TALLER DE PROCESAMIENTO DE SEÑALES

2do Cuatrimestre 2026 - Trabajo Práctico Nº 3 - Regresión Logística

---

Se realizar mantenimiento predictivo de una máquina de fresado.

(**a**) *Exploración del dataset:*

1. Descargar la bae de datos `https://www.kaggle.com/api/v1/datasets/download/stephanmatzka/predictive-maintenance-dataset-ai4i-2020`.
2. Se desea predecir si la máquina fallará (**Machine failure**) a partir de los siguientes predictores:
   - **Type:** Calidad del producto: L (baja), M (media) o H (alta).
   - **Air temperature:** Temperatura del aire alrededor de la máquina (Kelvin).
   - **Process temperature:** Temperatura del proceso de mecanizado (Kelvin).
   - **Rotational speed:** Velocidad de rotación de la herramienta (revoluciones por minuto).
   - **Torque:** Torque aplicado durante el mecanizado (Newton-metro).
   - **Tool wear:** Tiempo acumulado de desgaste de la herramienta (minutos).

   Construir la base de datos dejando solamente estas columnas.
3. Indicar la proporción de las variables categóricas representando probabilidades (incluido el target).
4. Para analizar las variables numéricas, utilice el comando `pairplot` (seaborn), indicando en colores diferentes los datos provenientes de fresadoras falladas y de fresadoras funcionales. Explique qué representan los gráficos.
5. Extraer conclusiones sobre los datos a partir de los últimos dos incisos.
6. Utilice el comando `train_test_split` (sklearn) para definir dos conjuntos de datos. El conjunto de entrenamiento debe contener el 80 % de las muestras, el resto serán de testeo.

(**b**) *Preprocesamiento:*

1. Los comandos `OneHotEncoder` y `OrdinalEncoder` (sklearn) se utilizan para codificar variables categóricas de más de dos clases. Aplicar una de ellas para cada predictor categórico, justificando la elección.
2. Utilice el comando `PolynomialFeatures` (sklearn) para crear un mapa polinómico de orden 2 sobre los predictores numéricos.
3. La vectorización correspondiente a un mapa polinómico de orden ν sobre d predictores, ¿cuantas columnas posee?. Justificar.
4. Combinar y posteriormente normalizar las salidas de las operaciones anteriores. : Herramientas recomendadas son `ColumnTransformer` y `StandardScaler` (sklearn).

(**c**) *Clasificación:*

1. Hallar una expresión para el gradiente del riesgo empírico regularizado. Justificar.
2. Implementar utilizando solamente `numpy` una regresión logística binaria con regularización L2, utilizando gradiente descendente. El código debe estar estructurado de la siguiente manera:

```
class RegresionLogistica:
    # Opcional, para inicializar atributos o declarar hiperparámetros
    def __init__(self,...

    # Etapa de entrenamiento
    def fit(self,X):
```

```
# Etapa de testeo soft
def predict_proba(self,X):

# Etapa de testeo hard
def predict(self,X):

# Computar el Accuracy
def accuracy(self,X,y):

# Computar la Cross-Entropy
def cross_entropy(self,X,y):

# Computar Precision
def precision(self,X,y):

# Computar Recall
def recall(self,X,y):

# Computar F1 score
def f1_score(self,X,y):
```

3. Utilizar `Pipeline` (sklearn) para combinar las operaciones anteriores.
4. Entrenar el algoritmo eligiendo un *learning rate* y una cantidad de iteraciones tal que el riesgo empírico de entrenamiento parezca converger (visualmente). Graficar dicho riesgo en función del número de iteraciones. Utilice un hiper-parámetro de regularización $\lambda = 0.1$.
5. OPCIONAL: Modificar el comando `Pipeline` para que facilite el uso de métodos no estandar como *accuracy*, la *cross entropy*, etc. Heredar sus atributos y métodos habituales de sklearn.
6. Reportar el *accuracy*, la *cross entropy*, *precision*, *recall* y *f1-score* de entrenamiento y testeo.
7. Extraer conclusiones de los resultados.

(**d**) *Curva ROC:* En este inciso en particular, utilizaremos el conjunto de testeo como validación, para calibrar el umbral asociado a las curvas ROC.

1. A partir de la salida de `predict_proba` del conjunto de testeo, implementar la curva ROC. Marcar con una cruz el umbral correspondiente a la decisión tomada en el inciso anterior.
2. Indicar el *Equal Error Rate* y el umbral necesario para alcanzarlo. Marcar con una cruz el umbral correspondiente a esta decisión.
3. Reportar el *accuracy*, *precision*, *recall* y *f1-score* de entrenamiento y testeo, para el umbral elegido en el punto anterior. Extraer conclusiones de los resultados. ☞: Deberá redefinir los métodos para que el `predict` permita cambio de umbral.
4. Elegir un umbral de manera de optimizar la *f1-score*. Reportar el *accuracy*, *precision*, *recall* y *f1-score* de entrenamiento y testeo y graficar el punto sobre la ROC.
5. OPCIONAL: Probar con otros grados de polinomio, hiper-parámetro de regularización y umbral para mejorar el resultado.
6. Para esta aplicación en particular, ¿que métrica considera más relevante? Justificar.