Universidad Nacional de Ingeniería
FACULTAD DE CIENCIAS

Proyecto de Fundamentos De Programación
2024-I

Integrantes:

Facultad de Ciencias - Escuela de Ingeniería Física - Mamani Benites Jimmy - 20211489D
Facultad de Ciencias - Escuela de Ingeniería Física - Rios Cusi David - 20230216J
Facultad de Ciencias - Escuela de Ingeniería Física - Cruz Quispe Luis - 20221664C

Índice:

Introducción

Este proyecto explora herramientas y técnicas clave de análisis de datos, machine learning, procesamiento de lenguaje natural, desarrollo web y automatización. Utilizamos las librerías de Python Pandas, Matplotlib y NumPy para la manipulación y visualización de datos. Scikit-learn se emplea para implementar y evaluar modelos de machine learning. En procesamiento de lenguaje natural (NLP), aplicamos técnicas de clasificación de textos y generación de contenido. Flask facilita el desarrollo de aplicaciones web dinámicas, integrando modelos de machine learning. Finalmente, Selenium automatiza pruebas y extracción de datos web. Este enfoque integral combina diversas tecnologías para resolver problemas complejos y desarrollar soluciones innovadoras.

Fundamento teórico

LIBRERIAS:

Pandas:

¿Qué es Pandas?

Pandas es una biblioteca de código abierto que proporciona estructuras de datos rápidas, flexibles y expresivas diseñadas para trabajar con datos relacionales o etiquetados de manera intuitiva. Es fundamentalmente esencial en el ecosistema de Python para la ciencia de datos debido a su capacidad para manejar y manipular fácilmente datos tabulares y series temporales.

Características principales de Pandas:

Estructuras de datos:

- DataFrame: Es una estructura de datos tabular bidimensional con columnas de diferentes tipos de datos. Es similar a una hoja de cálculo o una tabla SQL. Cada columna en un DataFrame es una serie de Pandas.

- Series: Es un arreglo unidimensional etiquetado capaz de contener cualquier tipo de datos (enteros, cadenas, números de punto flotante, objetos Python, etc.). Las series son básicamente las columnas de un DataFrame.

- Grouping:Pandas nos brinda la capacidad de agrupar data frames por valores de columna y luego fusionarlos nuevamente en un resultado con el método .groupby. Pandas group by nos permitirá lograr lo siguiente:

Aplicar una función de agregación a cada grupo de forma independiente y combinar los resultados del group by en una estructura de datos.

Funcionalidades principales:

-Lectura y escritura de datos: Pandas puede leer y escribir datos desde y hacia una amplia variedad de formatos de archivos, incluyendo CSV, Excel, HDF5, SQL, JSON, entre otros.

-Indexación y selección: Permite indexar, seleccionar y subconjuntar datos de manera eficiente utilizando etiquetas (loc) o índices enteros (iloc).

-Manipulación de datos: Ofrece funciones poderosas para limpiar, transformar y fusionar datos. Esto incluye operaciones como la eliminación de valores faltantes, relleno de datos faltantes, agrupación de datos, y más.

-Operaciones aritméticas y estadísticas: Facilita cálculos numéricos y estadísticos sobre datos, incluyendo agregaciones, funciones matemáticas, y métodos para calcular estadísticas descriptivas.

Matplotlib:

Matplotlib es una biblioteca de visualización en Python que permite crear gráficos estáticos, animados e interactivos de alta calidad. Es una de las herramientas más utilizadas en el ámbito científico y de análisis de datos debido a su flexibilidad y capacidad para crear una amplia variedad de visualizaciones.

Tipos de Gráficos: Ofrece soporte para una amplia gama de tipos de gráficos, incluyendo:

Gráficos de línea (plot())

Gráficos de dispersión (scatter())

Histogramas (hist())

Calidad de Salida: Matplotlib está diseñado para producir gráficos de alta calidad para su publicación en revistas y presentaciones. La salida puede ser guardada en varios formatos de archivo como PNG, PDF, SVG, y más.

Componentes principales de Matplotlib:

-Figure: Es el contenedor de nivel superior que contiene todos los elementos del gráfico. Puede contener uno o más Axes (subgráficos).

-Axes: Es la región del gráfico donde se dibujan los datos. Cada Axes tiene un sistema de coordenadas asociado (ejes x e y) y puede contener varios elementos gráficos como líneas, puntos, barras, etc.

Axis: Son los ejes x e y que muestran los límites de los datos. Se pueden configurar para mostrar etiquetas, marcas y ticks.

-Artist: Son todos los elementos que se pueden dibujar en un gráfico, como objetos Line2D (líneas), Text (texto), Patch (formas), etc.

NumPy:

NumPy es una biblioteca fundamental para la computación numérica en Python. Proporciona un poderoso objeto de matriz multidimensional (ndarray), así como funciones para operaciones matemáticas rápidas en arreglos, facilitando así el manejo de datos numéricos y científicos de manera eficiente.

Objeto ndarray:

Es la estructura principal de datos en NumPy.

Permite la creación de arreglos multidimensionales de tamaño fijo y tipo homogéneo.

Es eficiente en términos de espacio y tiempo para realizar operaciones matemáticas y lógicas en grandes conjuntos de datos.

Funciones y Operaciones:

Operaciones Vectorizadas: NumPy permite realizar operaciones matemáticas elementales en arreglos enteros sin la necesidad de bucles explícitos, lo que mejora considerablemente el rendimiento y la legibilidad del código.

Funciones de Álgebra Lineal: Proporciona rutinas para realizar operaciones de álgebra lineal básica, como multiplicación de matrices, descomposiciones, resolución de sistemas lineales, entre otros.

Funciones Matemáticas: Incluye funciones matemáticas básicas y avanzadas que operan en arreglos completos de datos, como trigonometría, funciones exponenciales y logarítmicas, funciones de estadísticas, y más.

Manipulación de Datos: Ofrece funciones para cambiar la forma (reshape), concatenar (concatenate), dividir (split), y otros métodos para manipular arreglos.

Indexación Avanzada: Permite indexar arreglos usando técnicas avanzadas como indexación booleana y por máscaras, lo que facilita la selección y manipulación de datos basados en criterios específicos.

Integración con otras bibliotecas:

NumPy es la base sobre la cual se construyen muchas otras bibliotecas científicas en Python, como Pandas, SciPy, Scikit-Learn, Matplotlib, y más. Esto permite un flujo de trabajo fluido para análisis de datos, visualización y aprendizaje automático.

Machine Learning

Machine Learning es una disciplina de la inteligencia artificial que se centra en el desarrollo de sistemas y algoritmos que pueden aprender y mejorar automáticamente a partir de la experiencia sin intervención humana explícita. En el contexto de scikit-learn, una biblioteca de Python para machine learning, se trabajan principalmente dos tipos de aprendizaje: supervisado y no supervisado.

Tipos de Aprendizaje:

- Scikit-learn es una biblioteca de aprendizaje automático de código abierto para Python que proporciona una amplia gama de herramientas y algoritmos para análisis predictivo y minería de datos. Aquí explico con más detalle qué es Scikit-learn y qué ofrece:

Propósito Principal:

Scikit-learn está diseñado para ser una biblioteca simple y eficiente para machine learning en Python. Ofrece herramientas para trabajar con datos de manera intuitiva y permite a los usuarios implementar rápidamente algoritmos de aprendizaje automático en sus proyectos.

Algoritmos y Funcionalidades:

Incluye una variedad de algoritmos de aprendizaje automático supervisados y no supervisados, como regresión lineal, regresión logística, árboles de decisión, máquinas de vectores de soporte (SVM), clustering (k-means, DBSCAN), reducción de dimensionalidad (PCA), entre otros.

También proporciona herramientas para preprocesamiento de datos, validación de modelos, selección de modelos y ajuste de hiperparámetros, lo que facilita el desarrollo y la evaluación de modelos predictivos

Procesamiento de lenguaje natural en clasificación

El aprendizaje por clasificación en machine learning es una técnica que se utiliza para predecir la clase o categoría a la que pertenece una observación nueva, basándose en el aprendizaje de patrones o estructuras presentes en datos de entrenamiento previamente etiquetados. Acontinuacion explico los conceptos principales relacionados con el aprendizaje por clasificación:

Conceptos Fundamentales:

Datos Etiquetados:

En el aprendizaje por clasificación, se requiere un conjunto de datos de entrenamiento donde cada observación está etiquetada con la clase a la que pertenece. Por ejemplo, en un problema de clasificación de correo electrónico, las observaciones podrían ser mensajes de correo y las etiquetas podrían indicar si son "spam" o "no spam".

Modelos de Clasificación:

Un modelo de clasificación es un algoritmo o conjunto de reglas que aprende de los datos de entrenamiento para asignar nuevas observaciones a una clase específica. Algunos ejemplos comunes de modelos de clasificación incluyen:

Regresión Logística: Se utiliza para problemas binarios y multiclase, estimando la probabilidad de que una observación pertenezca a cada clase.

Árboles de Decisión y Bosques Aleatorios: Utilizan estructuras de árbol para dividir los datos en nodos basados en características, haciendo predicciones basadas en las clases mayoritarias en las hojas.

Redes Neuronales: Modelos más complejos que pueden aprender representaciones no lineales de los datos para realizar clasificaciones.

Funciones de Decisión y Probabilidades:

Los modelos de clasificación suelen proporcionar una función de decisión que mapea características de entrada a una clase específica. Además, muchos modelos también pueden estimar probabilidades de pertenencia a cada clase, permitiendo una interpretación más refinada de las predicciones.

Aplicaciones Prácticas:

Diagnóstico Médico: Predecir enfermedades basadas en síntomas y resultados de pruebas.

Detección de Fraude: Identificar transacciones fraudulentas en sistemas de pago.

Reconocimiento de Imágenes: Clasificar imágenes en categorías como objetos, personas, paisajes.

FLASK:

Flask es un microframework web ligero y flexible para Python que proporciona las herramientas necesarias para construir aplicaciones web rápidamente y con un mínimo de líneas de código. A diferencia de otros frameworks más robustos como Django, Flask está diseñado para ser modular y fácilmente extensible, permitiendo a los desarrolladores elegir las herramientas y bibliotecas que mejor se adapten a sus necesidades.

SELENIUM:

¿Qué es Selenium?

Selenium es una suite de herramientas que permite la automatización de navegadores web en diferentes plataformas. Originalmente fue desarrollado como un proyecto de código abierto por Jason Huggins en 2004, y desde entonces ha evolucionado y se ha convertido en una herramienta fundamental para muchos desarrolladores y testers.

Componentes Principales de Selenium:

WebDriver:

WebDriver es la interfaz principal de Selenium que permite la comunicación con diferentes navegadores web. Proporciona métodos y funciones para interactuar con elementos de la página, enviar entradas, realizar clics y otras acciones como se haría manualmente en un navegador.

IDE (Integrated Development Environment):

Selenium IDE es una extensión para navegadores como Chrome y Firefox que permite grabar, editar y reproducir scripts de pruebas automatizadas directamente desde el navegador. Es útil para pruebas rápidas y prototipado de scripts de Selenium.

Grid:

Selenium Grid es una herramienta que permite la ejecución paralela de pruebas en múltiples navegadores y sistemas operativos. Es útil para reducir el tiempo de ejecución de pruebas al distribuir las pruebas en diferentes entornos de prueba simultáneamente.

Características Clave de Selenium:

Multiplataforma: Selenium es compatible con varios sistemas operativos como Windows, macOS y Linux, y con diferentes navegadores web como Chrome, Firefox, Safari, Edge, entre otros.

Lenguajes de Programación: Puede ser utilizado con diferentes lenguajes de programación como Python, Java, C#, Ruby, JavaScript, entre otros. Esto proporciona flexibilidad para los desarrolladores que prefieren trabajar en su lenguaje de programación favorito.

Interacción con Elementos Web: Selenium permite la identificación y manipulación de elementos HTML en una página web mediante selecciones por ID, clase, nombre, XPath, etc. Esto permite simular acciones de usuarios como clics, relleno de formularios, navegación por páginas, entre otros.

Pruebas Automatizadas: Es ampliamente utilizado en la automatización de pruebas funcionales y de regresión en aplicaciones web. Permite escribir scripts que simulan el comportamiento del usuario, verifican el funcionamiento de la aplicación y generan informes detallados de los resultados de las pruebas.

Desarrollo

El proyecto se centró en explorar y aplicar diversas herramientas y técnicas fundamentales en el análisis de datos, machine learning, procesamiento de lenguaje natural, desarrollo web y automatización. A continuación se describe el desarrollo de cada una de estas áreas.

Análisis de Datos

1. **Pandas:**

Estructuras de datos: DataFrame y Series para manipulación y análisis de datos tabulares.
Funcionalidades: Lectura y escritura de datos en múltiples formatos, indexación, selección, y operaciones de manipulación y transformación de datos.

2. **Matplotlib:**

Visualización: Creación de gráficos estáticos, animados e interactivos, incluyendo gráficos de línea, dispersión, histogramas, barras y gráficos 3D.

3. **NumPy:**

Computación numérica: Arreglos multidimensionales (ndarray) y operaciones matemáticas rápidas y eficientes, como álgebra lineal y funciones estadísticas.

Machine Learning

1. **Scikit-learn:**

Algoritmos: Implementación de modelos supervisados y no supervisados, incluyendo regresión, clasificación y clustering.
Herramientas: Preprocesamiento de datos, validación de modelos y ajuste de hiperparámetros para desarrollar y evaluar modelos predictivos.

Procesamiento de Lenguaje Natural (NLP) **Clasificación y generación de contenido:** Aplicación de técnicas para analizar y generar texto, como la clasificación de correos electrónicos y la creación de descripciones automáticas.

Desarrollo Web

1. **Flask:**

Microframework web: Desarrollo de aplicaciones web dinámicas que integran modelos de machine learning, proporcionando una interfaz de usuario intuitiva.

Automatización

1. **Selenium:**

Automatización de navegadores: Ejecución de pruebas automáticas y extracción de datos web mediante la simulación de acciones del usuario.

Ejercicios Prácticos en Python Implementación de funciones, manipulación de cadenas, uso de diccionarios, manejo de archivos y creación de clases, consolidando conceptos clave de programación.

Proyectos Desarrollados

1. **Proyectos básicos:**

Calculadora de operaciones básicas.
Juego de adivinanza de números.
Conversor de monedas.
Aplicación de gestión de tareas.

2. **Proyectos avanzados:**

Análisis de datos con Pandas.
Machine learning básico y clasificación.
Procesamiento de lenguaje natural.
Desarrollo web con Flask.
Automatización de tareas web con Selenium**.**

Conclusiones

Este proyecto ha demostrado cómo la integración de diversas tecnologías y herramientas puede facilitar la resolución de problemas complejos y la creación de soluciones innovadoras. Las principales conclusiones son:

1. **Versatilidad y Eficiencia de las Herramientas de Python:**

Pandas, Matplotlib y NumPy: Estas bibliotecas han mostrado ser fundamentales para el análisis y visualización de datos, permitiendo una manipulación eficiente y la creación de gráficos de alta calidad.

Scikit-learn: Su amplia gama de algoritmos y herramientas ha facilitado el desarrollo de modelos de machine learning robustos y precisos.

2. **Aplicaciones Prácticas de NLP:**

Las técnicas de clasificación y generación de texto han sido efectivas para tareas como la categorización de correos y la creación automática de contenido.

3. **Desarrollo Web con Flask:**

Flask ha demostrado ser un framework ligero y flexible, ideal para crear aplicaciones web dinámicas que integran modelos de machine learning.

4. **Automatización con Selenium:**

Selenium ha proporcionado una forma eficiente de automatizar tareas repetitivas en navegadores web, mejorando la productividad y precisión en pruebas y extracción de datos.

5. **Consolidación de Conceptos de Programación:**

Los ejercicios prácticos y proyectos han permitido aplicar conceptos teóricos de programación en escenarios del mundo real, fortaleciendo las habilidades técnicas y la comprensión de cada tecnología.

Documentación clara y completa de los códigos

Parte 1:

Archivos

archivos.py

```python
def copiar_archivo(origen, destino):
    try:
        with open(origen, 'r') as archivo_origen:
            contenido = archivo_origen.read()
        with open(destino, 'w') as archivo_destino:
            archivo_destino.write(contenido)
        print("El contenido del archivo se ha copiado correctamente.")
    except FileNotFoundError:
        print("Error: No se pudo encontrar el archivo.")
    except IOError:
        print("Error: Ocurrió un error al leer o escribir el archivo.")

copiar_archivo('archivo1.txt', 'archivo2.txt')

```

cuenta bancaria

cuenta bancaria.py

```python
class CuentaBancaria:
    def __init__(self, titular, saldo=0):
        self.titular = titular
        self.saldo = saldo

    def depositar(self, cantidad):
        if cantidad > 0:
            self.saldo += cantidad
            print(f"Depósito de {cantidad} realizado. Nuevo saldo: {self.saldo}")
        else:
            print("La cantidad a depositar debe ser mayor que cero.")

    def retirar(self, cantidad):
        if 0 < cantidad <= self.saldo:
            self.saldo -= cantidad
            print(f"Retiro de {cantidad} realizado. Nuevo saldo: {self.saldo}")
        else:
            print("Fondos insuficientes o cantidad inválida para retirar.")

    def consultar_saldo(self):
        print(f"Saldo actual de la cuenta de {self.titular}: {self.saldo}")

def interactuar_con_cuenta(cuenta):
    while True:
        print("\n¿Qué desea hacer?")
        print("1. Consultar saldo")
        print("2. Depositar dinero")
        print("3. Retirar dinero")
        print("4. Salir")

        opcion = input("Ingrese el número de la opción deseada: ")

        if opcion == '1':
            cuenta.consultar_saldo()
        elif opcion == '2':
            cantidad = float(input("Ingrese la cantidad a depositar: "))
            cuenta.depositar(cantidad)
        elif opcion == '3':
            cantidad = float(input("Ingrese la cantidad a retirar: "))
            cuenta.retirar(cantidad)
        elif opcion == '4':
            print("Saliendo del programa...")
            break
        else:
            print("Opción no válida. Por favor ingrese un número del 1 al 4.")

titular = input("Ingrese el nombre del titular de la cuenta: ")
saldo1 = float(input("Ingrese el saldo inicial de la cuenta: "))

cuenta = CuentaBancaria(titular, saldo1)
interactuar_con_cuenta(cuenta)
```

Factorial

factorial.py

```python
def factorial(n):
    resultado = 1
    for i in range(1, n + 1):
        resultado *= i
    return resultado

numero = int(input("Ingresa un número entero para calcular su factorial: "))
print(f"El factorial de {numero} es: {factorial(numero)}")
```

Invertircadena

invertircadena.py

```python
def invertir_cadena(cadena):
    invertida = ""
    for caracter in cadena:
        invertida = caracter + invertida
    return invertida
cadena1 = "Apruebeme profe"
cadena2 = invertir_cadena(cadena1)
print(cadena2)
```

Listatelefonica

listatelefonica.py

```python
agenda = {}

def agregar():
    nombre = input("Ingresa el nombre del contacto: ")
    telefono = input("Ingresa el número de teléfono del contacto: ")
    agenda[nombre] = telefono
    print(f"Contacto '{nombre}' agregado correctamente.")

def buscar():
    nombre = input("Ingresa el nombre del contacto que deseas buscar: ")
    if nombre in agenda:
        print(f"Nombre: {nombre} - Teléfono: {agenda[nombre]}")
    else:
        print(f"El contacto '{nombre}' no está en la agenda.")

def mostrar():
    print("Agenda Telefónica:")
    for nombre, telefono in agenda.items():
        print(f"Nombre: {nombre} - Teléfono: {telefono}")

while True:
    print("\n1. Agregar contacto")
    print("2. Buscar contacto")
    print("3. Mostrar agenda")
    print("4. Salir")
    opcion = input("Selecciona una opción : ")

    if opcion == '1':
        agregar()
    elif opcion == '2':
        buscar()
    elif opcion == '3':
        mostrar()
    elif opcion == '4':
        print("Saliendo del programa...")
        break
    else:
        print("Opción no válida. Por favor, selecciona una opción válida .")

```

Parte 2:

Adivinar

adivinar.py

```python
import random


def jugar_adivina_numero():
    print("Bienvenido al juego Adivina el Número!")
    print("Estoy pensando en un número entre 1 y 100. Intenta adivinarlo.")

    numero_secreto = random.randint(1, 100) #creamos el numero aleatorio
    intentos = 0

    while True:
        intentos += 1
        try:
            guess = int(input("Ingresa tu número (o '0' para salir): "))
        except ValueError:
            print("Por favor ingresa un número válido.")
            continue

        if guess == 0:
            print("Has decidido salir del juego. El número secreto era", numero_secreto)
            break
        elif guess < 1 or guess > 100:
            print("El número debe estar entre 1 y 100.")
        elif guess < numero_secreto:
            print("El número secreto es mayor. Intenta de nuevo.")
        elif guess > numero_secreto:
            print("El número secreto es menor. Intenta de nuevo.")
        else:
            print(f"Felicidades! Adivinaste el número secreto {numero_secreto} en {intentos} intentos.")
            break

jugar_adivina_numero()
```

Calculadora

calculadora.py

```python
def sumar(a, b):
    return a + b
def restar(a, b):
    return a - b
def multiplicar(a, b):
    return a * b
def dividir(a, b):
    if b != 0:
        return a / b
    else:
        return "Error: división por cero"
def calculadora():
    while True:
        print("\nCalculadora básica")
        print("Seleccione la operación:")
        print("1. Suma")
        print("2. Resta")
        print("3. Multiplicación")
        print("4. División")
        print("5. Salir")

        opcion = input("Ingrese el número de la operación deseada: ")

        if opcion == '1':
            num1 = float(input("Ingrese el primer número: "))
            num2 = float(input("Ingrese el segundo número: "))
            resultado = sumar(num1, num2)
            print(f"Resultado de la suma: {resultado}")

        elif opcion == '2':
            num1 = float(input("Ingrese el primer número: "))
            num2 = float(input("Ingrese el segundo número: "))
            resultado = restar(num1, num2)
            print(f"Resultado de la resta: {resultado}")

        elif opcion == '3':
            num1 = float(input("Ingrese el primer número: "))
            num2 = float(input("Ingrese el segundo número: "))
            resultado = multiplicar(num1, num2)
            print(f"Resultado de la multiplicación: {resultado}")

        elif opcion == '4':
            num1 = float(input("Ingrese el numerador: "))
            num2 = float(input("Ingrese el denominador: "))
            resultado = dividir(num1, num2)
            print(f"Resultado de la división: {resultado}")

        elif opcion == '5':
            print("Saliendo de la calculadora...")
            break

        else:
            print("Opción no válida. Por favor ingrese un número del 1 al 5.")

calculadora()

