TDVI: Inteligencia Artificial

Métricas de Performance

UTDT - LTD

Estructura de la clase

- Motivación
- Métricas de performance para regresión
- Métricas de performance para clasificación

Estructura de la clase

- Motivación
- Métricas de performance para regresión
- Métricas de performance para clasificación

Motivación

¿Qué queremos que "arroje" el modelo en cada caso?

Regresión

- Queremos un valor predicho de y_i (i.e., $\hat{y}_i$) que sea lo más cercano posible a lo que efectivamente valdrá y para la observación i

Clasificación

- Queremos que si y_i es igual a k, entonces $\hat{P}(y_i = k \mid X_i)$ sea lo más cercana a 1 (y, por ende, lo más cercana a 0 para las restantes clases)
- Sobre la base de los valores $\hat{P}(y_i = k \mid X_i)$ para todo $k \in K$ decidiremos qué clase predecir para la observación i

Vamos a utilizar métricas para ver medir qué tan bien se está logrando este objetivo

Motivación

Las métricas pueden (y suelen) cumplir dos roles:

1. Cuantificar la performance del modelo en distintos conjuntos de datos (training, validation, testing, producción, etc.)

2. Guiar el aprendizaje/entrenamiento de ciertos modelos. E.g.:
   - En árboles de decisión aplicados a regresión guiamos la construcción de los mismos minimizando el SSE sobre training
   - Regresión logística se busca minimizar *cross-entropy* en training

No es cierto las métricas utilizadas para guiar el aprendizaje y para evaluar un modelo deban ser las mismas, pueden diferir

Por ejemplo: regresión logística (un clasificador) se entrena minimizando cross-entropy, pero puede ser que se evalúe su performance con ROC-AUC

La selección de una buena métrica para evaluar un modelo es algo importante (dependerá del problema a atacar)

Estructura de la clase

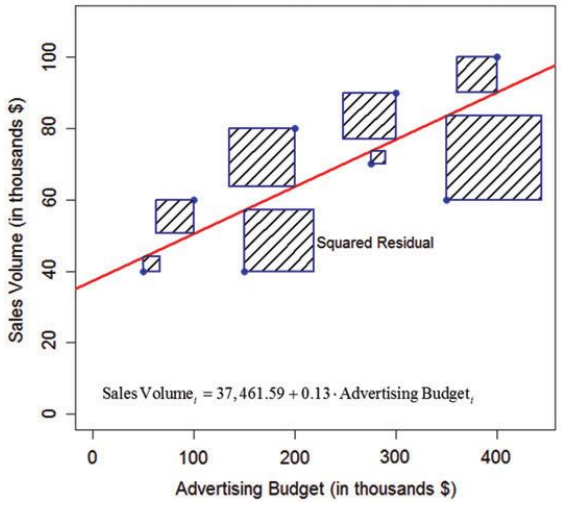
- Motivación
- Métricas de performance para regresión
- Métricas de performance para clasificación

Métricas de performance para regresión

Ya vimos la suma de los errores/residuos al cuadrado (*sum squares error*)

$$SSE = \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

Se puede visualizar como la suma del área de cuadrados



Noten que al considerar el cuadrado de los errores, la métrica es sensible a valores extremos de los mismos

Métricas de performance para regresión

Para que el valor del error no dependa de la cantidad de observaciones, se toma el promedio del SSE. Se obtiene el *mean squared error* (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

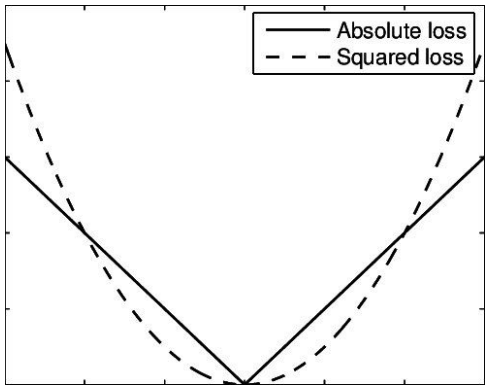
Para expresar al error en la unidad original, se suele tomar la raíz cuadrada del MSE. Se obtiene el *root mean squared error* (RMSE)

