PUCE Virtual
EDUCACIÓN FLEXIBLE

Aprendizaje automático (Machine Learning)

Aprendizaje supervisado - Regresión logística

Clase 4

MAESTRÍA EN
SISTEMAS DE INFORMACIÓN
Mención Data Science

La excelencia no se improvisa

1. INTRODUCCIÓN DE LA CLASE

El aprendizaje supervisado es un paradigma central en el campo del aprendizaje automático, en el que se utilizan conjuntos de datos etiquetados para entrenar modelos capaces de hacer predicciones o clasificaciones. Entre los algoritmos más destacados se encuentra la **regresión logística**, un modelo lineal utilizado principalmente en problemas de clasificación binaria, que estima la probabilidad de que una observación pertenezca a una clase particular (James, Witten, Hastie, & Tibshirani, 2013). A su vez, los **clasificadores lineales**, como la regresión logística, buscan trazar una frontera de decisión lineal para separar las clases en un espacio multidimensional (Bishop, 2006). Para casos más complejos, se emplean **clasificadores multiclase**, que permiten clasificar una observación en más de dos categorías, adaptando técnicas como la regresión logística o el **Support Vector Machine (SVM)** para manejar múltiples clases de manera eficiente (Müller & Guido, 2016). Estos métodos se complementan con la técnica de ***cross-validation***, que consiste en dividir el conjunto de datos en varios subconjuntos para validar la capacidad predictiva del modelo, evitando así el sobreajuste y asegurando su rendimiento en datos no vistos (James et al., 2013).

Además, los **árboles de decisión** son una opción popular en el aprendizaje supervisado debido a su capacidad para ofrecer modelos fácilmente interpretables, segmentando los datos a través de una estructura jerárquica que facilita la toma de decisiones basadas en características específicas de los datos (Bishop, 2006). En cuanto al tratamiento de variables categóricas, el ***one hot encoding*** es una técnica fundamental que convierte variables nominales en vectores binarios, permitiendo que los algoritmos las procesen correctamente (Müller & Guido, 2016). Estos métodos y técnicas son la base para construir modelos robustos y eficientes en problemas de clasificación y regresión, permitiendo una amplia gama de aplicaciones en diversos dominios del conocimiento.

RDA: Evalúa los modelos de aprendizaje automático para identificar y cuantificar potenciales sesgos y limitaciones, empleando métricas adecuadas.

Clase 4. Aprendizaje supervisado

Regresión logística

La **regresión logística** es un algoritmo ampliamente utilizado en problemas de **clasificación**, especialmente en aquellos que involucran una variable dependiente **binaria**, es decir, en los que el objetivo es predecir uno de dos posibles resultados. A diferencia de la regresión lineal, que predice un valor continuo, la regresión logística predice la probabilidad de que una observación pertenezca a una clase específica, mediante la aplicación de una función logística (o sigmoide) ver figura 1. Esta función transforma cualquier valor real de entrada en un valor entre 0 y 1, lo que permite interpretar los resultados como probabilidades (James, Witten, Hastie, & Tibshirani, 2013). Esta probabilidad se compara con un umbral -generalmente de 0,5- para asignar la observación a una de las dos clases posibles, l**o que facilita la clasificación**.

Uno de los aspectos claves de la **regresión logística** es su capacidad para manejar **múltiples variables independientes** (también llamadas características o predictores), lo que permite modelar relaciones complejas entre las variables. Sin embargo, es importante tener en cuenta que la **regresión logística asume** una **relación lineal** entre las variables predictoras y el logaritmo de las probabilidades, lo que puede ser una limitación si los datos presentan una relación no lineal (Bishop, 2006). Para resolver esta limitación, se pueden aplicar **técnicas** como la **regularización**, que ajusta el modelo para evitar el sobreajuste (***overfitting***) y mejorar su capacidad de generalización. Además, la regresión logística se usa ampliamente no solo en la estadística tradicional, sino también en áreas como el análisis de riesgo crediticio, diagnóstico médico y clasificación de textos (Müller & Guido, 2016).

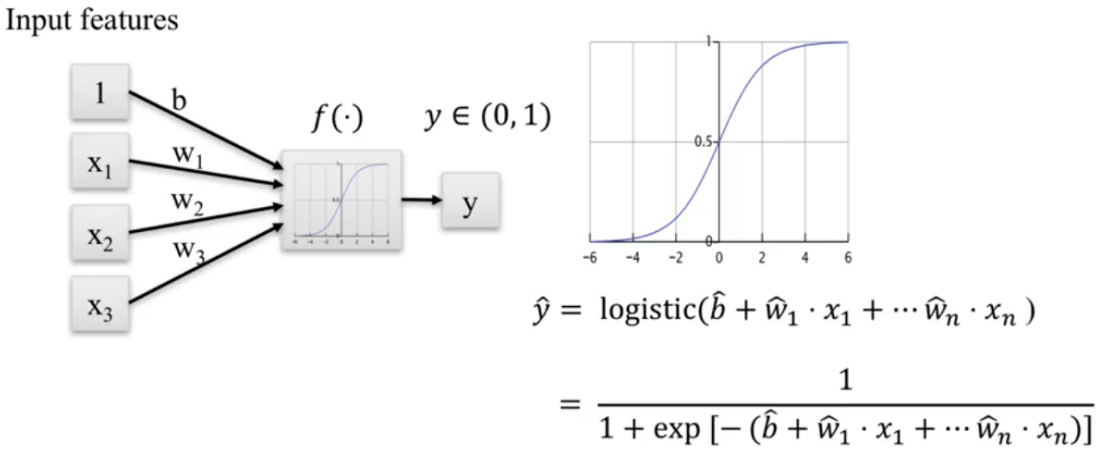


Figura N.º 1. Regresión logística.

La función **logística transforma** los valores de *input* a un valor de *output* entre 0 y 1, interpretado como la probabilidad de la instancia de pertenecer a la clase positiva dado los valores de los *feature* de entrada (x_0, x_1, x_2, ..., x_n). En otras palabras, la **regresión logística** genera una frontera de decisión lineal entre las instancias para poder clasificarlas en positivas y negativas. La siguiente figura ilustra ese concepto.

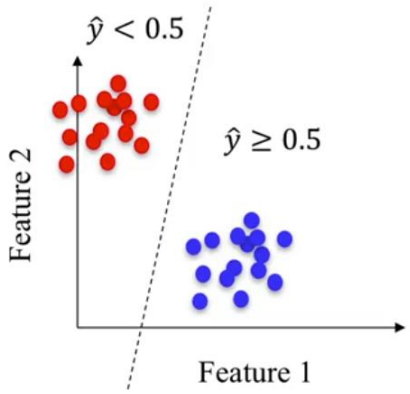


Figura N.º 2. Regresión logística para clasificación binaria.

Regularización

Al igual que *Lasso* y *Rigde regression,* se puede aplicar regularización en los coeficientes del modelo.

- Regularización L2 está habilitada por defecto en la implementación de la regresión logística en Python.
- El parámetro C controla la cantidad de regularización (por defecto es 1)
- Al igual que en el caso de la **regulación** para la regresión lineal, para la **regresión logística** es importante normalizar todos los *features* para que esté en la misma escala de valores.

Regresión logística aplicada en investigación
[https://repositorio.unprg.edu.pe/handle/20.500.12893/8623]

Clasificadores lineales

Los clasificadores lineales son un pilar fundamental en el aprendizaje automático (*machine learning*), utilizados principalmente para tareas de clasificación supervisada. Estos modelos se caracterizan por trazar una **frontera lineal para separar las clases en el espacio de características**, lo que permite asignar a una nueva instancia a una clase específica en función de sus características (Bishop, 2006). A continuación, exploraremos los conceptos básicos, los tipos más comunes y sus aplicaciones.

Un clasificador lineal toma un conjunto de características de entrada $x = (x1 + x2, \ldots, xn)$ y las combina linealmente para predecir una etiqueta de clase. Matemáticamente, un clasificador lineal puede representarse como:

$$f(x) = w1x1 + w2x2 + \ldots + wnxn + b$$

donde w son los pesos asociados a cada característica, b es el sesgo, y $f(x)$ es el valor que se utiliza para predecir la clase. Este valor es luego utilizado para clasificar la instancia según un umbral predeterminado (Hastie, Tibshirani & Friedman, 2009).

Para ilustrar el funcionamiento de los clasificadores lineales, en el caso de la clasificación binaria se toma el resultado de la función lineal y se aplica la función *sign,* la cual genera dos valores posibles: 1 si el resultado de la función lineal es positivo y –1 si el resultado de la función lineal es negativo. En la siguiente figura se ilustra el resultado de la función para las dos instancias.

