```
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_1.txt'
trainFileName = 'train_1.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 ####
23 Private 64520 10th 6 Never-married Craft-repair Not-in-family White Male 0 0 4
47 Private 182177 Some-college 10 Divorced Protective-serv Unmarried White Female
27 Private 203776 Bachelors 13 Married-civ-spouse Sales Husband White Male 7688 0
20 Private 143062 HS-grad 9 Never-married Machine-op-inspct Own-child White Male
52 Private 287454 Bachelors 13 Divorced Prof-specialty Unmarried White Female 0 0
24 Private 300275 HS-grad 9 Never-married Handlers-cleaners Not-in-family White M
45 Private 125892 Masters 14 Married-civ-spouse Exec-managerial Husband White Mal
41 Federal-gov 348059 Masters 14 Divorced Exec-managerial Unmarried White Female
23 Private 214236 HS-grad 9 Never-married Adm-clerical Not-in-family White Female
46 Self-emp-not-inc 103540 Some-college 10 Married-civ-spouse Exec-managerial Hus
41 Self-emp-inc 220821 Masters 14 Married-civ-spouse Prof-specialty Husband White
42 Private 360879 Some-college 10 Married-civ-spouse Craft-repair Husband White M
23 Private 148890 Some-college 10 Never-married Handlers-cleaners Own-child White
36 Private 49657 Bachelors 13 Married-civ-spouse Prof-specialty Husband White Mal
46 Local-gov 195418 Bachelors 13 Married-civ-spouse Prof-specialty Husband Black
42 Private 163985 HS-grad 9 Separated Transport-moving Not-in-family White Male 0
61 Private 255978 HS-grad 9 Widowed Sales Not-in-family White Male 0 0 50 United-
22 Private 180190 Some-college 10 Never-married Craft-repair Not-in-family White
45 Private 169324 9th 5 Divorced Other-service Unmarried Black Female 0 0 40 Hait
38 Self-emp-inc 184456 HS-grad 9 Married-civ-spouse Sales Husband White Male 0 0
```

```
# 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 ####
```

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | Private | 64520 | 10th | 6 | Never-married | Craft-repair | Not-in-family | White | Male | 0 | 0 | 40 | United-States |
| 47 | Private | 182177 | Some-college | 10 | Divorced | Protective-serv | Unmarried | White | Female | 0 | 0 | 23 | United-States |
| 27 | Private | 203776 | Bachelors | 13 | Married-civ-spouse | Sales | Husband | White | Male | 7688 | 0 | 45 | United-States |
| 20 | Private | 143062 | HS-grad | 9 | Never-married | Machine-op-inspct | Own-child | White | Male | 0 | 0 | 40 | United-States |
| 52 | Private | 287454 | Bachelors | 13 | Divorced | Prof-specialty | Unmarried | White | Female | 0 | 0 | 40 | United-States |
| 24 | Private | 300275 | HS-grad | 9 | Never-married | Handlers-cleaners | Not-in-family | White | Male | 0 | 0 | 40 | United-States |
| 45 | Private | 125892 | Masters | 14 | Married-civ-spouse | Exec-managerial | Husband | White | Male | 0 | 1977 | 60 | United-States |
| 41 | Federal-gov | 348059 | Masters | 14 | Divorced | Exec-managerial | Unmarried | White | Female | 0 | 0 | 40 | United-States |
| 23 | Private | 214236 | HS-grad | 9 | Never-married | Adm-clerical | Not-in-family | White | Female | 0 | 0 | 40 | United-States |
| 46 | Self-emp-not-inc | 103540 | Some-college | 10 | Married-civ-spouse | Exec-managerial | 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 ####
['17' 'Private' '269430' '10th' '6' 'Never-married' 'Machine-op-inspct'
 'Not-in-family' 'White' 'Male' '0' '0' '40' '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]
['18' '?' 'Some-college' 'Never-married' '?' 'Own-child' 'White' 'Male'
 '30' '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 )
```

```
Data point with missing value:
['52' '?' 'Bachelors' 'Widowed' '?' 'Not-in-family' 'White' 'Female' '8'
 'United-States']
My nearest equivalent:
['46' 'Private' 'HS-grad' 'Married-civ-spouse' 'Adm-clerical' 'Wife'
 'White' 'Female' '12' 'United-States']
My updated value:
['52' 'Private' 'Bachelors' 'Widowed' 'Adm-clerical' 'Not-in-family'
 'White' 'Female' '8' 'United-States']
-------------------------------
Data point with missing value:
['21' '?' 'Some-college' 'Never-married' '?' 'Own-child' 'White' 'Female'
 '40' 'United-States']
My nearest equivalent:
['21' 'Private' 'Some-college' 'Never-married' 'Adm-clerical' 'Own-child'
 'White' 'Female' '40' 'United-States']
My updated value:
['21' 'Private' 'Some-college' 'Never-married' 'Adm-clerical' 'Own-child'
 'White' 'Female' '40' 'United-States']
-------------------------------
Data point with missing value:
['77' '?' 'Assoc-acdm' 'Married-civ-spouse' '?' 'Husband' 'White' 'Male'
 '2' 'United-States']
My nearest equivalent:
['90' 'Federal-gov' 'Masters' 'Divorced' 'Prof-specialty' 'Not-in-family'
 'White' 'Male' '99' 'United-States']
My updated value:
['77' 'Federal-gov' 'Assoc-acdm' 'Married-civ-spouse' 'Prof-specialty'
 'Husband' 'White' 'Male' '2' 'United-States']
-------------------------------
Data point with missing value:
['74' '?' 'HS-grad' 'Married-civ-spouse' '?' 'Husband' 'White' 'Male' '48'
 'United-States']
My nearest equivalent:
['68' 'Private' 'HS-grad' 'Married-civ-spouse' 'Machine-op-inspct'
 'Husband' 'Black' 'Male' '40' 'United-States']
My updated value:
['74' 'Private' 'HS-grad' 'Married-civ-spouse' 'Machine-op-inspct'
 'Husband' 'White' 'Male' '48' 'United-States']
-------------------------------
Data point with missing value:
['22' '?' 'HS-grad' 'Never-married' '?' 'Own-child' 'Black' 'Male' '10'
 'United-States']
My nearest equivalent:
['17' 'Private' '10th' 'Never-married' 'Handlers-cleaners' 'Own-child'
 'White' 'Female' '15' 'United-States']
My updated value:
['22' 'Private' 'HS-grad' 'Never-married' 'Handlers-cleaners' 'Own-child'
 'Black' 'Male' '10' 'United-States']
-------------------------------
Data point with missing value:
['69' '?' 'Prof-school' 'Married-civ-spouse' '?' 'Husband' 'White' 'Male'
 '35' 'United-States']
My nearest equivalent:
['64' 'Private' 'Assoc-voc' 'Married-civ-spouse' 'Sales' 'Husband'
 'Asian-Pac-Islander' 'Male' '40' 'Philippines']
My updated value:
['69' 'Private' 'Prof-school' 'Married-civ-spouse' 'Sales' 'Husband'
 'White' 'Male' '35' 'United-States']
```

```
      White   Male   35   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("-------------")
```

```
['23' 'Private' '10th' 'Never-married' 'Craft-repair' 'Not-in-family'
 'White' 'Male' '40' 'United-States']
Predicted Class: >50K
-------------
['47' 'Private' 'Some-college' 'Divorced' 'Protective-serv' 'Unmarried'
 'White' 'Female' '23' 'United-States']
Predicted Class: <=50K
-------------
['27' 'Private' 'Bachelors' 'Married-civ-spouse' 'Sales' 'Husband' 'White'
 'Male' '45' 'United-States']
Predicted Class: <=50K
-------------
['20' 'Private' 'HS-grad' 'Never-married' 'Machine-op-inspct' 'Own-child'
 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
```