Random Forest: Churn Prediction Model

Hello, my name is Molly Miraglia and I’m self teaching myself machine learning, specifically, Random Forest. I’m lucky I have a background in phylogenetics in which I experienced Random Forest in academic research journals/presentations and lab discussion. As I go further into my career, I want to use machine learning for exploratory data analysis and business analytics. I have a strong background in Python, R, SQL, and Tableau and it would be easy to apply these skills to using Random Forest/other machine learning techniques and present my findings to make data driven decisions that can ultimately benefit all stakeholders. I’m using an R Markdown, as I’m most familiar with it, but also I can incorporate code from Python and SQL. I’m writing my understanding and knowledge of Random Forest, not only for others to understand, but also for myself. I feel like once you can explain something in simple terms, you understand the material well.

I also wanted to focus on a Churn Prediction model using Random Forest, as this is very applicable the type of work I want to do. I am borrowing heavily from the work of Natassha Selvaraj, as she wrote a wonderful, detailed article for 365 DataScience that gave an in-depth tutorial for a Churn Prediction model in Python. I also am commenting code and writing out my process as I go along so I can further understand Random Forest. I am also thankful to the countless Youtube and Google searches I’ve watched for better understanding of the Random Forest concepts.

### TL;DR

I’m bragging about my background. Below is my own Random forest explanation from what I’ve learned online and a Churn Prediction model from an online tutorial.

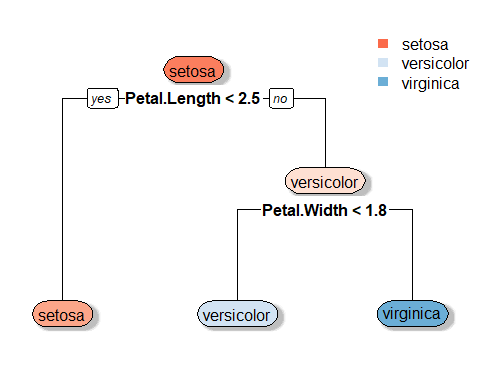
## Inspiration and Data Source:

* [365 DataScience Tutorial by Natassha Selvaraj](https://365datascience.com/tutorials/python-tutorials/how-to-build-a-customer-churn-prediction-model-in-python/)
* [Kaggle](https://www.kaggle.com/datasets/blastchar/telco-customer-churn)

## What do I know About Random Forest?

### Let’s start with Decision Trees

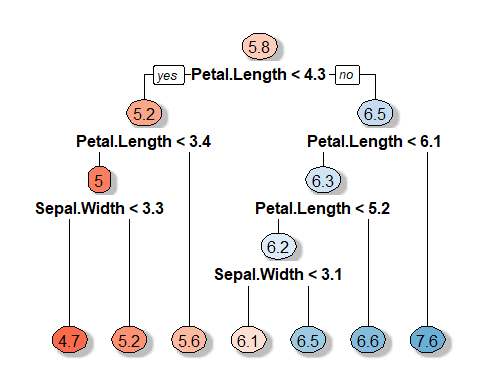
A random forest is composed of an ensemble of decision trees, which are models that look like a flow chart, but they actually help make predictions based on data. A decision tree makes an initial statement, known as the root, and then can branch off to different nodes based on a decision. There are two types of nodes, the internal nodes, which split the data into smaller subsets, and then there are leaf nodes, which is an outcome or prediction and does not further split the data. I’ll show an [example](https://www.spsanderson.com/steveondata/posts/2023-09-29/index.html) with the famous Iris (flower) data set.



The decision tree created shows the decisions leading to accurately naming the species of Iris based on petal measurements. If we selected an Iris from the data set used to create this decision tree, that had a petal length of 2.8 cm and a petal width of 1.9 cm, we would start at the root node of the decision tree where we would make our first decision about the petal length. Since the iris petal length is greater than 2.5 cm, we can move onto the right branch of the decision tree. This internal node then splits the data, and asks about petal width. Since the petal width is greater than 1.8, we can move to the right leaf node, which determines the species is ‘virginica’. If the petal length was less than 2.5, the data would not be needed to be split into further categorization, we could move to the left side of the branch, and get the outcome ‘setosa’.

The type of decision trees that random forest is based on is CART, which stands for Classification and Regression Trees. Classification means that the decision tree is splitting based on discrete or categorical data, and the leaf node will be a category. Categorical trees are usually found by finding the Gini Impurity or measure of Entropy. The plot above is considered a classification tree, as the leaf nodes are classifying what species the Iris is.

A regression tree means that each leaf node will be a numeric value, and is split based on a numeric measure. Regression trees are usually found using variance reduction or the mean squared error. The plot below is a regression tree which would predict the Sepal Length of a given Iris. If we selected an Iris from the data set used to create this decision tree, that had a petal length of 4.6 cm and sepal width of 2.9 cm, we would start at the root, which is 5.8 (average sepal length of entire data set). Since the given sepal length is NOT less than 4.3, we would move to the right branch of the tree. At the 6.5 internal node, we would move to left branch (4.6 < 6.1), and then at the 6.3 internal node, we would move to the left branch again (4.6 < 5.2). At the 6.2 internal node, we would then evaluate the sepal width (2.9 < 3.1), and move to the 6.1 leaf node.



Decision trees work well with the data used to create them (training data), but can be highly sensitive to any changes in training data. Notice in my examples, I said ‘selected an Iris from the data set used to create [the] decision tree’. If we took an Iris from outside this data set, it would not fit this model. The outside data set has different distributions and characteristics, and the decision tree could not generalize the new data correctly. There is also the problem of a decision tree overfitting the training data. That means the decision tree captured noise and specific patterns from the training data and those patterns and noise would skew a decision made with new data. Decision trees are a good way to form decisions if you’re working with one data set, however usually in research and other business intelligence roles, there will never be just one data set. This is why we use Random Forest!

In it’s simplest form, Random Forest creates an ensemble of decision trees. New data is ran through ALL of the decision trees created and the most popular outcome is the best prediction. Now, there are a bunch of steps in between, but the goal of random forest is to improve prediction accuracy and reduce overfitting.

Will continue writing (bootstrapping, etc.)

## Machine Learning Workflow

I am borrowing this [Machine Learning Workflow](https://towardsdatascience.com/random-forest-in-python-24d0893d51c0) roadmap.

1. State the question and determine required data
2. Acquire the data in an accessible format
3. Identify and correct missing data points/anomalies as required
4. Prepare the data for the machine learning model
5. Establish a baseline model that you aim to exceed
6. Train the model on the training data
7. Make predictions on the test data
8. Compare predictions to the known test set targets and calculate performance metrics
9. If performance is not satisfactory, adjust the model, acquire more data, or try a different modeling technique
10. Interpret model and report results visually and numerically

I don’t think we’ll get all the way to number 9, but I’m hopeful we can get to number 8!

## Let’s get started with the 365Data Science Tutorial

Following the Machine Learning Workflow, we must first state the question and determine required data. Luckily, we are working with given data from the tutorial. The main question in a Churn Prediction Model is usually, what is causing customers to Churn? We can look at the attributes of the data and depending on patterns in the data, we can try and accurately predict new things that will make a customer churn, and fix these issues.

We can now move onto number 2 in our machine learning worfklow and can read in the data set from [Kaggle](https://www.kaggle.com/datasets/blastchar/telco-customer-churn). This data set is customer information of a telephone company, including demographics, customer account information, services that the customer signed up for, and the Churn status. Churn means if a customer unsubscribes/stops using a business’s product. For this data set, the Churn status is if a customer stopped doing business *in the last month*. There are no dates for the churn, it’s just a Boolean of yes/no.

import pandas as pd  
from ydata\_profiling import ProfileReport  
  
df = pd.read\_csv('Customer\_Churn.csv')  
df.info()

## <class 'pandas.core.frame.DataFrame'>  
## RangeIndex: 7043 entries, 0 to 7042  
## Data columns (total 21 columns):  
## # Column Non-Null Count Dtype   
## --- ------ -------------- -----   
## 0 customerID 7043 non-null object   
## 1 gender 7043 non-null object   
## 2 SeniorCitizen 7043 non-null int64   
## 3 Partner 7043 non-null object   
## 4 Dependents 7043 non-null object   
## 5 tenure 7043 non-null int64   
## 6 PhoneService 7043 non-null object   
## 7 MultipleLines 7043 non-null object   
## 8 InternetService 7043 non-null object   
## 9 OnlineSecurity 7043 non-null object   
## 10 OnlineBackup 7043 non-null object   
## 11 DeviceProtection 7043 non-null object   
## 12 TechSupport 7043 non-null object   
## 13 StreamingTV 7043 non-null object   
## 14 StreamingMovies 7043 non-null object   
## 15 Contract 7043 non-null object   
## 16 PaperlessBilling 7043 non-null object   
## 17 PaymentMethod 7043 non-null object   
## 18 MonthlyCharges 7043 non-null float64  
## 19 TotalCharges 7043 non-null object   
## 20 Churn 7043 non-null object   
## dtypes: float64(1), int64(2), object(18)  
## memory usage: 1.1+ MB

Luckily I am using R Markdown and can incorporate more advanced plots (using ggplot!) than what the tutorial offers. The tutorial

churn\_df <- py$df

# Using ydata-profiling library to give a one line exploratory data analysis  
profile = ProfileReport(df, title = "Churn Profiling Report")  
profile.to\_notebook\_iframe()

