**MSc Data Science for Business**
***Research Paper***

*Academic Year 2020-2021*

# How does APP icon design affect download times?

**Delong LI**

Under the supervision of:

Prof. Cathy Yang

Public Report

# Executive Summary

Nowadays, with the popularization of smartphones, mobile app market has become unprecedented competitive. How to attract mobile phone users' attention while browsing app store has arisen as an important and meaningful problem for app developers.

This study focused on two visual elements that are underexplored in previous works, consistent text and face characteristics. Icon images are scraped using a web crawler and computer vision algorithms are applied to detect two features from the images. To obtain causal inference insights on downloading time, coarsened exact matching is then conducted to remove selection bias on other variables. The final findings of the paper are twofold. Neither consistent text nor face characteristics has a significant effect on the user's downloading behaviour.

This work contributes to the growing literature on mobile app by exploring concrete visual elements on an observable dataset with AI-based methodologies and assist app developers in understanding the user's visual preference better.

# Preface and Acknowledgements

I would like to thank my tutor Cathy YANG, who gave me clear guidance on research design as well as useful tips for collecting data. Her timely help and expertise in causal inference analysis are indispensable to the success of this study.

Moreover, I was appreciated to utilise some open-source data and projects, which allowed me to save a huge amount of time throughout the study. Thanks to Gautham Prakash and Jithin Koshy, I directly retrieved dataset and performed analysis.

All along the master, I am appreciated to have met wonderful professors and teaching assistants who helped me gained solid knowledge and skills, which enable me to conduct the study by myself. In the end, thanks to my parents, friends and classmates who always support and encourage me during my study in France.

# Table of Contents

# 1.Introduction

Thanks to the fast development of microchips and enormous progress in cellular network, digital devices are ever more indispensable for the internet users all over the world. As we spend more and more time on digital devices, most of which on mobile applications (hereafter referred to as 'apps'), the mobile application market is expanding at an unprecedented speed. The total number of app downloads from app stores worldwide is estimated 266 billion in 2021[1]. On Google Play Store, the biggest application store worldwide, 3739 apps are added every day on average. [2]

On almost all the popular operating systems, apps are presented through an icon, a visual sign, with a representative meaning, delivers information and is easy to identify and recall [3]. Icons essentially act as a first-pass filter for saturated app markets, which is why the visual design needs tactics in order to immediately capture a consumer's attention. Developers and designers make great efforts on app icon this to make sure it conveys the most appropriate message as well as attract potential users as much as possible.

In this context, a question arises: "Which visual elements or characteristics make a good app icon design?" Moreover, "How do these factors affect consumer's behaviour? More precisely, download times."

Despite the rapid growth of the mobile app market, little attention has been given to the visual attributes of apps and especially how do these attributes affect the download behaviour of users. In existing literature, only several aesthetic features were studied through experiments.  Nowadays, state-of-art algorithms in computer vision have enormously extended the research scope on images but also enhanced the scalability of application.  Therefore, this research paper intends to harness the power of artificial intelligence and expand the research on app icons to more concrete visual elements: text and facial characteristics. The key objective of this research is to check out whether text and facial characteristics on app icon have a significant influence on an app's download.

The rest of this paper is organized as follow: Section 2 introduces the previous literature and my hypothesis for this research. Section 3 presents the dataset in use as well as the data preparation pipeline. Section 4 covers the computer vision techniques to extract text and facial characteristics from icon images. In Section 5, I tried out several matching techniques to evaluate the causal effects. Section 6 summarizes the results as well as implications and proposes possible improvements in the future.

# 2. Background and Hypothesis

## 2.1 Related work

In this day and age, app popularity and user's downloading intention is a fast-growing topic, which has already received enough attention in both commercial and academic world. According to a study by Yahoo conducted in 2016, the top prompts for downloading new apps are: Looking for something new; recommendations by friends or others; solving specific problems [4]. Another report from TUNE published in 2015 almost revealed the same findings: specific tasks and word-of-mouth are dominant motivations to download an app [5]. Racherla et al. (2012), pointed out that 'top' lists and ordering effect, information cascades (number of downloads) and word-of-month network effect collectively as well as independently affect user decision. Hsu and Lin (2015) have found that value-for-money, app rating and free alternatives to paid apps were found to have a direct impact on intention to purchase paid apps [6].

Factors for downloading behaviour seem to be similar and consistent in previous works. Few of them focused on intrinsic features of app itself, but these findings give valuable suggestions on which variables to be controlled and treated for the following studies.

On the other hand, Iconograph as a separate research area has evolved from concept of signs to interactive representations on mobile devices. In the following of the paper, icon refers to the visual sign that represents an application on app store. Icons facilitate human-computer interaction because they are swiftly recognized and memorized [7]. A good icon design not only clearly represents the functions and identity of an application, but also stands out among its counterparts and earns more people's attention.

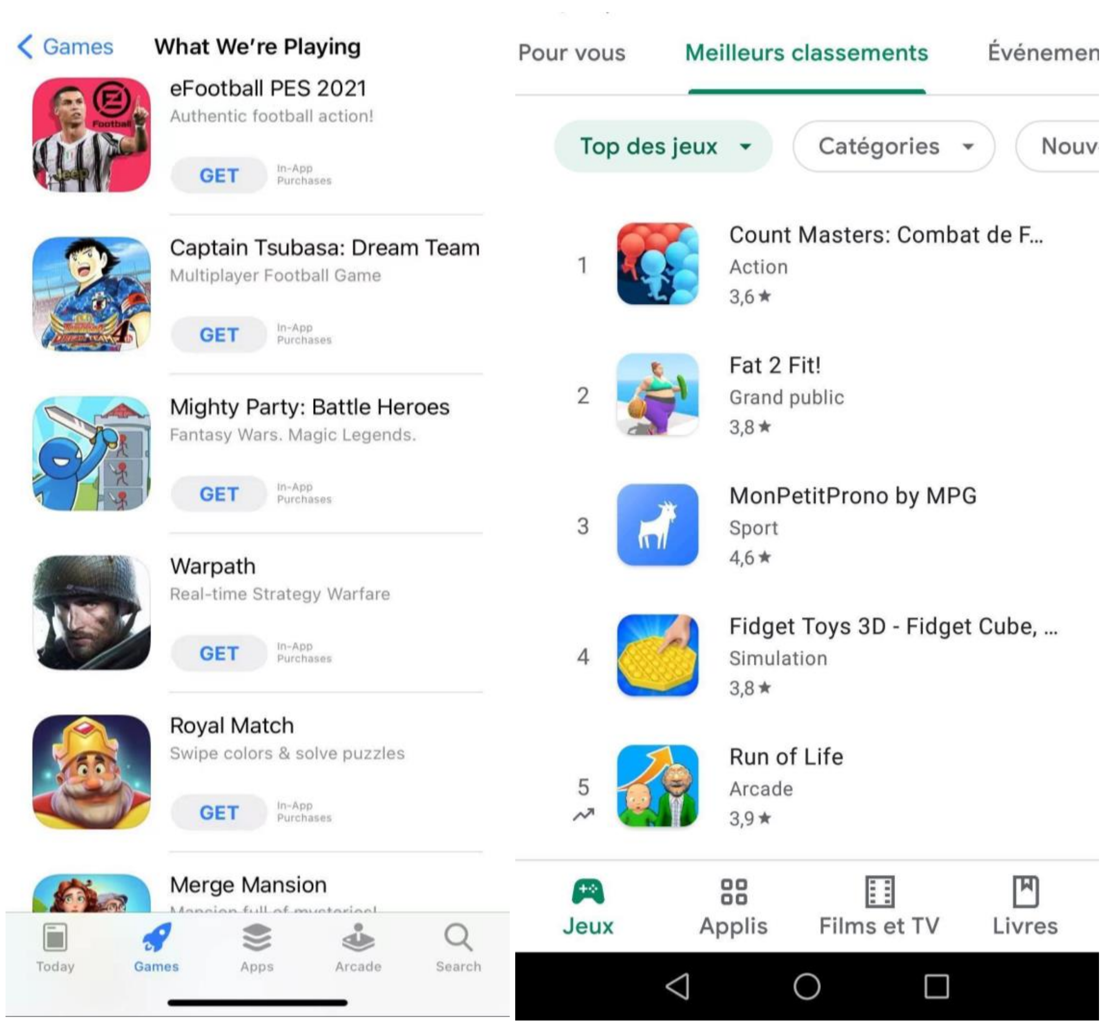


Figure 1 The Game apps on Apple Store and Google Play

Some previous studies have already focused on the design of icon and its effects. Goonetilleke et al. (2001), selected icons that represent daily activities and conducted experiments on the matching of these icons. Their results suggested ambiguity, uniqueness and dominance are three important aspects to consider when designing and developing icons [8]. Shu and Lin (2014) justified the significance of five attributes of game icon design through experiments, namely, active design, balance, complexity, depth, and organic value [9]. Wang and Li (2017) studied the effects of colour, complexity, and symmetry of the mobile app icon on number of downloads. Particularly, they found that apps with icons featuring higher colourfulness, proper complexity, and slight asymmetry lead to more downloads [10]. From a customer perception aspect, Jylhä and Hamari (2019) asserted that icons which are perceived beautiful, good and unique project a positive overall evaluation and willingness to click the app icon as well as download and purchase the imagined app [11].

To summarize, even though existing studies covered a wide range of topics, they are still quite limited in certain aspects. Aesthetic attributes like colour, complexity and uniqueness had clearer and more standardized measurement as well as theoretical foundation thus gained more attention than other visual elements. Additionally, most of the researchers conducted experiments, but few studies tackled the causal inference between visual attributes and downloads with observable data.

## 2.2 Text on Icons

In the academic domain, we could consider that icons are completely analogic to visual representations, or pictures. The relationship between text and pictures has always been an active research area in cognitive theories. Visual and verbal representations always appear side by side or even overlapped. In this context, text–image congruence facilitates the impression formation process by allowing consumers to form a clear and acute product image [12]. "The pictures double or parallel what is said in the text,'' but ''there is never complete redundancy because the picture is more concrete than the word'' [13].

Brand logo is a typical example in graphical design and image processing. It is a vital element in building corporate visual identity, and it has been found to accelerate users' recognition of a company or brand [14]. Bresciani and Del Ponte (2017) stated that logos composed of an icon plus a brand name are perceived as significantly more attractive than logos made of one component only [15].

As for icon design field, text did not receive too much attention. Some researchers such as Paivio (1971) have argued that multiple modalities enhance memorability and hence text and graphics together may be more effective than pure graphics [16]. Jylhä and

Hamari (2019) selected text as one of their four categories of app icon to study customer perceptions on app icons [11]. But unlike in the area of brand logo and printed ads, no study seems to touch upon the relationship between containing text on mobile app icon and customers' download behaviour using observable data.

To extend the above findings to the app icon's case, it is reasonable to hypothesize that appropriate text elements on the surface of icon will enhance the congruence of overall design thus makes the app more attractive. This is the first main effect that will be explored in this study.

**Hypothesis 1.** *Incorporating text that consistent with app name into icon design will have a positive effect on customer's download behaviour.*

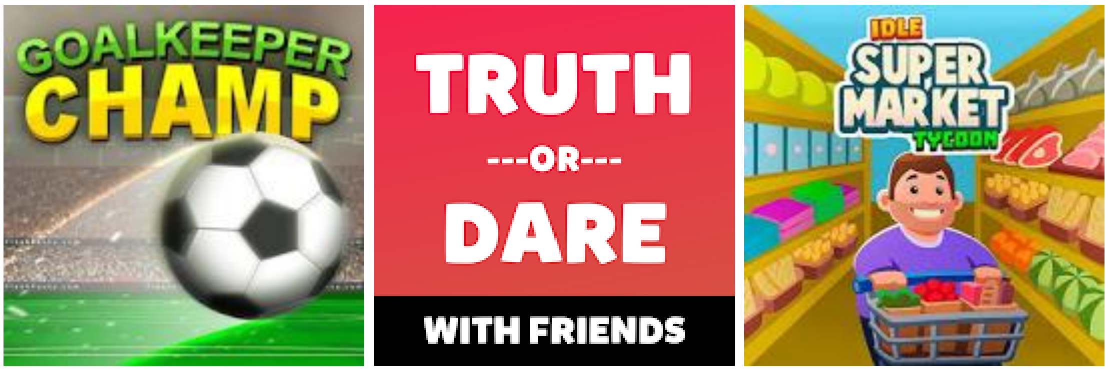


Figure 2. Examples of game icons containing text

## 2.3 Spokes-Characters and People on Icons

Cartoon characters, precisely spokes-characters, are prevalent elements usually seen on app icons. The spokes-characters at its most basic is defined as a fictional persona employed to sell a product or service [17]. For more than a century, marketers and advertisers have utilized spokes-characters in promotional campaigns and on product packages (Callcott 1993). Some of the earliest characters include the Michelin Man and the Cream of Wheat Chef [18]. Specifically, the characteristics (e.g., likability, expertise, and attractiveness) help shape brand perceptions (Callcott and Alvey,1991) [19]. The reason for its appeal is hard to determine. Some argued that features like face on app icons are widely used because of the immediate impact and memorability they have due to the neural processing of facial expressions [20].

Animated character has always been an active research topic in advertising. According to Huang et al. (2011), advertisements endorsed by animated spokes-characters enhance brand impression, improving advertisement communication effects; however, purchase intention is not guaranteed [21]. Li and Min (2014) pointed out that different faces do have an effect on people's attitude toward the advertisement, attitude toward the brand, and purchase intention [22].

As to app iconography, spokes-characters have just received some attention in the latest studies. Zhang et al. (2020) targeted a specific question: incompleteness and completeness of spokes-characters. Experimental results reveal that incomplete spokes-character faces create perceptions of anthropomorphism, and thus enhance users' brand evaluations. Here, anthropomorphism refers to the attribution of human characteristics to non-human entities.

