*"**Customer attrition**, also known as **customer churn**, **customer turnover**, or **customer defection**, is the loss of clients or customers" — Wikipedia*

For this project, let's agree on the universal assumption that "customer churn is bad", and "customer retention is good".

Abstract: This documentation takes a different slant on a well-traveled churn dataset that does not have the right features to blindly deploy even the best predictive model we can generate. Below I'll share the problem statement, data preparation steps, feature analysis, visualizations and select Python code from the best of the Scikit-learn classification models I tried for predicting customer churn. Most importantly, I'll show that by **moving the decision threshold probability along the precision-recall curve, you may find tranches of churn cases where you feel confident enough to deploy real retention actions.**

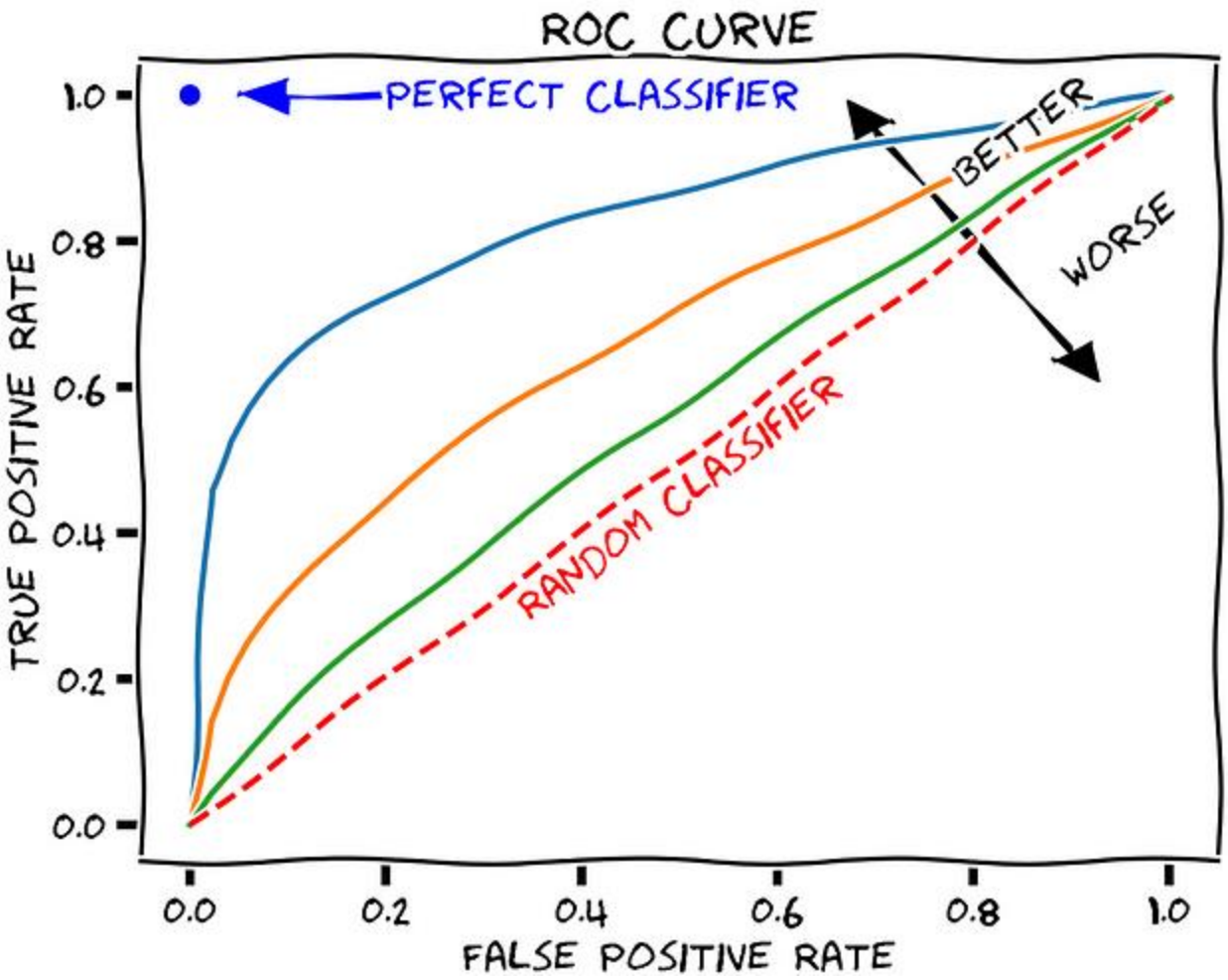
Business Objective

A telecommunications company (hereby "TelCo"), who sells residential Voice and Internet services, is experiencing a massive customer churn rate of nearly 27%. This level of churn may be enough to bankrupt a real-life company. Given the lack of publically available customer data, we are using the IBM Cognos Telco Customer Churn simulated data set which contains labeled churn for 7,043 customers.

