Introducción a la Inteligencia Artificial
Facultad de Ingeniería
Universidad de Buenos Aires

Ing. Lautaro Delgado
(lautarodc@unops.org)

Índice

1. Exposiciones ejercicios clase 1
2. Ejercicios k-means
    a. Datos sintéticos
    b. Clusterización
3. Teoría - Principal Component Analysis
    a. Concepto
    b. Demostración Matemática
        i. Enfoque de máxima varianza
        ii. Enfoque de error de reconstrucción mínimo
        iii. Enfoque de variables latentes
    c. Otros métodos
4. Práctica - Principal Component Analysis
    a. PCA en Numpy
    b. Ejercicios de Aplicación
5. Ejercicio Integrador

Repaso Ejercicios
Clase 1

Ejercicio #1 - C2 | Datasets sintéticos

En problemas de Machine Learning es muy importante contar con el dataset correcto. Pero muchas veces, el dataset nos es fácil de conseguir o el equipo de data engineers aún lo está generado.

Una manera sencilla de generar datos para probar soluciones de Machine Learning es crear datasets sintéticos. Por ejemplo, es simple crear datasets con grados de clusterización variables y probar cómo se comportan nuestros algoritmos en diferentes escenarios.

Objetivo: Utilizar numpy para crear datos clusterizados A/B en 4 dimensiones.

Hint:

- Definir una matriz con centroides [1,0,0,0] y [0,1,0,0]
- Utilizar una constante para separar o alejar los centroides entre sí.
- Utilizar np.repeat para crear n/2 muestras de cada centroide.
- Sumar a cada centroide un vector aleatorio normal i.i.d. con media 0 y desvío (np.random.normal).
- Armar un arreglo que tenga n enteros indicando si la muestra pertenece a A o a B.

Ejercicio #6 - C1 | Distancia a centroides

Dada una nube de puntos X y centroides C, obtener la distancia entre cada vector X y los centroides utilizando operaciones vectorizadas y broadcasting en NumPy. Utilizar como referencia los siguientes valores:

- X = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
- C = [[1, 0, 0], [0, 1, 1]]

Ejercicio #7 - C1 | Enunciado

Obtener para cada fila en X, el índice de la fila en C con distancia euclídea más pequeña.
Es decir, decir para cada fila en X a qué cluster pertenece en C.
Por ejemplo, si el resultado anterior fue:

```
# [[ 3.60555128  8.36660027 13.45362405]
#  [ 2.44948974  7.54983444 12.72792206]]
```

El programa debería devolver [1, 1, 1]

Hint: utilizar np.argmin

Ejercicio #8 - C1 | Implementacion basica de K-means en Numpy

K-means es uno de los algoritmos más básicos en Machine Learning no supervisado. Es un algoritmo de clusterización, que agrupa los datos que comparten características similares. Recordemos que entendemos datos como n realizaciones del vector aleatorio X.

El algoritmo K-means funciona de la siguiente manera:

1. El usuario selecciona la cantidad de clusters a crear (n).
2. Se seleccionan n elementos aleatorios de X como posiciones iniciales del los centroides C.
3. Se calcula la distancia entre todos los puntos en X y todos los puntos en C.
4. Para cada punto en X se selecciona el centroide más cercano de C.
5. Se recalculan los centroides C a partir de usar las filas de X que pertenecen a cada centroide.
6. Se itera entre 3 y 5 una cantidad fija de veces o hasta que la posición de los centroides no cambie.

Implementar la función *def k_means(X, n)* de manera tal que al finalizar devuelva la posición de los centroides y a qué cluster pertenece cada fila de X.

Hint: para (2) utilizar funciones de np.random, para (3) y (4) usar los ejercicios anteriores, para (5) es válido utilizar un for. Iterar 10 veces entre (3) y (5).

Aprendizaje no supervisado

El objetivo de los modelos no supervisados es encontrar la “mejor” representación de los datos. Con mejor nos referimos a una representación que preserve la mayor cantidad de información posible de los datos, bajo una determinada “penalidad o restricción”, que haga que la representación sea más accesible o simple.

Ejemplos de representaciones más simples:

- Representación de menor dimensionalidad
- Representación sparsa
- Representación independiente

Ingeniería de Features - PCA

En ocasiones los datos de entrada tienen muchas features y se torna costoso en tiempo y recursos entrenar modelos de ML con todo el dataset. En la práctica se pueden utilizar técnicas de reducción de la dimensión no supervisadas como PCA (Principal Component Analysis).

Casos de Uso

- Compresión de datos
- Identificación de patrones
- Factores latentes
- Visualización

Conocimientos Previos

- Bases y cambio de bases
- Proyecciones
- Valores y vectores propios
- Distribución gaussiana
- Optimización con restricciones

Principal Component Analysis - Teoría

PCA

Queremos encontrar proyecciones … de observaciones de datos ..., que sean lo más similares posibles a los originales, pero con significativamente menos dimensiones.

Principal Component Analysis - Teoría

PCA

Dado un dataset i.i.d:

$$\chi = \{x_1, \cdots, x_N\}, x_N \in \mathbb{R}^D$$

x_{11}	$\cdots$	$\cdots$	x_{1n}
$\cdots$	$\cdots$	$\cdots$	$\cdots$
x_{d1}	$\cdots$	$\cdots$	x_{dn}

χ

con medio cero, la matriz de covarianza es:

$$S = \frac{1}{N} \sum_{n=1}^{N} x_n x_n^T$$

Definimos transformaciones lineales:

$$z_n = B^T x_n \in \mathbb{R}^M$$

$$B = [b_1, \cdots, b_m] \in \mathbb{R}^{DxM}, b_i^T b_j = 0 \ \forall \ i \neq j$$

Principal Component Analysis - Teoría

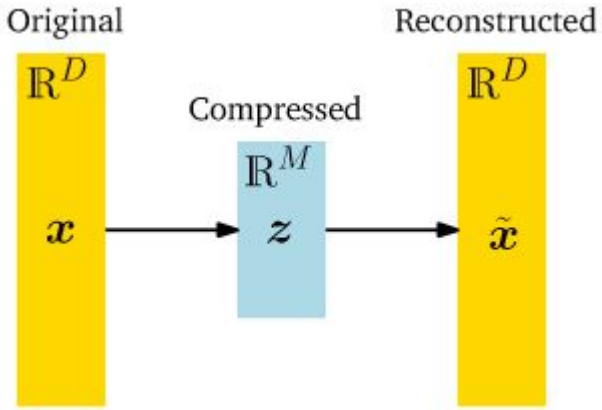
PCA

Buscamos un subespacio

$$U \subseteq \mathbb{R}^D / \ dim(U) = M < D$$

donde proyectar los datos. Es decir encontrar para:

$$\tilde{x}_n \in \mathbb{R}^D \rightarrow \begin{cases} z_n \\ [b_1, \cdots, b_m] \end{cases}$$



$$z_n = B^T x_n$$

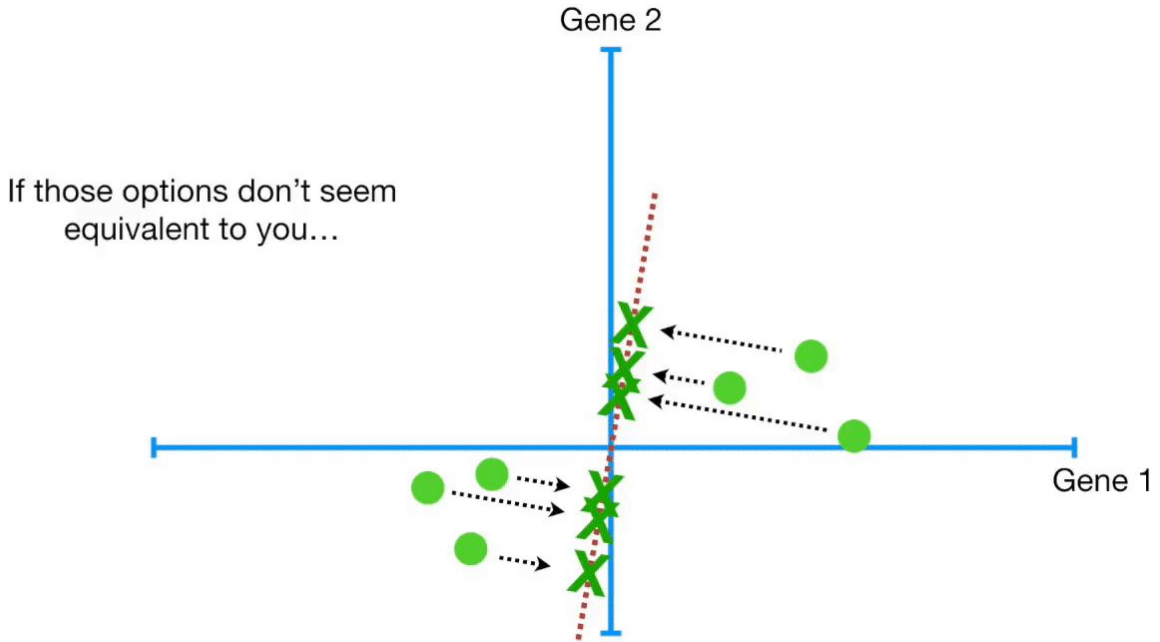
$$\tilde{x}_n = B z_n$$

i. Enfoque de máxima varianza
ii. Enfoque de error de reconstrucción mínimo
iii. Enfoque de variables latentes

Jamboard - Desarrollo Matemático PCA

PCA

Comparación métodos 1 y 2.



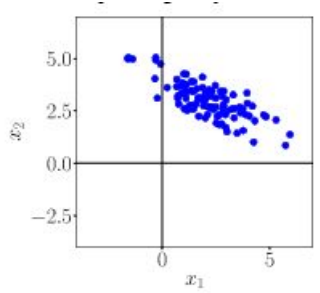
Principal Component Analysis - Teoría

PCA

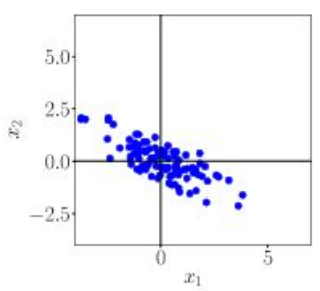
Pasos principales:

1. Centramos los datos
2. Estandarización
3. Autovalores de la matriz de covarianza
4. Proyección

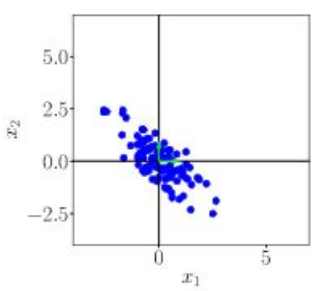
$$z_n = B^T x_n$$



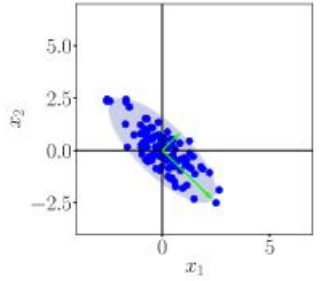
(a) Original dataset.



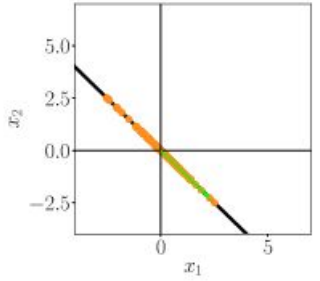
(b) Step 1: Centering by subtracting the mean from each data point.



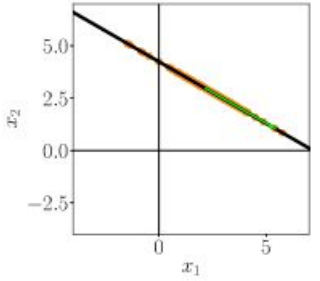
(c) Step 2: Dividing by the standard deviation to make the data unit free. Data has variance 1 along each axis.



(d) Step 3: Compute eigenvalues and eigenvectors (arrows) of the data covariance matrix (ellipse).



(e) Step 4: Project data onto the principal subspace.



(f) Undo the standardization and move projected data back into the original data space from (a).

