Classifiers_Transfusion_SVG_Python

September 12, 2020

[61]:
```
#Neural network classifier for Transfusion dataset
from numpy import loadtxt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow import keras
from tensorflow.keras import layers
```

[33]:
```
# load the dataset
transfusion_ds = pd.read_csv("C:/Venu/UCI DataSets/transfusion.
 ↪data",delimiter=',')
transfusion_ds
```

[33]:

	Recency (months)	Frequency (times)	Monetary (c.c. blood)	\
0	2	50	12500	
1	0	13	3250	
2	1	16	4000	
3	2	20	5000	
4	1	24	6000	
..	…	…	…	
743	23	2	500	
744	21	2	500	
745	23	3	750	
746	39	1	250	
747	72	1	250	

	Time (months)	whether he/she donated blood in March 2007
0	98	1
1	28	1
2	35	1
3	45	1

```
4               77                                              0
..             ...                                            ...
743             38                                              0
744             52                                              0
745             62                                              0
746             39                                              0
747             72                                              0

[748 rows x 5 columns]
```

[34]:
```
transfusion_ds.
 ↪columns=["Recency","Frequency","Blood_Donated","Time","Donated_Blood"]
transfusion_ds
```

[34]:
```
     Recency  Frequency  Blood_Donated  Time  Donated_Blood
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
..       ...        ...            ...   ...            ...
743       23          2            500    38              0
744       21          2            500    52              0
745       23          3            750    62              0
746       39          1            250    39              0
747       72          1            250    72              0

[748 rows x 5 columns]
```

[35]:
```
X = transfusion_ds.drop("Donated_Blood",axis=1)
X
```

[35]:
```
     Recency  Frequency  Blood_Donated  Time
0          2         50          12500    98
1          0         13           3250    28
2          1         16           4000    35
3          2         20           5000    45
4          1         24           6000    77
..       ...        ...            ...   ...
743       23          2            500    38
744       21          2            500    52
745       23          3            750    62
746       39          1            250    39
747       72          1            250    72

[748 rows x 4 columns]
```

[36]:
```
Y = transfusion_ds["Donated_Blood"]
Y
```

[36]:
```
0      1
1      1
2      1
3      1
4      0
      ..
743    0
744    0
745    0
746    0
747    0
Name: Donated_Blood, Length: 748, dtype: int64
```

[37]:
```
train_X,test_X,train_y,test_y = train_test_split(X,Y,random_state=5)
train_X
```

[37]:

	Recency	Frequency	Blood_Donated	Time
138	9	5	1250	19
689	14	1	250	14
312	12	9	2250	60
207	2	7	1750	76
680	20	14	3500	69
..	…	…	…	…
73	2	2	500	4
400	18	2	500	23
118	1	7	1750	57
701	16	1	250	16
206	2	2	500	16

[561 rows x 4 columns]

[38]:
```
test_y
```

[38]:
```
709    0
704    0
12     1
541    0
163    1
      ..
261    1
692    0
421    0
53     0
441    0
```

```
Name: Donated_Blood, Length: 187, dtype: int64
```

[39]:
```
# define the keras model
model = Sequential()
model.add(Dense(12, activation='relu', input_shape = (4,)))
```

[40]:
```
model.add(Dense(12, input_dim=4, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

[41]:
```
# compile the keras model
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
```

[42]:
```
# fit the keras model on the dataset
model.fit(train_X, train_y, epochs=150, batch_size=10)
```