TelCo wants to deploy customer retention strategies to reduce customer churn. The company has asked us to:

- Develop a predictive model to classify customer churn risk
- Explain the relative influence of each predictor on the model's predictions
- Suggest potential approaches to reduce customer churn

Here we have a binary classification problem to solve so we'll set our target dependent variable to 1 (churn) or 0 (retain). We will use metric ROC AUC to optimize our estimators. ROC AUC is the **A**rea **U**nder the **C**urve of a **R**eceiver **O**perator **C**haracteristic.



With ROC AUC, a random classifier scores 0.5 while a perfect classifier scores 1.0. I like ROC AUC as it recommends models that optimize both the true positive and false positive rates that are significantly above random chance, which is not assured with accuracy as the evaluation metric. This is important since ***we need confidence in our positive class predictions (churn)*** when taking retention actions.

Customer Churn Raw Data

Let's start by taking a look at the 33 columns we have to work with. After converting our source file to a pandas data frame, the top 5 records show:

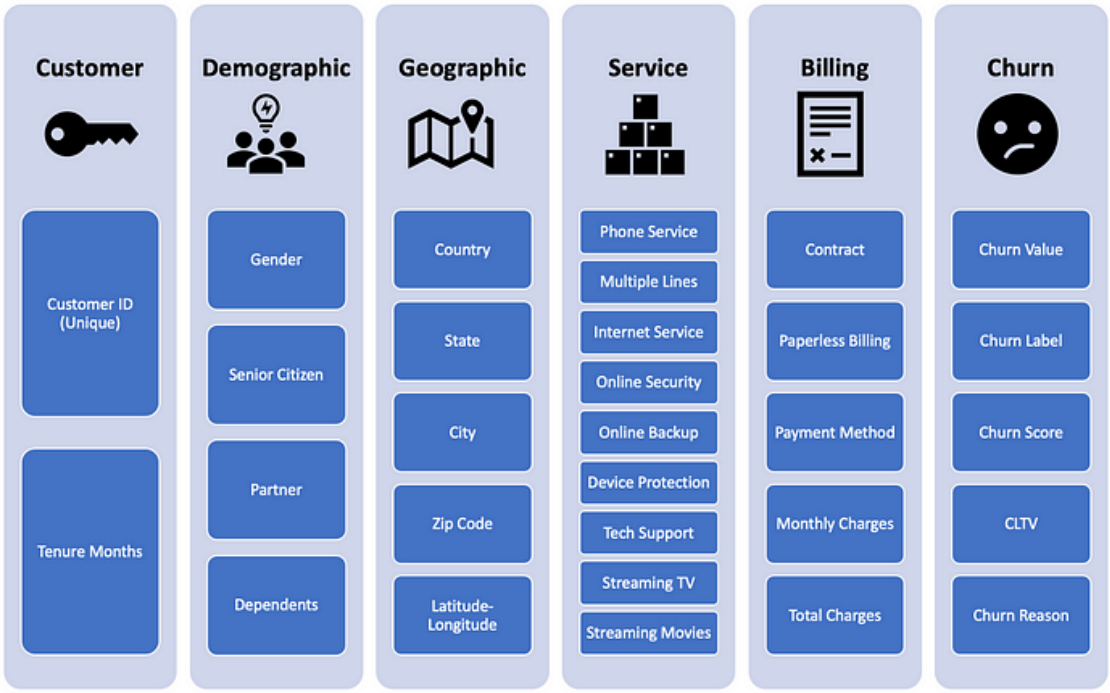
	CustomerID	Count	Country	State	City	Zip Code	Lat Long	Latitude	Longitude	Gender	Senior C	Partner	Dependent	Tenure	Phone Serv	Multiple	Internet S	Online Secur
3	9237-HQITU	1	United State	Californi	Los Ang	90005	34.059281,	34.059281	-118.307420	Female	No	No	Yes	2	Yes	No	Fiber optic	No
4	9305-CDSKC	1	United State	Californi	Los Ang	90006	34.048013,	34.048013	-118.293953	Female	No	No	Yes	8	Yes	Yes	Fiber optic	No
5	7892-POOKP	1	United State	Californi	Los Ang	90010	34.062125,	34.062125	-118.315709	Female	No	Yes	Yes	28	Yes	Yes	Fiber optic	No
6	0280-XJGEX	1	United State	Californi	Los Ang	90015	34.039224,	34.039224	-118.266293	Male	No	No	Yes	49	Yes	Yes	Fiber optic	No
10	6467-CHFZW	1	United State	Californi	Los Ang	90028	34.099869,	34.099869	-118.326843	Male	No	Yes	Yes	47	Yes	Yes	Fiber optic	No
13	6047-YHPVI	1	United State	Californi	Los Ang	90039	34.110845,	34.110845	-118.259595	Male	No	No	Yes	5	Yes	No	Fiber optic	No
14	5380-WJKOV	1	United State	Californi	Los Ang	90041	34.137412,	34.137412	-118.207607	Male	No	No	Yes	34	Yes	Yes	Fiber optic	No
15	8168-UQWWF	1	United State	Californi	Los Ang	90042	34.11572, -	34.115720	-118.192754	Female	No	No	Yes	11	Yes	Yes	Fiber optic	No