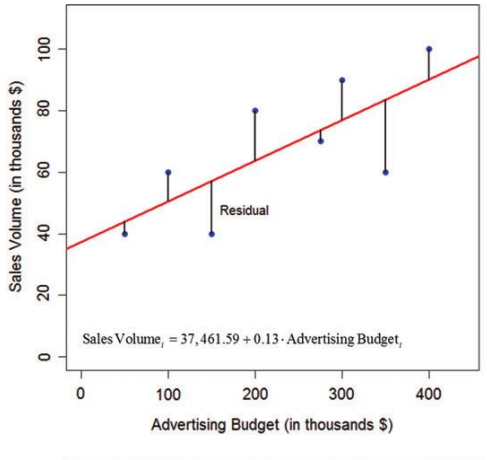
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2}$$

Métricas de performance para regresión

Para hacer al error más robusto a valores extremos, a veces se considera el valor absoluto de los errores. Al promediar dichos valores se obtiene el *mean absolute error* (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{f}(x_i)|$$





Diferencia:

- Errores absolutos. Una observación para la que se erró en 4 unidades genera el mismo error que 4 observaciones para las que se erró en 1 unidad
- Errores al cuadrado. Una observación para la que se erró en 4 unidades genera más error que 4 observaciones para las que se erró en 1 unidad (16 vs 4)

Métricas de performance para regresión

Existen múltiples métricas de errores en regresión. Generalmente, cada métrica capta alguna sutileza relevante/útil para determinados problemas de predicción

Vamos un último ejemplo: *root mean squared logarithmic error* (RMSLE)

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(ln(y_i + 1) - ln(\hat{f}(x_i) + 1))^2}$$

Propiedades:

- Dado que *ln(a) - ln(b) = ln(a/b)*, se tiene que captura algo cercano al error relativo (es más grave predecir 150 si era 100 — 0.4, que 1050 si era 1000 — 0.05)
- Penaliza de manera más fuerte predecir de menos que de más (i.e., es más grave predecir 800 si era 1000 — 0.22, que 1200 si era 1000 — 0.18)

Estructura de la clase

- Motivación
- Métricas de performance para regresión
- Métricas de performance para clasificación

Métricas de performance para clasificación

¿Qué decisión tomarían en cada uno de estos casos?

$p(y_1 = baja \mid x_1) = 0.14$

$p(y_2 = baja \mid x_2) = 0.87$

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

(un *threshold* (*t**) da lugar a una matriz de confusión)

¿Dada 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?

Métricas de performance para clasificación

Imaginemos dos escenarios referidos a enviar una oferta:

- Nuestra prioridad podría ser no gastar innecesariamente plata y mandar una oferta sólo a clientes para los que estemos seguros que servirá → ↑t^*
- Nuestra prioridad podría ser captar la mayor cantidad clientes que podrían tomar el servicio → ↓t*

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

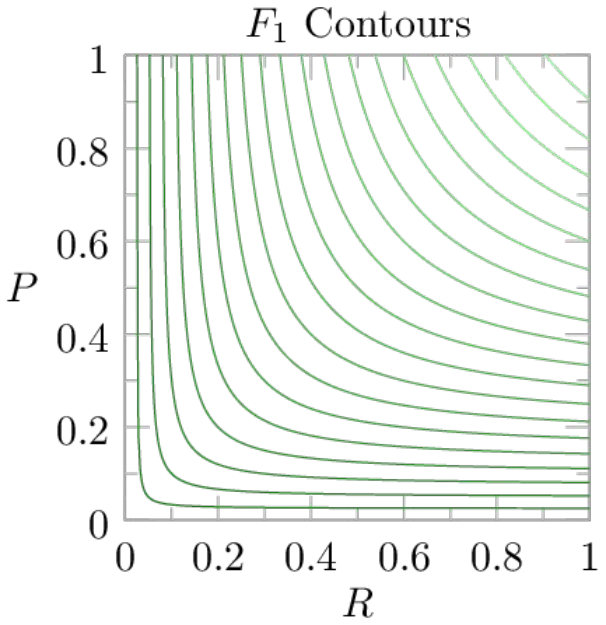
Uno puede captar estas ideas en la matriz de confusión:

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

Noten que entre estas medias existe un *trade-off* (piensen en los casos extremos t^*=0 y t^*=1)

Métricas de performance para clasificación

Una medida comúnmente usada es el *F_1-score*. La misma pondera de manera conjunta precision y recall



F_1-score = (2 * prec * rec) / (prec + rec)

F_1-score tiende a estar cerca del mínimo entre precision y recall (subir sólo precision o sólo recall no ayuda tanto como subir los dos)

Métricas de performance para clasificación

Hasta ahora todas las medidas que vimos dependen del threshold elegido (t^*)

Un modelo quizás sea bueno para un threshold dado, pero malo para otros. Esto agrega un grado más de complejidad al proceso de modelado

¿Habrá métricas que prescindan del threshold?

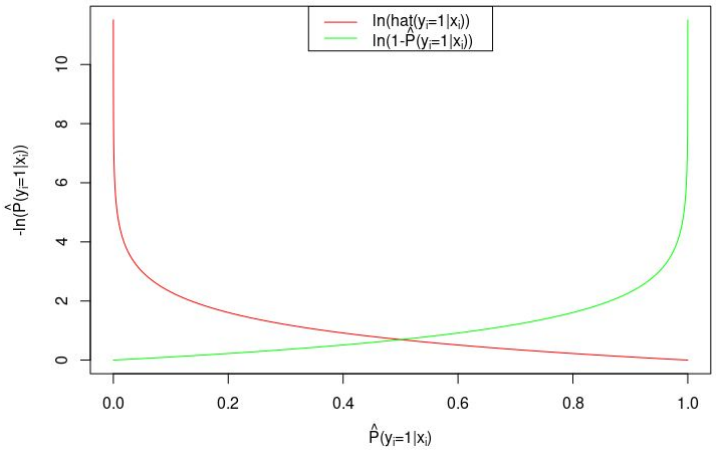
OJO: es patear el problema para adelante, en algún momento un threshold se deberá elegir

Métricas de performance para clasificación

Cross-entropy (o *Log-loss*). Se utiliza tanto para guiar la construcción de modelos (e.g., regresión logística y redes neuronales) como para evaluarlos

Para una observación *i*, toma valores altos si no se predice una alta probabilidad a la clase correcta

Para el caso binario: $$CE = -\frac{1}{n}\sum_{i=1}^{n}(y_i \cdot ln(\hat{P}(y_i = 1|x_i)) + (1 - y_i) \cdot ln(1 - \hat{P}(y_i = 1|x_i)))$$



Para problemas multiclase: $$CE = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} I(y_i = k) \cdot ln(\hat{P}(y_i = k|x_i)))$$

Métricas de performance para clasificación

Una métrica popular que no depende de t^* se deduce a partir de la curva ROC (ejemplo interactivo)

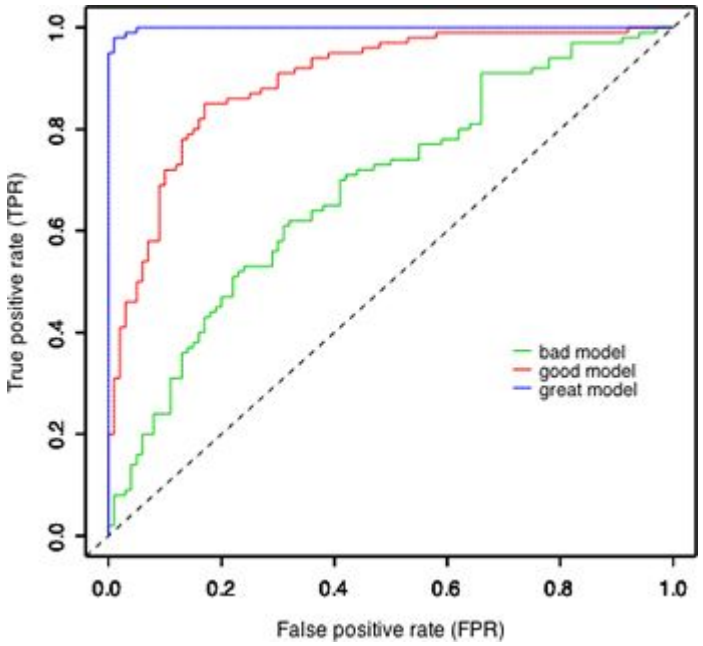
Relación entre:

True Positive Rate (TP | P, es Recall)

