MedTourEasy

PROJECT REPORT
Predicting Blood Donations

By Suba Shree V S
February 2021
Data Analyst Trainee

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The internship opportunity that I had with MedTourEasy was a great change for learning and understanding the intricacies of the subject of Data Visualizations in Data Analytics; and also, for personal as well as professional development. I am very obliged for having a chance to interact with so many professionals who guided me throughout the internship project and made it a great learning curve for me. I would also like to thank the team of MedTourEasy and my colleagues who made the working environment productive and very conducive.

Firstly, I express my deepest gratitude and special thanks to the Training & Developement Team of MedTourEasy who gave me an opportunity to carry out my internship at their esteemed organization. Also, I express my thanks to the team for making me understand the details of the Data Analytics profile and training me in the same so that I can carry out the project properly and with maximum client satisfaction and also for spearing his valuable time in spite of his busy schedule.

I would also like to thank the team of MedTourEasy and my colleagues who made the working environment productive and very conducive.

ABSTRACT

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.

Forecasting blood supply is a serious and recurrent problem for blood collection managers. In this Project, we will work with data collected from the donor database of Blood Transfusion Service Center. The dataset consists of random sample of 748 donors. Our task will be to predict if a blood donor will donate within a given time window.

ABOUT THE COMPANY

MedTourEasy, a global healthcare company, provides you the informational resources needed to evaluate your global options. It helps you find the right healthcare solution based on specific health needs, affordable care while meeting the quality standards that you expect to have in healthcare. MedTourEasy improves access to healthcare for people everywhere. It is an easy to use platform and service that helps patients to get medical second opinions and to schedule affordable, high-quality medical treatment abroad.

ABOUT THE PROJECT

"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. We are going to predict if a blood donor will donate within a given time window. We will look at the full model-building process: from inspecting the dataset to using the tpot library to automate your Machine Learning pipeline.

OBJECTIVES AND DELIVERABLES

- **Inspecting the transfusion data file:**

  One way to correct for high variance is to use log normalization.

- **Loading the blood donations data into memory:**

  We are working with a typical CSV file (i.e., the delimiter is , etc.)

- **Inspecting transfusion DataFrame:**

  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. RFMTC is a variation of the RFM model.

- **Creating target column for convenience**

- **Checking target incidence:**

  Target incidence is defined as the number of cases of each individual target value in a dataset.

- **Splitting transfusion into train and test datasets**

- **Selecting model using TPOT:**

  TPOT is a Python Automated Machine Learning tool that optimizes machine learning pipelines using genetic programming. TPOT will automatically explore hundreds of possible pipelines to find the best one for our dataset.

- **Checking the variance:**

  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. Correcting for high variance is called normalization. It is one of the possible transformations you do before training a model.

- **Log normalization:**

  One way to correct for high variance is to use log normalization.

- **Training the logistic regression model**

- **Conclusion:**

  Predicting if a blood donor willdonate blood within a given time window.

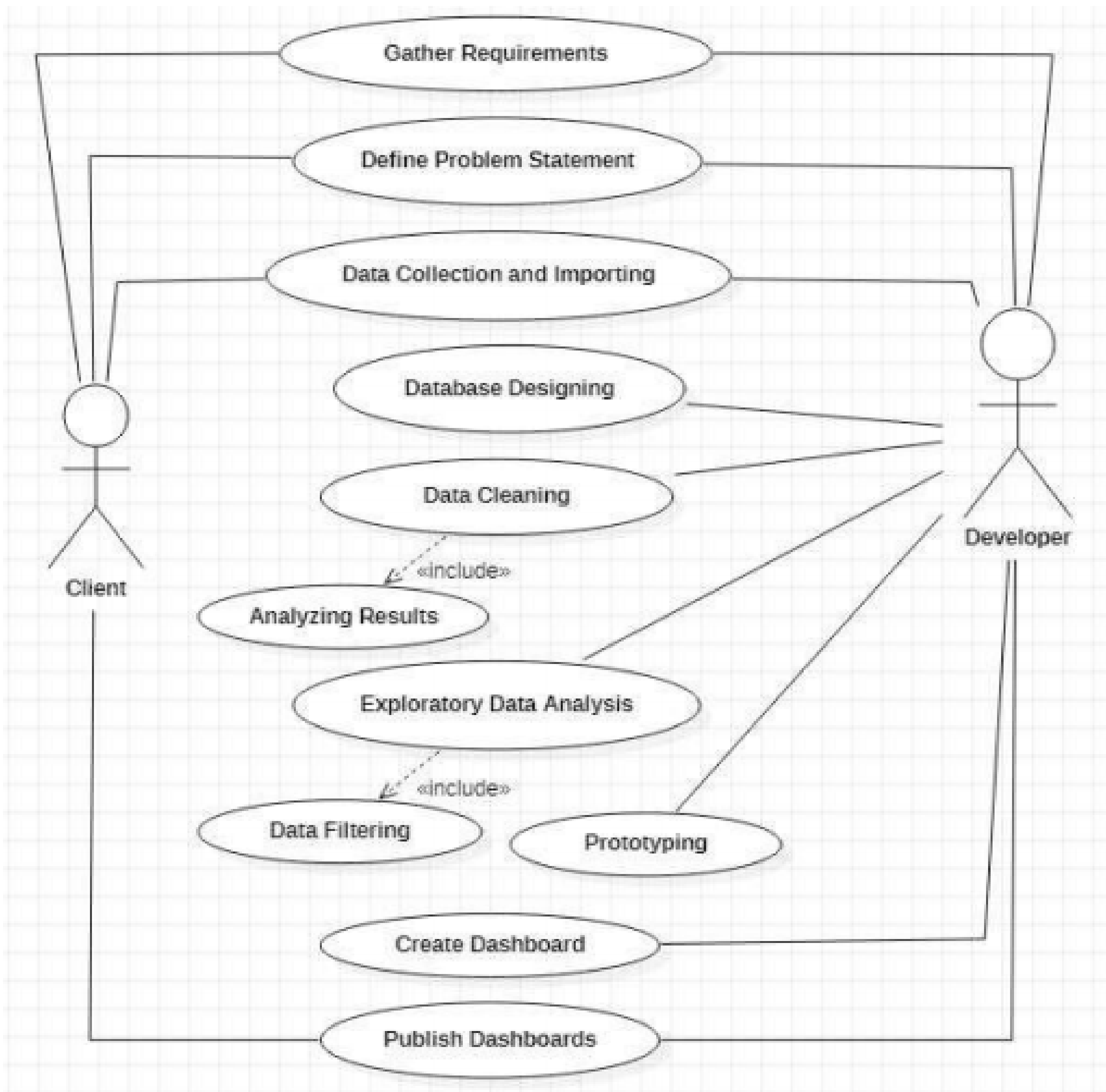
METHODOLOGY

FLOW OF THE PROJECT:

The project followed the following steps to accomplish the desired objectives and deliverables. Each step has been explained in detail in the following section.

USE CASE DIAGRAM:

The following figure shows the use case of the project. There are two main actors in the same: The Client and Developer. The developer will first gather requirements and define the problem statement then collecting the required data and importing it. Then the developer will design databases so as to identify various constraints and relations in the data. Next step is to clean the data to remove irregular values, blank values etc. Next, exploratory data analysis is conducted to filter the data according to the requirements of the project. Then a prototype of the dashboards is created using PowerBI to get a clear view of the visualizations to be developed. Finally, dashboard is developed and analyzed to publish the results to the client.



LANGUAGE USED & PLATFORM

Language Used: Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Characteristics of Python:

