Machine Learning Engineer Nanodegree

Capstone Project

Gino Nedilskyj

February 4, 2019

I. Definition

Project Overview

This project aims to present a machine learning solution for targeting telecom churn customers, i.e., customers that are likely to leave the company. As with most companies reliant on customer subscribers to maintain revenue growth, telecoms have been tackling the issue of trying to keep customers in a world where consumers have many other options. This is not an easy problem to solve as there are so many factors that can contribute to a decision to pivot to other companies. For this reason, it is an ideal candidate for a machine learning model. If we can successfully identify high risk customers, a targeted strategic campaign can be implemented by the telecom to reduce the risk, and ultimately minimize the loss of potential revenue associated with these customers leaving.

The idea for this project came from the kaggle.com website. I thought it would be interesting since I am currently in the telecom industry and have been aware of the issue of churn. The data comes from one of the datasets in the Kaggle website [2], originally from IBM Sample Data Sets. The set contains 7043 rows, each pertaining to a customer, with 21 total attributes.

Problem Statement

The problem that we will aim to solve is to create a model that will infer, from the given set of customer attributes, a prediction about whether a customer will leave the company or stay. The attributes of the each customer are either quantifiable (*Monthly Charges, Total Charges*, etc) or categorical (*gender, Senior Citizen*, etc.) in nature. Each customer data point contains a label of Churn/No Churn which should be well suited for a machine learning binary classification

model. We will use 4 classifier models and compare them against each other and a baseline model.

Metrics

The end goal is to provide a telecom company insight into which customers are more likely to leave. The company's strategy would be to efficiently target these customers and only these customers with a targeted retention campaign. Since there is a cost per customer associated with any campaign the company chooses to run, it is in the company's best interest to focus resources and capital on customers that are truly at risk and not waste resources on 'false positive' customers that would have stayed on regardless. However, assuming the campaign is successful in keeping customers on, the revenue saved (or losses averted) by a customer staying on is likely worth the wasted resources on a small number of false positives. This suggests we want our model to have a high recall score so that we cast a wide net and identify most of the potential deserters. Due to this, the metric most apt to measure the how well our model is doing is the F-2 score. Accuracy will also be used as a general metric to measure against an overall score when comparing other models. However, more weight should be given to F-Score since we have an imbalanced data set for which accuracy should not be the sole metric.

II. Analysis

Data Exploration

The data consists of one comma-delimited file of 21 columns and 7043 rows. This was directly downloaded from the Kaggle site as is. The 21 columns are labeled as follows:

Column Headers

customerID	gender	Senior Citizen	Partner	Dependents	tenure	Phone Service
Multiple Lines	Internet Service	Online Security	Online Backup	Device Protection	Tech Support	Streaming TV
Streaming Movies	Contract	Paperless Billing	Payment Method	Monthly Charges	Total Charges	Churn

The data elements consist of 20 features and 1 label, that being Churn (red). The blue fields represent numerical data types (3) while the white fields represent categorical data types (17). *Churn* is a boolean Yes/No data type that indicates whether that customer has left the company.

During the data inspection, it was noted that the 1st column, *CustomerID*, is merely an identifier field that was likely used to identify to individual customers. It was decided this field does not add value or information to the dataset, just as the customer name would not add value. This feature was dropped.

Additional data inspection revealed that there were 11 missing values in the *Total Charges* column. Because the value of *Total Charges* is potentially correlated to the likelihood of churn, imputing these blanks with a statistical value such as mean or median may not be prudent. Therefore, it was decided that all 11 data points be dropped.

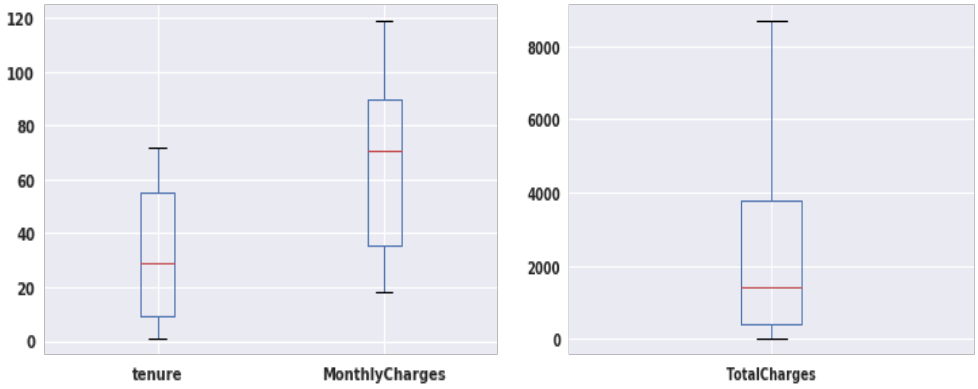
Basic Statistics:

	gender	Senior Citizen	Partner	Dependents	tenure	Monthly Charges	Total Charges
count	7032	7032	7032	7032	7032	7032	7032
mean	0.504693	0.1624	0.482509	0.298493	32.421786	64.798208	2283.300441
std	0.500014	0.368844	0.499729	0.457629	24.54526	30.085974	2266.771362
min	0	0	0	0	1	18.25	18.8
25%	0	0	0	0	9	35.5875	401.45
50%	1	0	0	0	29	70.35	1397.475
75%	1	0	1	1	55	89.8625	3794.7375
max	1	1	1	1	72	118.75	8684.8

	Phone Service	Online Security	Online Backup	Device Protection	Tech Support	Internet Service_ DSL	Internet Service_ Fiber optic
count	7032	7032	7032	7032	7032	7032	7032
mean	0.903299	0.286547	0.344852	0.343857	0.290102	0.343572	0.440273
std	0.295571	0.45218	0.475354	0.475028	0.453842	0.474934	0.496455
min	0	0	0	0	0	0	0
25%	1	0	0	0	0	0	0
50%	1	0	0	0	0	0	0
75%	1	1	1	1	1	1	1
max	1	1	1	1	1	1	1

	Internet Service_ No	Contract_ Month-to-month	Contract_ One year	Contract_ Two year	Payment Method_ Bank transfer (automatic)	Payment Method_ Credit card (automatic)	Payment Method_ Electronic check	Payment Method_ Mailed check
count	7032	7032	7032	7032	7032	7032	7032	7032
mean	0.216155	0.551052	0.209329	0.239619	0.219283	0.216297	0.33632	0.2281
std	0.41165	0.497422	0.406858	0.426881	0.41379	0.411748	0.472483	0.419637
min	0	0	0	0	0	0	0	0
25%	0	0	0	0	0	0	0	0
50%	0	1	0	0	0	0	0	0
75%	0	1	0	0	0	0	1	0
max	1	1	1	1	1	1	1	1

	Churn
count	7032
mean	0.265785
std	0.441782
min	0
25%	0
50%	0
75%	1
max	1



Some basic statistical exploration of the 3 numerical fields (*Tenure, Monthly Charges* and *Total Charges*) yields that none of them show significant outliers in their respective data sets, according to the boxplots.

Exploratory Visualization

An important finding about the label column Churn is that the data is imbalanced (Fig 1). Approximately 27% of the data points show churn, while 73% is not churn. This will have to be taken into account in model selection and hyper-parameter selection, as well how metrics and performance are measured.

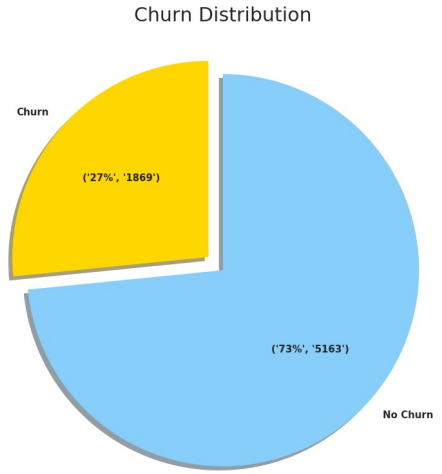


Fig 1

A correlation heatmap (Fig 2) of all the features shows some interesting correlations in the data. Some correlations make a lot of sense. *Tenure* is strongly inversely correlated to *Month-to-Month*, which makes sense since if a customer signing up for month-to-month contract is not tied to the company as would a 1 or 2 year contract customer. Similarly, there is a strong correlation between *Tenure* and *Total Charges*. Interestingly, there are 2 positively correlated features to *Churn*. Both *Internet_servicesFiberOptic* and *ContractMonth-Month* show a positive correlation, which may indicate that these 2 features may be driving factors in churn rate.

This visualization shows that there is correlation between some features, like between *TotalCharges*, *Tenure*, *Monthly Charges*. Given this result, it would be prudent to remove some of the dimensionality of the feature set with a PCA.

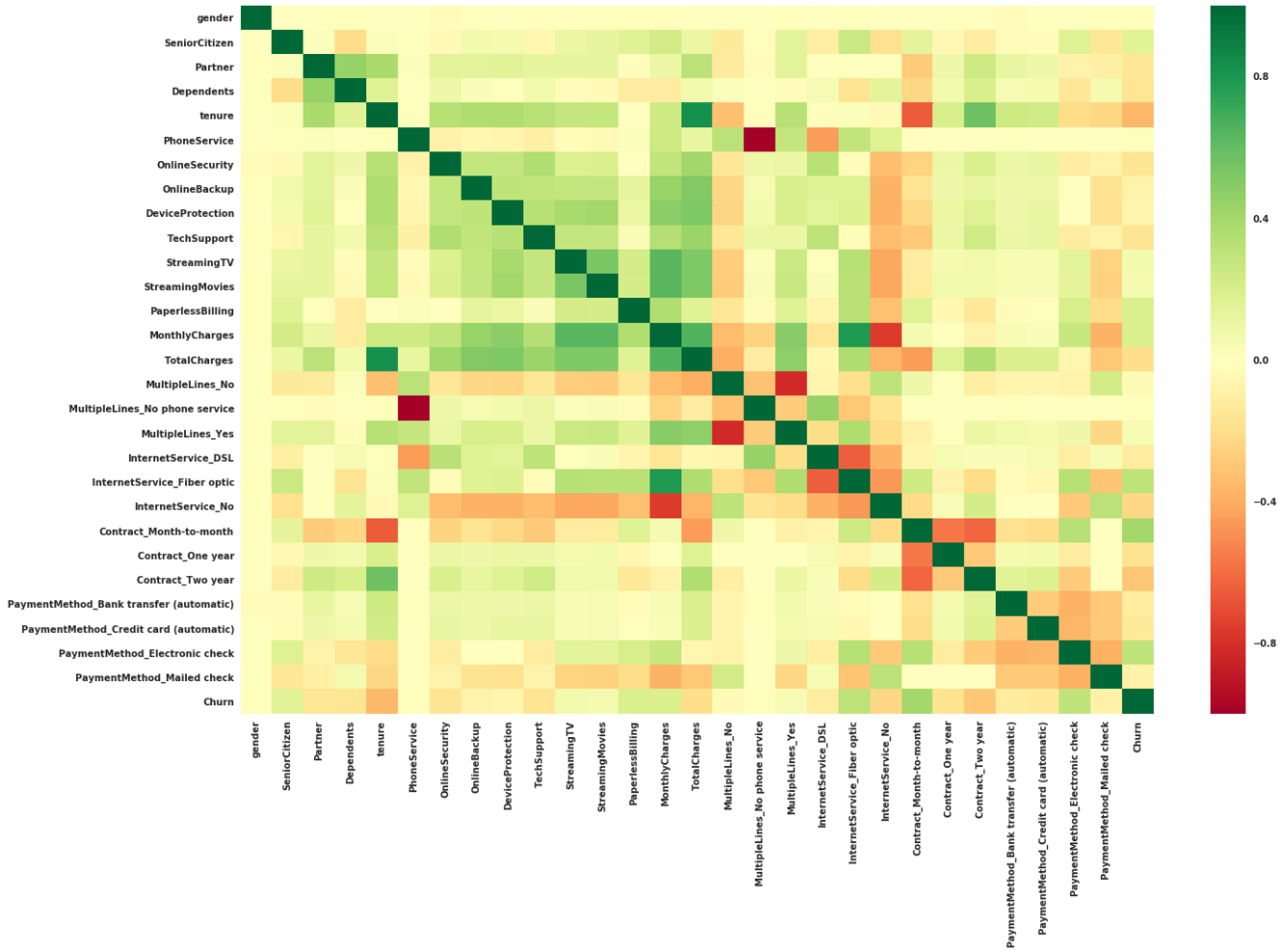


Fig 2

Algorithms and Techniques

The type of solution suited for this problem is a supervised binary classification machine learning model. Each customer data point has attributes associated that can be categorized by a model as a certain 'type' of customer. These types can then be matched against the label of churn/no churn by training over a majority of the dataset, leaving a small subset of data as a test set for evaluation. Because there are several categorical attributes of which many are binary in nature combined with the fact that we want to predict a category (churn/ no churn) the algorithm that seems most appropriate for this application is a decision tree classifier. However, we will test that theory by using other classifiers like XGBoost, LightGBM, which are gradient boosted decision tree algorithms as well as an SVM classifier.

