[[1]](#footnote-1)

Formato IEEE para reporte del proyecto final(julio 2021)

Daniel Behar A. (20200402), Cruz Leonel del Cid C. (20200394), Marcela Isabel Melgar G. (20200487) - *Computer Science, UFM*

***Resumen*—El proyecto final del curso consistió en analizar una base de datos relacionada a los niveles de glucosa de una persona, incluyendo los detalles relacionados al nivel de la glucosa en el tiempo (entiéndase la medicina y el momento cuando la ingirió). Aplicando los conocimientos adquiridos a lo largo del curso, al igual que técnicas estadísticas, se analizaron los datos en un periodo de tiempo determinado. El fin es comprender su comportamiento en varios lapsos de tiempo y poder visualizarlos gráficamente. Para ello se empleó el lenguaje de programación «Python» y se diseñó un programa que cumpliera con todos los requisitos mínimos de análisis, captura y presentación de datos.**

***Palabras clave*—glucosa, razón de cambio, rango, técnicas de integración y derivación, términos estadísticos.**

1. INTRODUCCIÓN

La diabetes es una enfermedad muy presente en nuestra sociedad hoy en día. Esta enfermedad proviene de altos niveles de azúcar en la sangre y la poca o nula producción de insulina por parte del cuerpo. Dicha hormona ayuda a la glucosa a poder entrar a la célula y así poder proveerle de energía. Mientras más tiempo permanezcan la glucosa en la sangre, más peligrosa se vuelve para el cuerpo. Este exceso de glucosa puede dañar órganos vitales, como el corazón, los nervios, entre otros.

Para las personas que sufren de esta enfermedad, es muy importante poder monitorear constantemente sus niveles de glucosa para poder buscar el tratamiento adecuado. Se emplea constantemente el uso de glucómetros para dicha actividad, pero es de vital importancia comprender cómo se está comportando la glucosa en la sangre conforme pasa el tiempo.

Se ha diseñado una aplicación que permite al usuario poder ingresar los datos del monitoreo, para poder comprender el comportamiento de la glucosa durante el día y así poder ayudarlo a predecir futuros niveles en la sangre. El programa devuelve los cambios que sufrió la glucosa, datos estadísticos como media, mediana y moda; varias gráficas y una función de predicción, que ayude a determinar en cuánto tiempo se puede alcanzar el nivel de glucosa deseado.

1. Diseño Experimental
   1. *Scripts*

Solo se adjuntaron los scripts más relevantes, todo lo demás (como el controlador, lo gráfico y el extractor de datos) se encuentra en este GitHub: https://github.com/Daniel07816/Glucom

Integración:

def TrapecioM (x, y):

n = len(x)

suma = 0

for i in range(n-1):

h = x[i+1]-x[i]

suma+= (y[i+1]+y[i])\*h

return suma\*0.5

Interpolación Newton:

import numpy as np

import sympy as sym

def Newton(time,gluco):

a=[]

divDIF=[]

if len(time) == len(gluco):

n=len(time)

a=[]

a.append(gluco[0])

divDIF = []

divDIF.append([])

for i in range (n-1):

divDIF[0].append((gluco[i+1]-gluco[i])/(time[i+1]-time[i]))

for j in range (1,n-1):

for i in range (0,n-j-1):

if i == 0:

divDIF.append([])

divDIF[j].append((divDIF[j-1][i+1]-divDIF[j-1][i])/(time[j+i+1]-time[i]))

for j in range (0,n-1):

a.append(divDIF[j][0])

x=sym.Symbol('x')

y=a[0]

for i in range (1,len(a)):

t=a[i]

for j in range(i):

t=t\*(x-time[j])

y+=t

return y

else:

return 0

Regresión Lineal:

import math

import numpy as np

import sympy as sym

import matplotlib.pyplot as plt

import pathlib

#matplotlib es para grafica, numpy vectoriza python

def reglin(x,y):

n=len(x)

if len(y)!= n:

print("DEBE DE HABER LA MISMA CANTIDAD DE VALORES EN AMBOS VECTORES")

sx=sum(x) #suma de todo el vector x

sy=sum(y) #suma de todo el vector y

#sumatorias cuadraticas

sx2=0

sxy=0

sy2=0

for i in range(n):

sx2+=(x[i]\*x[i])

for i in range(n):

sxy+=(x[i]\*y[i])

for i in range(n):

sy2+=(y[i]\*y[i])

pendiente=((n\*sxy-sx\*sy)/(n\*sx2-sx\*\*2))

intercepto = sy/n-pendiente\*sx/n

r2=((n\*sxy-sx\*sy)/math.sqrt(n\*sx2-sx\*\*2)/math.sqrt(n\*sy2-sy\*\*2))\*\*2 #coeficiente

return pendiente,intercepto,r2

def imagen(x,y,xp,yp):

#necesario para exportar grafica

fig=plt.figure()

#graficador + etiquetas

plt.plot(x,y,'b.',xp,yp)

plt.xlabel('Tiempo')

plt.ylabel('Glucosa en la Sangre') #se pone r para formato "raw" y los $ encierran el texto para armarlo mas legible

plt.title('Relación Tiempo-Glucosa')

plt.grid(True)

#exportador de grafica

ruta = str(pathlib.Path(\_\_file\_\_).parent.resolve()) # Obtiene la ruta del directorio

ruta = ruta.replace(chr(92),'/')+'/static/grafica.png' # Ruta final con el nombre del archivo

fig.savefig(ruta)

def grafreglin(x,y,a):

xp=np.linspace(min(x),max(x),2)

yp=a[0]\*xp+a[1]

return yp,xp

Derivada:

import numpy as np

#dx(1) = (y(2)-y(1))/(x(2)-x(1)); %Diferencia finita hacia adelante

#for i = 2 : n-1;

#dx(i) = (y(i+1)-y(i-1))/(x(i+1)-x(i-1)) %Diferencia finita centrada

#endfor

#dx(n) = (y(n) - y(n-1))/(x(n)-x(n-1)); %Diferencia finita hacia atras

#endfunction

def derivada(x,y):

n = len(x)

dx=np.zeros\_like(x)

dx[0]=((y[1]-y[0])/(x[1]-x[0])) # adelante

for i in range(n-2):

dx[i+1]=((y[i+2]-y[i])/(x[i+2]-x[i])) # centrada

dx[n-1]=((y[n-1]-y[n-2])/(x[n-1]-x[n-2])) # atras

return dx

Método de Lagrange:

def LagrangePol(x,y,Xint):

sum = 0

n = len(x)

for i in range(n):

producto=y[i]

for j in range(n):

print(j,end=' ')

if i is not j:

producto=producto\*((Xint-x[j])/(x[i]-x[j]))

sum=sum+producto

Yinter=sum

return Yinter

Métodos Estadísticos:

import numpy

from scipy import stats

def Media(x):

media=numpy.mean(x)

return media

def Mediana(x):

mediana=numpy.median(x)

return mediana

def Moda(x):

arreglo=numpy.array(x)

moda=stats.mode(arreglo, axis=None)

resp=moda[0]

fija=resp[0]

return resp

def Maximo(x):

maximo=max(x)

return maximo

def Minimo(x):

minimo=min(x)

return minimo

def Desviacion(x):

desviacion=numpy.std(x)

return desviacion

1. *Gráficos*

*Aclaración: los gráficos van a variar dependiendo del rango de fechas seleccionadas, la hora cuando consumió su medicamento y de la muestra aleatoria de diez elementos entre el rango (solo si la cantidad de observaciones excede a diez).*

Chart, bar chart, histogram

Description automatically generated

Resumen estadístico (sección 5.8)

Chart, line chart, scatter chart

Description automatically generated

Tendencia (sección 5.7)

Chart, line chart

Description automatically generated

Chart, scatter chart

Description automatically generated

Gráficas (sección 5.2)

1. Resultados

Elementos que influyen en estos resultados:

Graphical user interface, application

Description automatically generated

A picture containing company name

Description automatically generated

Resultados:

Table

Description automatically generated with medium confidence

Shape

Description automatically generated with low confidence

Promedio:

Text

Description automatically generated with low confidence

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

A picture containing table

Description automatically generated

1. Discusión de Resultados

*Sección 5.2:* La gráfica de dispersión se generó a partir de diez datos de la muestra (en el caso de que en ese rango se hayan ingresado más de diez observaciones). La curva se generó por medio del método: Polinomio Interpolante de Newton. Esté método requiere datos muy específicos. La variable independiente no puede repetirse en observaciones diferentes, ya que genera una división por cero, indefiniendo toda la función generada. Este problema no se presentó en el gráfico de dispersión.

*Sección 5.3:* Para calcular las razones de cambio se utilizó el método de Derivación por Diferencias Finitas. Después se presentaron las razones junto a la fecha de observación y la condición cuando se ingiere el medicamento.

*Sección 5.4:* Se empleó el mismo método que en la sección anterior. Para calcular la aceleración se derivó dos veces todos los elementos. Se imprimen únicamente el valor más grande y pequeño para exponer cuáles crecimientos o caídas fueron más rápidas o pronunciadas.

*Sección 5.5:* Para calcular el promedio se aplicó el método: Integración por Trapecio. Al resultado de la integral, que representa los datos bajo la curva, se le dividió la diferencia entre el elemento máximo y mínimo. Fue necesario ordenar de mayor a menor todos los datos de la muestra para que el resultado no se viera afectado negativamente.

*Sección 5.6:* El Método de Lagrange nos permitió estimar el tiempo requerido para alcanzar el nivel de glucosa deseado. Se evidenció que sería muy difícil estimar punto por punto cada parte en la función, por ello se usó Lagrange. En este método fue necesario invertir los valores de *x* y de *y*, tomando como *x* a la glucosa y como *y* al tiempo en horas.

*Sección 5.7:* Para determinar la tendencia de los datos se aplicó el método: Regresión Lineal Simple. Se calculó la recta de mejor ajuste y también el coeficiente de determinación. Este coeficiente solo nos ayuda a comprender cuan aproximados a la recta se encuentran los datos. Aunque la gráfica nos permite visualizar la aproximación, el coeficiente es más exacto.

*Sección 5.8:* Para el resumen se calculó la media, mediana, moda, valor máximo, valor mínimo y la desviación estándar. También se graficó el histograma de la muestra. Es importante aclarar que esta fue la única sección donde se tomaron todos los datos de la muestra, es decir, donde no se tomaron solo diez datos aleatorios.

1. Conclusiones

Fue muy importante determinar previamente el rango de fechas que se tomaron para el análisis, ya que con eso se pudo realizar un cálculo más detallado y preciso. Tomar todos los datos en la tabla podía complicar una predicción de movimiento en un lapso. Además, los datos se pueden dispersar mucho y varios cálculos requieren que ellos estén lo más cercanos entré sí.

Los gráficos, aunque detallan temas muy diferentes, si son consistentes con los datos que están expresando. La tendencia y la dispersión de los datos guardan una similitud considerable. Aunque la curva posee una forma no nada similar a la tendencia o la dispersión, si nos permite comprender cuál es el comportamiento que los datos presentan a lo largo del tiempo, comportamiento que podemos ver si analizamos a detalle la tabla de dispersión.

Las razones de cambio demuestran un movimiento errático. Hay varios crecimientos bastante grandes que no son compensados con caídas de un valor similar. Aunque no podemos afirmar que solo creció, ya que, si hubo una caída bastante grande, si podemos deducir que en ese lapso el paciente no fue consistente con sus cuidados. La aceleración de la glucosa revela que no hubo muchos cambios tan drásticos hacia arriba. La diferencia entre la aceleración mínima y máxima es de 281.0357, la cual es un rango bastante amplio, pero el mínimo se encuentra más alejado del cero que el máximo.

El nivel de glucosa deseado tiene una relación directa con el promedio. Mientras más cerca del promedio se encuentre el valor de glucosa deseado, menor será el tiempo necesario para alcanzarlo. Si el paciente o el medico desea llevar al usuario a un nivel de glucosa específico en el menor tiempo posible, calcular el promedio de glucosa en la sangre puede ser una herramienta valiosa para no gastar mucho tiempo y recursos en alcanzar el objetivo.

Al graficar la tendencia se hizo evidente que hay congruencia entre el resultado del coeficiente y la gráfica. Mientas más cerca de uno esté el coeficiente, más aproximados a la recta de mejor ajuste se encuentran los datos.

1. Recomendaciones

Aunque todas las funciones se tenían preparadas en «Octave», Python tenía mejores librerías que nos ayudaron a hacerlo más agradable a la vista y, en consecuencia, más comprensible. Es importante investigar como pasar todas las funciones a Python, ya que Octave fue creado en forma de vector, característica que el lenguaje empleado no posee. Es necesario «vectorizar» Python para no caer en errores de cálculo.

Normalmente las bases de datos contienen muchos errores, por ejemplo, datos «basura», observaciones en blanco, explicaciones sin un formato específico, entre otras. Es importante examinar y purificar la base de datos para eliminar dichos errores. No hacer esto puede generar errores al calcular o complicaciones cuando se analicen los datos de la muestra.

Para la mayoría de los cálculos se requirieron librerías de Python que no se instalan al momento de descargar el lenguaje. Detallar todas las librerías adicionales con el comando de descarga en un archivo adjunto, es importante para que el usuario pueda prepararse antes de ejecutar el programa.

El menú de interacción debe ser claro. Adicional se deben realizar muchas pruebas en situaciones diferentes para encontrar todos los errores de código ocultos.

Si no se tiene suficiente conocimiento sobre los métodos que se usaron, lo ideal es realizar una investigación previa para aclarar todas las dudas posibles. Tratar de optimizar el código de esta aplicación sin antes conocer de dichos métodos, puede causar fallos en los cálculos y fallas en la interpretación de los resultados.

1. Bibliografía
2. Chp, S., Canale, R. (2011) Métodos Numéricos para ingenieros. McGraw Hill. Séptima edición.
3. US National Library of Medicine. (2020, 13 octubre). Diabetes. Recuperado 7 de julio de 2021, de https://medlineplus.gov/spanish/diabetes.html

1. [↑](#footnote-ref-1)