Final Report of Internship Program 2022

On

"Predict Blood Donations"

MEDTOUREASY

SAJNA P

28th December 2022

ACKNOWLDEGMENTS

The internship opportunity that I had with MedTourEasy was a great chance 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 to have a chance to interact with so many professionals who guided me throughout the internship project and made it a great learning curve for me.

Firstly, I express my deepest gratitude and special thanks to the Training & Development 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 theworking environment productive and very conducive.

TABLE OF CONTENTS

ABSTRACT

A blood transfusion is a routine medical procedure in which donated blood is supplied to you through a narrow tube placed in a vein in your arm. This potentially life-saving procedure can help replace blood lost due to surgery or injury. Blood transfusions can also be helpful if an illness prevents your body from making blood or certain components of your blood correctly. According to WebMD, "about 5 million Americans need a blood transfusion every year".

Estimating blood supply is a serious and recurring challenge for bloodletting managers. In this project aims at work data collected from the Blood Transfusion Service Center's donor database. The dataset obtained from the Machine Learning Repository consists of 748 random donor samples. Predict whether a blood donor will donate within a given time frame and model building process: from examining the dataset to using the tpot library to automate Machine Learning pipeline.

1. NTRODUCTION

1.1 About the Company

MedTourEasy, a global healthcare company, provides you with 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.

1.2 About the Project

Blood transfusion saves millions of lives. But the need and the actual number of donations are not balanced worldwide. The actual reason is not clearly assessed; however, level of knowledge and attitude may be the main contributing factors. Thus, this project aimed to assess blood donors' knowledge and attitude towards blood donation. Data taken from the Blood Transfusion Service Center in Hsin-Chu City in Taiwan. This is a classification problem. Classification algorithms used in machine learning utilize input training data for the purpose of predicting the likelihood or probability that the data that follows will fall into one of the predetermined categories Classification is a form of "pattern recognition". Here Classification algorithms applied to the training data find the same pattern.

Additionally, MedTourEasy, being one of the globally foraying tele-medicine company in global healthcare, it is important for the firm to understand people's attitude towards blood transfusion and because of advancements in clinical blood requirement is increasing every day. medicine. Furthermore, based on the results of the analysis, it can be used to enhance their market presence and capacity planning.

1.3 Objectives and Deliverables

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. The data is stored in datasets/transfusion.data and it is structured according to RFMTC marketing model (a variation of RFM).

2. METHODOLOGY

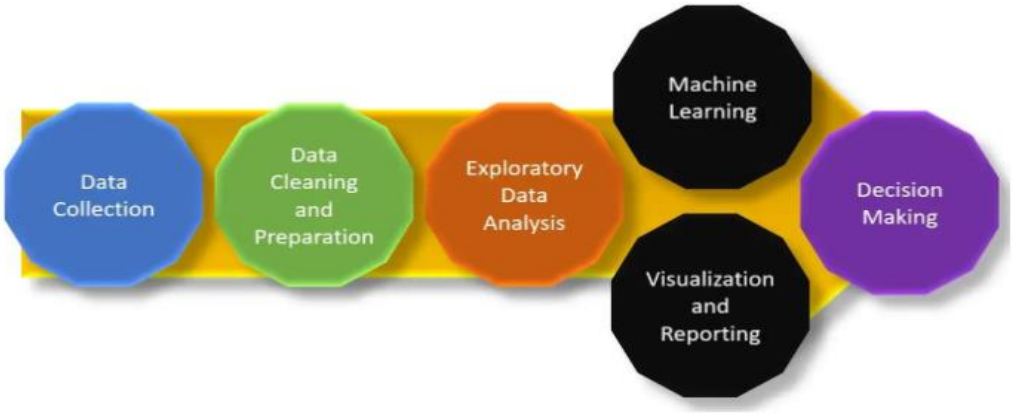
2.1 Importing Dataset

The donor database of Blood Transfusion Service Center in Hsin-Chu City in Taiwan. The center passes their blood transfusion service bus to one university in Hsin-Chu City to gather blood donated about every three months. This dataset contains information about the blood donor, e.g., duration of last month's blood donation, number of times blood donated, how much blood donated, how many times blood donated etc.

We selected 748 donors at random from the donor database. These 748-donor data, each one included R (Recency - months since last donation), F (Frequency - total number of donation), M (Monetary - total blood donated in c.c.), T (Time - months since first donation), and a binary variable representing whether he/she donated blood in March 2007 (1 stand for donating;0 stands for not donating)

2.2 Exploratory Data Analysis

Exploratory Data Analysis(EDA) refers to the critical process of performing initial investigations on data to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. EDA is an approach to analyze the data using visual techniques to obtain desired output.



Exploratory Data Analysis (EDA) is the primary building block of any data-centric project. The above figure shows the process flow from data collection to decision making.

Generally, EDA falls into two categories:-

- **The univariate analysis** involves analyzing one feature, such as summarizing and finding the feature patterns.
- **The multivariate analysis** technique shows the relationship between two or more features using cross-tabulation or statistics.

2.3 Language and Platform Used

2.3.1 Language: Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. It was created by Guido van Rossum and released in 1991.It is used for web development (server-side),software development, mathematics, system scripting. The most recent major version of Python is Python 3.

Python is the most widely used programming language today. When it comes to solving data science tasks and challenges, Python never ceases to surprise its users. Most data scientists are already leveraging the power of Python programming every day. Python has been built with extraordinary Python libraries for data science that are used by programmers every day in solving problems.

Why Python is so popular with developers?

- Python can be used on a server to create web applications. It works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.).
- Versatility, Efficiency, Reliability, Speed, and easy to learn and use.
- Big data, machine learning and cloud computing. and perform complex mathematics.

- Python can connect to database systems. It can also read and modify files.
- Python can be used for rapid prototyping, or for production-ready software development.

Why is Python opted for data analysis?

- Hundreds of Python libraries and frameworks which are valuable for analytics and complex calculations.
- There are several ways you can integrate python data analytics into your existing business intelligence and analytics tools. It includes time series and more complex data structures such as merging, pivoting, and slicing tables to create new views and perspectives on existing sets.
- Python combined with libraries such as iPython and NumPy itself, these tools can form the foundation of a powerful data analytics suite.

2.3.2 Python Libraries

A Python library is a reusable chunk of code that you may want to include in your programs/ projects. The Python Standard Library is a collection of exact syntax, token, and semantics of Python. It comes bundled with core Python distribution. It is written in C, and handles functionality like I/O and other core modules. All this functionality together makes Python the language it is.

More than 200 core modules sit at the heart of the standard library. This library ships with Python. But in addition to this library, you can also access a growing collection of several thousand components from the Python Package Index (PyPI). Here we used some Important Python Libraries for Data Analysis:-

1.Pandas

Pandas (Python data analysis) is a must in the data science life cycle. It is the most popular and widely used Python library for data science, along with NumPy in matplotlib.

It is heavily used for data analysis and cleaning. Pandas provides fast, flexible data structures, such as data frame CDs, which are designed to work with structured data very easily and intuitively.

Features:

- Eloquent syntax and rich functionalities that gives you the freedom to deal with missing data
- Enables you to create your own function and run it across a series of data
- High-level abstraction
- Contains high-level data structures and manipulation tools.

Applications:

- General data wrangling and data cleaning.
- ETL (extract, transform, load) jobs for data transformation and data storage, as it has excellent support for loading CSV files into its data frame format
- Used in a variety of academic and commercial areas, including statistics, finance and neuroscience.
- Time-series-specific functionality, such as date range generation, moving window, linear regression and date shifting.

2.NumPy

NumPy (Numerical Python) is the fundamental package for numerical computation in Python; it contains a powerful N-dimensional array object. It's a general-purpose array-processing package that provides high-performance multidimensional objects called arrays and tools for working with them. NumPy also addresses the slowness problem partly by providing these multidimensional arrays as well as providing functions and operators that operate efficiently on these arrays.

Features:

- Provides fast, precompiled functions for numerical routines
- Array-oriented computing for better efficiency
- Supports an object-oriented approach
- Compact and faster computations with vectorization

Applications:

- Extensively used in data analysis
- Creates powerful N-dimensional array
- Forms the base of other libraries, such as SciPy and scikit-learn.
- Replacement of MATLAB when used with SciPy and matplotlib.

3.Matplotlib

Matplotlib has powerful yet beautiful visualizations. It's a plotting library for Python with around 26,000 comments on GitHub and a very vibrant community of about 700 contributors. Because of the graphs and plots that it produces, it's extensively used for data visualization. It also provides an object-oriented API, which can be used to embed those plots into applications.

Features:

- Usable as a MATLAB replacement, with the advantage of being free and open source
- Supports dozens of backends and output types, which means you can use it regardless of which operating system you're using or which output format you wish to use
- Pandas itself can be used as wrappers around MATLAB API to drive MATLAB like a cleaner
- Low memory consumption and better runtime behavior.
- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Applications:

- Correlation analysis of variables
- Visualize 95 percent confidence intervals of the models
- Outlier detection using a scatter plot etc.
- Visualize the distribution of data to gain instant insights

4.Scikit-learn

Simple and efficient tools for predictive data analysis. Accessible to everybody, and reusable in various contexts. Scikit-learn, a machine learning library that provides almost all the machine learning algorithms you might need. Scikit-learn is designed to be interpolated into NumPy and SciPy.

Features:

- Datasets. Scikit-learn comes with several inbuilt datasets such as the iris dataset, house prices dataset, diabetes dataset, etc.
- Data Splitting.
- XG Boost.
- Machine learning algorithms.

Applications:

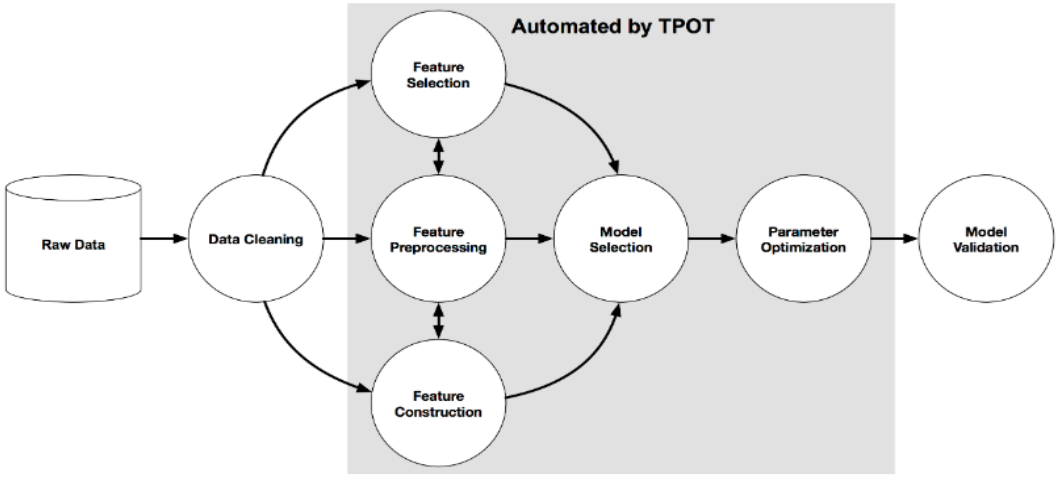
- Clustering
- Classification
- Regression
- Model selection
- Dimensionality reduction
- Preprocessing

2.3.3 TPOT Classifier

TPOT is a Python Automated Machine Learning tool that optimizes machine learning pipelines using genetic programming devices. Automated Machine Learning (AutoML) refers to techniques for automatically discovering well-performing models for predictive modeling tasks with very little user involvement. Once TPOT is finished searching (or you get tired of waiting), it provides you with the Python code for the best pipeline it found so you can tinker with the pipeline from there.

An optimization procedure is then performed to find a tree structure that performs best for a given dataset. Specifically, a genetic programming algorithm, designed to perform a stochastic global optimization on programs represented as trees. TPOT uses a tree-based structure to represent a model pipeline for a predictive modeling problem, including data preparation and modeling algorithms and model hyperparameters.

The figure below taken from the TPOT paper shows the elements involved in the pipeline search, including data cleaning, feature selection, feature processing, feature construction, model selection, and hyperparameter optimization.



2.3.4 Platform: Jupyter Notebook

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R.

3. IMPLEMENTATIONS

3.1 Gathering Requirements and Defining Problem Statement

It is the first step in which requirements are gathered from the resources and required applications are installed followed by defining a problem statement which is to be followed during the development of the project.

Data collection is a systematic approach for gathering and measuring information from a source to predict blood donation interest. It helps us to address specific questions, determine outcomes and forecast future probabilities and patterns.

The dataset transfusion.csv was taken from a blood transfusion service center in Taiwan. This dataset contains information about the blood donor, E.g. duration of last month blood donation, number of times blood donated, how much blood donated, how many times blood donated etc. This dataset consists of 748 instances and 5 attributes. We can use this dataset to predict whether he/she donated blood in March 2007. This dataset is locally available in the directory datasets, and it is stored as a comma separated value (CSV) file. Open Jupyter Notebook through Anaconda prompt then Inspect transfusion.data file

3.2 Inspecting transfusion.data file

Data importing is referred to as uploading the required data into the coding environment from internal sources (computer) or external sources (online websites and data repositories). This data can then be manipulated, aggregated, filtered according to the requirements, and needs of the project.

Once the data is imported in the environment, it is converted into a data frame using python library pandas through read.csv (). It is a wrapper function for read.table() that mandates a comma as separator and uses the input file's first line as header that specifies the table's column names. which makes it easy to maintain the data in the form of table.

The data is stored in datasets/transfusion.data and it is structured according to RFMTC marketing model.

First, open Jupyter Notebook for inspecting the dataset and then import the dataset. Once the data is imported into the environment, then import Importing the required libraries for Exploratory Data Analysis.

```
# Print out the first 5 lines from the transfusion.data file
!head -6 datasets/transfusion.data
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

3.3 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 load the data into memory and understand the attribute information.

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

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

	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

3.4 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. First we do Data cleaning, it is an important step in data preprocessing. Checking null values and attribute types are essential.

Let's verify our hypothesis.

```
# Print a concise summary of transfusion DataFrame
transfusion.info()
```

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

```
transfusion.shape
```

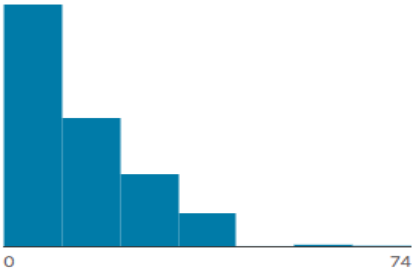
```
(748, 5)
```

```
transfusion.isnull().sum()
```

```
Recency (months)                            0
Frequency (times)                           0
Monetary (c.c. blood)                       0
Time (months)                               0
whether he/she donated blood in March 2007  0
dtype: int64
```

Recency (months)

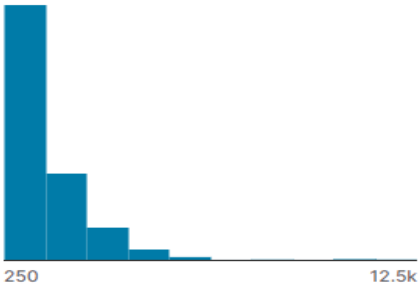
Recency



Valid	748	100%
Mismatched	0	0%
Missing	0	0%
Mean	9.51	
Std. Deviation	8.09	
Quantiles	0	Min
	3	25%
	7	50%
	14	75%
	74	Max

Monetary (c.c. blood)

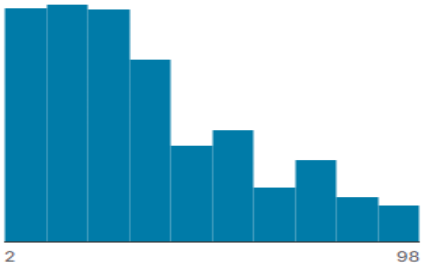
Monetary



Valid	748	100%
Mismatched	0	0%
Missing	0	0%
Mean	1.38k	
Std. Deviation	1.46k	
Quantiles	250	Min
	500	25%
	1000	50%
	1750	75%
	12.5k	Max

Time (months)

Time



Valid	748	100%
Mismatched	0	0%
Missing	0	0%
Mean	34.3	
Std. Deviation	24.4	
Quantiles	2	Min
	16	25%
	28	50%
	50	75%
	98	Max

whether he/she donated blood in March 2007

-

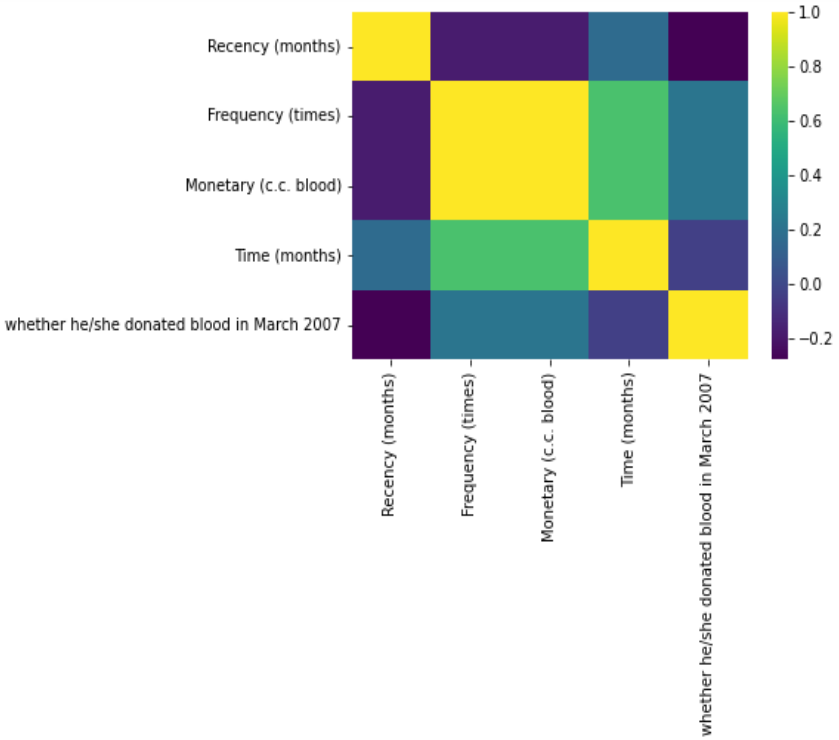
Valid	748	100%
Mismatched	0	0%
Missing	0	0%
Mean	0.24	
Std. Deviation	0.43	
Quantiles	0	Min
	0	25%
	0	50%
	0	75%
	1	Max

Dataset is made of 748 samples. All features are represented with integer numbers and there are no missing values. There is nothing shocking regarding the distributions. We only observe a high value range for the features "Recency", "Frequency", and "Monetary". It means that we have a few extremely high values for these features.

Checking correlation between attributes, Heatmaps helps us to understand the correlation between each variable. When looking at a pair of features, we don’t see any striking combinations as well. However, we can note that the "Monetary" and "Frequency" features are perfectly correlated: all the data points are aligned on a diagonal.

As a conclusion, this dataset would be a challenging dataset: it suffers from class imbalance, correlated features and thus very few features will be available to learn a model, and none of the feature combinations were found to help at predicting.

```
sns.heatmap(transfusion.corr(),cmap='viridis')
```



3.5 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.

```
# 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)
```

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

3.6 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) our dataset is. Plot the count plot of Target Variable.

```
# Print target incidence proportions, rounding output to 3 decimal places
transfusion.target.value_counts()
```

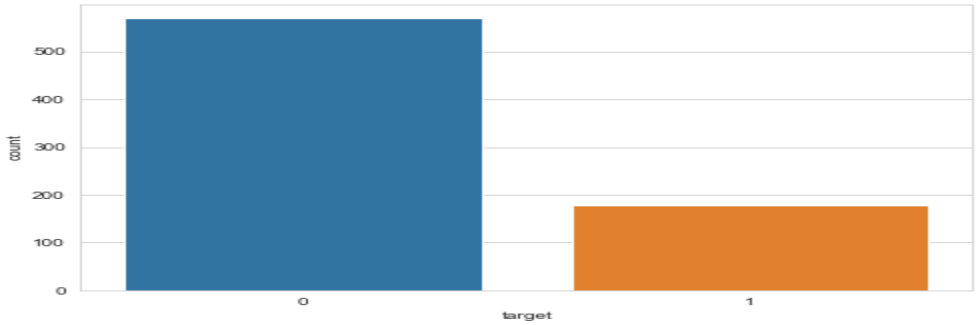
```
0    570
1    178
Name: target, dtype: int64
```

```
transfusion.target.value_counts(normalize=True).round(3)
```

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

```
# 0 means no, 1 - yes
sns.set_style("whitegrid")
plt.figure(figsize=(8,5))
sns.countplot(x="target",data=transfusion)
```

```
<AxesSubplot:xlabel='target', ylabel='count'>
```



From the total 748 donor data points, 570 donors, not donated blood, and 178 donors donated blood. Target incidence indicates that about 76% of the time an individual does not give blood. The present dataset is imbalanced due to (570 (NDB) and 178(DB) classes).

3.7 Splitting transfusion into train and test datasets

We'll now use train_test_split() method to split transfusion DataFrame from the scikit learn library. Target incidence informed us that in our dataset 0s 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%. In our case, we'll stratify on the target column.

```
# 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)
```

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

```
y_train.head(2)
```

```
334    0
99     1
Name: target, dtype: int64
```

3.8 Selecting model using TPOT Classifier

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.

```
# 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}')
```

```
AUC score: 0.7853

Best pipeline steps:
1. LogisticRegression(C=0.1, random_state=42)
```

3.9 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 regression 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.

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.

```
# X_train's variance, rounding the output to 3 decimal places
X_train.var().round(3)
```

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

3.10 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 gain 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.

```
# 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=[col_to_normalize], inplace=True)

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

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

3.11 Training the linear 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 linear regression model.

```
# 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}')
```

```
AUC score: 0.7891
```

4. 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.

5. REFERENCES

Data Collection

Input data and statistics:

a. https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center

Programming References

The following websites have been referred for Python coding and Jupyter Notebook tutorials:

a. https://datascienceplus.com/category/programming

b. https://www.python.org/

c. https://inria.github.io/scikit-learn-mooc/python_scripts/datasets_blood_transfusion.html

d. https://wesmckinney.com/book/

e. https://www.coursera.org/learn/data-analysis-with-python

f. https://medium.com/@manishbendale85/exploratory-data-analysis-blood-transfusion-service-center-dataset-e8102f0036e9

g. https://bmcresnotes.biomedcentral.com/articles/10.1186/s13104-019-4776-0#Abs1

h. https://jupyter.org/

i. https://www.anaconda.com/products/distribution