With almost 6,140 mobile applications released by the Google Play Store every day, developers and app industries face a tough challenge to come up with a unique concept-based app that performs successfully in the Android market. The expanding cellular sector raises the level of competition, but also increases the likelihood of failure.

Previous research has focused on using reviews and ratings to classify them into positive, negative, and neutral categories, but no work has been done to predict biased and unbiased ratings using machine learning models.

The objective of this manuscript is to address these issues in the following ways:

(I) Develop an intelligent system to assist users based on their needs and interests. Investigate the prediction of an automated review rating system that can accurately predict unbiased ratings.

(II) Focus on predicting biased and unbiased ratings using machine learning models, as previous research has primarily focused on sentiment classification.

(III) Leverage the capabilities of machine learning models to extract complex patterns and representations from review data.

(IV) Enhance the transparency and fairness of online review platforms by providing users with objective and reliable information through unbiased ratings.

(V) This article uses five different types of existing techniques to show a comprehensive comparison between the models to deliver the results as per the user's needs and interests.

In this study (5), the authors propose a framework that uses a deep learning architecture to predict the discrepancy between user ratings and reviews after analyzing the review. The sentiment of the review is considered one of the most significant features that can be used for analysis to solve this problem. The authors believe that using this approach will lead to more accurate prediction of ratings, which will help users make the right decision.

Highlight box

Key findings

- By evaluating various factors such as application name, pricing, reviews, and category, we can predict which types of apps are most likely to be successful.

What is known and what is new?

- Previous research has focused on using reviews and ratings to classify them into positive, negative, and neutral categories, but no work has been done to predict biased and unbiased ratings using machine learning models.

What is the implication, and what should change now?

- Investigate the prediction of an automated review rating system that can accurately predict unbiased ratings.
- Enhancing the transparency and fairness of online review platforms by providing users with objective and reliable information through unbiased ratings.
- This article uses five different types of existing techniques to show a comprehensive comparison between the models to deliver the results as per the user's needs and interests.

In the Android Market (5,6), developers receive 70% of the full cost, while the Play Store keeps the remaining 30%. According to the State of Developers of Mobile Apps survey, less than 10% of software developers have more than 1,000 installations, and only 15% have exceeded 10,000 installations, indicating that the likelihood of an app succeeding decreases as the number of downloads decreases. To address this issue, the authors conducted research on apps with promising ideas that performed poorly on the app store. Venkatakrishnan *et al.* [2020] (6) and other researchers employed the gates recurrent unit classifier to categorize and analyze the emotional sentiment views as optimistic, unfavorable, or impartial.

Machine learning has been applied in various industries such as Amazon's online shop and IBM's e-commerce, with product classification and recommendations being some examples. Google has also used machine intelligence to enhance ad placement and ad content design, as well as image search through picture processing (7). While there are numerous machine learning techniques for different applications, classification problems have become the most common in this field. This involves supervised learning, where a training set of correctly classified data is fed into the machine learning algorithm to train the system. This approach allows the algorithms to accurately identify new data.

Many studies have been conducted to improve classification in various domains, such as using machine learning algorithms to differentiate between individuals with schizophrenia and bipolar disorder from healthy patients using structural magnetic resonance imaging (MRI) data, as reported by Banumathy *et al.* [2023] (8). In other studies, the support vector machine (SVM) learning approach was used to create models for product classification and to focus consumer searches in data sets, as reported by Maheswari *et al.* [2014] (9) and SVM, respectively. Adeola *et al.* [2022] (10) identified the most detected features to determine the polarity distribution in shapeless reviews, such as optimistic, undesirable, and impersonal. Due to the increasing amount of personal and public information stored online, linguistic data can be extracted from bloggers, newsgroups, review sites, and other social media channels. This unstructured input can be converted into structured data using writeup prediction models, allowing for the analysis of customer sentiment towards specific apps, products, services, and brands through a relational database. Aralikatte *et al.* [2018] (4) found a 20% discrepancy between star ratings and reviews in their study of 8,600 reviews of popular Android apps. To address this issue, they propose a deep learning framework that can predict the discrepancy between user ratings and reviews. The authors believe that using this approach, which considers sentiment as a significant feature, will lead to more accurate predictions of ratings and help users make informed decisions. The authors conducted a survey to obtain both users' and app developers' perspectives on the issue of mismatched ratings and reviews. The results showed that detecting this discrepancy was beneficial because user's decisions to download an app were influenced by the existing ratings and reviews. It was also found that this inconsistency affected the download rate of upcoming and small apps. Furthermore, users often only posted reviews when they found problems with an app and did not update the star rating, which contributed significantly to the mismatch between rating and review.

Machine learning has been used in a variety of industries, including Amazon's shop, IBM' se-commerce, amongst other things. Machine learning has been used in product classification and suggestion. With machine intelligence, Google has substantially improved ad placement and ad content design. Google's image search also makes heavy use of machine learning in picture processing. Despite the fact that there are a variety of machine learning methods for diverse applications, classification problems have become the most common in this area. Classification is an example of supervised learning, as described by Zahoor *et al.* [2020] (11) in which a training set of correctly classified observations is supplied into the train the system, using a machine learning algorithm. The training sets help to gain the knowledge and such approach permits the machine learning algorithms to correctly recognize fresh data always. The unsupervised learning algorithm known as clustering involves classifying data into groups for measuring of data.

In study reviewed many studies have been conducted to increase classification in a variety of domains; example in Zahoor *et al.* [2020] (11) reported structural MRI data, distinguished individuals with schizophrenia and bipolar disease and health patients by using machine learning algorithms. SVM learning approach is created models using grey matter density pictures and similar studies reported in SVM-based product classification. In Mani [2023] (12), consumers are focusing their searches for a certain item in the data set by adding some value to the gates recurrent neural network (RNN) learning approach. And also, there is limited research on the proposed approach for product classification; the proposed approach

has proven that it is quite effective in a diversity of the classification's problems.

The amount of personal and public information saved online has grown over time. The article by Ho *et al.* [2023] (13), includes linguistic information gleaned from bloggers, newsgroups, review sites, and other channels of social media. It is possible to automatically convert this unstructured input into structured data that reflects popular sentiment by using writeup prediction models. In order to determine how customers, feel about specific apps, products, services, and brands, this relational database may then be employed. As a result, they can offer vital information for enhancing products and services. This style of sentiment analysis was used in the studies that followed.

Various information extraction approaches have been investigated to discover and organize text-based sentiments. Some studies (13-17) propose a model for annotating text with low-level demonstrations of sentiments, using a "scenario template" to summarize arguments based on personal judgment. These strategies are ideal for activities that require information from diverse sources.

In one study (14), a spin model for statistical analysis was recommended to extract semantic orientations from text. The proposed spin model computes estimated probabilities using the mean field approximation approach and provides a rating indicating whether the semantic orientation is acceptable or not. This approach generates highly accurate semantic orientations using only a small number of seed words from the English lexicon. In study (15), a sentiment analysis algorithm was proposed for summarizing ensembles of comments and reviews. The Gaussian distributions approach was used to overcome problems in sentiment analysis, which was a novel approach. Research (16) presented a machine learning strategy for predicting Google Play Store ratings using a database that included app classification, download quantity and size, download type, and Android version. Techniques such as k-means aggregation, k-nearest neighbors, SVMs, decision trees, linear regression, logistic regression, Naive Bayes classification, and artificial neural networks were employed to achieve the desired outcomes.

Mahmud *et al.* [2019] (17) analyzed the reviews and ratings on Google Play Store to prioritize categories. Umer *et al.* [2021] (18) explored different features, such as TF-IDF (Unigram, Bigram, and Trigram), and ensemble learning models to classify Google Play Store user ratings using machine learning models. They used TextBlob analysis to determine the sentiment of reviews and compared it with the star rating value to ensure accuracy. The machine learning models performed reasonably well, but require large amounts of high-quality data for training, and may not work well with imbalanced data. In contrast, deep learning models use word embedding techniques to learn features and perform well on challenging natural language processing problems. This is why deep learning models are used in this study.

Although many approaches have been used to explore user reviews in various aspects, including classification, summarization, analysis, and prediction of star ratings using different features, biased rating prediction based on actual user reviews has rarely been explored. Therefore, this study attempts to classify ratings as biased or unbiased using a deep learning model.

The paper is structured into different sections: (I) focuses on methods and techniques used in this work, (II) goes into detail on the results, (III) describes the discussion part, and (IV) concludes the work.

Methods

The dataset utilized for this work comprises 10,841 apps collected from the Google Play Store. This dataset is publicly available on the Kaggle website (19) and was last updated 2 months ago. *Figure 1* describes the Google Appstore dataset that offers comprehensive information provided by Google about the apps available on the Google Play Store. The dataset includes 13 attributes classified into three datatypes: String, Categorical, and Numeric. The Google Play Store Apps dataset available on Kaggle contains information about various Android apps. The attributes provide valuable information about the Android apps available on the Google Play Store, allowing for various analyses and insights.

Pre-processing

Tokenization is essential for processing application reviews and ratings. By converting raw text into individual tokens or words, it enables various analyses (20). Feature extraction: Tokenization helps extract meaningful features like words, phrases, or n-grams, capturing crucial patterns and information that influence ratings or sentiments expressed in the review. Sentiment analysis: Tokenization is a vital step in determining sentiment or emotion in text. Analyzing sentiment associated with individual tokens allows for