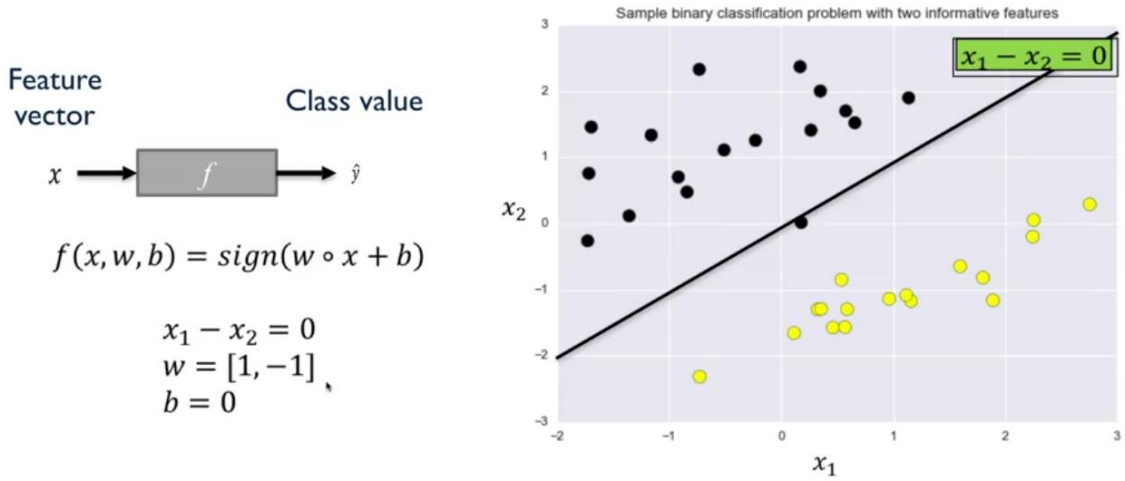


Figura N.º 3. Clasificadores lineales.

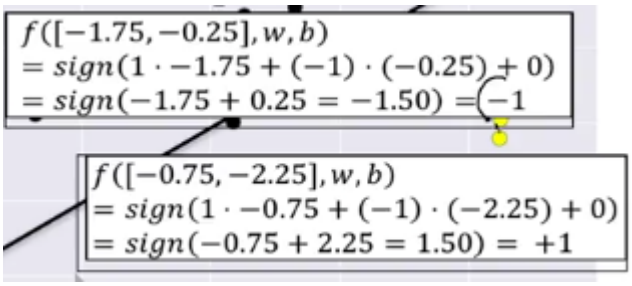


Figura N.º 4. Clasificadores lineales aplicado a dos instancias de ejemplo.

Margen de clasificación

Se define como el máximo espacio de *decision boundary* (frontera de clasificación), que se puede incrementar antes de clasificar una nueva instancia.

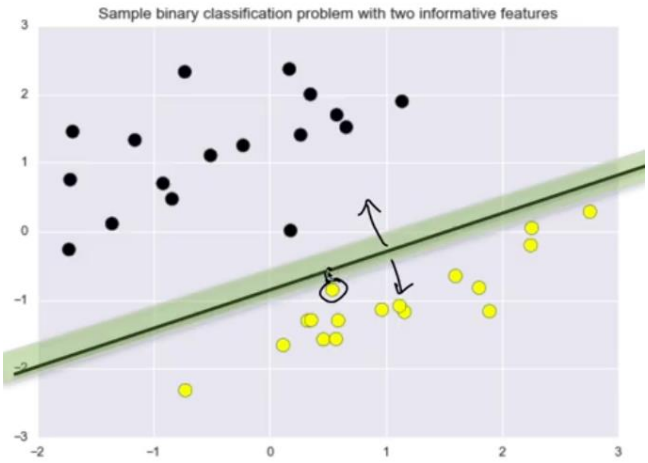


Figura N.º 5. Margen de clasificación.

Máquinas de vectores de soporte lineales (SVM)

Las máquinas de vectores de soporte (SVM, por sus siglas en inglés) son otro tipo de clasificador lineal. Su objetivo es **encontrar el hiperplano que maximice el margen entre las clases en el espacio de características**. Un SVM lineal **busca el mejor hiperplano** que **separa** las **clases** con la mayor **distancia posible entre los puntos más cercanos de cada clase**, llamados vectores de soporte (Cortes & Vapnik, 1995).

Regularización:

La fuerza de la regularización está determinada por el **parámetro C.**

- **Valores altos de C** representa **menor regularización.**
  - Esto hace que el modelo se entrene lo mejor posible con los datos de entrenamiento.
  - Todas las instancias son importantes para clasificar.
- **Valores bajos de C** implican **mayor regularización.**
  - Más tolerancia a errores en cada una de las instancias.