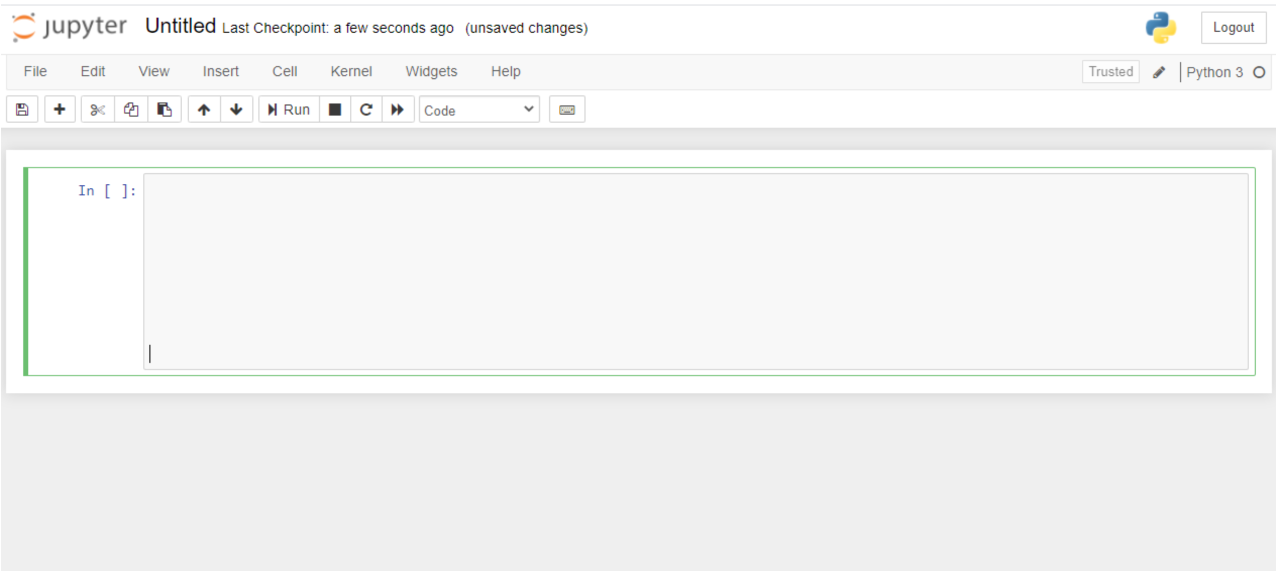
- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

IDE: Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

Uses of Jupyter Notebook:

- Data cleaning and transformation
- Numerical simulation
- Statistical modeling
- Data visualization
- Machine learning and much more.



IMPLEMENTATION

1. Inspect the file that contains the dataset

Print out the first 5 lines from datasets/transfusion.data using the head shell command.

2. Load the dataset

- Import the pandas library.
- Load the transfusion.data file from datasets/transfusion.data and assign it to the transfusion variable.
- Display the first rows of the DataFrame with the head() method to verify the file was loaded correctly.

3. Inspect the DataFrame's structure.

Print a concise summary of the transfusion DataFrame with the info() method. DataFrame's info() method prints some useful information about a DataFrame:

- index type
- column types
- non-null values
- memory usageincluding the index dtype and column dtypes, non-null values and memory usage.

4. Rename a column

- Rename whether he/she donated blood in March 2007 to target for brevity.

- Print the first 2 rows of the DataFrame with the head() method to verify the change was done correctly.

- 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.

5. Print target incidence

- Use value_counts() method on transfusion.target column to print target incidence proportions, setting normalize=True and rounding the output to 3 decimal places.
- 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.

6. Split the transfusion DataFrame into train and test datasets

- Import train_test_split from sklearn.model_selection module.
- Split transfusion into X_train, X_test, y_train and y_test datasets, stratifying on the target column.
- Print the first 2 rows of the X_train DataFrame with the head() method.
- 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.

7. Use the TPOT library to find the best machine learning pipeline

- Import TPOTClassifier from tpot and roc_auc_score from sklearn.metrics.

- Create an instance of TPOTClassifier and assign it to tpot variable.
- Print tpot_auc_score, rounding it to 4 decimal places.
- Print idx and transform in the for-loop to display the pipeline steps.
- 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.

8. Check the variance

- Print X_train's variance using var() method and round it to 3 decimal places.
- 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.

9. Correct for high variance

- Copy X_train and X_test into X_train_normed and X_test_normed respectively.
- Assign the column name (a string) that has the highest variance to col_to_normalize variable.
- For X_train and X_test DataFrames:
  - Log normalize col_to_normalize to add it to the DataFrame.
  - Drop col_to_normalize.

- Print X_train_normed variance using var() method and round it to 3 decimal places.
- 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 pre-processing before you split the data (it could be one of the steps in machine learning pipeline). Here, you are testing various ideas with the goal to improve model performance, and therefore this approach is fine.

