IA - Clase 2A

Aprendizaje de Máquina

(ML – Machine Learning)

Aprendizaje de Máquina
K-NN k-Nearest Neighbors

- k-Nearest Neighbors (k-NN) es un algoritmo de aprendizaje supervisado que se utiliza tanto para clasificación como para regresión.
  - Dado un conjunto de entrenamiento con pares (x_i, y_i) donde $x_i \in R^d$ es la etiqueta o valor de salida:
    - Para una nueva observación x, se calcula la **distancia** entre x y todos los puntos del conjunto de entrenamiento.
    - Se seleccionan los k vecinos más cercanos (los de menor distancia).
    - Se decide la salida en base a esos vecinos:
      - **Clasificación**: la clase más frecuente (voto mayoritario o ponderado por distancia).
      - **Regresión**: el promedio (o promedio ponderado) de los valores de salida.

Aprendizaje de Máquina
K-NN k-Nearest Neighbors

- Método no paramétrico y perezoso (lazy): no ajusta un modelo explícito; en predicción busca los k ejemplos más cercanos y decide por voto (clasificación) o promedio (regresión).
  - Dado un conjunto de entrenamiento con pares (x_i,y_i) donde $x_i \in R^d$ es la etiqueta o valor de salida:
    - Para una nueva observación x, se calcula la **distancia** entre x y todos los puntos del conjunto de entrenamiento.
    - Se seleccionan los k vecinos más cercanos (los de menor distancia).
    - Se decide la salida en base a esos vecinos:
      - **Clasificación**: la clase más frecuente (voto mayoritario o ponderado por distancia). Asigna la clase más frecuente (o ponderada) entre los k vecinos.
      - **Regresión**: el promedio (o promedio ponderado) de los valores de salida de los k vecinos.

Aprendizaje de Máquina
K-NN k-Nearest Neighbors – Metricas de Distancia

- Distancia de Minkowski L_p $$d_p(\mathbf{x}, \mathbf{z}) = \left(\sum_{j=1}^{d} |x_j - z_j|^p \right)^{1/p}$$

  Casos especiales:

  - $p = 2$: Euclídea d_2.
  - $p = 1$: Manhattan d_1.
  - $p \to \infty$: Chebyshev $d_\infty = \max_j |x_j - z_j|$.

- Estandarización (evita que una característica domine por escala) $$x_j^{(\text{std})} = \frac{x_j - \mu_j}{\sigma_j}$$

- Distancia de Mahalanobis (correlaciones entre catacterísticas) $$d_M(\mathbf{x}, \mathbf{z}) = \sqrt{(\mathbf{x} - \mathbf{z})^\top \mathbf{S}^{-1} (\mathbf{x} - \mathbf{z})}$$

- Distancia de Hamming $$d_H(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{d} \mathbf{1}[x_j \neq z_j]$$

Aprendizaje de Máquina
K-NN k-Nearest Neighbors – Clasificación

- Voto mayoritario (no ponderado)

$$\hat{y}(\mathbf{x}) = \arg\max_{c \in \mathcal{C}} \sum_{i \in \mathcal{N}_k(\mathbf{x})} \mathbf{1}[y_i = c]$$

- Voto ponderado por distancia

$$w_i = \frac{1}{(d(\mathbf{x}, \mathbf{x}_i) + \epsilon)^{\alpha}}, \qquad \hat{y}(\mathbf{x}) = \arg\max_{c} \sum_{i \in \mathcal{N}_k(\mathbf{x})} w_i \, \mathbf{1}[y_i = c]$$

Con $\alpha \in [1, 3]$ típico y $\epsilon > 0$ pequeño para estabilidad.

- Estimación de posterior de clase (suavizada)

$$\hat{P}(Y = c \mid \mathbf{x}) = \frac{\sum_{i \in \mathcal{N}_k(\mathbf{x})} w_i \, \mathbf{1}[y_i = c]}{\sum_{i \in \mathcal{N}_k(\mathbf{x})} w_i}$$

Aprendizaje de Máquina
K-NN k-Nearest Neighbors – Regresión

- Promedio (no ponderado)

$$\hat{f}(\mathbf{x}) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\mathbf{x})} y_i$$

- Ponderado / Nadaraya-Watson con kernel

$$w_i = K\left(\frac{d(\mathbf{x}, \mathbf{x}_i)}{h}\right), \qquad \hat{f}(\mathbf{x}) = \frac{\sum_{i \in \mathcal{N}_k(\mathbf{x})} w_i \, y_i}{\sum_{i \in \mathcal{N}_k(\mathbf{x})} w_i}$$

Aprendizaje de Máquina
K-NN – Selección de Hiperparámetro

- k: pequeño → baja varianza de sesgo pero alta varianza (sobreajuste)
- k: grande → más sesgo, menos varianza.
- Explorar k en {1,3,5,... } y seleccionar por validación cruzada.
- Métrica/ponderación: probar L1,L2, , Mahalanobis (o metric learning), y ponderaciones por distancia.

$$\text{CV-}k^\star = \arg\min_{k \in \mathcal{K}} \frac{1}{F} \sum_{f=1}^{F} \mathcal{L}^{(f)}(k)$$

Aprendizaje de Máquina
K-NN k-Nearest Neighbors - Ventajas

- Simplicidad conceptual y de implementación.
  - La idea es muy intuitiva: “los puntos cercanos tienden a compartir etiqueta/valor”.
  - Fácil de programar sin necesidad de librerías avanzadas.
  - El pseudocódigo se resume en calcular distancias y aplicar un voto o promedio.
- No paramétrico.
  - No hace supuestos sobre la distribución de los datos (a diferencia de métodos como Naive Bayes o regresión lineal).
  - Puede adaptarse a fronteras de decisión arbitrarias y muy no lineales.
  - Flexible para problemas en los que no se conoce la forma de la función subyacente.

Aprendizaje de Máquina
K-NN k-Nearest Neighbors - Ventajas

- Modelo “perezoso” (lazy learning).
  - No requiere entrenamiento costoso: solo se almacena el dataset.
  - Útil cuando la fase de entrenamiento es crítica en tiempo (ej. datos que cambian rápido).
  - Se entrena instantáneamente → el costo está en la predicción.
- Capacidad de adaptación local.
  - La predicción se hace en función del vecindario local de la muestra:
    - Permite capturar estructuras locales complejas.
    - Útil en problemas donde la relación entre variables cambia en distintas regiones del espacio.
- Eficaz con datasets pequeños o medianos.
  - Si el número de muestras es bajo y de baja dimensión, k-NN puede superar modelos más complejos.
  - Simplicidad se traduce en alta precisión.

Aprendizaje de Máquina
K-NN k-Nearest Neighbors - Ventajas

- Versatilidad.
  - Se aplica tanto a clasificación como a regresión.
  - Puede extenderse a problemas de estimación de densidad (Parzen windows).
  - Permite incorporar métricas especializadas (ej. Mahalanobis, distancia con kernels, metric learning).
- Explica soluciones en forma de ejemplos.
  - No da un modelo global explícito, permite explicar la decisión mostrando qué vecinos influyeron.
  - Tiene un valor pedagógico y práctico: “el resultado se parece a estos casos”.
- Puede ser usado con datos de distintas clases y categorías.
  - Maneja naturalmente problemas con más de dos clases.
  - Si se define una métrica adecuada, puede funcionar con variables mixtas y de diferentes categorías.

Aprendizaje de Máquina
K-NN k-Nearest Neighbors - Desventajas

- Complejidad computacional.
  - Entrenamiento: no hay fase de entrenamiento costosa (solo se almacenan los datos).
  - Predicción: para cada consulta, se calcula la distancia a los
    - n puntos $\rightarrow$ costo O(nd)
      - n número de instancias.
      - d número de dimensiones.
  - Esto lo hace poco eficiente con grandes volúmenes de datos.
- Requiere almacenamiento completo.
  - Todo el conjunto de entrenamiento debe estar disponible para realizar predicciones.
  - En aplicaciones con restricciones de memoria o en sistemas embebidos, esto es un problema.

