*Tarea de Investigación*

**Parte 1: Estimación del Mercado Laboral Tech en Ecuador (Expandido)**

A continuación, se presenta una tabla detallada con estimaciones de salarios, responsabilidades y tecnologías clave para roles de tecnología en Ecuador.

**Nota:**

Los salarios son estimaciones para roles de nivel Intermedio

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Profesión** | **Volumen de Ofertas** | **Salario Estimado (USD/hora)** | **Salario Estimado (USD/Anual)** | **Responsabilidades Clave** | **Frameworks / Tecnologías Comunes** |
| **Frontend Engineer** | Alto | $5.50 - $ 10.50 | $11,440 - $21,840 | Implementar la interfaz de usuario (UI) y la experiencia de usuario (UX).  Consumir APIs.  Asegurar la interactividad y el responsiveness del sitio. | React.js, Angular |
| **Backend Engineer** | Alto | $6.00 - $ 11.50 | $12,480 - $ 23,920 | Desarrollar la lógica del servidor, administrar bases de datos, crear APIs (REST/GraphQL) y manejar la autenticación y seguridad. | Node.js (Express), Python (Django, Flask), Java (Spring Boot), .NET, SQL  (PostgreSQL, MySQL). |
| **Data Engineer** | Medio | $6.00 - $ 13.00 | $13,520 - $ 27,040 | Diseñar, construir y mantener pipelines de datos (ETLs/ELTs). Gestionar la ingesta, almacenamiento y transformación de grandes volúmenes de datos. | Apache Spark, Airflow, SQL/NoSQL, Kafka, Databricks, MLflow, Kubeflow |
| **ML Engineer** | Bajo | $8.00 - $ 16.00 | $16,640 - $ 33,280 | Poner modelos de Machine Learning en producción (MLOps). Optimizar modelos para inferencia, crear APIs para su consumo y monitorear su rendimiento. | Scikit-learn, TensorFlow, Keras, PyTorch, MLflow, Kubeflow, Docker, Kubernetes |
| **AI Engineer** | Muy bajo | $8.50 - $ 17.50 | $17,680 - $ 36,400 | Investigar, entrenar e implementar modelos complejos (Deep Learning, LLMs, Computer Vision, NLP). Es un rol más enfocado en I+D. | TensorFlow, Keras, PyTorch, Hugging Face Transformers, LangChain, Scikit-learn |

**Observaciones**

1. **Roles de Datos (Data/ML/AI):**Los frameworks como Scikit-learn, TensorFlow, Keras, MLflow y Kubeflow son el estándar de la industria.
   * **Scikit-learn:** usado por casi todos para ML clásico.
   * **TensorFlow/Keras:** dominantes en Deep Learning (junto con PyTorch).
   * **MLflow/Kubeflow:** herramientas de MLOps para gestionar el ciclo de vida de los modelos.
2. **Demanda en Ecuador:**La demanda local de AI/ML Engineers puros es baja y se concentra en *startups* tecnológicas o centros de innovación.  
   Sin embargo, la demanda de *Data Engineers* está creciendo rápidamente.
3. **El Factor Remoto:**Los salarios se duplican o triplican si se trabaja de forma remota para empresas de EE.UU. o Europa.

**Parte 2: Importancia de Roles, Data Engineer y Feature Engineer**

Es primordial conocer los conceptos básicos antes de aplicarlos y resolver el Query.

**1. Ingeniería de Características (Feature Engineering / FE)**

Es el proceso de usar conocimiento del dominio para transformar datos crudos en variables (features) que representen mejor el problema.

**Objetivo:** facilitar el trabajo del modelo de Machine Learning.  
La calidad de las características tiene un mayor impacto que el algoritmo usado.

**Etapas principales:**

1. **Creación de características:**Crear nuevas variables a partir de las existentes. **Ejemplo:** de fecha\_construccion → crear antiguedad\_casa = Año actual - fecha\_construccion.
2. **Transformación y limpieza:**Los modelos necesitan datos limpios y numéricos. **Ejemplo:** *Encoding* = convertir texto a números.
3. **Selección de características:**Eliminar variables irrelevantes o redundantes.

**Parte 3: Rol del Data Engineer**

El Data Engineer es el habilitador clave.  
No diseña el modelo, pero construye y mantiene el *pipeline* de datos que:

* Genera millones de *embeddings* (vectores numéricos de significado).
* Los almacena en *Vector Databases* optimizadas.
* Permite al AI Engineer hacer consultas de similaridad en milisegundos.

**Parte 4: Conceptos Clave**

**Similaridad Coseno:**Mide el ángulo entre dos vectores.

* Si apuntan en la misma dirección → similaridad = 1 (significan lo mismo).
* Si son perpendiculares → similaridad = 0 (sin relación).
* Si apuntan en direcciones opuestas → similaridad = -1 (contrarios).

**Parte 5: Ejercicio a Resolver**

**Query:**

**Comparar el significado de: “dame el total de la factura IQ5430” con las siguientes funciones:**

* Función 1: find\_invoice
* Función 2: find\_user
* Función 3: create\_invoice

**Análisis**

Este es un problema clásico de *function calling* en sistemas de IA.

Se requiere una combinación de dos técnicas:

1. Embeddings (Vectores de Significado)
2. Similaridad Coseno (Cosine Similarity)

La similaridad sola no funciona con texto crudo. Primero hay que transformar el texto a vectores numéricos que representen su significado.

**Pasos para Resolver**

**Paso 1: Vectorización (Crear Embeddings)**

Debemos vectorizar tanto el *query* como las funciones.

**Definición:**Un *embedding* es un vector numérico que representa el significado semántico del texto.

**1.1 Vectorizar el Query**

* Texto: “dame el total de la factura IQ5430”
* Se pasa por un modelo de embeddings (por ejemplo, *sentence-transformers* en Python).
* Resultado:  
  Vector\_Query = [0.12, -0.45, 0.81, ...]

**1.2 Vectorizar las Funciones**

No solo se vectoriza el nombre, también su descripción funcional:

* Función 1: “find\_invoice: Busca y devuelve los detalles de una factura usando su ID.”
* Función 2: “find\_user: Encuentra los datos de un usuario por su nombre o ID.”
* Función 3: “create\_invoice: Genera una nueva factura para un cliente.”

**Se obtiene:**

* Vector\_Func1 = [0.10, -0.40, 0.79, ...] → muy similar al query
* Vector\_Func2 = [-0.50, 0.11, -0.23, ...] → muy diferente
* Vector\_Func3 = [0.05, -0.30, 0.65, ...] → algo similar

**Paso 2: Medición (Calcular Similaridad Coseno)**

**Ahora se comparan los vectores:**

* score\_1 = cosine\_similarity(Vector\_Query, Vector\_Func1) → 0.92
* score\_2 = cosine\_similarity(Vector\_Query, Vector\_Func2) → 0.15
* score\_3 = cosine\_similarity(Vector\_Query, Vector\_Func3) → 0.61

**Paso 3: Ranking (Ordenar Resultados)**

**Ordenar los resultados de mayor a menor:**

1. find\_invoice (Score: 0.92)
2. create\_invoice (Score: 0.61)
3. find\_user (Score: 0.15)

**Conclusión**

Con seguridad se determina que la función que debe llamarse es: “ **find\_invoice”**