Herramientas de Software para Ingeniería.

Integrantes:

- José Flores
- Jean Lalangui
- Felipe Salvador

NRC: 5217

Proyecto

Programa en Python (numpy, matplotlib), para realizar ajuste de curvas, mínimo 5 tipos de curvas.

1. **Planteamiento:**

   El programa nace de la necesidad para ingresar un masiva cantidad de datos y de forma organizada, con el fin de que al ingresar los datos en una tabla o una matriz y que de estos se pueda graficar. Pero hubo un problema que los tipos de datos podían ser tanto: Exponenciales, Logaritmos, Polinomios, Lineales y rectas. Entonces la función del programa es que de los datos que se ingresan puedan salir en las gráficas que deseemos, pero para eso pasan por varios pasos del programa para que cumplan con su función dando los puntos de la recta.

2. **Objetivos:**

   - Poder determinar la relación que existe entre una variable dependiente y una o más variables independientes, a través de distintos métodos según datos ingresados.
   - Obtener la mejor calidad de una curva ajustada.
   - Obtener un código eficaz y de sencillo entendimiento.
   -

3. **Diseño de Software:**

   MÓDULO 1 (messages.py)

El Módulo 1 consta de partes que no hacen procesos matemáticos simplemente envía menús adornados mensajes de saludo, limpiará la pantalla pero serán llamados en varias partes del programa.

```
# This Python file uses the following encoding: utf-8
# Modulo para funcionabilidad de impresion de menus

# Importamos chalk para textos con colores (https://pypi.org/project/pychalk/)
# Importamos modulo os para limpiar la consola
import chalk
import os

# Limpia la pantalla
def clear():
    os.system('clear')

# Mensaje de inicio de saludo
def Wellcome ():
    clear()
    print(chalk.blue(' X-GRAF \n'))


# Funcion para el menu principal envia al la variable op
def mainMenu ():
    print(chalk.blue('  typedata:   ', bold = True)+ 'Ingresar datos Manualmente')
    print(chalk.blue('  importdata: ', bold = True)+ 'Importar Datos de una direccion local ')
    print(chalk.blue('  editdata:   ', bold = True)+ 'Editar los datos ingresados / importados ')
    print(chalk.blue('  printdata:  ', bold = True)+ 'Imprimir tabla de datos ingresados')
    print(chalk.blue('  curve:      ', bold = True)+ 'Escojer un ajuste de curva para los datos ')
    print(chalk.blue('  exit:       ', bold = True)+ 'Salir...')
    return input('\n ► ')
```

- En esta parte es solo la introducción del programa donde se da orden al algoritmo para que recibir la opción que dará el menú.

```
# Funcion para el menu de curvas disponibles para la variable curve
def curvesMenu ():
    clear()
    Wellcome()
    print(chalk.blue('  1: ', bold = True)+ 'Regresion Polinomica grado N')
    print(chalk.blue('  2: ', bold = True)+ 'Regresion Logaritmica')
    print(chalk.blue('  3: ', bold = True)+ 'Regresion Exponencial')
    print(chalk.blue('  4: ', bold = True)+ 'Regresion Potencial')
    print(chalk.blue('  5: ', bold = True)+ 'Volver...')

    return input('\n ► ')
```

- Definimos la función curvesMenu() para que asigne qué curva desea ser ajustada, y le preguntará de qué tipo desea.

MÓDULO 2 (functionsMat.py)

```python
# Importamos modulo tabulate (https://pypi.org/project/tabulate/)
# Importamos el modulo messages ques creamos antes..
from tabulate import tabulate
from messages import clear

# Usando el modulo tabulate realizo una tabla con puntos x & y que son listas
def table(puntosX, puntosY):
    return tabulate({'Fila': range(1, len(puntosX) + 1), 'Datos X': puntosX,
                     'Datos Y': puntosY}, headers = "keys", tablefmt="grid")

# Defino 2 funciones para romper una matriz de X y Y que retorna los valores en una lista
def dataX(matrix):
    x = []
    for i in range(len(matrix)):
        x.append(matrix[i][0])
    return x

def dataY(matrix):
    y = []
    for i in range(len(matrix)):
        y.append(matrix[i][1])
    return y

# Funcion para validad entrada de datos numericos
# Si se ingresa x termina el bucle...
def validateData ( num ):
    while True:
        if num.isdigit(): break
        elif num == 'x': return -1
        else: num = input('Valor?: ')

    return float( num )
```

