Hands-on: Predict Customer Churn

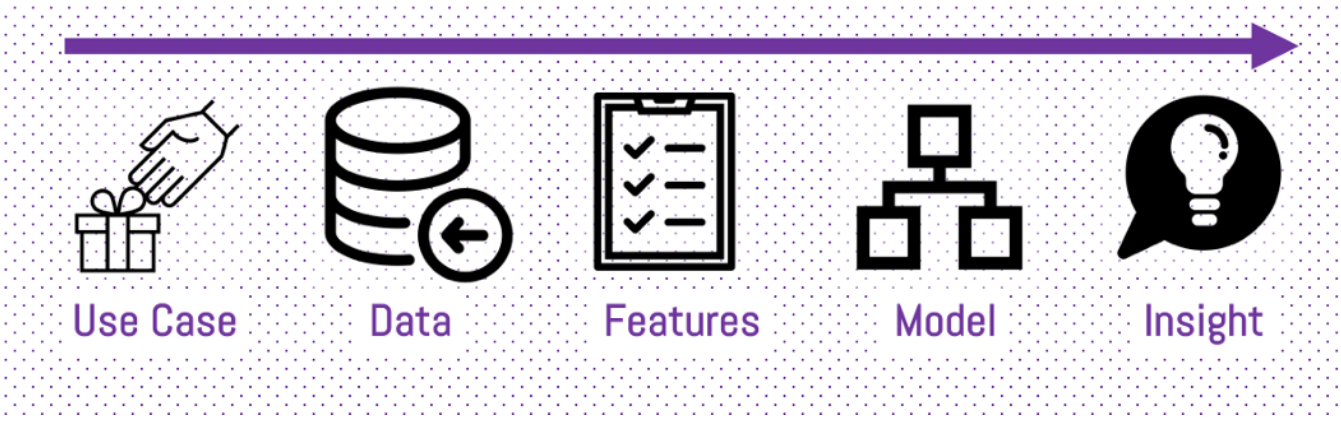
flo.tausend
Jan 16, 2019 · 10 min read ★

Long story short — in this article we want to get our hands dirty: building a model that identifies our beloved customers with the intention to leave us in the near future. We do this by implementing a predictive model with the help of python. Don't expect a perfect model, but expect something you can use in your own company / project today!

Before we start, let's briefly recap what churn actually is: Churn quantifies the number of customers who have unsubscribed or canceled their service contract. Customers turning their back to your service or product are no fun for any business. It is very expensive to win them back once lost, not even thinking that they will not do the best word to mouth marketing if unsatisfied. Learn all about the basics of customer churn in one of my previous articles. Now let's get started!

How Will We Predict Customer Churn?

The basic layer for predicting future customer churn is data from the past. We look at data from customers that already have churned (response) and their characteristics / behaviour (predictors) before the churn happened. By fitting a statistical model that relates the predictors to the response, we will try to predict the response for existing customers. This method belongs to the supervised learning category, just in case you needed one more buzzing expression. In practice we conduct the following steps to make these precise predictions:



High Level Process

1. Use Case / Business Case
   Step one is actually understanding the business or use case with the desired outcome. Only by understanding the final objective we can build a model that is actually of use. In our case the objective is reducing customer churn by identifying potential churn candidates beforehand, and take proactive actions to make them stay.