Screenshot by Author | First 19 columns…

	Online Backup	Device Protectic	Tech Supp	Streaming	Streaming Mo	Contra	Paperless Bil	Payment Met	Monthly Cha	Total Char	Churn	Churn	Churn Score	CLTV	Churn
3	No	No	No	No	No	Month-to	Yes	Electronic check	70.7	151.65	Yes	1	67	2701	Moved
4	No	Yes	No	Yes	Yes	Month-to	Yes	Electronic check	99.65	820.5	Yes	1	86	5372	Moved
5	No	Yes	Yes	Yes	Yes	Month-to	Yes	Electronic check	104.8	3046.05	Yes	1	84	5003	Moved
6	Yes	Yes	No	Yes	Yes	Month-to	Yes	Bank transfer (a	103.7	5036.3	Yes	1	89	5340	Competito
10	Yes	No	No	Yes	Yes	Month-to	Yes	Electronic check	99.35	4749.15	Yes	1	77	5789	Competito
13	No	No	No	No	No	Month-to	Yes	Electronic check	69.7	316.9	Yes	1	66	2454	Competito
14	Yes	Yes	No	Yes	Yes	Month-to	Yes	Electronic check	106.35	3549.25	Yes	1	65	2941	Competito
15	No	Yes	No	Yes	Yes	Month-to	Yes	Bank transfer (a	97.85	1105.4	Yes	1	70	5674	Competito

Screenshot by Author | …last 14 columns

It's helpful to group columns by domain when thinking about their potential usefulness as predictors. There's a data dictionary in the GitHub notebook, but here's how I'm viewing these:



- Customer — A unique customerID indicates 7,043 unique customers in our sample with customer tenure ranging from 0 to 72 months
- Demographic — Four useful customer demographic fields
- Geographic — Customer location down to geographic coordinates
- Service — Eight billable services (Internet splits to Fiber, DSL or None) and two streaming indicators
- Billing — Includes contract type, billing configuration and billed charges
- Churn — Churn value (our target), Churn Reason, and two internal TelCo metrics for Churn Score and Customer Lifetime Value.

Data Preparation

I wrote six functions for data preparation and feature creation. I exposed these functions through FunctionTransformer in a pipeline using scikit-learn libraries.

The most interesting 3 features I created were:

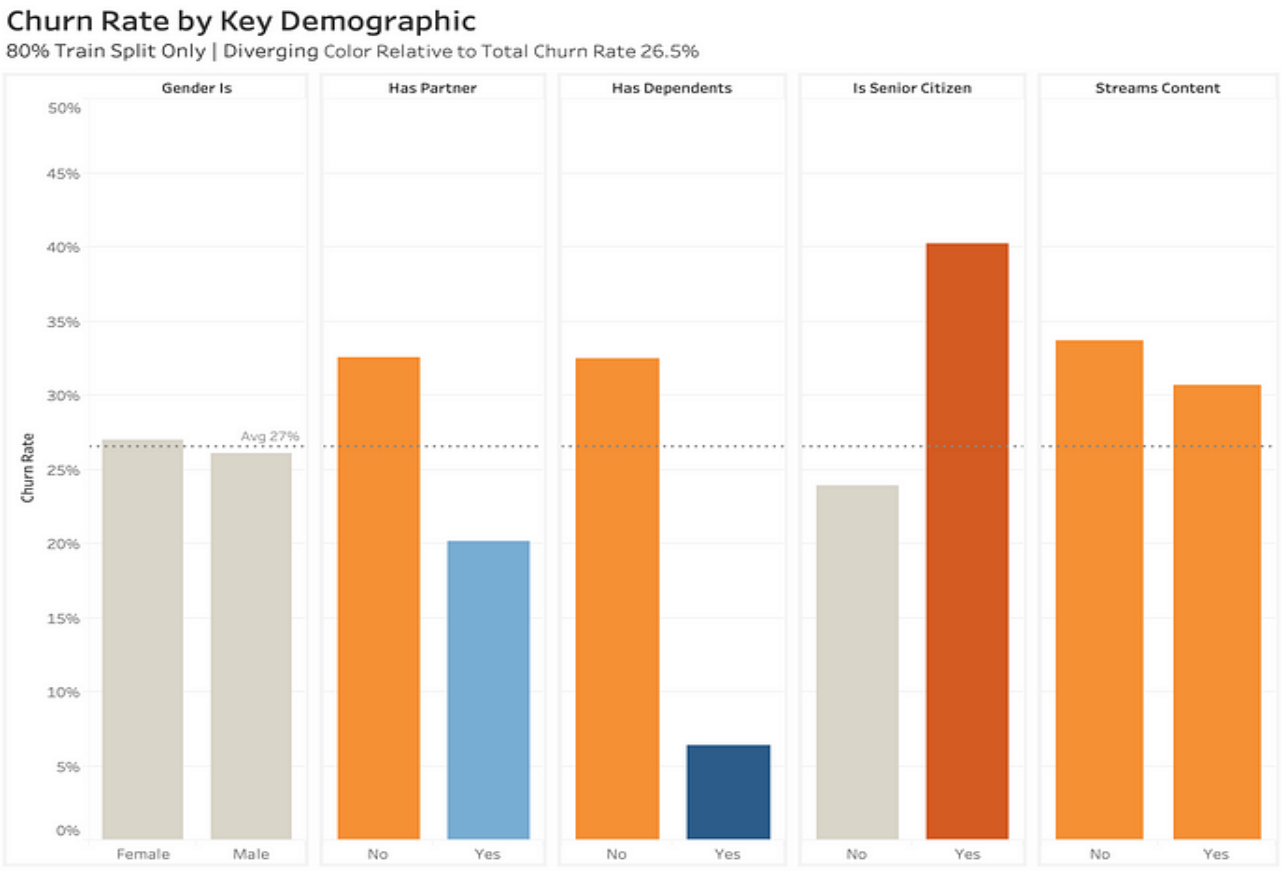
- Mapping 4 payment methods into “Payment Method Automated” (1=Yes).