App	Category	Rating	Reviews	Size	Installs	Type	Price
Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0
Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0
U Launcher Lite – FREE Live Cool Themes, Hide Apps	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0
Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0
Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0
Paper flowers instructions	ART_AND_DESIGN	4.4	167	5.6M	50,000+	Free	0
Smoke Effect Photo Maker - Smoke Editor	ART_AND_DESIGN	3.8	178	19M	50,000+	Free	0
Text on Photo - Fonteee	ART_AND_DESIGN	4.4	13880	28M	1,000,000+	Free	0
Name Art Photo Editor - Focus n Filters	ART_AND_DESIGN	4.4	8788	12M	1,000,000+	Free	0
Tattoo Name On My Photo Editor	ART_AND_DESIGN	4.2	44829	20M	10,000,000+	Free	0
Mandala Coloring Book	ART_AND_DESIGN	4.6	4326	21M	100,000+	Free	0
3D Color Pixel by Number - Sandbox Art Coloring	ART_AND_DESIGN	4.4	1518	37M	100,000+	Free	0
Learn To Draw Kawaii Characters	ART_AND_DESIGN	3.2	55	2.7M	5,000+	Free	0
Photo Designer - Write your name with shapes	ART_AND_DESIGN	4.7	3632	5.5M	500,000+	Free	0
350 Diy Room Decor Ideas	ART_AND_DESIGN	4.5	27	17M	10,000+	Free	0
FlipaClip - Cartoon animation	ART_AND_DESIGN	4.3	194216	39M	5,000,000+	Free	0

Figure 1 The overview of the used database of Google Appstore.

calculating an overall sentiment score, aiding in identifying positive, negative, or neutral reviews. Text classification (20,21): Tokenization facilitates categorizing reviews into topics or classes. Converting reviews into a suitable format for machine learning or deep learning models enables training these models to classify feedback based on content. In summary, tokenization is crucial for processing and analyzing application reviews. It enables feature extraction, sentiment analysis, and text classification, enhancing the understanding of user feedback and improving the overall

user experience.

Here's the algorithmic representation of the code snippet for Word Embeddings tokenization using Word2Vec:

```
Tokenization:
Input: Text
Output: List of tokens
Word Embedding
Training: Input: List of tokens
Output: Word Embedding
Model Accessing Word
Embeddings:
Input: Word Embedding Model, Word(s)
Output: Word Embeddings or Similar Words
Example Usage:
Input: Word Embedding Model, Word(s)
Output: Word Embedding Vector(s) or Similarity Score
```

Statistical analysis of classification techniques

Decision tree algorithm (DTA)

DTA is a tree-based technique that involves building multiple decision trees and then combining their results to improve a model's ability to generalize. This process is known as an ensemble approach, which aims to create a strong learner from a mixture of weak learners. In the RF approach, a group of trees is used as classifiers, denoted as h(x,g), where k is an independently and identically distributed random vector. In previous studies, random vectors have been used to divide trees into different inputs (22,23). The proposed RF algorithm generates a random finite set of trees discretely. The basic steps of this algorithm are as follows:

- Randomly select "N" records from the finite data set.
- Create a decision tree using the "N" selected records.
- Repeat steps 1 and 2 until the end of the operation.
- List out the decision trees generated in steps 1 and 2.

After generating multiple decision trees, the most voted class is assumed to be the original group. Ensemble techniques combine the outputs of "n" different trees to create a more accurate and consistent prediction. In other words, a RF generates numerous decision trees and combines their outputs to improve the overall accuracy of the model.

SVM classifier

The SVM classifier is a supervised Machine Learning technique that aims to generate an optimal hyper-plane to classify new samples based on the available training data (Sharma *et al.*, 2022) (24). Supervised learning symbolically relates where a model is trained using labeled examples (input data with corresponding target outputs) in order to predict unknown data. Although SVMs were first introduced in the 1960s, they gained popularity only in the 1990s due to their ability to produce remarkable results. Sentiment analysis, classification, and prediction are different tasks performed with a robust and reliable approach to extracting valuable insights.

In essence, a basic Machine Learning technique seeks to create a boundary that separates the data in a way that minimizes the misclassification error for linearly separable data in 2D, as shown in *Figure 2*. From *Figure 2*, it can be seen that there are several boundaries that correctly separate the data points. The two dashed lines and one solid line properly classify the data.

SVM differs from previous classification approaches by choosing a decision boundary that maximizes the distance between the nearest data points for all classes. SVM not only finds the decision boundary but also classifies the data using the best decision boundary. One of the most optimal decision boundaries is the one that has the largest margin among all the neighboring points of the classes.

The basic SVM algorithm cannot be applied to nonlinearly separable data. Instead, a modified form of SVM called Kernel SVM is used. Essentially, Kernel SVM converts nonlinearly separable data in lower dimensions to linearly separable data in higher dimensions, so that data points from different classes are assigned to different scopes.

RF classifier

RF is a popular supervised machine learning algorithm that is commonly used for classification and regression tasks. It employs an ensemble approach where multiple decision trees are constructed and combined to solve complex problems and improve model performance. The name "Random Forest" is derived from the fact that it comprises a large number of decision trees generated on different subsets of the input data, each aiming to improve the

```python
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize

def tokenization(text):
    tokens = word_tokenize(text)
    return tokens

def word_embedding_training(tokens):
    model = Word2Vec(sentences=[tokens], size=100, window=5, min_count=1
    return model

def access_word_embeddings(model, word):
    embedding = model.wv[word]
    return embedding

def find_similar_words(model, word):
    similar_words = model.wv.most_similar(word)
    return similar_words

def measure_similarity(model, word1, word2):
    similarity = model.wv.similarity(word1, word2)
    return similarity
```

Figure 2 Overview of tokenization algorithm.

projected efficiency of the model.

Unlike a single decision tree, RF (22) combines the predictions from all trees and generates a final output based on the majority vote. This approach not only produces more accurate results but also helps avoid overfitting issues. The RF algorithm can be executed using Anaconda Prompt software, as illustrated in the accompanying image. By following a few simple steps, the user can access Jupiter Notebook, where they can enter the code and run it line by line to produce the desired output.

Here's how the steps can be arranged in the form of an algorithm:

(I) Open the search bar and search for "Anaconda Prompt" software.
(II) Click on the "Anaconda Prompt" application to open the command prompt.
(III) Type "cd [directory name]" to change the directory to the desired location on your computer.
(IV) Press "Enter" to execute the command and change the directory.
(V) Type "ipython notebook [file name].ipynb" to open the Jupyter Notebook for the desired file.
(VI) Press "Enter" to execute the command and open the Jupyter Notebook in a new browser tab.
(VII) Scroll down to the code cells in the notebook.

Stochastic gradient boosting (SGB) classifier

SGB is a machine learning technique that incorporates stochastic gradient descent optimization with gradient boosting. By adding randomness to the training process, SGB is able to improve the speed and scalability of traditional gradient boosting.

SGB (25,26) functions by iteratively adding decision trees to the model. Each new tree aims to correct the errors made by the previous trees. However, unlike traditional gradient boosting, SGB fits each new tree to a random subset of the training data, rather than the residual errors of the previous trees.

The randomness introduced in SGB helps prevent overfitting and makes it more robust against noisy data. Additionally, the use of smaller subsets of data in each iteration allows for faster computation and parallel processing.

SGB (26) is effective in various applications, including classification, regression, and ranking problems. It has become a popular choice among machine learning practitioners due to its ease of use and strong performance.

Gated recurrent unit (GRU) classifier

The GRU is an advanced version of RNNs that was introduced by (12). It is commonly utilized to tackle classification problems and it solves the "vanishing gradient" issue that is typically associated with standard RNNs (27). The GRU employs two gates—the update gate and the reset gate—to handle the vanishing gradient issue that standard RNNs encounter (12,27). The gates can learn to keep relevant information from the past while discarding irrelevant information, and pass the relevant information to future events to improve prediction accuracy. The update gate controls the amount of information from previous time steps that should be passed on to future steps (27). In contrast, the reset gate decides how much of the previous information (history) should be disregarded by the network. Using a GRU can enhance the RNN's memory capacity and make it easier to train the model. The GRU architecture utilizes an update gate to regulate the flow of information from the past to the future, and a reset gate to decide how much previous information should be discarded. This design enhances the network's capacity to retain significant data and make better predictions. Adopting GRUs instead of standard RNNs can improve the network's memory capacity and ease the training process. This is especially beneficial for applications such as automatically identifying and categorizing requirements from app reviews, where accuracy and efficiency are critical for achieving success. The GRU architecture is very adaptable and simple as compared to long short-term memory (LSTM) which helps in faster computation speed and memory efficiency (27).

There are various advantages of GRU as compared to RNN and LSTM (28) based on specific databases and tasks.

To implement the GRU algorithm in Jupyter Notebook, you can follow these steps:

(I) Open Jupyter Notebook on your computer.
(II) Create a new notebook or open an existing one.
(III) Import the required libraries, such as NumPy, Pandas, TensorFlow, or Keras, depending on the implementation.
(IV) Load the dataset you want to use for training and testing the model.
(V) Preprocess the data as required, such as splitting it into training and validation sets, or encoding the features and labels.
(VI) Define the GRU model architecture, including the number of layers, number of neurons, activation functions, and other hyperparameters.
(VII) Compile the model by specifying the loss function, optimizer, and evaluation metrics.
(VIII) Train the model using the training data and validate it using the validation data.
(IX) Evaluate the model's performance on the test data and generate predictions on new data.
(X) Visualize the results and fine-tune the model as required to improve its accuracy and generalization.

Algorithmic representation:

(I) Initialize GRU model parameters randomly or using a predefined scheme.
(II) For each training iteration:
    i. Compute update gate (z) and reset gate (r) for current input and previous hidden state.
    ii. Compute candidate hidden state (h~) by applying activation function to a linear combination of input and reset gate multiplied by previous hidden state.
    iii. Compute current hidden state (h) by combining previous hidden state and candidate hidden state using the update gate.
    iv. Compute GRU model output for current input.
    v. Compute loss between predicted output and target output using a suitable loss function.
    vi. Backpropagation through time:
        a) Compute gradients of loss with respect to model parameters using gradient descent.
        b) Update model parameters using an optimization algorithm to minimize the loss.

Each step can be implemented using code cells in the Jupyter Notebook. You can execute each cell individually or run the entire notebook at once. It is recommended to comment the code to explain what each section does and to make it easier to understand and modify the implementation later.

Results

A system's logical design focuses on the flow of information, inputs, and outputs within the system. It is commonly accomplished through modeling, which involves creating an idealized representation of the actual system. Object association diagrams, or RR diagrams, is a type of logical design.

Physical architecture

On the other hand, the physical design of a system is concerned with the actual input and output processes of the system. This involves how information is received, validated/authorized, processed, and displayed as output. The architectural design phase determines the system's requirements, including input, output, storage, processing, and system management, including backups/retrieval.

To simplify, the visual appearance of a system design proposal is divided into three subtasks: (I) designing the user interface, (II) visualizing data, and (III) proposing the process. User interface involves the operator's input data in the system, while data is expressed and stored within the system in a process called data proposal. Finally, system planning is the process by which data is validated, secured, and/or altered by way of travels into, throughout, and accessible to the system. Certification defines three subtasks, which are developed and completed, and full availability is the following phase of the completion of the system design segment.

In this sense, the physical layout of a system does not refer to the physical appearance of the data or information within the system. For example, the physical design of a personal computer includes input through the controls, processing by the central processing unit (CPU), and outputs through a screen, printer, and other devices. This would not affect the physical layout of the hardware, which would include a screen, motherboard, CPU, hard disks, graphics cards, modems, universal serial bus (USB) ports, and so on. It involves creating a detailed operator and producing a catalog of computer building components, as well as a switch workstation. For a proposed system, individual hardware requirements are created.

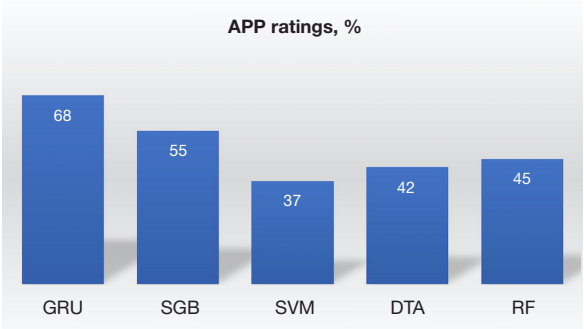

Figure 3 A bar chart that presents the anticipated outcomes of various algorithms based on the "reviews" feature. GRU, gated recurrent unit; SGB, stochastic gradient boosting; SVM, support vector machine; DTA, decision tree algorithm; RF, random forest.

Input design

Input design refers to the process of establishing a connection between the user and the system through input devices. This can be achieved by scanning documents, entering data directly into the system, or using the user's critical information system.

The goal of input design is to minimize the number of required inputs, reduce errors, eliminate unnecessary steps, and streamline the entire input process. Input design also ensures security and suitability while maintaining confidentiality. The following factors are taken into consideration when designing the input system:

- What data is required for input?
- How should the data be organized?
- Should any assistance be provided to help users enter data?
- How can input authentication be created and what should be done in case of errors?

Purpose

The main purpose of input design is to translate a user-oriented input description into a computer-based system. A well-designed input system is crucial for avoiding data entry errors and guiding users in collecting accurate information from the system. The design of user-friendly data entry panels helps to manage and avoid errors while making data entry easier and more efficient. This error-free and easy-to-use panel setup enables users to perform all required data operations and allows them to view and verify submitted records. Thus, the primary objective of input design is achieved and facilitates easy retrieval of data whenever required through an intuitive input system.

Output layout design

The primary goal of designing a high-quality output layout is to meet the user's needs and expectations and to present the required information accurately and clearly. The developers strive to achieve user satisfaction by creating an efficient and intelligent interface between the user and the system. To achieve this goal, the following steps are taken:

(I) Developing a well-planned and systematic approach for generating the system's output.
(II) Designing an appropriate output layout that appeals to users and meets their requirements.
(III) Creating a simple and precise output layout that meets user interaction standards.
(IV) Deciding on how the data should be presented in the output.
(V) Creating reports in specific user-required formats for better visibility and understanding of the output.

App: The title of the app

Category: The app's classification

Rating: The score given by users who rated the app

Reviews: The total number of reviews submitted by users

Size: The amount of storage space required by the app

Install: The number of times the app has been installed

Type: Whether the app is available for free or has a cost

Price: The cost of the app (if it's not free, otherwise 0)

Content Rating: The age group for which the app is deemed appropriate

Genres: The category or genre of the app

Last Updated: The date when the app was last modified

Current Ver: The version number of the app

Android Ver: The minimum version of the Android operating system required to use the app.

Discussion

The work showcases a variety of graphs that illustrate how different algorithms produce outputs for the considered features. The classification features included in the study are RF, DTA, SVM, SGB and GRU. *Figures 3*-7 present

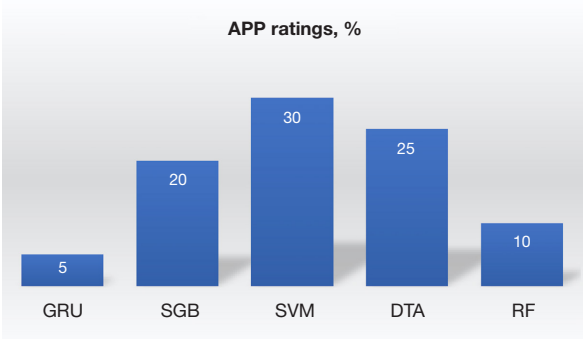

Figure 4 A bar chart that illustrates the expected results generated by all the algorithms mentioned above using the "price" feature. GRU, gated recurrent unit; SGB, stochastic gradient boosting; SVM, support vector machine; DTA, decision tree algorithm; RF, random forest.

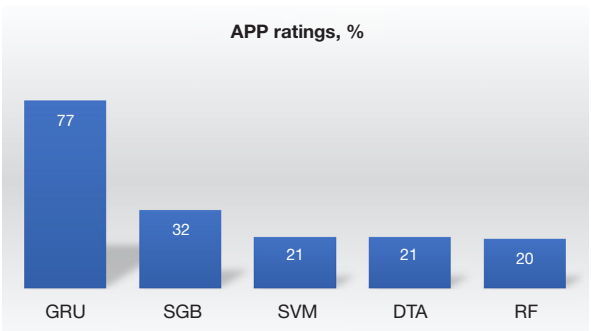

Figure 5 The predicted outcomes given by all the algorithms mentioned above based on the "rating score" feature. GRU, gated recurrent unit; SGB, stochastic gradient boosting; SVM, support vector machine; DTA, decision tree algorithm; RF, random forest.

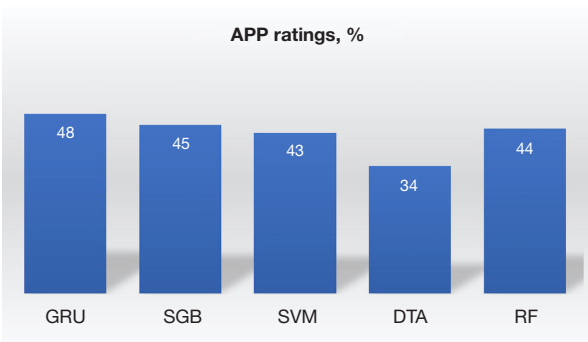

Figure 6 Bar chart that presents the predicted results provided by all the algorithms mentioned above based on the "size" feature. GRU, gated recurrent unit; SGB, stochastic gradient boosting; SVM, support vector machine; DTA, decision tree algorithm; RF, random forest.

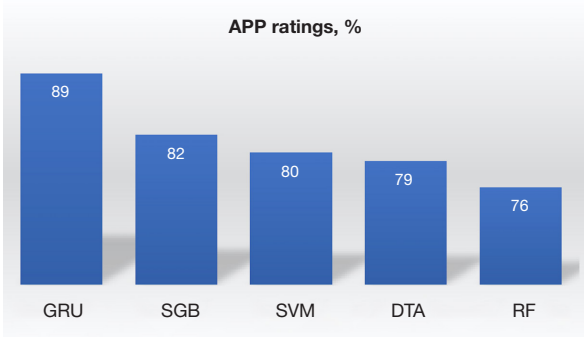

Figure 7 Bar chart displays the expected final results of several algorithms based on their accuracy, as shown on the Y-axis. GRU, gated recurrent unit; SGB, stochastic gradient boosting; SVM, support vector machine; DTA, decision tree algorithm; RF, random forest.

a comparison of the accuracy of the various algorithms in relation to reviews, prices, and apps, respective.

The X-axis shows different algorithms, while the Y-axis shows the accuracy of the output based on reviews. The output results for different algorithms such as RF =45%, DTA =42%, SVM =37%, SGB =55% and GRU =68% are displayed. The GRU is the best among the four algorithms, as it has the highest accuracy in terms of positive evaluations, which leads to more precise findings.

The X-axis represents different algorithms, while the Y-axis indicates the accuracy of the output based on pricing. The output outcomes for different algorithms such as RF =10%, DTA =25%, SVM =30%, SGB =20% and GRU =5% are shown. The GRU prediction is the best among the four algorithms, as the price for any classification should be as low as possible, while the results should be more precise.

The X-axis shows different algorithms, while the Y-axis indicates the predicted rating score. The output outcomes for different algorithms such as RF =20%, DTA =21%, SVM =21%, SGB =32% and GRU =77% are displayed. The GRU estimation is the best among the four algorithms, as the rating-score value for any classification should be as high as possible to ensure more precise findings.

The X-axis shows different algorithms, while the Y-axis indicates the size in percentages. The output outcomes for different algorithms such as RF =44%, DTA =34%, SVM =43%, SGB =45% and GRU =48% are shown. The GRU technique predicted results are the best among the four algorithms, as the size of the predicted apps in every classification should be as small as possible to make them more attractive for people to use.

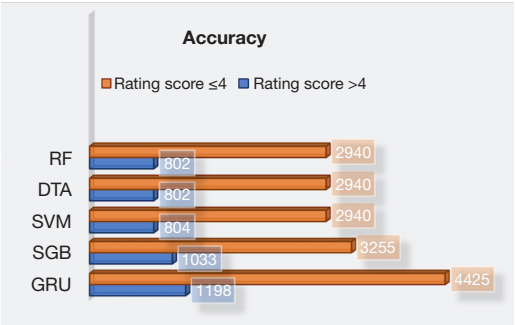

Figure 8 The bar chart represents the number of apps predicted with ratings less than and equal or greater than 4 using different algorithms, as shown in blue and red bars, respectively based on accuracy. RF, random forest; DTA, decision tree algorithm; SVM, support vector machine; SGB, stochastic gradient boosting; GRU, gated recurrent unit.

The X-axis represents the different algorithms, including RF with an output of 0.76, DTA with an output of 0.79, SVM with an output of 0.80, SGB with an output of 0.82, and GRU with 0.89. The GRU Algorithm outperforms the other algorithms significantly in terms of accuracy, which is crucial for any categorization result.

The expected final results of several algorithms based on their rating feature are shown in this bar chart. The Y-axis displays the different algorithms, and the X-axis represents the number of apps predicted with ratings less than and equal or greater than 4, as shown in blue and red bars, respectively. The algorithms include RF with an output of 802:1,120, DTA with an output of 802:1,310, SVM with an output of 804:1,140, SGB with 1,033, and GRU with an output of 1,198:2,123. Although the first five algorithms produce similar outcomes, the GRU Algorithm predicts significantly better than the others. It is essential to predict as many highly-rated apps as possible since users will be more interested in using them.

The *Figure 1* above shows the classification results based on various features of apps, including rating, reviews, size, installs, type, price, content rating, genres, and last update. The study focused on the art and design area of apps and categorized over ten apps based on their features to help users choose the best app for their needs. Similarly, different types of apps can be classified based on the given parameters.

The *Figure 8* above displays the projected final outcomes of the GRU classifier, with an overall accuracy of 0.89. Out of all the classifiers, GRU classified almost 5,616 apps, while the other classifiers only classified 3,744 apps. When considering various predictions, the GRU classifier provides the most precise results for both users and developers.

Conclusions

This dataset contains a significant amount of data that can be utilized in various ways. Moreover, future developers and the Google Play Store team could use the GRU model generated using this dataset to analyze the Android Market and determine which application categories should be emphasized to make the Android Market more appealing in the long run. This could increase the value of a business or the Play Store as a whole. Our research was not limited to the problem at hand; we explored various classification methods on the large dataset and concluded that the GRU was the most suitable for our problem definition. Additionally, developers discovered that certain methods worked better in certain situations. Researchers found that decision trees make model implementation easy to understand and explain while also saving processing power. Future research is focused on regression analysis, which includes the number of responses and app installations. This approach identifies different groups and statistics of the maximum number of installed apps and determines the similarity between app size and versions.

Acknowledgments

Funding: None.

Footnote

Peer Review File: Available at https://jmai.amegroups.com/article/view/10.21037/jmai-23-58/prf

Conflicts of Interest: Both authors have completed the ICMJE uniform disclosure form (available at https://jmai.amegroups.com/article/view/10.21037/jmai-23-58/coif). The authors have no conflicts of interest to declare.

Ethical Statement: The authors are accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved.

