Churn Predictor

Supervised Learning

Project Background

A little about me... Build a machine learning model in order to predict whether a customer will churn. Currently we have a save mode flag that informs whether a client is expected to churn in 30 days based on a customer request but I'd like to proactively address this.

History lesson... Historically, churn predictors have been attempted but haven't had much success. As a company we were more focused on churn metrics but in the last couple years, we turned our attention to retention tactics by becoming more operationally embedded in the client's day-to-day business and selling less of our Ads and Organic offerings.

I thought you should know... We have a large number of clients with different product packages, price points, onboarding paths, and customer service levels. Therefore it is difficult to treat all customers the same and expect one type of model to take into account all the nuances. I'll be focusing less on the clients on retention products as they have fairly low churn already.

Project Summary

#goals... Create a model that will pinpoint customers that are at high risk of churning based on my feature set.

What are next steps... Come up with the dataset! I used SQL to pull data from our databases. I went back and forth with this a few times in order to get the right mix.

Keep in mind... Two things that I kept in mind while creating this dataset was the cohort I wanted to use and which features to use. There are hundreds of features to pick from and I wanted features that could scale across almost all clients, meaning it would not be null for some and not null for others. Additionally these features had to be high indicators of churn. Secondly, the cohort had to be somewhat uniform in their characteristics and exclude clients that did not have enough data as they would most likely produce outliers. I did this part exclusively in an SQL query.

EXPLORATORY DATA ANALYSIS

Who's Your Data?

of observations: 10,000 clients (OFF = 5032, LIVE = 4968)

Cast.... The customer set had the following characteristics:

1. Currently LIVE or OFF
    a. This is my results variable. LIVE = 0 and OFF = 1
2. Launched within the last 3 years
3. Local channel
4. Not a test account
5. Not a Retention client
    a. These are clients that are on our Lighthouse product that already have a high retention.

    They also have different features from our Ads/Organic product set and therefore I did not want to mix these in using the same features.

Now Featuring....

*30 days is defined as the last 30 days as of today (or when I ran the dataset) for a LIVE client. For OFF clients, it is 30 days prior to their termination date (turned off our service).

- logins_30days - the # of client user dashboard logins
  - Each client has a dashboard that displays various KPI metrics to see how much they've spent, leads received, billing history, etc. I chose this metric as a way to determine a client's engagement. My hypothesis is that a client that logins more frequently in the last 30 days are less likely to churn that one that does not login.
- spend_30days - amount of ad spend with Google and MSN
  - This will vary for various segments. For example, a Lawyer will have a higher budget than a Painter. I chose this metric because clients that are spending less in the last 30 days prior to churning means they haven't refilled their budget and are spending out their remaining balance. Therefore I determined this to be a good predictor for potential churn.
- Invoices_30days - total billed amount
  - This differs from spend since spend is an SEM related metric. All clients get invoiced. A client that is about to churn will most likely not invoice for the following period and therefore a client with zero invoices would be an indication they are not continuing their service.
- leads_30days - # of leads a client received through a phone call or email.
  - A client that is not getting a return on their investment is obviously more likely to churn. A client that has low lead count in the last 30 days will be more likely to churn.
- days_live - the count of days in which the client status is LIVE.
  - A client that has been with us longer will have a higher days live. A client with a higher days live indicate they're happy with our service and are more likely to stay.

