```
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_2.txt'
trainFileName = 'train_2.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 ####
62 Private 166691 HS-grad 9 Divorced Exec-managerial Unmarried White Female 0 0 4
55 Local-gov 134042 Masters 14 Married-civ-spouse Exec-managerial Wife White Fema
49 Private 196360 Some-college 10 Married-civ-spouse Exec-managerial Husband Whit
66 Self-emp-not-inc 37170 HS-grad 9 Married-civ-spouse Adm-clerical Husband White
35 Private 151835 Masters 14 Married-civ-spouse Prof-specialty Husband White Male
49 Local-gov 371886 Assoc-voc 11 Married-civ-spouse Protective-serv Husband White
21 Private 315470 HS-grad 9 Never-married Sales Own-child White Female 0 0 30 Uni
34 Private 49469 HS-grad 9 Never-married Exec-managerial Not-in-family White Male
45 Local-gov 251786 HS-grad 9 Never-married Adm-clerical Not-in-family White Fema
28 Private 68393 HS-grad 9 Never-married Other-service Not-in-family White Female
39 Private 108943 11th 7 Divorced Other-service Unmarried White Female 0 0 40 Uni
27 Self-emp-inc 217848 12th 8 Married-civ-spouse Adm-clerical Husband White Male
28 Private 113635 12th 8 Never-married Craft-repair Not-in-family White Male 0 0
25 Private 193379 HS-grad 9 Never-married Transport-moving Not-in-family White Ma
37 Private 175185 11th 7 Never-married Machine-op-inspct Not-in-family White Male
37 Private 421633 Assoc-voc 11 Divorced Handlers-cleaners Unmarried Black Female
32 Private 108247 Some-college 10 Married-civ-spouse Craft-repair Husband White M
23 Private 437940 HS-grad 9 Married-civ-spouse Machine-op-inspct Husband White Ma
17 Private 194946 11th 7 Never-married Other-service Own-child White Female 0 0 2
27 Private 113866 HS-grad 9 Never-married Other-service Not-in-family White Femal
```

```
# 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

62	Private	166691	HS-grad	9	Divorced	Exec-managerial	Unmarried	White	Female	0	0	40	United-States
55	Local-gov	134042	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States
49	Private	196360	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	46	United-States
66	Self-emp-not-inc	37170	HS-grad	9	Married-civ-spouse	Adm-clerical	Husband	White	Male	0	0	42	United-States
35	Private	151835	Masters	14	Married-civ-spouse	Prof-specialty	Husband	White	Male	99999	0	50	United-States
49	Local-gov	371886	Assoc-voc	11	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	56	United-States
21	Private	315470	HS-grad	9	Never-married	Sales	Own-child	White	Female	0	0	30	United-States
34	Private	49469	HS-grad	9	Never-married	Exec-managerial	Not-in-family	White	Male	0	0	50	United-States
45	Local-gov	251786	HS-grad	9	Never-married	Adm-clerical	Not-in-family	White	Female	0	0	37	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 ####
['38' 'Private' '89814' 'HS-grad' '9' 'Married-civ-spouse'
 'Farming-fishing' 'Husband' 'White' 'Male' '0' '0' '50' '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]
['48' '?' '5th-6th' 'Divorced' '?' 'Not-in-family' 'White' 'Male' '99'
 '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_a
    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, ca
        dist = np.append( dist, temp, axis = 0 )
    dist = np.delete(dist, 0, axis = 0)

    # 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_attr

    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 )
```

```
['57' 'Private' 'Assoc-voc' 'Widowed' 'Craft-repair' 'Not-in-family'
 'Black' 'Male' '40' '?']
My nearest equivalent:
['52' 'Local-gov' 'HS-grad' 'Divorced' 'Craft-repair' 'Own-child' 'White'
 'Male' '40' 'United-States']
My updated value:
['57' 'Private' 'Assoc-voc' 'Widowed' 'Craft-repair' 'Not-in-family'
 'Black' 'Male' '40' 'United-States']
-------------------------------
Data point with missing value:
['38' '?' 'Some-college' 'Married-civ-spouse' '?' 'Husband' 'White' 'Male'
 '75' 'United-States']
My nearest equivalent:
['43' 'State-gov' 'Some-college' 'Married-civ-spouse' 'Transport-moving'
 'Husband' 'White' 'Male' '84' 'United-States']
My updated value:
['38' 'State-gov' 'Some-college' 'Married-civ-spouse' 'Transport-moving'
 'Husband' 'White' 'Male' '75' 'United-States']
-------------------------------
Data point with missing value:
['41' 'Private' 'Masters' 'Married-civ-spouse' 'Prof-specialty' 'Husband'
 'White' 'Male' '40' '?']
