TDVI: Inteligencia Artificial

Selección de Atributos

UTDT - LTD

Estructura de la clase

- Motivación
- Embedded methods
- Wrapper methods
- Filter methods
- Data leakage

Estructura de la clase

- Motivación
- Embedded methods
- Wrapper methods
- Filter methods
- Data leakage

Motivación

Es común enfrentarse a conjuntos de datos con un gran número de variables predictoras

Empíricamente se observa que reducir el número de variables de manera apropiada, puede mejorar la performance predictiva de los modelos

Puede haber 1) variables redundantes (en donde una variable muestra lo mismo que otra) o 2) variables irrelevantes (se sabe que no tiene información útil; e.g., una variable que tenga los últimos 4 dígitos del dni, ¿del cuit también?)

También se puede querer reducir la cantidad de variables por cuestiones operativas (el dataset es demasiado grande, ya sea para entrenamiento o producción)

Motivación

Existen diferentes estrategias para seleccionar atributos. Se pueden catalogar en tres grupos:

- *Embedded methods*
- *Wrapper approaches*
- *Filter approaches*

Veamos cada una en detalle

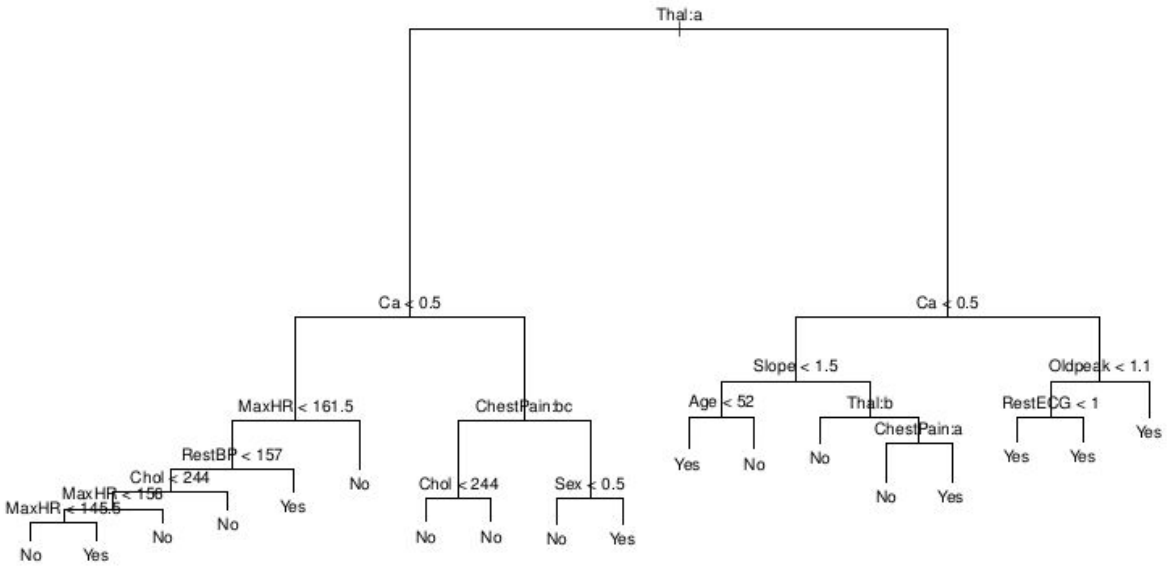
Estructura de la clase

- Motivación
- Embedded methods
- Wrapper methods
- Filter methods
- Data leakage

Embedded methods

La selección de atributos ocurre de manera natural al entrenar algoritmos. Está "incrustada" en el proceso de aprendizaje

Árboles de decisión hacen esto de manera natural



Ignorando el caso de manejo de missings con variables subrogadas, las variables no utilizadas en los splits podrían no haber estado en el conjunto de datos de entrenamiento y se obtendría el mismo árbol

Embedded methods

Veamos el caso de las regresiones *lasso* y *ridge*. Antes recordemos qué hace regresión lineal

Se asume la siguiente relación entre la variable a predecir y los atributos

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

Dado unos coeficientes estimados, se predice utilizando la siguiente expresión:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p$$

Medimos qué tan bien ajusta un modelo al conjunto de entrenamiento utilizando la suma de los errores/residuos al cuadrado

$$\sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2$$

Los valores estimados de los coeficientes serán aquellos que, para un conjunto de datos de entrenamiento, minimizan la suma de los errores al cuadrado

Embedded methods

Aún siendo un modelo simple, un modelo de regresión lineal puede hacer overfitting si hubiera muchas variables (recordar el ejemplo de underfitting y overfitting que vimos en clase)

Tampoco es un buen modelo al momento de seleccionar atributos (en este contexto consisten en estimar $\beta_j=0$ para una variable j)

Alternativa: modificar la función de costos de la siguiente manera

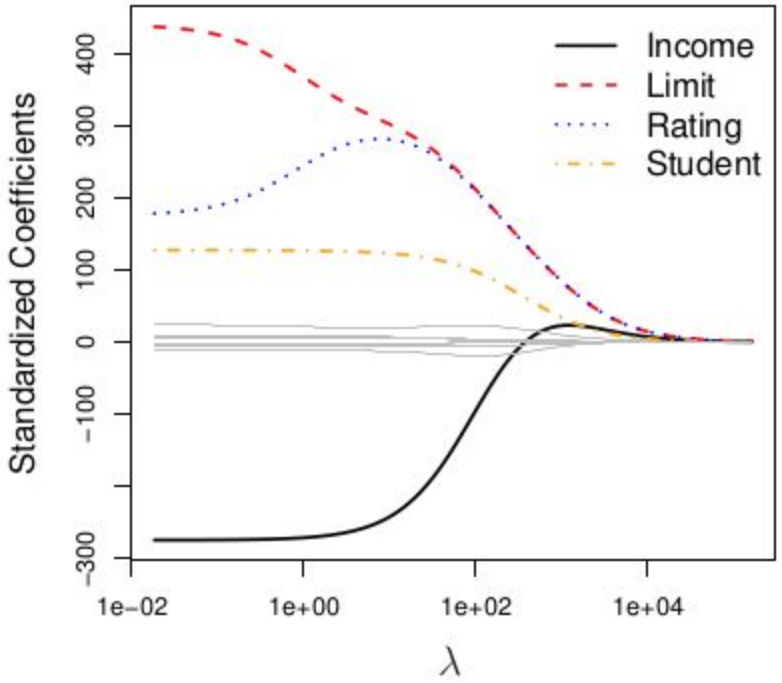
$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\mathrm{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2$$

En la nueva función de costo se penalizan altos valores de los coeficientes (salvo la constante). Este modelo se conoce como *ridge regression*

¿$\lambda=0$ y con $\lambda\to\infty$?

Embedded methods

Valor de los coeficientes estimados en función de λ



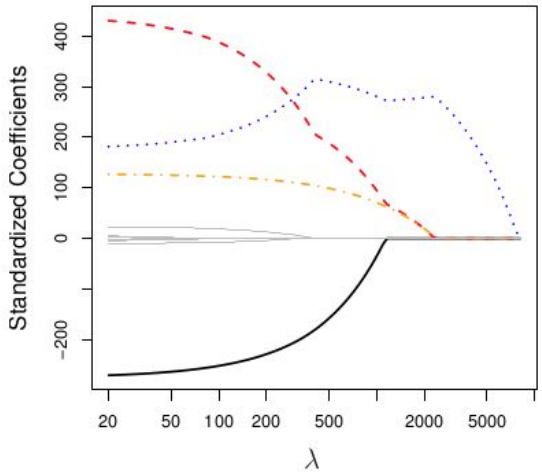
Suele ser común estandarizar las variables antes de correr rige regressión

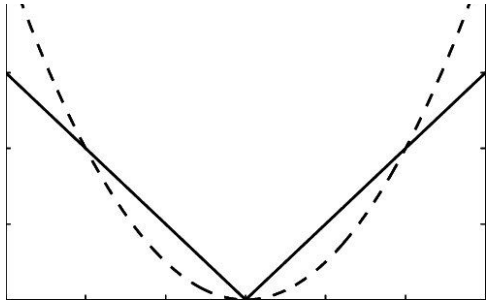
¿Por qué?

Embedded methods

Otra alternativa: penalizar el valor absoluto de los coeficientes estimados (salvo la constante). A este modelo se lo conoce como *lasso regression*

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|$$





- Ridge tiende a hacer los coeficientes más cercanos a 0, pero no igual a 0
- Lasso tiende a igualarlos a 0; es decir, hace feature selection

Embedded methods

Existen muchas variantes de penalizaciones, por ejemplo *elastic net*:

$$\lambda \sum_{j=1}^{p} \left(\alpha \beta_j^2 + (1 - \alpha) |\beta_j| \right)$$

Noten que es una combinación de *rigde* y *lasso*

Recordatorio: al proceso de quitar flexibilidad a un modelo se lo conoce como regularización. De modo, que a incorporar este tipo de penalizaciones en las funciones de costo se lo conoce como incorporar un componente de regularización

¿Habíamos visto este tipo de penalización en la materia?

Embedded methods

Veámos el primer bloque de código

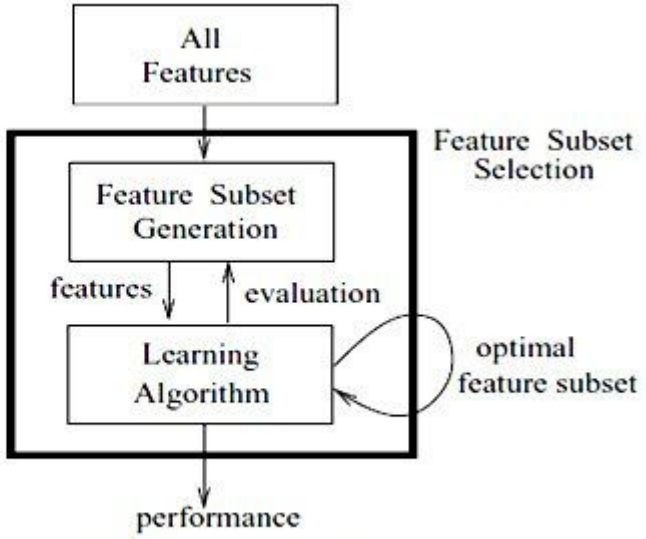
Estructura de la clase

- Motivación
- Embedded methods
- Wrapper methods
- Filter methods
- Data leakage

Wrapper methods

Estrategia:

Disponer de algún esquema para estimar la performance en datos nuevos e ir evaluando la performance distintas combinaciones de atributos (siguiendo alguna estrategia de búsqueda)



b) Wrapper method

Wrapper methods

Un caso extremo sería probar todas las combinaciones de predictores

Algorithm 6.1 *Best subset selection*

1. Let $\mathcal{M}_0$ denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, \ldots p$:
   (a) Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   (b) Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here *best* is defined as having the smallest RSS, or equivalently largest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

El libro sugiere que en el paso 2.b se use la performance sobre training. Otra opción podría ser utilizar la performance sobre validación

Muy costoso computacionalmente, ¿cuántos modelos se entrenan?

Wrapper methods

Alternativamente se puede ir agregando variables de a una a la vez

Forward selection

Algorithm 6.2 *Forward stepwise selection*

1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
   (a) Consider all $p-k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   (b) Choose the *best* among these $p-k$ models, and call it $\mathcal{M}_{k+1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Se entrenan *1 + p(p+1)/2* modelos. Es greedy, puede no encontrar la mejor combinación de variables

Wrapper methods

Alternativamente se puede ir quitando variables de a una a la vez

Backward selection

Algorithm 6.3 *Backward stepwise selection*

1. Let $\mathcal{M}_p$ denote the *full* model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   (a) Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   (b) Choose the *best* among these k models, and call it $\mathcal{M}_{k-1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

También se entrenan *1 + p(p+1)/2* modelos. No necesariamente se obtiene el mismo resultado que forward selection

Existen métodos híbridos que combinan forward y backward selection

Wrapper methods

Alternativa a backward selection tal cual la presenta el libro

Algorithm 6.3 *Backward stepwise selection*

1. Let $\mathcal{M}_p$ denote the *full* model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   (a) Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   (b) Choose the *best* among these k models, and call it $\mathcal{M}_{k-1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Modificar el paso 2 de manera que, dada una cantidad de features, se 1) entrene un único modelo que asigne importancia a los atributos y luego 2) se elimine aquel atributo con menor importancia estimada

En este caso solo se entrenan *p* modelos

Wrapper methods

Probemos el segundo bloque de código

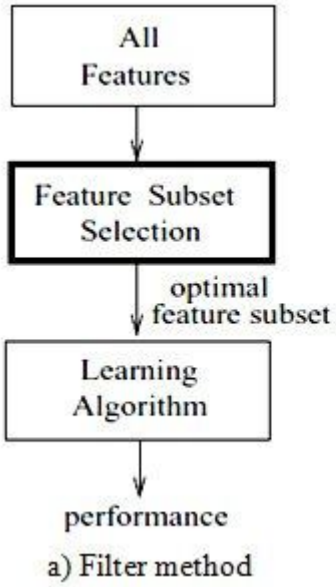
Estructura de la clase

- Motivación
- Embedded methods
- Wrapper methods
- Filter methods
- Data leakage

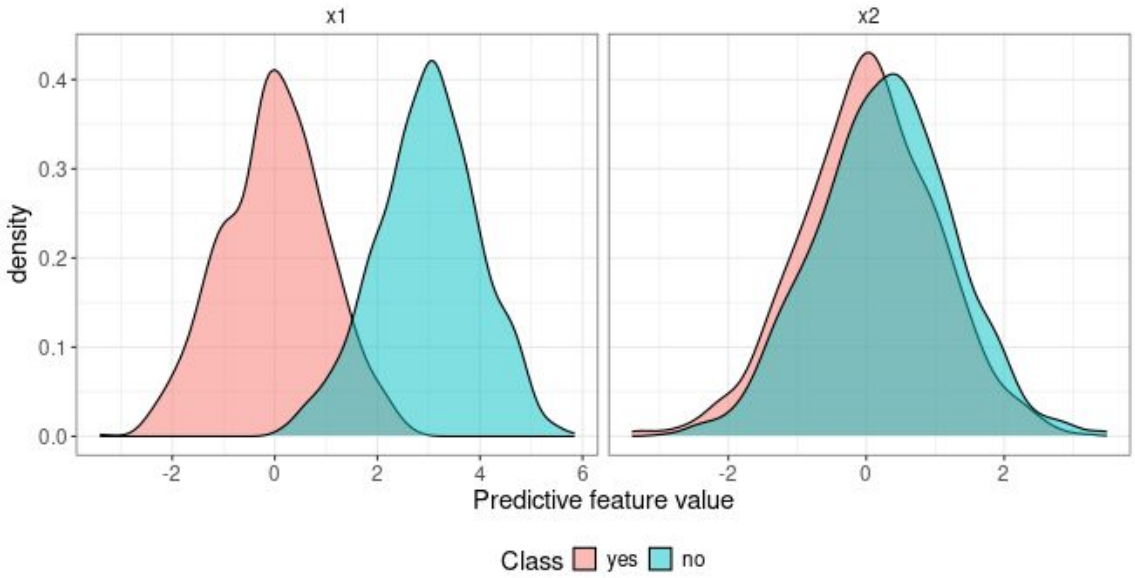
Filter methods

La selección se hace antes de entrenar el algoritmo de aprendizaje

Se utilizan en test estadísticos que de alguna manera miden qué tanto se relaciona un predictor con la variable a predecir (e.g., correlaciones, test chi^2, test anova, test de ganancia de información, etc.)



a) Filter method



¿Qué variable tiene más chances de ser un buen predictor, X_1 o X_2?

Wrapper methods

Probemos el tercer bloque de código

Estructura de la clase

- Motivación
- Embedded methods
- Wrapper methods
- Filter methods
- Data leakage

Data leakage

Veamos el cuarto bloque de código

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

¿En qué paso nos equivocamos?

Data leakage

Ocurre cuando el conjunto de entrenamiento contiene, de alguna manera, información del conjunto de validación (y/o testeo) que no debiera estar disponible al momento de entrenar el modelo

La filtración de información entre los conjuntos puede ser sutil

Cuando 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/testear 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/test set (ejemplo visto en código)
- Escalar/Discretizar variables utilizando todo el dataset (no es tan grave si uno asume que los distribuciones se mantendrán estables)
- Hacer *bin-counting* utilizando los datos de validación/testeo
- Hacer *oversampling* antes de separar el conjunto 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%

$$y(day_t) = x_1(day_t+5) / x_1(day_t) - 1$$

Data leakage

Algunos ejemplos de data leakage:

- Seleccionar atributos utilizando información de la variable a predecir proveniente del validation/test set (ejemplo visto en código)
- Escalar/Discretizar variables utilizando todo el dataset (no es tan grave si uno asume que los distribuciones se mantendrán estables)
- Hacer *bin-counting* utilizando los datos de validación/testeo
- Hacer *oversampling* antes de separar el conjunto 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%

$$y(day_t) = x_1(day_t+5) / x_1(day_t) - 1$$

Data leakage

Aclaraciones:

- Transformaciones para las cuales no se debe ver otra instancia salvo la que se transforma no generan data leakage (por ejemplo, tomar logaritmos, interacciones entre variables, etc.)

- Data leakage NO es overfitting

  - Overfitting es entrenar un modelo permitiendo que sea demasiado flexible. Esto genera buenos resultados en el training set, pero malos en validation, testing y producción (es facilmente detectable)

  - Data leakage es “hacer trampa”. La performance estimada en validación y en testeo (si hubiera conjunto de test) dejá de ser fidedigna

Bibliografía

- ISLP. Secciones 6.1 y 6.2

- Hastie, Tibshirani & Friedman, "The Elements of Statistical Learning", Sección 7.10.2