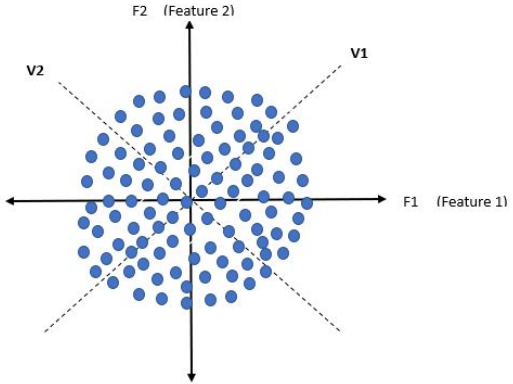
PCA

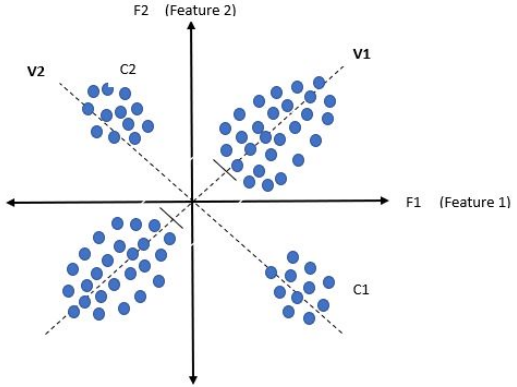
Derivaciones

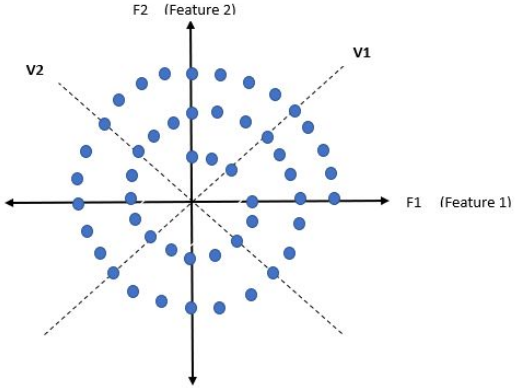
- Si en PCA cambiamos el mapeo lineal por uno no-lineal, obtenemos un auto-encoder. Si el mapeo no-lineal es una red neuronal, tenemos un deep auto-encoder.
- Cuando la varianza del ruido gaussiano es cero, PPCA → PCA.
- Si para cada dimensión, el ruido tiene una varianza distinta → Factor Analysis.
- Si cambiamos la distribución a priori de z por una no gaussiana → ICA

PCA

Limitaciones







Ejercicio #2 | Dado un dataset X, calcular PCA para reducir dimensión.

Siguiendo los pasos vistos en la teoría, se requiere utilizar numpy para calcular PCA del dataset de entrada X, utilizando la componente más importantes.

```
X = np.array( [ [0.8, 0.7] , [0.1, -0.1] ] )
```

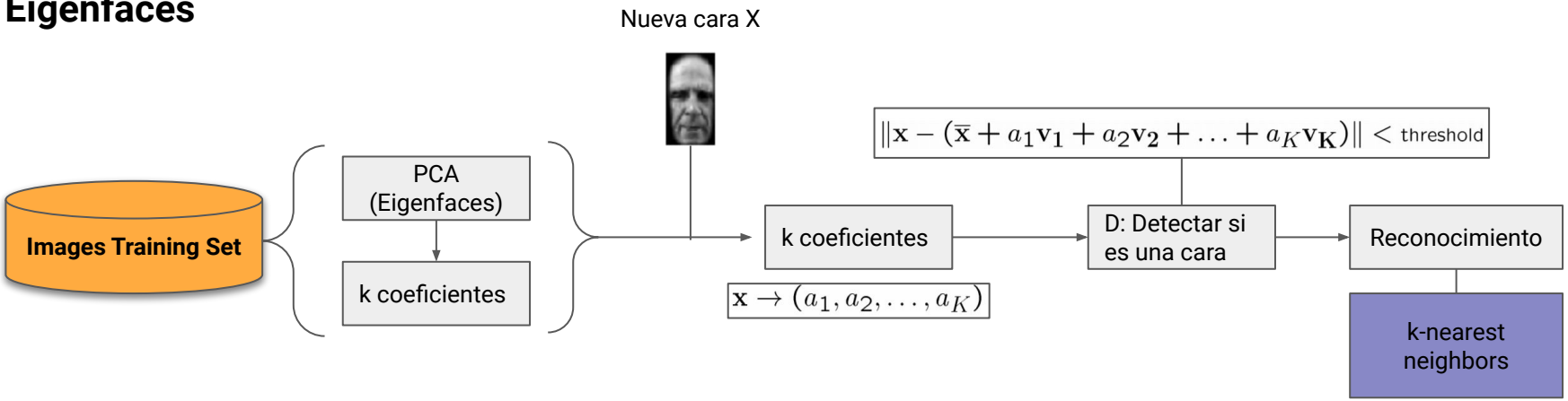
Al finalizar la implementación en numpy, corroborar obtener los mismos resultados que utilizando el código de la librería scikit-learn. Escribir un test para comparar las matrices.

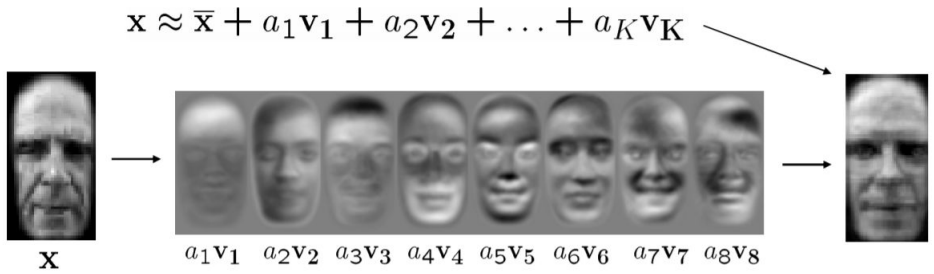
PCA- Casos Prácticos

Principal Component Analysis - Teoría

PCA

Eigenfaces





Más Ingeniería de Features

Missing Values

Es muy común en la práctica, recibir como datos de entrada, datasets que tienen información incompleta (“NaN”).

ID	City	Degree	Age	Salary	Married ?
1	Lisbon	NaN	25	45,000	0
2	Berlin	Bachelor	25	NaN	1
3	Lisbon	NaN	30	NaN	1
4	Lisbon	Bachelor	30	NaN	1
5	Berlin	Bachelor	18	NaN	0
6	Lisbon	Bachelor	NaN	NaN	0
7	Berlin	Masters	30	NaN	1
8	Berlin	No Degree	NaN	NaN	0
9	Berlin	Masters	25	NaN	1
10	Madrid	Masters	25	NaN	1

Missing Values - Solución # 1

Una forma de solucionar el problema es remover las filas y las columnas que contienen dichos valores.

ID	City	Degree	Age	Salary	Married ?
1	Lisbon	NaN	25	45,000	0
2	Berlin	Bachelor	25	NaN	1
3	Lisbon	NaN	30	NaN	1
4	Lisbon	Bachelor	30	NaN	1
5	Berlin	Bachelor	18	NaN	0
6	Lisbon	Bachelor	NaN	NaN	0
7	Berlin	Masters	30	NaN	1
8	Berlin	No Degree	NaN	NaN	0
9	Berlin	Masters	25	NaN	1
10	Madrid	Masters	25	NaN	1

Missing Values - Solución # 2

En columnas donde el % de NaNs es relativamente bajo, es aceptable reemplazar los NaNs por la media o mediana de la columna.

Average_Age = 26.0

ID	City	Age	Married ?
1	Lisbon	25	0
2	Berlin	25	1
3	Lisbon	30	1
4	Lisbon	30	1
5	Berlin	18	0
6	Lisbon	NaN	0
7	Berlin	30	1
8	Berlin	NaN	0
9	Berlin	25	1
10	Madrid	25	1

ID	City	Age	Married ?
1	Lisbon	25	0
2	Berlin	25	1
3	Lisbon	30	1
4	Lisbon	30	1
5	Berlin	18	0
6	Lisbon	26	0
7	Berlin	30	1
8	Berlin	26	0
9	Berlin	25	1
10	Madrid	25	1

Missing Values - Solución avanzada

Las técnicas mencionadas producen distorsiones en la distribución conjunta del vector aleatorio. Estas distorsiones pueden ser muy considerables y afectar en gran medida el entrenamiento del modelo. Para reducir este efecto se puede utilizar **MICE (Multivariate Imputation by Chained Equation)**

1. Se trata cada columna con missing values como la variable dependiente de un problema de regresión.
2. Se van haciendo los fits de cada columna de manera secuencial.
3. Se utiliza la regresión para completar los missing values.