- Mapping 8 services into “Product Profile” columns that delineate core service (phone, fiber/DSL, or bundled voice & internet) while also indicating layered “add-on” services like multi-line, tech support, etc.
- Created “Customer Charge Index” which measures the relative pricing of each customer to standard prices. I used combinations of services across all training data and incremental average pricing to derive the standard, stand-alone price for each service. From there I indexed each customers monthly charges against the standard prices for their basket of services.

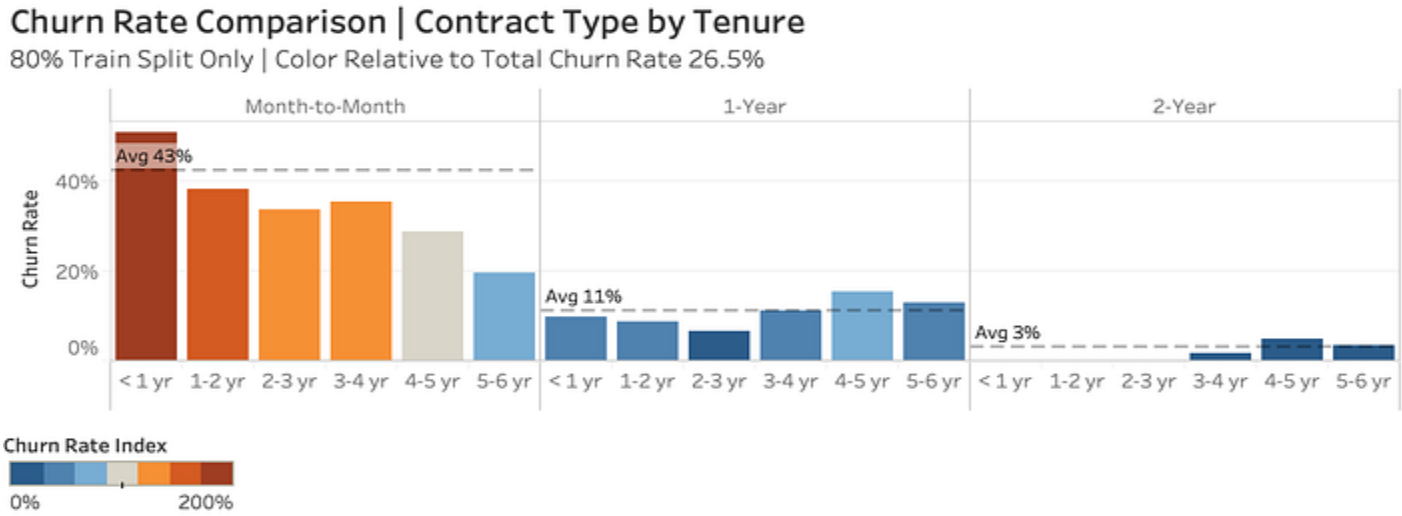
Prior to data profiling and feature analysis, I split the data into 80% train and 20% test. All analysis was performed only on training data to avoid data leakage into any predictive model.

