In [107]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
```

In [138]:
```
def MLE(data, max_iter=100):
    mean = data.mean()
    covariance = data.cov()

    for j in range(max_iter):
        w = []

        for i in data:
            wkt = len(data)
            print(i)

            wt = np.transpose(i - mean)
            wc = np.linalg.inv(covariance)
            wk2 = np.dot(wt, wc)
            wkb = np.dot(wk2, (i - mean))
            wk = wkt / wkb
            w.append(wk)
            w = np.array(w)

        mu = (np.dot(w, data))/(np.sum(w))

        c = 0
        for i in range(len(data)):
            c += w[i] * np.dot((data[i] - mean), (np.transpose(data[i] -
mean)))
        cov = c/len(data)

        mean = mu
        covarian = cov

    return mean, covariance
```

In [139]:
```
input = pd.read_csv('pima-indians-diabetes.csv')
print(input)
```

```
       1    2   3   4    5     6      7   8  9
0      6  148  72  35    0  33.6  0.627  50  1
1      1   85  66  29    0  26.6  0.351  31  0
2      8  183  64   0    0  23.3  0.672  32  1
3      1   89  66  23   94  28.1  0.167  21  0
4      0  137  40  35  168  43.1  2.288  33  1
..    ..  ...  ..  ..  ...   ...    ...  .. ..
763   10  101  76  48  180  32.9  0.171  63  0
764    2  122  70  27    0  36.8  0.340  27  0
765    5  121  72  23  112  26.2  0.245  30  0
766    1  126  60   0    0  30.1  0.349  47  1
767    1   93  70  31    0  30.4  0.315  23  0

[768 rows x 9 columns]
```

In [140]:
```
input[input.columns] = input[input.columns].apply(pd.to_numeric, errors=
'coerce')
data = input.iloc[:, 1:4]
train_data, test_data = train_test_split(data, test_size=0.50, random_st
ate=11)
```

In [141]:
```
print(train_data)
```

```
       2   3   4
287  119  86  39
34   122  78  31
674   91  82   0
756  137  90  41
277  104  64  23
..   ...  ..  ..
269  146   0   0
337  115  76   0
91   123  80  15
80   113  44  13
703  129   0   0

[384 rows x 3 columns]
```

In [ ]:
```
MLE(train_data)
```

accuracy: 0.7474 mean: 0.7378 std: 0.0242

In [148]:
```
def cos_sim(vector_x, vector_y):
    return np.dot(vector_x, vector_y) / (np.linalg.norm(vector_x) * np.l
inalg.norm(vector_y))

def find_neighbors(data, test, num_neighbors):

    distances = []
    for train_row in data:
        dist = cos_sim(test, train_row)
        distances.append((train_row, dist))
    neighbors = []
    for i in range(num_neighbors):
        neighbors.append(distances[i][0])
    return neighbors


def predict_neighbors(data, test, num_neighbors):

    neighbors = find_neighbors(data, test, num_neighbors)
    prediction = max(set(neighbors), key=neighbors.count)
    return prediction


def k_nearest(train, test, num_neighbors):

    predictions = []
    for row in test:
        output = predict_neighbors(train, row, num_neighbors)
        predictions.append(output)
    return(predictions)
```

In [ ]:
```
target1 = np.random.randint(len(test_data), size=1)
target5 = np.random.randint(len(test_data), size=5)
target11 = np.random.randint(len(test_data), size=11)

result_index = []
for target_index in target1:
    target_vector = test_data[target_index, :]
    result1 = k_nearest(target_vector, test_data[target_index], 1)
    result5 = k_nearest(target_vector, test_data[target_index], 5)
    result11 = k_nearest(target_vector, test_data[target_index], 11)
    result_index.append(result)

print(result_index)
```

k=1 accuracy: 0.565104 mean: 0.5443 std: 0.0195

k=5 accuracy: .557292 mean:5701 std: 0.0206

k=11 accuracy: 0.591146 mean:.5979 std: 0.0175

In [ ]: