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

In [2]:
```
df=pd.read_csv('heart.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
```

```
oldpeak     303 non-null float64
slope       303 non-null int64
ca          303 non-null int64
thal        303 non-null int64
target      303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.2 KB
```

In [5]:
```
df.describe()
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	t
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.(
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.(
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.9
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.(
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.5
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.(
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.(
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.(

In [6]:
```
from sklearn.preprocessing import StandardScaler
```

In [7]:
```
std=StandardScaler()
```

In [8]:
```
std.fit(df.drop('target',axis=1))
```

Out[8]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [9]:
```
a=std.transform(df.drop('target',axis=1))
a
```

Out[9]:
```
array([[ 0.9521966 ,  0.68100522,  1.97312292, ..., -2.27457861,
        -0.71442887, -2.14887271],
       [-1.91531289,  0.68100522,  1.00257707, ..., -2.27457861,
        -0.71442887, -0.51292188],
       [-1.47415758, -1.46841752,  0.03203122, ...,  0.97635214,
        -0.71442887, -0.51292188],
       ...,
       [ 1.50364073,  0.68100522, -0.93851463, ..., -0.64911323,
         1.24459328,  1.12302895],
       [ 0.29046364,  0.68100522, -0.93851463, ..., -0.64911323,
         0.26508221,  1.12302895],
       [ 0.29046364, -1.46841752,  0.03203122, ..., -0.64911323,
         0.26508221, -0.51292188]])
```

In [10]:
```
df_h=pd.DataFrame(a,columns=df.columns[:-1])
```

In [11]:
```
df_h.head()
```

Out[11]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang
0	0.952197	0.681005	1.973123	0.763956	-0.256334	2.394438	-1.005832	0.015443	-0.696631
1	-1.915313	0.681005	1.002577	-0.092738	0.072199	-0.417635	0.898962	1.633471	-0.696631
2	-1.474158	-1.468418	0.032031	-0.092738	-0.816773	-0.417635	-1.005832	0.977514	-0.696631
3	0.180175	0.681005	0.032031	-0.663867	-0.198357	-0.417635	0.898962	1.239897	-0.696631
4	0.290464	-1.468418	-0.938515	-0.663867	2.082050	-0.417635	0.898962	0.583939	1.435481

In [12]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [13]:
```
knn=KNeighborsClassifier(n_neighbors=2)
```

In [14]:
```
from sklearn.cross_validation import train_test_split
```

```
C:\Users\Dell\Anaconda3\lib\site-packages\sklearn\cross_validation.py:4
1: DeprecationWarning: This module was deprecated in version 0.18 in fa
```

```
vor of the model_selection module into which all the refactored classes
and functions are moved. Also note that the interface of the new CV ite
rators are different from that of this module. This module will be remo
ved in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [15]:
```
df.columns
```

Out[15]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thala
ch',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [16]:
```
X=df_h[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thal
ach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal']]
y=df['target']
```

In [17]:
```
 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
4, random_state=101)
```

In [18]:
```
X=df_h
y=df['target']
```

In [19]:
```
knn.fit(X_train,y_train)
```

Out[19]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
                     metric_params=None, n_jobs=1, n_neighbors=2, p=2,
                     weights='uniform')
```

In [20]:
```
predict=knn.predict(X_test)
```

In [21]:
```
from sklearn.metrics import classification_report
```

In [22]:
```
print(classification_report(y_test,predict))
```

```
                precision    recall  f1-score   support

             0       0.77      0.88      0.82        60
             1       0.87      0.74      0.80        62

avg / total       0.82      0.81      0.81       122
```

In [23]:
```
from sklearn.metrics import confusion_matrix
```

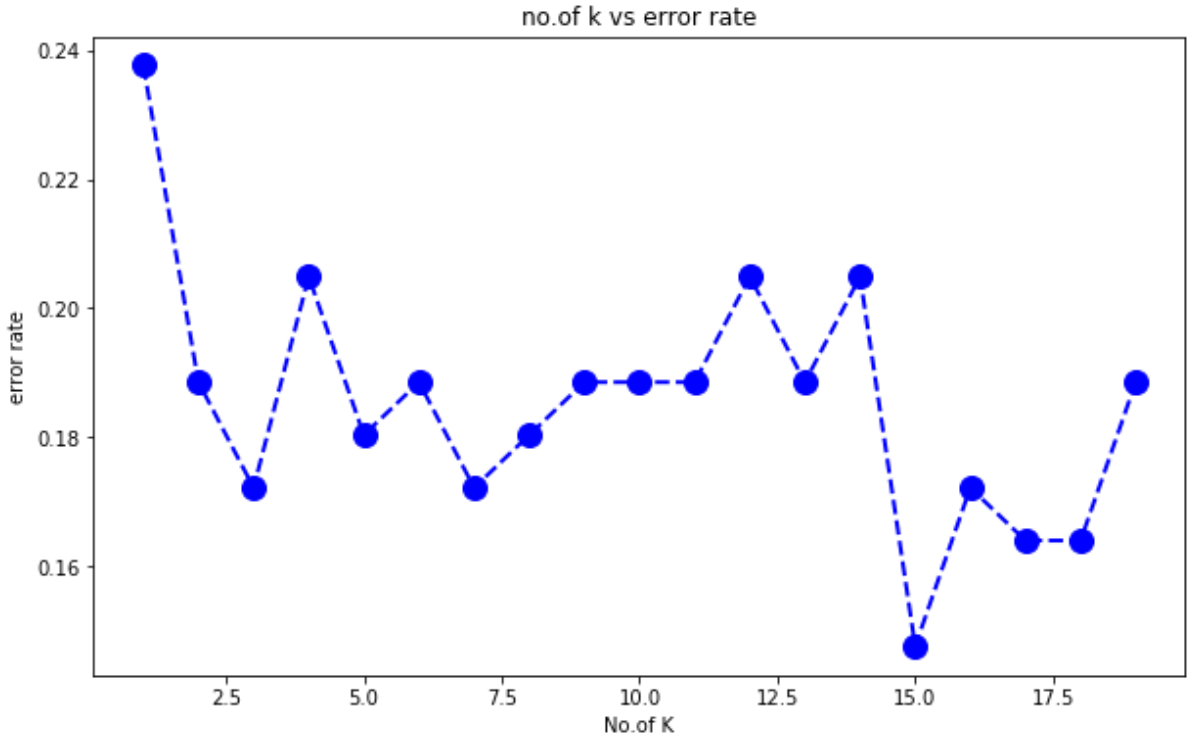
In [27]:
```
print(confusion_matrix(y_test,predict))
```

```
[[53  7]
 [16 46]]
```

In [39]:
```
error_rate=[]
for i in range(1,20):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    predict_i=knn.predict(X_test)
    error_rate.append(np.mean(predict_i != y_test))
```

In [44]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,20),error_rate,color='blue', marker='o', linestyle='da
shed',
         linewidth=2, markersize=12)
plt.xlabel('No.of K')
plt.ylabel('error rate')
plt.title('no.of k vs error rate')
```

Out[44]:
```
Text(0.5,1,'no.of k vs error rate')
```



In [24]:
```
knn=KNeighborsClassifier(n_neighbors=15)
```

In [25]:
```
knn.fit(X_train,y_train)
```

Out[25]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=1, n_neighbors=15, p=2,
                     weights='uniform')
```

In [26]:
```
predic=knn.predict(X_test)
```

In [27]:
```
print(classification_report(y_test,predic))
```

```
              precision    recall  f1-score   support
```

```
                  0       0.96      0.73      0.83        60
                  1       0.79      0.97      0.87        62

avg / total       0.87      0.85      0.85       122
```

In [28]:
```
print(confusion_matrix(y_test,predict))
```

```
[[53  7]
 [16 46]]
```

In [29]:
```
# what if i take k=7
```

In [30]:
```
knn=KNeighborsClassifier(n_neighbors=7)
```

In [31]:
```
knn.fit(X_train,y_train)
```

Out[31]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
           metric_params=None, n_jobs=1, n_neighbors=7, p=2,
           weights='uniform')
```

In [32]:
```
predictions=knn.predict(X_test)
```

In [34]:
```
print(classification_report(y_test,predictions))
```

```
             precision    recall  f1-score   support

          0       0.84      0.80      0.82        60
          1       0.82      0.85      0.83        62

avg / total       0.83      0.83      0.83       122
```

In [37]:
```
print(confusion_matrix(y_test,predictions))
```

```
[[48 12]
```

```
[ 9 53]]
```

In [36]:
```
# result is  better with k=7
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: