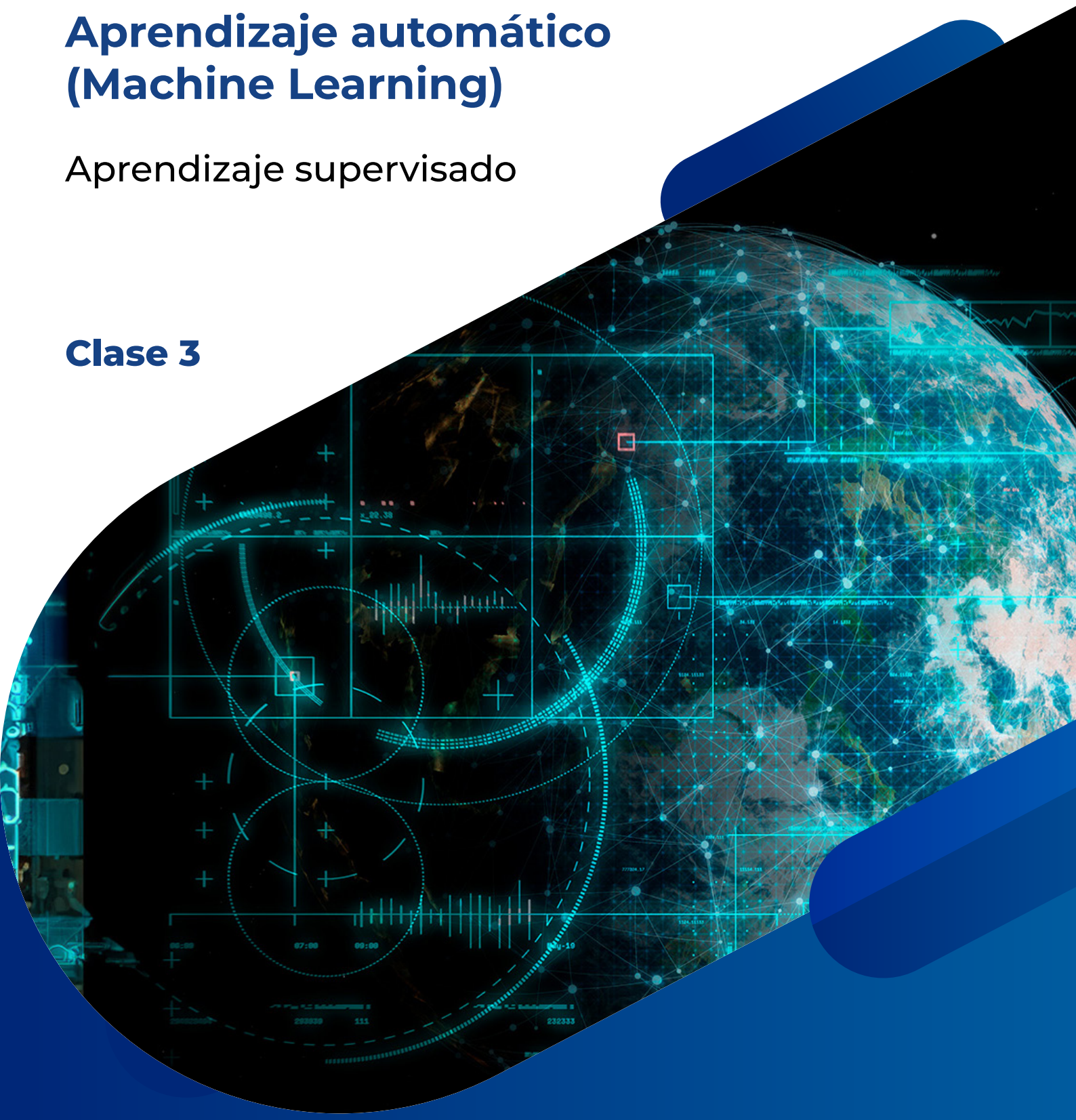
PUCE Virtual
EDUCACIÓN FLEXIBLE

Aprendizaje automático (Machine Learning)

Aprendizaje supervisado



Clase 3

MAESTRÍA EN
SISTEMAS DE INFORMACIÓN
Mención Data Science

La excelencia no se improvisa

1. INTRODUCCIÓN DE LA CLASE

El **aprendizaje supervisado** es un enfoque fundamental dentro del campo del **aprendizaje automático**, en el cual se entrena un modelo utilizando un conjunto de datos etiquetados. Estos datos consisten en pares de entradas y salidas, donde las salidas son conocidas de antemano, lo que permite al modelo aprender una relación o función que mapea las entradas a sus correspondientes salidas. El **objetivo** es que el modelo, una vez entrenado, sea capaz de **hacer predicciones** o inferencias sobre **nuevos datos no etiquetados**. El aprendizaje supervisado se aplica en tareas como **clasificación**, donde el modelo asigna categorías a las entradas, y **regresión**, donde predice valores continuos (James, Witten, Hastie, & Tibshirani, 2013).

En la práctica, el aprendizaje supervisado implica la selección de un modelo adecuado y un proceso de entrenamiento, donde el modelo ajusta sus parámetros para minimizar el error en las predicciones realizadas sobre el conjunto de datos de entrenamiento. Existen diversos algoritmos supervisados, tales como la **regresión lineal**, **máquinas** de **soporte vectorial** y **redes neuronales**, que son capaces de abordar distintos tipos de problemas. Sin embargo, uno de los desafíos más importantes en este tipo de aprendizaje es la capacidad del modelo para generalizar, es decir, hacer predicciones precisas no solo sobre los datos de entrenamiento, sino también sobre datos nuevos y no vistos (Bishop, 2006). Esta clase se centrará en los siguientes aspectos:

- Entender el **funcionamiento** de varios algoritmos de aprendizaje supervisado estimando sus **parámetros** desde los datos para generar nuevas predicciones.
- Entender las **fortalezas** y **debilidades** de algunos métodos de aprendizaje supervisado.
- Aprender cómo aplicar algoritmos de aprendizaje supervisado en Python usando ***scikit-learn*.**
- Aprender principios básicos del aprendizaje supervisado como por ejemplo el ***overfitting*** y cómo evitarlo.

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

Clase 3. Aprendizaje supervisado

Introducción al aprendizaje supervisado

El aprendizaje supervisado es un paradigma fundamental dentro del campo del aprendizaje automático, utilizado ampliamente para **resolver problemas de predicción** y **clasificación**. En este enfoque, se entrena un modelo utilizando un conjunto de datos etiquetados, lo que significa que cada instancia de entrada está asociada con una salida conocida. El objetivo es que el **modelo aprenda una función o relación que permita predecir la salida para nuevos datos no etiquetados.** Este tipo de aprendizaje es especialmente valioso en tareas como la **clasificación**, donde se asignan categorías a las entradas, o la **regresión**, donde se predicen valores continuos (Hastie, Tibshirani, & Friedman, 2009). El aprendizaje supervisado, por lo tanto, depende de la calidad y de la cantidad de los datos etiquetados, siendo clave para el rendimiento del modelo.

A medida que los datos de entrenamiento se procesan, el modelo ajusta sus parámetros internos para **minimizar el error de predicción,** utilizando técnicas **matemáticas** y **estadísticas**. Algunos de los algoritmos más conocidos en aprendizaje supervisado incluyen la r**egresión lineal, máquinas de soporte vectorial y redes neuronales**, cada uno adecuado para distintos tipos de tareas y características de los datos. A pesar de su efectividad, el aprendizaje supervisado enfrenta desafíos como el ***overfitting*** y el ***underfitting***, situaciones que pueden afectar la capacidad de generalización del modelo y, por ende, su rendimiento en datos nuevos. Estos desafíos resaltan la importancia de una correcta elección del modelo y de las técnicas de regularización para asegurar un aprendizaje efectivo y robusto (James, Witten, Hastie, & Tibshirani, 2013).

Términos importantes:

- **Representación de *features* (variables):**
  Convertir un objeto en números de forma que la computadora pueda entender. En la siguiente tabla se puede ver que las ***features*** son: *mass, width, height, color_score,* mismas que caracterizan a cada uno de los objetos o instancias.

- **Data instances/samples/examples (X):**
  Es la agrupación de las features por cada una de las instancias o filas del *dataset*. Normalmente se representa con la letra X

- **Target value (y):**
  Es la variable que se debe predecir, también conocida como *labels;* normalmente se la representa con la letra *y.*

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89
10	1	apple	braeburn	166	6.9	7.3	0.93

Figura N.º 1. *Fruits dataset*.

- ***Datasets* de entrenamiento y de testeo:**
  El *dataset* de entrenamiento es el conjunto de datos con los cuales se va a crear el modelo y que permite estimar los parámetros del mismo para que después pueda ser usado con datos nuevos. Por otro lado, el *dataset* de testeo se usa para esencialmente para medir la capacidad predictiva del modelo. Normalmente, de los datos disponibles se suele usar un 75 % de ellos para entrenar y el 25 % restante se usa para testear el modelo.

- ***Fitting, traning***
  Es el entrenamiento del modelo en sí mismo. Mientras que el entrenamiento es el proceso de estimación de los parámetros del modelo en función de los *dataset* de entrenamiento.

- **Método de evaluación**
  Métrica usada para cuantificar la capacidad predictiva del modelo *(accuracy, roc auc, presicion, recall, etc.).*

```
%matplotlib notebook
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

fruits = pd.read_table('fruit_data_with_colors.txt')

X = fruits[['height', 'width', 'mass', 'color_score']]
y = fruits['fruit_label']

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train, y_train)
print("Accuracy of K-NN classifier on test set: ", knn.score(X_test, y_test))

example_fruit = [[5.5, 2.2, 10, 0.70]]
print("Predicted fruit type for ", example_fruit, " is ", knn.predict(example_fruit))
```

Figura N.º 2. Entrenamiento de un modelo.

Clasificación y regresión

- En ambos casos, tanto para modelos de clasificación como para modelos de regresión es necesario contar con **instancias** de entrenamiento con su variable ***target*** correspondiente.
- Para los modelos de **clasificación**, la variable *target* es una variable discreta; y pueden existir dos tipos de modelos de clasificación:
    - **Modelos binarios**: valores entre 0 (clase negativa) o 1 (clase positiva). Por ejemplo: detectar transacciones fraudulentas en tarjetas de crédito. En la siguiente ilustración se muestra un ejemplo de detección de una instancia de transacción en tarjetas de crédito con su respectivo *target* (positivo o negativo).

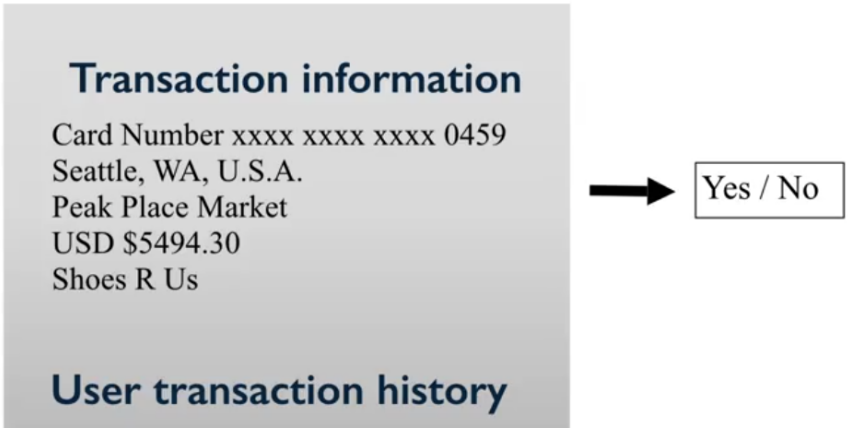


Figura N.º 3. *Binary classification:* detección de fraude en tarjetas de crédito.

- **Modelos multiclase:** es cuando la variable *target* es categórica y puede contener más de dos valores.
  Por ejemplo: ***Labelling*** el tipo de fruta en base a sus características físicas.

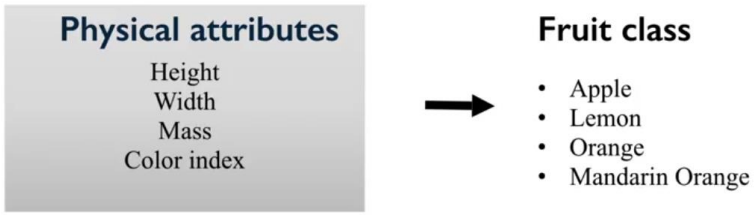


Figura N.º 4. *Multiclass classification*: reconocimiento de frutas.

- Para modelos de **regresión,** la variable *target* es una variable continua. Por ejemplo: predecir el precio de venta de una casa en base a sus atributos.
- La selección del modelo de aprendizaje supervisado depende del tipo de dato que tenga la variable *target.*
- Muchos de los métodos de aprendizaje supervisado tienen implementaciones para ambos: clasificación y regresión.

Generalización *overfitting* y *underfitting*

La **generalización** se refiere a la habilidad que tienen los algoritmos de generar predicciones precisas para datos nuevos que el modelo no ha visto. Partiendo de las siguientes suposiciones:

- El conjunto de datos de entrenamiento o ***test set*** tendrá las mismas características que el ***traning set*** actual.
- Se esperaría que los modelos que tienen un buen performance en los datos de entrenamiento tengan también un buen performance en los datos de testeo.
- Los modelos que son muy **complejos,** dada la cantidad de datos de entrenamiento disponibles, se dice que tienen a caer en *overfitting* y no llegan a generalizar correctamente los datos nuevos.
- Por otro lado, los modelos que son muy **simples**, que ni siquiera tiene un buen performance en los datos de entrenamiento se dice que caen en *underfitting* y por supuesto no generalizan correctamente.

Overfitting

El ***overfitting*** o sobreajuste es un problema en el aprendizaje automático, donde un modelo aprende demasiado bien los datos de entrenamiento, incluyendo ruido y patrones irrelevantes, lo que afecta su capacidad de **generalización** (Bishop, 2006). Esto significa que el modelo tiene un desempeño excelente en los datos de entrenamiento, pero falla al predecir con nuevos datos.

Ejemplos prácticos del *overfitting*

- **Reconocimiento de imágenes.** Un modelo de clasificación de imágenes entrenado con un conjunto de datos muy pequeño podría memorizar características específicas de cada imagen en lugar de aprender patrones generales. Por ejemplo, si un modelo de **detección**

de gatos solo ha sido entrenado con imágenes de gatos en interiores, podría fallar al reconocer gatos en entornos exteriores.

- **Predicción de ventas.** Supongamos que se entrena un modelo para predecir las ventas de una tienda basándose en datos históricos. Si el modelo aprende patrones específicos de ciertos días festivos sin generalizar, podría predecir incorrectamente las ventas en días similares que no fueron parte del conjunto de entrenamiento (Goodfellow, Bengio & Courville, 2016).
- **Análisis financiero.** En el mercado de valores, un modelo de predicción de precios de acciones podría ajustarse demasiado a tendencias pasadas sin captar fluctuaciones reales del mercado. Esto resultaría en predicciones poco confiables cuando surjan nuevos factores económicos (Ng, 2018).

Prevención del *overfitting*

Para evitar el sobreajuste, se pueden aplicar varias técnicas:

- **Regularización** (como L1 y L2), para penalizar modelos excesivamente complejos.
- **Aumento de datos** para diversificar el conjunto de entrenamiento.
- **Validación cruzada** para evaluar el modelo con distintos subconjuntos de datos.
- **Reducción de complejidad del modelo** para evitar que se ajuste a detalles irrelevantes.

Underfitting

El ***underfitting*** o subajuste es un problema en el aprendizaje automático, donde un modelo es demasiado simple para capturar patrones significativos en los datos, lo que resulta en un bajo desempeño tanto en los datos de entrenamiento como en los datos de prueba (Bishop, 2006). Esto significa que el modelo no logra aprender lo suficiente de los datos, lo que lleva a predicciones inexactas.

Ejemplos prácticos del *underfitting*

- **Reconocimiento de imágenes.** Un modelo de clasificación de imágenes que utiliza una regresión lineal para distinguir entre perros y gatos probablemente no capture la complejidad de las diferencias visuales entre ambas especies, lo que resultaría en un desempeño deficiente.
- **Predicción de ventas.** Si un modelo intenta predecir las ventas de una tienda utilizando solo el promedio de ventas diarias sin considerar factores como días festivos o tendencias de mercado, su capacidad predictiva será muy baja (Goodfellow, Bengio & Courville, 2016).
- **Análisis financiero.** Un modelo de predicción de precios de acciones que solo usa un promedio móvil de precios históricos sin incorporar variables como tasas de interés o noticias económicas probablemente no prediga con precisión las fluctuaciones del mercado (Ng, 2018).

Prevención del *underfitting*

Para evitar el subajuste, se pueden aplicar varias estrategias:

- **Aumento de la complejidad del modelo**, usando algoritmos más sofisticados como redes neuronales profundas.
- **Incorporación de más variables** relevantes en el modelo.
- **Mayor entrenamiento** para permitir que el modelo aprenda mejor los patrones de los datos.
- **Reducción del sesgo** mediante algoritmos menos restrictivos y mayor iteración en el proceso de ajuste.

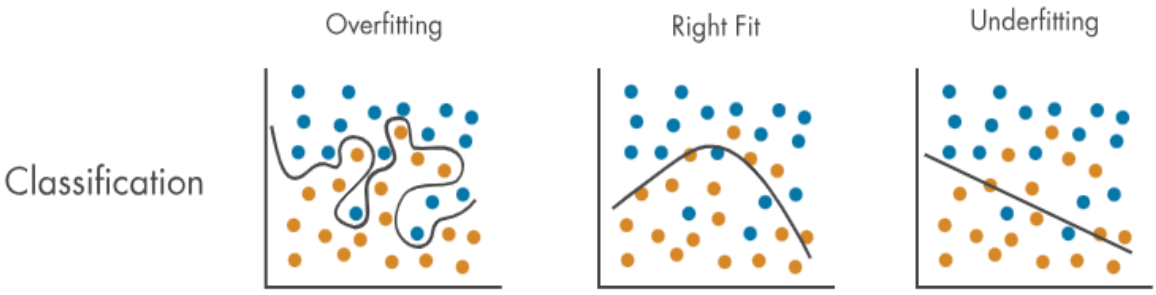


Figura N.º 5. *Overfitting* y *underfitting* en problemas de clasificación.
https://la.mathworks.com/discovery/overfitting.html

En la Figura N.º 5 se puede observar gráficamente el problema del *overfitting,* dado que el modelo se ajusta exactamente a los modelos de **entrenamiento,** logrando ser 100 % preciso en detectar sus ***labels***. Por otro lado, en la esquina derecha se puede observar que el modelo no es lo suficientemente preciso (es demasiado simple), ya que clasifica mal muchos casos. Finalmente, en el gráfico del medio se puede ver una curva que separa los grupos y, si bien no clasifica correctamente todos los casos, se puede ver una buena **generalización.**

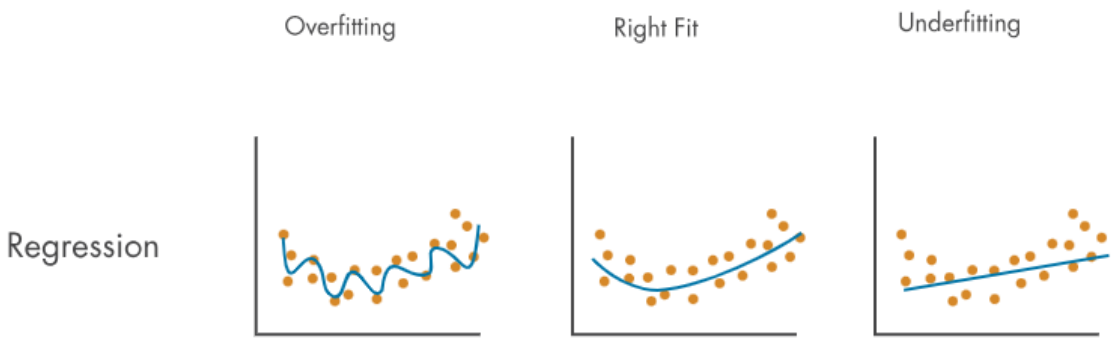


Figura N.º 6. *Overfitting* y *underfitting* en problemas de regresión.
https://la.mathworks.com/discovery/overfitting.html

En la Figura N.º 6, en el gráfico de la izquierda vemos que el modelo de regresión tiene problemas de ***overfitting,*** dado que la curva que predice los puntos se ajusta 100 % a cada uno de ellos llegando a tener una **predicción perfecta** en los datos de **entrenamiento**. Por otro lado, en el caso del gráfico del extremo derecho la predicción se traduce en una línea que está muy distante de predecir muchos de los puntos, sobre todo los que se encuentran en los extremos. Finalmente, el gráfico del medio si bien no está distante de algunos de los puntos, tiende a tener una buena generalización ya que detecta los patrones de **distribución** de los datos.

Overfitting and underfitting [https://la.mathworks.com/discovery/overfitting.html]

K–vecinos más cercanos para clasificación y regresión

K–vecinos más cercanos es un algoritmo de *machine learning* que puede ser usado tanto para clasificación como para regresión, y es uno de los algoritmos más simples de entender. Para modelos de **clasificación,** el algoritmo **k-vecinos más cercanos** funciona de la siguiente manera:

Dado un conjunto de entrenamiento X_train con labels y_train y dada una instancia de datos a ser clasificada x_test los pasos son los siguientes:

1. Buscar las **instancias más similares** (las cuales las llamaremos X_NN) a x_test que se encuentran en X_train
2. **Obtener** los ***labels*** **y_NN** para cada una de las instancias en X_NN
3. **Predecir** el ***label*** para x_test combinando los labels **y_NN** Por ejemplo: por mayoría de votos.

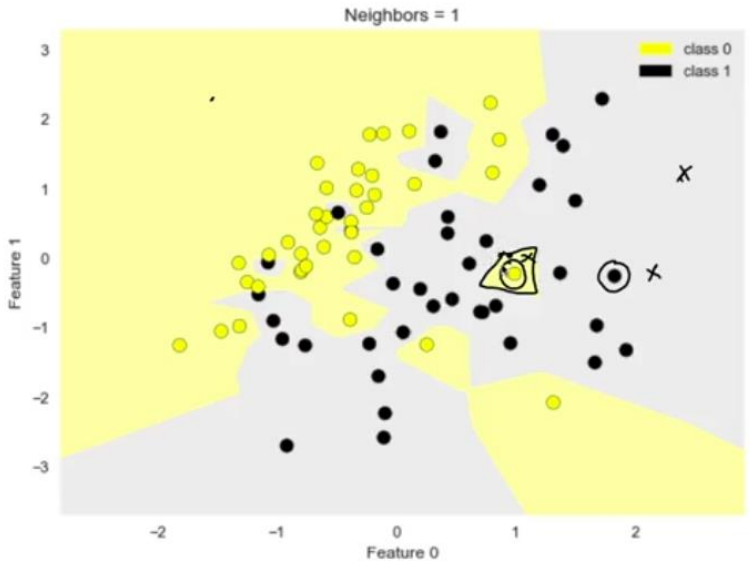


Figura N.º 7. K=1 vecinos más cercanos.

En la figura anterior se ilustra el funcionamiento del algoritmo **k–vecinos más cercanos** para clasificación con un k = 1. En este caso el modelo es demasiado complejo y cae en ***overfitting*** debido a que se ajusta demasiado bien a los datos de entrenamiento. El color de fondo (amarillo y gris) indica el *decision boundary* e indica que si la observación está en uno de esos límites pertenecerá a la clase amarilla o a la clase gris.

Para el siguiente caso siguiente con k = 11 podemos ver que el *decision boundary* es mucho más y mejor definido, y no complejiza el modelo puedo que las dos clases casi se las puede separar con una línea con una inclinación de 45 grados.

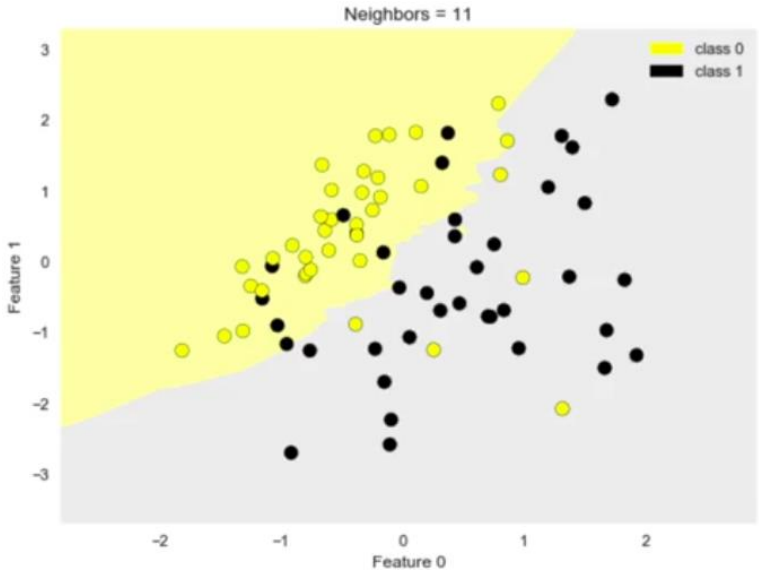


Figura N.º 8. K=11 vecinos más cercanos.

A diferencia del caso de *k vecinos más cercanos* para clasificación para modelos de regresión la predicción final de una instancia nueva será el **promedio** de los k–vecinos más cercanos. En la figura anterior, en extremo izquierdo tenemos un ***scatter plot*** de los datos de entrenamiento, es decir cada valor de instancia con su respectivo *target.* Luego al aplicar el modelo de regresión con datos nuevos podemos ver en el gráfico del medio que los triángulos son los valores predichos para cada uno de los valores de entrada. El valor de predicción final será el promedio entre los k–valores más cercanos al ***target value*** de cada instancia.

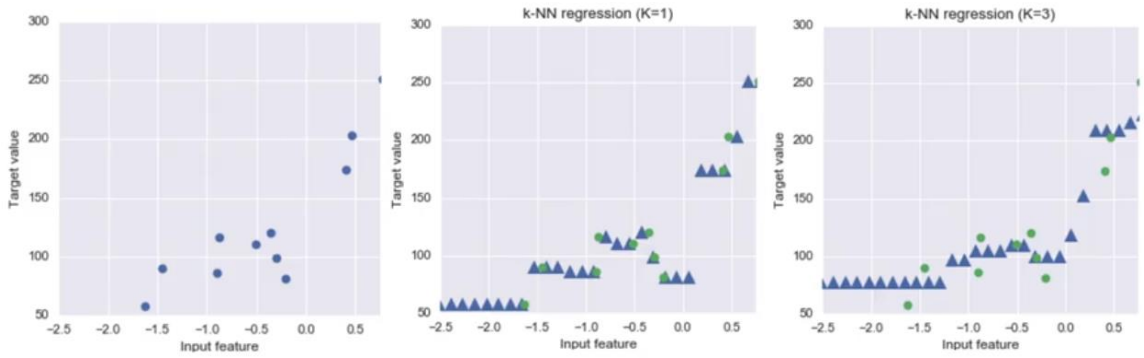


Figura N.º 9. K–vecinos más cercanos para regresión.

Una de las métricas más usadas para poder evaluar la capacidad predictiva de un modelo de regresión es el **R^2**, la cual mide cuán buena es la predicción en función del valor real.

El valor puede fluctuar entre 0 y 1, en donde el valor de 0 significa que el modelo está prediciendo el **promedio** de los *target values* de entrenamiento, y en consecuencia tiene pésima **capacidad predictiva**. Por otro lado, el valor de 1 corresponde a una predicción perfecta. El **R^2** también es conocido como coeficiente de determinación

Parámetros importantes para los modelos de clasificación y regresión usando k–vecinos más cercanos

Complejidad de los modelos:

n_neighbors: número de k–vecinos más cercanos a considerar; por defecto, en la implementación de Python se usa el valor de 5.

***Model Fitting*:**

metric: función de distancia entre los puntos de datos; por defecto, en la implementación de Python se usa la distancia de Minkowski.

Regresión lineal mínimos cuadrados

Los modelos lineales son uno de los tipos de modelos de aprendizaje supervisado más simples que hay. Un modelo lineal expresa la variable *target* en función de la suma de los pesos de las variables predictoras dada un *input instance.* La regresión lineal es un ejemplo de un modelo lineal.

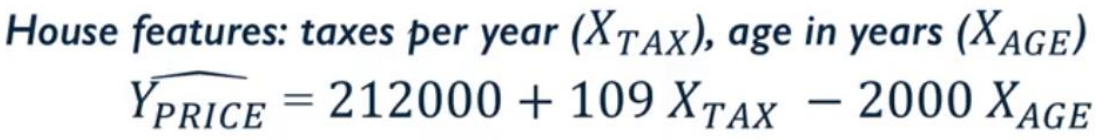


Figura N.º 10. Predicción del precio de casos.

En la figura anterior se muestra la ecuación que explica el precio de la casa en función de los impuestos por año y la edad de la casa expresada en años.

Una casa con los siguientes valores de *features* (Xtax, Xage) de (10 000, 75) tendría un valor de predicción de la casa de:

$$Yprice = 212\,000 + 109 * 10\,000 - 2\,000 * 75 = 1\,152\,000$$

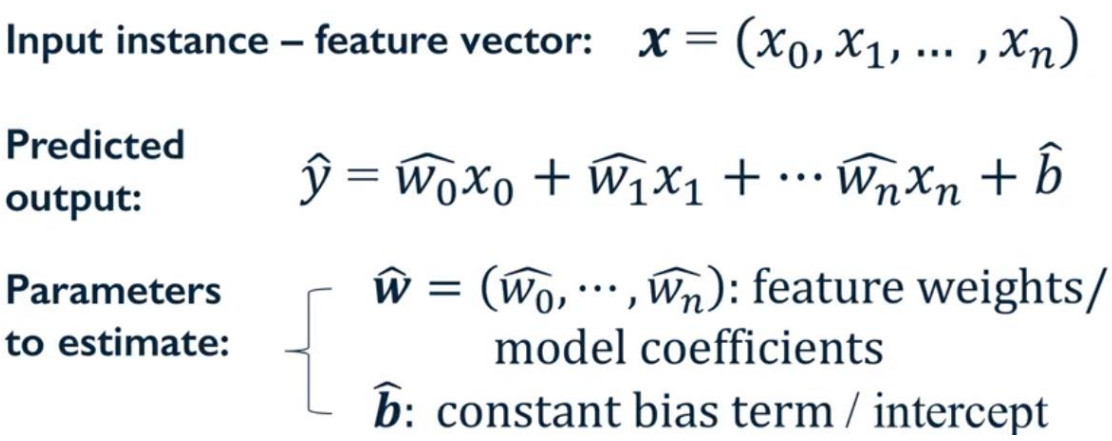


Figura N.º 11. La regresión lineal es un ejemplo de un modelo lineal.

En la figura anterior se pueden apreciar los diferentes componentes del modelo de regresión lineal; por un lado, tenemos el vector de *features.* En el ejemplo anterior eran los impuestos por año y la edad de la casa en años.

El proceso de entrenamiento de una regresión lineal consiste en estimar las **cargas** de cada una de sus variables y el ***intercept, slope* o pendiente.** La siguiente figura muestra la ecuación resultante de entrenar el modelo de regresión lineal con una variable predictora. El método que se aplica para estimar el *intercept* y las cargas se llama **mínimos cuadrados ordinarios.**

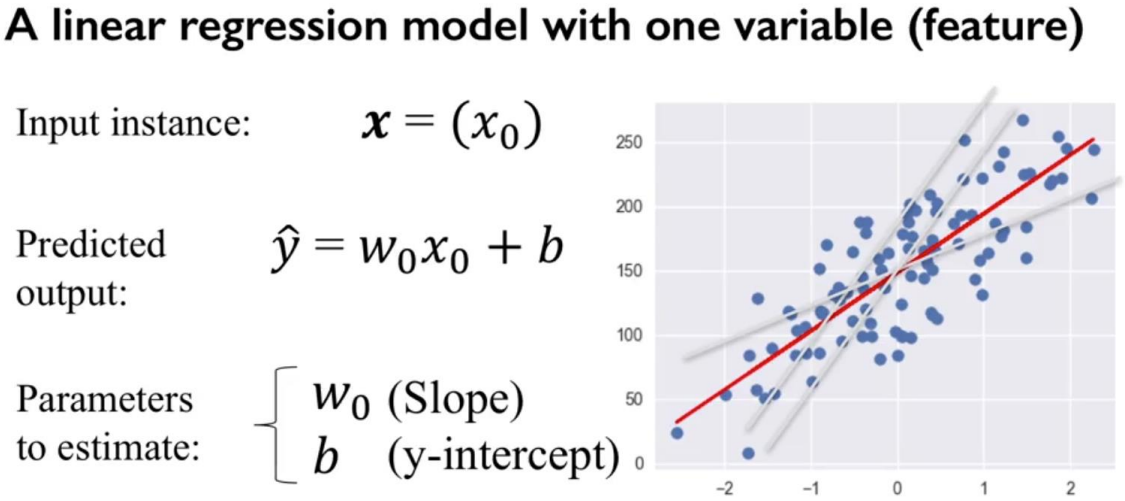


Figura N.º 12. La regresión lineal con una variable.

Los mínimos cuadrados ordinarios consisten en encontrar los valores de *w* y *b que minimizan error cuadrático medio del modelo.* Esto es la suma de las diferencias al cuadrado del *target* con el valor predicho. Este concepto se ilustra con el siguiente gráfico, las líneas negras representan la diferencia entre el valor predicho y el valor real.

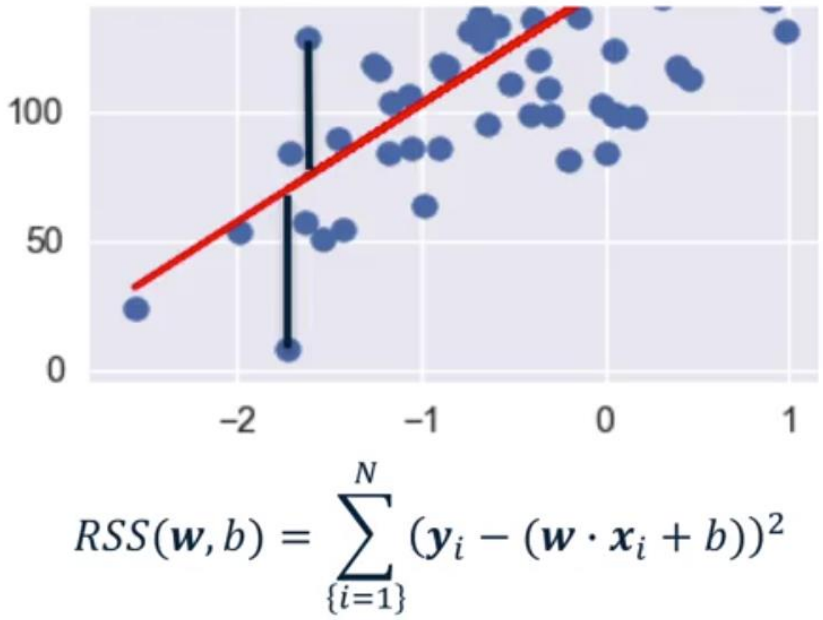


Figura N.º 13. Mínimos cuadrados ordinarios.

En la ecuación anterior, el valor de *y* representa el valor real a predecir, mientras que $w * xi + b$ representa la predicción generada por el modelo.

Ejemplo de implementación Lasso
[https://revistas.unaaa.edu.pe/index.php/unaaaciencia/article/view/29]

Ridge regression
Ridge regression usa el mismo criterio que los mínimos cuadros ordinarios para estimar el ***intercept*** y los coeficientes de cada una de las variables (*w*), pero añade una **penalidad** cuando existen grandes variaciones en los parámetros (*w*).

$$RSS_{RIDGE}(\boldsymbol{w}, b) = \sum_{\{i=1\}}^{N} (\boldsymbol{y}_i - (\boldsymbol{w} \cdot \boldsymbol{x}_i + b))^2 + \alpha \sum_{\{j=1\}}^{p} w_j^2$$

Figura N.º 14. *Rigde regression*.

- Una vez que los parámetros son aprendidos, la fórmula de la *rigde regression* es la misma que la que generaría los mínimos cuadrados ordinarios.
- El parámetro de penalidad que le añade se denomina **regularización.** La regularización previene el *overfitting,* reduciendo la complejidad de los modelos; es decir, que el modelo se ajuste demasiado a los datos de entrenamiento.
- *Ridge regression* usa la regularización L2. La cual minimiza la suma de los cuadrados de los pesos (*w*) ver Figura N.º 14.
- La influencia de la regularización es controlada por el parámetro alfa (α).
- Valores altos de α implican mayor regularización y, en consecuencia, modelos más simples.

Normalización de *features*

La normalización de *features* es muy importante para muchos de los algoritmos de *machine learning;* es decir, que todas las variables tengan la misma escala. Algunos de los algoritmos que necesitan son regresión regularizada, k–vecinos más cercanos, *support vector machines,* redes neuronales. Hay diferentes maneras de normalizar los datos, una de las formas más usadas es por *MinMax scaling.*

$$x_i' = (x_i - x_i^{MIN})/(x_i^{MAX} - x_i^{MIN})$$

Figura N.º 15. Fórmula *MinMax Scaling.*

Lasso regression
Lasso regression usa el mismo criterio que los mínimos cuadros ordinarios para estimar el ***intercept*** y los coeficientes de cada una de las variables (*w*) pero añade una **penalidad** cuando existen grandes variaciones en los parámetros (*w*). En este caso incluye regularización L2.

$$RSS_{LASSO}(\boldsymbol{w}, b) = \sum_{\{i=1\}}^{N} (\boldsymbol{y}_i - (\boldsymbol{w} \cdot \boldsymbol{x}_i + b))^2 + \alpha \sum_{\{j=1\}}^{p} |w_j|$$

Figura N.º 16. *Lasso regression*.

- La regularización/penalización L1 minimiza la suma de los valores absolutos de los coeficientes.
- El parámetro α controla la cantidad de regularización L1 (por defecto es 1.0).
- La fórmula de predicción es la misma que la de los mínimos cuadrados ordinarios.

***Feature* polinomiales con regresión logística**

$$\boldsymbol{x} = (x_0, x_1) \Rightarrow \boldsymbol{x}' = (x_0, x_1, x_0^2, x_0 x_1, x_1^2)$$

$$\hat{y} = \hat{w}_0 x_0 + \hat{w}_1 x_1 + \hat{w}_{00} x_0^2 + \hat{w}_{01} x_0 x_1 + \hat{w}_{11} x_1^2 + b$$

Figura N.º 17. *Feature* polinomiales.

- Consiste en generar nuevas variables basadas en combinaciones polinomiales de dos variables originales.
- El grado del polinomio especifica cuántas variables participarán cada vez en cada nueva *feature.* En el ejemplo de la Figura N.º 17, el polinomio es de grado 2.
- Los *feature* polinomiales siguen siendo combinaciones lineales de otras variables; por lo tanto, siguen siendo modelos lineales y pueden usar tambien los **mínimos cuadrado ordinarios** como método de estimación de los parámetros de la regresión *w* y *b*.

Referencias citadas en la Clase 3

Bishop, C. M. (2006). *Pattern recognition and machine learning* (1st ed.). Springer.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: with applications in R* (1st ed.). Springer.

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

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Ng, A. (2018). *Machine learning yearning: Technical strategy for AI engineers, in the era of deep learning*. AI Publishing.

Definición de los términos citados en la Clase 3

***Feature* polinomiales.** Las *feature* polinomiales en regresión lineal son transformaciones de las variables originales que crean nuevas características elevando las variables a potencias mayores. Por ejemplo, si tienes una variable x, una característica polinómica podría ser x^2 o x^3 . Esto permite que el modelo capture relaciones no lineales entre las variables **independientes** y la variable **dependiente**, lo que mejora la capacidad de predicción en casos donde la relación es curvilínea. Básicamente, amplía la flexibilidad del modelo para ajustarse a patrones más complejos en los datos.

Regularización L1. La regularización L1, también conocida como ***Lasso*** (Least Absolute Shrinkage and Selection Operator), consiste en agregar una penalización al modelo de **regresión** que es proporcional a la suma de los valores absolutos de los coeficientes de las características. Esto tiene el efecto de reducir algunos coeficientes a cero, lo que resulta en un modelo más simple y puede ayudar a la selección automática de características. En resumen, la regularización L1 promueve la **'esparcidad'** en el modelo, eliminando características irrelevantes y mejorando su generalización.

Regularización L2. La regularización L2, también conocida como ***ridge***, agrega una penalización al modelo de regresión que es proporcional a la suma de los cuadrados de los coeficientes de las características. Esto ayuda a evitar que los coeficientes crezcan demasiado grandes, lo que podría llevar a un sobreajuste (***overfitting***). A diferencia de la regularización L1, L2 no hace que los coeficientes se vuelvan exactamente cero, sino que los reduce de manera más uniforme. En resumen, L2 favorece modelos con coeficientes pequeños y distribuidos de manera más equitativa.

PUCE Virtual
EDUCACIÓN FLEXIBLE

La excelencia no se improvisa

síguenos

MDASC - 12 - 241