Final Report of
Internship Program 2021

“Project Give Life: Predict Blood Donations”

MEDTOUREASY

ACKNOWLDEGMENTS

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.

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.

TABLE OF CONTENTS

1. ABSTRACT

In this Project, you will work with data collected from the donor database of Blood Transfusion Service Centre in Hsin-Chu City in Taiwan. The centre 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. Your 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.

To complete this Project, you need to know some Python, pandas, and logistic regression. We recommend one is familiar with the content in Data Camp’s Manipulating Data Frames with pandas, Pre-processing for Machine Learning in Python, and Foundations of Predictive Analytics in Python (Part 1) courses.

"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, you 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. Your 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.

1.1 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.

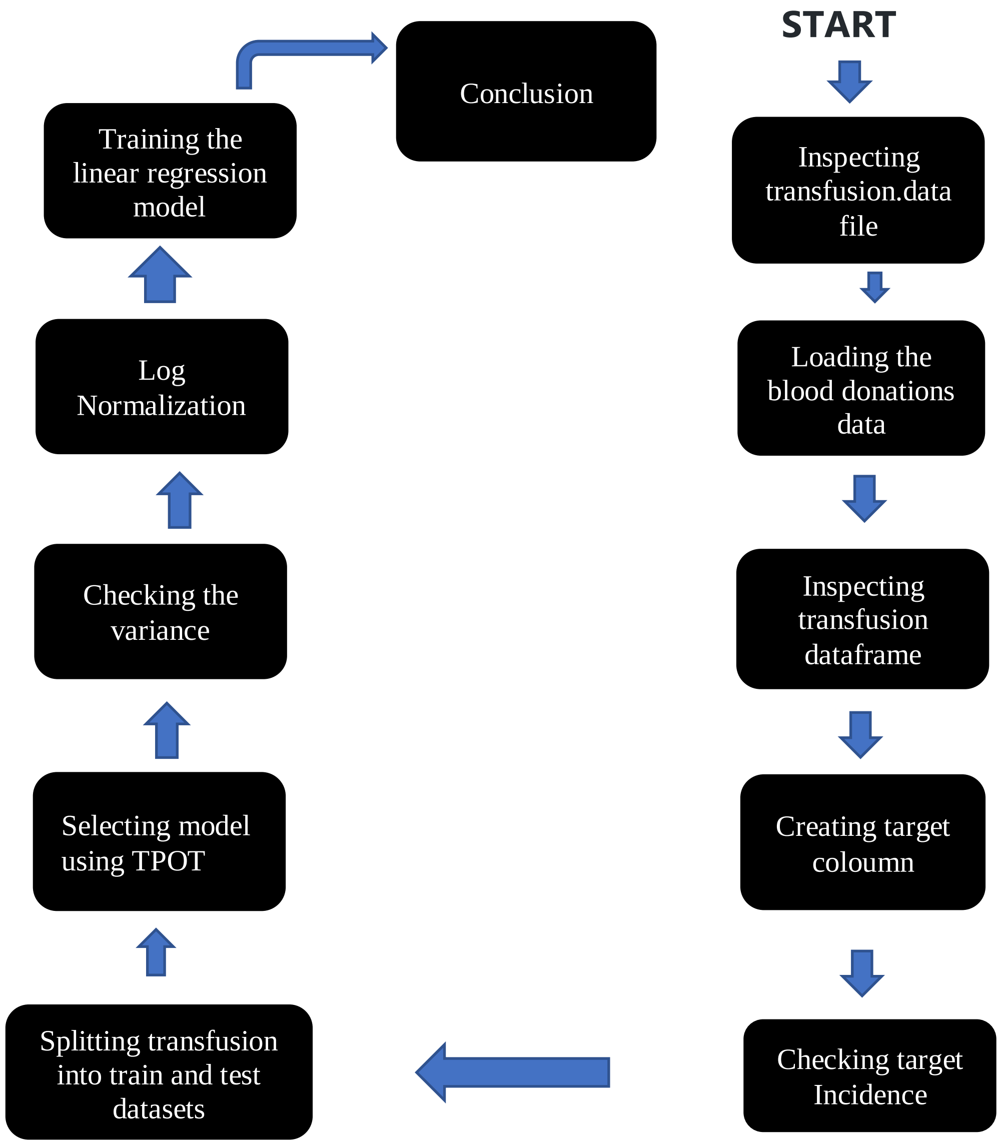
1.2 About the Project

The donation of blood is important because most often people requiring blood do not receive it on time causing loss of life. Examples include severe accidents, patients suffering from dengue or malaria, or organ transplants. Extreme health conditions such as Leukemia and bone marrow cancer, where affected individuals experience sudden high blood loss and need an urgent supply of blood and do not have it can also lead to loss of life. Sound data-driven systems for tracking and predicting donations and supply needs can improve the entire supply chain, making sure that more patients get the blood transfusions they need, which can reduce mortality risk. One of the interesting aspects about blood is that it is not a typical commodity. First, there is the perishable nature of blood. Grocery stores face the dilemma of perishable products such as milk, which can be challenging to predict accurately so as to not lose sales due to an expired product. Blood has a shelf life of approximately 42 days according to the American Red Cross (Darwiche, Feuilloy et al. 2010). However, what makes this problem more challenging than milk is the stochastic behavior of blood supply to the system as compared to the more deterministic nature of milk supply. Whole blood is often split into platelets, red blood cells, and plasma, each having their own storage requirements and shelf life. For example, platelets must be stored around 22 degrees Celsius, while red blood cells 4 degree Celsius, and plasma at -25 degrees Celsius. Moreover, platelets can often be stored for at most 5 days, red blood cells up to 42 days, and plasma up to a one calendar year. Amazingly, only around 5% of the eligible donor population actually donate (Linden, Gregorio et al. 1988, Katsaliaki 2008). This low percentage highlights the risk humans are faced today as blood and blood products are forecasted to increase year-on-year. This is likely why so many researchers continue to try to understand the social and behavioral drivers for why people donate to begin with. The primary way to satisfy demand is to have regularly occurring donations from healthy volunteers. In our study, we focus on building a data-driven system for tracking and predicting potential blood donors. We investigate the use of various binary classification techniques to estimate the probability that a person will donate blood in March 2007 or not based on his past donation behavior. There is a time lag between the demand of blood required by patients suffering extreme blood loss and the supply of blood from blood banks. We try to improve this supply-demand lag by building a predictive model that helps identify the potential donors.

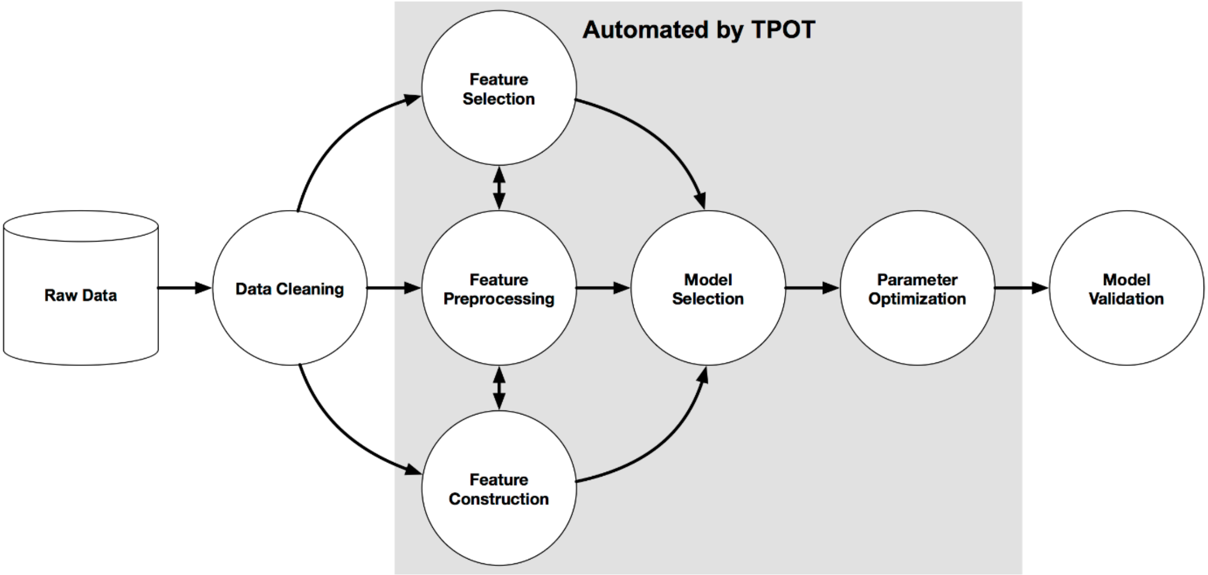
2. Methodology

2.1 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.


START
Inspecting transfusion.data file
Loading the blood donations data
Inspecting transfusion dataframe
Creating target coloumn
Checking target Incidence
Splitting transfusion into train and test datasets
Selecting model using TPOT
Checking the variance
Log Normalization
Training the linear regression model
Conclusion

2.2 Use of Case Diagram



Above figure shows the use case of the project. 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. 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.

2.3 Language and Platform Used:

1. Language: Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Major Features are:

- Easy to code: Python is a high-level programming language. ...
- Free and Open Source: ...
- **Object**-Oriented Language: ...
- GUI Programming Support: ...
- High-Level Language: ...
- Extensible feature: ...
- Python is **Portable** language: ...
- Python is Integrated language

2. Application Used: Jupyter notebook

One major feature of the Jupyter notebook is the ability to display plots that are the output of running code cells. The IPython kernel is designed to work seamlessly with the matplotlib plotting library to provide this functionality. Specific plotting library integration is a feature of the kernel.

Main features of the application

- In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection.
- The ability to execute code from the browser, with the results of computations attached to the code which generated them.
- Displaying the result of computation using rich media representations, such as HTML, LaTeX, PNG, SVG, etc. For example, publication-quality figures rendered by the matplotlib library, can be included inline.

- In-browser editing for rich text using the Markdown markup language, which can provide commentary for the code, is not limited to plain text.
- The ability to easily include mathematical notation within markdown cells using LaTeX, and rendered natively by MathJax.

3. IMPLEMENTATION

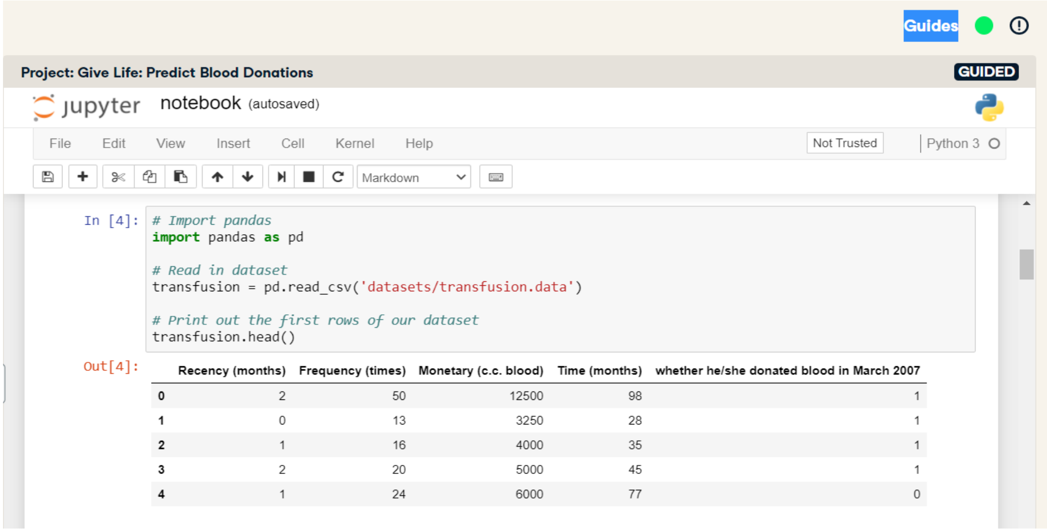
3.1 Data Collection and Importing

Data collection is a systematic approach for gathering and measuring information from a variety of sources in order to obtain a complete and accurate picture of an interest area. It helps an individual or organization to address specific questions, determine outcomes and forecast future probabilities and patterns.

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.

Packages Used:

pd.read_csv: read_csv is an important pandas function to read csv files and do operations on it. ... For instance, one can read a csv file not only locally, but from a URL through read_csv or one can choose what columns needed to export so that we don't have to edit the array later.


Guides
Project: Give Life: Predict Blood Donations
GUIDED
jupyter notebook (autosaved)
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3
Markdown

In [4]:

```
# 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()
```

Out[4]:

	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

3.2 Inspecting transfusion Data Frame

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:

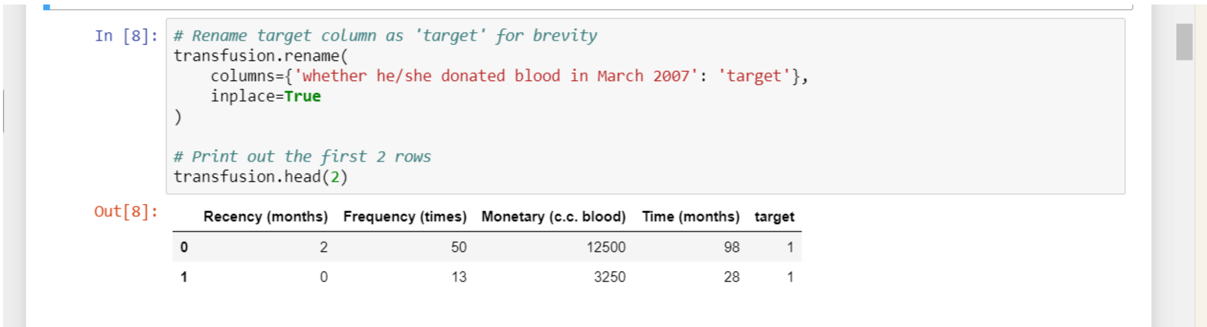
- R (Recency - months since the last donation)
- F (Frequency - total number of donation)
- M (Monetary - total blood donated in c.c.)
- T (Time - months since the first donation)
- a binary variable representing whether he/she donated blood in March 2007 (1 stands for donating blood; 0 stands for not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

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

3.3 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.

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 [8]:

```
# 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)
```

Out[8]:

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

3.4 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:

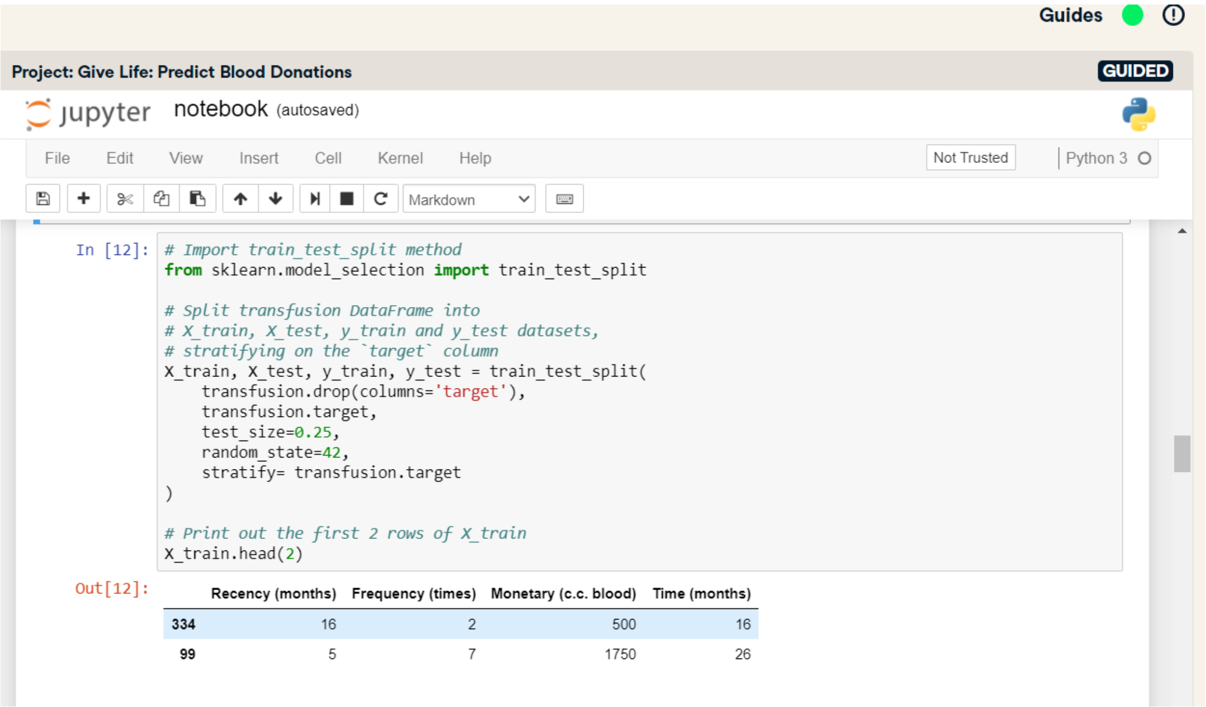
- 0 - the donor will not give blood
- 1 - the donor will give blood

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.

3.5 Splitting transfusion into train and test datasets

We'll now use train test_split() method to plit transfusion DataFrame.

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%. 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.


Guides
Project: Give Life: Predict Blood Donations
GUIDED
jupyter notebook (autosaved)
File
Edit
View
Insert
Cell
Kernel
Help
Not Trusted
Python 3
Markdown

```
In [12]: # 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)
```

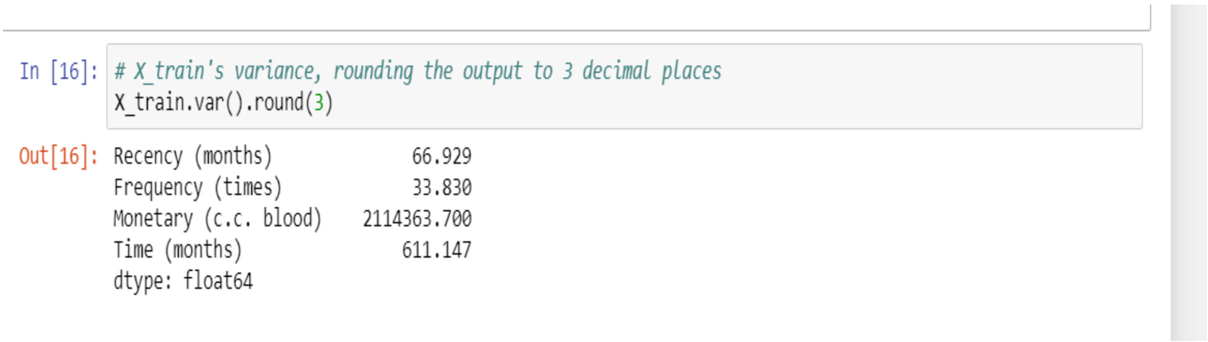
```
Out[12]:
      Recency (months)  Frequency (times)  Monetary (c.c. blood)  Time (months)
 334                16                  2                    500             16
  99                 5                  7                   1750             26
```

3.6 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.

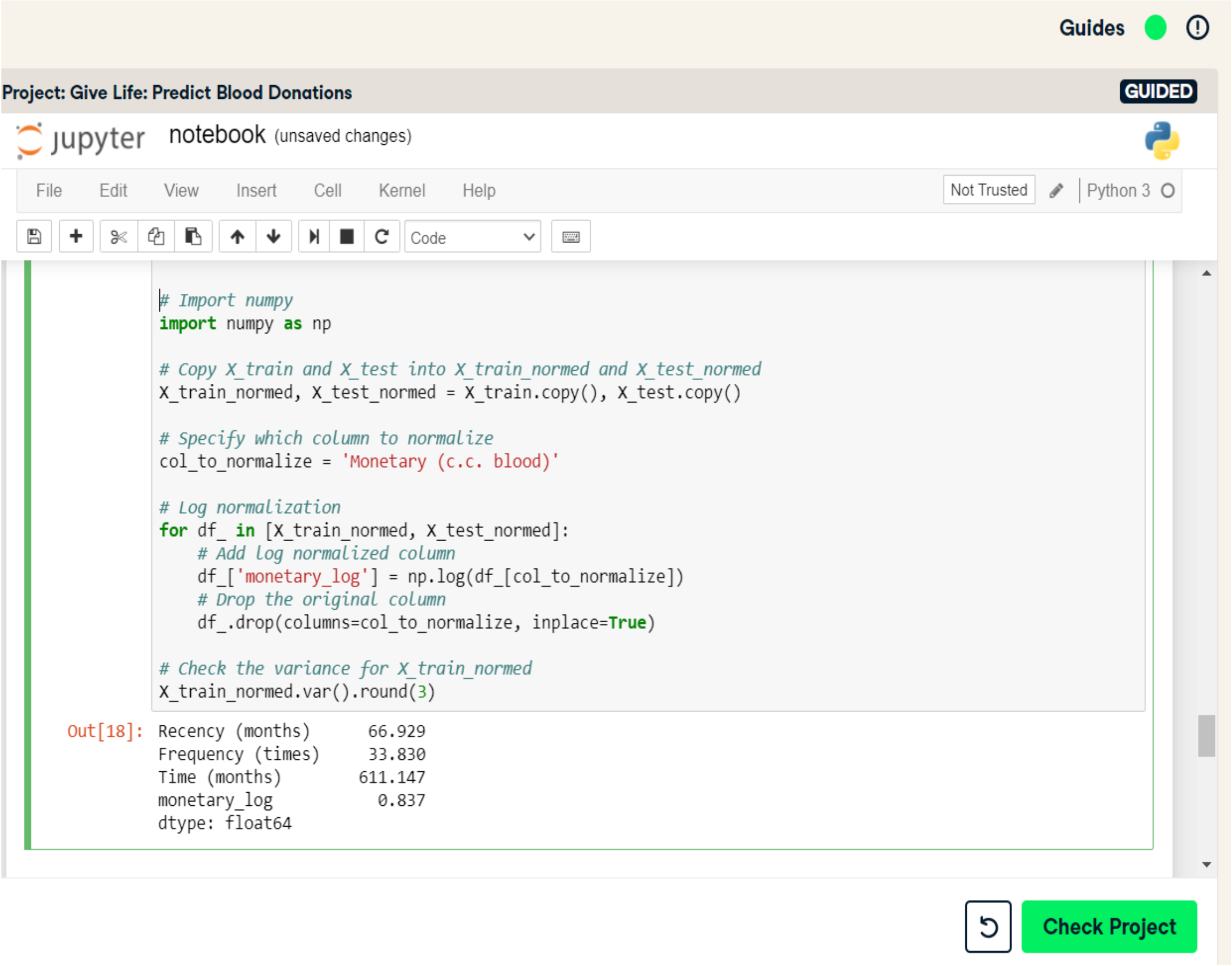


```
In [16]: # X_train's variance, rounding the output to 3 decimal places
         X_train.var().round(3)

Out[16]: Recency (months)               66.929
         Frequency (times)              33.830
         Monetary (c.c. blood)     2114363.700
         Time (months)                 611.147
         dtype: float64
```

3.7 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.



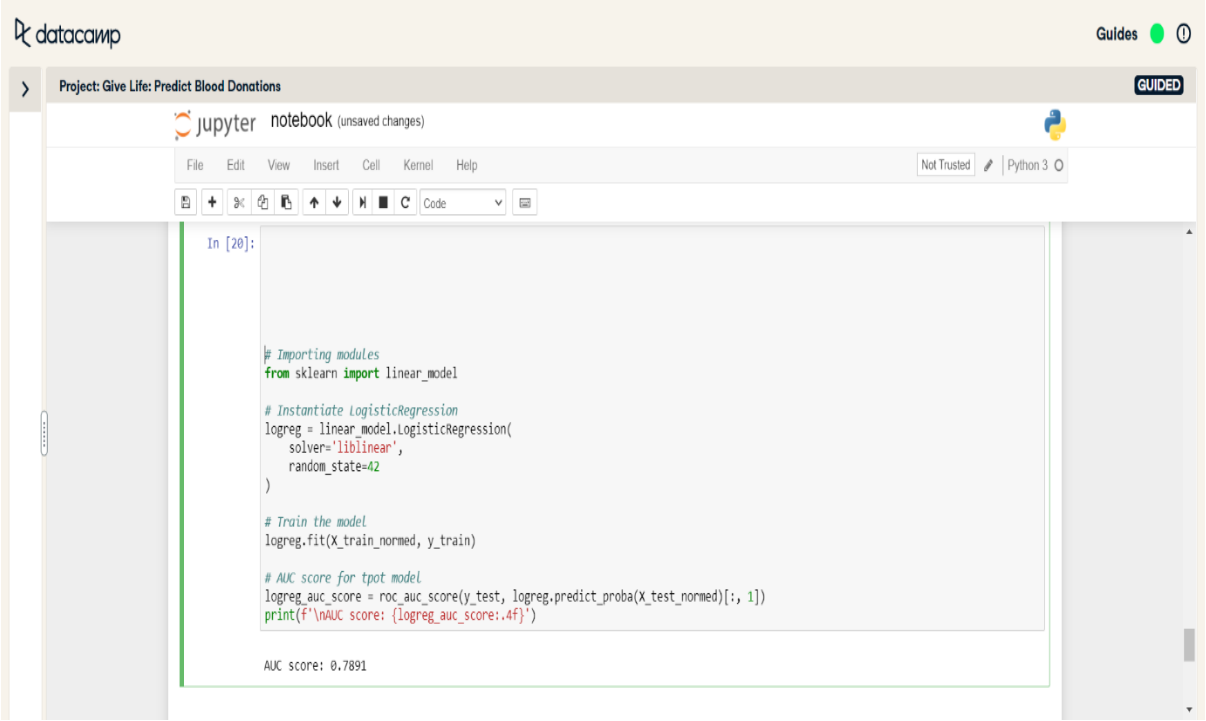
```
# Import numpy
import numpy as np

# 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=col_to_normalize, inplace=True)

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

```
Out[18]: Recency (months)      66.929
         Frequency (times)     33.830
         Time (months)        611.147
         monetary_log           0.837
         dtype: float64
```

3.8 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.



datacamp
Guides
Project: Give Life: Predict Blood Donations
GUIDED
jupyter notebook (unsaved changes)
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3
Code
In [20]:

```
# 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
```

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.

With regards to the future work, the firm aims at regularly updating the dashboards with time and integrating it with their systems so as to continually draw conclusions and analyze the results.

REFERENCES

Programing Refrences:

https://projects.datacamp.com/projects/646