

# Minería de Datos

UTDT

Master in Management + Analytics

*Sesión 3*

# ¿Qué vimos hasta ahora?

- Qué es la minería de datos.
- Distintos tipos de aprendizaje automático.
- Aprendizaje supervisado.
- Naïve bayes.
- K-nearest neighbors.
- Noción de overfitting & underfitting
- Selección de modelos
- Árboles de decisión

# Caret

Caret es una librería que facilita entrenar y validar modelos de predicción en R de manera simple.

Veamos el primer bloque de código de esta clase para ver cómo se usa.

De paso vamos a generar un archivo para hacer un submit en Kaggle.

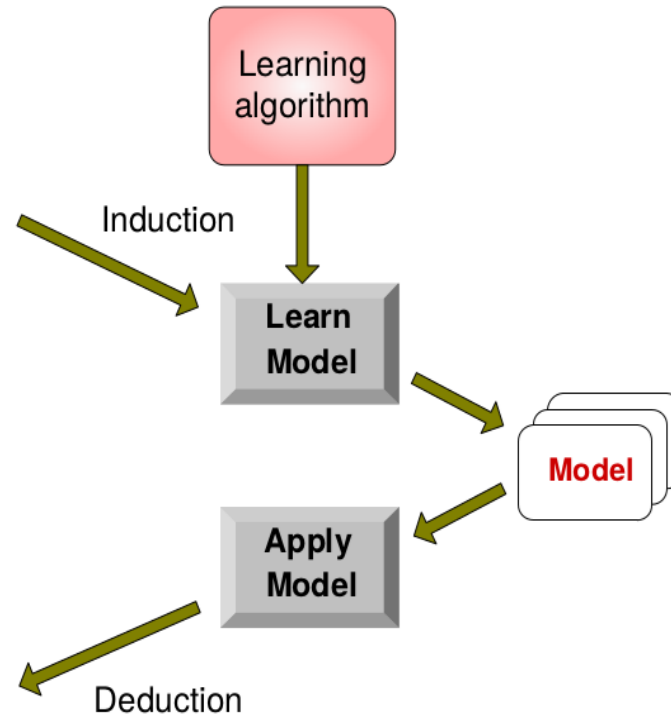
# Aprendizaje supervisado

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

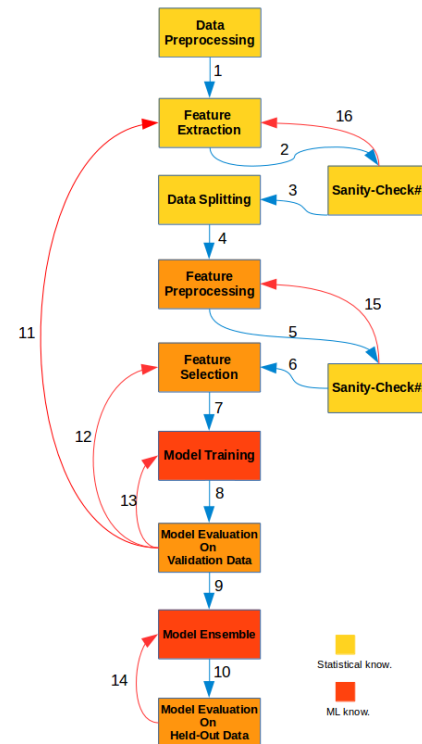
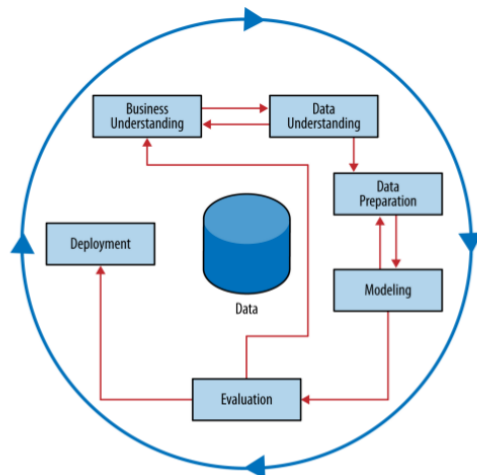
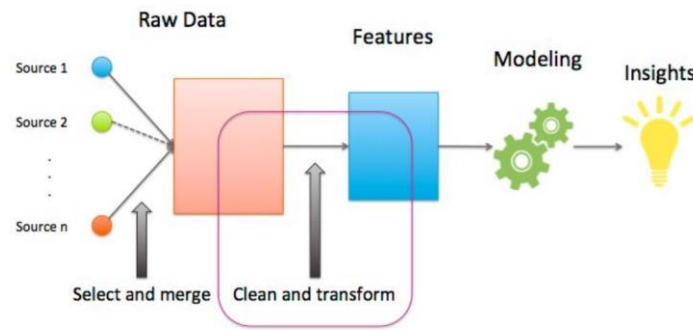
Test Set



¿Qué es nuestro “modelo”?

# Feature Engineering

No importa qué fuente tomemos para esquematizar el proceso aprendizaje, siempre se va a mencionar que **generar atributos para entrenar el modelo de aprendizaje es una etapa importante** (*garbage in - garbage out*).



# Feature Engineering

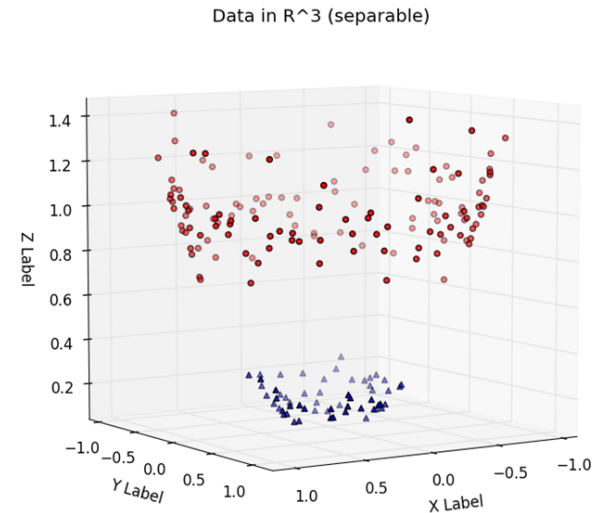
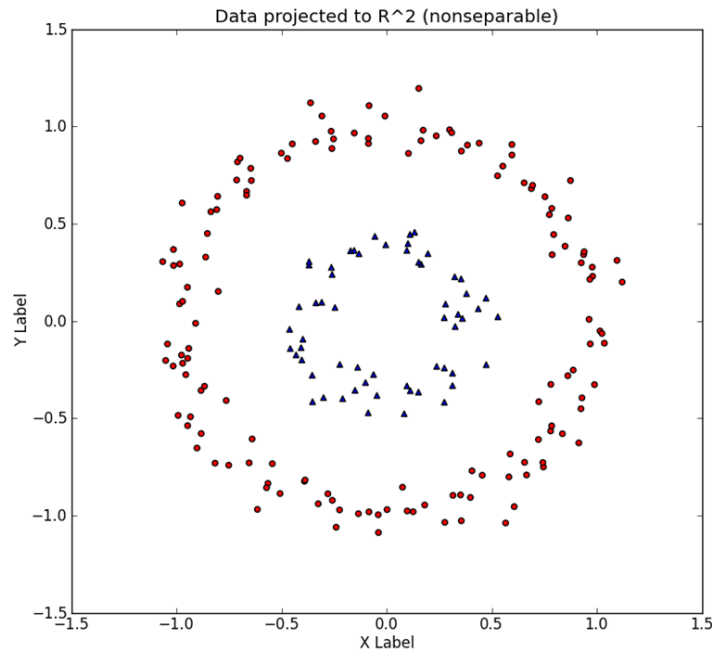
Lo que aprenda el algoritmo de aprendizaje dependerá en buena medida de los atributos de los que disponga para aprender.

Cuando uno ve las soluciones ganadoras en competencias, ve que una porción importante de la descripción de las soluciones se enfoca en el análisis exploratorio de datos y en la "ingeniería de atributos":

- *Santander Product Recommendation*.
- *Two Sigma Financial Modeling Code*.
- *Rossmann Store Sales* (*"I spent 50% on feature engineering, 40% on feature selection plus model ensembling, and less than 10% on model selection and tuning."*).

# Feature Engineering

“Good features should not only represent salient aspects of the data, but also conform to *assumptions of the model*. Hence transformations are often necessary.” (Zheng & Casari, Cap. 2) [Yo agregaría también de la implementación del algoritmo]



$$(z = x^2 + y^2)$$

# Feature Engineering

Ahora veremos técnicas tradicionales de **ingeniería de atributos**. Dependiendo del dominio y del análisis exploratorio de los datos, pueden surgir nuevas ideas.

La clave es **primero disponer de un sistema de evaluación de modelos** (*validation set*, *k-fold cross-validation*... Lo que sea que replique bien datos "desconocidos"), que nos permita evaluar si las nuevas variables creadas mejoran o no la performance del modelo.

Las técnicas tradicionales varían en función de si se trata de atributos **numéricos** o **categoricos**.



# Feature Engineering - Variables numéricas

## Log transformation:

Muchas veces las variables siguen una **distribución asimétrica positiva**. En donde la gran masa de los datos se encuentra en los valores pequeños, mientras que valores más grandes tienen poca densidad. En esos casos puede ser razonable que el modelo trabaje con la variable en **escala logarítmica**.

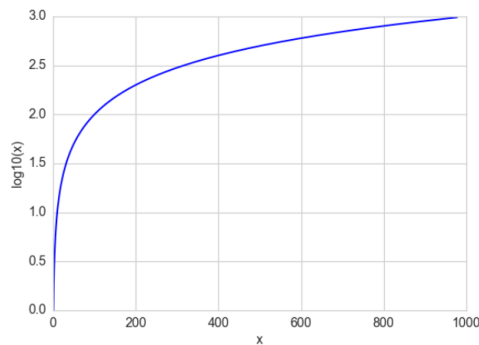
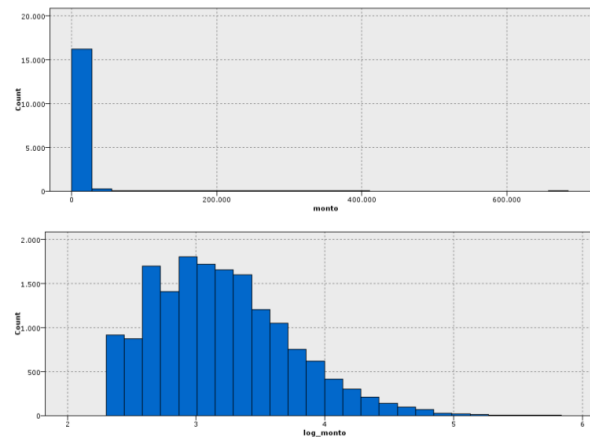


Figure 2-6. The log function compresses the high numeric range and expands the low range. Note how the horizontal  $x$  values from 100 to 1000 got compressed into just 2.0 to 3.0 in the vertical  $y$  range, while the tiny horizontal portion of  $x$  values less than 100 are mapped to the rest of the vertical range.



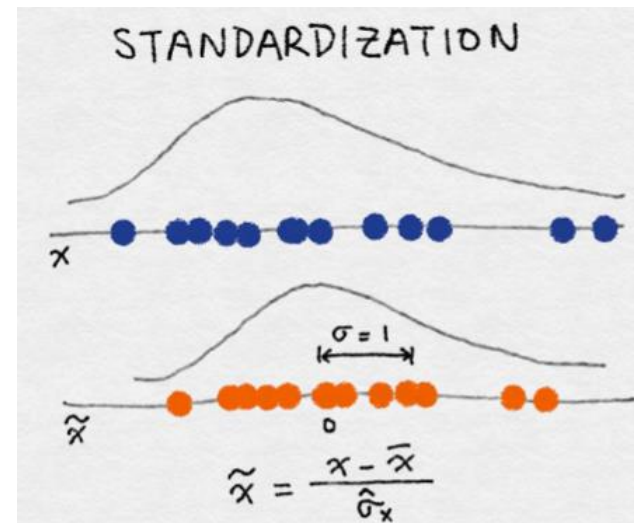
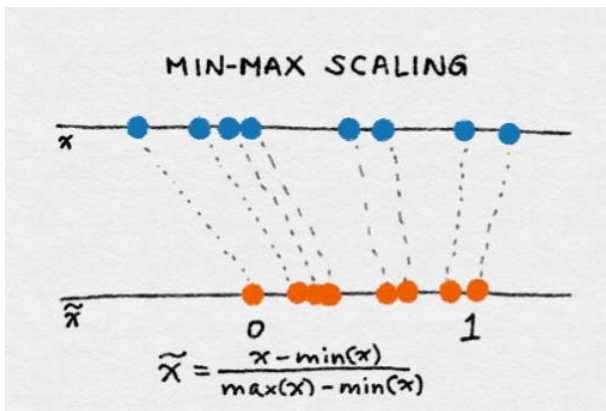
Si la variable original tiene valores negativos una opción es transformarla con la siguiente expresión  $x_{log} = \log(x + 1 - \min(x))$ .

¿Esto sólo importa en vecinos más cercanos?

