Grant Aguinaldo
Machine Learning Nanodegree Capstone Project
Revision 2

Identifying the Key Features of a Customer Dataset that can be used to Predict Chrun

1. ***Definition***

a. Project Overview

We live in a subscription-based economy. The global consulting firm, McKinsey & Company noted that:

> *The subscription e-commerce market has grown by more than 100 percent a year over the past five years. The largest such retailers generated more than $2.6 billion in sales in 2016, up from a mere $57.0 million in 2011.*

However, while it seems that things are rosy for these businesses there is a stark reality: churn. Simply put, churn can be defined as the rate at which a company loses customers during a given time period. Indeed, despite record and rapid growth for these subscription-based businesses, churn can bring a subscription-based down, as quickly as it has grown, if companies do not deliver a superior product. Given the importance of the churn rate to the bottom line of any subscription-based business, it is key that business leaders not only track and monitor the churn rate but also identify the key factors that contribute to customer churn. By doing so, it then becomes possible to not only focus the business on customer acquisition efforts but more important than that, are the customer retention strategies that are needed for sustainable growth.

This project was inspired by previous work from the New York University where Deep Learning was used to predict customer churn. For this project, we developed an approach for identifying the key indicators from within a customer data set that most contribute to churn. Further, we also developed a predictive model, based on the features in the dataset that is able to predict customer churn.

The motivation for undertaking this project is two-fold. First, the large shifts of traditional based business to subscription-based business models (like Adobe) as well as a large amount of activity of funding these businesses by venture capital firms. Second, is the notable acquisition of Whole Foods by Amazon in 2017 for which many believe was a move to strengthen Amazon's understanding of their current customer base via the data that Whole Foods have collected on their customer base. Finally, For this project, we will be using a dataset named "`WA_Fn UseC_ Telco Customer Churn.csv` "that was obtained from IBM's Watson's Analytics ("telco dataset").

b. Problem Statement

As part of this project, we developed a supervised machine learning model/classifier to predict customer churn from a given set of customer metadata (as provided within the dataset). On a high level, the problem does have one potential, supervised learning, solution since there is a clear set of independent and dependent variables (discussed further in the following section). In

addition, this problem is quantifiable in that all of the features of the data set are either continuous or categorical and can be converted or vectorized in order to build a numerical representation of each customer.

To solve this classification problem, we will utilize supervised machine learning in order to build a mathematical relationship between the features in the input space and the ground truth outcome (i.e., response variable). Overall, this project followed the following framework: exploration, cleaning, baseline model development, model optimization/tuning, and model assessment.

c. Metrics

For this analysis, the primary metric that was used to determine the quality of both the baseline and the final model is the Recall score of the minor class (i.e., Class 1). Within this project, the terms “minor” and “major” classes refer to the classes in a given imbalanced dataset. This project involved building a binary classifier that will predict, given a set of customer data, if the customer will churn or not. That being said, we sought to build a model that afforded the highest Recall score (value can be between 0-1) out of all of the models that are tested. Within the context of this problem, a higher Recall score meant that the model not only capture all of the true positives but also minimizes all of the false negatives. By using a high recall model, that is, a model that minimizes the number of false negatives, we are ensuring that all customers that are classified as being lost to churn have the opportunity to be reviewed by the analyst. In passing we define a false negative as being a situation where the model predicted that a customer was labeled as "no churn," but the customer was indeed lost to churn. Finally, as it relates to the utility of this model, we have taken the position that, it's okay for this model to predict that a customer did "churn" even though it wasn't labeled as such; however, it is not okay for our model to predict that a customer did not churn when in actuality, we did lose the customer due to churn.

2. *Analysis*

a. Data Exploration

The dataset used in this project was provided by IBM, and includes data that was obtained from a customer relationship management platform for a telecommunications (“telco”) company. It is unknown of the data is an export of real data or not, but when looking at the dataset, one cannot discern a difference. In term of features, the dataset includes 21 types of metadata (or features) for 7,043 different customers. Included in the 21 features is a binary flag that identifies if the particular customer has churned or not (Churn = 1, Not Churn = 0).

The 21 features of the dataset include a mixture of categorical and continuous data types. As an example, the `SeniorCitizen` field includes two responses, 0 and 1. During this analysis, we have assumed that a zero means that the customer is not a `SeniorCitizen` and a one means that the customer is a senior citizen. In another example, the field `InternetService` has three uniques values that include: DSL, Fiber optic, and No. In a final example, the field

`MonthlyCharges` includes continuous values ranging from $18.80 to $8,684.80. One observation that we did not, was that while none of the 21 fields presented had any missing data, there were 11 null values that was present in one feature of the dataset, namely, `TotalCharges`. For these values we needed to further analyze the data within the context of the problem in order to properly impute missing values for these data points.
A summary of all of the features and the types of responses included with each is provided below as Table 1. We used this table to re encode the feature values so that we could use the dataset in this project.

Table 1: Summary of all Features Included in Telco Dataset

Feature Name	Description	Unique Values
customerID	Unique customer ID number.	Infinitely many
gender	Gender of customer.	[Female, Male]
Senior Citizen	Flag if customer is a senior citizen.	[0, 1]
Partner	Flag if customer has a partner.	[Yes, No]
Dependents	Flag if customer has dependents.	[Yes, No]
tenure	Length of time customer has been active.	Continuous
Phone Service	Flag if customer has phone service.	[Yes, No]
Multiple Lines	Flag if customer has multiple lines of service.	[No phone service, No, Yes]
Internet Service	Flag if customer has internet service.	[DSL, Fiber optic, No]
OnlineSecurity	Flag if customer has online security feature.	[No, Yes, No internet service]
Online Backup	Flag if customer has online backup feature.	[No, Yes, No internet service]
Device Protection	Flag if customer has device protection.	[No, Yes, No internet service]
TechSupport	Flag if customer has used technical support.	[No, Yes, No internet service]
Streaming TV	Flag if customer has the streaming TV service.	[No, Yes, No internet service]
Streaming Movies	Flag if customer has the streaming movie service.	[No, Yes, No internet service]
Contract	Flag if customer is on a contract.	[Month-to-month, One year, Two year]
Paperless Billing	Flag if customer is enrolled in paperless billing.	[Yes, No]
Payment Method	Payment method used by customer.	[Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic)]
Monthly Charges	Total amount charged on a monthly basis.	Continuous
Total Charges	Total amount charged over tenure of the customer.	Continuous
Churn	Flag is the customer has churned at the time of the report.	[No,Yes]

For this project, it is important to note that the out of the 7,043 customers, the data set includes 1,869 customers who “churned” (i.e., 26% Class 1) and 5,174 who have not (i.e., 74% Class 0). This is important since more “real world” datasets similar to this are “imbalanced” meaning that there are more observations one one class over another. The methods that we employed to address the class imbalance is discussed within this document.
Finally, for this data set, summary statistics was determined within the provided notebook and presented in the data below.

Table 2: Summary Statistics of Continuous Features in Telco Dataset

Parameter	Tenure	Monthly Charges	Total Charges
count	7,043.00	7,043.00	7,043.00
mean	32.37	$ 64.76	$ 2,279.80
std	24.56	$ 30.09	$ 2,266.73
min	-	$ 18.25	$ 18.80
25%	9.00	$ 35.50	$ 398.55
50%	29.00	$ 70.35	$ 1,394.55
75%	55.00	$ 89.85	$ 3,786.60
max	72.00	$ 118.75	$ 8,684.80

b. Exploratory Visualization

Before embarking on this project, a few visualizations of the data was completed and include barcharts, pairchats and scatter plots.

i. Class Distribution

In the chart below, we sought to understand the distribution between the “Churn” and “No Churn” classes in order to visualize the class imbalance.

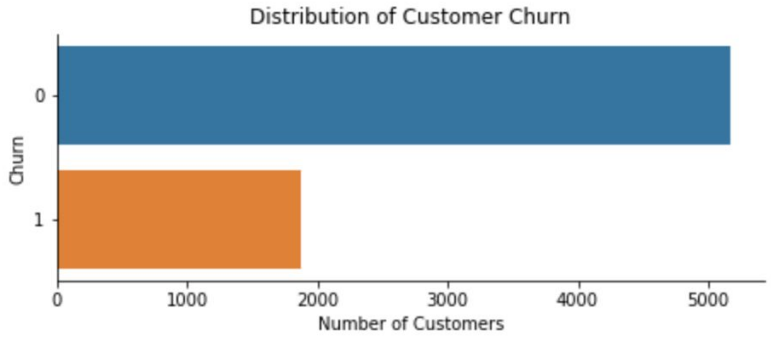


Figure 1: Distribution of Customer Churn within the Telco dataset (0: No Churn; 1: Churn).

As seen in Figure 1, there is a 74%-26% split between the no churn and churn classes. This is important for two reasons. First, if we were to use the dataset as it, it is possible that the machine learning model could be biased to the No Churn (i.e, Class 1) class since that is the majority class and thus the is the type of data that the model will see more of. Second, in real-world datasets, is is common to have an imbalance between the categories within the dataset, and this is even more so, when you are dealing with a problem where you are trying to find rare events, like customer churn, credit card fraud, or cancer detection.

i. Effect of paperless billing and churn

In November 2007, Aspen Analytics investigated the effect of paperless billing on customer attrition. In the executive summary of that report, it is noted that "Customers that adopt e-bill are much less likely to attrite." and "Customers using e-bill products are substantially more profitable."

In the telco dataset, we were interested in determining if the same conclusions could be made. As shown in Figure 3, the telco dataset does show consistent results in that, overall, customers who have not churned and are enrolled in paperless billing do have higher "TotalCharges" (interpreted to be gross profit) than customers who are not enrolled in paperless billing. This finding is consistent with the Aspen study.

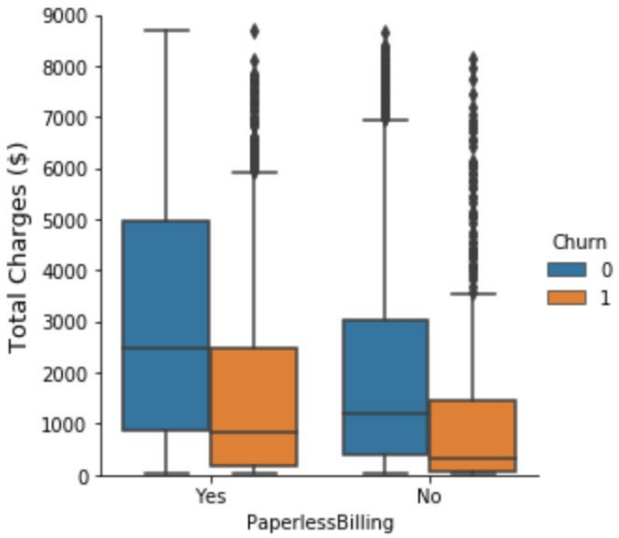


Figure 2: Distribution of Customer Churn within the Telco dataset as a function of Paperless billing (0: No Churn; 1: Churn).

c. Algorithms and Techniques

For this project, we have used the following algorithms and techniques.

Resampling

Class imbalance results when each class in the data set is not of an equal size and is quite common when working with actual data as opposed to toy data. As it relates to machine

learning, there can be several problems that can arise when working with an imbalanced dataset that include the development of a model that is biased to the majority class since those are the data points that the model sees the most of.

To mitigate this problem, and not increase performance of the classifier, is is common to use a one or more methods to assess class imbalanced in a dataset. While there are several methods available to address class imbalance, the two that was used in this project include upsampling of the minor class and downsampling of the major class.

Recall vs. Precision

Recall and precision are two common metrics that are used to assess the quality of a classification model. In general, the recall of a model is the ratio of the true posibies to all of the total actual positives. On the other hand, precision of a model is the ratio of the true positives to all of the predicted positives. As it relates to this problem, the model that we was built was optimized to be a high recall model since we wanted to be sure that we found all instances of customer churn, even at the expense of doing additional work when investigating any false positive classifications.

Train/Test/Split

In practice, the desired result of building a model is to take a current dataset, and generate a model that can make predictions based off of data the model has not seen. Over- and under-fitting of a model to the dataset can learn to poor generalization and utility of the overall model. Overfitting a model means that the model has learned all of the nuances of the dataset being used too well and will perform very well on the data set provided, but will tend to perform poorly on new data. In contrast, underfitting a model means that that the model does not pick up all of the nuances of the dataset and again, will tend to perform poorly on new data. In both cases, the model will not generalize well and will be of very little utility to the end user.

To avoid either model under- or over-fitting, this project used the `train_test_split` function within sk-learn. The benefit of using `train_test_split` in modeling is that it allows you to ensure that the model does not over- or under-fit the data since you will evaluate the model using data (i.e., the 'test' set) that the model has not seen. This in-turn ensures that the model is able to generalize well to out of sample data.

Grid Search

Each of three machine learning models used in this project have several hyperparameters that can be adjusted to enhance the performance of the model. To tune each model, this project used the Grid Search function within sk-learn. A Grid Search is an exhaustive process that determines the performance of each model using a given set of input parameters. During this process, the model will run through each combination of hyperparameters (as a Cartesian product) and determine the performance of the model based on those parameters. At the end of

the process, the set of parameters that affords the best performance is selected and returned to the user.

Learning Curves

Learning curves allow us to quickly determine the performance of the model as a function of the number of data points in the training set. Specifically, the learning curve will allow us to know if the model has an issue with balancing the bias (an underfit model) and variance (an overfit model). In practice we will need to determine the level of bias and variance that will be acceptable in order to provide the most utility to the end user.

Label encoding and scaling of data

As noted previously, the telco dataset contains a mixture of both continuous and categorical variables. In order to use the telco data in a machine learning model we will need to convert the categorical data (i.e., convert the churn field from “yes” or “no” into a binary output of 1 and 0) into numeric vector in d-dimensional space. In addition, this project also scaled, using sk-learn’s `StandardScaler` function, all of the input features since all of the features were on different scales. For example, the total cost column in the dataset was in dollars, but the churn column, after vectorization was in 0 and 1. This can be a problem since it is possible for a model to to use the the features with the greatest magnitude as the most important feature in the dataset. Once all of the features have been standardized, the mean of the column will be set to zero and the standard deviation will be set to 1.

Imputation of missing data

For any data set, it is possible that certain features may be missing for each observation. In this case, determining which data elements are missing and more importantly how to address this missing points are important to the overall analysis. For this project missing data was imputed based on the context of the data point in question and will be explained in the results section of this report.

Logistic regression

Logistic regression is a common machine learning algorithm for binary classifications. The algorithm takes a series of input features, and computes the probability that the input sample belongs to one class or another based on the logit function.

$$\frac{1}{1 + e^{-value}}$$

The value used in the logit function is determined/estimated from the coefficients determined by training a logistic regression algorithm using the maximum-likelihood estimation. An example of a logistic regression equation for one feature is given below.

$$y = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

In general, the maximum-likelihood estimation is an algorithm that determines the probability that a given sample y belongs to a given class, given the training data and the coefficients (i.e., β_0 and β_1) of logistic regression equation. The process of actually determining the best coefficients for the logistic regression equation is an interactive process that involves minimizing a given cost function using stochastic gradient descent. The cost function that is used when training a logistic regression model is called the inverse logistic cost function and is shown below (the image provided below of the inverse logistic cost function was obtained from this Quora post).

