**TP Inicial – Plataforma de Gestión de Recursos Humanos con ERP y CRM con Inteligencia Artifical**

**Introducción**

Los sistemas ERP (*Enterprise Resource Planning*) y CRM (*Customer Relationship Management*) son herramientas esenciales para mejorar la gestión de recursos humanos (RRHH) en las empresas. Al combinarlos con inteligencia artificial (IA), pueden automatizar procesos como el reclutamiento, la evaluación de empleados y la administración del personal. En este TP Inicial, van a investigar cómo funcionan estas tecnologías y a desarrollar un prototipo práctico usando *machine learning*. Este trabajo les dará las bases para el TP Principal, donde construirán una plataforma más completa.

**Objetivos del TP**

* Entender cómo los sistemas ERP y CRM se aplican a la gestión de RRHH.
* Analizar cómo la IA mejora la eficiencia en procesos de RRHH.
* Identificar ventajas y desafíos de estas tecnologías.
* Aprender sobre herramientas y técnicas de *machine learning* que usarán en la práctica.
* Desarrollar un prototipo funcional con *machine learning* aplicado a un aspecto de RRHH.

**Enunciado del TP**

Van a trabajar en equipos y realizar las siguientes actividades:

1. **Investigación Teórica (en dos partes):**
   * **Parte 1: ERP, CRM e IA en RRHH**
     + Investiguen ejemplos reales de sistemas ERP y CRM usados en RRHH (por ejemplo, SAP SuccessFactors, Oracle HCM, Salesforce). Describan sus características principales.
     + Analicen cómo la IA mejora la eficiencia en procesos como reclutamiento, evaluación de candidatos y gestión del desempeño. Incluyan ejemplos concretos.
     + Identifiquen ventajas (como automatización o precisión) y desafíos (como costos o seguridad) de implementar estas soluciones.
   * **Parte 2: Herramientas y Técnicas de *Machine Learning***
     + Investiguen qué es Scikit-learn y cómo se usa en Python para desarrollar modelos de *machine learning*.
     + Exploren las técnicas de *machine learning* que van a usar en la parte práctica:
       - **Regresión logística y árboles de decisión** (clasificación).
       - **Regresión lineal** (predicción numérica).
       - ***K-means*** (agrupamiento).
       - ***Isolation Forest*** (detección de anomalías).
     + Expliquen qué hace cada técnica, cómo funciona y en qué casos se aplica. Esto les evitará dudas más adelante, por lo que deberán ser detallistas.
2. **Parte Práctica: Desarrollo de un prototipo con *Machine Learning***

En esta sección del TP Inicial, van a elegir una de las seis opciones que les presentamos abajo para desarrollar un prototipo sencillo usando machine learning. Cada opción está vinculada a uno de los módulos del TP Principal y les servirá como una introducción práctica a los conceptos que van a explorar más adelante. El objetivo es que se familiaricen con herramientas como Python y Scikit-learn, y que empiecen a aplicar machine learning en contextos reales de gestión de recursos humanos.

#### ****Instrucciones Generales****

* **Elección de la opción:** Elijan una de las seis opciones que están listadas más abajo.
* **Requisitos técnicos:** Todas las opciones tienen que implementarse en Python usando la biblioteca Scikit-learn.
* **Entrega:**
  + Código del modelo entrenado.
  + Conjunto de datos ficticio que crearon para entrenar el modelo.
  + Una breve demostración del resultado (por ejemplo, predicciones, clasificaciones o recomendaciones).

**Opciones para la Parte Práctica**

#### ****Opción 1: Clasificación de Candidatos Basada en Habilidades****

* **Descripción:**  
  Creen un modelo de clasificación (como regresión logística o árbol de decisión) para predecir si un candidato es apto para un puesto basado en datos como años de experiencia, nivel educativo y habilidades. Primero, generen un conjunto de datos ficticio con al menos 50 candidatos, incluyendo las variables mencionadas y una etiqueta ("Apto" o "No Apto"). Luego, entrenen el modelo para que clasifique automáticamente nuevos candidatos.
* **Resultado esperado:** Un modelo que clasifique candidatos y muestre su precisión (por ejemplo, "Candidato con 5 años de experiencia y habilidad en Python → Apto").

#### ****Opción 2: Predicción de Puntaje de Desempeño****

* **Descripción:**  
  Desarrollen un modelo de regresión lineal para predecir el puntaje de desempeño de un empleado basado en variables como horas de capacitación, antigüedad y calificación previa. Creen un conjunto de datos ficticio con al menos 30 empleados y entrenen el modelo para predecir el puntaje.
* **Resultado esperado:** Un modelo que prediga puntajes (por ejemplo, "Empleado con 20 horas de capacitación y 3 años de antigüedad → Puntaje 78") y muestre el error cuadrático medio (MSE).

#### ****Opción 3: Recomendación de Cursos Basada en Habilidades****

* **Descripción:**  
  Implementen un sistema de recomendación usando k-means para agrupar empleados según sus habilidades y sugerir cursos personalizados. Generen un conjunto de datos con al menos 40 empleados, incluyendo habilidades y nivel de experiencia. El modelo debe agruparlos en clústeres y recomendar cursos según el clúster al que pertenezcan.
* **Resultado esperado:** Un modelo que agrupe empleados y sugiera cursos (por ejemplo, "Empleado Junior con Python → Clúster ‘Desarrolladores’ → Curso: Python Avanzado").

#### ****Opción 4: Detección de Anomalías en Asistencia****

* **Descripción:**  
  Usen un modelo de detección de anomalías (como Isolation Forest) para identificar patrones inusuales en registros de asistencia, como ausencias repetidas. Simulen un conjunto de datos con al menos 100 registros de asistencia y entrenen el modelo para marcar anomalías.
* **Resultado esperado:** Un modelo que detecte anomalías (por ejemplo, "Empleado con 3 ausencias seguidas → Anomalía").

#### ****Opción 5: Predicción de Riesgo de Salud Ocupacional****

* **Descripción:**  
  Creen un modelo de clasificación para predecir si un empleado tiene alto riesgo de problemas de salud basado en datos como ausencias por enfermedad y horas trabajadas. Generen un conjunto de datos con al menos 50 empleados y entrenen el modelo para clasificarlos en "Alto Riesgo" o "Bajo Riesgo".
* **Resultado esperado:** Un modelo que clasifique empleados (por ejemplo, "Empleado con 60 horas semanales y 6 ausencias → Alto Riesgo").

#### ****Opción 6: Predicción de Preferencias para Notificaciones de Eventos Sociales****

* **Descripción:**  
  Desarrollen un modelo de clasificación para predecir cómo prefieren los empleados recibir notificaciones sobre eventos como cumpleaños o aniversarios laborales. Generen un conjunto de datos con al menos 50 empleados, incluyendo edad, antigüedad y preferencia ("Correo", "Slack", "Notificación Push"). Entrenen el modelo para sugerir el canal óptimo.
* **Resultado esperado:** Un modelo que sugiera el canal adecuado (por ejemplo, "Empleado de 25 años → Slack").

**Fechas de Entrega**

Las entregas son acumulativas: cada fecha incluye lo anterior corregido y mejorado si es necesario. Suban todo al repositorio compartido que van a crear.

* **19/03**
  + Entreguen la **Parte 1 de la investigación** (ERP, CRM e IA en RRHH) en un documento (máximo 3 páginas).
* **26/03**
  + Entreguen la investigación completa:
    - Parte 1 corregida y mejorada si había errores.
    - Parte 2 de la investigación (herramientas y técnicas de *machine learning*).
  + Incluyan evidencia de que crearon el repositorio compartido (por ejemplo, enlace a GitHub y captura de pantalla).
* **04/04**
  + Creación del dataset ficticio, implementación preliminar del modelo y resultados básicos.
  + Incluye un informe breve de progreso (máximo 1 página).
* **11/04**
  + Prototipo completo y funcional, con:
    - Código fuente completo y comentado.
    - Dataset ficticio final.
    - Documentación técnica detallada (descripción del modelo, instrucciones para ejecutarlo, cómo entrenarlo y hacer predicciones).
    - Demo funcional (video corto o capturas de pantalla).
    - Todo subido al repositorio compartido.

**RECORDATORIO**: La entrega en el Moodle se hace el día anterior hasta las 18:00hs

Si no limitás la profundidad del árbol (max\_depth), **crecerá indefinidamente hasta que cada hoja contenga solo una clase**. Esto significa que:

* El árbol memoriza cada caso del conjunto de entrenamiento.
* Pierde la capacidad de generalizar a datos nuevos.
* En la práctica, se convierte en una tabla de búsqueda en lugar de un modelo predictivo.

**🔍 Cómo comprobar si el árbol es demasiado profundo**

Podés ver la **profundidad real** que alcanzó el árbol con:

python

CopiarEditar

print(f"Profundidad del árbol: {clf.get\_depth()}")

print(f"Número de nodos en el árbol: {clf.tree\_.node\_count}")

Si la profundidad es muy alta (por ejemplo, más de 15 o 20 niveles), **definitivamente está sobreajustado**.

**🛠 Cómo controlar la profundidad**

Podés limitar el crecimiento del árbol con max\_depth para que no memorice los datos:

python

CopiarEditar

clf = tree.DecisionTreeClassifier(max\_depth=5, random\_state=42)

También podés probar **otros parámetros** para evitar sobreajuste:

* min\_samples\_split=10 → Mínimo de muestras requeridas para dividir un nodo.
* min\_samples\_leaf=5 → Mínimo de muestras en una hoja final.

python

CopiarEditar

clf = tree.DecisionTreeClassifier(max\_depth=5, min\_samples\_split=10, min\_samples\_leaf=5, random\_state=42)

**🚀 Próximo paso**

Probá imprimir la profundidad del árbol actual y vemos qué tan grande es. También podemos ajustar max\_depth progresivamente hasta encontrar un equilibrio. ¿Te parece? 😊