Aprendizaje de Máquina
K-NN k-Nearest Neighbors - Desventajas

- Sensibilidad a la escala de las variables.
  - Una variable con valores grandes puede dominar la distancia.
  - Ejemplo: edad (20–60) y salario (1000–100000).
  - Es Necesario normalizar/estandarizar:

$$x_j^{(\text{std})} = \frac{x_j - \mu_j}{\sigma_j}$$

- Sensibilidad al ruido y a outliers.
  - Un solo vecino atípico puede alterar el voto o el promedio.
  - Ejemplo: en regresión, un valor extremo en los vecinos puede sesgar el resultado.
  - Se suele usar ponderación por distancia para mitigar.
    - Un outlier (valor atípico o extremo) es una observación que se desvía significativamente del patrón general de los datos.
    - Puede ser un valor muy grande o muy pequeño respecto a la mayoría.
    - Puede estar causado por errores de medición, ruido, o bien ser un dato real pero poco frecuente.
    - *Un outlier es un dato cuya distancia respecto a la media o a la mediana es bastante mayor que la mayoría de las observaciones.*

Aprendizaje de Máquina
K-NN k-Nearest Neighbors - Desventajas

- Dificultad con clases desbalanceadas.
  - Si una clase es mayoritaria, los vecinos tenderán a pertenecer a ella.
  - Resultado: baja sensibilidad hacia clases minoritarias.
  - Se pueden usar pesos por clase o técnicas de re-muestreo (SMOTE, undersampling).

- Problema de la dimensionalidad.
  - En espacios de alta dimensión, las distancias tienden a ser similares entre todos los puntos:

$$\lim_{d \to \infty} \frac{\max_i d(\mathbf{x}, \mathbf{x}_i) - \min_i d(\mathbf{x}, \mathbf{x}_i)}{\min_i d(\mathbf{x}, \mathbf{x}_i)} \to 0$$

  - Significa que no hay vecinos “cercanos” reales → el método pierde eficacia.
  - Requier usar reducción de dimensión (PCA, LDA, autoencoders).

Aprendizaje de Máquina
K-NN k-Nearest Neighbors - Desventajas

- Problemas en variables categóricas complejas.
  - No existe una métrica “natural” universal.
  - Hamming o codificación one-hot puede llevar a distancias artificiales.
  - El desempeño depende de cómo se representen las categorías.
- Selección de hiperparámetros.
  - Elección de k es crítica:
    - Muy chico → sobreajuste (alta varianza).
    - Muy grande → subajuste (alto sesgo).
  - También afecta la métrica de distancia y si se usan o no ponderaciones.
  - Depende de validación cruzada para encontrar el óptimo.
- Interpretabilidad limitada.
  - Simple en definición pero las fronteras de decisión pueden ser altamente no lineales y difíciles de interpretar.
  - No ofrece un modelo explícito de la relación entre las variables y la salida.

Aprendizaje de Máquina
Ejemplo de uso de K-NN

- En SLA para comunicaciones las características (features) como retardo y variación de retardo, nos definen la calidad de servicio.
- Se recopilan datos que alimentan a K-NN con comandos.
- Y posteriormente, con K-NN analizamos datos para la reconfiguración del perfil de ancho de banda del dispositivo que lo haga cumplir con el SLA, si es posible para la infraestructura definida en la red.
- La cantidad de datos disponibles para la evaluación justifica la implementación de la captura y evaluación de datos supervisados y no supervisados, detectando grupos de métricas aceptables y no aceptables.
- Los métodos de agrupación permiten aprender la mezcla de parámetros a partir de datos. K-NN se adopta para esta tarea y para obtener una clasificación verosímil en un problema donde la media y la varianza de los grupos de métricas están ocultas.

Aprendizaje de Máquina
Ejemplo de uso de K-NN

- Selección de muestras obtenidas a modo de ejemplo con la clasificación de tipo de servicio 1 a 4 (valores de retardo, retardo esperado, variación de retardo y el esperado, pérdida de paquetes, si cumple el SLA, las distancias calculadas, K y orden)
- Se toma K=3 para el parámetro observado a clasificar.
- Buscamos el mejor valor entre las K instancias más similares del parámetro a encontrar.
- Para definir la similaridad utilizamos una métrica que mida la distancia entre dos puntos.
- La métrica de similaridad puede ser la distancia Euclideana u otras opciones como Manhattan, Chebyshev o Hamming.

$$d(x, x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 \ldots + (x_n - x'_n)^2}$$

$K = entero\ positivo; x = parámetro\ a\ encontrar; d = métrica\ de\ similaridad$

Aprendizaje de Máquina
Ejemplo de uso de K-NN

- K-NN se ejecuta sobre la información de tráfico obtenida computando entre y cada dato observado.
- Definimos al conjunto *A* como a los *K* puntos de los datos observados cercanos a *x*.
- Se recomienda *K* que sea impar para impedir una situación en la que dos o más participantes en competencia se asignen por igual.
- Estimamos la probabilidad condicional para cada clase o fracción de puntos en A con una misma clase indicada.

$$I(x) = función\ clasificadora$$
$$1\ si\ x\ es\ vedadero - 0\ si\ x\ es\ falso$$
$$P(y = j|X = x) = \frac{1}{K}\sum_{i\epsilon A} I(y^{(i)} = j)$$

Aprendizaje de Máquina
Ejemplo de uso de K-NN

- (como dato) es asignado a la clase que tenga la probabilidad más alta.
- K-NN realiza la búsqueda dentro del grupo de datos observados para las instancias que más se parezcan a cada nueva instancia, asignando a ella la clase más común del grupo.
- Esto lleva a la definición o cálculo de una frontera o borde de decisión.
- Uno de los problemas de este método radica en cómo seleccionar , por lo que debemos evaluar los efectos de K dentro del clasificador K-NN.
- Se dice que K es un “hiperparámetro” que controla la forma de la frontera o borde de decisión. K en K-NN representa el número de datos vecinos que votan para determinar la posición de un nuevo caso observado.

Aprendizaje de Máquina
Ejemplo de uso de K-NN

- Cuanto más pequeño sea K más limitada será la distribución de los grupos de datos, haciendo que el resultado sea muy sesgado.
- Conviene seleccionar K impares para el caso de problemas con 2 clases.
- El principal problema de K-NN radica en la complexidad de su implementación ya que para que funcione necesitamos muestras de referencia dentro de los grupos correctos,
- Para poder predecir otros grupos en el futuro. K-NN es supervisado pues tratamos de clasificar un parámetro observado en base a clasificaciones conocidas de otros puntos.
- Si aplicásemos una técnica del tipo K-Mean (EM) tomaríamos un conjunto de datos no clasificados para tratar en agruparlos en manojos y sería no supervisada pues esos datos no
- devienen de clasificaciones externa previas.
- K en K-Mean determina el número de agrupaciones con la cuáles queremos terminar.

• Aprendizaje de Máquina
Ejemplo de uso de K-NN
Normalización de los Datos

- Las variables independientes en los datos de entrenamiento se suelen medir en diferentes unidades.
- Los datos discretos debemos convertirlos a numéricos.
- Es necesario estandarizar las variables antes de realizar los cálculos
- Método de estandarización:

$$x_s = \frac{x - promedio}{desviación\ estándard}$$

- Ejemplo (SLA en Comunicaciones):

	Calidad	Retardo	Variación de retardo
Sin estandarizar	1.00	4.00	3.00
Estandarizado	-1.24	-0.90	0.59

• Aprendizaje de Máquina
Ejemplo de uso de K-NN
Normalización de los Datos

- Ejemplo SLA:

Calidad de Servicio	CoS ID	Bandwidth Profile EVC & per CoS ID	Service Performance
Real Time IP- IP Telephony-IP Video			
1	6,7	CIR > 0 EIR = 0	Latency < 5 ms Jitter < 1 ms Packet Loss. < 0.001%
Critical burst data applications críticos with low latency and low packet loss (like Cloud storage)			
2	4,5	CIR > 0 EIR <= UNI Speed	Latency < 5 ms Jitter = N/S Packet Loss. < 0.01%
Burst data applications that requires Bandwidth assurance			
3	3,4	CIR > 0 EIR <= UNI Speed	Latency < 15 ms Jitter = N/S Packet Loss. < 0.1%
Best effort services			
4	0,1,2	CIR = 0 EIR = UNI Speed	Latency < 30 ms Jitter = N/S Packet Loss. < 0.5%

Aprendizaje de Máquina
Ejemplo de uso de K-NN

- Mostramos una porción de muestras obtenidas a modo de ejemplo con la clasificación de tipo de servicio 1 a 4 indicada antes, los valores de retardo, retardo esperado, variación de retardo y el esperado, pérdida de paquetes.
- Si cumple el SLA, las distancias calculadas, K y orden.
- Se toma K=3 para el parámetro observado a clasificar.
- Al final de la tabla pueden observarse los promedios y DS necesarios para la estandarización aplicada.
- Para el entrenamiento dividimos los datos entre un conjunto sobre el cual k-NN hace predicciones y otro conjunto
- de prueba o testeo que usamos para evaluar la precisión del modelo.
- La división es aleatoria y usamos un valor para establecer la proporción o relación de esa división (variable “split”).
- En el ejemplo usamos un valor de “split”de 0.60. Se realiza el testeo y entrenamiento sobre el mismo conjunto de datos (dataset).

Aprendizaje de Máquina
Ejemplo de uso de K-NN

Parámetros													
Muestra	Ancho de Banda B (Mbps)	Tipo de Servicio	Calidad	Retardo (ms)	Valor Esperado (ms)	Variación de retardo (ms)	Valor esperado (ms)	Pérdida Paquetes < a	Valor esperado < a	Cumple	Distancia	K	Orden
1	768.87	CIR > 0 EIR = 0	1	5	5	2.58	1	0.001%	0.001%	NO	1.08	1	0.42
2	111.45	CIR > 0 EIR <= UNI Speed	2	4	5	2.00	5	0.01%	0.01%	SI	1.41	2	1.41
3	153.40	CIR > 0 EIR <= UNI Speed	3	12	15	4.31	4	0.1%	0.1%	SI	8.35	3	2.00
4	775.33	CIR = 0 EIR = UNI Speed	4	26	30	2.10	2	0.5%	0.5%	SI	22.22	4	2.24
5	861.76	CIR > 0 EIR <= UNI Speed	2	6	5	2.00	5	0.01%	0.01%	NO	2.45		2.24
6	423.88	CIR > 0 EIR = 0	1	9	5	2.58	1	0.001%	0.001%	NO	5.02		2.45
7	554.66	CIR > 0 EIR <= UNI Speed	3	17	15	4.31	4	0.1%	0.1%	NO	13.22		2.45
8	975.15	CIR > 0 EIR <= UNI Speed	2	6	5	2.00	5	0.01%	0.01%	NO	2.45		2.45
9	814.01	CIR > 0 EIR = 0	1	5	5	1.00	1	0.001%	0.001%	SI	2.24		2.45
10	50.36	CIR > 0 EIR <= UNI Speed	3	12	15	4.31	4	0.1%	0.1%	SI	8.35		2.45
11	429.50	CIR > 0 EIR <= UNI Speed	3	14	15	4.31	4	0.1%	0.1%	SI	10.28		2.45
		CIR > 0											
23	948.50	CIR > 0 EIR <= UNI Speed	2	6	5	2.00	5	0.01%	0.01%	NO	2.45		18.27
24	485.66	CIR = 0 EIR = UNI Speed	4	27	30	2.10	2	0.5%	0.5%	SI	23.21		19.26
25	265.84	CIR > 0 EIR = 0	1	5	5	1.00	1	0.001%	0.001%	SI	2.24		22.22
26	216.72	CIR > 0 EIR <= UNI Speed	2	6	5	2.00	5	0.01%	0.01%	NO	2.45		23.21
27	749.62	CIR = 0 EIR = UNI Speed	4	22	30	2.10	2	0.5%	0.5%	SI	18.27		25.20
		Promedio	2.30	10.93	11.48	2.42	3.30						
		DS	1.05	7.73	9.60	0.99	1.70						

Aprendizaje de Máquina
Ejemplo de uso de K-NN

Parámetros Estandarizados													
Muestra	Ancho de Banda B (Mbps)	Tipo de Servicio	Calidad	Retardo (ms)	Valor Esperado (ms)	Variación de retardo (ms)	Valor esperado (ms)	Pérdida Paquetes < a	Valor esperado < a	Cumple	Distancia	K	
1	768.87	CIR > 0 EIR = 0	-1.24	-0.77	-0.67	0.16	-1.35	0.001%	0.001%	NO	0.44	1	0.43
2	111.45	CIR > 0 EIR <= UNI Speed	-0.28	-0.90	-0.67	-0.43	1.00	0.01%	0.01%	SI	1.39	2	0.44
3	153.40	CIR > 0 EIR <= UNI Speed	0.67	0.14	0.37	1.91	1.91	0.1%	0.1%	SI	2.55	3	0.77
4	775.33	CIR = 0 EIR = UNI Speed	1.63	1.95	1.93	-0.33	-0.33	0.5%	0.5%	NO	4.14	4	0.77
5	861.76	CIR > 0 EIR <= UNI Speed	-0.28	-0.64	-0.67	-0.43	1.00	0.01%	0.01%	NO	1.42		1.39
6	423.88	CIR > 0 EIR = 0	-1.24	-0.25	-0.67	0.16	-1.35	0.001%	0.001%	NO	0.77		1.42
7	554.66	CIR > 0 EIR <= UNI Speed	0.67	0.79	0.37	1.91	1.91	0.1%	0.1%	NO	2.87		1.42

Parámetros													
Muestra	Ancho de Banda B (Mbps)	Tipo de Servicio	Calidad	Retardo (ms)	Valor Esperado (ms)	Variación de retardo (ms)	Valor esperado (ms)	Pérdida Paquetes < a	Valor esperado < a	Cumple	Distancia	K	Orden
1	768.87	CIR > 0 EIR = 0	1	5	5	2.58	1	0.001%	0.001%	NO	1.08	1	0.42
2	111.45	CIR > 0 EIR <= UNI Speed	2	4	5	2.00	5	0.01%	0.01%	SI	1.41	2	1.41
3	153.40	CIR > 0 EIR <= UNI Speed	3	12	15	4.31	4	0.1%	0.1%	SI	8.35	3	2.00

Aprendizaje de Máquina (ML – Machine Learning)
Caso de Uso – Steel Frame

- Objetivo:
  - Predecir la geometría óptima de un perfil estructural en steel frame (espesor, sección, etc.) para resistir una determinada presión de viento, sabiendo otras variables del diseño (altura, ancho, separación entre montantes, etc.).

Entrada	Presión + altura + zona
Salida	Espesor e, tipo de perfil C100, etc.
Tipo de problema	Ingeniería asistida, Modelo ML Inverso (regresión supervisada)

Aprendizaje de Máquina (ML – Machine Learning)
Caso de Uso – Steel Frame

- Queremos aprender la función que resuelva el problema.
- Queremos un modelo que, dado el vector de condiciones x (carga de viento y geometría del panel), prediga la geometría óptima del perfil: espesor e (regresión) y/o tipo de sección c (clasificación).
  - *f(P,h,w,s,z)→espesor del perfil e*
    - P: presión de viento (N/m^2)
    - h: altura del panel
    - w: ancho del panel
    - s: separación entre montantes (ej. 0.40 m)
    - z: zona o categoría (puede codificarse)
  - Predecir:
    - e: espesor del perfil (en mm)
    - tipo de perfil si planteamos una clasificación (C90, C100, C150, etc.)

Aprendizaje de Máquina (ML – Machine Learning)
Caso de Uso – Steel Frame

- ¿Cómo obtener los datos para entrenar?
  - Hay 3 opciones:
    - Simulación con software de cálculo: generar combinaciones (pressión, altura, ancho) y registrar qué espesor cumple.
    - Normativas (CIRSOC 303, AISI, etc.): usar tablas de diseño estructural como dataset base.
    - Diseños reales existentes.

$$\mathbf{x} = [P,\ h,\ w,\ s,\ z]$$