Chapter 2
Sentiment Analysis on Google Play Store Data Using Deep Learning

Swathi Venkatakrishnan, Abhishek Kaushik, and Jitendra Kumar Verma

1 Introduction

Mobile app stores such as Google and Apple have a wide range of applications to suffice the basic needs of customers in the digital platform. These days, with new renovated augmentations in technologies and ascent of opportunities, various development tool kits are readily available in the market for software developers. The developers are wary of interests of the customers and they also have an idea of the target audience with respect to the application in question, but unless they do get an appropriate feedback, there is little scope for improvement and credibility. This is what the developers yearn for. Customer feedback and ratings have always been one of the major metrics that can be used to review the performance and provide suitable recommendations to enhance the functionality provided by the app [1].

Application developers usually deal with large number of critical reviews which they have to filter out to suit the requirements of the appropriate stakeholders. The objectives of this research can be further explained as:

- Can the application-specific metrics, when combined as a whole, along with details and reviews, be used to determine the user rating?
- How can deep learning be adopted to combine different categorical, textual and numerical data to determine user app ratings?

S. Venkatakrishnan
School of Computing, Dublin Business School, Dublin, Ireland
e-mail: swathivenkat0191@gmail.com

A. Kaushik (✉)
ADAPT Centre, School of Computing, Dublin City University, Dublin, Ireland
e-mail: abhishek.kaushik2@mail.dcu.ie

J. K. Verma
Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Haryana, Gurugram (Manesar), Haryana, India
e-mail: jitendra.verma.in@ieee.org

P. Johri et al. (eds.), *Applications of Machine Learning*,
Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-15-3357-0_2

- Can vectorization techniques affect the overall outcome of the classification?

We analyse the different variants of data together by combining text, numerical and categorical data. The sole purpose to take up this task was to put forth the various intricate involved in data extraction and pre-processing stage, and further study the performance of a deep learning model that would integrate the different forms of data to perform multi-class classification. This study describes both; source of the data and concatenation of deep learning models that use the functional layer of Keras for combining text vectors and numerical columns of data for performing classification [2].

2 Literature Review

Hengshu Zhu et al. [3] mainly emphasize on how the mobile app market is way more susceptible to fraud activities such as pushing up the apps in the popularity list. To counter this, they have explained a view of the current ranking fraud cases and proposed an approach to study the fraud detection system to determine which app ranks have spam markings. The two main features considered for training the model are ranking-based evidences and application-based ratings. Hypothetical statistical tests are performed on both; app rankings and rating variables; to study the varying trends of these features to determine the fraud occurrence. This study helped us to understand which mobile app features have more significance to study the varying trends in rankings.

Data mining is not just restricted to text or image processing but it involves a lot of other important factors such as multimedia mining or graph mining [4]. So this knowledge discovery process is elaborated in depth by the categorization process that performs on the same ideology as that of bag of words and the natural language processing (NLP) technique. Vijayarani et al. have given a detailed description of the extraction process, the stop words elimination techniques, stemming and comparison of methodologies adopted for removing stop words like using pre-compiled lists, Zipf's law, mutual information methods and term-based random sampling. They also showcase scenarios in which the stemming operations based on statistical analysis are performed. Here, it helped us in understanding the significance of stemming operation on text data to improve the overall performance of the model.

Abhishek and Sudhanshu [2, 5] have provided an extensive critique about various text mining techniques, tools and applications in this article. A wide-scale analysis was performed on several terminologies and techniques of data mining such as categorization, clustering, topic tracking and sentiment analysis to name a few. Kaur et al. [6] have investigated the sentiment analysis on Hinglish data using the semi-supervised learning using different forms of vectorizer in parametric and non-parametric model.

Zhang et al. [7] basically provide an overview of how deep learning and neural network can be used to showcase a comprehensive survey of the present scenario with respect to the applications used for sentiment analysis. It is stated how deep learning

uses multiple layers of processing models for main feature extraction. This study introduces various in-depth explanations about various deep learning architectures and how each of them, when used as an individual model or as a combination of various neural nets can thereby depict state of the art results for various text processing techniques.