- Se importan los datos para que regresen de una manera matricial hacia una forma genérica que es una lista, luego de eso se asignan valores para tanto x,y de forma numérica y se cierra la entrada de valores poniendo x.

```python
# Funcion mas importante..
# Rellena una matriz validando datos
# Rellena una matriz validando datos
# Retorna un diccionario con:
# Puntos en X & Y
# Matriz de los puntos y numero de datos
def fillMatrixData ( imported ):
    matrix, puntosX, puntosY = [], [], []
    go = True
    data = 0
    # En caso de recibir datos por importacion de numpy le preguntamos es reutilizable
    if imported:
        data = len(imported)
        matrix = imported
        puntosX = dataX(matrix)
        puntosY = dataY(matrix)

    else:
        # Agrega datos a una lista hasta presionar letra x
        while(go):
            matrix.append([])
            for j in range( 2 ):
                if (j == 0):
                    matrix[ data ].append(validateData(input(' [ X ]: ')))
                else:
                    matrix[ data ].append(validateData(input(' [ Y ]: ')))
                if(matrix[ data ][j] == -1):
                    go = False
                    break
            data += 1

        data = data - 1
```

- Se requiere ingresar los datos de forma x,y, en el caso de que se requiera importar datos escribir numpy y así se podrá reutilizar los datos obtenidos.

```
    # Eliminar los ultimos elementos de la matriz y vectores para salir del blucle
    matrix.pop()
    
    while True:
        puntosX = dataX(matrix)
        puntosY = dataY(matrix)

        clear()
        print("\n xx La Tabla de datos ingresada es: \n")
        print( table(puntosX, puntosY) )
        
        # Edita la matriz despues de ingresarse
        editar = input('\n Desea editar un dato? (y/n): ► ')

        if editar == 'y':
            matrix = editMatrix(matrix)
        if editar =='n':
            break
        else:
            print('\n xx Ingresa (y o n) para si o no.. xx')

    puntosX = dataX(matrix)
    puntosY = dataY(matrix)
  #Retorna los puntos x & y la matriz completa y el numero de datos..
  return { 'x': puntosX, 'y': puntosY, 'matriz': matrix, 'data': data }
```

- Al momento de ingresar los datos y estar erróneos se puede volver a editar la matriz, al final retornan los datos y completa la matriz.

MÓDULO 3 PRINCIPAL (x-graf.py)

- Importamos los módulos creados anteriormente junto con otros instalados en el sistema

```
# This Python file uses the following encoding: utf-8
# Importamos funciones a utilizar del modulo2
# Importamos funciones del modulo 1
from functionsMat import table, dataX, dataY, validateData, editMatrix, fillMatrixData
from messages import clear, Wellcome, mainMenu, curvesMenu

# Importamos numpy, matplotlib, scipy y math
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import math

# Operador del menu principal
op = ''
```

- Utilizamos numpy para encontrar de forma matricial los coeficientes de los vectores x e y según los datos ingresados.

```
# Calcula de forma matricial los coeficientes de vectores de puntos X y Y
def calcMatrix(vecx, vecy):
  
  print("\n xx Matriz de ajuste de polinomio: \n")
  print(vecx)
  print("\n xx Matriz de datos en Y: \n")
  print(vecy)

  X = np.array(vecx)
  Y = np.array(vecy)
  K = np.dot(np.transpose(X), X)
  V = np.dot(Y,X)
  return np.dot(np.linalg.inv(K), np.transpose(V))

def rd(num):
  return round(num,5)
```

- Según los valores de x e y que se tenga en este punto, el programa segun la opcion que se escoja, preguntará si se desea importar algún nuevo valor a la tabla de datos de x e y, imprimir los datos o dar paso al ajuste de curva.

```
#Vatiable para saber si se ingreso datos..
fillTable = False

while True:·

··Wellcome()·····#Lanzamos el saludo
··op = mainMenu() #Recibimos opcion del menu 1·

··# Opcion 1 TypeData
··# Ingrasara datos por teclado·
··if op == 'typedata':
····print('\n xx Ingrese los valores de X & Y para los datos:··\n')
····recivedData = fillMatrixData(False)
····fillTable = True

··# Opcion 1 Importdata
··# Ingrasara datos mediante un path del archivo y el delimitador
··if op == 'importdata':
····path = input(' xx Ingrese la direccion del archivo ►··')
····delimit = input(' xx Ingrese el delimitador de datos ►··')
····data = np.loadtxt(path, delimiter = delimit)

····print('\n xx Los datos importados son: \n')
····recivedData = fillMatrixData(data.tolist())
····print(table(recivedData['x'], recivedData['y']))
····fillTable = True
```

```
·# Opcion 3 Editdata
·# Editara la matriz con el algoritmo editMatrix()
·if (fillTable and op == 'editdata'):
···recivedData['matriz'] = editMatrix(recivedData['matriz'])
···recivedData = fillMatrixData(recivedData['matriz'])


·# Opcion 4 Printdata
·# Imprimira los datos con la funcion table del modulo 1
·if (fillTable and op == 'printdata'):
···print('\n Los datos ingesados son: \n')
···print(table(recivedData['x'], recivedData['y']))



·# Opcion 5 Cruve
·# Lanzara el otro menu del modulo 1 curvesMenu()
·# Se divide en partes para cada ajuste de curva
·if (fillTable and op == 'curve'):
```

- Dependiendo de la opción que se escogiese , estas serian las operaciones que se llevarían a cabo para la obtención de los datos ajustados según cada tipo de curva.

Curva polinómica

```
··# Inicializo los puntos y matriz para comenzar a trabajar
··# Variables importantes··x, y, xp
··Wellcome()
··curve = int(curvesMenu())

··recivedData = fillMatrixData(recivedData['matriz'])
··x = np.array(recivedData['x'])
··y = np.array(recivedData['y'])
··xp = np.linspace(0.1, max(x), 100)

··#Ajuste polinomico..
··if(curve == 1):
····clear()
····Wellcome()
····grado = int(input('\n Grado del polinomoio: ► '))

····#Calculamos los datos de coeficientes mediante operacion matricial calMatrix()
····mat··= grafPolinomy(recivedData, grado + 1 )
····S = calcMatrix(mat['coefx'], mat['vecy'])

····print("\n xx Matriz de coeficientes de grado [", grado ,"] \n")
····print(S)

····z = np.polyfit(x, y, grado)
····p = np.poly1d(z)

····#Preparamos el plot para comenzar a graficar
····print('\n Ecuacion: Y = ', p)
····plt.title("REGRESIÓN POLINÓMICA")
····plt.plot(xp, p(xp), label = p)
····plt.legend(loc="upper left")

····slope, intercept, r_value, p_value, std_err = stats.linregress(y, x)
····print('\n R_cuadrado: ',r_value**2)
```

Curva Logarítmica

```
·# Ajuste Logaritmico
·if(curve == 2):
···clear()
···Wellcome()

···z = np.polyfit(np.log(x), y, 1)
···ec = "Y =" + str(rd(z[0])) + "ln(x) + " + str(rd(z[1]))
···print('\n Ecuacion: ', ec)

···def flog(x):
·····return z[0]*np.log(x) + z[0]
···
···plt.title("REGRESIÓN LOGARITMICA")
···plt.plot(xp, [flog(i) for i in xp], label = ec)
···plt.legend(loc="upper left")

···nx = []
···for i in range(0,len(x)):
·····nx.append(math.log(x[i]))
···slope, intercept, r_value, p_value, std_err = stats.linregress(y, nx)
···print('\n R_cuadrado: ',r_value**2)
```

Curva Exponencial

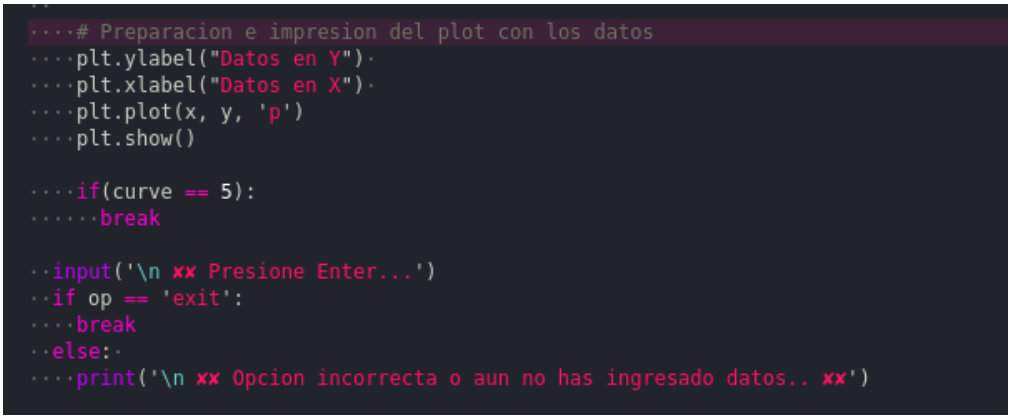
```
# Ajuste Exponencial
if(curve == 3):
    clear()
    Wellcome()

    z = np.polyfit(x, np.log(y), 1, w = np.sqrt(y))

    a = round(z[0],4)
    b = round( np.exp(z[1]),4)
    ec = 'Y = ' + str(rd(b)) + 'e^(' + str(rd(a)) + 'x)'
    print('\n Ecuacion:  Y=',b,'e','^','(',a,')','x')

    def fexp(x):
        return b*(2.718281**(a*x))

    plt.title("REGRESIÓN EXPONENCIAL")
    plt.plot(xp, [fexp(i) for i in xp], label = ec)
    plt.legend(loc="upper left")

    nx = []
    for i in range(0,len(x)):
        nx.append(np.exp(x[i]))
    slope, intercept, r_value, p_value, std_err = stats.linregress(y, nx)
    print('\n R_cuadrado: ',r_value**2)
```

Curva Potencial

```
if(curve == 4):
  clear()
  Wellcome()

  x = x.tolist()
  y = y.tolist()
  
  sumlogy = sum(math.log10(y[i]) for i in range(len(x)))
  sumlogx = sum(math.log10(x[i]) for i in range(len(x)))
  sumxy = sum(math.log10(x[i])*(math.log10(y[i])) for i in range(len(x)))
  sumlogx2 = sum(math.log10(x[i])**2 for i in range(len(x)))
  incognitas = np.array([[len(x),sumlogx],[sumlogx, sumlogx2]])
  valores = np.array([sumlogy, sumxy])

  res = np.linalg.solve(incognitas, valores)

  a = np.power(10, res[0])
  b = res[1]
  ec = 'Y = ' + str(rd(a)) + 'X^' + str(rd(b))
  print('Ecuacion: Y = ',a,'X^',b)

  def fpow(x):
      return a*x**b

  plt.title("REGRESIÓN POTENCIAL")
  plt.plot(xp, [fpow(i) for i in xp], label = ec)
  plt.legend(loc="upper left")

  nx = []
  for i in range(0,len(x)):
      nx.append(np.exp(x[i]))
  slope, intercept, r_value, p_value, std_err = stats.linregress(y, nx)
  print('\n R_cuadrado: ',r_value**2)
```

Impresión del plot y cierre del Main

- Aquí es donde los datos ajustados a la curva cuando ya están listos para ser impresos en la gráfica según lo que se haya seleccionado, si son correctos se mostrará la gráfica, pero si se seleccionó alguna opcion no valida o no se ingresaran datos saldra una texto mostrando lo mismo.



```
    # Preparacion e impresion del plot con los datos
    plt.ylabel("Datos en Y")
    plt.xlabel("Datos en X")
    plt.plot(x, y, 'p')
    plt.show()

    if(curve == 5):
        break

input('\n xx Presione Enter...')
if op == 'exit':
    break
else:
    print('\n xx Opcion incorrecta o aun no has ingresado datos.. xx')
```

4. Conclusiones:

- Las técnicas de regresión y correlación cuantifican la asociación estadística entre dos o más variables. La regresión lineal simple expresa la relación entre variable dependiente Y y una variable independiente X, en otras palabras será la intersección de la línea que mejor se ajuste.

- Para poder linealizar una función se necesita estrictamente tener claro su tipo de comportamiento, es decir si es una función: exponencial, logarítmica, polinómica o una recta, con esto se puede tener claro para su algoritmo.

- El Programa nos ayudó mucho a entender de qué formas puede funcionar una simple función matemática.

5. Recomendaciones:

- Tener conocimientos básicos de estadística y álgebra lineal para la elaboración del programa

6. Emprendimiento:

- Pronosticar la demanda:

  El objetivo del análisis de regresión como método causal es pronosticar la demanda a partir de una o más causas (variables independientes), las cuales pueden ser por ejemplo el tiempo, precios del producto o servicio, precios de la competencia, economía del país, acciones del gobierno o fomentos publicitarios.

- Aplicación en medicina:

  En medicina, se han efectuado estudios de la reducción del peso de una persona en términos del número de semanas que ha seguido una dieta específica; o la cantidad de medicamento absorbido por el organismo en función del tiempo. El procedimiento estadístico que se utiliza para este fin se conoce como análisis de regresión que permite establecer la relación funcional o ecuación matemática que relaciona las variables, así como la fuerza de esa relación.

- Aplicación dentro de laboratorios:

  Dentro de los análisis que se realizan en un laboratorio, ya sea en una materia como mecánica en la que se tiene una ecuación que sirve para estudiar elementos sometidos a fatiga en función del número de ciclos a los que se somete un material, el ajuste de curva podrá conjuntos de valores que tendrán relación con la ecuación dada; o en electricidad que se puede obtener el valor de una resistencia en un circuito y su error mediante un ajuste de regresión lineal de pares de datos experimentales de voltaje e intensidad obtenidos mediante un voltímetro y un amperímetro; además de otras materias se podrá facilitar la interpretación de datos obtenidos a través de los experimentos.

- Mercado de Valores (Bolsa de Valores):
  En este mercado existe una variación tan grande que hubo que hacer un programa que pueda leer los mas rápido posible las variaciones del mercado, que quiere decir esto, por ejemplo el Forex que por sus siglas es Foreign Exchange Market (Mercado de Divisas), en este mercado se maneja las monedas y como el mundo se maneja con esto pues es demasiado grande para solo ponerlo como en papeles, entonces se hizo la idea de manejar este mercado mediante una gráfica que representa su variación en todo los tiempos que se desea, ver cómo se alza el precio de una moneda o como se desvaloriza al mismo tiempo.