10. Train the logestic regression model

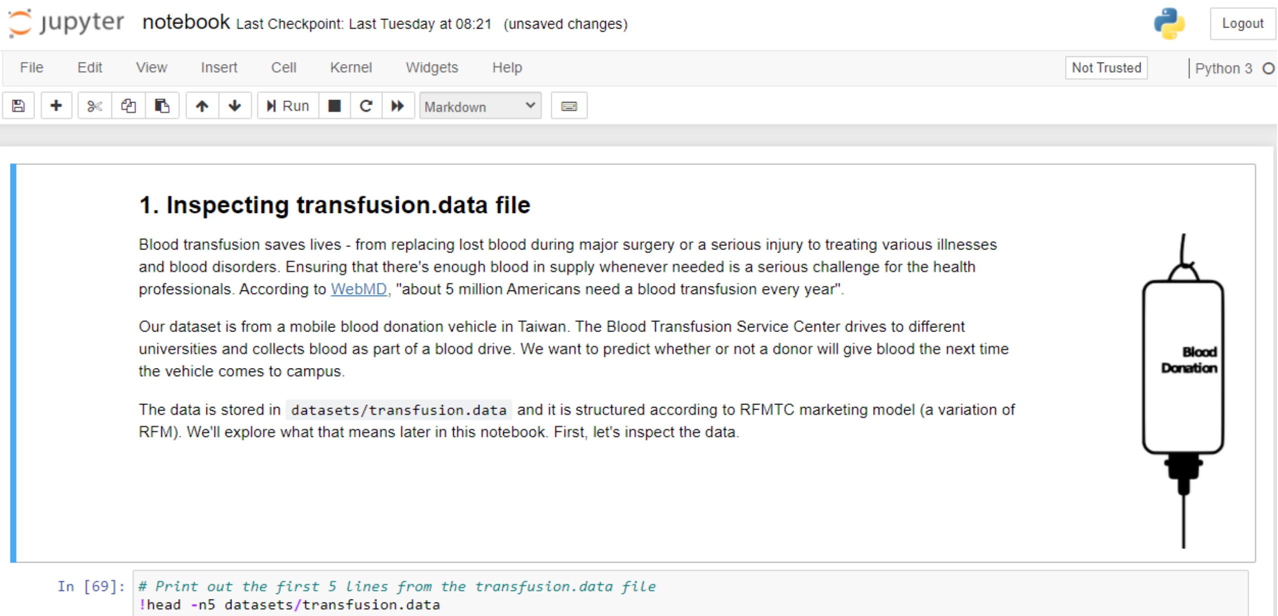
- Import linear_model from sklearn.
- Create an instance of linear_model.LogisticRegression and assign it to logreg variable.
- Train logreg model using the fit() method.
- Print logreg_auc_score.
- The scikit-learn library has a consistent API when it comes to fitting a model:
    - Create an instance 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 TPOT model. This is the beauty of the scikit-learn library: you can quickly try out different models with only a few code changes.

11. Sort your models based on their AUC score from highest to lowest.

- Import itemgetter from operator module.
- Sort the list of (model_name, model_score) pairs from highest to lowest using reverse=True parameter.

SAMPLE SCREENSHOTS AND OBSERVATIONS

1. Inspecting transfusion.data file



Observation:

```
Recency (months),Frequency (times),Monetary (c.c. blood),Time
(months),"whether he/she donated blood in March 2007"
2 ,50,12500,98 ,1
0 ,13,3250,28 ,1
1 ,16,4000,35 ,1
2 ,20,5000,45 ,1
```

2. Loading the blood donations data

jupyter notebook Last Checkpoint: Last Tuesday at 08:21 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Markdown

2. Loading the blood donations data

We now know that we are working with a typical CSV file (i.e., the delimiter is `,` , etc.). We proceed to loading the data into memory.