# Feature Engineering - Variables numéricas

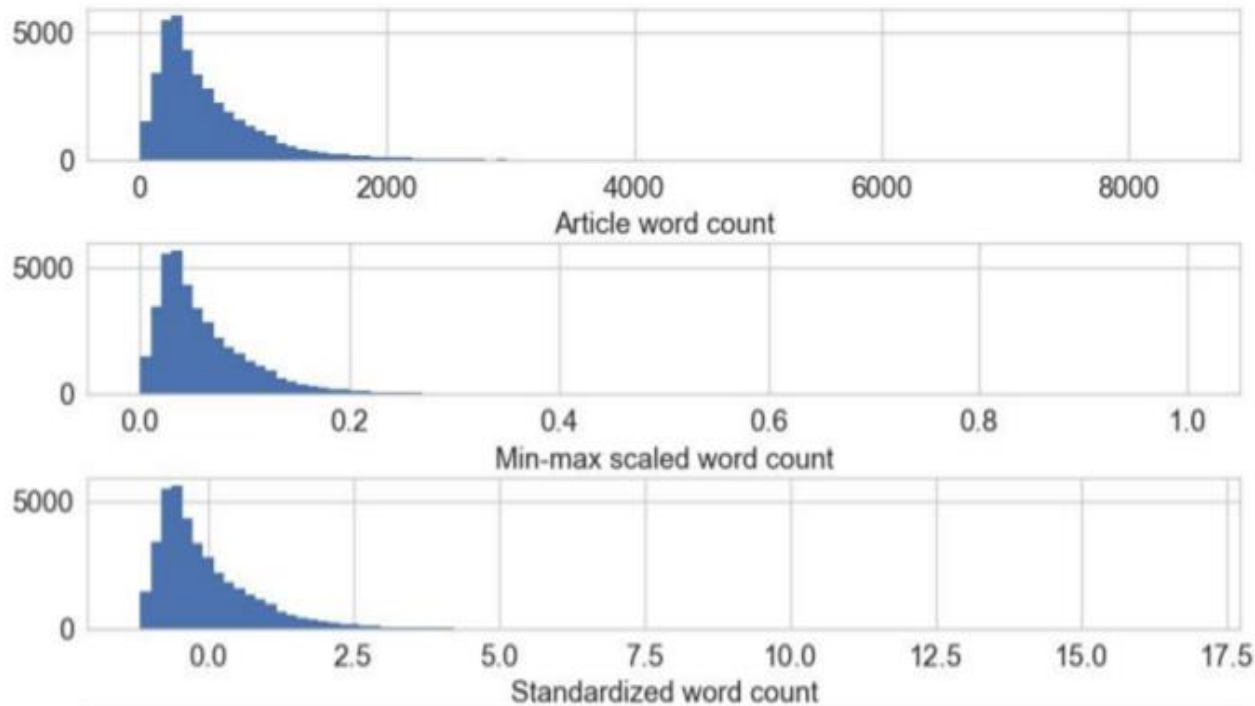
## *Feature Scaling or Normalization (variance scaling):*

Muchos modelos son sensibles a la escala (varianza) de los atributos (por ej.: los que trabajan con distancias).



# Feature Engineering - Variables numéricas

Noten que estas estrategias para normalizar/escalar variables **no modifican la forma de la distribución** (sólo cambian los valores de los ejes).



# Feature Engineering - Variables numéricas

Probemos el segundo bloque de código.

En el mismo se predice si una empresa entrará en bancarrota. Usaremos el dataset en su totalidad (noten que la clase a predecir es bastante desbalanceada).

¿Parece buena decisión escalar las variables?

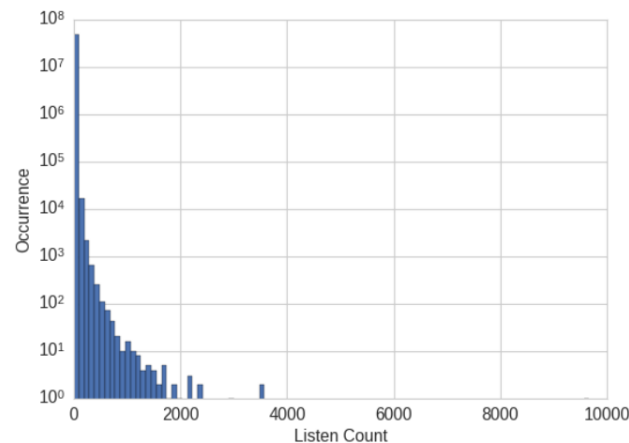
¿Mejora la performance sólo tomando logaritmos de las mismas?

¿Es aún mejor la performance si se combina tomar logaritmos con escalar variables?

# Feature Engineering - Variables numéricas

## *Binarization:*

Hay situaciones donde podría sólo interesar si el valor de una variable numérica es **mayor a una constante  $c$**  (por ej. en conteos donde predominan los ceros, ver si algo es mayor a 0).



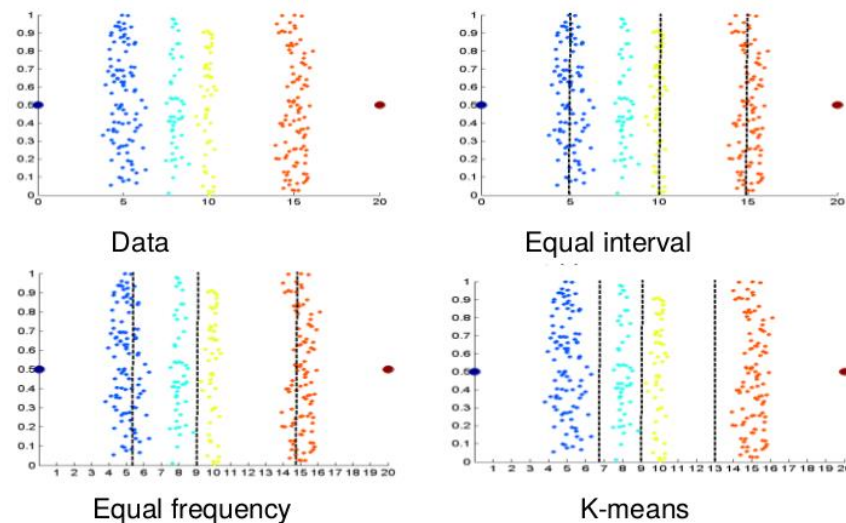
*Figure 2-3. Histogram of listen counts in the user taste profile of the Million Song Dataset.  
Note that the y-axis is on a log scale.*

Se puede crear una nueva variable que valga 1 si la original es mayor a  $c$  y 0 en caso contrario (pudiendo la variable creada reemplazar a la original).

# Feature Engineering - Variables numéricas

## *Quantization or binning:*

Consiste en **agrupar las observaciones en bins** (en donde los bins tienen un orden). Luego, opcionalmente, a estos bins se les puede poner un número que indique su orden. Existen múltiples criterios para elegir los cortes (uno totalmente válido es: "a ojo").



Muchas implementaciones de algoritmos de aprendizaje supervisado internamente hacen esto.

# Feature Engineering - Variables numéricas

Probemos el tercer bloque de código:

¿Tal cual vienen los datos, es buena la performance de Naïve Bayes?

¿Mejora la performance performance discretizando las variables? ¿Por qué cree que lo hace?

Como tarea:

- Evalúen si la performance depende del número de bins.

# Feature Engineering - Variables categóricas

## *Encoding Categorical Variables:*

No todos los algoritmos de aprendizaje incorporan formas de lidiar con atributos categóricos (como sí lo tiene árboles). En estos casos uno debe **transformar variables categóricas en variables continuas** intentando perder poca información.

## *One-hot encoding:*

Sample	Category		Human	Penguin	Octopus	Alien
1	Human	→	1	0	0	0
2	Human		1	0	0	0
3	Penguin		0	1	0	0
4	Octopus		0	0	1	0
5	Alien		0	0	0	1
6	Octopus		0	0	1	0
7	Alien		0	0	0	1

**Problemático para variables con muchos niveles.** Opciones: *hashing-trick* (verlo del libro), uso de matrices esparsas/ralas (más detalles en la próxima clase).



# Feature Engineering - Variables categóricas

## *Encoding Categorical Variables:*

Ojos con lo que hace caret cuando se usa la sintatxis de fórmulas.

```
nb_fit_form <- train(class ~ .,  
  data=data_set_disc_eq,  
  method="nb",  
  trControl = fitControl,  
  metric = "ROC")
```

¿Si la variable fuera categórica ordinal (ej. malo, bueno, regular), perdemos información usando one-hot encoding? ¿Qué alternativa tenemos?


# Feature Engineering - Otras variables derivadas

**A partir de fechas y horas:** año, mes, día, hora, minuto, día de la semana, semana del año, día desde el 1970-01-01 (tendencia), etc.

**Si los datos tienen estructura de panel:** considerar medidas móviles de variables, máximo o mínimos en periodos de tiempo.

**A partir de variables categóricas:** conteos, promedios/proporciones de  $y_i$  según el valor de una variable categórica (*bin-counting*, ¿A qué les recuerda?).

n	usuario	y
1	A	0
2	B	1
3	B	1
4	A	0
5	B	0
6	A	0
7	A	1
8	C	0
9	A	0
10	B	1



usuario_conteo	usuario_bc
5	0.2
4	0.75
4	0.75
5	0.2
4	0.75
5	0.2
5	0.2
1	0
5	0.2
4	0.75

# Feature Selection

Tres estrategias comunes:

- *Filtering*: consiste en eliminar atributos antes de entrenar por considerar tienen poca probabilidad de tener poder predictivo (ya sea por exploración manual, o por algún criterio estadístico. Ahora vamos a ver un ejemplo)
- *Wrapper methods*: consiste en probar un modelo de aprendizaje con distintas variables, medir su performance y usarla como criterio de selección.
- *Embedded methods*: consiste en que el mismo algoritmo de aprendizaje elija qué variables ignorar (árboles de decisión lo hace con naturalidad, otros modelos de regresión tipo LASSO también lo hacen).

También se pueden hacer combinaciones de estrategias, por ej.: **recursive feature elimination** (<https://topepo.github.io/caret/recursive-feature-elimination.html>).

# Feature Selection

Feature selection mediante *filtering*:

Probemos el cuarto bloque de código, en el mismo generamos datos simulados y usamos una técnica de filtering para elegir variables.



¿Cómo es posible que estemos obteniendo predicciones mejor que el azar!?

¿En qué paso nos equivocamos?

# Data Leakage

Consiste en validar y seleccionar modelos incorporando en los mismos **información que no debiera estar disponible** al momento de puesta en producción.

Cuando esto ocurre, lo común es **sobrestimar la performance del modelo propuesto**. Lo más común es que se filtre de alguna manera información de la variable a predecir desde el holdout set al training set (pero hay otras variantes).

Lo importante es que **hay que validar en situaciones comparables a las que se tendrá en producción**.

# Data Leakage

Algunos ejemplos de *data leakage*:

- Seleccionar atributos utilizando información de la variable a predecir proveniente del validation set (ejemplo visto en código).
- Escalar/Discretizar variables utilizando todo el dataset (no es tan grave si uno asume que los distribuciones se mantienen estables en testing).
- Hacer bin-counting utilizando los datos de validación.
- No trabajar bien la temporalidad de los datos. Ejemplo: si quisiera predecir el rendimiento de un activo a 5 días, no dejar fuera del análisis al menos 5 días entre training y validation.

	Entrenamiento										Validación				
Var	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14	Day 15
x1	0.017	0.014	0.014	0.016	0.006	0.003	0.002	-0.001	-0.014	-0.017	-0.017	-0.017	-0.015	-0.011	0.002
x2	1.8386	1.5950	1.9864	2.0133	1.8348	2.1233	2.0687	1.8020	2.2015	1.9766	1.9605	1.8175	2.1286	1.8712	1.5672
y	-84%	-86%	-110%	-193%	-401%	-711%	-1017%	1006%	-24%	-110%	-90%	-89%	-161%	-227%	896%

$$y(\text{day}_t) = x_1(\text{day}_{t+5}) / x_1(\text{day}_t) - 1$$

# Data Leakage

Algunos ejemplos de *data leakage*:

- Seleccionar atributos utilizando información de la variable a predecir proveniente del validation set (ejemplo visto en código).
- Escalar/Discretizar variables utilizando todo el dataset (no es tan grave si uno asume que los distribuciones se mantienen estables en testing).
- Hacer bin-counting utilizando los datos de validación.
- No trabajar bien la temporalidad de los datos. Ejemplo: si quisiera predecir el rendimiento de un activo a 5 días, no dejar fuera del análisis al menos 5 días entre training y validation.

	Entrenamiento										Validación				
Var	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14	Day 15
x1	0.017	0.014	0.014	0.016	0.006	0.003	0.002	-0.001	-0.014	-0.017	-0.017	-0.017	-0.015	-0.011	0.002
x2	1.8386	1.5950	1.9864	2.0133	1.8348	2.1233	2.0687	1.8020	2.2015	1.9766	1.9605	1.8175	2.1286	1.8712	1.5672
y	-84%	-86%	-110%	-193%	-401%	-711%	-1017%	1006%	-24%	-110%	-90%	-89%	-161%	-227%	896%

$$y(\text{day}_t) = x_1(\text{day}_{t+5}) / x_1(\text{day}_t) - 1$$

# Data Leakage

Algunos ejemplos de *data leakage*:

- Seleccionar atributos utilizando información de la variable a predecir proveniente del validation set (ejemplo visto en código).
- Escalar/Discretizar variables utilizando todo el dataset (no es tan grave si uno asume que los distribuciones se mantienen estables en testing).
- Hacer bin-counting utilizando los datos de validación.
- No trabajar bien la temporalidad de los datos. Ejemplo: si quisiera predecir el rendimiento de un activo a 5 días, no dejar fuera del análisis al menos 5 días entre training y validation.

	Entrenamiento										Validación				
Var	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14	Day 15
x1	0.017	0.014	0.014	0.016	0.006	0.003	0.002	-0.001	-0.014	-0.017	-0.017	-0.017	-0.015	-0.011	0.002
x2	1.8386	1.5950	1.9864	2.0133	1.8348	2.1233	2.0687	1.8020	2.2015	1.9766	1.9605	1.8175	2.1286	1.8712	1.5672
y	-84%	-86%	-110%	-193%	-401%	-711%	-1017%	1006%	-24%	-110%	-90%	-89%	-161%	-227%	896%

$$y(\text{day}_t) = x_1(\text{day}_{t+5}) / x_1(\text{day}_t) - 1$$



# Medidas de performance en clasificación

Naïve Bayes:

$$Pr( C_k | X_i ) \propto ( Pr( v_1 = x_{i1} | C_k ) * Pr( v_2 = x_{i2} | C_k ) * \dots * Pr( v_q = x_{iq} | C_k ) ) * Pr( C_k ) )$$

Vecinos más cercanos:

$$Pr(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

Árboles de decisión:

$$Pr(Y = j | X = x_0) = \frac{1}{|R_{x_0}|} \sum_{l \in R_{x_0}} I(y_l = j)$$

¿Estimamos probabilidades condicionales, pero es esto lo que finalmente nos interesa?

# Medidas de performance en clasificación

¿Qué decisión tomarían en cada uno de estos casos?

$$p(y1 = \text{baja} \mid x1) = 0.14$$

$$p(y2 = \text{baja} \mid x2) = 0.87$$

		Predicted class		Total instances
		+	−	
Actual class	+	TP	FN	P
	−	FP	TN	N

(una valor **prob\*** da lugar a una matriz de confusión)

¿Cómo evaluamos si el resultado fue bueno?

Necesitamos alguna métrica. Una primera opción sería evaluar en base a *accuracy*:  $(TP+TN) / (P+N)$ .

¿Qué sucede si las clases son muy desbalanceadas?

# Medidas de performance en clasificación

Imaginemos distintas situaciones:

- Nuestra prioridad es no gastar innecesariamente plata y mandar la oferta sólo a clientes para los que estemos seguros que servirá  $\rightarrow \uparrow \text{prob}^*$
- Nuestra prioridad podría ser captar la mayor cantidad clientes que tomarían el servicio  $\rightarrow \downarrow \text{prob}^*$

		Predicted class		Total instances
		+	−	
Actual class	+	TP	FN	P
	−	FP	TN	N

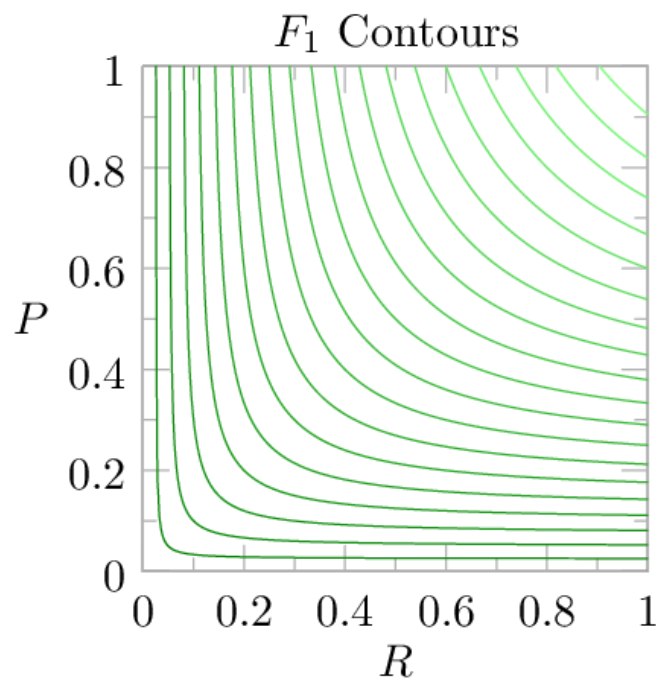
Uno puede captar estas ideas en la matriz de confusión:

- *Precision*: de los que dije que son + cuántos lo son.  $TP / (TP+FP)$ .
- *Recall*: de los que son + cuántos dije que son +.  $TP / (TP+FN)$ .

Noten que entre estas medias existe un trade-off (piensen en casos extremos:  $\text{prob}^*=0$  y  $\text{prob}^*=1$ ).

# Medidas de performance en clasificación

Una medida comúnmente usada es el *f1-score*. El mismo trata de nivel ponderar de manera conjunta precision y recall.



$$\text{f1-score} = (2 * \text{prec} * \text{rec}) / (\text{prec} + \text{rec})$$

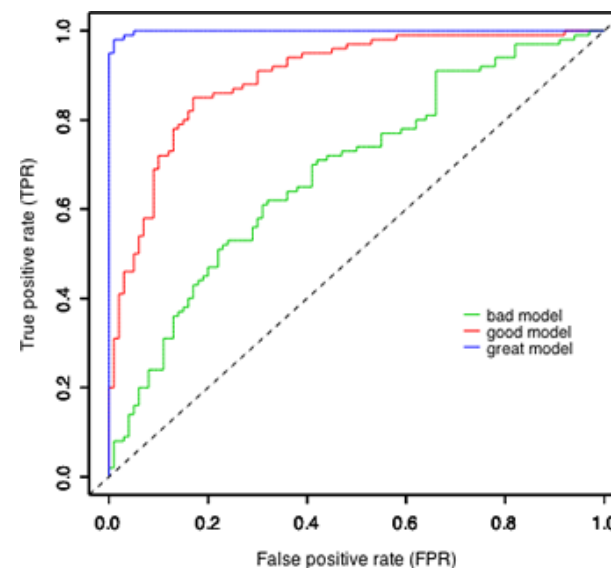
f1-score tiende a estar cerca del mínimo entre precision y recall.

# Medidas de performance en clasificación

Existen medidas que tratan de independizarse de  $prob^*$ . Una métrica popular se deduce de la curva ROC ([ejemplo interactivo](#)).

Relación entre:

- True Positive Rate (TP | P).
- False Positive Rate (FP | N).



Se toma el **área bajo de la curva ROC** (AUC) como un indicador de la capacidad de predicción del clasificador.

# Medidas de performance en clasificación

Supongamos que tengamos el siguiente escenario:

Matriz de confusión				Matriz de costos			
		Predicho				Acción	
		no baja	baja			no oferta	oferta
Actual	no baja	7713	219	Actual	no baja	0	120
	baja	600	479		baja	1000	120

El costo de esta campaña sería de \$683,760.

Enviar la oferta a todos costaría \$951,840.

No enviarle la oferta a nadie costaría \$1,079,000.

¿Podremos incorporar esta información a los fines de mejorar nuestra decisión?

# Medidas de performance en clasificación

Supongamos que tenemos el siguiente escenario:

		Matriz de costos	
		Acción	
		no oferta	oferta
Actual	no baja	0	120
	baja	1000	120

El costo esperado de hacer la oferta (A) es:  $120 * \text{prob}^{\wedge} + 120 * (1 - \text{prob}^{\wedge})$

El costo esperado de no hacer la oferta (B) es:  $1000 * \text{prob}^{\wedge} + 0 * (1 - \text{prob}^{\wedge})$

Conviene mandar la oferta cuando  $A < B$ , es decir:

$$120 * \text{prob}^{\wedge} + 120 * (1 - \text{prob}^{\wedge}) < 1000 * \text{prob}^{\wedge}$$
$$\text{prob}^{\wedge} > 0.12$$

# Medidas de performance en clasificación

Probemos en el último bloque de código cómo se comportan distintas métricas de performance.



# Medidas de performance en clasificación

Conclusion:

Le métrica de interés depende en gran medida del problema en cuestión.

Lo que sí es buena práctica es **utilizar una única métrica** para guiar las decisiones.

En determinada ocasiones se puede modificar el algoritmo de manera que tenga como objetivo minimizar la métrica de costo que uno proponga (por ej: maximizar el recall sujero a una precisión mayor al 60%).

# Lecturas recomendadas para los temas vistos hoy

Ingeniería de atributos:

- Feature Engineering for Machine Learning (Cap 2 y 5)

Métricas de performance:

- Tan (Sección 5.7)

# Práctica de laboratorio

Para hacer:

Vean la resolución subida de la práctica de la sesión anterior (la que usa los datos "Adult"). Usando Caret entrene modelos "rpart" y "nb". Pruebe en cada uno cómo impactan los siguientes preprocesamientos que abajo se listan. Utilicen AUC como métrica de performance y k-fold cross validation con 5 folds.

Antes de hacerlos, piense si cree que deberían afectar la performance de estos algoritmos para bien, para mal o en lo absoluto.

Preprocesamientos:

- Discretizar todas las variables continuas utilizando intervalos con la misma frecuencia de observaciones.
- Hacer one-hot-encoding en todas las variables categóricas.
- Pasar toda las variables continuas a escala de logaritmo con base 10 .

# Práctica teórica

¿Usted espera que discretizar variables continuas utilizando un número ridículamente grande de intervalos afecte en gran medida cómo predicen los árboles de decisión? Justifique su respuesta.

Estudie por su cuenta qué es hacer oversampling de clases minoritarias. ¿Cree que existe alguna manera de usar mal esta técnica y dar lugar a data leakage?

¿Es cierto que valores altos de AUC (e.g., 0.99) generan siempre valores altos de accuracy? Justifique su respuesta.

Suponga que en el TP usted está interesado en que aquellos para los que predice churn efectivamente lo hagan, ¿en qué considera más relevante enfocarse, en precision o en recall?