Hinton and Salakhutdinov [8] have stated that deep learning methodology, in the long run, will be a widely used approach for text processing owing to its high tendency to study the data in a more refined way. If a data of high dimension is showcasing its constraints with respect to its varsity and shape; it can very well be converted to a low-dimensional form by training a multi-layer neural network. It also goes ahead to state that this auto-encoder network might be a more effective way than principal component analysis to learn the low-dimensional form of vectors.

Neural networks have always been a star performer by achieving remarkable results in sentence and document modelling [9]. When two popular and prevailing architectures, CNN and recurrent neural networks (RNN), are implemented for language processing tasks, we can observe that they adopt totally different methodologies for implementation. The end result indicates distinctly that the combined model outperforms both the individual models.

Isa et al. [10] implement a modified hybrid classification technique through the Naive Bayes approach and the support vector machine (SVM). This project focuses on the Bayes formula to vectorize a document rather than using the traditional classification technique. This technique comparatively uses less training time period than the L Squared methodology and has no constraints with respect to the type of data being used. On further evaluation, the classification accuracy increases to a legit scale compared to Naive Bayes systems.

Abhishek et al. [5] have proposed a system where one can automate rate the opinions in the form of unstructured data. Although it is a challenge in the present scenario, this paper presents an in-depth ideology on sentiment analysis methods and tools being used for the same. They have further elaborated the workflow of opinion mining where the techniques used for opinion mining involves opinion retrieval, classification and summarization.

Blessy et al. [11] have put forth different approaches of sentiment classification and the existing techniques with the framework. It states that the feature extraction stage deals with feature types specifically for opinion mining; in this context, feature selection phase opted for opinion classification, feature weighting strategy and finally reduction mechanisms for optimization purpose.

3 Data

3.1 Data Extraction

It is essential that the data source for research should be completely authentic. The play scraper authentic API package is available for users for extraction purposes. This package consists of major filters which can be used while fetching the data.

Table 1 Sample of app categories

Categories	Top free	Top trending
Weather	TOPFREE_WEATHER	TRENDING_WEATHER
Maps and navigation	TOPFREE_MAP_ AND_NAVIGATION	TRENDING_MAPS_ AND_NAVIGATION
Medical	TOPFREE_MEDICAL	TRENDING_MEDICAL
Photography	TOPFREE_PHOTOGRAPHY	TRENDING_PHOTOGRAPHY
Games	TOPFREE_GAMES	TRENDING_GAMES

Some of the user-friendly pointers used would be fetching data as per collection of sections such as trending, top ranked; top paid, top free, country or region-specific since it involves reviews from varied customers; developer-specific applications; to fetch details as per a developer's offered set of applications. Further, each of these sections is bifurcated into multiple categories. The whole list of packages and python files used is as listed out below:

1. Play scraper link[1]
2. Enhanced version of reviews module[2]
3. Built Python scripts.

For this particular research work, the data fetched mainly belonged to top trending and top free collection. Each collection had 58 categories. To mention a few as shown in Table 1. The data fetched through play scraper API (application programme interface) was in JavaScript Object Notation (JSON) format. A python script was used to convert the nested JSON structure to a CSV file format. This JSON form was converted to a CSV file form using a python script named *FetchJSONConvertToCSVDetails.py*.

3.1.1 Structure of Play Store

This third-party package is mainly used for extracting application details. However, this package does not have a module for extracting reviews for the apps. One enhanced version of the same module is available on GitHub for fetching reviews. This module works on this principle of fetching application reviews with respect to the application ID specified. They have basically created a new function that deals with reviews section and invokes the review URL. It fetches reviews and related details of the application ID specified.

Since the scope of this project involves dealing with multiple applications within each category that further falls under major collections; it would be tedious to invoke

[1] https://pypi.org/project/play-scraper/#details.

[2] https://github.com/danieliu/play-scraper/commit/3906d4f6814073cdc5c83b88635607dcc99ff3d5.

reviews of each application one-by-one. So, we have created a new script *MultipleAppsReviews.py* that can be forked for extracting reviews of all applications of the specified category name. It also involved tweaking the code in the python scraper main class *scraper.py* where we have augmented that part of the code where now, it accepts a list of apps instead of a single app name as an input and passes this entire list as one parameter in the POST request to the specified Uniform Resource Locator (URL). This is in turn retrieves the reviews of all the applications appended in the list.

3.1.2 Work Flow of Web Scraping

This play scraper is a python API. Figure 1 indicates the stage-by-stage functioning of each script used for getting the data from the play store. The detailed technical parameters and purpose of each script are given in the Github link provided in the references [12]. Further, the scraper.py file that was enhanced for fetching reviews of multiple applications falling under a category is also provided in Github [13].

3.2 App Details and Review Details Specifics

3.2.1 Details

- Fetch app details as per collections: TOPFREE and TRENDING.
- Every collection has 58 small categories.
- Each category has multiple applications that are sorted under it.
- Every application has a set of detailed metrics to be used for analytical processing.

Fig. 1 Entire workflow

- Columns for application details: application Id, category, content rating, current version, description, description HTML, developer, developer address, developer email, developer ID, developer URL, editor's choice, free, histogram, IAP, IAP range, icon, installs, interactive elements, price.

3.2.2 Reviews

- Since we have fetched application details with respect to collection and categories; it is advisable to fetch application reviews too on similar lines.
- List of columns fetched in the reviews case are as follows: author image, author name, current rating, review app ID, review body, review date, review ID, review permanent link, review title.

Since this research involved studying just application detail-specific and review-specific features, we have dropped out on all the other unnecessary columns to avoid clogging the idea.

1. Merging the app details within two collections:
   So, as per the design plan, there are two major collections: trending and top free. Within each of these collections, there are 58 CSV files containing application details of apps where each sheet represents a unique category within that collection. We combine all the 58 categories data sheets falling under one collection 'top free' with other 58 categories CSV sheets falling under the other collection 'trending'. A python script *CombineAllDetailCollections.py* was used to merge them.
2. Merging reviews within two collections:
   Similar to the details case, we merge the reviews under two different collections within one sheet. Both the merge functions are performed using Python scripts.

4 Dataset Description

While there were minor technical glitches while concatenating different types of data; the main focus was to maintain the overall quality of the data; after we combined the details and the reviews. As seen, the dataset seemed quite imbalanced with respect to the number of rows maintained in both these sheets. Each app had unequal number of reviews. To resolve this, we have considered just 20 rows of reviews for each app using newly created python script *ReviewHead.py*. To ensure that we have complete information of an app; all numerical and categorical columns in details are adjoined to the textual columns in reviews; stacked up horizontally against each other; a python script named *CombineDetailsToReviews.py* was created to combine the details of the apps present in the details sheet to that of the reviews sheet keeping application ID as the base identifier. When all 58 categories are present, it would be a little tedious to work through.

To make it simpler, sub-groupings of categories was added. Since there were a varied range of 58 categories involved, we segregated the categories into 14 broad unique categories as shown in Table 2.

The main broad categories are further label encoded to pass through the learning model.

- Improved performance of predictors
- Cost effective.

Table 2 lists apps having similar base functions under one major category.

Table 2 Number of sub-categories falling under each broad category

Broad categories	Sub-categories grouping
ART_AND_DESIGN	Art and design
AUTOSHOP	Auto and vehicles
PERFORMANCE_BOOST	Lifestyle/beauty/parenting/personalization/productivity/ tools/video players
KNOWLEDGE	Books and reference/business/education/libraries and demo
ENTERTAINMENT_LIST	Entertainment/music and audio/comics/events/photography/shopping
COMMUNICATION	Communication/dating/social
GAMES & FAMILY ACTVITIES	Game arcade/game casual/game card/game action/game adventure/game board/game card/game casino/game casual/game educational/game music/game puzzle/game racing/game role playing/game stimulation/game sports/game strategy/game trivia/game word/sports family pretend/family music video/family education/family create/family brain games/family action/family
HEALTHCARE	Medical/health and fitness
FINANCE	Finance
FOOD_AND_DRINK	Food and drink
HOUSING	House and home
MAPS_AND_TRAVEL	Maps and navigation/travel/travel and local
ENTERTAINMENT_LIST	Entertainment/music and audio/comics/events/photography/shopping
NEWS	News and magazines
WEATHER	Weather

5 Data Exploration

To study the impact of each of the weighted attributes such as installations, rating score in each of these categories, exploratory data analysis was performed. The distribution of the different major categories across the dataset is shown in Fig. 2.

Clearly, the value counts of GAMES category was the highest and the number of applications and categories falling under WEATHER was the least. The range of total average rating against each of the main categories and distribution of installs against every category is depicted using a scatter plot in Figs. 3 and 4.

6 Methods

The feature variables being selected for this research work were application ID, unique broad category, total average rating, number of installations, required android version, user review text, user rating, total number of reviews.

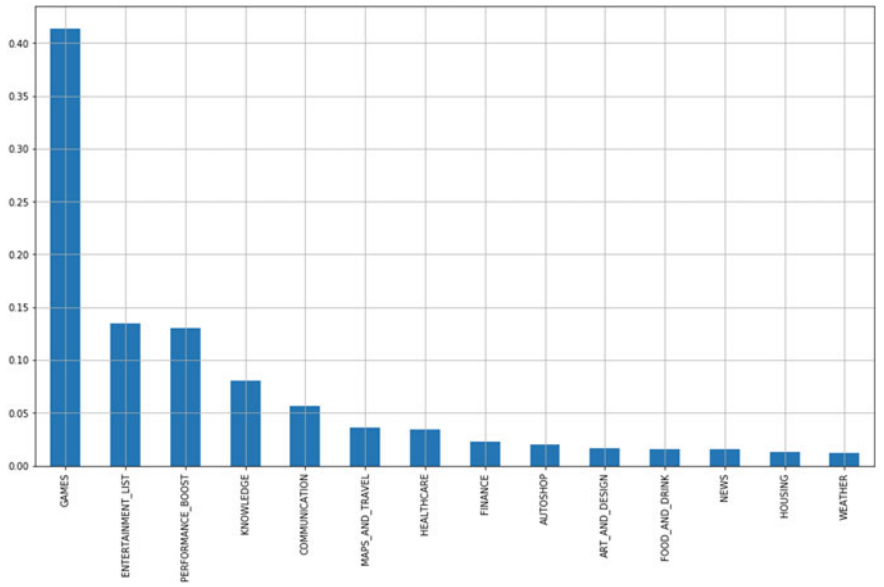


Fig. 2 Distribution of the different major categories

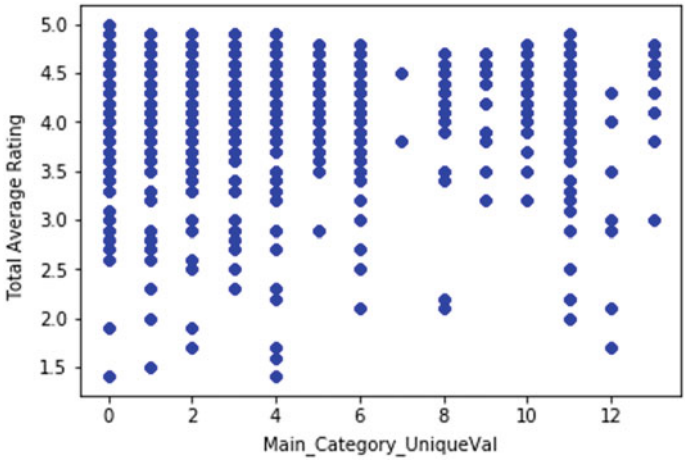
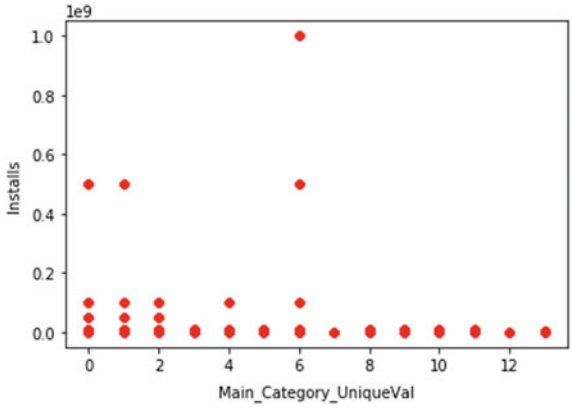


Fig. 3 Distribution of main categories versus average rating

Fig. 4 Distribution of main categories versus installs



6.1 Data Processing

Since we were dealing with raw extracted data, our primary goal was to clean it to avoid any discrepancy in our results.

6.1.1 Text Vectorization

The Scikit-learn library provides user-friendly libraries to use count vectorization and TF-IDF vectorizer on real text data. Before we could jump into the vectorization part, it was crucial to remove stop words, encoded punctuations, and lemmatize our content to maintain the quality decorum during processing.

Table 3 Results of some tested classifiers

ML classifier	CV accuracy (in %)	TF-IDF accuracy (in %)
Naive Bayes	67.29	70.13
XG Boost classifier	66.92	67.63
MLP classifier	67.58	65.68

1. **Count Vectorization**: This is one of the most basic ways that can be utilized to represent text data numerically. This concept is also called as one-hot encoding. The basic process that is being implemented here is we create vectors that have a dimensionality equal to the size of the vocabulary. At every instance of encountering a vocab word within a textual review; we increase the count by 1 and add 0 in place of null encounters.
2. **TF-IDF Vectorization**: The term simply means "term frequency—inverse document" (TF-IDF) frequency. This basically implements the vectorization by performing the TF-IDF transformation of the matrix of counts.

Both count vectorization (CV) and TF-IDF were applied on the same column to evaluate the performance of both the results in the case of text classification classifier model Table 3.

6.2 Numerical Data Columns

There were multiple numerical columns of different data types that were involved in this research work. Each value range was quite extensive, so it was essential to scale it down to a normalized range. List of columns that were normalized are as follows:

- Total Number of Reviews: Total number of reviews ranged from 10–15 to maximum range of 48160293. To ensure it stays in the same range as the other columns used in experiment, we use Min-Max Scalar libraries to reshape our column to lesser range.
- Installs: Total number of installations that have occurred for each application also has huge variations in the values. To bring it down to a normalized form, we divide the entire column by the maximum value in the column '1000000000'.
- Required Android Version: Required android version was one more column that had varying values which would have been time consuming for the model to interpret. This too was scaled down to a reasonable value using Min Max scalar.
- Average Rating: Total average rating basically ranges from 1 to 5. To bring it down to a value similar to other columns and in the [0,1] range, we divide the entire column using max value of the column.

6.2.1 Categorical Encoding

Since the output variable is varied range from 1 to 5; we need to label encode it to be used while processing the model. This is done using the to_categorical package provided by scikit-learn.

7 Modelling Results

Using our refined dataset [14], we have used deep learning classifiers to determine user ratings on the basis of all other application metrics. Functional layers of multi-layered perceptron classifier were used to build the model architecture. The entire design specification is as shown in Fig. 5. After performing web scraping and combining multiple sheets into one dataset, we divide the data frame into 30% test set and 70% train set. After the pre-processing stage, we divide our modelling stage into two parts:

1. For application details, we select the best performing classifier amongst MLP and DDMLP.
2. Similarly, we select the best performing model amongst MLP models in case of application reviews; the only difference being the vectorization method used for converting text reviews to the vector form.

