Ex. NO. 10.b

Date: 01/11/25

A python program to Implement K-means model

Aim:

To implement a python program using a Kmeans Algorithm is a model.

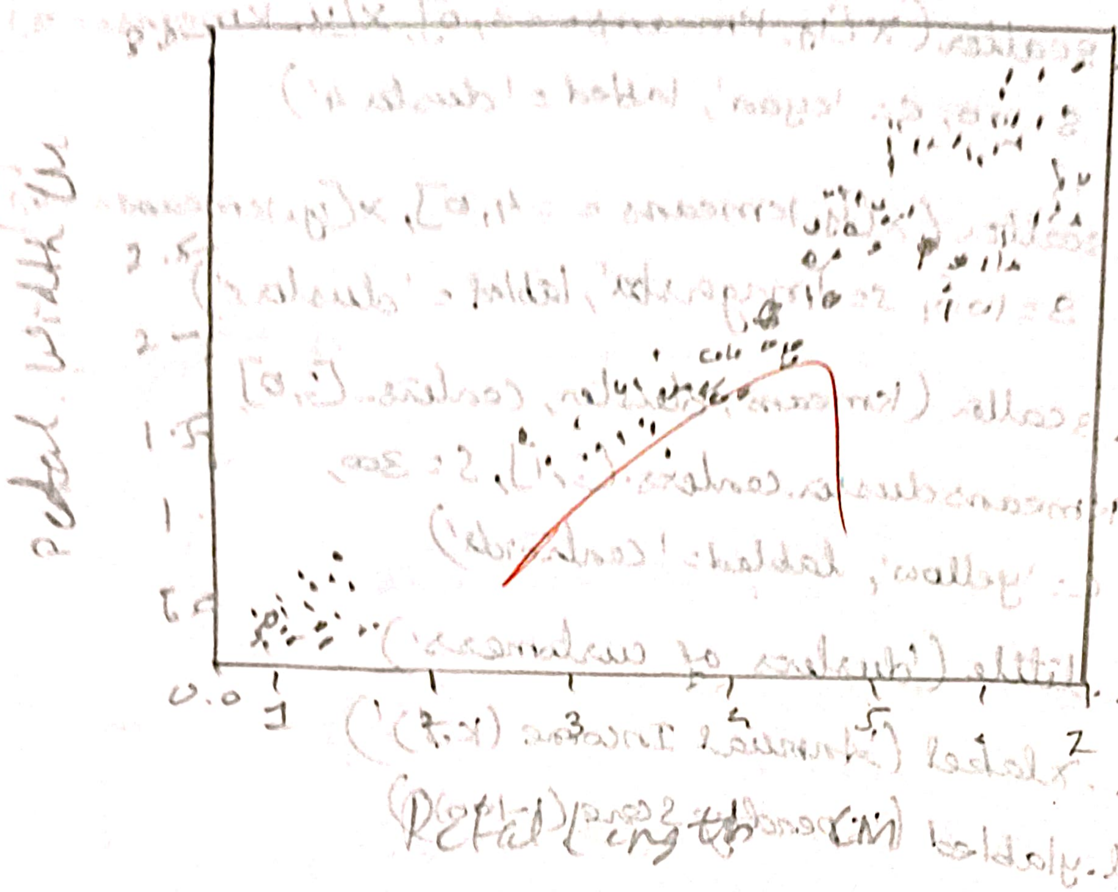
Algorithm:-

1. Import Necessary libraries
2. Load and preprocess data
3. Initialize cluster centers
4. Assign data points to clusters.
5. Update cluster centers
6. Repeat step 4 and 5
7. Plot the clusters

Program:-

```
data = pd.read_csv (.....)
data.head (5)
reg.data = data.iloc [:1,:]
 reg.data.head (5)
shuffle_index = np.random.permutation
                (reg.data.shape [0])

reg.data = reg.data.iloc [shuffle_index]
           reg.data.head (5)
```


Petal Width (cm)
0.5
1
1.5
2
2.5
0.0
1
3
4
5
7
PETAL Length (CM)

```
train_ size = int (reg. data. shape [0] * 07]
train_df = reg_ data. iloc [: train_ size, :]
test _df : reg_ data. Iloc [train_ size :, :]
train = train.df. values
test = test.df. values
 y_ true = test [:, -1]
print ('Train shape: ', train_df. shape]
print ('Test_shape.' test_df_shape)

 from math import sqrt
 def euclidean_distance (x. test , x_ train):
     distance = 0
  for.. in range (len (x_test) - 1):
      distance + = (x. test [i]. x. train [i] ** 2
  return sqrt (distance)

def get-neighboors (x_ test, x_train, num_neighboors):
      distances = []
    date = []
for i is x_ train
      distance append (euclidean_distance (x, test))
      date. append (i)
     distances = np. array (distances)
     data = np. array (data)
     Sortindexes = distances. arg sort ()
    distances data in assending order
     date = data [sort_ indexees)
     return data [: num. neighbours]
def predictions (x_text, x_train, num, neighbors):
    classes = []
neighbors = get.neighbors(x_text, x_train,
                          num_neighbors)
  for i in neighbors:
  classes.append (i [-1])
  predicted = max (classes, key-classes-count)
return predicted.
def predict-classifier (x-text).
    classes = []
   neighbors = get-neighbors (x-Text, reg-data.
                              value, 5)
   for i in neighbors:.
        classes.append (i [-1])
   predicted = max (classes key = classes.count)
   print(predicted).
   return predicted.
def accuracy (y_true, y-pred)):
  num_correct = 0
 for i in range (len (y-true)):
   If y_true[i] = y_pred [i]
     num-correct += 1
 accompany = num_correct / len (y-true)
return accuracy
y-pred = []
for i in test :
    y - pred append (prediction (i, trains))
    y - pred
```

```
accuracy = accuracy (y_true,y_pred)
accuracy
     0.9555555555556
```

Result:-

Thus, the phython program to emplement the kmeans model has been successfully emplemented.