In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

df = pd.read_csv('Thesis data.csv')
df.head()
```

Out[1]:

	Res_Age	Resp_weight	Resp_height	Anemia_level	Tetanus_BBirth	HepatitisB_atBirth	ShortBreaths	Married_age	Smoke_at Home	Alcohol	...	HeartDisease	Car
0	38	76.0	1.614	1	3	0	0	26	0	0	...	0	
1	39	43.8	1.406	3	2	1	1	24	1	0	...	0	
2	30	65.5	1.597	0	2	1	0	28	1	0	...	0	
3	32	60.5	1.073	0	3	0	0	24	1	0	...	0	
4	29	80.1	1.617	0	2	0	0	24	0	0	...	0	

5 rows × 24 columns

In [2]:
```
# حذف مقادیر خالی
df.dropna(inplace=True)
# Check for NaN values:
df.isnull().sum()
```

Out[2]:
```
Res_Age               0
Resp_weight           0
Resp_height           0
Anemia_level          0
Tetanus_BBirth        0
HepatitisB_atBirth    0
ShortBreaths          0
Married_age           0
Smoke_at Home         0
Alcohol               0
Diabetes              0
Hypertension          0
RespDisease           0
Thyroid               0
HeartDisease          0
Cancer                0
Kidney                0
Donatable             0
DPTB                  0
MMR                   0
Breastfeed_duration   0
B_ChildSex_Male       0
ResidenceType_Urban   0
Transfusion           0
dtype: int64
```

In [3]:
```
# حذف داده‌های تکراری
df = df.drop_duplicates()
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Res_Age', 'Resp_weight', 'Resp_height', 'Anemia_level',
       'Tetanus_BBirth', 'HepatitisB_atBirth', 'ShortBreaths', 'Married_age',
       'Smoke_at Home', 'Alcohol', 'Diabetes', 'Hypertension', 'RespDisease',
       'Thyroid', 'HeartDisease', 'Cancer', 'Kidney', 'Donatable', 'DPTB',
       'MMR', 'Breastfeed_duration', 'B_ChildSex_Male', 'ResidenceType_Urban',
       'Transfusion'],
      dtype='object')
```

In [5]:
```
from sklearn.model_selection import train_test_split

X = df[['Res_Age', 'Resp_weight', 'Resp_height', 'Anemia_level',
       'Tetanus_BBirth', 'HepatitisB_atBirth', 'ShortBreaths', 'Married_age',
       'Smoke_at Home', 'Alcohol', 'Diabetes', 'Hypertension', 'RespDisease',
       'Thyroid', 'HeartDisease', 'Cancer', 'Kidney', 'Donatable', 'DPTB',
       'MMR', 'Breastfeed_duration', 'B_ChildSex_Male'
       ]]
y = df['Transfusion']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [6]:

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=1)

# Feed the training data through the pipeline
knn.fit(X_train, y_train)
```

Out[6]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=1, p=2,
           weights='uniform')
```

In [7]:

```
# Form a prediction set
predictions = knn.predict(X_test)
```

In [8]:

```
# Report the confusion matrix
from sklearn import metrics
print(metrics.confusion_matrix(y_test,predictions))
```

```
[[ 6191  8049]
 [ 6841 23844]]
```

In [9]:

```
# Print a classification report
print(metrics.classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

         neg       0.48      0.43      0.45     14240
         pos       0.75      0.78      0.76     30685

   micro avg       0.67      0.67      0.67     44925
   macro avg       0.61      0.61      0.61     44925
weighted avg       0.66      0.67      0.66     44925
```

In [10]:

```
# Print the overall accuracy
print(metrics.accuracy_score(y_test,predictions))
```

```
0.6685587089593767
```

In [11]:

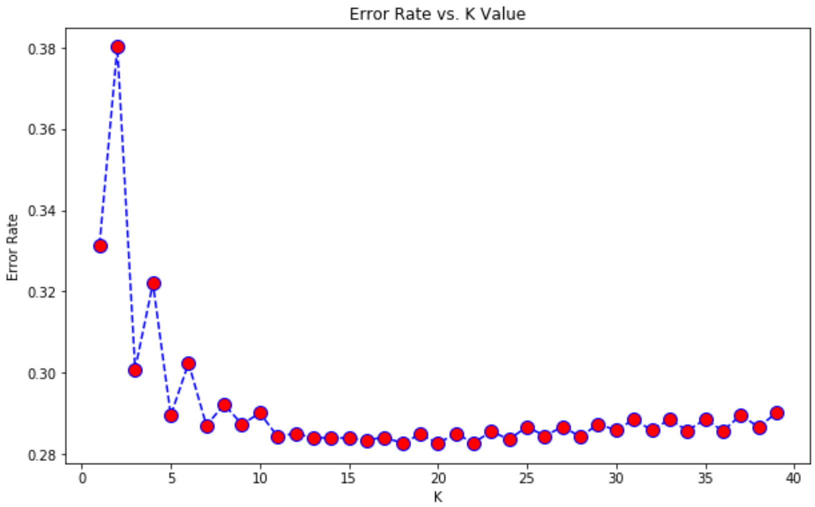
```
error_rate = []

# Will take some time
for i in range(1,40):

    knn = KNeighborsClassifier(n_neighbors=i)

    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

In [12]:

```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

Out[12]:

```
Text(0, 0.5, 'Error Rate')
```



In [13]:

```
knn =  KNeighborsClassifier(n_neighbors=18)


# Feed the training data through the pipeline
knn.fit(X_train, y_train)

pred = knn.predict(X_test)

print(metrics.confusion_matrix(y_test,pred))

print(metrics.classification_report(y_test,pred))

print(metrics.accuracy_score(y_test,pred))
```

```
[[ 3765 10475]
 [ 2224 28461]]
              precision    recall  f1-score   support

         neg       0.63      0.26      0.37     14240
         pos       0.73      0.93      0.82     30685

   micro avg       0.72      0.72      0.72     44925
   macro avg       0.68      0.60      0.59     44925
weighted avg       0.70      0.72      0.68     44925

0.7173288814691152
```

In [ ]: