

Data Preparation



Dr. José Ramón Iglesias

DSP-ASIC BUILDER GROUP

Director Semillero TRIAC

Ingeniería Electrónica

Universidad Popular del Cesar

Regularización - Optimización y Feature Engineering



EN ESTA CLASE

01 Regularización

Definición, Motivación, Ridge, Lasso y ElasticNet

02 División del Conjunto de Datos

Train, Validación, Test y Cross - Validation

03 Optimización

Hiperparámetros, Búsqueda de Grilla, Búsqueda Aleatoria

04 Feature Engineering

Datos faltantes, Bining, Transformaciones, Agrupación, Escalado

05 Pipelines

Pipeline en Sklearn y pre-procesamiento



01

Regularización

Definición, Motivación, Ridge, Lasso y
ElasticNet

Formulación del Problema

El valor esperado del error de un modelo se puede descomponer en tres elementos

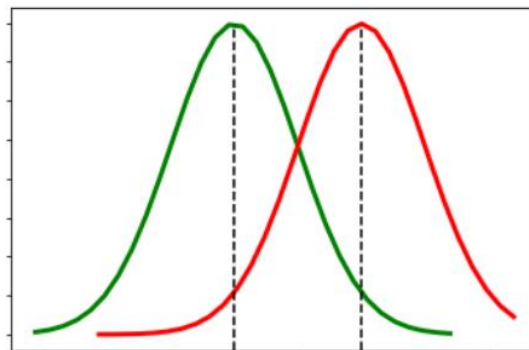
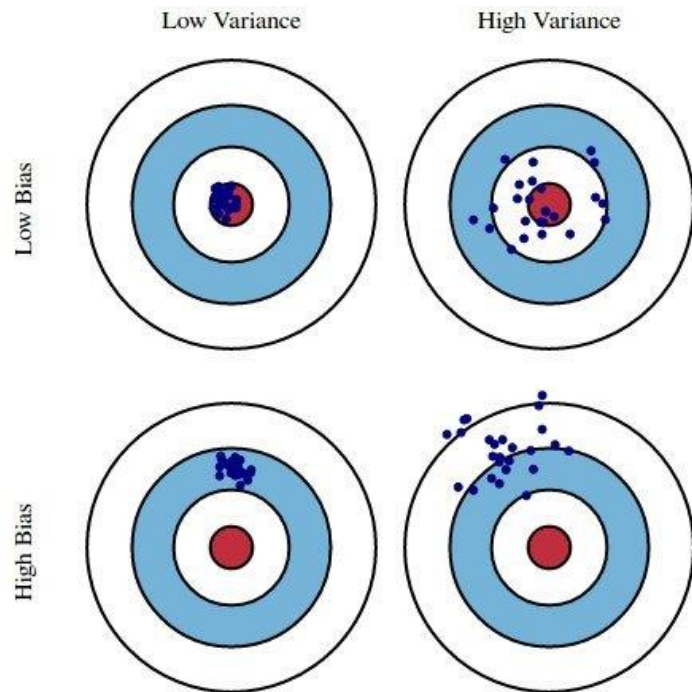
$$E(\epsilon) = \underbrace{E(\hat{Y} - Y)^2}_{\downarrow} + \underbrace{E(\hat{Y} - E(\hat{Y}))^2}_{\downarrow} + \sigma^2$$
$$E(\epsilon) = \text{Sesgo}^2 + \text{Varianza} + \sigma^2$$

$$Y = \sum_{i=1}^n \sum_{j=1}^p \beta_j \cdot x_{ij}$$

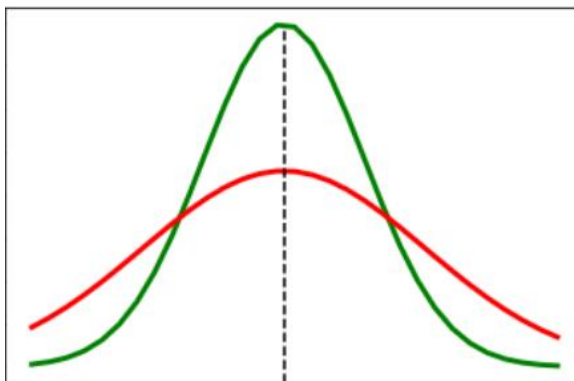
$$\hat{Y} = \sum_{i=1}^n \sum_{j=1}^p \hat{\beta}_j \cdot x_{ij}$$

Por propiedad de los estimadores, el estimador OLS es [el mejor estimador insesgado](#). Sin embargo, esto no garantiza que no tenga alta varianza

Formulación del Problema



Sesgo



varianza

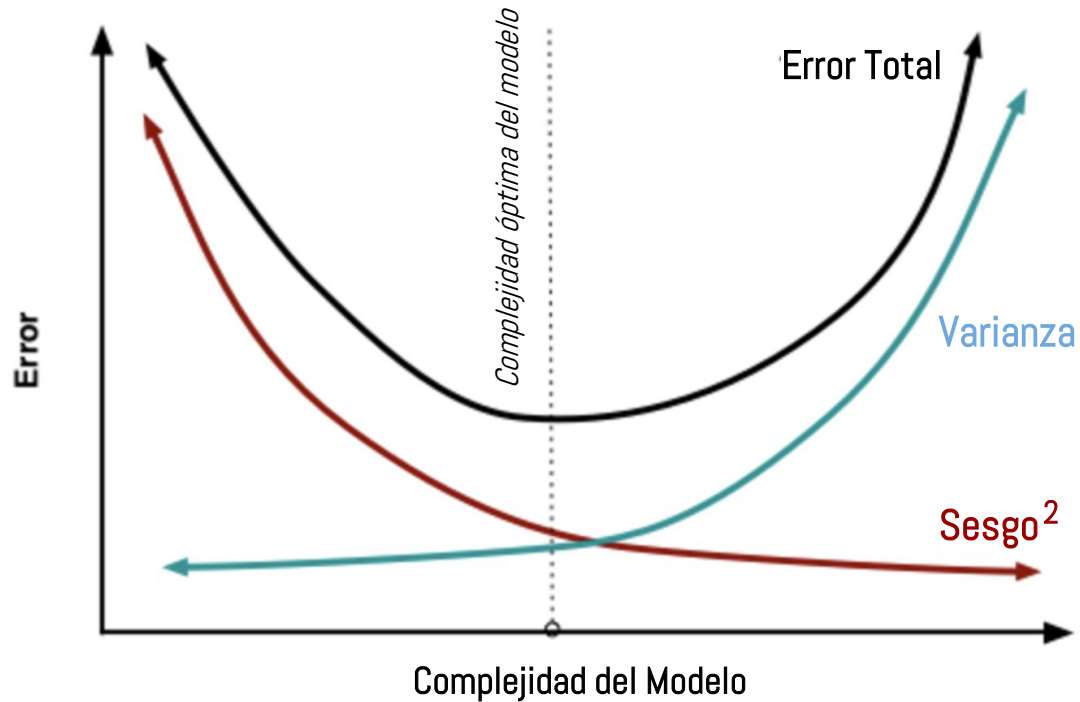
Formulación del Problema

Causas que originan estimadores con alta varianza:

- Hay presencia de multicolinealidad de los estimadores
- Hay demasiados estimadores en relación a la cantidad de observaciones y el modelo se vuelve muy complejo

Solución: Reducir la varianza de los estimadores, logrando reducir el error del modelo a costa de resignar un poco de sesgo. A esto se lo llama Regularización

Formulación del Problema



Ridge

La regularización **Ridge** se utiliza para reducir la complejidad de los modelos. Para lograrlo se agrega a la función de error del modelo un parámetro(λ) que **penaliza el tamaño de los estimadores**, forzándolos a tender hacia el cero

$$\hat{\beta}_{ridge} = \arg \min_{\beta} \underbrace{\sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2}_{\text{Función de Error}} + \underbrace{\lambda}_{\text{Parámetro de Penalización}} \underbrace{\sum_{j=1}^p \beta_j^2}_{\text{Penalización cuadrática (L2)}}$$

Ridge

$$\lambda \rightarrow 0 \Rightarrow \hat{\beta}_{ridge} \rightarrow \hat{\beta}_{OLS}$$

$$\lambda \rightarrow \infty \Rightarrow \hat{\beta}_{ridge} \rightarrow 0$$

Conclusión

$$\hat{\beta}_{ridge} = \arg \min_{\beta} \underbrace{\sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2}_{\hat{\beta}_{OLS}} + \underbrace{\lambda \sum_{j=1}^p \beta_j^2}_{Sesgo}$$

Ridge

Corolario

$$\exists \lambda \text{ tal que } MSE_{ridge} < MSE_{OLS}$$

La **disminución de la varianza** que se obtiene mediante la regularización **Ridge** más que compensa el **aumento del sesgo**

Lasso

Lasso a diferencia de Ridge tiene una penalización mediante el **valor absoluto** de los coeficientes. Lo que genera que para cierto valores de λ algunos parámetros tome valor **igual** a cero. Por este motivo se puede utilizar como **selector de variables**

$$\hat{\beta}_{lasso} = \arg \min_{\beta} \underbrace{\sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2}_{\text{Función de Error}} + \lambda \underbrace{\sum_{j=1}^p |\beta_j|}_{\substack{\text{Parámetro de} \\ \text{Penalización}} \quad \text{Penalización lineal (L1)}}$$

Elastic Net

Elastic Net surge de la combinación de las regularizaciones Ridge y Lasso. Mediante el parámetro α (alpha) se controla el tipo de penalidad del modelo.

$$ECM + \lambda[\alpha \sum_j |\beta_j| + (1 - \alpha) \sum_j \beta_j^2]$$

$$\alpha = 0 \Rightarrow \textit{Ridge}$$

$$\alpha = 1 \Rightarrow \textit{Lasso}$$

Regularización

Beneficios

1 Previene el Overfitting

2 Controla el desbalance entre la cantidad de observaciones y de variables

3 Selector automático de variables (Lasso)

4 Reduce la multicolinealidad (Ridge)

Puntos de Atención

1 Reduce la varianza del modelo a costa de incorporar cierto nivel de sesgo

2 No es invariante a la escala de las variables. Siempre se requiere estandarizar



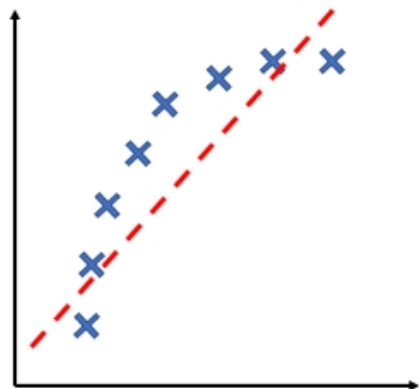
02

Train - Test - Cross Validation

Train, Validación, Test y Cross -
Validation

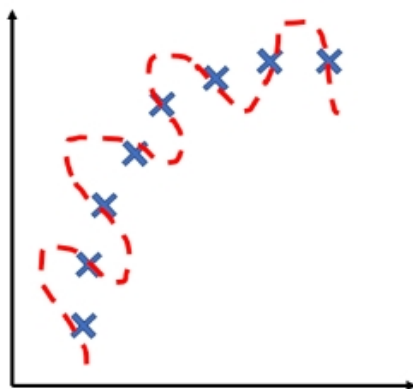
Overfitting - Underfitting

Underfitting



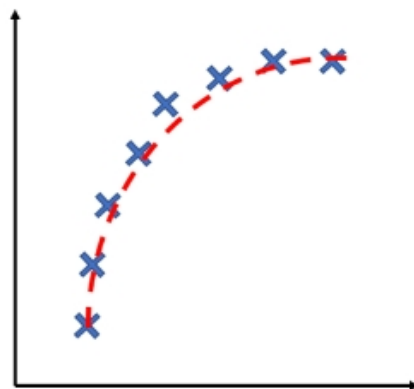
- Mal ajuste
- Baja generalización

Overfitting



- Buen ajuste
- Baja generalización

Buen Ajuste

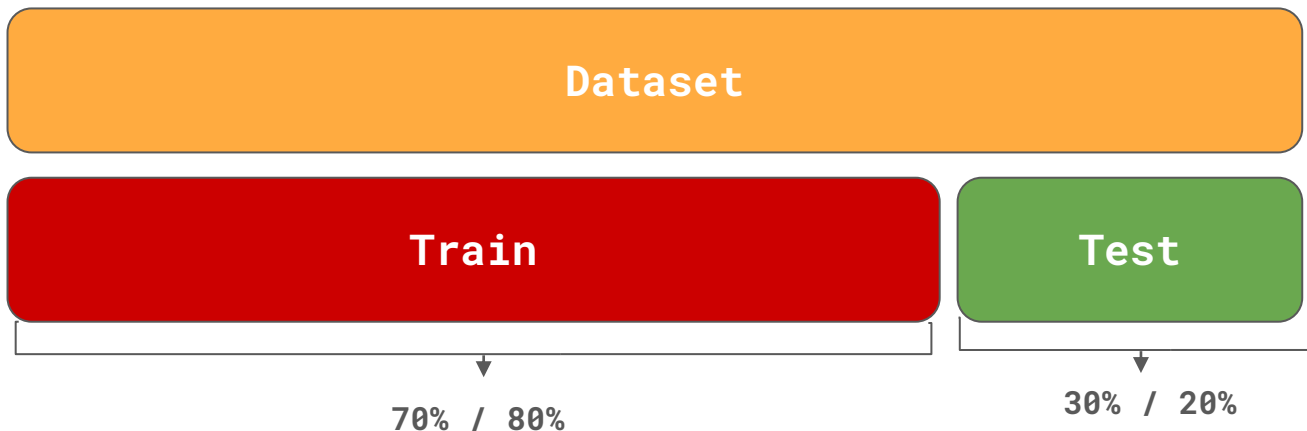


- Buen ajuste
- Buena generalización

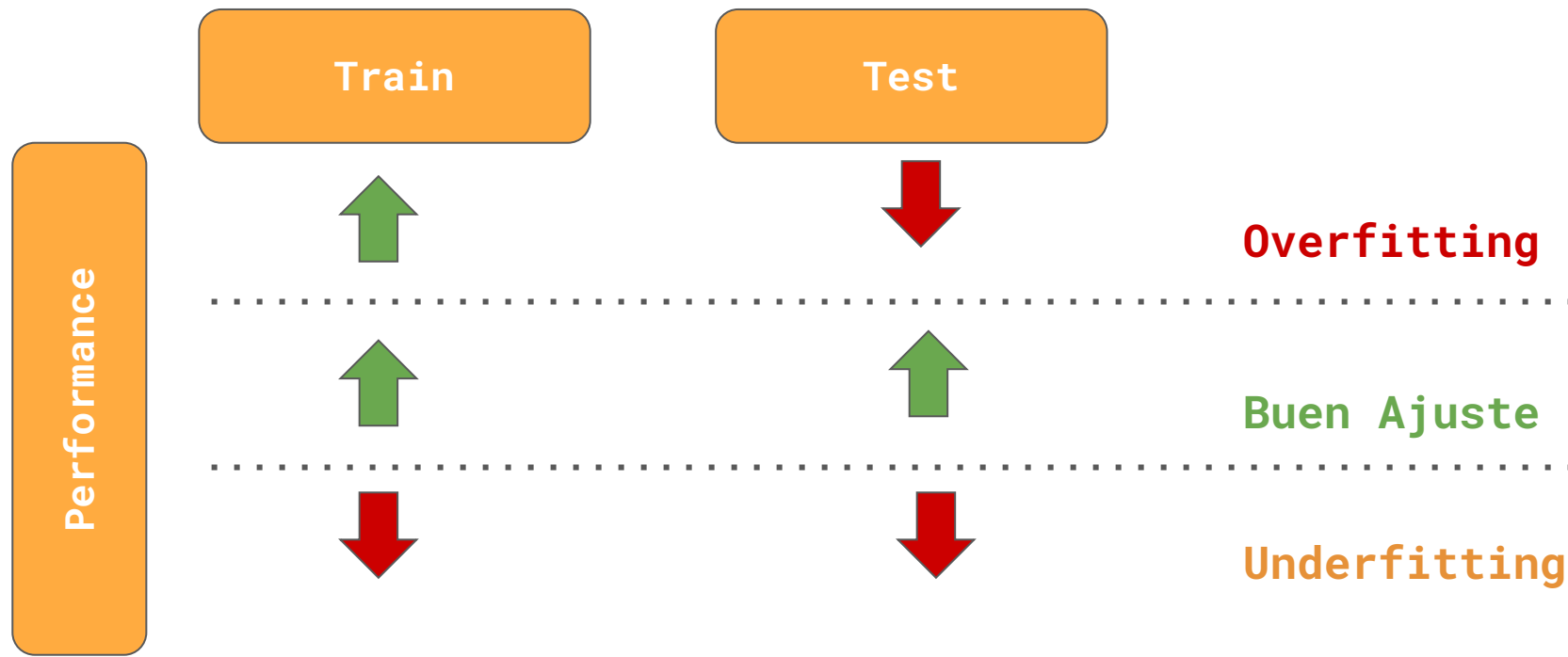
Train - Test

Para evaluar la capacidad de generalizar del modelo se puede separar el conjunto de datos original en **Train** y **Test**

- **Train:** subconjunto de datos con el cual se entrena el modelo
- **Test:** subconjunto de datos con el cual se evalúa el modelo



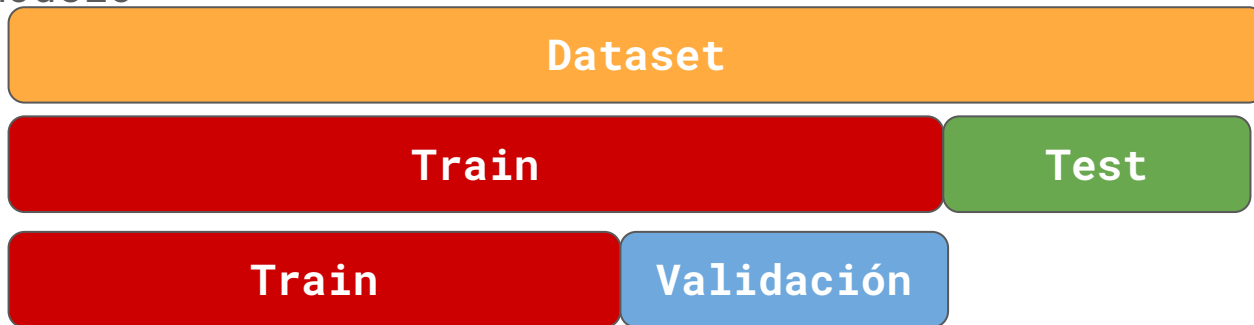
Train - Test



Train - Validación - Test

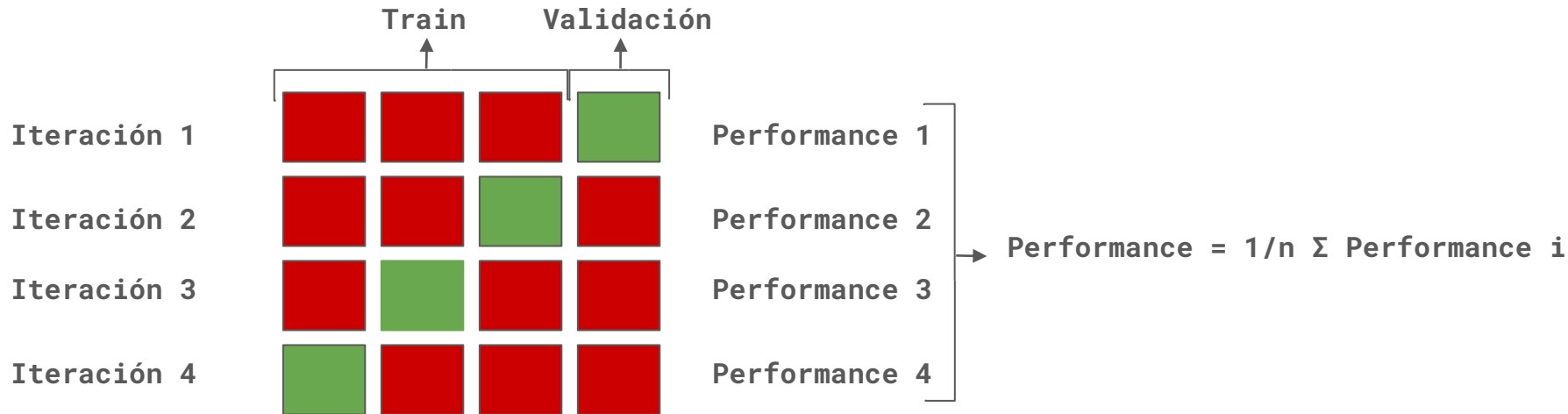
Para evaluar la capacidad de generalizar del modelo se puede separar el conjunto de datos original en **Train** y **Test**

- **Train:** subconjunto de datos con el cual se entrena el modelo
- **Validación:** subconjunto con el cual se realiza la búsqueda hiperparametros
- **Test:** subconjunto de datos con el cual se evalúa el modelo



Cross - Validation

Otra forma de ajustar nuestro modelo y evitar el overfitting es mediante el **Cross-Validation**. Es un proceso iterativo en el cual el dataset se divide en k partes (folds) y el modelo se entrena con k-1 conjuntos y se evalúa con el restante. El proceso se repite k veces, generando k estimaciones del error para luego promediarlas





03

Hiperparametros

Hiperparametros, Búsqueda de Grilla,
Búsqueda Aleatoria

Tipos de Parámetros

- **Parámetros del Modelo:** Son obtenidos durante el proceso de ajuste / entrenamiento del modelo. En el caso de las regresiones van a ser los β
- **Hiperparámetros:** Se establecen de forma previa al ajuste del modelo mediante conocimiento experto o a través de técnicas de búsqueda. Tienen una influencia directa en el posterior ajuste del modelo. Para las técnicas de regularización estos son λ (intensidad de la penalización) y α (tipo de penalización)

Train

Parámetros
 β

Validación

Hiperparámetros
 λ y α

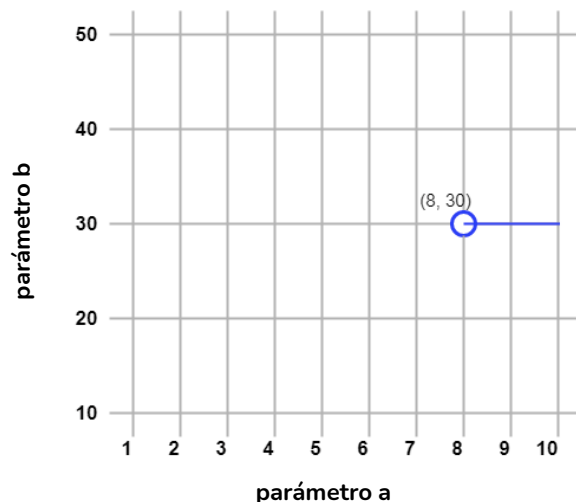
Test

Evaluación
ECM

Optimización Hiperparametros

Las distintas combinaciones de posibles hiperparametros van a generar lo que se conoce como **espacio de búsqueda**. En donde cada dimensión va a representar a un hiperparámetro distinto y cada punto va a ser una configuración distinta del modelo

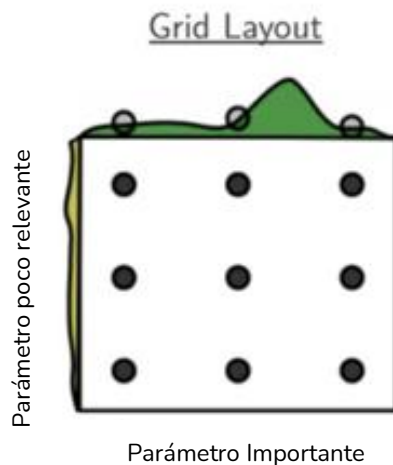
Ejemplo de un Espacio de Búsqueda de 2D



Grid Search

Es una estrategia de búsqueda en donde el usuario define una grilla con los valores que desea evaluar para cada parámetro. La cantidad de iteraciones a probar viene dada por el combinatorio resultante de la cantidad de valores que se desea evaluar para cada parámetro.

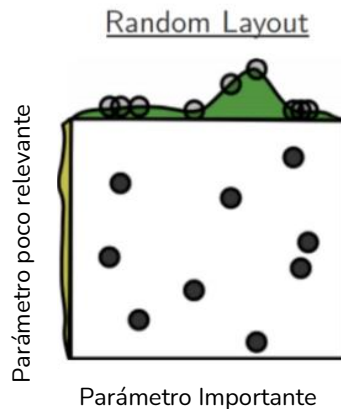
- Requiere cierto conocimiento previos
- Elevado costo computacional para evaluar muchas combinaciones



Random Search

En este caso el usuario define lo límites entre los que quiere que se busque cada parámetro y método va a computar distintas combinaciones aleatorias entre ellos. El usuario **debe definir** la cantidad de iteraciones a evaluar

- Genera una elevada probabilidad de encontrar combinaciones óptimas
- Su uso es recomendable cuando se desean evaluar una elevada cantidad de parámetros





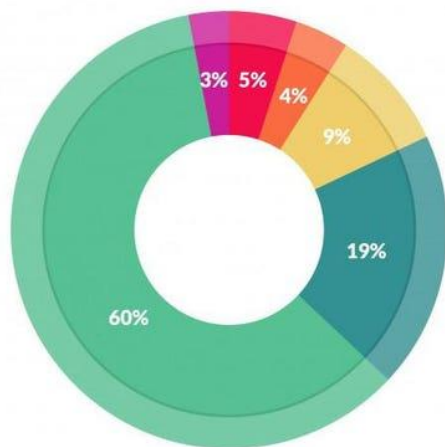
04

Feature Engineering

Datos faltantes, Bining,
Transformaciones, Agrupación, Escalado

Feature Engineering

De acuerdo a un estudio realizado por la revista Forbes, un Data Scientist pasa alrededor de un 80% del tiempo de un proyecto dedicado exclusivamente a la preparación de los datos. Esto se puede deber tanto a la obtención de los mismos como a su limpieza y tratamiento



What data scientists spend the most time doing

- Building training sets: 3%
- Cleaning and organizing data: 60%
- Collecting data sets: 19%
- Mining data for patterns: 9%
- Refining algorithms: 4%
- Other: 5%

Feature Engineering

El **Feature Engineering** consiste una serie de procedimientos y/o técnicas que se aplican sobre las distintas variables del conjunto de datos. Las transformaciones a aplicar cobran relevancia porque:

- Ayudan a preparar el conjunto de datos a ser utilizados por los algoritmos. Por ejemplo hay librerías y algoritmos que no pueden trabajar con datos faltantes
- Enriquecer el conjunto de datos para obtener un mejor rendimiento de los algoritmos que vayan a ser utilizados. Hay ciertos modelos que performan peor si las variables tienen distintas magnitudes o si las variables categóricas tienen mucha cardinalidad

Datos Faltantes

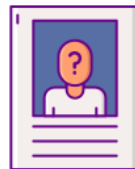


A la hora de trabajar con registros con datos faltantes es importante intentar conocer la causa de los mismos. Estos se pueden deber a problemas en la extracción de datos de otras fuentes o problemas cuando se realizó la recolección.

Estrategias

1. Borrado
 - a. Registros: eliminar aquellos registros que presenten al menos una columna sin datos. **ATENCIÓN!**: esto puede llevarnos a quedarnos con muy pocos registros
 - b. Columnas: eliminar aquellas columnas que poseen un porcentaje elevado de datos faltantes. Se suele usar un umbral de 70%
1. Imputación: consiste en completar los campos sin datos con algún tipo de valor

Datos Faltantes - Imputación



Numéricas

- **Con ceros:** completar los datos faltantes imputando el valor 0
 - Puede alterar la distribución de los datos
- **Media:** imputar calculando la media de cada columna
 - Hay pérdida de información por pérdida de variabilidad
- **Media condicionada:** calcula la media condicionada a otra variable (Ej. Salario promedio según la posición)

Categóricas

- **Nueva Categoría:** se crea una nueva categoría que identifique al campo (Ej: 'Sin Datos')
- **Categoría Mayoritaria:** se imputa con la categoría que más apariciones tiene en cada columna

- **Regresión:** se puede ajustar una regresión en función de las variables con las cuales se cuentan datos para calcular el valor faltante

Binning



Es una estrategia que se utiliza para agrupar en **bins** los registros en función de cierto criterio. Con esto se resigna variabilidad en las variables pero se gana robustez en el modelo general.

Numéricas

- **Deciles:** se agrupan los datos en deciles y los valores pertenecientes al mismo decil tomarán como valor en número del decil
- **Según la variable:** de acuerdo de que variable se trata se puede seguir la lógica de la variable
 - Años en décadas
 - Rango de pesos
 - etc.

Categóricas

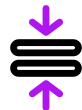
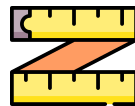
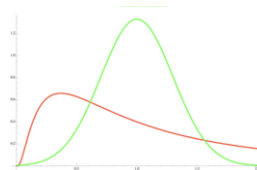
- **Baja frecuencia:** aquellas categorías con baja frecuencia pueden ser agrupadas en una nueva categoría llamada 'Otras'
- **Según la variable:** de acuerdo de que variable se trata se puede conseguir una agrupación más general
 - provincias → países
 - modelo de autos → marca del auto
 - etc.

Transformación Logarítmica



Es una de las transformaciones más comunes que se realizan a las variables numéricas. El logaritmo es una función matemática que cuenta con innumerables cantidad de propiedades, en particular nos interesan:

- Ayuda a que una distribución sesgada en una que se aproxime más a una normal
- En ciertos casos igual diferencia de magnitud no representa lo mismo (Ej. estudiar 2hs luego de haber estudiado 4hs respecto de haber estudiado 20hs). El logaritmo transforman y normaliza las diferencias de magnitud
- Suaviza el efecto de los outliers en el modelo a partir de la normalización de magnitud, lo que hace al modelo más robusto



IMPORTANTE:

La función logaritmo solo puede tomar como input valores mayores a 0

Otras transformaciones

Transformaciones Polinómicas

Consiste en elevar a la potencia que deseemos alguna variable categórica. Esto nos permite crear nuevas variables a partir de la existente. La transformación polinómica modifica la distribución de probabilidad de la variable separando las observaciones con valores bajos respecto de las que tienen valores altos. La separación incrementa con el valor del exponente

$$salary = \beta_0 + \beta_1 \cdot mp_per_g + \beta_2 \cdot mp_per_g^2$$

Interacción

En este caso generaremos una nueva variable a partir de dos de las que conforman el conjunto original de datos. Es posible, por ejemplo, realizar una multiplicación entre ellas

$$salary = \beta_0 + \beta_1 \cdot mp_per_g + \beta_2 \cdot mp_per_g \cdot ast_per_g$$

One-hot-encoding



Dada una variable categórica con N filas y M categorías, mediante el **One-hot-encoding** la variable se va a expresar como una matriz de N filas por M columnas (NxM). En donde cada columna representa una de las categorías de la variable original y están codificadas con los valores 1 y 0.

Jugador	Posicion
1	PG
2	SG
3	SF
4	PF
5	C



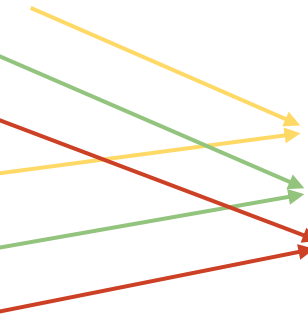
Jugador	PG	SG	SF	PF	C
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1

Agrupación - Group By

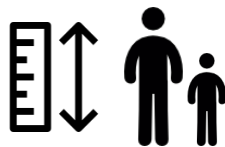
Mediante la **agrupación** se puede generar nuevas variables resumiendo de la información de distintas observaciones y agrupandolas según algún criterio que surte a partir de otra/otras variables. Usualmente es utilizado para las variables numéricas y las medidas de resumen más usuales son:

- *Media*
- Máximo
- Mínimo
- Suma

Jugador	Posición	Salario		Posición	Salario Promedio
1	PG	100.000		PG	$(100.000 + 75.000) / 2$ 87.500
2	SG	150.000		SG	$(150.000 + 125.000) / 2$ 137.500
3	SF	25.000		SF	$(25.000 + 50.000) / 2$ 37.500
4	PG	75.000			
5	SG	125.000			
6	SF	50.000			



Escalado de Datos



Hay ciertos algoritmos que se ven severamente afectados por la diferencia en escala de las variables (Ej Altura vs Salario). Para esto es necesario reescalar las variables para que todas estén en el mismo orden de magnitud

Normalización

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- Escala los valores entre 0 y 1
- No modifica la distribución de los datos
- Amplifica la magnitud de los outliers debido a que baja de los desvíos standard

Estandarización

$$z = \frac{x - \mu}{\sigma}$$

- Escala los valores a partir de la media (μ) y el desvío (σ) de la distribución
- Suaviza el efecto de los outliers

Fechas



A partir de una columna que sea una fecha se pueden realizar al menos tres transformaciones que nos van a permitir extraer más información:

- Descomponer la fecha en sus distintos elementos y asignarlos a distintas variables. Por ejemplo se podría crear una columna para el *día*, *mes*, *año*, *hora*, etc.
- Crear nuevas variables a partir de diferencias de tiempo y expresarlo en la unidad que se necesite (año, días, etc.)
- Extraer features a partir de la fecha. El nombre del día, si es alguna fecha especial del estilo feriado nacional, Navidad, etc., si es fin de semana o no, etc.



05

Pipelines

Algunas definiciones básicas

Pipelines

De acuerdo a la [documentación](#) de scikit learn un **Pipeline** es:

‘Una lista de transformaciones aplicadas de forma secuencial sobre el estimador. Los pasos intermedios deben ser “transforms” es decir, implementan los métodos fit y transform. El estimador final sólo necesita implementar el fit’

Beneficios

- Permite ensamblar varias operaciones y aplicarlas en un solo estimador
- Garantiza la ejecución ordenada de los distintos pasos brindándole reproducibilidad al proceso de estimación
- Previene el **data leakage** que se puede producir cuando se pre procesan los datos de train y test
 - Ej. Escalar los datos del conjunto de train y aplicar cross-validation

Pipelines

Se deben expresar los pasos del pipeline como una lista de tuplas, donde los elementos de la tupla son el nombre del proceso y la función que se desea aplicar. El orden en que se presentan las tuplas define el orden de ejecución del pipeline

```
#Agregamos un imputador con una estrategia de media
pasos = [('imputer', SimpleImputer(strategy='mean')), ('scaler', StandardScaler()), ('enet', ElasticNet())]
```

```
# Definimos el objeto que va a contener nuestro pipeline
pipeline = Pipeline(pasos)
```

```
# Como vamos a evaluar una búsqueda de grilla definimos los parametros a evaluar
#debemos poner el nombre del estimador seguido de __ y luego el parametro
parametros = {'enet__alpha':[0.001,0.01,0.1,1,10,100],
              'enet__l1_ratio':[0,0.25,0.5,0.75,1]}
```

```
# Instanciamos la búsqueda de grilla pasandole el pipeline que deseamos que ejecute

grid = GridSearchCV(pipeline, param_grid=parametros, cv=20, scoring= 'neg_mean_squared_error')
```

```
# Ajustamos nuestro modelo al conjunto de entrenamiento
grid.fit(X_train, y_train)
```

Pipelines - Transformaciones más usuales

- **StandardScaler()**: Ejecuta una estandarización restando a las observaciones la media y dividiendo por el desvío
- **SimpleImputer()**: Permite imputar los valores faltantes con distintas estrategias (media, mediana, valor más frecuente)
- **OrdinalEncoder()**: permite codificar las variables numéricas en una lista ordenada de enteros
- **OneHotEncoder()**: codifica las variables categóricas con la estrategia One-hot encoding
- **ColumnTransformer()**: permite segmentar los transformadores que se aplican a las variables (Ej. transformadores para variables categóricas por un lado y numericas por el otro)