- De los que son + qué proporción se acierta

False Positive Rate (FP | N)

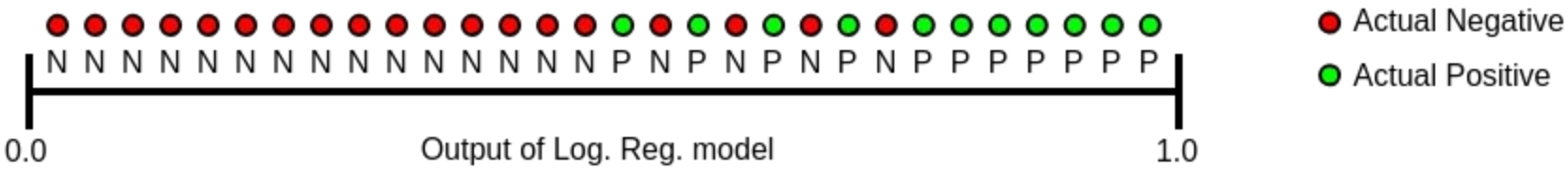
- De los que son - en qué proporción se erra



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

Métricas de performance para clasificación

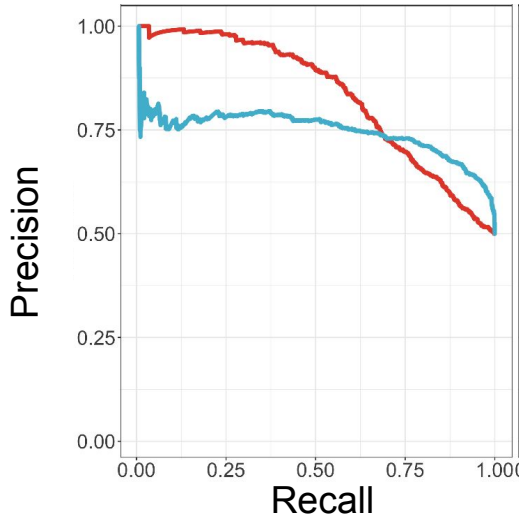
Interpretación probabilística del *ROC-AUC*



Es igual a la probabilidad de que, al elegir una observación positiva al azar y una observación negativa al azar, el modelo haya predicho una probabilidad mayor para la observación positiva que para la negativa

Métricas de performance para clasificación

Vimos que existe un trade-off entre precision y recall. Se pueden graficar los valores de precision y recall a medida que se modifica el threshold y obtener la *curva precision-recall* (PR)



Se suele tomar el área bajo esta curva (*PR-AUC*) como una métrica de performance (se la conoce también como *average precision*)

Una ventaja por sobre *ROC-AUC* es que se ve menos afectada por el desbalance de clases (*artículo de discusión*)

Métricas de performance para clasificación

Supongamos que ahora podemos asignar un costo a cada uno de los diferentes aciertos/errores

Matriz de confusión

		Predicho	
		no baja	**baja**
Actual	**no baja**	7713	219
	baja	600	479

Matriz de costos

		Decisión	
		no oferta	**oferta**
Actual	**no baja**	0	120
	baja	1000	120

- El costo de esta campaña sería de $683,760
- Enviar la oferta a todos costaría $1.081.320
- No enviarle la oferta a nadie costaría $1,079,000

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

Métricas de performance para clasificación

Supongamos que tengamos el siguiente escenario:

Matriz de costos

		Decisión	
		no oferta	**oferta**
Actual	**no baja**	0	120
	baja	1000	120

- El costo esperado de hacer la oferta (A) es: 120 * prob^ + 120 * (1-prob^)
- El costo esperado de no hacer la oferta (B) es: 1000 * prob^ + 0 * (1-prob^)

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

$$120 * prob^ + 120*(1-prob^) < 1000 * prob^$$

$$prob^ > 0.12$$

Métricas de performance para clasificación

Probemos en código cómo se calcular las distintas métricas de performance en clasificación

Bibliografía

- Zheng A., “*Evaluation Machine Learning Models*”, Capítulo 2

- Müller & Guido, "*Introduction to machine learning with Python: a guide for data scientists*”, Capítulo 5 (lo referido a métricas)

- Matrices de costos:

  https://mlr.mlr-org.com/articles/tutorial/cost_sensitive_classif.html