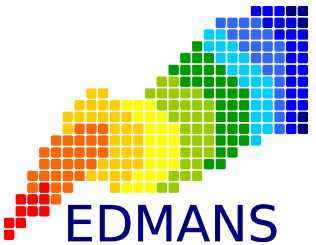
# SISAL: Algoritmo de selección secuencial de variables implementado en R

## III Jornadas de Usuarios de R (17-18 Noviembre, 2011)

Sanz, A., González, A., Fernández, J. & Antoñanzas, F.

Grupo EDMANS (http://www.mineriadatos.com)

# Índice

# Introducción

- La predicción de series temporales (*short/long-term prediction*) sigue siendo un importante campo de investigación

Pred. lineales: ARX, ARMA, ARIMA ; pred. no-lineales: MLP, SVM

- Uno de los problemas comunes en los modelos es la **selección de las variables correctas**
- **Estrategia 'meto todas'** $\rightarrow$ No es correcta
- Conclusión: selección de variables (*feature selection*)**(SV)**, ¿cuáles?

**Modelado con criterio de parsimonia (parsimonious modeling)**

Aquellas que generan un **conjunto parsimonio de variables de entrada**

- Lograr modelos más compactos con un proceso de análisis de datos.
- Mejorar el rendimiento y la precisión del modelo de predicción

# Introducción

- **SV** ayuda a lograr modelos parsimonios reduciendo el num. de variables de entrada

## Estrategias

**SV por Fuerza Bruta;** Num. var. muy elevado $\rightarrow$ No es factible

- 3 alternativas o enfoques:
  1. Filtros (*filter*)
  2. Envolvente (*wrapper*)
  3. Incrustado (*embedded*)

- Diferencia $\rightarrow$ forma en la que el **SV** se integra en el modelo de predicción

En este trabajo se presenta la implementación en **R** del algoritmo **SISAL** (***S**equential **I**nput **S**election **AL**gorithm*), un *filtro* para elegir un conjunto parsimonio de variables de entrada.

# Índice

# Objetivos

- Implementación del algoritmo **SISAL** bajo un programa como **R-project** con licencia ***GNU General Public License***
- Implementación de los métodos clásicos (comparativas)
- Re-implementación de las funciones gráficas creadas en MATLAB©
- Expansión del algoritmo a un enfoque tipo *wrapper* y/o *embedded*
- Mejorar la representación gráfica del proceso de **SV**
- Ampliar el num. de experimentos y *dataset*

# Índice

# SISAL - Sequential Input Selection ALgorithm

- El algoritmo SISAL está pensado para problemas *long-term*. $\Uparrow$ horizonte
- El enfoque es similar a un algoritmo *backward selection* donde las variables de entrada son eliminadas progresivamente de un modelo de auto-regresión (AR).
- La eliminación está basada en la mediana y un ancho empíricamente estimado mediante un procedimiento de remuestreo de validación cruzada.
- Los estadisticos reflejan la importancia de cada entrada en la tarea de predicción:
  1. Función (fsal) Forward selection **with** background knowledge
  2. Función (fsal) Forward selection **without** background knowledge
  3. Función (sisal) Backward selection **with** background knowledge
  4. Función (sisal) Backward input selection **without** background knowledge

# Índice

# Listado principal de funciones

1. `read.fs.results()`
2. `read.bs.results()`
3. `experiments()`
4. `crossvalidation()`
5. `beta.estimation()`
6. `sisal()`
7. `lm.bs.fs()`
8. `plot.coef()`
9. `plot.residuals()`
10. `plot.errors()`
11. `plot.predict()`
12. `plot.sis.results()`
13. `plot.models()`

# Función: `read.fs.results()`

**function**: `read.fs.results (folder.name,name.base,num.experiments)`

- **Resultado**: Lee los datos de un archivo de experimentos de un modelo de FS (R.Data)

- **Parámetros de Entrada**:
  - **folder.name** = Path del archivo de experimentos de modelo FS
  - **name.base** = Nombre del archivo
  - **num.experiments** = Numero de experimentos en la lectura new.results
- **Parámetros de Salida**:
  - **new.results** = Lista con los resultados de los experimentos del archivo

# Función: `read.bs.results()`

**function**: `read.bs.results (folder.name,name.base,num.experiments)`

- **Resultado**: Lee los datos de un archivo de experimentos de un modelo BS (R.Data)

- **Parámetros de Entrada**:
  - **folder.name** = Path del archivo de experimentos de modelo BS
  - **name.base** = Nombre del archivo
  - **num.experiments** = Numero de experimentos en la lectura new.results
- **Parámetros de Salida**:
  - **new.results** = Lista con los resultados de los experimentos del archivo

# Función: `experiments()`

**function**: `experiments f(x, M=10, k=5)`

- **Resultado**: Función para crear una lista con los indices para hacer los diversos experimentos de entrenamiento o para fase de testeo de los modelos

- **Parámetros de Entrada**:
  - **x** = *Dataframe* que contiene los datos de los experimentos
  - **M** = (10) Argumento que especifica el número de grupos formados
  - **k** = (5) Variable que indica el número de divisiones para cada experimento
- **Parámetros de Salida**:
  - **num.it** = Lista con $M * k$ matrices de indices para experimentos.

# Función: `crossvalidation()`

**function**: `crossvalidation (elements, x, var.in, var.out, theta.fit, k=10, y.mean=NULL, y.sd=NULL)`

- **Resultado**: Estimacion de los coeficientes de regresión utilizando $k$-fold cross-validation

- **Parámetros de Entrada**:
  - **elements** = Indices a utilizar en la validación cruzada (de la función de `experiments()`)
  - **x** = *Dataframe* que contiene el vector de datos a procesar
  - **var.in** = Indices de las entradas seleccionadas
  - **var.out** = Indices de las salidas
  - **theta.fit** = (lm) Función seleccionada `lm(y ~ . -1,data=data.frame(x))`
  - **k** = (10) Argumento que especifica el número de grupos formados
  - **y.mean** = (NULL) Media de la variable de salida
  - **y.sd** = (NULL) Desviación estandar de la variable de salida

- **Parámetros de Salida**:
  - **t(coef.matrix),t(MSE.matrix)** = Una matriz con los coeficientes estimados (primeras $k$ filas) y ambos los MSEs de entrenamiento ($k+1$ a $2 \cdot k$ filas) y validación (últimas $k$ filas)

# Función: `beta.estimation()`

**function**: `beta.estimation (x, var.in, var.out, M=100, k=10, theta.fit = NULL, y.mean=NULL, y.sd=NULL)`

- **Resultado**: Estimacion de los ceficientes de regresión utilizando k-fold cross-validation

- **Parámetros de Entrada**:
  - **x** = *Dataframe* con los datos de entrada
  - **var.in** = Indice de las variables de salida de entrada
  - **var.out** = Indice de la variable de salida de salida
  - **M** = Iteraciones. Bootstrap
  - **k** = (10) Argumento que especifica el número de grupos formados
  - **theta.fit** = (lm) Función seleccionada `lm(y ~ . -1,data=data.frame(x))`
  - **y.mean** = (NULL) Media de la variable de salida
  - **y.sd** = (NULL) Desviación estandar de la variable de salida
- **Parámetros de Salida**:
  - **list(coefficients, error)** = Lista que contiene el vector conteniendo los coeficientes estimados y una matriz con ambos errores (MSE) de entrenamiento y de validación

# Función: `sisal()`. Parte I. Entradas

**function**: `sisal (x, var.in, var.out, fix.var.in=NULL, M=100, k=10, y.mean=NULL, y.sd=NULL, sisal.plot=TRUE, ...)`

- **Resultado**: Matriz de coeficientes y matriz de errores MSE como resultados de la ejecución del algoritmo SISAL

- **Parámetros de Entrada**:
  - **x** = *Dataframe* con los datos de entrada
  - **var.in** = Indice de las variables de entrada
  - **var.out** = Indice de la variable de salida
  - **fix.var.in** = Indice de var. entrada fijas (*background knowledge*)
  - **M** = Iteraciones. Bootstrap
  - **k** = (10) Argumento del *k*-crossvalidation
  - **sisal.plot** = (TRUE) Salida gráfica al finalizar el proceso
  - **...** = (-) Variables para modificar la función de dibujo.

# Función: `sisal()`. Parte II. Salida

**function**: `sisal (x, var.in, var.out, fix.var.in=NULL, M=100, k=10, y.mean=NULL, y.sd=NULL, sisal.plot=TRUE, ...)`

- **Resultado**: Matriz de coeficientes y matriz de errores MSE como resultados de la ejecución del algoritmo SISAL

- **Parámetros de Salida**: **[sis]**
  - *c1:* **num.in** = num. de entradas usadas para ajustar el modelo
  - *c2:* **mse.tr.mean** = media de los MSEs de entrenamiento
  - *c3:* **mse.tr.sd** = desviación estandar de los MSEs de entrenamiento
  - *c4:* **mse.val.mean** = media de los MSEs de validación
  - *c5:* **mse.val.sd** = desviación estandar de los MSEs de validación
  - *c6:* **var.del.pos** = posición de la proxima var. a ser eliminada
  - *c7:* **var.del.name** = nombre de la proxima var. a ser eliminada
  - *c8:* **min.vmn** = modelo con minimo error de validacion
  - *c9:* **min.vmn.tsd** = modelo con menor complejidad
  - *c10:* **min.vmn.tsd** = matriz de coeficientes estimados en cada iteración

# Función: `lm.bs.fs()`. Parte I. Entradas

**function:** `lm.bs.fs` (x, var.in, var.out, fix.var.in = NULL, y.mean=NULL, y.sd=NULL, pct.test=0.15, method="bs", train=NULL, test=NULL, new.plot=TRUE)

- **Resultado**: Función que calcula el modelo de regresión lineal inicial, con SV por *forward selection* y por *SISAL*

- **Parámetros de Entrada**:
  - **x** = *Dataframe* con los datos de entrada
  - **var.in** = Indice de las variables de entrada
  - **var.out** = Indice de la variable de salida
  - **fix.var.in** = (NULL) Indice de la variable de entrada fijas (*background knowledge*)
  - **y.mean** = (NULL) Media de la variable de salida
  - **y.sd** = (NULL) Desviación estandar de la variable de salida
  - **pct.test** = (0.15) Tanto por cierto reservado para testeo del modelo
  - **method** = (bs) Método de selección de variables implementado
  - **train** = (NULL) Archivo de entrenamiento diferente
  - **test** = (NULL) Archivo de testeo
  - **new.plot** = (TRUE) Se crea la gráfica en una nueva ventana

# Función: `lm.bs.fs()`. Parte II. Salida

**function:** `lm.bs.fs` (x, var.in, var.out, fix.var.in = NULL, y.mean=NULL, y.sd=NULL, pct.test=0.15, method="bs", train=NULL, test=NULL, new.plot=TRUE)

- **Resultado**: Función que calcula el modelo de regresión lineal inicial, con SV por *forward selection* y por *SISAL*

- **Parámetros de Salida**:
  - *c1:* **original.out** = Salida original de los datos de entradas
  - *c2:* **var** = Vector con las var. del modelo
  - *c3:* **method** = Método de SV utilizado
  - *c4:* **train** = Datos de entrenamiento
  - *c5:* **test** = Datos de testeo
  - *c6:* **initial.lm** = modelo lineal inicial con todas las var.
  - *c7:* **sisal.vmn** = modelo SISAL con minimo error de validacion
  - *c8:* **sisal.vmn.tsd** = modelo SISAL con menor complejidad
  - *c9:* **sisal.bk.vmn** = modelo SISAL con minimo error de validacion (*background knowledge*)
  - *c10:* **sisal.bk.vmn.tsd** = modelo SISAL con menor complejidad (*background knowledge*)

# Función: `plot.coef()`

**function**: `plot.coef (x, sis=NULL, method=NULL, main=NULL, ylim=NULL, col=NULL, text.all=FALSE, new.plot=TRUE)`

- **Resultado**: Visualiza la evolución del los coeficientes del modelo de regresión durante el procedimiento de **SV**

- **Parámetros de Entrada**:
  - **x** = *Dataframe* con los resultados de TR y VL de un modelo ($sis)
  - **sis** = (NULL) Nombre de la variable del sistema
  - **method** = (NULL) Tipo de selección de variables elegido
  - **main** = (NULL) El titulo principal para la gráfica
  - **ylim** = (NULL) El límite vertical de la gráfica
  - **col** = (NULL) Controla el color para a gráfica
  - **text.all** = (logical) Si TRUE crea la gráfica en una nueva ventana
  - **new.plot** = (logical) Si TRUE crea la gráfica en una nueva ventana
- **Parámetros de Salida**:
  - Ninguno.

# Función: `plot.residuals()`

**function**: `plot.residuals (out.original, out.predict, new.plot = TRUE, main=NULL)`

- **Resultado**: Visualiza los errores en la predicción, un histograma de residuos y la gráfica Q-Q

- **Parámetros de Entrada**:
  - **out.original** = Nombre del vector con los datos originales
  - **out.predict** = Nombre del vector con los datos originales
  - **new.plot** = (logical) Si TRUE crea la gráfica en una nueva ventana
  - **main** = (NULL) El titulo principal de la gráfica
- **Parámetros de Salida**:
  - Ninguno

# Función: `plot.errors()`

**function:** `plot.errors (bs.results,num.experiments,y.sd=NULL,new.plot=FALSE)`

- **Resultado**: Visualiza todos los errores calculados. Entrenamiento y validación

- **Parámetros de Entrada**:
  - **bs.results** = Data.frame conteniendo la salida de la selección de variables.
  - **num.experiments** = Numero de experimentos
  - **y.sd** = (NULL) Limite vertical de el gráfico
  - **new.plot** = (logical) Si TRUE crea la gráfica en una nueva ventana
- **Parámetros de Salida**:
  - Ninguno

# Función: `plot.predict()`

**function**: `plot.predict (out.original, out.predict, new.plot = TRUE, main=NULL)`

- **Resultado**: Visualiza la evolución del error MSE de testeo para una serie de datos ya almacenada previamente y no utilizada en la fase de entrenamiento.

- **Parámetros de Entrada**:
  - **out.original** = Nombre del vector con los datos originales
  - **out.predict** = Nombre del vector con los datos originales
  - **new.plot** = (logical) Si TRUE crea la gráfica en una nueva ventana
  - **main** = (NULL) El titulo principal de la gráfica
- **Parámetros de Salida**:
  - Ninguno

# Función: `plot.sis.results()`

**function:** `plot.sis.results (lm.sis,plot.coef=TRUE,plot.residuals=FALSE)`

- **Resultado**: Visualiza como se modifica el modelo, los coeficientes y residuos. Se considera la función de salida global.

- **Parámetros de Entrada**:
  - **lm.sis** = *Dataframe* que contiene todos los datos del sistema
  - **plot.residuals** = (logical) Si TRUE presenta la gráfica de residuos
  - **plot.coef** = (logical) Si TRUE presenta la gráfica de coeficientes
- **Parámetros de Salida**:
  - Ninguno

