

UNIVERSIDAD CATÓLICA SANTO TORIBIO DE MOGROVEJO
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN



**“TRADUCTOR POR VOZ CON PYTHON -
TRABAJO DE INVESTIGACIÓN”**

ALUMNOS:

CHUNG MESONES, CARLOS LEE

CURSO:

DESARROLLO DE APLICACIONES WEB

DOCENTE:

CACHAY MACO, JUNIOR EUGENIO

I. INTRODUCCIÓN:

En la actualidad cada vez es más conocido el término de inteligencia artificial, muchos al escuchar este término debieron haberse imaginado un robot con forma humanoide realizando acciones que un humano puede hacer. Pero eso no está tan alejado de la realidad, hoy en día existen sistemas que emplean modelos de IA para mejorar la experiencia del usuario, analizar datos, procesar voces, incluso hasta para traducir textos de un idioma a otro.

Muchas más personas están interesadas en estas tecnologías actualmente, pero ¿Sabías que ahora mismo ya puedes desarrollar una programa para traducir lo que tu digas por voz? Pues si no lo sabías, hoy te mostraremos un pequeño programa que hemos desarrollado, en python, en donde podrás traducir lo que tu digas a más de 100 idiomas diferentes.

Además, hablaremos sobre los modelos de inteligencia artificial más utilizados actualmente y también las librerías más famosas para desarrollar inteligencia artificial con el lenguaje de programación Python.

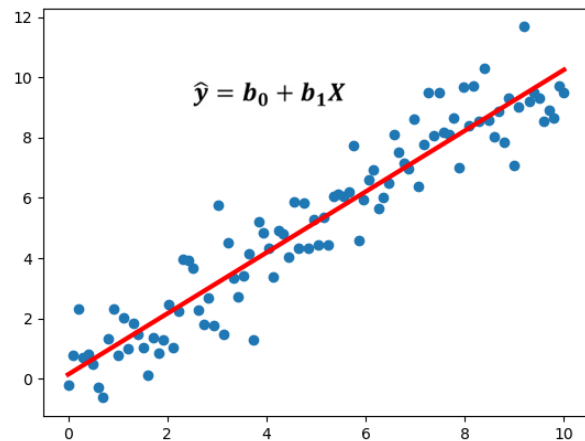
II. MODELOS DE IA MÁS UTILIZADOS EN PYTHON:

A. Regresión Lineal

La regresión lineal es un método paramétrico utilizado para predecir una variable dependiente continua dado un conjunto de variables independientes.

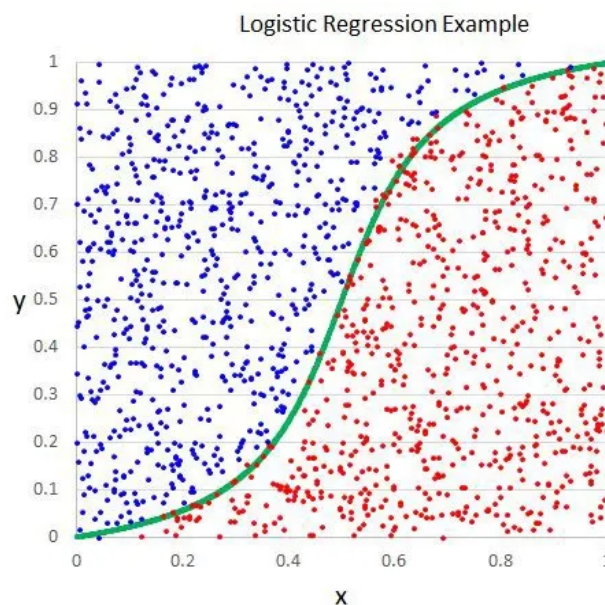
Se puede usar en situaciones en las que desea predecir una cantidad continua, como predecir el tráfico en una tienda minorista, predecir el tiempo de permanencia del usuario o las visitas al blog, etc. Es de naturaleza paramétrica porque hace ciertas suposiciones basadas en el conjunto de

datos. La regresión produce resultados inverosímiles si el conjunto de datos cumple con estos supuestos; de lo contrario, se esfuerza por proporcionar una precisión convincente.[1]



B. Regresión Logística

La regresión logística es un algoritmo de clasificación utilizado para predecir la probabilidad de una variable dependiente categórica. En la regresión logística, la variable dependiente es una variable binaria que contiene datos codificados como 1-0, sí-no, abierto-cerrado, etc. La variable de resultado o objetivo tiene una naturaleza dicotómica. Dicotomía significa que sólo hay dos clases posibles. Por ejemplo, puede usarse para problemas de detección de cáncer o para calcular la probabilidad de un evento.



La regresión logística es uno de los algoritmos de aprendizaje automático más simples y más utilizados para la clasificación binaria. Es fácil de implementar y se puede utilizar como base para cualquier problema de clasificación binaria. Describir y evaluar relaciones entre variables dependientes e independientes binarias.[2]

C. Árbol de Decisiones

Un árbol de decisiones en el aprendizaje automático es una estructura similar a un árbol similar a un diagrama de flujo, donde un nodo interno representa una función (o atributo), una rama representa una regla de decisión y cada nodo hoja representa un resultado.



El nodo superior de un árbol de decisión se denomina nodo raíz en el aprendizaje automático. Aprende a particionar en función de los valores de los atributos. Divide el árbol de una manera recursiva llamada división recursiva. Esta estructura similar a un diagrama de flujo lo ayuda a tomar decisiones. Esta es una visualización similar a un diagrama de flujo que imita fácilmente el pensamiento humano. Por lo tanto, los árboles de decisión son fáciles de entender e interpretar.

Los árboles de decisión clasifican las instancias ordenando el árbol desde la raíz hasta algún nodo hoja que proporciona una clasificación de las instancias, un método llamado de arriba hacia abajo. Cada nodo en el árbol actúa como un caso de prueba para alguna propiedad, y cada borde que desciende de ese nodo corresponde a una de las posibles respuestas del caso de prueba. Este proceso es recursivo y se repite para cada subárbol enraizado en el nuevo nodo.[3]

D. Redes Neuronales Profundas

Una red neuronal profunda (DNN) es una red neuronal artificial (ANN) con múltiples capas ocultas entre las capas de entrada y salida. Al igual que las ANN poco profundas, las DNN pueden modelar relaciones no lineales complejas.

El objetivo principal de una red neuronal es tomar un conjunto de datos de entrada, realizar cálculos progresivamente complejos y luego generarlos para resolver problemas del mundo real, como la clasificación. Simplemente alimentamos la red neuronal. Tenemos una entrada, una salida y un flujo de datos secuenciales en una red profunda. Las redes neuronales son ampliamente utilizadas en problemas de aprendizaje supervisado y aprendizaje por refuerzo. Estas redes se basan en un conjunto de capas interconectadas. [4]

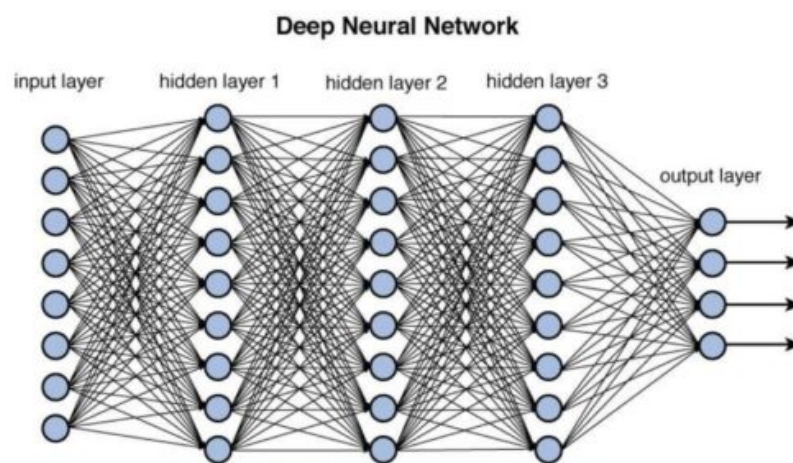


Figure 12.2 Deep network architecture with multiple layers.

III. LIBRERÍAS/BIBLIOTECAS MÁS UTILIZADAS EN PYTHON PARA IA:

A. Tensor Flow

Tensorflow es una de las bibliotecas de código abierto más importantes para el aprendizaje profundo creada por Google. Consistente en un ecosistema flexible de herramientas, bibliotecas y recursos comunitarios, ayuda a los investigadores a innovar utilizando técnicas de aprendizaje automático. Además, permite a los desarrolladores incorporar este tipo de tecnología a sus aplicaciones.



TensorFlow

Tensorflow facilita la creación y el entrenamiento de modelos ML mediante su API. Entre ellos destaca Keras, una API de alto nivel para la creación rápida de prototipos, la investigación avanzada y las soluciones de productos. Entre las características de Keras, destaca su interfaz simple y optimizada para casos comunes, su modularidad mediante bloques, y el hecho de que estos bloques son fáciles de adaptar para aplicar nuevos resultados de última generación.[5]

B. Pandas

Es una de las bibliotecas de manipulación de datos más utilizadas en Python. Entre otras cosas, esta herramienta de análisis y manipulación de datos incluye nuevas funciones de datos basadas en la definición de matriz de la biblioteca NumPy. Le permite leer y escribir fácilmente CSV, archivos con formato Excel y consultar bases de datos SQL.[7]



C. Numpy

Numpy es el estándar de Python, de hecho, Pandas y muchas otras bibliotecas que dependen de él lo usan como base.

Esta biblioteca de Python está diseñada para cálculos matemáticos y análisis de big data. Permite el uso de arrays, que te permiten representar colecciones de un mismo tipo de datos en diferentes dimensiones, así como funciones muy eficientes para manipularlos.[6]



Numpy nos permite crear diversas estructuras numéricas, multidimensionales, nos permite transformarlas, realizar operaciones aritméticas, filtrar y en general es útil para inicializar datos aleatorios.[7]

D. SciPy

La biblioteca Scipy consta de varios módulos que brindan funcionalidad para resolver tareas analíticas y de ciencia computacional. En estos módulos podemos encontrar soluciones de álgebra lineal, integración, optimización, interpolación, procesamiento de señales, procesamiento de imágenes y otros. Las estructuras de datos utilizadas por SciPy son matrices multidimensionales proporcionadas por el módulo Numpy, es decir, SciPy se basa en Numpy para realizar sus operaciones.

Debido al rápido desarrollo de ML, los desarrolladores intentan crear diferentes bibliotecas de Python para el aprendizaje automático. Este hecho llevó a los creadores de SciPy en 2001 a combinar y estandarizar varias funciones, dando como resultado esta biblioteca.[5]



E. Scikit-learn

Una de las principales bibliotecas de Python centradas en el aprendizaje automático es Scikit-Learn. La biblioteca consta de una gran cantidad de algoritmos de ML (clasificación, regresión, agrupamiento, etc.) y proporciona funcionalidades básicas para facilitar el trabajo diario de los ingenieros que trabajan en dichas tareas. Entre sus funciones, podemos destacar herramientas de preprocesamiento de datos, funciones de evaluación de modelos y mecanismos de ajuste para cada parámetro del modelo.[5]



Una de las características más notables es la capacidad de canalizar transformaciones y reutilizarlas. No podemos ignorar que para proyectos comerciales, los datos “en bruto” que nos llegan siempre deben ser transformados de la misma manera para obtener un modelo ML.[6]

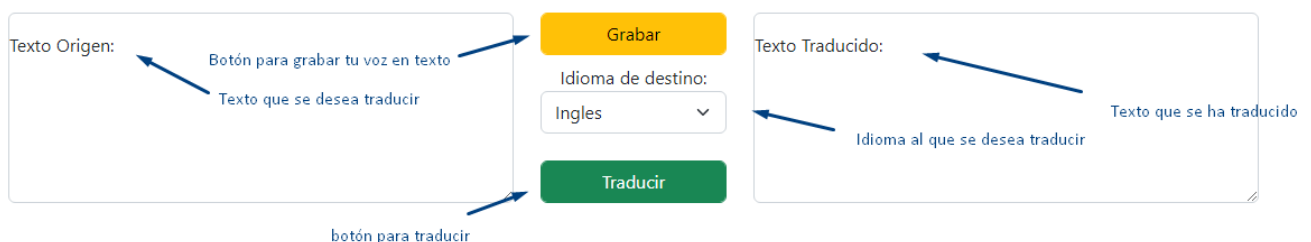
Finalmente, cabe señalar que Scikit-Learn tiene una de las interfaces más fáciles de usar. Esto complementa la ayuda proporcionada por la documentación, que proporciona información detallada sobre algoritmos y funciones, así como ejemplos prácticos, haciendo que el uso de la biblioteca sea una experiencia agradable.[5]

IV. RESULTADOS DE PRÁCTICA DE UNA LIBRERÍA IA:

Para esta demostración optamos por desarrollar una traductor por voz utilizando distintas librerías, tales como:

- googletrans: Librería que utiliza el api de google translate
- speech recognition: Librería que nos permite reconocer la voz y convertirla en texto
- pyaudio: Esta librería es requerida por speech recognition, y la usaremos para poder acceder al micrófono del dispositivo
- pyttsx3: Librería que convierte texto a voz

El resultado final de la interfaz que hemos implementado es la siguiente:



Ahora pasaremos a explicar las funciones de este programa:

1. **Reconocimiento de voz:** Esta es la primera función de este programa y nos permite convertir nuestra voz a texto para su posterior uso. Para esto hacemos uso de la librería speech recognition que emplea en su modelo una red neuronal de aprendizaje profundo. A continuación la funciones que nosotros utilizamos:

```
import speech_recognition as sr
def reconocerTexto():
    r = sr.Recognizer()
    mic = sr.Microphone(device_index=0)
    with mic as source:
        audio = r.listen(source)
    result = r.recognize_google(audio, language='es-Mx')
    return result
```

2. **Traducción de texto:** Esta es la principal función del programa y nos permite traducir un texto de cualquier idioma a un idioma en específico, retornando el texto traducido, para ello hacemos uso de googletrans, una librería que utiliza un modelo de traducción automática neuronal entrenado por google. En la siguiente imagen se aprecia la función en la que nosotros la utilizamos:

```
import googletrans as gt
def traducirTexto(texto, dest):
    p = gt.Translator()
    k = p.translate(texto, dest=dest)
    traducido = str(k.text)
    return traducido
```

3. **Lector de texto:** Esta función convierte un texto en voz, esto lo usaremos para que nuestro programa lea el texto traducido. Podemos observar esta función en la siguiente imagen:

```
import pyttsx3
import time
def leerTexto(texto):
    engine = pyttsx3.init()
    engine.setProperty('rate', 120)
    time.sleep(1)
    engine.say(texto)
    engine.runAndWait()
```

Como podemos ver, cada una de estas funciones son independientes entre sí, por lo que nos permite modular nuestro programa a nuestro beneficio.

A continuación imágenes del uso de nuestro programa:

Texto Origen: hola mundo	Grabar Idioma de destino: Ingles ▼ Traducir	Texto Traducido: Hello World
-----------------------------	--	---------------------------------

Texto Origen: hola mundo	Grabar Idioma de destino: Italiano ▼ Traducir	Texto Traducido: Ciao mondo
-----------------------------	--	--------------------------------





V. CONCLUSIONES:

Con este trabajo de investigación podemos darnos cuenta que el área de la inteligencia artificial se ha desarrollado bastante en los últimos tiempos. Ahora existen muchas herramientas que nos ayudan con el proceso de creación de estas IAs. También, tenemos modelos ya entrenados disponibles para usarse, como por ejemplo googletrans que emplea un modelo de traducción automática neuronal.

Además, es necesario resaltar la importancia de que los modelos se pongan a disposición de la comunidad, ya que permite que esta se nutra y genere interés en temas afines, haciendo que cada vez se desarrollen más y mejores modelos de inteligencia artificial.

Para finalizar podemos concluir que, como vimos en los resultados de práctica, cualquier persona puede hacer uso de librerías con modelos entrenados de inteligencia artificial. Así como googletrans y speechrecognition, existen una infinidad de librerías con modelos de distintos tipos, usos y fines que nos permiten desarrollar software cada vez más complejo y sofisticado.

VI. REFERENCIAS BIBLIOGRÁFICAS:

1. L. Gonzalez. "Regresión Lineal - Teoría -  Aprende IA".  Aprende IA. <https://aprendeia.com/algoritmo-regresion-lineal-simple-machine-learning/> (accedido el 5 de noviembre de 2022).
2. L. Gonzalez. "Regresión Logística - Teoría -  Aprende IA".  Aprende IA. <https://aprendeia.com/algoritmo-regresion-logistica-machine-learning-teoria/> (accedido el 5 de noviembre de 2022).
3. "Árbol de decisión en Machine Learning (Parte 1) - sitiobigdata.com". <https://sitiobigdata.com/2019/12/14/arbol-de-decision-en-machine-learning-parte-1/> (accedido el 5 de noviembre de 2022).
4. "Redes neuronales profundas - Tipos y Características - Código Fuente". Código Fuente. <https://www.codigofuente.org/redes-neuronales-profundas-tipos-caracteristicas/> (accedido el 5 de noviembre de 2022).
5. J. Mansanet y J. J. Pérez. "6 Librerías imprescindibles de Python para Machine Learning". IA SOLVER. <https://iasolver.es/6-librerias-de-python-para-machine-learning/> (accedido el 5 de noviembre de 2022).
6. Verne Academy. "10 Librerías Python para Data Science y Machine Learning". Verne academy. <https://verneacademy.com/blog/articulos-ia/10-librerias-python-data-science-machine-learning/> (accedido el 5 de noviembre de 2022).
7. Equipo de datos.gob.es. "datos.gob.es". 10 librerías populares de análisis de datos y Machine Learning. <https://datos.gob.es/es/blog/10-librerias-populares-de-analisis-de-datos-y-machine-learning> (accedido el 5 de noviembre de 2022).