Branch: master

Find file Copy path

03MAIR-Algoritmos-de-optimizacion / Seminario / MiguelAngelSotoCollada_Seminario.ipynb

mcollada Creado con Colaboratory db56bd7 a minute ago

1 contributor

947 lines (947 sloc) 40.6 KB

Raw Blame History

Open in Colab

Algoritmos de optimización - Seminario

Nombre y Apellidos: Miguel Angel Soto Collada
Url: https://github.com/mcollada/03MAIR-Algoritmos-de-optimizacion/blob/master/Seminario/MiguelAngelSotoCollada_Seminario.ipynb

Problema:

> 1. Elección de grupos de población homogéneos

Descripción del problema:(copiar enunciado)

Problema Elección de grupos de población homogéneos:

• Una productora ganadera nos encarga la tarea de seleccionar grupos de terneros para aplicar 3 tratamientos diferentes. Para cada uno de los tratamientos debemos seleccionar 3 grupos de terneros que sean lo mas homogéneos posible en peso para que en los resultados del tratamiento influya lo menos posible el peso del animal. Disponemos de una población de N animales entre machos y hembras

• Se solicita diseñar un algoritmo para conseguir una agrupación que cumpla de la mejor manera posible las especificaciones de la productora.

• Aun se desconocen los datos concretos de los animales por lo que debemos trabajar con datos que debemos generar de forma aleatoria.

• Hemos podido averigua que el peso medio del ternero es 37kg con una desviación estándar de 2.1

Nos proporcionan la siguiente plantilla de los datos:

Id	Peso	Sexo
1	38.7	M
2	37.1	H

3	39.2	M

(*) La respuesta es obligatoria

In [0]:

(*)¿Cuantas posibilidades hay sin tener en cuenta las restricciones?

¿Cuantas posibilidades hay teniendo en cuenta todas las restricciones.

Respuesta:

Sin tener en cuenta las restricciones y por lo tanto agrupando todos los terneros en los 9 grupos creo que sería combinaciones de N (tamaño de la población elegido) elementos tomados de N/9 en N/9

In [0]:

Modelo para el espacio de soluciones
(*) ¿Cual es la estructura de datos que mejor se adapta al problema? Argumentalo.(Es posible que hayas elegido una al principio y veas la necesidad de cambiar, arguentalo)

Respuesta:

Almacenamos los terneros en un dataframe de inicio (df) de python y en el mismo guardamos los siguientes datos:

- id: para poder identificar al ternero en este caso usamos el id de la fila del dataframe, si más adelante fuera necesario crear otro tipo de identificador con alguna etiqueta de código de barras que identificara de forma única al ternero lo único que tendríamos que hacer es crear una columna más con ese dato.
- Peso: almacena el peso del termero en kg
- Sexo: Sexo del ternero (H: Hembra, M: Macho)

Las estructuras de datos que van a almacenar las mejores clasificaciones de terneros para cada uno de los tratamientos y grupos las almacenamos en dataframes independientes con la misma estructura de datos que el original (entiendo que en la vida real además debería llevar un identificador único con la etiqueta del ternero, pero para el estudio del problema actual no es necesario y unicamente supondría añadir esa columna en el dataframe original y ya se iría teniendo en todos los demás dataframes de salida

al ir asignandole los datos del dataframe original):

```
df_Tratamiento1_Machos_G1
df_Tratamiento1_Hembras_G2
df_Tratamiento1_Machos_G3

df_Tratamiento2_Machos_G1
df_Tratamiento2_Hembras_G2
df_Tratamiento2_Machos_G3

df_Tratamiento3_Machos_G1
df_Tratamiento3_Hembras_G2
df_Tratamiento3_Machos_G3
```

Según el modelo para el espacio de soluciones
(*)¿Cual es la función objetivo?

(*)¿Es un problema de maximización o minimización?