The input data will be preprocessed as explained in the next section, split into training and test sets, and fed into each of the classifiers. The SVM classifier

is sensitive to outliers in the data. However, as stated, the numerical feature fields in this dataset have been shown to not have any significant outlier data points.

Benchmark

In order to see how our chosen models compare, we need to have a baseline model identified. In my research, I wasn't able to find a comparable benchmark model for the churn problem in this project. Therefore, we create a naive model, one that predicts that all customers will churn. Given this basic model and using the metrics we'd like to measure on, the results are as follows:

**** BASIC NAIVE MODEL ****
Accuracy = 0.265784982935
F2 Beta Score = 0.6441273779983457
Recall Score = 1.0
Precision Score = 0.26578498293515357

However, we can perhaps do better.

In the context of churn analysis, we can theorize that perhaps the longer a customer has been with the company, the less likely they are to leave. Conversely, the shorter a customer has been with the company, the likelier they are to leave. We can subsequently derive a model that will simply look at this attribute alone to determine churn risk. In this case, the assumption was made that every customer will churn if *tenure* is less than 12 months. In this case, the results are better:

**** IMPROVED NAIVE Benchmark ****
Naive Tenure-based Benchmark Accuracy = 0.7256825938566553
Naive Tenure-based Benchmark F2 Beta Score = 0.5239144115796098
Naive Tenure-based Benchmark Recall Score = 0.5345104333868379
Naive Tenure-based Benchmark Precision Score = 0.48542274052478135

Here we see that the F2 Score is slightly worse than the naive model, however accuracy and precision are much higher. Recall is significantly lower but this is compared to a naive model that has perfect recall because it predicts all customers churn. The perfect recall of the naive model is not a realistic value to compare. Therefore, this improved benchmark will be what we will use to compare against a chosen machine learning model.

III. Methodology

Data Preprocessing

The input data required preprocessing in order to prepare it for machine learning models. We've already addressed the 11 missing values and have removed those data points. We've also dropped the *CustomerID* column as it does not add useful information to a machine learning model.

Next, we simplify the data by replacing the string value 'No Internet Service' to 'No' for the '*OnlineSecurity*', '*OnlineBackup*', '*DeviceProtection*', '*TechSupport*', '*StreamingTV*', '*StreamingMovies*' columns. This now makes each of these columns a binary categorical field, which will aid in the future step of one-hot labeling.

Because our numerical columns (*MonthlyCharges*, *TotalCharges* and *Tenure*) are on significantly different scales, we perform a standard scaling of these fields. This will enable classifiers like SVM to perform better.

Once the numerical columns have been scaled, we can focus on the categorical columns. Most of the categorical columns are binary (yes/no), however, some are multiclass. The non-binary categorical columns (*'MultipleLines', 'InternetService',' Contract',' PaymentMethod'*) are one-hot encoded, producing multiple new binary columns of 1's and 0's. Similarly, the binary categorical columns are one-hot encoded as well. Finally, the same treatment is done to the label column, which has now been separated into its own dataframe, *y.*

At this point we have scaled and one-hot encoded features, *X*, and label, *y*. The one-hot encoded columns have added new columns, or features, to our dataset. Our new feature dataframe, *X*, now consists of 28 columns. To reduce potential high variance and overfitting in our learning models to be tested, it is prudent to reduce dimensionality, where possible. We apply Principal Component Analysis (PCA) to the feature dataset. It is determined that the 1st 21 components comprise 98.3%. Therefore, only these components will be used for our learning models, dropping the 8 least significant components.

Implementation

Now that our data is processed, dimensionally reduced and ready for ingesting into our classifiers, we first split our data into training and testing sets. We choose a 15% testing set for our model evaluations.

To maximize model accuracy, we use a Gridsearch cross validation to choose the optimal hyperparameters for each model. The validation set of 20% is chosen. For each model, we select a range of sensible hyperparameters to run our models on and select the parameters that maximize our F2 score metric.

Since we know our data is imbalanced, we use a stratified shuffle split method for our cross validation sets with a 5-fold parameter. This will help balance the data sets and increase prediction accuracy in our models.

The models selected for this project are as follows: Support Vector Classifier (SVC), Decision Tree, XGBoost, LightGBM. The latter 3 are decision tree based classifiers that should work well for this type of binary classification task with a relatively small data size. The SVC model works well in general classification problems such as this one.

SVC

The support vector machine model is run through a gridsearch cross validation with the default RBF kernel and the C value from the list of 1000, 1e07 and 1e09. The gamma values were chosen from the list 1e-09, 1e-10. The gridsearch cross validation yielded the optimal parameters: *{'C': 1e07, 'gamma': 1e-10}*. These parameters are used to run the SVC model on the training set. The final complete parameter set for the SVC model is:

SVC(C=10000000.0, cache_size=200, class_weight=None, coef0=0.0,decision_function_shape='ovr', degree=3, gamma=1e-10, kernel='rbf', max_iter=-1, probability=False, random_state=1, shrinking=True, tol=0.001, verbose=False)

Decision Tree

The decision tree model is run through a gridsearch cross validation with the following hyperparameter lists:

{'criterion':['gini', 'entropy'],
'max_depth':[3, 5, 8,12],
'splitter':['random','best'],
'min_samples_leaf':[1,2,5,10]}

The optimal parameters that maximized our F2 score metric were:

{'splitter': 'random', 'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 10}

These parameters were used to run the decision tree model on the training set. The final complete parameter set used for this model is:

DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=3,max_features=None, max_leaf_nodes=None,min_impurity_decrease=0.0, min_impurity_split=None,min_samples_leaf=10, min_samples_split=2,min_weight_fraction_leaf=0.0, presort=False, random_state=1, splitter='random')

XGBoost

The XGBoost model is run through a gridsearch cross validation with the following hyperparameter lists:

{'booster':['gblinear', 'gbtree','dart'],
'learning_rate':[0.1, 0.01, 0.3, 0.6],

'max_depth': [1,3,9],
'scale_pos_weight': [2.76, 5.5, 6.0]}

The optimal parameters that maximized our F2 score metric were:

{'scale_pos_weight': 6.0, 'learning_rate': 0.3, 'max_depth': 1, 'booster': 'gbtree'}

These parameters were used to run the XGBoost model on the training set. The final complete parameter set used for this model is:

XGBClassifier(base_score=0.5, booster='gbtree',
colsample_bylevel=1,colsample_bytree=1, gamma=0, learning_rate=0.3,
max_delta_step=0,max_depth=1, min_child_weight=1, missing=None,
n_estimators=100,n_jobs=1, nthread=None, objective='binary:logistic',
random_state=1,reg_alpha=0,reg_lambda=1,scale_pos_weight=6.0,seed=None,s
ilent=True, subsample=1)

LighGBM

The LightGBM model is run through a gridsearch cross validation with the following hyperparameter lists:

{'boosting':['gbdt','goss','dart'],
'min_data_per_leaf':[5,10,30,500],
'num_leaves':[5,31,80,150],
'max_depth': [1,3,7,9],
'is_unbalance':[True],
'objective':['binary'],
'random_state': [1]}

The optimal parameters that maximized our F2 score metric were:
{'num_leaves': 5, 'min_data_per_leaf': 5, 'random_state': 1, 'is_unbalance': True,
'boosting': 'dart', 'objective': 'binary', 'max_depth': 3}

These parameters were used to run the LightGBM model on the training set. The final complete parameter set used for this model is:

LGBMClassifier(boosting='dart', boosting_type='gbdt',
class_weight=None,colsample_bytree=1.0, importance_type='split',
is_unbalance=True,learning_rate=0.1, max_depth=3,
min_child_samples=20,min_child_weight=0.001, min_data_per_leaf=5,
min_split_gain=0.0,n_estimators=100, n_jobs=-1, num_leaves=5,
objective='binary',random_state=1,reg_alpha=0.0,reg_lambda=0.0,silent=True,s
ubsample=1.0, subsample_for_bin=200000, subsample_freq=0)

Refinement

The decision tree algorithm was initially run without a balanced class weight parameter. This yielded poorer results. An F2 score of 0.43 was obtained without a 'balanced' weight parameter. After studying the parameter definitions for the classifier, I learned that for imbalanced datasets such as this one, the class weight parameters will yield better results overall. This was evident after the parameter was turned on, the F2 score jumped to 71.8%.

IV. Results

Model Evaluation and Validation

After running all 4 models, we evaluate and compare them against our desired metric, which we determined to be the F2 score and also to a lesser degree the recall, precision and accuracy as a collective. The model that attained the highest F2 score was the XGBoost model, with a 0.75 score. It also attained an impressive 0.91 recall score, meaning that very few churn customers were falsely categorized as "no churn". The final metrics for all models is as follows:

Model	F2 Score	Recall	Precision	Accuracy
Support Vector Machine	0.702718	0.855634	0.409781	0.629384
Decision Tree	0.718149	0.841549	0.452652	0.683412
XGBoost Classifier	0.750144	0.915493	0.435511	0.65782
LightGB Classifier	0.701587	0.778169	0.503417	0.733649
Naive Benchmark	0.644127	1	0.265785	0.265785
Improved Naive Benchmark	0.523914	0.53451	0.485423	0.725683

Fig 3

The low precision value of this model does mean that many non-churn customers are identified as churn. As we can see in Fig 4, the confusion matrix for each of the models tested clearly shows that the XGBoost has the highest prediction rate for churn customers, only missing 24 of 284 churn customers in the test set as compared to 41, 45 and 63 for SVC, Decision Tree and LightGBM, respectively. LightGBM had the least number of false positives with 218, compared to 350, 289, and 337 for SVC, Decision Tree and XGBoost, respectively.

Depending on the application of this model, precision might be a factor to consider. For example, if the desired strategy is to target churn customers with an expensive preventative campaign, a low precision score means much higher costs and a lower return on investment. However, sticking to our desire for a high F2 score and high recall, the XGBoost model delivers the optimum results for our application.

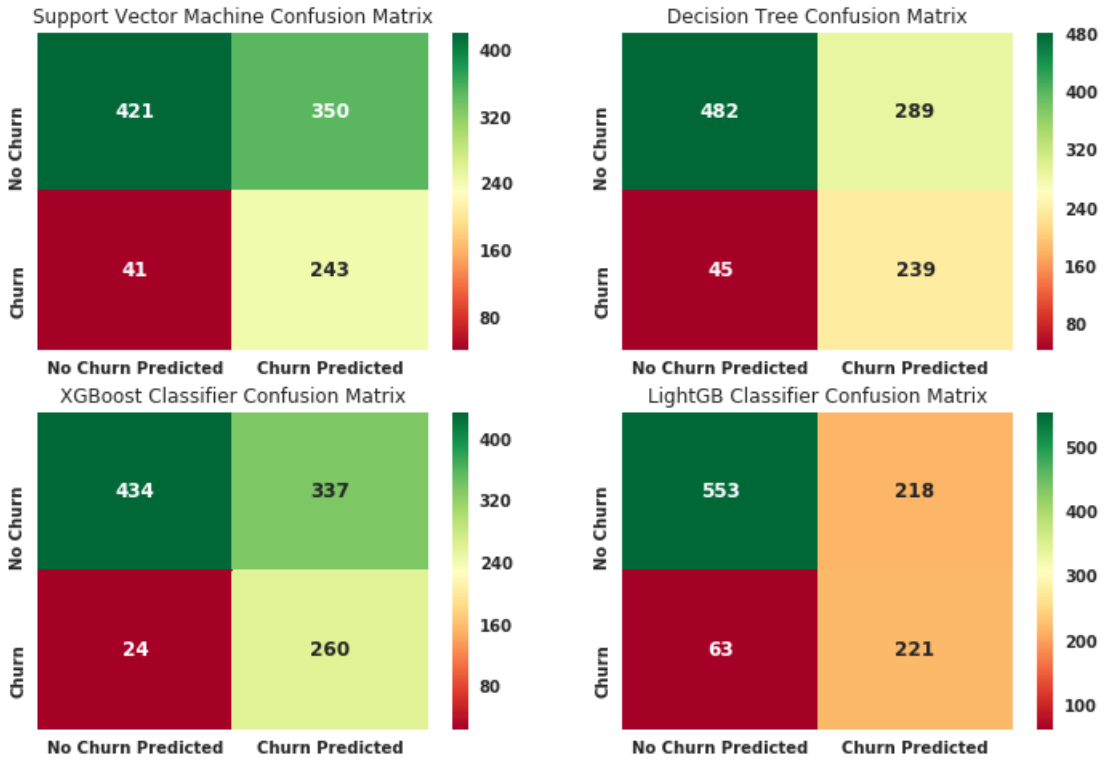


Fig 4

The model's robustness was tested by comparing the results with a different test-training set split. Specifically, a 20% test set model was run, as opposed to the initial 15%. The result was as follows:

Model	F2 Score	Recall	Precision	Accuracy
Support Vector Machine	0.715849	0.874332	0.414975	0.638948
Decision Tree	0.716887	0.842246	0.449358	0.683724
XGBoost Classifier	0.744401	0.906417	0.434059	0.660981
LightGB Classifier	0.717017	0.802139	0.503356	0.737029
Naive Benchmark	0.644127	1	0.265785	0.265785
Improved Naive Benchmark	0.523914	0.53451	0.485423	0.725683

Fig 5

The result shows that the XGBoost F2 score decreased by only about 0.006. The F2 scores for the other 3 models increased, with the biggest increase in the SVC model, by about 0.013.
This shows that small changes do not seem to cause significant swings in performance in our chosen model.

Justification

At this point, we compare our chosen model, XGBoost, to our benchmark models we derived earlier in this project. Our improved naive benchmark has a 0.52 F2 score which is significantly lower than our XGBoost model's 0.75. We can stop there and declare that our benchmark was beaten but there are some more interesting results we can explore. The naive benchmark actually did have a better accuracy score, with 0.726 as compared to 0.658 for XGBoost. This is significant, however because accuracy is not the best measure for our application in this case, this finding is not relevant to finding the best model. For example, the poor recall of the naive model of 0.53 means that it misses almost half of the churn customers in it's predictions. The XGBoost predicts 91.5% of the churn customers correctly.

I feel confident that the chosen XGBoost classifier is a more robust model that any naive solution we can contrive. It correctly identifies 91.5% of churn customers with an F2 score of 0.75. I believe this model is the correct solution for a company that is willing to spend resources to prevent every last potential churn customer from leaving.

V. Conclusion

Free-Form Visualization

Now that we've selected a model, we can take a look at what exactly our model considers when making a prediction. Machine learning models are many times considered black boxes in which we can't easily peer inside and see how things work. However, because XGBoost is essentially a decision tree, we are able to access some important information about how it uses the features in our dataset. One way to look at this is looking at XGBoost's 'Feature Importance' bar chart (Fig 6).

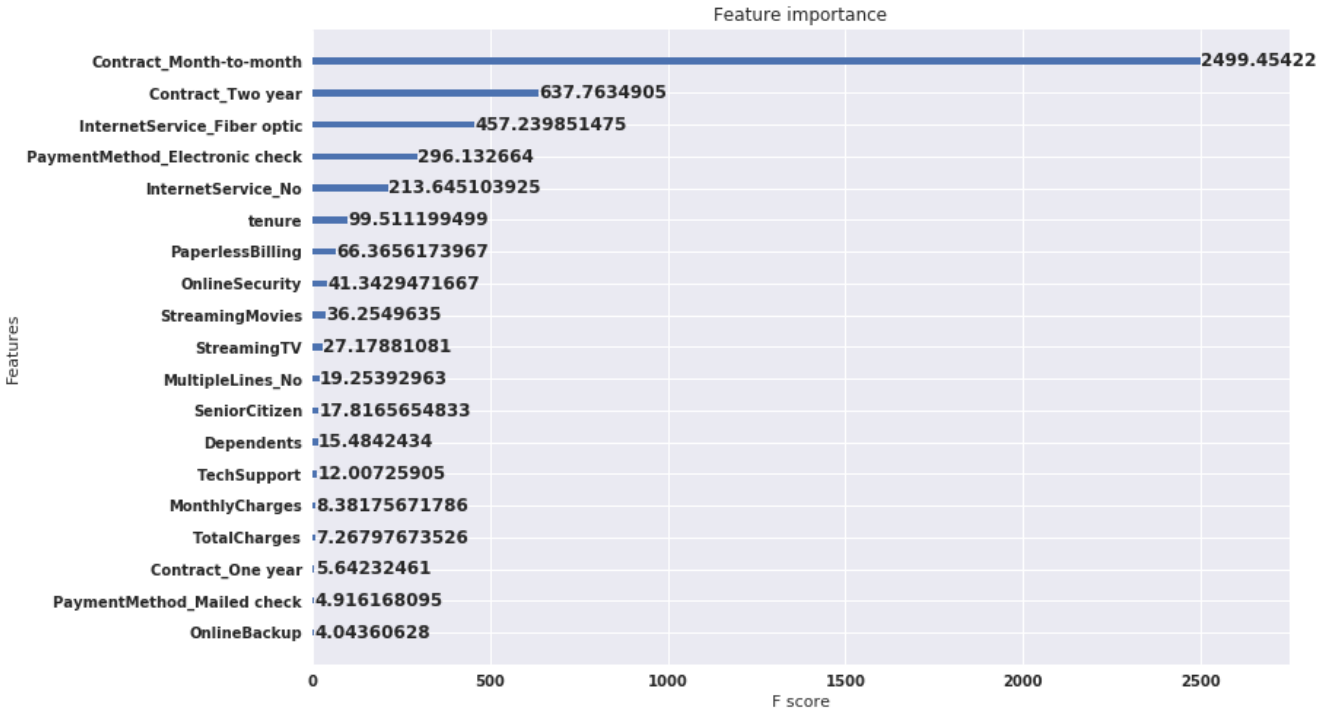


Fig 6

This report measures feature importance based on 'gain', which is the average training loss reduction gained when using a feature for splitting [1]. As we can see, this chart makes it easy to see that *Contract-Month-to-Month* is by far, our most important feature, followed by *Contract_Two-year* and *Internet_Service_FiberOptic*.

Evaluating this result, it does makes sense. It's sensible that a customer not on long term contract would be at higher risk to defect. It also makes sense that having a 2 year contract is likely strongly negatively correlated to churn. Surprisingly, the model is indicating that having Fiber Optic service is also an important feature in determine churn likelihood. This provides our telecom valuable information as to what strategies it may employ to minimize churn.

Reflection

So far, we have arrived at an adequate machine learning model that predicts whether a customer is likely to defect from our telecom company. This model accurately predicts 91.5% of potential churn customers. To get here, we imported our data from kaggle, cleaned up missing data, then one-hot encoded our feature and label data. We scaled our numerical features columns using a standard scaler for optimum model performance. After exploring for outliers, and looking for correlations, we produced pie charts showing some of our variable distributions. We performed Principle Component Analysis on the data and determine that the 1st 21 components described more than 98% of the variance and therefore were able to reduce the dimensionality from 28 to 21. We created 2 baseline benchmark models, with one being more accurate. We chose 4 supervised classification models to test: SVC, Decision Tree, XGBoost, and

LightGBM. We used a gridsearch method to find optimal parameters that maximized our F2 score for each model. The models were then compared against each other using F2 score, Recall, Precision and Accuracy, with an emphasis on F2 score. Our winning model, XGBoost, was chosen due to it's high F2 score and excellent Recall.

One of the interesting aspects of this project was taking into account the imbalanced nature of the data in terms of the label class. Since only about 27% of the data was identified as Churn, I had to take that into account with the model parameters. Also, the cross validation set was set originally as regular k = 5 folds until I realized that random folds can possibly cause greater imbalance in these validation sets. Upon investigating further, I found that Stratified Shuffle Split would do a much better job of finding balanced but still, random folds.

I would trust this model to be used in the wild with some important caveats. The model is trained on a particular customer set from an undisclosed company. I would not feel comfortable using this model on any other telecom other than the one from which this data was sourced due to many dependencies related to differences in telecom customer base. For example, one telecom may attract a very young demographic base compared to another. The churn label we strive to predict may be dependent on these differences.

Improvement

One improvement I would make would be to test more models. In particular, I would have liked to test an Artificial Neural Network. However, there was a good reason this was not tested in this implementation, as our data volume was very limited and not adequate for an ANN. A neural network would have been interesting because these types of models, when built right, learn which features are important and therefore theoretically come up with very efficient and accurate solutions. The drawback is that these types of models are compute-heavy and require hefty hardware resources. If a shallow algorithm does the job as well as a neural network, it might just make sense to use a shallow learning model.

It's very likely there is a better solution out there. However, given the sparsity of the dataset, I doubt there is a solution that is significantly better.

References

[1] https://towardsdatascience.com/interpretable-machine-learning-with-xgboost-9ec80d148d27

[2] https://www.kaggle.com/blastchar/telco-customer-churn

[3] https://xgboost.readthedocs.io/en/latest/