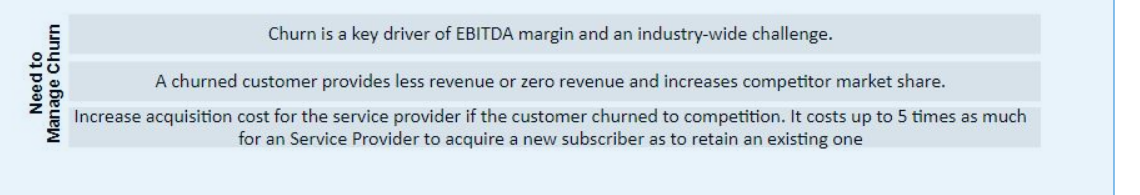
**What is Churn Rate?**

The churn rate, also known as the rate of attrition or customer churn, is the rate at which customers stop doing business with an entity. It is most commonly expressed as the percentage of service subscribers who discontinue their subscriptions within a given time period. It is also the rate at which employees leave their jobs within a certain period. For a company to expand its clientele, its growth rate (measured by the number of new customers) must exceed its churn rate.

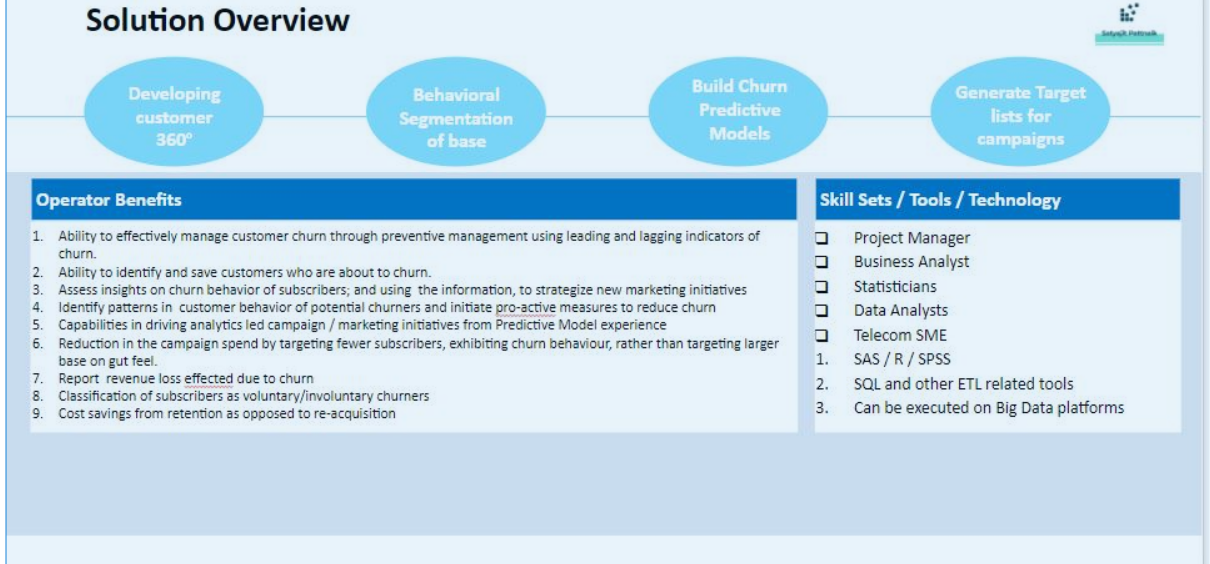
A high churn rate could adversely affect profits and impede growth. Churn rate is an important factor in the telecommunications industry. In most areas, many of these companies compete, making it easy for people to transfer from one provider to another.

The churn rate not only includes when customers switch carriers but also includes when customers terminate service without switching. This measurement is most valuable in subscriber-based businesses in which subscription fees comprise most of the revenues.

Need to manage Churn



Solution Overview



Aim of the Project:

In this project we take a dataset collected from a telecommunications company conatining information on 7000 customers and whether they terminated their service (Churn) or not. The information includes dmeographic data like age and gender and other relavent data like cost of plan, length of contract etc. The aim is to use the dataset to train a classification model that after training will be able to give us accurate predictions (based on a sample customer’s informtation) on the likelihood of the customer to ‘Churn’

Exploratory Data Analysis

The first step of any project is exploring the dataset. We first look at the basic information regarding the dataset like no. of rows and columns, distribution of numeric variables and so on. We then look for the presence of missing values and any instance of wrong formatting. Since there were only 11 missing values for TotalCharges column, I decided to delete them as it is a very small number. After that we went for visualizations on the various variables (performed both univariate and bivariate analysis of the relavent columns). We also looked at the correlation matrix and generated a seaborn heatmap to visualize the correlations of the variables with each other. Lastly, for two of the most significant variables, monthly charges,total charges and churn, we decided to draw up a kernal desnsity estimation (KDE) plot. KDE Plot described as Kernel Density Estimate is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at different values in a continuous variable. We can also plot a single graph for multiple samples which helps in more efficient data visualization which we did for both churn and no churn.

