Redes Neuronales Artificiales para la Detección del Trastorno Depresivo Persistente y Trastorno Depresivo Mayor.

# Artificial Neural Networks for the Detection of Persistent Depressive Disorder and Major Depressive Disorder.

Ricardo de Jesús González Morales (1), estudiante, I.T. de Tuxtla Gutiérrez. [0.ricardogonzalez.0@gmail.com.](mailto:0.ricardogonzalez.0@gmail.com.)

Rafael Antonio Bolaños Cameras (2), estudiante, I.T. de Tuxtla Gutiérrez, [animasdelmundo2@gmail.com.](mailto:animasdelmundo2@gmail.com.)

Aída Guillermina Cossío Martínez (3), I.T. de Tuxtla Gutiérrez, [acossio\_m@yahoo.com.](mailto:acossio_m@yahoo.com)

## Artículo recibido en MES DIA, AÑO; aceptado en MES DIA, AÑO.

**Resumen.**

*Este trabajo presenta el diseño de un sistema de 3 redes neuronales conectado a un aplicativo web para la detección del trastorno depresivo mayor y distímico mediante salidas probabilísticas derivadas de cada una de las redes neuronales. El modelo implementa redes neuronales tipo feedforward. En el proceso de aprendizaje, se emplean valores obtenidos de exámenes realizados por psicólogos y psiquiatras para diagnosticar los trastornos en un individuo mediante el Inventario Depresivo de Beck. Dicho diagnóstico será realizado mediante el llenado de un formulario en el aplicativo web, el cual pasará los datos a las redes neuronales para realizar la predicción y obtener el resultado del diagnóstico. El aplicativo, así como las redes neuronales, serán desarrolladas en el lenguaje Python.*

**Palabras clave:** Trastorno depresivo mayor, trastorno distímico, Django, Python, redes neuronales.

# Abstract.

*This work presents the design of a system of 3 neural networks connected to a web application for the detection of major depressive and dysthymic disorder through probabilistic outputs derived from each of the neural networks. The model uses feedforward neural networks. In the learning process, values ​​are obtained from tests carried out by psychologists and psychiatrists that are used to diagnose disorders in an individual using the Beck Depressive Inventory. Such diagnosis will be made by filling in a form in the web application, which will pass the data to the neural networks to make the prediction and obtain the result of the diagnosis. The application, as well as the neural networks, will be developed in Python the programming language.*

**Keywords**: Major depressive disorder, dysthymic disorder, Django, Python, neural networks.

# Introducción.

Según la Organización Mundial de la Salud (2019), el Trastorno Depresivo Mayor es un trastorno mental bastante frecuente que tiene como sintomatología, la tristeza, la perdida de interés o placer, sentimiento de culpa, falta de concentración y baja autoestima. Puede llegar a hacerse crónico y recurrente, dificultando el desempeño y calidad de vida de la persona en todos los ámbitos; en la escuela, con la familia y en las relaciones interpersonales. Si este trastorno se hace grave, puede conducir al suicidio. La depresión puede ser tratada con terapias, pero si esta llega a un nivel moderado o grave, se necesitará de medicación y psicoterapia profesional. En México, la Secretaria de Salud nos indica que el Trastorno Depresivo Mayor se manifiesta a través de varios síntomas, como una respuesta a largos periodos de estrés y angustia. Existen multitud de factores que están asociados a la depresión y ansiedad. Los problemas que tenemos diariamente, pueden juntarse y desencadenar algún cuadro de estos padecimientos, especialmente si tenemos antecedentes de haber padecido alguno de estos trastornos. Es difícil que un paciente pueda detectar tempranamente la depresión, regularmente puede llegar a ser confundido con la tristeza, sin embargo, es de vital importancia poder detectarla a tiempo, para que esta no se convierta en un trastorno permanente. La diferencia entre la tristeza y la depresión, es que los síntomas suelen ser más graves, persistentes, e intensos. Este trastorno tiene como tratamiento, la psicoterapia, la cual consiste en ejercicios de relajación, que de manera conjunta ayudan a disminuir los síntomas de tristeza, ansiedad y melancolía, además de que aportan calma y mayor control en situaciones de estrés (IMMS, 2019). Sin embargo, existen casos donde la psicoterapia no es suficiente y se tiene que recurrir a medicación por medio de un especialista psiquiátrico, e incluso en casos más severos se llega a internar al paciente para tenerlo en constante observación, debido al riesgo que conlleva a que la depresión pueda ser el desencadenante de algún intento de suicidio.

Son los jóvenes de edades comprendidas entre 15 a 29 años y en su mayoría pertenecientes a instituciones escolares los que suelen desarrollar este tipo de problemas, y uno de los mayores retos a los que Chiapas se enfrenta es que no todas las instituciones poseen un psicólogo laborando dentro de ellas, además, la cantidad de especialistas en psiquiatría y psicología en nuestro estado es pequeña comparada con la población que se enfrenta a estos trastornos. Aunado a esto, cuando se habla del Trastorno Depresivo Mayor, en una etapa crónica o catatónica, el paciente necesita de medicamentos antidepresivos, e inclusive se puede llegar hasta el grado de hospitalizarlo en algún centro de salud mental por su seguridad, debido a que algunos presentan pensamientos suicidas que, las personas a su alrededor no pueden detectar (Organización Mundial de la Salud, 2019).

El objetivo principal de este sistema de redes neuronales, es que, a partir de diagnósticos realizados previamente por psicólogos, podamos detectar si el paciente presenta trastornos del estado de ánimo, y/o pensamientos suicidas, con el fin de poder canalizarlo con un psicólogo/psiquiatra a una clínica especializada. Estos factores pueden ser de diferente índole, ya sean culturales, sociales, o inclusive religiosos, y que no todos los pacientes presentan en conjunto.

El sistema servirá a:

1. Psicólogos que estén en instituciones educativas y de servicio que no cuentan con una especialidad en el ámbito de Trastorno Mayor Depresivo o Distimia, les proporciona un diagnóstico, donde pueda determinar el tipo y nivel de trastorno para asignar terapias o indicar el apoyo médico de un psiquiatra.
2. Psiquiatras, como un segundo diagnóstico y que pueda determinar un tratamiento, en su defecto para asegurar la respuesta a todas las necesidades psiquiátricas, del paciente.

# Métodos.

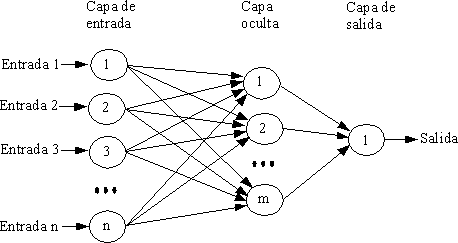
## Redes Neuronales.

Las redes neuronales son otra forma de emular ciertas características propias de los humanos, tales como la capacidad de memorizar y asociar hechos. Si se examinaran con atención todos los problemas que no son capaz de expresar su contenido en un algoritmo hay una característica en común: la experiencia. En sí, las redes neuronales no son más que un modelo artificial y simplificado del cerebro humano. (Matich, Informática Aplicada a la Ingeniería de Procesos – Orientación I, 2001, pág. 4)

Esto mismo hace que sea una herramienta útil para el sistema que queremos realizar ya que un diagnostico se basa en la experiencia previa del psicólogo o psiquiatra para dar un resultado con el cual se puede determinar si una persona tiene o no depresión y la terapia a seguir. Para el funcionamiento correcto de una red neuronal, se necesita de los datasets, que son los archivos que contienen la información con la que será entrenada la red neuronal.

Las redes neuronales pueden ser clasificadas según el tipo de aprendizaje (supervisado, no supervisado), el tipo de aplicación y la arquitectura de la conexión (monocapa, multicapa). Una de las principales características de las redes neuronales, es su capacidad de aprendizaje; a la hora de entrenar la red neuronal, se busca conseguir que una aplicación determinada, para un conjunto de entradas produzca el conjunto de salidas deseadas o mínimamente consistentes. Este proceso de entrenamiento consiste en la aplicación secuencial de diferentes conjuntos o vectores de entrada para que se ajusten los pesos de las interconexiones según un procedimiento predeterminado.

En las redes neuronales artificiales existen dos capas con conexiones, con el mundo exterior. Una capa de entrada, donde se presentan los datos a la red, y una capa de salida que mantiene la respuesta de la red a una entrada. El resto de las capas reciben el nombre de capas ocultas. La Figura 1, muestra el aspecto de una red neuronal artificial multicapa.

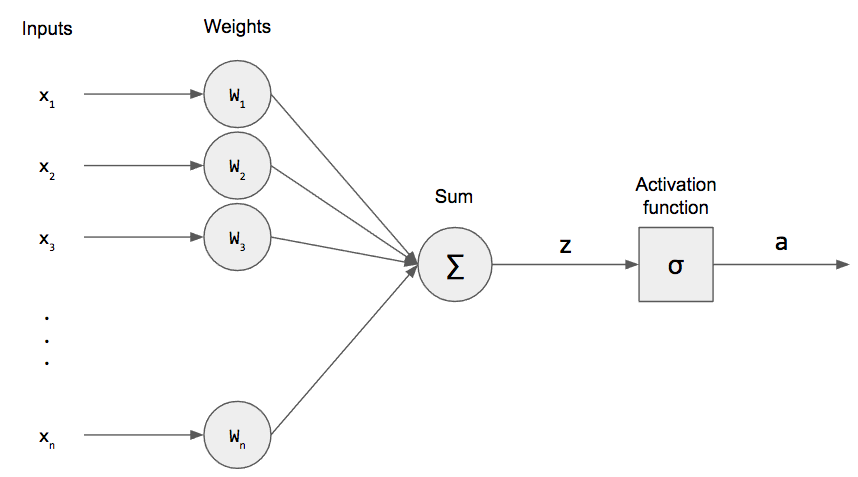


***Figura 1. Arquitectura de una Red Neuronal Artificial Multicapa.***

**Tipos de redes neuronales según su topología:**

1. Red neuronal monocapa o perceptrón simple:

La red neuronal monocapa (figura 2) se corresponde con la red neuronal más simple, está compuesta por una capa de neuronas que proyectan las entradas a una capa de neuronas de salida donde se realizan los diferentes cálculos. Básicamente está compuesta de una sola neurona, con sus respectivas entradas, pesos y función de activación. Principalmente pueden resolver problemas simples como las compuertas lógicas.



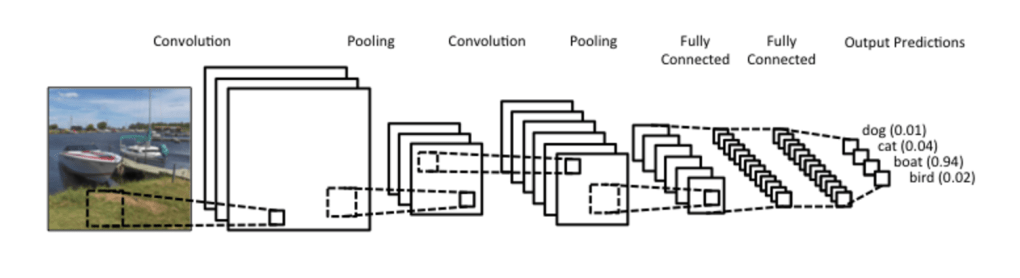
**Figura 2. Perceptrón Simple**

1. Red neuronal multicapa o perceptrón multicapa:

La red neuronal multicapa es una generalización del perceptrón simple, la principal diferencia reside en que mientras la red neuronal multicapa dispone de un conjunto de capas intermedias (capas ocultas) entre la capa de entrada y la de salida (figura 1). Dependiendo del número de conexiones que presente la red esta puede estar total o parcialmente conectada.

1. Red Neuronal Convolucional (CNN)

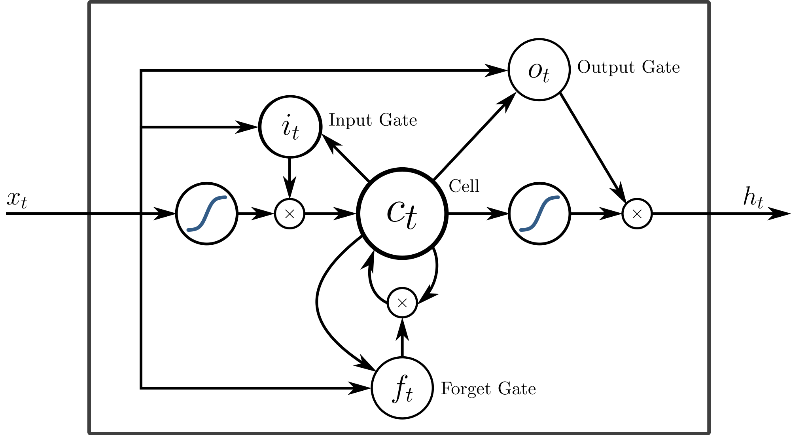
La principal diferencia de la red neuronal convolucional (figura 3) con el perceptrón multicapa viene en que cada neurona no se une con todas y cada una de las capas siguientes, sino que solo con un subgrupo de ellas (se especializa), con esto se consigue reducir el número de neuronas necesarias y la complejidad computacional necesaria para su ejecución. Este tipo de redes son usadas principalmente en tareas de análisis de imágenes, visión artificial o análisis de voz.



**Figura 3. Red Neuronal Convolucional**

1. Red neuronal recurrente (RNN)

Las redes neuronales recurrentes no tienen una estructura de capas, sino que permiten conexiones arbitrarias entre las neuronas, incluso pudiendo crear ciclos, con esto se consigue crear la temporalidad, permitiendo que la red tenga memoria (figura 4). Los datos introducidos en el momento t en la entrada, son transformados y van circulando por la red incluso en los instantes de tiempo siguientes t + 1, t + 2, …



**Figura 4. Red Neuronal Recurrente**

1. Red neuronal prealimentada (Feed-forward)

Una red neuronal prealimentada (es una red neuronal artificial donde las conexiones entre las unidades no forman un ciclo. Estas son diferentes de las redes neuronales recurrentes. La red neuronal prealimentada fue la primera y más sencilla forma de red neuronal artificial ideada. En esta red, la información se mueve en una única dirección: adelante. De los nodos de entrada, a través de los nodos escondidos (si los hay) hacia los nodos de salida. No hay ningún ciclo o bucle en estas redes, un ejemplo es la que podemos observar en la figura 1.

1. Redes de base radial (RBF)

Las redes de base radial calculan la salida de la función en función de la distancia a un punto denominado centro. La salida es una combinación lineal de las funciones de activación radiales utilizadas por las neuronas individuales. Las redes de base radial tienen la ventaja de que no presentan mínimos locales donde la retro propagación pueda quedarse bloqueada.

## Base de Hechos.

Una base de hechos incorpora una representación de hechos referidos al dominio de aplicación del sistema. En este caso, es la espina dorsal de todo el sistema ya que aquí se guardarán las experiencias que la red neuronal deberá almacenar para poder aprender de ellas. Martínez (2006) comenta: “La base de hechos, es relevante para determinar la potencia del sistema con el que se está trabajando ya que este depende de la amplitud de los conocimientos contenidos en ella” (p. 47).

Esto nos dice que la elección de la forma en la que se pueda representar un conocimiento es fundamental para obtener un buen rendimiento del sistema. Esta relación debe permitir identificar conocimientos nuevos siendo lo flexiblemente necesario como para decidir incorporarlo a la base de hechos, eliminar un conocimiento contradictorio u obsoleto o sustituir conocimientos por otros más actualizados o precisos. (Martinez, 2006, p. 48)

## Aplicativo Web.

La World Wide Web (Red Informática Mundial) es un sistema de distribución de hipertexto o hipermedia interconectados y accesibles a través de Internet mediante el uso de los navegadores web. Un usuario de este tipo de sistemas podrá visualizar páginas web compuestas de archivos que regularmente tienen extensión HTML que pueden contener multitud de contenidos multimedia. Estos sistemas web pueden ser accesados navegando a través de hiperenlaces.

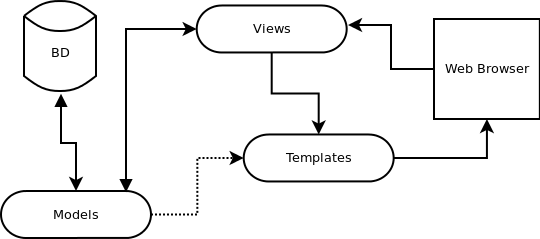
Esta manera de difusión de información ha ido en constante actualización comenzando desde las primeras páginas con formato de un boletín de periódico con el único fin de difundir noticias hasta los aplicativos más complejos como lo son las redes sociales o aplicaciones que permiten una interacción más compleja entre un usuario y el aplicativo.

Gracias al avance informático que se ha presentado en los últimos años, mediante los diferentes lenguajes y frameworks de desarrollo web que se han ido creando, existe un mundo de posibilidades en cuanto a los procesos que pueden ser automatizados. Con el fin de una mejor administración y seguimiento del sistema, hemos concluido que esta tecnología es la indicada para el análisis de información que se desea realizar. En nuestro caso en particular, se requiere integrar Redes Neuronales Artificiales a un Sistema Web que permita el diagnóstico de los trastornos previamente mencionados, para facilitar el acceso de los usuarios a este. Como las Redes Neuronales estarán programadas en el lenguaje Python, se necesita de una aplicación que pueda ser accedida desde cualquier dispositivo por un profesional en la salud mental para usar la Red y realizar diagnósticos un aplicativo web usando las tecnologías Mobile First daría la solución óptima, y para facilitar la compatibilidad entre el aplicativo web y las redes neuronales, se utilizará Django el cual es un framework escrito con Python.

**Django**

Django es un framework de aplicaciones web gratuito y de código abierto (open source) escrito en Python. Un framework web es un conjunto de componentes que facilitan y agilizan el desarrollo de sitios web. Cuando se construye un sitio web, se necesita de un conjunto de componentes similares: una manera de manejar la autenticación de usuarios (registrarse, iniciar sesión, cerrar sesión), un panel de administración para el sitio web, formularios, una forma de subir archivos, etc. Los frameworks sirven para que no se tenga que reinventar la rueda cada vez y se avance de una manera más rápida al construir un nuevo sitio.

Se utilizará el framework previamente mencionado el cual provee una estructura ideal con el patrón MVT. Para Django el Modelo-Vista-Template es similar al Modelo-Vista-Controlador utilizado en otros frameworks como Laravel o Spring Boot.



**Figura 5. Modelo MVT**

El modelo es la capa de acceso a la base de datos. En esta capa contiene toda la información sobre los datos: accesos, validaciones, comportamientos y relaciones. El template (plantilla) es la capa de la presentación. Esta capa contiene los archivos HTML con la estructura de cómo se mostrará el contenido del aplicativo. El view (vista) es la capa de la lógica de negocios. Dentro de esta capa se contiene la lógica que accede al modelo y la delega a la plantilla correspondiente, este es el puente entre los modelos y las plantillas.

Se tiene contemplado el uso de SQLite para almacenar la base de datos utilizadas tanto para la red neuronal como para el aplicativo web. Dentro de la base de datos para el aplicativo, las tablas principales para que el sistema funcione correctamente deben de incluir los diagnósticos generados, el historial de usuarios y los usuarios registrados con sus respectivas relaciones para la gestión de los resultados de la red.

El aplicativo manejara 3 tipos de usuarios:

1.- Anónimo: Todo usuario sin registro previo, pueden visualizar las páginas de registro, inicio de sesión y la página principal del aplicativo.

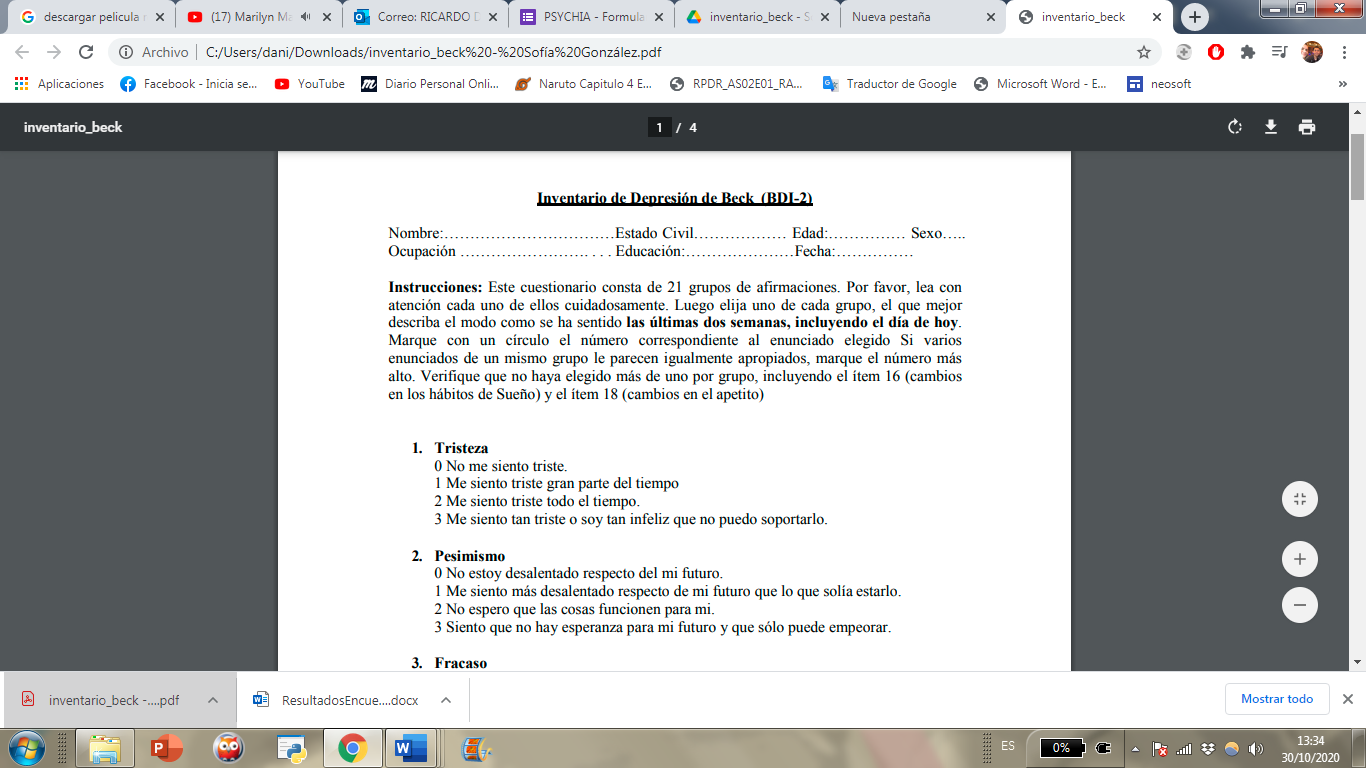
2.- Doctores: Usuarios que hayan sido registrados con éxito demostrando su profesión con la cédula profesional, pueden visualizar la respectiva página de inicio, el historial de diagnósticos realizados por mes/año, la ventana para realizar un nuevo diagnóstico, la página de modificación de credenciales y un apartado estadístico de la cantidad de pacientes que ha diagnosticado y se haya encontrado uno de los trastornos.

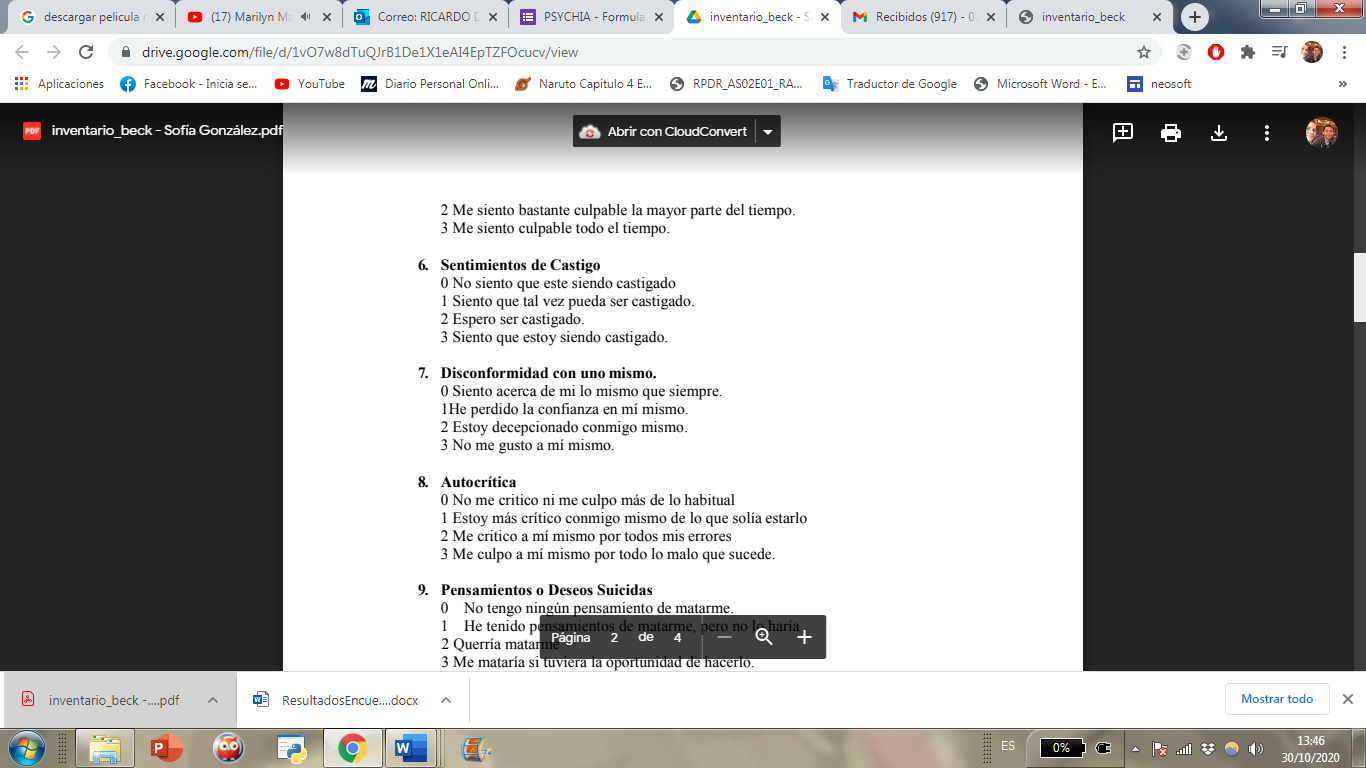
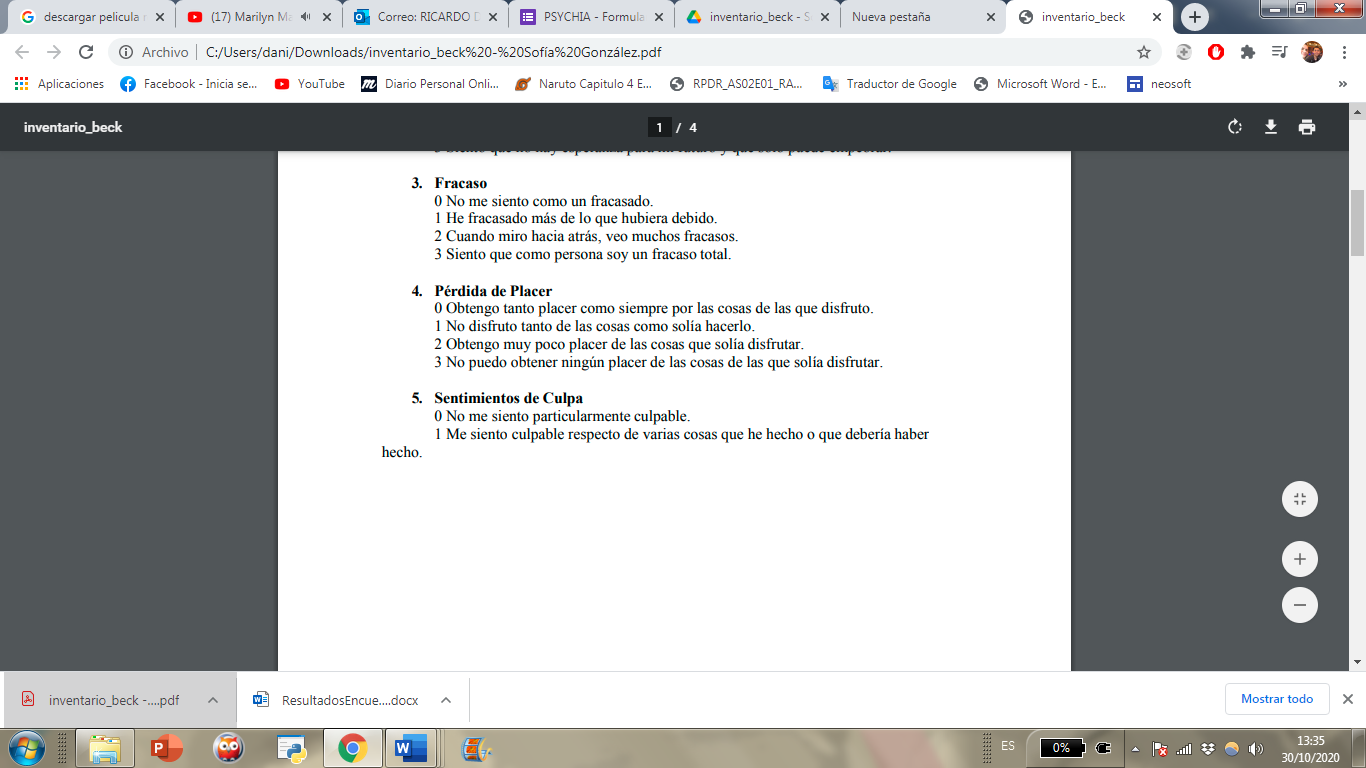
3.- Administradores: Los creadores del aplicativo o personal encargado de la administración de la base de datos. Tendrá acceso a una página donde se muestre una tabla con los usuarios registrados en el sistema por orden alfabético donde podrá editar, eliminar o restringir privilegios a los usuarios para mantener una gestión de las cuentas activas.

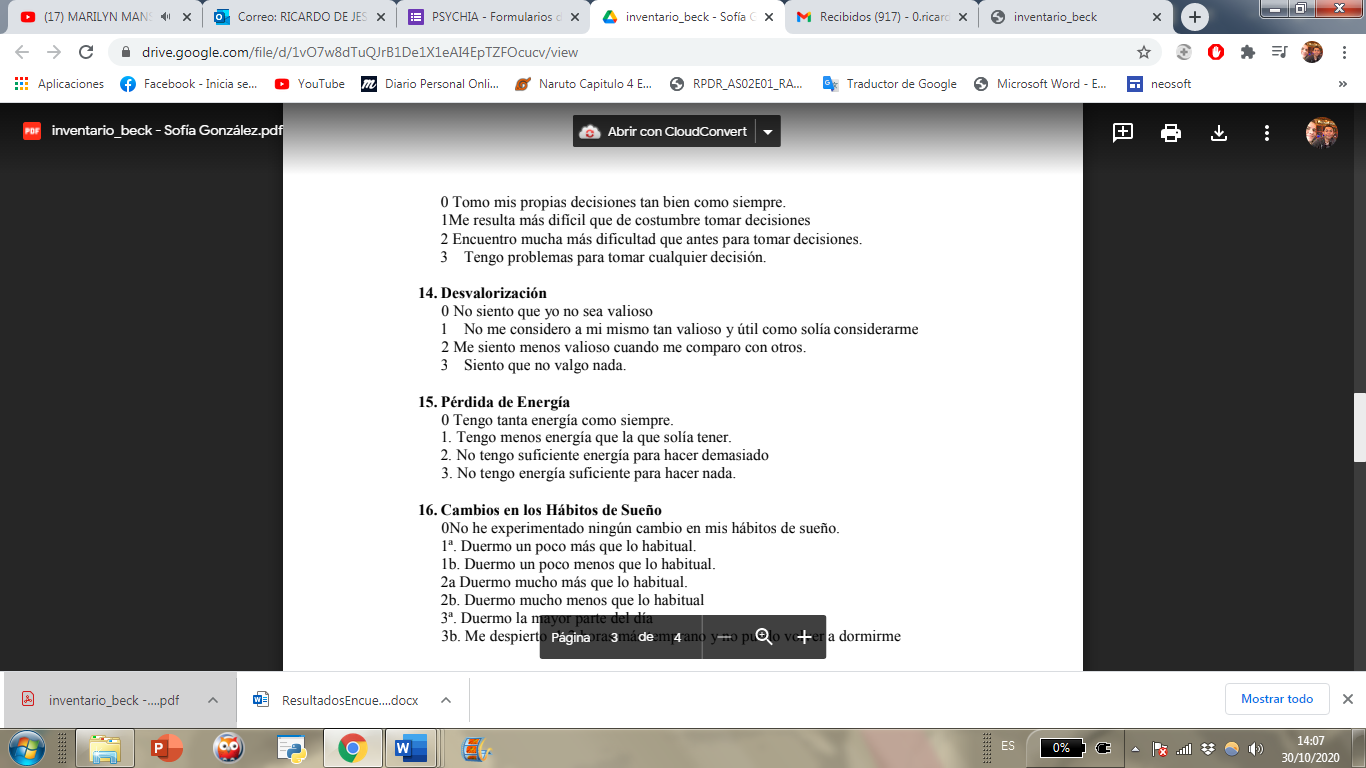
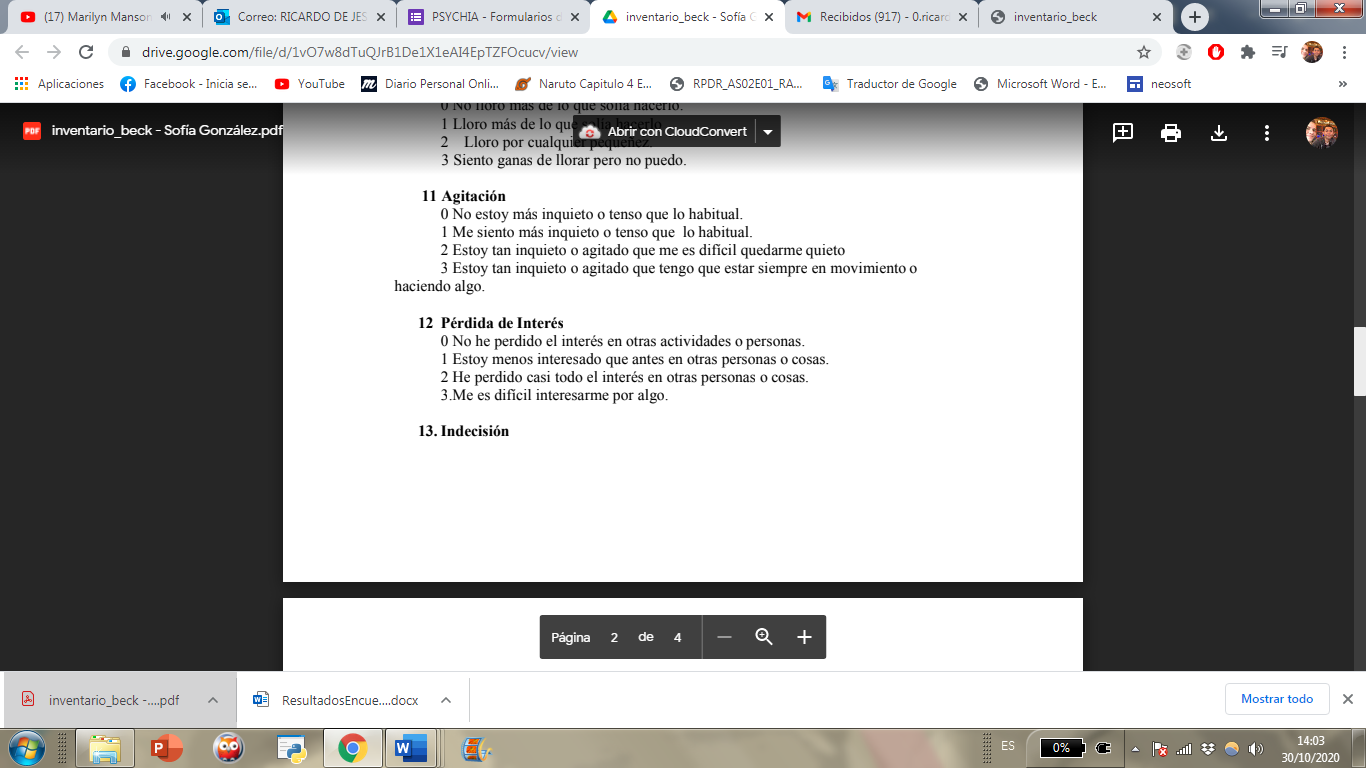
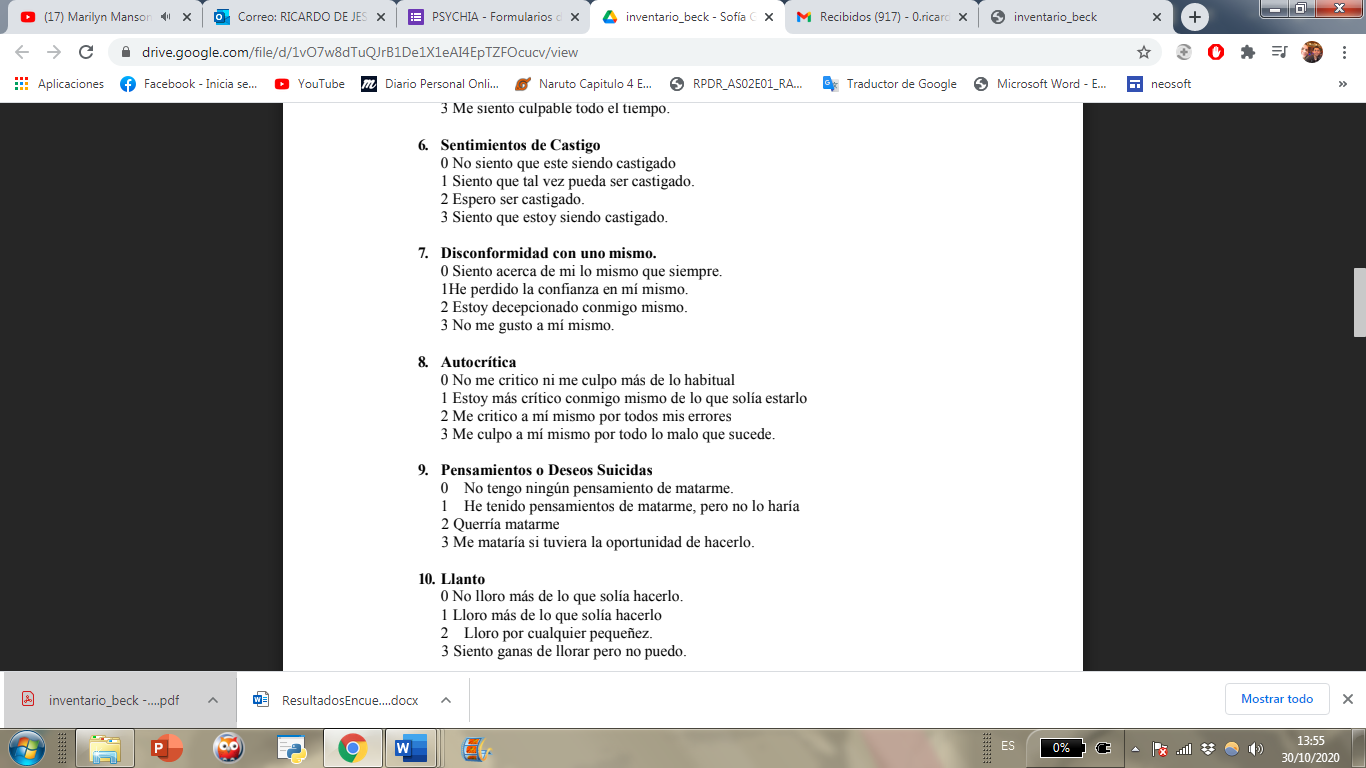
Se utilizará Ajax para realizar peticiones (get, post, put, delete), Bootstrap para los estilos y JQuery para la interacción que tendrá el usuario con el aplicativo. Mediante estas librerías se planea utilizar una interfaz llamativa e intuitiva para el usuario buscando que todo quede como una Single Page Application (SPA).

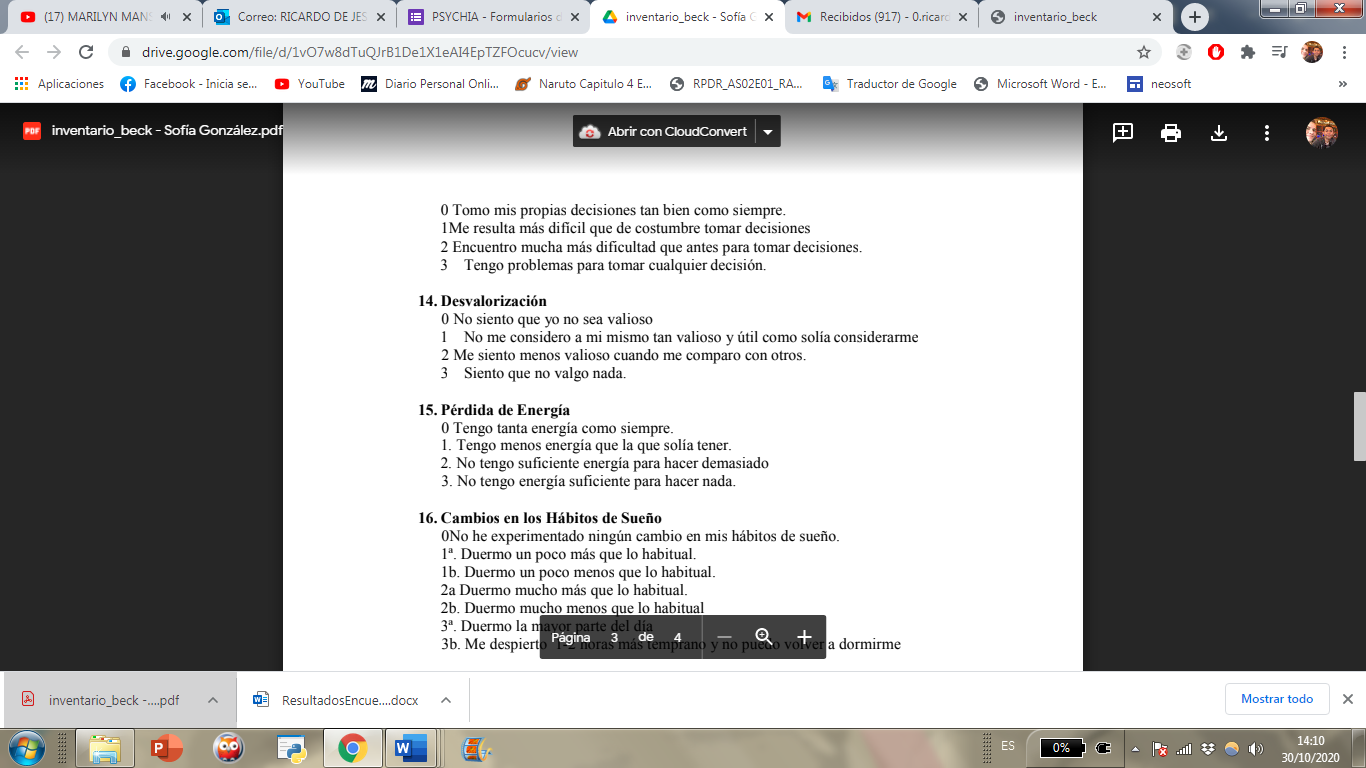
**Inventario Depresivo de Beck**

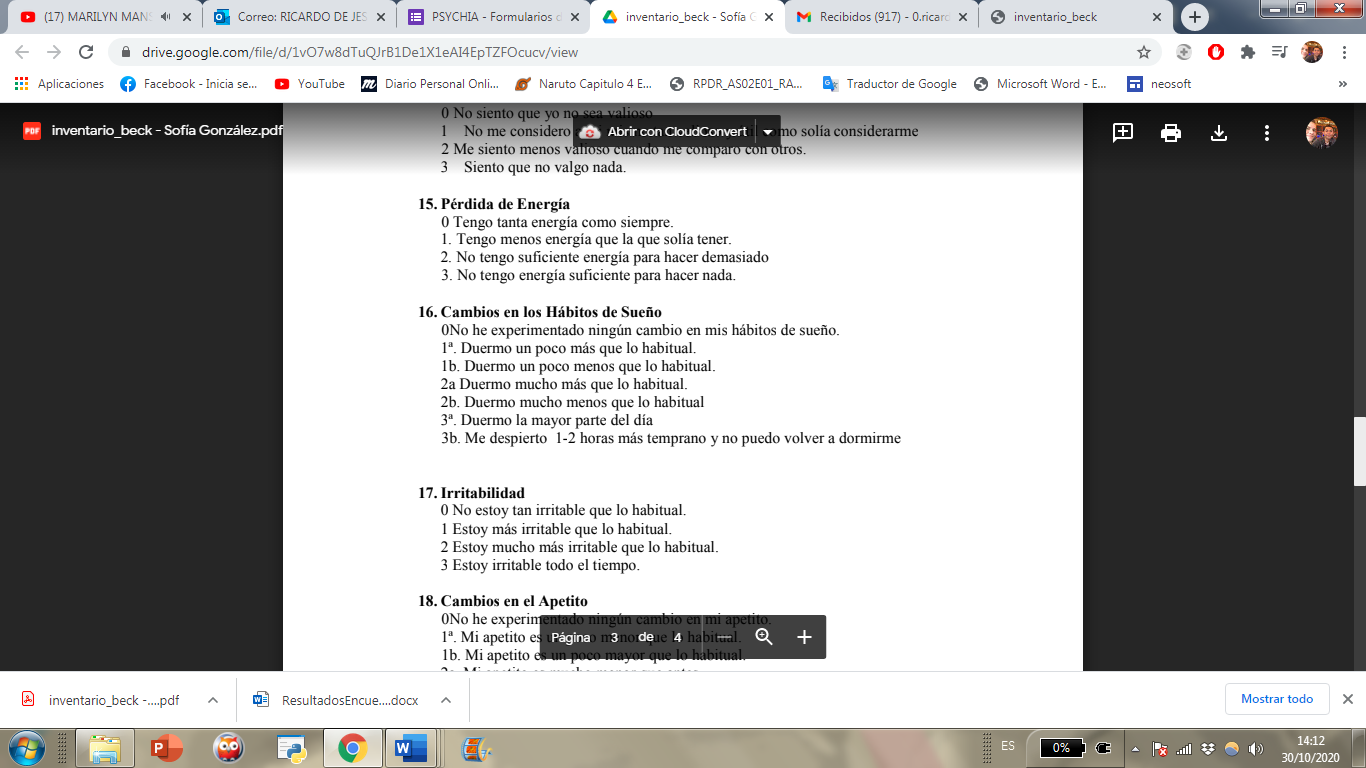
El Inventario de Depresión de Beck-II (también conocido como BDI-II) es un autoinforme de lápiz y papel compuesto por 21 ítems de tipo Likert. El inventario inicialmente propuesto por Beck y sus versiones posteriores han sido los instrumentos más utilizados para detectar y evaluar la gravedad de la depresión. (Consejo General de Consejos Oficiales de Psicólogos, 2013, pág. 3)

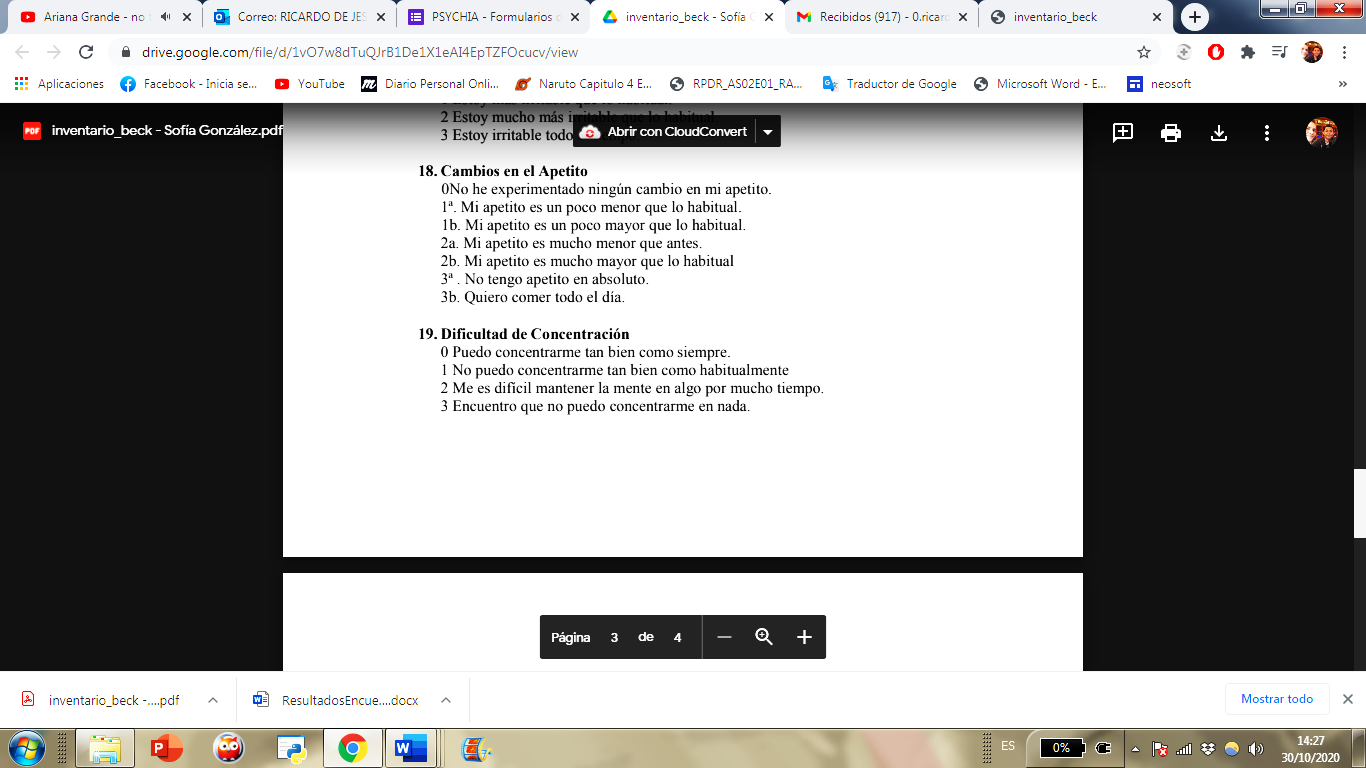












# Desarrollo.

## Aplicativo Web.

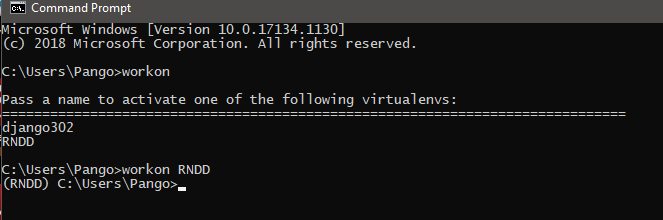
**Creación del Proyecto**

Para poder utilizar Django es necesario tener instalado Python y PIP en el sistema operativo donde se vaya a desarrollar cualquier sistema con este framework, para esto basta con entrar a la página oficial ([www.python.org](http://www.python.org)), ir a la sección de descargas y posteriormente descargar e instalar la última versión de Python ejecutando el archivo ejecutable obtenido de la página oficial de Python.

El otro componente necesario para las librerías y dependencias que se vayan a ocupar en el proyecto es ***PIP***. Para poder instalar PIP basta con ejecutar el comando *python get-pip.py* dentro de la consola de comandos en un sistema operativo que ya tenga instalado Python previamente.

Python tiene un manejador de entornos virtuales, pueden existir N número de estos los cuales pueden configurarse de manera específica para cada proyecto creado, estos servirán para levantar el servidor del proyecto para manejar una configuración adecua dependiendo de la versión del framework o del lenguaje en la que se esté desarrollando. Para esto se tiene que ejecutar el comando *pip install virutalenvwrapper-win* en la consola y esto se encargará de descargar e instalar las paqueterías necesarias. Posteriormente se necesitará instalar ***workon***, la cual es una librería que permite usar los diferentes entornos virtuales creados con la paquetería anterior.

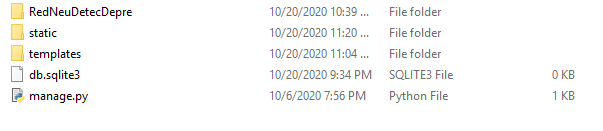
Para crear un entorno se utiliza el comando *mkvirtualenv “Nombre del proyecto sin comillas”* . Se puede ejecutar el comando *workon* para listar los entornos actuales y se debe entrar al entorno actual escribiendo *workon “nombre del proyecto sin comillas”.*



**Figura 6. Comandos para la creación de entornos virtuales**

Una vez se esté dentro del entorno virtual, todas las librerías y paqueterías que se vayan a utilizar y deban ser instaladas al proyecto, se instalaran en el entorno virtual encargado del levantamiento del servidor local del mismo. Para poder usar Django primero debemos instalarlo, lo cual es llevado a cabo mediante el comando *pip install django* dentro del entorno virtual. Posteriormente se debe de crear un proyecto en Django ejecutando el comando *django-admin startproject “nombre del proyecto sin comillas”*. Esto es lo necesario para la configuración básica de un proyecto nuevo en Django.

El comando anterior habrá creado una nueva carpeta con los archivos necesarios para el framework como en la Figura 7. La carpeta *static* contendrá los archivos de estilo .css y los archivos de funcionalidades .js al igual que las librerías que se vayan a incorporar. *Templates* tendrá los archivos .html que tendrán la estructura de como se irán mostrando los datos para el usuario. *RedNeuDetecDepre* es el nombre del proyecto y esta carpeta tiene una estructura interna que se explicara posteriormente. El archivo *db.sqlite3* es el encargado de la base de datos, aquí se crearán las tablas para los usuarios, diagnósticos y resultados junto con sus respectivas conexiones. Por último, *magane.py* es el archivo que utiliza Django para compilar y ejecutar el servidor local con el proyecto, este mismo contiene comandos los cuales configuran ciertas opciones del proyecto.



**Figura 7. Estructura básica de un proyecto en Django**

Dentro de la carpeta con el nombre del proyecto se encuentran varios archivos. Los archivos que se usarán principalmente son *views.py*, *urls.py*, *settings.py*.

1. **Views.py**

Este archivo es el que contiene toda la lógica que se manejará en la página. Una ventaja de Django es que permite separara los módulos mediante *apps*. Estás serán explicadas más adelante, por lo pronto en este archivo se llevará la lógica para el módulo principal del aplicativo el cual tendrá las funciones de inicio de sesión y registro. Tomando como referencia a un framework que maneje MVC, este archivo sería el controlador, pero en este caso es llamado la *vista*.

1. **Urls.py**

Para que la lógica se ejecute primero tiene que llamarse un link/hipervínculo/url, estás serán declaradas dentro de este archivo. Dichas urls contienen el hipervínculo que tendrá que seguir el navegador para acceder, la vista a la que hará referencia y el nombre que se le dará al link.

1. **Settings.py**

Por último, está el archivo encargado de las configuraciones globales para el proyecto. Aquí se define si el proyecto sigue en modo de desarrollo o producción, se importan las librerías descargadas para usar y las rutas para la carpeta static y templates que normalmente no vienen programadas.

Este conjunto de archivos serán los encargados para el módulo del registro de usuarios e inicio de sesión, al igual que la protección de los links a los cuales solo tendrán accesos las personas que ya tengan credenciales válidas para el aplicativo.

Dicho conjunto se puede repetir dependiendo de los módulos que se vayan a manejar, de esta manera podemos mantener la lógica separada para cada sección que se vaya a necesitar. En el caso del aplicativo que se va a desarrollar, necesitamos de dos módulos más. Uno se encargará de los componentes necesarios para los doctores y otro se encargará de lo necesario para los administradores.

Para poder crear los módulos previamente mencionados, se tienen que ejecutar en la consola de comandos que tiene el entorno las siguientes instrucciones: *python manage.py startapp doctores* y *python manage.py startapp administradores* y luego registrarlas dentro del archivo *settings.py*. Se crearán las carpetas respectivas con los mismos archivos que el módulo de inicio. De esta manera se manejarán urls diferentes y vistas diferentes para cada módulo, facilitando la programación del sistema.

Una vez se tengan las configuraciones anteriores, se puede probar compilando el proyecto verificando que todas las configuraciones se hayan cargado de manera correcta y no exista algún error en la sintaxis. Esto se hace ejecutando el comando *python manage.py runserver* (todos los comandos deben ser ejecutados en la carpeta raíz del proyecto, justo donde se encuentra el archivo *manage.py*). Al ser un framework interpretado no se necesita compilar cada cambio, cada que se guarda un documento la consola se reinicia y revisa que todo este correcto, de lo contrario muestra donde se encuentra el error y no permite cargar el aplicativo. De todo estar correcto se mostrará una pantalla de inicio como en la figura 8.



**Figura 8. Página de inicio predetermina Django**

Lo siguiente es crear los *objetos* que se encargaran de almacenar los datos de los usuarios, los diagnósticos y los resultados de estos. Hay un concepto en el mundo de la programación llamado *programación orientada a objetos*. La idea es escribir las tablas como un conjunto de propiedades y acciones (tales como se haría en, por ejemplo, java). Estas tablas que ahora se llamarán *objetos* estarán dentro del archivo *models.py* dentro del módulo respectivo. El modulo de usuarios tendrá los *objetos* para las tablas de registro y roles para los usuarios, doctores tendrá los diagnósticos y los resultados. Estos archivos tendrán la siguiente sintaxis:

from django.db import models

from django.utils import timezone

class ”Nombre del modelo sin comillas” (models.Model):

dato1 = models.ForeignKey('tabla/objeto de donde se obtiene la llave', on\_delete=models.CASCADE) //esto es en caso de que exista una relación con algúna tabla que requiera de una llave foranea

dato2 = models.CharField(max\_length=200)

dato3 = models.TextField()

created\_date = models.DateTimeField(

default=timezone.now)

published\_date = models.DateTimeField(

blank=True, null=True)

def “Nombre de función sin comillas”(self):

self.published\_date = timezone.now()

self.save() //en este caso devuelve la fecha actual para el campo published\_date

Con todos los modelos registrados en sus respectivos archivos hay que agregar estos modelos a la base datos. El primer paso es hacer saber a Django que se han realizado o agregado los nuevos modelos. Esto se hace entrando a la terminal y escribiendo el comando *python manage.py makemigrations “nombre de la app sin comillas”*, en este caso, el comando se ejecutará dos veces, uno para cargar los modelos del módulo de inicio (*python manage.py makemigrations RedNeuDetecDepre)* y el otro para el módulo de doctores (*python manage.py makemigrations doctores)*.

Con este comando Django ha preparado archivos de migración los cuales tienen que ser aplicados a la base de datos mediante el comando *python manage.py migrate “nombre de la app”* (en este caso se sustituiría el nombre por *RedNeuDetecDepre* y luego por *doctores*). De esta manera ya se tendrían los modelos dentro de la base de datos.

Django tiene un administrador propio para las bases de datos, este trae su propio apartado visual para la gestión de las tablas de esta. Primero se debe de crear el archivo *admin.py* en cada app creada donde se registrarán las tablas migradas. La sintaxis es de esta manera:

from django.contrib import admin

from .models import “NombreDelModelo”

admin.site.register(“NombreDelModelo”)

Para ver el modelo en la sección gráfica se debe de crear un super usuario, esto se realiza ejecutando el comando *python manage.py createsuperuser* y posteriormente llenar los campos solicitados escribiendo la respuesta a lo que solicite el *wizard* dando “enter” tras cada entrada. Los campos que solicita son los siguientes:

Username: ejemplo

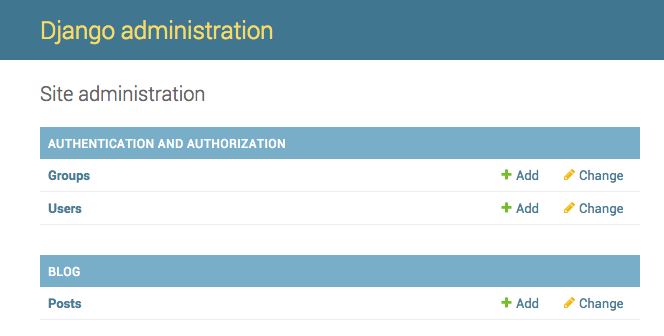
Email address: ejemplo@example.com

Password:

Password (again):

Superuser created successfully.

Una vez se tenga esto, se debe de ingresar mediante la url que otorgue el comando *python manage.py runserver* de la siguiente manera: *url/admin*. Nos desplegará un formulario para el inicio de sesión el cual tras ser llenado mostrará una interfaz con las tablas creadas tras la migración (figura 9). Aquí es donde se podrá ver de manera directa los datos que se van llenado en las tablas, básicamente esta será la pantalla del administrador adaptando los estilos a los colores y logos seleccionados para este proyecto.



**Doctores**

**Diagnósticos**

**Figura 9. Django Administration**

Aquí el administrador podrá directamente hacer modificaciones a los datos en la base, puede agregar a nivel SQL así como eliminar tablas, entre otras operaciones.

Estas serán las configuraciones para la estructura lógica y funcional del proyecto. Las pantallas prototipo serán creadas utilizando la herramienta JustInMind Prototyper para luego ser pasadas a la estructura HTML, agregar estilos con los archivos CSS y agregar las funcionalidades en las vistas de Django.

# Redes Neuronales:

Se llevará a cabo el siguiente proceso de desarrollo para la implementación de cada una de las redes neuronales:

1. Recolección de los datos
2. Preparación de los datos
3. Elección de los hiperparámetros de las redes neuronales.
4. Diseño de la arquitectura de las redes neuronales.
5. Pruebas y validación.
6. Afinación.

# 1. Recolección de los datos.

Para la etapa de recolección de los datos, se necesita recolectar datos de pacientes diagnosticados por profesionales de la salud mental, con el objetivo principal de entrenar las redes neuronales para su posterior aprendizaje. Principalmente los datos que se buscan en esta etapa y para este sistema son:

1. Datos sociodemográficos del paciente diagnosticado tales como la edad, sexo, ocupación, estado civil, etc. Estos datos se encuentran en la ficha de identificación psicológica del paciente.
2. Inventario de depresión de Beck, el cual contiene todos los ítems/métricas relacionados con el Trastorno Depresivo Mayor y Trastorno Distímico, que indican si el paciente presenta uno de los trastornos o ideación suicida.
3. El diagnóstico final de los profesionales, que indica si el paciente padeció/padece de los trastornos mencionados o no, además de la ideación suicida.

Una vez determinados los datos que se requieren, se debe construir el dataset a partir del análisis de características del problema, en nuestro caso, el dataset tendrá la siguiente arquitectura (figura 10):

* Las columnas marcadas por el color verde, corresponden a los datos sociodemográficos del paciente.
* Las columnas color amarillo corresponden a los ítems del Inventario de Depresión de Beck.

# Las columnas color rojo, indican las salidas/etiquetas de la red neuronal, y corresponden a los trastornos a predecir.

**Figura 10. Arquitectura del dataset**

# 2. Preparación de los datos.

Cuando recolectamos los datos en el paso anterior, lo más seguro es que estos datos estén representados bajo nuestro propio lenguaje (español en esta instancia), el cual no es útil cuando se trata de alimentar una red neuronal. Es por ello que es necesario realizar una serie de pasos para que los datos recolectados sean útiles en el proceso de aprendizaje de las redes neuronales. El primer paso que se debe llevar a cabo es la codificación de todos aquellos datos que no sean números, como por ejemplo sexo, estado civil, profesión, etc. A este proceso se le conoce como **Label Encoding**. Estas columnas están representadas por palabras, las cuales no pueden ser procesadas por una red neuronal, por consiguiente, gracias a la codificación de etiquetas, podemos convertir los datos de estas columnas en un número representativo, por ejemplo:

Para codificar estos datos de ejemplo (figura 11), representados por cadenas de caracteres en español…



**Figura 11 Datos sin codificación**

Podemos definir un id para cada categoría en las columnas (figura 12), por ejemplo:



**Figura 12. Codificación de las etiquetas/categorías de cada columna del dataset**

E intercambiarlas las columnas del dataset, con sus respectivos ID (figura 13).

# 

**Figura 13. Dataset codificado**

Sin embargo, esto no es suficiente para que una red neuronal pueda comprender y entender los datos de manera que exista un aprendizaje. Esto es debido a que Label Encoding tiene un problema: los valores numéricos pueden ser malinterpretados por algunos algoritmos.

En nuestro ejemplo hemos codificado varias profesiones con los valores 0, 1, 2 y 3, por lo que ¿esto significa que la profesión correspondiente al valor 3 es el triple que la que ha recibido el valor 1 (según algún criterio)? La respuesta claramente es NO.

Una alternativa al Label Encoding es el método llamado One Hot Encoding. Este es un formato ampliamente usado para datos categóricos, también llamado “codificación categórica”. La estrategia que implementa es crear una columna para cada valor distinto que exista en la característica que estamos codificando y, para cada registro, marcar con un 1 la columna a la que pertenezca dicho registro y dejar las demás con 0 (Chollet, 2018).

En nuestro caso, el dataset puede ser modificado usando esta estrategia, para aquellas columnas que pueden ser de más de una categoría (Profesión, Escolaridad, Nivel socioeconómico, Religión), de manera que el dataset sea representado con una notación binaria (figura 14).

# 

**Figura 14. Dataset Codificado con One-Hot Encoding**

Con esto tenemos listos los datos de tipo sociodemográficos, ya que, con esta representación, la red neuronal podrá aprender de una manera más eficiente y precisa. De igual manera se tienen que preparar los datos depresivos, que son representaciones de los ítems del Inventario Depresivo de Beck. Como podemos ver en la figura 15, el inventario depresivo de Beck, puntúa para cada uno de los ítems, en una escala de 0 a 3, la variable que representa el grado de malestar del paciente. En cada una de estas escalas, el paciente puede pertenecer solamente a una categoría de cada ítem. Estos nos indica que, de igual manera que en los datos sociodemográficos, podemos usar el One Hot Encoding, ya que cada ítem contiene 4 categorías mutuamente excluyentes, por lo que podemos codificar en base a una notación binaria. Por ejemplo, en el caso de que los pacientes tuvieran lo siguientes datos en los primeros 5 ítems:



**Figura 15. Datos sociodemográficos sin codificación**

Utilizando el One Hot Encoding, para los primeros 5 ítems, el dataset se modificaría de la siguiente manera (figura 16):



**Figura 16. Datos sociodemográficos codificados con One-Hot Encoding**

Las etiquetas de salida, son los pronósticos o resultados a los que queremos llegar, y por supuesto corresponden a los datos que la red neuronal intentará alcanzar, en esta instancia corresponden a las columnas de Suicidio, Depresión y Distímia. Como podemos observar en la figura 17, estos campos de igual manera están codificados en binario, en donde 0 significa la ausencia del trastorno y 1 la existencia.



**Figura 17. Datos de salida del dataset**

Por supuesto, gracias al framework Keras, la codificación del dataset, y el One Hot Encoding, puede ser realizado automáticamente desde el código (figura 18) por medio del método to\_categorical:

# 

**Figura 18. Código para implementar One-Hot Encoding**

Dando como resultado una matriz, de tamaño (5,85) que contiene nuestro dataset codificado (figura 19), podemos compararlo con las imágenes en donde codificamos el dataset manualmente, y ver que ambos corresponden a lo mismo.

# 

**Figura 19. Dataset codificado sin estandarización**

Seria problemático alimentar la red neuronal con datos con diferentes rangos (en este caso la edad no puede ser codificada categóricamente), ya que con el tiempo a pesar de que la red se adaptará a datos heterogéneos, será más difícil para esta aprender.

Chollet nos dice que una práctica muy empleada en el mundo del Deep Learning para tratar con datos de este tipo, es la normalización o estandarización según las características, para esto se sustrae la media de cada columna y se divide entre la desviación estándar esto para que la característica representada en la columna este centrada en 0 y tenga una desviación estándar en común con las otras características. (Chollet, 2018, pág. 86)

A continuación, realizaremos este proceso de normalización en nuestro dataset, para ello usaremos el método **zscore** de **scipy**, dando como resultado la siguiente tabla (figura 20).

# 

**Figura 20 Dataset codificado y estandarizado**

Una vez que los datos están estandarizados, el dataset está listo para alimentar a la red neuronal. La preparación y limpieza de los datos es un paso primordial para el análisis de datos, debido a que muchos algoritmos no funcionan de una forma correcta si los alimentamos con “raw data” que no puede entender. Antes de que comencemos a construir nuestras redes neuronales, es sumamente importante separar el dataset en 3 secciones, las cuales nos servirán para el entrenamiento (los datos que servirán para entrenar a la red neuronal), la validación (la sección que usará la red en el momento de observar cómo se comporta con datos que nunca ha visto), y el test (una última prueba después de haber afinado las redes neuronales, para observar que no existe overfitting).

# 3. Elección de los hiperparámetros de las redes neuronales.

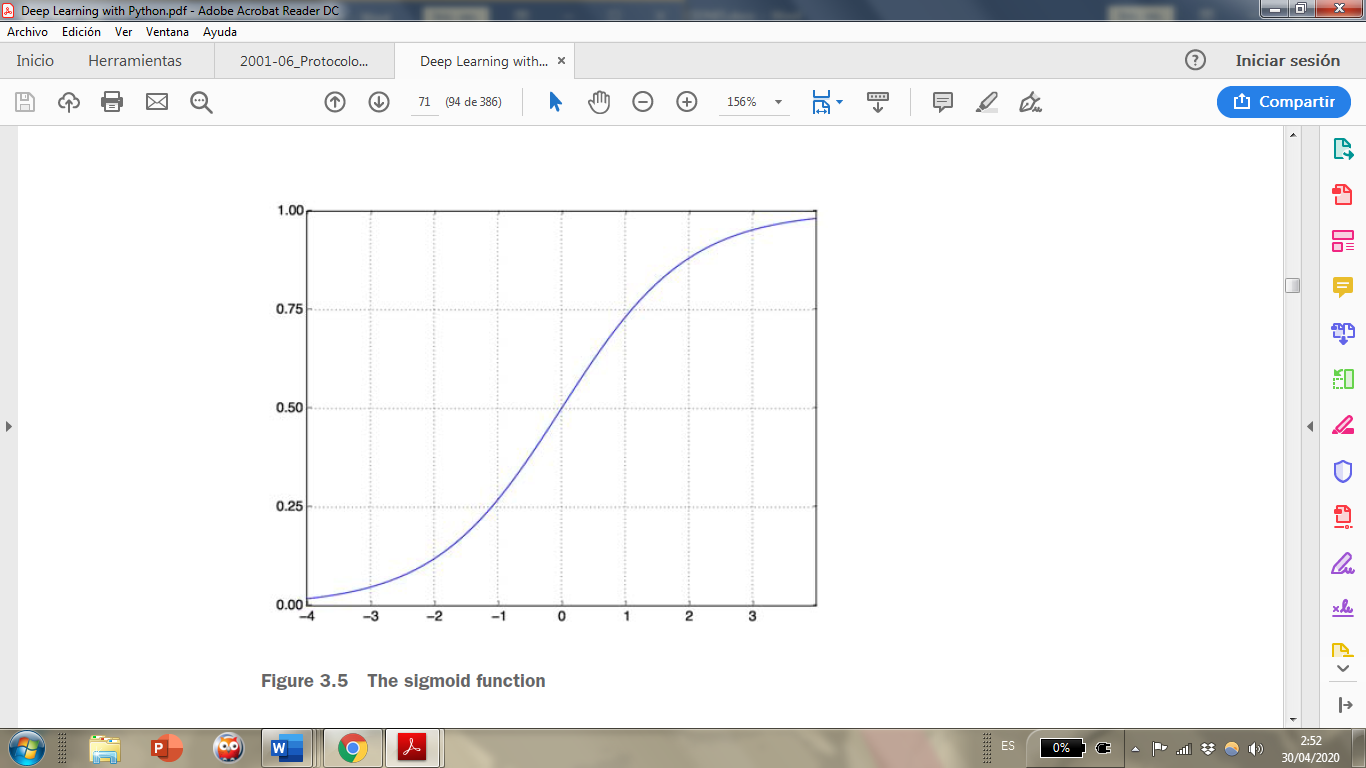
Ya tenemos listos los datos, por lo que a partir de los 3 problemas de clasificación que se quieren resolver con las redes, tendremos que elegir los hiperparámetros que usaremos para cada una:

1. Para la primera red neuronal, la cual se encargará de detectar si el paciente sufre del Trastorno Depresivo Mayor, o Trastorno Distímico se utilizará la función de pérdida “**categorical\_crossentropy**” debido a que es la función determinada por Keras para los problemas de clasificación multiclase (cada trastorno corresponde a una clase excluyente), la capa de salida implementará la función de activación “**softmax**” (figura 21) ya que esta función permite que los datos de salida de la red sean una distribución de probabilidades, cuya suma total dará como resultado 1, lo cual asegura que solamente una etiqueta será asignada a cada una de las muestras de entrada. El optimizador será “**RMSProp**”, ya que es el mejor optimizador que nos proporciona el framework Keras para problemas de clasificación. (Chollet, 2018, págs. 78-84)



**Figura 21. Función Softmax**

1. Para la segunda red neuronal, la cual está encargada de clasificar los tipos de depresión que puede sufrir una persona, (atípico, melancólico, y catatónico), se nos presenta un problema de clasificación multiclase con una sola etiqueta, donde claramente tenemos 3 clases, por lo que los parámetros utilizados y la capa de salida será igual a la anterior red neuronal.
2. En el caso de la tercera red, se nos presenta un problema de clasificación binaria o de “dos clases” (un sí o no), por lo que la función de perdida que utilizaremos será “**binary\_crossentropy**”, la cual es la que nos ofrece Keras para este tipo de problemas binarios (siendo los valores cercanos a 0 la probabilidad de que el paciente no refiera ideación suicida y cercanos a 1 que sí. Nuevamente el optimizador será “**RMSProp**” mientras que la capa de salida tendrá como función de activación “**sigmoid**” (figura 22) debido a que, para todos los valores de x, solamente puede arrojar datos entre 0 y 1, lo cual es lo que se busca. (Chollet, 2018, págs. 68-77)



**Figura 22. Función Sigmoid**

# 4. Diseño de la arquitectura de las redes neuronales.

Para que podamos elegir la arquitectura de cada una de las redes neuronales, hay que analizar las características del dataset que utilizamos, además de lo que se requiere que resuelva el software, para determinar dicha arquitectura usaremos nuestro dataset de ejemplo.

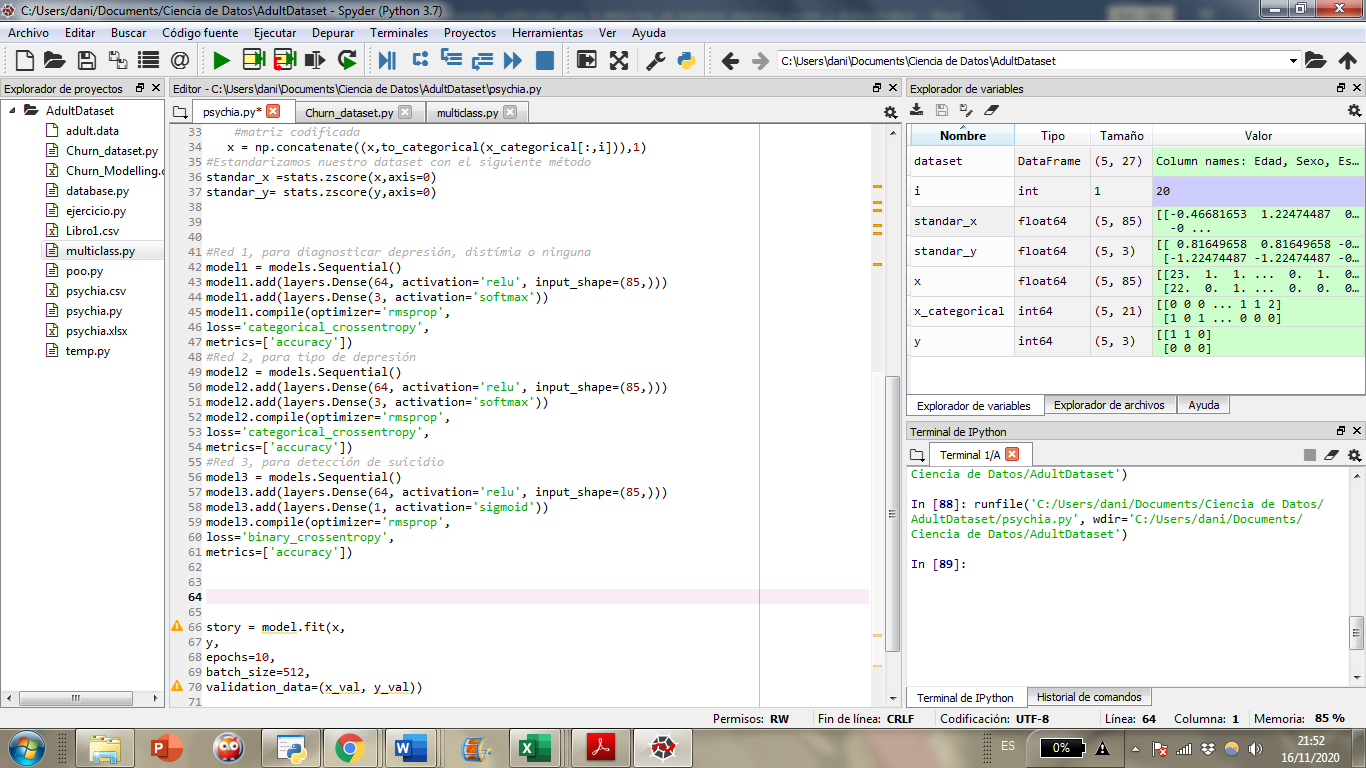
1. La primera red neuronal, será de tipo “clasificación multiclase con una sola etiqueta”, debido a que nos permitirá clasificar si el paciente padece del Trastorno Depresivo Mayor, Trastorno Distímico o ninguno de los anteriores (el paciente solamente puede caer en una de estas situaciones).

Según el número de columnas del dataset, la capa de entrada tendrá 85 neuronas, mientras que la capa de salida estará compuesta de tres neuronas, la cual arrojará la probabilidad de que sea un trastorno u otro (una neurona destinada a la depresión, otra a la distimia y otra a ninguna).

1. Para la segunda red, se mantendrá la arquitectura de 85 neuronas y 3 capas de salida (una para cada tipo de depresión: atípica, melancólica y catatónica).
2. En la tercera red neuronal, se encargará de predecir si el paciente sufre de pensamientos suicidas o no. Para que esta red neuronal se active, primero se deberá llegar a la conclusión de que el paciente sufre de depresión. La capa de entrada implementará las mismas 85 neuronas de entrada, mientras que su salida tendrá una sola neurona ya que la salida será una distribución probabilística y por ende escalar.
3. Las 3 redes neuronales comenzarán con 1 sola capa oculta, esto debido a lo que nos dice:

Esta capa oculta tendrá 64 neuronas, para observar cómo se comportan las redes al principio en el momento de aprendizaje.

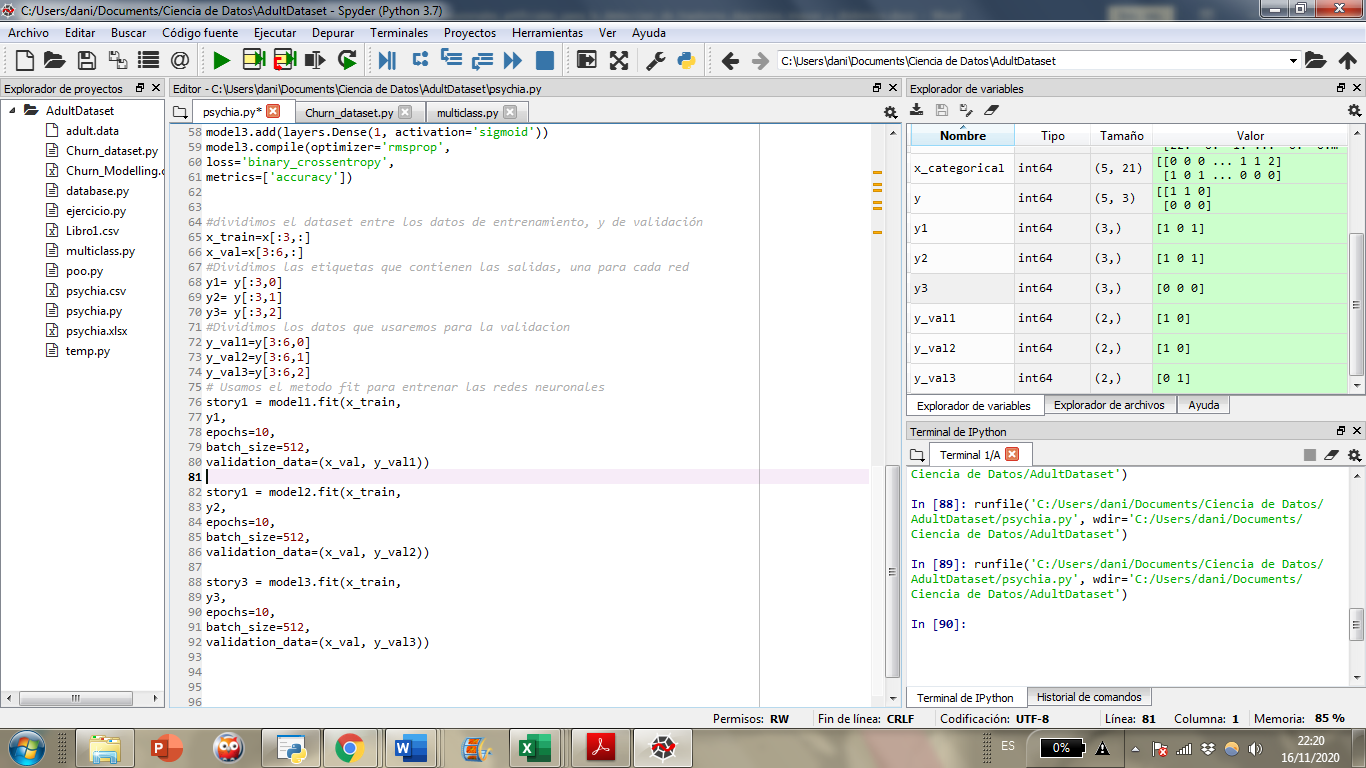
Una vez establecido esto, podemos comenzar con escribir el código de la red neuronal como se ve en la figura 23.



**Figura 23. Código de las redes neuronales**

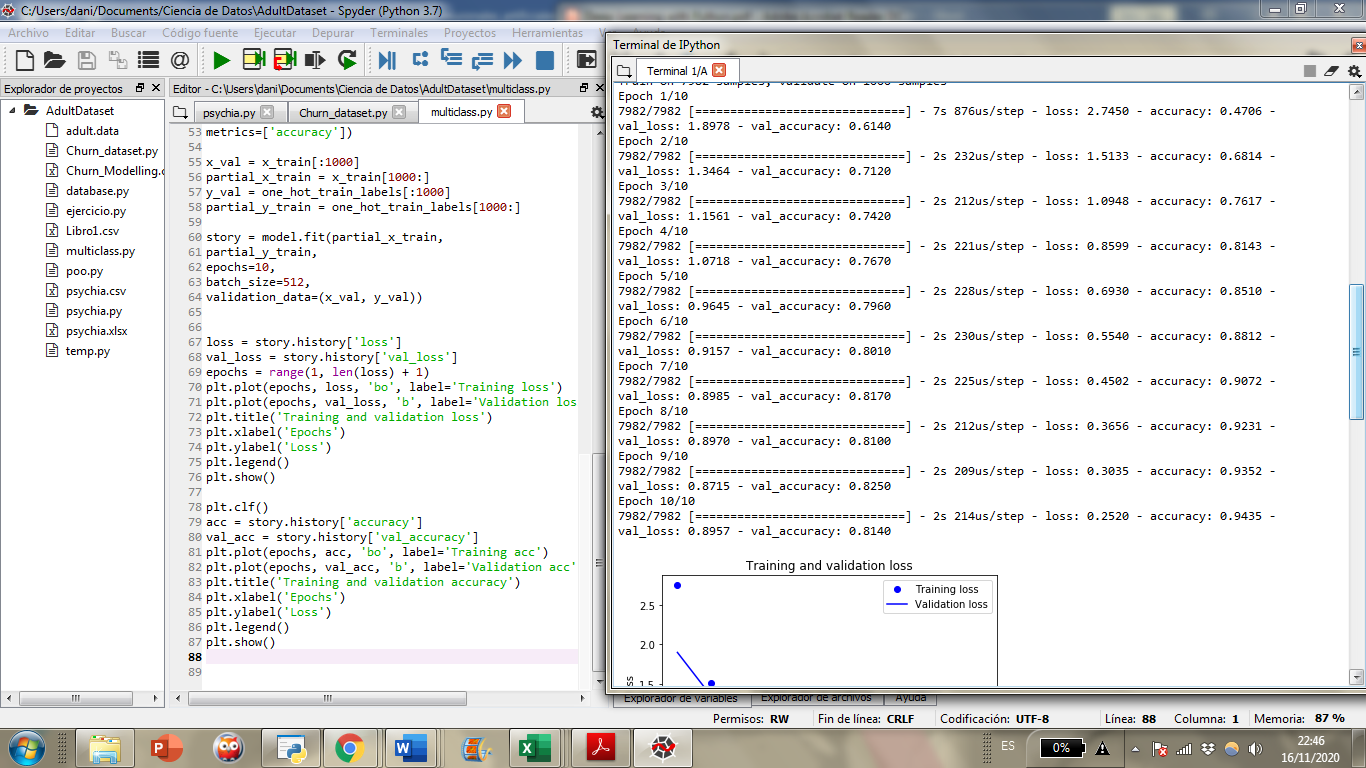
# 5. Pruebas y validación.

Una vez construida la red neuronal, podemos comenzar a entrenarla con nuestros datos de entrenamiento y validación (figura 24).



**Figura 24 Código para entrenar las redes neuronales**

La función “**fit”** es la encargada de entrenar las redes neuronales, y recibe como parámetros los datos de entrenamiento “**x\_train**” , las etiquetas o datos de salida “**y**”, el número de épocas (el cual es el número de veces que el dataset pasará a través de la red neuronal), el tamaño del lote (el número de filas que pasará a través de la red en cada iteración, ya que no se puede pasar el dataset completo inmediatamente), y los datos de validación (que no servirán para visualizar cómo se comporta el modelo neuronal con datos que no existían en el dataset de entrenamiento).



**Figura 25. Red siendo entrenada con 10 épocas**

Podemos observar que mientras aumenta el número de épocas, se reduce la perdida (loss: 0.2520) e incrementa la precisión (accuracy: 94.3%), lo cual es una señal de que el modelo está aprendiendo. La métrica **val\_loss** indica la pérdida en el conjunto de validación la cual de igual manera va disminuyendo, aunque de manera más lenta que la perdida de entrenamiento, y la métrica **val\_accuracy**, nos indica cómo se comporta la red con datos que nunca ha visto, como podemos observar la precisión real de esta red de ejemplo es de 81% .



**Figura 26. Gráfica de la precisión de la red durante el entrenamiento y la validación**

# 5. Afinación.

Ahora que sabemos cómo se comporta el modelo con respecto a la arquitectura que se eligió en cuanto a número de capas y número de neuronas ocultas, podemos comenzar a afinar la red neuronal. Esto se consigue agregando o disminuyendo el número de capas ocultas, el número de neuronas ocultas (la capa de entrada y la de salida no pueden cambiar, ya que estas están establecidas por el problema), eliminando características (columnas) del dataset, cambiando los hiperparámetros de las capas ocultas, por ejemplo, usar otra función de activación, o un optimizador distinto (el optimizador “ADAM”, es bastante utilizado en las redes neuronales feedforward, y es bastante similar a RMSPROP). Una vez que hayamos obtenido los resultados deseados, es importante usar el dataset de test para probar el nuevo modelo, ya que, al estar afinando la red, esta pudo haber obtenido overfitting con respecto al conjunto de validación, incluso sin haber sido entrenado con este. Es por ello la importancia de dividir el dataset en 3 conjuntos.

# Conclusiones.

El uso de un modelo que implementa redes neuronales en este tipo de problemas, donde se tiene que tener cierto conocimiento especializado, es cada vez más frecuente e incluso en los últimos años se ha estado investigando como aprovechar el gran poder de estos algoritmos de inteligencia artificial, en tareas que anteriormente se pensaba que solamente los humanos podían realizar. Las redes neuronales son ampliamente utilizadas en el ámbito médico, para la detección de múltiples enfermedades, tales como el cáncer o tumores (por medio de redes neuronales convolucionales en este último), debido a que han demostrado tener un grado de precisión, por supuesto las métricas y el alcance de estas dependerá del proceso de entrenamiento.

Como pudimos ver en este proyecto, el proceso de construcción de las redes neuronales depende de un análisis de los datos a profundidad, ya que a partir de la arquitectura que tenga nuestro dataset, será el tipo de la red y los hiperparámetros que usaremos para ella. No solo debemos preocuparnos por la red neuronal, si no que la base de todo esta dentro de los datos, no podemos alimentar una red con los datos sin antes procesarlos, es por ello importante llevar a cabo la ingeniería de características y limpieza de datos, de manera que puedan ser una representación útil del problema en cuestión. En la actualidad no existen reglas como tal para adivinar el número de capas ni de unidades ocultas, por lo que el entrenamiento es un proceso iterativo, en donde se tiene que probar una arquitectura para tratar que el modelo muestre resultados óptimos, si estos no son los esperados tendremos que probar modificando aquellos hiperparámetros que puedan ser cambiados, además de la arquitectura de la capa oculta. Sin embargo, las capas de entrada y de salida, dependerán enteramente del problema, por lo que el proceso de afinación de las redes es muy importante si se quiere alcanzar resultados óptimos (mayores a 85% de precisión).

# Agradecimientos.

Le extendemos nuestro principal agradecimiento a la M.C. Aida Guillermina Cossio Martínez por el apoyo que nos han brindado durante el desarrollo del proyecto, así como a la Lic. Sofia Amairani González Morales por facilitarnos el Inventario Depresivo de Beck, además del conocimiento brindado en el ámbito de los trastornos psicológicos.

# Referencias

1. OMS. (s.f de s.f de 2019). *Organización Mundial de la Salud*. Obtenido de Organización Mundial de la Salud: https://www.who.int/topics/depression/es/
2. IMMS. (17 de Noviembre de 2019). *IMMS*. Obtenido de Atención a Conducta Suicida: <https://www.gob.mx/salud/sap/documentos/guia-practica-para-la-atencion-del-pacientecon-conducta-suicida-en-hospitales-generales>
3. OMS. (2 de Septiembre de 2019). *Organización Mundial de la Salud*. Obtenido de Suicidio: <https://www.who.int/es/news-room/fact-sheets/detail/suicide>
4. Matich, D. (s.f de Marzo de 2001). *https://www.frro.utn.edu.ar.* Obtenido de Informática Aplicada a la Ingeniería de Procesos- Orientación I.: ://www.frro.utn.edu.ar/repositorio/catedras/quimica/5\_anio/orientadora1/monogra ias/matich-redesneuronales.pdf
5. Martinez, M. (s.f de s.f de 2006). *repositorio.unican.es.* Obtenido de CONOCIMIENTO Y BASES DE DATOS: UNA PROPUESTA DE INTEGRACION INTELIGENTE: https://repositorio.unican.es/xmlui/bitstream/handle/10902/1172/1de3.MAMcap2.pdf?s equence=2
6. Chollet, F. (2018). *Deep Learning with Python.* Shelter Island, NY 11964: Manning Publications.
7. Consejo General de Consejos Oficiales de Psicólogos. (2013). *EVALUACIÓN DEL INVENTARIO.* Madrid España: EFPA.

# Información de los autores.