In [1]:

```
import matplotlib.pyplot as plt
from matplotlib import style
import numpy as np
from sklearn import preprocessing, cross_validation
import pandas as pd


class K_Means:
    def __init__(self, k=2, tol=0.001, max_iter=300):
        self.k = k
        self.tol = tol
        self.max_iter = max_iter

    def fit(self,data):

        self.centroids = {}

        for i in range(self.k):
            self.centroids[i] = data[i]

        for i in range(self.max_iter):
            self.classifications = {}

            for i in range(self.k):
                self.classifications[i] = []

            for featureset in X:
                distances = [np.linalg.norm(featureset-self.centroids[centroid
]) for centroid in self.centroids]
                classification = distances.index(min(distances))
                self.classifications[classification].append(featureset)

            prev_centroids = dict(self.centroids)

            for classification in self.classifications:
                self.centroids[classification] = np.average(self.classificatio
ns[classification],axis=0)

            optimized = True

            for c in self.centroids:
                original_centroid = prev_centroids[c]
                current_centroid = self.centroids[c]
                if np.sum((current_centroid-original_centroid)/original_centro
id*100.0) > self.tol:
                    #print(np.sum((current_centroid-original_centroid)/origina
l_centroid*100.0))
                    optimized = False

            if optimized:
                break

    def predict(self,data):
```

```
        distances = [np.linalg.norm(data-self.centroids[centroid]) for centroi
d in self.centroids]
        classification = distances.index(min(distances))
        return classification


df = pd.read_excel('titanic.xls')
df.drop(['body','name'], 1, inplace=True)
#df.convert_objects(convert_numeric=True)
#print(df.head())
df.fillna(0,inplace=True)

def handle_non_numerical_data(df):

    # handling non-numerical data: must convert.
    columns = df.columns.values

    for column in columns:
        text_digit_vals = {}
        def convert_to_int(val):
            return text_digit_vals[val]

        #print(column,df[column].dtype)
        if df[column].dtype != np.int64 and df[column].dtype != np.float64:

            column_contents = df[column].values.tolist()
            #finding just the uniques
            unique_elements = set(column_contents)
            # great, found them.
            x = 0
            for unique in unique_elements:
                if unique not in text_digit_vals:
                    # creating dict that contains new
                    # id per unique string
                    text_digit_vals[unique] = x
                    x+=1
            # now we map the new "id" vlaue
            # to replace the string.
            df[column] = list(map(convert_to_int,df[column]))

    return df

df = handle_non_numerical_data(df)
#print(df.head())

# add/remove features just to see impact they have.
df.drop(['ticket','home.dest'], 1, inplace=True)


X = np.array(df.drop(['survived'], 1).astype(float))
X = preprocessing.scale(X)
y = np.array(df['survived'])

#X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, te
st_size=0.5)

clf = K_Means()
clf.fit(X)
```

```
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 = clf.predict(predict_me)
    if prediction == y[i]:
        correct += 1


print(correct/len(X))
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/cross_validation.py
:41: DeprecationWarning: This module was deprecated in version 0.1
8 in favor of the model_selection module into which all the refact
ored classes and functions are moved. Also note that the interface
of the new CV iterators are different from that of this module. Th
is module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)

0.2727272727272727


In [1]:
```

```
# Conclusion: On Given Dataset Using Clustering Algorithm Kmeans Sklearn Libra
ry Inbuild Library Perform Better Than Kmeans From Scrath.
```