```

Cambiomoneda

cambiomoneda.py

```python
tasas_de_cambio = {
    'USD': 1.0,  # Dólar estadounidense
    'EUR': 0.85,  # Euro
    'GBP': 0.77,  # Libra esterlina
    'JPY': 110.15,  # Yen japonés
    'CAD': 1.21  # Dólar canadiense
}


def convertir_moneda(monto, moneda_origen, moneda_destino):
    if moneda_origen in tasas_de_cambio and moneda_destino in tasas_de_cambio:
        tasa_origen = tasas_de_cambio[moneda_origen]
        tasa_destino = tasas_de_cambio[moneda_destino]
        monto_convertido = monto / tasa_origen * tasa_destino
        return monto_convertido
    else:
        return None

def main():
    print("Bienvenido al convertidor de moneda")
    print("Las monedas disponibles son: USD, EUR, GBP, JPY, CAD")

    while True:
        monto = float(input("Ingrese el monto a convertir: "))
        moneda_origen = input("Ingrese la moneda de origen (ej. USD): ").upper()
        moneda_destino = input("Ingrese la moneda de destino (ej. EUR): ").upper()

        if moneda_origen not in tasas_de_cambio or moneda_destino not in tasas_de_cambio:
            print("Moneda no válida. Por favor ingrese una moneda válida.")
            continue

        resultado = convertir_moneda(monto, moneda_origen, moneda_destino)

        if resultado is not None:
            print(f"{monto} {moneda_origen} equivale a {resultado:.2f} {moneda_destino}") # el :.2f hace que tenga 2 decimal
        else:
            print("No se pudo realizar la conversión. Por favor verifique las monedas ingresadas.")

        continuar = input("¿Desea realizar otra conversión? (s/n): ")
        if continuar.lower() != 's':
            break


if __name__ == "__main__":
    main()
```

tareas4

tareas4.py

```python
def mostrar_menu():
    print("\n===== Lista de Tareas =====")
    print("1. Mostrar tareas")
    print("2. Agregar tarea")
    print("3. Marcar tarea como completada")
    print("4. Eliminar tarea")
    print("5. Salir")
    print("============================")

def mostrar_tareas(tareas):
    print("\n===== Tareas Pendientes =====")
    if not tareas:
        print("No hay tareas pendientes")
    else:
        for i, tarea in enumerate(tareas, 1):
            estado = "Completada" if tarea['completada'] else "Pendiente"
            print(f"{i}. [{estado}] {tarea['descripcion']}")
    print("============================")

def agregar_tarea(tareas):
    descripcion = input("Ingrese la descripción de la tarea: ")
    tarea = {'descripcion': descripcion, 'completada': False}
    tareas.append(tarea)
    print(f"Tarea '{descripcion}' agregada correctamente.")

def marcar_completada(tareas):
    mostrar_tareas(tareas)
    try:
        indice = int(input("Ingrese el número de la tarea que desea marcar como completada: ")) - 1
        if 0 <= indice < len(tareas):
            tareas[indice]['completada'] = True
            print(f"Tarea '{tareas[indice]['descripcion']}' marcada como completada.")
        else:
            print("Número de tarea fuera de rango.")
    except ValueError:
        print("Entrada no válida. Ingrese un número válido.")

def eliminar_tarea(tareas):
    mostrar_tareas(tareas)
    try:
        indice = int(input("Ingrese el número de la tarea que desea eliminar: ")) - 1
        if 0 <= indice < len(tareas):
            descripcion = tareas[indice]['descripcion']
            del tareas[indice]
            print(f"Tarea '{descripcion}' eliminada correctamente.")
        else:
            print("Número de tarea fuera de rango.")
    except ValueError:
        print("Entrada no válida. Ingrese un número válido.")

# Función principal para ejecutar la aplicación
def main():
    tareas = []
    while True:
        mostrar_menu()
        opcion = input("Ingrese el número de la opción que desea realizar: ")

        if opcion == '1':
            mostrar_tareas(tareas)
        elif opcion == '2':
            agregar_tarea(tareas)
        elif opcion == '3':
            marcar_completada(tareas)
        elif opcion == '4':
            eliminar_tarea(tareas)
        elif opcion == '5':
            print("Saliendo del programa...")
            break
        else:
            print("Opción no válida. Por favor ingrese un número del 1 al 5.")

# Ejecutar la aplicación
if __name__ == "__main__":
    main()
```

Parte 3:

Parte1

Matplotlib(1)

matplotlib1.py

```python
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 2.5])
plt.ylabel('Valores')
plt.show()
```

Numpy(2)

numpy2.py

```python
import numpy as np
#creo una matriz
matriz = np.array([[ 0,  1,  2,  3],
                   [ 4,  5,  6,  7],
                   [ 8,  9, 10, 11]])

print("Matriz original:")
print(matriz)
print()
submatriz = matriz[:2, 1:3]

print("Submatriz extraída (primeras 2 filas y columnas 1 y 2):")
print(submatriz)
print()
submatriz[0, 0] = 99

print("Submatriz después de modificar submatriz[0, 0] a 99:")
print(submatriz)
print()
print("Matriz original después de modificar la submatriz:")
print(matriz)

```

Panda(2)

panda2.py

```python
import pandas as pd
import numpy as np

df = pd.DataFrame({'Nombre': ['jack', 'jane', 'jack', 'jane', 'jack', 'jane', 'jack', 'jane'],
                   'Estado': ['SFO', 'SFO', 'NYK', 'CA', 'NYK', 'NYK', 'SFO', 'CA'],
                   'Genero': ['A', 'A', 'B', 'A', 'C', 'B', 'C', 'A'],
                   'Edad': np.random.uniform(24, 50, size=8),
                   'Salario': np.random.uniform(3000, 5000, size=8)})

suma_por_nombres = df.groupby('Nombre').sum()
#se está utilizando el método groupby de pandas para agrupar los datos del DataFrame df
#por la columna 'Nombre'. Después, se aplica el método sum() sobre los grupos resultantes.

print("Suma por nombres:")
print(suma_por_nombres)
print()
agregaciones = {
    'Edad': ['max'],
    'Salario': ['max']
}
maximos_por_nombre_estado = df.groupby(['Nombre', 'Estado']).agg(agregaciones)

print("Edad máxima y salario máximo por nombre/estado:")
print(maximos_por_nombre_estado)

```

Parte2

Scikit-Learn(diabetes)

diabetes.py

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_diabetes #para asignar variables
from sklearn.model_selection import train_test_split #para dividir y probar el conjunto de datos
from sklearn.linear_model import LinearRegression, Ridge, Lasso #para usar las regresiones
from sklearn.preprocessing import StandardScaler

# Cargar el conjunto de datos
diabetes = load_diabetes()
X = diabetes.data
y = diabetes.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Inicializar los modelos
linear_reg = LinearRegression()
#crea un objeto de regresión con regularizacion
ridge_reg = Ridge(alpha=1.0)  # Usaremos alpha=1.0 por defecto
lasso_reg = Lasso(alpha=0.1)  # Usaremos alpha=0.1 por defecto

# Ajustar los modelos
linear_reg.fit(X_train, y_train)
ridge_reg.fit(X_train, y_train)
lasso_reg.fit(X_train, y_train)

coefficients = {
    'Linear Regression': linear_reg.coef_,
    'Ridge': ridge_reg.coef_,
    'Lasso': lasso_reg.coef_
}

# Visualizar los coeficientes
plt.figure(figsize=(10, 6))
plt.barh(range(len(linear_reg.coef_)), linear_reg.coef_, color='b', align='center', label='Linear Regression'
plt.barh(range(len(ridge_reg.coef_)), ridge_reg.coef_, color='g', align='center', label='Ridge')
plt.barh(range(len(lasso_reg.coef_)), lasso_reg.coef_, color='r', align='center', label='Lasso')
plt.yticks(range(len(diabetes.feature_names)), diabetes.feature_names)
plt.xlabel('Coefficient Value')
plt.ylabel('Feature')
plt.title('Comparison of Coefficients')
plt.legend()
plt.tight_layout()
plt.show()
```

Parte3

Clasificación(2)

ejercicio2.py

```python
import numpy as np
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error # para calcular el error cuadratico medio

diabetes = load_diabetes()
X = diabetes.data
y = diabetes.target
#elimina las columnas de índice 0 y 8 de la matriz
X_filtered = np.delete(X, [0, 8], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X_filtered, y, test_size=0.2, random_sta
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
#calcular el error cuadratico medio
mse = mean_squared_error(y_test, y_pred)

print(f"Mean Squared Error (MSE) con características menos útiles eliminadas: {mse:.2f}")
'''
La característica "contract" probablemente contiene información crucial
sobre el tipo de contrato del paciente. Esto puede estar directamente relacionado
con su nivel de atención médica y, por lo tanto, afectar la respuesta a tratamientos
y la evolución de la enfermedad. Al eliminar esta característica,
el modelo podría perder parte de su capacidad para capturar estas variaciones,
lo que podría resultar en una reducción en la precisión de las predicciones.
'''
```

Parte4

Metricas de clasificacion(4)

ejercicio4.py

```python
import numpy as np
import matplotlib.pyplot as plt

def entrenamiento(df, y, C=1.0):
    cat = df[categoricas + numericas].to_dict(orient='records')
    dv = DictVectorizer(sparse=False)
    dv.fit(cat)

    X = dv.transform(cat)

    modelRL = LogisticRegression(solver='liblinear', C=C)
    modelRL.fit(X, y)

    return dv, modelRL

nfolds = 10
nrepeats = 10
rkf = RepeatedKFold(n_splits=nfolds, n_repeats=nrepeats, random_state=1)
valores_C = [0.001, 0.01, 0.1, 0.5, 1, 10]
auc_scores = []
for C in valores_C:
    auc_scores_C = []
    for train_idx, val_idx in rkf.split(df_train_completo):
        df_train = df_train_completo.iloc[train_idx]
        df_val = df_train_completo.iloc[val_idx]

        y_train = df_train.churn.values
        y_val = df_val.churn.values

        dv, modelo = entrenamiento(df_train, y_train, C=C)
        y_pred_proba = modelo.predict_proba(dv.transform(df_val[categoricas + numericas].to_dict(orient='records')))[:

        auc = roc_auc_score(y_val, y_pred_proba)
        auc_scores_C.append(auc)
    auc_scores.append(auc_scores_C)
auc_scores = np.array(auc_scores)
plt.figure(figsize=(10, 6))
plt.boxplot(auc_scores.T, labels=valores_C)
plt.xlabel('Valor de C')
plt.ylabel('Puntuación AUC')
plt.title(f'Distribución de las puntuaciones AUC con {nfolds}-fold {nrepeats}-repeats CV')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Parte5

Generación de textos(ejercicio dentro de la teoria)

ejercicio dentro de la teoria.py

```python
import pdfplumber # PARA EXTRAER TEXTO
import pytesseract # PARA RECONOCER TEXTOS E IMAGENES
from PIL import Image #ABRIR Y MANIPULAR IMAGENES
from google.cloud import translate_v2 as translate  # CAMBIO DE IDIOMA
from bs4 import BeautifulSoup # PARA ANALIZAR DOCUMENTOS HTML Y XML
import requests #DESCARGAR DE WEB
import nltk #PROCESAMIENTO DEL LENGUAJE
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.probability import FreqDist

translate_client = translate.Client()


pdf_text = """
I hope you are well. I wanted to thank
him for his dedication in teaching this course.
 I have been working hard and I have learned.
  (I'm Rios) I write this here so that I can consider this part that I had to,
  please approve me to transfer to chemistry: c. I very much appreciate your time and consideration.
"""
imagen = Image.open('imagen.jpeg')

url = 'https://www.example.com'

def extract_text_from_pdf(pdf_text):
    sentences = sent_tokenize(pdf_text)
    words = word_tokenize(pdf_text)
    stop_words = set(stopwords.words('english'))
    filtered_words = [word.lower() for word in words if word.isalnum() and word.lower() not in stop_words]

    freq = FreqDist(filtered_words)
    top_sentences = sorted(sentences, key=lambda x: sum(freq[word.lower()] for word in word_tokenize(x) if word.lower() not in stop_words), reverse=Tr

    return top_sentences


def translate_text(texto, target_language='es'):

    translation = translate_client.translate(texto, target_language=target_language)
    return translation["translatedText"]

def ocr_text_from_image(imagen):

    texto_reconocido = pytesseract.image_to_string(imagen)
    return texto_reconocido

def extract_text_from_web(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    texto_web = soup.get_text()
    return texto_web

print("Resumen del texto del PDF:")
pdf_summary = extract_text_from_pdf(pdf_text)
for sentence in pdf_summary:
```

Parte6

Flask

flask.py

```python
from flask import Flask
app = Flask(__name__)

@app.route("/")
# ruta donde se va a ejecutar la funcion, osea en el navegador
#"/" se va a ejecutar en lapagina principal

def hola_mundo():
    return "Hola mundo!!"
if __name__ == "__main__":
    app.run()
```

Parte7

Proyecto

proyecto.side


```
{
  "id": "c9bad6b7-95d0-4c7e-86e1-4857ecbb85fb",
  "version": "2.0",
  "name": "proyecto",
  "url": "https://univirtual.uni.pe",
  "tests": [{
    "id": "530bf16f-f3fd-4767-b335-766fd0889967",
    "name": "proyectofinal",
    "commands": [{
      "id": "6844404e-0b21-4f9e-bef3-94373191f0f3",
      "comment": "",
      "command": "open",
      "target": "/",
      "targets": [],
      "value": ""
    }, {
      "id": "505d33dc-3e87-4d7d-a72f-9aa7a9343d71",
      "comment": "",
      "command": "setWindowSize",
      "target": "997x870",
      "targets": [],
      "value": ""
    }, {
      "id": "01400150-55f5-4900-bc20-770eba25506d",
      "comment": "",
      "command": "click",
      "target": "linkText=Acceder",
      "targets": [
        ["linkText=Acceder", "linkText"],
        ["css=.login > a", "css:finder"],
        ["xpath=//a[contains(text(),'Acceder')]", "xpath:link"],
        ["xpath=//div[@id='usernavigation']/div[2]/div/span/a", "xpath:idRelative"],
        ["xpath=//a[contains(@href, 'https://univirtual.uni.pe/login/index.php')]", "xpath:href"],
        ["xpath=//span/a", "xpath:position"],
        ["xpath=//a[contains(.,'Acceder')]", "xpath:innerText"]
      ],
      "value": ""
    }, {
      "id": "3d4763da-deb6-4ff3-bba4-af75de0b975c",
      "comment": "",
```