Respuesta:

- La función objetivo que he elegido ha sido que el peso del ternero no se desvíe del peso medio del grupo que estamos tratando (valor dado por el enunciado) en + - 0,25kg. Para elegir ese valor me he centrado en que la media de peso del grupo 1 de los machos se pide que sea lo más aproximada a 38.7 kg y de los machos del grupo 3 sea lo más aproximada a 39.2kg de tal forma que hay una distancia de 0.5 kg entre esos 2 grupos de machos entonces para clasificar grupos homogéneos me centro en los terneros cuyo peso esta en la media del grupo que estemos clasificando + - 0.25:

  Media_Grupo - 0.25 <= peso_del_ternero <= Media_Grupo + 0.25

- Lo he abordado como un problema de mínizar las diferencias en peso para cada uno de los tratamientos y grupos, como indíca el enunciado lo más homogenéos posible en peso.

In [0]:

Diseña un algoritmo para resolver el problema por fuerza bruta

Respuesta

In [0]:
```
# import y funciones generales que voy a usar a  lo largo del programa
import random
import pandas as pd
import numpy as np
from scipy import stats
import pdb
import math
from time import time


#Función para calcular el tiempo de ejecución
def calcular_tiempo(f):
    def wrapper(*args,**kwargs):
        inicio = time()
        resultado = f(*args,**kwargs)
        tiempo = time() - inicio
        print("\r\nTiempo de ejecución para algoritmo: "+"{0:.25f}".format(tiempo))
        return resultado
    return wrapper

# Función para chequear los datos que se vayan generando
def ChequearDatosGenerados(df, titulo):
    print('------------------ Chequeo Datos --------------------')
    print('Dataframe: ' +titulo )
    print('Media: ' + str(np.mean(df[0:])))
    print('Desviación: '+ str(np.std(df[1:])))
    print('Número individuos grupo: '+ str(df.shape[0]))
    print('-----------------------------------------------------')
```

In [0]:
```
# FUERZA BRUTA
def Clasificacion_Fuerza_Bruta (df,sexo, pesoMedioPedido, desviacionPermitidaGrupo):
    dfSeleccionados=pd.DataFrame(columns=('peso', 'sexo'))
    for i in range(df.shape[0]) :
        if df.loc[i,'sexo']==sexo and df.loc[i,'peso'] >= pesoMedioPedido - desviacionPermitida
Grupo and df.loc[i,'peso'] <= pesoMedioPedido + desviacionPermitidaGrupo :
            dfSeleccionados.loc[dfSeleccionados.shape[0]]=df.loc[i,:].tolist()
```

```
    return dfSeleccionados


@calcular_tiempo
def Lanzadera_Algoritmo_Fuerza_Bruta():

    # Realizar 3 llamadas al algoritmo de Clasificacion_Fuerza_Bruta para cada uno de los grupo
s pedidos
    dfSeleccionados_Machos_G1=Clasificacion_Fuerza_Bruta(df,'M',38.7,desviacionStandarPermitida
EnGrupo)
    dfSeleccionados_Hembras_G2=Clasificacion_Fuerza_Bruta(df,'H',37.1,desviacionStandarPermitid
aEnGrupo)
    dfSeleccionados_Machos_G3=Clasificacion_Fuerza_Bruta(df,'M',39.2,desviacionStandarPermitida
EnGrupo)

    # Dividimos los machos seleccionados del grupo 1 para los 3 tratamientos
    df_Tratamiento1_Machos_G1=dfSeleccionados_Machos_G1.loc[:dfSeleccionados_Machos_G1.shape[0]
/3-1, :]
    df_Tratamiento2_Machos_G1=dfSeleccionados_Machos_G1.loc[dfSeleccionados_Machos_G1.shape[0]/
3:2*dfSeleccionados_Machos_G1.shape[0]/3-1, :]
    df_Tratamiento3_Machos_G1=dfSeleccionados_Machos_G1.loc[2*dfSeleccionados_Machos_G1.shape[0
]/3:, :]


    # Dividimos las hembras seleccionadas del grupo 2 para los 3 tratamientos
    df_Tratamiento1_Hembras_G2=dfSeleccionados_Hembras_G2.loc[:dfSeleccionados_Hembras_G2.shape
[0]/3-1, :]
    df_Tratamiento2_Hembras_G2=dfSeleccionados_Hembras_G2.loc[dfSeleccionados_Hembras_G2.shape[
0]/3:2*dfSeleccionados_Hembras_G2.shape[0]/3-1, :]
    df_Tratamiento3_Hembras_G2=dfSeleccionados_Hembras_G2.loc[2*dfSeleccionados_Hembras_G2.shap
e[0]/3:, :]


    # Dividimos los machos seleccionados del grupo 3 para los 3 tratamientos
    df_Tratamiento1_Machos_G3=dfSeleccionados_Machos_G3.loc[:dfSeleccionados_Machos_G3.shape[0]
/3-1, :]
    df_Tratamiento2_Machos_G3=dfSeleccionados_Machos_G3.loc[dfSeleccionados_Machos_G3.shape[0]/
3:2*dfSeleccionados_Machos_G3.shape[0]/3-1, :]
    df_Tratamiento3_Machos_G3=dfSeleccionados_Machos_G3.loc[2*dfSeleccionados_Machos_G3.shape[0
```

```
df_Tratamiento3_Machos_G3=dfSeleccionados_Machos_G3.loc[2*dfSeleccionados_Machos_G3.shape[0
]/3:, :]

    # Chequeamos datos iniciales y generados
    ChequearDatosGenerados(df, 'Datos originales generados')
    ChequearDatosGenerados(df_Tratamiento1_Machos_G1, 'Datos Tratamiento 1 Machos Grupo 1')
    ChequearDatosGenerados(df_Tratamiento2_Machos_G1, 'Datos Tratamiento 2 Machos Grupo 1')
    ChequearDatosGenerados(df_Tratamiento3_Machos_G1, 'Datos Tratamiento 3 Machos Grupo 1')
    ChequearDatosGenerados(df_Tratamiento1_Hembras_G2,'Datos Tratamiento 1 Hembras Grupo 2')
    ChequearDatosGenerados(df_Tratamiento2_Hembras_G2,'Datos Tratamiento 2 Hembras Grupo 2')
    ChequearDatosGenerados(df_Tratamiento3_Hembras_G2,'Datos Tratamiento 3 Hembras Grupo 2')
    ChequearDatosGenerados(df_Tratamiento1_Machos_G3, 'Datos Tratamiento 1 Machos Grupo 3')
    ChequearDatosGenerados(df_Tratamiento2_Machos_G3, 'Datos Tratamiento 2 Machos Grupo 3')
    ChequearDatosGenerados(df_Tratamiento3_Machos_G3, 'Datos Tratamiento 3 Machos Grupo 3')
```

Calcula la complejidad del algoritmo por fuerza bruta

Respuesta:

El orden de complejidad es 3n ya que recorremos la lista entera para clasificar dada uno de los grupos pedidos (no la recorremos 9 veces porque después cada uno de los dataframes obtenidos lo divido en tres partes para cada uno de los tratamientos de esa forma obtengo los 9 grupos):

dfSeleccionados_Machos_G1=Clasificacion_Fuerza_Bruta(df,'M',38.7,desviacionStandarPermitidaEnGrupo)
dfSeleccionados_Hembras_G2=Clasificacion_Fuerza_Bruta(df,'H',37.1,desviacionStandarPermitidaEnGrupo)
dfSeleccionados_Machos_G3=Clasificacion_Fuerza_Bruta(df,'M',39.2,desviacionStandarPermitidaEnGrupo)

In [0]:

(*)Diseña un algoritmo que mejore la complejidad del algortimo por fuerza bruta. Argumenta porque crees que mejora el algoritmo por fuerza bruta

Respuesta

La mejora respecto a fuerza bruta es porque tenemos que recorrer la lista entera una únicamente vez en lugar de 3 veces que lo haríamos con fuerza bruta y de esa forma ya obtenemos la clasificación completa pedida de los grupos.

Nota: Como la población es pequeña la mejora en tiempos no se aprecia bien pero en pruebas que he realizado con 1.000.000 de

terneros en fuerza bruta nos vamos a 831.85 segundos aproximadamente (13.85 minutos) y en cambio con técnica voraz 688.22 (11.47 minutos) no sé si esta prueba es demasiado fiable porque mi equpo no es nada del otro mundo, pero bueno ya que la he hecho la aporto.