```
Epoch 1/150
57/57 [==============================] - 0s 4ms/step - loss: 34.5922 - accuracy:
0.7094
Epoch 2/150
57/57 [==============================] - 0s 2ms/step - loss: 3.2427 - accuracy:
0.4456
Epoch 3/150
57/57 [==============================] - 0s 2ms/step - loss: 2.1828 - accuracy:
0.3957
Epoch 4/150
57/57 [==============================] - 0s 4ms/step - loss: 1.5440 - accuracy:
0.4421
Epoch 5/150
57/57 [==============================] - 0s 5ms/step - loss: 0.6998 - accuracy:
0.6488
Epoch 6/150
57/57 [==============================] - 0s 4ms/step - loss: 0.6698 - accuracy:
0.7255
Epoch 7/150
57/57 [==============================] - 0s 3ms/step - loss: 0.6344 - accuracy:
0.7611
Epoch 8/150
57/57 [==============================] - 0s 3ms/step - loss: 0.6510 - accuracy:
0.7594
Epoch 9/150
57/57 [==============================] - 0s 3ms/step - loss: 0.6406 - accuracy:
0.7201
Epoch 10/150
57/57 [==============================] - 0s 3ms/step - loss: 0.6715 - accuracy:
0.7504
```

```
Epoch 11/150
57/57 [==============================] - 0s 3ms/step - loss: 0.6000 - accuracy:
0.7487
Epoch 12/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5802 - accuracy:
0.7558
Epoch 13/150
57/57 [==============================] - 0s 3ms/step - loss: 0.6172 - accuracy:
0.7504
Epoch 14/150
57/57 [==============================] - 0s 3ms/step - loss: 0.6072 - accuracy:
0.7398
Epoch 15/150
57/57 [==============================] - 0s 3ms/step - loss: 0.6193 - accuracy:
0.7362
Epoch 16/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5547 - accuracy:
0.7540
Epoch 17/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5825 - accuracy:
0.7558
Epoch 18/150
57/57 [==============================] - 0s 5ms/step - loss: 0.5913 - accuracy:
0.7540
Epoch 19/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5599 - accuracy:
0.7558
Epoch 20/150
57/57 [==============================] - 0s 4ms/step - loss: 0.6244 - accuracy:
0.7362
Epoch 21/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5838 - accuracy:
0.7504
Epoch 22/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5578 - accuracy:
0.7701
Epoch 23/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5724 - accuracy:
0.7558
Epoch 24/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5410 - accuracy:
0.7540
Epoch 25/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5543 - accuracy:
0.7576
Epoch 26/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5340 - accuracy:
0.7629
```

```
Epoch 27/150
57/57 [==============================] - 0s 3ms/step - loss: 0.6014 - accuracy:
0.7540
Epoch 28/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5348 - accuracy:
0.7665
Epoch 29/150
57/57 [==============================] - 0s 5ms/step - loss: 0.5976 - accuracy:
0.7451
Epoch 30/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5746 - accuracy:
0.7291
Epoch 31/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5455 - accuracy:
0.7629
Epoch 32/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5413 - accuracy:
0.7576
Epoch 33/150
57/57 [==============================] - 0s 3ms/step - loss: 0.6588 - accuracy:
0.7433
Epoch 34/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5231 - accuracy:
0.7576
Epoch 35/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5281 - accuracy:
0.7611
Epoch 36/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5580 - accuracy:
0.7522
Epoch 37/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5989 - accuracy:
0.7380
Epoch 38/150
57/57 [==============================] - 0s 4ms/step - loss: 0.6172 - accuracy:
0.7522
Epoch 39/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5594 - accuracy:
0.7540
Epoch 40/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5399 - accuracy:
0.7558
Epoch 41/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5666 - accuracy:
0.7504
Epoch 42/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5626 - accuracy:
0.7558
```

```
Epoch 43/150
57/57 [==============================] - 0s 4ms/step - loss: 0.6863 - accuracy:
0.7184
Epoch 44/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5442 - accuracy:
0.7683
Epoch 45/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5614 - accuracy:
0.7522
Epoch 46/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5207 - accuracy:
0.7718
Epoch 47/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5464 - accuracy:
0.7576
Epoch 48/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5073 - accuracy:
0.7665
Epoch 49/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5431 - accuracy:
0.7629
Epoch 50/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5121 - accuracy:
0.7665
Epoch 51/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5347 - accuracy:
0.7736
Epoch 52/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5304 - accuracy:
0.7540
Epoch 53/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5550 - accuracy:
0.7558
Epoch 54/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5776 - accuracy:
0.7540
Epoch 55/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5089 - accuracy:
0.7665
Epoch 56/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5731 - accuracy:
0.7522
Epoch 57/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5206 - accuracy:
0.7558
Epoch 58/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5476 - accuracy:
0.7522
```

