IES DOMINGO PÉREZ MINIK

Asignatura: Inteligencia Artificial

Trabajo: UD01A02

Alumno: Martínez Caballero Darel

Fecha: Septiembre 2025

Introducción

En este documento se presentan las respuestas a las preguntas de la unidad didáctica UD01A02 relacionadas con los diferentes enfoques de la inteligencia artificial, aprendizaje automático y casos prácticos de aplicación. Se abordan conceptos de IA simbólica, Machine Learning clásico y Deep Learning, así como reflexiones sobre la importancia de los datos y los riesgos asociados a su sesgo.

Desarrollo

1. Decide si son IA simbólica, ML clásico o Deep Learning:

- - Un sistema experto médico de los 80. → IA simbólica.
- - Reconocimiento facial en el móvil. → Deep Learning.
- - Un chatbot basado en reglas (FAQ preprogramadas). → IA simbólica.
- - Netflix recomendando series. → Machine Learning clásico (filtrado colaborativo).
- - AlphaGo. → Deep Learning + Aprendizaje por refuerzo.

2. Agrupa por supervisada, no supervisada o por refuerzo:

- - Caso 1: Clasificar correos como "spam/no spam". → Supervisada.
- - Caso 2: Agrupar clientes por sus hábitos de compra. → No supervisada.
- - Caso 3: Un dron que aprende a volar evitando obstáculos. → Por refuerzo.
- - Caso 4: Predecir precio de casas según características. → Supervisada.

3. ¿Qué enfoque crees que es más usado en aplicaciones web hoy en día y por qué?

Hoy en día el más usado es el Machine Learning clásico y el Deep Learning en algunos casos, principalmente porque permiten personalización (recomendaciones), predicciones en tiempo real y clasificación de datos de forma más eficiente que la IA simbólica. El Deep Learning se ha popularizado con procesamiento de imágenes y NLP.

4. Análisis del dataset Titanic

Archivos y problemas detectados:

gender_submission.csv

- Contenido: PassengerId y Survived.
- Estado: Completo y bien estructurado.
- Problemas: Ninguno relevante; sin valores nulos, duplicados ni inconsistencias.

test.csv

- Columnas clave: PassengerId, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked.
- Problemas detectados:
  - Valores faltantes: Age, Cabin; posibles en Fare o Embarked.
  - Tipos de datos inconsistentes: Age con decimales; Name con comillas y paréntesis.
  - Datos truncados: vista parcial indica más filas no visibles.
  - Outliers: Fare con valores extremos (ej. 512.33).
  - Variables categóricas: Sex, Embarked, Cabin y Ticket requieren encoding.
  - Posibles duplicados en nombres o tickets.

train.csv

- Columnas clave: mismas que test.csv + Survived.
- Problemas detectados:
  • Valores faltantes: Age, Cabin, algunos en Embarked.
  • Tipos de datos complejos: mismos que test.csv.
  • Outliers: Age muy alto o muy bajo; Fare extremo.
  • Desbalanceo de clases: más 0 que 1 en Survived.
  • Columnas irrelevantes: PassengerId, Name, Ticket, Cabin requieren feature engineering.

Problemas generales del dataset

- Incompletitud: muchos nulos en Cabin (~77%), Age (~20%) y algunos en Embarked.
- Ruido: nombres con formatos variados; tickets alfanuméricos inconsistentes.
- Redundancia: SibSp y Parch podrían combinarse en FamilySize.
- Escalabilidad: Fare necesita normalización para modelos de ML.

Riesgos de entrenar con datos incompletos:

- Sesgo en el modelo (Bias):
  • Los nulos no son aleatorios. Ejemplo: Cabin falta más en clases bajas (Pclass=3), sesgando predicciones hacia clases altas.
  • El modelo podría aprender relaciones incorrectas y predecir peor para grupos con más nulos.
- Pérdida de precisión y generalización:
  • Modelos clásicos no manejan bien nulos.
  • Overfitting: se ajusta demasiado a datos completos.
  • Underfitting: eliminar filas con nulos reduce el tamaño del dataset (~891 → ~700 filas).
  • Métricas como accuracy, precision y recall se reducen.
- Errores en predicción:
  • Aparición de NaN en test.csv al no manejar nulos → fallos o resultados inválidos.
  • Riesgo de crashes en producción.
- Problemas éticos y prácticos:
  • Refuerzo de desigualdades (ej. sesgo por clase o género en Titanic).
  • Mayor coste al tener que reentrenar varias veces.
- Solución recomendada:
  • Evitar entrenar directamente con datos incompletos.
  • Realizar imputación (mediana/moda) o eliminar columnas con muchos nulos (ej. Cabin).
  • Resultado esperado: modelo más robusto y confiable.

