Machine Learning Engineer Nanodegree

Capstone Project

Scott Etheridge
November 5th, 2018

Project Overview

The Telecom industry has experienced greatly increased levels of competition over the last decade. While the competition is coming from the traditional Telecommunications and Cable industry players, it is also coming from companies in completely different industries. The ability to provide telecommunications service over the top (OTT) of the internet service has given companies in different industries the ability to become direct competitors. Google and Facebook have both implemented applications and services to complete in the traditional telecom markets. Likewise, the Wireless and Internet service markets have become commoditized services making it difficult to compete based on the quality of service/network.

The result has been that the telecommunications companies have competed based on price. It is never a good sign when an industry enters the stage of the product lifecycle where the only way to attract customers is based on price. This has led to the price wars that were observed from 2015 to 2018. Revenues have suffered greatly as a direct result of the price-based competition. In turn, the stock price of most of the telecommunications companies has remained stagnant during this period.

I have been working in the Telecommunications industry for the last thirteen years. I have a strong desire to help the Telecommunications companies thrive and improve their profitable - it is a matter of saving my job.

Problem Statement

While the Telecommunications companies are increasingly entering new markets to escape the declining wireline and wireless industries, they are also placing a laser focus on growing the revenues from their existing customer bases. They are placing a great emphasis on reducing the number of customers that are voluntarily terminating their contracts. The industry calls this “customer churn” – and the churn rate is defined as the percentage of the customer based that terminated their contract in the last month. The Telecommunications industry has a churn rate of ~2% (1).

The problem to be solved is to determine models that can be used to predict whether a customer is a candidate to terminate their contract. The goal will be to reduce the churn rate of the standard Telecommunications Company by being able to identify customers that are likely to churn. Then, the Telecommunications Company can target them with offers that will transform them into customers that are not likely to terminate their contract.

The problem will be set up as a classification problem. The labeled input data contains numerical and categorical data about the customer. Churn is the output (target) feature, which is a binary value of "Churn" or "Not Churn".

Metrics

The model will be evaluated based on the model's AUC or area under the curve associated with ROC curves. The AUC is a common evaluation metric for binary classification problems. Consider a plot of the true positive rate vs the false positive rate as the threshold value for classifying an item as 0 or is increased from 0 to 1: if the classifier is very good, the true positive rate will increase quickly and the area under the curve will be close to 1. If the classifier is no better than random guessing, the true positive rate will increase linearly with the false positive rate and the area under the curve will be around 0.5. One characteristic of the AUC is that it is independent of the fraction of the test population which is class 0 or class 1: this makes the AUC useful for evaluating the performance of classifiers on unbalanced data sets.

The AUC score is recognized as the most predominantly used metric when analyzing an imbalanced dataset. While the accuracy score (and f-score) can be heavily influenced to the positive by an imbalanced dataset, the AUC score is not. Since this project is using an imbalanced dataset, the AUC score has been selected as the benchmark metric. While the dataset for this project is modified to be balanced using the SMOTE function, AUC is still a valid selection as the benchmark metric due to its robust resistance to bias. Likewise, AUC is a very popular metric used to measure success in classification projects using balanced datasets - many of the classification competitions on Kaggle use AUC as the benchmark metric.

II. Analysis

Data Exploration

The dataset called "telco-customer-churn" will be used for this project. (https://www.kaggle.com/blastchar/telco-customer-churn)

This dataset is from a telecommunications company and has been expressly created to support Churn analysis. Therefore, I do believe it is relevant to this project. Since it includes labeled variables as well as the target variable "churn", I believe it is a good candidate for supervised machine learning algorithms.

The dataset contains 7,043 rows (customers) with 21 labeled columns. The input data contains several continuous numerical features and several binary and multi-value categorical features about the customer. The labeled Churn feature will be the output (target), which is a binary value of "Yes" or "No".

Seventy-six percent of the customers in the base dataset are "No", while the remaining twenty-four percent of the customers are "Yes".

The data contains the following variables:

Input variables

1. customerID - Customer ID
2. gender - Customer gender (Categorical: female, male)
3. SeniorCitizen - Whether the customer is a senior citizen or not (Categorical: 1, 0)
4. Partner - Whether the customer has a partner or not (Categorical: Yes, No)
5. Dependents - Whether the customer has dependents or not (Categorical: Yes, No)
6. Tenure - Number of months the customer has stayed with the company (Numeric)
7. PhoneService - Whether the customer has a phone service or not (Categorical: Yes, No)
8. MultipleLines - Whether the customer has multiple lines or not (Categorical: Yes, No, No phone service)
9. InternetService - Customer's internet service provider (Categorical: DSL, Fiber optic, No)
10. OnlineSecurity - Whether the customer has online security or not (Categorical: Yes, No, No internet service)
11. OnlineBackup - Whether the customer has online backup or not (Categorical: Yes, No, No internet service)
12. DeviceProtection - Whether the customer has device protection or not (Categorical: Yes, No, No internet service)
13. TechSupport - Whether the customer has tech support or not (Categorical: Yes, No, No internet service)
14. StreamingTV - Whether the customer has streaming TV or not (Categorical: Yes, No, No internet service)
15. StreamingMovies - Whether the customer has streaming movies or not (Categorical: Yes, No, No internet service)
16. Contract - The contract term of the customer (Categorical: Month-to-month, One year, Two year)
17. PaperlessBilling - Whether the customer has paperless billing or not (Categorical: Yes, No)
18. PaymentMethod - The customer's payment method (Categorical: Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))
19. MonthlyCharges - The amount charged to the customer monthly (Numeric)
20. TotalCharges - The total amount charged to the customer (Numeric)
21. Churn - Whether the customer churned or not (Categorical: Yes or No)

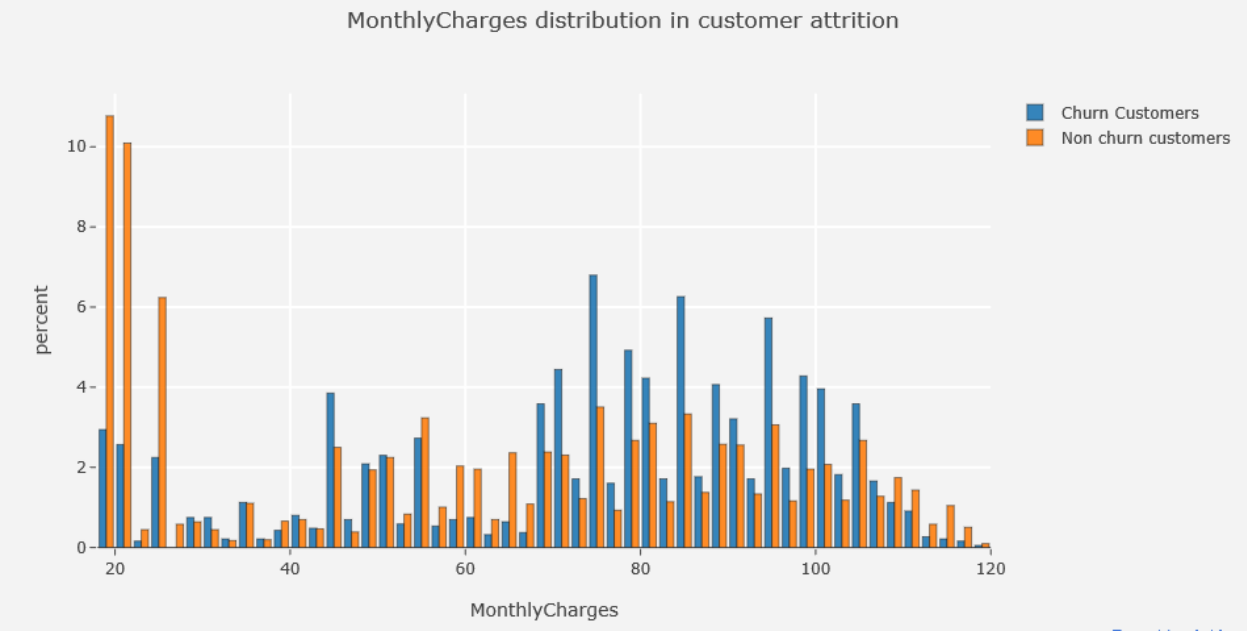
Predict variable (desired target):

Churn - Whether the customer churned or not (Categorical: Yes = 1 or No = 0)

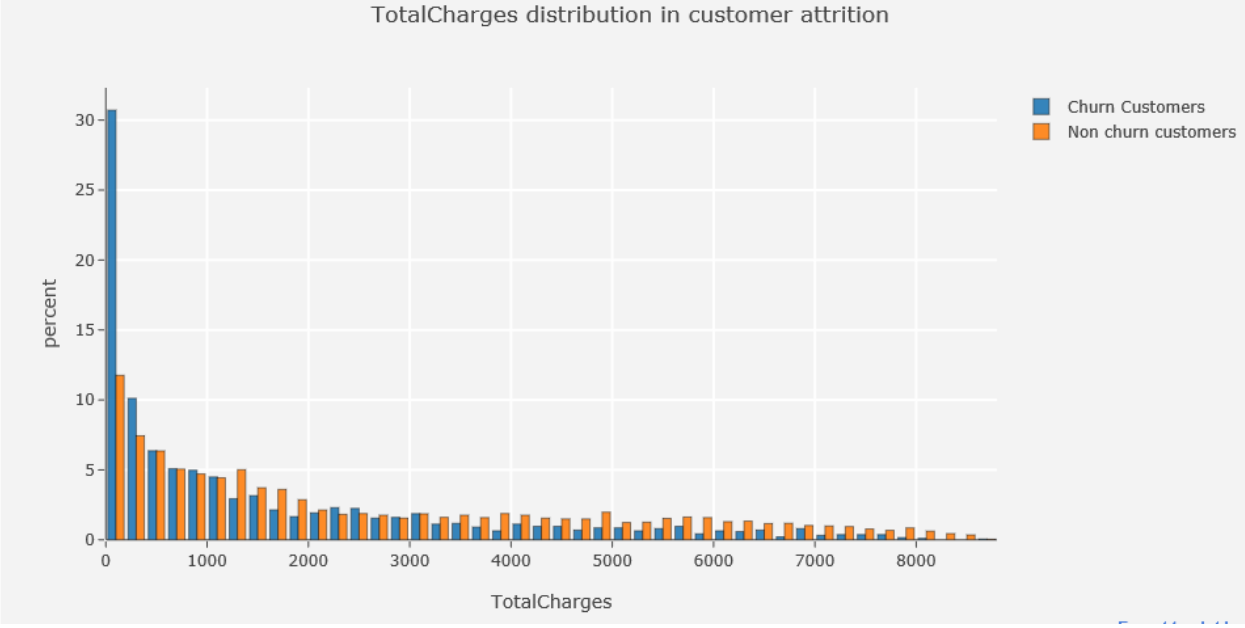
Exploratory Visualization

The following five charts show features that may be useful in predicting whether or not a customer will churn.

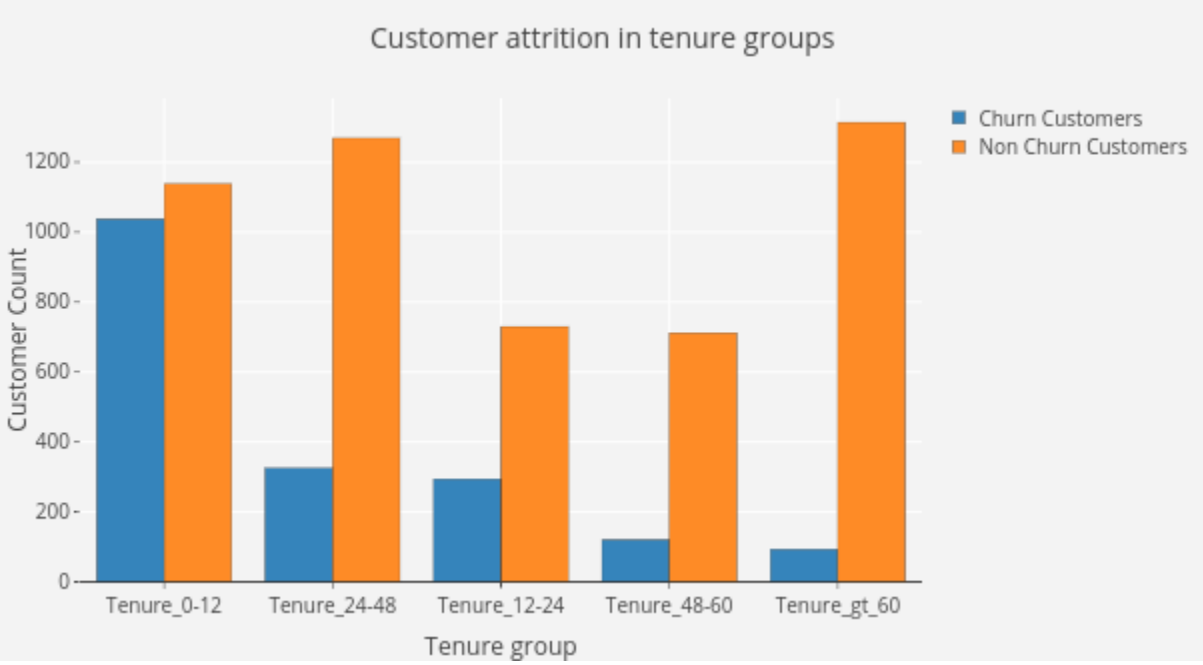
1. Monthly Charges distribution in customer attrition - The churn customer has higher than average Monthly Charges. It is clear that the customers who pay more than $65 have a higher rate of churn than the customers with monthly bills less than $65.
    1. x axis – Dollar amount of Monthly Charges
    2. y axis – Percent of total customers within each churn and non churn customer groups
    3. Labels – Blue – Churn customers; Orange – non churn customers



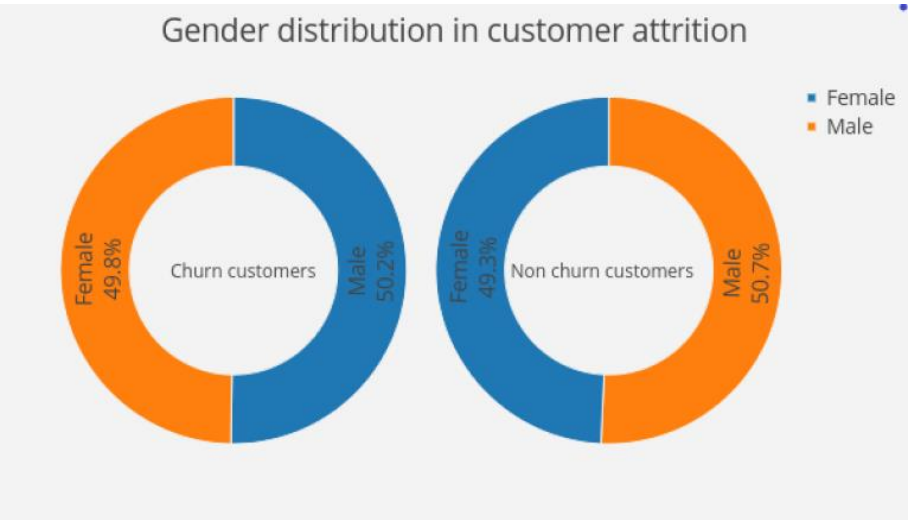
2. Total Charges distribution in customer attrition – A high percentage of churn customers have less than $1,000 in total charges.
    1. x axis – Dollar amount of Total Charges
    2. y axis – Percent of total customers within each churn and non churn customer groups
    3. Labels – Blue – Churn customers; Orange – non churn customers



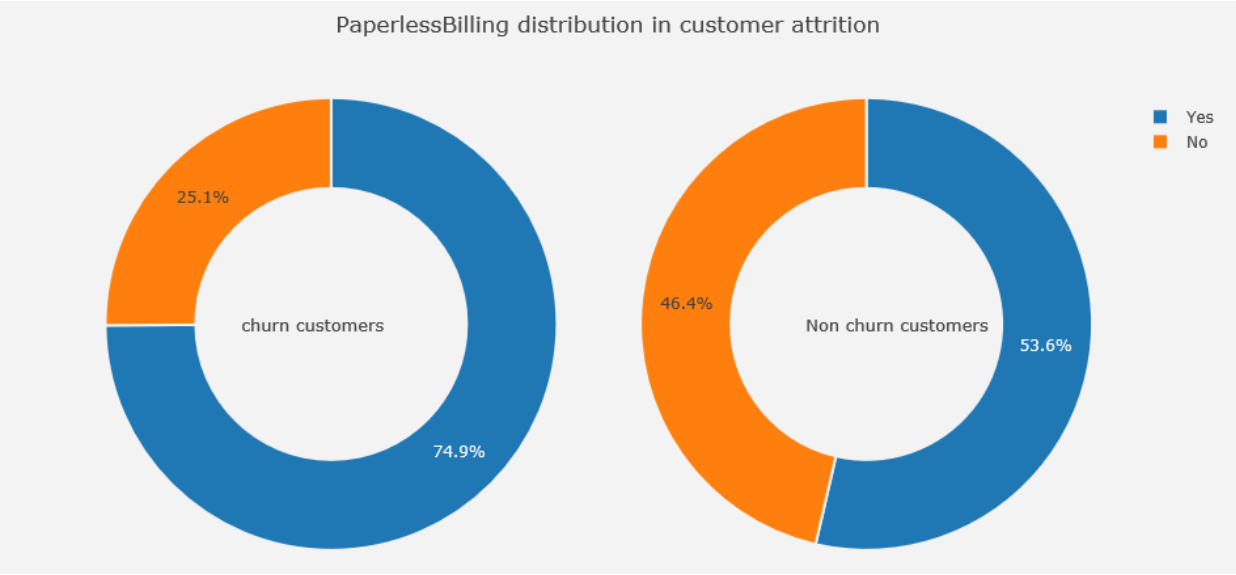
3. Customer attrition in tenure groups - The median tenure of the churn customer is 10 months. This graph shows that most customers that churn leave before 12 month. This helps make sense of the fact that the majority of the churn customers have less than $1,000 in total charges.
    1. x axis – Tenure group in months (ie. Tenure_0-12 = 0 to 12 months)
    2. y axis – Customer count
    3. Labels – Blue – Churn customers; Orange – non churn customers



4. Gender distribution in customer attrition – It is interesting that churn customers are split evenly by gender – 49.8% Female and 50.2% Male.
    1. Left diagram - Churn customers. Right diagram – Non churn customers
    2. Bars – Percent of males and female that churn or do not churn.
    3. Labels – Blue – Female customers; Orange – Male customers



5. Paperless Billing distribution in customer attrition - Customers who have paperless billing are 21% more likely to churn than paper based billing customers.
    1. Left diagram - Churn customers. Right diagram – Non churn customers
    2. Bars – Percent of males and female that churn or do not churn.
    3. Labels – Blue – Female customers; Orange – Male customers



Algorithms and Techniques

The Churn problem is set up as a classification problem. The labeled input data contains numerical and categorical data about the customer. Churn is the output (target) feature, which is a binary value of "Churn" or "Not Churn".

Classification is concerned with building models that separate data into distinct classes. These models are built by inputting a set of training data for which the classes are pre-labelled in order for the algorithm to learn from. The model is then used by inputting a different dataset for which the classes are withheld, allowing the model to predict their class membership based on what it has learned from the training set. Well-known classification schemes include logistic regression and support vector machines. As this type of algorithm requires explicit class labelling, classification is a form of supervised learning.

Algorithms to be used:

All of the following classifiers were selected because they perform well with classification problems using labeled data. The default parameters will be used for each of the following classifiers. I will select one classifier to tune adjusting the parameters accordingly.

1. Logistic Regression – Benchmark model –
    1. Logistic regression, which is borrowed from the field of classical statistics, is one of the simpler machine learning algorithms. This machine learning technique is commonly used for binary classification problems, meaning those in which there are two possible outcomes that are influenced by one or more explanatory variables. The algorithm estimates the probability of an outcome given a set of observed variables.
    2. Where logistic regression differs from other methods is in its interpretability. Since this algorithm is derived from the highly interpretable linear regression algorithm, the influence of each data feature can be interpreted without much effort. As a result, logistic regression is often favored when interpretability and inference is paramount. This versatile algorithm is used to determine the outcome of binary events such as customer churn, marketing click-throughs, or fraud detection.
2. Logistic Regression – SMOTE
    1. The Logistic Regression model will be trained using the balance training dataset created by SMOTE (Synthetic Minority Oversampling Technique)
3. Gaussian Naive Bayes
    1. Bayesian methods are those that explicitly apply the Bayesian inference theorem and facilitate conditional probability in modelling. Bayes' Theorem include a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data. It finds the probability of an event occurring given the probability of another event that has already occurred.
    2. The Bayesian method based algorithms afford fast, highly scalable model building and scoring. These solve both classification and regression problems and scale linearly with the number of predictors and rows.
4. Support Vector Machine – Linear

1. Support Vector Machines (SVMs) are a particular classification strategy. SMVs work by transforming the training dataset into a higher dimension, which is then inspected for the optimal separation boundary, or boundaries, between classes. In SVMs, these boundaries are referred to as hyperplanes, which are identified by locating support vectors, or the instances that most essentially define classes, and their margins, which are the lines parallel to the hyperplane defined by the shortest distance between a hyperplane and its support vectors. Consequently, SVMs are able to classify both linear and nonlinear data.

5. Support Vector Machine – RBF
    1. The RBF kernel maps samples into a higher dimensional space so it, unlike the linear kernel, can handle the case when the relation between class lables and attributes is nonlinear.
    2. The RBF kernel SVM decision region is actually also a linear decision region. What RBF kernel SVM actually does is to create non-linear combinations of your features to uplift your samples onto a higher-dimensional feature space where you can use a linear decision boundary to separate your classes.
6. Background information for XGBoost and Light GBM classifiers:
    4. Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a weighted vote of their predictions. The original ensemble method is Bayesian averaging, but more recent algorithms include error-correcting output coding, bagging, and boosting.
        - So how do ensemble methods work and why are they superior to individual models?
            - They average out biases: If you average a bunch of democratic-leaning polls and republican-leaning polls together, you will get an average something that isn't leaning either way.
            - They reduce the variance: The aggregate opinion of a bunch of models is less noisy than the single opinion of one of the models. In finance, this is called diversification—a mixed portfolio of many stocks will be much less variable than just one of the stocks alone. This is why your models will be better with more data points rather than fewer.
            - They are unlikely to over-fit: If you have individual models that didn't over-fit, and you are combining the predictions from each model in a simple way (average, weighted average, logistic regression), then there's no room for over-fitting.
    2. Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.
    3. Decision trees are a method of splitting the data based on features to either classify or predict some value. Each branch in a decision tree divides the data into one of two (or several, if the tree is not binary) groups. Each leaf node is allocated with a single label

(class or predicted value). When predicting using the decision tree, the data is allocated to the appropriate leaf node, and the prediction is the label of that leaf node.

6. XGBoost Classifier
    1. Extreme Gradient Boosting (XGBoost) is an implementation of the gradient boosting machines that is highly flexible and versatile while being scalable and fast. XGBoost works with most regression, classification, and ranking problems as well as other objective functions; the framework also gained its popularity in recent years because of its compatibility with most platforms and distributed solutions like Amazon AWS, Apache Hadoop, Spark among others.
    2. In short, XGBoost is a variation of boosting - an ensemble method algorithm that tries to fit the data by using a number of "weak" models, typically decision trees. The idea is that a "weak" classifier which only performs slightly better than random guessing can be improved ("boosted") into a "stronger" one that is arbitrarily more accurate (source: Y. Freund, R. E. Schapire)(source: R. E. Schapire). Building on the weak learners sequentially, at every round each learner aims to reduce the bias of the whole ensemble of learners, thus the weaker learners eventually combined into a powerful model. This idea gave birth to various boosting algorithms such as AdaBoost, Gradient Tree Boosting, etc.
    3. XGBoost is an example of gradient boosting model, which is built in stages just like any other boosting method. In gradient boosting, weak learners are generalized by optimizing an arbitrary loss function using its gradient.
    4. XGBoost, as a variation of boosting, features a novel tree learning algorithm for handling sparse data; a theoretically justified weighted quantile sketch procedure enables handling instance weights in approximate tree learning. source: T. Chen, C. Guestrin
    5. There is a number of advantages in using XGBoost over other classification methods: Work with large data: XGBoost packs many advantageous features to facilitate working with data of enormous size that typically can't fit into the system's memory such as distributed or cloud computing. It is also implemented with automatic handling of missing data (sparse) and allows continuation of training, or batch training.
7. LightGBM Classifier
    1. Though XGBoost seemed to be the go-to algorithm in Kaggle for a while, a new contender is quickly gaining traction: lightGBM. Released from Microsoft, this algorithm has been claimed to be more efficient (better predictive performance for the same running time) than xgboost.
    2. Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks.
    3. LightGBM speeds up the training process of conventional GBDT by up to over 20 times while achieving almost the same accuracy.
    4. Two techniques used to make LightGBM faster than XGBoost:
        - {Gradient-based One-Side Sampling} (GOSS), we exclude a significant proportion of data instances with small gradients, and only use the rest to estimate the information gain. We prove that, since the data instances with larger gradients play a more important role in the computation of information gain, GOSS can obtain quite accurate estimation of the information gain with a much smaller data size.

- {Exclusive Feature Bundling} (EFB) - bundle mutually exclusive features (i.e., they rarely take nonzero values simultaneously), to reduce the number of features.

Benchmark

Applying Machine Learning techniques to customer data to identify contributors to customer churn is not a new phenomenon. It seems to be an industry standard to use the Logistical Regression model as the benchmark model (2). The Logistical Regression model will provide a benchmark f-score, precision and recall scores, as well as the ROC or area under the curve to be used for model comparison. Likewise, it will provide a benchmark set of features that contribute to the prediction of the target variable, as well as their respective weights.

III. Methodology

Data Preprocessing

The dataset was preprocessed using the following techniques:

1. Deleted the 11 records that had empty values in the “total charges” field
2. Converted the “total charges” field to a float type
3. Replaced the value 'No internet service' to ‘No’ for the following columns: 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection','TechSupport','StreamingTV', 'StreamingMovies'
4. Label encoded the binary columns to “0” for “No” and “1” for “Yes” using LabelEncoder() from sklearn.
5. Created duplicate columns for each value in multi-value columns using get_dummies.
6. Scaled numeric columns to prevent bias resulting from skewed data using StandardScaler().

As a result of the preprocessing the number of columns increased from 21 to 30.

```
Features :
 ['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'OnlineSecurity', 'OnlineBackup', 'DeviceP
rotection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling', 'Churn', 'MultipleLines_No', 'MultipleLines_N
o phone service', 'MultipleLines_Yes', 'InternetService_DSL', 'InternetService_Fiber optic', 'InternetService_No', 'Contract_M
onth-to-month', 'Contract_One year', 'Contract_Two year', 'PaymentMethod_Bank transfer (automatic)', 'PaymentMethod_Credit car
d (automatic)', 'PaymentMethod_Electronic check', 'PaymentMethod_Mailed check', 'tenure', 'MonthlyCharges', 'TotalCharges']
```

Data Abnormality - Imbalanced Data

Most classification algorithms will only perform optimally when the number of samples of each class is roughly the same. Highly skewed datasets, where the minority is heavily outnumbered by one or more classes, have proven to be a challenge while at the same time becoming more and more common.

Since seventy-six percent of the records in the dataset are for customers that do not churn – and only twenty-four percent of the records contain the target value of customer's who churn, the dataset is imbalanced. The target value is a minority case in the existing dataset. Running models against an imbalanced dataset do not produce the best prediction results. As a result, a technique called SMOTE will be used to create a balance between the records that contain the churn and non-churn values. SMOTE will address the imbalance issue by re-sampling the dataset as to offset this imbalance with the hope of arriving at a more robust and fair decision boundary.

Implementation

Software Requirements:

Python >= 3.6
numpy >= 1.15.1
pandas >= 0.23.4
scikit-learn >= 0.20
matplotlib >= 2.2.2
Seaborn >= 0.9.0
Plotly >= 0.4.0
Yellowbrick >= 0.8
imbalanced-learn >= 0.4.1
lightgbm >= 2.2.1
py-xgboost >= 0.72

The following process was used to implement the algorithms:

1. The dataset was shuffled and split into training and testing sets using train_test_split from sklearn. Seventy-five percent of the data will be used for training with the remaining twenty-five percent to be used for testing.
2. The target column for the train and test data set is the "Churn" column.
3. A modeling pipeline function called "telecom_churn_prediction" is used to train, predict, measure, and report on each algorithm.
4. The following information is gathered and displayed in chart format for each algorithm:
    a. Precision
    b. Recall
    c. F1-score
    d. Support
    e. Accuracy score
    f. Area under the curve (This is the metric that will be used to determine success)
5. The following information is displayed in visual graph format for each algorithm:

    a. Confusion matrix
    b. Receiver operating characteristic (Area under the curve)
    c. Feature importance
6. The Logistic Regression baseline model is the first model run resulting in a 72% Area under the curve (AUC) score. This will be used as the baseline to compare and measure the remaining algorithms.
7. Addressed the imbalanced nature of the dataset using SMOTE up-sampling algorithm (Synthetic Minority Oversampling Technique) on the training dataset. At a high level, SMOTE:
    a. Works by creating synthetic samples from the minor class (churn) instead of creating copies
    b. Randomly choosing one of the k-nearest neighbors and using it to create a similar, but randomly tweaked, new observation.
    c. Statistics:
        i. Original dataset: 26% Churn, 74% not Churn
        ii. Training dataset: 20% Churn, 80% not Churn
        iii. SMOTE training dataset: 50% Churn, 50% not Churn
8. Reran the Logistic Regression model using the default parameters and the SMOTE training dataset. The AUC increased from 72% for the baseline to 77.5%
9. All of the subsequent models were run using default parameters and the SMOTE training dataset
10. The following table contains the results for each algorithm:

Model	Accuracy_score	Recall_score	Precision	f1_score	Area_under_curve	Kappa_metric
Log Regr (Baseline)	0.8009	0.5388	0.6804	0.6014	0.7205	0.4711
Log Regr (SMOTE)	0.7651	0.798	0.5546	0.6544	0.7752	0.485
Naive Bayes	0.7531	0.7714	0.54	0.6353	0.7587	0.4574
SVM Classifier Linear	0.7537	0.8061	0.5389	0.646	0.7698	0.4683
SVM Classifier RBF	0.8805	0.8367	0.7593	0.7961	0.8671	0.7119
LGBM Classifier	0.8532	0.7143	0.7479	0.7307	0.8106	0.6299
XGBoost Classifier	0.8066	0.7082	0.6379	0.6712	0.7764	0.5347

Export to plot.ly

During the coding of this project, I had difficulty using several packages and models that I had never used before: Seaborn, Yellowbrick, imbalanced-learn, lightgbm, and py-xgboost. While I was able to find code examples, I had to research the details of each package to understand how to implement specific features. I was very fortunate to be able to find lots of good code examples, so I did not have any specific issues that proved to be overly difficult to resolve.

Refinement

While the SVM classifier using the RBF kernel and the LGBM classifier have higher initial AUC scores, I am going to tune the XGBoost classifier. This classifier has been selected for tuning, because it is currently receiving such high praise from the Machine Learning community. The majority of the Kaggle competitions in the last year have been won using this model.

The GridSearchCV function from sklearn will be used to tune the parameters of the XGBoost classifier. While there are over a dozen parameters that can be tuned, I selected max_depth and the learning_rate parameters to tune because they have the highest potential of all of the parameters to improve the AUC metric.

Max_depth is used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.

Learning_rate - Gradient boosting involves creating and adding trees to the model sequentially. New trees are created to correct the residual errors in the predictions from the existing sequence of trees. The effect is that the model can quickly fit, then overfit the training dataset. A technique to slow down the learning in the gradient boosting model is to apply a weighting factor for the corrections by new trees when added to the model. This weighting is called the shrinkage factor or the learning rate, depending on the literature or the tool. Naive gradient boosting is the same as gradient boosting with shrinkage where the shrinkage factor is set to 1.0. Setting values less than 1.0 has the effect of making less corrections for each tree added to the model. This in turn results in more trees that must be added to the model. It is common to have small values in the range of 0.1 to 0.3, as well as values less than 0.1. The learning rate controls how fast the model tries to converge.

The following values were used for each of these parameters:

- 'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- 'learning_rate': [.1, .2, .3, .4, .5, .6, .7, .8, .9, 1

GridSearchCV will run all permutations of these values to find the combination that provides the best results for the XGBoost classifier. While the default model uses 3 for max_depth and .1 for learning_rate, the GridSearchCV function determined that the optimal values are 10 for max_depth and .1 for learning_rate. Using these values resulted in an increase in AUC from 77.64% to 88.96%. The following chart shows that the tuned XGBoost classifier has the highest AUC of all of the tested algorithms.

Model	Accuracy_score	Recall_score	Precision	f1_score	Area_under_curve	Kappa_metric
Log Regr (Baseline)	0.8009	0.5388	0.6804	0.6014	0.7205	0.4711
Log Regr (SMOTE)	0.7651	0.798	0.5546	0.6544	0.7752	0.485
Naive Bayes	0.7531	0.7714	0.54	0.6353	0.7587	0.4574
SVM Classifier Linear	0.7537	0.8061	0.5389	0.646	0.7698	0.4683
SVM Classifier RBF	0.8805	0.8367	0.7593	0.7961	0.8671	0.7119
LGBM Classifier	0.8532	0.7143	0.7479	0.7307	0.8106	0.6299
XGBoost Classifier	0.8066	0.7082	0.6379	0.6712	0.7764	0.5347
Tuned XGBoost	0.9084	0.8469	0.8283	0.8375	0.8896	0.7738

Export to plot.ly »

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

Sensitivity analysis has been performed using cross-validation. Cross-validation, sometimes called rotation estimation, is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

The k-fold cross-validation technique is used to make sure that model parameters are not "overfit" to the test data split. When using just one set of training data, the model can adapt and tune itself to a high degree, thus "overfitting". This will result in a poor score when applying the model to the test data. In order to make sure that the parameters are generalized and not "overfit", the training data split can be split again into multiple training sets. For example, with cv of cross_val_score set to 4, the training data set is broken into 4 equal segments. One of the segments is kept as the validation data and the remaining 3 segments are used as learning data. The model is run three times using each learning segments to model and then predict the 1 segment of validation data. The average of the three prediction scores is used as the efficacy score of the model.

While the XGBoost algorithm uses 3 folds by default (Tuned XGBoost), I increased the number of folds to 5 and received the following results (XGBoost CV test).

Results of the XGBoost model run using the 3 default k-fold for cross validation:

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits

[Parallel(n_jobs=16)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=16)]: Done  18 tasks      | elapsed:    9.2s
[Parallel(n_jobs=16)]: Done 168 tasks      | elapsed:   45.2s
[Parallel(n_jobs=16)]: Done 300 out of 300 | elapsed:  1.3min finished

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=10, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)

 Classification report :
              precision    recall  f1-score   support

           0       0.94      0.94      0.94      1268
           1       0.84      0.84      0.84       490

   micro avg       0.91      0.91      0.91      1758
   macro avg       0.89      0.89      0.89      1758
weighted avg       0.91      0.91      0.91      1758

Accuracy   Score :  0.9084186575654153
Area under curve :  0.8864272838472929
```

Results of the XGBoost model run using the 3 default k-fold for cross validation:

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
ble click to hide
(n_jobs=16)]: Using backend LokyBackend with 16 concurrent workers.
(n_jobs=16)]: Done  18 tasks      | elapsed:    8.5s
[Parallel(n_jobs=16)]: Done 168 tasks      | elapsed:   51.7s
[Parallel(n_jobs=16)]: Done 418 tasks      | elapsed:  2.0min
[Parallel(n_jobs=16)]: Done 500 out of 500 | elapsed:  2.4min finished

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=10, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)

 Classification report :
               precision    recall  f1-score   support

           0       0.94      0.94      0.94      1268
           1       0.84      0.84      0.84       490

   micro avg       0.91      0.91      0.91      1758
   macro avg       0.89      0.89      0.89      1758
weighted avg       0.91      0.91      0.91      1758

Accuracy   Score :  0.9084186575654153
Area under curve :  0.8864272838472929
```

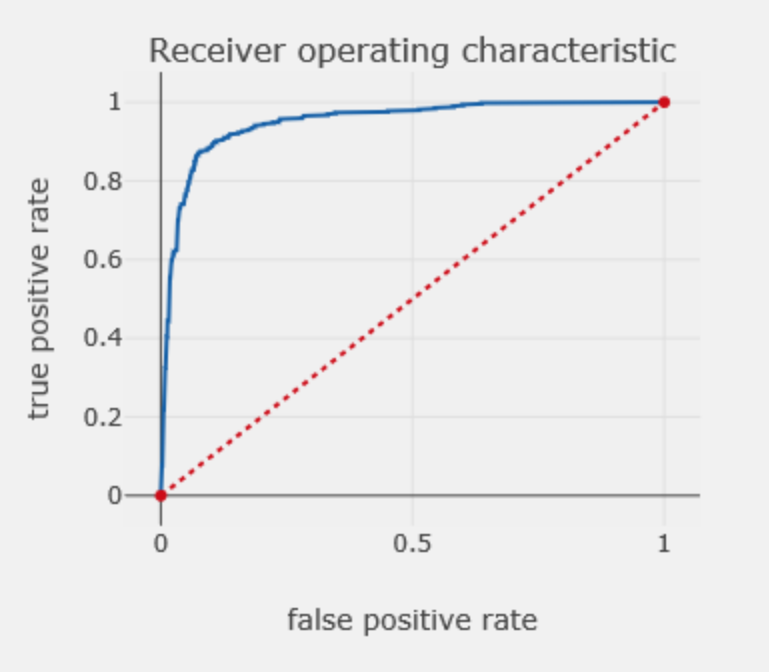
Since the AUC score is exactly the same for the same for the model when run using 3 (Tuned XGBoost) and 5 folds (XGBoost CV test), it is a strong indicator that the model is very robust. The scores below are the average scores across the three and five folds.

Model	Accuracy_score	Recall_score	Precision	f1_score	Area_under_curve	Kappa_metric
XGBoost Classifier	0.8083	0.698	0.6441	0.6699	0.7745	0.5352
Tuned XGBoost	0.9084	0.8367	0.835	0.8359	0.8864	0.7724
XGBoost CV test	0.9084	0.8367	0.835	0.8359	0.8864	0.7724

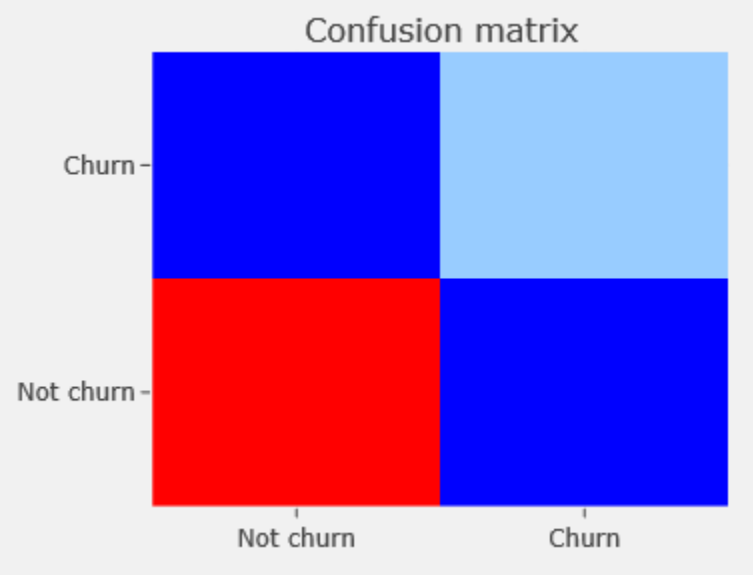
Export to plot.

Justification

The baseline Logistical Regression model has a 76% percent Area Under the Curve score. After reviewing the Area Under the Curve (AUC) data in the Model Comparison Chart above, the XGBoost classifier is the winner with a 88.64% AUC score. This model also has a very high accuracy score of 90.84%. As a result, the XGBoost model is the new benchmark. The following “Receiver operating characteristic” diagram displays the Area Under the Curve.



This diagram shows that the precision and recall of the model are very close to the ultimate goal of one.



The confusion matrix heat map is showing that there are 1187+410 correct predictions and 81+80 incorrect predictions. This is a 90% accuracy score.

While there is definitely room for improvement, the final model is significant enough to have solved the problem.

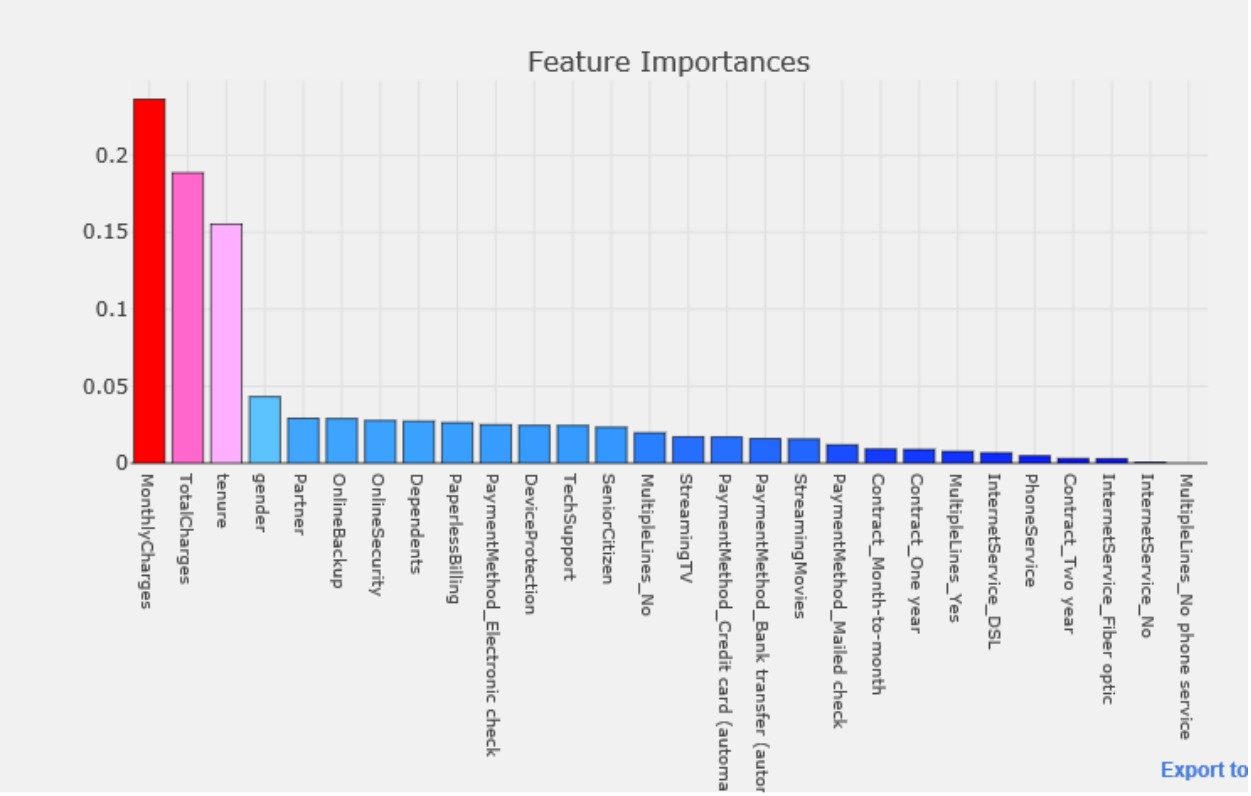
V. Conclusion

Free-Form Visualization

Feature Importance - Business Implications

The final tuned XGBoost model lists the following as the most important features for predicting churn.

1. Monthly Charges - 24%
2. Total Charges - 19%
3. Tenure - 16%
4. Gender - 4%
5. Partner - 3%



The following observation can be drawn from the charts in the "Data Analysis - Visuals" section above:

1. Monthly Charges - The churn customer has higher than average Monthly Charges.

2. Total Charges - The Total Charges of the churn customer is on average higher than the average customer at that tenure.

3. Tenure - The median tenure of the churn customer is 10 months.

4. Gender - It is not clear from the Gender chart below how this feature influences the XGBoost model. However, the females churn 1% more than the males. This feature must influence the model when it is in a specific combination with the other features.

A particularly important fact is that 59% of the model's prediction of whether or not a customer will churn is based on just three features: Monthly Charges, Total Charges, and Tenure. It is clear that the combination of monthly charges, total charges, tenure, and gender are very important features to determine the propensity of a customer to churn. For example, a female customer who has paperless billing, high monthly charges, has been with the company less than 10 months, and as a result has higher than average total charges for their tenure is strong candidate to leave the company. This information can be used to develop business strategies to target these specific customers. Female customers could be offered 10% discounts on their monthly charges for the first year. This would give them incentive to stay with the company past the crucial 10 month window within which the churn rate is high.

It is interesting that the decision to churn appears to be more influenced by the cost of the services as opposed to the services or features themselves. For example, if all of the churn customers have DSL, this would indicate that this particular service is unsatisfactory and customers are leaving as a result. It appears quality of service is not as important as price. This again re-enforces the notion that the services provided by telecommunications companies have become commodities. Unfortunately, price is the only competitive lever to use when the product you are selling is a commodity. It is not a surprise that most telecommunications companies are searching to expand into vertical and horizontal industries to combat this reality. I believe the model created by this project accurately reflects the reality within the telecommunications industry.

Reflection

Since the dataset came from an existing Kaggle competition, I found it very interesting to review the kernels that had been created previously. While I learned a lot about Python coding and the Classification algorithms, I also learned a lot about the use of different visualization packages. I found it most interesting to be introduced to the popular decision tree boosting algorithms XGBoost and LightGBM that I had not experimented with in the previous projects. Having the guidance of so many good kernels made the process of implementing this project much easier.

However, I did not find any kernels that performed parameter tuning or data sensitivity analysis. As I will discuss in the following section, the parameter tuning was the hardest part of the project for me.

I do not feel that the final model can be applied to the gain insight into the overall Telecommunication Churn issue. The dataset was way too small to assume that it could successfully predict churn. However, I do think the final model could be used as a beginning benchmark when applied to a large dataset.

Improvement

I have taken a very minimal and unsophisticated approach to tuning the XGBoost model. The articles "Complete Guide to Parameter Tuning in XGBoost (with codes in Python)" and "Tune Learning Rate for Gradient Boosting with XGBoost in Python" provide excellent examples of rigorous tuning methodologies. I know the model's predictive results can be improved if I had followed a very detailed approach to the tuning process.

I also would like to get the opportunity to perform analysis on a dataset that includes not only the services and features, but also how satisfied the customer is with each of the services and features. This information could be used to further determine if there is any correlation with a particular service and the churn rate. However, I think that keeping behavioral information out of a dataset like this is important. This keeps the analysis based on explicit facts as opposed to subjective feelings that may not be relative between customers.

While the models were selected because they lend themselves perfectly to "churn" classification, I would love to run deep learning models on the data. I would be interesting to see if they would be able to uncover any hidden correlations between the specific services and features and "churn". Likewise, while the final XGBoost is the new benchmark, I believe an even better solution exists (maybe even simply a better tuned XGBoost classifier).

References

1. **"**Facebook is building a camera TV set-top box**."** TechCrunch, 17 October 2018, https://techcrunch.com/2018/10/16/facebook-ripley-set-top-box/. Accessed 17 October 2018.
2. "Google: Your new phone carrier." CNN Money, 1 January 2010. https://money.cnn.com/2010/12/30/technology/google_wireless_carrier/index.htm**.** Accessed 14 October 2018.
3. **"**Churn reduction in the telecom industry**."** Database Marketing Institute, 15 October 2018, http://www.dbmarketing.com/telecom/churnreduction.html. Accessed 12 October 2018.
4. "Managing Churn to Maximize Profits." Harvard Business Review, 15 October 2018, https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=2ahUKEwi_4Oi1iYneAhUHI6wKHSbbCgoQFjADegQIBBAC&url=https%3A%2F%2Fwww.hbs.edu%2Ffaculty%2FPublication%2520Files%2F14-020_3553a2f4-8c7b-44e6-9711-f75dd56f624e.pdf&usg=AOvVaw37Ykq-D-WIkcESKB5xoaYy . Accessed 12 October 2018.
5. AUC explanation - https://stats.stackexchange.com/questions/132777/what-does-auc-stand-for-and-what-is-it
6. **"**CHURN IN THE TELECOM INDUSTRY – IDENTIFYING CUSTOMERS LIKELY TO CHURN AND HOW TO RETAIN THEM**."** Database Marketing Institute, 17 February 2017, https://wp.nyu.edu/adityakapoor/2017/02/17/churn-in-the-telecom-industry-identifying-customers-likely-to-churn-and-how-to-retain-them/. Accessed 13 October 2018.

7. "Reducing churn in telecom through advanced analytics." McKinsey and Co., https://www.mckinsey.com/industries/telecommunications/our-insights/reducing-churn-in-telecom-through-advanced-analytics. Accessed 12 October 2018.
8. **"**Predict Customer Churn – Logistic Regression, Decision Tree and Random Forest**."** datascience, 20 November 2020, https://datascienceplus.com/predict-customer-churn-logistic-regression-decision-tree-and-random-forest/. Accessed 14 October 2018.
9. **"**Kaggle Dataset." https://www.kaggle.com/blastchar/telco-customer-churn
10. Taiwo Oladipupo Ayodele (February 1st 2010). Types of Machine Learning Algorithms, New Advances in Machine Learning Yagang Zhang, IntechOpen, DOI: 10.5772/9385. Available from: https://www.intechopen.com/books/new-advances-in-machine-learning/types-of-machine-learning-algorithms
11. A Tour of Machine Learning Algorithms. https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/
12. Machine Learning Key Terms, Explained. https://www.kdnuggets.com/2016/05/machine-learning-key-terms-explained.html
13. Machine Learning Key Terms, Explained. https://www.datascience.com/blog/introduction-to-machine-learning-algorithms
14. Machine Learning Algorithms: A Concise Technical Overview – Part 1. https://www.kdnuggets.com/2017/08/machine-learning-algorithms-concise-technical-overview-part-1.html
15. Building A Logistic Regression in Python, Step by Step. https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8
16. 10 groups of Machine Learning Algorithms. http://adataanalyst.com/machine-learning/10-groups-machine-learning-algorithms/
17. 6 Easy Steps to Learn Naive Bayes Algorithmv.https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
18. Support Vector Machines: A Simple Explanation. https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html
19. How to Select Support Vector Machine Kernels. https://www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html
20. The 10 Algorithms Machine Learning Engineers Need to Know. https://www.kdnuggets.com/2016/08/10-algorithms-machine-learning-engineers.html/2
21. LightGBM. https://lightgbm.readthedocs.io/en/latest/Features.html
22. Which algorithm takes the crown: Light GBM vs XGBOOST?. https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/
23. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. https://www.microsoft.com/en-us/research/publication/lightgbm-a-highly-efficient-gradient-boosting-decision-tree/
24. XGBoost Documentation. https://xgboost.readthedocs.io/en/latest/
25. Using XGBoost in Python. https://www.datacamp.com/community/tutorials/xgboost-in-python
26. Machine Learning Engineer Nanodegree Capstone Project by Nguyen Quoc Bao https://github.com/bqnguyen94/Udacity-MLND-Capstone/blob/master/report.pdf
27. Boosting (machine learning). https://en.wikipedia.org/wiki/Boosting_(machine_learning)

28. Starter Guide : Preprocessing & RandomForest. https://www.kaggle.com/tanetboss/starter-guide-preprocessing-randomforest#
29. imbalanced-learn.org. https://github.com/scikit-learn-contrib/imbalanced-learn
30. Kaggle Kernel. https://www.kaggleusercontent.com/kf/5566622/eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..AxMOalhBBcGJtnnTJ4ZomQ.yCIv3RHlWldTvc8QR3ssSSKIcBxNs_smNiIvmeRLfjKgbzUe1kk1RCgGK9FV1EacAvdm7YQvTBI3XLwzurMKAcGaUYTb4Wb-VqNzCGgGjHXBLXn1vpXIf0gAt6tj2nBcSv3iQXonGsNzxMmtnJQOXllheOUxqvRkm927HrHrQSc.WcZGJYDProlgioDhAxXk_w/__results__.html>
31. Kaggle Kernel. https://www.kaggleusercontent.com/kf/5681558/eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..-K13b1W_cWQfTDfpMFjuZA.vE_PPrcKKPpfURhgXnvhAp3tXIVHH7Eel2SLkegWMcznHXn7TCCh3vwj520hXnSkUOClI6663LUW2hB6PV-Q9NN8it_o8wD59GK2wXsqwCPrsAnko7T63_-XD8yVss9MJQJoIq2r0PfWQxCl2iwMO_2ZIrljiJj63fYnIxf3QJA.bOg8lp3ZUkNIicmOwJUJHQ/__results__.html
32. Kaggle Kernel. https://www.kaggleusercontent.com/kf/6005067/eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..IfIsUuiEB4f_RsmtoRwqFQ.DrUmelWOgz6__h2IanPi-vEYGY_Sjm7ncGPdF1vccwMrM8Ng8jBs_DUyubnNYfuBg7HgFSarP8IQM5jWP7DYCfwg7_2OpdU1iSlebYSyS8txPgD7yZvm7yu31bxrGvBXfzWSH2Zb6O3X3vDLpaoP2cyhZacPiItGFnQUcfZyd3s.SvpVcLNHYAuWwmzkgNak4A/__results__.html
33. Kaggle Kernel. https://www.kaggle.com/pavanraj159/telecom-customer-churn-prediction
34. Complete Guide to Parameter Tuning in XGBoost (with codes in Python). **https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/**
35. Tune Learning Rate for Gradient Boosting with XGBoost in Python. https://machinelearningmastery.com/tune-learning-rate-for-gradient-boosting-with-xgboost-in-python/
36. What does AUC stand for and what is it?. https://stats.stackexchange.com/questions/132777/what-does-auc-stand-for-and-what-is-it
37. Advantages of AUC vs standard accuracy. https://datascience.stackexchange.com/questions/806/advantages-of-auc-vs-standard-accuracy