

faraankhan /
predicting_churn_on_ott




 Code  Issues  Pull requests  Actions  Projects  Wiki  Security  Insights  Set



☆ 0 stars  0 forks  1 watching  Activity

 Public repository

 main ▾



 Branches  Tags



faraankhan Add files via upload ...

3 minutes ago ⌚ 2

[View code](#)

☰ README.md



Predicting Churn for Video Streaming Services

Project Overview

The purpose of this project was to build and optimize a classification model to predict the likelihood of a target variable belonging to one or more groups. Given the rise of digital streaming services over the past decade, it has become increasingly important for companies to identify users who are likely to churn (cancel their service), and then act to retain these users. Most companies rely on monthly subscriptions as their primary revenue stream, so being able to identify users who are likely to churn can help these companies stay competitive in an increasingly saturated market.

I built two classification models (logistic regression and a decision tree), and optimized each through a series of feature engineering, transformation, validation, and fine tuning steps. Ultimately, the models were compared on their performance for recall, or minimizing false negative cases, as these predicitions represent users who were predicted not to churn, but who actually did churn. Recall is the most important metric with regard to churn as false negatives represent users whom the model could not identify as likely to churn, but who actually did cancel their service. A classification model with high recall will allow for companies to miss fewer accounts that are likely to churn, and act preemptively to retain these users.

Business Understanding

Since Netflix began streaming original content with 2012's 'House of Cards,' the emergence of a new type of media company, video subscription, has become a mainstay in consumers' daily lives. The consumption of video content has more or less shifted from cable television to online streaming platforms, with dozens of options that offer users specialized content. More than a decade after 'House of Cards,' streaming services, which most often charge users a monthly subscription, have begun to split their focus from pure growth to growth AND retention. Churn, or the rate at which users cancel their service, has become an increasing concern; the Data Analytics firm Antenna found that average monthly churn grew from 3.2% to 5.2% from 2019 to 2021. Similarly, a survey conducted by Deloitte in 2022 found that nearly 37% of consumers cancelled a streaming service within the last six months.

If streaming services were able to predict the likelihood that a user churns, based on platform engagement and user behavior, it could preemptively deploy strategy to retain these high risk users, and ultimately sustain revenue for additional platform growth. Some important questions to consider:

1. Which factors play the most important role in leading to customer churn?
2. Do user demographics (age, sex, etc) play a role in churn?
3. Of the users likely to churn, how engaged are they with a given streaming platform?

Data Understanding and Analysis

The dataset used for this analysis was an anonymized dataset found on Kaggle of customer behavior on an unnamed streaming platform from 2020. It contained 2000 rows, and 16 columns, the most relevant of which were:

- gender
- age
- videos watched
- number of days subscribed
- maximum days inactive
- customer support calls
- mail subscription
- churn (target variable)

Data Exploration & Logistic Regression

Upon opening the dataset, the data appeared to be in decent working condition. There were few NaN values (most of which were dropped, except for 'maximum days inactive,' for which the median value was imputed on NaNs), and all but three variables were numeric. The biggest problem, however, was that the data was unbalanced. Nearly 87% of the entries indicated no churn, and 13% indicated churn. I knew that to build a usable model, some form of oversampling or undersampling would be required to reestablish a 50% baseline for churn/no-churn.

After conducting a train/test split, standard scaling the data, and one-hot encoding categorical variables, I ran a correlation matrix to eliminate variables that appeared to be colinear. Unsurprisingly, it appeared that both maximum and minimum daily minutes watched correlated strongly with some variables, so I dropped both before running a baseline model. Once the data was ready for baseline modeling, I ran a logistic regression with the remaining variables to predict the target variable, churn. At first glance, the confusion matrix appeared to perform well with .86 accuracy. However, closer analysis shows that this simply reflects the imbalance in the data, as 87% of observations were classified as 'non-churn.' Furthermore, the baseline model performed poorly for precision (.39), and even worse for recall (.18), the metric for which we want to optimize. This was reflected by the model's f1-score (.24), which showed overall poor performance.

Fine Tuning and Hyper-parameter Adjustments

The first step in beginning to optimize the model was to account for the class imbalance. First, I attempted to reweigh the classes to see if the overall model would improve by observing an improvement in ROC AUC. After plotting ROC curves for various weights, it was clear that model performance would steadily improve as the minority class weight increased as AUC values steadily rose from .69 for a balanced class to .78 for a 1000:1 weight rebalance. Next, I had to decide whether to undersample, which would dramatically reduce the observations in the dataset and potentially cause underfitting, or oversample, which would provide more datapoints and risk overfitting the model. Since the initial dataset was not dramatically large, oversampling was conducted via SMOTE. After plotting various ROC curves for a range of sampling strategy ratios, it was determined that .9 would provide the optimal ratio, as it yielded the highest AUC score (.74). After applying SMOTE with a ratio of .9, the new training dataset contained 1257 non-churn and 1131 churn observations (a total of 2388 observations).

Running a new logistic regression yielded improved performance. While accuracy dropped to .72, recall dramatically increased to .76, and f1-score increased to .41, indicating an improvement in model performance. Next, I applied both L1 lasso and L2 ridge penalties to the model to observe any noticeable improvement. While the model did not really improve in any specific metric, the f1-score very slightly improved for L1 lasso regularization, so this penalty was applied moving forward.

Finally, polynomial transformation was applied to the dataset to observe any non-linear relationships between variables. Logistic regressions were run on polynomial transformations of levels 2, 3, and 4 (squares, cubes, or to the power 4). As a higher order polynomial is introduced, the number of features exponentially increases (66, 286, and 1001 columns for levels 2, 3, 4, respectively). Realistically, higher order polynomials become increasingly difficult to interpret, so a 4th order polynomial transformation was set as the limit. Upon assessing model performance metrics, the square transformation performed best with accuracy, recall, and f1-scores of .8, .87, and .52, respectively. Therefore, the 2nd order transformation was selected for validation.

Cross Validation

To validate the second order model, cross validation was performed using 10 folds. Mean recall, accuracy, and f1-scores for the 10 folds yielded values of .78, .81, and .79, respectively. Average coefficients were extracted, which represented the optimized logistic regression model.

Decision Tree

The optimized logistic regression model was compared to an optimized decision tree to determine which model would best predict churn while optimizing for recall. After solving for the optimal hyperparameters max features, max depth, minimum sample split, and minimum leaf split, the optimized decision tree yielded accuracy, recall, and f1-scores of .72, .89, and .44, respectively, with an ROC AUC of .79. According to the decision tree, the most important features for predicting churn in descending order were 'multi-screen,' 'weekly minutes watched,' and 'customer support calls.'

Discussion and Recommendations

Between the optimized logistic regression model and the optimized decision tree, it makes more sense to use the decision tree when predicting whether a customer churns or not, as it yielded the higher recall score of the two. However, the overall model of the logistic regression (polynomial transformation of degree 2) yielded a higher f1-score, and so if it were to be more fine tuned to remove irrelevant and harder to interpret variables, there is a change that it would be a more useful model when predicting churn. The addition of more data, like types of videos viewed, number of users per account, and account creation date, might help predict with more certainty if a user is likely to churn. Additionally, knowing whether a user is likely to churn may not be useful enough for a company to take action. It may be useful for a company to know how likely a user will churn within a certain time period (like if a user is likely to churn within the next 6 months, as opposed to 1 month, a company may have a longer runway to act to retain said user).

Based on the model, the variables 'multi-screen' and 'weekly minutes watched' were ranked as most important when predicting whether a customer would churn. Incentiving customers to use the platform on multiple screens (like on a laptop or other mobile device) may help retain a user. One such incentive could be releasing behind-the-scenes content, or other exclusive features only available on a platform's mobile app. In the long run, streaming services might consider partnering with each other to offer content bundles that may incentivize users to stay engaged with multiple platforms for a longer amount of time at a lower cost, increasing the likelihood that the user will be retained.

Finally, the amount of customer support calls had a dramatic impact on whether a user churned or not. Identifying a critical call number, at which point a customer's likelihood to churn dramatically increases, would help in deploying a strategy to retain said customer. If, after 3 calls with customer support, a customer's probability of churning doubles, then it is may be prudent to reserve a subset of customer support that can be assigned to a customer to ensure that optimal service is provided. It may even make sense to begin A/B testing a customer support bot integration, on which customers can log specific issues that can then be triaged to appropriate customer support for resolution. This type of tool would also allow platforms to begin collecting data on common issues that could then be streamlined and resolved in an efficient manner.

All these strategies that seek to retain customers should be A/B tested among a randomly selected subset of customers who are likely and unlikely to churn (based on the decision tree). Through A/B testing, we will be able to successfully test whether a particular strategy was effective in retaining a customer or not.

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)