Ventajas y desventajas de los modelos lineales

Ventaja	**Desventaja**
Simple y fácil de entrenar	Otros modelos tienden a generalizar mejor con pocas dimensiones (*features*)
Predicción rápida	En temas de clasificación los datos podrían no ser linealmente separables
Buen escalamiento para *datasets* grandes	
Funciona bien con *sparse data*	
Modelo fácil de explicar	

Clasificadores multiclase

La clasificación multiclase se refiere a la asignación de una instancia a una de varias clases posibles. Para extender los modelos lineales a la clasificación multiclase, se utilizan técnicas como 'One-vs-All' y 'One-vs-One'. En el enfoque '**One-vs-All'** (también conocido como 'uno contra todos'), se entrena un clasificador binario para cada clase, y la predicción final se obtiene seleccionando el clasificador con la mayor salida (Hastie, Tibshirani, & Friedman, 2009). En el enfoque '**One-vs-One'**, se entrenan clasificadores binarios para cada par de clases, y la predicción se basa en el voto mayoritario entre ellos (Bishop, 2006).

Otra alternativa es la **regresión logística multiclase** mediante la función **Softmax**, que convierte las salidas lineales en probabilidades, permitiendo que cada clase tenga una probabilidad asociada. Esta técnica es útil cuando se busca una interpretación probabilística directa de las predicciones (Hastie et al., 2009). Los modelos lineales para clasificación multiclase se implementan fácilmente en bibliotecas como **Scikit-learn** (Pedregosa et al., 2011).

La siguiente ilustración muestra un ejemplo de clasificación multiclase usando *datasets* de frutas.

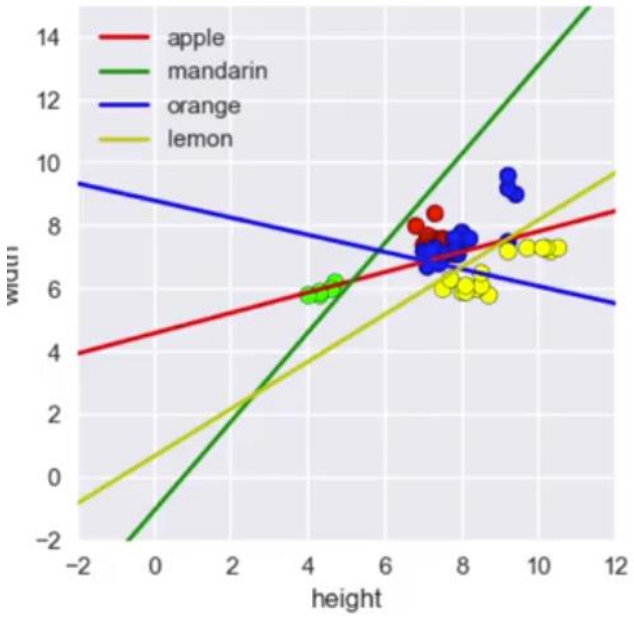


Figura N.º 6. Clasificación multiclase con modelos lineales.

Kernelized Support Vector Machine (SVM)

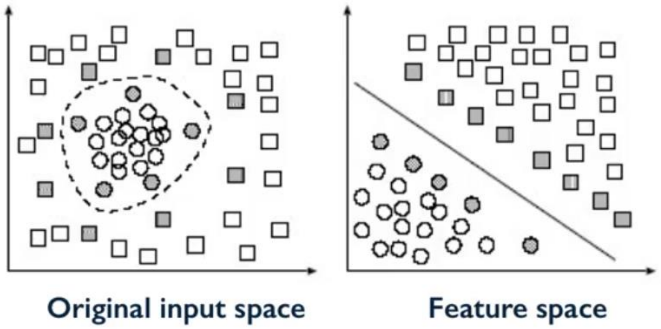
El **Support Vector Machine** (SVM) es un algoritmo de aprendizaje supervisado utilizado principalmente para clasificación y regresión. Su objetivo principal es encontrar un **hiperplano óptimo** que separe los datos en diferentes clases con el mayor margen posible (Cortes & Vapnik,

1995). El margen se define como la distancia entre el hiperplano y los puntos de datos más cercanos, denominados vectores de soporte. Este enfoque garantiza una mejor generalización del modelo a nuevos datos (Schölkopf & Smola, 2002).

