

Predicting Churn

Ashton Reed
Springboard's Data Science Track
Capstone 2
October 2020

Problem

Companies based on a recurring revenue/subscription model can lose thousands of dollars annually because of customer churn. However, there is something that companies can do to prevent this.



Solution

After a company looks at a typical customer lifecycle and reviews any procedural changes that can be made, they can then use a survival analysis model to predict customer churn and better understand what customer characteristics are indicative of a loyal customer vs. one who is likely to churn. As a result, the company can look to market to potential customers who may have similar interests or characteristics as loyal customers. Looking at customers who have left the company may also help a company determine the biggest reasons for leaving and whether or not they are something the company has any power over.



Who Benefits?

- The business itself maintains a higher percentage of its recurring revenue
- Customers have a better user experience
- The customer experience/retention team has more time to focus on high target clients
- Account executives/salespeople whose earnings are affected by renewals can spend more time finding new customers instead of trying to convince current customers to stay with the company



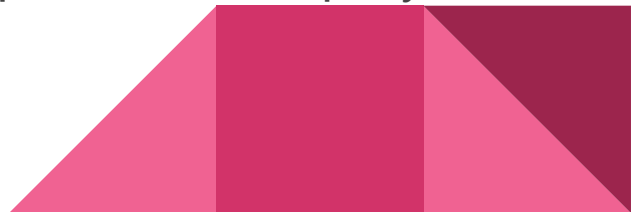
Dataset

- Only a single dataset was used for this project. It was a free telecom dataset that can be found at <https://www.kaggle.com/raumonsa11/churn-telco-europa>
- Conflicting statements about data in the Data Dictionary portion found here: <https://www.kaggle.com/raumonsa11/churn-telco-europa-spa-eng>
 - Explanation of dataset says it is a new telecom company that has been around for 14 months
 - The tenure of multiple customers is > 1000 (days)
 - As a result, I ignored the claim of it being a 14-month-old company and stuck to the numbers in the dataset
 - Bounds were given for multiple features, but there were many instances in which data did not fall within these bounds



Approach

- Dropped all observations with negative numbers for tenure (DAYS_LIFE)
- Noted there were several values that fell outside of the bounds outlined in the Data Dictionary that was uploaded with the original dataset. These values were all for categorical features, so they were left as-is until being one-hot-encoded
- CITY_VOICE, STATE_DATA, CITY_DATA, and STATE_VOICE were all missing 5621 entries. They were filled with 9999 immediately before these categorical features were one-hot-encoded
- Not all CITY_* and STATE_* values were the same for a single observation but were left as-is--in the absence of a subject matter expert at the company



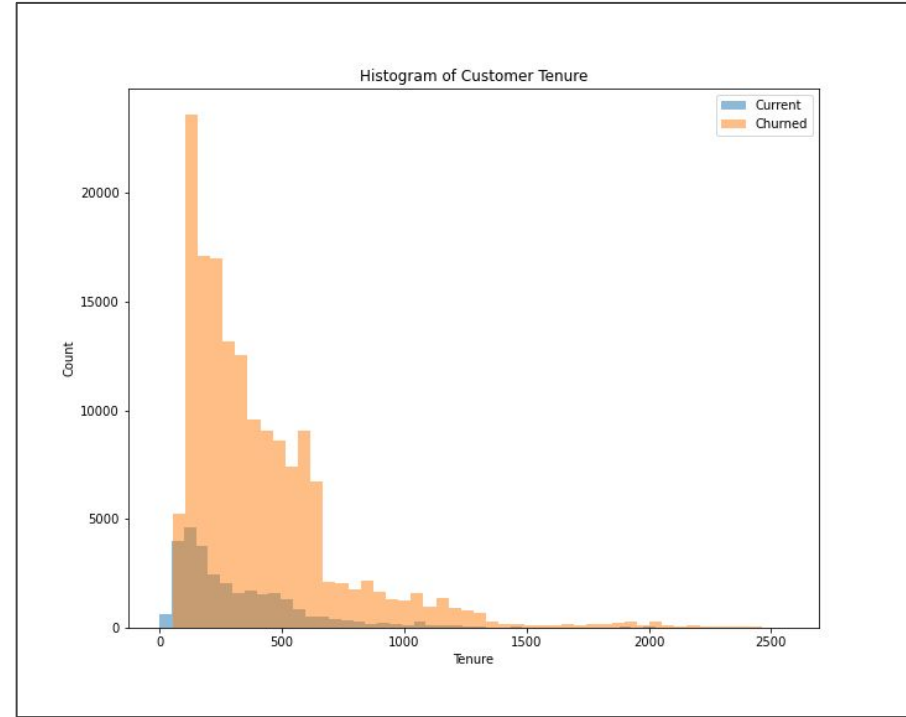
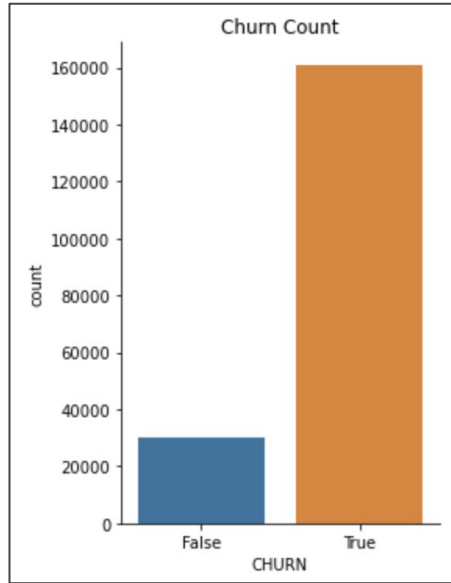
Insights

- Multiple instances of high collinearity between features
 - Notably between *_VOICE *_DATA features (i.e. CITY_VOICE and CITY_DATA often had the same values, but not always)
 - Collinearity between a customer's behavior over the last 1-3 months vs. their entire lifespan with the company
 - Collinearity of 0.5 for Price Plan vs. Minute Plan, indicating that the price of plans have not remained constant during the company's lifetime
- Not all CITY_* and STATE_* values were the same for a single observation



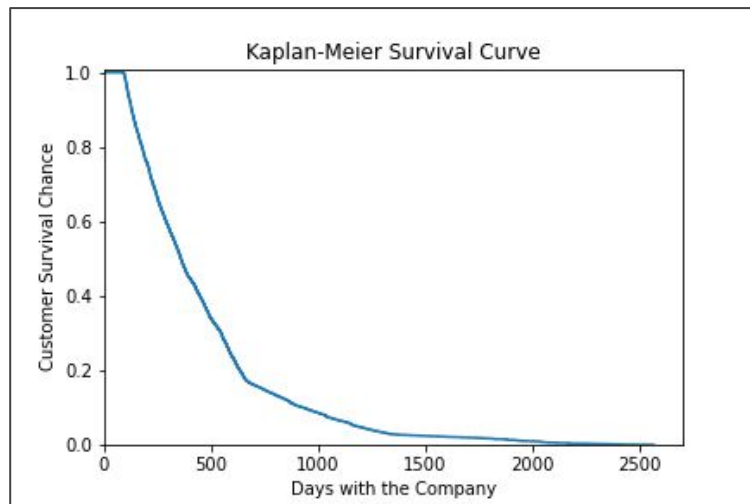
Insights (cont.)

- The company has had a lot of customers, but the majority of them are no longer with the company




Modeling

- First model built on full (one-hot-encoded) dataset
 - CoxPHSurvivalAnalysis
- Second model built on dataset with no categorical features
 - Used a pipeline
 - Train-test split/cross-validation
 - CoxPHSurvivalAnalysis
- Kaplan-Meier Survival Curve (same for both models)



Limitations and Future Iterations

- Had to drop multiple columns with higher collinearity so as not to break the model
 - This Python package (scikit-survival) is not optimized for a dataset with so many columns of one-hot-encoded features
 - Large, one-hot-encoded dataset did not work with `train_test_split` or `cross_validate` functions, but the smaller dataset without one-hot-encoded features worked with both functions in addition to the pipeline function
 - Can explore the “lifelines” package in the future to see how it works with a dataset like the main one used in this project
- 

Questions?

