PROGRAM

```
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
from sklearn.datasets import load_iris

iris = load_iris()
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()

df['flower'] = iris.target
df.head()

df.drop(['sepal length (cm)', 'sepal width (cm)', 'flower'], axis=1, inplace=True)
print(df.head())

km = KMeans(n_clusters=3)
yp = km.fit_predict(df)
yp

df['cluster'] = yp
df.head(2)

df.cluster.unique()

df1 = df[df.cluster==0]
```

```
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'],color='blue')
plt.scatter(df2['petal length (cm)'],df2['petal width (cm)'],color='green')
plt.scatter(df3['petal length (cm)'],df3['petal width (cm)'],color='yellow')

sse = []

k_rng = range(1,10)
for k in k_rng:
    km = KMeans(n_clusters=k)
    km.fit(df)
    sse.append(km.inertia_)
plt.xlabel('K')

plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```

PROGRAM

```
import numpy as np
np.random.seed(42)

true_prob_A = 0.6
true_prob_B_given_A = np.array([[0.8, 0.2], [0.3, 0.7]])

sample_size = 1000
data_A = np.random.choice([0, 1], size=sample_size, p=[1-true_prob_A,
true_prob_A])
data_B = np.zeros(sample_size)
for i in range(sample_size):
    data_B[i] = np.random.choice([0, 1], p=true_prob_B_given_A[data_A[i]])

def expectation_step(data_A, data_B, prob_A, prob_B_given_A):
    # Compute the probabilities of A and B given A
    prob_B = np.dot(prob_A, prob_B_given_A)

    # Compute the probability of each hidden variable
    hidden_vars = np.zeros((len(data_A), 2))
    for i in range(len(data_A)):
        hidden_vars[i] = prob_A * prob_B_given_A[data_A[i]]
        hidden_vars[i] /= np.sum(hidden_vars[i])

    return hidden_vars

def maximization_step(data_A, data_B, hidden_variables):
    # Update the probability of A
```

```
    prob_A = np.mean(hidden_variables[:, 1])

    # Update the conditional probability of B given A
    prob_B_given_A = np.zeros((2, 2))
    for a in [0, 1]:
        data_B_given_A = data_B[data_A == a]
        prob_B_given_A[a] = [np.mean(data_B_given_A == 0),
np.mean(data_B_given_A == 1)]

    return prob_A, prob_B_given_A

estimated_prob_A = 0.5
estimated_prob_B_given_A = np.array([[0.5, 0.5], [0.5, 0.5]])

num_iterations = 10
for i in range(num_iterations):
    hidden_vars = expectation_step(data_A, data_B, estimated_prob_A,
estimated_prob_B_given_A)
    estimated_prob_A, estimated_prob_B_given_A =
maximization_step(data_A, data_B, hidden_vars)

print("Estimated probability of A:", estimated_prob_A)
print("Estimated conditional probability of B given A:")
print(estimated_prob_B_given_A)
```

PROGRAM

```
import numpy as np
def activation(x):
  return 1 / (1 + np.exp(-x))

weights = np.random.uniform(-1,1,size = (2, 1))

training_inputs = np.array([[0, 0, 1, 1, 0, 1]]).reshape(3, 2)
training_outputs = np.array([[0, 1, 1]]).reshape(3,1)
for i in range(15000):
 dot_product = np.dot(training_inputs, weights)
 output = activation(dot_product)

 temp2 = -(training_outputs - output) * output * (1 - output)
 adj = np.dot(training_inputs.transpose(), temp2)

weights = weights - 0.5 * adj
test_input = np.array([1, 0])

test_output = activation(np.dot(test_input, weights))
print(test_output)
```

PROGRAM

```
from numpy import loadtxt

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
dataset = loadtxt('C:/python/pima-indians-diabetes.csv', delimiter=',')
X = dataset[:,0:8]
y = dataset[:,8]

model = Sequential()

model.add(Dense(12, input_shape=(8,), activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',  optimizer='adam',
metrics=['accuracy'])
model.fit(X, y, epochs=150, batch_size=10)

_, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy*100))

model.fit(X, y, epochs=150, batch_size=10, verbose=0)
_, accuracy = model.evaluate(X, y, verbose=0)
predictions = model.predict(X)
rounded = [round(x[0]) for x in predictions]
predictions = (model.predict(X) > 0.5).astype(int)
```

```
from numpy import loadtxt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

dataset = loadtxt('C:/python/pima-indians-diabetes.csv', delimiter=',')

X = dataset[:,0:8]
y = dataset[:,8]
model = Sequential()
model.add(Dense(12, input_shape=(8,), activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',
metrics=['accuracy'])
model.fit(X, y, epochs=150, batch_size=10, verbose=0)
predictions = (model.predict(X) > 0.5).astype(int)
for i in range(5):
    print('%s => %d (expected %d)' % (X[i].tolist(), predictions[i], y[i]))
```