Give Life: Predict Blood Donations

Build a binary classifier to predict if a blood donor is likely to donate again.

Project Description

"Blood is the most precious gift that anyone can give to another person — the gift of life." ~ World Health Organization

Forecasting blood supply is a serious and recurrent problem for blood collection managers: in January 2019, "Nationwide, the Red Cross saw 27,000 fewer blood donations over the holidays than they see at other times of the year."
Machine learning can be used to learn the patterns in the data to help to predict future blood donations and therefore save more lives.

In this Project, we will work with data collected from the donor database of Blood Transfusion Service Center in Hsin-Chu City in Taiwan. The center passes its blood transfusion service bus to one university in Hsin-Chu City to gather blood donated about every three months. The dataset, obtained from the UCI Machine Learning Repository, consists of a random sample of 748 donors.
Our task will be to predict if a blood donor will donate within a given time window. You will look at the full model-building process: from inspecting the dataset to using the tpot library to automate your Machine Learning pipeline.

Project Instructions
1.business understanding
Blood transfusion saves lives - from replacing lost blood during major surgery or a serious injury to treating various illnesses and blood disorders. Ensuring that there's enough blood in supply whenever needed is a serious challenge for the health professionals. According to WebMD, "about 5 million Americans need a blood transfusion every year".

Our dataset is from a mobile blood donation vehicle in Taiwan. The Blood Transfusion Service Center drives to different universities and collects blood as part of a blood drive. We want to predict whether or not a donor will give blood the next time the vehicle comes to campus.

2. ML process

- Load the dataset.

Lemme first talk about the RFM model. RFM stands for Recency, Frequency and Monetary Value and it is commonly used in marketing for identifying your best customers. In our case, our customers are blood donors.

Our dataset is builded with RFMTC, which is a variation of the RFM model. Below is a description of what each column means in our dataset:

- ➔ R (Recency - months since the last donation)
- ➔ F (Frequency - total number of donation)
- ➔ M (Monetary - total blood donated in c.c.)
- ➔ T (Time - months since the first donation)
- ➔ a binary variable representing whether he/she donated blood in March 2007 (1 stands for donating blood; 0 stands for not donating blood)

It looks like every column in our DataFrame has the numeric type, which is exactly what we want when building a machine learning model. Let's verify our hypothesis**.**

- Creating target column

By setting the inplace parameter of the rename() method to True, the transfusion DataFrame is changed in-place, i.e., the transfusion variable will now point to the updated DataFrame as you'll verify by printing the first 2 rows.

- Checking target incidence

By default, value_counts() method returns counts of unique values. By setting normalize=True, the value_counts() will return the relative frequencies of the unique values instead.

- Splitting transfusion into train and test datasets

Writing the code to split the data into the 4 datasets needed would require a lot of work. Instead, you will use the train_test_split() method in the scikit-learn library.

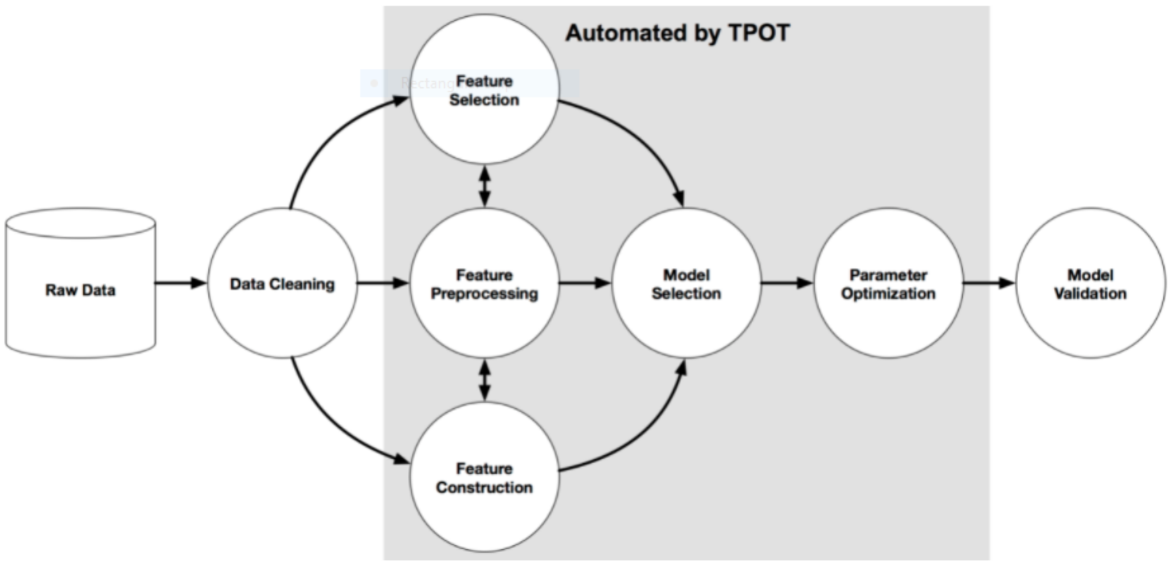
- Selecting model using TPOT

TPOT is a Python Automated Machine Learning tool that optimizes machine learning pipelines using genetic programming.

You will adapt the classification example from the TPOT's documentation. In particular, you will specify scoring='roc_auc' because this is the metric that you want to optimize for and add random_state=42 for reproducibility. You'll also use TPOT light configuration with only fast models and preprocessors.

The nice thing about TPOT is that it has the same API as scikit-learn, i.e., you first instantiate a model and then you train it, using the fit method.

Data pre-processing affects the model's performance, and tpot's fitted_pipeline_ attribute will allow you to see what pre-processing (if any) was done in the best pipeline.



Helpful links:

TPOT tutorial
Format strings using Python 3's f-strings tutorial

- checking the variance

TPOT picked **LogisticRegression** as the best model for our dataset with no pre-processing steps, giving us the AUC score of 0.7850. This is a great starting point. Let's see if we can make it better.

One of the assumptions for linear models is that the data and the features we are giving it are related in a linear fashion, or can be measured with a linear distance metric. If a feature in our dataset has a high variance that's orders of magnitude greater than the other features, this could impact the model's ability to learn from other features in the dataset.

pandas.DataFrame.var() method returns column-wise variance of a DataFrame, which makes comparing the variance across the features in X_train simple and straightforward.

- Log normalization

X_train and X_test must have the same structure. To keep your code "DRY" (Don't Repeat Yourself), you are using a for-loop to apply the same set of transformations to each of the DataFrames.

Normally, you'll do preprocessing before you split the data (it could be one of the steps in the machine learning pipeline). Here, you are testing various ideas with the goal to improve model performance, and therefore this approach is fine.

- Training the logistic regression model

The scikit-learn library has a consistent API when it comes to fitting a model:
Create an instance of a model you want to train.

Train it on your train datasets using the fit method.
You may recognise this pattern from when you trained the TPOT model. This is the beauty of the scikit-learn library: you can quickly try out different models with only a few code changes.

- conclusion

The demand for blood fluctuates throughout the year. As one prominent example, blood donations slow down during busy holiday seasons. An accurate forecast for the future supply of blood allows for an appropriate action to be taken ahead of time and therefore saving more lives.

We explored automatic model selection using TPOT and the AUC score we got was 0.7850. This is better than simply choosing 0 all the time (the target incidence suggests that such a model would have a 76% success rate). We then log normalized our training data and improved the AUC score by 0.5%. In the field of machine learning, even small improvements in accuracy can be important, depending on the purpose.

Another benefit of using a logistic regression model is that it is interpretable. We can analyze how much of the variance in the response variable (target) can be explained by other variables in our dataset.