UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC)
BarcelonaTech

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

MASTER IN INNOVATION AND RESEARCH IN INFORMATICS

Data Science

Predicting the number of likes on Instagram with TensorFlow

Author
Joel Cantero Priego

Advisor
Alfredo Vellido Alcacena
Department
Computer Science

Date of Defense
October 26, 2020

Abstract

1 billion people use Instagram every month[1], which makes it one of the most popular social networks worldwide [2]. Currently, there is an enormous scope market with the potential to be optimized to increase Instagram posts popularity and engagement. In this project, the main goal is to predict the number of likes given a post, building a deep learning model. This model will use convolutional neural networks, natural language processing and other deep learning techniques within the TensorFlow framework. The input data is composed of categorical, numerical data, as well as some image and text data.

Index

List of Figures

List of Tables

1 Introduction

1.1 Context

1 billion people use Instagram every month [1], which makes it one of the most popular social networks worldwide [2]. Besides, 90% of accounts follow a business account on Instagram [3]. Due to the app's content, it is a valuable social media marketing tool for business, a huge opportunity for brands.

On the other hand, influencers, who are creators with a large number of followers that share content on social networks, can build communities around topics and niches. They use Instagram, as a visual platform, to make regular posts and generate large followings of engaged people who pay close attention to their views.

For that reason, there is a special relationship between influencers and brands. Influencers know their followers and are conscious about what kind of content they want to consume, that way, it can help brands to communicate their messages. Commonly, Instagram influencers get paid by brands to make the promotion of a product and service.

Currently, there is an enormous scope market with the potential to be optimized to increase Instagram posts popularity and engagement. So for that reason, I proposed to Alfredo Vellido to create a system that predicts the number of likes given a post and identify which variables help to increase it. Using that system, it will generate an estimation so that posts can be optimized to gather the most amount of visibility as well as engagement. This outcome is interesting to many stakeholders, such as digital marketers, influencers, and even regular users.

1.2 Goals

As mentioned before, the main project goal is to predict the number of likes given a post, building a Deep Learning model. This model will use convolutional neural networks, natural language processing and other Deep Learning techniques within the TensorFlow framework. The input data is composed of categorical, numerical data, as well as some image and text data.

Beforehand, we will perform a state of the art study of predicting the popularity of an image. Next, we will perform a full univariate and multivariate

analysis in our exploratory data analysis process to study our dataset obtained.

To measure our model performance, we will need to establish a methodology that evaluates our model versions through accuracy indicators.

These goals are going to be detailed and explained in the following sections.

1.3 Motivation

When I was coursing the last year of the bachelor's degree in Informatics Engineering, one of my lasts subjects was Applied Engineering Project [4]. The goal of this subject is to work on a multi-disciplinary project taking a base on a challenge defined by a company. In my case, I had the first opportunity to work on a machine learning project in which the aim was to classify dental implants given a dental x-ray. I really enjoyed the project development as well as I was discovering the Data Science world for the first time. When I completed my degree, I understood that there are numerous fields of computer science and I had to choose one of them. For that reason, I decided to keep studying coursing a master's degree in Innovation and Research in Informatics, major in Data Science.

When I had to choose my master thesis, it was clear to me that I wanted it to be TensorFlow [5] related project, so I asked to Alfredo Vellido to be my advisor. I chose TensorFlow framework to develop this project for the following reasons:

- **I have been working as a data scientist** for a year and a half at Catalana Occident [6] and we use TensorFlow to solve our machine learning problems.
- **TensorFlow is one of the most popular deep learning frameworks.** Besides, it is still growing fast because of its developer community, it is an open source library, and of course, because it is created by Google, the second most valuable brand in the world [7] .
- **TensorFlow uses Keras** API which allows us to build complex deep neural networks from a higher implementation level. Keras helps to manage input layers with different data types: numerical, categorical, text and image data.

2 State-of-the-art

Actually, there is much interest among regular users to predict the popularity in social networks and has received a lot of attention from the researchers. While the greater part of the past works has focused on predicting popularity of text content, such as messages on Twitter, and some recent works on YouTube video popularity, significantly less effort has been expended in the prediction of image popularity.

There is some literature and systems about predicting the popularity of an image before it is uploaded.

In line with Kosha's work, What Makes An Image Popular? [8], image analysis is divided into two key components:

- **Image cues.** The image content itself. We can investigate many features that could be used to explain the popularity of images. In our project goal, these features will be identified by a convolutional neural network.
- **Social cues.** In the Instagram context, social cues play a significant role in the number of likes an image will receive. A user with a larger number of followers would naturally receive a higher number of likes. Additionally, we can assume that the post time of the post and the average of number of likes are as well as social indicators, among others.

Crystal J. Qian's [9] demonstrated that some interesting features that can predict the popularity of an image. Comment count has the strongest predictive power, corresponding positively to popularity. A greater side length image also corresponds positively, likely because larger images tend to be of higher quality.

2.1 Existing applications

Nowadays, many applications try to predict the popularity of Instagram posts.

2.1.1 LikelyAI

LikelyAI [10] was developed by former Facebook and Google employees, it extracts thousands of data points from an image and recognizes the popular patterns. Objects, colors, emotions, shapes, lighting, size and position – all of them are data points. The user has to select multiple photos to evaluate, then the algorithm evaluates every single photo and finally the app sorts his photos and display results.

It seems that the project was discontinued because they have not updated the app since October 2017.

2.1.2 Beautiful Destinations

Beautiful Destinations [11], a travel and lifestyle brand that has more than 10,5 million followers in 180 countries, built an algorithm [12] that tells how many likes and comments a photo will get.

They created the algorithm by collecting a lot of photos from many social networks (Instagram, Flicker and Pinterest) that had reactions to make correlations.

This software helps to their community managers to choose which Instagram photo to use, the one that will obtain the most likes.

The fact is that the app is just for company use, it is not public. The algorithm is trained with just travel and lifestyle pictures, and for this reason, the algorithm works better in this specific topic.

Beautiful Destinations is already working on a new task: predicting audience engagement with videos.

In line with Beautiful Destinations, this project will focus on just one topic to predict image popularity. This is one of the most key points to fit our algorithm on a specific image topic.

3 Methodology

In this following section we are going to specify our methodology that is going to be applied in this project. Firstly, all the steps included in our workflow will be described. Also, all the technologies that will be used to successfully

build the model will be listed, as well as the justification for why we have chosen them.

3.1 Strategy

Using a correct machine learning strategy helps to get our machine learning systems much more quickly and efficiently. As Andrew Ng states in his course AI For Everyone [13], there are three key steps of a machine learning project:

- **Collect data.** Data collection is one of the most important tasks in a machine learning project. In this step we gather information from one or different sources. Collecting data will allows us to use data analysis to find recurring patterns and to build predictive models. Predictive models are only as satisfactory as the data from which they are built. Putting all our effort in that step is crucial to developing fine models. After the data is collected, there are two more steps: data preparation and data preprocessing.
  - **Data preparation:** is the process to transform the collected data into a form that is more appropriate for modeling. It includes a data cleaning (identifying missing values), data selection and data transformation (changing the type of variables).
  - **Data preprocessing:** is the step in which the data is encoded to bring it to a state that can now be easily interpreted by the machine learning algorithm [14]. This includes proper labeling of categorical variables (converting them to numerical variables), cleaning text, and standardization of numeric variables.
- **Train model.** This means we will use a deep learning model to learn our input (variables, text and image) to output the prediction, in this case, the number of likes of the post. The process of training a model is to determine correct values for all the weights and the bias from labeled examples, in this case, posts that we know the number of likes gathered. As we have labeled examples, a supervised learning algorithm is used. These sorts of algorithms build a model by examining a large number of labeled examples and try to figure out a model that minimizes loss, the penalty for an incorrect prediction.
- **Deploy model.** Once the model is trained and tested, the model is

deployed. However, the process is not done: we can get data back, obtain new examples of Instagram posts with their number of likes (new labeled data) to maintain and update the model. Hopefully, the model continually gets better and better to the point where we end up with a software that can do a good job predicting the number of likes from an unpublished Instagram post.

These steps are not sequential. As Andrew Ng states in *Machine Learning Yearning* [15], machine learning workflow is a very iterative process:

"Build your first system quickly, then iterate".

Andrew Ng

For that reason, the first model will be designed and built quickly and then we will try many ideas before finding the final model version. This means that probably we will iterate collecting, preparing and preprocessing data, even when the model is deployed.

3.2 Technologies

3.2.1 R

R [16] is a language and environment for statistical computing and graphics .R provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

R will be used as a programming language to perform the exploratory data analysis, using an Integrated Development Environment (IDE) called RStudio.

3.2.2 RStudio

RStudio [17] is a set of integrated tools designed to help you be more productive with R and Python. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.

RStudio will be used to perform the exploratory data analysis and to show the results.

3.2.3 Git

Git [18] is a distributed version control system. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

3.2.4 GitHub

GitHub [19] is a distributed version-control platform that uses Git here users can collaborate on or adopt open source code projects, fork code, share ideas and more. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. GitHub offers its basic services free of charge. Its more advanced professional and enterprise services are commercial.

GitHub will be used as a version control system to manage the project code.

3.2.5 TensorFlow

TensorFlow [5] is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow will be used as a deep learning framework to train the neural network of our project.

3.2.6 Python packages

In this section, all the python packages needed for the project development will be listed.

3.2.6.1 Pandas

Pandas [20] is a fast, powerful, flexible and easy to use open source tool for data manipulation and analysis, written for the Python programming language. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Pandas will be used as a data manipulation tool for load our dataset and preprocess it.

3.2.6.2 Selenium WebDriver

Selenium [21] Selenium WebDriver drives a browser natively, as a real user would, either locally or on remote machines.

Pandas will be used as a data manipulation tool to load our dataset and preprocess it.

3.2.6.3 Scikit-learn

Scikit-learn [22] is a free software machine learning library for Python. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Scikit-learn will be used as a data manipulation tool to split our dataset into train, validation and test set.

3.2.6.4 fastText

fastText [23] is a library for the learning of word embeddings and text classification created by Facebook's AI Research. The model allows one to create an unsupervised learning or supervised learning algorithms for obtaining vector representations for words. fastText uses a neural network for word embedding. The algorithm of fasttext is based on these two papers. *Enriching Word Vectors with Subword Information* [24] and *Bag of Tricks for Efficient Text Classification* [25]

fastText will be used for obtaining vector representations for our text data (post caption and user biography). An own fasttext model will be trained (train_unsupervised function) using cbow (continuous bag of words). Cbow model predicts the target word according to its context. The context is represented as a bag of the words contained in a fixed size window around the target word.

4 Development

In this chapter, the available data for our project are summarily described, followed by the data acquisition methods, data preparation, data preprocessing and the strategies to increase the interpretability of the data.

Finally, the model architecture and the training process are explained.

4.1 Available data

First of all, the data that are available through Public Instagram API for each user is obtained as a JSON file. For example, if we take a look at the user page, *https://www.instagram.com/cristiano/*, we can read all the posts information if we append the query string *?__a=1* to the url. This is a summary of the available data we can obtain:

Table 1: Available data for user.

Variable	**Type**
biography	String
blocked_by_viewer	Boolean
country_block	Boolean
external_url	String
external_url_linkshimmed	Boolean
edge_followed_by	Numerical
edge_follow	Numerical
follows_viewer	Boolean
full_name	String
has_ar_effects	Boolean
has_channel	Boolean
has_blocked_viewer	Boolean
highlight_reel_count	Boolean
has_requested_viewer	Boolean
id	Numerical
is_business_account	Boolean
is_joined_recently	Boolean
business_category_name	String
category_id	Numerical

Continued on next page

Table 1 – continued from previous page

Variable	Type
overall_category_name	String
is_private	Boolean
is_verified	Boolean
edge_mutual_followed_by	Numerical
profile_pic_url	String
profile_pic_url_hd	String
requested_by_viewer	Boolean
username	String
connected_fb_page	Categorical
higlight_reel_count	Numerical
edge_felix_video_timeline	Numerical
edge_media_collections	Numerical
edge_owner_to_timeline_media	Numerical
edge_saved_media	Numerical
edge_felix_video_timeline	Numerical

Table 2: Available data for post.

Variable	Type
__typename	String
id	Numerical
shortcode	String
height	Numerical
width	Numerical
display_url	String
gating_info	Numerical
fact_check_overall_rating	Boolean
fact_check_information	String
media_preview	String
is_video	Boolean
accessibility_caption	Boolean
edge_media_to_caption	String
edge_media_to_comment	Numerical
comments_disabled	Boolean

Continued on next page

Table 2 – continued from previous page

Variable	Type
taken_at_timestamp	Boolean
edge_liked_by	Numerical
edge_media_preview_like	Numerical
location	String
thumbnail_src	String

4.2 Gathering Instagram influencers

On the other hand, we need to obtain a list of Instagram influencers. We have analyzed more than 10 pages of influencers to choose what website fits better for our project. We are interested in retrieve influencers that accomplish some features: location (Spain), number of followers (between 10.000 and 1.000.000 followers) and category (beauty and/or fashion).

- **Heepsy** [26]: We can access more than 7 million influencers with highly precise search filtering by category, location, engagement rate, etc. Nevertheless, the free plan just retrieves 7 influencers searching by filters.
- **HypeAuditor** [27]: HypeAuditor is an AI-powered Instagram account authenticity checking platform. Its influencers database has more than 16 filters to find the best content creators. The website allows us to filter by location, age, gender, number of followers, level of engagement, a filter that will allow us to check if the followers of their accounts are real or not. On the other hand, we can export search results to CSV file, that is an interesting feature.
- **Coobis** [28]: Coobis allows us for free to filter and select more than 25.000 influencers by different criteria such as the number of followers, country, category, social network, etc.

We decide to obtain our Instagram influencers list through Coobis using its free search tool.

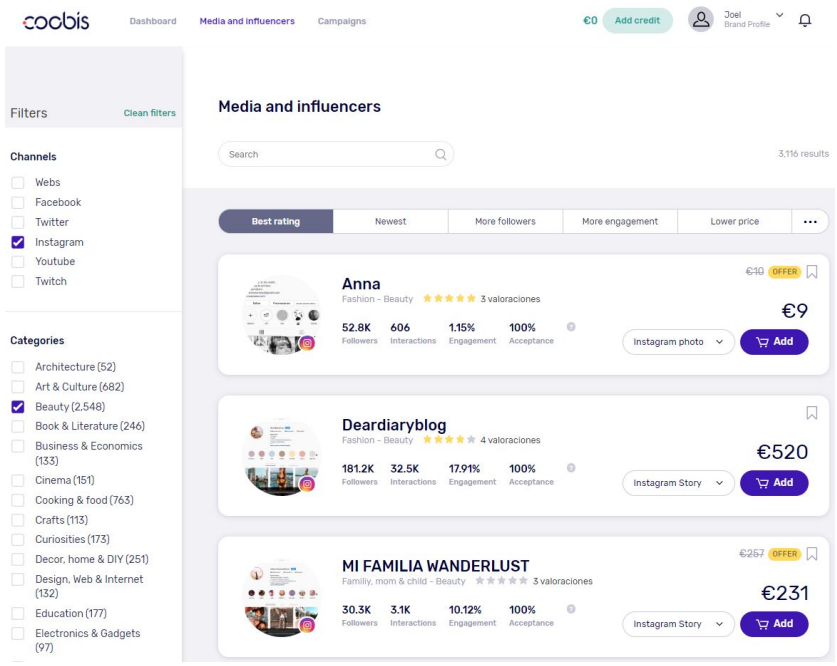

Figure 1: Coobis: Content Marketing, Influencers & Branded Content

According to our goal project, we need to add these filters:

- **Channels**: *Instagram*.
- **Categories**: *Beauty* and *Fashion*.
- **Language**: *Spanish*.
- **Country**: *Spain*.
- **Instagram followers**: from 10.000 to 1.000.000 followers.

The search tool returns us 1012 influencers that accomplish our search settings. Now, executing gathering_usernames python function we gather all these usernames into a CSV file *influencers.csv*.

This function uses Selenium WebDriver [21] that allows us to drive a browser natively, as a real user would, either locally or on remote machines. Thanks to that python library, we can drive Coobis website and gather automatically each username into our list.

4.3 Selecting variables

Before we have to gather each post information, we should select a reduced number of variables available on the JSON file mentioned in the previous section Available Data. In this first iteration, we have to infer which variables have the strongest relationship with the target variable, in this case, the number of likes for each post. We decide to select these variables:

Table 3: Selected variables.

Variable	Description
username	Influencer's username
edge_followed_by	Influencer's followers count
edge_follow	Influencer's following count
edge_owner_to _timeline_media	Total number of posts
is_verified	If the influencer's profile is verified by Instagram
is_business_account	If the influencer's profile is a business account
highlight_reel_count	Number of highlighted stories
biography	Text to share some details about the influencer's profile
display_url	Photo URL of the post
edge_liked_by	Number of likes
caption	Description about the post photo
timestamp	Exact posting time of the post
is_sidecar	If the post is a collection of multiple permanent photos
shortcode	Post identifier

4.4 Gathering post information

Once the selection of variables are decided, the next step is to scrap the Instagram profiles of these influencers obtained in the previous section. This involves executing *gathering_posts* function to read our selected variables from each profile through JSON metadata. For each influencer, *gathering_posts* function will crawl his/her latest 12 posts. The function will not crawl a post if any of these conditions are met:

- **Is a video post**: If the post is a video the function is not interested to crawl it. We are looking for posts that contain an image because later a convolutional neural network (CNN) will be used to extract some features of the image.
- **Is a recent post**: if the post has been published recently (one day), we are not interested in it because the number of likes, our target variable, may not be optimal, it will keep growing.
- **It does not contain a caption**: If the post has not a caption the function will not crawl it. We will extract text features for each post caption using Natural Language Processing techniques.

The *gathering_posts* function returns the final result is a hierarchical JSON file that for each influencer contains each post data. It looks like this:

Figure 2: JSON file obtained for a given influencer

However, executing *gathering_posts* function we figure out that some profiles that are private (37) or do not exists (48). So, the total number of profiles gathered is 931.

First of all, we need to convert this JSON file to a CSV file to provide our future model to ingest the data. The *json_to_csv* python will adapt our JSON to a single CSV file.

9905 post data from 931 Instagram Beauty & Fashion influencers from Spain were collected to train and test our model.

4.5 First data preparation

Before performing the exploratory data analysis, we need to transform the raw data that was collected into a form that can be used in modeling. In *first_ dataprep* function, we use some techniques such as data cleaning (identify missing values) and data transformation (changing the type of variables or adding new ones in the dataset).

- **Converting timestamp**: One of the variables we have selected (Table 3) is timestamp. The timestamp is the exact posting time of the post, but it is just an integer that is not understandable. For that reason, the datetime python package is used to convert it into 3 new variables: weekday, hour, and minute.
- **Discarding users with just one post**: if we have gathered a user with just one post, it will be discarded because the average number of likes will not be consistent.

Table 4: Prepared data variables.

Variable	Description
username	Influencer's username
followers	Influencer's followers count
following	Influencer's following count
total_scrapped_posts	Total number of posts
is_verified	If the influencer's profile is verified by Instagram
is_business_account	If the influencer's profile is a business account
highlight_reel_count	Number of highlighted stories
biography	Text to share some details about the influencer's profile
total_posts	Total number of user posts
post_id	Instagram post identifier
display_url	Image URL of the post
number_of_likes	Number of post likes
caption	Description about the post
is_sidecar	If the post is a collection of multiple permanent photos
weekday	Posting weekday of the post
hour	Posting hour of the post
minute	Posting minute of the post
avg_number_of_likes	Average of user likes count

Once the data is prepared and we can understand all these variables, the data is ready to perform an exploratory data analysis.

4.6 Exploratory Data Analysis

4.6.1 Univariate analysis

For the univariate analysis, density plots were obtained for the numerical variables and bar plots were generated for the categorical variables. These are the results obtained:

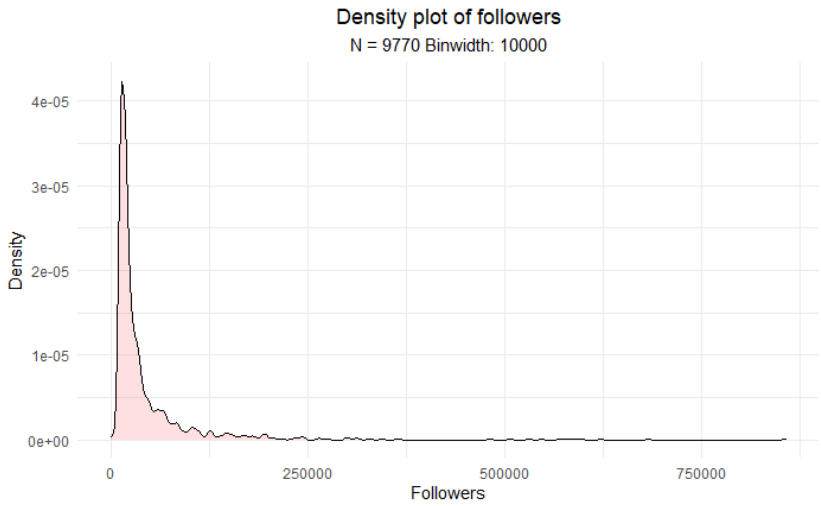

Figure 3: Density plot of followers

When analyzing the followers field, we can see that there are some values with very low representation, but with a clear normal tendency.

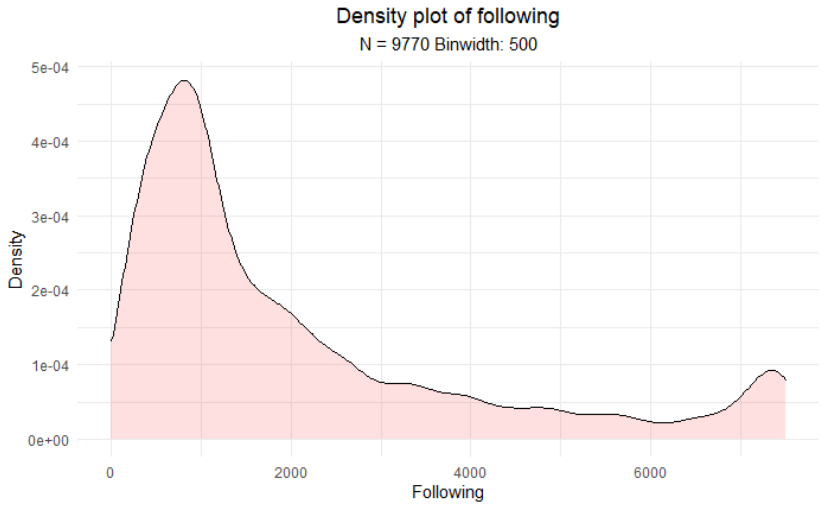

Figure 4: Density plot of following

Here, we can see that the following values are between 0 and 7.500, because Instagram does not allow any user to follow more than 7.500 people, to help reduce spam. Also following field seems sort of normal, so there are no problems here.

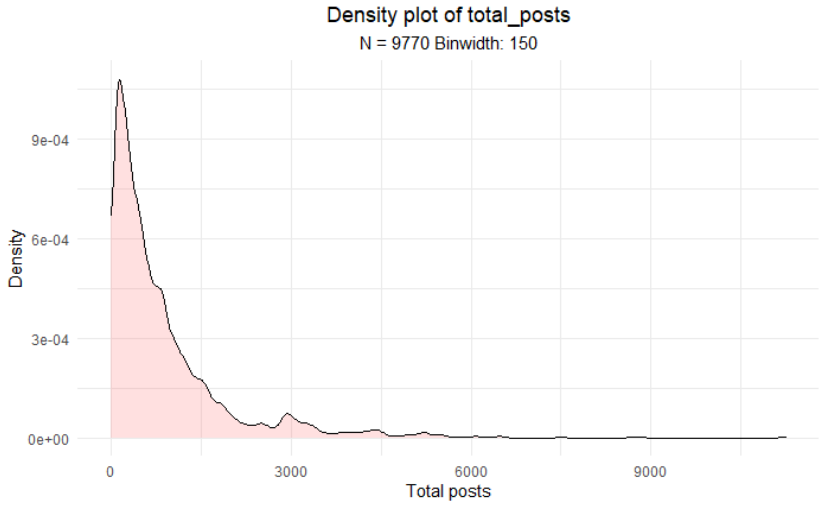

Figure 5: Density plot of total posts

In the total_posts density plot we can observe some outliers values above 5000 posts. It seems sort of a normal tendency.

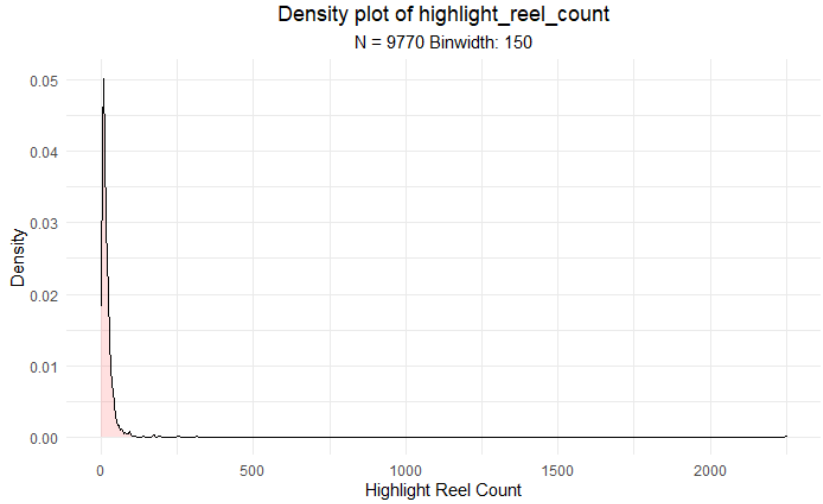

Figure 6: Density plot of highlight reel count

In the highlight_reel_count density plot we can observe as well many outliers values above 20.

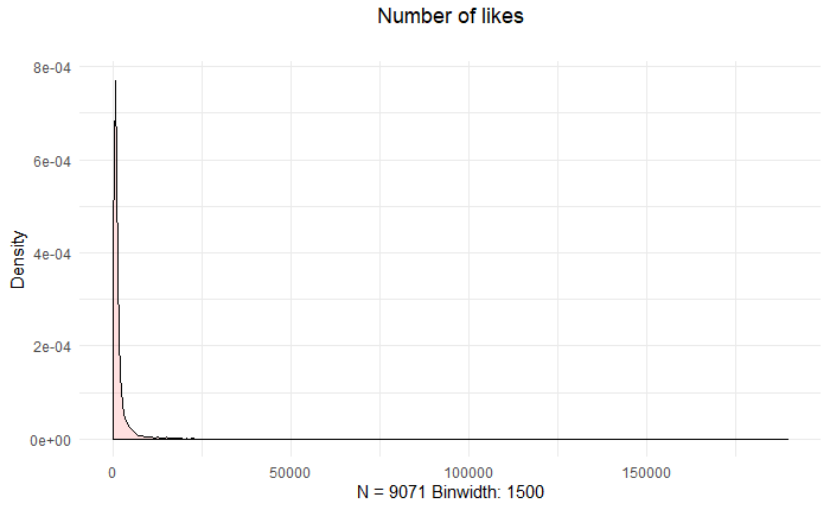

Figure 7: Density plot of the number of likes

In our target variable, number_of_likes, in its density plot can observe also many outliers values above 1678 likes.

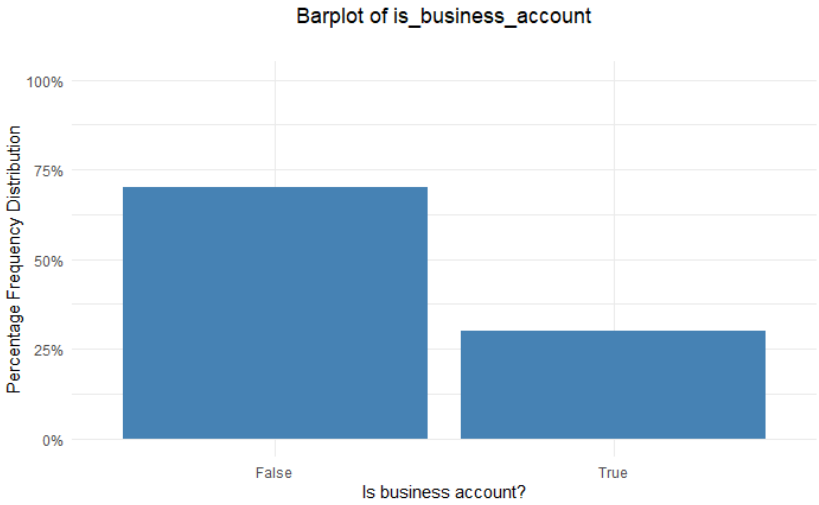

Figure 8: Bar plot of is_business_account variable

In **is_business_account** bar plot, we can see that over 60% of the influencers profiles are not business account.

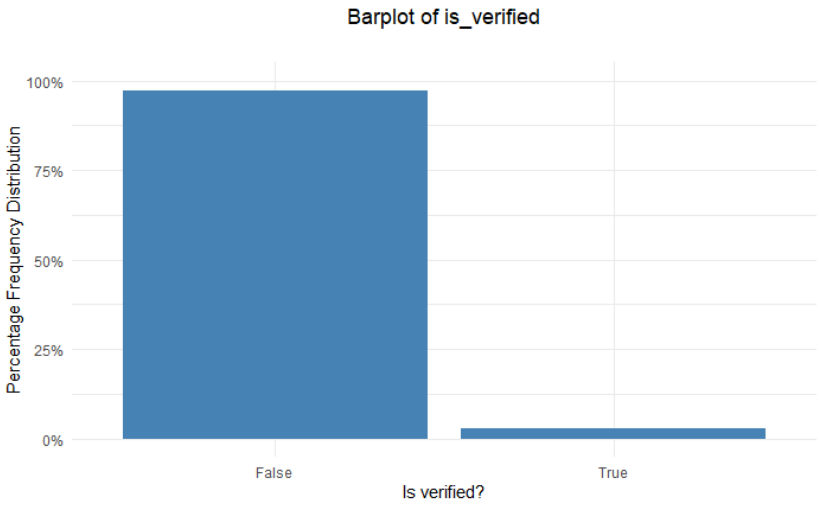

Figure 9: Bar plot of is_verified variable

On the other hand, no problems for **is_verified**, except for a certain expected unbalanced in the True-False proportion; most of the accounts are not verified.

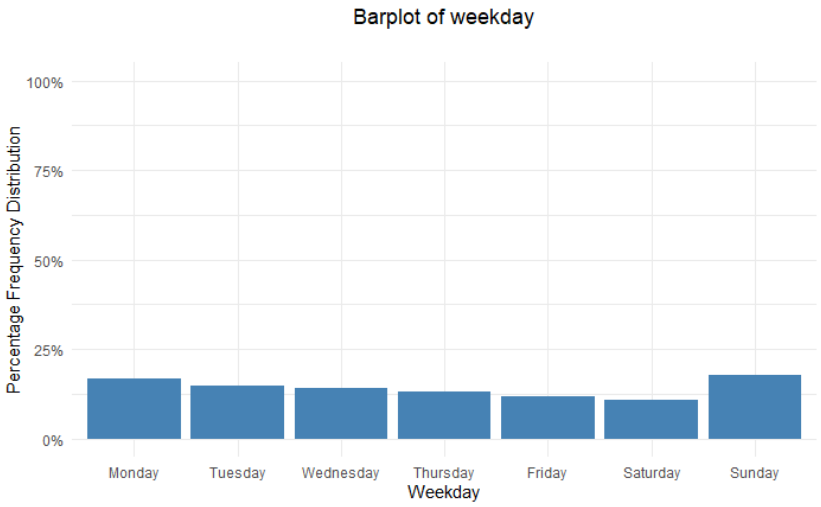

Figure 10: Bar plot of weekdays

In the **weekday** variable, we can see that the most usual weekday to post is Sunday and then it decays until Saturday, the less popular day to post. There is a certain expected balanced proportion.

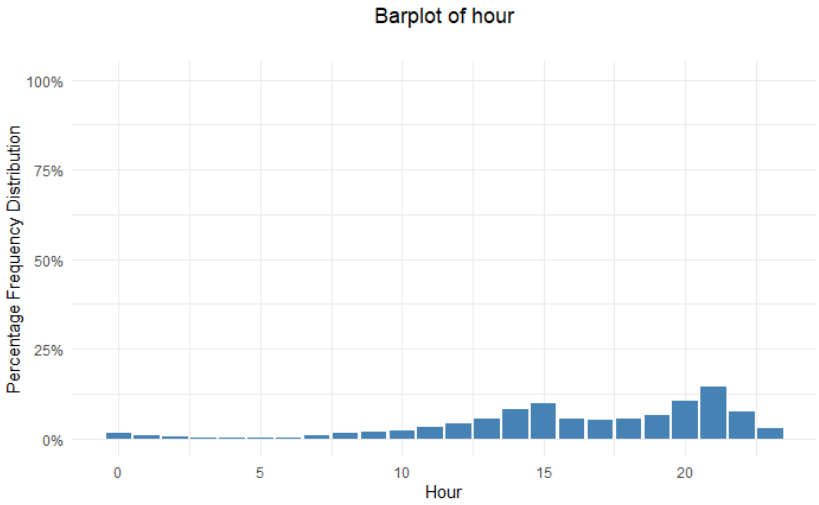

Figure 11: Bar plot of hours

In the **hour** variable, we can observe that the most common hours to post is about 14h-15h and 20h-21h hour range.

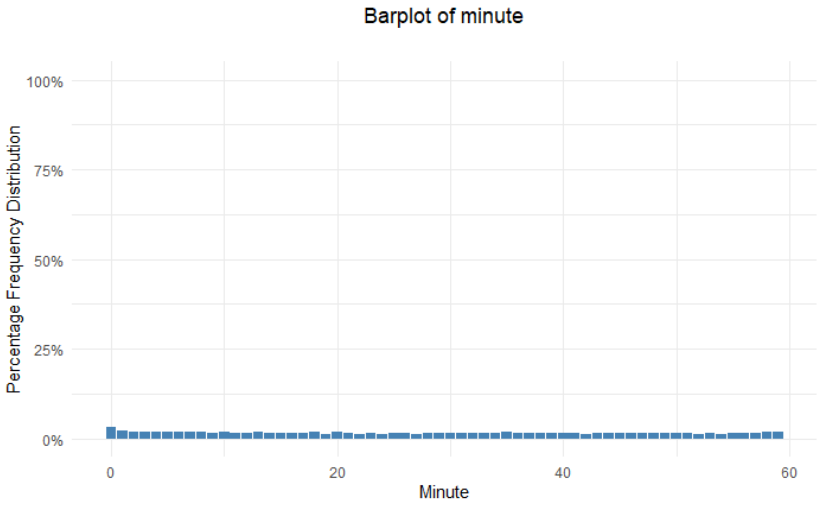

Figure 12: Bar plot of minutes

In the **minute** variable, we can observe a certain balanced proportion. Otherwise, we can observe that the most usual minute to post is 00, it is because

some users can schedule its posts at specific hours using third party tools.

4.7 Second data preparation

After performing the first exploratory data analysis, we applied all the simplification suggestions we had based on the data exploration. It is basically a data cleaning (identifying missing values), data selection and reconstructing data.

4.7.1 Data selection

Due to the huge number of outliers in the number of likes variable, we have selected all the posts that have obtained less than 5000 likes.

4.7.2 Reconstructing data

Since our mean number of likes variable for each user contains the own observation that the model will predict, we should change it. For this reason, a new average will be calculated without their own observation. Given N the number of total user posts, x_j the number of likes received for that post:

$$new\_average_j = \frac{Nx - x_j}{N - 1}$$

4.8 Second exploratory data analysis

Since the previous data preparation modified significantly the previous analysis, it is interesting to repeat it with new values of the dataset. Since, this project aims to perform number of likes prediction, the relation of this variable with the rest of variables is also studied.

4.8.1 Univariate analysis

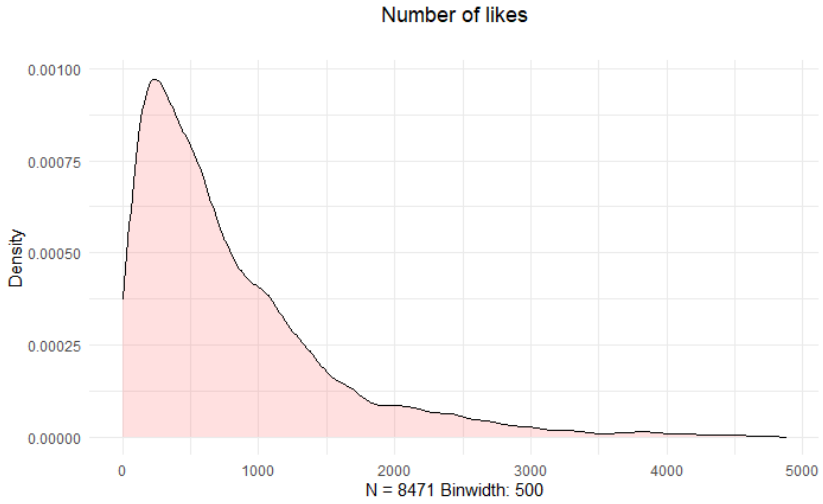

Figure 13: Bar plot of number of likes (second exploratory data analysis)

In this plot, we can see that the number of likes target is more normalized now, and so is the followers variable.

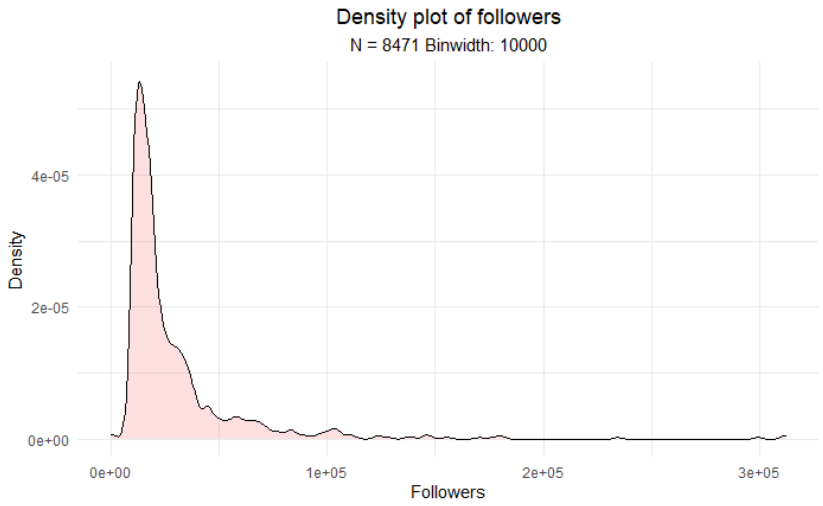

Figure 14: Bar plot of followers (second exploratory data analysis)

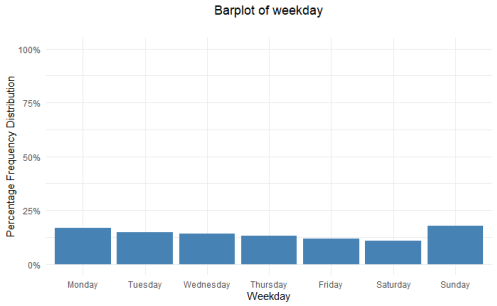

Figure 15: Bar plot of weekdays (second exploratory data analysis)

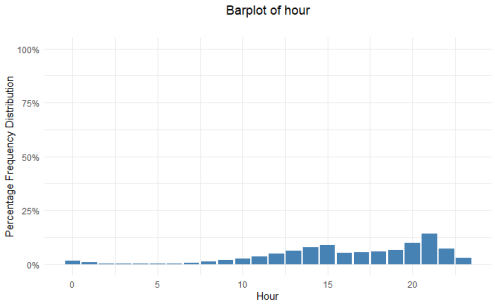

Figure 16: Bar plot of hours (second exploratory data analysis)

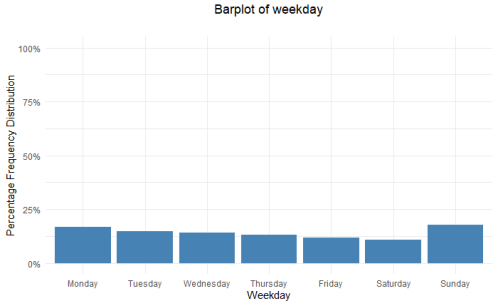

Figure 17: Bar plot of weekdays (second exploratory data analysis)

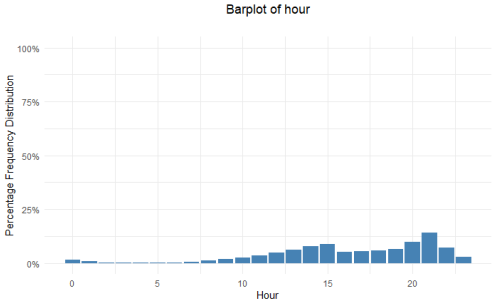

Figure 18: Bar plot of hours (second exploratory data analysis)

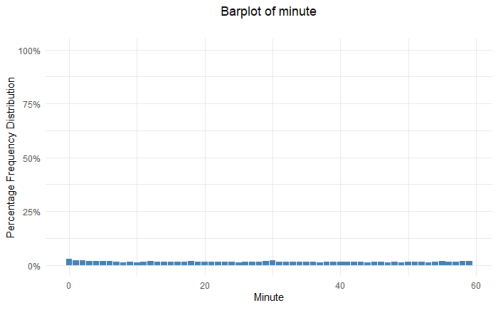

Figure 19: Bar plot of minutes (second exploratory data analysis)

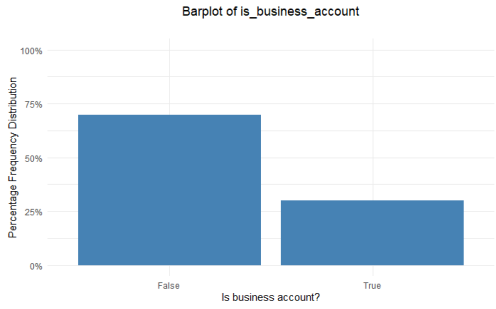

Figure 20: Bar plot of is_business_account variable (second exploratory data analysis)

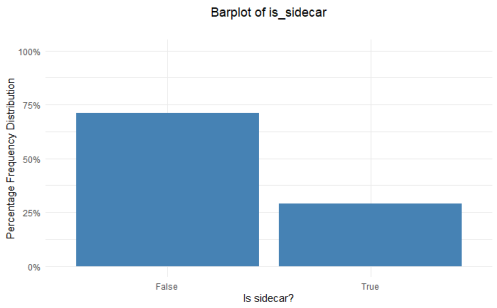

Figure 21: Bar plot of is_sidecar variable (second exploratory data analysis)

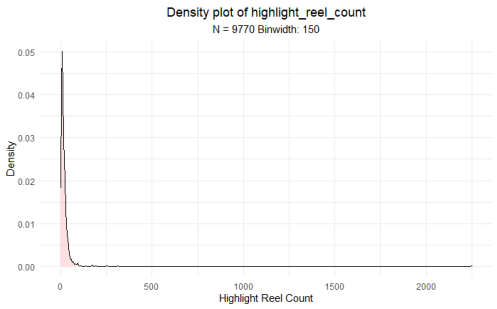

Figure 22: Density plot of highlight-reel count (second exploratory data analysis)

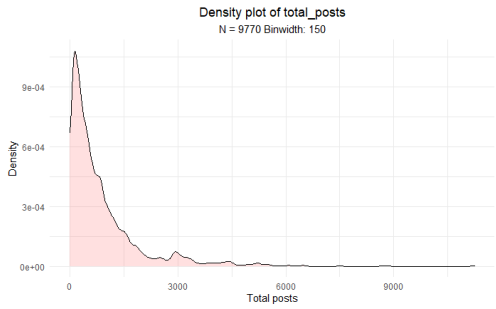

Figure 23: Density plot of total posts (second exploratory data analysis)

Also, the distribution in the weekday, hour, minute variables is almost the same even 1299 instances have been discarded.

In the other plots, we can see that the other categorical variables distribution has not changed. In addition, in numerical variables the density distribution is almost identical.

4.8.2 Bivariate analysis

For the bivariate analysis, we are not able to perform all binary combinations, since d for features there are

$$\frac{d(d-1)}{2}$$

binary combinations of features. In our case, that is 66 combinations! Therefore, only the most interesting pairs are analyzed, the ones we suspect there might be some interaction.

This bivariate analysis is oriented towards the number of likes prediction. This means that all variables have been plotted against the number of likes variable to get an insight into its relation.

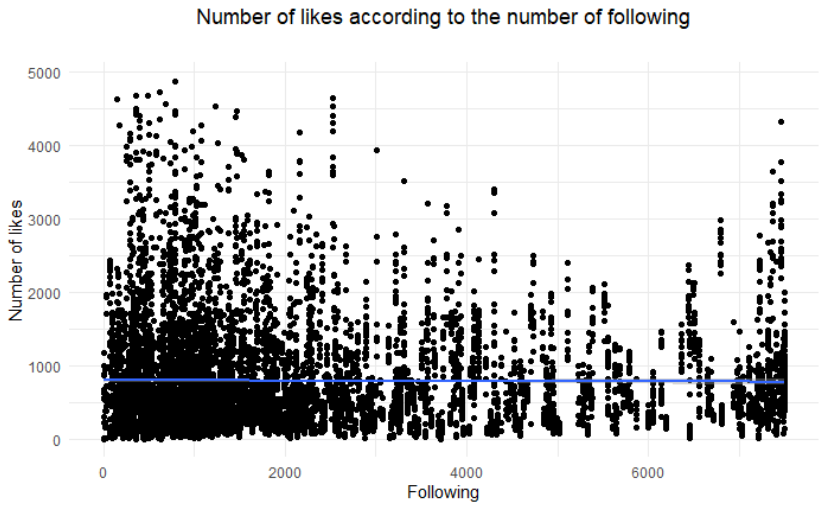

Figure 24: Number of likes according to the number of following

In this plot, we can see that the following variable does not seem a major factor in terms of the number of likes.

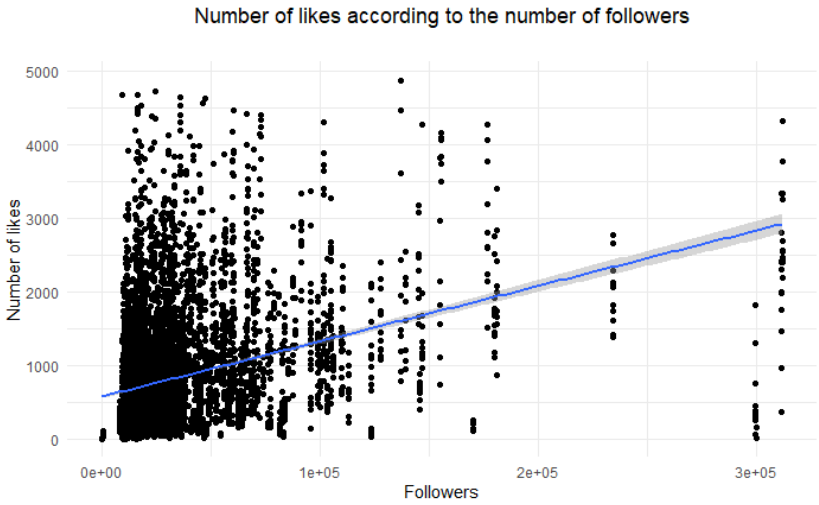

Figure 25: Number of likes according to followers

As for the followers, it seems that users with more followers are more likely to have a higher number of likes in their posts. This may be because only users who reach more people can obtain more likes.

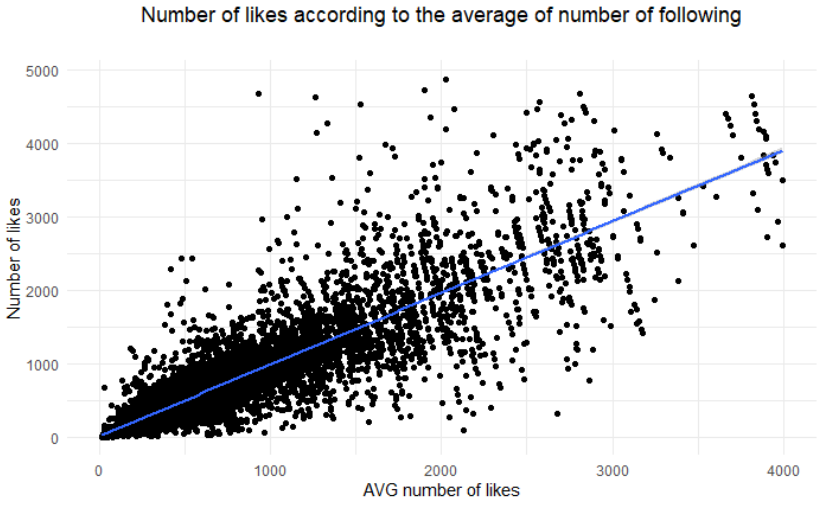

Figure 26: Number of likes according to the average number of likes

In this plot, we can clearly see that exists a linear relationship between these

two variables. For this reason, this variable will help the model to perform accurate predictions. As we have explained in the previous section, this new average is calculated without our own observation.

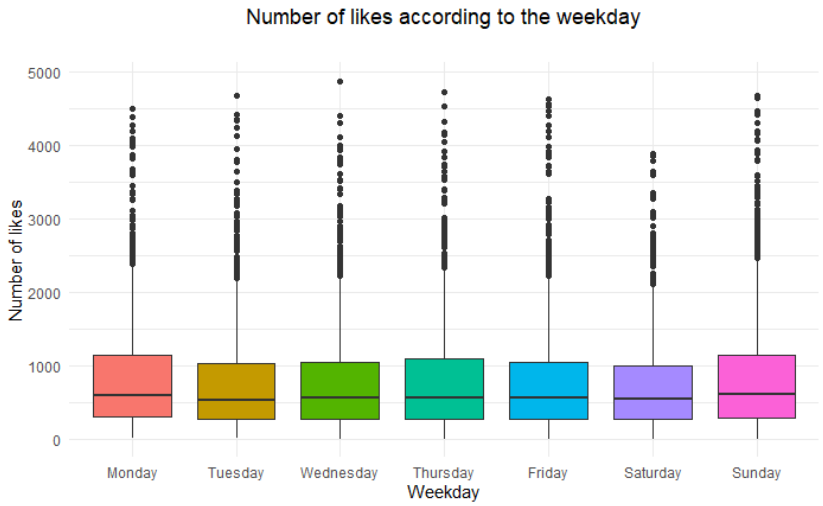

Figure 27: Number of likes according to the weekday

As for the weekday, it seems that the distribution is very balanced but we can observe that on Sundays and Mondays are slightly more likely to obtain more likes than on other days. However, the post weekday does not seem relevant.

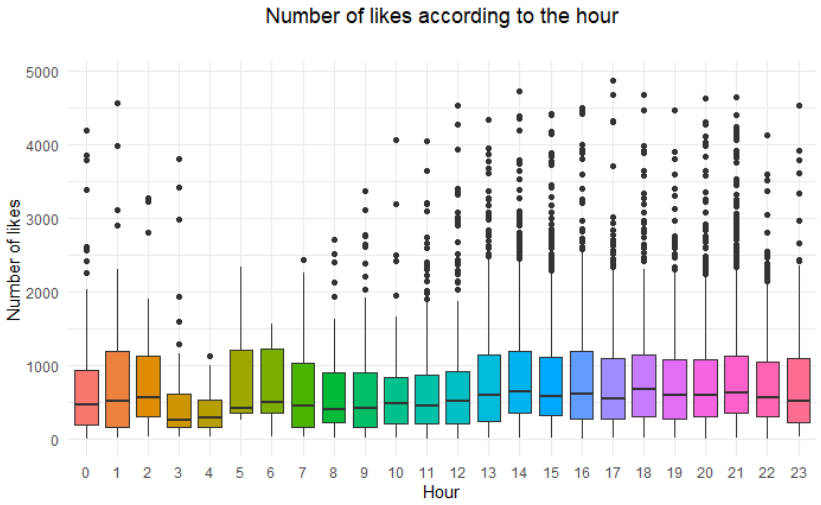

Figure 28: Number of likes according to the hour

When we look at the hour of the post, it does seem that at 14:00h and 19:00h is a good hour range to post on Instagram. This may be because these hours are when most people are online, and for this reason, the post reach is higher and consequently, the probability to obtain more likes.

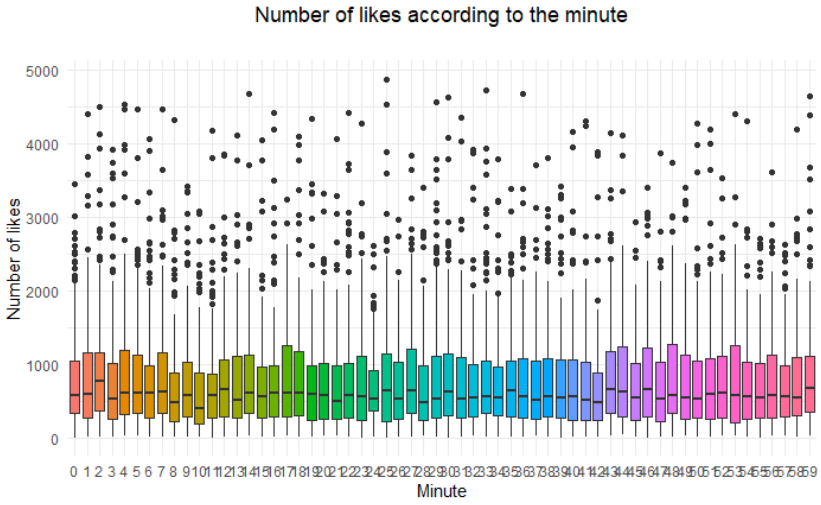

Figure 29: Number of likes according to the minute

As for the minute, it does not seem that there is any relation between post minute and the number of likes. The distribution is balanced and we cannot assert any conclusion.

Figure 30: Number of likes according to is_business_account variable

In this plot, we can see that is_business_account does not seem is correlated to our target variable even we can observe that accounts that are not business obtain a bit more likes.

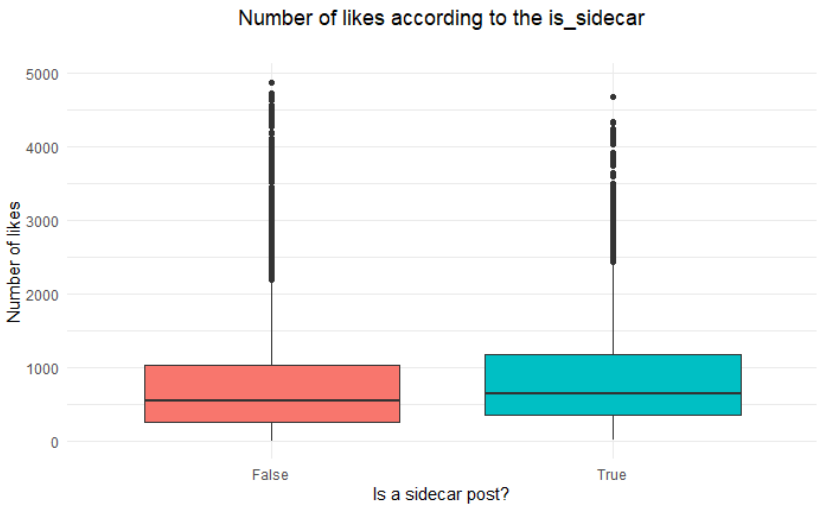

Figure 31: Number of likes according to is_sidecar variable

As for the sidecar variable, there is an insignificant difference between both means. However, it seems that users could prefer sidecar posts.

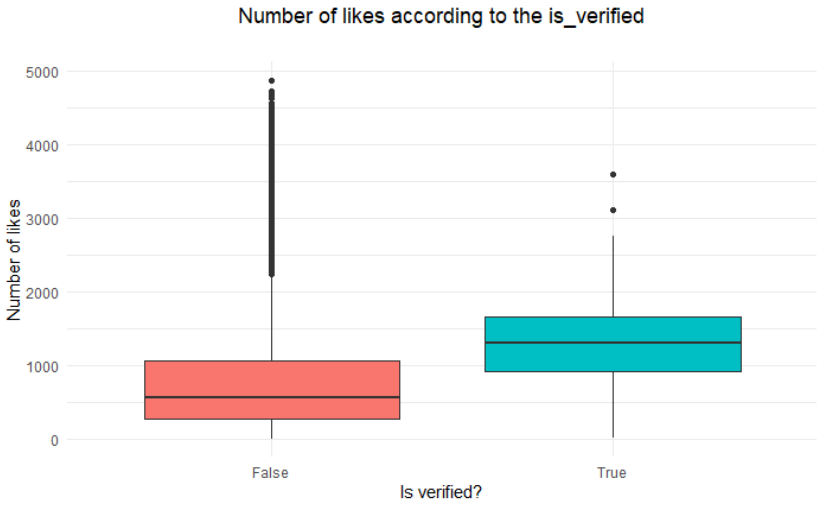

Figure 32: Number of likes according to is_verified variable

In this plot, we can observe that if an account is verified, their posts have more chance of success. As we can see, the number of likes average is signifi-

cantly higher than no verified accounts. This may be because the Instagram algorithm increases verified account visibility.

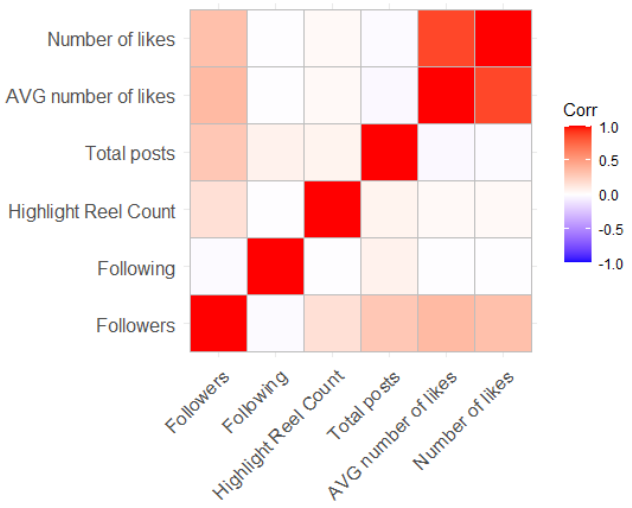

Figure 33: Correlation matrix for numerical variables

In this matrix, we can see the correlation between all numerical variables of our dataset. As we said before, for the average of number of likes, it is clear that there is a strong correlation between it and our target variable. Also, as more followers a user has, the more likely to achieve a higher number of likes. For the other numerical variables, there is not any relevant correlation between them.

4.9 Data preprocess

At this stage, the data preprocessing is performed. This includes reading prepared data, proper labeling of categorical variables, cleaning text, and purely statistical processing (standardization of numeric variables).

This is a fundamental step while building a deep learning model. If the data is quite well preprocessed the results would be confident.

4.9.1 Feature selection

Feature selection is the step of reducing the number of input features to improve the performance of the predictive model. The main focus is to choose these variables that represent the data set well, thanks to the exploratory data analysis performed we can identify irrelevant data to predict our target variable. As we have seen in the bivariate analysis, highlight_reel_count and total_posts are not correlated to the number of likes and their standard deviation is very high, so they are removed.

4.9.2 Feature engineering time

A mathematical method [30] for treating periodic features will be used to map weekday, hour and minute variables. By mapping post time (hour and minute variables) and weekday to a 2D circle that represents 24 hours we can reduce the discontinuity that occurs when the hour goes from 23 to 0 and when the day goes from Sunday to Monday.

Each cyclical variable is mapped onto a circle. The largest value for that variable appears right before to the lowest value. Both x and y components are calculated using trigonometric functions: sine and cosine.

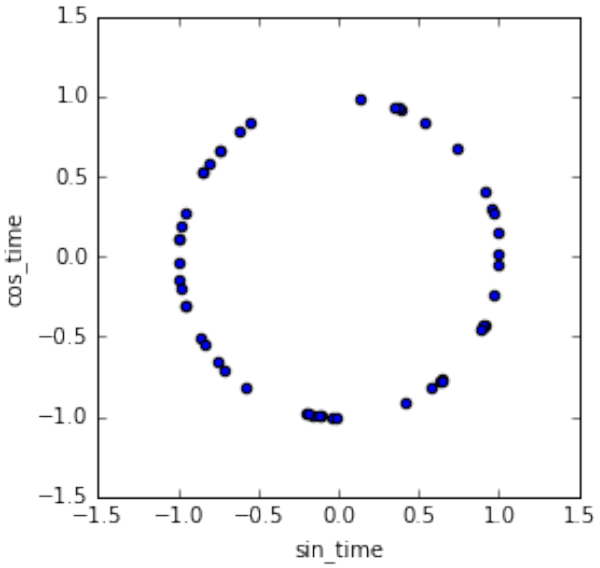

Figure 34: x_time and y_time

First of all, a post time has to be calculated by hour and minute variables:

$$post\_time_i = hour_i + \frac{minute_i}{60}$$

Then, using sin mathematical functions we can compute x component.

$$x\_time_i = sin(2 * \pi * \frac{post\_time_i}{24})$$

Finally, using cos mathematical function we can do the same for y component.

$$y\_time_i = cos(2 * \pi * \frac{post\_time_i}{24})$$

Now, we will map the weekday using the same procedure:

$$x\_weekday_i = sin(2 * \pi * \frac{weekday_i}{7})$$

$$y\_weekday_i = cos(2 * \pi * \frac{weekday_i}{7})$$

4.9.3 Normalization

In the data preprocess step, is usual to change the values of numeric columns to a common scale when features have different ranges. The numeric variables with values that have a higher range will intrinsically influence the predicted result more due to its larger value, but this does not mean it is more important as a predictor. So, we normalize the data to bring all the variables to the same range.

In this case, number_of_likes, avg_likes, followers, and following have different ranges. For that reason, numeric variables are normalized using Min-Max scaling.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Thanks to the exploratory data analysis the maximum and minimum values for each variable that we need normalize are identified:

$$avg\_likes_{norm} = \frac{avg\_likes - 0}{5000 - 0}$$

$$followers_{norm} = \frac{followers - 44}{312195 - 44}$$

$$following_{norm} = \frac{following - 2}{7502 - 2}$$

4.9.4 Split dataset

In this project, since the amount of available data is very low, we decided to split our dataset into training (and validation) and test sets.

Therefore, it was decided that each split would hold exactly 95% of the data, reducing the total dataset from 8471 data samples to two different data sets holding 112658 and 5936 data samples each.

The partition was performed randomly using the train_test_split Sci-kit learn function, and having previously set a seed of 7 to make all the results reproducible.

4.9.5 Cleaning text

This dataset contains two variables that consist of text: user biography and post caption. These steps are needed before tokenize words. Tokenization is a process in which sentences are converted into words and then transferring text from human language to machine-readable format.

We must remove all the punctuation, non-alphabets, and any other kind of characters that might not be a part of the language. The methods of text cleaning involve regular expressions that we are going to use to filter out most of the unwanted texts.

For that reason, clean_text will use regular expression and Python functions for:

- **Lowercasing**: converting all words to lower.
- **Removing unwanted characters**: removing punctuations, accent marks and other diacritics.
- **Remove whitespaces**: removing additional whitespaces between words and ending/leading spaces.

4.9.6 Building corpus

After text cleaning preprocess, the next step is to build a corpus. To compute word vectors, a large text corpus file is needed. It consists of a text file containing all the sentences we can obtain in our dataset, specifically, user biography and post caption (after cleaned).

Depending on the corpus, the word vectors will capture different information. So for that reason, we are going to train our fastText model using a corpus built for our project purpose.

4.9.7 Training fastText model

Once we have built our corpus file, the fastText model is trained.

As mentioned in a previous section, we will use cbow ('continuous-bag-of-words') model for computing word representations predicting the target word according to its context. On the other hand, the skip-gram model learns to predict a target word just by seeing a nearby word.

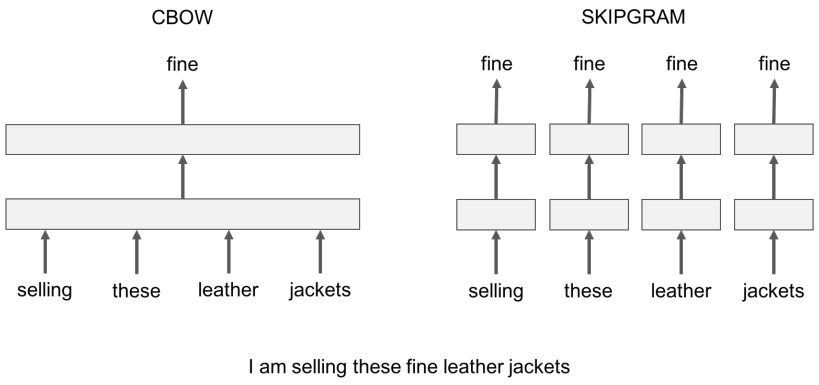

Figure 35: Continuous bag of words example

Given the sentence 'I am selling these fine leather jackets' and the target word 'fine', a skip-gram model tries to predict the target using a random close-by word, like 'these' or 'leather'. The cbow model takes all the words in a surrounding window, like selling, these, leather, jackets

These arguments are passed in the function parameters:

- **Input path (input)**: text file path that was built before.
- **Model (model)**: unsupervised fastText model, cbow or skipgram. cbow is the argument value passed to the parameter.
- **Learning rate (lr)**: tuning parameter that determines the step size at each iteration while moving toward a minimum of the loss function. 0,05 is the argument value passed to the parameter.
- **Epochs (epoch)**: number of epochs. An epoch is one complete pass through the corpus file. 100 is the argument value passed to the parameter.
- **Dimension (dim)**: the size of word vectors. 50 is the argument value passed to the parameter.
- **Thread (thread)**: number of threads to train the model. 32 is the argument value passed to the parameter.

4.10 Modeling

As we decided to use neural networks to solve our problem, a model architecture is needed. Because our problem has mixed data (variables, text and image), our model must be able to use all of our data. For this reason, a three-branch neural network will be used.

4.10.1 Neural network model architecture

First of all, we need to define three different model architectures to process each type of data. After this, the three branches will be concatenated to make a final prediction, adding some layers to this and finally a linear activation output layer which is the number of likes prediction.

This following figure shows the model architecture proposed:

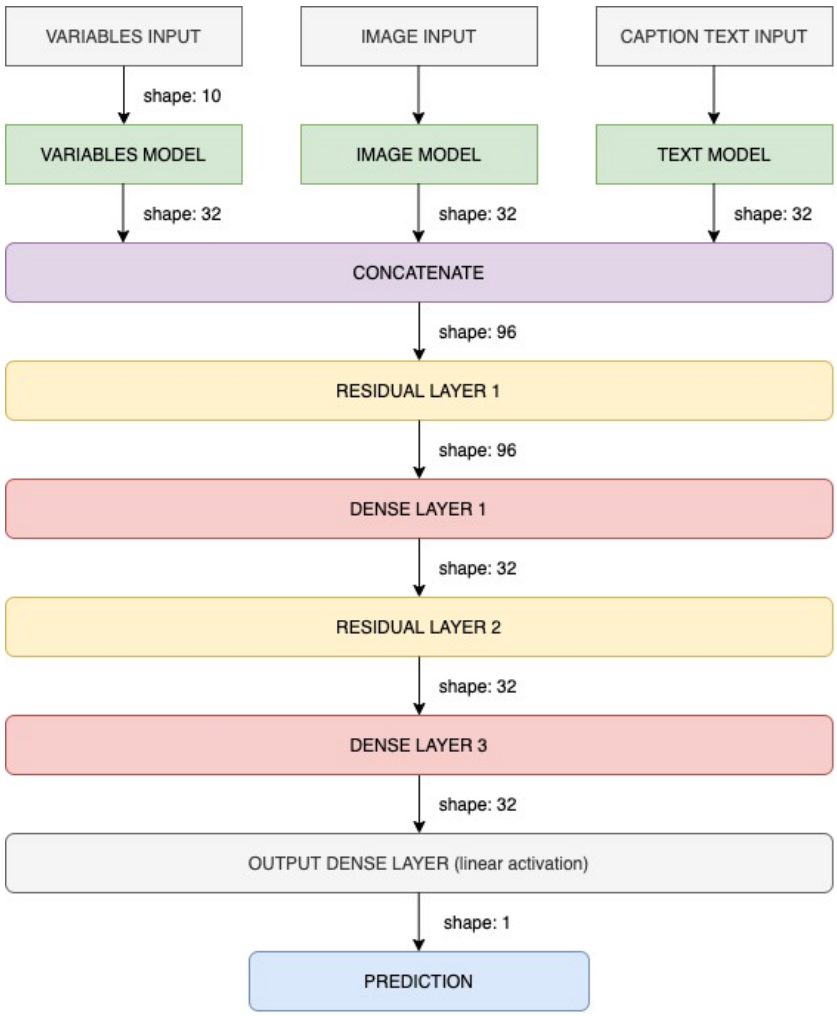

Figure 36: Model architecture.

One of the problems of the neural networks that have multiple layers, such as, in our case, is vanishing gradient [31]. When a larger neural network is trained through gradient descent, the calculated partial derivatives used to compute the gradient go deeper into the network. Since the gradients control how much the network learns during training, the gradients will be vanishingly small or zero, preventing the weight from changing its value, leading to poor predictive performance. In the worst case, it could completely stop the model training.

To solve this problem, we will use three methods to overcome the vanishing gradient:

- **Multi-level hierarchy**: as we mentioned before, a three-branch neural network will be used. As Jürgen Schmidhuber states in *Learning complex, extended sequences using the principle of history compression* [32], this technique pretrains one branch at a time, and then performs backpropagation for fine-tuning.
- **Residual layers**: this technique introduces bypass connections, also known as skip connections or shortcuts, that connect layers further behind the previous layer to a given layer. Using residual layers on our model allows gradients to propagate quicker to deep layers before they can be attenuated to zero or small values reusing activations from a previous layer until the adjacent layer learns its weights. [33].

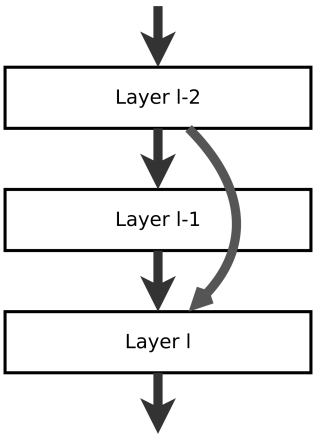

Figure 37: The canonical form of a residual neural network. A layer $\ell 1$ is skipped over activation from $\ell 2$ [33].

As we can see in the model architecture, the combined outputs of the three branches (an encoding for variables, image and text) are the input to the concatenate layer. Then, the embedding concatenated is connected to the final layers. It goes from 64 neurons to 32, 16 and finally a linear activation on our final neuron which is the predicted number of likes.

In the variables branch, all numerical and categorical variables are the input to the model: followers, following, is_verified, is_business_account, is_sidecar,

x_time, y_time, x_weekday and y_weekday. These variables have been properly preprocessed, as mentioned in the previous section, so now the model algorithm can interpret them. The resulting architecture is shown below:

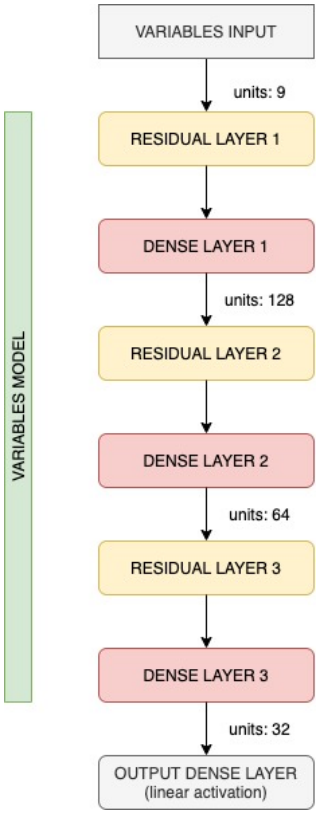

Figure 38: Variables model architecture.

These inputs are connected to some residual and dense layers. These dense layers transform the number of nodes and it goes from 128 to 64, 32, and finally a linear activation on our final neuron which is the predicted number of likes.

On image and text branches, an alternative target variable is introduced. As we mentioned in the exploratory data analysis section, followers and avg_number_of_likes features are very correlated to our target number of likes. The problem is that image and text branches do not have these inputs in their models and they can not learn about the social context related to

an image or text. For this reason, these models will predict an alternative variable that tries to exclude social context information: engagement rate.

Engagement rate is the amount of total engagement received by an Instagram account over a specific post, expressed as a percentage of followers. Engagement is defined as anytime another user likes on one of your posts. Given a post i:

$$engagement\_rate_i = 100 \cdot \frac{number\_of\_likes_i}{followers_i}$$

In text branch, the post caption, and the biography description are the input to the model. These variables are preprocessed and then converted to embedding by fastText model, as mentioned in the data preprocess section.

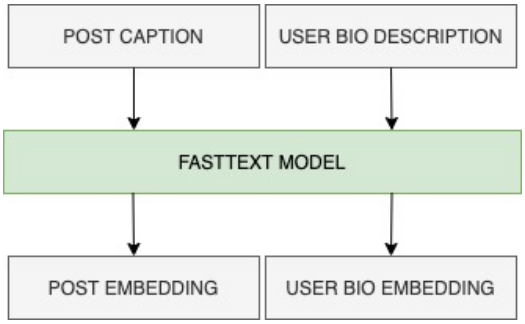

Figure 39: fastText model architecture

Both variables are represented as a 50-dimensional dense vector using *ftencoder.encode* function and then are concatenated to convert it to a single embedding. Then, the resulting architecture for the text branch is shown below:

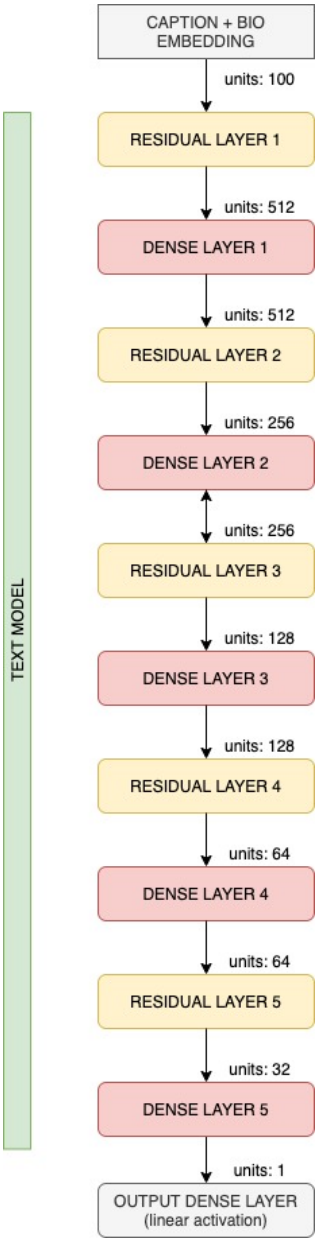

Figure 40: Text model branch architecture

The concatenated embedding is connected to some residual and dense layers. These dense layers transform the number of nodes and it goes from 512, 256,

128, 64, 32, and finally a linear activation on our final neuron which is the predicted engagement ratio.

In the image branch, the post image is the input to the model. Thanks to ImageDataGenerator class from Keras, it generates batches of tensor image data with real-time data-augmentation. The rescale parameter is used to scale the array of original image pixel values to be between [0, 1].

Data augmentation is also performed by applying some image transformation techniques. It is used to expand the training samples in order to improve the performance of the model to generalize and to reduce overfitting [34].

- **horizontal_flip**: randomly flip images horizontally.
- **rotation_range**: randomly rotates the image by a given value of degrees from 0 to 360. 45 is the argument value passed to the parameter.
- **width_shift_range and height_shift_range**: randomly shifts the image horizontally or vertically. It means moving all pixels of the image while keeping the image dimensions the same. 0,2 is the argument value passed to the width and height parameters.

Figure 41: Output images after applying data augmentation techniques

Then, as our image paths are in a Pandas dataframe, the flow_from_dataframe function from Keras is used to take the dataframe and the path to a directory and then, generates batches. The target_size parameter is used to resize dimension image to 100x100px.

The CNN architecture is based on the original ResNet paper: *Deep Residual Learning for Image Recognition* [35] The next figure is what the ResNet model architecture looks like:

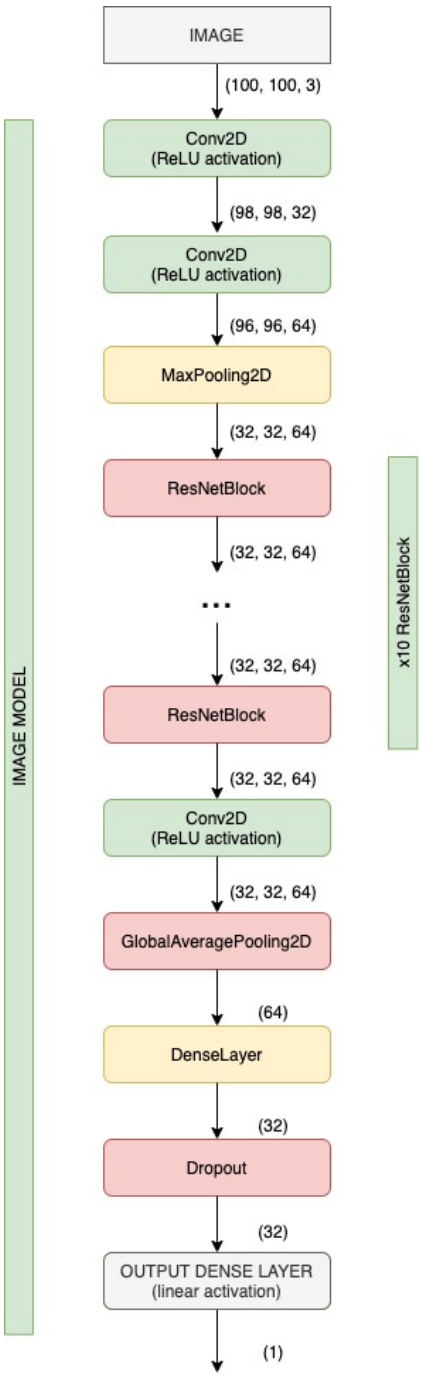

Figure 42: ResNet block architecture

First, we specify the input dimensions to Keras. The preprocessed images have a size of (100, 100, 3). Next, 2 standard CNN layers, Conv2D, are created with 32 and 64 filters respectively. The filter window sizes are 3x3, in line with the original ResNet architectures mentioned previously. Then, some MaxPooling2D is performed, and then it is time to build some ResNet building blocks. In our case, 10 ResNet blocks are created. The resulting architecture for the ResNet block is shown below:

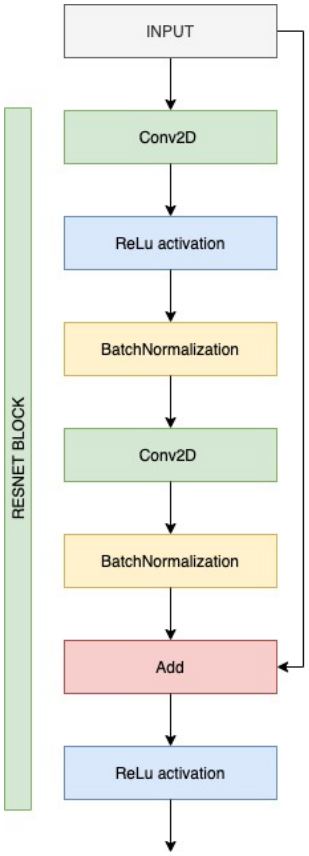

Figure 43: ResNet block architecture

The first layers of ResNet blocks are Conv2D with Batch Normalization CNN layers. The second Conv does not have an activation function because one will be applied after the residual addition part of the block). Then, the Keras Add layer is used (which simply adds two tensors together). The input data is

added to the CNN output. In the end, a ReLU activation is connected to the result of this addition and the output is the returned value of the function.

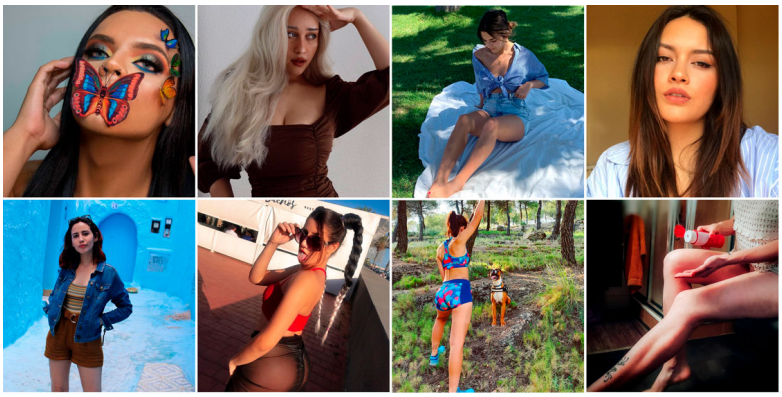

Figure 44: Random image subset of high predicted engagement rate

Figure 45: Random image subset of low predicted engagement rate

Because these three models (variable, text and image) will be concatenated, the final layer of each text, image and variable model is removed. The regression is computed at the end by these three model branches: concatenating them, including some residual and dense layers, and a final output dense layer (with linear activation). The hybrid model architecture is shown below:

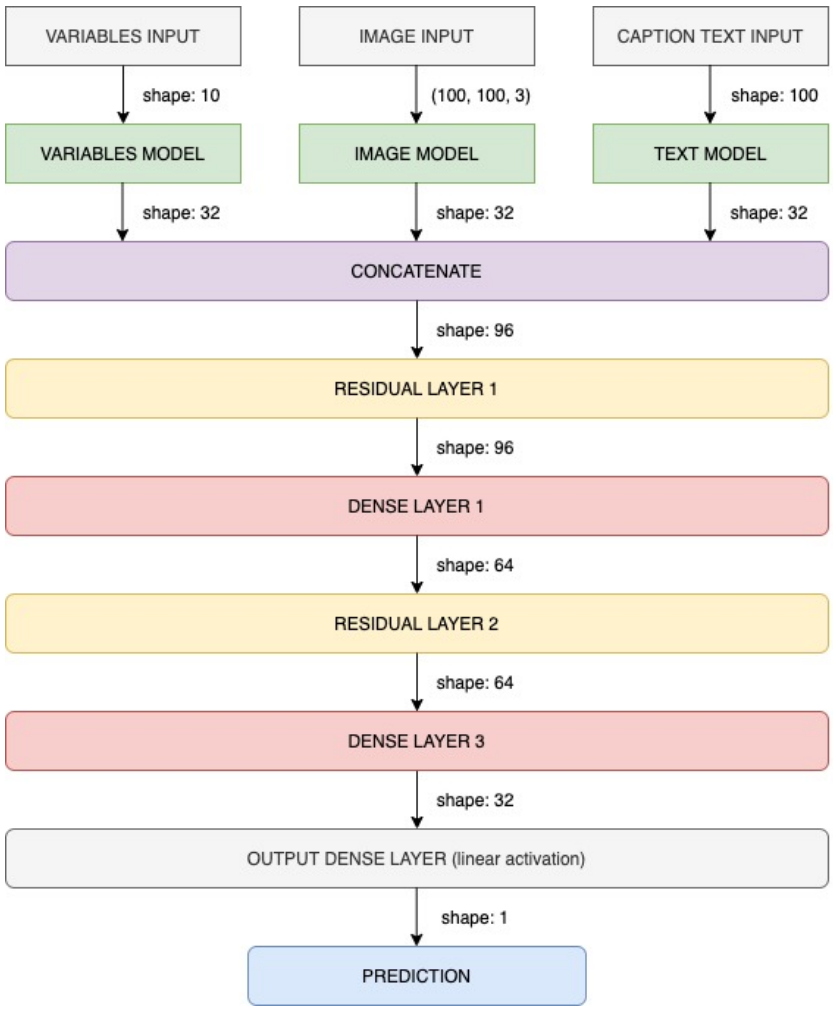

Figure 46: Hybrid model architecture

Thanks to Keras, we can handle three different inputs. We already had the three models that we mentioned above so the next step was to combine them into one single neural network. Each model branch is tasked to train on each type of input data.

4.10.2 Optimizer

To train our neural network model, Adam optimizer is used. Adam is a stochastic gradient descent optimization algorithm that provides more effi-

cient neural network weights by running repeated cycles of "adaptive moment estimation. It is most effective in larger datasets by keeping the gradients more compacted over many learning iterations.

According to Kingma et al. [36], the method is "computationally efficient, has little memory requirement, invariant to a diagonal rescaling of gradients and is well suited for problems that are large in terms of data/parameters".

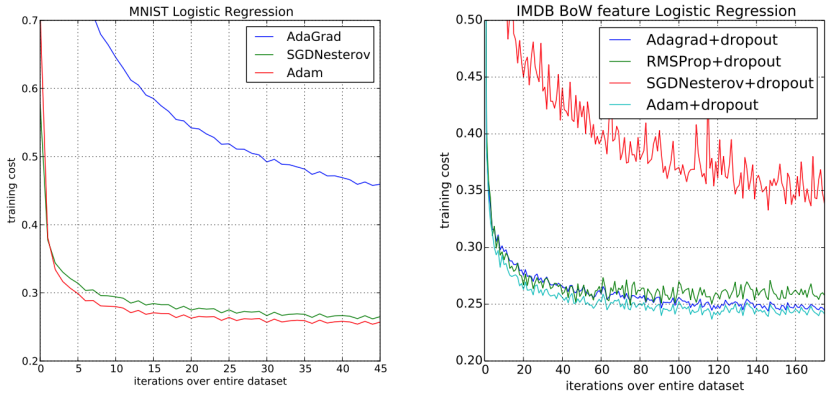

Figure 47: Logistic regression training negative log-likelihood on MNIST images and IMDB movie reviews with 10,000 bag-of-words (BoW) feature vectors. [36]

Adam combines the advantages of two other stochastic gradient techniques, Adaptive Gradients and Root Mean Square Propagation, to create a new learning approach to optimize a variety of neural networks.

4.10.3 Loss function

As our project is a regression problem, the loss function will be mean squared error. It is a regression problem because the output variable is a real or continuous value, such as the number of likes.

Given n the number of samples of the data set, Y the vector of observed values of the number of likes being predicted and $\hat{Y}$ being the predicted values, the mean squared error (MSE) is:

$$MSE = (\frac{1}{n}) \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i \right)$$

MSE measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value [37].

Our goal is to minimize this value, which will provide us with the best regression line that goes through to the set of points.

4.10.4 Metric

The metric refers to the predictive accuracy of the model. In this case, as we predict a number, we should use a function used for regression models. Mean absolute error (MAE) is chosen. MAE is the sum of absolute differences between predicted and observed variables. The mean absolute error uses the same scale as the data being measured, in this case, the number_of_likes variable scale.

Given n the number of samples of the data set, Y the vector of observed values of the number of likes being predicted and being the predicted values, the mean absolute error (MAE) is:

$$MAE = \frac{\sum_{i=1}^{n} y_i - x_i}{n}$$

This performance metric is also used to compare multiple models if it is analyzed in the same data set. As we mentioned before, test data is built, holding 5936 data samples.

Root mean squared error (RMSE) is also one of the most common metrics used to measure accuracy for regression problems:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y_i - x_i)^2}{n}}$$

Both metrics express average model prediction error in units of the prediction variable and are indifferent to the direction of errors (positive or negative

errors). However, in our case, we have chosen for MAE for these reasons [38]:

- **RMSE increases with the variance of the frequency distribution of error magnitudes**: as the variance of our target variable is large, RMSE gives a will give a high weight to large errors. RMSE should be more useful when large errors are particularly undesirable, but this is not our case.
- **RMSE tends to be larger than MAE as the test sample size increases**: this would be a problem when we compare other RMSE results calculated on different sized test samples on other models.
- **More difficult to understand**: RMSE does not describe average error alone because the errors are squared before they are averaged. RMSE by itself does not tell whether our predictions are good.

4.10.5 Training models

After introducing all model architectures in the previous section, the three model branches (text, image and variables model) and the hybrid one is trained. While the models are training, TensorBoard is used to tracking Mean Absolute Error and Root Mean Squared Error metrics through every epoch. TensorBoard is a tool for providing the measurements and visualizations needed during the machine learning workflow. It enables tracking experiment metrics like loss and accuracy, visualizing the model graph and projecting embeddings to a lower dimensional space. [39]

First, the variable model branch is trained:

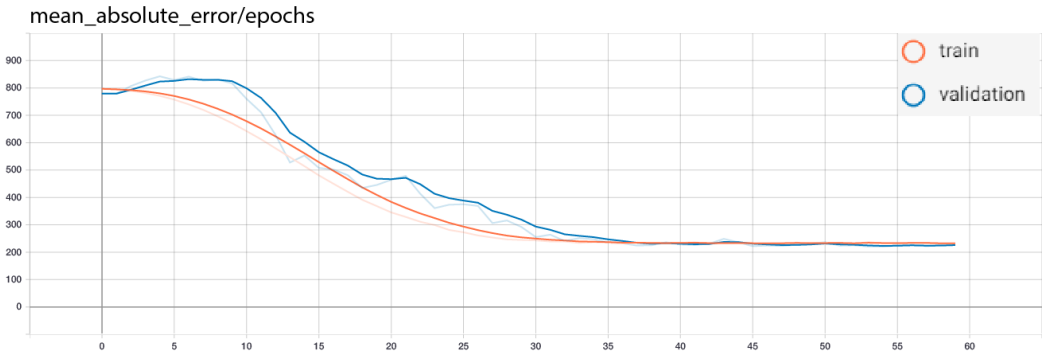

Figure 48: Variables model training. Mean Absolute Error along with the number of epochs.

In the previous figure, we can see Mean Absolute Errors values along with the number of epochs. An epoch is one learning cycle where the trainer sees the whole training data set. The variables model trains 35 epochs until the mean absolute error metric converges, reaching the threshold. We obtain a mean absolute error of 231,69 in the train set (RMSE of 377,79) and 232,64 in the test set (RMSE of 354,76).

Table 5: Variables model metrics

Set	Loss	Mean Absolute Error	Root Mean Squared Error
Train	141978	231.69	377.79
Validation	142128	242.37	381.92
Test	125861.60	232.64	354.76

Taken 8 random observations from the test split and their predictions, the absolute error between the predicted value and the actual value is calculated.

Table 6: Variables model: actual number of likes vs the predicted number of likes

ID	Average #likes	Actual #likes	Predicted #likes	Absolute Error
1	1230.11	1156	1158.33	2.33
2	915.81	890	903.24	13.24
3	306.30	314	409.81	95.81
4	370.33	451	476.43	25.43
5	808.81	773	779.43	6.43
6	216.40	218	341.22	123.22
7	915.81	890	903.24	13.24
8	1248.27	935	1228.23	293.23

As we can observe, there are really accurate predictions, for example, number 1 or 5, but some bad ones like 6 or 8.

Next, the image model branch is trained:

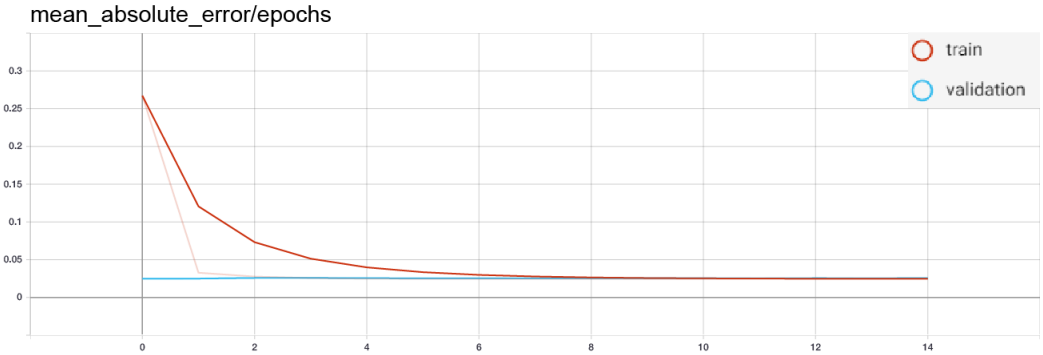

Figure 49: Image model training. Mean Absolute Error along with the number of epochs.

The image model trains 8 epochs until the mean absolute error metric converges, reaching the threshold. We obtain a mean absolute error of 0,0247 in the train set (RMSE of 0,03557) and 0,0257 in the validation set (RMSE of 0,03461). The engagement rate is the prediction target of the image model. It means that the mean absolute error of the engagement rate is about 2,5%, a really interesting result.

Table 7: Image model metrics

Set	Loss	Mean Absolute Error	Root Mean Squared Error
Train	0,001265	0,0247	0,03557
Validation	0,001976	0,0257	0,03461
Test	0,0014	0,0251	0,0383

As we did before, 8 random observations are taken from the test split and their predictions, the absolute error between the predicted engagement rate value and the actual engagement rate value is calculated.

Table 8: Image model: actual engagement rate vs predicted engagement rate

ID	Actual engagement rate	Predicted engagement rate	Absolute Error
1	3,31%	3,53%	0,21%
2	1,78%	3,53%	1,76%
3	3,56%	3,44%	0,11%
4	1,41%	3,46%	2,05%
5	0,73%	3,47%	2,75%
6	6,20%	3,52%	2,67%
7	3,57%	3,52%	0,04%
8	1,49%	3,52%	2,03%

Next, the text model branch is trained:

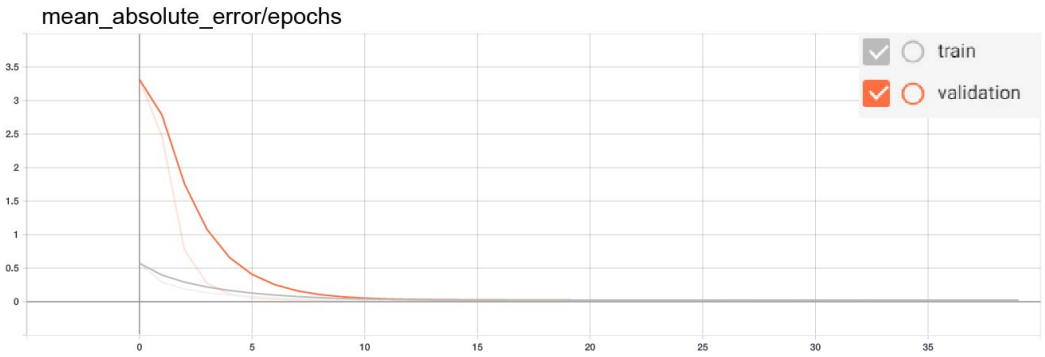

Figure 50: Text model training. Mean Absolute Error along the number of epochs.

The text model trains 12 epochs until the mean absolute error metric converges, reaching the threshold. A mean absolute error of 0,0244 is obtained

in the train set (RMSE of 0,0345) and 0,0245 in the validation set (RMSE of 0,0350). The engagement rate is the prediction target of the text model.

Table 9: Text model metrics

Set	Loss	Mean Absolute Error	Root Mean Squared Error
Train	0,001105	0,0244	0,0345
Validation	0,001251	0,0245	0,0350
Test	0,001231	0,0246	0,0362

As we can observe, the text model works slightly better than the image model predicting engagement rate.

Table 10: Text model: actual engagement rate vs Predicted engagement rate

ID	Actual rate	Predicted engagement rate	Absolute Error
1	3,31%	3,42%	0,21%
2	1,78%	2,53%	1,76%
3	3,56%	3,58%	0,11%
4	1,41%	1,85%	2,05%
5	0,73%	2,14%	2,75%
6	6,20%	4,85%	2,67%
7	3,57%	2,92%	0,04%
8	1,49%	2,52%	2,03%

Taking the same 8 random observations that we have seen testing the image model, we can observe slightly better mean absolute errors in most cases.

Finally, the hybrid model is trained:

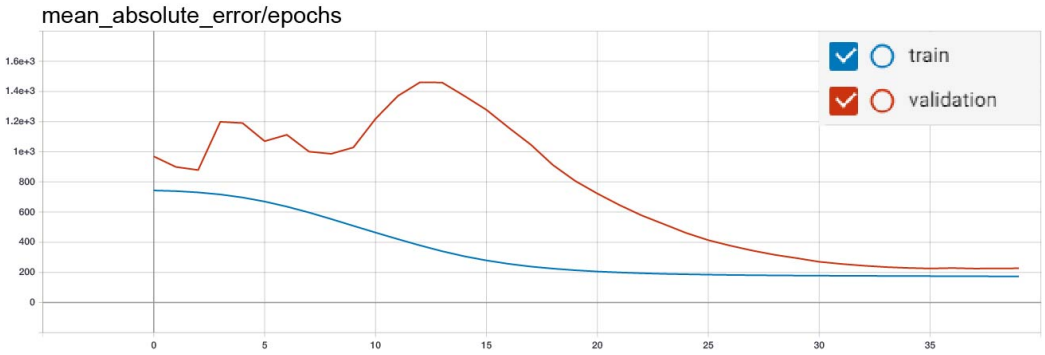

Figure 51: Hybrid model training. Mean Absolute Error along the number of epochs.

The hybrid model trains 35 epochs until the mean absolute error metric converges, reaching the threshold. A mean absolute error of 190,34 is obtained in the train set (RMSE of 298,39) and 209,34 in the validation set (RMSE of 352,42). In this case, the number of likes is the prediction target of the hybrid model, as in the variables model branch.

Table 11: Hybrid model metrics

Set	Loss	Mean Absolute Error	Root Mean Squared Error
Train	98963,83	190,34	298,39
Validation	100551,60	209,77	325,92
Test	102861,60	210,75	327,89

As we can observe, the hybrid model works better than the variables model predicting the number of likes.

Table 12: Hybrid model: actual number of likes vs the predicted number of likes

ID	Average #likes	Actual #likes	Predicted #likes	Absolute Error
1	1230,11	1156	1159,20	3,20
2	915,81	890	900,47	10,47
3	306,30	314	384,58	70,58
4	370,33	451	470,73	19,73
5	808,81	773	778,58	5,58
6	216,40	218	323,53	105,53
7	915,81	890	904,75	14,75
8	1248,27	935	1223,75	288,75

Taking the same 8 random observations that we have seen testing the variables model, we can observe that the hybrid model performance is better than the variables model performance and its MAE is also lower (210,75 against 232,32).

5 Evaluation

Evaluating machine learning model performance is very relative. It depends on the amount of data we have and the algorithms or tools we are using. In this case, as we are working on a regression problem, achieving 0,0 error is impossible because all predictive models contain errors and a perfect score is impossible in practice given the stochastic nature of the domain.

Otherwise, we can compare the performance of various models trained on the same data. For example, the performance of the hybrid model is better than the variables model because both are trained on the same train set and both predict the same target: number of likes. Also, the text model predicts a better engagement rate than the image model in our train set.

For this reason, the baseline model performance can be used as the standard in which we can compare other models and be evaluated. The baseline model is the result of a very basic solution for our problem, the simplest possible results. The skill of this model to predict our target provides the base for the lowest performance of a machine learning model on our specific dataset.

When it is a regression problem, the most common baseline model is when it predicts the mean or the median value of our target. On the other hand, predicting the mode outcome value can be an example of a baseline model for a classification problem.

For this project, the mean value baseline model is chosen. The average number of likes in the test set is the result of all predictions. As we have used the mean absolute error for the accuracy score, we will use it as well to calculate the error of the baseline model.

Mean Absolute Error for the baseline model is 557,78.

As we can observe, the baseline is a really bad result. It may indicate a particular difficulty with the project problem. On the other hand, if our model achieves a performance below that this result, indicates that our model is not appropriate for the problem.

Table 13: Comparing results with the baseline model

Model	Mean Absolute Error
Baseline model	557,78
Hybrid model	210,75
Difference	**347,03**

We know that our model is good when the performance falls within a range between the best possible score (0,0) and the baseline. As we can observe, our model performance is 347,03 lowers than the baseline.

Now, we have explored the space of possible models and, in general, the obtained results can be considered to be quite good. In the future, it would be possible to find another model that its performance falls within a range between hybrid model performance (210,75) and 0.

6 Conclusions

After performing all the proposed tasks over the dataset, I could obtain very interesting information about the number of likes prediction in Instagram posts. First, we could access to Instagram post data through a web scraping python script. After, we have been able to analyze individually all its features and perform a significant exploratory data analysis.

To test further insights, the dataset was properly divided into a train, validation and a test partitions. The train partition was used to perform the exploration and the modeling, whereas the validation and test dataset was used to validate the knowledge extracted from that modeling.

Finally, a model was trained to perform the proposed task. The chosen model is an hybrid model, because it works particularly well with the image, numerical and text data and works with the amount of samples in the dataset. The trained model achieved good on the test dataset, given the difficulty of the task and the amount of data available.

Also, three is better than one; the idea of combining variable, image and text models has been a good choice. Our accuracy has definitely increased.

The quality of a deep learning model is constrained by the quality of the dataset. In this project, 8471 data samples were available. It is not considered a larger dataset to train a deep learning neural network. So, if I could have access to more samples through the Instagram API, I could create a much robust model. Machine learning techniques get better with more data, deep learning especially.

Therefore, all the required tasks have been performed successfully, providing me a lot of information and insights from the dataset. It has also been very interesting to work with a dataset so interpretable, because I have learned to work with a more real-world oriented project. It has helped me a lot in understanding the techniques and algorithms used in deep learning, natural language processing and convolutional neural networks and how to take them into practice. It was a very interesting project I enjoyed a lot and in which I learned many interesting concepts I will surely be using in my future professional career.

6.1 Future work

As I mentioned before, In the future, I would like to get access to the Instagram API to obtain many Instagram post data in batch since a scrappy based web python script is not a model for scalability. It would mitigate the majority of problems currently present in the process. Nowadays, we can only access public accounts due to privacy. With the right amount of data, this hybrid model could predict Instagram likes through every kind of user, not only beauty and fashion account categories.

During the project development, an interesting new was published: *Instagram removing likes*[40]. Instagram is testing to hide the number of likes on posts in several countries, including Australia and Japan. Likes counters will be hidden, so users can still see how many likes an own post obtained, but no

one else can. Adam Mosseri, Instagram's CEO, has said that removing likes was the platform's way to "depressurize" Instagram for young people. Data science teams at Facebook (which owns Instagram), believe that hiding likes might motivate people to post more frequently. They also think that it might extend the length of time that people spend on the app.

It was interesting to notice that maybe someday this project will not be reproducible due likes counter privacy. Then, it will not be to predict the number of likes. The number of comments could be the alternative variable to measure the popularity of a post.

7 List of References

References

[1] Instagram. Official Instagram Blog. https://about.instagram.com/about-us

[2] Most used social media platforms. Statista. https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

[3] Instagram Business: Marketing on Instagram. https://business.instagram.com/?locale=en_EN

[4] Applied Project Engineering. https://www.fib.upc.edu/en/studies/bachelors-degrees/bachelor-degree-informatics-engineering/curriculum/syllabus/PAE

[5] TensorFlow. https://www.tensorflow.org/

[6] Seguros Catalana Occidente. https://www.seguroscatalanaoccidente.com/

[7] Ranked: The Most Valuable Brands in the World. https://www.cisualcapitalist.com/ranked-the-most-valuable-brands-in-the-world/

[8] Khosla, Aditya & Sarma, Atish & Hamid, Raffay. (2014). What makes an image popular?

[9] Crystal J. Qian, Jonathan D. Tang, Matthew A. Penza, Christopher M. Ferri. Instagram Popularity Prediction via Neural Networks and Regression Analysis

[10] LikelyAI webpage
https://www.likelyai.com/

[11] Beautiful Destinations webpage
https://beautifuldestinations.com/

[12] Forbes. This Company Can Predict The Number Of Likes An Instagram Photo Will Get
https://www.forbes.com/sites/alysonkrueger/2016/07/21/this-company-can-predict-the-number-of-likes-an-instagram-photo-will-get/#2260c0195a47

[13] AI for everyone.
https://www.coursera.org/learn/ai-for-everyone/

[14] Pranjal Pandev. Data Preprocessing : Concepts.
https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825

[15] Andrew Ng. Machine Learning Yearning.
https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825

[16] R webpage
https://www.r-project.org/about.html

[17] RStudio webpage
https://www.rstudio.com

[18] Git webpage
https://git-scm.com/

[19] Github webpage
https://github.com/

[20] Pandas webpage
https://pandas.pydata.org/

[21] Selenium WebDriver
https://www.selenium.dev/projects/

[22] Scikit-learn
https://scikit-learn.org/stable/

[23] fastText
https://fasttext.cc/

[24] Piotr Bojanowski, Edouard Grave, Armand Joulin and Tomas Mikolov. (2016). Enriching Word Vectors with Subword Information
https://arxiv.org/abs/1607.04606

[25] Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov. (2016). Bag of Tricks for Efficient Text Classification
https://arxiv.org/abs/1607.01759

[26] Heepsy
https://www.heepsy.com/

[27] HypeAuditor webpage
https://hypeauditor.com/en/

[28] Coobis: Content Marketing, Influencers & Branded Content.
https://suite.coobis.com/

[29] Chrome website
https://www.google.es/chrome/browser/desktop/index.html

[30] Ian London. Encoding cyclical continuous features - 24-hour time.
https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/

[31] What is the vanishing gradient problem?
https://deepai.org/machine-learning-glossary-and-terms/vanishing-gradient-problem

[32] J. Schmidhuber. (1992). Learning complex, extended sequences using the principle of history compression.

[33] Wikipedia. Residual neural network.
https://en.wikipedia.org/wiki/Residual_neural_network

[34] Wikipedia. Data augmentation.
https://en.wikipedia.org/wiki/Data_augmentation

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015. Deep Residual Learning for Image Recognition.
https://arxiv.org/pdf/1512.03385.pdf

[36] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization.
https://arxiv.org/abs/1412.6980

[37] Wikipedia. Mean squared error.
https://en.wikipedia.org/wiki/Mean_squared_error

[38] JJ. MAE and RMSE — Which Metric is Better?
https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

[39] Tensorflow. TensorBoard.
https://www.tensorflow.org/tensorboard

[40] MacPaw. Instagram removing likes: Why, when, and what it will look like.
https://macpaw.com/how-to/instagram-removing-likes

8 Annexes

8.1 R packages used

The packages used and required to make the website work are the following:

- dplyr
- ggplot2
- ggpubr
- tidyr
- dplyr
- scales
- ggcorrplot
- readr

- RColorBrewer
- tm

8.2 Python packages used

- tensorflow
- tensorboard
- scikit-learn
- pillow
- pandas
- numpy
- selenium
- urllib3
- fasttext