MODEL EVALUATION

Test/Train on Entire Dataset vs Split

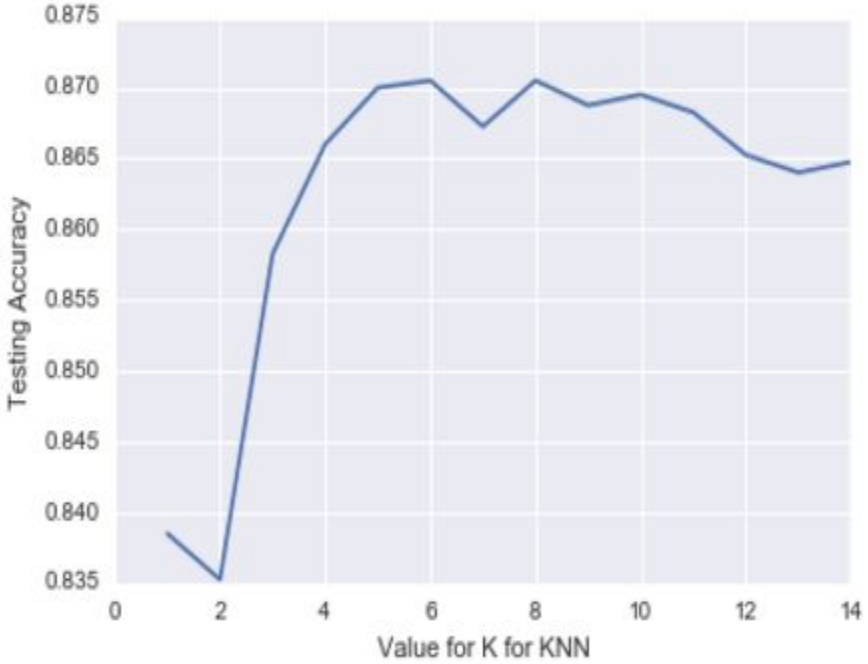
1) Define X and y.
   a) X = feature columns
   b) Y = churned
2) Fit Logistic Regression model with data
3) Predict the response values for observations in X
4) Results: **Classification accuracy (80%) ; KNN=5 (90%) ; KNN=1 (99.6%)**

1. Split X and Y into training and testing sets
2. Train the model on the training set using Logistic Regression
3. Make predictions on the training set
4. Compare actual response values (y_test) with predicted response values (y_pred)
5. Results: **Classification accuracy (80%) ; KNN=5 (87%) ; KNN=1 (84%)**

Although K=1 has better accuracy, this does not mean this is better. KNN looks for the nearest observation and uses it as the predicted response value for the unknown observation. Therefore K=1 will most likely always be near 1. It has memorized the training set and we're testing on the same data. Therefore this is not useful procedure to choosing our model.

Our goal is to estimate likely performance of a model on out-of-sample data. Maximizing training accuracy rewards overly complex models that won't necessarily generalize which overfit the training data. Lower value of K will learn the noise in the data, instead of the signal.

Tuning: Can we find a better value for K?



Plot the relationship between K and testing accuracy

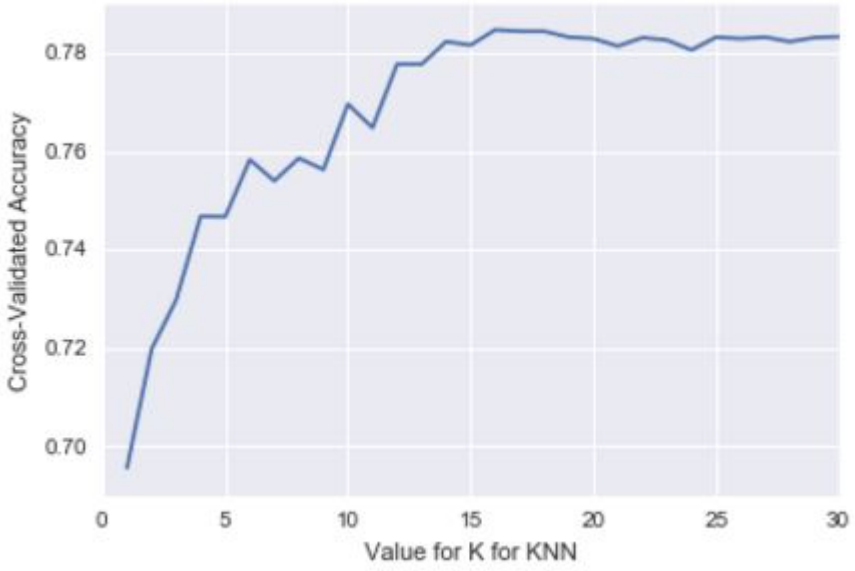
In general, as K increases, there's a rise in the testing accuracy and then a fall. This rise and fall is typical when examining the relationship between model complexity and testing accuracy.

Training accuracy rises as model complexity increases. For KNN, this is determined by value of K.

Testing accuracy penalizes models that are too complex or not complex enough. Maximum testing accuracy when model has the right amount of complexity. Here, we see the highest K around K=5, 6, and 8. Let's choose K=5 as the best parameter.

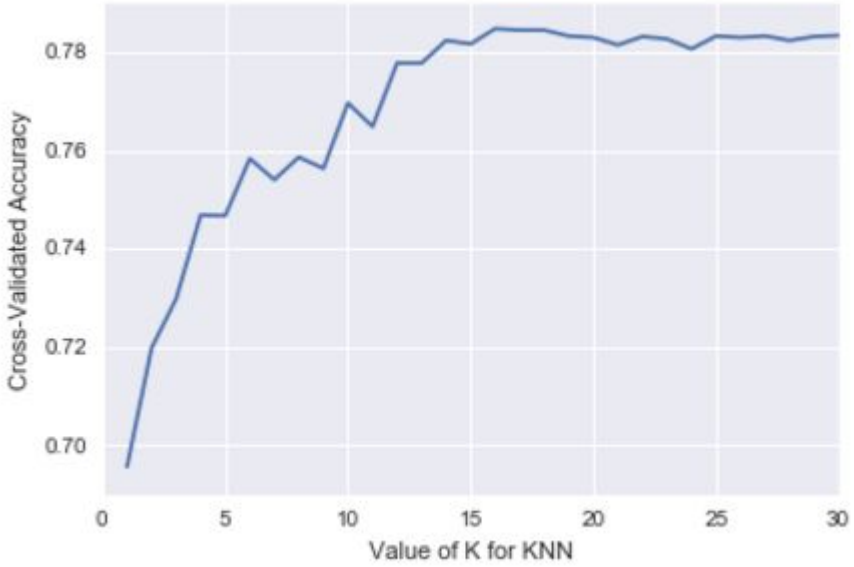
Cross Validation: Tuning & Model Selection

1. Train/Test Split can change a lot depending what observations happen to be in training vs testing set. CV largely overcomes this limitation by repeating train test split multiple times and averages those results. It is more accurate estimate of out-of-sample accuracy.
2. Dataset contains 10,000 observations, 10-fold CV runs for 10 iterations, For each iteration, every observation is either in the training set or testing set. Every observation is in the testing set once.
3. 10-fold cross validation with K=5 for KNN.
4. Mean score → **77%**
5. Search for optimal value of K between range 1-30
6. Plot the value of K for KNN versus the cross validated accuracy
7. 10-fold CV with best KNN model (n_neighbors=16): **78%**
8. 10-fold CV with Logistic Regression: **79%**



Logistic Regression and best KNN performed almost the same. Either would work for model selection.

GridSearchCV



Cross validation can be computationally expensive especially when dataset is large. GridSearchCV is a more efficient tuning parameter that allows you to define a grid of parameters that will searched using K-fold cross-validation.

1. Define parameter that should be searched (k_range: 1-30)
2. Create a parameter grid and do 10 fold CV on a KNN model using classification accuracy as evaluation metric.
3. Fit the grid with data (X, y)
4. Create a list of the mean scores
5. Plot the results
6. Examine the best model
    a. Best score: **78%**
    b. Best params: **n_neighbors=16**
    c. Best estimator: **weights = uniform** and other default parameters

CONFUSION MATRIX

Confusion Matrix

n=2500	Predicted: 0	Predicted: 1	
Actual: 0	TN = 891	FP = 343	1234
Actual: 1	FN = 195	TP = 1071	1266
	1086	1414	

True Positive (TP): We correctly predicted that the customer churned.
True Negative (TN): We correctly predicted that the customer did not churn.
False Positive (FP): We incorrectly predicted that the customer churned.
False Negative (FN): We incorrectly predicted that the customer did not churn.

"Classification Accuracy": How often is the classifier correct?
(TP+TN) / (TP+TN+FP+FN) = 1962/2500 = **78%**

"Classification Error": How often is the classifier incorrect? Also misclassification rate.
(FP + FN) /(TP+TN+FP+FN) =538/2500 = **22%**

What would Confus-cius do?

n=2500	Predicted: 0	Predicted: 1	
Actual: 0	TN = 891	FP = 343	1234
Actual: 1	FN = 195	TP = 1071	1266
	1086	1414	

"**Sensitivity/Recall/True Positive Rate**": when the actual value is positive, how often is the prediction correct?
TP/ (TP + FN) = 1071/ 1266 = **85%**

"Specificity": when the actual value is negative, how often is the prediction correct?
TN/ (TN + FP) = 891/1234 = .7220 = **72%**

"False Positive Rate': when the actual value is negative, how often is the prediction correct?
FP/ (TN + FP) = 343/1234 = **28%**

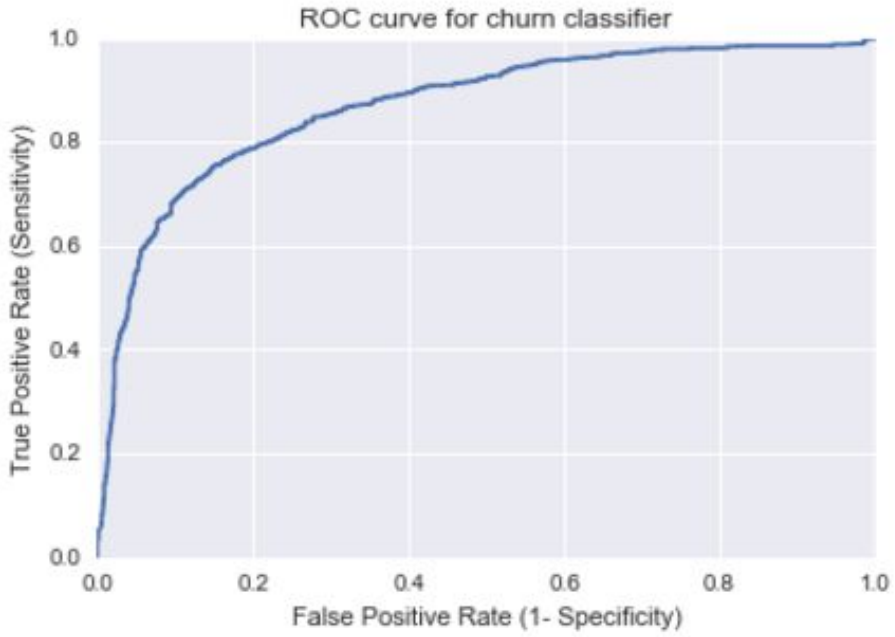
"Precision": when a positive value is predicted, how often is the prediction correct?
TP/ (TP+FP) = 1071/1414 = **76%**

Please be nice, I'm a sensitive model...

Optimize for sensitivity because false positives (incorrectly predicting a customer will churn) is more acceptable than false negatives (incorrectly predicting a customer will not churn).

Lifetime Recurring Revenue is more important than small marketing dollars that would be spent saving a client that we predicted would churn but actually did not.

ROC/AUC Curve



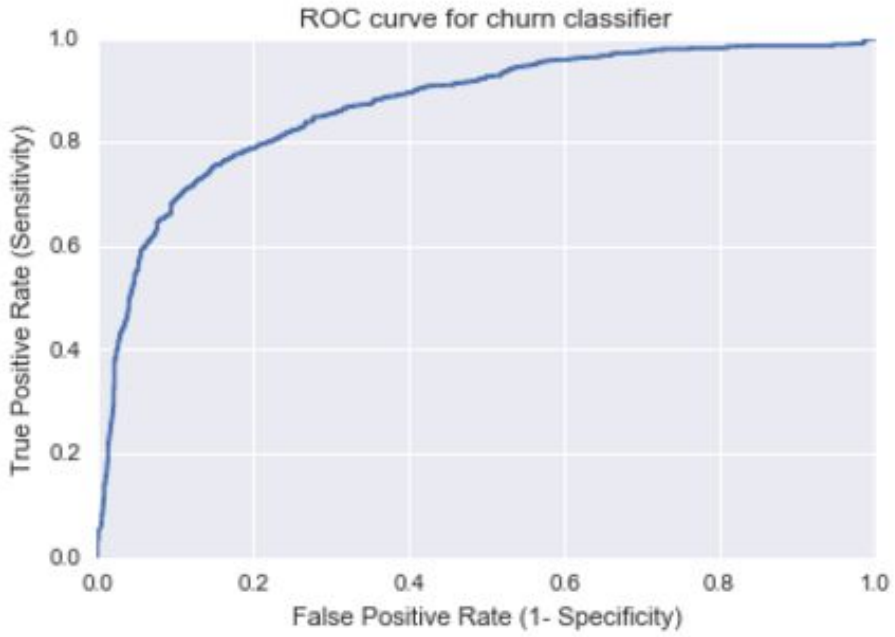
Y-axis is the True Positive Rate (true positives/all positives), which answers the question when the actual classification is positive, meaning churned, how often does the classifier predict positive.

X-axis is the False Positive Rate (false positives/all negatives), which answers the question when the actual classification is negative, meaning not churned, how often does the classifier incorrectly predict positive.

ROC curve plots all possible thresholds for (x, y) or (TPR, FPR) which range from 0 to 1 versus misclassification rate is only for a single threshold. Note you can't see the thresholds on the ROC curve.

A good ROC curve that does a good job at separating the classes will have a high TPR and a low FPR and therefore will hug the upper left corner of the plot. Conversely, a classifier that does a poor job of separating the classes will have an ROC close to the diagonal of the plot, which represents a classifier that does no better than random guessing.

ROC/AUC Curve



Area Under the Curve is the percentage of the box that is underneath the curve. A bad classifier would have an AUC close to 0.5 which is random guessing. A good classifier would be close to 1.0, a perfect classifier.

Threshold (0.5) → Sensitivity: .85; Specificity: .72

Threshold (0.3) → Sensitivity: .91; Specificity: .57

The ROC can help choose a threshold that balances sensitivity and specificity. For example, to accept a sensitivity of .91, I must be willing to accept a specificity of .57.

ROC AUC Score → **87%**

FEATURE IMPORTANCE

Logistic Regression vs Random Forest

1. Standard Scalar
2. Fit Logistic Regression
3. Coefficient Results
    a. days_live: -0.2057
    b. invoices_30days: -0.1609
    c. leads_30days: -0.0448
    d. spend_30days: 0.0183
    e. logins_30days: -0.0037

Except for my logins_30days feature, all my other P values were close to zero and confidence interval did not include zero.

ROC AUC Score - **87%**

1. Fit Decision Tree Classifier
2. Importance Score
    a. invoices_30days: 0.4820
    b. days_live: 0.2729
    c. leads_30days: 0.1084
    d. spend_30days: 0.1004
    e. logins_30days: 0.0363

This model also categorized logins as least important and invoices/days live as most important and leads and spend as in the middle.

ROC AUC Score - **62%**

Results Summary

1. Model Evaluation
    a. Training and testing on the same data
        i. Classification accuracy **(80%)**
        ii. KNN=5 **(90%)** vs KNN=1 **(99.6%)**
    b. Train/test split
        i. Classification accuracy **(80%)**
        ii. KNN=5 **(87%)** vs KNN=1 **(84%)**
    c. K-fold cross validation/GridSearchCV
        i. 10 fold cross validation with K=5 for KNN → Mean score: **(77%)**
        ii. 10 fold cross validation with n_neighbors=16 with best KNN model → **(78%)**
        iii. 10 fold cross validation with Logistic Regression → **(79%)**
2. Confusion Matrix
    a. Classification accuracy **(78%)**
    b. Sensitivity (with 0.5 threshold): **(85%)** vs Specificity (with 0.5 threshold): **(72%)**
    c. ROC/AUC Curve: **(87%)**
3. Feature Importance:
    a. Random Forest CV ROC AUC: **(67%)**
    b. Logistic Regression CV ROC AUC: **(87%)**
4.

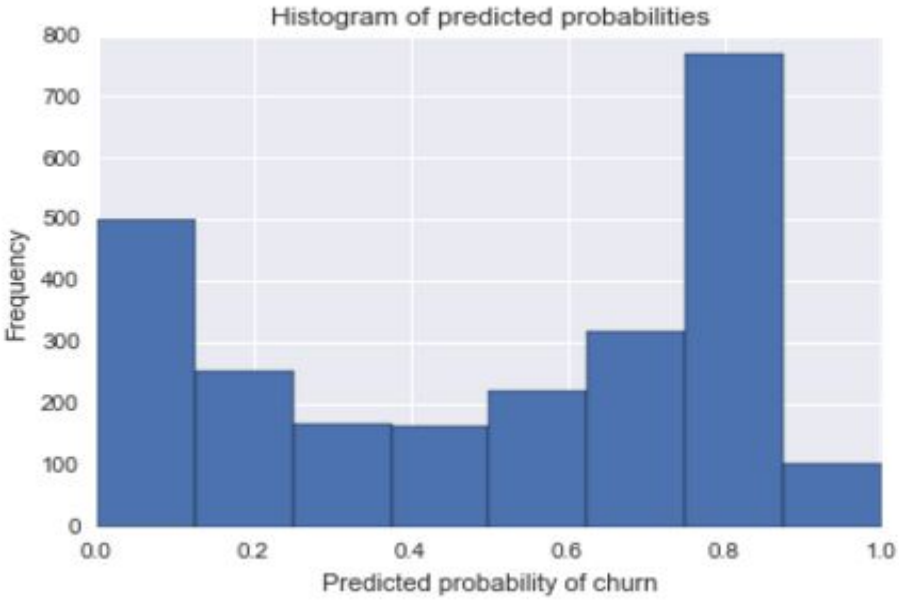
Conclusion

Select Logistic Regression as my best model based on this cohort and feature set based on my results.

Running predicted probabilities of churn shows about ~880 of my accounts have a >80% chance of churning.

I believe that my dataset and feature selection have the biggest impact. When looking for optimal parameters, we saw that the accuracy score between KNN vs Logistic Regression didn't have a big difference in terms of accuracy.

It appears that features and cohort size are more influential when using a Decision Tree Classifier as we saw huge differences in the ROC/AUC score and may be something to test when expanding the features and cohort



What's Next?

Things to try...

1. Expand cohort - Currently it's limited to 10K clients and fall within a certain cohort that doesn't cover the majority of our client base and product offerings. I'd like to be able to understand how to improve churn amongst the various channels/products. One model may not fit all types of clients and use the same features as not all features scale across all accounts.
2. Sampling techniques - use different sampling techniques to efficiently do GridSearchCV for a larger client base.
3. Feature importance - testing more features and see if it improves model accuracy as we saw logins wasn't important as I initially thought.
4. Random Forest with more features and larger sample size.

Actionable insights and next steps...

1. Set up a report in our automated reporting system that spits out predicted probabilities using the Logistic Regression that fall >=80% chance of churning ordered by highest churn rate to lowest.
2. Have Retention team go through this list and proactively address issues such as low lead count or other underperforming issues. The goal is to provide our retention team clients that are at high risk of churning and enact save tactics proactively. Additionally, being able to identify the features that are more highly correlated to churn to focus company efforts.

That’s All Folks!

Questions?