# Función: `plot.models()`

**function**: `plot.models`
`(x,var.out,test=NULL,sis.bs,sis.fs,nlm.data,models=c("SVM","NN LMLS","Linear")`

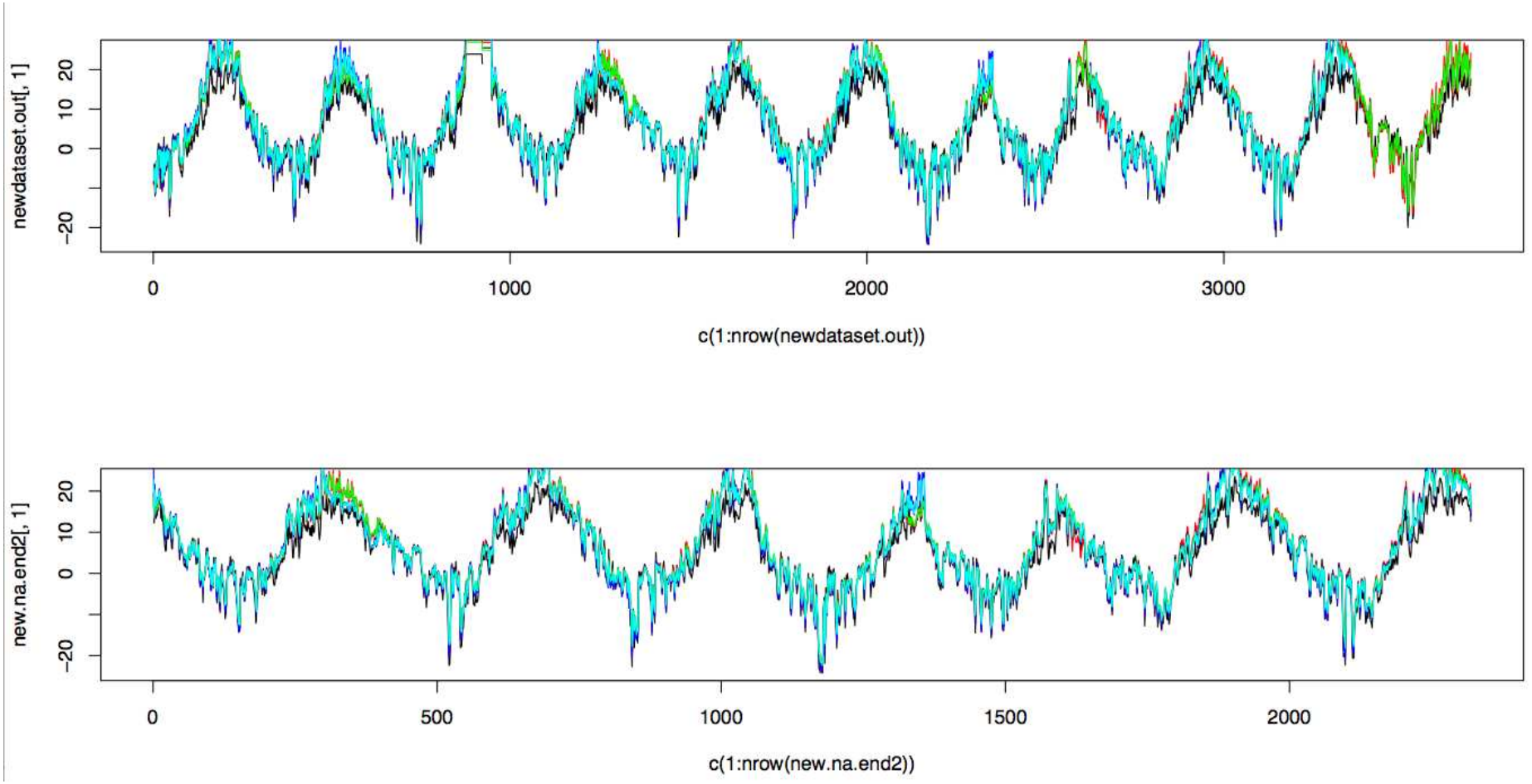
- **Resultado**: Visualiza los resultados del proceso de SV y la salida predecida y MSE para el mejor de los modelo seleccionados

- **Parámetros de Entrada**:
  - **x** = *Dataframe* con los datos de entrada
  - **var.out** = Indice de la variable de salida
  - **sis.bs** = Sistema de seleccion SISAL Forward Sel.
  - **sis.fs** = Sistema de seleccion SISAL Backward Sel.
  - **nlm.data** = Sistema de seleccion SISAL non-linear
  - **models** = Lista de modelos elegidos como regresores
- **Parámetros de Salida**:
  - Ninguno

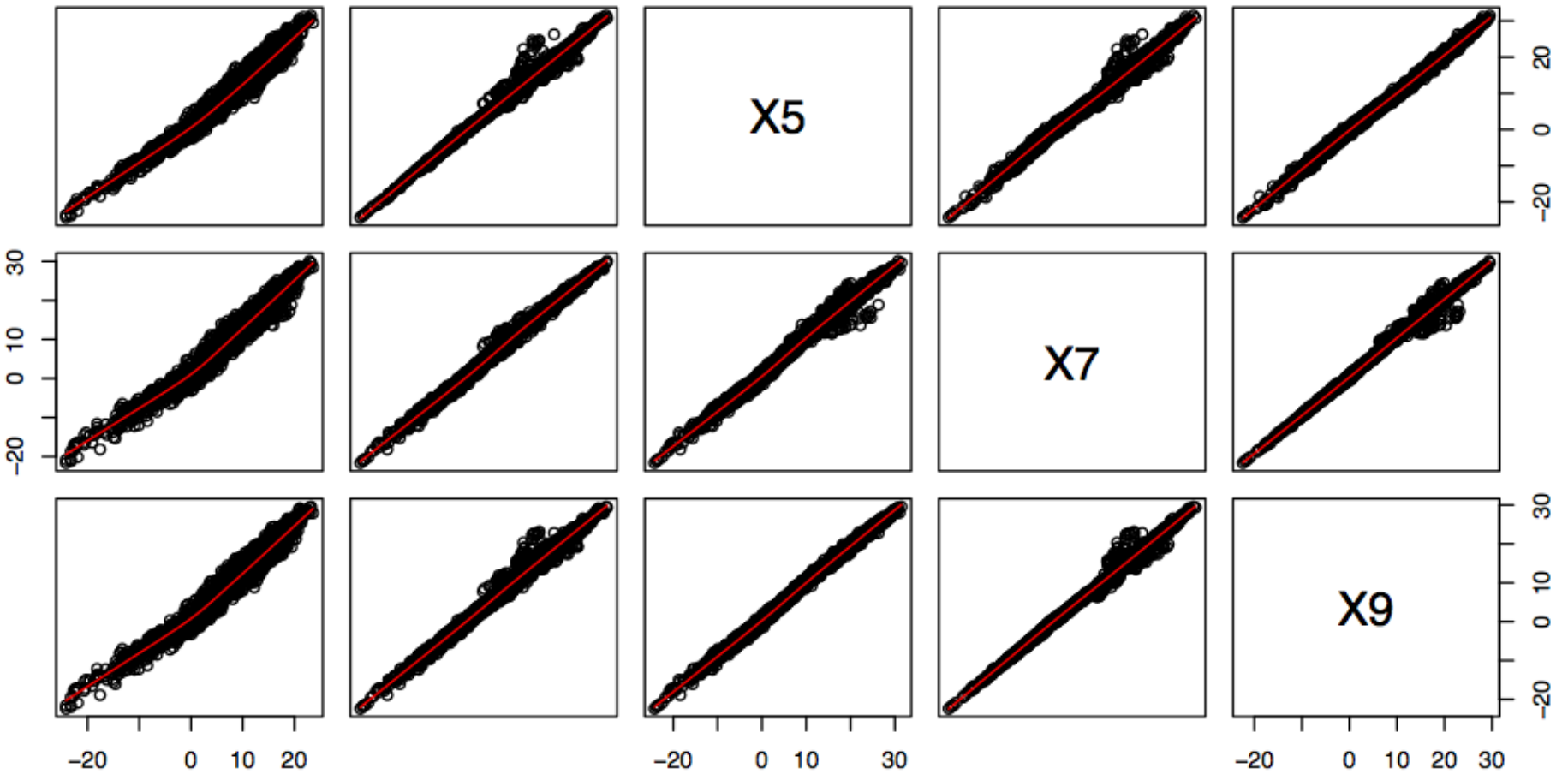
# Índice

# Descripción de la base de datos

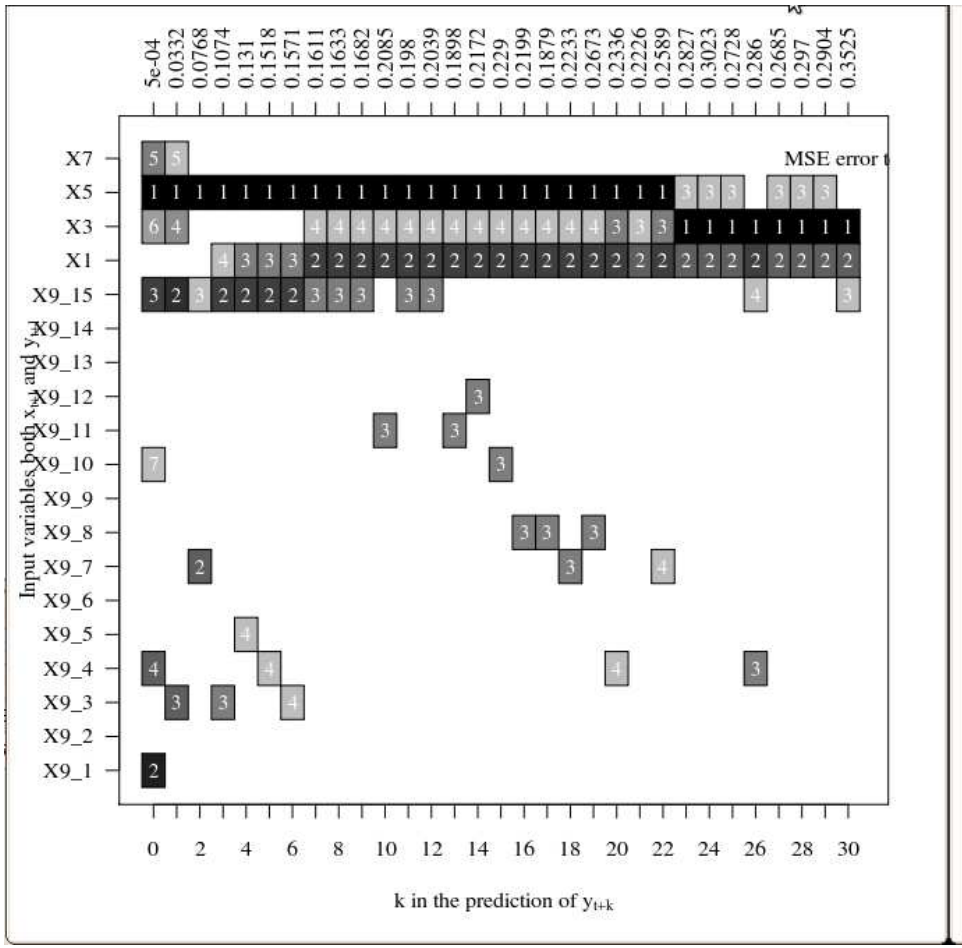
La BD son temperaturas anuales durante 13 sucesivos años, con largos periodos por debajo de cero en invierno. Las máximas temperaturas rondan los 25-30ºC. La serie tiene 5 variables altamente correlacionadas.

