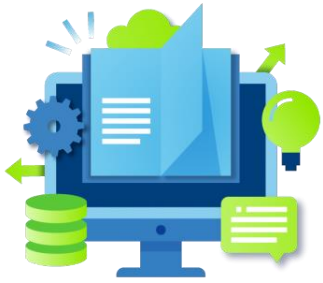
Manual del Usuario

Versión 1.0



Base del Conocimiento

REDU

ESPE

1 Índice

2 Acerca de Sistema

2.1 Información

El sistema relacionado con la Base del Conocimiento fue desarrollado en Python y los datos son almacenados en MongoDB, se divide en tres módulos principales.

Lógica difusa, es aplicada al analizar los coeficientes haralick, una vez que se tiene en que escala se encuentran (alto, medio, bajo), dependiendo el coeficiente, se genera un valor dentro de la escala señalada.

Red neuronal, es la encargada de verificar si los valores ingresados corresponden a una calcificación benigna o maligna, los datos que se analizan son: homogeneidad, contraste, disimilitud, entropía, energía, media, agrupación, lateralidad, profundidad, distancia pezón, distancia torácica, distancia esternón, forma, bordes y área.

Almacenamiento de datos, al tener el análisis completo, se lo guarda en una estructura json que permite almacenar la información en la base de datos en MongoDB.

2.2 Requerimientos

El programa se encuentra ya compilado y disponible para las siguientes plataformas

Hardware (Mínimo)

- PC compatible x86 (i386, i486, Pentium I, II, III, IV, K6, K6-2, Duron, Athlon, etc.)
- 10 Mb libres en el disco rígido

Software

- Windows 7 en adelante
- MongoDB instalado
- Python instalado con las librerías:
  - Pymongo: compatibilidad con MongoDB
  - Keras: importante para la red neuronal

3 Instalación

3.1 Instalación

(Aun no hay un archivo ejecutable por eso esta sección queda pendiente)

4 Uso y formas de operación

4.1 Consideraciones generales

El sistema consta de dos archivos importantes: programa en Python y la base de datos en MongoDB.

4.2 Uso

La base del conocimiento se divide en tres funciones principales que son: lógica difusa, predicción y almacenamiento de información

4.2.1 Lógica Difusa

Es la primera función que debe ser llamada para hacer el análisis de los coeficientes haralick, los datos de entrada que recibe son: homogeneidad, contraste, disimilitud, entropía, energía y media. Además, recibe los meta datos correspondientes para poder llamar a la siguiente función.

4.2.2 Red Neuronal

Una vez que el sistema tiene cada uno de los datos necesarios analizados, se llama a la función predicción, la cual es la encarga de crear el modelo, compilarlo, ajustarlo y finalmente hacer la predicción de la calcificación, es decir estudiar si la calcificación es benigna o maligna. Los datos necesarios y las escalas que se deben utilizar están descritos en Tabla 1.

Dato Utilizado	Descripción	Escala
Calcificación	Existen diferentes tipos de calcificaciones	Benignas: 1. Cutáneas 2. Vasculares 3. Grosera 4. En vara larga 5. Redondeada 6. Puntuada 7. Esférica 8. Aro 9. Calcificación láctea 10. Sutura 11. Distrófica Malignas 12. Amorfas 13. Pleomorfas 14. Finas

		15. Ramificadas 16. Groseras Heterogéneas
Haralick	Los valores haralick que se calculan son: homogeneidad, contraste, disimilitud, entropía, energía, media	1. Alta 2. Media 3. Baja
Agrupación	Manera que se encuentra distribuida la calcificación	1. Agrupada 2. Lineal 3. Segmentaria 4. Regional
Lateralidad	Característica de la ubicación	1. Derecha 2. Izquierda
Profundidad	Característica de la ubicación	1. Anterior 2. Medio 3. Posterior
Distancia Pezón	Característica de la ubicación	Cualquier valor numérico
Distancia Tórax	Característica de la ubicación	Cualquier valor numérico
Distancia Esternón	Característica de la ubicación	Cualquier valor numérico
Forma	Característica de la detección del patrón	1. Ovalada 2. Redonda 3. Circular 4. Irregular
Borde	-	-
Área		Cualquier valor numérico
Maligna o Benigna	Respuesta que da la predicción	1. Maligna 2. Benigna

Tabla 1. Especificación de datos necesarios para el sistema

4.2.3 Almacenamiento

Al finalizar, el usuario sabrá la predicción sobre la calcificación ingresada, toda la información se almacena en la base de datos MongoDB, la cual tiene la siguiente estructura:

```
{
  "calcificaciones": {
    "Nombre": "",
    "Haralick": {
      "Homogeneidad": "",
      "Contraste": "",
      "Disimilitud": "",
      "Entropia": "",
      "Energia": ""
    },
    "Agrupacion": "",
    "Ubicacion": {
      "Lateralidad": "",
      "Profunidad": "",
      "DistanciaPezon": "",
      "DistanciaToracica": "",
      "DistanciaExternon": ""
    },
    "Forma": "",
    "Bordes": "",
    "Area": "",
    "Maligna": "1 o 0"
  }
}
```

Figura 1: Estructura Json correspondiente a calcificación

4.3 Formas de Operación

4.3.1 Lógica Difusa

Aplicada en el análisis haralick, cuando el usuario ingresa la escala de cada uno de los coeficientes, el sistema va procesando que valor asignar a dicho coeficiente, un ejemplo de la manera que se analiza, se observa en la Figura 2.

```
########################## HOMOGENEIDAD ##############
homog=int(input('Homogeneidad: '))
if homog ==1:
    hckHmg=random.uniform(0,0.32)
else:
    if homog==2:
        hckHmg=random.uniform(0.33,0.65)
    else:
        hckHmg=random.uniform(0.66-1)
########################## CONTRASTE #################
nivelGrises=255;
contraste=int(input('Contraste: '))
mini=nivelGrises/3
if contraste ==1:
    hckCntr=random.uniform(0,mini)
else:
    if contraste==2:
        mini=mini+1
        limite=2*nivelGrises/3;
        hckCntr=random.uniform(mini,limite)
    else:
        hckCntr=random.uniform(limite, nivelGrises)
```

Figura 2. Código de análisis de los coeficientes de homogeneidad y contraste

4.3.2 Red Neuronal

Primero se debe cargar datos que funcionen como entrenamiento para la red neuronal, luego se establece las variables de entrada que serán analizadas y la variable de salida que será la predicción.

Segundo se compila el modelo establecido anteriormente y se lo ajusta, dependiendo la cantidad de entrenamiento deseado.
Tercero, una vez que el usuario ingrese todos los datos, se realiza la predicción deseada. El código de la red neuronal está en Figura (3,4,5).

```
dataset = numpy.loadtxt("datosEntrenamiento.csv", delimiter=",")
# dividido en variables de entrada (X) y salida (Y)
X = dataset[:,0:16]
Y = dataset[:,16]
# crea el modelo
model = Sequential()
model.add(Dense(12, input_dim=16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Figura 3. Carga de datos y creación del modelo con variables de entrada y salida

```
# Compila el modelo
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Ajusta el modelo
model.fit(X, Y, epochs=150, batch_size=10)
```

Figura 4. Código para compilación del modelo

```
Z=[nombre,hckHmg,hckCntr,hckDismlt,hckEnt,hckEng,hckMedia,agrupacion,lateralidad,profundidad,distanciaPezon,distanciaToracica,
predictions = model.predict(Z)

maligna = [round(x[0]) for x in predictions]
if maligna == 1:
    print("Es una calcificación maligna")
else:
     print ("No es una calcificacion maligna")
print(maligna)
```

Figura 5. Predicción de la calcificación

4.3.3 Almacenamiento

Una vez finalizado el proceso de análisis de datos, con la ayuda de la librería pymongo, se guarda los datos temporalmente en una estructura json, la cual por comandos es enviada a la base de datos en MongoDB, como se observa en la Figura 6.

```
almacenamiento={"Haralick":{"Homogeneidad":hckHmg,"Contraste":hckCntr,"Disimilitud":hckDismlt,
                "Entropia":hckEnt,"Energia":hckEng},"Agrupacion":agrupacion,
                "Ubicación":{"Lateralidad":lateralidad,
                "Profundidad":profundidad,"DistanciaPezon":distanciaPezon,"DistanciaToracica":distanciaToracica,
                "DistanciaExternon":distanciaExternon},"Forma:":forma,
                "Bordes:":borde,"Area":area,"Maligna":maligna}
mongoClient = MongoClient("mongodb://localhost:27017/")
mydb=mongoClient["Calcificacion"]
mycol=mydb["Calcificacion"]
db=mongoClient.Calcificacion
x=mycol.insert_one(almacenamiento)
```

Figura 6. Almacenamiento de datos

En caso de verificación para saber si se almaceno la información, se puede imprimir el id generado automáticamente. Figura (7, 8).

```
x=mycol.insert_one(almacenamiento)
print(x.inserted_id)
```

Figura 7. Mostrar en pantalla id de la calcificación almacenada

```
5dc8c06129eb727ff086ceb2
```

Figura 8. Id impreso de una calcificación almacenada