Laboratorio 1: Métricas de Desempeño

Inteligencia Artificial - CC3085

José Antonio Mérida Castejón

24 de enero de 2026

Task 1: El Dilema del Negocio

Imagine que usted ha sido contratado como Lead Data Scientist para tres startups diferentes. Para cada caso, responda las preguntas justificando su respuesta.

Caso A: "MediScan AI"

Están desarrollando un modelo para detectar un tipo de cáncer raro en etapas tempranas a partir de radiografías.

1. *En este contexto, ¿qué es peor: un **Falso Positivo** o **Falso Negativo**?*

   En este caso, un **falso negativo** es significativamente más peligroso (o *peor*). En el caso de un **falso negativo**, el paciente abandona completamente la posibilidad de tener la enfermedad y se elimina cualquier posibilidad de recibir tratamiento oportuno. Mientras que en un **falso positivo**, el paciente continúa con el monitoreo y exámenes necesarios a pesar de no tener la enfermdad. Una opción genera gastos monetarios en exámenes (o tratamientos), mientras que el otro resulta siendo potencialmente mortal.

   En mi opinión, el modelo debería de tener un *threshold* sumamente bajo para indicar una potencial presencia de la enfermedad. La información proveída por el modelo podría servir como un tipo de *screening* temprano, seguido de un régimen establecido de exámenes para continuar con el monitoreo del paciente. En caso que la tecnología se vea limitada las unicas opciones fueran *tratamiento* o *no tratamiento*, seguiría considerando peor un **falso negativo** aunque se deberían de examinar los efectos secundarios o demás consecuencias del tratamiento.

2. *Basado en lo anterior, si tuviera que optimizar el modelo priorizando una sola métrica entre **Precisión (Precision)** y **Sensibilidad (Recall / Sensitivity)**, ¿cuál escogería y por qué?*

   Definitivamente sería **recall**. Utilizando precisión, el modelo se estaría enfocando en minimizar los **falsos positivos**. Por otro lado, el **recall** busca minimizar los **falsos negativos**. En otras palabras, utilizando **recall** estamos entrenando al modelo para identificar la mayor cantidad de casos de cáncer correctamente sin importar que tan seguro esté, mientras que utilizando **precision** lo estamos entrenando a priorizar la certeza absoluta antes de indicar cáncer. Esto sería como un doctor que observa ciertas indicaciones de la enfermedad, dónde priorizar **precision** incentiva al doctor a ignorar los indicadores y no informar al paciente (negligencia, si me preguntan mi opinón).

3. *¿Por qué el Accuracy sería una métrica peligrosa para presentar a los inversionistas en este caso específico?*

   Primero que nada, el **accuracy** realmente no cuenta la historia completa. Para cualquier modelo que estemos diseñando, es clave tomar múltiples métricas en cuenta a la hora de analizar su rendimiento y utilidad en el mundo real. Esto cubre las métricas de **recall** y **precision** discutidas anteriormente, dónde buscamos analizar más allá y descubrir un poco más sobre que tipos de errores comete el modelo. Luego, podemos considerar que el **accuracy** de por si es sumamente inapropiado debido a la rareza de la enfermedad que buscamos predecir. Si tuviésemos 1 radiografía (o paciente) de cada 20 presentando esta enfermedad, podríamos obtener un **accuracy** del 95 % simplemente prediciendo que ninguno de los pacientes está enfermo. Por último, como discutimos anteriormente, tenemos una situación de la vida real dónde las consecuencias de cada tipo de error son completamente diferentes. El presentar esta métrica a los accionistas sería un error total, ya que no representa adecuadamente el rendimiento del modelo *respecto al problema que se busca resolver.*

Caso B: "SpamGuard" (Filtro de Correos)

Están creando un filtro de spam para una corporación grande.

1. *¿Qué error causaría más molestia y pérdida de productividad a los empleados: que un correo de spam llegue al inbox (FP o FN dependiendo de su definición) o que un correo importante de un cliente se vaya a la carpeta de spam?*

   En este caso, un correo de spam llegando al inbox sería una inconveniencia ligera para el trabajador que lo reciba. Mientras tanto, si un correo importante va a la carpeta de spam tenemos una situación catastrófica. El no ver el correo puede tener consecuencias dentro de la empresa, económicas o hasta legales para la persona que lo reciba. Adicionalmente, esta incertidumbre llevaría a que el resto de empleados estuviesen revisando la carpeta de spam constantemente. Como resultante, existiría una pérdida de productividad significativa, dónde el modelo contribuye al problema en vez de resolverlo.

2. *¿Qué métrica priorizaría aquí: **Precision** o **Recall**? (Defina cual es su clase positiva)*

   Empezamos definiendo *spam* como nuestra clase positiva. En este caso, definitivamente se debería priorizar el **precision** en lugar de **recall**. Utilizando una analogía, digamos que tenemos una persona encargada de revisar cada correo individualmente. Priorizando **precision**, esta persona clasificaría un correo como spam si tuviera certeza que un correo pertenece a esta categoría. Si fuéramos en cambio por **recall**, esta persona elegiría clasificar como spam a cualquier correo que tuviera índices de serlo.

Caso C: "Zillow 2.0" (Predicción de Precios de Casas)

Están prediciendo el valor de mercado de propiedades. Tienen un modelo con las variables: Metros Cuadrados, Ubicación, Número de Cuartos. Ahora, un junior engineer sugiere agregar 50 variables nuevas (como "color de la puerta", "nombre del dueño anterior", etc.) y nota que el R^2 subió ligeramente.

1. *¿Deberíamos confiar en este aumento del R^2 para decir que el modelo es mejor?*

   No. Primero, al diseñar modelos debemos de tomar en cuenta la *explicabilidad* y no únicamente el rendimiento. Puede ser favorable tener un modelo con rendimiento similar pero con

menos variables, así podemos identificar más fácilmente *por que* y *como* funciona el modelo. Luego, la métrica de de R^2 sigue aumentando al agregar nuevas variables independientes, sin importar cuán relacionadas estén con nuestra variable objetivo. De manera que, aunque prefiriésemos un modelo con una cantidad de variables considerablemente mayor por un incremento pequeño de rendimiento, este incremento es un derivado de las variables adicionales y no refleja adecuadamente la calidad del modelo. Por último, deberíamos de tener bastante claro que algunas de estas variables muy probablemente solo estén agregando ruido al modelo. No necesitamos ser expertos en el tema para darnos cuenta que probablemente no exista una correlación significativa entre el nombre del dueño anterior y el valor de mercado de una propiedad. Esto también resalta la necesidad de un EDA, dónde las decisiones a tomar al diseñar el modelo deben de estar bien fundamentadas.

2. *¿Qué métrica debería observar para saber si esas nuevas variables aportan valor o son solo ruido? Justifíquese basándose en la métrica de R^2 ajustado.*

   La métrica a observar sería R^2 *ajustado*. Aquí se modifica la fórmula, agregando una penalización por agregar variables nuevas. La fórmula modificada es la siguiente:

   $$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

   Donde:

   - R^2: es el R^2 de la muestra
   - N: es el tamaño de la muestra
   - p: es el número de variables independientes

   Esto significa que, al agregar una variable el denominador $N - p - 1$ se vuelve más pequeño. Al dividir dentro de un valor más pequeño, estamos volviendo la cifra del lado derecho más grande y esto resulta en un R^2 *ajustado* más pequeño. Esto resulta en que las variables adicionales puedan *disminuir* el rendimiento del modelo si no proveen valor, mientras que utilizando R^2 únicamente pueden mantenerlo igual o aumentarlo. Esto quiere decir que para tener un R^2 *ajustado* más alto, cada variable independiente debe proveer un valor más alto que la penalización dada por el ajuste.

   Adicionalmente, para el entrenamiento del modelo podemos tomar en cuenta diferentes técnicas de regularización como *Lasso (L1)* o *Ridge (L2)*. En este caso, personalmente optaría por *Lasso* para ayudar a eliminar algunas de las variables nuevas del junior engineer (al ser la regularización más *agresiva*) y darle algo de validez a su sugerencia de implementar nuevas variables. De esta manera podemos explorar brevemente algunas sugerencias, manteniendo clara la visión de optimizar en cuanto a la métrica R^2 *ajustado*.

Task 2 - Ingeniería de Datos

Utilizando Python (Pandas/Numpy), simule y procese un dataset. No se permite utilizar funciones mágicas de limpieza automática (como SimpleImputer de sklearn), deben hacerlo con lógica de programación para demostrar que entienden el proceso.

Todo este Task fue entregado como `task_2.py` en Canvas, decidí colocar bloques de código tomados del archivo aquí en el informe. También se encuentran ambos este informe y los archivos .py en el *repositorio de GitHub*.

1. **Generación de Dataset Sucio**.

   a) *Cree un DataFrame de 100 filas y 3 columnas: Edad, Salario y Compro_Producto (0 o 1).*

   b) *Introduzca intencionalmente valores NaN en el 10 % de la columna Edad.*

   c) *Genere un desbalance de clases en Compro_Producto: 90 filas deben ser '0' y 10 filas '1'.*

```python
# Initialize empty list of dictionaries to store data
data_list = []

# Add 100 entries
for i in range(100):
    # Initialize row with random values, all '0's initially for 'Compro_Producto'
    row = {
        "Edad": np.random.randint(16, 90),
        "Salario": np.random.uniform(4000, 50000),
        "Compro_Producto": 0,
    }
    # Add row to list
    data_list.append(row)

# Turn list into DF
df = pd.DataFrame(data_list)

# Set 10 random entries in 'Edad' to NaN
df.loc[df.sample(10).index, "Edad"] = np.nan

# Set 10 random samples in 'Compro_Producto' to '1'
df.loc[df.sample(10).index, "Compro_Producto"] = 1
```

2. **Manejo de Datos Faltantes**

   a) *Escriba un algoritmo que recorra la columna Edad.*

   b) *Si encuentra algun valor faltante, rellénelo con el promedio de las edades existentes.*

   c) *Pregunta extra en código (comentario): ¿En qué situación usar el promedio sería una mala idea y sería mejor usar la mediana?*

```python
# Calculate mean for 'Edad' column
mean = df["Edad"].mean()

# Manually iterate through all rows replacing nans
for i, row in df.iterrows():
    if pd.isna(row["Edad"]):
        df.at[i, "Edad"] = mean
```

Para el inciso C, la respuesta se encuentra en el código y adicionalmente se responde a continuación:

El promedio es una mala idea si se tienen outliers extremos. Por ejemplo, para la lista [1,1,x,1000] (a parte de probablemente ser un error de entrada) el promedio es de 334 mientras que viendo los datos crudos la mejor idea podría ser un valor cercano a 1. También se

puede tomar un "approach" más visual y basarse en la longitud de las colas para identificar el sesgo. Otro ejemplo sería el salaio o ingresos mensuales promedios en USA donde el top 0.1 % tiene del 13-14 % de las riquezas. Aquí el promedio y mediana de los ingresos son $75000 y $55000 aproximadamente, definitivamente deberíamos utilizar la mediana para representar los valores faltantes.

3. **Manejo de Datos Desbalanceados (Undersampling Manual):**

   a) Debe mantener todas las filas de la clase minoritaria ('1')

   b) Debe seleccionar aleatoriamente un número de filas de la clase mayoritaria ('0') igual al número de filas de la clase minoritaria.

   c) El resultado debe ser un nuevo DataFrame balanceado

```python
# Function to balance the dataset
def undersample(df, target, min, maj):
    minority = df[df[target] == min]
    majority = df[df[target] == maj]

    n_min = len(minority)

    majority_sample = majority.sample(n=n_min)

    balanced_df = pd.concat([minority, majority_sample])

    return balanced_df


# Balance the dataset
balanced_df = undersample(df, "Compro_Producto", 1, 0)
```

Task 3 - Ingeniería de Datos

Escriba dos funciones en Python desde 0 (usando math o numpy) para calcular el error de dos listas de valores:

- y_real = [100, 150, 200, 250, 300] (Valores reales)
- y_pred = [110, 140, 210, 240, 500] (Predicciones - Note el error masivo en el último dato)

Todo este Task fue entregado como `task_3.py` en Canvas, decidí colocar bloques de código tomados del archivo aquí en el informe. También se encuentran ambos este informe y los archivos .py en el *repositorio de GitHub*.

1. *Implemente la fórmula de RMSE vista en clase, como una función*

```python
def rmse(y_real, y_pred):
    sum = 0
    n = len(y_real)
    for i in range(n):
        sum += (y_pred[i] - y_real[i]) ** 2

    avg = sum / n

    return math.sqrt(avg)
```

2. *Implemente la fórmula de MAE vista en clase, como una función*

```
def mae(y_real, y_pred):
    sum = 0
    n = len(y_real)
    for i in range(n):
        sum += abs(y_pred[i] - y_real[i])

    return sum / n
```

3. *Comparación (Adicionalmente, se encuentran los resultados y el análisis como prints dentro del código):*

   - **Resultados**

     MAE: 48.0, RMSE: 89.89

   - **¿Cuál de las dos métricas penalizó más el error del último dato?**

     El *RMSE*, definitivamiente. Esto se debe a la manera que se realizan las mediciones, dónde el *MAE* todos los errores se tratan por igual. Mientras que en el *RMSE*, los errores más grandes se ven más penalizados. Podemos tomar también como ejemplo, digamos que tenemos una lista con 10 observaciones. Bajo el *MAE*, es equivalente que todas nuestras predicciones tuvieran un error de 50 a que una sola predicción tuviera un error de 500. Mientras que el *RMSE*, similar a como fue en los resultados de este código, penalizaría más la predicción *absurda* con un error de 500.

   - **¿Por qué esto es importante si estamos prediciendo, por ejemplo, dosis de medicamentos?**

     En el caso de la dosis de medicamentos, las predicciones muy poco certeras pueden llegar a tener repercusiones sobre la salud del paciente. Es un caso dónde es completamente diferente tener 10 casos con un error pequeño, en comparación a 9 casos perfectos y un caso dónde la dosis es letal. Siguiendo el ejemplo del inciso anterior, digamos que las dosis se mantienen en valores alrededor de 250mg. Con todos los errores alrededor de 50mg, todos los pacientes estarían recibiendo +/- 50mg (20 % de una dosis promedio) en comparación a su dosis adecuada. Por otro lado, tendríamos 9 pacientes con su dosis perfecta pero uno de ellos recibió 500mg extra. Es decir, alrededor de 3x la dosis promedio de medicamento que deberían recibir los pacientes. Claramente, deberíamos penalizar más los errores catastróficos utilizando métricas como *RMSE*.

     Como nota adicional, también tenemos diferentes métricas que podemos utilizar. *MAE* utiliza la norma L_1, mientras que *RMSE* la norma L_2. Existen normas L_p (que no voy a definir por brevedad), dónde mientras p aumenta se penalizan aún más los valores extremos. Aquí podría ser una buena idea utilizar la Norma de Chebyshev (L_∞), dónde se ignora completamente el promedio y únicamente se enfoca en el peor caso posible.