PERSONAL PROJECT

ANALYZING USER SATISFACTION IN APPLICATIONS

Developer: Marcos Matheus de Paiva Silva

Barra Mansa - RJ

September 2022

Contents

1 Objective

With today's digital age revolution, some developers are able to automate processes, to create customer support management, analyze user feedback, explore survey responses and evaluate chats. Thus, developing a program that performs sentiment analysis is essential in the modern era.

In the current project, we fully contemplate the stages of a data science project, so that from the information collected on the website Kaggle and data visualization we can probe some questions that would help users, companies, shareholders and developers understand about:

1. What days were the updates made that most influenced the number of installations?
2. Which apps have the most installs?
3. What categories, genres, and maturity ratings of apps were installed the most?
4. What categories, genres, maturity ratings were rated the best by users?
5. What exactly were the highest scoring applications?

Finally, after creating the model, we developed an application, which allows any user to predict whether or not he is satisfied with an application (Deploy). In addition, if the user performs a very robust and enlightening critique, it is possible to generate an exploratory data analysis in the application. Every procedure can be performed based on information collected or provided by the user, about their experience with the application.

2 Methodology and Proposed Solution

Initially we used dataset from google play, in order to find the answers to the questions raised in the previous section. The dataset is on github, for better execution of the algorithm in the tool Google Colaboratoty[1].

To achieve our goals, throughout this project we took advantage of numerous python libraries, from data analysis to visualization: numpy, pandas, matplotlib, seaborn, os,urllib, zipfile, wordcloud, re, sys, math, nltk, collections, sklearn, spacy. All python libraries have several modules and ready-made functions that make coding easier. For data pre-processing and creation of our model we use the libraries: tensorflow, Keras , finally to implement the model we use the streamlit library.

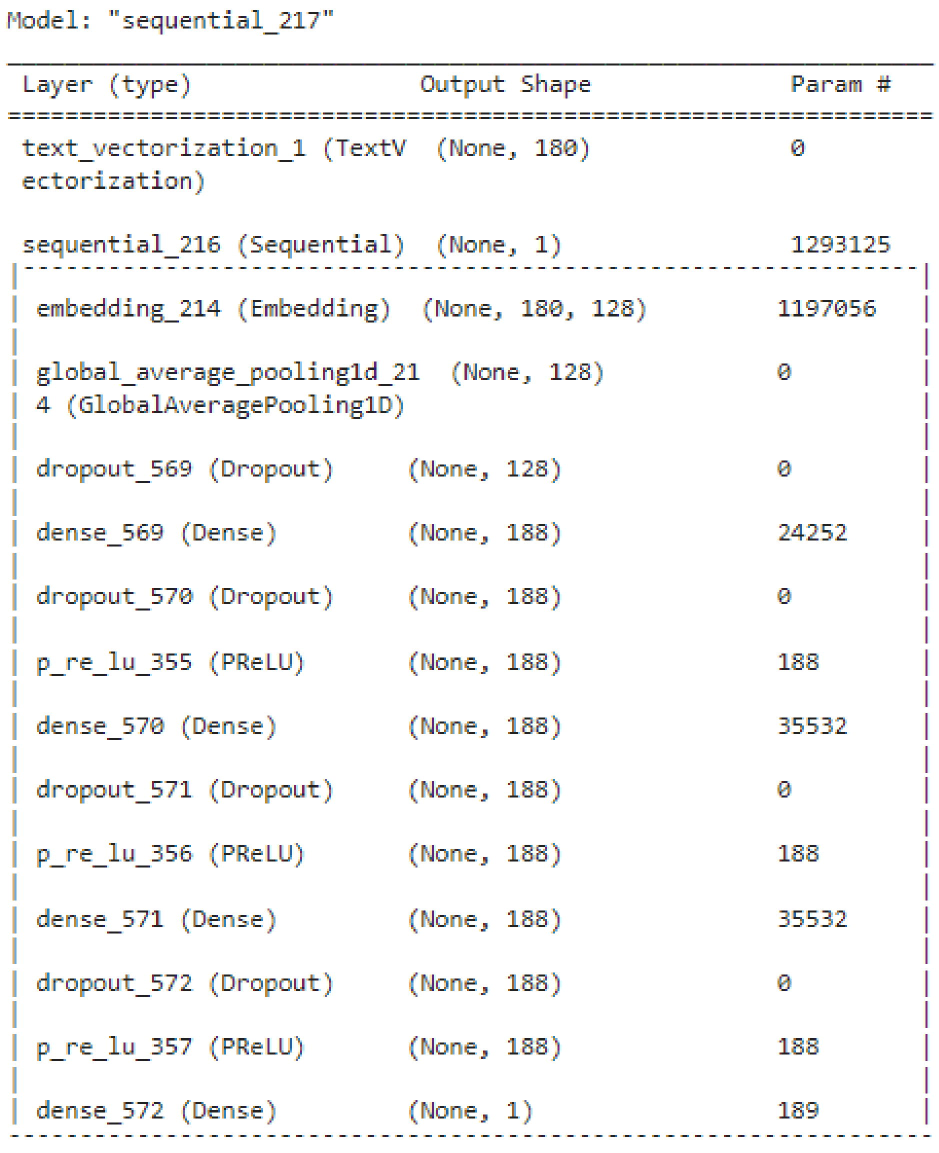
In addition, we had to use several models of **neural networks**, but in the end we only had one, to use in our final model, so the models we tested were embedding, LSTM, GRU and SimpleRNN. Neural networks are based on biological neurons and can make predictions from input data, as they are based on biological neurons enjoying layers, neural networks have input layers, fully connected layers and output layers. Such layers have nodes that are neurons that connect with other neurons, if the neuron reaches a threshold value, it will be triggered, taking information to the next layer of the network, if this does not occur, there will be no data transmission. In theory, our neural network receives user comments as input data, and generates a good or bad rating as an output, then we train our model through comments and ratings that users made from google applications, with this, it can predict user rating based on their comment.

In our neural network (fig: 1) we use Sequential() to assemble the layers into a ctf.keras file. Model in a linear stack, the first layer we add is to ensure that there is exactly one input string per batch, the next is the preprocessing layer. Continuing in our network, we create a layer embedding(Transforms positive integers (indices) into dense fixed-length vectors) taking as input several parameters . The next layer we add is GlobalAveragePooling1D(), which in the 2D case is usually applied to images to reduce resolution or computational load, but in our case we can use this to decrease the number of trainable parameters. I didn't use Flatten because with Pooling1D we got better scores.

Therefore, we use a regularization technique called Dropout, which basically skips a few neurons at a rate of 10% during training, but can be activated in a later step.
Next, we use dense layers that link the neurons of one layer, with the previous layers, and apply the weight initializer, which sets the weights of a neural network to short

[1] Google Colaboratoty "is a tool data analysis and machine learning that allows you to combine executable Python code and rich text with graphics, images, HTML, LaTeX and more in a single document stored in Google Drive".(GOOGLE, 2022)

Figure 1 – Neural Network Structure



```
Model: "sequential_217"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization_1 (TextV  (None, 180)              0
 ectorization)

 sequential_216 (Sequential)  (None, 1)                1293125
|---------------------------------------------------------------|
| embedding_214 (Embedding)  (None, 180, 128)        1197056   |
|                                                               |
| global_average_pooling1d_21  (None, 128)           0         |
| 4 (GlobalAveragePooling1D)                                    |
|                                                               |
| dropout_569 (Dropout)     (None, 128)              0         |
|                                                               |
| dense_569 (Dense)         (None, 188)              24252     |
|                                                               |
| dropout_570 (Dropout)     (None, 188)              0         |
|                                                               |
| p_re_lu_355 (PReLU)       (None, 188)              188       |
|                                                               |
| dense_570 (Dense)         (None, 188)              35532     |
|                                                               |
| dropout_571 (Dropout)     (None, 188)              0         |
|                                                               |
| p_re_lu_356 (PReLU)       (None, 188)              188       |
|                                                               |
| dense_571 (Dense)         (None, 188)              35532     |
|                                                               |
| dropout_572 (Dropout)     (None, 188)              0         |
|                                                               |
| p_re_lu_357 (PReLU)       (None, 188)              188       |
|                                                               |
| dense_572 (Dense)         (None, 1)                189       |
-----------------------------------------------------------------
```

Fonte: Autor

random values, in addition to determining the kick-off to optimize the model. of neural network. In our model we use the initialization He which receives an arbitrary number whose probability distribution is the Gaussian, with standard deviation $(\sqrt{2}/n)$, such that n is the number of connections of a response across the layers and averaged 0.0.

Furthermore, we employ what we call the activation function, which determines the way in which the weighted sum of the neural network input is converted into an output of nodes in a layer of the network. The activation function we use is PreLU, which adapts as it understands the parameters of the rectifiers and improves accuracy with additional computational cost insignificant. But what is a rectifier? Basically, a linear rectified activation unit is a unit or node, which elaborates the activation function reported above, therefore, rectified networks are those that benefit from the rectifier function for the hidden layers. Finally, we use an output neuron and an activation function sigmoid, where sigmoid is a logistic function whose result can be 0 or 1 (0 dissatisfied with the application, 1 satisfied).

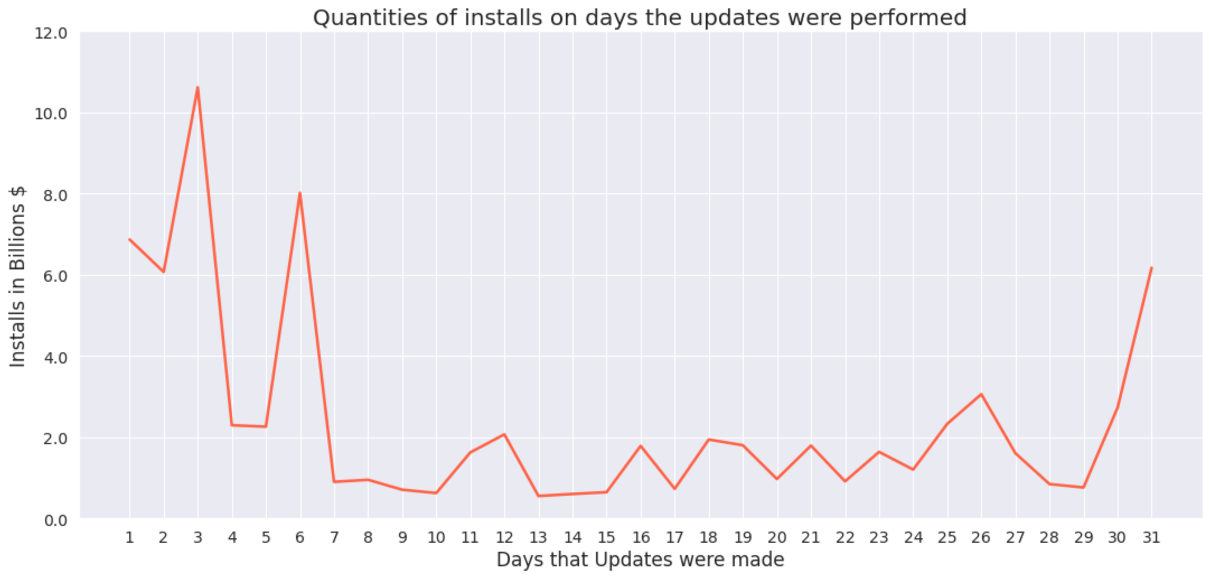
In neural network models we must compile the model, to initially configure the loss functions, optimizers and which metrics will be applied to it. The optimizer we use is Adam which is based on momentum, momentum aggregates exponentially declining moving averages from previous gradients and proceeds to move towards he. The Adam optimizer, on the other hand, measures individual learning rates that adapt to different parameters, through approximations of the first and second moments of the gradients. In addition we use what we call a loss function, a loss function returns error values, that is, the smaller the better, the loss function we will use is binary cross entropy, which tells us an exit probability between zero and one. Furthermore, such a loss function grows as the predicted probability diverges from the current label. Finally, we use accuracy as an auxiliary metric.

This project was developed following the steps of the book (GéRON, 2019), also based on some precepts found in the books (MüLLER; GUID, 2016), (GOODFELLOW; BENGIO; COURVILLE, 2016), (TATSAT; PURI; LOOKABAUGH, 2020), (MIRTAHERI; SHAHBAZIAN, 2022) and in the articles (HE et al., 2015), (XU et al., 2015), (MAAS; HANNUN; NG, 2013). In summary, this text was written in Latex.

3 Results

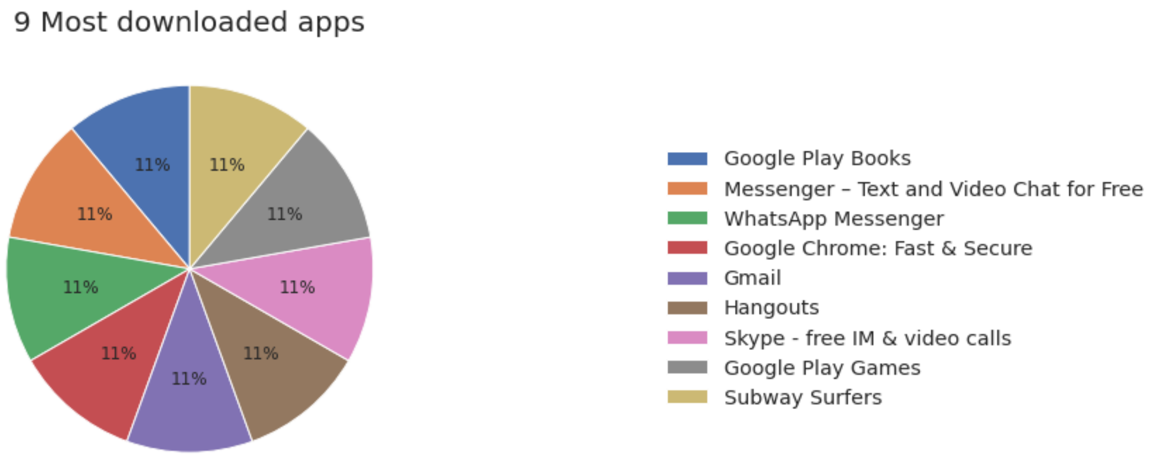
3.1 Answering Business Questions

We started our jupyter notebook by extracting some insights from the data, and also plotting some graphs to answer the business intelligence questions, the first question raised was what is the best day to perform an application update:



source: Autor

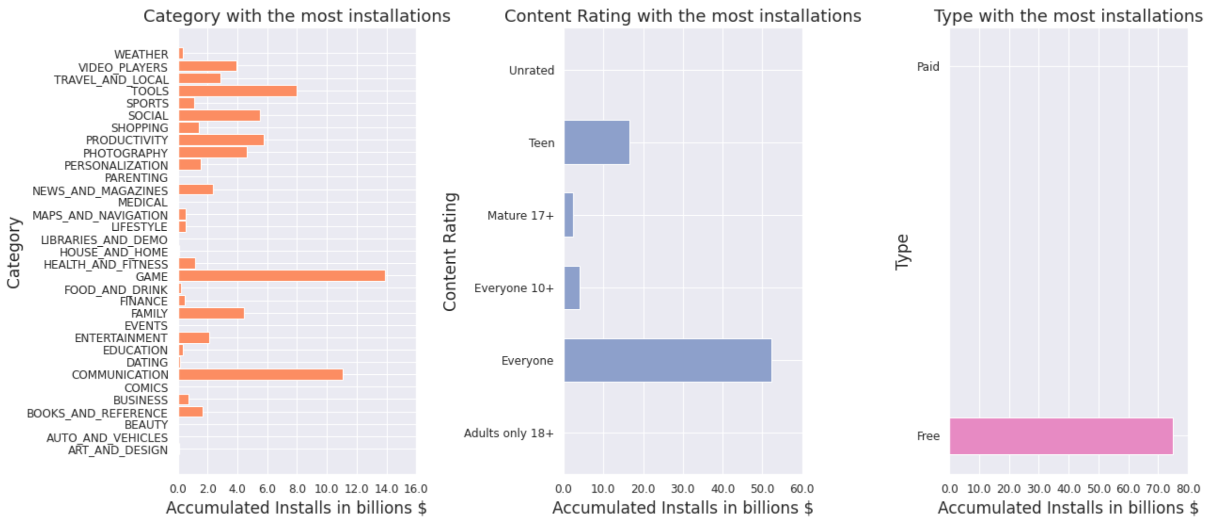
we noticed from the chart above that the day with the most installations was the 3rd. Then we wanted to find the most downloaded apps, and with that we plotted the pie chart:



source: Autor

The graph above is self-explanatory, however either our dataset is bad and doesn't have significant app information, or all this data has the same download numbers.

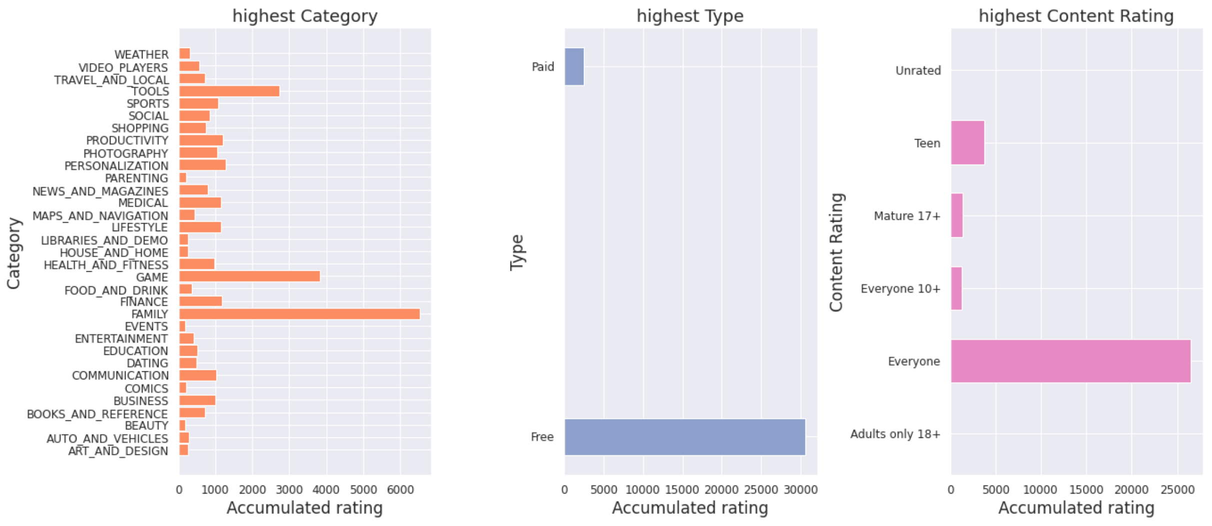
Since we didn't find good information about individual apps, we're going to group some of the apps into classes, so we've plotted Category, Content Rating, and Type charts by Accumulated Installs in billions, in order to find the classes that had the most installs.



source: Autor

Here we see that the category of apps that got the most installs were games and communication apps, as well as those indicated for any audience and free.

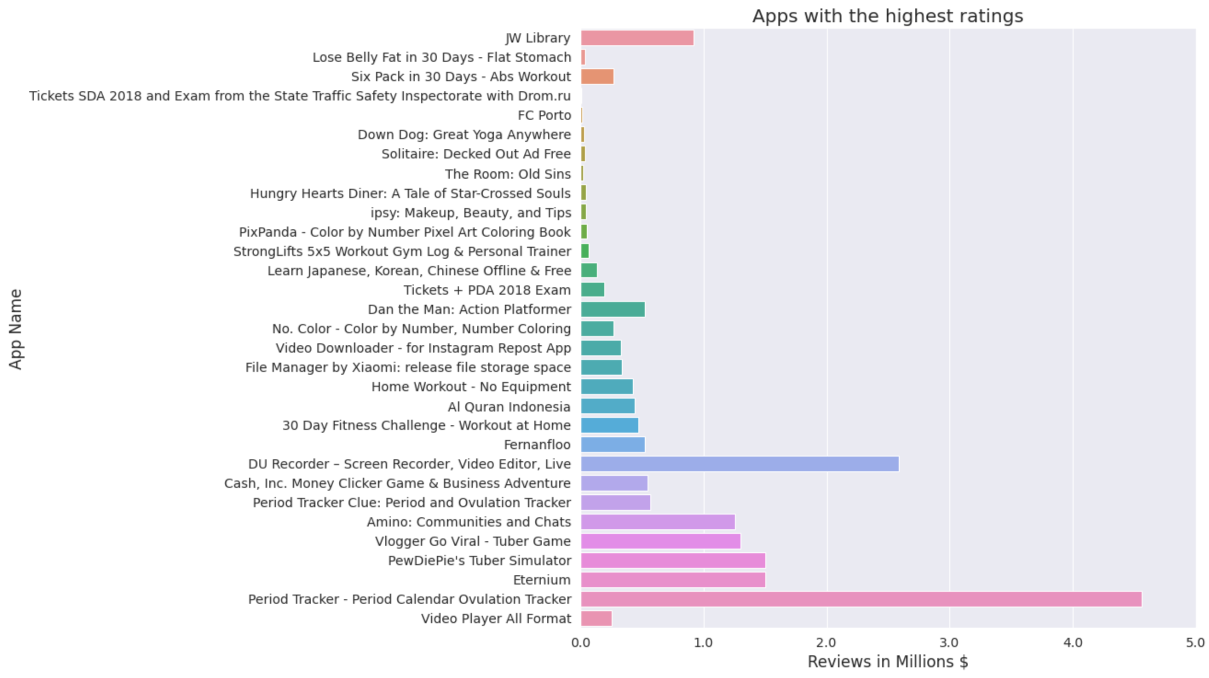
We do something similar for these same classes, comparing them by the number of accumulated ratings.



source: Autor

We can see from these charts that the apps with the highest ratings are the free and familiar apps that anyone of any age can play.

At the end of the data visualization section we select the apps with the highest Rating, sort them in descending order in the data, and plot the name of the apps by the number of reviews. This was done, since the amount of criticism also influences for an app to have a high rating or not.



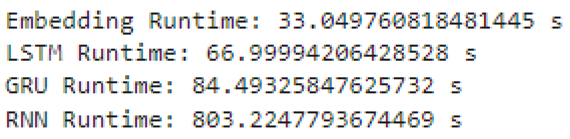
source: Autor

From the chart above, as the ratings are in descending order, the jm library is the highest rated app. After the data visualization part, we also go through the exploratory data analysis, for more information, visit: jupyter

3.2 Choice of Model for Sentiment Analysis

In order to classify users' feelings as good or bad, based on their comments, we trained some models as discussed in the 2 section, the time results of the models are shown below.

Comparison of Model Times

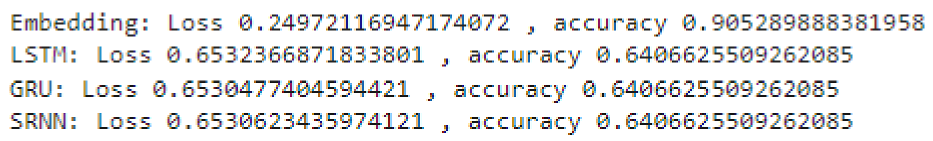


```
Embedding Runtime: 33.049760818481445 s
LSTM Runtime: 66.99994206428528 s
GRU Runtime: 84.49325847625732 s
RNN Runtime: 803.2247793674469 s
```

source: Autor

From the figure above, we understand that the best model was the Embedding with the shortest execution time. In addition to comparing the time of the programs, we also

Comparação dos tempos dos Modelos



```
Embedding: Loss 0.24972116947174072 , accuracy 0.905289888381958
LSTM: Loss 0.6532366871833801 , accuracy 0.6406625509262085
GRU: Loss 0.6530477404594421 , accuracy 0.6406625509262085
SRNN: Loss 0.6530623435974121 , accuracy 0.6406625509262085
```

source: Autor

compare their loss functions.

We therefore note that the embedding neural network model is better in terms of both time and loss function. A very important factor that we must point out is the fact that if we add the parameter mask_zero=true in the GRU, LSTM, SRNN layers, the model achieves low loss functions, however such models take much longer to run with this parameter . Furthermore, if we put the parameter return_sequences = true in the last layers GRU, LSTM, SRNN, these same models achieve good loss function results, but such an operation, besides not being a good practice, does not give us the output dimension. expected since the output dimension for return_sequences = true is 3D, and we don't want that.

4 Conclusion

In the course of this project, we developed relevant concepts in data science and analysis in order to suggest the solution to know if a user is satisfied with google apps, based on their feedback. In addition to this objective, we answered some business intelligence questions, and performed an exploratory analysis on our data.

In this way, we go through several steps precisely to extract as much information as possible from our dataset. As a result of our investigation done on the data, we found that the best day to perform an app update is day 3 as it has more install records. In addition, we found that the apps that had the most installs were games and communication apps, as well as those indicated for any audience and free. In the later results we found the apps with the highest ratings, they are the free and familiar apps that anyone of any age can play. At the end of the data visualization part we find that the application named jm library is the highest rated application found in our dataset.

Therefore, we performed an exploratory data analysis in jupyter notebook, with this we were able to understand in our data, the most frequent words, bigrams and entities in each type of classification (good, neutral and bad). At the end of our project, we tested several models of neural networks and employed numerous techniques that can be verified in the jupyter notebook in order to further improve the model and increase its efficiency. Thus, the best model found was a neural network with an embedding layer, due to its lower loss function result, otherwise, this model was also more efficient than its competitors in terms of time. Finally, we implement our model using streamlit so that any user, developer or shareholder can use it.

Bibliography

GOODFELLOW; BENGIO, Y.; COURVILLE, A. *Deep Learning (Adaptive Computation and Machine Learning series)*. 1ª. ed. Massachusetts: MIT Press, 2016. Quoted on page 5.

GOOGLE. *Colaboratory*. 2022. Disponível em: <https://workspace.google.com/marketplace/app/colaboratory/1014160490159>. Quoted on page 3.

GéRON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2ª. ed. Canadá: O'Reilly, 2019. Quoted on page 5.

HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. Disponível em: <http://arxiv.org/abs/1502.01852>. Quoted on page 5.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. [S.l.: s.n.], 2013. Quoted on page 5.

MIRTAHERI, S. L.; SHAHBAZIAN, R. *Machine Learning: Theory to Applications*. 1ª. ed. Florida - USA: CRC Press, 2022. Quoted on page 5.

MüLLER, A. C.; GUID, S. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. 1ª. ed. United States of America: O'Reilly, 2016. Quoted on page 5.

TATSAT, H.; PURI, S.; LOOKABAUGH, B. *Machine Learning and Data Science Blueprints for Finance: From Building Trading Strategies to Robo-Advisors Using Python*. 1ª. ed. United States of America.: O'Reilly, 2020. Quoted on page 5.

XU, B. et al. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015. Disponível em: <http://arxiv.org/abs/1505.00853>. Quoted on page 5.