Minería de Flujo de Datos - 2024-2025

Minería de Flujo de Datos - Reinforcement Learning

Universidad de Granada

Práctica de Flujos de Datos - Balanceo del Poste

Miguel García López

Índice

Índice de figuras

Índice de cuadros

1. Introducción

En esta práctica se requiere dar solución a un problema de aprendizaje por refuerzo (*reinforcement learning*. El problema consiste en un carrito con un poste anclado a este, el cuál ha de equilibrarse a medida que el carrito se mueve a la izquierda o a la derecha para que no se caiga.

Es un problema de flujo de datos pues en cada momento de simulación entran nuevos datos sobre la posición, velocidad, ángulo, etc, del carrito y el poste. Según la entidad se va moviendo, se actualizan estos datos y han de ser utilizados para la predicción de la acción más correcta, mover el carrito a la izquierda o a la derecha.

El objetivo principal, además de conseguir un modelo robusto y preciso para este problema, es el de analizar tres tipos de clasificador y compararlos entre ellos. Como clasificador base se impone el uso de *Naive Bayes* con modelado Gaussiano de clases.

Se deben comparar los distintos algoritmos por el *accuracy* conseguido en los datos, así como el rendimiento de ellos en el entorno simulado (recompensa obtenida).

2. Datos

Se dispone de un fichero *csv* con datos que contienen las siguientes características:

- **Posición del carro (Atributo Cart Position)**: Numérico (Valores > 0 hacia la derecha, valores < 0 hacia la izquierda, valor $= 0$ en el centro).
- **Velocidad del carro (Atributo Cart Velocity)**: Numérico. Contiene la velocidad del carro (< 0 hacia la izquierda; > 0 hacia la derecha; 0 = sin movimiento).
- **Ángulo del poste con respecto a la vertical (Atributo Pole Angle)**: Numérico (< 0 inclinado a la izquierda; > 0 inclinado a la derecha).
- **Velocidad angular a la que se mueve el poste (Atributo Pole Angular Velocity)**: Numérico (< 0 hacia la izquierda; > 0 hacia la derecha).
- **Pseudo-mejor acción posible a realizar (Atributo objetivo o clase, Action)**: Valor entero (0 = mover a la izquierda; 1 = mover a la derecha).

3. Algoritmos usados

4. Naive Bayes

El algoritmo base para las comparaciones es el *Naive Bayes*. El clasificador bayesiano es un algoritmo de clasificación basado en el Teorema de Bayes, el cual calcula la probabilidad a posteriori de una clase dado un conjunto de características de entrada. A pesar de su simplicidad, es una técnica poderosa y ampliamente utilizada en problemas de clasificación.

El modelo recibe el calificativo de "*naive*" (ingenuo) debido a su principal asunción: **independencia condicional entre las características**, es decir, se supone que todas las variables predictoras son independientes entre sí dado el valor de la clase.

Aunque esta asunción rara vez se cumple en escenarios del mundo real, *Naive Bayes* funciona de manera efectiva en muchos contextos, especialmente en dominios donde las relaciones entre variables no son complejas.

4.1. Asunciones del Naive Bayes

- **Independencia condicional**: Se asume que las características son independientes entre sí. En este caso, se asume que la variabilidad de una característica no depende del valor de otra. No obstante, las variables con las que se van a trabajar seguramente no cumplan esta condición. Es obvio que si la velocidad angular del poste cambia, lo hará el ángulo, al igual que si la posición y velocidad del carrito se modifican, éstas tendrán gran influencia en el ángulo del poste. Todas las variables están relacionadas.
- **Distribución de los datos:** *Naive Bayes* asume que las características siguen una distribución normal dentro de cada clase.
- **Balance de clases**: El modelo puede ser sensible al desequilibrio de clases (cuando una clase es mucho más frecuente que otra), lo que puede sesgar las predicciones hacia la clase mayoritaria si no se toma en cuenta.

4.2. Preprocesamiento

Para implementar el *Naive Bayes* de manera efectiva, es crucial preparar los datos adecuadamente. En este caso, no existen variables categóricas que necesiten codificarse numéricamente ya que todas las características son numéricas y continuas. El algoritmo, dado que calcula distribuciones de probabilidad para cada clase [1] sin basarse en

distancia, es invariante a la escala de los datos y, por ello, no es necesario escalar los datos.

5. HoeffdingTreeClassifier

El *HoeffdingTreeClassifier* o *VFDT* es un algoritmo de clasificación en flujo de datos diseñado para entornos no estacionarios [2]. Basado en el árbol de Hoeffding tradicional. Utiliza el límite de Hoeffding para tomar decisiones de división de nodos con garantías estadísticas, optimizando el equilibrio entre precisión y eficiencia computacional en contextos de alta velocidad de datos.

Este clasificador es especialmente eficaz en escenarios donde las relaciones entre características y clases evolucionan con el tiempo, manteniendo un modelo actualizado sin requerir re-procesamiento completo de los datos.

5.1. Propiedades del HoeffdingTreeClassifier

- **Procesamiento incremental**: Cada instancia se procesa una sola vez, consumiendo memoria constante en tiempo de ejecución, ideal para flujos de datos continuos.
- **Tolerancia a no estacionariedad**: No asume una distribución fija de los datos, permitiendo adaptarse a patrones cambiantes mediante mecanismos de detección y corrección activa.
- **Uso del límite de Hoeffding**: Determina la confiabilidad estadística de las divisiones utilizando muestras limitadas, garantizando decisiones cercanas a las que se obtendrían con datos completos.

5.2. Preprocesamiento

Los árboles de decisión, incluyendo esta variante adaptativa, son invariantes a la escala de las variables al realizar divisiones basadas en umbrales relativos. Por tanto, no se requiere normalización ni estandarización de los datos.

6. ADWINBoostingClassifier

El *ADWINBoostingClassifier* [3] es un método de ensamble adaptativo que combina la técnica de boosting con el detector de deriva *ADWIN*. Diseñado para flujos de datos con posibles cambios conceptuales, ajusta dinámicamente los pesos de las instancias y los

componentes del ensamble cuando detecta deriva, manteniendo la precisión en entornos evolutivos.

Utiliza modelos base débiles (típicamente árboles de decisión) y actualiza iterativamente sus pesos, focalizándose en instancias mal clasificadas mientras descarta componentes del ensamble afectados por deriva mediante el monitoreo continuo de *ADWIN*.

Para este caso concreto y dado que se está comparando la potencia del método de *ensemble* de *ADWIN*, se utiliza el modelo base de *Naive Bayes*. De esta forma se podrá comparar con los resultados del modelo *Naive Bayes* en singular.

6.1. Propiedades del ADWINBoostingClassifier

- **Boosting adaptativo**: Ajusta los pesos de las instancias en tiempo real, priorizando aquellas con errores recurrentes, mientras *ADWIN* verifica si los errores reflejan deriva de concepto.
- **Eliminación selectiva de componentes**: Cuando se detecta deriva, elimina los modelos base asociados a distribuciones obsoletas, preservando solo aquellos relevantes para el contexto actual.
- **Balance precisión-velocidad**: Limita la profundidad de los árboles base y el tamaño del ensamble para garantizar eficiencia computacional en flujos de alta velocidad.
- **Automonitoreo de rendimiento**: Evalúa continuamente la precisión predictiva mediante ventanas adaptativas para activar mecanismos de corrección proactivos.

6.2. Preprocesamiento

Al operar con *Naive Bayes* como modelo base, el algoritmo no requiere escalado de características. Las variables categóricas (inexistentes en este caso) necesitarían codificación, pero al trabajar con datos puramente numéricos, el preprocesamiento se reduce a garantizar un flujo continuo de datos sin transformaciones adicionales.

7. Experimentación

7.1. Parámetros

En la tabla 1 se describen los parámetros usados por cada modelo. En general se han dejado los parámetros por defecto, para no influenciar demasiado en la comparación de

Cuadro 1: Parámetros por defecto de los modelos implementados

Modelo	**Parámetros por defecto**
Naive Bayes	- No tiene hiperparámetros configurables
Hoeffding Tree	- `grace_period=200` - `split_criterion='info_gain'` - `split_confidence=1e-7` - `leaf_prediction='nba'` - `nb_threshold=0` - `nominal_attributes=None`
ADWIN Boosting	- `model=GaussianNB()` - `n_models=10`

los algoritmos. Otra opción sería realizar una búsqueda de hiperparámetros, pero es muy costoso en tiempo.

El único parámetro que ha sido modificado es el número de modelos que tiene el modelo de *boosting*, ya que por defecto solo implementa tres.

7.2. Entreno online

Se realiza un entrenamiento sobre los datos se realiza de manera secuencial, según estos van llegando. Además se guardan los valores de cada variable cada 100 iteraciones para realizar un análisis posterior de *concept drift*.

7.3. Análisis del desvío de concepto

En este análisis se grafican tendencias temporales para detectar cambios en las distribuciones de las distintas variables. Para ello se utiliza la técnica de media móvil ($ventana = min(100, len(data)//10)$), donde se suavizan las fluctuaciones para destacar tendencias a largo plazo. El objetivo es identificar cambios abruptos o graduales en las estadísticas de las variables.

Otro análisis realizado es el siguiente: se dividen los datos en segmentos temporales y se comparan sus distribuciones. Se compara el *coeficiente de variación de medias* para comprobar esos cambios de distribución mencionados.

$$CV = \frac{std(segmento)}{mean(segmento)}$$

7.4. Detección del desvío de concepto

Se realiza una detección activa de *concept drift* utilizando el algoritmo *ADWIN* [4] (*Adaptive Windowing*) en tiempo real sobre el flujo de datos del textitCartPole. Se intenta detectar tanto el desvío en las variables características como en la variable objetivo.

ADWIN es un algoritmo que ajusta dinámicamente el tamaño de una ventana de datos, descartando datos antiguos cuando detecta cambios en las estadísticas. Dada una ventana W, si existen dos subventanas W_0 y W_1 suficientemente grandes y con medias suficientemente distintas, se puede concluir que los valores esperados son diferentes y se puede eliminar la parte antigua de W. Se utiliza un umbral $\lambda = 0{,}002$.

Este método difiere con el anterior descrito dado que es una técnica de detección en tiempo real, mientras que el análisis anteriormente mencionado es más un método *post-hoc*.

7.5. Comparación estadística de métricas

Se realiza un análisis estadístico comparativo entre dos métricas clave de los modelos evaluados, la precisión (*accuracy*) de los modelos y en rendimiento basado en recompensas en el entorno de simulación.

Se mide por correlación de *Pearson* [5] y correlación de *Spearman* [6] si las métricas de precisión y rendimiento están relacionadas entre ellas.

8. Resultados

8.1. Análisis/Detección del desvío de concepto

Los resultados de la detección por medio del algoritmo *ADWIN* han dado como resultado un total de 0 puntos detectados, por lo que este método no ha detectado ningún desvío significativo durante el flujo de datos.

En cuanto al análisis *post-hoc*, si se establece un umbral de *CV=0.1*, los resultados son que se detectan potenciales desvíos tras la recolecta de datos durante el flujo. Las variables que son potencialmente afectadas por este efecto son: velocidad angular del poste, ángulo del poste, velocidad del carro y posición del carro.

El problema principal de este método es trata cada segmento como independiente, ignorando patrones temporales. Concretamente en *CartPole*, variables como la velocidad angular (y prácticamente todas) suelen tener autocorrelación temporal.

Los resultados de media y desviación estándar quedan registradas en las tablas 2, 3, 4, 5 y 6.

Segment	Mean	Std
Segment 1	0.0081	0.1099
Segment 2	0.0180	0.0887
Segment 3	0.0005	0.1154
Segment 4	0.0130	0.1164

Cuadro 2: Cart Position Statistics

Segment	Mean	Std
Segment 1	-0.0013	0.0375
Segment 2	0.0026	0.0266
Segment 3	0.0051	0.0381
Segment 4	-0.0017	0.0305

Cuadro 3: Cart Velocity Statistics

Segment	Mean	Std
Segment 1	0.0012	0.0139
Segment 2	0.0011	0.0125
Segment 3	0.0008	0.0144
Segment 4	0.0007	0.0148

Cuadro 4: Pole Angle Statistics

En las gráficas 1 se pueden observar los valores de cada variable en el tiempo. Los gráficos de *Cart Position vs Pole Angle* tienen tendencias similares con picos y caídas en momentos parecidos. Esto sugiere una correlación positiva: cuando el carrito se mueve en una dirección, el ángulo del poste también cambia en la misma dirección, lo cual tiene sentido desde el punto de vista físico.

Comparándo *Cart Velocity vs Pole Angular Velocity*, a simple vista, los picos y valles de ambos gráficos parecen estar desfasados. Esto podría indicar una correlación inversa: cuando la velocidad del carrito aumenta en una dirección, la velocidad angular del poste se ajusta en la dirección opuesta para compensar el movimiento. Este comportamiento es consistente con el control del sistema: si el carrito acelera en un sentido, el poste intenta inclinarse en la dirección contraria para recuperar el equilibrio.

Como es de esperar, muchas variables parecen estar relacionadas entre sí, por lo que la asunción de *Naive Bayes* seguramente no se cumpliría.

Segment	Mean	Std
Segment 1	0.0002	0.0590
Segment 2	-0.0032	0.0424
Segment 3	-0.0075	0.0555
Segment 4	0.0010	0.0551

Cuadro 5: Pole Angular Velocity Statistics

Segment	Mean	Std
Segment 1	0.5480	0.4977
Segment 2	0.5160	0.4997
Segment 3	0.4880	0.4999
Segment 4	0.4960	0.5000

Cuadro 6: Action Statistics

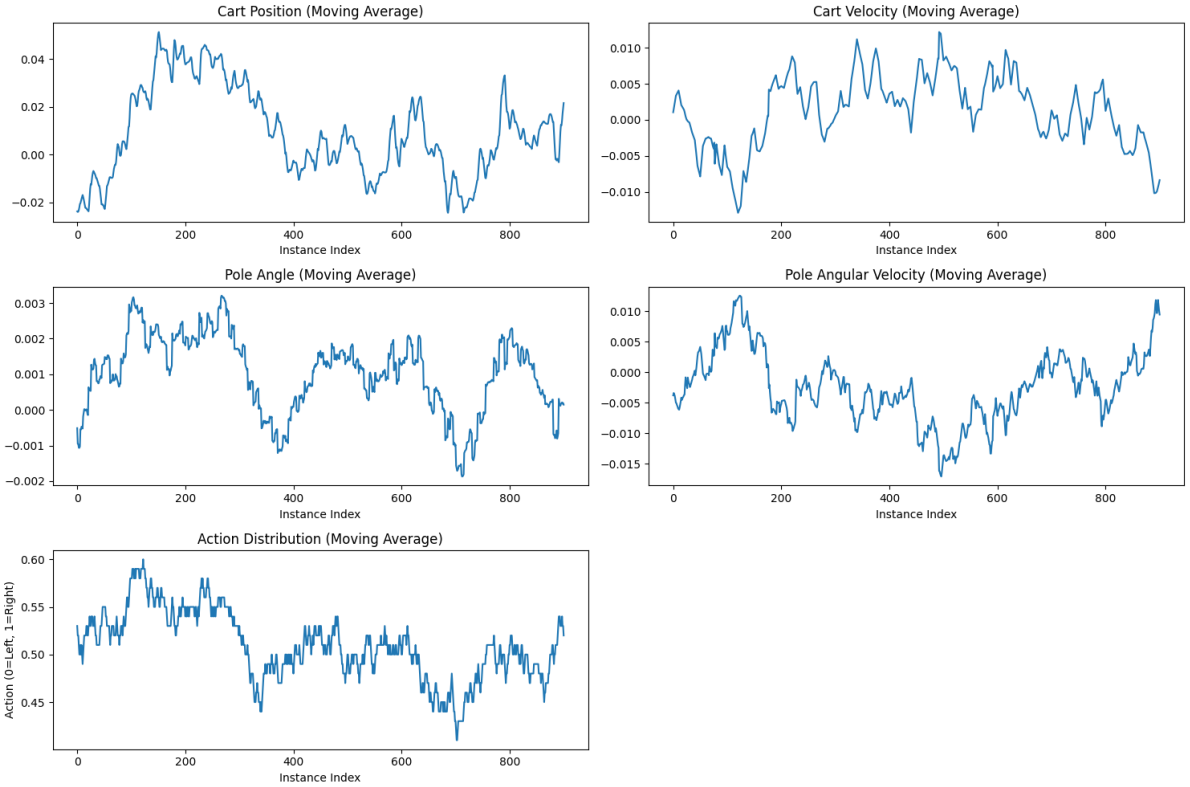

Figura 1: Moving average of each variable

8.2. Métricas de los clasificadores

Según se puede ver en la tabla de resultados en 8, el mejor modelo es *HoeffdingTreeClassifier*, el cual es un modelo estacionario, es decir, asume que la distribución de los datos es constante (no hay *concept drift*). Esto cuadra con los resultados de la detección de *ADWIN*, por lo cuál tiene sentido que este modelo haya batido a Naive Bayes con tantísima diferencia.

No solo es el que mejor *accuracy* tiene, con una diferencia grande, sino que su rendimiento en *test* es también el mejor. Se puede ver en la gráfica 2, que es el que mejor y más rápido converge, mientras que los dos modelos basados en *Naive Bayes* rinden prácticamente igual. Lo mismo puede observarse en la figura de distribución de valores de recompensa 3. Los valores son muy dispersos para *HoeffdingTreeClassifier*, ya que el modelo va adaptándose e incrementando este valor según aprende, mientras que los otros dos no superan, salvo en pocas ocasiones, el umbral de los 100 puntos.

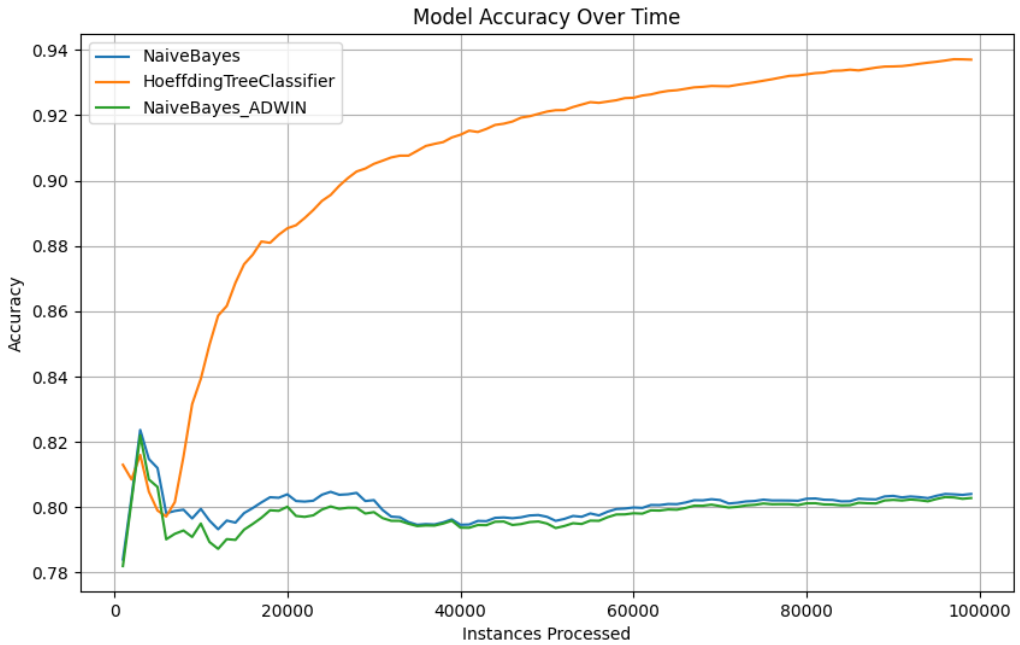

Figura 2: Accuracy over time for each model

Model	Mean Reward	Std
NaiveBayes	64.50	21.91
HoeffdingTreeClassifier	365.50	205.47
NaiveBayes_ADWIN	52.80	12.41

Cuadro 7: Model Performance in CartPole Environment

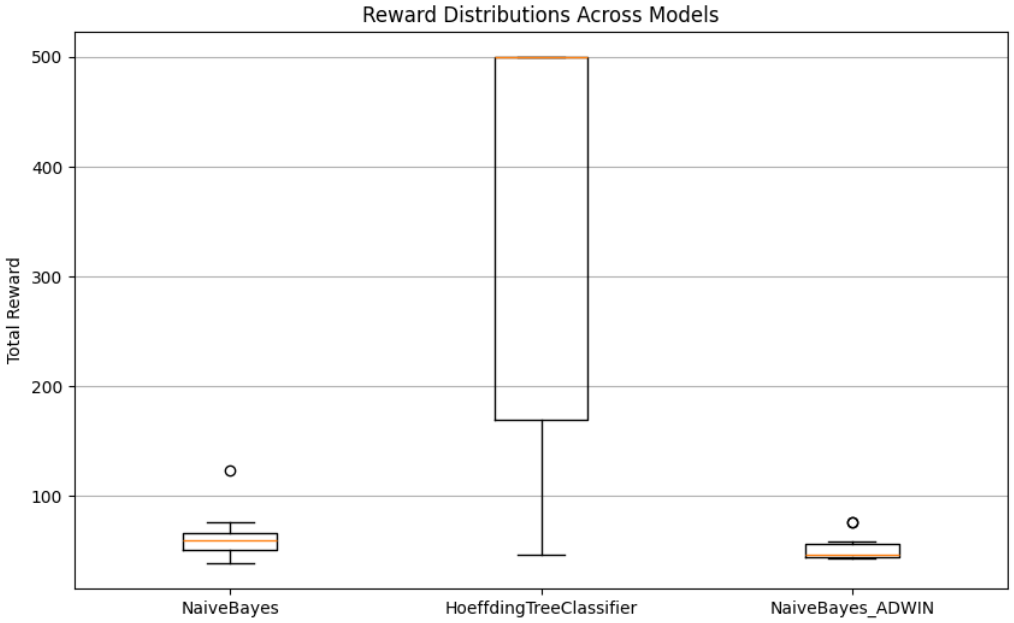

Figura 3: Rewards distribution

Model	Training Accuracy	Test Performance	Rank (Acc/Perf)
NaiveBayes	0.8046	64.50	2 / 2
HoeffdingTreeClassifier	0.9370	365.50	1 / 1
NaiveBayes_ADWIN	0.8034	52.80	3 / 3

Cuadro 8: Statistical Comparison of Models

Tanto el *test* de *Spearman* como el de *Pearson* dan valores de correlación cercanos a la unidad. Estos resultados, si bien relacionan directamente la métrica de *accuracy* con los puntos de recompensa, no son concluyentes ni generalizables a otros problemas, pues los datos para poder relacionar ambas métricas son escasos.

Si bien esto es así, podría decirse que en este problema, si parece que un modelo con buena precisión también funcionará bien en la simulación (*test*).

9. Conclusiones

9.1. ¿Presenta el flujo de datos algún tipo de Concept Drift?

Los resultados experimentales muestran una discrepancia entre métodos de detección. *ADWIN* no detectó ningún punto de deriva (*0 drift points*), sugiriendo estabilidad en el flujo.

El análisis *post-hoc* con $CV = 0{,}1$ identificó cambios potenciales en variables clave: posición del carro, velocidad angular del poste, y ángulo (Tablas 2–6).

Sin embargo, la naturaleza física del CartPole explica estas variaciones: las oscilaciones naturales del sistema (Figura 1) generan fluctuaciones estadísticas que el CV interpreta como drift, aunque corresponden a comportamientos esperados del entorno. Por tanto, no hay evidencia concluyente de *concept drift* relevante que afecte la política de control óptima.

9.2. ¿Es un modelo adaptativo mejor que uno estacionario en este problema?

El modelo estacionario *HoeffdingTreeClassifier* superó ampliamente a las alternativas adaptativas. Esto se debe ausencia de *drift* significativo y a que *HoeffdingTree* aprovecha mejor las relaciones no lineales entre variables (Figura 1), a diferencia de *Naive Bayes*, que asume independencia condicional (incorrecta en este caso).

En este escenario específico, los modelos estacionarios son suficientes y superiores.

9.3. ¿Técnicas como ADWIN o similares son relevantes en este problema?

Aunque *ADWIN* es un método robusto para detección de deriva, su utilidad aquí fue limitada ya que no identificó cambios estadísticamente significativos.

Fruto de este resultado es que el algoritmo de *boosting* basado en *ADWIN* no supera al modelo singular en el que se basa, de hecho obtienen resultados idénticos (Figura 2).

9.4. ¿Se necesita algún tipo de preprocesamiento para resolver el problema?

No es necesario por:

1. Invariabilidad a escala: Tanto *Naive Bayes* (basado en distribuciones) como *HoeffdingTree* (divisiones por umbrales relativos) son insensibles a la normalización.
2. Datos puramente numéricos: La ausencia de variables categóricas elimina la necesidad de codificación.
3. Estructura temporal: Los algoritmos procesan directamente las observaciones en bruto (posición, velocidad, etc.) sin transformaciones.

10. Bibliografía

[1] Vikramkumar, V. B y Trilochan, *Bayes and Naive Bayes Classifier*, 2014. arXiv: 1404.0933 [cs.LG]. URL: https://arxiv.org/abs/1404.0933.

[2] P. Domingos y G. Hulten, "Mining High-Speed Data Streams," en *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, págs. 71-80.

[3] N. C. Oza y S. Russell, "Online bagging and boosting," en *IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, 2001, págs. 2340-2345.

[4] A. Bifet, G. Holmes, B. Pfahringer y R. Gavaldà, "Adaptive Hoeffding Trees for Learning from Data Streams," en *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM)*, 2009, págs. 139-147.

[5] K. Pearson, "VII. Note on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, págs. 240-242, 1895, Trabajo original que introduce el coeficiente de Pearson.

[6] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 15, n.º 1, págs. 72-101, 1904, Artículo seminal de Spearman.