References

1. Guendouz M, Amine A. A New Feature Selection Method Based on Dragonfly Algorithm for Android Malware Detection Using Machine Learning Techniques. International Journal of Information Security and Privacy (IJISP) 2023;17:1-18.
2. Aslam N, Alzamzami O, Xia K, et al. Improving the review classification of Google apps using combined feature embedding and deep convolutional neural network model. Journal of Ambient Intelligence and Humanized Computing 2023;14:4257-72.
3. Mawardi VC, Darmaja E. Logistic Regression Method for Sentiment Analysis Application on Google Playstore. International Journal of Application on Sciences, Technology and Engineering 2023;1:243-9.
4. Aralikatte R, Sridhara G, Gantayat N, et al. Fault in your stars: an analysis of android app reviews. In Proceedings of the ACM India Joint International Conference on Data Science and Management of Data 2018:57-66.
5. Sadiq S, Umer M, Ullah S, et al. Discrepancy detection between actual user reviews and numeric ratings of Google App store using deep learning. Expert Systems with Applications 2021;181:115111.
6. Venkatakrishnan S, Kaushik A, Verma JK. 2020. Sentiment analysis on google play store data using deep learning. Applications of Machine Learning. Singapore: Springer, 2020:15-30.
7. Suleman M, Malik A, Hussain SS. Google play store app ranking prediction using machine learning algorithm. Urdu News Headline, Text Classification by Using Different Machine Learning Algorithms, 2019;57.
8. Banumathy D, Khalaf OI, Romero CAT, et al. Breast calcifications and histopathological analysis on tumor detection by CNN. Comput Syst Sci Eng 2023;44:595-612.
9. Maheswari RV, Subburaj P, Vigneshwaran B, et al. Nonlinear support vector machine-based partial discharge patterns recognition using fractal features. Journal of Intelligent & Fuzzy Systems 2014;27:2649-64.
10. Adeola O, Edeh JN, Evans O, et al. Africa's Digital Marketplace: The Role of Social Media in Customer Engagement. In: Adeola O, Edeh JN, Hinson RE. editors. Digital Business in Africa: Social Media and Related Technologies. Cham: Springer International Publishing, 2022:145-68.
11. Zahoor S, Lali IU, Khan MA, et al. Breast Cancer Detection and Classification using Traditional Computer Vision Techniques: A Comprehensive Review. Curr Med Imaging 2020;16:1187-200.
12. Mani S, Gunasekaran G, Geetha S. Email spam detection using gated recurrent neural network. International Journal of Progressive Research in Engineering Management and Science (IJPREMS) 2023;3:90-9.
13. Ho SPS, Chow MYC. The role of artificial intelligence in consumers' brand preference for retail banks in Hong Kong. Journal of Financial Services Marketing 2023. Doi: 10.1057/s41264-022-00207-3.
14. Sprugnoli R, Tonelli S. Novel event detection and classification for historical texts. Computational Linguistics 2019;45:229-65.
15. Takamura H, Inui T, Okumura M. Extracting semantic orientations of words using spin model. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05). Association for Computational Linguistics, USA, 2005:133-40.
16. Abulhaija S, Hattab S, Abdeen A, et al. Mobile Applications Rating Performance: A Survey. International Journal of Interactive Mobile Technologies 2022;16:133-46.
17. Mahmud O, Niloy NT, Rahman MA, et al. Predicting an effective android application release based on user reviews and ratings. 2019 7th International Conference on Smart Computing & Communications (ICSCC); 28-30 June 2019; Sarawak, Malaysia. IEEE, 2019:1-5.
18. Umer M, Ashraf I, Mehmood A, et al. Predicting numeric ratings for google apps using text features and ensemble learning. ETRI Journal 2021;43:95-108.
19. Fuad A, Bayoumi S, Al-Yahya H. A Recommender System for Mobile Applications of Google Play Store. International Journal of Advanced Computer Science and Applications 2020;11:42-50.
20. Kathuria A, Gupta A, Singla RK. A Review of Tools and Techniques for Preprocessing of Textual Data. In: Singh V, Asari V, Kumar S, et al. editors. Computational Methods and Data Engineering. Advances in Intelligent Systems and Computing. Singapore: Springer, 2021;1227.
21. Wahyuni WA, Saepudin S, Sembiring F. Sentiment Analysis of Online Investment Applications on Google

Play Store using Random Forest Algorithm Method. Journal Mantik 2022;5:2203-9.
22. Fitri VA, Andreswari R, Hasibuan MA. Sentiment analysis of social media Twitter with case of Anti-LGBT campaign in Indonesia using Naïve Bayes, decision tree, and random forest algorithm. Procedia Computer Science 2019;161:765-72.
23. Aufar M, Andreswari R, Pramesti D. Sentiment analysis on youtube social media using decision tree and random forest algorithm: A case study. 2020 International Conference on Data Science and Its Applications (ICoDSA); 05-06 August 2020; Bandung, Indonesia. IEEE; 2020:1-7.
24. Sharma D, Singh P, Kumar A. A Comparative Study of Classical and Quantum Machine Learning Models for Sentimental Analysis. arXiv:2209.05142 2022. [Preprint]. Available online: https://doi.org/10.48550/arXiv.2209.05142
25. Santhosh Baboo S, Amirthapriya M. Comparison of machine learning techniques on Twitter emotions classification. SN Computer Science 2022;3:35.
26. Naeem MZ, Rustam F, Mehmood A, et al. Classification of movie reviews using term frequency-inverse document frequency and optimized machine learning algorithms. PeerJ Comput Sci 2022;8:e914.
27. Agarap AF. Statistical analysis on E-commerce reviews, with sentiment classification using bidirectional recurrent neural network (RNN). arXiv preprint 2018. arXiv:1805.03687.
28. Adyasha Pattanaik P, Wang Z, Horain P. Deep CNN frameworks comparison for malaria diagnosis. arXiv e-prints 2019. arXiv:1909.02829v1.

doi: 10.21037/jmai-23-58
Cite this article as: Pattanaik P, Nagpal D. Comparison of machine learning algorithms used to catalog Google Appstore. J Med Artif Intell 2023;6:22.