Machine Learning Engineer Nanodegree

Capstone Project

Peter Shoukry
December 31st, 2050

I. Definition

Project Overview

Running a profitable business is quite complicated. You need to juggle lots of balls at the sametime. The ability to do so comes down to the ability to make lots of small decisions everyday. Taking out the guesswork out of the decision process can make all the difference between successful and failed businesses.

Software as a service (SaaS), specially subscription based services, rely on the existing customer base to cover their expenses and to generally maintain a healthy cashflow. Customer churning, leaving or not renewing their subscription, can cost a business dearly in;

- Cost of aquiring new customers
- Bad publicity through word of mouth

ref: https://www.salesforce.com/blog/2013/08/customer-service-stats.html

Since understanding clients is an important part of running any business lots of similar research have been carried out, ex:

https://thesai.org/Downloads/Volume9No2/Paper_38-Machine_Learning_Techniques_for_Customer_Retention.pdf

https://www.h2o.ai/wp-content/uploads/2019/02/Case-Studies_PayPal.pdf

Because of the great difference between services types, user activity, etc... each research is unique and hence any machine learning approach would be higly dependent on the dataset. During the initial research I identified two different open data sets:

- https://www.kaggle.com/c/kkbox-churn-prediction-challenge/data (~8GB of SAAS user data)
- https://community.watsonanalytics.com/wp-content/uploads/2015/03/WA_Fn-UseC_-Telco-Customer-Churn.csv

Although the first dataset is interesting and would give lots of insights into customer churning in SaaS industry specifically, the size of the dataset makes tackling it a problem on it's own and will require lots of effort in data cleaning, preparation and extraction of a representative sample hence I opted for the second dataset which would help me focus more on the ML aspect of the problem.

Quick Statistics

- The data shape is (7043, 21)
- The number of churning users are 1869 (~26.5%) hence the dataset is unbalanced

Problem Statement

In specific words we are trying to predict if a user is going to stop using our services (churn) by studying his current usage behaviour.

The solution to this problem would be building a model to predict if a customer is going to churn or not. Given data about the user usage of the service, the model can classify into two classes: churning or not-churning. Because of the small size of the selected dataset it is more suitable to use classic ML techniques to deep learning.

Metrics

I will use F1 score as the metrics to evaluate the model because the dataset is unbalanced. The F1 (https://en.wikipedia.org/wiki/F1_score) is the harmonic mean of the precision and recall, where F1 = 1 is the best and 0 is the worst.

```
F1 = 2 * (precision * recall) / (precision + recall)
```

we will also use a simple benchmark model that assumes all users are going to keep using our service (not-churning). By comparing the two models' performance we will be able to identify how much business value we have gained since we have linked the difference between the models with the amount of users we can now try to gain back before they actually churn.

II. Analysis

Data Exploration

In this section, you will be expected to analyze the data you are using for the problem. This data can either be in the form of a dataset (or datasets), input data (or input files), or even an environment. The type of data should be thoroughly described and, if possible, have basic statistics and information presented (such as discussion of input features or defining characteristics about the input or environment). Any abnormalities or interesting qualities about the data that may need to be addressed have been identified (such as features that need to be transformed or the possibility of outliers). Questions to ask yourself when writing this section:

The data set is part of IBM watson community data sets. It consists of 7043 records each with 20 features plus one column to indicate if the user did churn or not. A sample of the records is provided below.

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

OnlineSecurity	No	Yes	Yes	Yes	No
OnlineBackup	Yes	No	Yes	No	No
DeviceProtection	No	Yes	No	Yes	No
TechSupport	No	No	No	Yes	No
StreamingTV	No	No	No	No	No
StreamingMovies	No	No	No	No	No
Contract	Month-to-month	One year	Month-to-month	One year	Month-to-month
PaperlessBilling	Yes	No	Yes	No	Yes
PaymentMethod	Electronic check	Mailed check	Mailed check	Bank transfer (automatic)	Electronic check
MonthlyCharges	29.85	56.95	53.85	42.3	70.7
TotalCharges	29.85	1889.5	108.15	1840.75	151.65
Churn	No	No	Yes	No	Yes

Note: The sample was transposed to fit the report width

Because of the size of the dataset classical ML techniques will be used to build the prediction model and we will not use deep learning.

The dataset has the following fields:

customerID: String - The customer ID and is unique for each user

- The customer ID is unique per user and will be removed

gender: String - Whether the customer is a male or female (Male, Female)

- will use label encoder to change to 1 or 0

SeniorCitizen: Number - Whether the customer is senior citizen or not (1, 0)
Partner: String - Whether the customer has a partner or not (Yes, No)

- will use label encoder to change to 1 or 0

Dependents: String - Whether the customer has dependents or not (Yes, No)

- will use label encoder to change to 1 or 0

tenure: Number - Number of months the customer has stayed with the company
PhoneService: String - Whether the customer has a phone service or not (Yes, No)

- will use label encoder to change to 1 or 0

MultipleLines: String - Whether the customer has multiple lines or not (Yes, No, No phone service)

