```
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_5.txt'
trainFileName = 'train_5.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 ####
33 Private 48520 HS-grad 9 Separated Handlers-cleaners Not-in-family White Male (
39 Private 259716 HS-grad 9 Divorced Machine-op-inspct Unmarried White Female 0 (
23 Private 211345 Some-college 10 Never-married Adm-clerical Own-child White Fema
35 Private 37655 HS-grad 9 Divorced Machine-op-inspct Unmarried White Female 0 0
34 Private 398874 Bachelors 13 Never-married Prof-specialty Not-in-family White M
65 Local-gov 172646 9th 5 Married-civ-spouse Exec-managerial Husband White Male (
30 Private 48520 HS-grad 9 Divorced Transport-moving Not-in-family White Male 0 (
19 Private 93762 Some-college 10 Never-married Adm-clerical Not-in-family White F
27 Private 173944 Bachelors 13 Never-married Prof-specialty Not-in-family White M
20 Private 177896 HS-grad 9 Never-married Handlers-cleaners Other-relative Black
41 Self-emp-inc 56019 Prof-school 15 Married-civ-spouse Prof-specialty Husband As
49 Private 41294 Some-college 10 Divorced Adm-clerical Not-in-family White Female
61 Private 373099 7th-8th 4 Married-civ-spouse Craft-repair Husband White Male 0
49 Private 61885 Assoc-acdm 12 Married-civ-spouse Sales Husband White Male 0 0 45
32 Private 335569 HS-grad 9 Married-civ-spouse Craft-repair Husband White Male 0
30 Private 209768 Assoc-voc 11 Married-civ-spouse Machine-op-inspct Husband White
45 Private 149169 Some-college 10 Divorced Craft-repair Not-in-family White Male
53 Private 231865 12th 8 Married-civ-spouse Craft-repair Husband White Male 0 0 4
53 Private 176185 HS-grad 9 Married-civ-spouse Craft-repair Husband White Male 0
35 Private 241306 HS-grad 9 Married-civ-spouse Machine-op-inspct Husband White Ma
```

```
# 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 ####
```

33	Private	48520	HS-grad	9	Separated	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States
39	Private	259716	HS-grad	9	Divorced	Machine-op-inspct	Unmarried	White	Female	0	0	40	Mexico
23	Private	211345	Some-college	10	Never-married	Adm-clerical	Own-child	White	Female	0	0	40	Nicaragua
35	Private	37655	HS-grad	9	Divorced	Machine-op-inspct	Unmarried	White	Female	0	0	60	United-States
34	Private	398874	Bachelors	13	Never-married	Prof-specialty	Not-in-family	White	Male	0	0	42	United-States
65	Local-gov	172646	9th	5	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	40	United-States
30	Private	48520	HS-grad	9	Divorced	Transport-moving	Not-in-family	White	Male	0	0	43	United-States
19	Private	93762	Some-college	10	Never-married	Adm-clerical	Not-in-family	White	Female	0	0	40	United-States
27	Private	173944	Bachelors	13	Never-married	Prof-specialty	Not-in-family	White	Male	0	0	50	United-States
20	Private	177896	HS-grad	9	Never-married	Handlers-cleaners	Other-relative	Black	Male	0	0	40	United-States
41	Self-emp-	56019	Prof-school	15	Married-civ-	Prof-specialty	Husband	Asian-Pac-	Male	99999	0	50	India

```
# 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 ####
['28' 'Local-gov' '336951' 'Assoc-acdm' '12' 'Married-civ-spouse'
 'Protective-serv' 'Husband' '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]
['51' '?' 'Some-college' 'Married-civ-spouse' '?' 'Wife' 'White' 'Female'
 '18' '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
   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 )
```

```
------------------------------
Data point with missing value:
['51' '?' 'Some-college' 'Married-civ-spouse' '?' 'Wife' 'White' 'Female'
 '18' 'United-States']
My nearest equivalent:
['48' 'Private' '11th' 'Separated' 'Other-service' 'Unmarried' 'Black'
 'Male' '22' 'United-States']
My updated value:
['51' 'Private' 'Some-college' 'Married-civ-spouse' 'Other-service' 'Wife'
 'White' 'Female' '18' 'United-States']
------------------------------
Data point with missing value:
['73' '?' 'HS-grad' 'Married-civ-spouse' '?' 'Husband' 'White' 'Male' '8'
 'United-States']
My nearest equivalent:
['34' 'Private' 'HS-grad' 'Never-married' 'Craft-repair' 'Not-in-family'
 'White' 'Male' '99' 'United-States']
My updated value:
['73' 'Private' 'HS-grad' 'Married-civ-spouse' 'Craft-repair' 'Husband'
 'White' 'Male' '8' 'United-States']
------------------------------
Data point with missing value:
['32' '?' '9th' 'Married-civ-spouse' '?' 'Husband' 'White' 'Male' '45'
 'United-States']
My nearest equivalent:
['25' 'Private' 'Assoc-acdm' 'Never-married' 'Exec-managerial'
 'Not-in-family' 'White' 'Male' '45' 'United-States']
My updated value:
['32' 'Private' '9th' 'Married-civ-spouse' 'Exec-managerial' 'Husband'
 'White' 'Male' '45' 'United-States']
------------------------------
Data point with missing value:
['57' '?' '9th' 'Married-civ-spouse' '?' 'Husband' 'White' 'Male' '40'
 'United-States']
My nearest equivalent:
['57' 'Private' 'HS-grad' 'Married-civ-spouse' 'Craft-repair' 'Husband'
 'White' 'Male' '40' 'United-States']
My updated value:
['57' 'Private' '9th' 'Married-civ-spouse' 'Craft-repair' 'Husband'
 'White' 'Male' '40' 'United-States']
------------------------------
Data point with missing value:
['29' '?' 'Bachelors' 'Married-spouse-absent' '?' 'Not-in-family' 'White'
 'Female' '4' 'India']
My nearest equivalent:
['33' 'Private' 'Bachelors' 'Never-married' 'Prof-specialty'
 'Not-in-family' 'Asian-Pac-Islander' 'Male' '10' 'India']
My updated value:
['29' 'Private' 'Bachelors' 'Married-spouse-absent' 'Prof-specialty'
 'Not-in-family' 'White' 'Female' '4' 'India']
------------------------------
Data point with missing value:
['23' '?' 'Some-college' 'Never-married' '?' 'Own-child' 'White' 'Female'
 '30' 'United-States']
My nearest equivalent:
['22' 'Private' 'Some-college' 'Never-married' 'Adm-clerical' 'Own-child'
 'White' 'Female' '30' 'United-States']
```

```
My updated value:
['23' 'Private' 'Some-college' 'Never-married' 'Adm-clerical' 'Own-child'
 'White' 'Female' '30' 'United-States']
--------------------------------
Data point with missing value:
['32' '?' '11th' 'Married-civ-spouse' '?' 'Wife' 'White' 'Female' '15'
 'United-States']
My nearest equivalent:
['37' 'State-gov' 'Bachelors' 'Never-married' 'Prof-specialty'
 'Not-in-family' 'White' 'Female' '20' 'United-States']
My updated value:
['32' 'State-gov' '11th' 'Married-civ-spouse' 'Prof-specialty' 'Wife'
 'White' 'Female' '15' 'United-States']
--------------------------------
Data point with missing value:
['19' '?' 'Some-college' 'Never-married' '?' 'Own-child' 'White' 'Male'
 '30' 'United-States']
My nearest equivalent:
['27' 'Private' 'Assoc-acdm' 'Never-married' 'Other-service'
 'Not-in-family' 'White' 'Male' '30' 'United-States']
My updated value:
['19' 'Private' 'Some-college' 'Never-married' 'Other-service' 'Own-child'
 'White' 'Male' '30' 'United-States']
--------------------------------
Data point with missing value:
['29' 'Self-emp-not-inc' 'Some-college' 'Never-married' 'Farming-fishing'
 'Not-in-family' 'White' 'Male' '40' '?']
My nearest equivalent:
['24' 'Private' 'Bachelors' 'Married-civ-spouse' 'Sales' 'Husband' 'White'
 'Male' '40' 'United-States']
My updated value:
['29' 'Self-emp-not-inc' 'Some-college' 'Never-married' 'Farming-fishing'
 'Not-in-family' 'White' 'Male' '40' 'United-States']
--------------------------------
Data point with missing value:
['62' '?' 'Some-college' 'Married-civ-spouse' '?' 'Husband' 'White' 'Male'
 '40' 'United-States']
My nearest equivalent:
['62' 'Federal-gov' 'Some-college' 'Married-civ-spouse' 'Adm-clerical'
 'Husband' 'White' 'Male' '40' 'United-States']
My updated value:
['62' 'Federal-gov' 'Some-college' 'Married-civ-spouse' 'Adm-clerical'
 'Husband' 'White' 'Male' '40' '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
-------------
['65' 'Local-gov' '9th' 'Married-civ-spouse' 'Exec-managerial' 'Husband'
 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['30' 'Private' 'HS-grad' 'Divorced' 'Transport-moving' 'Not-in-family'
 'White' 'Male' '43' 'United-States']
Predicted Class: <=50K
-------------
['19' 'Private' 'Some-college' 'Never-married' 'Adm-clerical'
 'Not-in-family' 'White' 'Female' '40' 'United-States']
Predicted Class: <=50K
-------------
['27' 'Private' 'Bachelors' 'Never-married' 'Prof-specialty'
 'Not-in-family' 'White' 'Male' '50' 'United-States']
Predicted Class: <=50K
-------------
['20' 'Private' 'HS-grad' 'Never-married' 'Handlers-cleaners'
 'Other-relative' 'Black' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['41' 'Self-emp-inc' 'Prof-school' 'Married-civ-spouse' 'Prof-specialty'
 'Husband' 'Asian-Pac-Islander' 'Male' '50' 'India']
Predicted Class: >50K
-------------
['49' 'Private' 'Some-college' 'Divorced' 'Adm-clerical' 'Not-in-family'
 'White' 'Female' '40' 'United-States']
Predicted Class: >50K
-------------
['61' 'Private' '7th-8th' 'Married-civ-spouse' 'Craft-repair' 'Husband'
 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['49' 'Private' 'Assoc-acdm' 'Married-civ-spouse' 'Sales' 'Husband'
 'White' 'Male' '45' 'United-States']
Predicted Class: <=50K
-------------
['32' 'Private' 'HS-grad' 'Married-civ-spouse' 'Craft-repair' 'Husband'
 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['30' 'Private' 'Assoc-voc' 'Married-civ-spouse' 'Machine-op-inspct'
 'Husband' 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
-------------
['45' 'Private' 'Some-college' 'Divorced' 'Craft-repair' 'Not-in-family'
 'White' 'Male' '50' 'United-States']
Predicted Class: >50K
-------------
['53' 'Private' '12th' 'Married-civ-spouse' 'Craft-repair' 'Husband'
 'White' 'Male' '40' 'United-States']
Predicted Class: >50K
-------------
['53' 'Private' 'HS-grad' 'Married-civ-spouse' 'Craft-repair' 'Husband'
 'White' 'Male' '40' 'United-States']
Predicted Class: <=50K
```