Mobile App Category Forecasting: A Neural Network Model

(Presentation Video: https://youtu.be/MFzceQydCAM)

1. Introduction

Our project aims to (1) develop a context-aware mobile application (app) recommendation model; (2) identify key features influencing context-aware app recommendations. In simpler terms, our goal is to design a model that predicts app categories based on specific contexts. At the same time, we want to analyze feature dependencies to discern the most influential ones in category prediction.

To achieve these goals, we utilize two datasets: Frappé, which focuses on the context of app usage, and Meta, which provides general information about mobile apps. Given the classification nature of our research questions, we implement multi-class logistic regression and neural network models on the combined dataset. Both models analyze usage contexts and app properties to predict the app category that is likely to be of the most interest to the user. The models are then compared based on metrics such as accuracy, precision and recall.

Our research question holds significance within the context-aware recommendation systems (CARS) framework, which recognizes the dynamic nature of users' app preferences. Our approach addresses this dynamism by incorporating users' temporal and spatial conditions. By predicting the app category aligning most with the user's interests based on their context, we anticipate an increased likelihood of users installing the advertised app. Simultaneously, we foresee a potential decrease in aversion to app advertisements. These outcomes are not only beneficial for profit-seeking advertisers but also guarantee users discover apps aligning with their preferences, enhancing the overall user experience.

Adomavicius et al. are the very first to examine and define the concept of context-aware recommender systems [1]. Despite Baltrunas et al.'s implementation of the context-aware mobile app recommendation system Frappé in real-world scenarios, negative user feedback was received in a small-scale survey [2]. Stitini et al. conducted multi-class classification on contextual information, employing a combination of four algorithms (logistic regression, multinomial naive Bayes, linear support vector machine, and stochastic gradient descent) to enhance recommendation outcomes [3]. However, the computational complexity of combining multiple algorithms might pose a limitation, particularly with extensive datasets. Zhu et al. integrated user app, time, and location into behavior trajectories, constructing a context Voronoi diagram to model user similarity and generate app recommendations. Notably, their approach lacks consideration for app properties [4]. To address these limitations, we propose adopting a neural network architecture that incorporates both user behavior context and app properties as inputs. Our approach facilitates the effective learning of context factors, thereby efficiently creating more intelligent and tailored recommendations within recommendation systems.

Our workflow unfolds in the following steps. First, we perform exploratory data analysis (EDA) to understand the combined dataset. During this phase, we address issues such as revising unclear data structures and inappropriate formats, handling missing values, and filtering out irrelevant and unwieldy features. Next, due to the abundance of non-numeric features, we employ feature engineering to transform existing features and utilize visualization techniques to discern correlations between different features. Then, considering the classification nature of our task, we implement multi-class logistic regression and compare its performance with a neural network. Lastly, based on cross-validation results for both models, we select the neural network as our modeling approach to predict the app category.

2. Description of Data

2.1 Data Summary

In this project, we incorporate two datasets: Frappé and Meta. Frappé is about app usage, with each row representing an app usage context. It consists of 96,203 entries by 957 users for 4,082 apps. It was collected from a large-scale market deployment of Frappé, an Android app. A detailed overview of the features included in this dataset can be found in Table 1.

Table 1: Overview of the features in Frappé dataset

Column Name	Feature Information
user	Each numeric value represents a user
item	Each numeric value represents an app
cnt	Number of times an app was used in a given context by a given user.
daytime	Time of the day: morning (6am to 12am), afternoon (12am to 6pm), evening (6pm to 12pm), or night (12pm to 6am)
weekday	Day of the week: monday, tuesday, wednesday, thursday, friday, saturday, or sunday.
isweekend	Period of the week: weekend or workday (i.e. weekday)
homework	Location: home, work, or other.
cost	Cost of an app: free or paid.
weather	Weather condition: sunny, cloudy, foggy, windy, drizzle, rainy, stormy, sleet, or snowy.
country	Name of the country where the user is currently located.
city	Likely to be the distance from the user location to the center of a major city.

Meta dataset is about apps. It consists of 4,082 entries, each representing a unique app. A detailed overview of the features included in this dataset can be found in Table 2.

Table 2: Overview of the features in Meta dataset

Column Name	Feature Information
item	Each numeric value represents an application
package	The application package
category	The application category
downloads	The range of number of downloads for the app
developer	Developer of the app
icon	The link to the image of the app icon
language	The language of the app
description	The description of the app
name	The name of the app
price	The price of the app
rating	The rating of the app
short desc	The short description of the app

After combining the two datasets, our data granularity is at the user-application level. Each entry in the data frame represents a specific user using a specific app and various features are described about the user-application interaction. Each entry captures the characteristics of the user using a specific app under certain conditions.

2.2 Exploratory Data Analysis

2.2.1 Data Sampling and Collection

When studying Frappé, we find certain selection biases inherent in the dataset. Frappé dataset comes from a study on Frappé app adoption involving both large-scale and small-scale approaches. However, it should be noted that both approaches only target Android users as Frappé is only exclusively available on Google Play [2]. Furthermore, when visualizing the distribution of entries per country (Figure 1), discrepancies

arise between the recorded numbers and the actual population sizes of each country. This implies that certain countries may be overlooked in the study, while others are disproportionately emphasized.

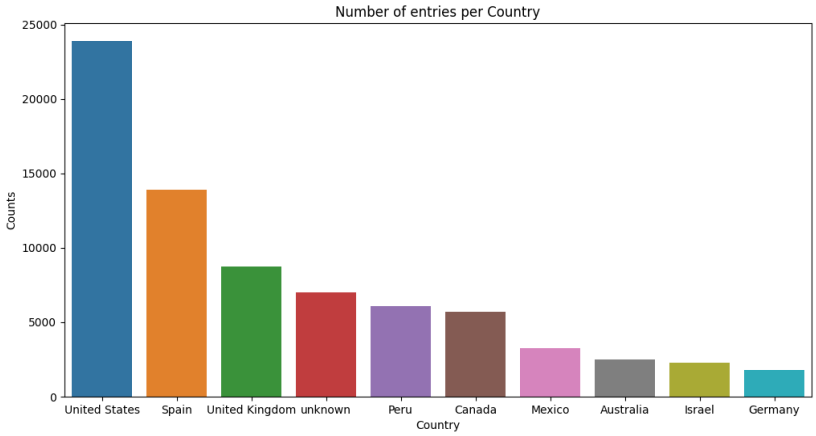

Figure 1: The Distribution of Number of Entries per Country

2.2.2 Data Cleaning

(1) Missing Values

First, we check for missing and invalid entries and identify the label ‘unknown’ for these entries. To further illustrate this, we visualize the percentage of 'unknown' entries in each column (see Figure 2). Remarkably, the 'homework' column exhibits the highest prevalence of 'unknown' values within the combined dataset, constituting 78.65% of the total observations. Given the high percentage of missing values, we eliminate ‘homework’ from our features.

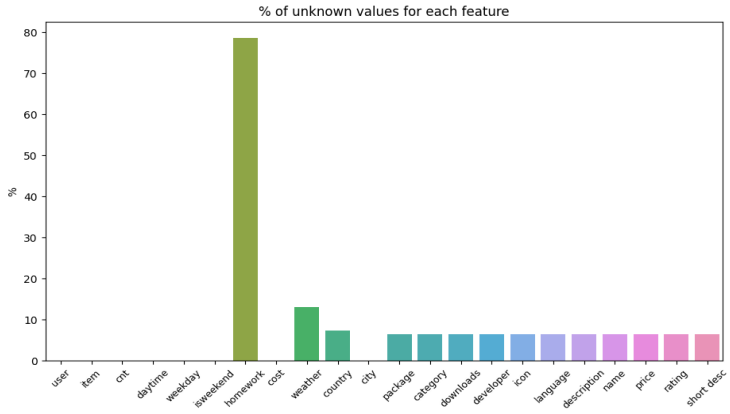

Figure 2: The Distribution of Percentage of Unknown Values for Each Feature

(2) Inappropriate Data Types

We examine the data type of each column and find most features to be of string type, except for ‘user’, ‘item’, ‘cnt’ and ‘city’ being of int type. However, we notice multiple quantitative variables to be formatted as string type. For instance, values in the ‘price’ column display ‘Free’ or ‘$’ symbol followed by a string-type decimal number. There are also a few other monetary symbols involved. To handle this, we assign 0 to the entries labeled with ‘Free’ values. We also design a function with a regex expression to retrieve the numerical value of price. With these efforts, we convert ‘price’ into ‘price_num’, a new feature containing app prices in numerical form. Similarly, we create a column ‘rating_num’ with numerical values of app ratings.

In addition, we observe the ‘downloads’ column to be characterized as a range of the number of downloads, such as ‘5000-10000’, containing useful numeric information but unwieldy for modeling input. To convert it into a numeric value while preserving the original range attribute, we substitute the range with the mean value of each range. Furthermore, we transform the notation for binary feature ‘isweekend’ from workday-weekend to 0-1.

(3) Data Distribution and Outliers

After converting the non-numeric features into numeric ones, we then proceed to plot the distribution of each feature. For instance, we plot the overall distribution of all ‘rating’ values (see Figure 3) and the category-specific distributions for the top 10 popular categories (see Figure 4).

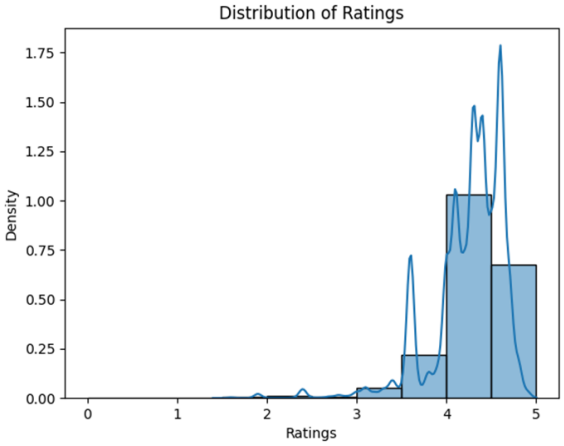

Figure 3: The Distribution of Ratings

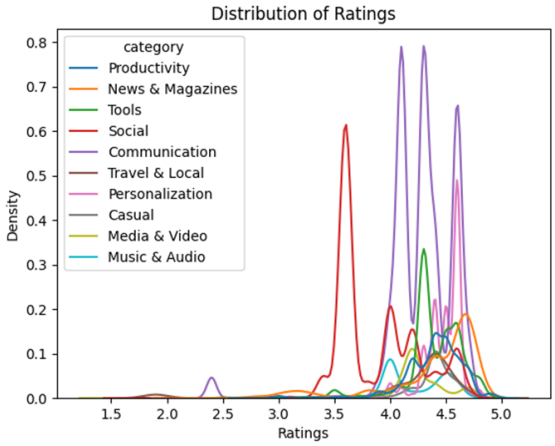

Figure 4: The Distribution of Ratings for Popular Categories

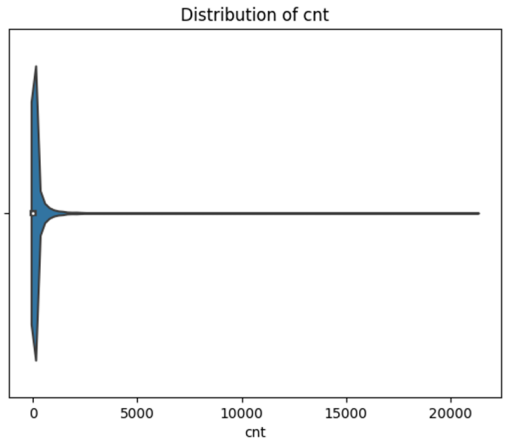

Figure 5: The Distribution of Count

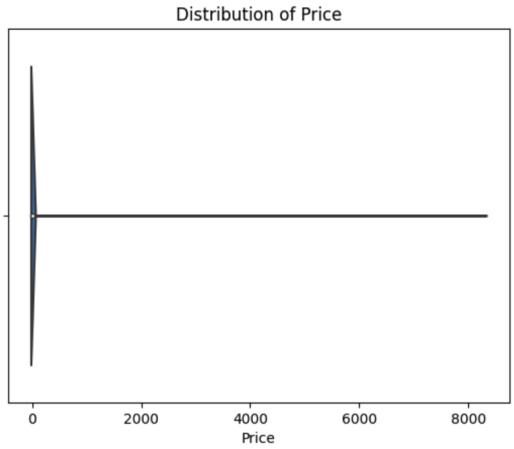

Figure 6: The Distribution of Price

Upon examining the distributions, we find outliers in ‘cnt’ and ‘price’ (see Figure 5,6). Specifically, the distribution of ‘cnt’ reveals extraordinarily huge numbers like 28752, 21262. Such numbers of app usage seems unreasonable in a day for a given context. Therefore, we drop entries greater than 3000. Moreover, the feature ‘price’ also contains unreasonable values like $8313 for an application. Such entries are also removed from the dataset.

2.2.3 Data Transformation

Plotting the distribution of each feature, we examine if data transformation is needed according to Tukey-Mosteller Bulge Diagram.

Based on the right-skewness in the distribution of ‘cnt’, we perform a log transformation on it. In Figure 7,8, the distributions of ‘cnt’ and ‘log_cnt’ are shown in violin plots, which evidently shows that the log transformation more effectively captures the distribution of the ‘cnt’ data. Similarly, we apply the log transformation on ‘price’.