- Categorial value, we will one hot encode it to work with the model

InternetService: String - Type of internet service (DSL, Fiber optic, No)

- Categorial value, we will one hot encode it to work with the model

OnlineSecurity: String - Whether the customer has online security or not (Yes, No, No internet service)

- Categorial value, we will one hot encode it to work with the model

OnlineBackup: String – Whether the customer has online backup or not (Yes, No, No internet service)

- Categorial value, we will one hot encode it to work with the model

DeviceProtection: String – Whether the customer has device protection or not (Yes, No, No internet service)

- Categorial value, we will one hot encode it to work with the model

TechSupport: String – Whether the customer has tech support or not (Yes, No, No internet service)

- Categorial value, we will one hot encode it to work with the model

StreamingTV: String – Whether the customer has streaming TV or not (Yes, No, No internet service)

- Categorial value, we will one hot encode it to work with the model

StreamingMovies: String – Whether the customer has streaming movies or not (Yes, No, No internet service)

- Categorial value, we will one hot encode it to work with the model

Contract: String – The contract term of the customer (Month-to-month, One year, Two year)

- Categorial value, we will one hot encode it to work with the model

PaperlessBilling: String – Whether the customer has paperless billing or not (Yes, No)

- will use label encoder to change to 1 or 0

PaymentMethod: String – The customer's payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))

- Categorial value, we will one hot encode it to work with the model

MonthlyCharges: String – The amount charged to the customer monthly
TotalCharges: String – The total amount charged to the customer

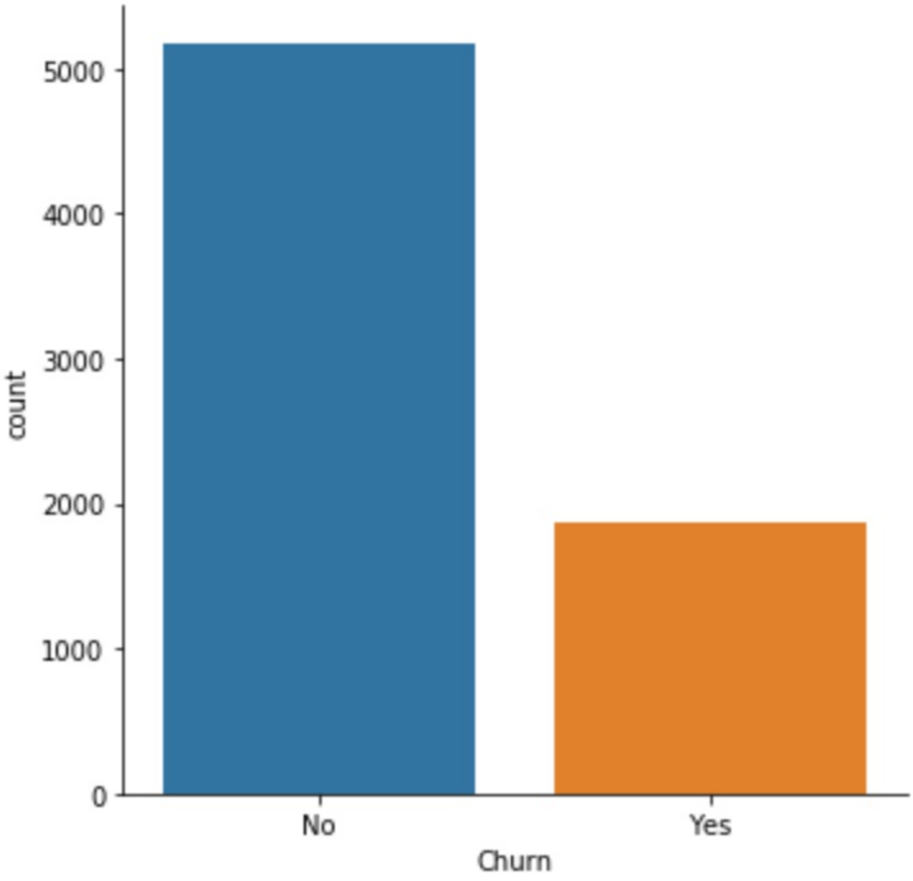
- The column is detected as string so we need to convert it to numeric
- Fill missing values with mean

Churn: String – Whether the customer churned or not (Yes, No)

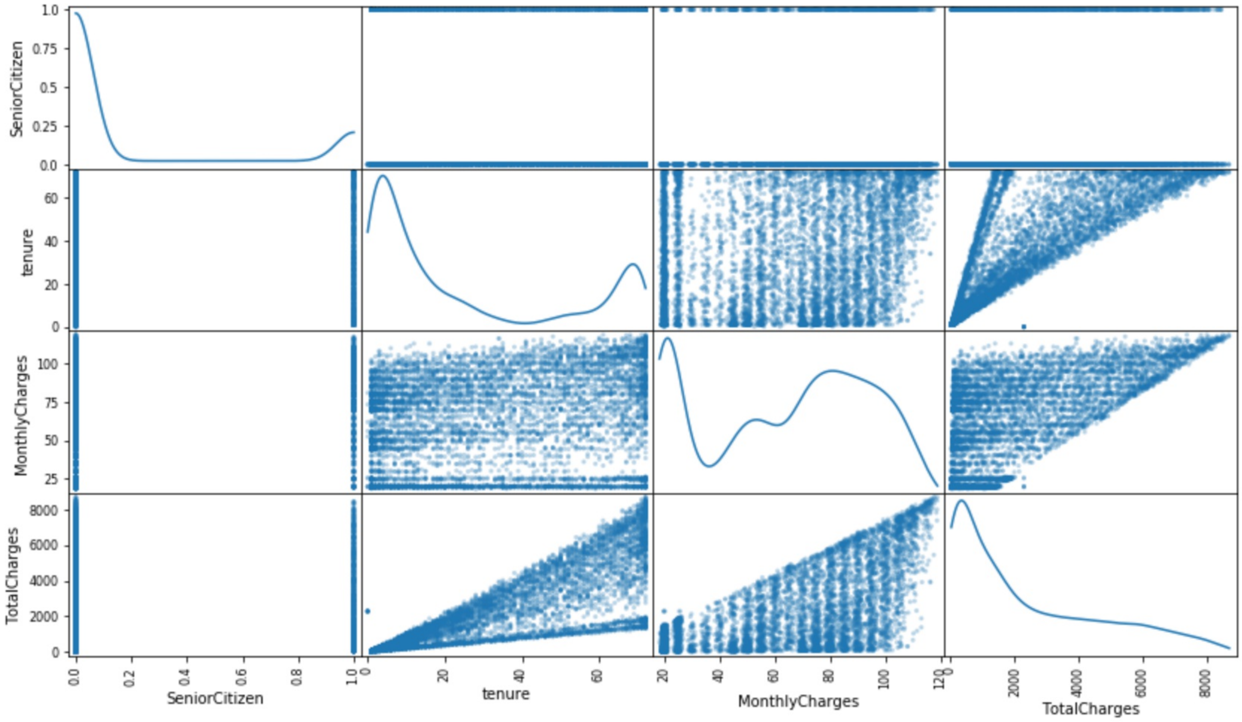
- will use label encoder to change to 1 or 0

Exploratory Visualization

The following plot shows the churn distribution in the data set and since the data is unbalanced we will choose F1 score as our metric.



Next we check the scatter matrix for each feature pair:



From the figure we notice:

1. The continues features are skewed and require transformation

2. The numerical features require scaling normalization
3. There is a correlation between tenure and total charges

Algorithms and Techniques

XGBoost is used as it dominates structured or tabular datasets on classification and regression predictive modeling problems.

The evidence is that it is the go-to algorithm for competition winners on the Kaggle competitive data science platform.

Strengths:

- they produce similar results as that of models with a lot of feature engineering in a fraction of the time and effort
- Robust to overfitting

Weaknesses:

- It has several key parameters that need to be set correctly to achieve the best classification results for any given problem

why?:

- Generally works well out of the box with most of problems

Benchmark

We will use a naive classifier as a benchmark the benchmark assumes all users will churn. This means the model will have no true or false negatives) as we aren't making any negative predictions (churn = 0).

In this case:
Precision = accuracy = TP / Total Predictions = 1869/7043
Recall = 1

$ F_{\beta} = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\left(\beta^2 \cdot precision \right) + recall} $
$ F_{\beta} = 0.3110748643520523 $

III. Methodology

Data Preprocessing

1. Transforming Skewed Continuous Features
2. Normalizing Numerical Features
3. label encoding yes / no values to 1 / 0
4. one-hot-encoding categorial values

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

- *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?*
- *Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?*
- *Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

- *Has an initial solution been found and clearly reported?*
- *Is the process of improvement clearly documented, such as what techniques were used?*
- *Are intermediate and final solutions clearly reported as the process is improved?*

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model's solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

- *Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?*
- *Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?*
- *Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?*
- *Can results found from the model be trusted?*

Justification

In this section, your model's final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

- *Are the final results found stronger than the benchmark result reported earlier?*
- *Have you thoroughly analyzed and discussed the final solution?*
- *Is the final solution significant enough to have solved the problem?*

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

- *Have you visualized a relevant or important quality about the problem, dataset, input data, or results?*
- *Is the visualization thoroughly analyzed and discussed?*
- *If a plot is provided, are the axes, title, and datum clearly defined?*

Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

- *Have you thoroughly summarized the entire process you used for this project?*
- *Were there any interesting aspects of the project?*
- *Were there any difficult aspects of the project?*
- *Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?*

Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

- *Are there further improvements that could be made on the algorithms or techniques you used in this project?*
- *Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?*
- *If you used your final solution as the new benchmark, do you think an even better solution exists?*

Before submitting, ask yourself. . .

- Does the project report you've written follow a well-organized structure similar to that of the project template?
- Is each section (particularly Analysis and Methodology) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
- Would the intended audience of your project be able to understand your analysis, methods, and results?
- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?

- Is the code that implements your solution easily readable and properly commented?
- Does the code execute without error and produce results similar to those reported?