Internal report

Saurav Chowdhury, Sirine Chahma, Reiko Okamoto, Tani Barasch 17/06/2020

Purpose

This report serves four purposes: (1) help individuals navigate our GitHub repository; (2) discuss our analysis and findings; (3) report the performances of selected models: and (4) provide recommendations on how to move forward.

Description of the data

Exploratory data analysis was performed at the beginning of the project. The code for the exploratory data analysis can be found in the `/eda` directory.

The key observations are as follows. First, the marginal distribution of the `unacast_session_count` has a positive skew (Figure 1). Second, Figure 2 shows the sparsity of the data. Many of the features derived from data collected through the app are sparse. Third, missing values were present in both the explanatory and response variables. The presence of missing values across the explanatory variables is summarized in Figure 3. The next two histograms illustrate the distribution of missing `unacast_session_count`. As shown in Figure 4, there are a handful of playgrounds that are missing the target value for over half of the months. Figure 5 suggests that there is a temporal pattern in the distribution of missing target values. `unacast_session_count` is more likely to be missing in the winter months; notably, the target value for January 2018 is missing for many playgrounds.

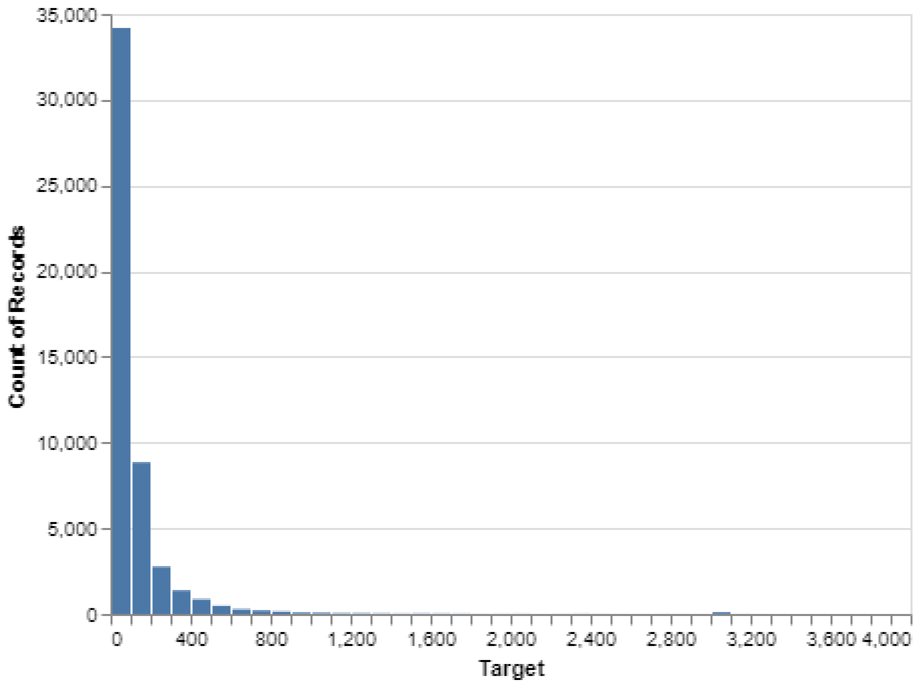


Figure 1. Marginal distribution of the target variable.

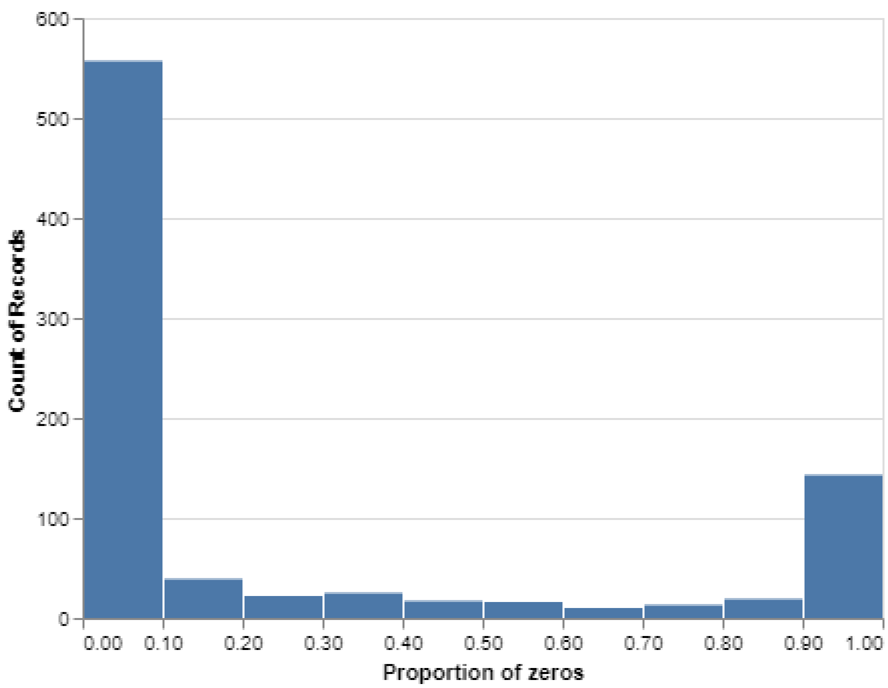


Figure 2. Counts of features by proportion of zeros.

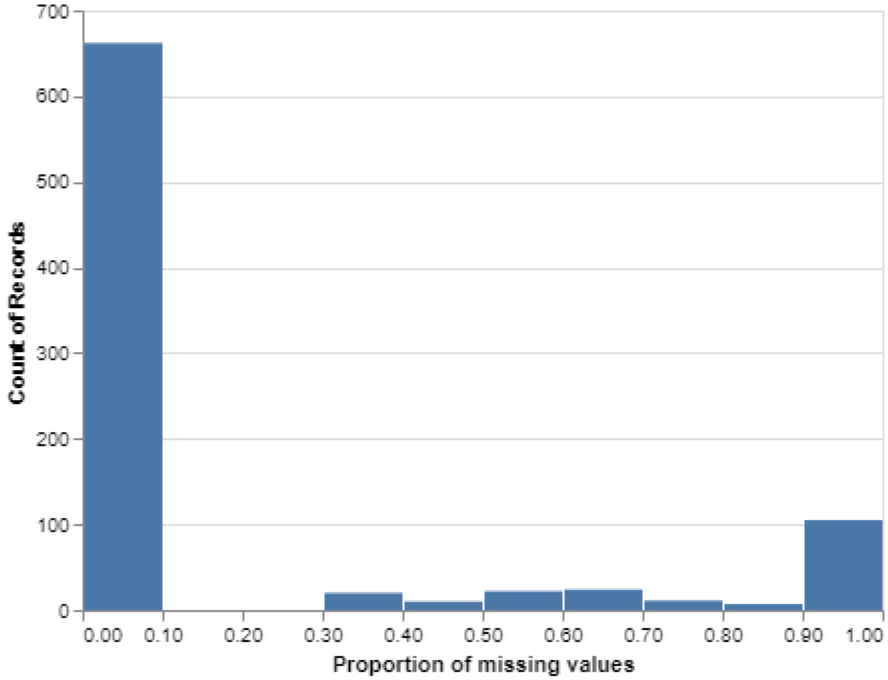


Figure 3. Counts of features by proportion of missing values

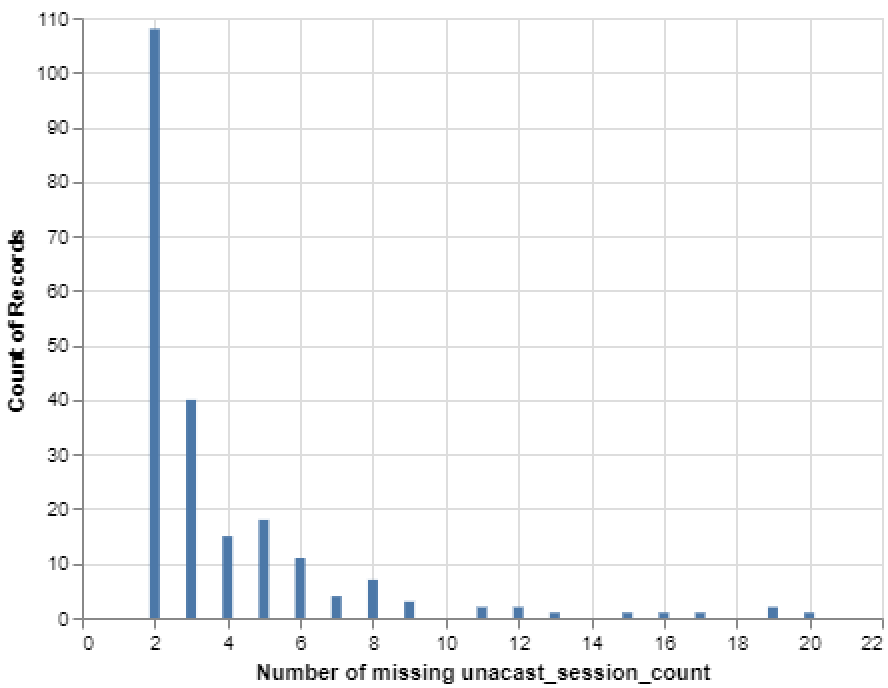


Figure 4. Counts of playgrounds by number of missing target values. Playgrounds missing less than two observations are excluded from this plot for readability.

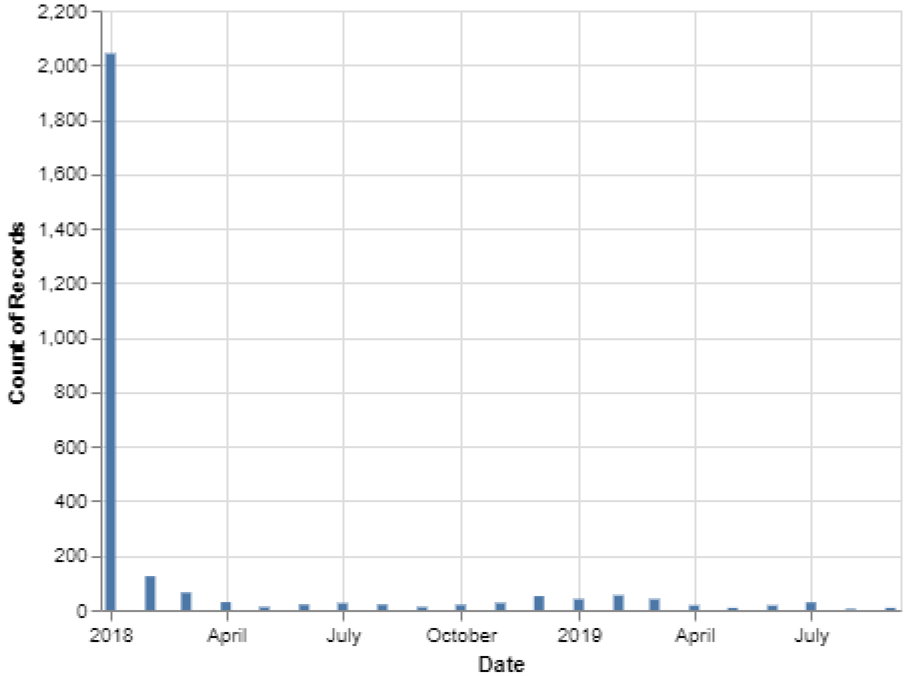


Figure 5. Distribution of missing target values across months.

Exploratory data analysis also revealed the possible duplication of information in the dataset. For example, among the features derived from the U.S. Census, information related to sex is encoded in several places (i.e. “Sex by Age”, “Sex by Marital Status”, “Sex by School Enrollment”). Upon closer inspection of the dataset, we also discovered that some columns are merely the sum of others. The pattern can be observed among the U.S. Census-related features; for example, `B13016e2` (“Women Who Had a Birth by Age: Total”) is the sum of `B13016e3` through `B13016e9`. On a similar note, we also noticed that some columns can be obtained by linear transformations of other columns. `streets_per_node_counts_*` and `streets_per_node_proportion_*` are a great example of this because one is just a normalized version of the other.

Rationale behind the output

At first, we considered creating regression models that output either a confidence interval or probability distribution. Since the marginal distribution of `unacast_session_count` is skewed, we thought that these kinds of estimates would be more robust to outliers than a single-value prediction. However, given that the end users (i.e. playground owners and managers) are more comfortable working with single-value predictions than estimates that incorporate uncertainty, we chose to build models that predict either the mean or median `unacast_session_count`. The performance of these models were evaluated using the root mean squared error (RMSE) and mean absolute error (MAE), respectively. Quantile regression was also pursued here because the median is less sensitive to extreme values than the mean.

Rationale behind the data split

The dataset consists of 24 monthly observations for 2506 Biba-enabled playgrounds in the United States. The dates ranged from January 2018 to December 2019. Data from January 2018 were excluded from our analysis because many observations are missing the target value for this month, as shown in Figure 5. Therefore, our training set consisted of observations from February 2018 through September 2019. The observations from October 2019 to December 2019 were aside for model testing. This strategy enabled us to avoid data leakage when pursuing a time series approach.

Analysis with the old dataset

The data used in this iteration of modeling can be found here. On Google Drive, it is saved as `playground_stats.csv`. Since the focus of this iteration was not on preprocessing, rows missing the target value were dropped and missing values in the explanatory variables were imputed with zeros. `/src/preprocessing_old.py` contains the functions that were used to clean the data prior to modeling.

Ten algorithms were used. Table 1 shows where the .ipynb file for each algorithm can be found.

Table 1. Locations of .ipynb files containing modeling work using the old dataset.

Filename	Algorithms
`/src/training_LGBM_01.ipynb`	`LinearRegression`, `Ridge`, `Lasso`, `ElasticNet`, `LGBMRegressor`
`/src/training_random_forest_01.ipynb`	`RandomForestRegressor`
`/src/training_gradient_boost_01.ipynb`	`GradientBoostingRegressor`, `XGBRegressor`
`/src/training_SVR_CatBoost_01.ipynb`	`SVR`, `CatBoostRegressor`

Across the board, these rudimentary models performed poorly. The validation RMSE values were in the range of 300 to 600.

We also fit models to data in which the number of dimensions was reduced via PCA. PCA was performed in two ways: (1) on the whole dataset and (2) on groups of related columns. `/src/PCA_data.py` contains the functions used to perform PCA. Table 2 describes where the work can be found.

Table 2. Locations of .ipynb files containing modeling work on data in which PCA was applied.

Filename	Algorithms
`/src/training_SVR_CatBoost_02.ipynb`	`CatBoostRegressor`
`/src/training_gradient_boost_01.ipynb`	`GradientBoostingRegressor`
`/src/training_LGBM_01.ipynb`	`Ridge`, `LGBMRegressor`

Out of curiousity, we also tried fitting models to data in which the playgrounds with historic session counts greater than 70000 were removed. This dramatically improved the fit of the model, decreasing the error in both the training and validation set. `/src/training_gradient_boost_01.ipynb` illustrates the improvement in model performance.

Analysis with the new dataset

The data used in this iteration of modeling can be found here. On Google Drive, it is saved as `playground_stats_capped.csv`. The major difference between the new and old dataset is the range of the target variable. `unacast_session_count` values over 3000 were normalized to fall between 3000 and 4000.

Preprocessing techniques were reconsidered. As for imputation, we began by taking a closer look at each feature. In some features, missing values were in fact synonymous with zeros. In others, it made more sense to replace missing values with the mean or a specific value. For example, missing values related to presidential election results in Alaska were filled in manually. Feature engineering and selection were also performed to reduce dimensionality. This involved dropping columns with low fill rates, removing correlated features, and combining columns using domain knowledge. The discussion surrounding feature engineering and selection is documented here. `/src/imputer.py`, `/src/feature_eng.py`, and `/src/drop.py` contain the functions that were used to preprocess the input data.

Ten different kinds of models were pursued during this iteration. Table 3 shows where the work for each model can be located. It should be mentioned that Jupyter notebooks were run on Amazon EC2 to reduce computation time.

Table 3. Locations of .ipynb files containing modeling work using the new dataset.

Filename	Model
`/src/training_SVR_CatBoost_02.ipynb`	`SVR`, `CatBoostRegressor`
`/src/training_LGBM_02.ipynb`	`LGBMRegressor`
`/src/training_random_forest_02.ipynb`	`RandomForestRegressor`
`/src/training_gradient_boost_02.ipynb`	`GradientBoostingRegressor`, `XGBRegressor`
`/src/training_time_dependent.ipynb`	Time-series approach
`/src/training_mixed_effects_Python.ipynb`	Mixed effects model (Python)
`/src/training_mixed_effects_R.ipynb`	Mixed effects model (R)
`/src/training_tiered.ipynb`	Tiered model

With respect to the off-the-shelf regression algorithms (random forest and boosting methods), the validation RMSE values were in the range of 100 and 130. However, we believe that this improvement in model performance is most likely attributed to the capping of the target variable.

Time-series approach

A time-series approach was pursued in which the lagged target variable was included as an explanatory variable. We assumed that session counts would be similar across consecutive months for a given playground and would therefore serve as useful input signals. It is worth mentioning that, when training this model, ordinary *k*-fold cross validation could not be used since the random partition of the data could result in data leakage. Moreover, we could not use an off-the-shelf time-series cross-validator because the number of observations for each month was inconsistent. As a result, we had to create our own implementation of nested cross-validation. With a validation RMSE value of 176, it did not outperform the off-the-shelf regression models mentioned above.

Mixed effects

We assumed that not every playground observes the same behaviour with respect to visits. For example, playgrounds located in warmer climates may observe more visits in the spring or fall because it is unpleasantly hot in the summer. Meanwhile, playgrounds located in cooler climates may observe more visits in the summer because it is too cold to play outside in the winter. We could have fit a unique regression surface to observations derived from each playground, but it would have been unreasonable to create 2506 models and impossible to predict `unacast_session_count` for new playgrounds. A reasonable solution to incorporating these across-group differences was to build mixed effects models. We grouped the playgrounds using a specific characteristic and then fit a regression surface. However, each group was allowed to have its own intercept. As a result, information was shared across the groups to determine the regression surface, but the generated model was less generic than a pooled regression model.

We build these models using both R (using `lmer` function) and Python (`smf` function from the `statsmodels.formula.api` library).

Unfortunately, this didn't improve neither our RMSE (200) nor our MAE (100).

In R, we grouped the observations using levels of categorical variables (i.e. `state` , `climate` , `density_class` , `income_class`). None of the models outperformed the off-the-shelf regression models mentioned earlier. The validation RMSE and MAE values were around 200 and 100, respectively. We also applied k-means clustering with k = 2, 4 to see if more meaningful groups could be obtained. However, it should be mentioned that, with this grouping strategy, observations from the same playground could have been placed in different clusters. Unfortunately, implementing the new grouping strategy resulted in the same validation RMSE and MAE values as described earlier.

Building a mixed effects model in Python was troublesome because the `smf` function did not have an argument that enabled automatic dropping of columns that made the algorithm not converge. To compensate, we had to write a function that dropped these problematic columns. Similar to the implementation in R, observations were grouped using levels of categorical variables (i.e. `climate`, `density_class`, `income_class`). The results were similar to that of the R implementation with validation RMSE and MAE values of around 200 and 100, respectively.

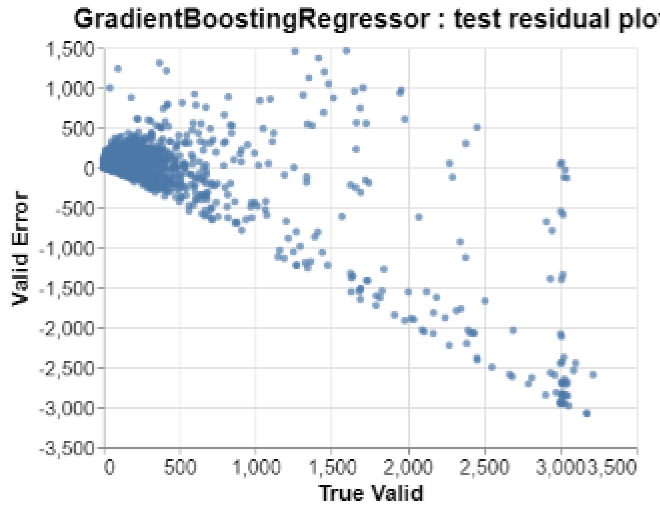
Tiered approach

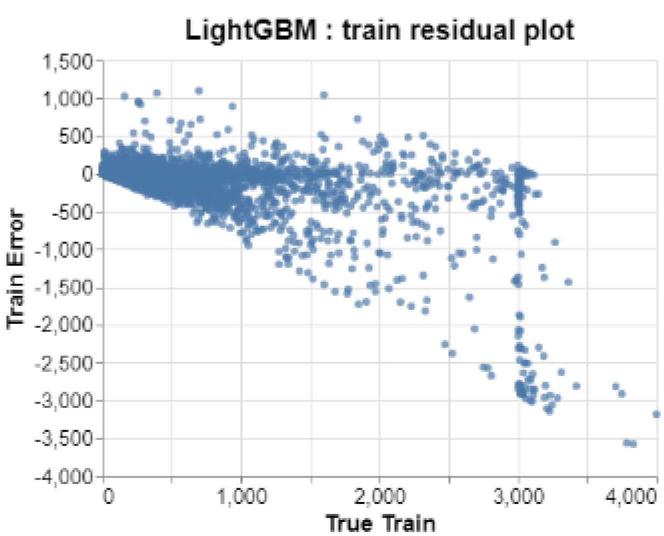
We also considered a tiered approach. This model consisted of a classifier which would predict an observation to be either low count or high count (i.e. below or above 300 sessions). Based on that decision, a prediction would be made using a regressor that was trained on low-count or high-count data. An `XGBClassifier` was created and its F1 score for the low-count class was 0.99 and that of the high-count class was 0.85. Given the skewness in the data, poisson regression was used to predict `unacast_session_count` for the high-count data. However, its validation RMSE was 1.22287e+73. Other generalized linear models suitable for count data were considered (e.g. negative binomial); however, the algorithm would not converge and model coefficients could not be obtained. On the other hand, regression models build for the low-count data using `LinearRegression` and `XGBRegressor` had validation RMSE values of 54 and 35, respectively. However, we acknowledge that these values are not reflective of the performance of the tiered model as a whole.

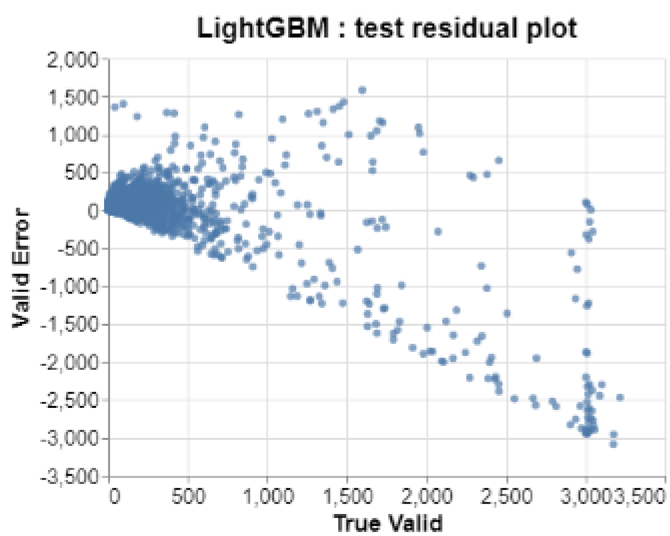
Residual plots

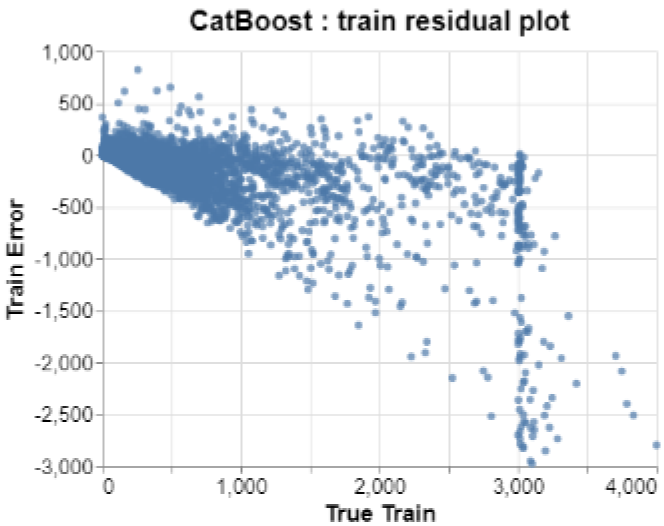
In order to assess the fit of the model, we created residual plots. In these plots, we observed a trend in the residuals. Residual plots derived from the median-predicting `GradientBoostingRegressor`, `LightGBM`, and `CatBoost` models are shown below.

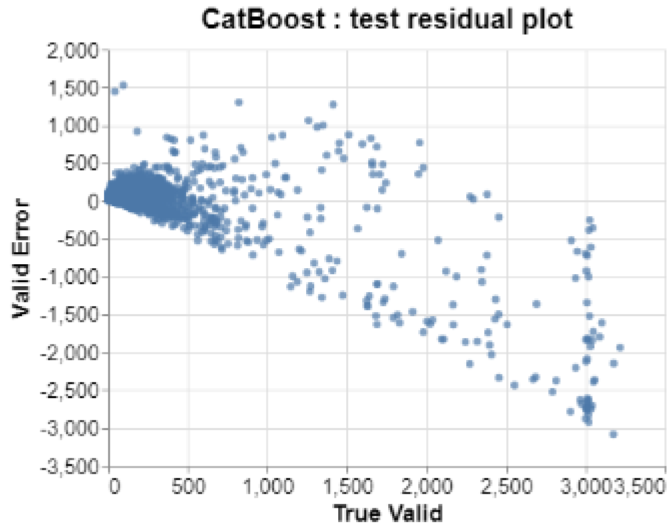

GradientBoostingRegressor : train residual plot
1,000
500
0
-500
-1,000
-1,500
-2,000
-2,500
-3,000
-3,500
-4,000
Train Error
0
1,000
2,000
3,000
4,000
True Train


GradientBoostingRegressor : test residual plot
1,500
1,000
500
0
-500
-1,000
-1,500
-2,000
-2,500
-3,000
-3,500
Valid Error
0
500
1,000
1,500
2,000
2,500
3,0003,500
True Valid


LightGBM : train residual plot
1,500
1,000
500
0
-500
-1,000
-1,500
-2,000
-2,500
-3,000
-3,500
-4,000
Train Error
0
1,000
2,000
3,000
4,000
True Train


LightGBM : test residual plot
2,000
1,500
1,000
500
0
-500
-1,000
-1,500
-2,000
-2,500
-3,000
-3,500
Valid Error
0
500
1,000
1,500
2,000
2,500
3,0003,500
True Valid





Data product

Results

Our data product consists of three boosting models that predict the median `unacast_session_count`. We selected these models because they are least worst-performing models we came across in our analysis, they are relatively fast to train, and the median is less sensitive to extreme values than the mean, as mentioned earlier. Their performance is as follows:

```
## # A tibble: 3 x 3
##   model                     `train mae` `test mae`
##   <chr>                           <dbl>      <dbl>
## 1 GradientBoostedRegressor         38.8       98.6
## 2 LightGBM                         27.7      104.
## 3 CatBoost                         35.8       95.6
```

Reproducing the data analysis

Instructions on how to run the makefile to reproduce this report can be found here. The makefile automates the execution of six scripts. The first script `/src/01_split_data.py` splits the raw data into the training and test sets. The second script `/src/02_preprocessing.py` fits an imputer and one-hot encoder on the training set and saves them as .joblib files for later use. It also transforms the training and test sets and saves them as .csv files in the `/data` directory. Smaller versions of the training and test sets are also saved as .csv files to serve as dummy data for testing. It should be noted that the preprocessing methods used here are identical to those used in the second iteration of modeling. *Note: features are not scaled because tree-based models are not sensitive to scaling.*

`/src/03_gbr_model.py`, `src/04_catboost_model.py`, and `/src/05_lgbm_model.py` are the scripts in which modeling take place. The hyperparameters are hard coded based on the results of the random searches performed in the second iteration of modeling. In each script, a model is fit and then used to predict on the training and test sets. Since these models can predict negative values, the nonsensical predictions are converted to zero prior to calculating the MAE. Each model is saved as a .joblib file and its performance metrics are saved as a .csv file in the `/results` directory. This file, which is not part of the makefile, renders this report.

Predicting on new data

It is also possible to predict `unacast_session_count` values for an unseen dataset using the models described above. Instructions on how to run the makefile to predict on new data can be found here. The data is preprocessed in the same way as described above: the imputer and one-hot encoder from earlier is loaded to transform the data. `/src/07_prediction.py` outputs a .csv file in the `/results` directory. Non-negative predictions from the three models are added as new columns to the input data.

Recommendations

Outliers in the target

It was demonstrated again and again that outliers in `unacast_session_count` were detrimental to model fit. In the future, other statistical or machine learning models that are more robust to outliers could be pursued. Alternatively, the strategy of fitting multiple hyperplanes to the data could be considered further. However, we believe that the most effective strategy for improving model fit is to reevaluate how the target value is calculated using cell phone location data. Perhaps the polygon that is drawn around some playgrounds are ill-shaped, giving the impression that more playground visits took place than there actually were.

Missing values in the target

Prior to modeling, we simply removed rows with missing target values. However, we stress that these values can be dealt with more elegantly. For example, if evidence emerges that a playground sees visitors year round, but is located in an area with poor network coverage, missing values could be filled by multiple imputation. On the other hand, an observation missing `unacast_session_count` could be dropped if evidence shows that the playground was not in operation that month. These scenarios demonstrate that there is no one-size-fits-all solution.

Missing values in explanatory variables

We dropped a handful of columns with a high proportion of missing values. However, some of those features may in fact be important predictors of playground usage. If time and resources permit, it may be worth consulting additional external sources to fill in those values.

Feature engineering and selection

This may not improve model fit, but further manipulation of the raw data may allow for other algorithms, which were initially not used for reasons related to time complexity, to be considered.

Conclusion

Although the models are in need of improvement, we hope that our work has at least brought the organization closer to attaining a reliable predictor that can be used to inform the decision-making process around community play spaces. It should also be mentioned that these models were fit to data collected before the pandemic. Although people are returning to playgrounds as restrictions ease, these models may only be reflective of pre-pandemic behaviour. Further model tuning may be required to incorporate the behavioural changes that took place and are continuing to take place in society.

Acknowledgements

We would like to thank Biba Ventures Inc. for sharing their resources and providing unparalleled support over the course of this project. We extend our gratitude to our mentor, Vincenzo Coia, and the UBC MDS program for making this experience possible.