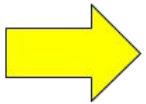
One hot encoding

En muchos problemas de Machine Learning, puedo tener como dato de entrada variables categóricas. Por ejemplo, una columna con información sobre el color: {rojo, amarillo, azul}

Para este tipo de información, donde no existe una relación ordinal natural entre las categorías, no sería correcto asignar números a las categorías.

Una forma más expresiva de resolver el problema es utilizar “one hot encoding” y transformar la información en binaria de la siguiente manera.

Color
Red
Red
Yellow
Green
Yellow



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Machine Learning Terminology

- Raw vs. Tidy Data
- Training vs. Holdout Sets
- Baseline
- Parameters vs. Hyperparameters
- Classification vs. Regression
- Model-Based vs. Instance-Based Learning
- Shallow vs. Deep Learning

Input Analysis - Machine Learning Pipelines

Dataset pipeline

Acciones que generalmente se ejecutan sobre los datasets.

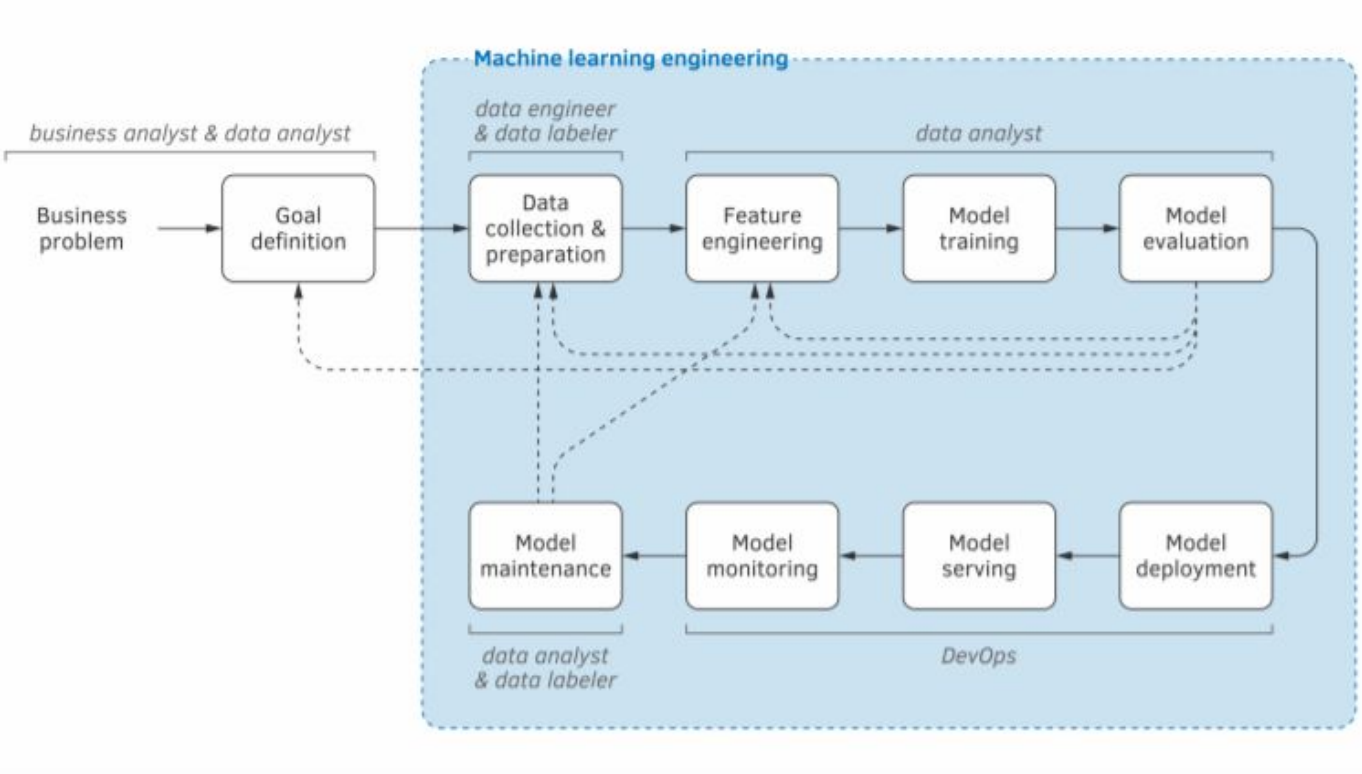
Obtención de datos o synthetic dataset	Pre-procesamiento de Missing Values	Cómputo de media, desvío y cuantiles	Estandarización de datos (z-score)	Ingeniería de Features (PCA)	Data augmentation	Split en Train, Validation y Test

Model pipeline

Pasos involucrados al entrenar un modelo de Machine Learning

Obtener el dataset para train	Definir métricas de evaluación y train	Calcular métricas para modelos base	Entrenar el modelo con el dataset train	Computar métricas con validation	HPs optimization	Evaluación sobre el dataset test

Machine Learning Pipeline



Ejercicio
Integrador ML

Ejercicios

Ejercicio #3 | Calcular la inversa generalizada y simular

Sabiendo que una variable aleatoria es continua, la función de densidad de probabilidad se define según:

$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)dt$$

Para una variable aleatoria con función de densidad de probabilidad:

$$f_X(x) = 3x^2\{0 < x < 1\}$$

Obtener la inversa generalizada y utilizar numpy para simular n muestras.

Ejercicio #4 | Normalización

Muchos algoritmos de Machine Learning necesitan datos de entrada centrados y normalizados. Una normalización habitual es el z-score, que implica restarle la media y dividir por el desvío a cada feature de mi dataset.

Dado un dataset X de n muestras y m features, implementar un método en numpy para normalizar con z-score. Pueden utilizar np.mean() y np.std().

Ejercicio #5 | Remover filas y columnas con NaNs en un dataset

Dado un dataset, hacer una función que, utilizando numpy, filtre las columnas y las filas que tienen NaNs.

Ejercicio #6 | Reemplazar NaNs por la media de la columna.

Dado un dataset, hacer una función que utilizando numpy reemplace los NaNs por la media de la columna.

Ejercicio #7 | Dado un dataset X separarlo en 70 / 20 / 10

Como vimos en el ejercicio integrador, en problemas de Machine Learning es fundamental que separemos los datasets de n muestras, en 3 datasets de la siguiente manera:

- **Training dataset:** los datos que utilizaremos para entrenar nuestros modelos. Ej: 70% de las muestras.
- **Validation dataset:** los datos que usamos para calcular métricas y ajustar los hiperparámetros de nuestros modelos. Ej: 20% de las muestras.
- **Testing dataset:** una vez que entrenamos los modelos y encontramos los hiperparámetros óptimos de los mísmos, el testing dataset se lo utiliza para computar las métricas finales de nuestros modelos y analizar cómo se comporta respecto a la generalización. Ej: 10% de las muestras.

A partir de utilizar np.random.permutation, hacer un método que dado un dataset, devuelva los 3 datasets como nuevos numpy arrays.

Ejercicio # 8 - Integrador Clase #1 y Clase #2

Aplicar todo lo visto en clase a un ejercicio de reducción de dimensionalidad y clusterización básico.

1. Utilizar un dataset de su preferencia, puede ser de su proyecto final u otro.
2. Cambiar algunos puntos de manera aleatoria y agregar NaN (0.1% del dataset).
3. Guardar el dataset en un .pkl
4. Cargar el dataset con Numpy desde el .pkl
5. Completar NaN con la media de cada feature.
6. Calcular la norma l2, la media y el desvío de cada feature con funciones numpy vectorizadas.
7. Aplicar PCA al dataset reduciendo a M dimensiones.
    a. Analizar la contribución de cada componente.
    b. Realizar un scree plot.
    c. Graficar el cluster en 2 y 3 dimensiones.
    d. A través de análisis estadísticos, sacar conclusiones respecto de los resultados de PCA.
8. Hacer la clusterización con el k-means desarrollado en clase.
9. Volver a graficar el cluster con lo obtenido en (8) y comparar resultados con (7).

Bibliografía

- The Elements of Statistical Learning | Trevor Hastie | Springer
- An Introduction to Statistical Learning | Gareth James | Springer
- Deep Learning | Ian Goodfellow | https://www.deeplearningbook.org/
- Stanford | CS229T/STATS231: Statistical Learning Theory | http://web.stanford.edu/class/cs229t/
- Mathematics for Machine Learning | Deisenroth, Faisal, Ong
- Artificial Intelligence, A Modern Approach | Stuart J. Russell, Peter Norvig