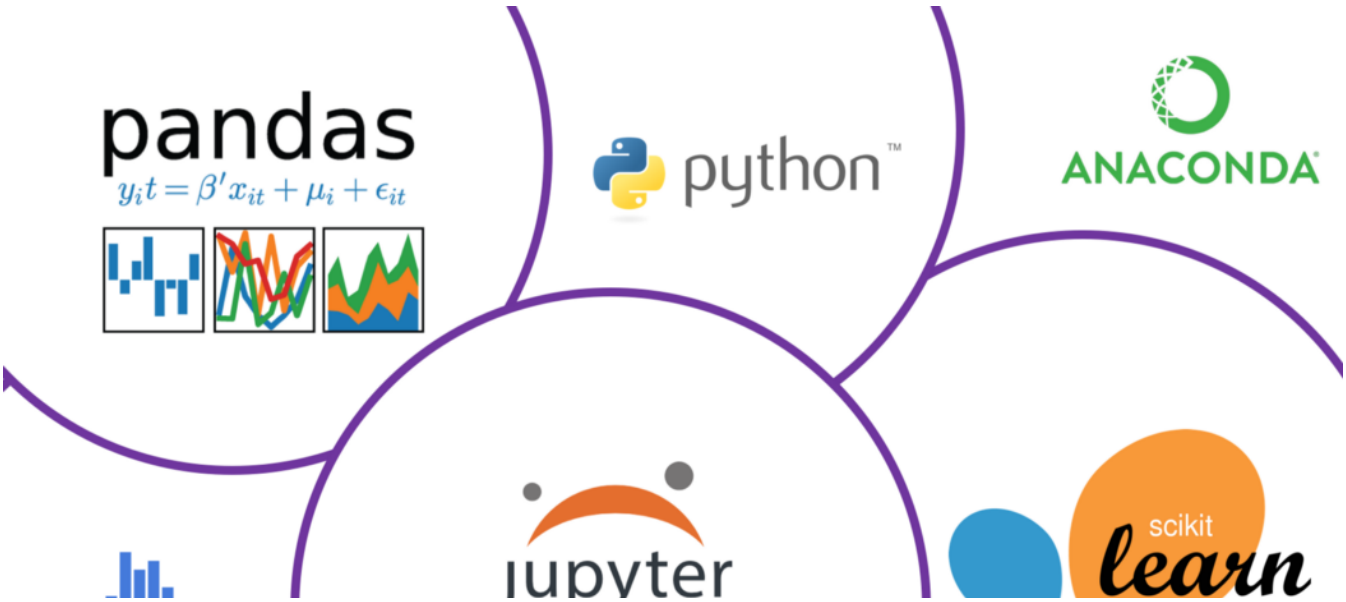
2. Data collection & cleaning
   With understanding the context it is possible to identify the right data sources, cleansing the data sets and preparing for feature selection or engineering. It sounds quite simple, but this is likely the hardest part. The predicting model is only as good as the data source. And especially Startups or small companies have often trouble to find enough data to train the model adequately.

3. Feature selection & engineering
   With the third step we decide which features we want to include in our model and prepare the cleansed data to be used for the machine learning algorithm to predict customer churn.

4. Modelling
   With the prepared data we are ready to feed our model. But to make good predictions, we firstly need to find the right model (selection) and secondly need to evaluate that the algorithm actually works. While this usually takes a few iterations, we will keep this quite simple and stop as soon as the results fit our needs.

5. Insights and Actions
   Last but not least we have to evaluate and interpret the outcomes. What does it mean and what actions can we derive from the results? Because predicting customer churn is only half of the part and many people forget that by just predicting, they can still leave. In our case we actually want to make them stop leaving.

Tools We Use



Tools to predict churn in python

To predict if a customer will churn or not, we are working with Python and it's amazing open source libraries. First of all we use Jupyter Notebook, that is an open-source application for live coding and it allows us to tell a story with the code. Furthermore we import Pandas, which puts our data in an easy-to-use structure for data analysis and data transformation. To make data exploration more graspable, we use plotly to visualise some of our insights. Finally with scikit-learn we will split our dataset and train our predictive model.

The Dataset

One of the most valuable assets a company has is data. As data is rarely shared publicly, we take an available dataset you can find on IBMs website as well as on other pages like Kaggle: Telcom Customer Churn Dataset. The raw dataset contains more than 7000 entries. All entries have several features and of course a column stating if the customer has churned or not.
To better understand the data we will first load it into pandas and explore it with the help of some very basic commands.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

from pylab import rcParams

%matplotlib inline

# Loading the CSV with pandas
data = pd.read_csv('..Customer Churn/Telco-Customer-Churn.csv')
```

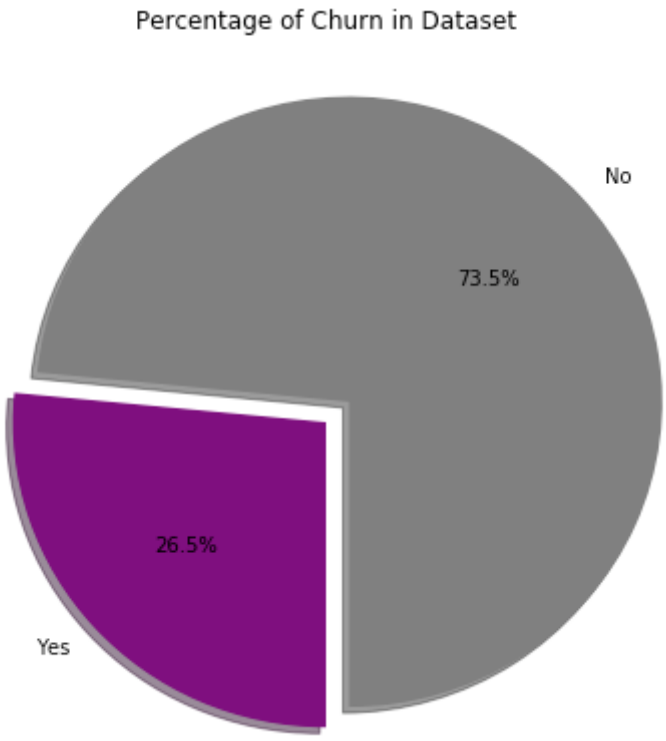
Exploration and Feature Selection

This section is kept quite short as you can learn more about general data exploration in better tutorials. Nevertheless to get initial insights and learn what story you can tell with the data, data exploration makes definitely sense. By using the python functions data.head(5) and "data.shape" we get a general overview of the dataset.

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSup
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

Glimpse of the dataset — overall 7043 rows with 21 columns

In detail we have a look at the target feature, the actual “Churn”. Therefore we plot it accordingly and see that 26,5% Of the total amount of customer churn. This is important to know so we have the same proportion of Churned Customers to Non-Churned Customers in our training data.




```
# Data to plot
sizes = data['Churn'].value_counts(sort = True)
colors = ["grey","purple"]
rcParams['figure.figsize'] = 5,5

# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=270,)

plt.title('Percentage of Churn in Dataset')
plt.show()
```

Data Preparation and Feature Engineering

Be aware that the better we prepare our data for the machine learning model, the better our prediction will be. We can have to most advanced algorithm, but if our training data sucks, our result will suck too. For that reason data scientists spend so much time on preparing the data. And as data preprocessing takes a lot of time, but is not the focus here, we will work through a few transformations exemplary.

1. Dropping irrelevant data
   There may be data included that is not needed to improve our results. Best is that to identify by logic thinking or by creating a correlation matrix. In this data set we have the customerID for example. As it does not influence our predicted outcome, we drop the column with the pandas “drop()” function.

```
data.drop(['customerID'], axis=1, inplace=True)
```

2. Missing Values
Furthermore it is important to handle missing data. The values can be identified by the “.isnull()” function in pandas for example. After identifying the null values it depends on each case if it makes sense to fill the missing value for example with the mean, median or the mode, or in case there is enough training data drop the entry completely. In the data set we are working with there is a very unusual case — there are no null values. Lucky us for today, but important to know usually we have to handle this.

3. Converting Numerical Features From Object
From our data exploration (in this case “data.dtypes()”) we can see that the the columns MonthlyCharges and TotalCharges are numbers, but actually in the object format. Why is this bad? Our machine learning model can only work with actual numeric data. Therefore with the “to_numeric” function we can change the format and prepare the data for our machine learning model.

```
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'])
```

4. Categorical data into numerical data
As we cannot calculate anything with string values, we have to convert these values

intro numeric ones. A simple example in the Telcom dataset is the gender. By using the Pandas function “get_dummies()” two columns will replace the gender column with “gender_Female” and “gender_Male”.

:eProtection	...	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male
No	...	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No	1	0
Yes	...	No	No	One year	No	Mailed check	56.95	1889.5	No	0	1
No	...	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes	0	1
Yes	...	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No	0	1
No	...	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes	1	0

Additionally to that we could use the “get_dummies()” function for all the categorical variables in the dataset. This is a powerful function, but it may be disturbing to have so many additional columns.

5. Splitting the dataset

First our model needs to be trained, second our model needs to be tested. Therefore it is best to have two different dataset. As for now we only have one, it is very common to split the data accordingly. X is the data with the independent variables, Y is the data with the dependent variable. The test size variable determines in which ratio the data will be split. It is quite common to do this in a 80 Training / 20 Test ratio.

```
data["Churn"] = data["Churn"].astype(int)

Y = data["Churn"].values
X = data.drop(labels = ["Churn"],axis = 1)

# Create Train & Test Data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=101)
```

Logistic Regression & Model Testing

Logistic Regression is one of the most used machine learning algorithm and mainly used when the dependent variable (here churn 1 or churn 0) is categorical. The independent variables in contrary can be categorical or numerical. Please note that of course it makes sense to understand the theory behind the model in detail, but in this case our goal is to make use of the predictions we won’t go through this in this article.

Step 1. Let's Import the model we want to use from sci-kit learn
Step 2. We make an instance of the Model
Step 3. Is training the model on the training data set and storing the information learned from the data

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
result = model.fit(X_train, y_train)
```

With the trained model we can now predict if a customer churned or not for our test dataset. The results are saved in "prediction_test" and afterwards the accuracy score is measured and printed.

```
from sklearn import metrics
prediction_test = model.predict(X_test)

# Print the prediction accuracy
print (metrics.accuracy_score(y_test, prediction_test))
```

```
0.8061611374407583
```

The score show us that in 80% of the cases our model predicted the right outcome for our binary classification problem. That's considered quite good for a first run, especially when we look which impact each variable has and if that makes sense. So with the final objective to reduce churn and take the right preventing actions in time, we want to know which independent variables have to most influence on our predicted outcome. Therefore we set the coefficients in our model to zero and look at the weights of each variable.

```
# To get the weights of all the variables
weights = pd.Series(model.coef_[0],
 index=X.columns.values)
weights.sort_values(ascending = False)
```

```
Fiber_optic                              0.498273
Contract_Month-to-month                  0.471609
```

```
Has_InternetService                          0.436689
SeniorCitizen                                0.279543
PaperlessBilling                             0.251657
StreamingTV                                  0.170260
MultipleLines                                0.142375
PaymentMethod_Electronic check               0.077876
MonthlyCharges                               0.013344
StreamingMovies                              0.005586
TotalCharges                                 0.000246
Partner                                     -0.002809
tenure                                      -0.054602
DSL                                         -0.061584
Dependents                                  -0.121068
DeviceProtection                            -0.139696
PaymentMethod_Bank transfer (automatic)     -0.162099
OnlineBackup                                -0.183691
PaymentMethod_Credit card (automatic)       -0.233960
PaymentMethod_Mailed check                  -0.253860
gender_Female                               -0.276228
Contract_One year                           -0.281264
gender_Male                                 -0.295816
OnlineSecurity                              -0.463323
TechSupport                                 -0.501160
Contract_Two year                           -0.762389
PhoneService                                -0.895864
dtype: float64
```

It can be observed that some variables have a positive relation to our predicted variable and some have a negative relation. A positive value has a positive impact on our predicted variable. A good example is “Contract_Month-to-month”: The positive relation to churn means that having this type of contract also increases the probability of a customer to churn. On the other hand that “Contract_Two year” is in a highly negative relation to the predicted variable, which means that customers with this type of contract are very unlikely to churn. But we can also see that some variables do not make sense in the first point. “Fiber_Optic” is on top position in terms of a positive impact on churn. While we would expect that this makes a customer stay, as it provides him with fast internet, our model says different. Here it is important to dig deeper and get some context for the data.

Yes We Did It — What’s Next?

Hug your customer and make him stay :)

One — Talk to your team.
We did not only find out which customers are likely to churn, but also which features have the most impact on a customer leaving. Therefore it is highly recommended to share these insights with your customer success team and adapt their focus. Only if the team knows where to put emphasis on, the team is able to to guide a customer to features that make him/her stick around longer. Speak openly, discuss options and make sure you understand the complete context. In many cases the customer success or support team is able to give additional qualitative insights that might underline your findings or totally tell a different story to your findings.

Two — Engage with the customers likely to churn.
Yes, there is the story that you should let sleeping dogs lie. But in the case of potential churn this is bullshit. Being quiet and hoping that your customer does not leave will 100% end up in a churn sooner or later. Instead don't be scared, go out and engage with your customers. There are many options but the best one is the most obvious one: Talk to them.

Look at their profile, identify characteristics and analyse past interactions with your product and then simply talk to them. Ask for feedback, communicate latest developments that might be from interest or educate them on new product features. Approach customers likely to churn, but make sure that you come up with relevant things that may fit their individual needs. It will create a feeling of being understood and bind them to you and your business.

****

Articles related to this one:

Hands-on: Setup Your Data Environment With Docker

Eliminating Churn is Growth Hacking 2.0

Misleading with Data & Statistics

****

Most of what I write about keeps me busy in our own Startup investory.io.
Looking forward to hearing your experience :)

****

Data Science | Predictive Analytics | Startup | Growth | Entrepreneurship