Conceptos fundamentales

El SVM funciona mediante la **identificación** de un **hiperplano** en un espacio de **características de dimensión elevada**. Si los datos son linealmente separables, el SVM busca maximizar el margen entre las clases. Sin embargo, cuando los datos no son linealmente separables, se emplean funciones de kernel para mapear los datos a un espacio de mayor dimensión donde sean separables (Cristianini & Shawe-Taylor, 2000). Algunos de los kernels más utilizados incluyen el kernel lineal, el polinómico, radial (RBF) y el sigmoide. La siguiente ilustración explica el funcionamiento de la función de kernel.

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\ [-\gamma \cdot \|\boldsymbol{x} - \boldsymbol{x}'\|^2]$$



A kernel is a similarity measure (modified dot product) between data points

Figura N.º 7. Radial Basic Function Kernel.

Ventajas y desventajas

Entre las ventajas del SVM se incluyen su capacidad para manejar datos de alta dimensión y su eficacia en conjuntos de datos con pocas muestras (Burges, 1998). No obstante, su desventaja radica en el alto costo computacional cuando se trabaja con grandes volúmenes de datos y la necesidad de ajustar correctamente los hiperparámetros, como el tipo de kernel y el parámetro de regularización (Hastie, Tibshirani, & Friedman, 2009).

Support Vector Machine aplicado en investigaciones [https://ieeexplore.ieee.org/abstract/document/6914200]

Parámetros importantes:

***Model complexity*:**

- **Kernel: t**ipo de función que usa por defecto 'rbf', que significa *radial basis function.*
- **C:** parámetro de regularización

- **Kernel *parameters:*** *gamma (γ): ancho del kernel RBF.*

Ventajas y desventajas de *Kernelized Support Vector Machine*

Ventaja	**Desventaja**
Puede 'performar' bien en muchos *datasets.*	El tiempo de procesamiento incrementa cuando el *dataset* de entrenamiento supera las 50 000 instancias.
Versátil: se pueden usar diferentes versiones de *kernel* o se pueden definir *kernels* personalizados.	Necesita una cuidadosa normalización de las variables y tuneo de hiperparámetros.
Funciona bien para *dataset* con alta o baja dimensional.	Su capacidad descriptiva es más alta que otros algoritmos.

Cross validation

La **validación cruzada** (*cross validation*) es una técnica estadística utilizada en el aprendizaje automático para evaluar el rendimiento de un modelo y **reducir el riesgo de sobreajuste**. Consiste en **dividir** el conjunto de datos en múltiples subconjuntos o particiones, donde el modelo es entrenado en una parte de los datos y validado en otra, asegurando una evaluación más robusta de su capacidad predictiva (Kohavi, 1995).

Tipos de validación cruzada

Existen diversos tipos de validación cruzada, siendo los más comunes la validación cruzada ***k-fold, leave-one-out*** (LOO) y **validación cruzada estratificada**. En la **validación *k-fold,*** *el conjunto de datos se divide en k partes, utilizando k-1 partes para el entrenamiento y la restante para la prueba*. Este proceso se **repite k veces**, permitiendo una evaluación más estable del modelo (Stone, 1974). **La técnica *leave-one-out* es un caso especial de *k-fold,* donde k es igual al número de observaciones**, lo que puede proporcionar una evaluación precisa pero con alto costo computacional (Geisser, 1975). La siguiente figura ilustra la validación cruzada utilizando 5 folds.

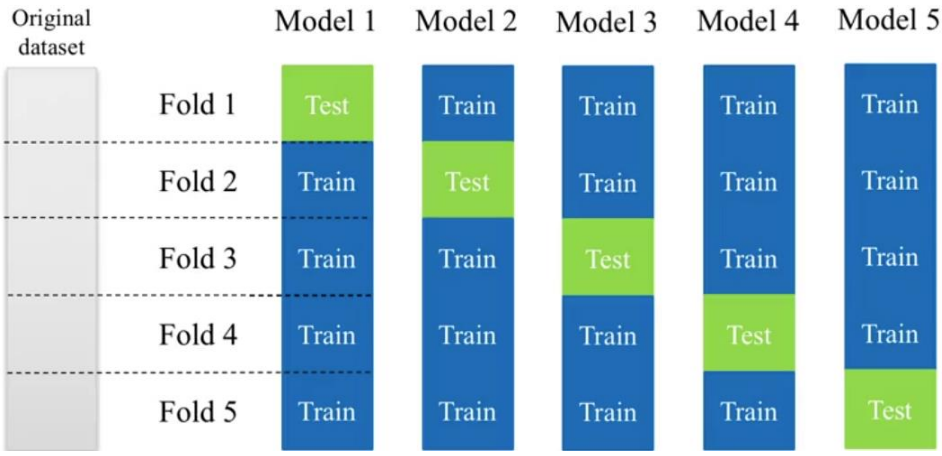


Figura N.º 8. Validación cruzada 5 *folds.*

Si el *dataset* está ordenado en función de, por ejemplo la variable *target*, esto podría generar problemas en el momento de generar los *k folds*, dado que tomaría los primeros k elementos (o proporciones) para validar, pudiendo tener un *dataset* de testeo totalmente desbalanceado. La mejor forma de lidiar con este problema es usar la validación cruzada estratificada, la cual permite preservar la proporción de cada una de las clases.

Importancia de la validación cruzada

La validación cruzada es fundamental para evitar problemas de sobreajuste y seleccionar modelos con mejor capacidad de generalización. Se ha demostrado que al utilizar validación cruzada, especialmente en *datasets* pequeños, se pueden obtener estimaciones más precisas del error del modelo en datos no vistos (Bengio & Grandvalet, 2004). Además, permite comparar diferentes modelos y seleccionar aquel con mejor desempeño antes de su implementación en problemas reales.

Aplicaciones de la validación cruzada

La validación cruzada es ampliamente utilizada en diversas aplicaciones del aprendizaje automático, incluyendo la clasificación de textos, análisis de imágenes y predicción de series temporales. En el análisis de imágenes, por ejemplo, la validación cruzada ayuda a seleccionar modelos de reconocimiento de patrones con alta precisión (Varma & Zisserman, 2005). **En problemas de series temporales, se utilizan variantes como la validación en ventana deslizante (*rolling window*) para evaluar modelos predictivos** (Bergmeir & Benítez, 2012).

Árboles de decisión

Los **árboles de decisión** son uno de los algoritmos más **utilizados** en *machine learning* debido a su **simplicidad**, **interpretabilidad** y **capacidad** para manejar tanto **problemas** de **clasificación** como de **regresión**. Un árbol de decisión es un modelo predictivo que toma decisiones **basadas en características de los datos**, representadas en **nodos de un árbol jerárquico**. Cada nodo interno realiza una prueba sobre una característica, y cada hoja del árbol representa una etiqueta de clase (para clasificación) o un valor (para regresión). Esta estructura permite representar relaciones complejas de manera comprensible (Breiman et al., 1986).

El siguiente gráfico brinda información sobre las ramas del árbol de decisión; la lista *value* indica el número de ejemplos de cada clase que quedaron en la hoja durante el proceso de entrenamiento. Dado que el *dataset* iris tiene 3 clases, hay 3 elementos en la lista. Esta rama tiene 37 ejemplos de setosa, 0 de versicolor y 0 de virginica.

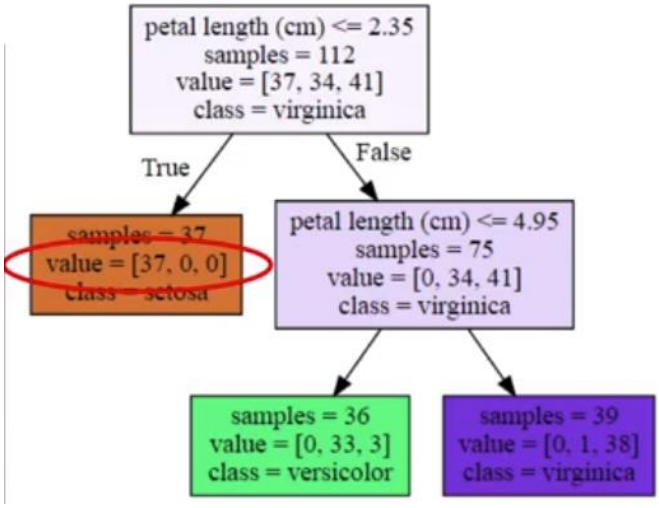


Figura N.º 9. Informatividad de las separaciones del árbol de decisión.

Funcionamiento de los árboles de decisión

El funcionamiento de los árboles de decisión se basa en un proceso de **particionamiento recursivo del conjunto de datos**. Este proceso busca **dividir** el conjunto de datos de manera que las **particiones** resultantes sean lo más **homogéneas** posible, en términos de la variable objetivo. Para lograrlo, se emplean medidas como la ***entropía*** y el ***coeficiente o índice de Gini*** para elegir el mejor atributo en cada nodo (Quinlan, 1986). A continuación, se presenta un ejemplo básico de implementación de un árbol de decisión utilizando la biblioteca *scikit-learn* de Python, para un problema de clasificación con el conjunto de datos *Iris*:

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Cargar el conjunto de datos Iris
iris = load_iris()
X = iris.data
y = iris.target

# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Crear un modelo de árbol de decisión
clf = DecisionTreeClassifier(random_state=42)

# Entrenar el modelo
clf.fit(X_train, y_train)

# Hacer predicciones
y_pred = clf.predict(X_test)

# Evaluar el modelo
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

Figura N.º 10. Implementación de árbol de decisión con sklearn.

En este ejemplo, un árbol de decisión se entrena con el conjunto de datos *Iris*, y se **evalúa** el **rendimiento del modelo** mediante la **métrica de precisión**. El modelo **clasifica** las **flores** del conjunto de datos en **tres categorías basadas en sus características morfológicas.**

Ventajes y desventajas de los árboles de decisión

Los **árboles de decisión** tienen varias ventajas, entre las cuales se destaca su **capacidad** para ser **interpretados fácilmente por los humanos**. Esto es crucial en aplicaciones donde la **explicabilidad** es importante, como en el sector financiero o en la atención médica (Breiman et al., 1986). Sin embargo, una de sus principales **desventajas** es que pueden **sobreajustarse** si no se controlan adecuadamente. El sobreajuste ocurre cuando el árbol se adapta demasiado a los datos de entrenamiento, perdiendo capacidad de generalización para datos nuevos (Breiman, 2001). Para mitigar este problema, es común emplear **técnicas de poda**, que eliminan ramas del árbol que no aportan valor significativo.

Técnicas de poda y ensamble de modelos

Una técnica importante para mejorar los árboles de decisión es la **poda**, que consiste en **eliminar nodos innecesarios o ramas que no contribuyen al rendimiento del modelo**. La **poda** ayuda a **reducir el sobreajuste** y **mejora la capacidad de generalización** del modelo. A continuación, se presenta un ejemplo de implementación de poda utilizando el parámetro ***max_depth*** de *scikit-learn* para limitar la profundidad del árbol y **evitar** el **sobreajuste**:

```
# Crear un modelo de árbol de decisión con poda (limitando la profundidad)
clf_poda = DecisionTreeClassifier(max_depth=3, random_state=42)

# Entrenar el modelo
clf_poda.fit(X_train, y_train)

# Hacer predicciones
y_pred_poda = clf_poda.predict(X_test)

# Evaluar el modelo
print(f"Accuracy con poda: {accuracy_score(y_test, y_pred_poda)}")
```

Figura N.º 11. Poda de árbol de decisión con sklearn.

En este ejemplo, se limita la **profundidad máxima del árbol a 3**, lo que evita que el modelo crezca excesivamente y se sobreajuste a los datos de entrenamiento.

Otras técnicas avanzadas que pueden mejorar el rendimiento de los árboles de decisión incluyen ***Random Forest*** **y** ***Gradient Boosting***. Estos enfoques construyen **múltiples árboles de decisión** y combinan sus resultados para mejorar la precisión general del modelo. Los ***Random Forest*** funcionan entrenando **muchos árboles sobre subconjuntos aleatorios de los datos**, lo que **reduce** la **varianza** y **mejora** la **robustez** del modelo. Por otro lado, ***Gradient Boosting*** **construye árboles**

secuenciales, donde **cada árbol corrige los errores del anterior**, lo que puede llevar a un rendimiento significativamente superior en ciertos contextos (Breiman, 2001).

Ventajas	**Desventajas**
Fácil de visualizar e implementar	Puede presentar problemas de *overfitting*.
No se necesita normalización de datos	Usualmente se necesita ensamblar árboles para obtener un mejor *performance*.

One Hot Encoding

En el ámbito del aprendizaje automático, uno de los retos más comunes al trabajar con datos categóricos es cómo representar dichos datos de manera adecuada para los algoritmos. Una de las técnicas más utilizadas para este fin es el ***One Hot Encoding*** (OHE), que convierte variables categóricas en una forma binaria que los modelos de *machine learning* pueden interpretar sin perder información relevante. Esta técnica es fundamental para preparar datos antes de ser alimentados a modelos como regresión logística, árboles de decisión o redes neuronales (Harris, 2015).

Definición y funcionamiento

El ***One Hot Encoding*** consiste en transformar cada categoría de una variable en una columna binaria; es decir, por cada valor único de la variable categórica, se crea una nueva columna, y en cada fila de datos, la columna correspondiente a la categoría de la muestra se marca con un valor 1, mientras que las demás columnas reciben el valor 0 (Chollet, 2018). Este proceso asegura que no haya una relación ordinal implícita entre las categorías y permite que el modelo interprete cada categoría como una entidad independiente.

Por ejemplo, supongamos que tenemos una variable ‘color’ con tres categorías: ‘rojo’, ‘verde’ y ‘azul’. Después de aplicar *One Hot Encoding*, obtendríamos tres columnas binarias como sigue:

Color	Rojo	Verde	Azul
Rojo	1	0	0
Verde	0	1	0
Azul	0	0	1

Figura N.º 12. Ejemplo de *One Hot Encoding*.

Ventajas del *One Hot Encoding*

El principal beneficio del *One Hot Encoding* es su capacidad para manejar variables categóricas sin hacer suposiciones de orden o distancia entre las categorías (Géron, 2019). Además, es una técnica fácil de aplicar y entender. Sin embargo, el principal inconveniente surge cuando el número de categorías es elevado, lo que lleva a una expansión significativa del número de características,

lo cual puede generar un aumento en el tiempo de entrenamiento y en la complejidad del modelo (Géron, 2019).

Implementación en Python usando *scikit-learn*

A continuación, se presenta un ejemplo práctico de cómo implementar *One Hot Encoding* utilizando la librería ***scikit-learn*** en Python. En este caso, usaremos la clase *One Hot Encoding* para transformar una columna categórica.

```python
# Importamos las librerías necesarias
import numpy as np
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# Creamos un DataFrame de ejemplo
data = {'Color': ['Rojo', 'Verde', 'Azul', 'Rojo', 'Verde']}
df = pd.DataFrame(data)

# Inicializamos el OneHotEncoder
encoder = OneHotEncoder(sparse=False)

# Ajustamos y transformamos los datos categóricos
encoded_data = encoder.fit_transform(df[['Color']])

# Mostramos el resultado
encoded_df = pd.DataFrame(encoded_data, columns=encoder.categories_[0])
print(encoded_df)
```

Figura N.º 13. Implementación de *One Hot Encoding* con *sklearn*.

Este código transforma la columna ‘color’ en tres columnas binarias: ‘rojo’, ‘verde’ y ‘azul’, representando la presencia de cada color en las filas del *dataset*.

Referencias citadas en la Clase 4

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2), 215-232.

Cortes, C., & Vapnik, V. (1995). *Support-vector networks*. *Machine learning*, 20(3), 273-297.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R*. Springer.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11), 2278-2324.

Pang, B., Lee, L., & Vaithyanathan, S. (2002*). Thumbs up? Sentiment classification using machine learning techniques*. Proceedings of the ACL-02 conference on Empirical methods in natural language processing, 79-86.

Sebastiani, F. (2002). *Machine learning in automated text categorization*. ACM Computing Surveys (CSUR*)*, 34(1), 1-47.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … & Duchesnay, E. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research*, 12*, 2825-2830.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1986). *Classification and regression trees*. Wadsworth & Brooks/Cole Advanced Books & Software.

Quinlan, J. R. (1986). *Induction of decision trees*. Machine Learning, 1(1), 81-106.

Chollet, F. (2018). *Deep Learning with Python*. Manning Publications.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.

Harris, C. R. (2015). *Python for Data Analysis*. O'Reilly Media.

Definición de los términos citados en la Clase 4

Clasificadores lineales. Los clasificadores lineales son en esencia adaptaciones de la regresión lineal, dado que aplican una función específica a la salida de tal forma que el resultado sea una **categoría** y no un valor numérico continuo.

Poda. La **poda de un árbol de decisión** es el proceso de reducir su tamaño eliminando ramas que no aportan significativamente al rendimiento del modelo, con el fin de evitar el sobreajuste. Se puede realizar de manera ***pre-pruning***, limitando el crecimiento del árbol, o ***post-pruning***, recortando ramas después de su construcción. Esto mejora la generalización y reduce la complejidad del modelo.

PUCE Virtual
EDUCACIÓN FLEXIBLE

La excelencia no se improvisa

síguenos

MDASC - 12 - 241