Based on these previous findings, we can conclude that spokes-characters, more importantly, their human characteristics do have a significant effect on both user's attitudes and behaviour. Similar to the topic of text, mobile app is an emerging research area. There is still a research gap in terms of studying the effects on downloads using observable data. This leads me to propose the following hypothesis:

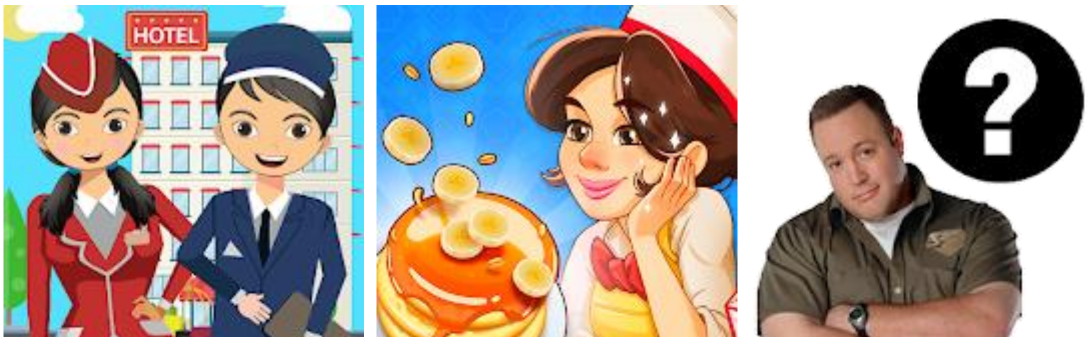


Figure 3. Examples of game icons containing characters or people

**Hypothesis 2.** *Incorporating spokes-characters or portrait photos into icon design will have a positive effect on customer's download behaviour.*

## 2.4 Research Framework

To ensure a higher enteral validity, this research is conducted on an observable as well as comprehensive dataset (see Section 3) rather than on experimental data. At the same time, feasibility of large sample analysis is assured by advanced computer vision algorithms which can recognise text and facial characteristics on a large scale (see Section 4). Finally, matching is applied to control selection bias on all the other covariates, and treatment effects on download times are estimated based on matched data (see Section 5).

Figure 4 illustrates the general research framework.

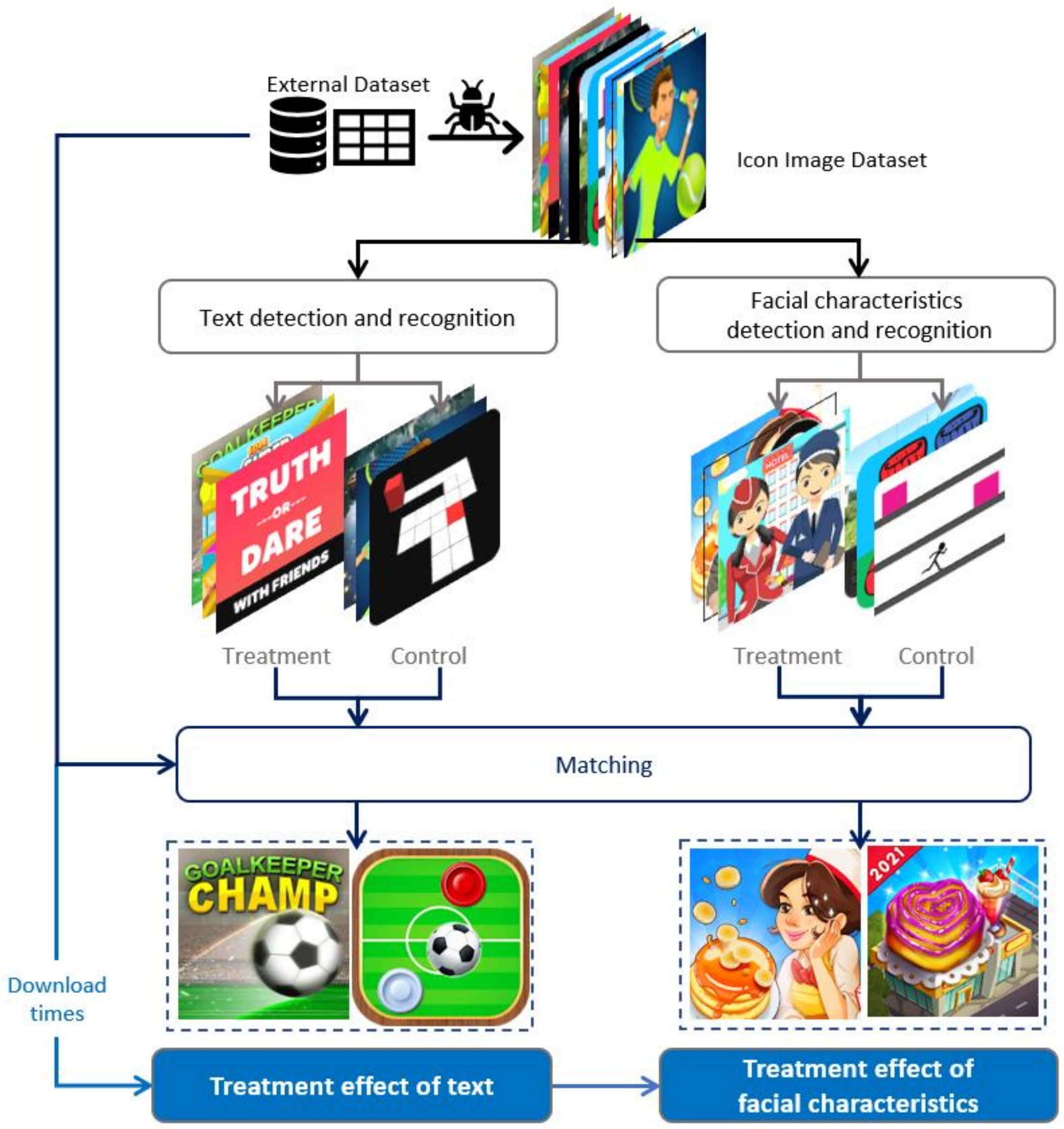

External Dataset
Icon Image Dataset
Text detection and recognition
Facial characteristics detection and recognition
Treatment
Control
Treatment
Control
Matching
Download times
Treatment effect of text
Treatment effect of facial characteristics

Figure 4 Research Framework

# 3. Data

## 3.1 Data Collection

Data is especially crucial to non-experimental research. Due to the limit of time and budget, two possible approaches are feasible: web scraping and online open data sources. After examination of main app stores, I found some critical information including the exact download times and released date are not accessible to the public. Fortunately, thanks to the contribution of Prakash and Koshy, an open dataset of Google Play Store is available on GitHub and Kaggle [23]. Updated in December 2020, the original dataset consists of 1.1 million records of mobile app on Google Play, with 23 attributes (see Appendix A).

## 3.2 Data Pre-processing

### 3.2.1 Cleaning and Filter

Before analysing the images and building causal inference model, some data pre-processing is essential. To narrow down the huge total sample as well as to be more precise, I only kept game apps. Games account for the largest proportion of mobile app revenue. As to meet their entertainment needs rather than information needs, people are generally low-involved when browsing game apps on app store, and thus they are inclined to take peripheral route according to the Elaboration Likelihood Model (Petty & Cacioppo, 1986) [24]. In this scenario, visual attractions, especially the icon will play a more important role in the information processing of app store users.

Furthermore, a threshold of last update date was introduced to lower the risk of icon change, cause the icons were only downloaded at present (May 2021). Only the apps that were ever updated after June 30th ,2020 were kept. Also, I restrained this study only on free apps, as price is a significant factor on download times. Therefore, Free and

Price columns were dropped, along with Developer Website, Developer Email and Currency, which are irrelevant information to this research.

The last issue to consider before moving forward is the language of the app. Since text detection on app icons is crucial to this study, having text from many languages will not only extend running time but also pose a threat to the accuracy of detection model. I assume the text on icon image, if exists, is of the same language as in its app name, so I only selected apps with English name and erased those with letters or characters outside the English alphabet. After all these filters, the original dataset was narrowed down to 56422 rows.

### 3.2.2 Missing values

To handle missing values, I first plotted the correlation heatmap of missing values, as shown in Figure 5.

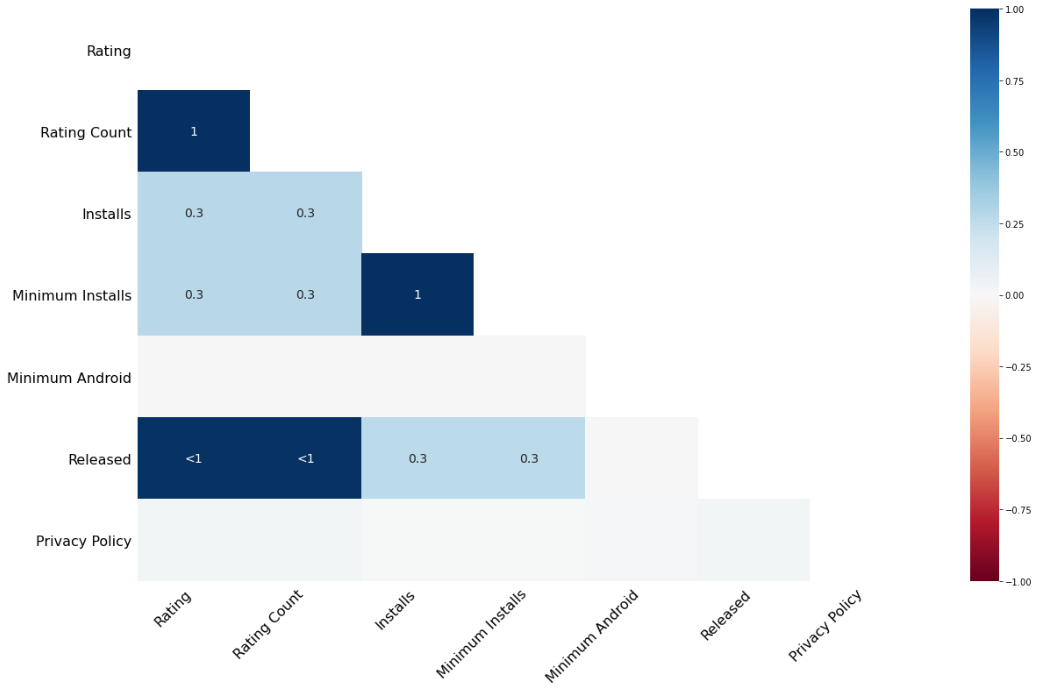


Figure 5 Correlation of missing values

Seven variables are not complete. It turns out that Minimum Installs and Installs are perfectly correlated (132 missing values). Rating Count, Rating and Released are almost perfectly correlated (1668, 1668, 1694 missing values respectively). Among these variables, the most incomplete has a missing rate of 3.0%, which implies that these samples could be simply discarded. After this manipulation, Minimum Android remains 176 missing, whereas Privacy Policy has 3567 nan values. I dropped the records missing in Minimum Android and transform Privacy Policy to a Boolean variable, which will be introduced in the following part.

## 3.3 Data Transformation

Transformation implies converting non-numerical data into numerical ones. This is twofold in my study. Some are simply categorical variables which should be coded into dummy ones, while others are in various data type such as text and date. Category, Minimum Android, Content Rating, Ad Supported, In App Purchases, Editors Choice are categorical or binary and I converted and replaced them with multiple dummy columns. After this step, there are five columns remaining to be treated.

For Size column, I converted all the storage unit to megabyte, and truncated the last character to keep only the number; Released date and Last update date were replaced by computing their time difference with the date when the data were scraped (December 4th, 2020); Privacy Policy column stores the URL of privacy policy of the app developer. As mentioned above, there are 3567 apps that do not have this information. Obviously, the web address is irrelevant to this study, a dummy variable signifying whether the app has published privacy policy was created to replace the URL.

App Name represents important information, which constitutes along with app icon a very important first impression on customers. However, extracting numerical features from App Name is complicated. An explicit one is the length of name in terms of characters. Zelinsky and Murphy (2000) claimed a tight coupling between visual and linguistic processing, and the faster of these two processings will wait for the slower process to be completed. More fixations and longer gaze durations are given to an object that has more syllables in its name [25]. Apparently, the display of apps applies to this coupling of visual and verbal processing, thus I assume the length of App Name is an important feature for matching of apps.

Another possible approach to capture the information conveyed by the app name is using word embedding[1]. Sentence Transformers (Reimers and Iryna, 2019) [26] is a framework to compute dense vector representations for sentences, paragraphs, and images. Although it is a feasible and efficient tool, the output vector space is more than 500 dimensions that are too large compared to the existing dataset. Due to limited computation power, this transformation is discarded and remained for future inspection.

## 3.4 Outlier Removal

The descriptive statistics reveal that some attributes have extremely skewed or imbalanced distributions, especially for Maximum Install and Released. The former is the outcome variable which deserves more caution and attention.

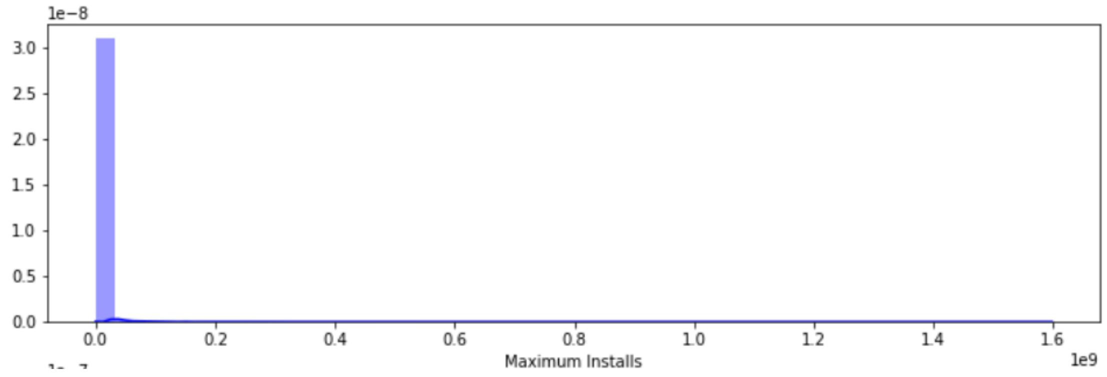


Figure 6 Distribution of Maximum Installs

A priori, Maximum Installs is Poisson-distributed, matching will be performed over the entire dataset on all the attributes, it is not reasonable to remove some data points using univariate approach like variance threshold.

[1] Representation of words for text analysis, especially in the form of a real-valued vector that encodes the meaning of the word.

Local Outlier Factor (LOF) is an unsupervised outlier detection algorithm particularly suitable for moderately high dimensional datasets. LOF algorithm computes a score (called local outlier factor) reflecting the degree of abnormality of the observations. It measures the local density deviation of a given data point with respect to its neighbours. The idea is to detect the samples that have a substantially lower density than their neighbours [27]. This serves as a pre-filter to remove isolate samples before matching.

Seven continuous attributes (Rating, Rating Count, Maximum Installs, Size, Released Days, Last Version Days, Name length) were input into LOF model, 663 apps out of 50670 were filtered in this stage.

## 3.5 Downloading Images

Not surprisingly, the dataset does not contain icon images, which I still need to resort to web scraping. In order to save time and memory space, the crawling was actually performed after data pre-processing and transformation, thus only relevant apps will be crawled.

I utilised Requests Python package and the App Id in the dataset to form URL requests of the apps' web pages on Google Play. Then lxml Python package is used to parse and localize the icon image on the html pages. Finally, images were scraped with urlretrieve method in urllib package.

I tried to scrape the images for all 50307 samples that are remaining in the dataset. However, 43732 icon images were successfully downloaded, whereas the other apps' webpages could not be found. In light of the dataset was updated in December 2020, it is absolutely normal that some apps have been taken off from the shelf during these past 6 months.

## 3.6 Exploratory Data Analysis

Finally, I obtained a complete dataset with 43732 samples and 58 attributes. Although nan values are excluded and isolate data are removed, it is still necessary to inspect the dataset by columns.

With the help of pandas profiling, I implemented an exploratory data analysis. Among the 58 attributes, Minimum Android dummy variables take up 24, which makes a very sparse matrix. Because a dummy variable with extremely low variance (extremely unbalanced binary variable) will cause a heavy computation cost in the logistic regression model for propensity score, I merged some of the levels together to get a denser matrix with 14 columns. Figure 7 displays the value counts of the 24 levels in original Minimum Android as well as my merge operation.

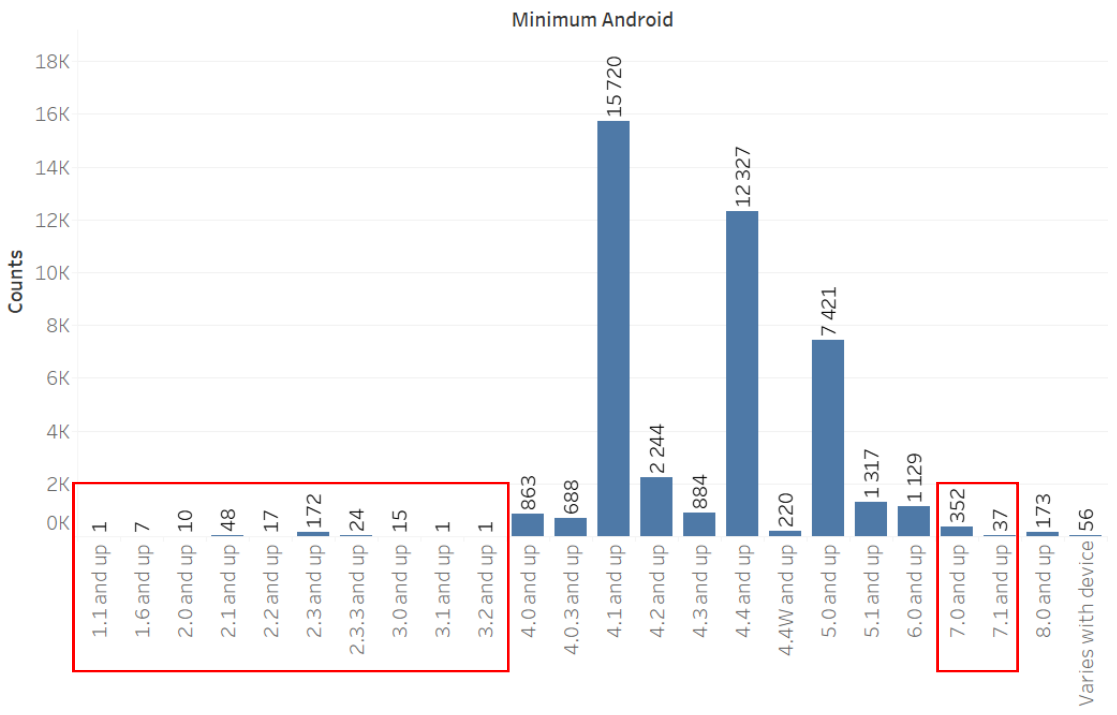


Figure 7 Value Distribution of Minimum Android

A high correlation between variables is detrimental to the analysis of regression models. Figure 8 is a person correlation plot of all the variables after merge on Minimum Android dummies. A perfect correlation exists between Ad Supported and In App

Purchases. The latter is deleted to eliminate multicollinearity. On the other hand, dummy variables are supposed to be correlated, especially when the level of the original categorical variable is small. Despite some significant negative correlations among dummies of Content Rating, it is acceptable to retain them.

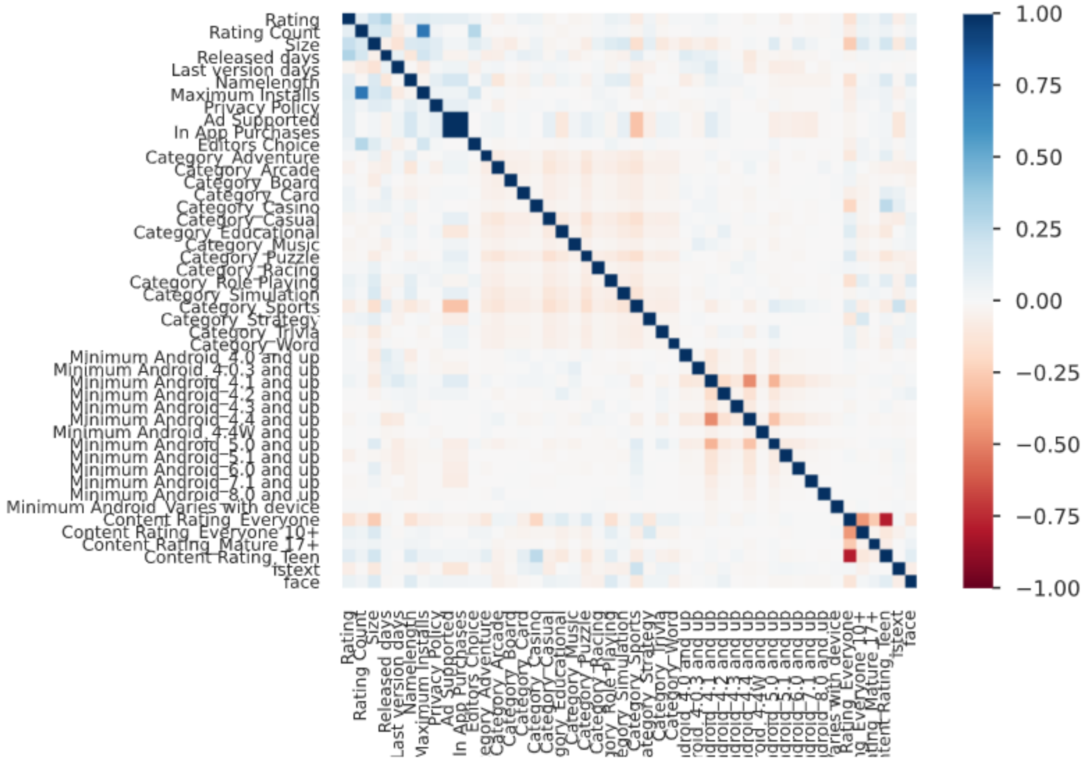


Figure 8 Correlation Plot of Variables

# 4. Image Feature Extraction

## 4.1 Text Detection and Recognition

Following Hypothesis 1 in section 2.2, this section discusses the extraction of text elements on the app icon image and the consistency of these text on images with the app name. In this study, this task is divided into three steps: text detection, text recognition and computation of similarity between strings.

### 4.1.1 Text Detection

Text detection has witnessed rapid development in recent years. However, many approaches usually fall short when dealing with challenging scenarios, even when equipped with deep neural network models [28]. The various size of text and arbitrary orientation makes the prediction of bounding boxes even more difficult.

To better tackle the diversity and the complexity of app icon images, I utilised the state-of-art EAST text detector that consists of two parts: Fully Convolutional Network (FCN) and an NMS merging stage. The detector is named as EAST since it is an Efficient and Accuracy Scene Text detection pipeline.

Figure 9 generally depicts the structure of FCN in EAST model, which can be decomposed into three parts: feature extractor stem, feature-merging branch and output layer. In the first stage, four different sizes of feature maps of the image are generated. And in the merging stage, feature maps are concatenated gradually, while the largest feature map is the first to be merged. This allows a better capturing of features for both local and global region while keeping a small computation cost. By incorporating novel, carefully designed loss functions, the detector outputs two geometry shapes, rotated box (RBOX) and quadrangle (QUAD) including words and lines of text at arbitrary orientations and quadrilateral shapes.

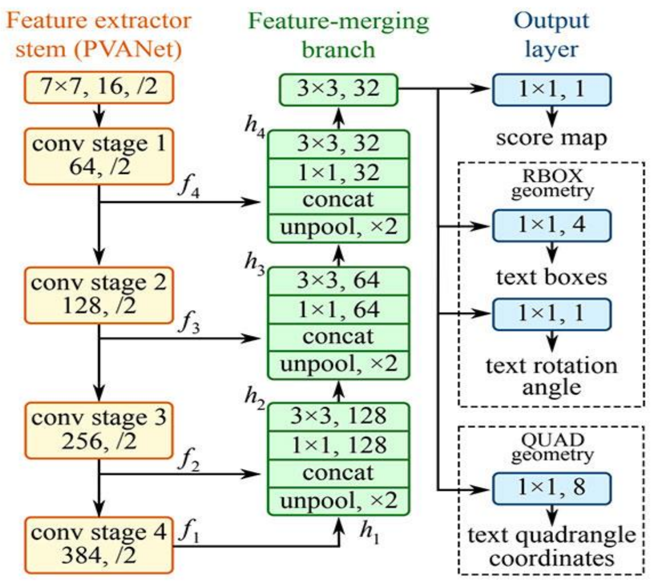


Figure 9 The structure of Fully Convolutional Network in EAST model

Thresholding is then applied to each predicted geometry, where only the ones whose scores are over the predefined threshold are considered valid and saved for later non-maximum-suppression, a technique in object detection to filter the proposals of candidate region and keep only the most appropriate entity.

In this study, frozen-east-text-detection.pb [29] is directly loaded as pre-trained model to detect text on the app icon dataset. After this process, the best geometry candidates were passed to the recognition step.

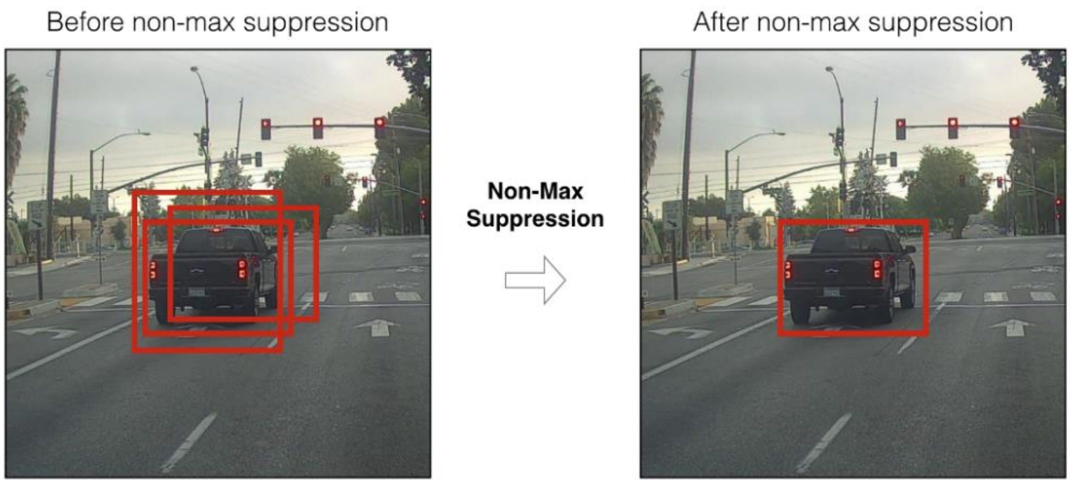


Figure 10 Non-max suppression

### 4.1.2 Text Recognition

Following the detection, text recognition algorithms read from geometries and output machine-encoded text on the images. The process is usually named Optical character recognition (OCR).

Tesseract is an open-source OCR engine that was developed at HP between 1984 and 1994 [30]. Further development in tesseract has been sponsored by Google since 2006. Figure 11 illustrates the general architecture of Tesseract. Tesseract assumes that its input is a binary image with optional polygonal text regions defined. At the first stage, adaptive thresholding transforms the input images (either colour or grey) into binary images. Then a connected component analysis is performed to store the outlines of the components. In the next step, outlines are gathered together, purely by nesting, into Blobs. Text lines are analysed for fixed pitch and proportional text and are broken into words accordingly to the character spacing.

Recognition consists of two passes. In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. During the second pass, the words which were not recognized well in the first pass are recognized again through run over the page [31].

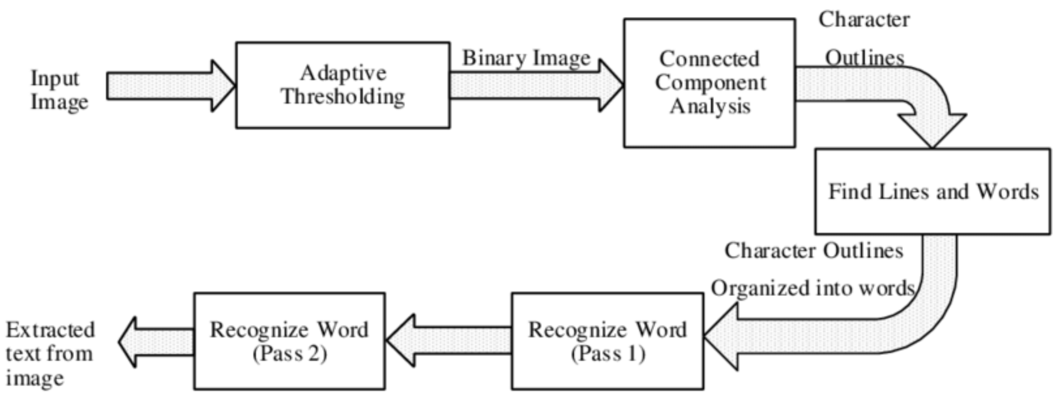


Figure 11 Architecture of Tesseract

As mentioned above, image pre-processing (including adaptive thresholding) is essential to obtaining high-quality recognition results, as it significantly removes noise. Using Opencv, I sequentially applied grayscale, gaussinblur, binary thresholding, morphological transformations. These operations output a smooth binary image on black and white that are ready for extraction.

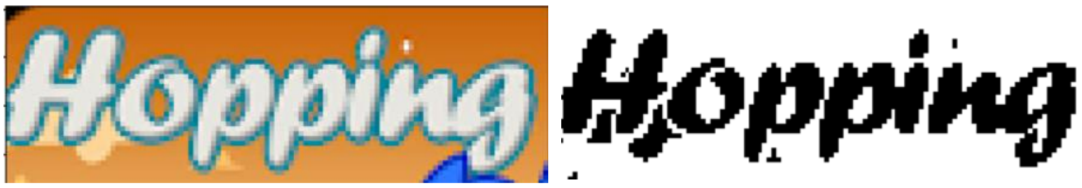


Figure 12 Image Pre-processing on text geometry

There are three core parameters in Tesseract model: the language of the text that you want to detect (language); the neural network setting of the model (engine mode) and how does the model segment an image into lines and words (page segmentation mode).

Undoubtedly, English words are the most common on icons. Here I set language to English, and samples that contain non-English characters in the app name were already discarded in data pre-processing. The new OCR engine based on LSTM neural networks is selected. Finally, because the inputs for tesseract are already rotated boxes and quadrangles around text elements, I set 'assuming a single uniform block of text' as page segmentation method.

### 4.1.3 Text – Icon Consistency

The last step to obtain an indicator of text-icon consistency is to evaluate the similarity of recognised text and app name. This entails the string similarity algorithms, whose application scenarios range from spelling correction to record linkage in database. The best-known character-based string similarity metric is Levenshtein distance, defined as the minimum number of insertions, deletions or substitutions necessary to transform one string into another [32]. The formal definition of the Levenshtein distance between two strings a and b can be seen as follows:

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & if \min(i,j) = 0 \\ min\begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & otherwise \end{cases} \quad (1)$$

Here $lev_{a,b}(i,j)$ is the Levenshtein distance between the first $i$ characters of $a$ and the first $j$ characters of $b$. And $1_{(a_i \neq b_j)}$ denotes 0 when the $i^{th}$ character of $a$ equals the $j^{th}$ character of $b$, otherwise denotes 1.

In practice, Levenshtein similarity ratio is calculated to realize standardized comparison. This is given by the formula:

$$lev\ ratio(a,b) = \frac{|a|+|b|-lev_{a,b}}{|a|+|b|} \quad (2)$$

where $|a|$ and $|b|$ are the lengths of sequence $a$ and sequence $b$, respectively.

Considering that some images are too fancy and complex for tesseract to extract the correct text, it is better to be more fault-tolerant for similarity comparison. In most cases, the text on image only covers the keyword in the app name. Therefore, I calculated the text-icon consistency score as maximum partial string similarity between the recognised text and the app name, which is defined as follow:

$$\max partial\ lev\ ratio(text_i, name) \quad (3)$$

Partial Levenshtein ratio follows the same formula as (1) but it truncates the longer string to the same length as the shorter one. Here the $text_i$ is the $i^{th}$ recognised text by tesseract, and the $name$ is the string of app name. To make the score more robust, single letters were eliminated from the set of the recognised text. Table 1 lists three examples explaining the pipeline of getting the text-icon consistency score.

By going through hundreds of random samples, I set a threshold at 66 for the consistent score to qualify an app having consistent text with its name on its icon, which means a three-letter recognition result from Tesseract OCR must have at least two same letters at the same positions from a three-letter long slice of the app name to be recognised as consistent (e.g., 'OPO' will be qualified as a consistent text if the real app name is 'ORO').

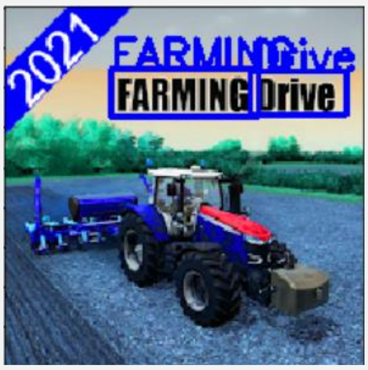
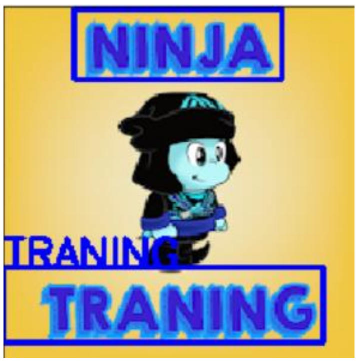
| App Name | EAST Detection | Tesseract OCR | Consistency Score |
|---|---|---|---|
| **Modern Tractor Farmer Simulator 2020:Tractor Games** |  | ['Drive', 'FARMING'] | 57 × |
| **George Ninja Training** |  | [NINJA, TRANING] | 100 √ |
| **Knife Toss** | | [] | 0 × |

Table 1 Examples of text-icon consistency pipeline

In this way, I obtained a bool variable for consistent text. 6254 apps in the final dataset have corresponding text to their name on their icons, while 37478 apps do not.

## 4.2 Face and Spokes-Character Detection

In this section, I introduce the methods used for spokes-character and human face detection as to further examine Hypothesis 2.

As discussed in section 2.3, the essential feature of spokes-character is fictional. Although it must contain certain quasi-human characteristics, some parts must be exaggerated or deformed. Before looking into the detection method for animated objects, I went through some human face detection models.

In computer vision, human face detection has seen rapid progress as the basis of many application cases, such as face id key on digital devices and selfless driving. Perhaps the most used and classic approach is Haar Cascades, proposed by Paul Viola and Michael Jones in 2001 [33]. In this method, effective features are learned using the AdaBoost algorithm, although importantly, multiple models are organized into a hierarchy or "cascade", where for each window of the image, different features are grouped into stages to pass different classifiers. If a window fails the first stage, it will be discarded. We do not consider remaining features on it. Only the window which passes all stages is classified as a face region [34].

Another state-of-art model is The MTCNN (MultiTask Cascaded Convolutional Neural Network) [35] architecture offers a deep cascaded multi-task framework with three sequential deep CNNs: the Proposal Net, the Residual Net and the Output Net [36]. First, the image is rescaled to a range of different sizes (called an image pyramid), then the first model (Proposal Net) proposes candidate facial regions, the second model (Refine Net) filters the bounding boxes, and the third model (Output Net) proposes facial landmarks.

But as discussed before, a model adapted to animated features is needed. Thankfully, there are several open-source models that trained on animation images, especially on

Japanese manga. Anime-Face-Detector [37] is a Faster R-CNN based model which trained on samples collecting from Pixiv's daily ranking[2].

Faster R-CNN [38] is an object detection architecture presented by Ren et al. (2015). It consists of two stages. The first stage called a Region Proposal Network (RPN), proposes candidate object bounding boxes. The second stage, which is in essence Fast R-CNN[3], extracts features using RoIPool from each candidate box and performs classification and bounding-box regression [39].

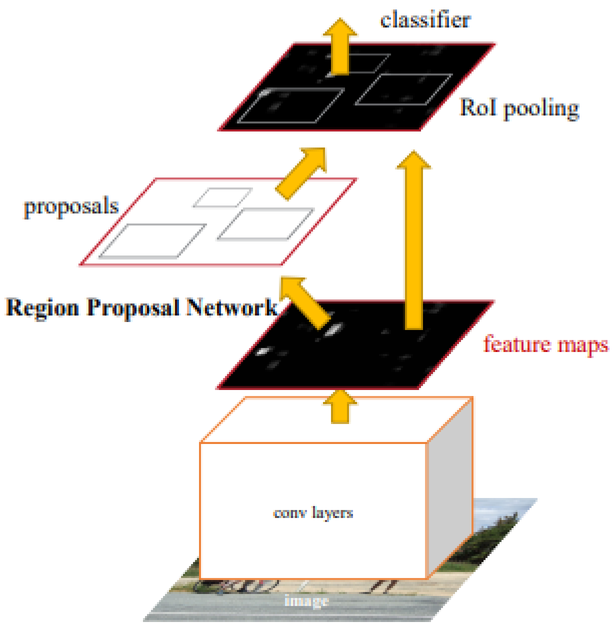


Figure 13 Faster R-CNN architecture

As an extension to both R-CNN and Fast R-CNN, the RPN generates efficient and accurate region proposals. It achieves a better mean Average Precision with less time.

As to have an empirical comparison of the three models above (Haar Cascades, MTCNN, Anime-Face-Detector), I selected 100 sample images from the dataset, in which 13 contain spokes-characters or portrait photos. The detection results are shown in Table 2.

[2] Pixiv is a Japanese online community mainly for manga and anime artists. https://www.pixiv.net/ranking.php?mode=daily

[3] Region-based CNN

| Detections | spokes-character or portrait photo | Others | Accuracy | F1 Score | Run Time |
|---|---|---|---|---|---|
| **Haar Cascades** | 6/13 | 6/87 | 0.87 | 0.48 | **1s** |
| **MTCNN** | 7/13 | **0/87** | 0.94 | 0.7 | 84s |
| **Anime-Face-Detector** | **13/13** | 1/87 | **0.99** | **0.96** | 429s |

Table 2 Trials on three face detection algorithms

Because the dataset is unbalanced, with a small percentage of photos having spokes-character or portrait photo, I calculated both accuracy and F1 Score[4] to evaluate detection performance. While Haar Cascade is super-fast, its performance is way lower than MTCNN and Anime-Face-Detector. Although the run-time of Anime-Face-Detector is costly, I still chose to apply it due to the accuracy concern. Here below in Figure 14 are some detection results from Anime-Face-Detector with bounding boxes.

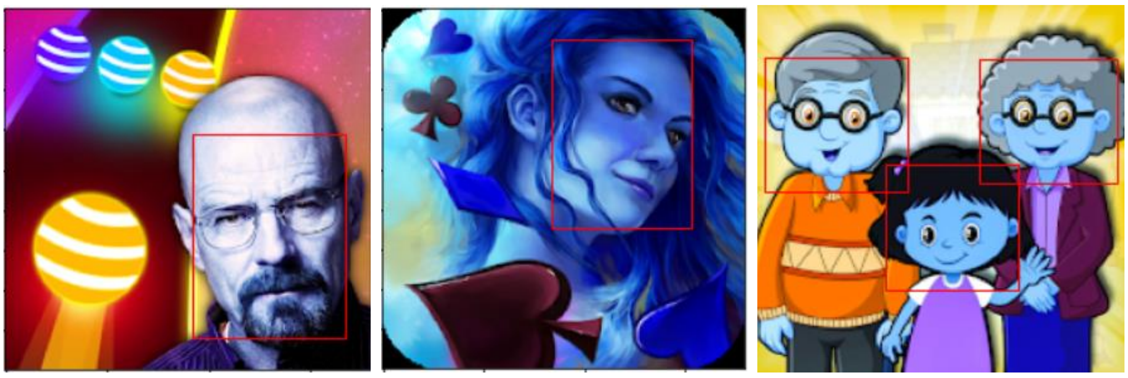


Figure 14 Detection results of Anime-Face-Detector

Given that MTCNN is a cautious method and generates conservative predictions without any false positive in the trial, I also implemented it and took the union of predictions from two models as the final bool variable of spoke-characters or human faces. In total, 7326 out of 43732 apps have facial characteristics on their icons.

[4] F1 score is the harmonic mean of the precision and recall in binary classification problem. See: https://g.co/kgs/KTs5LQ

# 5. Causality Analysis and Matching

The aim of this study is to examine the impacts of text and real or fictional faces on app downloads, which can be measured by the average treatment effect on the treated (ATT). The average treatment effect in the treated (ATT) is the average effect of the treatment for units like those who actually were treated [40]. This could be estimated through propensity score matching (PSM). In this paper, the treatments (D) are text and faces, while the outcome variable (Y) is the download times. The following formula gives the estimation process of ATT:

$$ATT = E\left(Y_{i,1} \middle| D_i = 1\right) - E\left(Y_{i,0} \middle| D_i = 1\right)$$

$$= E\left(Y_{i,1} \middle| D_i = 1\right) - E\left(Y_{i,0} \middle| D_i = 0\right) + E\left(Y_{i,0} \middle| D_i = 0\right) - E\left(Y_{i,0} \middle| D_i = 1\right) \quad (4)$$

where $E$ denotes the expectation operator, $Y_{i,1}$ is the download times of app $i$ that contains text (or faces) on its icon; $Y_{i,0}$ is the download times of the app $i$ that does not contain text (or faces) on its icon. And $D_i$ is a binary indicator that equals 1 if the app icon does contain text (or faces) and 0 otherwise.

In reality, we cannot observe $Y_{i,1}|D_i = 1$ and $Y_{i,0}|D_i = 1$ simultaneously for a same app, but instead $Y_{i,1}|D_i = 1$ and $Y_{i,0}|D_i = 0$. However, the simple difference between these two observed outcomes involves selection bias that is represented by $E\left(Y_{i,0} \middle| D_i = 0\right) - E\left(Y_{i,0} \middle| D_i = 1\right)$. That is to say, the fact of containing text (or faces) on icon is not random and the impact of treatment could not be justified.

Therefore, matching is used in this step to eliminate selection bias. Proposed by Rosenbaum and Rubin (1983) [41], Propensity Score Matching is a technique in causal inference to match treated and untreated observations on the estimated probability of being treated, and thus estimate the effect of treatment. To be specific, the method assumes that the outcome is independent of treatment indicator conditional on a set of observable characteristics (W) [42]. This can be expressed as $Y_{i,0} \perp D_i|p(W)$, with $p(W)$ denotes the probability (propensity score) of $D = 1$ with certain covariates $W$.

By matching on these covariates, the selection bias term between the new treated group and new control group equals zero.

It is worth noting that PSM assumes no unobserved variable affect the treatment assignment. Besides, covariates must be measured prior to treatment (or otherwise not be affected by the treatment). Recalling the variables in the dataset, all the attributes of an app can be seen as independent to the icon design. Also, considering the variety of attributes covering almost all the available information of an app, we can suppose our setting meets the assumptions of PSM.

## 5.1 Estimation of Propensity Score

The most common method to calculate propensity score is logistic regression. Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable [43]. Assumes that Y is a binary dependent variable with two possible values 0 and 1 and X are a set of independent variables, the logistic regression equation is written as follow:

$$P(Y = 1|X) = \frac{1}{1+e^{-(\beta^T X+b)}} \quad (5)$$

Hence, fitting the treatment assignment with covariates using logistic regression gives the probabilities, namely propensity score (with a range of 0 to 1), of receiving the treatment.

| Covariates | Model 1 | | Model 2 | | Model 3 | |
|---|---|---|---|---|---|---|
| | coefficient | p-value | coefficient | p-value | coefficient | p-value |
| Rating | -0.0754 | 0.0000 | -0.0697 | 0.0000 | -0.0542 | 0.0000 |
| Rating_Count | 0.0000 | 0.3852 | 0.0000 | 0.0423 | 0.0000 | 0.0264 |
| Size | -0.0116 | 0.0000 | -0.0093 | 0.0000 | -0.0069 | 0.0000 |
| Released_days | 0.0000 | 0.0680 | -0.0001 | 0.0000 | -0.0001 | 0.0005 |
| Last_version_days | -0.0027 | 0.0000 | -0.0020 | 0.0000 | -0.0003 | 0.2962 |
| Namelength | 0.0015 | 0.1488 | 0.0003 | 0.7979 | 0.0051 | 0.0000 |
| Privacy_Policy | -0.2921 | 0.0000 | -0.4249 | 0.0000 | -0.0139 | 0.8093 |
| Ad_Supported | -0.8888 | 0.0000 | -0.5892 | 0.0000 | -0.4004 | 0.0000 |
| Editors_Choice | -0.1777 | 0.4581 | -0.1485 | 0.5425 | -0.1311 | 0.5909 |
| Category_Adventure | | | -0.5385 | 0.0000 | 0.2532 | 0.0119 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Category_Arcade | | | -0.8611 | 0.0000 | 0.0653 | 0.4959 |
| Category_Board | | | -0.4157 | 0.0000 | 0.4830 | 0.0000 |
| Category_Card | | | 0.6418 | 0.0000 | 1.4798 | 0.0000 |
| Category_Casino | | | 1.4431 | 0.0000 | 2.1390 | 0.0000 |
| Category_Casual | | | -0.9088 | 0.0000 | -0.0584 | 0.5231 |
| Category_Educational | | | -0.2360 | 0.0008 | 0.6154 | 0.0000 |
| Category_Music | | | 0.1581 | 0.1247 | 0.9770 | 0.0000 |
| Category_Puzzle | | | -0.7389 | 0.0000 | 0.1399 | 0.1131 |
| Category_Racing | | | -0.6142 | 0.0000 | 0.1826 | 0.1603 |
| Category_Role_Playing | | | -0.7767 | 0.0000 | -0.0089 | 0.9385 |
| Category_Simulation | | | -0.5638 | 0.0000 | 0.2072 | 0.0251 |
| Category_Sports | | | 0.6536 | 0.0000 | 1.5963 | 0.0000 |
| Category_Strategy | | | -0.5206 | 0.0000 | 0.3937 | 0.0015 |
| Category_Trivia | | | 0.3699 | 0.0000 | 1.2387 | 0.0000 |
| Category_Word | | | 0.4222 | 0.0000 | 1.2725 | 0.0000 |
| Min_Android_4.0 | | | | | 0.2389 | 0.2396 |
| Min_Android_4.0.3 | | | | | 0.3685 | 0.0720 |
| Min_Android_4.1 | | | | | 0.2565 | 0.1498 |
| Min_Android_4.2 | | | | | 0.5774 | 0.0019 |
| Min_Android_4.3 | | | | | -0.0282 | 0.8928 |
| Min_Android_4.4 | | | | | 0.3002 | 0.0927 |
| Min_Android_4.4W | | | | | -0.0790 | 0.7888 |
| Min_Android_5.0 | | | | | 0.2518 | 0.1615 |
| Min_Android_5.1 | | | | | 0.3108 | 0.1049 |
| Min_Android_6.0 | | | | | 0.4679 | 0.0162 |
| Min_Android_7.1 | | | | | 0.4510 | 0.0443 |
| Min_Android_8.0 | | | | | 0.2093 | 0.4625 |
| Min_Android_Varies | | | | | 1.1171 | 0.0029 |
| Content_Rating_Everyone | | | | | -2.0710 | 0.0000 |
| Content_Rating_10+ | | | | | -2.1785 | 0.0000 |
| Content_Rating_17+ | | | | | -1.6832 | 0.0000 |
| Content_Rating_Teen | | | | | -2.0022 | 0.0000 |
| Number of observations | 42732 | | 43732 | | 43732 | |
| Log Likelihood | -17488. | | -16490. | | -16279. | |
| AIC | 34993.9367 | | 33029.8884 | | **32641.8169** | |
| Pseudo R square | 0.026 | | 0.081 | | **0.093** | |

Table 3 Summary of Logit Model for treatment of Text

| Covariates | Model 1 | | Model 2 | | Model 3 | |
|---|---|---|---|---|---|---|
| | coefficient | p-value | coefficient | p-value | coefficient | p-value |
| Rating | -0.0227 | 0.0024 | -0.0057 | 0.0000 | 0.0386 | 0.0000 |
| Rating_Count | 0.0000 | 0.0003 | 0.0000 | 0.1675 | 0.0000 | 0.0924 |
| Size | 0.0054 | 0.0000 | 0.0040 | 0.0000 | 0.0082 | 0.0000 |
| Released_days | -0.0001 | 0.0000 | -0.0000 | 0.0000 | -0.0000 | 0.2887 |
| Last_version_days | -0.0031 | 0.0000 | -0.0018 | 0.0000 | 0.0007 | 0.0291 |
| Namelength | 0.0082 | 0.0000 | 0.0100 | 0.0000 | 0.0172 | 0.0000 |
| Privacy_Policy | -1.1938 | 0.0000 | -0.7029 | 0.0000 | 0.2112 | 0.0012 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Ad_Supported | -0.5836 | 0.0000 | -0.4487 | 0.0000 | 0.0674 | 0.1401 |
| Editors_Choice | 0.3469 | 0.0170 | 0.2066 | 0.0000 | 0.2122 | 0.1559 |
| Category_Adventure | | | -0.6720 | 0.0413 | 0.2392 | 0.0008 |
| Category_Arcade | | | -1.5341 | 0.0000 | -0.3106 | 0.0001 |
| Category_Board | | | -1.4024 | 0.0000 | -0.1998 | 0.0537 |
| Category_Card | | | -1.0193 | 0.0000 | 0.0103 | 0.9145 |
| Category_Casino | | | -0.9158 | 0.0000 | -0.3022 | 0.0104 |
| Category_Casual | | | -0.4336 | 0.0000 | 0.6830 | 0.0000 |
| Category_Educational | | | -0.5371 | 0.0000 | 0.6668 | 0.0000 |
| Category_Music | | | -0.1808 | 0.0014 | 0.9207 | 0.0000 |
| Category_Puzzle | | | -1.1619 | 0.0000 | 0.0326 | 0.6238 |
| Category_Racing | | | -2.0766 | 0.0000 | -1.0594 | 0.0000 |
| Category_Role_Playing | | | 0.2497 | 0.1675 | 1.1457 | 0.0000 |
| Category_Simulation | | | -0.7410 | 0.0000 | 0.2119 | 0.0010 |
| Category_Sports | | | -1.8165 | 0.0000 | -0.4052 | 0.0000 |
| Category_Strategy | | | -0.5098 | 0.0000 | 0.6108 | 0.0000 |
| Category_Trivia | | | -0.2238 | 0.0000 | 1.0344 | 0.0000 |
| Category_Word | | | -1.8774 | 0.0000 | -0.6898 | 0.0000 |
| Min_Android_4.0 | | | | | -0.4404 | 0.0143 |
| Min_Android_4.0.3 | | | | | -0.5478 | 0.0028 |
| Min_Android_4.1 | | | | | -0.5537 | 0.0003 |
| Min_Android_4.2 | | | | | -0.0735 | 0.6429 |
| Min_Android_4.3 | | | | | -0.7573 | 0.0000 |
| Min_Android_4.4 | | | | | -0.6165 | 0.0001 |
| Min_Android_4.4W | | | | | -0.9885 | 0.0001 |
| Min_Android_5.0 | | | | | -0.7987 | 0.0000 |
| Min_Android_5.1 | | | | | -0.7633 | 0.0000 |
| Min_Android_6.0 | | | | | -0.7312 | 0.0000 |
| Min_Android_7.1 | | | | | -1.0607 | 0.0000 |
| Min_Android_8.0 | | | | | -1.3951 | 0.0001 |
| Min_Android_Varies | | | | | -0.1464 | 0.6888 |
| Content_Rating_Everyone | | | | | -2.6231 | 0.0000 |
| Content_Rating_10+ | | | | | -2.5721 | 0.0000 |
| Content_Rating_17+ | | | | | -1.8254 | 0.0000 |
| Content_Rating_Teen | | | | | -2.1266 | 0.0000 |
| Number of observations | 42732 | | 43732 | | 43732 | |
| Log Likelihood | -20251. | | -18912. | | -18083. | |
| AIC | 40520.3869 | | 37873.7125 | | **36249.8978** | |
| Pseudo R square | 0.025 | | 0.043 | | **0.085** | |

Table 4 Summary of Logit Model for treatment of Face

The estimated results of logistic regression models are presented in Table 3 and Table 4, with text and face treatment as the dependent variable, respectively. The objective of this stage is not to examine significances of variables, but to have an insight of the fitting. I divided the covariates into three groups: basic information, categories and requirements, and added them to the model in order. In both two tables, the goodness of fit (measured by AIC and pseudo $R^2$) reached the optimal when all three groups of covariates are presented (Model 3), therefore the probability prediction of Model 3 will be used as propensity score for both text and face treatments.

## 5.2 Matching Criteria and Results

Matching on propensity scores generates a balanced Treated-vs-Control dataset. There are several different classes and methods of matching available. Generally, there are two classes of methods: distance matching and stratum matching. Distance matching involves selecting members of the control group to pair with each member of the treated group. Members of either group that are not paired are dropped from the sample. This class includes nearest neighbour, optimal matching and genetic matching. Stratum matching involves creating strata based on unique values of the covariates and assigning units with those covariate values into those strata. Exact matching, subclassification and coarsened exact matching belong to the stratum matching class [44].

Here below is a short description of several matching methods[5]:

- **Nearest neighbor**: greedy method, the closest control unit is chosen for each treated unit, one at a time without replacement. without trying to minimize a global distance measure.
- **Optimal matching**: finding the matched samples with the smallest average absolute distance across all the matched pairs.
- **Genetic matching**: using a computationally intensive genetic search algorithm to match treatment and control units.
- **Exact matching**: matching each treated unit with a control unit that has exactly the same values on each covariate.

[5] See the documentation on MatchIt for more details: https://cran.r-project.org/web/packages/MatchIt/vignettes/matching-methods.html

- **Coarsened exact matching**: first coarsening the covariates by creating bins and then performing exact matching on the new coarsened versions of the covariates.
- **Subclassification**: a form of coarsened exact matching with the propensity score as the sole covariate to be coarsened and matched on.

To implement matching, I choose MatchIt[6] package in R. It implements the suggestions of Ho, Imai, King, and Stuart (2007) [45] for improving parametric statistical models for estimating treatment effects in observational studies.

Using MatchIt, I tried out nearest neighbour matching and optimal matching and coarsened exact matching. But due to the huge sample size, optimal matching cannot be executed on my computing device. Only nearest neighbour matching and coarsened exact matching were tested on this dataset.

To improve the goodness of nearest neighbour matching, I imposed an exact match on category dummy variables and content rating variables. From a common point of view, these are the most general classifications of an app that must be equal in each matching pair. Moreover, an exact matching restriction on certain covariates will dramatically reduce searching space and speed up the matching process, especially for computationally costly methods like optimal matching. In addition, caliper is used for all distance matching approaches that only the units whose distance (normally propensity score) smaller than the caliper are allowed to match. The calipers usually take 0.2 and 0.25 of the propensity score standard deviation [46]. Here I set caliper to 0.2. Ratio is set to 1, implying a one-to-one match without replacement is performed.

I assessed the quality of match by checking the covariates' balance before and after the matching. Nearest neighbour matching gave relatively low average absolute within-pair difference in after-matching results. Meanwhile, the vast majority of the treated samples

[6] MatchIt is an **R** package that easily enables **R** users to conduct propensity score matching.

are retained. Coarsened exact matching is even more precise in terms of within-pair difference, but only 317 and 289 treatment samples are kept for two different treatments.

However, after further examination on matched pairs in nearest neighbour matching results, I found that the pairwise distances are not satisfying enough for some key features (Released days, Rating count etc.). As we can see huge differences in sheer numbers on Rating Count and Released Days for some pairs (see Table 5). The matching did not improve the overall distribution balance on these variables do neither (see Figure 15). For credibility consideration, coarsened exact matching will be used to estimate treatment effects.

| App Id | Rating Count | Released days | … | Maximum Installs | Pair No. |
|---|---|---|---|---|---|
| com.yinzcam.nfl.cardinals | 2960 | 3056 | … | 192661 | 3789 |
| com.axitama.superfiresoccer | 89 | 184 | … | 9228 | 3789 |
| nl.lisa_is.pwhockey | 9 | 2649 | | 693 | 2366 |
| com.ballz.billiard.free.fungames | 843 | 329 | | 58731 | 2366 |

Table 5 Extract pairs of nearest neighbour matching result

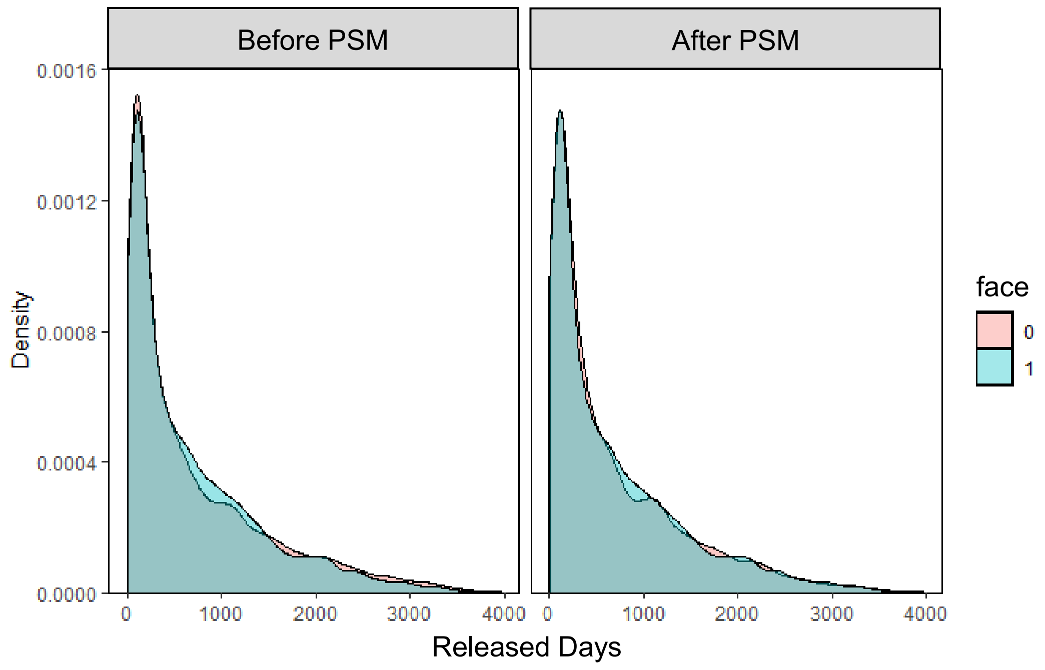


Figure 15 Distribution of Released Days for nearest neighbour matching on face treatment.

Table 6 and Table 7 show the coarsened exact matching results, with text and face as the treatment variable, respectively. The summaries of nearest neighbour matching can be found in Appendix B.

| | **Before Matching – All Data** | | | **After Matching – Matched Data** | | | |
|---|---|---|---|---|---|---|---|
| | Means Treated | Means Control | Std. Mean Diff. | Means Treated | Means Control | Std. Mean Diff. | Std. Pair Dist. |
| **Rating** | 2.8697 | 3.1956 | -0.1674 | 1.0950 | 1.0914 | 0.0018 | 0.0084 |
| **Rating_Count** | 16654.71 | 22250.19 | -0.0364 | 1440.3123 | 654.0336 | 0.0051 | 0.0073 |
| **Size** | 32.6925 | 41.4498 | -0.2997 | 23.9943 | 23.9721 | 0.0008 | 0.0621 |
| **Released_days** | 718.9146 | 695.7503 | 0.0293 | 465.1388 | 460.5827 | 0.0058 | 0.0439 |
| **Last_version_days** | 64.4402 | 65.9707 | -0.0353 | 57.4353 | 57.4968 | -0.0014 | 0.0357 |
| **Namelength** | 26.7467 | 27.3983 | -0.0503 | 23.1136 | 23.0974 | 0.0013 | 0.0609 |
| **Privacy_Policy** | 0.9287 | 0.9372 | -0.0331 | 0.9779 | 0.9779 | 0.0000 | 0.0000 |
| **Ad_Supported** | 0.7681 | 0.8797 | -0.2644 | 0.7539 | 0.7539 | -0.0000 | 0.0000 |
| **Editors_Choice** | 0.0034 | 0.0064 | -0.0531 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| **Category_Adventure** | 0.0361 | 0.0576 | -0.1148 | 0.0221 | 0.0221 | 0.0000 | 0.0000 |
| **Category_Arcade** | 0.0476 | 0.0886 | -0.1924 | 0.0473 | 0.0473 | -0.0000 | 0.0000 |
| **Category_Board** | 0.0283 | 0.0328 | -0.0269 | 0.0032 | 0.0032 | -0.0000 | 0.0000 |
| **Category_Card** | 0.0534 | 0.0256 | 0.1238 | 0.0158 | 0.0158 | -0.0000 | 0.0000 |
| **Category_Casino** | 0.0403 | 0.0104 | 0.1520 | 0.0126 | 0.0126 | -0.0000 | 0.0000 |
| **Category_Casual** | 0.0603 | 0.1278 | -0.2835 | 0.0883 | 0.0883 | 0.0000 | 0.0000 |
| **Category_Educational** | 0.0585 | 0.0585 | 0.0001 | 0.0568 | 0.0568 | -0.0000 | 0.0000 |
| **Category_Music** | 0.0225 | 0.0162 | 0.0424 | 0.0158 | 0.0158 | -0.0000 | 0.0000 |
| **Category_Puzzle** | 0.0815 | 0.1411 | -0.2175 | 0.1640 | 0.1640 | 0.0000 | 0.0000 |
| **Category_Racing** | 0.0154 | 0.0290 | -0.1108 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| **Category_Role_Playing** | 0.0217 | 0.0470 | -0.1732 | 0.0095 | 0.0095 | 0.0000 | 0.0000 |
| **Category_Simulation** | 0.0544 | 0.0989 | -0.1962 | 0.0221 | 0.0221 | 0.0000 | 0.0000 |
| **Category_Sports** | 0.3310 | 0.1146 | 0.4598 | 0.4763 | 0.4763 | -0.0000 | 0.0000 |
| **Category_Strategy** | 0.0179 | 0.0259 | -0.0599 | 0.0095 | 0.0095 | 0.0000 | 0.0000 |
| **Category_Trivia** | 0.0536 | 0.0283 | 0.1123 | 0.0315 | 0.0315 | -0.0000 | 0.0000 |
| **Category_Word** | 0.0438 | 0.0247 | 0.0933 | 0.0221 | 0.0221 | 0.0000 | 0.0000 |
| **Min_Android_4.0** | 0.0193 | 0.0198 | -0.0033 | 0.0095 | 0.0095 | 0.0000 | 0.0000 |
| **Min_Android_4.0.3** | 0.0192 | 0.0152 | 0.0294 | 0.0158 | 0.0158 | -0.0000 | 0.0000 |
| **Min_Android_4.1** | 0.3331 | 0.3640 | -0.0656 | 0.3438 | 0.3438 | 0.0000 | 0.0000 |
| **Min_Android_4.2** | 0.0641 | 0.0492 | 0.0610 | 0.0631 | 0.0631 | -0.0000 | 0.0000 |
| **Min_Android_4.3** | 0.0155 | 0.0210 | -0.0444 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| **Min_Android_4.4** | 0.2592 | 0.2856 | -0.0602 | 0.2555 | 0.2555 | 0.0000 | 0.0000 |
| **Min_Android_4.4W** | 0.0034 | 0.0053 | -0.0337 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| **Min_Android_5.0** | 0.1884 | 0.1667 | 0.0555 | 0.2050 | 0.2050 | -0.0000 | 0.0000 |
| **Min_Android_5.1** | 0.0401 | 0.0284 | 0.0596 | 0.0662 | 0.0662 | 0.0000 | 0.0000 |
| **Min_Android_6.0** | 0.0341 | 0.0244 | 0.0530 | 0.0347 | 0.0347 | -0.0000 | 0.0000 |
| **Min_Android_7.1** | 0.0117 | 0.0085 | 0.0299 | 0.0063 | 0.0063 | -0.0000 | 0.0000 |
| **Min_Android_8.0** | 0.0042 | 0.0039 | 0.0032 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Min_Android_Varies** | 0.0021 | 0.0011 | 0.0204 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| **Content_Rating_Everyone** | 0.7841 | 0.7597 | 0.0594 | 0.9716 | 0.9716 | -0.0000 | 0.0000 |
| **Content_Rating_10+** | 0.0317 | 0.0589 | -0.1555 | 0.0063 | 0.0063 | -0.0000 | 0.0000 |
| **Content_Rating_17+** | 0.0221 | 0.0199 | 0.0145 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| **Content_Rating_Teen** | 0.1610 | 0.1613 | -0.0009 | 0.0221 | 0.0221 | 0.0000 | 0.0000 |
| **Treatment sample** | | 6254 | | | 317 | | |
| **Control sample** | | 37478 | | | 356 | | |

Table 6 Balance Summary of using Coarsened Exact Matching for text treatment.

| | **Before Matching – All Data** | | | | **After Matching – Matched Data** | | |
|---|---|---|---|---|---|---|---|
| | Means Treated | Means Control | Std. Mean Diff. | Means Treated | Means Control | Std. Mean Diff. | Std. Pair Dist. |
| **Rating** | 2.8697 | 3.1956 | -0.1674 | 1.6471 | 1.6470 | 0.0000 | 0.0180 |
| **Rating_Count** | 16654.71 | 22250.19 | -0.0364 | 1863.5571 | 1520.8761 | 0.0017 | 0.0082 |
| **Size** | 32.6925 | 41.4498 | -0.2997 | 28.9003 | 28.6038 | 0.0092 | 0.0608 |
| **Released_days** | 718.9146 | 695.7503 | 0.0293 | 493.9412 | 491.4206 | 0.0035 | 0.0460 |
| **Last_version_days** | 64.4402 | 65.9707 | -0.0353 | 61.5882 | 61.4213 | 0.0037 | 0.0377 |
| **Namelength** | 26.7467 | 27.3983 | -0.0503 | 32.7093 | 32.5569 | 0.0118 | 0.0673 |
| **Privacy_Policy** | 0.9287 | 0.9372 | -0.0331 | 0.9758 | 0.9758 | -0.0000 | 0.0000 |
| **Ad_Supported** | 0.7681 | 0.8797 | -0.2644 | 0.9446 | 0.9446 | -0.0000 | 0.0000 |
| **Editors_Choice** | 0.0034 | 0.0064 | -0.0531 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| **Category_Adventure** | 0.0361 | 0.0576 | -0.1148 | 0.0346 | 0.0346 | 0.0000 | 0.0000 |
| **Category_Arcade** | 0.0476 | 0.0886 | -0.1924 | 0.0623 | 0.0623 | -0.0000 | 0.0000 |
| **Category_Board** | 0.0283 | 0.0328 | -0.0269 | 0.0035 | 0.0035 | 0.0000 | 0.0000 |
| **Category_Card** | 0.0534 | 0.0256 | 0.1238 | 0.0208 | 0.0208 | 0.0000 | 0.0000 |
| **Category_Casino** | 0.0403 | 0.0104 | 0.1520 | 0.0242 | 0.0242 | -0.0000 | 0.0000 |
| **Category_Casual** | 0.0603 | 0.1278 | -0.2835 | 0.1696 | 0.1696 | -0.0000 | 0.0000 |
| **Category_Educational** | 0.0585 | 0.0585 | 0.0001 | 0.0692 | 0.0692 | 0.0000 | 0.0000 |
| **Category_Music** | 0.0225 | 0.0162 | 0.0424 | 0.0519 | 0.0519 | 0.0000 | 0.0000 |
| **Category_Puzzle** | 0.0815 | 0.1411 | -0.2175 | 0.1938 | 0.1938 | -0.0000 | 0.0000 |
| **Category_Racing** | 0.0154 | 0.0290 | -0.1108 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| **Category_Role_Playing** | 0.0217 | 0.0470 | -0.1732 | 0.0242 | 0.0242 | -0.0000 | 0.0000 |
| **Category_Simulation** | 0.0544 | 0.0989 | -0.1962 | 0.0311 | 0.0311 | -0.0000 | 0.0000 |
| **Category_Sports** | 0.3310 | 0.1146 | 0.4598 | 0.2180 | 0.2180 | 0.0000 | 0.0000 |
| **Category_Strategy** | 0.0179 | 0.0259 | -0.0599 | 0.0035 | 0.0035 | 0.0000 | 0.0000 |
| **Category_Trivia** | 0.0536 | 0.0283 | 0.1123 | 0.0623 | 0.0623 | -0.0000 | 0.0000 |
| **Category_Word** | 0.0438 | 0.0247 | 0.0933 | 0.0035 | 0.0035 | 0.0000 | 0.0000 |
| **Min_Android_4.0** | 0.0193 | 0.0198 | -0.0033 | 0.0069 | 0.0069 | 0.0000 | 0.0000 |
| **Min_Android_4.0.3** | 0.0192 | 0.0152 | 0.0294 | 0.0311 | 0.0311 | -0.0000 | 0.0000 |
| **Min_Android_4.1** | 0.3331 | 0.3640 | -0.0656 | 0.4464 | 0.4464 | 0.0000 | 0.0000 |
| **Min_Android_4.2** | 0.0641 | 0.0492 | 0.0610 | 0.1315 | 0.1315 | -0.0000 | 0.0000 |
| **Min_Android_4.3** | 0.0155 | 0.0210 | -0.0444 | 0.0104 | 0.0104 | -0.0000 | 0.0000 |
| **Min_Android_4.4** | 0.2592 | 0.2856 | -0.0602 | 0.2111 | 0.2111 | -0.0000 | 0.0000 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Min_Android_4.4W** | 0.0034 | 0.0053 | -0.0337 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| **Min_Android_5.0** | 0.1884 | 0.1667 | 0.0555 | 0.1280 | 0.1280 | 0.0000 | 0.0000 |
| **Min_Android_5.1** | 0.0401 | 0.0284 | 0.0596 | 0.0277 | 0.0277 | 0.0000 | 0.0000 |
| **Min_Android_6.0** | 0.0341 | 0.0244 | 0.0530 | 0.0069 | 0.0069 | 0.0000 | 0.0000 |
| **Min_Android_7.1** | 0.0117 | 0.0085 | 0.0299 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| **Min_Android_8.0** | 0.0042 | 0.0039 | 0.0032 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| **Min_Android_Varies** | 0.0021 | 0.0011 | 0.0204 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| **Content_Rating_Everyone** | 0.7841 | 0.7597 | 0.0594 | 0.8927 | 0.8927 | 0.0000 | 0.0000 |
| **Content_Rating_10+** | 0.0317 | 0.0589 | -0.1555 | 0.0035 | 0.0035 | 0.0000 | 0.0000 |
| **Content_Rating_17+** | 0.0221 | 0.0199 | 0.0145 | 0.0346 | 0.0346 | 0.0000 | 0.0000 |
| **Content_Rating_Teen** | 0.1610 | 0.1613 | -0.0009 | 0.0692 | 0.0692 | 0.0000 | 0.0000 |
| **Treatment sample** | | 6254 | | | 289 | | |
| **Control sample** | | 37478 | | | 311 | | |

Table 7 Balance Summary of using Coarsened Exact Matching for text treatment.

## 5.3 Estimation of Treatment Effects

After assessing balance, it is time to estimate the effects of treatments: whether consistent text and spokes-characters or portrait photos on icons influence the popularity of an app. How treatment effects are estimated depends on the matching model and specifications. Since pairwise nearest neighbour matching without replacement is adopted, the effect could be simply estimated through linear regression to fit the outcome variable (Maximum Install) using matched dataset. However, it is important to avoid interpreting coefficients in the outcome model on predictors other than the treatment, as the coefficients do not correspond to causal effects and are likely confounded [47].

Only the coefficients of the two treatment variables are valid and of my interest. Two simple linear regressions were carried out and summarized in Table 8 and Table 9. Meanwhile, highly influential observations are removed whose Cook's distance is higher than 4/n[7].

[7] n is the number of observations in regression model

11 and 17 influential points are eliminated from regression model for text and face, respectively. Figure 16 and Figure 17 are Cook's distance plots of simple regression with text and face bool variables.

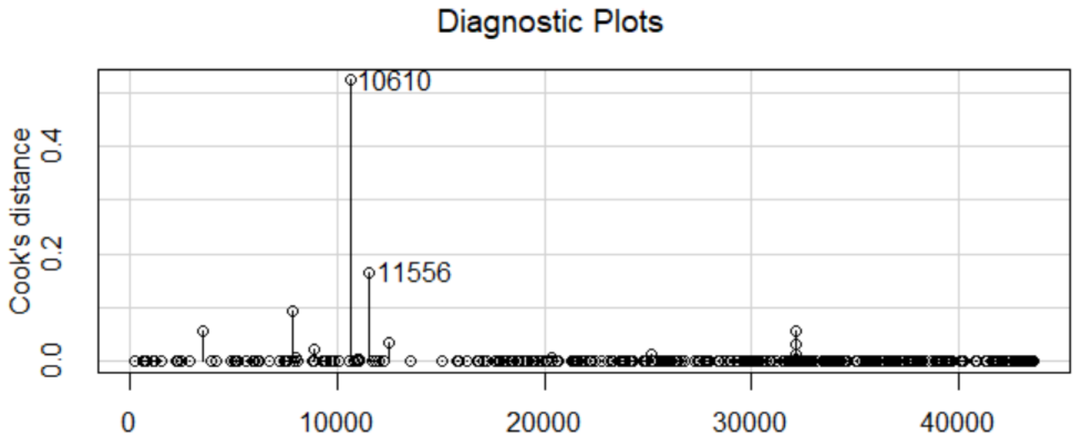


Figure 16 Cook's Distance Plot of simple regression with text bool variable

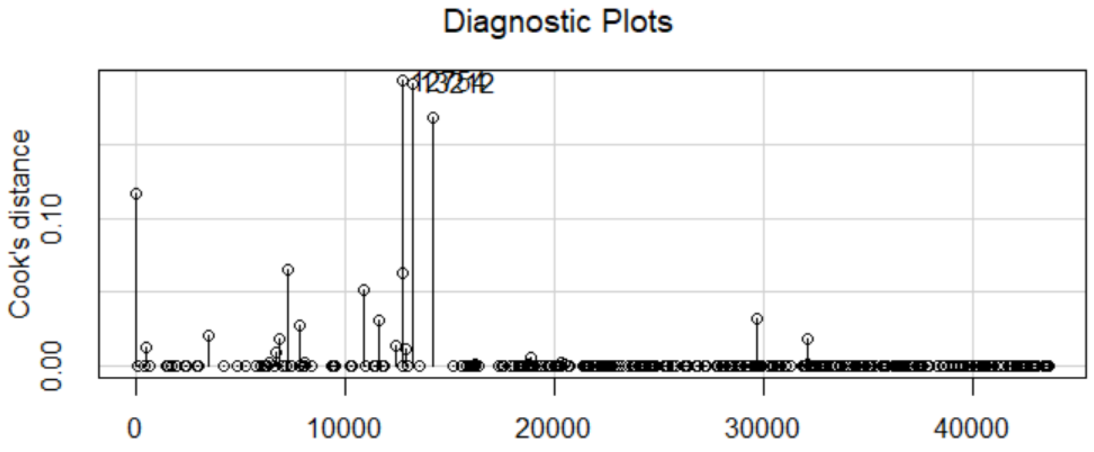


Figure 17 Cook's Distance Plot of simple regression with face bool variable

| | **coefficient** | **Std. Error** | **t value** | **p value** |
|---|---|---|---|---|
| **text** | 954.09 | 10714.94 | 0.0890 | 0.9291 |
| **Observations:** 662 | | | | $R^2$: 0.000008 |

Table 8 Effect of Consistent Text on App Download Times

| | **coefficient** | **Std. Error** | **t value** | **p value** |
|---|---|---|---|---|
| **face** | -14319 | 21737 | -0.6587 | 0.5103 |
| **Observations:** 583 | | | | $R^2$: 0.0005 |

Table 9 Effect of Facial Characteristics on App Download Times

Nevertheless, the coefficients and significance of the two treatments are not aligned with the two hypotheses presented in section 2. Based on the simple linear regression on the matched dataset, the binary text variable has a coefficient at around $954$ with a near-one p-value (0.9146); the face treatment has a negative coefficient at $-14319$, and a slightly higher significance level (0.5103).

# 6. Conclusion

## 6.1 Discussion of findings

The main purpose of the study is to access the effects of two underexplored visual elements on smartphone user’s download behaviours, text and facial characteristics. This study has made full use of advanced computer vision techniques and a comprehensive dataset collected from Google Play Store. Two features are extracted from more than 40000 icon images with EAST text detector, Tesseract, MTCNN and Anime-Face-Detector. Finally, coarsened exact matching and regression were then applied to obtain sheer causal effects. Accordingly, the study has two major findings.

First, incorporating text that consistent with app name into icon design does not have a significant effect on the download times of the app. Although many previous research has proved a positive effect of text and graph combination, a piece of text corresponding to the app name does not help boost the attractivity of the app. A possible explanation would be that the app name juxtaposed with icon on app store page overshadow the text displayed on icon itself. Furthermore, it is not surprising to see that a general layout of the app store page makes the text on icon tinier and illegible.

Second, incorporating spokes-characters or portrait photos into icon design does not have a significant effect on the download times of the app neither. Previous works have confirmed the importance of spokes-character and human face in advertising. However, these characteristics do not distinctly increase the appeal of apps. The exact reasons for this are hard to tell. But by viewing the icons in the dataset, I have an impression that not only the presence of face characteristics counts but the overall design and the effect with other visual elements (colour, balance etc.) or app category might also play a role.

## 6.2 Implications

This study enriches current literature on mobile app analysis visual complexity. The study differentiates from previous ones from two aspects. First, external validity is ensured with real-world data. Experimental studies give definite conclusions, but sometimes fall short of representativeness and suffers from subjectivity. This paper aims to provide practical insights. Second, two new factors are explored in a large scale with the help of state-of-art computer vision models. Text and spokes-characters have both received much attention in marketing and advertising domains. This study stands out to propose and test the same theories in app icon design field.

As to the managerial implications, the study clarifies two potential misunderstandings on icon design. User's downloading behaviour is an intricate issue that is hard to predict and manipulate, but app developers can avoid thinking twice about whether to embellish text, design spokes characters or select a portrait photo. These are definitely not the decisive factors for the success of an app.

## 6.3 Limitations

The paper has some limitations that could act as fruitful areas for future research. First, fine-tuned models on the icon image dataset could possibly achieve better recognition results. Compared to experiment design, research on observable data requires a high-level accuracy of the data. In this study, two treatment variables obtained from pre-trained computer vision models are not 100% accurate, which add noise to the results. Second, some prior studies presented that word-of-month and information cascade (conformity) exert influences on people's downloading decision. In the matching phase, these factors are hard to be excluded. Future studies may focus on alternatives to address this issue. Third, although several criteria are applied and I limit this study to free English game apps, it is still interesting to investigate the same hypotheses on other subclassifications of apps. Last but not least, future work could consider moderating and mediating effects with other visual elements on the icon.

To sum up, I hope that this work can pave the way for future research in this emerging and important area.

# Bibliography

[1] Number of mobile app downloads worldwide from 2014 to 2023, by region (in billions), Retrieved May 15, 2021, from Statista website: https://www.statista.com/statistics/266488/forecast-of-mobile-app-downloads/
[2] Top Google Play Store Statistics 2019-2020 You Must Know https://appinventiv.com/
[3] Qiang, S., & Fei, H. (2016). An icon design approach based on symbolic and users' cognitive psychology. Indones J Electr Eng Comput Sci., 4(3), 695-705.
[4] What Prompts Smartphone Users to Download Apps – and Why Do They Delete Them? Retrieved May 28,2021, from marketingcharts website: https://www.marketingcharts.com/digital-64747
[5] The State of App Discovery in 2015. Retrieved May 28,2021, from TUNE website: https://www.tune.com/blog/new-white-paper-the-state-of-app-discovery-in-2015/
[6] Hsu, C. L., & Lin, J. C. C. (2015). What drives purchase intention for paid mobile apps?–An expectation confirmation model with perceived value. Electronic Commerce Research and Applications, 14(1), 46-57.
[7] Horton, W. K. (1994). The icon book: Visual symbols for computer systems and documentation. John Wiley & Sons, Inc.
[8] Goonetilleke, R. S., Shih, H. M., & FRITSCH, J. (2001). Effects of training and representational characteristics in icon design. International Journal of Human-Computer Studies, 55(5), 741-760.
[9] Shu, W., & Lin, C. S. (2014). Icon design and game app adoption.
[10] Wang, M., & Li, X. (2017). Effects of the aesthetic design of icons on app downloads: evidence from an android market. Electronic Commerce Research, 17(1), 83-102.
[11] Jylhä, H., & Hamari, J. (2019). An icon that everyone wants to click: How perceived aesthetic qualities predict app icon successfulness. International Journal of Human-Computer Studies, 130, 73-85.
[12] Van Rompay, T. J., De Vries, P. W., & Van Venrooij, X. G. (2010). More than words: on the importance of picture–text congruence in the online environment. Journal of Interactive Marketing, 24(1), 22-30.
[13] Sipe, L. R. (2012). Revisiting the relationships between text and pictures. Children's Literature in Education, 43(1), 4-21.
[14] Pittard, N., Ewing, M., & Jevons, C. (2007). Aesthetic theory and logo design: examining consumer response to proportion across cultures. International Marketing Review.
[15] Bresciani, S., & Del Ponte, P. (2017). New brand logo design: customers' preference for brand name and icon. Journal of Brand Management, 24(5), 375-390.
[16] Paivio, A. (2013). Imagery and verbal processes. Psychology Press.
[17] Callcott, M. F., & Phillips, B. J. (1996). Observations: Elves make good cookies: Creating likable spokes-character advertising. Journal of Advertising Research, 36(5), 73-73.

[18] Garretson, J. A., & Niedrich, R. W. (2004). Spokes-characters: Creating character trust and positive brand attitudes. Journal of advertising, 33(2), 25-36.
[19] Callcott, M. F., & Alvey, P. A. (1991). Toons sell... and sometimes they don't: An advertising spokes-character typology and exploratory study. In Proceedings of the 1991 Conference of the American Academy of Advertising (pp. 43-52). New York: D'Arcy Masius Benton and Bowles.
[20] Chartboost, Power-Up Report – July 2015. Retrieved June 4, 2021, from https://chartboost.s3.amazonaws.com/blog/power-up-report-july-2015-building-an-empire-mobile-strategy-games.pdf
[21] Huang, W. S., Hsieh, T., & Chen, H. S. (2011). The advertisement effectiveness of animated spokes-characters. African journal of business management, 5(23), 9971-9978.
[22] Xiao, L., & Ding, M. (2014). Just the faces: Exploring the effects of facial features in print advertising. Marketing Science, 33(3), 338-352.
[23] Gautham, P. (2020) Google-Playstore-Dataset [Source code]. https://www.kaggle.com/gauthamp10/google-playstore-apps
[24] Petty, R. E., & Cacioppo, J. T. (1986). The elaboration likelihood model of persuasion. In Communication and persuasion (pp. 1-24). Springer, New York, NY.
[25] Zelinsky, G. J., & Murphy, G. L. (2000). Synchronizing visual and language processing: An effect of object name length on eye movements. Psychological Science, 11(2), 125-131.
[26] Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084.
[27] Novelty and Outlier Detection. Retrieved June 10,2021, from scikit-learn website:https://scikit-learn.org/stable/modules/outlier_detection.html#overview-of-outlier-detection-methods
[28] Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). East: an efficient and accurate scene text detector. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 5551-5560).
[29] Abhishek, S. (2019) EAST Detector for Text Detection [Source code]. https://github.com/ZER-0-NE/EAST-Detector-for-text-detection-using-OpenCV#east-detector-for-text-detection
[30] Smith, R. (2007, September). An overview of the Tesseract OCR engine. In Ninth international conference on document analysis and recognition (ICDAR 2007) (Vol. 2, pp. 629-633). IEEE.
[31] Mithe, R., Indalkar, S., & Divekar, N. (2013). Optical character recognition. International journal of recent technology and engineering (IJRTE), 2(1), 72-75.
[32] Bilenko, M., & Mooney, R. J. (2003, August). Adaptive duplicate detection using learnable string similarity measures. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 39-48).
[33] Viola, P., & Jones, M. (2001, December). Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001 (Vol. 1, pp. I-I). IEEE.

[34] Face Detection using Haar Cascades. Retrieved June 10, 2021, from Opencv website: https://docs.opencv.org/master/d2/d99/tutorial_js_face_detection.html
[35] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10), 1499-1503.
[36] Jha, S., Agarwal, N., & Agarwal, S. (2018). Bringing Cartoons to Life: Towards Improved Cartoon Face Detection and Recognition Systems. arXiv preprint arXiv:1804.01753.
[37] Zhou, X. (2018) Anime-Face-Detector [Source code]. https://github.com/qhgz2013/anime-face-detector
[38] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497.
[39] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).
[40] MatchIt: Getting Started. Retrieved June 15, 2021, from The Comprehensive R Archive Network website: https://cran.r-project.org/web/packages/MatchIt/vignettes/MatchIt.html
[41] Rosenbaum, P. R., & Rubin, D. B. (1983). The central role of the propensity score in observational studies for causal effects. Biometrika, 70(1), 41-55.
[42] Hoken, H., & Su, Q. (2015). Measuring the effect of agricultural cooperatives on household income using PSM-DID: a case study of a rice-producing cooperative in China (No. 539). Institute of Developing Economies, Japan External Trade Organization (JETRO).
[43] Tolles, J., & Meurer, W. J. (2016). Logistic regression: relating patient characteristics to outcomes. Jama, 316(5), 533-534.
[44] Matching Methods. Retrieved June 21, 2021, from The Comprehensive R Archive Network website: https://cran.r-project.org/web/packages/MatchIt/vignettes/matching-methods.html
[45] Ho, D. E., Imai, K., King, G., & Stuart, E. A. (2007). Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference. Political analysis, 15(3), 199-236.
[46] Lunt, M. (2014). Selecting an appropriate caliper can be essential for achieving good balance with propensity score matching. American journal of epidemiology, 179(2), 226-235.
[47] Estimating Effects After Matching. Retrieved June 21, 2021, from The Comprehensive R Archive Network website: https://cran.r-project.org/web/packages/MatchIt/vignettes/estimating-effects.html

# Appendix

## A. Google Play-Store Dataset Summary

| Column Name | Description | Data Type | Missing Percentage |
|---|---|---|---|
| **App Name** | Name of the app | String | 0% |
| **App Id** | Package name | String | 0% |
| **Category** | App category | String | 0% |
| **Rating** | Average rating | Integer | 1% |
| **Rating Count** | Number of rating | Integer | 1% |
| **Installs** | Approximate install count | String | 0% |
| **Minimum Installs** | Approximate minimum app install count | Integer | 0% |
| **Maximum Installs** | Approximate maximum app install count | Integer | 0% |
| **Free** | Whether app is Free or Paid | Boolean | 0% |
| **Price** | App price | Integer | 0% |
| **Currency** | App currency | String | 0% |
| **Size** | Size of application package | String | 0% |
| **Minimum Android** | Minimum android version supported | String | 0% |
| **Developer Id** | Developer Id in Google Playstore | String | 0% |
| **Developer Website** | Website of the developer | String | 33% |
| **Developer Email** | Email-id of developer | String | 0% |
| **Released** | App launch date on Google Playstore | Date | 3% |
| **Last Updated** | Last app update date | Date | 0% |
| **Content Rating** | Maturity level of app | String | 0% |
| **Privacy Policy** | Privacy policy from developer | String | 18% |
| **Ad Supported** | Ad support in app | Boolean | 0% |
| **In app purchases** | In-App purchases in app | Boolean | 0% |
| **Editor Choice** | Whether rated as Editor Choice. | Boolean | 0% |

## B. The summaries of nearest neighbour matching

| | **Before Matching – All Data** | | | **After Matching – Matched Data** | | | |
|---|---|---|---|---|---|---|---|
| | Means Treated | Means Control | Std. Mean Diff. | Means Treated | Means Control | Std. Mean Diff. | Std. Pair Dist. |
| **distance** | 0.2176 | 0.1306 | 0.6887 | 0.2165 | 0.2161 | 0.0025 | 0.0042 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Rating** | 2.8697 | 3.1956 | -0.1674 | 2.8642 | 2.8523 | 0.0061 | 0.6868 |
| **Rating_Count** | 16654.71 | 22250.19 | -0.0364 | 16338.6869 | 15671.3579 | 0.0043 | 0.1892 |
| **Size** | 32.6925 | 41.4498 | -0.2997 | 32.6554 | 33.4911 | -0.0286 | 0.6230 |
| **Released_days** | 718.9146 | 695.7503 | 0.0293 | 718.2495 | 721.3077 | -0.0039 | 0.8901 |
| **Last_version_days** | 64.4402 | 65.9707 | -0.0353 | 64.3640 | 64.5206 | -0.0036 | 1.1259 |
| **Namelength** | 26.7467 | 27.3983 | -0.0503 | 26.7065 | 26.5751 | 0.0101 | 1.0770 |
| **Privacy_Policy** | 0.9287 | 0.9372 | -0.0331 | 0.9286 | 0.9246 | 0.0156 | 0.5126 |
| **Ad_Supported** | 0.7681 | 0.8797 | -0.2644 | 0.7693 | 0.7556 | 0.0324 | 0.4577 |
| **Editors_Choice** | 0.0034 | 0.0064 | -0.0531 | 0.0032 | 0.0032 | 0.0000 | 0.0058 |
| **Category_Adventure** | 0.0361 | 0.0576 | -0.1148 | 0.0364 | 0.0364 | 0.0000 | 0.0000 |
| **Category_Arcade** | 0.0476 | 0.0886 | -0.1924 | 0.0478 | 0.0478 | 0.0000 | 0.0000 |
| **Category_Board** | 0.0283 | 0.0328 | -0.0269 | 0.0282 | 0.0282 | 0.0000 | 0.0000 |
| **Category_Card** | 0.0534 | 0.0256 | 0.1238 | 0.0525 | 0.0525 | 0.0000 | 0.0000 |
| **Category_Casino** | 0.0403 | 0.0104 | 0.1520 | 0.0398 | 0.0398 | 0.0000 | 0.0000 |
| **Category_Casual** | 0.0603 | 0.1278 | -0.2835 | 0.0607 | 0.0607 | 0.0000 | 0.0000 |
| **Category_Educational** | 0.0585 | 0.0585 | 0.0001 | 0.0588 | 0.0588 | 0.0000 | 0.0000 |
| **Category_Music** | 0.0225 | 0.0162 | 0.0424 | 0.0227 | 0.0227 | 0.0000 | 0.0000 |
| **Category_Puzzle** | 0.0815 | 0.1411 | -0.2175 | 0.0822 | 0.0822 | 0.0000 | 0.0000 |
| **Category_Racing** | 0.0154 | 0.0290 | -0.1108 | 0.0151 | 0.0151 | 0.0000 | 0.0000 |
| **Category_Role_Playing** | 0.0217 | 0.0470 | -0.1732 | 0.0219 | 0.0219 | 0.0000 | 0.0000 |
| **Category_Simulation** | 0.0544 | 0.0989 | -0.1962 | 0.0548 | 0.0548 | 0.0000 | 0.0000 |
| **Category_Sports** | 0.3310 | 0.1146 | 0.4598 | 0.3312 | 0.3312 | 0.0000 | 0.0000 |
| **Category_Strategy** | 0.0179 | 0.0259 | -0.0599 | 0.0180 | 0.0180 | 0.0000 | 0.0000 |
| **Category_Trivia** | 0.0536 | 0.0283 | 0.1123 | 0.0522 | 0.0522 | 0.0000 | 0.0000 |
| **Category_Word** | 0.0438 | 0.0247 | 0.0933 | 0.0438 | 0.0438 | 0.0000 | 0.0000 |
| **Min_Android_4.0** | 0.0193 | 0.0198 | -0.0033 | 0.0195 | 0.0184 | 0.0082 | 0.2725 |
| **Min_Android_4.0.3** | 0.0192 | 0.0152 | 0.0294 | 0.0193 | 0.0214 | -0.0153 | 0.2877 |
| **Min_Android_4.1** | 0.3331 | 0.3640 | -0.0656 | 0.3330 | 0.3246 | 0.0178 | 0.8674 |
| **Min_Android_4.2** | 0.0641 | 0.0492 | 0.0610 | 0.0643 | 0.0630 | 0.0053 | 0.4156 |
| **Min_Android_4.3** | 0.0155 | 0.0210 | -0.0444 | 0.0156 | 0.0179 | -0.0183 | 0.2633 |
| **Min_Android_4.4** | 0.2592 | 0.2856 | -0.0602 | 0.2601 | 0.2613 | -0.0026 | 0.8385 |
| **Min_Android_4.4W** | 0.0034 | 0.0053 | -0.0337 | 0.0034 | 0.0042 | -0.0139 | 0.1197 |
| **Min_Android_5.0** | 0.1884 | 0.1667 | 0.0555 | 0.1877 | 0.1931 | -0.0140 | 0.7210 |
| **Min_Android_5.1** | 0.0401 | 0.0284 | 0.0596 | 0.0404 | 0.0403 | 0.0008 | 0.3800 |
| **Min_Android_6.0** | 0.0341 | 0.0244 | 0.0530 | 0.0338 | 0.0324 | 0.0080 | 0.3455 |
| **Min_Android_7.1** | 0.0117 | 0.0085 | 0.0299 | 0.0116 | 0.0121 | -0.0045 | 0.2085 |
| **Min_Android_8.0** | 0.0042 | 0.0039 | 0.0032 | 0.0042 | 0.0043 | -0.0025 | 0.1277 |
| **Min_Android_Varies** | 0.0021 | 0.0011 | 0.0204 | 0.0014 | 0.0018 | -0.0071 | 0.0707 |
| **Content_Rating_Everyone** | 0.7841 | 0.7597 | 0.0594 | 0.7893 | 0.7893 | 0.0000 | 0.0000 |
| **Content_Rating_10+** | 0.0317 | 0.0589 | -0.1555 | 0.0312 | 0.0312 | 0.0000 | 0.0000 |
| **Content_Rating_17+** | 0.0221 | 0.0199 | 0.0145 | 0.0184 | 0.0184 | 0.0000 | 0.0000 |
| **Content_Rating_Teen** | 0.1610 | 0.1613 | -0.0009 | 0.1609 | 0.1609 | 0.0000 | 0.0000 |
| **Treatment sample size** | | 6254 | | | 6208 | | |
| **Control sample size** | | 37478 | | | 6208 | | |

Balance Summary of using Nearest Neighbour Matching for text treatment.

| | **Before Matching – All Data** | | | **After Matching – Matched Data** | | | |
|---|---|---|---|---|---|---|---|
| | Means Treated | Means Control | Std. Mean Diff. | Means Treated | Means Control | Std. Mean Diff. | Std. Pair Dist. |
| **distance** | 0.2347 | 0.1540 | 0.6612 | 0.2318 | 0.2308 | 0.0086 | 0.0113 |
| **Rating** | 3.4148 | 3.0956 | 0.2092 | 3.4089 | 3.4176 | -0.0057 | 0.8282 |
| **Rating_Count** | 32089.8366 | 19308.9336 | 0.0616 | 31223.0512 | 26156.1598 | 0.0244 | 0.2517 |
| **Size** | 49.3027 | 38.3652 | 0.3385 | 48.8800 | 48.0620 | 0.0253 | 0.7693 |
| **Released_days** | 678.6152 | 703.1776 | -0.0344 | 678.0441 | 676.3377 | 0.0024 | 0.9913 |
| **Last_version_days** | 65.8494 | 65.7322 | 0.0026 | 65.9111 | 66.2648 | -0.0078 | 1.1198 |
| **Namelength** | 31.2013 | 26.5210 | 0.3622 | 31.1288 | 31.1591 | -0.0023 | 0.8188 |
| **Privacy_Policy** | 0.9601 | 0.9311 | 0.1483 | 0.9595 | 0.9599 | -0.0021 | 0.3424 |
| **Ad_Supported** | 0.8909 | 0.8583 | 0.1047 | 0.8908 | 0.8989 | -0.0258 | 0.5446 |
| **Editors_Choice** | 0.0109 | 0.0050 | 0.0570 | 0.0104 | 0.0085 | 0.0187 | 0.1708 |
| **Category_Adventure** | 0.0669 | 0.0520 | 0.0596 | 0.0674 | 0.0674 | 0.0000 | 0.0000 |
| **Category_Arcade** | 0.0446 | 0.0905 | -0.2219 | 0.0452 | 0.0452 | 0.0000 | 0.0000 |
| **Category_Board** | 0.0195 | 0.0347 | -0.1095 | 0.0197 | 0.0197 | 0.0000 | 0.0000 |
| **Category_Card** | 0.0248 | 0.0305 | -0.0363 | 0.0248 | 0.0248 | 0.0000 | 0.0000 |
| **Category_Casino** | 0.0153 | 0.0146 | 0.0059 | 0.0151 | 0.0151 | 0.0000 | 0.0000 |
| **Category_Casual** | 0.1590 | 0.1099 | 0.1344 | 0.1574 | 0.1574 | 0.0000 | 0.0000 |
| **Category_Educational** | 0.0800 | 0.0542 | 0.0952 | 0.0810 | 0.0810 | 0.0000 | 0.0000 |
| **Category_Music** | 0.0284 | 0.0149 | 0.0813 | 0.0280 | 0.0280 | 0.0000 | 0.0000 |
| **Category_Puzzle** | 0.0984 | 0.1394 | -0.1377 | 0.0981 | 0.0981 | 0.0000 | 0.0000 |
| **Category_Racing** | 0.0093 | 0.0306 | -0.2223 | 0.0093 | 0.0093 | 0.0000 | 0.0000 |
| **Category_Role_Playing** | 0.1065 | 0.0307 | 0.2456 | 0.1064 | 0.1064 | 0.0000 | 0.0000 |
| **Category_Simulation** | 0.1080 | 0.0894 | 0.0599 | 0.1071 | 0.1071 | 0.0000 | 0.0000 |
| **Category_Sports** | 0.0661 | 0.1615 | -0.3844 | 0.0670 | 0.0670 | 0.0000 | 0.0000 |
| **Category_Strategy** | 0.0384 | 0.0220 | 0.0853 | 0.0377 | 0.0377 | 0.0000 | 0.0000 |
| **Category_Trivia** | 0.0487 | 0.0285 | 0.0939 | 0.0483 | 0.0483 | 0.0000 | 0.0000 |
| **Category_Word** | 0.0108 | 0.0308 | -0.1937 | 0.0110 | 0.0110 | 0.0000 | 0.0000 |
| **Min_Android_4.0** | 0.0172 | 0.0202 | -0.0234 | 0.0175 | 0.0184 | -0.0075 | 0.2699 |
| **Min_Android_4.0.3** | 0.0158 | 0.0157 | 0.0010 | 0.0160 | 0.0157 | 0.0022 | 0.2355 |
| **Min_Android_4.1** | 0.3778 | 0.3559 | 0.0453 | 0.3774 | 0.3886 | -0.0232 | 0.9346 |
| **Min_Android_4.2** | 0.0815 | 0.0452 | 0.1325 | 0.0793 | 0.0782 | 0.0041 | 0.4208 |
| **Min_Android_4.3** | 0.0190 | 0.0205 | -0.0109 | 0.0191 | 0.0182 | 0.0071 | 0.2592 |
| **Min_Android_4.4** | 0.2734 | 0.2835 | -0.0226 | 0.2752 | 0.2703 | 0.0109 | 0.8542 |
| **Min_Android_4.4W** | 0.0038 | 0.0053 | -0.0235 | 0.0039 | 0.0043 | -0.0067 | 0.1326 |
| **Min_Android_5.0** | 0.1506 | 0.1736 | -0.0645 | 0.1508 | 0.1455 | 0.0147 | 0.6640 |
| **Min_Android_5.1** | 0.0222 | 0.0317 | -0.0641 | 0.0225 | 0.0229 | -0.0028 | 0.2924 |
| **Min_Android_6.0** | 0.0227 | 0.0265 | -0.0255 | 0.0226 | 0.0202 | 0.0158 | 0.2843 |
| **Min_Android_7.1** | 0.0056 | 0.0096 | -0.0535 | 0.0055 | 0.0062 | -0.0093 | 0.1469 |
| **Min_Android_8.0** | 0.0015 | 0.0045 | -0.0769 | 0.0014 | 0.0015 | -0.0036 | 0.0681 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Min_Android_Varies** | 0.0018 | 0.0012 | 0.0141 | 0.0017 | 0.0022 | -0.0132 | 0.0923 |
| **Content_Rating_Everyone** | 0.6500 | 0.7860 | -0.2850 | 0.6592 | 0.6592 | 0.0000 | 0.0000 |
| **Content_Rating_10+** | 0.0657 | 0.0528 | 0.0517 | 0.0645 | 0.0645 | 0.0000 | 0.0000 |
| **Content_Rating_17+** | 0.0373 | 0.0168 | 0.1080 | 0.0304 | 0.0304 | 0.0000 | 0.0000 |
| **Content_Rating_Teen** | 0.2469 | 0.1441 | 0.2385 | 0.2459 | 0.2459 | 0.0000 | 0.0000 |
| **Treatment sample size** | | 7326 | | | 7210 | | |
| **Control sample size** | | 36406 | | | 7210 | | |

Balance Summary of using Nearest Neighbour Matching for face treatment.