Predicting why the customer gave 1-star review - Prodigy Education

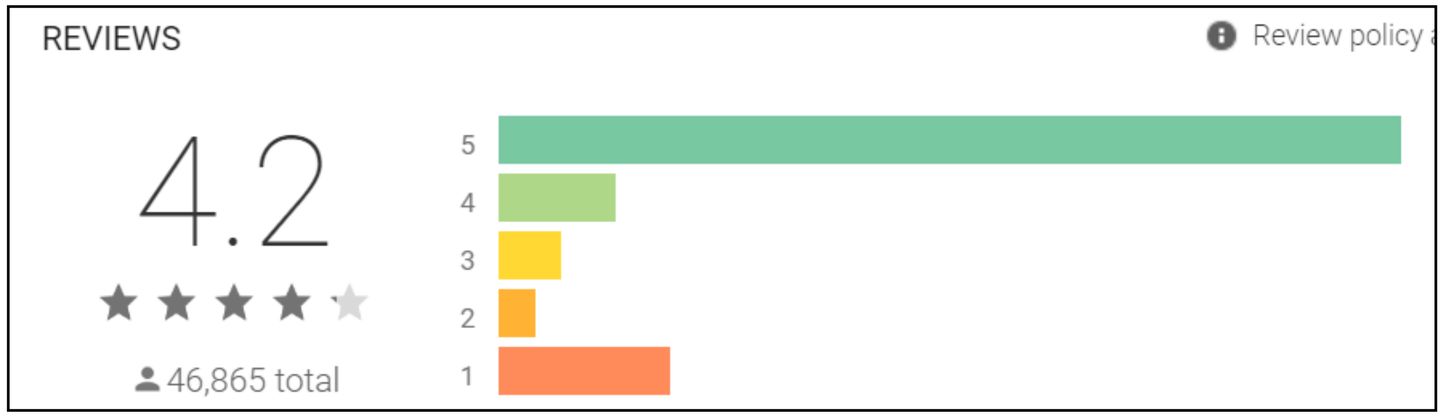
Building a machine learning model that predicts the issue from reviews

Raam Regunathan

GitHub Link

Prodigy Education review classifier

Problem Statement: How can Prodigy Education (app for learning math online) achieve a 5 star rating on google play store by identifying why customers are giving low ratings to their app ?



Over the past 12 months prodigy education has 500 "1 star" reviews on Google play store.

Sample of 1 star reviews:

THE WORST. These questions are too HARD

No more membership

It's way easier to sign in with a computer than on a phone.

GitHub Link

Constraints & success criteria

Constraints of building a classification model:

1. Customer posts the review but it is not grouped into different labels based on the issue. So, data labels need to be added.
2. Need to avoid very old versions as these issues might be already resolved.

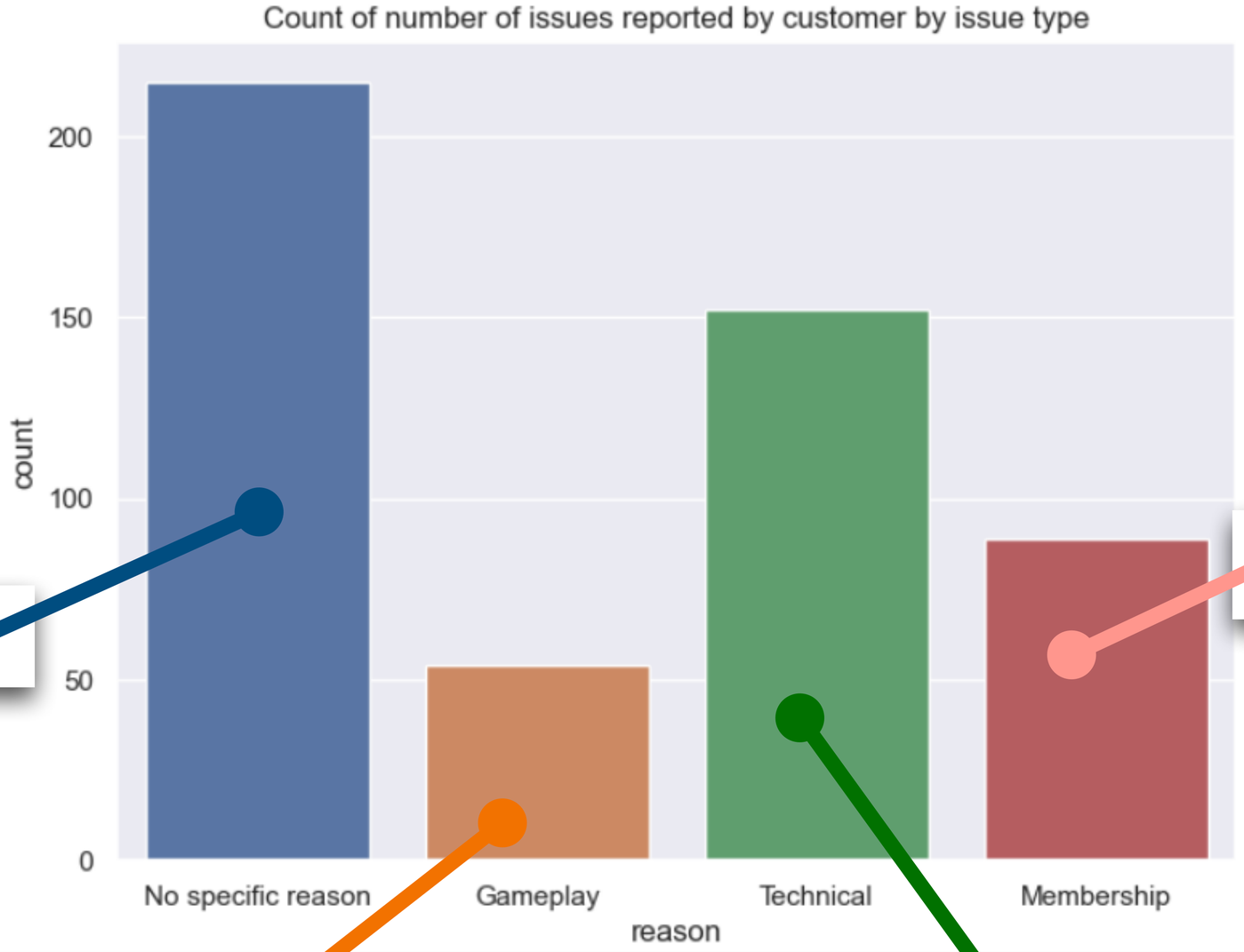
Success Criteria:

Success for this project involves in coming up with a model to predict why the customer is complaining and send the issues to specific teams.

GitHub Link

Creating labels for the issues

4 Labels are defined and each review is manually labelled. The distribution is as follows:



Terrible

No more membership

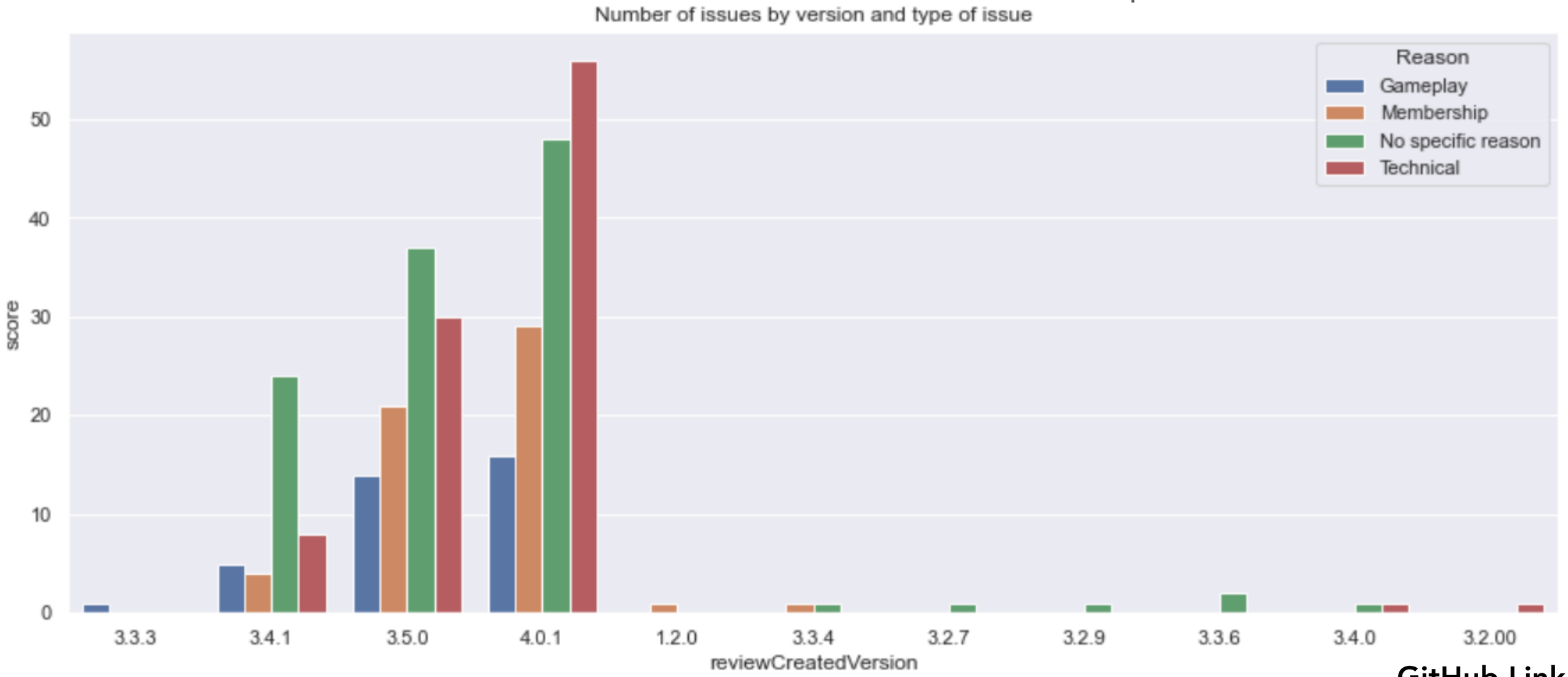
These questions are too HARD

It's way easier to sign in with a computer than on a phone.

Distribution of issues across versions

The play store data also gives the versions for which the reviews were given.

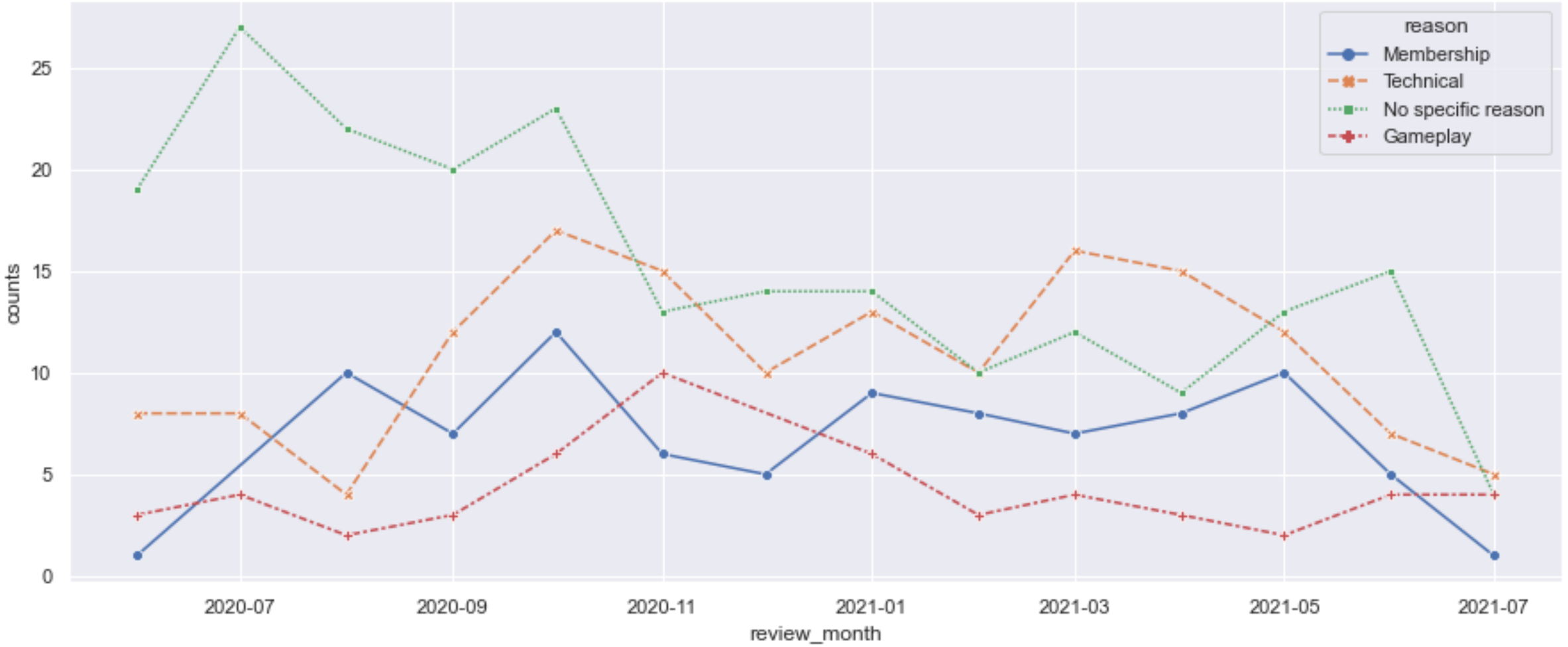
The distribution of issues across versions follows a similar pattern.



GitHub Link

Distribution of issues over time

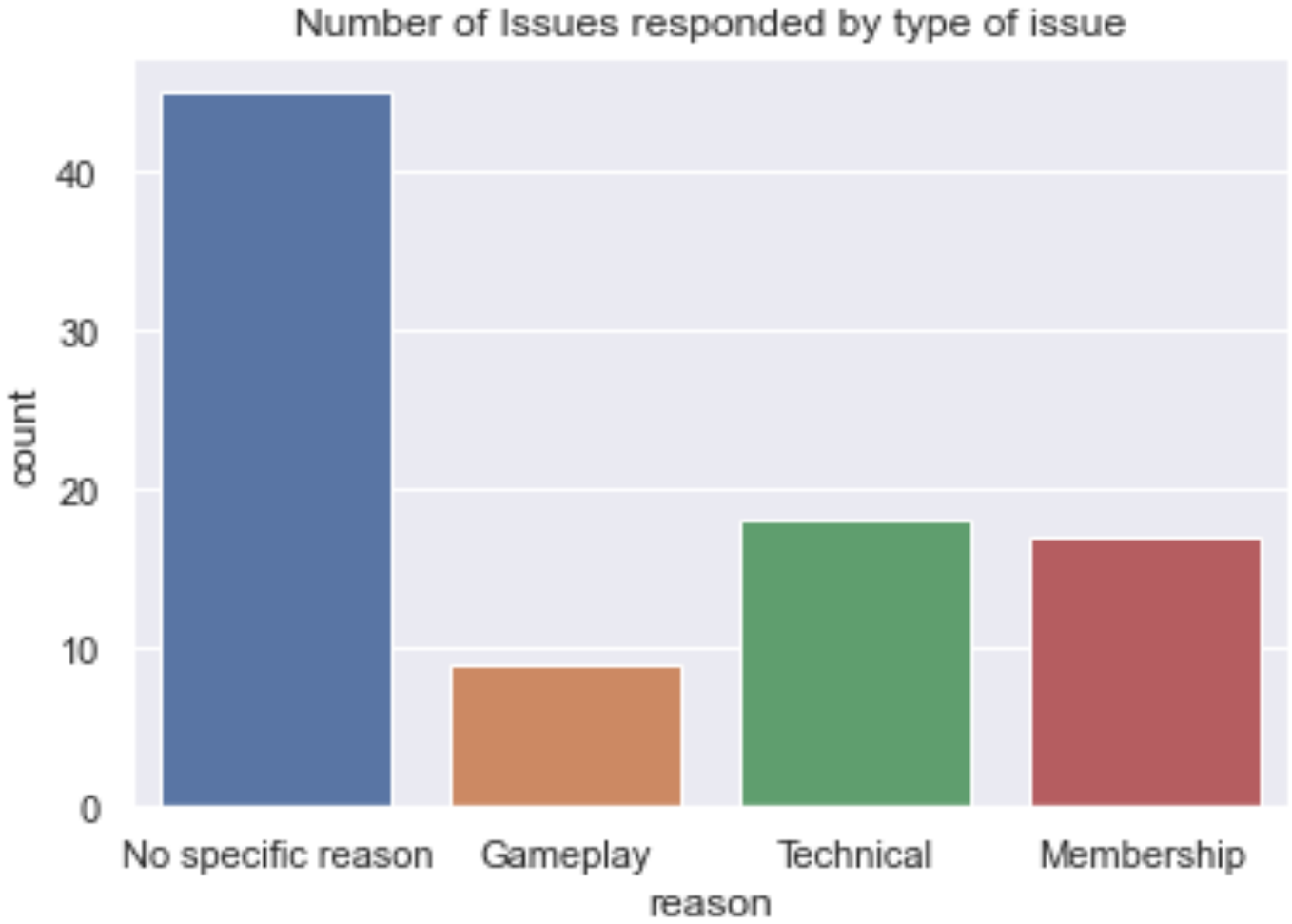
This is the distribution of different issues over time. The data is aggregated over each month. There is no particular trend among meaningful issues.



GitHub Link

Distribution of responses from developer

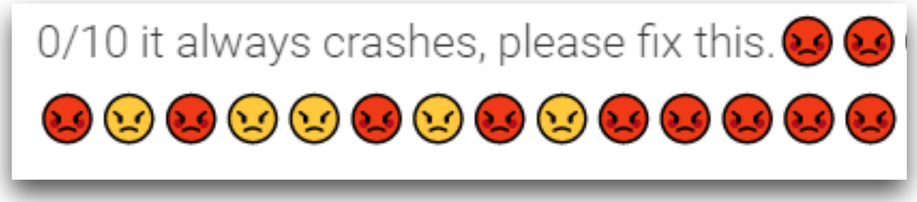
This is the distribution of responses for the issues from the developer. Out of 500 issues, only about 100 have responses. So, **only 20% of people get any communication.**



GitHub Link

Data Preprocessing

The raw review data has emojis, formatting issues and non-ascii values. This is cleaned up by following process:



Raw data

↓

Convert to lower case, only keep alphabets

↓

Lemmatize words to convert to root forms of word

↓

Identify TF-IDF scores to convert words into features

↓

Data ready for modeling

GitHub Link

Modeling

5 different models were used for classification and a Randomized search was done to find the best hyperparamter for each model type.

Decision metric: Matthew's Correlation Coefficient

S.No	Model	Training score	Testing score
1	**Logistic Regression**	0.622	0.670
2	**Multinomial Naive Bayes**	0.584	0.569
3	**Random Forest**	0.624	0.615
4	**XGBoost**	0.615	0.519
5	**Neural Network**	0.959	0.705

WINNER! →

GitHub Link

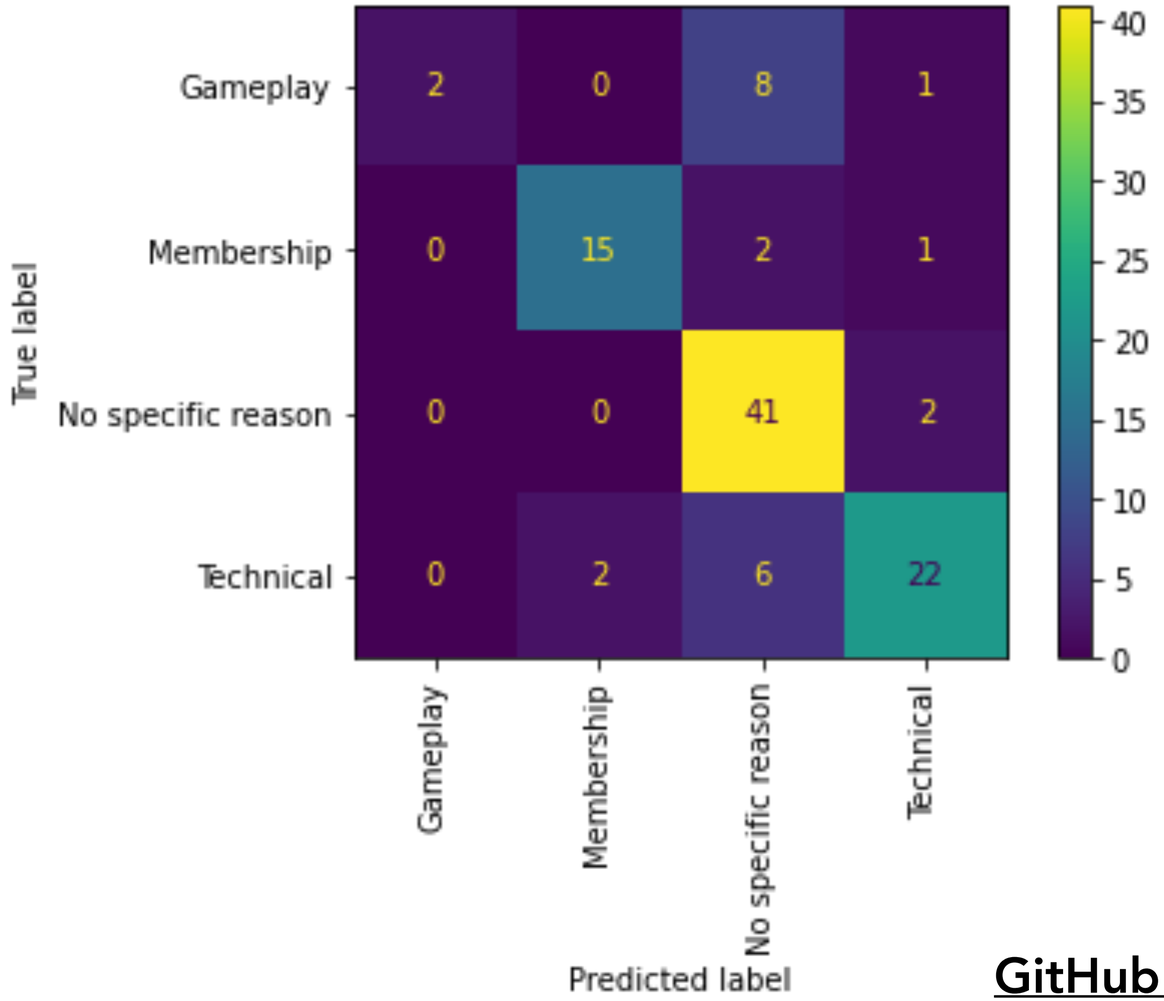
Confusion matrix for final model

Since the neural network has the highest score, it is selected for classifying the reviews into selective issues.

The confusion matrix for the final model is shown here

Matthew's Correlation Coefficient:
0.705

Accuracy: 80/102= 78%



GitHub Link

Business impact

Thus, using this machine learning model, the issues can be automatically classified into different categories. This helps to:

1. Automatically classify the customer issues into different segments and send pre-programmed responses
2. Monitor what type of issue is on the rise in real time
3. Send issues to respective teams for resolution
4. Use this as a means to predict why customers churn

GitHub Link

Summary

This final model can be run frequently to see how the app is being received by the customers. The major challenge is that new issues might be misclassified so model might need to reviewed and updated periodically.

GitHub Link

Thank You