Data pre-processing and cleaning

The next step is to get the dataset ready for the purpose of training our classifier model. We divided customers into bins based on tenure e.g. for tenure < 12 months: assign a tenure group if 1-12, for tenure between 1 to 2 Yrs, tenure group of 13-24; so on. This is done primarily primarily for the sake of simplicity and visualization but because this has a smoothing effect on the input data and may also reduce the chances of overfitting

There were a number of categorical variables in the dataset (No/Yes). Computers can only process information represented with numbers. That is reason why we cannot just give a machine learning models categories or strings as input. So we convert these into dummy variables that will consist of zeros and ones only. The number of dummy variables we must create is equal to k-1 where k is the number of different values that the categorical variable can take on.

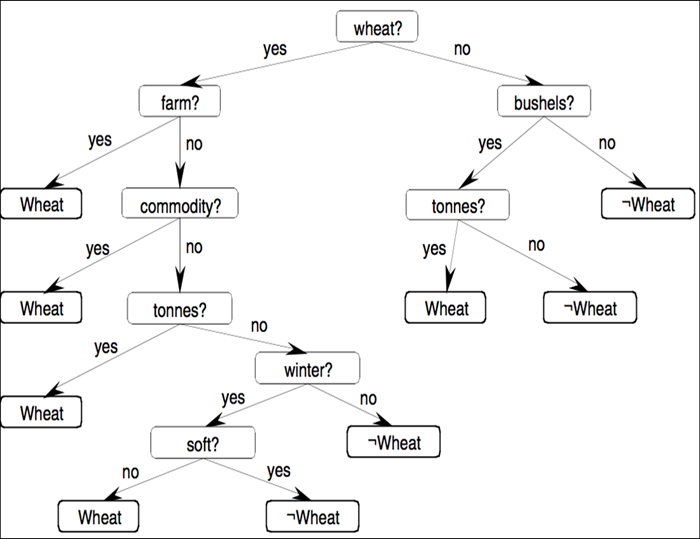
Lastly we dropped some unnecessary columns that were not relavent to churn rate in any way and would only increase the complexity of our model. Now our dataset is ready for our model to be trained.

Model Building and Evaluation

Since we have already had our data cleaned and pre-processed, we can jump straight into the model building part. First task is to divide the dataset into a training and test set. We will train our model using the training set (80%) and evaluate the performance using the test set (20%). We use sklearn train\_test\_split for this purpose.

Basics of Decision Tree Classifier:

Decision tree classifiers work like flowcharts. Each node of a decision tree represents a decision point that splits into two leaf nodes. Each of these nodes represents the outcome of the decision and each of the decisions can also turn into decision nodes. Eventually, the different decisions will lead to a final classification.



We first build a decisiontree classifier model using the following hyperparamters.

* criterion = "gini"

The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “log\_loss” and “entropy”

* random\_state = 100

Controls the randomness of the estimator. Practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.

* max\_depth=6

The name of hyperparameter max\_depth is suggested the maximum depth that we allow the tree to grow to. The deeper you allow, the more complex our model will become. If we increase the max\_depth value, training error will always go down but a very high value would lead to overfitting of the model

* min\_samples\_leaf=80

We know that a leaf node is a node without any children, so we cannot split a leaf node any further, so min\_samples\_leaf is the minimum number of samples that we can specify to term a given node as a leaf node so that we do not want to split it further.

Here we start off with 10,000 samples, and we reach a node wherein we just have 80 samples; there is no point in splitting further as you would tend to overfit the training data that you have, so by using this hyperparameter smartly, we can also avoid overfitting

Need for resampling

The accuracy of the previous model was very low (around 71%) when tested upon the test\_set. This was mainly because it is an imbalanced dataset i.e. the number of records that churned was significantly less than the no of customers that did not churn (27-73). Since the class distibution is not balanced, the machine learning algorithm performed poorly and at most times simply predicted the majority class.

To fix this problem, we will use oversampling by using SMOT-EEN. In simple words, oversampling is increasing the number of minority class samples. However, it’s not just about replicating samples of the minority class.We will get the resampled training and test sets and then use them to train and evaluate our model. After oversampling, we get much better results with an accuracy of above 90%. We also use sklearn.metrics to get the classification report consisting of precision, recall, support and f1 score.

Random Forest Classifier

We also use a random forest classifier model to check if it performs better on the given dataset. Random forest algorithm contains a large number of decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. So we expect a better performing model as compared to the decision tree classifier.

We build a Random forest classifier using the following hyperparameters:

* n\_estimators=100

We know that a random forest is nothing but a group of many decision trees, the n\_estimator parameter controls the number of trees inside the classifier. We may think that using many trees to fit a model will help us to get a more generalized result, but this is not always the case. However, it will not cause any overfitting but can certainly increase the time complexity of the model.

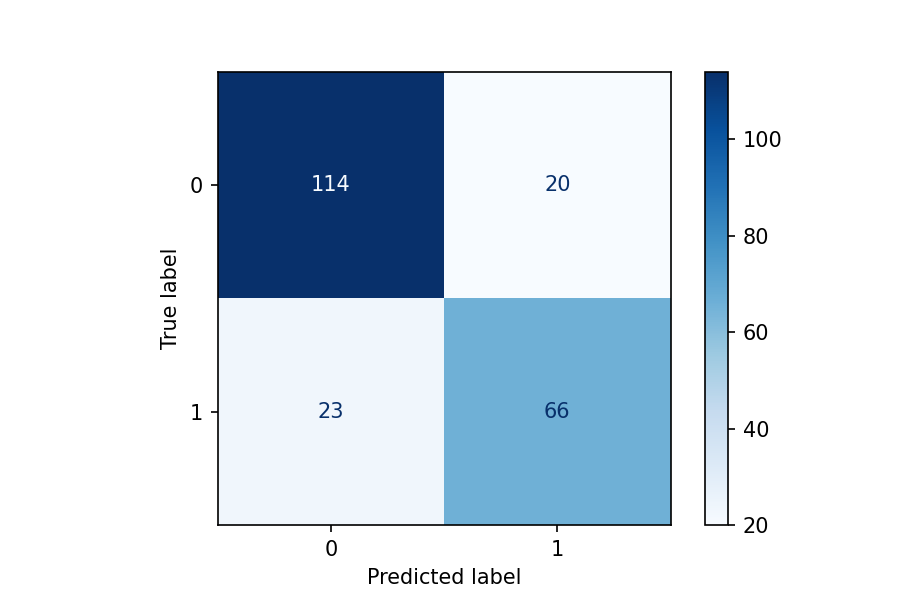
* criterion='gini'
* random\_state = 100
* max\_depth=6
* min\_samples\_leaf=8

Upon evaluation we get a slightly higher accuracy aroun 93% as compared to the decision tree classifier. Hence we will be choosing the RF classifier as our model to deploy.

Evaluation parameters

1)Confusion matrix:

A confusion matrix, despite its name, is fairly straightforward. It shows us how the predictions of a model stack up against the true and correct values, also known as the ground-truth.



There are four key takeaways from this, one for each quadrant:

* True Negatives (TN) are the number of predictions where the predicted label was 0 and the ground-truth label was also 0. This can be found in the top left quadrant. Note: negative in this context does not necessarily mean a negative value but rather one part of a binary representing true/false, on/off, alive/dead, etc.
* True Positives (TP) are similar to true negatives but for the opposing label (in this case, 1). This can be found in the bottom right quadrant.
* False Negatives (FN) are scenarios in which the model predicted a negative value when the ground-truth was actually positive. This can be found in the bottom left quadrant.
* False Positives (FP) are scenarios in which the model predicted a positive value when the ground-truth was actually negative. This can be found in the top right quadrant.

2)Metrics

* Accuracy: The most intuitive of the metrics, accuracy is essentially a measure of how many predictions of a model were correct — that is, aligned with the ground-truth.

While it may seem that nothing more is needed beyond this metric, relying solely on accuracy to evaluate a classification model is a mistake. Consider the following commonly referenced scenario used to highlight this issue: 100 patients are being tested for a disease that occurs in only 1% of people. A model which predicts that no one has the disease at all would technically have a 99% accuracy rate yet be completely useless for actually finding patients that are infected!

* Precision: Precision is defined as the ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples (either correctly or incorrectly).

Precision = True Positive/True Positive + False Positive

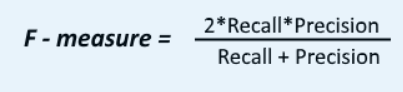
In our case it will be the ratio of the number of customers the model correctly predicted as ‘likely to churn’ divided by the total number of positives it predicted

* Recall: The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected.

Recall = True Positive/True Positive + False Negative

In our case it will be the ratio of the number of customers the model correctly predicted as ‘likely to churn’ divided by the total number of positives present in the test set.

* F1 Score: F1-Score or F-measure is an evaluation metric for a classification defined as the harmonic mean of precision and recall. It is a statistical measure of the accuracy of a test or model. Mathematically, it is expressed as follows,



Here, the value of F-measure(F1-score) reaches the best value at 1 and the worst value at 0. F1-score 1 represents the perfect accuracy and recall of the model.