```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call dr
```

```
from google.colab import files
from IPython.display import HTML, display

import numpy as np
import io
import re
from copy import deepcopy

# REQUIRED
testFileName = 'test_4.txt'
trainFileName = 'train_4.txt'
classAttributeIndex = 14
attributesIgnore = [2, 4, 10, 11]

# PARAMETERS
dataSplitRatio = 0



# Function to read a file
def readFile( fileName ):
  with open( fileName, 'r' ) as f:
    lines = f.read().split( '\n' )
    return lines

print("#### FILE DATA ####")
trainData = readFile( trainFileName )
testData = readFile( testFileName )
for line in testData:
  print( line )
```

```
#### FILE DATA ####
24 Private 325596 Assoc-voc 11 Married-civ-spouse Machine-op-inspct Husband White
39 Private 98886 7th-8th 4 Married-civ-spouse Other-service Husband White Male 45
41 Private 266439 HS-grad 9 Married-civ-spouse Machine-op-inspct Husband White Ma
21 Private 200121 HS-grad 9 Never-married Handlers-cleaners Own-child White Male
18 Private 111019 10th 6 Never-married Other-service Own-child White Male 0 0 24
47 Self-emp-inc 308241 HS-grad 9 Married-civ-spouse Sales Wife White Female 0 0 4
57 State-gov 170108 Bachelors 13 Married-civ-spouse Exec-managerial Husband White
54 Self-emp-inc 195904 Assoc-voc 11 Married-civ-spouse Craft-repair Husband White
47 Federal-gov 218325 Assoc-acdm 12 Married-civ-spouse Handlers-cleaners Husband
51 Private 126528 HS-grad 9 Separated Craft-repair Not-in-family White Male 0 0 (
30 Private 210541 HS-grad 9 Divorced Craft-repair Not-in-family White Female 0 0
36 Private 34364 Assoc-acdm 12 Separated Tech-support Not-in-family White Female
51 Self-emp-not-inc 195634 Masters 14 Never-married Exec-managerial Not-in-family
30 Private 201697 Bachelors 13 Never-married Other-service Not-in-family White Ma
33 Private 30612 Some-college 10 Married-civ-spouse Sales Husband White Male 0 0
61 Private 27086 HS-grad 9 Divorced Transport-moving Not-in-family White Male 0 (
27 Private 292883 HS-grad 9 Married-civ-spouse Machine-op-inspct Husband Black Ma
23 Private 194630 HS-grad 9 Separated Machine-op-inspct Own-child White Male 0 0
31 Private 92179 10th 6 Divorced Machine-op-inspct Not-in-family White Male 0 0 4
22 Federal-gov 316438 HS-grad 9 Never-married Prof-specialty Own-child White Male
```

```
# Converting the file data into a 2D array
def tabulateData( data, delimiter = ' ', hasHeader = True ):
  X = []
  for line in data:
    words = line.split(delimiter)
    X.append(words)
  return X

print("#### TABULATED DATA ####")
trainTabulatedData = tabulateData( trainData )
testTabulatedData = tabulateData( testData )
display(HTML(
   '<table><tr>{}</tr></table>'.format(
       '</tr><tr>'.join(
           '<td>{}</td>'.format('</td><td>'.join(str(_) for _ in row)) for row in tes
       )
))
```

```
#### TABULATED DATA ####
```

24	Private	325596	Assoc-voc	11	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	45	United-States
39	Private	98886	7th-8th	4	Married-civ-spouse	Other-service	Husband	White	Male	4508	0	40	Mexico
41	Private	266439	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States
21	Private	200121	HS-grad	9	Never-married	Handlers-cleaners	Own-child	White	Male	0	0	40	United-States
18	Private	111019	10th	6	Never-married	Other-service	Own-child	White	Male	0	0	24	United-States
47	Self-emp-inc	308241	HS-grad	9	Married-civ-spouse	Sales	Wife	White	Female	0	0	40	United-States
57	State-gov	170108	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	1902	40	United-States
54	Self-emp-inc	195904	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40	United-States
47	Federal-gov	218325	Assoc-acdm	12	Married-civ-spouse	Handlers-cleaners	Husband	Asian-Pac-Islander	Male	0	0	40	Philippir
51	Private	126528	HS-grad	9	Separated	Craft-repair	Not-in-family	White	Male	0	0	60	United-States
30	Private	210541	HS-grad	9	Divorced	Craft-repair	Not-in-family	White	Female	0	0	40	United-States
36	Private	34364	Assoc-acdm	12	Separated	Tech-support	Not-in-family	White	Female	0	0	3	United-States
51	Self-emp-not-inc	195634	Masters	14	Never-married	Exec-managerial	Not-in-family	White	Male	10520	0	20	United-States
30	Private	201697	Bachelors	13	Never-married	Other-service	Not-in-family	White	Male	0	0	40	United-States
33	Private	30612	Some-college	10	Married-civ-spouse	Sales	Husband	White	Male	0	0	40	United-States

```
# Removing data points which consists of null values
def preprocessData( tabulatedData, classAttributeIndex, train = True ):
  X = []
  Y_train = [ ]
  requiredLength = len( tabulatedData[0] )
  for dataPoint in tabulatedData:
    if( len(dataPoint) < requiredLength ):
      continue
    # if "none" in dataPoint:
    #   continue
```

```
      X.append( dataPoint[ :requiredLength ] )

    X = np.asanyarray(X)
    if(train is True):
      Y_train = X[:, classAttributeIndex]
      X = np.delete(X, classAttributeIndex, axis = 1)
    return X, Y_train

print("#### PREPROCESSED DATA ####")
X_train, Y_train = preprocessData( trainTabulatedData, classAttributeIndex = classAtt
X_test, Y_test = preprocessData( testTabulatedData, classAttributeIndex = classAttrib
print(X_train[10,:])
# print(Y_train)
```

```
#### PREPROCESSED DATA ####
['23' 'Private' '134446' 'HS-grad' '9' 'Separated' 'Machine-op-inspct'
 'Unmarried' 'Black' 'Male' '0' '0' '54' 'United-States']
```

```
def categorical_distance(ptA, ptB):
  diff = ( ptA == ptB )
  return np.size(diff) - np.sum(diff)

def euclidean_distance(ptA, ptB):
  a = ptA.astype(np.float)
  b = ptB.astype(np.float)
  return (np.sum((a - b)**2)**0.5)

def distance(ptA, ptB, numeric_attributes, categorical_attributes):
  dist = 0
  dist += euclidean_distance(ptA[numeric_attributes], ptB[numeric_attributes])
  dist += categorical_distance(ptA[categorical_attributes], ptB[categorical_attribute
  return dist


def findAttributeTypes(X):
  N, M = np.shape(X)
  i = 0
  dataSet = X[0,:]
  while('?' in dataSet):
    i += 1
    dataSet = X[i, :]
  categorical_attributes = []
  numeric_attributes = []
  array = dataSet

  for i in range(len(array)):
    regex_output = None
    x = re.search('^[A-Za-z]+[-]*', array[i])
    if x is not None:
```

```
      categorical_attributes.append(i)
      continue
    x = re.search('^[0-9]+[.]*[0-9]+$', array[i])
    if x is not None:
      numeric_attributes.append(i)
      continue
    else:
      categorical_attributes.append(i)

  return numeric_attributes, categorical_attributes


# Function to process Data that is removing the columns
def processData( data, removeColumns ):
  data = np.delete( data, removeColumns, axis = 1 )
  numeric_attributes, categorical_attributes = findAttributeTypes( data )
  return data, numeric_attributes, categorical_attributes

X_train, numeric_attributes, categorical_attributes = processData( X_train, attribute
X_test, numeric_attributes2, categorical_attributes2 = processData( X_test, attribute

print(numeric_attributes)
print(categorical_attributes)
print(X_train[0,:])
```

```
[0, 8]
[1, 2, 3, 4, 5, 6, 7, 9]
['65' '?' 'HS-grad' 'Married-civ-spouse' '?' 'Husband' 'White' 'Male' '40'
 'United-States']
```

```
def getMissedDataPoint( data ):
 N, M = np.shape(data)
 for i in range(N):
   point = data[i, :]
   if '?' in point:
     return i
 return -1

def getKNNeighbours(X_train, Y_train, testPoint, k, numeric_attributes, categorical_at
   dist = np.empty((1,3))

   # Finding distance with all the training nodes and storing in dist matrix
   N_train, M = np.shape(X_train)

   for j in range( N_train ):
     trainPoint = X_train[j, :]
     if not '?' in trainPoint:
       temp = np.array([[ j, distance( trainPoint, testPoint, numeric_attributes, cat
       dist = np.append( dist, temp, axis = 0 )
   dist = np.delete(dist, 0, axis = 0)

   # Sorting the distances
```

```
    # Sorting the distances
    dist = dist[dist[:, 1].argsort()]

    # Selecting top K elements as our neighbours
    neighbours = dist[:k, :]

    return neighbours

def fillTrainSet( data, Y, numeric_attributes, categorical_attributes ):
  N, M = np.shape( data )

  # i = getMissedDataPoint( data )
  for i in range(10):
    myPoint = data[i, :]
    missingAttributesIndex = []
    print("-------------------------------")
    print("Data point with missing value:")
    print(data[i, :])
    for j in range( len(myPoint) ):
      if '?' in myPoint[j]:
        missingAttributesIndex.append(j)
    copy_myPoint = deepcopy(myPoint)
    copy_data = deepcopy(data)

    copy_myPoint = np.delete(copy_myPoint, missingAttributesIndex )
    copy_data = np.delete(copy_data, missingAttributesIndex, axis = 1)

    numeric_attributes, categorical_attributes = findAttributeTypes( copy_data )

    myNeighbour = getKNNeighbours( copy_data[10:,:], Y, copy_myPoint, 1, numeric_attri

    for j in range( len(myPoint) ):
      if '?' in myPoint[j]:
        # Adding 10 as we started with 10 training instances
        data[i, j] = data[ int(myNeighbour[0][0]) + 10, j ]
    print("My nearest equivalent:")
    print((data[int(myNeighbour[0][0])+10, :]) )
    print("My updated value:")
    print(data[i, :])
  return data


X_train = fillTrainSet(X_train, Y_train, numeric_attributes, categorical_attributes )
```

```
['64' '?' 'Bachelors' 'Married-civ-spouse' '?' 'Husband' 'White' 'Male'
 '40' 'United-States']
My nearest equivalent:
['64' 'Local-gov' 'HS-grad' 'Married-civ-spouse' 'Adm-clerical' 'Husband'
 'White' 'Male' '40' 'United-States']
My updated value:
['64' 'Local-gov' 'Bachelors' 'Married-civ-spouse' 'Adm-clerical'
 'Husband' 'White' 'Male' '40' 'United-States']
--------------------------------
Data point with missing value:
['24' '?' 'Some-college' 'Never-married' '?' 'Not-in-family' 'White'
 'Female' '38' 'United-States']
My nearest equivalent:
['23' 'Private' 'Some-college' 'Never-married' 'Other-service'
 'Not-in-family' 'Black' 'Male' '30' 'United-States']
My updated value:
['24' 'Private' 'Some-college' 'Never-married' 'Other-service'
 'Not-in-family' 'White' 'Female' '38' 'United-States']
--------------------------------
Data point with missing value:
['19' '?' 'Some-college' 'Never-married' '?' 'Own-child' 'White' 'Female'
 '12' 'United-States']
My nearest equivalent:
['18' 'Private' 'Some-college' 'Never-married' 'Other-service' 'Own-child'
 'White' 'Female' '12' 'United-States']
My updated value:
['19' 'Private' 'Some-college' 'Never-married' 'Other-service' 'Own-child'
 'White' 'Female' '12' 'United-States']
--------------------------------
Data point with missing value:
['58' '?' '11th' 'Married-spouse-absent' '?' 'Not-in-family' 'White'
 'Female' '20' 'United-States']
My nearest equivalent:
['63' 'Private' 'Some-college' 'Married-civ-spouse' 'Sales' 'Husband'
 'White' 'Male' '15' 'United-States']
My updated value:
['58' 'Private' '11th' 'Married-spouse-absent' 'Sales' 'Not-in-family'
 'White' 'Female' '20' 'United-States']
--------------------------------
Data point with missing value:
['28' '?' 'Some-college' 'Married-civ-spouse' '?' 'Other-relative' 'White'
 'Female' '20' 'United-States']
My nearest equivalent:
['23' 'Private' 'HS-grad' 'Never-married' 'Sales' 'Own-child' 'Black'
 'Male' '20' 'United-States']
My updated value:
['28' 'Private' 'Some-college' 'Married-civ-spouse' 'Sales'
 'Other-relative' 'White' 'Female' '20' 'United-States']
--------------------------------
Data point with missing value:
['28' '?' '11th' 'Never-married' '?' 'Unmarried' 'Black' 'Female' '30'
 'United-States']
My nearest equivalent:
['28' 'Private' 'Assoc-voc' 'Never-married' 'Exec-managerial'
 'Not-in-family' 'White' 'Female' '23' 'United-States']
My updated value:
['28' 'Private' '11th' 'Never-married' 'Exec-managerial' 'Unmarried'
 'Black' 'Female' '30' 'United-States']
```

```
 Black   Female   30   United-States ]
```

```
numeric_attributes, categorical_attributes = findAttributeTypes(X_train[10:, :])

def predict( neighbours ):
  # Finding count class
  countClass = dict()
  for neighbourClass in neighbours[:, 2]:
    if( neighbourClass in countClass ):
      countClass[neighbourClass] += 1
    else:
      countClass[neighbourClass] = 1

  # Finding prediction
```

```
    maxCount = -1
    predClass = None
    for countClassKey in countClass:
      if( maxCount < countClass[ countClassKey ] ):
        maxCount = countClass[ countClassKey ]
        predClass = countClassKey

    return predClass

  N_test, M = np.shape(X_test)
  for i in range(N_test):
    testPoint = X_test[i]
    numeric_attributes, categorical_attributes = findAttributeTypes(X_train)
    neighbours = getKNNeighbours(X_train, Y_train, testPoint, 3 , numeric_attributes, c
    print(testPoint)
    print("Predicted Class: " + str(predict(neighbours)))
    print("-------------")
```

```
Predicted Class: <=50K
-------------
['57' 'State-gov' 'Bachelors' 'Married-civ-spouse' 'Exec-managerial'
 'Husband' 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['54' 'Self-emp-inc' 'Assoc-voc' 'Married-civ-spouse' 'Craft-repair'
 'Husband' 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['47' 'Federal-gov' 'Assoc-acdm' 'Married-civ-spouse' 'Handlers-cleaners'
 'Husband' 'Asian-Pac-Islander' 'Male' '40' 'Philippines']
Predicted Class: <=50K
-------------
['51' 'Private' 'HS-grad' 'Separated' 'Craft-repair' 'Not-in-family'
 'White' 'Male' '60' 'United-States']
Predicted Class: >50K
-------------
['30' 'Private' 'HS-grad' 'Divorced' 'Craft-repair' 'Not-in-family'
 'White' 'Female' '40' 'United-States']
Predicted Class: <=50K
-------------
['36' 'Private' 'Assoc-acdm' 'Separated' 'Tech-support' 'Not-in-family'
 'White' 'Female' '3' 'United-States']
Predicted Class: <=50K
-------------
['51' 'Self-emp-not-inc' 'Masters' 'Never-married' 'Exec-managerial'
 'Not-in-family' 'White' 'Male' '20' 'United-States']
Predicted Class: <=50K
-------------
['30' 'Private' 'Bachelors' 'Never-married' 'Other-service'
 'Not-in-family' 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['33' 'Private' 'Some-college' 'Married-civ-spouse' 'Sales' 'Husband'
 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['61' 'Private' 'HS-grad' 'Divorced' 'Transport-moving' 'Not-in-family'
 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['27' 'Private' 'HS-grad' 'Married-civ-spouse' 'Machine-op-inspct'
 'Husband' 'Black' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['23' 'Private' 'HS-grad' 'Separated' 'Machine-op-inspct' 'Own-child'
 'White' 'Male' '53' 'United-States']
Predicted Class: <=50K
-------------
['31' 'Private' '10th' 'Divorced' 'Machine-op-inspct' 'Not-in-family'
 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['22' 'Federal-gov' 'HS-grad' 'Never-married' 'Prof-specialty' 'Own-child'
 'White' 'Male' '35' 'United-States']
Predicted Class: <=50K
```