My nearest equivalent:
['33' 'Private' 'HS-grad' 'Married-civ-spouse' 'Craft-repair' 'Husband'
 'White' 'Male' '40' 'United-States']
My updated value:
['41' 'Private' 'Masters' 'Married-civ-spouse' 'Prof-specialty' 'Husband'
 'White' 'Male' '40' 'United-States']
-------------------------------
Data point with missing value:
['80' '?' '1st-4th' 'Separated' '?' 'Not-in-family' 'Black' 'Male' '15'
 'United-States']
My nearest equivalent:
['80' 'Private' 'HS-grad' 'Divorced' 'Adm-clerical' 'Other-relative'
 'White' 'Female' '20' 'United-States']
My updated value:
['80' 'Private' '1st-4th' 'Separated' 'Adm-clerical' 'Not-in-family'
 'Black' 'Male' '15' 'United-States']
-------------------------------
Data point with missing value:
['31' '?' 'Some-college' 'Never-married' '?' 'Own-child' 'Black' 'Male'
 '40' 'United-States']
My nearest equivalent:
['25' 'Private' 'HS-grad' 'Married-civ-spouse' 'Machine-op-inspct'
 'Husband' 'White' 'Male' '40' 'United-States']
My updated value:
['31' 'Private' 'Some-college' 'Never-married' 'Machine-op-inspct'
 'Own-child' 'Black' 'Male' '40' 'United-States']
-------------------------------
Data point with missing value:
['70' '?' '7th-8th' 'Married-civ-spouse' '?' 'Husband' 'White' 'Male' '8'
 'United-States']
My nearest equivalent:
['65' 'Self-emp-not-inc' 'Masters' 'Married-civ-spouse' 'Prof-specialty'
 'Husband' 'White' 'Male' '16' 'United-States']
My updated value:
['70' 'Self-emp-not-inc' '7th-8th' 'Married-civ-spouse' 'Prof-specialty'
 'Husband' 'White' 'Male' '8' 'United-States']
```

```
    Husband   White   Male   8   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("-------------")
```

```
['62' 'Private' 'HS-grad' 'Divorced' 'Exec-managerial' 'Unmarried' 'White'
 'Female' '40' 'United-States']
Predicted Class: >50K
-------------
['55' 'Local-gov' 'Masters' 'Married-civ-spouse' 'Exec-managerial' 'Wife'
 'White' 'Female' '40' 'United-States']
Predicted Class: <=50K
-------------
['49' 'Private' 'Some-college' 'Married-civ-spouse' 'Exec-managerial'
 'Husband' 'White' 'Male' '46' 'United-States']
Predicted Class: <=50K
-------------
['66' 'Self-emp-not-inc' 'HS-grad' 'Married-civ-spouse' 'Adm-clerical'
 'Husband' 'White' 'Male' '42' 'United-States']
Predicted Class: <=50K
-------------
['35' 'Private' 'Masters' 'Married-civ-spouse' 'Prof-specialty' 'Husband'
 'White' 'Male' '50' 'United-States']
Predicted Class: <=50K
-------------
['49' 'Local-gov' 'Assoc-voc' 'Married-civ-spouse' 'Protective-serv'
 'Husband' 'White' 'Male' '56' 'United-States']
Predicted Class: <=50K
-------------
['21' 'Private' 'HS-grad' 'Never-married' 'Sales' 'Own-child' 'White'
 'Female' '30' 'United-States']
Predicted Class: <=50K
-------------
['34' 'Private' 'HS-grad' 'Never-married' 'Exec-managerial'
 'Not-in-family' 'White' 'Male' '50' 'United-States']
Predicted Class: <=50K
-------------
['45' 'Local-gov' 'HS-grad' 'Never-married' 'Adm-clerical' 'Not-in-family'
 'White' 'Female' '37' 'United-States']
Predicted Class: <=50K
-------------
['28' 'Private' 'HS-grad' 'Never-married' 'Other-service' 'Not-in-family'
 'White' 'Female' '46' 'United-States']
Predicted Class: >50K
-------------
['39' 'Private' '11th' 'Divorced' 'Other-service' 'Unmarried' 'White'
 'Female' '40' 'United-States']
Predicted Class: <=50K
-------------
['27' 'Self-emp-inc' '12th' 'Married-civ-spouse' 'Adm-clerical' 'Husband'
 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['28' 'Private' '12th' 'Never-married' 'Craft-repair' 'Not-in-family'
 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['25' 'Private' 'HS-grad' 'Never-married' 'Transport-moving'
 'Not-in-family' 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['37' 'Private' '11th' 'Never-married' 'Machine-op-inspct' 'Not-in-family'
```