## Summarize dataset: 0%| | 0/5 [00:00<?, ?it/s]Summarize dataset: 0%| | 0/26 [00:00<?, ?it/s, Describe variable:tenure]Summarize dataset: 4%|3 | 1/26 [00:00<00:01, 18.92it/s, Describe variable:SeniorCitizen]Summarize dataset: 8%|7 | 2/26 [00:00<00:01, 20.41it/s, Describe variable:gender] Summarize dataset: 12%|#1 | 3/26 [00:00<00:00, 24.90it/s, Describe variable:Dependents]Summarize dataset: 15%|#5 | 4/26 [00:00<00:00, 33.20it/s, Describe variable:Dependents]Summarize dataset: 15%|#5 | 4/26 [00:00<00:00, 33.20it/s, Describe variable:Partner] Summarize dataset: 19%|#9 | 5/26 [00:00<00:00, 33.20it/s, Describe variable:PhoneService]Summarize dataset: 23%|##3 | 6/26 [00:00<00:00, 33.20it/s, Describe variable:MultipleLines]Summarize dataset: 27%|##6 | 7/26 [00:00<00:00, 33.20it/s, Describe variable:InternetService]Summarize dataset: 31%|### | 8/26 [00:00<00:00, 33.20it/s, Describe variable:OnlineSecurity] Summarize dataset: 35%|###4 | 9/26 [00:00<00:00, 33.20it/s, Describe variable:TechSupport] Summarize dataset: 38%|###8 | 10/26 [00:00<00:00, 33.20it/s, Describe variable:Contract] Summarize dataset: 42%|####2 | 11/26 [00:00<00:00, 53.14it/s, Describe variable:Contract]Summarize dataset: 42%|####2 | 11/26 [00:00<00:00, 53.14it/s, Describe variable:customerID]Summarize dataset: 46%|####6 | 12/26 [00:00<00:00, 53.14it/s, Describe variable:StreamingTV]Summarize dataset: 50%|##### | 13/26 [00:00<00:00, 53.14it/s, Describe variable:PaperlessBilling]Summarize dataset: 54%|#####3 | 14/26 [00:00<00:00, 53.14it/s, Describe variable:PaymentMethod] Summarize dataset: 58%|#####7 | 15/26 [00:00<00:00, 53.14it/s, Describe variable:MonthlyCharges]Summarize dataset: 62%|######1 | 16/26 [00:00<00:00, 53.14it/s, Describe variable:DeviceProtection]Summarize dataset: 65%|######5 | 17/26 [00:00<00:00, 53.14it/s, Describe variable:OnlineBackup] Summarize dataset: 69%|######9 | 18/26 [00:00<00:00, 53.14it/s, Describe variable:Churn] Summarize dataset: 73%|#######3 | 19/26 [00:00<00:00, 53.14it/s, Describe variable:StreamingMovies]Summarize dataset: 77%|#######6 | 20/26 [00:00<00:00, 53.14it/s, Describe variable:TotalCharges] Summarize dataset: 81%|######## | 21/26 [00:00<00:00, 70.96it/s, Describe variable:TotalCharges]Summarize dataset: 81%|######## | 21/26 [00:00<00:00, 70.96it/s, Get variable types] Summarize dataset: 81%|########1 | 22/27 [00:00<00:00, 70.96it/s, Get dataframe statistics]Summarize dataset: 82%|########2 | 23/28 [00:00<00:00, 70.96it/s, Calculate auto correlation]Summarize dataset: 86%|########5 | 24/28 [00:00<00:00, 70.96it/s, Get scatter matrix] Summarize dataset: 75%|#######5 | 24/32 [00:00<00:00, 70.96it/s, scatter tenure, tenure]Summarize dataset: 78%|#######8 | 25/32 [00:00<00:00, 70.96it/s, scatter MonthlyCharges, tenure]Summarize dataset: 81%|########1 | 26/32 [00:00<00:00, 70.96it/s, scatter tenure, MonthlyCharges]Summarize dataset: 84%|########4 | 27/32 [00:00<00:00, 70.96it/s, scatter MonthlyCharges, MonthlyCharges]Summarize dataset: 82%|########2 | 28/34 [00:00<00:00, 70.96it/s, Missing diagram bar] Summarize dataset: 85%|########5 | 29/34 [00:01<00:00, 21.44it/s, Missing diagram bar]Summarize dataset: 85%|########5 | 29/34 [00:01<00:00, 21.44it/s, Missing diagram matrix]Summarize dataset: 88%|########8 | 30/34 [00:01<00:00, 21.44it/s, Take sample] Summarize dataset: 91%|#########1| 31/34 [00:01<00:00, 21.44it/s, Detecting duplicates]Summarize dataset: 94%|#########4| 32/34 [00:01<00:00, 21.44it/s, Get alerts] Summarize dataset: 97%|#########7| 33/34 [00:01<00:00, 21.44it/s, Get reproduction details]Summarize dataset: 100%|##########| 34/34 [00:01<00:00, 22.91it/s, Get reproduction details]Summarize dataset: 100%|##########| 34/34 [00:01<00:00, 22.91it/s, Completed] Summarize dataset: 100%|##########| 34/34 [00:01<00:00, 26.92it/s, Completed]  
## Generate report structure: 0%| | 0/1 [00:00<?, ?it/s]Generate report structure: 100%|##########| 1/1 [00:05<00:00, 5.33s/it]Generate report structure: 100%|##########| 1/1 [00:05<00:00, 5.33s/it]  
## Render HTML: 0%| | 0/1 [00:00<?, ?it/s]Render HTML: 100%|##########| 1/1 [00:00<00:00, 1.67it/s]Render HTML: 100%|##########| 1/1 [00:00<00:00, 1.67it/s]  
## <IPython.core.display.HTML object>