In [1]:
```
import os

current_dir = os.getcwd()
print("Present Working Directory:\n", current_dir, sep="")
```

```
Present Working Directory:
C:\myProject\WinPY\WPy64-3830\notebooks
```

In [2]:
```
# Change the location of current directory to the required locations where datasets resides
os.chdir(r"D:/NYIT (Fall2020-ACADEMICS)/MACHINE LEARNING (DTSC710) [PROFF Xueqing Huang]/PROJECT/dataset")
current_dir = os.getcwd()
print("Changed Directory:\n", current_dir, sep="")

# List all files in the current directory
ls = os.listdir()
print("\nList Of Files:")
for element in range(len(ls)):
    print(ls[element], sep="")

# View the first 20 lines of datasets/transfusion.csv
with open("transfusion.csv") as mysecond_dataset:
    head = [next(mysecond_dataset) for x in range(20)]
print("\nFirst 20 Lines of Dataset transfusion.csv:\n", str(head).strip("[]"), sep="")
```

```
Changed Directory:
D:\NYIT (Fall2020-ACADEMICS)\MACHINE LEARNING (DTSC710) [PROFF Xueqing Huang]\PROJECT\dataset

List Of Files:
DataSet Link.txt
pre-requisite info.txt
transfusion.csv

First 20 Lines of Dataset transfusion.csv:
'Recency (months),Frequency (times),Monetary (c.c. blood),Time (months),"whether he/she donated blood in Mar
ch 2007"\n', '2 ,50,12500,98 ,1\n', '0 ,13,3250,28 ,1\n', '1 ,16,4000,35 ,1\n', '2 ,20,5000,45 ,1\n', '1 ,2
4,6000,77 ,0\n', '4 ,4,1000,4 ,0\n', '2 ,7,1750,14 ,1\n', '1 ,12,3000,35 ,0\n', '2 ,9,2250,22 ,1\n', '5 ,46,
11500,98 ,1\n', '4 ,23,5750,58 ,0\n', '0 ,3,750,4 ,0\n', '2 ,10,2500,28 ,1\n', '1 ,13,3250,47 ,0\n', '2 ,6,1
500,15 ,1\n', '2 ,5,1250,11 ,1\n', '2 ,14,3500,48 ,1\n', '2 ,15,3750,49 ,1\n', '2 ,6,1500,15 ,1\n'
```

In [3]:

```
# Import the `pandas` module as "pd"
import pandas as pd

# Read in `transfusion.csv`
transfusion = pd.read_csv("D:/NYIT (Fall2020-ACADEMICS)/MACHINE LEARNING (DTSC710) [PROFF Xueqing Huang]/PROJE

# Save the number of rows columns as a tuple
rows_and_cols = transfusion.shape
print("Brief Description Of Dataset transfusion.csv Stored In DataFrame Object transfusion:\n")
print('There are {} rows and {} columns.\n'.format(rows_and_cols[0], rows_and_cols[1]))

# Generate an overview of the DataFrame
transfusion_information = transfusion.info()
print(transfusion_information)

# Check whether there are Nan values or not
print("\nBrief Description Whether DataFrame Object transfusion Consists Any NaN Values:\n", transfusion.isna(

# Display the first five rows of the DataFrame
print("\nFollowing Is The Generic Overview Of DataFrame Object transfusion:\n")
transfusion.head()
```

```
Brief Description Of Dataset transfusion.csv Stored In DataFrame Object transfusion:

There are 748 rows and 5 columns.

<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
None

Brief Description Whether DataFrame Object transfusion Consists Any NaN Values:
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

Following Is The Generic Overview Of DataFrame Object transfusion:
```

Out[3]:

	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

In [4]:
```
# Compute the summary statistics of all columns in the `transfusion` DataFrame
sum_stat_transfusion = transfusion.describe()
print("Summary Statistics Of DataFrame Object transfusion:")
sum_stat_transfusion
```

```
Summary Statistics Of DataFrame Object transfusion:
```

Out[4]:

	Recency (months)	Frequency (times)	Monetary (c.c. blood)	Time (months)	whether he/she donated blood in March 2007
count	748.000000	748.000000	748.000000	748.000000	748.000000
mean	9.506684	5.514706	1378.676471	34.282086	0.237968
std	8.095396	5.839307	1459.826781	24.376714	0.426124
min	0.000000	1.000000	250.000000	2.000000	0.000000
25%	2.750000	2.000000	500.000000	16.000000	0.000000
50%	7.000000	4.000000	1000.000000	28.000000	0.000000
75%	14.000000	7.000000	1750.000000	50.000000	0.000000
max	74.000000	50.000000	12500.000000	98.000000	1.000000

In [5]:
```
# Rename "whether he/she donated blood in March 2007" column as "'"target' for brevity
transfusion.rename(columns={'whether he/she donated blood in March 2007': 'target'}, inplace=True)

# Display the first five rows of the DataFrame
print("\nFollowing Is The Generic Overview Of DataFrame Object transfusion:\n")
transfusion.head()
```

```
Following Is The Generic Overview Of DataFrame Object transfusion:
```

Out[5]:

	Recency (months)	Frequency (times)	Monetary (c.c. blood)	Time (months)	target
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

In [6]:
```
# Compute the summary statistics of all columns in the `transfusion` DataFrame
sum_stat_transfusion = transfusion.describe()
print("Summary Statistics Of DataFrame Object transfusion:")
sum_stat_transfusion
```

```
Summary Statistics Of DataFrame Object transfusion:
```

Out[6]:

	Recency (months)	Frequency (times)	Monetary (c.c. blood)	Time (months)	target
count	748.000000	748.000000	748.000000	748.000000	748.000000
mean	9.506684	5.514706	1378.676471	34.282086	0.237968
std	8.095396	5.839307	1459.826781	24.376714	0.426124
min	0.000000	1.000000	250.000000	2.000000	0.000000
25%	2.750000	2.000000	500.000000	16.000000	0.000000
50%	7.000000	4.000000	1000.000000	28.000000	0.000000
75%	14.000000	7.000000	1750.000000	50.000000	0.000000
max	74.000000	50.000000	12500.000000	98.000000	1.000000

In [7]:
```
# Print target incidence proportions, rounding output to 3 decimal places
transfusion.target.value_counts(normalize=True).round(3)
```

Out[7]:
```
0    0.762
1    0.238
Name: target, dtype: float64
```

In [8]:
```
# 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(5)
```

Out[8]:

	Recency (months)	Frequency (times)	Monetary (c.c. blood)	Time (months)
334	16	2	500	16
99	5	7	1750	26
116	2	7	1750	46
661	16	2	500	16
154	2	1	250	2

In [9]:
```
# Import TPOTClassifier
from tpot import TPOTClassifier
# Import roc_auc_score
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: 0.7422459184429089

Generation 5 - Current best internal CV score: 0.7456308339276876

Best pipeline: MultinomialNB(Normalizer(input_matrix, norm=l2), alpha=0.001, fit_prior=True)

AUC score: 0.7637

Best pipeline steps:
1. Normalizer()
2. MultinomialNB(alpha=0.001)
```

In [10]:
```
# X_train's variance, rounding the output to 3 decimal places
X_train.var().round(3)
```

Out[10]:
```
Recency (months)              66.929
Frequency (times)             33.830
Monetary (c.c. blood)    2114363.700
Time (months)                611.147
dtype: float64
```

In [11]:
```
# 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
X_train_normed.var().round(3)
```

Out[11]:
```
Recency (months)      66.929
Frequency (times)     33.830
Time (months)        611.147
monetary_log           0.837
dtype: float64
```

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

In [13]:
```
# 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[13]:
```
[('logreg', 0.7890972663699937), ('tpot', 0.7637476160203432)]
```