Limpieza Básica con Python:

- Eliminación de columnas:
  • Cabin → eliminada (77% nulos).
- Imputación de nulos:
  • Age → mediana.
  • Embarked → moda (ej. 'S').
  • Fare → mediana en test.csv.
- Duplicados:
  • Eliminación de filas duplicadas (poco probable en este dataset).
- Codificación:
  • Sex → binario (male=0, female=1).
  • Embarked → one-hot encoding (Embarked_Q, Embarked_S; C como referencia).
- Columnas irrelevantes:
  • PassengerId, Name, Ticket eliminadas.
- Resultado:
  • Los datasets limpios (train_clean.csv, test_clean.csv) tienen columnas numéricas y categóricas codificadas, sin nulos, listos para entrenar un modelo.

Código el ejemplo:
https://carbon.now.sh/V7vDnklADRrgksw39E8F

```python
import pandas as pd

# Leer los archivos CSV
try:
    gender_submission = pd.read_csv('gender_submission.csv')
    test = pd.read_csv('test.csv')
    train = pd.read_csv('train.csv')
except FileNotFoundError:
    print("Error: Uno o más archivos CSV no se encuentran en el directorio actual. Verifica las
rutas.")
    exit()

# Función para analizar problemas en un DataFrame
def analyze_problems(df, name):
    print(f"\nAnálisis de problemas en {name}:")
    print("\n1. Información general:")
    print(df.info())  # Tipos de datos y nulos
    print("\n2. Valores nulos:")
    print(df.isnull().sum())  # Conteo de nulos por columna
    print("\n3. Duplicados:")
    print(f"Filas duplicadas: {df.duplicated().sum()}")
    print("\n4. Estadísticas descriptivas (numéricas):")
    print(df.describe())  # Para detectar outliers
    print("\n5. Valores únicos en categóricas:")
    for col in df.select_dtypes(include='object').columns:
        print(f"{col}: {df[col].nunique()} valores únicos, ejemplos: {df[col].head(3).tolist()}")

# Analizar cada dataset
analyze_problems(gender_submission, "gender_submission.csv")
analyze_problems(test, "test.csv")
analyze_problems(train, "train.csv")
```

5. ¿Por qué se dice que “los datos son el nuevo petróleo”?

Porque los datos en bruto no tienen valor hasta que se procesan, analizan y refinan, igual que el petróleo. Los datos impulsan modelos de IA, decisiones empresariales y nuevos productos, convirtiéndose en un recurso estratégico.

6. ¿Qué riesgos existen si los datos están sesgados?

El principal riesgo es que el modelo aprenda y refuerce esos sesgos, lo que puede llevar a discriminación. Ejemplos: sistemas de selección de personal que penalizan a mujeres, algoritmos judiciales que perpetúan prejuicios raciales, o recomendaciones que excluyen minorías.

7. Imagina que queremos entrenar un sistema que recomiende restaurantes:

- - Datos necesarios: ubicación de usuarios, tipos de cocina preferida, reseñas, horarios, precios, historial de visitas.
- - Cómo recopilarlos: encuestas, aplicaciones móviles, datos públicos de Google Maps, Yelp o TripAdvisor.
- - Problemas éticos: privacidad de usuarios, manipulación de reseñas falsas, sesgo hacia restaurantes grandes frente a pequeños negocios locales.

8. Realiza los siguientes pasos y responde con Iris dataset

(https://scikit-learn.org/1.5/auto_examples/datasets/plot_iris_dataset.html)

Cargar el dataset Iris con scikit-learn.

- ¿Cuántas muestras y cuántas clases tiene?

  El dataset Iris tiene 150 muestras y 3 clases (setosa, versicolor, virginica)

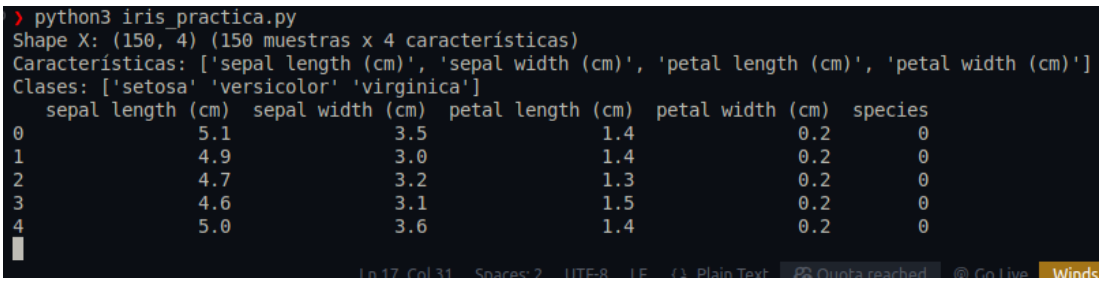
- ¿Qué representan las features (columnas)?

  Las features representan medidas en centímetros de las flores

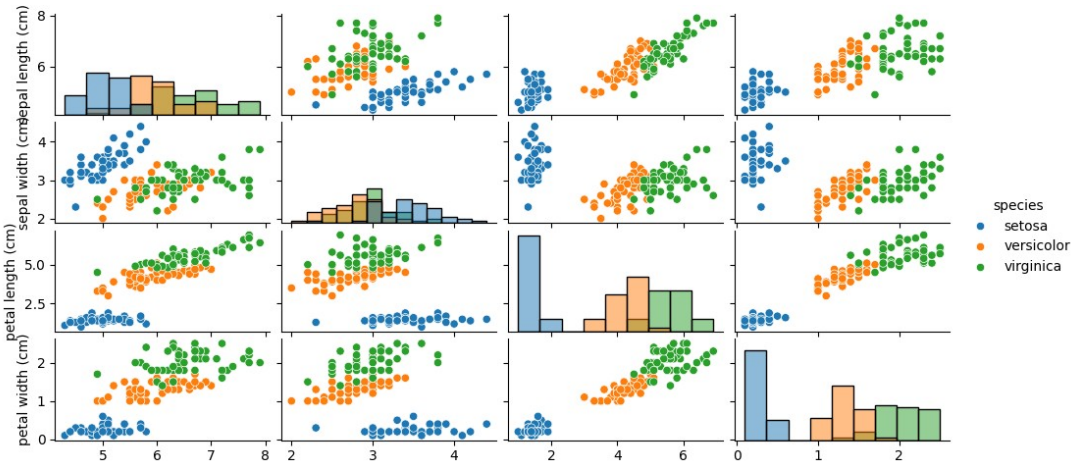
  - sepal length (cm): longitud del sépalo.
  - sepal width (cm): ancho del sépalo.
  - petal length (cm): longitud del pétalo.
  - petal width (cm): ancho del pétalo.

Exploración inicial

- Mostrar las primeras 5 filas.



- Graficar un pairplot (usando seaborn.pairplot) coloreado por especie.



División de datos

- Separar en entrenamiento (70%) y prueba (30%) con train_test_split.
- **Descripción de la división:**

  Se dividió el dataset Iris (150 muestras, 4 características) en conjuntos de entrenamiento y prueba usando train_test_split con test_size=0.3, random_state=42 y stratify=y. Esto resultó en:

  - Conjunto de entrenamiento: 105 muestras (70%), con 4 características.
  - Conjunto de prueba: 45 muestras (30%), con 4 características. La opción stratify=y asegura que la proporción de las clases (setosa, versicolor, virginica) sea la misma en ambos conjuntos, manteniendo el balance del dataset original.

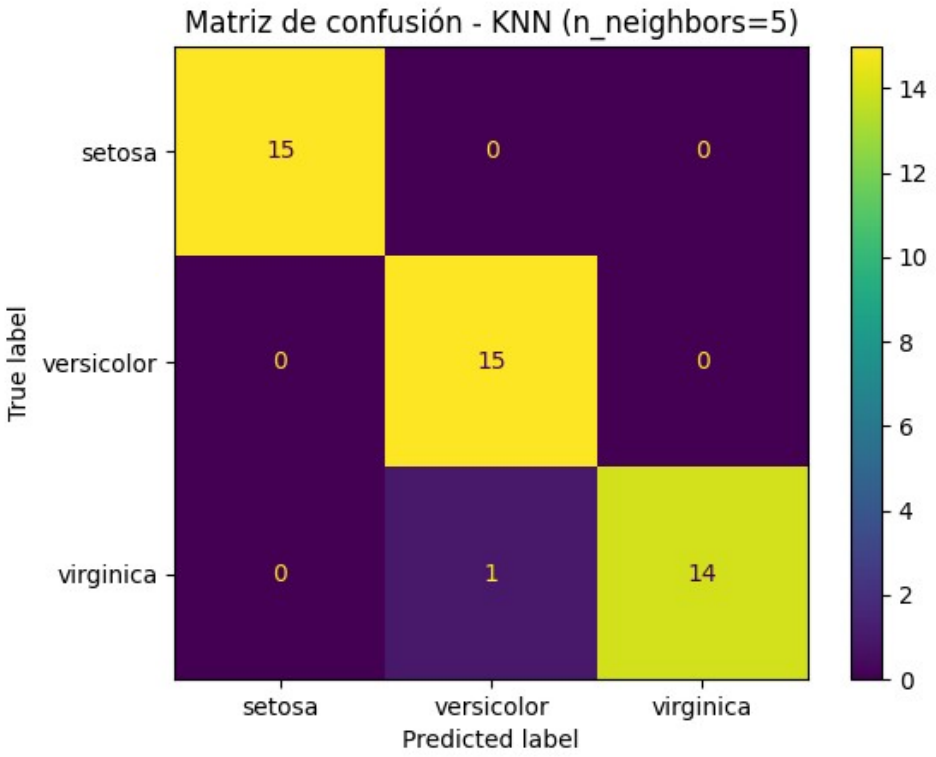
Entrenamiento con KNN

- Entrenar un clasificador KNeighborsClassifier con n_neighbors = 5.

```
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       0.94      1.00      0.97        15
   virginica       1.00      0.93      0.97        15

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```



- Predecir en el conjunto de prueba.
- Calcular accuracy, matriz de confusión y classification report.

  Entrenamiento con KNN: Se entrenó un clasificador KNeighborsClassifier con n_neighbors=5 usando el conjunto de entrenamiento (105 muestras). Se hicieron predicciones en el conjunto de prueba (45 muestras). Los resultados son:

  - **Accuracy**: 0.978 (97.8%), indicando un alto rendimiento en la clasificación.
  - **Matriz de confusión**:

    [[15 0 0]

    [0 15 0]

    [0 1 14]]

- Setosa: 15/15 correctas (100%).
- Versicolor: 15/15 correctas (100%).
- Virginica: 14/15 correctas (93%), con 1 muestra clasificada erróneamente como versicolor.
- **Classification Report**:

  Setosa: precisión=1.00, recall=1.00, f1-score=1.00.
  Versicolor: precisión=0.94, recall=1.00, f1-score=0.97.

Virginica: precisión=1.00, recall=0.93, f1-score=0.97. Accuracy global: 0.98. La matriz de confusión visualizada muestra que el modelo es muy preciso, con un solo error en la clase virginica, clasificada como versicolor.

Entrenamiento con SVM

- Entrenar un clasificador SVC (con kernel lineal).

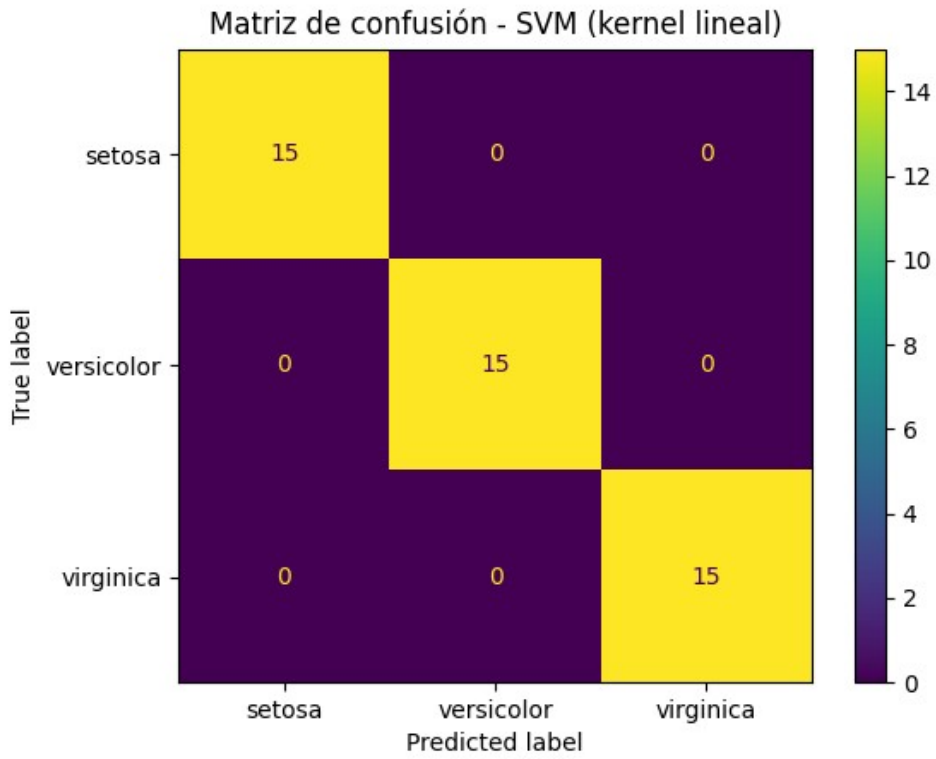
```
Accuracy del modelo SVM (kernel lineal): 1.0

Matriz de confusión:
[[15  0  0]
 [ 0 15  0]
 [ 0  0 15]]

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       1.00      1.00      1.00        15
   virginica       1.00      1.00      1.00        15

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```



Se entrenó un clasificador SVC con kernel='linear' usando el conjunto de entrenamiento (105 muestras). Se hicieron predicciones en el conjunto de prueba (45 muestras). Los resultados son:

- **Accuracy**: 1.0 (100%), indicando un rendimiento perfecto en la clasificación.
- **Matriz de confusión**:

  [[15 0 0]

  [0 15 0]

  [0 0 15]]

- Setosa: 15/15 correctas (100%).
- Versicolor: 15/15 correctas (100%).
- Virginica: 15/15 correctas (100%).
- **Classification Report**:
  - Setosa: precisión=1.00, recall=1.00, f1-score=1.00.
  - Versicolor: precisión=1.00, recall=1.00, f1-score=1.00.
  - Virginica: precisión=1.00, recall=1.00, f1-score=1.00.
  - Accuracy global: 1.00. La matriz de confusión visualizada muestra que el modelo no cometió errores, clasificando correctamente todas las muestras de las tres clases.

Comparación de modelos

- **¿Cuál tiene mejor accuracy?**

  El modelo SVM (kernel lineal) tiene mejor accuracy, con un valor de 1.0 (100%), comparado con el modelo KNN (n_neighbors=5), que obtuvo un accuracy de 0.978 (97.8%). Esto indica que SVM clasificó correctamente todas las muestras del conjunto de prueba, mientras que KNN cometió un error.

- ¿Dónde se confunden más (mirando la matriz de confusión)?

  KNN: La matriz de confusión muestra un error: 1 muestra de la clase virginica fue clasificada incorrectamente como versicolor (valor 1 en la posición [virginica, versicolor]). No hubo errores en setosa ni versicolor.

  [[15 0 0]

  [0 15 0]

  [0 1 14]]

  SVM: La matriz de confusión no muestra errores; todas las muestras fueron clasificadas correctamente (diagonal con 15, 15, 15 y ceros fuera de la diagonal).

  [[15 0 0]

  [0 15 0]

[0 0 15]]

Conclusión: KNN se confunde más, específicamente entre virginica y versicolor, mientras que SVM no presenta confusiones. Esto es consistente con el pairplot, donde virginica y versicolor tienen cierta superposición en algunas características (como sepal length y sepal width), pero SVM logra una separación perfecta con un hiperplano lineal.

Conclusiones

La inteligencia artificial abarca distintos enfoques que han evolucionado a lo largo de la historia. La IA simbólica sentó las bases en los años 80, pero hoy en día los enfoques más utilizados en el ámbito tecnológico y en aplicaciones web son el Machine Learning y el Deep Learning. Estos permiten procesar grandes cantidades de datos y generar predicciones más precisas. Sin embargo, los riesgos asociados a los datos sesgados ponen de manifiesto la importancia de la ética en el desarrollo y aplicación de sistemas de IA.

Bibliografía

- Russell, S., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach. Prentice Hall.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
- Documentación oficial de scikit-learn: https://scikit-learn.org/
- Kaggle - Titanic Dataset: https://www.kaggle.com/competitions/titanic