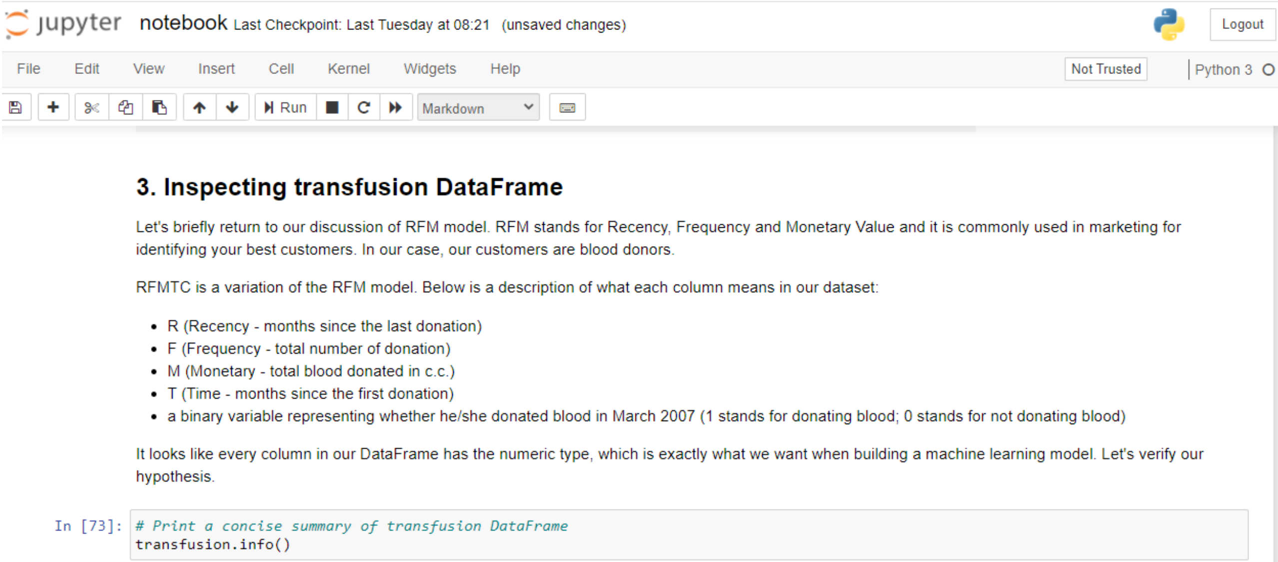
In [71]:

```
# Import pandas
import pandas as pd

# Read in dataset
transfusion = pd.read_csv('datasets/transfusion.data')

# Print out the first rows of our dataset
transfusion.head()
```

Observation:

	Recency (months)	Frequency (times)	Monetary (c.c. blood)	Time (months)	whether he/she donated blood in March 2007
0	2	50	12500	98	1
1	0	13	3250	28	1
2	1	16	4000	35	1
3	2	20	5000	45	1
4	1	24	6000	77	0

3. Inspecting transfusion DataFrame



3. Inspecting transfusion DataFrame

Let's briefly return to our discussion of 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.

RFMTC 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.

In [73]:
```
# Print a concise summary of transfusion DataFrame
transfusion.info()
```

Observation:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 748 entries, 0 to 747
Data columns (total 5 columns):
Recency (months)                              748 non-null int64
Frequency (times)                             748 non-null int64
Monetary (c.c. blood)                         748 non-null int64
Time (months)                                 748 non-null int64
whether he/she donated blood in March 2007    748 non-null int64
dtypes: int64(5)
memory usage: 29.3 KB
```

4. Checking target column

jupyter notebook Last Checkpoint: Last Tuesday at 08:21 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Markdown

```
whether he/she donated blood in March 2007    748 non-null int64
dtypes: int64(5)
memory usage: 29.3 KB
```

4. Creating target column

We are aiming to predict the value in `whether he/she donated blood in March 2007` column. Let's rename this it to `target` so that it's more convenient to work with.

In [75]:

```python
# Rename target column as 'target' for brevity
transfusion.rename(
    columns={'whether he/she donated blood in March 2007': 'target'},
    inplace=True
)

# Print out the first 2 rows
transfusion.head(2)
```

Observation:

	Recency (months)	Frequency (times)	Monetary (c.c. blood)	Time (months)	target
0	2	50	12500	98	1
1	0	13	3250	28	1

5. Checking target incidence

jupyter notebook Last Checkpoint: Last Tuesday at 08:21 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Markdown

1	0	13	3250	28	1

5. Checking target incidence

We want to predict whether or not the same donor will give blood the next time the vehicle comes to campus. The model for this is a binary classifier, meaning that there are only 2 possible outcomes:

- `0` - the donor will not give blood
- `1` - the donor will give blood

Target incidence is defined as the number of cases of each individual target value in a dataset. That is, how many 0s in the target column compared to how many 1s? Target incidence gives us an idea of how balanced (or imbalanced) is our dataset.

In [77]:

```python
# Print target incidence proportions, rounding output to 3 decimal places
transfusion.target.value_counts(normalize=True).round(3)
```

Observation:

```
0     0.762
1     0.238
Name: target, dtype: float64
```

6. Splitting transfusion into train and test datasets

jupyter notebook Last Checkpoint: Last Tuesday at 08:21 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted

Python 3

Run Markdown

6. Splitting transfusion into train and test datasets

We'll now use `train_test_split()` method to split `transfusion` DataFrame.

Target incidence informed us that in our dataset `0` s appear 76% of the time. We want to keep the same structure in train and test datasets, i.e., both datasets must have 0 target incidence of 76%. This is very easy to do using the `train_test_split()` method from the `scikit learn` library - all we need to do is specify the `stratify` parameter. In our case, we'll stratify on the `target` column.

In [79]:

```
# Import train_test_split method
from sklearn.model_selection import train_test_split

# Split transfusion DataFrame into
# X_train, X_test, y_train and y_test datasets,
# stratifying on the `target` column
X_train, X_test, y_train, y_test = train_test_split(
    transfusion.drop(columns='target'),
    transfusion.target,
    test_size=0.25,
    random_state=42,
    stratify=transfusion.target
)

# Print out the first 2 rows of X_train
X_train.head(2)
```

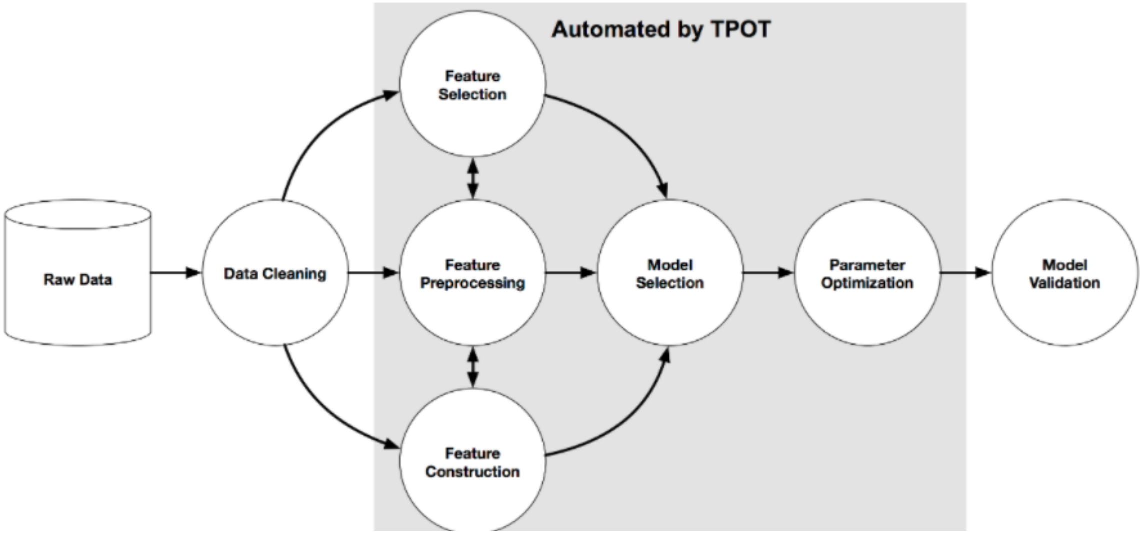
Observation:

	Recency (months)	Frequency (times)	Monetary (c.c. blood)	Time (months)
334	16	2	500	16
99	5	7	1750	26

7. Selecting model using TPOT

7. Selecting model using TPOT

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



jupyter notebook Last Checkpoint: Last Tuesday at 08:21 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Markdown

TPOT will automatically explore hundreds of possible pipelines to find the best one for our dataset. Note, the outcome of this search will be a scikit-learn pipeline, meaning it will include any pre-processing steps as well as the model.

We are using TPOT to help us zero in on one model that we can then explore and optimize further.

In [81]:

```python
# Import TPOTClassifier and roc_auc_score
from tpot import TPOTClassifier
from sklearn.metrics import roc_auc_score

# Instantiate TPOTClassifier
tpot = TPOTClassifier(
    generations=5,
    population_size=20,
    verbosity=2,
    scoring='roc_auc',
    random_state=42,
    disable_update_check=True,
    config_dict='TPOT light'
)
tpot.fit(X_train, y_train)

# AUC score for tpot model
tpot_auc_score = roc_auc_score(y_test, tpot.predict_proba(X_test)[:, 1])
print(f'\nAUC score: {tpot_auc_score:.4f}')

# Print best pipeline steps
print('\nBest pipeline steps:', end='\n')
for idx, (name, transform) in enumerate(tpot.fitted_pipeline_.steps, start=1):
    # Print idx and transform
    print(f'{idx}. {transform}')
```

Observation:

HBox(children=(HTML(value='Optimization Progress'), FloatProgress(value=0.0, max=120.0), HTML(value='')))

```
Generation 1 - Current best internal CV score: 0.7433977184592779
Generation 2 - Current best internal CV score: 0.7433977184592779
Generation 3 - Current best internal CV score: 0.7433977184592779
Generation 4 - Current best internal CV score: 0.7433977184592779
Generation 5 - Current best internal CV score: 0.7433977184592779

Best pipeline: LogisticRegression(input_matrix, C=0.5, dual=False, penalty=l2)

AUC score: 0.7850

Best pipeline steps:
1. LogisticRegression(C=0.5, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

8. Checking the variance

jupyter notebook Last Checkpoint: Last Tuesday at 08:21 (autosaved) Logou

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python 3

Run Markdown

8. 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.

Correcting for high variance is called normalization. It is one of the possible transformations you do before training a model. Let's check the variance to see if such transformation is needed.

In [83]:
```
# X_train's variance, rounding the output to 3 decimal places
X_train.var().round(3)
```

Observation:

```
Recency (months)              66.929
Frequency (times)             33.830
Monetary (c.c. blood)    2114363.700
Time (months)                611.147
dtype: float64
```

9. Log normalization

jupyter notebook Last Checkpoint: Last Tuesday at 08:21 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

9. Log normalization

`Monetary (c.c. blood)`'s variance is very high in comparison to any other column in the dataset. This means that, unless accounted for, this feature may get more weight by the model (i.e., be seen as more important) than any other feature.

One way to correct for high variance is to use log normalization.

In [85]:

```
# Import numpy
import numpy as np

# Copy X_train and X_test into X_train_normed and X_test_normed
X_train_normed, X_test_normed = X_train.copy(), X_test.copy()

# Specify which column to normalize
col_to_normalize = 'Monetary (c.c. blood)'

# Log normalization
for df_ in [X_train_normed, X_test_normed]:
    # Add log normalized column
    df_['monetary_log'] = np.log(df_[col_to_normalize])
    # Drop the original column
    df_.drop(columns='Monetary (c.c. blood)', inplace=True)

# Check the variance for X_train_normed
X_train_normed.var().round(3)
```

Observation:

```
Recency (months)        66.929
Frequency (times)       33.830
Time (months)          611.147
monetary_log             0.837
dtype: float64
```

10. Training the logistic regression model

jupyter notebook Last Checkpoint: Last Tuesday at 08:21 (autosaved)

Logou

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Markdown

10. Training the logistic regression model

The variance looks much better now. Notice that now `Time (months)` has the largest variance, but it's not the orders of magnitude higher than the rest of the variables, so we'll leave it as is.

We are now ready to train the logistic regression model.

In [87]:

```python
# Importing modules
from sklearn import linear_model

# Instantiate LogisticRegression
logreg = linear_model.LogisticRegression(
    solver='liblinear',
    random_state=42
)

# Train the model
logreg.fit(X_train_normed, y_train)

# AUC score for tpot model
logreg_auc_score = roc_auc_score(y_test, logreg.predict_proba(X_test_normed)[:, 1])
print(f'\nAUC score: {logreg_auc_score:.4f}')
```

Observation:

```
AUC score: 0.7891
```

11. Conclusion

jupyter notebook Last Checkpoint: Last Tuesday at 08:21 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Markdown

11. 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.

In this notebook, we explored automatic model selection using TPOT and 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 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 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.

In [89]:

```
# Importing itemgetter
from operator import itemgetter

# Sort models based on their AUC score from highest to lowest
sorted(
    [('tpot', tpot_auc_score), ('logreg', logreg_auc_score)],
    key=itemgetter(1),
    reverse=True
)
```

Observation:

```
[('logreg', 0.7890972663699937), ('tpot', 0.7849650349650349)]
```