Final Report of Traineeship Program 2023

PREDICT BLOOD DONATIONS

MEDTOUREASY

28th May 2023

Abstract

Blood transfusions are critical in various medical procedures and emergencies, necessitating the availability of an adequate blood supply. However, blood banks often need help with predicting blood demand and maintaining a consistent supply due to the unpredictable nature of blood donation patterns. This project aims to address this challenge by developing a machine-learning model that predicts whether a blood donor will donate blood or not based on their past donation history. The project utilizes a dataset containing pertinent information about blood donors, including the number of previous donations, donation frequency, time since the last donation and time since the first donation. By analyzing this data, the model seeks to identify underlying patterns and factors influencing a donor's decision-making process. TPOT selected a supervised learning technique, logistic regression, which is employed to train the predictive model on the collected dataset. These algorithms enable the model to learn from historical data and establish relationships between input variables (donor characteristics) and the output variable (whether the donor will donate blood or not). The performance of the developed model is evaluated and validated using a combination of training and testing datasets. The model's performance is evaluated using an AUC score, and recall on a testing dataset evaluates the model's performance. This project also shows improvement in the model after normalizing significant features in data. The potential impact of this project includes optimizing blood supply management, identifying potential donors, and improving the efficiency of blood donation campaigns. By accurately predicting blood donation behavior, this model aims to minimize blood shortages, streamline resource allocation, and ultimately contribute to saving lives by ensuring a reliable and sufficient supply of blood products.

1. Introduction

1.1 About MedTourEasy

MedTourEasy, a global healthcare company, provides the information needed to evaluate your global options. It helps you find the right healthcare solution based on specific health needs and affordable care while meeting the quality standards 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 lives - from replacing lost blood during major surgery or a serious injury to treating various illnesses and blood disorders. Ensuring enough blood is needed whenever needed is a serious challenge for health professionals. According to WebMD, "about 5 million Americans need a blood transfusion every year" [1].

Blood transfusions are crucial in various medical procedures, emergencies, and treatments, making maintaining an adequate supply of blood products essential. However, blood banks often face challenges in predicting blood demand and ensuring a consistent supply due to the unpredictable nature of blood donation patterns. We can create a model identifying patterns and factors influencing blood donation behavior by analyzing historical data and using machine learning algorithms.

Predicting whether a blood donor will donate blood or not can significantly aid blood banks and healthcare organizations in planning and managing their blood supply effectively. By leveraging machine learning techniques, we can develop a predictive model that utilizes past data of blood donors to determine the likelihood of future blood donations. This project aims to construct such a model, which can assist in identifying potential donors and optimizing donation campaigns.

1.3 Objectives and Deliverables

The primary objective of this project is to develop a machine-learning model that accurately predicts whether a blood donor will donate blood or not based on their past donation history. Healthcare organizations can proactively target potential donors, plan donation campaigns more effectively, and ensure an adequate supply of blood products by understanding the factors contributing to a donor's decision-making process.

We have data on blood donors about how many months since the last donation, the total number of blood donations, the total blood donated, and how many months from the first donation. Our model should predict whether the blood donor will donate next time or not. We will train past blood donor data and their donation state (0 for not donated, 1 for donated) into our model at a particular time.

2. Methodology

2.1 Project Sequence

First, we analyze the dataset to address questions like how many columns there are, what they describe and what their data type is. After that, ensure no null data points are in the dataset. Subsequently, identify the target column which we want to predict using the model and rename it with an appropriate name. We also check target column incidence, which means how many 0s are in the target column compared to 1s?

After preprocessing the dataset, we proceed to split the dataset into test and train data. Then we select an appropriate model using the TPOT library and find out the AUC score for this model. Then we analyze the effect of normalization on model accuracy. For that, we check out the variance for each column and log-normalize the column with a higher variance. We again train the new model and compare its AUC score with the previous model.

2.2 Used Language and Platform

2.2.1 Language: Python

Python is a popular programming language in scientific computing because it has many data-oriented feature packages that can speed up and simplify data processing, thus saving time [2]. It has many completely free libraries that are open to the public. That critical factor makes Python essential for data analysis and data science.

2.2.2 Platform: Jupyter Notebook

Jupyter Notebooks is one of the leading open-source tools for developing and managing data analytics [3]. It produces documents (notebooks) that combine inputs (code) and outputs into a single file. This single document approach enables users to develop, visualize the results and add information, charts, and formulas that make work more understandable, repeatable, and shareable. Jupyter Notebooks support over 40 programming languages, focusing significantly on Python [4]. Since it is a free and open-source tool, anyone can use it freely for their data science projects.

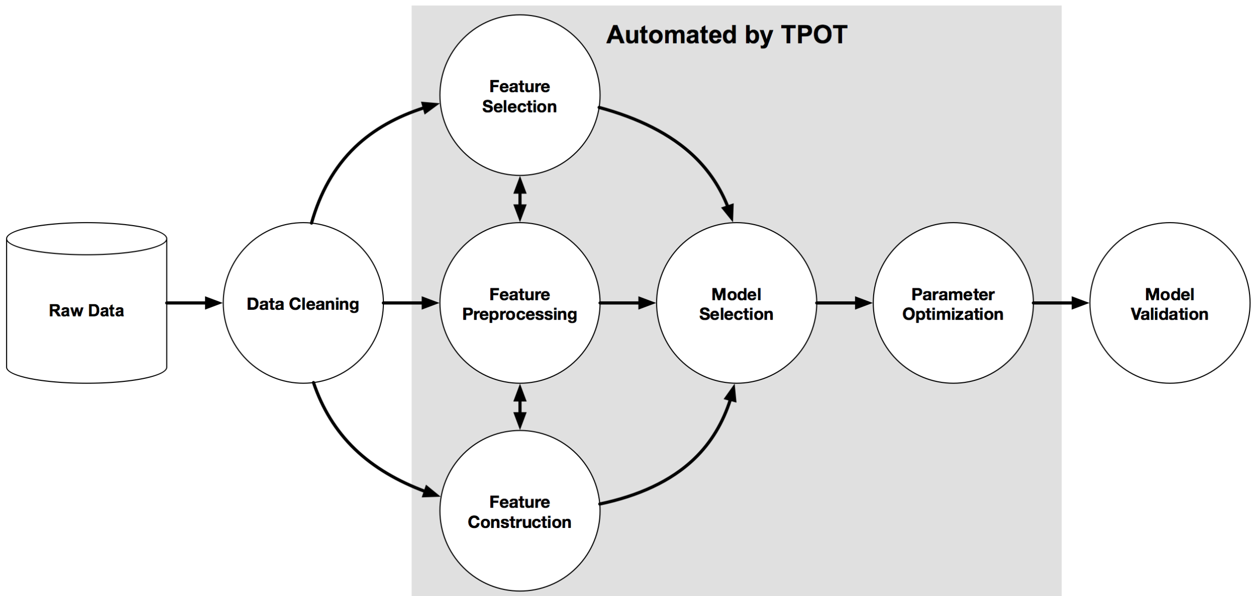
2.2.3 Python Packages

Our code uses 4 libraries: NumPy, Pandas, Tpot and Sci-kit learn.

NumPy includes features of mathematical operations like average, mode, the sum of array and functions like sin, cos, log and least integer. NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for array processing. Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package, Numarray, was also developed, having some additional functionalities. In 2005, Travis Oliphant created the NumPy package by incorporating the features of Numarray into the Numeric package. There are many contributors to this open-source project. Using NumPy, a developer can perform the following operations: 1. Mathematical and logical operations on arrays. 2. Fourier transforms and routines for shape manipulation. Operations related to linear algebra. 3. NumPy has in-built functions for linear algebra and random number generation [5].

Pandas library can read data from Excel and CSV files and make a data frame [6]. Pandas is an open-source Python Library providing high-performance data manipulation and analysis tools using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data. In 2008, developer Wes McKinney started developing pandas when needing high-performance, flexible data analysis tools. Using Pandas, we can accomplish five typical steps in processing and analyzing data, regardless of the origin of data — load, prepare, manipulate, model, and analyze. Python with Pandas is used in various fields, including academic and commercial domains, including finance, economics, Statistics, analytics, etc. [7]

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 [8]. Note the outcome of this search will be a scikit-learn pipeline, meaning it will include any pre-processing steps and the model [9].



Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling, including classification, regression, clustering and dimensionality reduction via a consistency interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. It was originally called scikits.learn and was initially developed by David Cournapeau as a Google Summer of Code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel from FIRCA (French Institute for Research in Computer Science and Automation) took this project to another level and made the first public release (v0.1 beta) on 1st Feb. 2010 [10].

3. Implementation

3.1 Inspecting the “transfusion.data” file

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 dataset is structured according to the RFMTC marketing model (a variation of RFM). RFMTC is a variation of the RFM model. RFM stands for Recency, Frequency and Monetary Value, and it is commonly used in marketing to identify your best customers. In our case, our customers are blood donors. Below is a description of what each column means in our dataset:

- R (Recency - months since the last donation)
- F (Frequency - total number of donations)
- 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 stand for donating blood; 0 stands for not donating blood)

The data is stored in `datasets/transfusion.data` and structured according to the RFMTC marketing model (a variation of RFM). Let's inspect the data. First, we print out the first 5 lines from `datasets/transfusion.data` using the `head` shell command. The syntax is as follows: `!head -n datasets/transfusion.data` where n is the number of lines we want to print.

In [1]:
```
#Print out the first 5 lines from the transfusion.data file
!head -5 datasets/transfusion.data
```

```
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
```

3.2 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 load the data into memory. To load the dataset,

- Imported the Pandas library using `import pandas as pd`
- Created path variable and stored data file path into it
- Loaded data into transfusion variable using `pd.read_csv()` function
- To verify the dataset was loaded correctly, we printed the first 5 rows using the head() function

pd.read_csv(path, sep=','): Takes a file path and "," delimiter as a separator and returns pandas Dataframe
df.head(): Display first 5 rows of the dataset including columns names

In [2]:
```python
# Import pandas
import pandas as pd

# Read in dataset
path = 'datasets/transfusion.data'
transfusion = pd.read_csv(path, sep=",")

# Print out the first rows of our dataset
# ... YOUR CODE FOR TASK 2 ...
transfusion.head()
```

Out[2]:

	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

3.3 Inspecting transfusion DataFrame

Let's analyze every column in our dataset. We want to check if every column in our DataFrame has a numeric type, precisely what we want when building a machine learning model. Printed a concise summary of the transfusion Dataframe with the df.info() function. Our dataset has 748 rows and all columns have int64 data type with 0 null value.

df.info(): Returns index types, column types, non-null values and memory usage, including the index dtype and column dtypes, non-null values and memory usage

In [3]:
```python
# Print a concise summary of transfusion DataFrame
# ... YOUR CODE FOR TASK 3 ...
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
```

3.4 Creating a target column

We are aiming to predict the value of `whether he/she donated blood in March 2007` column. So renamed this to `target` so it's more convenient to work with. Renamed column name using `transfusion.rename(columns = {‘whether he/she donated blood in March 2007’:‘target’}, inplace = True)`. 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.

df.rename(columns = {}, inplace=True): Rename a column name
df.head(n): Prints first n rows with column names

In [4]:
```
# 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
# ... YOUR CODE FOR TASK 4 ...
transfusion.head(2)
```

Out[4]:

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

3.5 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 target value in a dataset. How many 0s are in the target column compared to how many 1s? Target incidence gives us an idea of how balanced (or imbalanced) our dataset is.

We used `value_counts()` method on `transfusion.target` column to print target incidence proportions, set `normalize=True` and rounded the output to 3 decimal places using `round()` function. The dataset has 76.2% `0`'s and 23.8% `1`'s.

value_counts(): Returns counts of unique values
value_counts(normalize=True): Returns the relative frequencies of the unique values instead
round(n): Rounding number upto n decimal places

```
In [5]:
# Print target incidence proportions, rounding output to 3 decimal places
# ... YOUR CODE FOR TASK 5 ...
transfusion['target'].value_counts(normalize=True).round(3)
```

```
Out[5]: 0    0.762
        1    0.238
        Name: target, dtype: float64
```

3.6 Splitting transfusion into train and test datasets

We used `train_test_split()` method to split `transfusion` DataFrame. First, we imported the `train_test_split()` function using the code below:

`from sklearn.model_selection import train_test_split`.

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 done using the `train_test_split()` method from the `scikit-learn` library by specifying the `stratify` parameter. In our case, we stratified the `target` column.

train_test_split(X,y,test_size,random_state,satisfy): Split X, y array into X_train, X_test, y_train and y_test datasets, stratifying on the target column having test size = test_size and random state = randon_state.

In [6]:
```
# 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
# ... YOUR CODE FOR TASK 6 ...
X_train.head(2)
```

Out[6]:

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

3.7 Selecting model using TPOT

We are using TPOT to help us zero in on one model that we can then explore and optimize further. First imported `TPOTClassifier` and `roc_auc_score` as follows:

```
from tpot import TPOTClassifier
from Sklearn.metrics import roc_auc_score
```

Created an instance of `TPOTClassifier` and assigned it to `tpot` variable. We have taken `generations=5`, `population_size=50`, `verbosity=2`, `random_state=42` according to TPOT's documentation. [6] We have specified `scoring='roc_auc'` for optimizing `roc_auc` matrix for and added `random_state=42` for reproducibility. We used `TPOT light` configuration. `TPOT` has the same API as `sci-kit learn`, so we can train the model using `fit()` method. Data pre-processing affects the model's performance, and `tpot`'s `fitted_pipeline_` attribute allows us to see what pre-processing (if any) was done in the best pipeline.

TPOT gave us the best pipeline as follows, with an AUC score of 0.7634.

```
MultinomialNB(Normalizer(input_matrix, norm=l1), alpha=0.01, fit_prior=True)
```

TPOTClassifier(): Performs an intelligent search over machine learning pipelines that can contain supervised classification models, preprocessors, feature selection techniques, and any other estimator or transformer that follows the scikit-learn API

fit(X,y): Trains the model

roc_auc_score(y, X): Calculates AUC score for a model

In [7]:

```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}')
```

```
Generation 1 - Current best internal CV score: 0.7422459184429089

Generation 2 - Current best internal CV score: 0.7422459184429089

Generation 3 - Current best internal CV score: 0.7422459184429089

Generation 4 - Current best internal CV score: 0.7423330644124078

Generation 5 - Current best internal CV score: 0.7457169665719596

Best pipeline: MultinomialNB(Normalizer(input_matrix, norm=l1), alpha=0.01, fit_prior=True)

AUC score: 0.7634

Best pipeline steps:
1. Normalizer(norm='l1')
2. MultinomialNB(alpha=0.01)
```

3.8 Checking the variance

We saw that in the previous section, TPOT picked `LogisticRegression` as the best model for our dataset with no pre-processing steps, giving us an AUC score of 0.7634. This is a great starting point. We are now proceeding to make it better.

One of the assumptions for linear regression models was that the data and the features we give it are related linearly or can be measured with a linear distance metric. If a feature in our dataset has a high variance that's an order of magnitude or greater than the other features, this could impact the model's ability to learn from other features. Correcting for high variance is called normalization. First, we checked the variance for all features in the input matrix by `var()` function.

df.var(): Returns column-wise variance in data frame df

```
In [8]:
# X_train's variance, rounding the output to 3 decimal places
# ... YOUR CODE FOR TASK 8 ...
X_train.var().round(3)
```

```
Out[8]: Recency (months)               66.929
        Frequency (times)              33.830
        Monetary (c.c. blood)     2114363.700
        Time (months)                 611.147
        dtype: float64
```

3.9 Log normalization

`Monetary (c.c. blood)`'s variance is very high compared 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. This is done as follows:

- Copied `X_train` and `X_test` into `X_train_normed` and `X_test_normed`, respectively, using `copy()` function.
- Assigned the column name (a string) with the highest variance to the `col_to_normalize` variable.
- For `X_train` and `X_test` DataFrames:
  - Log normalized `col_to_normalize` using `np.log()` to add it to the DataFrame.
  - Dropped `col_to_normalize` using `drop()` function.
- Printed `X_train_normed` variance using `var()`.

df_arr.copy(): Copies a data frame array (column) into another variable
np.log(): Logarithm function from Numpy library
df.drop(columns = col_name, inplace=True): Drops (Delete) a column in data frame

```
In [9]:
# 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
# ... YOUR CODE FOR TASK 9 ...
X_train_normed.var().round(3)
```

```
Out[9]: Recency (months)       66.929
        Frequency (times)      33.830
        Time (months)         611.147
        monetary_log            0.837
        dtype: float64
```

3.10 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 left it as it is. We trained the linear regression model as follows:

- Imported libraries: `from sklearn import linear_model`.
- Created an instance of `linear_model.LogisticRegression` and assigned it to `logreg` variable.
- Trained logreg model using the `fit()` method.
- Calculated AUC score for this model using `roc_auc_score()` function and stored into `logreg_auc_score` variable.

The scikit-learn library has a consistent API for fitting a model:

1. Create an instance of a model you want to train
2. Train it on your train datasets using the fit method

In [10]:
```
# 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 blood supply forecast allows for appropriate action to be taken ahead of time, saving more lives [11].

This report explored automatic model selection using TPOT and the AUC score of 0.7634. This is better than simply choosing `0` always (the target incidence suggests that such a model would have a 76% success rate). We then logged, normalized our training data and improved the AUC score by 3.37%. In machine learning, even minor improvements in accuracy can be necessary, 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.

In [11]:
```
# 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
)
```

Out[11]:
```
[('logreg', 0.7890972663699937), ('tpot', 0.7634297520661156)]
```

5. Future Scopes

The future scope of this project includes incorporating additional data such as donor information such as age, gender, location, socio-economic factors and lifestyle information to enhance the predictive power of the model, integrating real-time data feeds for dynamic predictions, conducting geographic analysis to identify regional patterns in blood donation behavior, considering user feedback and sentiment analysis to understand donor motivations better, collaborating with blood banks and organizations for validation and practical implementation, and expanding the model to include other blood-related predictions such as blood type identification and demand forecasting. These future enhancements aim to provide more robust and customized solutions for blood supply management, resource allocation, and effective blood donation campaigns.

Acknowledgment

I express my sincere gratitude for the opportunity to intern with MedTourEasy in the field of Data visualization in Data Analytics and Data Science. I would like to extend my heartfelt appreciation for considering my application and providing me with this invaluable learning experience. The traineeship opportunity that I had with MedTourEasy was a great chance to learn and understand the intricacies of Data Analytics and personal and professional development.

I am genuinely thrilled to be a part of your team and contribute to the exciting projects and initiatives your organization undertakes in data analysis and visualization. This traineeship represents a significant step in my career journey, and I am grateful for the chance to gain practical exposure in such a dynamic and evolving field. I am very obliged to have a chance to interact with my training mentor Mr. Ankit Hasija who guided me throughout the traineeship project and made it a great learning curve for me. I am confident that working alongside your experienced professionals will not only enhance my technical abilities but also enable me to understand the practical applications of these disciplines in real-world scenarios.

Thank you once again for your confidence in my abilities. I am excited to commence this internship and contribute to the growth and success of your organization.

References

[1] WebMD. (n.d.). *Blood transfusion: Purpose, procedure, risks, Complications*. WebMD. https://www.webmd.com/a-to-z-guides/blood-transfusion-what-to-know#1
[2] McKinney, W. (2010). Data structures for statistical computing in python. Proceedings of the 9th Python in Science Conference, 445(1), 51–56.
[3] Mendez, K. M., Pritchard, L., Reinke, S. N., & Broadhurst, D. I. (2019). Toward collaborative open data science in metabolomics using Jupyter Notebooks and cloud computWorking1–16.
[4] Nagar, S. (2018). IPython. In Introduction to Python for Engineers and Scientists (pp. 31–45). Springer.
[5] *Numpy - introduction Online Courses and eBooks Library*. Available at: https://www.tutorialspoint.com/numpy/numpy_introduction.htm.
[6] Wade, R. (2020). Reading CSV Files. In Advanced Analytics in Power BI with R and Python (pp. 151–175). Springer.
[7] *Python pandas - introduction*. Online Courses and eBooks Library. (n.d.-a). https://www.tutorialspoint.com/python_pandas/python_pandas_introduction.htm
[8] EpistasisLab. (n.d.). *EpistasisLab/tpot: A Python Automated Machine Learning Tool that optimizes machine learning pipelines using genetic programming.* GitHub. https://github.com/EpistasisLab/tpot
[9] *Sklearn.pipeline.pipeline*. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
[10] *Scikit learn - introduction*. Online Courses and eBooks Library. (n.d.). https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm
[11] 2 News Oklahoma KJRH Tulsa. (2019, January 18). *Red Cross in blood donation "crisis."* 2 News Oklahoma KJRH Tulsa. https://www.kjrh.com/news/local-news/red-cross-in-blood-donation-crisis

Appendix

Colab File link of code
https://colab.research.google.com/drive/1VqTojckSa9DIdAwqP4m93oEMOLSWZ5LJ?usp=sharing