Churn Prediction using Interpretable Machine Learning

Dhaval Potdar

Table of Contents

1 Background *2*

1.1 Introduction 2

1.2 Data Description 2

2 Methods *3*

2.1 Globally Optimized Sparse Decision Trees 3

2.2 Generalized Additive Models 3

2.3 Sparse Generalized Linear and Additive Models 4

2.4 Explainable Boosting Machines 4

3 Experiments *4*

3.1 Results 4

3.1.1 GOSDT 5

3.1.2 GAM 5

3.1.3 L0Learn 6

3.1.4 EBM 7

3.2 Hyperparameter Selection 9

3.3 Variable Importance Analysis 9

4 Insight *10*

5 Conclusion *10*

6 Citations *11*

1 Background

1.1 Introduction

Churn prediction is a classic problem in Machine Learning. A quick survey of Kaggle shows numerous churn prediction competitions being hosted every year by data-rich companies such as online retailers and network providers. Most of these competitions are won by black-box models such as highly tuned Random Forests or the all-time classic XGBoost. Although they achieve state-of-the-art performance in terms of classification accuracy, they leave little room for interpretability. They can make individual predictions with high confidence but fall apart when someone starts probing the *whys*. The aim of this project is two-fold: (i) to describe a whole suite of models that preserve both global interpretability and local explainability, and (ii) to make the case that interpretable models are the best way to gain actionable insight into the data they are trained on, and why one might accept a slight drop in predictive capability in favor of interpretability.

1.2 Data Description

The data comes from Maven Analytics, 2022[1] as part of their churn prediction challenge. The main table contains information on all 7,043 customers from a telecommunications company in California in Q2 2022. An additional table contains the zip-code level population, which I join with the main dataset. Each record represents one customer and contains details about their demographics, location, tenure, subscription services, and status for the quarter (joined, stayed, or churned). I ignore the "joined" category (454 customers), thus framing this as a binary classification problem.

Observing several rare values in the city column I collapse all cities with a frequency of less than 6 into one category. I impute the missing values in 10 of the categorical columns such as Internet Type, Premium Tech Support, etc. by "NA", since no explanation for their missingness is found at the source. Lastly, I impute "Avg Monthly GB Download" and "Avg Monthly Long-Distance Charges" using the Multi-iteration Stochastic Estimator[2] algorithm.

To make sense of the data I use PaCMAP[3] to see if any natural separation occurs between the customers who churn vs. those who don't. Although several separations occur, none naturally align with the target classification.

[1] (Maven Analytics, 2022)
[2] (Carlon, Espath, Lopez, & Tempone, 2020)
[3] (Wang, Huang, Rudin, & Shaposhnik, 2021)

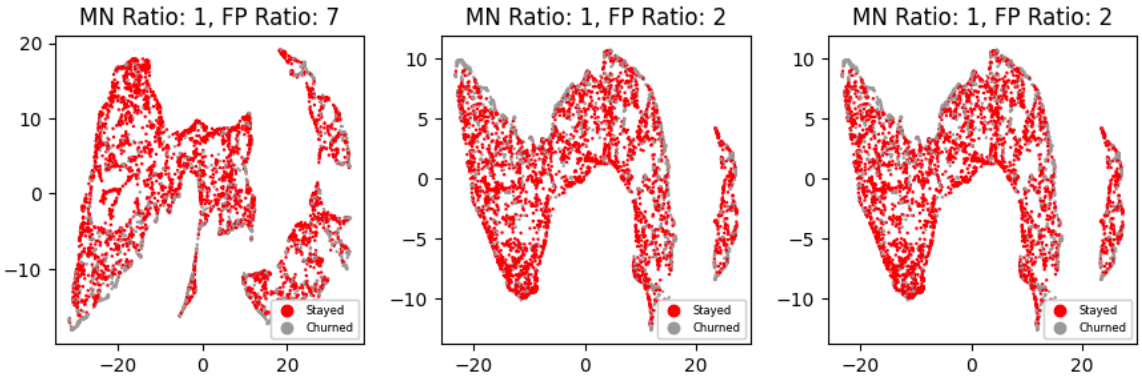

Figure 1: PaCMAP results

2 Methods

It is not enough to say which customer will churn. Identifying *why* the customer will churn is equally important. Keeping interpretability at the forefront, while minimizing the performance cost heavily informs the choice of algorithms for this problem. The sections below discuss the four algorithms used.

2.1 Globally Optimized Sparse Decision Trees

Also known as GOSDT[4], this algorithm uses branch-and-bound methods to fully optimize decision trees. Traditional decision trees such as C4.5[5] and CART[6] use greedy splitting and pruning to build trees that fit the data, which don't guarantee optimality. GOSDTs on the other hand, aim for full minimization of the cost complexity objective. They use the pairing of theoretical bounds for reducing the search space with careful storage and lookup of previously solved subproblems. In short, they use analytical bounds to limit the search space at each new split and use dynamic programming in solving sub-problems generated by each new split. The main advantage of GOSDTs over traditional decision trees is that they produce simple and near-optimal trees that are sparse and easy to interpret.

2.2 Generalized Additive Models

Also known as GAMs[7], these models make two assumptions: (a.) relationships between individual predictors and the dependent variable follow smooth patterns that can be linear or non-linear, and (b.) we can estimate these relationships simultaneously and then predict $g(E(Y))$, by adding them up, where g denotes the link function. The GAM model can be written as $g\big(E(Y)\big) = \alpha + s_1(x_1) + s_2(x_2) + \cdots s_p(x_p)$, where s_p denotes a smooth non-parametric function. For this project, I use the logit link function $g\big(E(Y)\big) = \log\frac{P(Y=1)}{P(Y=0)}$. The advantage of GAMs is that they allow a

[4] (Hu, Rudin, & Seltzer, 2019)
[5] (Quinlan, 1993)
[6] (Breiman, Friedman, Charles, & Olshen, 1984)
[7] (Hastie & Tibshirani, 1986)

direct relationship between a predictor variable and the log odds of the outcome variable to be visualized after training.

2.3 Sparse Generalized Linear and Additive Models

These slightly more modern (and optimized) variants of GAMs use L0 regularization to punish some parameters to zero. For this project, I use L0Learn[8], which is a fast L0 regularized framework that can handle 50,000 features in a few minutes. More specifically I use the L0Learn model with L0 penalty and Logistic loss. The advantage of L0Learn over GAMs is that it performs fast feature selection and allows for the coefficient values to be visualized as the support size (number of features) increases. This leads to an interpretable model that uses only a small subset of the full feature space.

2.4 Explainable Boosting Machines

EBM is a glass-box model that is designed to thwart the interpretability vs. accuracy trade-off. At its core, an EBM is a GAM that uses bagging and gradient boosting to learn each feature function f_i. EBMs can automatically detect and include pairwise interaction terms. The equation of an EBM thus becomes: $g(E(y)) = \beta_0 + \sum f_i(x_i) + \sum f_{i,j}(x_i, x_j)$. Each feature function f_i acts as a lookup table per feature and returns a term contribution. EBMs are highly intelligible because the contribution of each feature to a final prediction can be visualized and understood by plotting. Besides they give results comparable to powerful black-box models such as XGBoost. For this project, I use IntrepretML[9], a framework that implements EBMs.

3 Experiments

3.1 Results

The table below shows the performance results of each model discussed here, along with the best competition-winning performance reported on Kaggle for this dataset.

Model	F1 Score	Interpretable
Random Forest (Kaggle)[10]	90.93%	No
EBM	77.17%	Yes
GAM	76.27%	Yes
L0Learn	73.74%	Yes
GOSDT	73.44%	Yes

Table 1: Model Performances

The following sections will showcase the interpretability of each model implemented in this project and make the case for why anyone would accept lower performance in favor of complete interpretability.

[8] (Hazimeh, Mazumder, & Nonet, 2022)
[9] (Nori, Jenkins, Koch, & Caruana, 2019)
[10] (Saad, 2022)

3.1.1 GOSDT

GOSDT gives a highly interpretable sparse tree that is provably optimal. The figure below describes the process to decide whether a certain customer will churn. The interpretability of this tree clearly outlines the interventions that must be made by the telecom company to reduce the churn rate. For example, customers who are not subscribed to a month-to-month contract are the group that the company should not focus its retention strategies on. Further, among customers less than 60 years old and those who have monthly charges of less than 70$, the possibility of churn is higher if the tenure is less than 3.5 months. This means that, within this group, the company should focus on getting more customers to subscribe to plans that have a longer tenure.

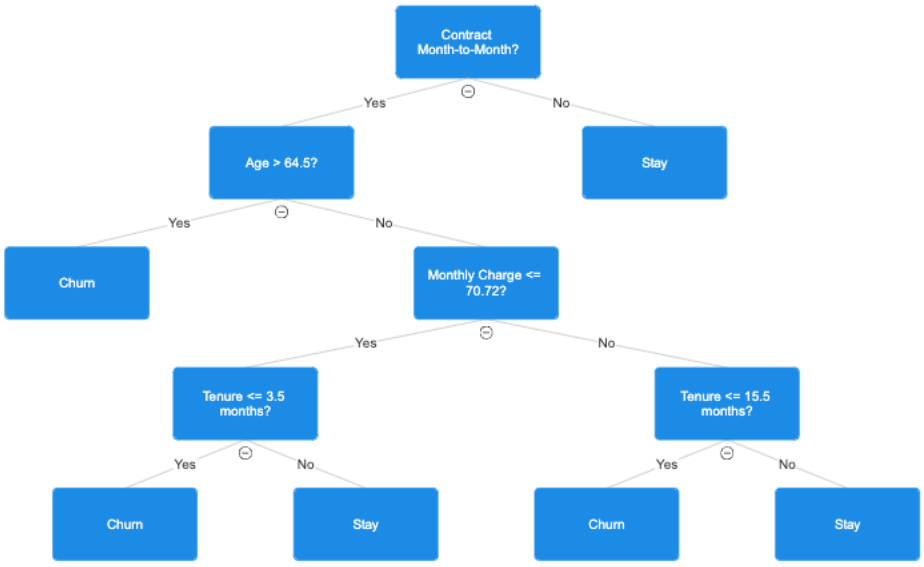

Figure 2: Interpretable and provably optimal tree given by GOSDT

3.1.2 GAM

I fit the GAM model using 6 spline terms, 9 factor terms, and an intercept. Since GAM does not do automatic feature selection, I had to narrow down the feature space based on the EDA. The figure below shows the spline/factor function for each of the 15 variables, with the log-odds on the y-axis. The plot for tenure shows that as it increases, the odds of a customer churning decrease; this aligns with my intuition. The three contract-related plots in the middle show that the longer the contract, the lower the odds of churn. Customers on a month-to-month contract are most likely to churn; this too aligns with intuition. What doesn't align with intuition, is that as the number of dependents approaches 6, there is a bump in the possibility of churn. This could simply be because of noise in the dataset, but if it isn't, this bump points to a specific target group that the company could focus on to bring down the overall churn rate. Whether it is an actual insight or noise, can only be determined following a deeper analysis.

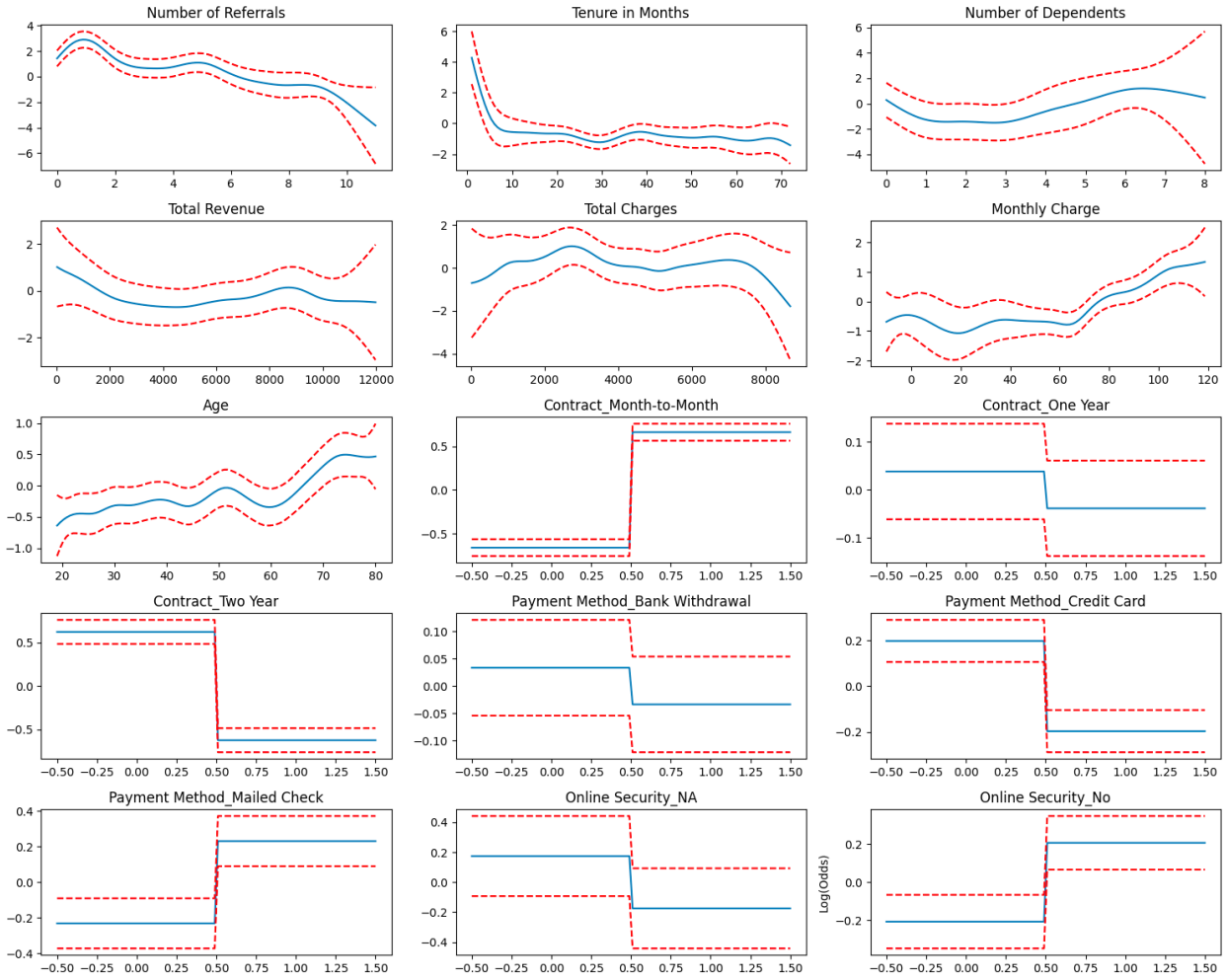

Figure 3: Insights from GAM

3.1.3 L0Learn

L0Learn had the advantage of L0 regularization and thus fast feature selection. I did a search over 4 values for the parameter *lambda* which governs how strongly features are penalized. The table below shows how the support size (the number of features used by the model) decreases as *lambda* increases.

Lambda	Support Size	Loss
10	13	0.346
4	15	0.343
2	19	0.340
1	31	0.341

Table 2: Tuning Lambda for L0Learn

In the figure below each dashed line represents the regularization path of a feature and the circles represent the support sizes. The most important features are those with a non-zero coefficient, regardless of the support size. The pink line at the top represents the coefficient for the binary

feature “Contract Month-to-month”. Notice that it has the highest magnitude of coefficient value, and thus is the most important feature in predicting churn. This is consistent with GOSDT, which chose this feature as the root node.

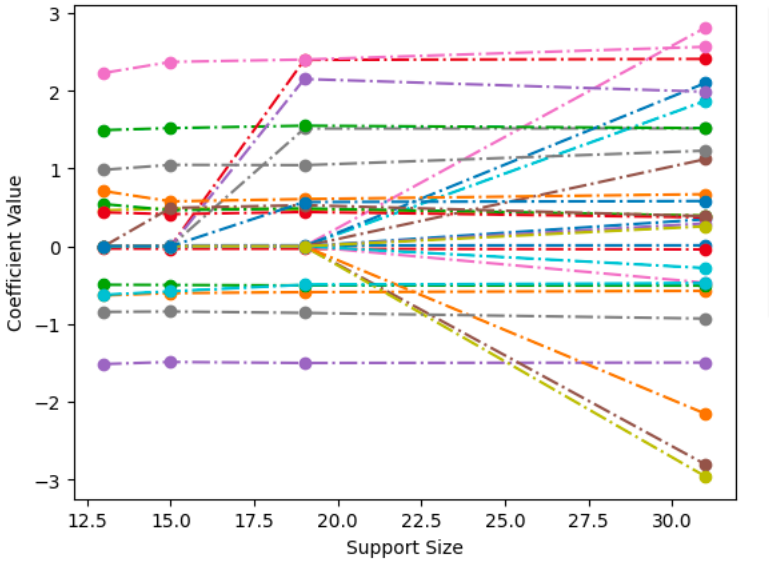

Figure 4: Regularization paths for features given to L0Learn

3.1.4 EBM

EBM gave the best F1 score of the models used in this project. The graph below shows the global feature importances of the fitted model. Contract is again the most important feature in predicting customer churn.

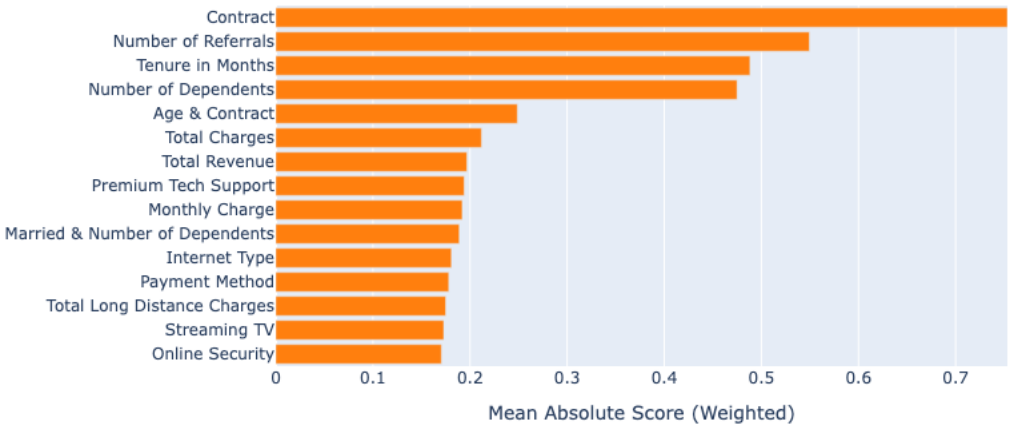

Figure 5: Global feature importances of EBM

Like GAM, EBM can visualize the relationship between a single predictor and the log odds of the outcome. As was the case with GAM, a longer tenure here too leads to a lower churn possibility.

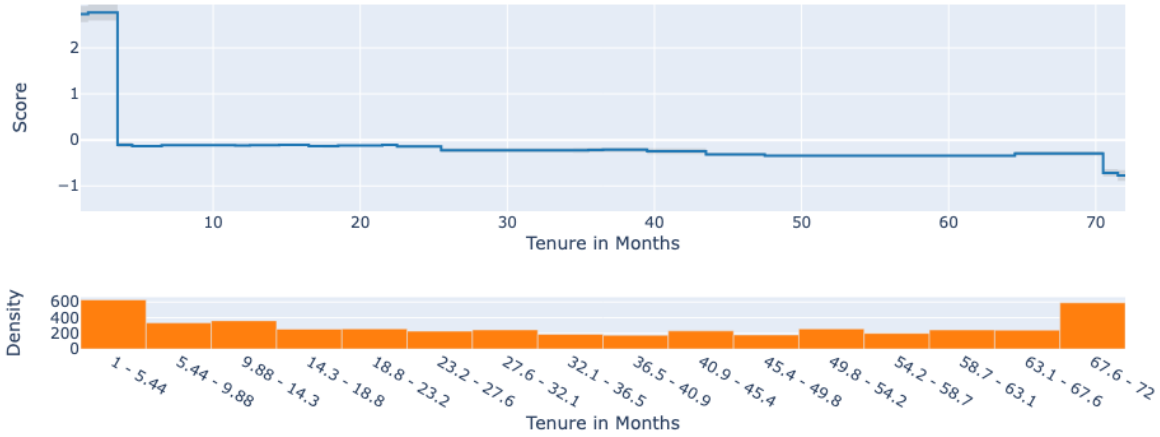

Figure 6: Tenure v/s Log Odds of Churn

The Python library that implements EBM also gives a convenient way to identify the contributions for each feature toward the prediction. Below is an example of a specific positive-class prediction made by the model.

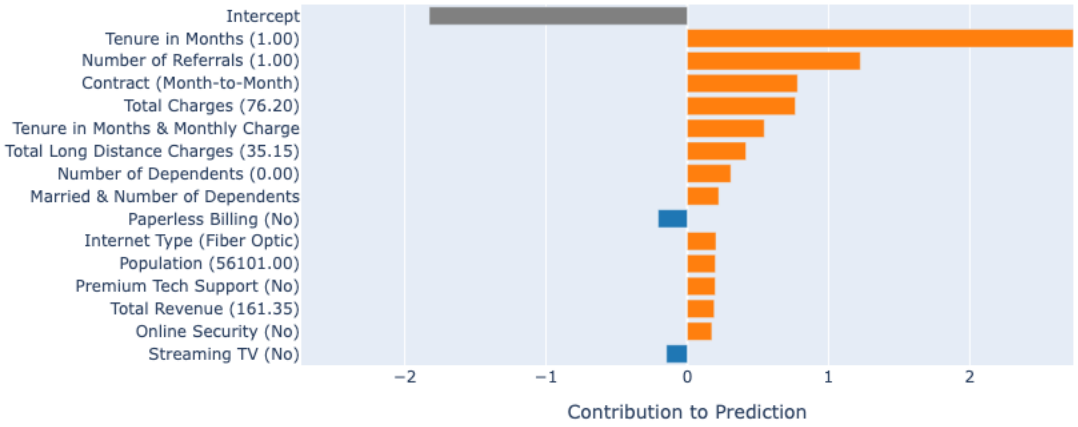

Figure 7: Local Explanation by EBM (Predicted Class: Churn; P(Y=Churn) = 0.998)

One can see that for an acceptable reduction in predictive performance, complete transparency of predictions can be achieved. With high-stakes decisions, interpretability trumps performance to a certain degree. To what extent one values interpretability over performance, of course, depends on the specific use case, margin of error, health risk, and a multitude of factors. In this project, I make the case that data scientists can use a wide array of tools that preserve the interpretability of predictions and that this allows complex decisions to be made with complete end-to-end transparency.

3.2 Hyperparameter Selection

All models were tuned using Grid Search with cross-validation. Model evaluation was done using the F1 Score and the confusion matrix. I use a 70-30 stratified test split to report the performance of each model. Complexity penalty (inherent to GOSDT), and regularization were used to prevent overfitting. Regularization (L0) was also used to limit the support size, to have a more interpretable model. The plot below shows the relationship between the L0 penalty, the resultant support size, and the corresponding F1 score on the test set.

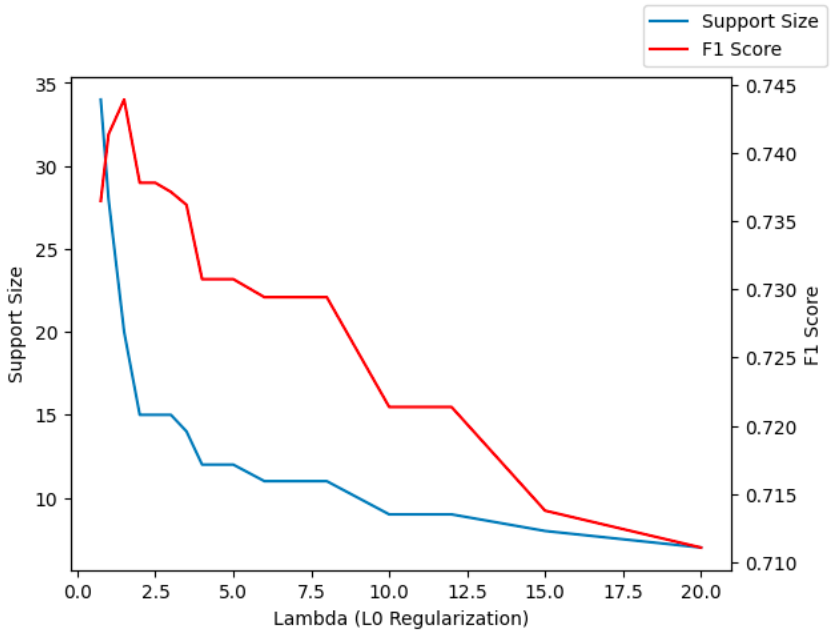

Figure 8: Functional Relationship between Lambda and F1 Score for L0Learn

We see the best overall performance at $\lambda = 1.5$. The support size at this point is 20, which is still interpretable. However, for a reduction of 3.2% in the F1 Score, we get a model that merely uses 7 features in the complete dataset, making it highly interpretable. For the use case of churn prediction, depending on factors like the company budget for interventions, this drop in performance in favor of greatly increased interpretability may be acceptable.

3.3 Variable Importance Analysis

For doing generic variable importance analysis, I fit a decision tree to the data, tune the hyperparameters, and use the Python library eli5 to compute feature importances. This library calculates the feature importances by replacing different columns in the test data with random noise and noting the drop in performance. The table below shows some of the most important features that emerge through this process. The results are consistent with all four models discussed previously.

Weight	Feature
0.1172 ± 0.0052	Contract = Month-to-Month
0.0735 ± 0.0087	Tenure in Months
0.0313 ± 0.0060	Age
0.0309 ± 0.0082	Number of Referrals
0.0185 ± 0.0038	Monthly Charge
0.0094 ± 0.0039	Number of Dependents
0.0060 ± 0.0033	Total Charges
0.0035 ± 0.0033	City = San Diego

Table 3: Global Variable Importances

4 Insight

Several insights about the data emerged from this project:

- People who are in a month-to-month contract with the telecom company are most likely to churn; this is as expected.
- After being subscribed for about 4 months to the company, there is a near-vertical drop in the likelihood of churn. After this point, there is no further tangible drop. This means that the telecom company should expend vastly more effort in retaining customers up to the 4-month mark; things get easier after that.
- A surprising insight that came from the analysis is that Age is positively correlated with odds of churn. One would assume that younger customers would be more proactive in switching between network providers and try out different options, but as it turns out, the exact opposite is true.
- Finally, customers who pay by credit card or bank withdrawal are less likely to churn compared to those who mail a check. This should tell the telecom company to incentivize the former two options over the latter one by giving attractive offers or cash-back opportunities.

5 Conclusion

Although the predictive performance of interpretable models did not match that of black-box ensemble methods, interpretable models give deep insights into the data. Not only do they highlight global trends and observable relationships, but they also give local explanations for every single prediction they make. These models can be fine-tuned to minimize false negatives or false positives, and in that respect, get comparable performance to black-box models with the added advantage of complete end-to-end transparency. In practice, high-stakes decisions are rarely made based on predictions of black-box models that nobody can vouch for. Interpretable models can aid human decision-making, rather than substituting it, and thus help data science take a long stride in the direction of “open” AI.

6 Citations

Breiman, L., Friedman, J., Charles, S. J., & Olshen, R. A. (1984). *Classification and Regression Trees.*

Carlon, A., Espath, L., Lopez, R., & Tempone, R. (2020, November). Multi-Iteration Stochastic Optimizers.

Hastie, T., & Tibshirani, R. (1986). Generalized Additive Models. *Statistical Science, 1*(3), 297-318.

Hazimeh, H., Mazumder, R., & Nonet, T. (2022). L0Learn: A Scalable Package for Sparse Learning using L0 Regularization.

Hu, X., Rudin, C., & Seltzer, M. (2019). Optimal Sparse Decision Trees. *Advances in Neural Information Processing Systems*, 7267–7275.

Maven Analytics. (2022). *Maven Churn Challenge*. Retrieved from https://www.mavenanalytics.io/blog/maven-churn-challenge

Nori, H., Jenkins, S., Koch, P., & Caruana, R. (2019). InterpretML: A Unified Framework for Machine Learning Interpretability.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning.* Baltimore, Maryland, United States of America.

Saad, Z. (2022). *Customer Churn Prediction on Telecom Dataset*. Retrieved from Kaggle: https://www.kaggle.com/code/zakriasaad1/customer-churn-prediction-on-telecom-dataset

Wang, Y., Huang, H., Rudin, C., & Shaposhnik, Y. (2021). Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization. *Journal of Machine Learning Research, 22*, 1-73.