falloutb1tch / ChurnBackTime

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

GPL-3.0 license

0 stars 0 forks 1 watching 1 Branch 0 Tags Activity

Public repository

main 1 Branch 0 Tags Go to file t Go to file + Add file Code

falloutb1tch Update README.md now

Data	workies	yesterday
Images	final push	10 minutes ago
Notebooks	updates	last week
.gitignore	Create .gitignore	2 weeks ago
Final.ipynb	final push	10 minutes ago
LICENSE	Create LICENSE	2 weeks ago
README.md	Update README.md	now

README GPL-3.0 license

If I Could Churn Back Time

Overview

For my capstone project, I have chosen to continue in a similar vein to my Phase 3 project with entirely different data. Churn, which is what it is called when a customer stops doing business with a company, is one of the largest issues facing almost every industry today. In context of subscription industries, for example, "churn" is a massive ongoing issue, as it is infinitely cheaper to keep existing customers than to lure in new ones. Therefore, establishing a procedure to keep customers from "churning" is of the utmost importance. The first step in this process is identifying the reasons behind customer churn, such as dissatisfaction, area of service provided, quality of service, and other such considerations. Following that, we will build a model that will accurately predict whether or not a customer will churn, and thereby provide the limitlessly valuable opportunity for a company to prevent that churn.

Business Understanding

As previously stated, it is much cheaper to keep existing customers than to continually lure new customers in. In other words, the longer a customer stays with a company, the more money that company stands to make from them. It seems fairly simple in that context, right? So it also follows that if there was a way to accurately predict whether or not a customer would churn, that method would be likely to save a company a boatload of money. And no, "boatload" in this case is not an exaggeration - the data we are using today comes from a fictional telecommunications company, but the very real company known as Verizon walked away from 2023 with $134 billion in total operating revenue. So our goal here was quite simple - build a model that could accurately predict when a customer would stop doing business with a company, and provide that company an opportunity to stop that customer from churning, thereby saving them money.

Data Understanding

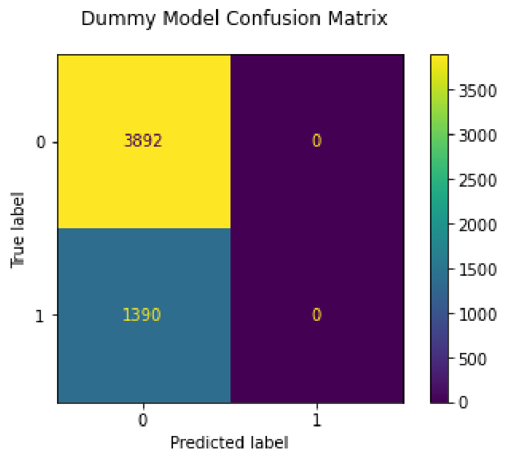
The data we've used for this endeavor comes to us from IBM, and is a publicly available dataset created for this exact purpose - to help deal with churn problems, and allow students like me to cut our teeth on such before we enter the wider world of data science. This sample data module tracks a fictional Telco company's customer churn based on a variety of possible factors, such as gender, monthly charges, and usage information, as well as whether the customer churned or not. Due to the nature of the data we used, any location based features were excluded because the data was all centered in California specifically. There is also a class imbalance present, meaning that 73% of customers have not churned and 26% have, so our data is "imbalanced" from the start.

Data Preparation

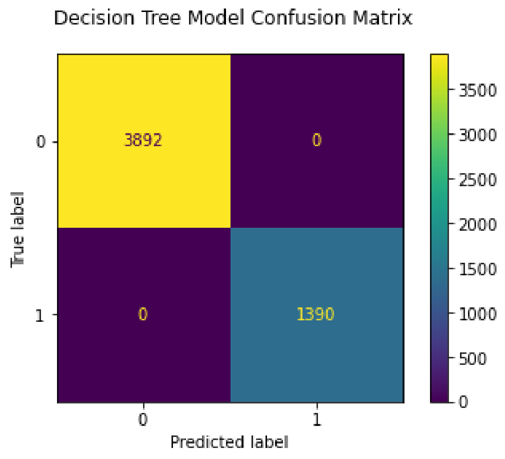
we will remove any columns that are unnecessary or contain information that we already have in our other two available dataframes. This way, we will prevent overlapping of information and inaccurate feature impact on our modeling later. Once again there are many columns in this dataframe that were repeated in the first (or second) dataframe, and therefore are unnecessary. Furthermore, any of the quarter or count information isn't relevant to our modeling process. These things have nothing to do with whether or not a customer churned, because no consumer is keeping track of fiscal quarters of their cellphone company to plan to cut their service, but gender or tenure with company might very well be relevant. Now, throughout all this data exploration and preparation, we have created three separate dataframes with one common column ("Customer ID"), 7043 entries that correspond to each other in each of the three dataframes, and various information on their services, plans, and habits. At this point, we must weld together those three separate dataframes into one usable one, and then begin the analysis process.

Analysis

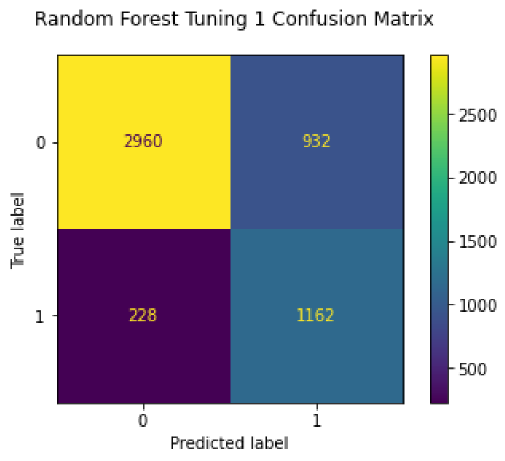
Our first model was a dummy model that would, as dummy models do, predict "no" each time. Despite that obvious handicap, our cross-validated accuracy score was still 73% and gave us a reasonable baseline. Due to being a dummy model, there is 0% recall, however, and recall is our most important metric in this context.



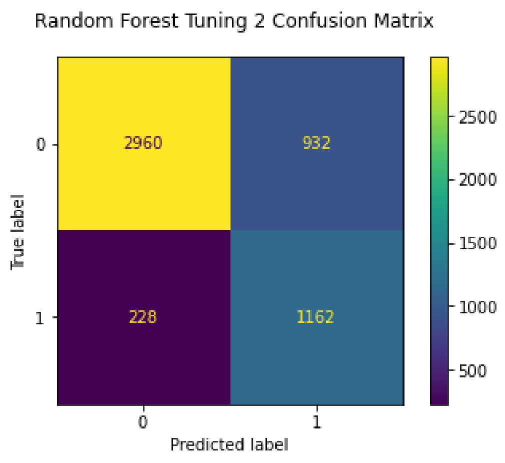
Our next step was a simple decision tree model, which netted us a cross-validated accuracy score of 75% but was incredibly overfit. Due to this, the idea of using a decision tree model for this purpose was rejected.



The next attempt was a completely untuned Random Forest ensemble method, which is truly a collection of decision trees meant to be tuned with various hyperparameters. In any case, this untuned Random Forest achieved a cross-validated accuracy score of 81%. It was also very overfit and required further tuning. I have not included the visual here because I find it to be somewhat unnecessary. Thus, we began the process of tuning the Random Forest model to achieve peak recall, as that is our most important metric in this project. Our first attempt at tuning used the parameters n_estimators, max_depth and class_weight, and achieved a cross-validated accuracy score of 81% and a recall score of 84%.



A second pass at tuning used more specific variations of the same three hyperparameters, and actually resulted in a loss of accuracy with the same rate of recall. This model had a cross-validated accuracy score of 77% and a recall score of 84%.



Model Recommendation

Therefore, the final model was the first hypertuning attempt, with hyperparameters n_estimators of 100, max_depth of 5, and class_weight of balanced_subsample. These parameters allowed us to achieve a recall score of 84% and a cross-validated accuracy score of 81%.

Conlusion and Next Steps

In conclusion, it is entirely possible and very beneficial to employ a similar predictive model for the purpose of customer churn issues on a real-world scale. With what limited time and resources I had available, I was able to properly tune a model that was 81% accurate and had 84% recall, and I'm sure that those numbers could be tweaked and increased with further time and resources applied. Earlier in this notebook, I mentioned that Verizon had clocked something to the tune of $134 billion in revenue in 2023; I do not know what Verizon is doing to solve any churn problems they may be having, but imagine if a company like Verizon could successfully stop customers leaving

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● **Jupyter Notebook** 100.0%