```
Epoch 59/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5239 - accuracy:
0.7683
Epoch 60/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5398 - accuracy:
0.7683
Epoch 61/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5191 - accuracy:
0.7594
Epoch 62/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5167 - accuracy:
0.7594
Epoch 63/150
57/57 [==============================] - 0s 2ms/step - loss: 0.5025 - accuracy:
0.7611
Epoch 64/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5435 - accuracy:
0.7647
Epoch 65/150
57/57 [==============================] - 0s 2ms/step - loss: 0.5355 - accuracy:
0.7594
Epoch 66/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5293 - accuracy:
0.7647
Epoch 67/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5349 - accuracy:
0.7647
Epoch 68/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5403 - accuracy:
0.7701
Epoch 69/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5165 - accuracy:
0.7558
Epoch 70/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5103 - accuracy:
0.7629
Epoch 71/150
57/57 [==============================] - 0s 5ms/step - loss: 0.5295 - accuracy:
0.7576
Epoch 72/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5157 - accuracy:
0.7504
Epoch 73/150
57/57 [==============================] - 0s 6ms/step - loss: 0.5120 - accuracy:
0.7629
Epoch 74/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5295 - accuracy:
0.7611
```

```
Epoch 75/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5080 - accuracy:
0.7790
Epoch 76/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5313 - accuracy:
0.7647
Epoch 77/150
57/57 [==============================] - 0s 3ms/step - loss: 1.3439 - accuracy:
0.7504
Epoch 78/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5586 - accuracy:
0.7647
Epoch 79/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5308 - accuracy:
0.7647
Epoch 80/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5111 - accuracy:
0.7629
Epoch 81/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5619 - accuracy:
0.7665
Epoch 82/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5504 - accuracy:
0.7647
Epoch 83/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5272 - accuracy:
0.7647
Epoch 84/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5225 - accuracy:
0.7647
Epoch 85/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5101 - accuracy:
0.7629
Epoch 86/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4999 - accuracy:
0.7665
Epoch 87/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5052 - accuracy:
0.7611
Epoch 88/150
57/57 [==============================] - 0s 4ms/step - loss: 0.4942 - accuracy:
0.7683
Epoch 89/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5152 - accuracy:
0.7647
Epoch 90/150
57/57 [==============================] - 0s 5ms/step - loss: 0.5050 - accuracy:
0.7665
```

```
Epoch 91/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5064 - accuracy:
0.7629
Epoch 92/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5317 - accuracy:
0.7647
Epoch 93/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5068 - accuracy:
0.7647
Epoch 94/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4952 - accuracy:
0.7647
Epoch 95/150
57/57 [==============================] - 0s 4ms/step - loss: 0.4904 - accuracy:
0.7665
Epoch 96/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5100 - accuracy:
0.7647
Epoch 97/150
57/57 [==============================] - 0s 4ms/step - loss: 0.4987 - accuracy:
0.7629
Epoch 98/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5003 - accuracy:
0.7611
Epoch 99/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4898 - accuracy:
0.7718
Epoch 100/150
57/57 [==============================] - 0s 5ms/step - loss: 0.5012 - accuracy:
0.7558
Epoch 101/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5067 - accuracy:
0.7683
Epoch 102/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4992 - accuracy:
0.7611
Epoch 103/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5102 - accuracy:
0.7683
Epoch 104/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5043 - accuracy:
0.7683
Epoch 105/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5105 - accuracy:
0.7629
Epoch 106/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4947 - accuracy:
0.7647
```

```
Epoch 107/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4974 - accuracy:
0.7576
Epoch 108/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5017 - accuracy:
0.7647
Epoch 109/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4965 - accuracy:
0.7629
Epoch 110/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5014 - accuracy:
0.7665
Epoch 111/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5005 - accuracy:
0.7647
Epoch 112/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4962 - accuracy:
0.7647
Epoch 113/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4920 - accuracy:
0.7647
Epoch 114/150
57/57 [==============================] - 0s 2ms/step - loss: 0.5177 - accuracy:
0.7594
Epoch 115/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4978 - accuracy:
0.7647
Epoch 116/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4982 - accuracy:
0.7611
Epoch 117/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5153 - accuracy:
0.7558
Epoch 118/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5036 - accuracy:
0.7594
Epoch 119/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5034 - accuracy:
0.7594
Epoch 120/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5057 - accuracy:
0.7647
Epoch 121/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5098 - accuracy:
0.7647
Epoch 122/150
57/57 [==============================] - 0s 4ms/step - loss: 0.4993 - accuracy:
0.7665
```

```
Epoch 123/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5044 - accuracy:
0.7647
Epoch 124/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4976 - accuracy:
0.7647
Epoch 125/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5109 - accuracy:
0.7611
Epoch 126/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4955 - accuracy:
0.7665
Epoch 127/150
57/57 [==============================] - 0s 4ms/step - loss: 0.4958 - accuracy:
0.7611
Epoch 128/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5047 - accuracy:
0.7594
Epoch 129/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5024 - accuracy:
0.7647
Epoch 130/150
57/57 [==============================] - 0s 4ms/step - loss: 0.4959 - accuracy:
0.7647
Epoch 131/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5106 - accuracy:
0.7594
Epoch 132/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5008 - accuracy:
0.7629
Epoch 133/150
57/57 [==============================] - ETA: 0s - loss: 0.4981 - accuracy: 0.76
- 0s 3ms/step - loss: 0.4981 - accuracy: 0.7611
Epoch 134/150
57/57 [==============================] - 0s 5ms/step - loss: 0.5096 - accuracy:
0.7665
Epoch 135/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5039 - accuracy:
0.7665
Epoch 136/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4968 - accuracy:
0.7683
Epoch 137/150
57/57 [==============================] - 0s 4ms/step - loss: 0.4969 - accuracy:
0.7683
Epoch 138/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5078 - accuracy:
0.7576
```

```
Epoch 139/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5011 - accuracy:
0.7647
Epoch 140/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5100 - accuracy:
0.7683
Epoch 141/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5025 - accuracy:
0.7665
Epoch 142/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4982 - accuracy:
0.7594
Epoch 143/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5038 - accuracy:
0.7665
Epoch 144/150
57/57 [==============================] - 0s 3ms/step - loss: 0.5061 - accuracy:
0.7683
Epoch 145/150
57/57 [==============================] - 0s 4ms/step - loss: 0.4922 - accuracy:
0.7629
Epoch 146/150
57/57 [==============================] - 0s 4ms/step - loss: 0.4964 - accuracy:
0.7647
Epoch 147/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4999 - accuracy:
0.7683
Epoch 148/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4995 - accuracy:
0.7665
Epoch 149/150
57/57 [==============================] - 0s 4ms/step - loss: 0.5011 - accuracy:
0.7647
Epoch 150/150
57/57 [==============================] - 0s 3ms/step - loss: 0.4992 - accuracy:
0.7647
```

[42]:
```
<tensorflow.python.keras.callbacks.History at 0x22e46d3a5f8>
```

[43]:
```
# evaluate the keras model
_, accuracy = model.evaluate(test_X, test_y)
print('Accuracy: %.2f' % (accuracy*100))
```

```
6/6 [==============================] - 0s 3ms/step - loss: 0.4628 - accuracy:
0.7594
Accuracy: 75.94
```

[63]:
```
# make probability predictions with the model
predictions = model.predict(X)
```

[64]:
```
# make class predictions with the model
predictions_class = model.predict_classes(test_X)
```

[65]:
```
print("Confusion Matrix: \n",confusion_matrix(test_y,predictions_class))
```

```
Confusion Matrix:
 [[141   0]
 [ 45   1]]
```

[47]:
```
print("Classification Report:␣
 ↪\n",classification_report(test_y,predictions_class))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.76      1.00      0.86       141
           1       1.00      0.02      0.04        46

    accuracy                           0.76       187
   macro avg       0.88      0.51      0.45       187
weighted avg       0.82      0.76      0.66       187
```

[48]:
```
#Logistic Regression Classier
transfusion_logit = LogisticRegression()
```

[49]:
```
transfusion_logit = transfusion_logit.fit(train_X,train_y)
```

[50]:
```
transfusion_logit_pred = transfusion_logit.predict(test_X)
```

[51]:
```
transfusion_logit_pred
```

[51]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

[52]:
```
transfusion_logit.score(test_X,test_y)
```

[52]: 0.786096256684492

[53]:
```
print("Confusion Matrix: \n",confusion_matrix(test_y,transfusion_logit_pred))
print("Classification Report:␣
 ↪\n",classification_report(test_y,transfusion_logit_pred))
```

```
Confusion Matrix:
 [[140   1]
 [ 39   7]]
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.99      0.88       141
           1       0.88      0.15      0.26        46

    accuracy                           0.79       187
   macro avg       0.83      0.57      0.57       187
weighted avg       0.80      0.79      0.72       187
```

[54]:
```
#Support Vector Machine Classifier
transfusion_svm = svm.LinearSVC(C=0.0001, tol = 1e-2, max_iter=1000000)
transfusion_svm = transfusion_svm.fit(train_X,train_y)
```

[55]:
```
transfusion_svm_pred = transfusion_svm.predict(test_X)
transfusion_svm.score(test_X,test_y)
```

[55]: 0.7807486631016043

[56]:
```
print("Confusion Matrix: \n",confusion_matrix(test_y,transfusion_svm_pred))
print("Classification Report:␣
 ↪\n",classification_report(test_y,transfusion_svm_pred))
```

```
Confusion Matrix:
 [[140   1]
 [ 40   6]]
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.99      0.87       141
           1       0.86      0.13      0.23        46

    accuracy                           0.78       187
   macro avg       0.82      0.56      0.55       187
weighted avg       0.80      0.78      0.71       187
```

[57]:
```
#KNN Classifier with K=5
transfusion_knn = KNeighborsClassifier()
transfusion_knn = transfusion_knn.fit(train_X,train_y)
```

[58]:
```
transfusion_knn_pred = transfusion_knn.predict(test_X)
```

[59]:
```
print("Confusion Matrix: \n",confusion_matrix(test_y,transfusion_knn_pred))
print("Classification Report:␣
 ↪\n",classification_report(test_y,transfusion_knn_pred))
```

```
Confusion Matrix:
 [[128  13]
 [ 33  13]]
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.91      0.85       141
           1       0.50      0.28      0.36        46

    accuracy                           0.75       187
   macro avg       0.65      0.60      0.60       187
weighted avg       0.72      0.75      0.73       187
```

[60]:
```
print('Accuracy using K-Nearest Neighbourhood:',transfusion_knn.
 ↪score(test_X,test_y)*100)
print('Accuracy using Support Vector Machine:',transfusion_svm.
 ↪score(test_X,test_y)*100)
print('Accuracy using Logistic Regression:',transfusion_logit.
 ↪score(test_X,test_y)*100)
print('Accuracy using Multi-Layered Percentron/Artificial Neural Networks: %.
 ↪2f' % (accuracy*100))
```

```
Accuracy using K-Nearest Neighbourhood: 75.40106951871658
Accuracy using Support Vector Machine: 78.07486631016043
Accuracy using Logistic Regression: 78.6096256684492
Accuracy using Multi-Layered Percentron/Artificial Neural Networks: 75.94
```

[ ]: