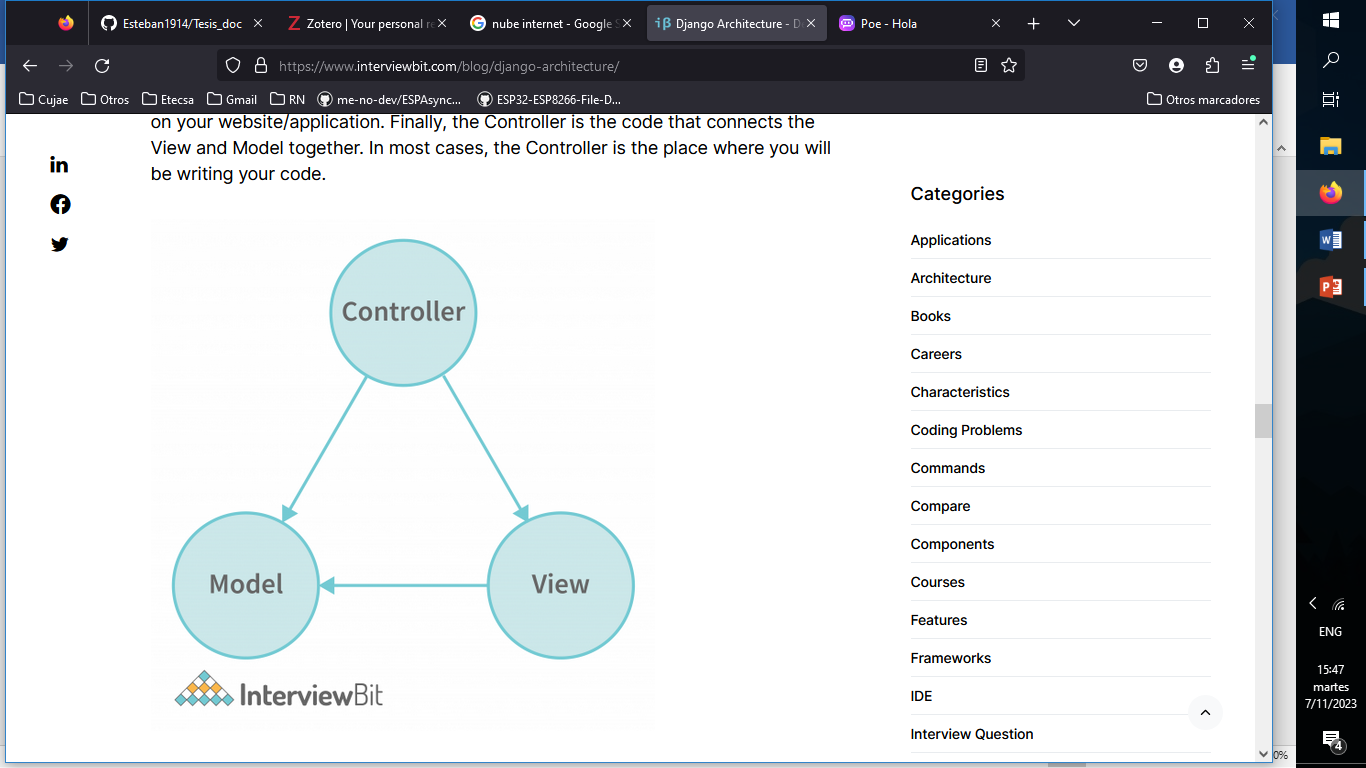
**Django**

****

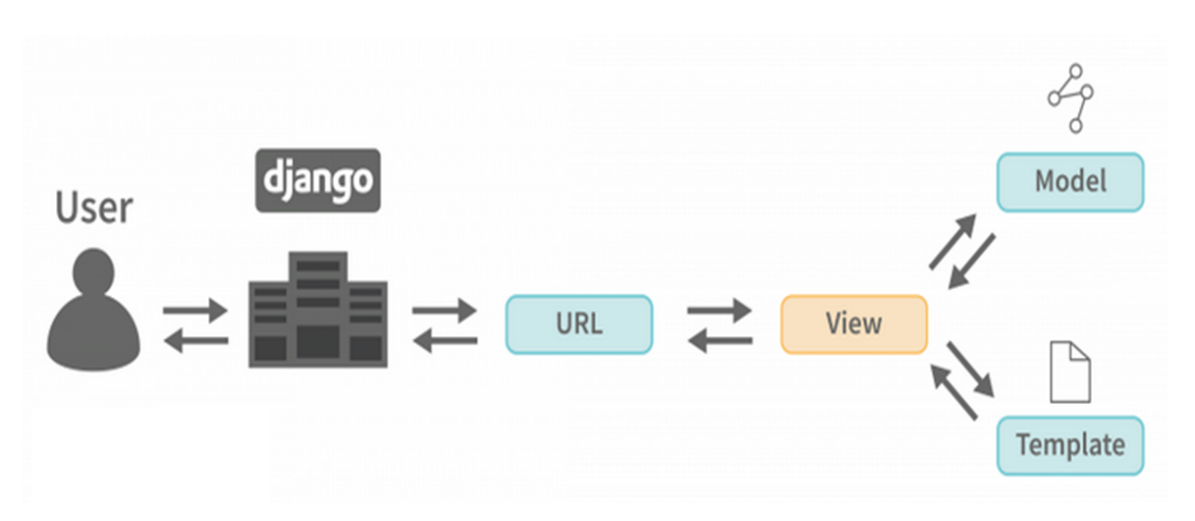
**Introducción a Django**  
Django es un framework web de alto nivel escrito en Python que se utiliza para el desarrollo rápido y eficiente de aplicaciones web. Fue creado por Adrian Holovaty y Simon Willison y lanzado por primera vez en 2005. Django se basa en el principio del desarrollo ágil de software y sigue una filosofía de diseño DRY (Don't Repeat Yourself), lo que significa que promueve la reutilización de código y la eficiencia en el desarrollo.

**Arquitectura MVC**  
Django sigue el patrón de arquitectura Modelo-Vista-Controlador (MVC). En este patrón, el modelo representa la estructura de datos de la aplicación y se encarga de interactuar con la base de datos. La vista se encarga de la lógica de presentación y la generación de la interfaz de usuario. El controlador maneja las solicitudes del usuario y coordina la interacción entre el modelo y la vista. Esta división clara de responsabilidades facilita el desarrollo y el mantenimiento de aplicaciones web en Django.



**ORM (Object-Relational Mapping)**  
Django proporciona un ORM (Mapeo Objeto-Relacional) que permite interactuar con la base de datos utilizando objetos Python en lugar de escribir consultas SQL directamente. Esto simplifica el acceso y la manipulación de datos, ya que las consultas se expresan en términos de modelos y objetos Python en lugar de tablas y filas de la base de datos. El ORM de Django también se encarga de la abstracción de la base de datos, lo que significa que es compatible con varios motores de base de datos, como PostgreSQL, MySQL, SQLite, etc.

**Routing y URL Handling**  
Django utiliza un enfoque basado en URL para manejar las solicitudes web. Un archivo de configuración de URL define las URL y las asocia con las vistas correspondientes. Cuando un usuario realiza una solicitud a una URL específica, Django utiliza el archivo de configuración de URL para determinar qué vista debe manejar la solicitud. Esto permite una organización clara y mantenible de las rutas de la aplicación.

****

Ventajas de usar Django:

1. Alto nivel de productividad: Django es conocido por su alto nivel de productividad debido a su diseño y conjunto de herramientas. Proporciona una arquitectura MVC (Modelo-Vista-Controlador) que facilita la organización del código y la separación de preocupaciones. Además, Django ofrece una amplia gama de funcionalidades incorporadas, como autenticación de usuarios, administración de bases de datos y generación automática de formularios, lo que acelera el proceso de desarrollo.
2. Seguridad integrada: Django tiene una sólida capa de seguridad integrada. Proporciona protección contra amenazas comunes, como ataques de inyección de SQL y cross-site scripting (XSS). Django también ofrece características de autenticación y autorización flexibles, lo que facilita la implementación de sistemas de autenticación seguros y el control de acceso a las funcionalidades de la aplicación.
3. ORM potente: Django incluye un ORM (Object-Relational Mapping) que permite interactuar con la base de datos utilizando objetos Python en lugar de escribir consultas SQL directamente. Esto simplifica el proceso de acceso y manipulación de datos, y proporciona una capa de abstracción que facilita el cambio de base de datos subyacente sin tener que reescribir todo el código.
4. Escalabilidad: Django es capaz de manejar aplicaciones web de gran escala y alto tráfico. Proporciona herramientas y técnicas para optimizar el rendimiento, como la implementación de cachés, la división de carga (load balancing) y el escalado horizontal. Además, Django es compatible con la integración de sistemas de caché externos y servicios de almacenamiento en la nube, lo que facilita la escalabilidad según las necesidades del proyecto.
5. Comunidad activa y madura: Django cuenta con una comunidad activa y madura de desarrolladores y usuarios que brindan soporte, comparten conocimientos y contribuyen con bibliotecas y complementos. Esto significa que hay una amplia cantidad de recursos, documentación y ejemplos disponibles, lo que facilita el aprendizaje y el desarrollo de proyectos utilizando Django.

Ventajas de usar Django sobre Flask:

1. Mayor escalabilidad: Django es un framework más completo y orientado a aplicaciones de gran escala, lo que lo hace más adecuado para proyectos de mayor envergadura. Proporciona una estructura y un conjunto de herramientas más robusto para manejar aplicaciones complejas y de alto tráfico.
2. Administración de bases de datos: Django incluye una interfaz de administración de bases de datos incorporada que permite la gestión y visualización de datos de manera sencilla. Esta característica es especialmente útil durante el desarrollo y la depuración de la aplicación.
3. Capa de seguridad integrada: Django ofrece una capa de seguridad integrada más completa en comparación con Flask. Proporciona características de autenticación y autorización, protección contra amenazas comunes y opciones de configuración para mitigar vulnerabilidades.
4. Mayor cantidad de funcionalidades incorporadas: Django incluye una amplia gama de funcionalidades incorporadas, como autenticación de usuarios, generación automática de formularios, soporte para internacionalización, administración de bases de datos y más. Esto reduce la necesidad de desarrollar estas funcionalidades desde cero y acelera el proceso de desarrollo.
5. Comunidad más grande: Django cuenta con una comunidad de usuarios más grande y activa en comparación con Flask. Esto significa que hay más recursos, documentación, bibliotecas y ejemplos disponibles para facilitar el desarrollo y el aprendizaje.

**Referencias:**

* Django Documentation: <https://docs.djangoproject.com/>
* "Django for Beginners" de William S. Vincent.
* "Two Scoops of Django" de Daniel Roy Greenfeld y Audrey Roy Greenfeld.
* "Django Design Patterns and Best Practices" de Arun Ravindran.

**SQLite3**

SQLite3 es un sistema de gestión de bases de datos relacional de código abierto y ligero que se utiliza ampliamente en aplicaciones web y móviles. Django, como framework web de Python, ofrece una integración sólida con SQLite3, lo que permite a los desarrolladores utilizarlo como opción de base de datos en sus proyectos. Algunos aspectos clave a considerar sobre la integración de SQLite3 en Django son:

1. Configuración de la base de datos: Django proporciona una configuración sencilla para utilizar SQLite3 como base de datos en una aplicación. A través del archivo de configuración de Django (settings.py), los desarrolladores pueden especificar la ruta y el nombre del archivo de la base de datos SQLite3.
2. ORM y migraciones: Django utiliza un ORM (Object-Relational Mapping) que facilita la interacción con la base de datos. El ORM de Django proporciona una capa de abstracción que permite a los desarrolladores trabajar con objetos Python en lugar de tener que escribir consultas SQL directamente. Además, Django también ofrece migraciones, que permiten la creación y modificación de la estructura de la base de datos de manera controlada y consistente.
3. Características y limitaciones: SQLite3 es una base de datos ligera y fácil de usar, pero también tiene algunas limitaciones a considerar. Por ejemplo, no es adecuada para aplicaciones que requieren un alto volumen de transacciones concurrentes o para escenarios de escalabilidad extrema. Sin embargo, para aplicaciones de tamaño mediano o pequeño, SQLite3 es una opción sólida debido a su simplicidad y bajo consumo de recursos.

**Ventajas y casos de uso de SQLite3 en Django**  
La integración de SQLite3 en Django ofrece varias ventajas y es adecuada para diversos casos de uso. Algunos puntos destacados para mencionar en tu tesis son:

1. Portabilidad: SQLite3 es un sistema de base de datos de archivo único, lo que significa que toda la base de datos se almacena en un archivo local. Esto facilita la portabilidad y el despliegue de aplicaciones, ya que solo se necesita transferir el archivo de la base de datos.
2. Desarrollo rápido y prototipado: SQLite3 es rápido de configurar y no requiere una configuración de servidor separada. Esto lo convierte en una opción ideal para el desarrollo rápido y el prototipado de aplicaciones web, ya que no se requiere una infraestructura adicional.
3. Aplicaciones de tamaño mediano o pequeño: SQLite3 es adecuada para aplicaciones web de tamaño mediano o pequeño, donde los volúmenes de datos y las demandas de transacciones concurrentes no son extremadamente altos. Es especialmente útil en aplicaciones de una sola instancia o de uso personal.

**ESP32**

ESP32 es un microcontrolador de bajo costo y baja potencia que se utiliza en el desarrollo de aplicaciones IoT (Internet de las cosas). Fue desarrollado por Espressif Systems y es considerado uno de los microcontroladores más populares en el ámbito de la IoT debido a su versatilidad y potencia. El ESP32 combina un procesador de doble núcleo, conectividad Wi-Fi y Bluetooth, capacidades de bajo consumo de energía y una amplia variedad de interfaces y periféricos, lo que lo convierte en una opción atractiva para proyectos IoT de diferentes escalas y complejidades.

**Características y funcionalidades**  
El ESP32 cuenta con una amplia gama de características y funcionalidades que lo hacen adecuado para una variedad de aplicaciones IoT. Algunas de estas características incluyen:

1. Procesador de doble núcleo: El ESP32 está equipado con un procesador de doble núcleo Xtensa LX6, lo que permite ejecutar múltiples tareas de manera eficiente y simultánea.
2. Conectividad Wi-Fi y Bluetooth: El ESP32 tiene soporte integrado para Wi-Fi 802.11 b/g/n, lo que permite la conexión a redes inalámbricas. Además, también ofrece soporte para Bluetooth clásico y Bluetooth de baja energía (BLE), lo que facilita la comunicación inalámbrica con otros dispositivos.
3. Interfaces y periféricos: El ESP32 cuenta con una variedad de interfaces y periféricos, como puertos UART, SPI, I2C, GPIO y ADC, que permiten la conexión y comunicación con otros dispositivos y sensores.
4. Bajo consumo de energía: El ESP32 está diseñado para minimizar el consumo de energía, lo que lo hace adecuado para aplicaciones que requieren una larga duración de la batería.

**Aplicaciones de ESP32**  
El ESP32 se utiliza en una amplia gama de aplicaciones IoT. Algunos ejemplos de aplicaciones incluyen:

1. Domótica: El ESP32 se utiliza en proyectos de automatización del hogar, permitiendo el control y monitoreo remoto de luces, sistemas de seguridad, electrodomésticos, etc.
2. Agricultura inteligente: El ESP32 se utiliza en sistemas de monitoreo y control de cultivos, permitiendo la medición de parámetros como la humedad del suelo, la temperatura y la luminosidad, y facilitando la toma de decisiones basada en datos en tiempo real.
3. Dispositivos portátiles: El ESP32 se utiliza en el desarrollo de dispositivos portátiles como relojes inteligentes, brazaletes de actividad física y monitores de salud, permitiendo la conectividad inalámbrica y el procesamiento de datos.
4. Automatización industrial: El ESP32 se utiliza en proyectos de automatización industrial, permitiendo el control y supervisión remotos de procesos y máquinas.

**Referncias:**

* Documentación oficial de Espressif Systems: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/>
* "Getting Started with ESP32" de Simon Monk.

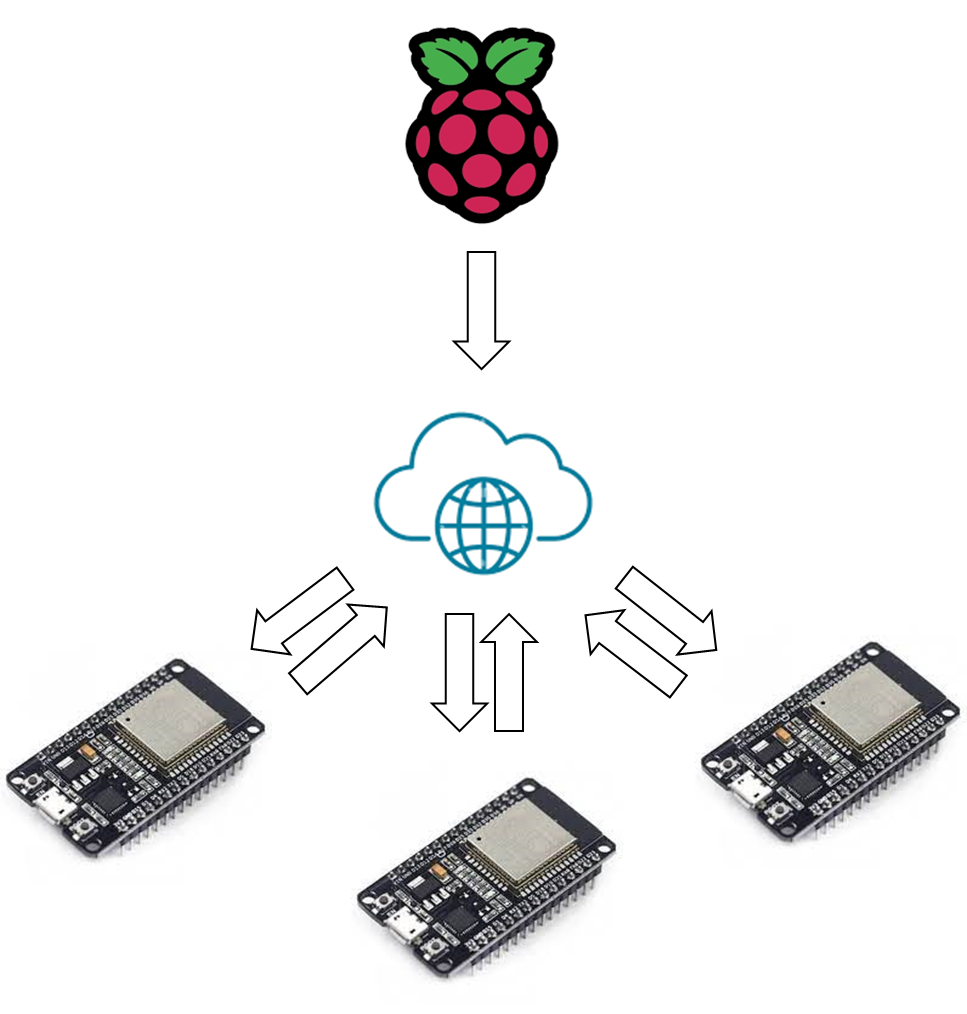
“ESP32 Development Workshop" de Neil Kolban.

**Funcionamiento:**

Para el desarrollo de este proyecto se analizarán dos dispositivos:

* Dispositivo de laboratorio (Raspberry)
* Dispositivo de campo (ESP32)

El dispositivo de laboratorio en general, tendrá la tarea de tomar las muestras necesarias para entrenar un modelo optimo y alojarlo en un servidor web público. Los dispositivos de campo podrán seleccionar el modelo a descargar para la predicción de muestras.



**Dispositivo de laboratorio**

Este dispositivo se encargará de:

* Interfaz visual
* Comunicación con el sensor
* Almacenar muestras (Base de datos)
* Entrenamiento del modelo
* Pruebas de predicción con modelo entrenado
* Subir modelo hacia servidores para ser usado por los distintivos de campo

Todo este procesamiento requiere un alto nivel computacional, por lo tanto, una Raspberry Pi es adecuada

**Interfaz Visual**

La interfaz visual que se utilizara el framework Django, el cual está basado en Python y permite crear una interfaz agradable y segura para la toma de datos. La programación de este framwork sería orientada a clase basada en vistas, se implantaron tres niveles de seguridad:

* Desarrollador
* Administrador
* Invitado

Esta interfaz visual será desplegada sobre apache2, para poder servirlo desde cualquier red externa.

**Comunicación con el sensor**

Para la comunicación con el sensor As7265x, es necesario la comunicación por el protocolo I2C, protocolo que la Raspberry pi soporta, la libraría smbus de Python permitirá la lectura de datos.

**Almacenamiento**

Django trae implementado por defecto sqlite3, base de datos que permite alancear de manera organizada las muestras

**Clasificación de datos**

Para la clasificación de datos existen múltiples métodos :

1. LDA (Análisis de Discriminante Lineal): El Análisis de Discriminante Lineal busca encontrar una combinación lineal de características que maximice la separación entre diferentes clases de datos. Proyecta los datos en un espacio de menor dimensión mientras maximiza la varianza entre las clases y minimiza la varianza dentro de cada clase.  
   Referencia: Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. Annals of Eugenics, 7(2), 179-188.
2. KNN (K-vecinos más cercanos): El algoritmo K-vecinos más cercanos clasifica un nuevo punto de datos según la clase mayoritaria de sus k vecinos más cercanos en el espacio de características. La distancia se calcula utilizando una medida de distancia, como la distancia euclidiana.  
   Referencia: Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), 21-27.
3. Redes Neuronales: Las redes neuronales son modelos computacionales inspirados en el cerebro humano. Consisten en capas de nodos interconectados (neuronas) que procesan y transmiten información. Las redes neuronales pueden aprender a clasificar datos ajustando los pesos de las conexiones entre las neuronas durante un proceso de entrenamiento.  
   Referencia: Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536
4. Árboles de Decisión: Los árboles de decisión son estructuras de tipo árbol donde cada nodo interno representa una característica o atributo, y las ramas del árbol representan las posibles opciones o valores para esa característica. Los árboles de decisión se construyen dividiendo los datos en función de las características más relevantes y se utilizan para clasificar nuevos ejemplos siguiendo el camino desde la raíz hasta las hojas del árbol.  
   Referencia: Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1), 81-106.
5. SVM (Support Vector Machines): Las Máquinas de Vectores de Soporte son modelos de aprendizaje automático que buscan encontrar un hiperplano óptimo que separe las diferentes clases de datos en un espacio de características. El objetivo es maximizar el margen entre las instancias de diferentes clases más cercanas al hiperplano de decisión. SVM también puede utilizar kernels para manejar datos no linealmente separables.  
   Referencia: Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297.
6. Naive Bayes (Clasificador Bayesiano Ingenuo): El clasificador bayesiano ingenuo se basa en el teorema de Bayes y asume independencia condicional entre las características. Calcula la probabilidad de que un ejemplo pertenezca a una clase determinada utilizando la probabilidad de la clase y la probabilidad de cada característica dado un valor de clase. A pesar de su suposición simplificada, este método puede ser efectivo en muchos casos.  
   Referencia: Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning, 29(2-3), 103-130.
7. Random Forest (Bosques Aleatorios): Los bosques aleatorios son un conjunto de árboles de decisión combinados. Cada árbol se entrena con una muestra aleatoria del conjunto de datos y produce una clasificación. La clasificación final se obtiene por votación o promediando las predicciones de los árboles individuales. Los bosques aleatorios ayudan a mitigar el sobreajuste y suelen ser robustos y precisos en la clasificación.  
   Referencia: Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32

Para la selección del método a utilizar, es importante destacar que el esp32 es un microcontrolador de bajo consumo de energía y recursos limitados en comparación con sistemas más potentes como una Raspberry o una computadora convencional. Esto implica que las técnicas de análisis de datos que se pueden implementar en el esp32 deben ser adaptadas a sus capacidades y restricciones.

Por lo tanto métodos como el usado en [Tesis]; LDA con PCA, en el caso del esp32, su implementación puede ser desafiante debido a las limitaciones de memoria y capacidad de cálculo del microcontrolador. El PCA, en particular, puede requerir una gran cantidad de memoria y poder de procesamiento para calcular las componentes principales.

Por otro lado, las redes neuronales son una opción para implementar en el esp32, ya que se pueden adaptar a sus restricciones y aprovechar al máximo sus capacidades. Las redes neuronales, especialmente las redes neuronales convoluciones (CNN) y las redes neuronales recurrentes (RNN), han demostrado ser efectivas en tareas de clasificación y reconocimiento de patrones en diversos dominios.

*Referncia:*

*Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.*

Las redes neuronales también ofrecen oportunidades de optimización específicas para dispositivos de recursos limitados. Técnicas como la cuantificación de pesos y la compresión del modelo permiten reducir aún más el tamaño del modelo y los requisitos de memoria, mientras que las técnicas de aceleración de hardware como la inferencia en hardware especializado o la optimización de código pueden mejorar el rendimiento y la eficiencia energética en el ESP32.

Referencias:

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or-1. arXiv preprint arXiv:1602.02830.

Por lo tanto, el método de redes neuronales se puede implementar correctamente a una placa esp32.

**Redes Neuronales:**

Las redes neuronales son un modelo computacional inspirado en el funcionamiento del cerebro humano. Están diseñadas para aprender y reconocer patrones complejos en conjuntos de datos mediante el procesamiento de información a través de múltiples capas de neuronas interconectadas. Estas pueden resolver problemas complejos de clasificación, regresión, reconocimiento de imágenes, procesamiento de lenguaje natural

La neurona y su funcionamiento:  
La neurona es la unidad fundamental de procesamiento en una red neuronal. Se compone de tres partes principales: las dendritas, el cuerpo celular y el axón. Las dendritas reciben señales de entrada de otras neuronas o del entorno, y el cuerpo celular procesa estas señales. Luego, el axón transmite la señal procesada a otras neuronas.

El perceptrón y sus características:  
El perceptrón es un tipo básico de neurona artificial utilizado en redes neuronales. Tiene conexiones ponderadas que multiplican las señales de entrada y las suman. Luego, se aplica una función de activación para determinar si la neurona se activa o no. El perceptrón puede aprender a ajustar los pesos de las conexiones para lograr una salida deseada mediante un proceso llamado aprendizaje supervisado.

Referncias

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press

Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford University Press.

**Implemtacion de Redes Neuronales**

Para la implementación de redes neuronales, se utilizara la librería de Tensorflow y Keras, las cuales traen toda la implementación para crear redes neuronales, para entrenarlas y utilizarlas para clasificar.

**Arquitectura Red Neuronal**

Para la arquitectura interna no existen métodos específicos para se selección , pero Keras trae un nuevo método de obtención de hiperparámetros óptimos con su librería keras-tuner para Python .

Se hicieron pruebas con distintas redes neuronales como ….

Resultados…..

**Servidor**

Para poder utilizar el modelo por en instrumento de campo será necesario subir a un servidor la información del modelo de keras, , para ello se utilizara la propia plataforma de git para actualizar los modelos en la web , los modelos seran genrados por la propia librería de tensorflow , pero en su versión lite para dispositivos de menores prestaciones , para que puedan ser utilizadas estas redes entrenadas; por otros lenguajes y plataformas.

**Dispositivo de Campo**

El dispositivo de campo se encargara de tomar los modelos hechos por el dispositivo de campo , y lograr encontrar el estado de la leche , tiene que ser un dispositivo que e pueda comunicar con el sensor, que pueda contarse al servidor online, descargar los modelos y almacenarlos , además de ser un dispositivo configurable , para ello se utilizara el esp32.

**Lenguaje**

Esp32 sera programado en C++ sobre la placa de Arduino , programado con el ide de VSC en PlattaformIO.

**Interfaz Visual**

El esp32 implantará una interfaz visual para permitir actualizar los modelos de