Customer Churn Feature Analysis

Now on to some basic insights.



From above, the churn rate is lower for “has partner”, “has dependents”, “is not senior citizen” and “streams content”. Gender was not differentiated.

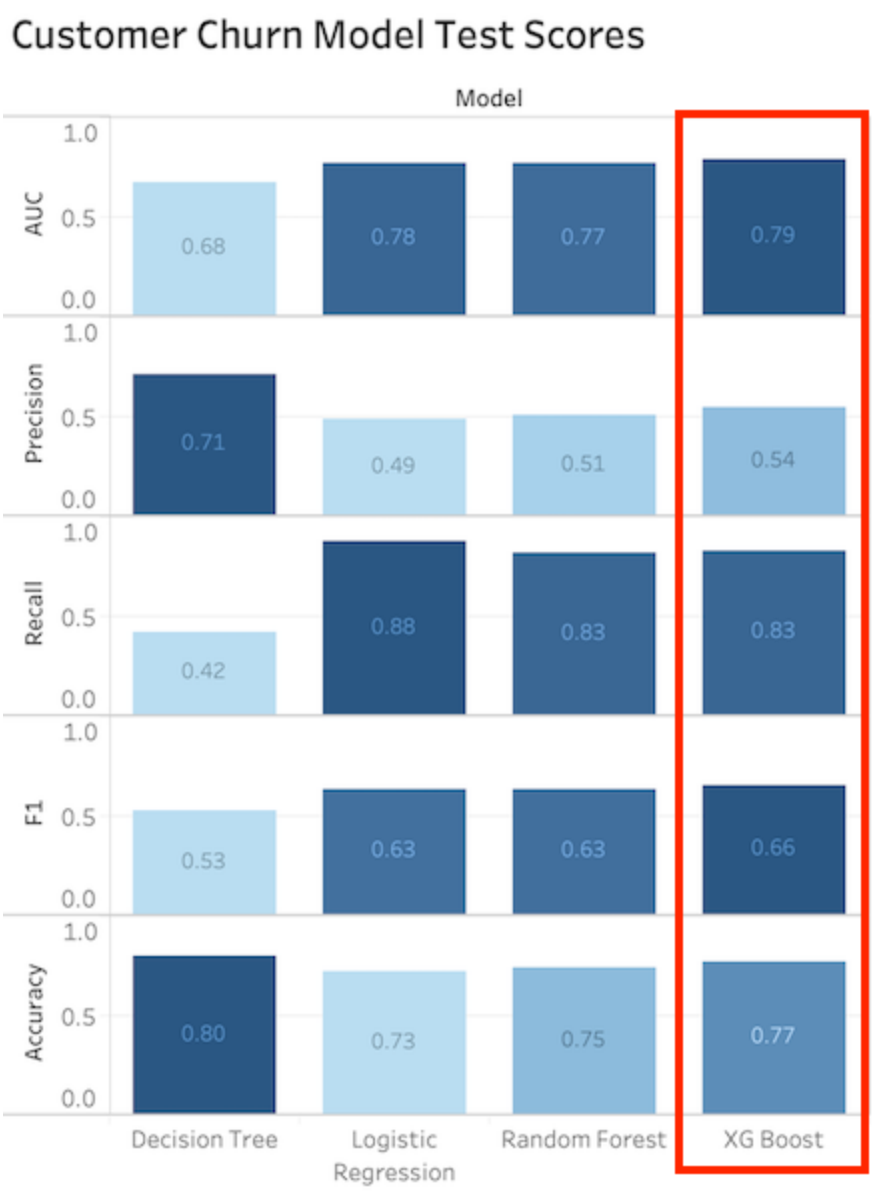


There is a major issue with Month-To-Month contracts which show a 43% churn rate versus customers on term-based contracts (11% and 3%). Higher customer tenure reduces the churn rate on M-T-M contacts, but not until 4–5 years tenure does the churn rate achieve overall average of 26.7%. Clearly M-T-M contracts, and likely the associated pricing, are problematic.From a product perspective, Internet Fiber on M-T-M contracts drives churn rates north of 50%! Note that TelCo always requires a bundling of Phone with Internet Fiber, and the largest group of customers are on M-T-M contracts for Fiber. Customers who add-on "Plus" services for their M-T-M contracts churn slightly less.

Customer tenure positively correlates with higher level of services and monthly charges (with statistically significant p-values). I'm assuming here that customers add services over time. We can also see from the above scatter plot that average monthly charges for M-T-M contracts with any Internet service are actually ***lower*** than term contracts. This is not intuitive, given the churn rate is higher. This likely indicates customers requiring M-T-M contracts are more price-sensitive than those willing to sign-up for terms.

Running Pearson correlations in Python largely confirms associations we've seen in prior plots.

Predicting Customer Churn



I trained four classification models on the 80% training split: Decision Tree, Logistic Regression, Random Forest and XGBoost.

For the latter 3 models, I used GridSearchCV to iterate through relevant parameters and refit the best estimator based on highest mean ROC AUC from 5-fold cross validation. Performance for the latter 3 models was similar.

I chose the XGBoost ensemble model as it had the highest AUC of 0.79, a strong recall of 0.83 and higher precision of 0.54.

I applied standard scaling to 16 features with an objective set to “binary:logistic” to predict the binary churn outcome in the XGBoost model.

A function instantiate_grid takes a model and grid parameters and establishes standard scoring metrics to instantiate the GridSearchCV object.

Through trial and error, I iterated through feature combinations and key parameters for XGBoost to optimize the model’s scoring. For all iterations, I included a positive class weight of 2.8 to correct the modest class imbalance of churn (e.g. 1) occurring 26.7% of time.

The best estimator’s final parameters chose only 4 max depth levels, 100 estimators, and a more conservative L2 regularization of 2.

Using the coefficients of the model as a proxy for feature importance, we see our 1-yr and 2-yr contracts (over the baseline M-T-M) are by far the strongest influence. Internet Fiber products and customers with Dependents had moderate importance, with the remaining features have minor impact.

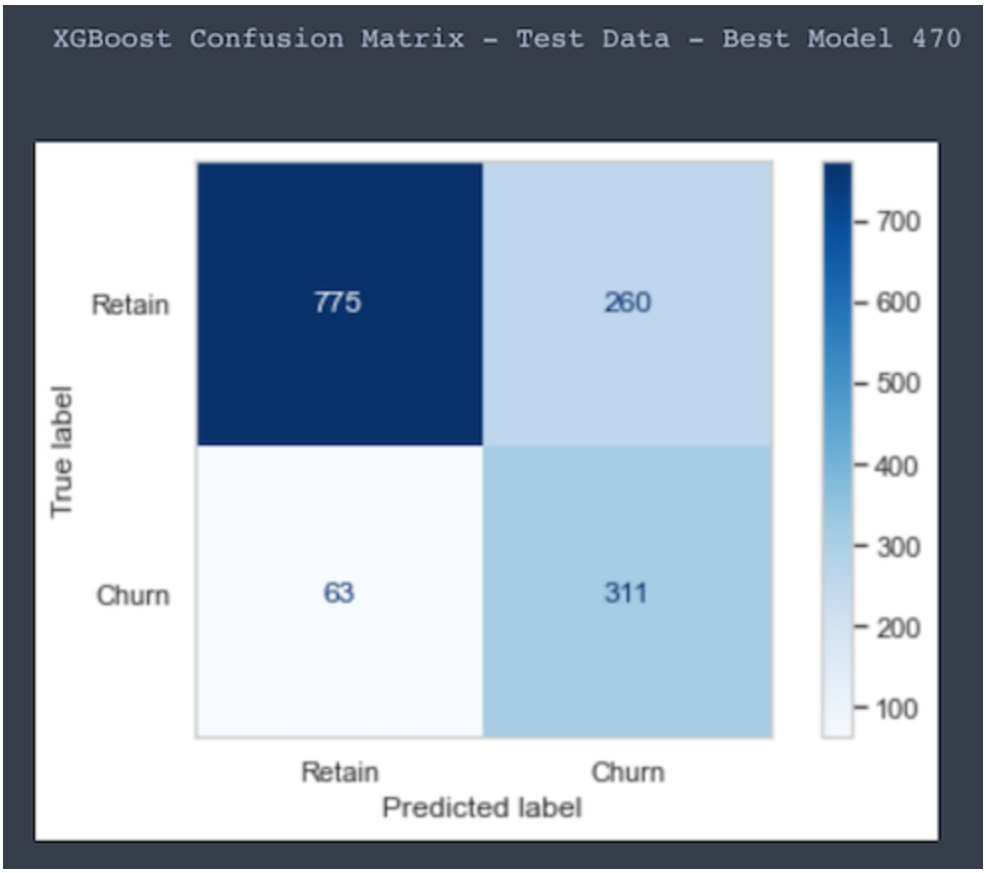
The cross-validation and other parameters helped the model avoid overfitting as the scores on Test were in-line with Train.

```
# compare xgboost best estimator metrics for Train vs Test data sets
print('Train Performance - XGBoost Model\n--------------------------------')
y_hat_train4 = gs_xg.predict(X_train4)
scores_train_xg = score_pred('xg', 'train', y_train, y_hat_train4)

print('\nTest Performance - XGBoost Model\n--------------------------------')
y_hat_test4 = gs_xg.predict(X_test4)
scores_test_xg = score_pred('xg', 'test', y_test, y_hat_test4)
```

```
executed in 40ms, finished 13:08:39 2021-04-21
```

```
Train Performance - XGBoost Model
---------------------------------
Model xg Predictions: AUC 0.81 | Accuracy 0.78 | Recall 0.86 | Precision 0.56 | F1 0.68

Test Performance - XGBoost Model
--------------------------------
Model xg Predictions: AUC 0.79 | Accuracy 0.77 | Recall 0.83 | Precision 0.54 | F1 0.66
```

And so we come to the confusion matrix for our Test predictions.