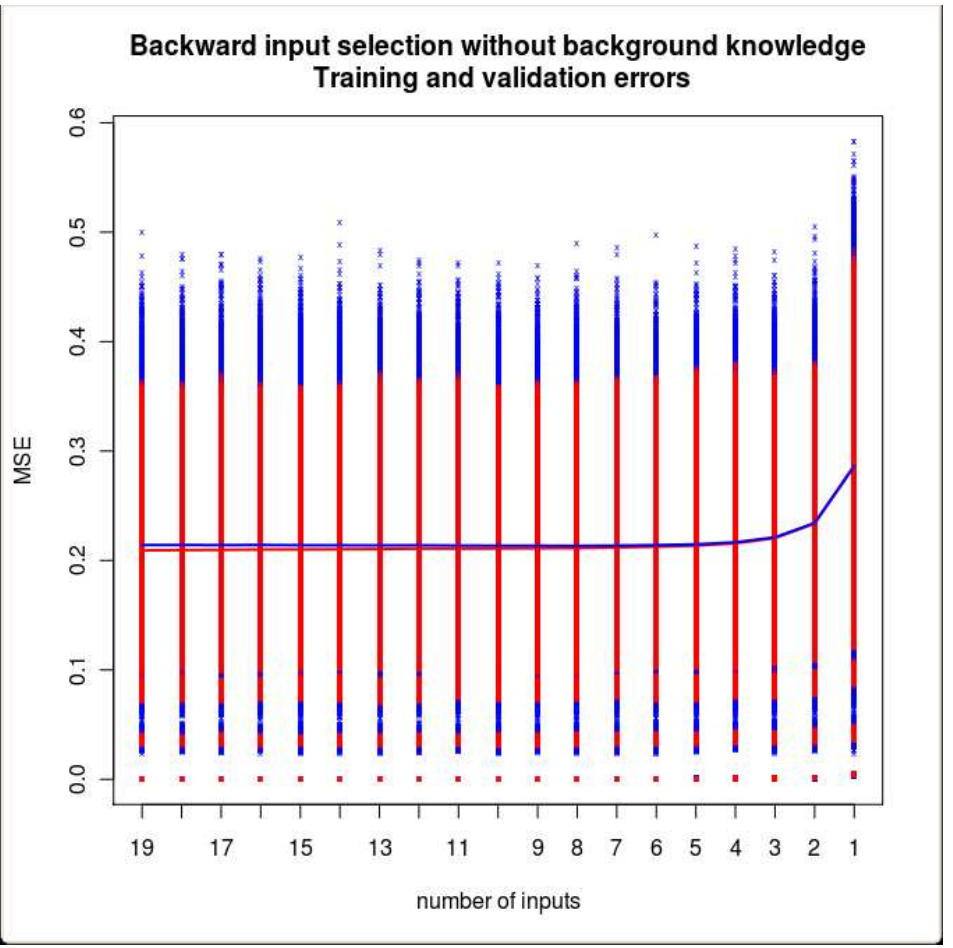


# Resultados obtenidos I

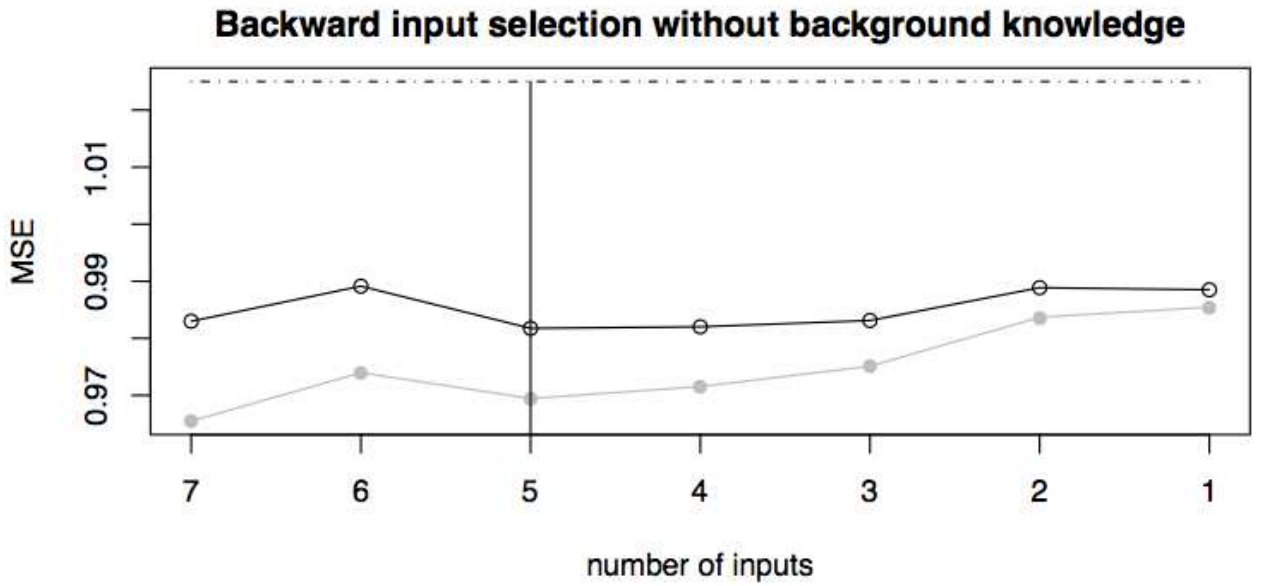


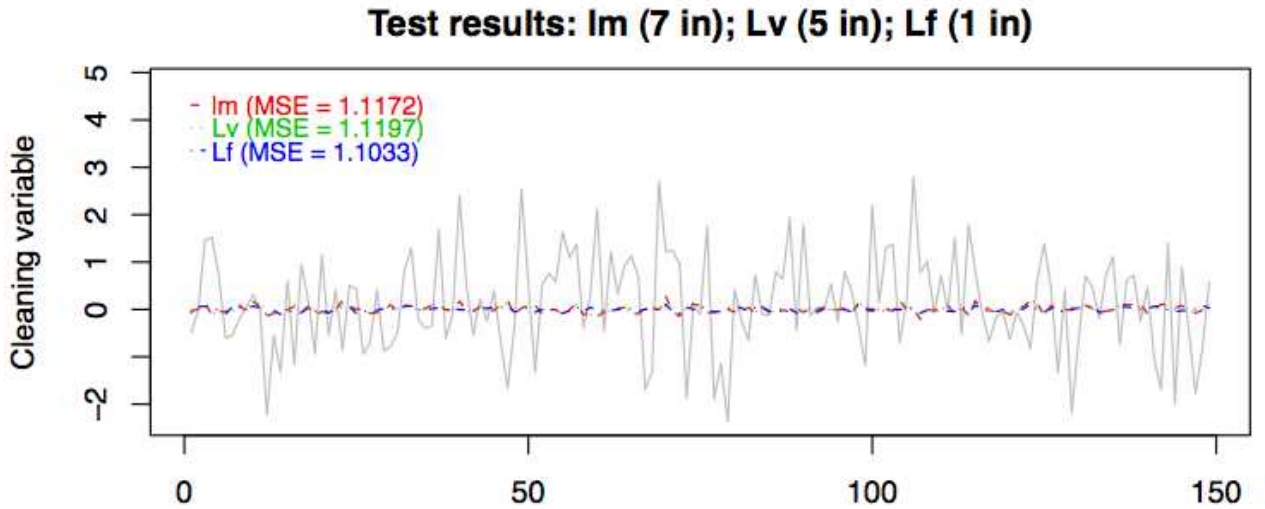
# Resultados obtenidos II



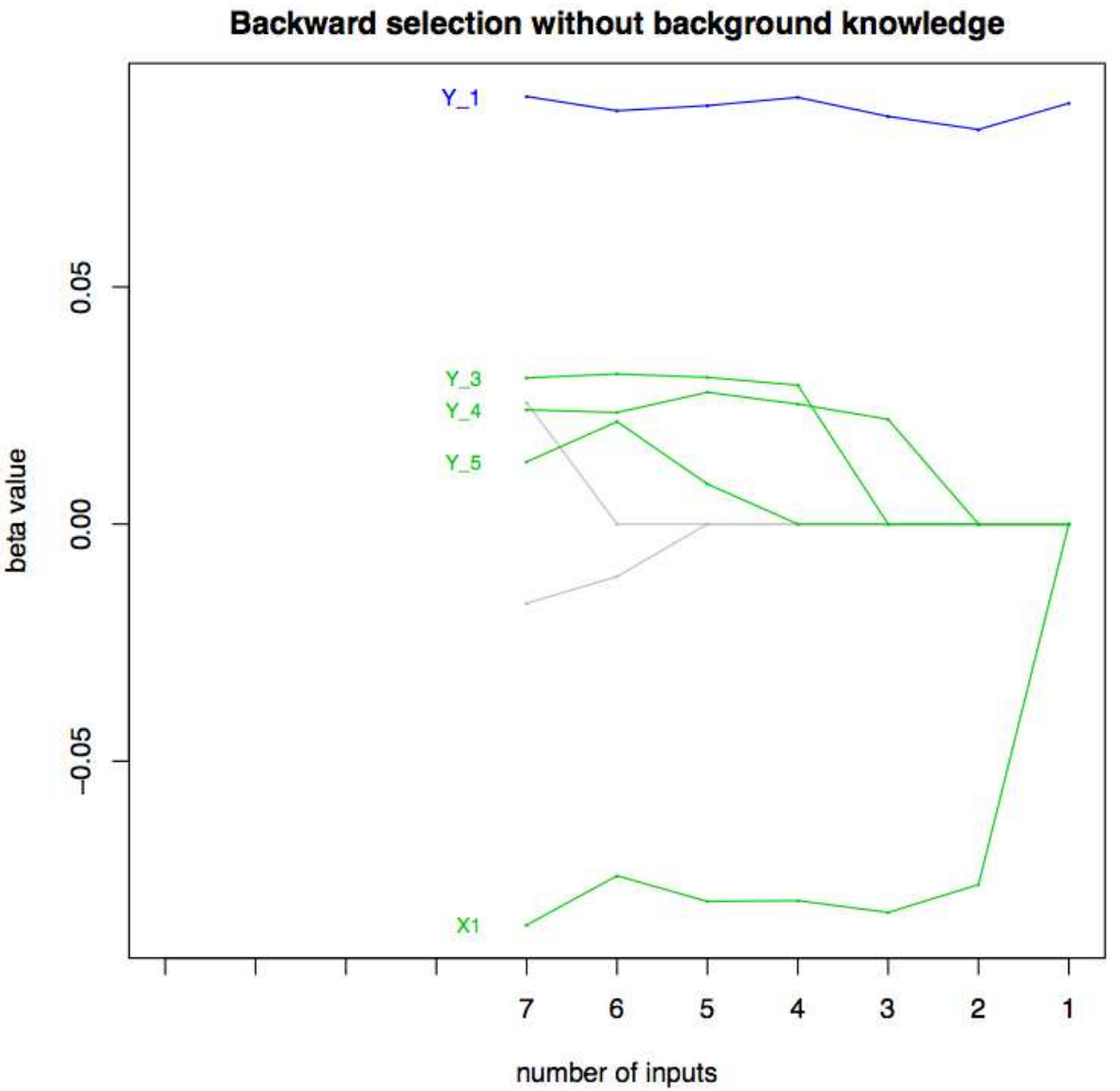



# Resultados obtenidos III

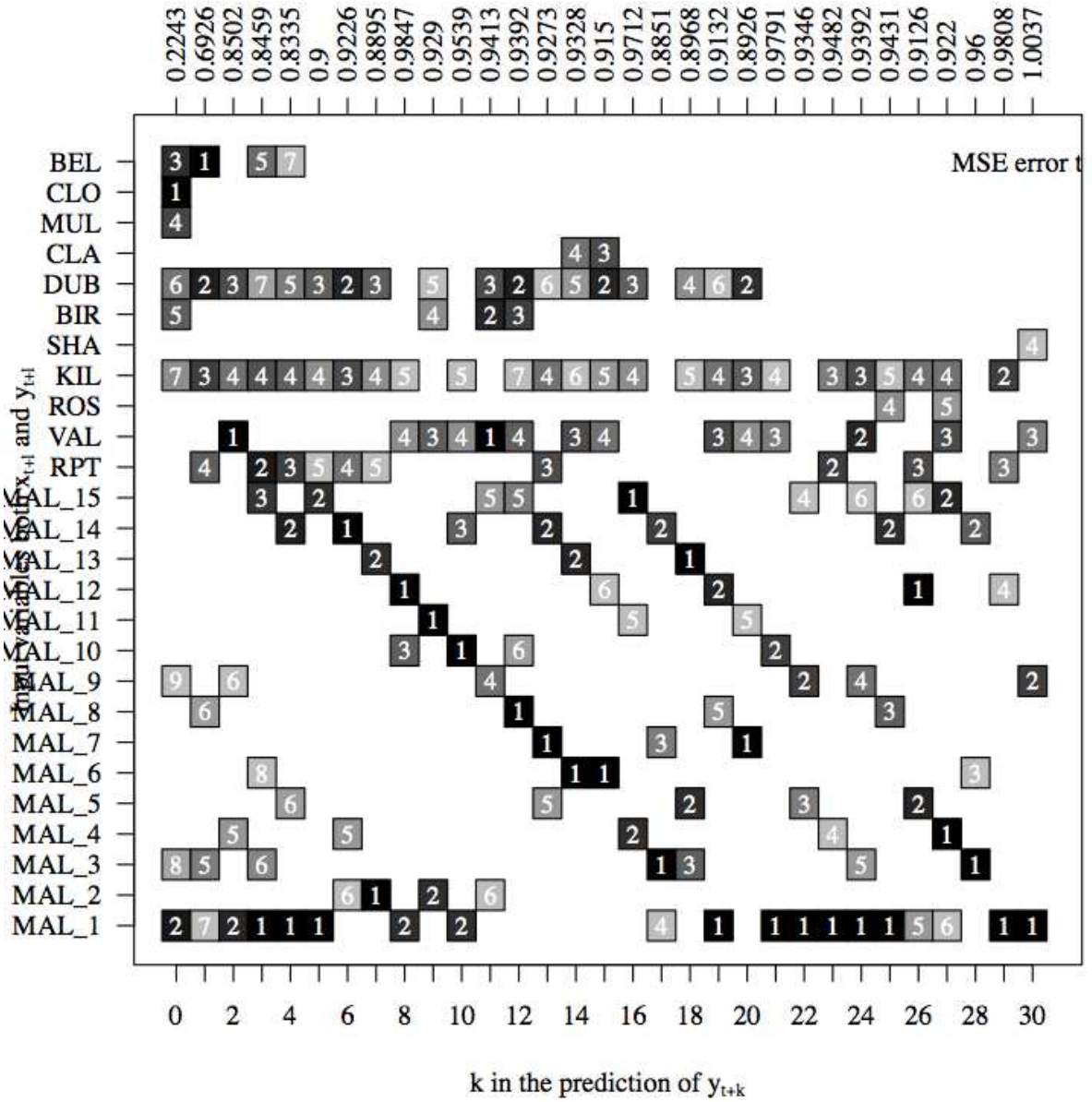


# Resultados obtenidos IV



# Resultados obtenidos. Otros casos



# Índice

# Bibliografía


Guyon, I. and Elisseeff, A.
*An introduction to variable and feature selection*
J. Mach. Learn. Res., 3 : 1157-1182, March 2003

Tikka, J. ; Hollmén J. and Lendasse, A.
*Input Selection for Long Term Prediction of Time Series. In: Proceedings of the 8th International Work Conference on Artificial Neural Networks (IWANN 2005)*
pp. 1002-1009, 2005

Tikka, J. and Hollmén, J.
*Sequential input selection algorithm for long term prediction of time series.*
Neurocomput., 71: 2604-2615, August 2008

Tikka, J.
*Input variable selection methods for construction of interpretable regression models.*
*Doctoral thesis, Helsinki University of Technology, Dissertations in Information and Computer Science, TKK-ICS-D11, Espoo, Finland.*