In [0]:
```
#### Resolución por Técnica Voraz ####

def Clasificacion_Tecnica_Voraz (df,desviacionPermitidaGrupo):
    dfSeleccionados_Machos_G1=pd.DataFrame(columns=('peso', 'sexo'))
    dfSeleccionados_Hembras_G2=pd.DataFrame(columns=('peso', 'sexo'))
    dfSeleccionados_Machos_G3=pd.DataFrame(columns=('peso', 'sexo'))
    for i in range(df.shape[0]) :
        sexo_ternero=df.loc[i,'sexo']
        peso_ternero=df.loc[i,'peso']

        if sexo_ternero =='M':
            if peso_ternero >= 38.7 - desviacionPermitidaGrupo and peso_ternero <= 38.7 + desvi
acionPermitidaGrupo :
                dfSeleccionados_Machos_G1.loc[dfSeleccionados_Machos_G1.shape[0]]=df.loc[i,:].t
olist()
            elif peso_ternero >= 39.2 - desviacionPermitidaGrupo and peso_ternero <= 39.2 + des
viacionPermitidaGrupo :
                dfSeleccionados_Machos_G3.loc[dfSeleccionados_Machos_G3.shape[0]]=df.loc[i,:].t
olist()

        elif sexo_ternero=='H' and peso_ternero >= 37.1 - desviacionPermitidaGrupo and peso_ter
nero <= 37.1 + desviacionPermitidaGrupo :
            dfSeleccionados_Hembras_G2.loc[dfSeleccionados_Hembras_G2.shape[0]]=df.loc[i,:].tol
ist()


    return dfSeleccionados_Machos_G1,dfSeleccionados_Hembras_G2,dfSeleccionados_Machos_G3



@calcular_tiempo
def Lanzadera_Algoritmo_Tecnica_Voraz():

    # Realizar 1 llamada al algoritmo de Clasificacion_Tecnica_Voraz
    dfSeleccionados_Machos_G1,dfSeleccionados_Hembras_G2,dfSeleccionados_Machos_G3=Clasificacio
```

```
n_Tecnica_Voraz(df,desviacionStandarPermitidaEnGrupo)

    # Dividimos los machos seleccionados del grupo 1 para los 3 tratamientos
    df_Tratamiento1_Machos_G1=dfSeleccionados_Machos_G1.loc[:dfSeleccionados_Machos_G1.shape[0]
/3-1, :]
    df_Tratamiento2_Machos_G1=dfSeleccionados_Machos_G1.loc[dfSeleccionados_Machos_G1.shape[0]/
3:2*dfSeleccionados_Machos_G1.shape[0]/3-1, :]
    df_Tratamiento3_Machos_G1=dfSeleccionados_Machos_G1.loc[2*dfSeleccionados_Machos_G1.shape[0
]/3:, :]


    # Dividimos las hembras seleccionadas del grupo 2 para los 3 tratamientos
    df_Tratamiento1_Hembras_G2=dfSeleccionados_Hembras_G2.loc[:dfSeleccionados_Hembras_G2.shape
[0]/3-1, :]
    df_Tratamiento2_Hembras_G2=dfSeleccionados_Hembras_G2.loc[dfSeleccionados_Hembras_G2.shape[
0]/3:2*dfSeleccionados_Hembras_G2.shape[0]/3-1, :]
    df_Tratamiento3_Hembras_G2=dfSeleccionados_Hembras_G2.loc[2*dfSeleccionados_Hembras_G2.shap
e[0]/3:, :]


    # Dividimos los machos seleccionados del grupo 3 para los 3 tratamientos
    df_Tratamiento1_Machos_G3=dfSeleccionados_Machos_G3.loc[:dfSeleccionados_Machos_G3.shape[0]
/3-1, :]
    df_Tratamiento2_Machos_G3=dfSeleccionados_Machos_G3.loc[dfSeleccionados_Machos_G3.shape[0]/
3:2*dfSeleccionados_Machos_G3.shape[0]/3-1, :]
    df_Tratamiento3_Machos_G3=dfSeleccionados_Machos_G3.loc[2*dfSeleccionados_Machos_G3.shape[0
]/3:, :]

     # Chequeamos datos iniciales y generados
    ChequearDatosGenerados(df, 'Datos originales generados')
    ChequearDatosGenerados(df_Tratamiento1_Machos_G1, 'Datos Tratamiento 1 Machos Grupo 1')
    ChequearDatosGenerados(df_Tratamiento2_Machos_G1, 'Datos Tratamiento 2 Machos Grupo 1')
    ChequearDatosGenerados(df_Tratamiento3_Machos_G1, 'Datos Tratamiento 3 Machos Grupo 1')
    ChequearDatosGenerados(df_Tratamiento1_Hembras_G2,'Datos Tratamiento 1 Hembras Grupo 2')
    ChequearDatosGenerados(df_Tratamiento2_Hembras_G2,'Datos Tratamiento 2 Hembras Grupo 2')
    ChequearDatosGenerados(df_Tratamiento3_Hembras_G2,'Datos Tratamiento 3 Hembras Grupo 2')
    ChequearDatosGenerados(df_Tratamiento1_Machos_G3, 'Datos Tratamiento 1 Machos Grupo 3')
    ChequearDatosGenerados(df_Tratamiento2_Machos_G3, 'Datos Tratamiento 2 Machos Grupo 3')
    ChequearDatosGenerados(df_Tratamiento3_Machos_G3, 'Datos Tratamiento 3 Machos Grupo 3')
```

(*)Calcula la complejidad del algoritmo

Respuesta:

El orden de complejidad del algoritmo con técnica voraz que se ha obtenido es n, ya que recorremos la lista entera una única vez para clasificar todos los grupos pedidos.

In [0]:

Según el problema (y tenga sentido), diseña un juego de datos de entrada aleatorios

Respuesta

Obtengo la población de terneros del tamaño pasado como parametro a la función, para generar los datos uso una función aleatoria que genera datos según una distribución normal con media y desviación indicadas, de esta forma podemos generar los datos como se han pedido media 37kg y desviación 2,1kg

In [4]:
```
# Obtener población de terneros del tamaño pasado como parametro, para generar
# los datos usamos una función aleatoria que genera datos según una distribución
# normal con media y desviación indicadas, de esta forma podemos generar los datos
# como se han pedido media 37kg y desviación 2,1kg
def ObtenerPoblacionTerneros(tamanioPoblacion,media,desviacionStandar):
    df=pd.DataFrame(columns=('peso', 'sexo'))
    df['peso']=[np.random.normal(media, desviacionStandar, 1) for x in range(tamanioPoblacion)]
    df['sexo']=[random.randint(0,1) for x in range(tamanioPoblacion) ]
    df['sexo'] = np.where(df['sexo'], 'M', 'H')

    return df

# Inicializamos valores a los valores indicados en el problema y asignamos también el tamaño de
 la población
N=1000
media=37
desviacionStandarPoblacion=2.1

# A la desviacionStandarPermitidaEnGrupo le asigno el valor de 0.25kg ya que
# la media del grupo 1 de los machos se pide que sea lo más aproximada a 38.7
# y de los machos del grupo 3 39.2 de tal forma que hay una distancia de 0.5 kg
```

```
# entre esos 2 grupos de machos entonces para clasificar grupos homogeneos en peso me centro
# en los terneros cuyo peso esta en la media del grupo que estemos clasificando +-0.25
desviacionStandarPermitidaEnGrupo=0.25


# Obtener población de terneros del tamaño pasado como parametro, para generar
# los datos usamos una función aleatoria que genera datos según una distribución
# normal con media y desviación indicadas
df=ObtenerPoblacionTerneros(N, media,desviacionStandarPoblacion)

# chequeamos los datos iniciales generados para ver si cumplen los criterios
# de la media y la desviación que se piden en el problema
ChequearDatosGenerados(df, 'Datos originales generados ')
```

```
------------------ Chequeo Datos -------------------
Dataframe: Datos originales generados
Media: peso    36.80693
Name: 0, dtype: float64
Desviación: peso    2.052442
dtype: float64
Número individuos grupo: 1000
----------------------------------------------------
```

Aplica el algoritmo al juego de datos generado

Respuesta

In [5]:
```
# Aplicamos los algoritmos construidos al juego de datos generado
print ('**********************************************************************************
********')
print ('INICIO Fuerza Bruta **********************************************************************
********')
print ('**********************************************************************************
********')
print (' ')
Lanzadera_Algoritmo_Fuerza_Bruta()
print (' ')
print ('**********************************************************************************
********')
print ('FIN Fuerza Bruta **************************************************************
```

```
********')
print ('**********************************************************************************
********')
print (' ')
print (' ')
print (' ')
print ('**********************************************************************************
********')
print ('INICIO Técnica Voraz ***********************************************************
********')
print ('**********************************************************************************
********')
print (' ')
Lanzadera_Algoritmo_Tecnica_Voraz()
print (' ')
print ('**********************************************************************************
********')
print ('FIN Técnica Voraz **************************************************************
********')
print ('**********************************************************************************
********')
print (' ')
print (' ')
print (' ')
```

```
**********************************************************************************************
INICIO Fuerza Bruta ************************************************************************
**********************************************************************************************

------------------ Chequeo Datos ------------------
Dataframe: Datos originales generados
Media: peso    36.80693
Name: 0, dtype: float64
Desviación: peso    2.052442
dtype: float64
Número individuos grupo: 1000
---------------------------------------------------
------------------ Chequeo Datos ------------------
Dataframe: Datos Tratamiento 1 Machos Grupo 1
Media: peso    38.635088
```

```
Name: 0, dtype: float64
Desviación: peso    0.083287
dtype: float64
Número individuos grupo: 10
--------------------------------------------------
------------------ Chequeo Datos -------------------
Dataframe: Datos Tratamiento 2 Machos Grupo 1
Media: peso    38.590969
Name: 10, dtype: float64
Desviación: peso    0.154296
dtype: float64
Número individuos grupo: 10
--------------------------------------------------
------------------ Chequeo Datos -------------------
Dataframe: Datos Tratamiento 3 Machos Grupo 1
Media: peso    38.721216
Name: 20, dtype: float64
Desviación: peso    0.15097
dtype: float64
Número individuos grupo: 10
--------------------------------------------------
------------------ Chequeo Datos -------------------
Dataframe: Datos Tratamiento 1 Hembras Grupo 2
Media: peso    37.034088
Name: 0, dtype: float64
Desviación: peso    0.14444
dtype: float64
Número individuos grupo: 12
--------------------------------------------------
------------------ Chequeo Datos -------------------
Dataframe: Datos Tratamiento 2 Hembras Grupo 2
Media: peso    37.095167
Name: 13, dtype: float64
Desviación: peso    0.083931
dtype: float64
Número individuos grupo: 11
--------------------------------------------------
------------------ Chequeo Datos -------------------
Dataframe: Datos Tratamiento 3 Hembras Grupo 2
Media: peso    37.052583
Name: 25, dtype: float64
```

```
Name: 25, dtype: float64
Desviación: peso    0.146002
dtype: float64
Número individuos grupo: 12
--------------------------------------------------
------------------ Chequeo Datos -------------------
Dataframe: Datos Tratamiento 1 Machos Grupo 3
Media: peso    39.321909
Name: 0, dtype: float64
Desviación: peso    0.051221
dtype: float64
Número individuos grupo: 4
--------------------------------------------------
------------------ Chequeo Datos -------------------
Dataframe: Datos Tratamiento 2 Machos Grupo 3
Media: peso    39.196039
Name: 5, dtype: float64
Desviación: peso    0.148464
dtype: float64
Número individuos grupo: 4
--------------------------------------------------
------------------ Chequeo Datos -------------------
Dataframe: Datos Tratamiento 3 Machos Grupo 3
Media: peso    39.222117
Name: 10, dtype: float64
Desviación: peso    0.154301
dtype: float64
Número individuos grupo: 4
--------------------------------------------------

Tiempo de ejecución para algoritmo: 0.319615602493286132812500 0

**********************************************************************************************
FIN Fuerza Bruta **************************************************************************
**********************************************************************************************



**********************************************************************************************
INICIO Técnica Voraz **********************************************************************
```

```
**********************************************************************************************

------------------ Chequeo Datos -------------------
Dataframe: Datos originales generados
Media: peso    36.80693
Name: 0, dtype: float64
Desviación: peso    2.052442
dtype: float64
Número individuos grupo: 1000
----------------------------------------------------
------------------ Chequeo Datos -------------------
Dataframe: Datos Tratamiento 1 Machos Grupo 1
Media: peso    38.635088
Name: 0, dtype: float64
Desviación: peso    0.083287
dtype: float64
Número individuos grupo: 10
----------------------------------------------------
------------------ Chequeo Datos -------------------
Dataframe: Datos Tratamiento 2 Machos Grupo 1
Media: peso    38.590969
Name: 10, dtype: float64
Desviación: peso    0.154296
dtype: float64
Número individuos grupo: 10
----------------------------------------------------
------------------ Chequeo Datos -------------------
Dataframe: Datos Tratamiento 3 Machos Grupo 1
Media: peso    38.721216
Name: 20, dtype: float64
Desviación: peso    0.15097
dtype: float64
Número individuos grupo: 10
----------------------------------------------------
------------------ Chequeo Datos -------------------
Dataframe: Datos Tratamiento 1 Hembras Grupo 2
Media: peso    37.034088
Name: 0, dtype: float64
Desviación: peso    0.14444
dtype: float64
```

```
Número individuos grupo: 12
-----------------------------------------------------
------------------ Chequeo Datos --------------------
Dataframe: Datos Tratamiento 2 Hembras Grupo 2
Media: peso    37.095167
Name: 13, dtype: float64
Desviación: peso    0.083931
dtype: float64
Número individuos grupo: 11
-----------------------------------------------------
------------------ Chequeo Datos --------------------
Dataframe: Datos Tratamiento 3 Hembras Grupo 2
Media: peso    37.052583
Name: 25, dtype: float64
Desviación: peso    0.146002
dtype: float64
Número individuos grupo: 12
-----------------------------------------------------
------------------ Chequeo Datos --------------------
Dataframe: Datos Tratamiento 1 Machos Grupo 3
Media: peso    39.321909
Name: 0, dtype: float64
Desviación: peso    0.051221
dtype: float64
Número individuos grupo: 4
-----------------------------------------------------
------------------ Chequeo Datos --------------------
Dataframe: Datos Tratamiento 2 Machos Grupo 3
Media: peso    39.196039
Name: 5, dtype: float64
Desviación: peso    0.148464
dtype: float64
Número individuos grupo: 4
-----------------------------------------------------
------------------ Chequeo Datos --------------------
Dataframe: Datos Tratamiento 3 Machos Grupo 3
Media: peso    39.222117
Name: 10, dtype: float64
Desviación: peso    0.154301
dtype: float64
```

```
Número individuos grupo: 4
--------------------------------------------------

Tiempo de ejecución para algoritmo: 0.276624679565429687500000 0

**********************************************************************************
FIN Técnica Voraz ****************************************************************
**********************************************************************************
```

In [0]:

Enumera las referencias que has utilizado(si ha sido necesario) para llevar a cabo el trabajo

- Documentación aportada en el curso
- Documentación de Python
- Wikipedia

Respuesta

Describe brevemente las lineas de como crees que es posible avanzar en el estudio del problema. Ten en cuenta incluso posibles variaciones del problema y/o variaciones al alza del tamaño

Respuesta

Si nos fueramos a tamaños de población muy elevados en los que los tiempos de respuesta de este algoritmo fueran inviables entonces rediseñaria el algoritmo y usaría algortimos heurísticos o genéticos siempre que no fuera necesario clasificar a toda la población.

Otra posibilidad también sería aplicar técnica voraz pero limitando el número de individuos a obtener en cada grupo, de tal forma, que si supieramos que los grupos por ejemplo con 1.000 individuos ya son suficientes para el estudio, en el momento que alcanzaramos ese tamaño de grupo pondría algún punto de control para que el algoritmo parara.