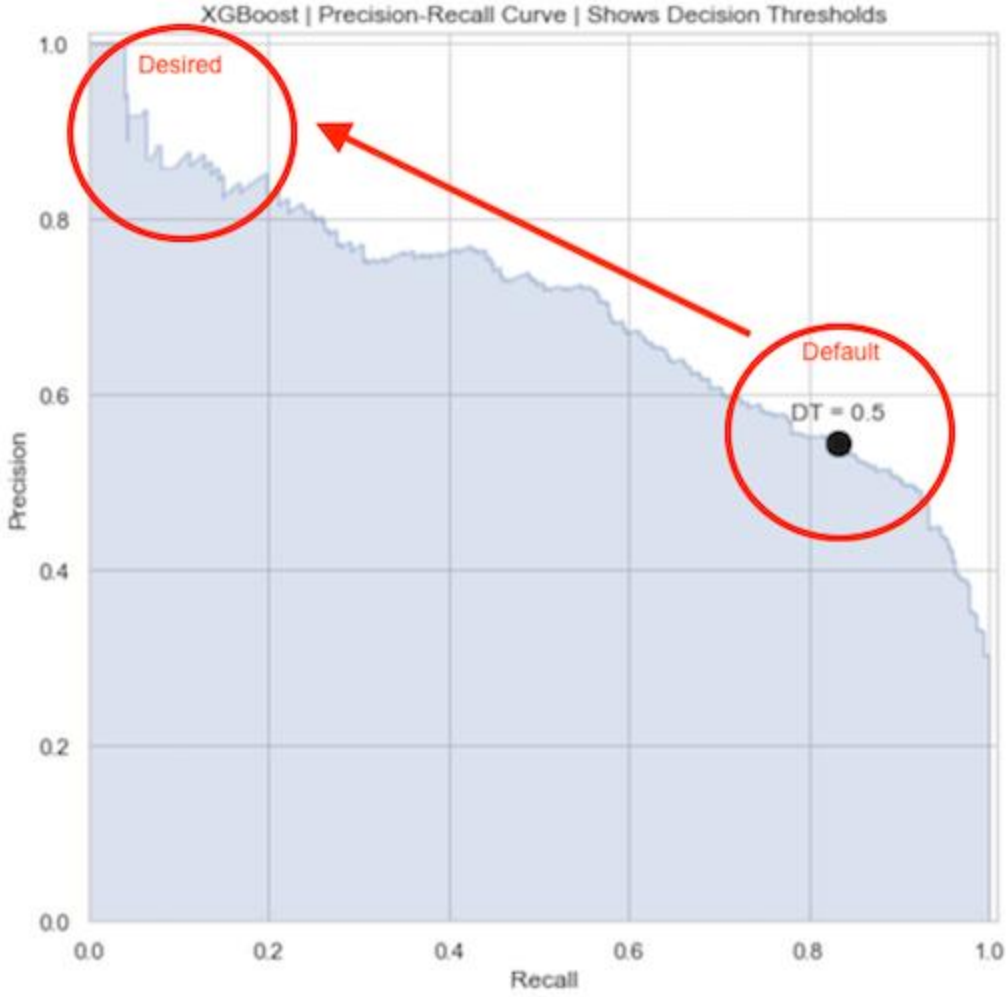
I was pleased with the Test predictive recall of 83% (311/374). Our XGBoost model can correctly predict 5 out 6 true churn cases which is great.

The best model's precision, however, is not acceptable at 54% (311/571). Nearly 1 of 2 churn predictions from our model are not correct (false positives). To deploy the model with confidence, we need better precision.

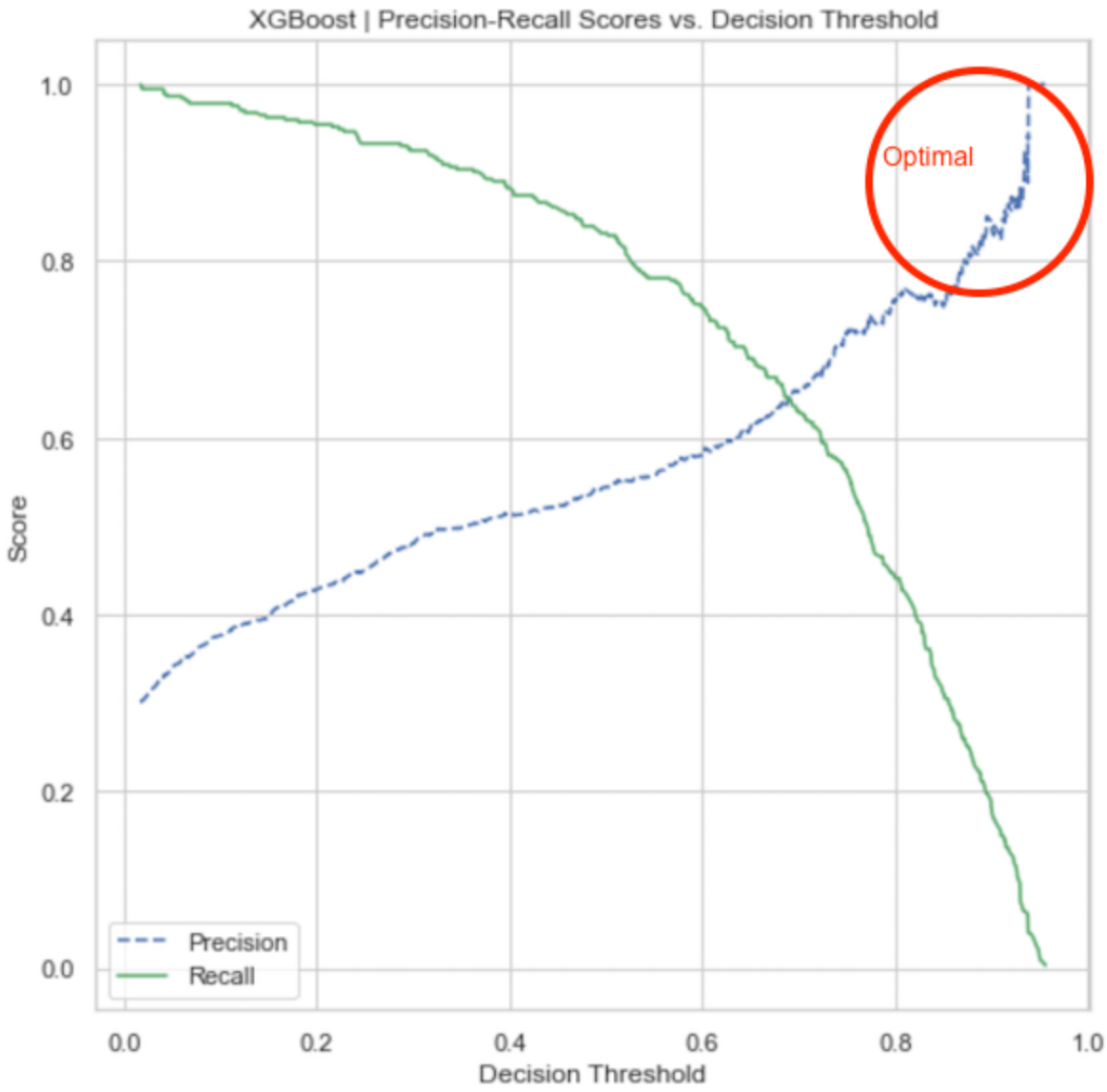
Moving the Decision Threshold

With this model, I can't recommend that TelCo blindly apply default predictions using the decision threshold (DT) of 0.5 churn probability.

By assuming all customers with greater than 50% probability will churn, we may experience excessive costs, customer confusion and possibly unneeded churn. We want to move the decision threshold higher to a churn probability that gives us precision above 80%.



By plotting precision and recall scores as a function of decision threshold, we can see the probability required to achieve the precision we want. While we'll limit the share of churn customers which we can address to under 20% of recall, we'll be more confident in taking action with higher precision.



Using method predict_proba on our best estimator, I reassigned predicted classess for a variety of decision thresholds. I recommend splitting the model into a 3-tier approach, covering these incremental churn cases:

1. **High Precision**: DT=0.9–1.0 / 84% Precision / 18% Recall (67 true pos, 13 false pos)

2. **Medium Precision**: DT=0.8–0.9 / 71% Precision / 26% Recall (98 true pos, 40 false pos)
3. **Low Precision**: DT=0.7–0.8 / 49% Precision / 19% Recall (70 true pos, 72 false pos)

TelCo can target their retention actions according to precision groups, covering 63% of all churn cases in these 3 scenarios:

1. **High Precision** — target most aggressive retention like proactive outreach with discounts or adding free services to retain the mostly likely-to-churn customers.
2. **Medium Precision** — target more general, and lower cost, approaches like empowerment of customer service agents to save customers in certain contexts.
3. **Low Precision** — add these customers to watch lists, or general retention communications efforts.