The first sub-model was built specifically for text classification on the basis of user ratings falling within the range of 1–5. Second sub-model was solely developed to predict the user ratings from (1–5) with respect to the other numerical feature variables. The combined hybrid model was used to merge two-layer outputs of both the sub-models and process the entire data frame that was a combination of text vectors, numerical values and categorical features. This combined hybrid model recorded an optimal test accuracy of 70.45%.

This functional input layer gives the feasibility to first build layers of input models. Here, each layer is assigned a different name so it can distinctively identify each layer weights while training the model. The first input layer is the visible layer that takes in the set of inputs given to model for training. The number of feature variables added here was 5. So the shape of the value was fed as 5 in the input layer itself. The next three layers added were hidden layers constructed with ReLu activation functions. The last hidden layer was the Softmax functional layer that gives out the output of the app details classification. This output layer is named as output 1. The second input model has one input layer, two hidden layers and an output layer. There were 25, 5 and 6 neurons in each of those layers, respectively. The input layer and the intermediate layer neurons are activated by the ReLu function. The output layer is aligned with the Softmax function that produces a set of six-dimensional output vector where each row would signify the probability that the sample belonged to that class. This app reviews classification would directly indicate the user rating that the class would fall under. The output layer of these set of layers is called as output 2.

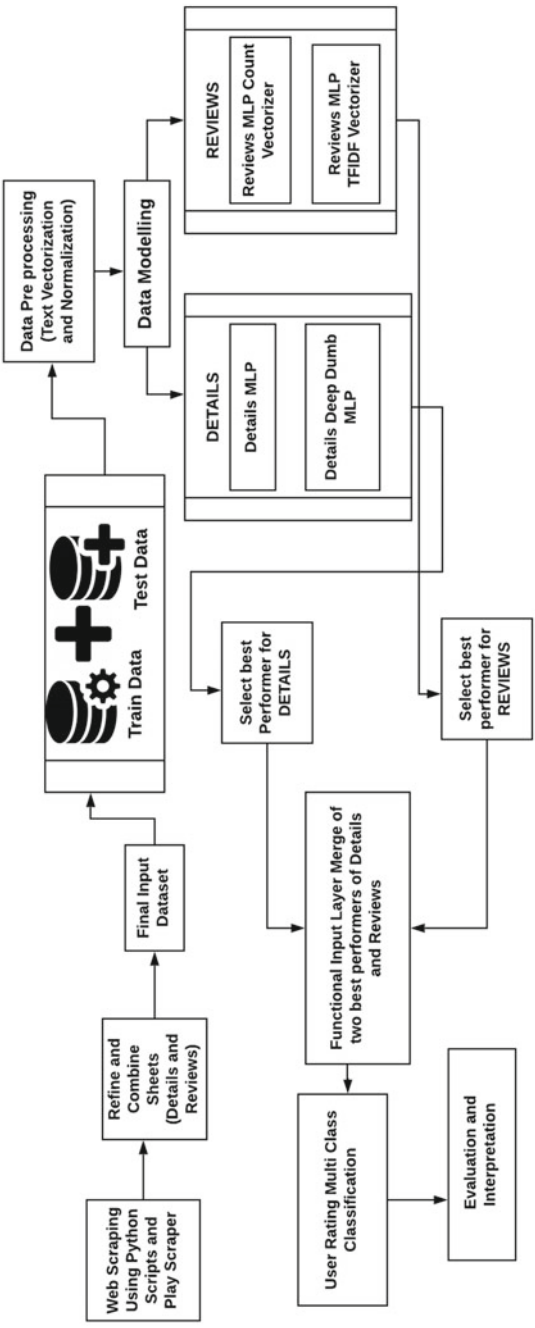


Fig. 5 Data modelling process flow

Keras provides few functions such as averaging the inputs and even concatenating the inputs to create a new input layer for the third set of layered network model to form a combination. The output 1 and output 2 are combined to form a merged content. This merged value is then provided as an input to the hidden dense layer activated by the ReLu function. The inputs consist of different sets of data, which would produce a six-dimensional vector of user ratings. This last layer is also embedded with Softmax function. The model is trained using the best probable optimizer rmsprop and sparse categorical cross-entropy loss function.

Since a number of models and their combinations were implemented, the variations of accuracy across different epochs and batch size were recorded which is shown in Table 4.

The ROC of the multi-class classification of MLP model was used to interpret the performance of the classifier. The curve shown in Fig. 6 shows the performance of the model embedded with text vectorized using TF-IDF concept. This is used

Table 4 Epochs and batch size for deep learning

Classifier	Epochs	Batch size	Train Acc	Test Acc
MLP-details	25	400	64.32	64.77
DDMLP-details	25	400	64.02	64.34
MLP-countVec	25	400	69.08	67.61
MLP-TF-IDF	25	400	88.73	66.56
Functional count	25	400	68.44	67.26
Functional-TF-IDF	25	400	85.04	68.62
MLP-details	12	450	64.22	64.45
DDMLP-details	12	450	63.42	63.4
MLP-countVec	12	450	68.18	68.16
MLP-TF-IDF	12	450	85.96	68.98
Functional count	12	450	67.63	67.83
Functional-TF-IDF	12	450	75.12	70.45
MLP-details	15	500	64.33	64.36
DDMLP-details	15	500	63.47	63.3
MLP-countVec	15	500	68.47	68.2
MLP-TF-IDF	15	500	87.77	67.86
Functional count	15	500	67.82	67.99
Functional-TF-IDF	15	500	75.92	69.94
MLP-details	13	480	63.96	63.99
DDMLP-details	13	480	63.45	63.44
MLP-countVec	13	480	68.24	68.17
MLP-TF-IDF	13	480	86.84	68.78
Functional count	13	480	67.99	68.07
Functional-TF-IDF	13	480	78.43	70.32

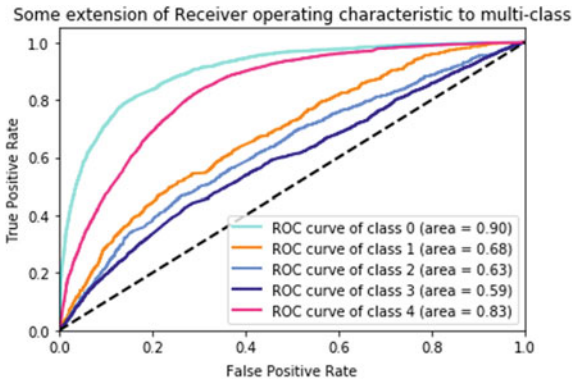


Fig. 6 ROC curve

to evaluate the quality of the output produced by the classifier. It aims in depicting the true positive rates on the *Y*-axis and the false positive rates are determined by the *X*-axis. The ROC works in a way to showcase that an ideal curve should indent more towards the true positive score of 1 and a false positive score close to 0. The projection should be such that the curve produced should have more area under them to determine the effectiveness of the compiler. The graph produced, indicates that class 0 and class 4 data, which means data classified under rating 1 and rating 5 were correctly classified as they projected a decent area score. Classes 2, 3 and 4 which correspond to ratings 2, 3 and 4, respectively could produce an improved area score by adding batch normalization and other pruning techniques to fetch a better result. This action is in pipeline for improving the overall score of the classifier.

8 Discussion

The main aim of this research process was to focus on the data refining process and combine different types of data; details and reviews of the mobile apps data, to determine the user rating by implementing deep learning model MLP. Although, there have been extensive research on the lines of text classification and implementation of deep learning as well as machine learning concepts for mobile apps data, not much light has been shed in the area of multiple inputs combination of variants of data using deep learning. TF-IDF does seem more appropriate and superior in some cases for text classification scenarios as it balances out the frequency of the entire text tensor under question using its inverse document frequency as opposed to count vectorizer which maintains the vector of frequency of a word occurrence in document or text corpus. It also depends on the content of the textual reviews under question, for differences in the performance scale of count vectorizer and TF-IDF. The design plan was to implement a combined approach using the best performing models of MLP for both details and reviews to attain maximum accuracy. From the table, for different epochs and batch sizes; it can be concluded that the combined functional model shows

better performance than as individual units. The MLP model gives a decent testing accuracy score of 70%, it can be improved by trying few combinations of neural nets. This, thereby, satisfies our initial research theory that this model performs well to determine the app user ratings, given various play store app metrics and user reviews. One major hurdle that was overcome was the ease with which the functional MLP neural model fits the entire data, combination of both the models, into its layers. Also, the combined model gave better results compared to individual models. Though there is still a lot of scope for improving the results, only a few variants of MLP network model was implemented. A lot of time was invested in the data extraction and data refining process. Owing to this time constraint, larger combinations of neural nets could not be implemented although it is in the pipeline. The scope for this research is huge. The various trends and key factors of mobile app stores can be exploited to unearth new findings and predictions.

9 Conclusion

Most of the prominent web-based applications could sustain their hold in top ranking charts for a long run mainly because the developers are well aware of the user requirements and user satisfaction criteria. App developers look forward to receiving credibility for their work in the form of rankings and reviews. The play store feature variables, irrespective of being largely unstructured and varied in its data format as well its content, can act as performance metrics if they can be pre-processed and well trained to determine the user-specific ratings. Although this research focused on the columns that we felt would be appropriate for analysis, we are keeping the topic open for other researchers too to experiment as per their needs. Although the DOI for the data [14] is given and the post-processing operations are irreversible, the original extracted files are provided in Github [12] for other further analysis. The future scope of the project includes testing and building the model on various other combinations of neural networks that can help in improving its performance.

References

1. Shah SR, Kaushik A (2019) Sentiment analysis on indian indigenous languages: a review on multilingual opinion mining
2. Kaushik A, Naithani S (2016) A comprehensive study of text mining approach. Int J Comput Sci Netw Secur (IJCSNS) 16(2):69
3. Zhu H, Xiong H, Ge Y, Chen E (2014) Discovery of ranking fraud for mobile apps. IEEE Trans Knowl Data Eng 27(1):74–87
4. Vijayarani S, Ilamathi MJ, Nithya M (2015) Preprocessing techniques for text mining-an overview. Int J Comput Sci Commun Netw 5(1):7–16
5. Kaushik A, Naithani S (2014) A study on sentiment analysis: methods and tools. Int J S Res (IJSR) 4:287–292

6. Kaur G, Kaushik A, Sharma S (2019) Cooking is creating emotion: a study on hinglish sentiments of youtube cookery channels using semi-supervised approach. Big Data Cogn Comput 3(3):37
7. Zhang L, Wang S, Liu B (2018) Deep learning for sentiment analysis: a survey. Wiley Interdiscip Rev Data Min Knowl Discov 8(4):e1253
8. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507
9. Zhou C, Sun C, Liu Z, Lau F (2015) A c-lstm neural network for text classification. arXiv preprint arXiv:1511.08630
10. Isa D, Lee LH, Kallimani V, Rajkumar R (2008) Text document preprocessing with the bayes formula for classification using the support vector machine. IEEE Trans Knowl Data Eng 20(9):1264–1272
11. Selvam B, Abirami S (2013) A survey on opinion mining framework. Int J Adv Res Comput Commun Eng 2(9):3544–3549
12. Venkatakrishnan S, Kaushik A (2019) Extracted csv files. https://github.com/abhishekkshk68/Play-Store-Deeplearning-Analysis/tree/master/Extracted_CSV_Files
13. Venkatakrishnan S, Kaushik A (2019) Data refining scripts. https://github.com/abhishekkshk68/Play-Store-Deeplearning-Analysis (June 2019)
14. Kaushik A, Venkatakrishnan S (2019) Google play store data. https://doi.org/10.5281/zenodo.2844022