# Título o Nombre de las notas

Cristo Daniel Alvarado

11 de marzo de 2025

# Índice general

# Información y Recursos

Página del libro (incluye los datasets) de donde estoy sacando algo de contenido: The Elements of Statistical Learning.

## §0.1 Libros

*Escrito por ChatGPT:* Si buscas un enfoque teórico y práctico para aprender Machine Learning, estos libros son altamente recomendados:

- **"Pattern Recognition and Machine Learning"** - Christopher M. Bishop Matemáticamente riguroso, ideal para alguien con formación en matemáticas.
- **"The Elements of Statistical Learning"** - Hastie, Tibshirani y Friedman Recomendado para obtener fundamentos teóricos sólidos.
- **"Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"** - Aurélien Géron Enfoque práctico con Python, excelente para comenzar con código.

## §0.2 Lenguajes de Programación

- **Python** (Principal) - Librerías clave: NumPy, Pandas, Scikit-Learn, TensorFlow, PyTorch.
- **R** - Útil para estadística avanzada y visualización de datos.

## §0.3 Páginas y Cursos en Línea

### §0.3.1 Cursos en Coursera

- Curso de Andrew Ng: Machine Learning Matemáticas y conceptos clave explicados de manera clara.
- Deep Learning Specialization Curso avanzado enfocado en redes neuronales.

### §0.3.2 Otros Cursos y Recursos

- Fast.ai: Practical Deep Learning for Coders Curso gratuito con enfoque práctico en código.

- Kaggle Learn Mini cursos interactivos y competencias con datos reales.
- CS229 de Stanford Notas de clase de Andrew Ng en Stanford.
- **StatQuest (YouTube)** Explicaciones claras de teoría estadística aplicada a ML.

Lo siguiente es sacado de ChatGPT (habla de estadística y de donde encontar cosas sobre teoría de la desición y cosas así):

## §0.4 Referencias en Teoría de la Decisión

Si quieres profundizar en **Teoría de la Decisión**, aquí tienes algunos libros y recursos avanzados que se alinean con una formación matemática rigurosa.

### §0.4.1 Libros Clásicos y Rigurosos

- **Statistical Decision Theory and Bayesian Analysis** – James O. Berger
  Un enfoque matemático sólido de la teoría de la decisión estadística, con énfasis en métodos bayesianos.
- **Theory of Point Estimation** – Erich L. Lehmann y George Casella
  Cubre teoría de la estimación bajo diferentes criterios de optimalidad y conecta con la teoría de la decisión.
- **Decision Theory: Principles and Approaches** – Giovanni Parmigiani y Lurdes Inoue
  Presenta teoría de la decisión clásica y bayesiana con aplicaciones en estadística y machine learning.
- **Introduction to Statistical Decision Theory** – John Pratt, Howard Raiffa, Robert Schlaifer
  Un libro clásico que introduce la teoría de la decisión en un contexto estadístico.
- **Foundations of Decision Theory** – Itzhak Gilboa
  Más filosófico y matemático, ideal si te interesan los fundamentos axiomáticos de la teoría de la decisión.

### §0.4.2 Libros con Aplicaciones en Machine Learning y Data Science

- **Bayesian Decision Theory and Machine Learning** – David Barber (Parte del libro "Bayesian Reasoning and Machine Learning")
  Relaciona la teoría de la decisión con aprendizaje automático.
- **Pattern Recognition and Machine Learning** – Christopher M. Bishop
  Enfatiza la teoría de la decisión bayesiana en clasificación y reconocimiento de patrones.

### §0.4.3 Temas Clave en Teoría de la Decisión

Si ya tienes nociones de probabilidad y optimización, te pueden interesar:

- **Decisiones Bayesianas** (Bayesian Decision Theory)

- **Teoría de Juegos y Decisión** (Game Theory & Decision Making)
- **Riesgo y utilidad** (Risk & Utility Theory)
- **Reglas de decisión óptimas** (Optimal Decision Rules)
- **Criterios de Minimax y Bayes**

Si buscas algo más aplicado o enfocado en optimización estadística, hay muchos otros recursos disponibles.

### §0.4.4 Cosas que llamaron la atención

- Convolutional Neural Networks.
- Natural Language Processing.
- Recurrent Neural Networks.
- Transformers.
- Clustering.

# Capítulo 1

# Introducción

El objetivo de estas notas es el de dar un panorama general sobre el Machine Learning.

## §1.1 Aprendizaje Estadístico

**Definición 1.1.1** (**Aprendizaje Estadístico**)
El **aprendizaje estadístico** consiste en

El aprendizaje estadístico juega un rol clave en muchas áreas de la ciencia, finanzas y la industria. Tales ejemplos pueden verse en los siguientes problemas:

- Predecir cuando un paciente hospitalizado debido a un ataque cardiaco tendrá un segundo ataque cardiaco. Esta predicción debe basarse en la deomgrafía del paciente.
- Predecir precios de acciones a 6 meses a futuro.
- Identificar números escritos a mano en códigos postales.
- Identificar factores de riesgo para cáncer de próstata, basado en variables clínicas y demográficas.

**Definición 1.1.2** (**Demografía**)
La **demografía** es la ciencia que estudia las poblaciones humanas, su dimensión, estructura, evolución y características generales, así como los procesos que determinan su formación, conservación y desaparición.

El objetivo es tomar información, interpretarla y decir algo acerca de ella, más precisamente, se pretende dar una *predicción*.

Usando la información se construirá un **aprendedor** (o **learner**) que será la pieza clave para predecir la salida de nueva fuente de información. Queremos un *good learner*.

Hay dos tipos de aprendizaje:

- **Supervisado**: En este tipo de aprendizaje hay la presencia de una variable de salida que nos permite guiar el proceso de aprendizaje.
- **No Supervisado**: Observamos solo las características y no tenemos medida de que tan buena es la salida del proceso.

Hablaremos de algunos ejemplos de aprendizaje supervisado:

## §1.2 Spam de Email

**Ejercicio 1.2.1** (**Spam de Email**)
Se tienen 4601 mensajes en email hacia una persona y pretendemos determinar si cada uno de ellos era un email basura o *spam*. Diseñar un detector de spam automático que pueda filtrar el spam antes de que las bandejas de entrada de los usuarios se llenen.

**Solución:**
Para cada uno de los 4601 mensajes, podemos asignar dos estados de salida:

`email` ó `spam`

Este es un tipo de problema dentro del área de aprendizaje supervisado y es llamado **problema de clasificación**.

| | george | you | your | hp | free | hpl | ! | our | re | edu | remove |
|---|---|---|---|---|---|---|---|---|---|---|---|
| spam | 0.00 | 2.26 | 1.38 | 0.02 | 0.52 | 0.01 | 0.51 | 0.51 | 0.13 | 0.01 | 0.28 |
| email | 1.27 | 1.27 | 0.44 | 0.90 | 0.07 | 0.43 | 0.11 | 0.18 | 0.42 | 0.29 | 0.01 |

Figura 1. Tabla donde se muestran las probabilidades de que cierta palabra aparezca en un correo del tipo `spam` y `email`.

Nuestro método de aprendizaje decide cuáles de características usar y como: Por ejemplo puede usar una regla como la siguiente:

```
1 if george < 0.6 & you > 1.5 then spam, else email.
```

o también:

```
1 if 0.2*you - 0.3*george > 0 then spam, else email.
```

En este problema no todos los errores son iguales, queremos evitar filtrar email que realmente corresponda a `email`. Una forma de atacar este problema se discute más adelante. □

**Observación 1.2.1**
No existe un lenguaje de programación usado en este texto a menos que se indique lo contrario. Si no parece que es algún lenguaje conocido, como el caso del ejercicio anterior, asumiremos que este es pseudocódigo.

**Ejercicio 1.2.2** (**Reconocimiento de Números en Manuscrita**)
Se tienen miles de cartas escritas a mano con códigos postales en las mismas. Cada carta es escaneada y almacenada. De cada una de las cartas se extrae el área que contiene el código postal y se almacena. El objetivo es determinar el número que está escrito en el código postal.

**Solución:**
Lo que se hace es primero, separar cada uno de los números en la imagen para después, colocar ese número en un grid de $16 \times 16$ bits en escala de grises con cada uno de éstos variando de intensidad desde 0 hasta 255. Se pretende con esto determinar cada uno de los números en cada una de las imágenes.

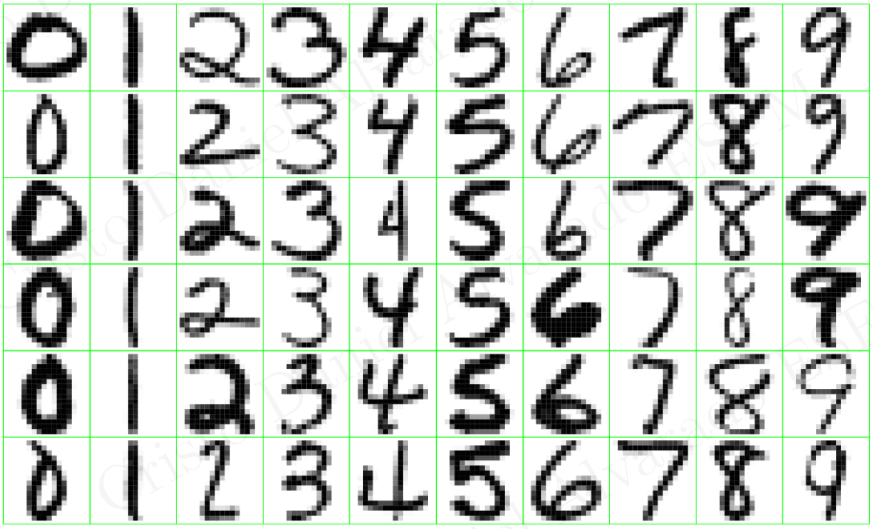

Figura 2. Diversos números sacados de cartas en códigos postales.

Este es un problema de clasificación cuyo error debe mantenerse bajo para poder dar un resultado lo mejor posible. □

Otro problema inteeresante es el de *regresión*, en el que básicamente se da un input de información varia y se requiere dar un output de cierto parámetro que puede o no ser deducido a partir de esos valores introducidos inicialmente.

Aplicar técnicas de Machine Learning para escarbar en una gran cantidad de información puede ayudar a descubrir patrones donde no es claro que éstos existan, esto es llamado **data mining**.

Podemos aplicar las técnicas del Machine Learning en tres áreas:

- Supervisado o no supervisado (entran como subcategorías semisupervisado y aprendizaje con refuerzo).
- Online and Batch Learning.
- Si trabajan comparando nueva información con puntos de información ya existentes o si a partir de la información pueden obtener predicciones, como haría un científico.

Los algoritmos de aprendizaje supervisado más importantes son los siguientes:

- $k$-nearest neighbors.
- Linear Regresssion.
- Logistic Regression.
- Support Vector Machines (SVMs).
- Decision Trees and Random Forests.
- Neural Networks (algunas pueden ser no supervisadas o semisupervisadas).

En no supervisado:

- Clustering:
  - $k$-means.
  - DBSCAN.

    - Hierarchical Cluster Analysis (HCA).
- Anomaly detection and novelty detection:
    - One-class SVM.
    - Isolation Forest.
- Visualization and dimensionality reduction:
    - Principal Component Analysis
    - Kernel PCA
    - Locality Linear Embedding
    - $t$-distributed stochastic neighbor embedding.
- Association rule learning:
    - A priori.
    - Eclat.

La gran mayoría de algoritmos de aprendizaje semisupervisado son mezclas de algorimos supervisados y no supervisados.

El aprendizaje reforzado es muy diferente. Hay un *agente* que recibe *rewards* o *penalties* en función de la acción que haga. Una política define que cosas debe elegir el *agent* y cuales no, dada una situación.

# Capítulo 2

# Vista Panorámica del Aprendizaje Supervisado

## §2.1 Introducción

El objetivo es que dada entrada de información demos una salida de la misma que podemos medir (ya que en aprendizaje supervisado eso es lo que se hace). Pueden ser muchas entradas de información de diferentes tipos y varias salidas igualmente de diferentes tipos.

**Definición 2.1.1** (**Terminología Machine Learning**)
Los *inputs* son a menudo llamados **predictors** o **independent variables** (**variables independientes** en español), los *outputs* son llamados **responeses** o **variables dependientes** (**respuestas** en español).

Se adoptará mucha terminología matemática vista en cursos de álgebra lineal y temas un poco más avanzados, por lo que no se ahondara mucho en ello. Sin embargo, es relevante mencionar el tipo de información con el que puede que estemos trabajando a menudo:

## §2.2 Mínimos Cuadrados y Vecinos Cercanos

**Definición 2.2.1**
Si $\vec{x}$ es un vector $n$-dimensional, entenderemos que $\vec{x}$ es una matriz columna y así, $\vec{x}^T$ denotará a la transpuesta de esta matriz.

El modelo más simple de predicción es el modelo líneal. Si $\vec{x} = (x_1, x_2, ..., x_n)$ es una entrada, entonces la salida del modelo líneal es el vector:

$$\vec{y} = \hat{\beta}_0 + \sum_{i=1}^{n} x_i \hat{\beta}_j \tag{2.1}$$

donde $\hat{\beta}_j$ son vectores $m$-dimensionales para todo $j \in \big[\big|0, n\big|\big]$. En particular, el vector $\hat{\beta}_0$ es llamado el **bias** en Machine Learning. Una forma más sencilla de representar lo anterior es haciendo $\vec{x'} = (1, x_1, ..., x_n)$ y tomar:

$$\vec{y} = \vec{x'}^T \hat{\beta} \tag{2.2}$$

donde $\hat{\beta} = (\hat{\beta}_0, ..., \hat{\beta}_0)$. Notemos que en este caso las coordenadas $x_1, ..., x_n$ del vector $\vec{x}$ son escalares y las de $\hat{\beta}$ pueden ser incluso vectores, con lo que $\hat{\beta}$ en general será una matriz. La salida tendrá la misma dimensión que las coordenadas de las entradas de $\hat{\beta}$.

Podemos construir así la función:

$$f(\vec{x}) = \vec{x'}^T \hat{\beta} \tag{2.3}$$

esta es una función de un espacio $n$-dimensional en uno $m$-dimensional.

Vista como función:

$$f(\vec{x}) = \vec{x}^T \beta \tag{2.4}$$

donde $\beta$ es una matriz $m \times n$ y es tal que $\nabla f(\vec{x}) = \beta$, es decir lo clásico que uno hace en un curso de Cálculo III para determinar derivadas de transformaciones lineales.

Nuestro objetivo es que este modelo lineal se acomple a nuestra información. Hay muchas formas de hacerlo pero la más sencilla es por **mínimos cuadrados**. En tal aproximación lo que uno hace es tomar los coeficientes de $\beta$ para minimizar el residuo de la suma de cuadrados:

$$\text{RSS}(\beta) = \sum_{i=1}^{k} (y_i - \vec{x_i}^T \beta)^2 \tag{2.5}$$

la cosa anterior es un escalar (el cuadrado es el producto interno del vector resultante consigo mismo) y asumimos que tenemos un dataset de $k$-vectores. De forma más simplificada podemos escribir:

$$\text{RSS}(\beta) = (\vec{y} - X^T \beta)^T (\vec{y} - X^T \beta) \tag{2.6}$$

donde $\vec{y} = (y_1, ..., y_n)$ y $X = (\vec{x_1}, ..., \vec{x_k})$. Notemos que $X^T \beta$ define la dimensión de las coordenadas del vector $\vec{y}$.

Se tiene que $X$ tiene a cada columna como un vector de entrada $\vec{x_i}$ y la salida está dada por $y_i$, esto en el conjunto de entrenamiento.

Esto es la cosa más simple del mundo de hacer en programación. Considere el siguiente ejemplo:

**Ejemplo 2.2.1**
Se tiene un documento .csv donde se encuestó a personas de ciudades con cierto *GDP per cápita (USD)* sobre su satisfacción de vida. Dada cierto GDP per cápita, determine la satisfacción de vida de la persona.

**Solución:**

Este claramente es un modelo de regresión. Observemos primero un plot de la información para inferir si es que hay algo que podamos determinar de ella:

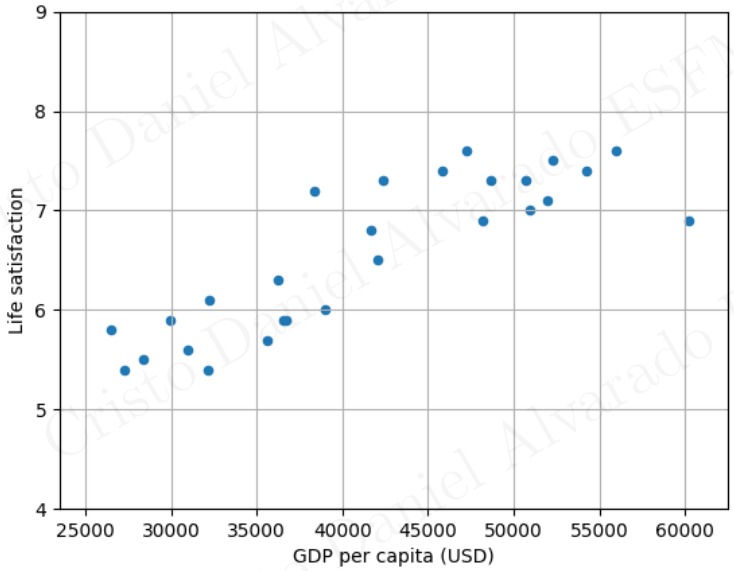


Figura 3. Caption.

rápidamente uno puede ver que tal vez haya alguna relación lineal, por lo que se implementa el siguiente código en Python para realizar un modelo de regresión lineal:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

# Download and prepare the data
data_root = "https://github.com/ageron/data/raw/main/"
lifesat = pd.read_csv(data_root + "lifesat/lifesat.csv")
X = lifesat[["GDP per capita (USD)"]].values
y = lifesat[["Life satisfaction"]].values

# Visualize the data
lifesat.plot(kind='scatter', grid=True,
             x="GDP per capita (USD)", y="Life satisfaction")
plt.axis([23_500, 62_500, 4, 9])

# Select a linear model
model = LinearRegression()

# Train the model
model.fit(X, y)

# Graph Model
y_pred = model.predict(X)
plt.plot(X, y_pred, color='red')
plt.show()
```

Listing 2.1: Ejemplo Regresión Lineal

El modelo lo que hizo fue generar la siguiente línea recta roja:

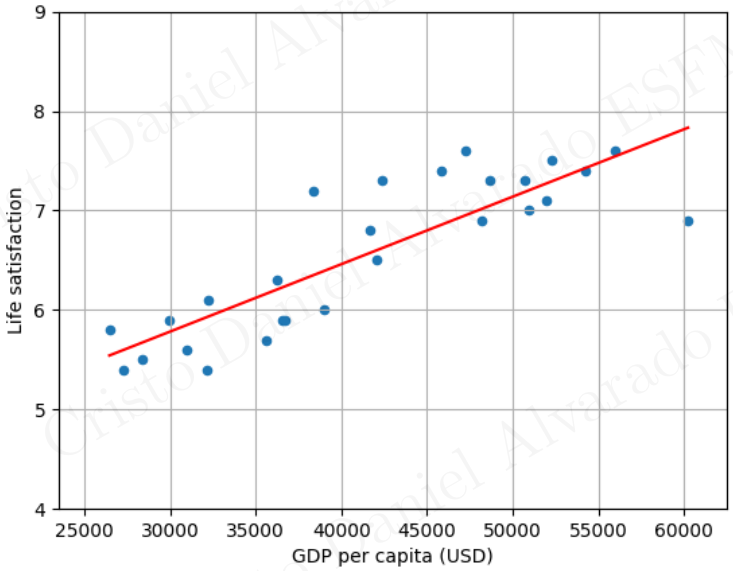


Figura 4. Caption.

Con este mismo ejemplo podemos implementar otro modelo, por ejemplo el de $k$-nearest neighbors con $k = 3$.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsRegressor

# Download and prepare the data
data_root = "https://github.com/ageron/data/raw/main/"
lifesat = pd.read_csv(data_root + "lifesat/lifesat.csv")
X = lifesat[["GDP per capita (USD)"]].values
y = lifesat[["Life satisfaction"]].values

# Visualize the data
lifesat.plot(kind='scatter', grid=True,
             x="GDP per capita (USD)", y="Life satisfaction")
plt.axis([23_500, 62_500, 4, 9])

# Select a linear model
model = KNeighborsRegressor(n_neighbors=3)

# Train the model
model.fit(X, y)

# Graph Model
y_pred = model.predict(X)
plt.plot(X, y_pred, color='red')
plt.show()
```

Listing 2.2: Ejemplo $k$-vecinos

En este caso, la gráfica queda de la siguiente manera:

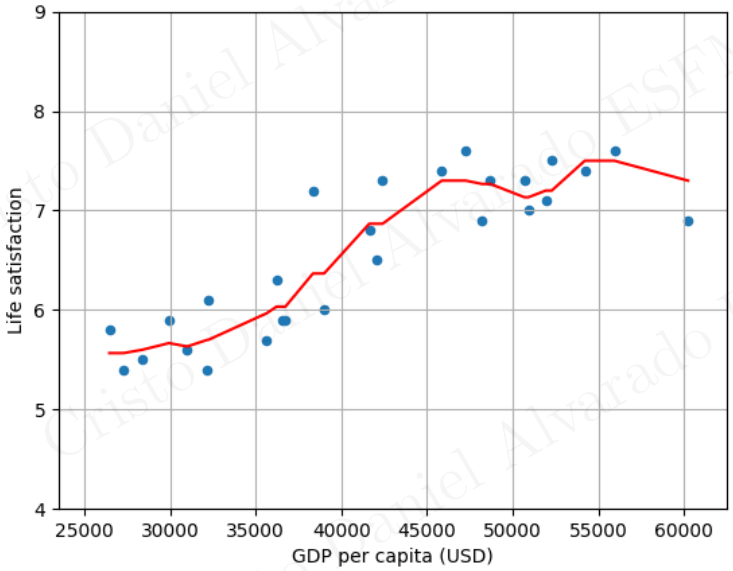


Figura 5. Caption.

El algoritmo de regresión lo que hace (de forma descriptiva) es encontrar los $k$ vecinos más cercanos de un punto y hace el promedio de los valores.

□

En síntesis, se siguen los 4 siguientes pasos:

- Estudiar la información.
- Limpiar la información.
- Preparar la información.
- Seleccionar un modelo.
- Entrenar el modelo con la información de entrenamiento.
- Aplicar el modelo para hacer predicciones.