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} y^i \log(h_\theta(x^i)) + (1 - y^i) \log(1 - h_\theta(x^i))$$

Where $h_\theta(x)$ is defined as follows,

$$h_\theta(x) = \frac{1}{1+e^{-\theta(x)}}$$

Once the coefficients are determined from the training data, making predictions (calculated as probabilities of being to one class) using the logistic regression equation is as easy as plugging in the values from the test data to an equation containing the learned coefficients.

Support vector classifier (SVC)

The support vector classifier is another common type of classification algorithm that can be used for binary classifications. The goal of an SVC is to fit a hyperplane through the data as a means of splitting the data into two (or more) classes. As expected, the coefficients used in the formula to describe the hyperplane is empirically determined using numerical methods like gradient descent to afford. Once the coefficients are determined, the test data is plotted and an assessment is made as to what side of the hyperplane does the test data lie. In other words, once you have the equation of the hyperplane, you can add the test data to the equation and determine whether the test data is above or below the hyperplane (in the case of a binary classifier).

When training the SVC, the best hyperplane will be the one that has the largest margin. Within this context, the margin is the distance between the hyperplane and the data points closest to the plane. In all, when training an SVC, you are using gradient descent to maximize the margin from the points in the training dataset.

The main point of an SVC is to use a hyperplane to separate the data, and thus, one of the key assumption is the that the data is linearly separable. In real life, however, it is rare that an actual dataset may indeed be linearly separable. In this case, the data may need to be transformed

into a higher dimension in order for the data to be linearly separable (i.e., non-linear data). This is called the "Kernel trick." Some of the common kernels used to separate, non-linear data, include polynomial kernel and the radial kernel.

XGBoost classifier

The XGBoost classifier is one of the most popular and best performing machine learning models in the community. In general, the XGBoost classifier is based off of an ensemble of decision tree classifiers that has been optimized for speed and model accuracy. That is, the model takes a series of low performing models (i.e., weak learners) and combines to form a higher performing model (i.e. a strong learner). The process of combining weak learners to create a strong learner is an iterative one where the model adds a second learner to compensate for the shortfalls (or errors) of the first one (i.e., a new model is added to correct for the residuals from the first model).

When training a boosted ensemble, the primary objective is to sequentially build a series of decision trees to minimize the residuals from the previous model outputs. During this process, gradient descent is used to used to minimize a loss function based on the pseudo predicted determined by the current iteration of the model. For the implementation of XGBoost used for this project, the loss function we used was the `binary:logistic` since we have a binary classification problem. While the core of the XGBoost algorithm is a boosted tree model, the XGBoost adds additional functionality that allows for the efficient processing of large data sets.

Principal component analysis

Principal component analysis ("PCA") is a method that is used to reduce the number of features in the input space and maintain the same amount of variance as was present in the original feature set. In other words, PCA searches for new features that show as much variation over all of the data points as possible. In addition, during the process, PCA is able to show you which features in a dataset are more important for explaining variance as well as showing you which features are correlated with each other

d. Benchmark

For this project, the benchmark metric that will be used to assess model performance is the null accuracy or the null error rate. The null accuracy can be defined as the accuracy if the model always predicts most common class, and is, for the telco dataset, 74%. Despite this however, it is important to note that we are seeking to build a classifier that is able to find all of the rare events (i.e, customers who did churn) and therefore, the null accuracy is not relevant. In its place, we will use the percentage of the customers who did churn (26%) as the benchmark for this problem.

3. Methodology

a. Data Preprocessing

As described earlier, the telco dataset required several preprocessing steps in order to be used with the selected machine learning algorithms of this project.

Vectorization of data via a label encoding

Since the telco dataset includes categorical data, we will need to convert these categorical data points into numerical values. In this project, this was done by first finding all of the unique values for each feature in the dataset, and then using the map function within Python to assign an integer value to each unique response.

Standard scaler

Once the categorical variables were encoding using label encoding, all of the data in the dataset (i.e., both continuous and categorical-encoded values) was scaled using the StandardScaler function within sk-learn. By using `StandardScaler`, all of the data is standardized to a mean of zero and a unit variance.

Changing data types and imputing missing data

When looking at the datatypes of telco dataset, we noticed that the `monthlycharges` and `totalcharges` columns in the dataset were made up of different data types. Specifically, we noticed that `monthlycharges` was an `object` and `totalcharges` was a `float64`. We also noticed that none of the fields had any missing data points since there were a total of 7,043 points for each feature. To convert `monthlycharges` to a numeric value, we utilized the pandas function `to_numeric` with the errors flag to `'coerce'`. By using `errors = 'coerce'`, a `NaN` would be inserted in any cell where the to_numeric function encounters an error.

Following the conversion, we found that there were 11 `NaN` points in the `totalcharges` column. This suggests that the no data related to the `totalcharges` was collected for these customers, even though the dataset did contain the corresponding `monthlycharges`. Upon further inspection, we found that the 11 customers that had `NaN` values for the `NaN` had a zero value in the `tenure` field. For these customers, we backfilled the missing `totalcharges` data with the corresponding `monthlycharges`. Since these 11 customers have been labeled as “no churn”, a reasonable explanation for these `NaN` value is that the dataset was produced in the same month that these customers signed up for telco service.

Analysis of outliers

For this project, an assessment of outliers was conducted using the Tukey method. Out of the 21 columns, the only two columns with continuous features are `tenure`, `MonthlyCharges` and `TotalCharges`. For these three columns, the Tukey method of outlier detection will be used. In short, the Tukey Method for outlier detection flags any values that are more than 1.5 times the interquartile range from the quartiles (i.e., Q1 and Q3) as an outlier.

For the `MonthlyCharges` feature in telco dataset, lower and upper Tukey Fences was calculated to be ($46.02), $171.37, respectively. Since the minimum and maximum values of the `MonthlyCharges` feature was determined to be $18.80 and $118.75, respectively, there we no outliers found in the `MonthlyCharges` feature since these values were within the lower and upper value of the Tukey Fences.

Likewise, for the For the `TotalCharges` feature in telco dataset, lower and upper Tukey Fences was calculated to be ($4,683.52), $8,868.67, respectively. Since the minimum and maximum values of the `TotalCharges` feature was determined to be $18.80 and $8,664.80, respectively, there we no outliers found in the `TotalCharges` feature since these values were within the lower and upper value of the Tukey Fences.

Resampling techniques

Given the class imbalance seen in the telco dataset, this project used two different methods to resample the data: random upsampling of the minor class and random downsampling of the major class. To ensure that the up- and downsampling was reproducible, `random_state = 42` was used in each sampling regimen. A summary of all three datasets used in this project is provided below.

Table 3: Overview of the Resampled Telco Dataset

Dataset	Total Observations	Size of Training Set	Size of Testing Set	Class 1 Count	Class 0 Count
Baseline	7,043	2,465	4,578	1,869	5,174
Upsample	10,348	3,621	6,727	5,174	5,174
Downsample	3,738	1,308	2,430	1,869	1,869

b. Implementation

The process that was used to complete this project can be explained in three steps.

Determine baseline model performance.

To determine the baseline performance of the machine learning model, we have following the standard instantiate/fit/predict workflow that is commonly used with sk-learn. Since the goal of this step of the process is to get a baseline performance metric, the the models will be instantiated using the default parameters and `random_state = 42` to ensure reproducibility. This process was used on the baseline, upsampled and downsampled datasets.

Once the models are run, the quality of the model was assessed by analyzing the Recall score and the shapes of the learning curves. To facilitate the creation of the learning curves functions were written ahead of time and used within the project workflow.

Tune model and assess model performance.

Once the baseline performance for the models have been determined, `GridSearchCV` was then used to tune each model using. During this process, the tuned model is defined as the one that afforded the highest Recall score out of all for the model fits. Once the best hyperparameters was obtained, we also analyzed the recall score, ROC-AUC score and overall shapes of the learning curves.

Identify the most important features for predicting customer churn.

Once we have developed a final model, we will then identify the most important features that relevant for predicting customer churn. To identify these features we will either use relevant

functionality built into the model (i.e., `plot_importance` function for XGBoost, or the `coef_` attribute for SVM or logistic regression).

Note to the file: Initially, we proposed to use PCA and a biplot to find the most important features, however, after initially completing the PCA with `n_components = 10` we found that dimensionality reduction was not going to afford us an explained variance that we found to be meaningful. For example, we saw that the overall explained variance approached ~41% after including the first two components (i.e., 0.28, 0.13, for principal component 1 and 2, respectively), and did not improve as more components were added. While this may not be a bad thing necessarily, we opted not to move forward with using the PCA analysis to identify the most important features of the dataset since we assumed that we would capture more than 70% of the variance in the first two principal components. The initial language explaining the PCA analysis is maintained for transparency.

> ~~**Conduct PCA and analyze results.**~~
> ~~Once the tuned model is identified, we will then use Principal Component Analysis (PCA) to determine the most important features in the dataset. To accomplish this, we will use the PCA function within sk-learn to fit and transform the entire dataset down to one or more of the principal components and use the Scree plot to determine the optimal amount of principal components to keep. Once we find the optimal amount of components to keep, we will then create a biplot of the data reduced two those components for visualization. This will allow to determine what are the most similar features in the dataset. Functions will be created to facilitate the generation of the biplot.~~

c. Refinement

As mentioned previously, the process to refine each of the three machine learning models (logistic regression, support vector classifier and XGBoost) was to use the `GridSearchCV` function within sk-learn. Since we have three models, and three datasets (baseline, upsampled and downsampled), there was a total of 18 model variations that were completed (calculated as three datasets, and the untuned and tuned variants of three machine learning models). A summary of all of the performance metrics is provided below.

Table 4: Summary Results from All Modeling Runs

Entry	Model	Model State	Dataset	Recall (Class 1)	Precision (Class 0)	AUC Score
1	Logistic Regression	Default	Base	0.58	0.85	0.72
2	SVC	Default	Base	0.52	0.84	0.70
3	XGBoost	Default	Base	0.53	0.84	0.71
4	Logistic Regression	Tuned	Base	0.58	0.85	0.72
5	SVC	Tuned	Base	0.57	0.85	0.72
6	XGBoost	Tuned	Base	0.54	0.84	0.71

7	Logistic Regression	Default	Up	0.80	0.78	0.75
8	SVC	Default	Up	0.81	0.79	0.75
9	XGBoost	Default	Up	0.84	0.81	0.77
10	Logistic Regression	Tuned	Up	0.80	0.78	0.75
11	SVC	Tuned	Up	1.00	0.00	0.50
12	***XGBoost***	***Tuned***	***Up***	***0.89***	***0.84***	***0.72***
13	Logistic Regression	Default	Down	0.77	0.76	0.74
14	SVC	Default	Down	0.78	0.76	0.74
15	XGBoost	Default	Down	0.80	0.78	0.76
16	Logistic Regression	Tuned	Down	0.77	0.76	0.74
17	SVC	Tuned	Down	0.99	0.88	0.54
18	XGBoost	Tuned	Down	0.88	0.82	0.71

As mentioned previously the models in this project was tuned via the `GridSearchCV` function within sk-learn using the following parameters.

Logistic Regression Parameters
To tune the LR model, the following grid search parameters were used:

```
penalty = ['l1', 'l2']
C = np.logspace(0, 1, 10, 100)
cv = 3
scoring = 'recall'
```

Support Vector Classifier Parameters
To tune the SVC model, the following grid search parameters were used:

```
C = [0.001, 0.10, 0.1, 10]
gamma = [0.01, 0.001, 0.0001]
kernel = ['rbf', 'poly']
```

XGBoost Parameters
To tune the XGBoost model, the following grid search parameter were used:

```
learning_rate = [0.01, 0.1, 1]
max_depth = [1, 3, 5, 7]
n_estimators = [100, 1000]
```

Results

d. Model Evaluation and Validation

As part of this project, we developed a machine learning model that was able to predict churn within a dataset containing customer-related information. In total, there were 18 models runs completed within this project and the results of all model runs are presented below.

Model tuning

To tune the machine learning models used in this project, we employed the gridsearch function within sk-learn.

When tuning the logistic regression and XGBoost models, we noticed that the performance of the models did not change as much as we expected. As noted in Table 4, the Recall scores for all of the models did not change before and after turning when using any of the three datasets. As an example, the Recall (Class 1) score for the logistic regression model remained at 0.58 before and after the tuning and this was a consistent pattern amongst all of the logistic regression and XGBoost models and datasets used. These results suggest that each of the logistic regression and XGBoost models are somewhat robust and "insensitive" to changes in hyperparameters.

On the other hand, the SVC model displayed somewhat different performance. When we tuned the SVC model, the model did not respond to hyperparameter changes for the base dataset, but did for the upsampled and down sampled datasets. For example, when the SVC model was tuned using the baseline dataset, the before and after Recall (Class 1) scores were similar to each other at 0.52 and 0.57, respectively. In contrast, the when the SVC model was turned on the upsampled or downsampled dataset, the Recall (Class 1) increased to 1.0 and 0.99, respectively. These results suggest that the SVC model is more sensitive to the changing of hyperparameters when the classes are more balanced.

Class imbalance

The telco dataset displayed a class imbalance meaning that the number of samples in the "Churn" and "No Churn" classes was not in equal proportion. From previous determinations, we calculated that the distribution was 74%-26% between both classes. Since we understood that class imbalance can affect the performance of machine learning models, we decided to resample the dataset using random upsampling of the minor class and random downsampling of the major class and train the three machine learning models on the three datasets. In all, we did see that model performance did increase as the the magnitude of the class imbalanced was diminished. For example, for the tuned XGBoost model, the Recall scores (Class 1) was 0.54, 0.89 and 0.88 for the base, upsampled and downsampled datasets, respectively.

Model validation

To validate the model, and determine if the model is able to generalize well to unseen data, we employed the k-fold cross validation, `train_test_split` methods as well as limiting the number of points used when training the model.

Indeed, when we employed 3-fold cross validation on all three models, we did see that the accuracy scores for each model to be close to each other. For example, in the case of the turned XGBoost model using the upsampled dataset, the Recall scores (Class 1) were 0.89, 0.88, and 0.88. These results suggest that across all of the splits, the model displays consistent performance and therefore, is able to generalize well to unseen data.

In another method to validate the model we also changed the number of training and testing points Based on previous observations (not presented), we noticed that accuracy and shape of the learning curves did not change as the number of training points increased. In one specific case using the updammped dataset and the untuned XGBoost omdel, we saw that the shape of the curve did not change once the model passed the ~1,250 point mark. This led us to believe that we could further validate the model, by increasing the number of unseen points in the testing test and not compromise performance. As expected, when we trained tuned XGBoost model using the upsampled dataset, with 35% of the data, we did see similar performance to the case when we used used 67% of the data points to train the same model on the upsampled dataset. For example, when using 35% of the data points for training and the upsampled dataset, the tuned XGBoost model displayed a Recall score (Class 1) of 0.89 This is compared to a Recall score (Class 1) of 0.95 when using 67% of all of the data points to train the tuned XGBoost model using the upsampled dataset. These results further substantiate the fact that the model is able to generalize well on unseen data. For clarity, all of the model runs presented in this project summary are trained on 35% of the data as a means to substantiate the generality of the overall model.

Most important features

The main goal of this project was to determine the most important features to the dataset when predicting customer churn. To determine the most important features, we used the feature importance functionality within the `plot_importance` function within the XGBoost library..

Based on this analysis, the four features that most contributed to customer churn, using the upsampled dataset and the turned XGBoost model were: `tenure`, `MonthlyCharges`, `hasInternetService`, and `hasContract`. In passing, it is interesting to note that if we increase the size of the training dataset to 67%, the important features as noted by the XGBoost model are different in that the two most `TotalCharges`, `MonthlyCharges` and `tenure` are the top three features that can predict churn. We did not explore this observation further in this project.

Given that the initial feature set contained 17 features, we also evaluated the performance of the model when only using these four features to predict customer churn. Based on the results, the Recall score (Class 1) for the final model and the upsampled dataset was both 0.89 when using the reduced feature set (4 features) or the entire feature set (17 features).

Final model

Based on all of the model tuning and validation efforts, the final model that we selected was the XGBoost model that was instantiated as noted below. The performance of the final model was noted to be 0.89 using the upsampled dataset that contained 3,621 training samples and 6,727 testing samples.

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bytree=1, gamma=0, learning_rate=0.01, 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=42,
reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None, silent=True,
subsample=1)
```

Overall, based on the results observed within this project, this model seems to be robust enough for the problem since small perturbations to the hyperparameters do not change the performance of the model. In fact, the most important attribute to the performance of the final mode was the level of class imbalance between the major and minor classes. Specifically, we saw that the model performs best when the minor class was randomly upsampled. In addition, we did not see a decrease in performance when this model was used on the full feature space (containing 17 features) or the reduced feature space containing only five features.

In addition, we feel that this model can be trusted to generalize well on unseen data since we did not see a decrease in the performance of the model when training the data with less data points.

For these reasons, we feel that the final model presented is the best solution for predicting customer churn from the telco dataset.

e. Justification

Earlier, we discussed the results from the turning and validation efforts of this project. From this project, we have selected a turned XGBoost model as the final model to predict customer churn.

The final model seems to be the best model since it has a both high recall and precision scores and was selected after various tuning and validation efforts. As a result, we considered the final model significant enough to solve the problem of predicting customer churn from a set of customer data as well as identifying the most important features needed to predict customer churn and have thoroughly analyzed and discussed the final model in the previous section.

As noted earlier, the benchmark that was established for this project was defined as the null accraucy of the model, or the percent of the most common class. For the telco dataset, this is 74% (for Class 0). While a benchmark was defined for this project, it is ill conceived to use this benchmark to performance of the model that was built since this project involves identifying all instances of a rare event, defined in this project, as being instances of customer churn (Class 1). In light of this, the final model does outperform the benchmark score of 26% since there is a Recall score (Class 1) of 95% and is stronger than the benchmark noted earlier.

4. Conclusion

a. Freeform visualization

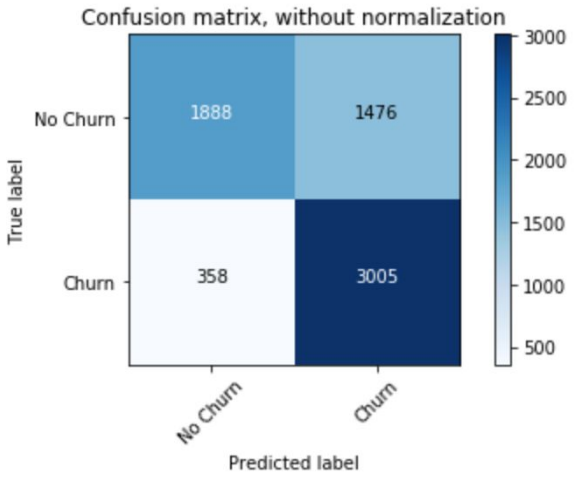
For the freeform visualization, we decided it would be best to present two confusion matrices from two different model outcomes. The first is from the final model (i.e., tuned XGBoost model trained on the upsampled dataset with a Recall (Class =1 of 0.89)) and a second model that had higher Recall (Class 1) than the final model, a tuned SVC model that was trained on the downsampled data set with a Recall (Class 1) of 0.99.

I found these two results to be interesting since both models have a high Recall rate for Class 1, which we did use to determine the success of our model, however, the tuned SVC model shown in Panel B did have a False positive rate that was nearly twice as high as the final model shown in Panel A.

The false positive rate is defined as being the ratio of the total amount of false positives to the total number of true negatives. Using the values in the confusion matrices below, that is calculated to be 43% (1476 / 1888 + 1476) for the tuned XGBoost model and 90% (1100 / 1100 115) for the tuned SVC model. Within the context of this problem, a false positive is defined as being customers who have not churned but have been classified as being churned by the model.

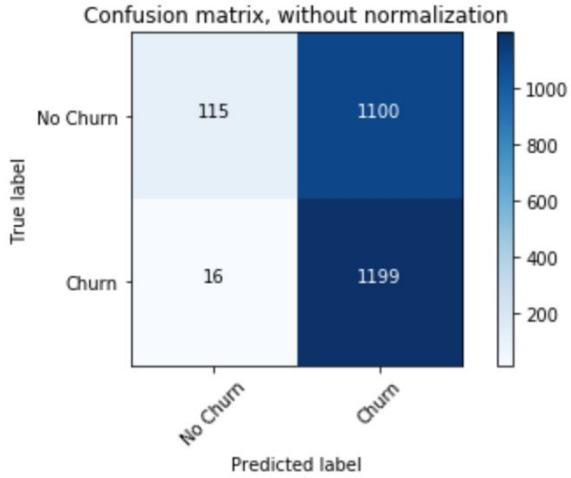
This one finding illustrates the importance of looking at the results as a whole when formulating a final conclusion. If we did not consider the false positive rate of the Tuned SVC model, is would have been possible to select that model as the final model since that model did afford the highest recall. Knowing the tradeoffs between the recall and false positive rate is something that is determined using the context of the problem.

Panel A: Tuned XGBoost/Upsample



Recall (Churn Class) = 0.89 (3005 / 3005 + 358)
Precision (No Churn Class) = 0.84 (1888 / 1888 + 358)

Panel B: Tuned SVC/Downsample



Recall (Churn Class) = 0.99 (1199 / 1199 + 16)
Precision (No Churn Class) = 0.88 (115 / 115 + 16)

b. Reflection

In all, I found this project to be challenging and rewarding since I was not only able to develop the high level proposal but also complete the technical side of the project scope.

The two aspects of this project that I found interesting was the handling of the class imbalance (using random upsampling and downsampling) as well as the hyperparameter tuning of each model.

Class imbalance is a common occurrence within actual datasets, and from this project, it was clear that the models can observe a local maximum in terms of performance due to the class imbalance. Despite this, however, it is clear from the work presented herein that the performance of the model improves if the dataset is resampled to address the class imbalance. To address the class imbalance, I took it upon myself to research and implement resampling methods into the project. While there are a variety of other methods to resample an imbalanced dataset (NearMiss 1, 2, or 3, or SMOTE), I decided that the best approach was to use basic upsampling and downsampling. Again, from the results observed, the best Recall was obtained with the upsampled dataset, suggesting the importance of having a balanced dataset.

Being able to turn a machine learning model ensures that the model is able to perform its best on the training dataset. The second aspect of this project that I found to be interesting was the hyperparameter tuning of the models. While all of this was done using the grid search function within sk-learn, I did find it fascinating that the the function cycled through all permutations of the hyperparameters provided and selected the one that afforded the highest Recall.

One of the challenging aspects of the project included keeping track of all of the Recall and precision scores since there was 18 total models that developed . Another challenging aspect of this project came up when resampling the dataset.

Overall, the final model and solution (a turned XGBoost model) does fit my expectations for the problem at hand. Previous Google searches have shown that the XGBoost model is one of the hottest machine learning models currently out there and when starting this project, I did expect that this model would be the best model of the group for this project. One of the specific reasons that I feel that this model did the best was that the XGBoost model was able to have a high Recall and Precisions scores. On a high level, this model can be used in a general setting to solve this type of binary classification problems.

c. Improvement

There are a few improvements that can be made to the final model. These include [1] collecting more data in the minor class, [2] assess performance using other resampling methods, [3] implementation of a recurrent neural network (“RNN”).

For any classification problem that seeks to find a “rare event,” or an uncommon event like credit card fraud, incidents of cancer or SPAM email messages, one common problem resides in the balance between the all of the example classes in the dataset that is used. For this dataset, only 26% of the data points used were of the minority class, or the class where customer churn was predicted. One important that I would propose is to collect more data points

of the customer churn class as to balance the classes in the overall dataset naturally as opposed to synthetically.

Another improvement that I would propose for this project is to assess the performance of the model using other sampling methods including Synthetic Minority Over-sampling TEchnique (SMOTE) and Near-Miss 1, 2, or 3. In some cases, collecting more data is not possible. In these cases, it becomes necessary to resample the dataset in order to avoid overfitting the major class. Since we only employed random oversampling and undersampling, I would propose to compare other methods of sampling to the final model that was selected in this project.

Finally, a last improvement that I would make to this project would be to implement a recurrent neural network as one of the classifiers. While this course did touch upon neural networks and transfer learning, I feel that I would need more time with the material in order to full yo impoement a RNN on my own for this project.

As it relates to performance of the final model, even though the the use SMOTE, Near Miss 1, 2 or 3 and an RNN are sophisticated methods, I would expect that the Recall and Precision of the final model using these methods to be the same as the final model given the new optimal performance of the XGBoost model on it's own. However, out of curiosity, I would still try and implement these changes to confirm my assumptions.

###