In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.rcParams["figure.figsize"] = (10, 20)
import mpld3
mpld3.enable_notebook()
```

In [3]:

```
df = pd.read_csv('heart.csv')
```

In [4]:

```
df.head()
```

Out[4]:

| | 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 [5]:

```
df.isnull().sum()
```

Out[5]:

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [6]:

```
df.shape
```

Out[6]:

```
(303, 14)
```

In [7]:

```
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: 33.3 KB
```

In [8]:

```
df.head(2)
```

Out[8]:

| | 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 |

In [9]:

```
df['sex'].value_counts()
```

Out[9]:

```
1    207
0     96
Name: sex, dtype: int64
```

In [10]:

```
df['cp'].value_counts()
```

Out[10]:

```
0    143
2     87
1     50
3     23
Name: cp, dtype: int64
```

In [11]:

```
df['fbs'].value_counts()
```

Out[11]:

```
0    258
1     45
Name: fbs, dtype: int64
```

In [12]:

```
df['restecg'].value_counts()
```

Out[12]:

```
1    152
0    147
2      4
Name: restecg, dtype: int64
```

In [13]:

```
df['exang'].value_counts()
```

Out[13]:

```
0    204
1     99
Name: exang, dtype: int64
```

In [14]:

```
df['slope'].value_counts()
```

Out[14]:

```
2    142
1    140
0     21
Name: slope, dtype: int64
```

In [15]:

```
df['ca'].value_counts()
```

Out[15]:

```
0    175
1     65
2     38
3     20
4      5
Name: ca, dtype: int64
```

In [16]:

```
df['thal'].value_counts()
```

Out[16]:

```
2    166
3    117
1     18
0      2
Name: thal, dtype: int64
```

In [17]:

```
data = pd.get_dummies(data = df, columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope'
```

In [22]:

```
df.head(2)
```

Out[22]:

| | 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 |

In [23]:

```
pd.set_option('display.max_columns', None)
data.head()
```

Out[23]:

| | age | trestbps | chol | thalach | oldpeak | target | sex_1 | cp_1 | cp_2 | cp_3 | fbs_1 | restecg_1 | r |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 63 | 145 | 233 | 150 | 2.3 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | |
| 1 | 37 | 130 | 250 | 187 | 3.5 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | |
| 2 | 41 | 130 | 204 | 172 | 1.4 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | |
| 3 | 56 | 120 | 236 | 178 | 0.8 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | |
| 4 | 57 | 120 | 354 | 163 | 0.6 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | |

In [26]:

```
from sklearn.preprocessing import StandardScaler
```

In [27]:

```
scale = StandardScaler()
```

In [29]:

```
columns_to_scale = ['trestbps', 'chol', 'thalach', 'oldpeak']
```

In [30]:

```
data[columns_to_scale] = scale.fit_transform(data[columns_to_scale])
```

In [31]:

```
data.head()
```

Out[31]:

| | age | trestbps | chol | thalach | oldpeak | target | sex_1 | cp_1 | cp_2 | cp_3 | fbs_1 | res |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 63 | 0.763956 | -0.256334 | 0.015443 | 1.087338 | 1 | 1 | 0 | 0 | 1 | 1 | |
| 1 | 37 | -0.092738 | 0.072199 | 1.633471 | 2.122573 | 1 | 1 | 0 | 1 | 0 | 0 | |
| 2 | 41 | -0.092738 | -0.816773 | 0.977514 | 0.310912 | 1 | 0 | 1 | 0 | 0 | 0 | |
| 3 | 56 | -0.663867 | -0.198357 | 1.239897 | -0.206705 | 1 | 1 | 1 | 0 | 0 | 0 | |
| 4 | 57 | -0.663867 | 2.082050 | 0.583939 | -0.379244 | 1 | 0 | 0 | 0 | 0 | 0 | |

In [34]:

```
X = data.drop(['target'], axis = 1)
y = data['target']
```

In [35]:

```
from sklearn.model_selection import train_test_split
```

In [36]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [38]:

```
from sklearn.linear_model import LogisticRegression
```

In [39]:

```
model = LogisticRegression()
```

In [40]:

```
model.fit(X_train, y_train)
```

```
C:\Users\SURYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Speci
fy a solver to silence this warning.
  FutureWarning)
```

Out[40]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [41]:

```
model.score(X_train, y_train)
```

Out[41]:

```
0.8677685950413223
```

In [42]:

```
predict = model.predict(X_test)
```

In [43]:

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

In [47]:

```
print(model.score(X_test, y_test)*100)
```

```
90.1639344262295
```

In [45]:

```
print(classification_report(predict, y_test))
```

```
              precision    recall  f1-score   support

           0       0.89      0.89      0.89        27
           1       0.91      0.91      0.91        34

    accuracy                           0.90        61
   macro avg       0.90      0.90      0.90        61
weighted avg       0.90      0.90      0.90        61
```

In [49]:

```
print(accuracy_score(predict, y_test)*100)
```

```
90.1639344262295
```

In [50]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [51]:

```
from sklearn.model_selection import cross_val_score
knn_scores = []
for k in range(1,21):
    knn_classifier = KNeighborsClassifier(n_neighbors = k)
    score=cross_val_score(knn_classifier,X,y,cv=10)
    knn_scores.append(score.mean())
```

In [52]:

```
plt.plot([k for k in range(1, 21)], knn_scores, color = 'red')
for i in range(1,21):
    plt.text(i, knn_scores[i-1], (i, knn_scores[i-1]))
plt.xticks([i for i in range(1, 21)])
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('K Neighbors Classifier scores for different K values')
```

Out[52]:

```
Text(0.5, 1.0, 'K Neighbors Classifier scores for different K values')
```

In [53]:

```
knn = KNeighborsClassifier(n_neighbors = 3)
```

In [54]:

```
knn.fit(X_train, y_train)
```

Out[54]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

In [55]:

```
knn.score(X_train, y_train)
```

Out[55]:

```
0.8347107438016529
```

In [56]:

```
pre = knn.predict(X_test)
```

In [58]:

```
print(knn.score(X_test, y_test)*100)
```

```
81.9672131147541
```

In [59]:

```
print(accuracy_score(pre, y_test)*100)
```

```
81.9672131147541
```

In [ ]:

```

```