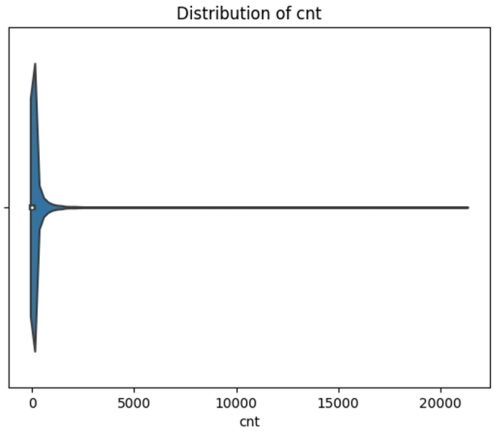

Figure 7: The distribution of ‘cnt’

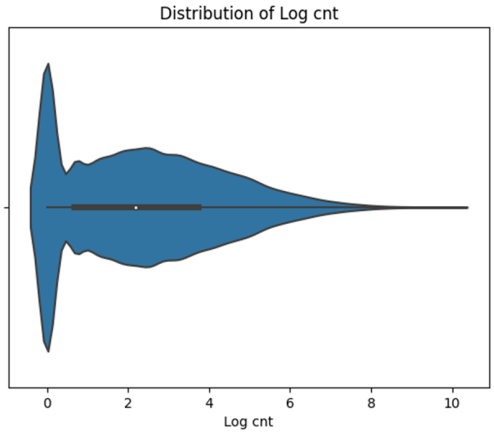

Figure 8: The distributions of ‘log_cnt’

2.3 Feature Engineering

In this section, we summarize the feature engineering techniques employed during the data cleaning phase (refer to Section 2.2) and introduce the application of one-hot encoding to categorical features.

Our prior feature engineering efforts center on: (1) extracting numeric values from the string representation of quantitative variables; (2) using log transformation to better capture the data

distribution. Moving forward, we use one-hot-encoding to transform categorical features ‘daytime’, ‘weekday’, and ‘weather’ into numeric ones.

Through these feature engineering efforts, we successfully convert all non-numeric features into numeric representations, except for the target feature that we aim to predict. This prepares the data for the subsequent correlation analysis and feature selection phase.

2.4 Data Correlations

2.4.1 Pearson Correlation

To understand the correlation between different variables, we first utilize Pearson correlation matrix and to avoid multicollinearity.

In particular, we utilize seaborn to generate Pearson Correlation plots for specific categories (refer to Figure 9). Notably, the correlation attributes in these plots for different categories exhibit similarities. ‘weather_sleet’ is found to be almost empty, with only one entry being 1. Additionally, both ‘weekday-saturday’ and ‘weekday-sunday’ show a strong correlation with ‘isweekend’. Subsequent correlation plots for other datasets yield consistent results, leading us to eliminate both ‘weather_sleet’ and ‘isweekend’ from our features.

2.4.2 Mutual Information Score

Alongside the correlation matrix, we compute the mutual information classification metric, assessing the dependency between random variables (see Figure 10). This metric further guides our feature selection by favoring features with the highest predictive information. Specifically, in our case, it quantifies how much information a feature provides about the class variable. A higher score indicates greater information that the feature carries about the target variable.

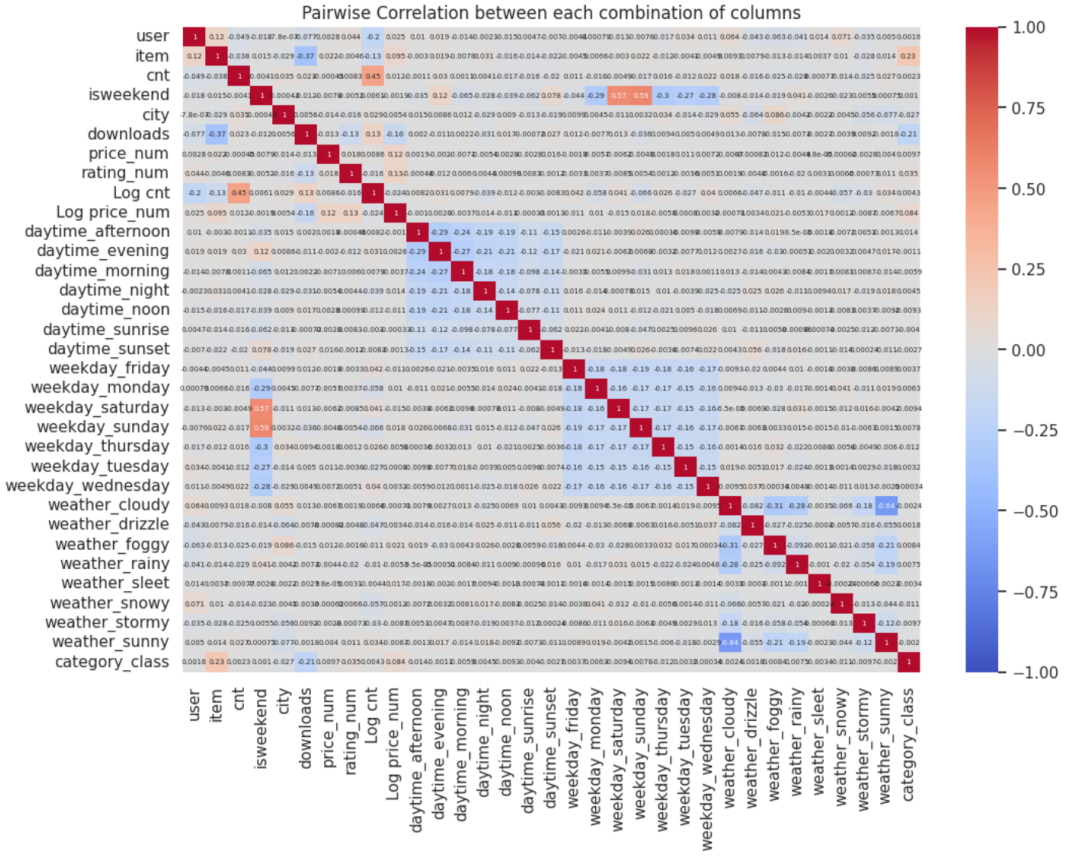

Figure 9: Pearson Correlation Plot for One Category

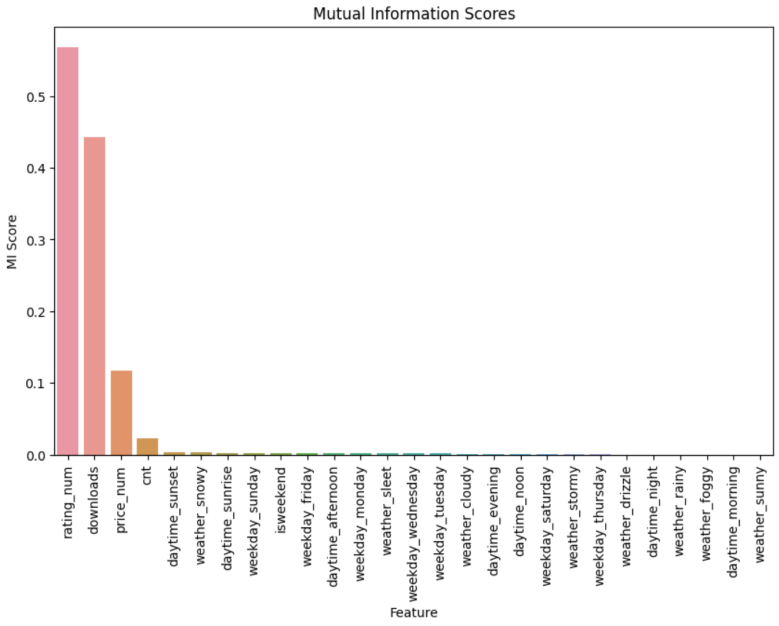

Figure 10: The Distribution of Mutual Information Score for All Features

3. Methodology

In this section, recognizing the inherently classification-oriented nature of our objective, we employ two distinct models: multi-class logistic regression and a neural network, for the purpose of predicting among 30 different categories. Subsequently, we conduct a comprehensive comparison of both models, evaluating their performance based on metrics such as accuracy, precision, and recall. The outcome of this comparison guides us in selecting the most suitable model for our predictive task.

3.1 Multi-class Logistic Regression

To predict among 30 different app categories, one natural choice is to employ a multi-class logistic regression (or softmax regression) model.

We apply the LogisticRegression function from the scikit-learn package. As seen in Section 2.3, the features highly correlated with the class variable include: ‘rating_num’, ‘price_num’, ‘downloads’, ‘cnt’, ‘weather_cloudy’, ‘weather_sunny’, ‘weekday_wednesday’, ‘weekday_sunday’, ‘daytime_night’ and ‘daytime_morning’. Therefore, the designed matrix X contains these 10 features as the foundation to build the logistic regression model for the prediction task. The solver chosen for optimization is ‘lbfgs’, which stands for Limited-memory Broyden–Fletcher–Goldfarb–Shanno, and the regularization parameter C is set to 2.

To quantify the performance of the multi-class classification, we compute the individual values of accuracy, precision and recall for each class (see Table 7). We also calculate the overall metrics and find the overall accuracy of the model to be 0.39. Due to the unsatisfactory results in the overall model performance, we opt to explore alternative models, such as neural networks.

3.2 Neural Network

In search of better overall performance, we choose a sequential neural network to perform the classification task based on different contexts.

The features utilized in our multi-class logistic regression analysis are incorporated into the construction of the neural network model. To ensure uniform scaling across all variables, we employ standardization using the 'StandardScaler' from scikit-learn. Additionally, we conduct one-hot encoding for the target variable 'category_class' target, which contains 30 categories.

The neural network itself consists of four to five layers including an input layer with 10 neurons (matching the number of features), two or three hidden layers with number of neurons between 10 and 30, and an output layer with 30 neurons representing the categorical output with the app categories. Given the size of the dataset and the complexity of the relationship, we opt for two hidden layers to effectively capture the underlying patterns in the data. The number of neurons in each hidden layer is selected based on the idea that the number of neurons lies between the number of input neurons (10) and the number of output neurons (30), therefore, 20 and 15 neurons for the two hidden layers respectively is a reasonable number for our model. The activation function used is the Exponential Linear Unit which could be beneficial to capture both positive and negative information.

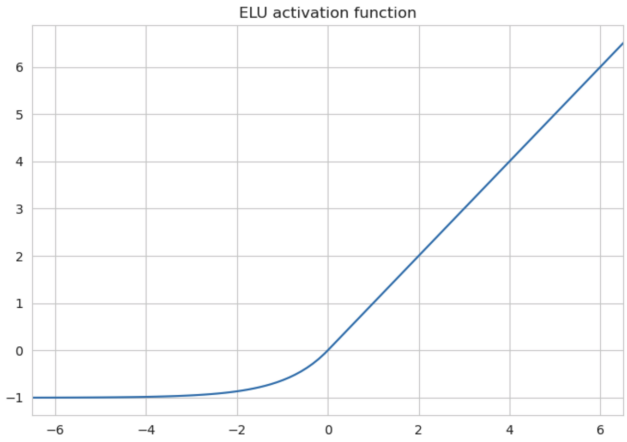

Figure 11: Exponential Linear Unit

The output layer uses a softmax activation function which is suitable for multi-class classification. It converts the raw output into probabilities. The category associated with the highest probability in each case is the final answer for the category. Weights are initialized using a random uniform distribution between -0.5 and 0.5. The model is compiled using the categorical cross-entropy loss function which already implements cross-validation into the data, and Adam optimizer. Here is a summary of the neural network including the layers, the number of neurons and the total parameters to be optimized for each layer that are going to be used for the model.

Table 3: Hyperparameter Choices in Neural Network Model

Hyperparameter	**Values**
Input neurons	10
Hidden layers	[2,3]
Neurons per hidden layer	[10,30]
Output neurons	30
Activation function	ELU, RELU, SELU
Output activation	Softmax
Loss function	Cross Entropy

As for the model training, the dataset is split into training and validation using 25% of test size. Early stopping is also implemented as a callback during training which can prevent overfitting as it stops the training when the loss stops improving. The model is trained for 1000 epochs. The best scenario is reached at epoch 993 where the smallest loss and highest accuracy, recall and precision on the validation set are achieved.

In order to find the best hyperparameters, we are going to perform the process of hyperparameters tuning to select the optimal values of NN layers, activation functions, minimum and maximum weight and bias values and learning rate for the Adam optimizer that result in higher accuracy, precision and recall for the training and validation datasets.

Table 4: Hyperparameter summary for Cases 1 - 5

Case	1	2	3	4	5
NN Layers	10, **20, 15**, 30	10, **20, 25**, 30	10, **25, 28**, 30	10, **15, 10, 15**, 30	10, **12, 18, 15**, 30
Act. Funct.	ELU	RELU	ELU	SELU	ELU
W,b minval	-0.5	-0.9	-1	-0.3	-0.5
W,b maxval	0.5	0.9	1	0.3	0.5
Learning rate	0.01	0.015	0.008	0.005	0.01

Table 5: Training Results: Loss and metrics

Cases	Loss	Accuracy	Precision	Recall
1	1.2558	0.6004	0.8238	0.4594
2	2.2196	0.2857	0.7974	0.1648
3	1.3264	0.5881	0.7923	0.4713
4	1.3380	0.5908	0.8010	0.4611
5	1.3733	0.5827	0.7937	0.4561

Table 6: Validation Results: Loss and metrics

Cases	Loss	Accuracy	Precision	Recall
1	**1.3454**	**0.5977**	**0.8229**	**0.4515**
2	2.2052	0.2739	0.8067	0.1709
3	1.3479	0.5858	0.7856	0.4723
4	1.3493	0.5831	0.8166	0.4489
5	1.3700	0.5912	0.8171	0.4384

The best result is achieved for case 1 with a validation loss of 1.3454. The following hyperparameter choices are: ELU as activation function, learning rate of 0.01 for a total of 1000 iterations, initializer of -0.5 and 0.5 and a neuron structure of 10, 20, 15, 30.

4. Summary of Results

Overall, after applying two distinct models, multi-class logistic regression and a neural network, to explore the classification task for the applications, we evaluate the performance of both methods and conduct a comprehensive comparison. We compared metrics such as accuracy, precision, and recall to see which method is more efficient and suitable for classification.

In terms of performance of multi-class logistic regression, we compute the metrics including accuracy, precision, recall and f-1 score for both each individual class and the overall dataset. The results are shown in Table 3. We can see that except for the categories like Communication and Social, the majority of the categories have very low precision, recall and f-1 scores. Many of them yield 0 in all the metrics. One possible explanation for the poor performance is the limited size and the unbalanced distribution of the dataset. Only Communication and Social categories have the relatively high "support" metric (the count of true entries belonging to each class), while the many categories only have less than a thousand supports. The percentage of records belonging to each category experiences a great variation across the categories, and the small size of the dataset makes it worse. As for overall performance, the model also can only yield weighted average precision, recall and f-1 score of 29%, 39%, and 29%. We also calculate the overall accuracy of the model to be 0.39. None of the performances give a satisfactory result in classification. We can conclude that multi-class logistic regression is not the optimal choice for this dataset.

Classes	Precision	Recall	F1 score	Support
Arcade & Action	0	0	0	1199
Books & Reference	0.33	0	0	810
Brain & Puzzle	0	0	0	1489
Business	0	0	0	480
Cards & Casino	0	0	0	100
Casual	0	0	0	1535
Comics	0	0	0	31
Communication	0.46	0.92	0.61	17542
Education	0	0	0	287
Entertainment	0	0	0	908
Finance	0	0	0	480
Health & Fitness	0	0	0	586
Libraries & Demo	0	0	0	9
Lifestyle	0	0	0	662
Media & Video	0	0	0	1853
Medical	0	0	0	14
Music & Audio	0	0	0	1818
News & Magazines	0.15	0.5	0.23	4679
Personalization	0.17	0.15	0.16	4459
Photography	0	0	0	526
Productivity	0.03	0	0	3731
Racing	0	0	0	125
Shopping	0	0	0	828
Social	0.67	0.54	0.6	9847
Sports	0	0	0	307
Sports Games	0	0	0	133
Tools	0.36	0	0	5210
Transportation	0	0	0	259
Travel & Local	0	0	0	2380
Weather	0	0	0	381
Total	0.29	0.39	0.29	62668

Table 7: Classification Report for Multi-class Linear Regression

For the neural network model, as discussed above, we use the best hyperparameters (ELU as activation function, learning rate of 0.01 for a total of 1000 iterations, initializer of -0.5 and 0.5 and a neuron structure of 10, 20, 15, 30) to adjust the model so it can have the best performance. The optimal model can yield 60% of prediction accuracy, 45% of recall and 82% of precision, which is much higher than the result of the Multiclass Logistic Regression model. The result shows that even with this unbalanced and small-size dataset, the model can still perform well in predicting which category out of all 30 categories

that one application belongs to. We can expect the model will have even more promising performance for the dataset that has more entries.

Dataset	Accuracy	Precision	Recall
Training Set	0.60	0.82	0.45
Validation Set	0.60	0.82	0.46
Overall Dataset	0.60	0.82	0.45

Table 8: Classification Report for Neural Network

To conclude, based on the comparison of metrics including accuracy, precision, recall (shown on Figure 12), the Neural Network Model is the better model to perform classification. This can effectively classify one application to the category it belongs to based on context features of one application usage entry.

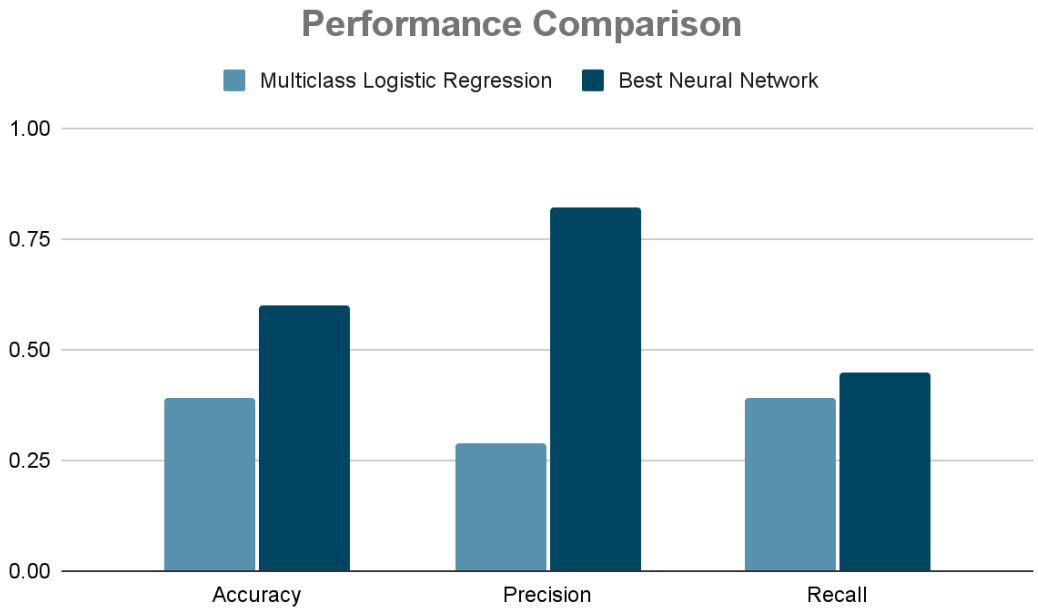

Figure 12: Metric comparison between MLR and NN

5. Discussion

In evaluating our approach, we acknowledge certain limitations associated with the Frappé dataset. The dataset's exclusive inclusion of Android users introduces an inherent demographic imbalance, limiting the generalizability of our study to users on different systems. Additionally, the dataset reveals a geographical imbalance, with significant variations in user records among countries, despite their population sizes. This imbalance could potentially amplify regional biases in user context, resulting in poor model performance for countries with limited or no records. Another limitation is the distribution imbalance of our target variable in the Frappé. Due the uneven distribution of application categories, it is difficult to predict those categories that have low data entries, and hence impact the overall model performance. Though the actual usage of applications in each category is not the same, a more evenly distributed dataset across all categories will enhance the model's accuracy and yield better performance in classification. Moreover, though the dataset has features like 'weather', 'daytime', 'weekday' that are related to the context of the usage, these might not contain enough information to describe the context that the users were in.

We believe that, with more features like 'location', 'activity' (walking, running, or driving, etc.), 'preferences/history' (frequency or usage pattern), the dataset may reflect a more comprehensive presentation on the usage context and further be used to train a more effective and precise model. Lastly, the dataset is overly disaggregated in terms of categories, and some of the categories are ambiguous or overlapping. For example, the category 'Books & Reference' might overlap with 'News & Magazines' or 'Comics'. This granularity may be helpful to elaborately depict the usage situation, but the large number of categories and the similarity across multiple categories bring difficulty in training a classification model.

Regarding methodology, while the neural network model can achieve promising results in classification, we notice that, among all the features we use to train the model, features like ‘rating’, ‘price’, ‘downloads’, ‘cnt’ are actually not context-related. They are more representative of the application attributes instead of the user’s habits or environment. From Figure 10, we see that, compared to the context features such as ‘daytime’, ‘weather’, and ‘weekday’, the non-context features have higher mutual information scores. Therefore, the non-context features may introduce a bias towards application popularity in the market.

One surprisingly interesting discovery we get during the model-building process is how significant the ratings of the application is to predict the categories. The data shows the subjective ‘rating’ feature has higher predictive information than the objective features like ‘downloads’, ‘price’, ‘cnt’.

For future extension, one improvement method is to bring in a more comprehensive dataset for classification. As discussed above, a dataset with less bias, more context information and well-defined categories granularity can make the model more accurate and productive. Also, built on the surprising discovery of the significance of rating, we consider collecting the rating from different platforms to further reduce the bias in this informative feature. Regarding the methodology, we hope to explore deep neural networks, allowing for a more comprehensive exploration of intricate relationships within the data and potentially refining the accuracy of our recommendations.

References

[1] Adomavicius, Gediminas, and Alexander Tuzhilin. "Context-aware recommender systems." Recommender systems handbook. Boston, MA: Springer US, 2010. 217-253.

[2] Baltrunas, Linas, et al. "Frappé: Understanding the usage and perception of mobile app recommendations in-the-wild." arXiv preprint arXiv:1505.03014 (2015).Baltrunas, L., Church, K., Karatzoglou, A., & Oliver, N. (2015). Frappé: Understanding the usage and perception of mobile app recommendations in-the-wild. arXiv preprint arXiv:1505.03014.

[3] Stitini, Oumaima, Soulaimane Kaloun, and Omar Bencharef. "Integrating contextual information into multi-class classification to improve the context-aware recommendation." Procedia Computer Science 198 (2022): 311-316.

[4] Zhu, Ke, et al. "A novel context-aware mobile application recommendation approach based on users behavior trajectories." IEEE Access 9 (2020): 1362-1375.