In [1]:
```
#import
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

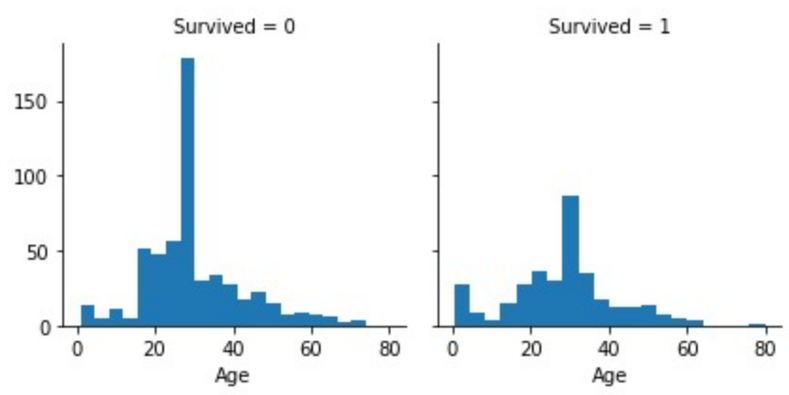
In [2]:
```
#import dataset
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

In [3]:
```
train.fillna(train.mean(), inplace=True)
```

In [4]:
```
test.fillna(test.mean(), inplace=True)
```

In [5]:
```
g = sns.FacetGrid(train, col='Survived')
g.map(plt.hist, 'Age', bins=20)
```

Out[5]: <seaborn.axisgrid.FacetGrid at 0x19f84464b38>



In [6]:
```
train = train.drop(['Name','Ticket', 'Cabin','Embarked'], axis=1)
test = test.drop(['Name','Ticket', 'Cabin','Embarked'], axis=1)
```

In [7]:
```
labelEncoder = LabelEncoder()
labelEncoder.fit(train['Sex'])
labelEncoder.fit(test['Sex'])
train['Sex'] = labelEncoder.transform(train['Sex'])
test['Sex'] = labelEncoder.transform(test['Sex'])
```

In [8]:
```
X = np.array(train.drop(['Survived'], 1).astype(float))
```

In [9]:
```
y = np.array(train['Survived'])
```

In [10]:
```
kmeans = KMeans(n_clusters=2) # You want cluster the passenger records into 2: Survived or Not survived
kmeans.fit(X)
```

Out[10]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [11]:
```
correct = 0
for i in range(len(X)):
    predict_me = np.array(X[i].astype(float))
    predict_me = predict_me.reshape(-1, len(predict_me))
    prediction = kmeans.predict(predict_me)
    if prediction[0] == y[i]:
        correct += 1

print('\nAccuracy is :- {}'.format(correct/len(X)))
```

```
Accuracy is :- 0.49158249158249157
```

In [12]:
```
kmeans = kmeans = KMeans(n_clusters=2, max_iter=600, algorithm = 'auto')
kmeans.fit(X)
```

Out[12]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=600,
    n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [13]:
```
correct = 0
for i in range(len(X)):
    predict_me = np.array(X[i].astype(float))
    predict_me = predict_me.reshape(-1, len(predict_me))
    prediction = kmeans.predict(predict_me)
    if prediction[0] == y[i]:
        correct += 1

print('\nAccuracy is :- {}'.format(correct/len(X)))
```

```
Accuracy is :- 0.5084175084175084
```

In [14]:
```
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

In [15]:
```
kmeans.fit(X_scaled)
```

Out[15]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=600,
    n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [16]:
```
correct = 0
for i in range(len(X)):
    predict_me = np.array(X[i].astype(float))
    predict_me = predict_me.reshape(-1, len(predict_me))
    prediction = kmeans.predict(predict_me)
    if prediction[0] == y[i]:
        correct += 1

print('\nAccuracy is :- {}'.format(correct/len(X)))
```

```
Accuracy is :- 0.6262626262626263
```