In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
transfusion=pd.read_csv('transfusion.csv.txt')
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

In [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
```

In [4]:
```
transfusion.rename(columns={'whether he/she donated blood in March 2007':'target'},inplace=True)
```

In [5]:
```
transfusion.head(2)
```

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

In [6]:
```
transfusion.target.value_counts(normalize=True).round(3)
```

Out[6]:
```
0    0.762
1    0.238
Name: target, dtype: float64
```

In [7]:
```
 from sklearn.model_selection import train_test_split
   # X_train,X_test,y_train,y_test=train_test_split(transfusion.drop(columns='target'),transfusion.targ
et,test_size=0.25,random_states=42,stratify=transfusion.target)
```

In [8]:
```
X_train,X_test,y_train,y_test=train_test_split(transfusion.drop(columns='target'),transfusion.target,te
st_size=0.25,random_state=42,stratify=transfusion.target)
```

In [9]:
```
X_train.head(2)
```

Out[9]:

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

In [10]:
```
#from tpot import TPOTClassifier
from sklearn.metrics import roc_auc_score
```

In [11]:
```
from tpot import TPOTClassifier
```

```
C:\Users\Sadaquat Hussain\anaconda3\lib\site-packages\tpot\builtins\__init__.py:36: UserWarning: Warn
ing: optional dependency `torch` is not available. - skipping import of NN models.
  warnings.warn("Warning: optional dependency `torch` is not available. - skipping import of NN model
s.")
```

In [12]:
```
tpot=TPOTClassifier(generations=4,population_size=20,verbosity=2,scoring='roc_auc',random_state=42,disa
ble_update_check=True,config_dict='TPOT light')
tpot.fit(X_train,y_train)
tpot_auc_score=roc_auc_score(y_test,tpot.predict_proba(X_test)[:,1]).round(4)
print(f'\nAUC scoren: {tpot_auc_score:.4f}')
print('\nBest pipeline steps:',end='\n')
for idx,(name,transform) in enumerate(tpot.fitted_pipeline_.steps,start=1):
    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

Best pipeline: LogisticRegression(input_matrix, C=25.0, dual=False, penalty=l2)

AUC scoren: 0.7858

Best pipeline steps:
1.LogisticRegression(C=25.0, random_state=42)
```

In [ ]:

In [21]:
```
print(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
```

In [22]:
```
X_train_normed,X_test_normed=X_train.copy(),X_test.copy()
col_to_normalize=X_train_normed.var().idxmax(axis=1)
for df_ in [X_train_normed,X_test_normed]:
    df_['monetary_log']=np.log(df_[col_to_normalize])
    df_.drop(columns=col_to_normalize,inplace=True)
print(X_train_normed.var().round(3).to_string())
```

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

In [23]:
```
from sklearn import linear_model
logreg=linear_model.LogisticRegression(solver='liblinear',random_state=42)
logreg.fit(X_train_normed,y_train)
logreg_auc_score=roc_auc_score(y_test,logreg.predict_proba(X_test_normed)[:,1])
print(f'AUC score:{logreg_auc_score:0.4f}')
```

```
AUC score:0.7891
```

In [24]:
```
from operator import itemgetter
sorted([('tpot',tpot_auc_score.round(4)),('logreg',logreg_auc_score.round(4))],key=itemgetter(1),revers
e=True)
```

Out[24]:
```
[('logreg', 0.7891), ('tpot', 0.7858)]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: