A Python package for performing penalized maximum likelihood estimation of conditional logit models using Kernel Logistic Regression

José Ángel Martín-Baos
Ricardo García-Ródenas
Luis Rodriguez-Benitez

1. Presentación
[Vídeo – Primer plano]

2. Motivación
[Vídeo con efectos]

Revisión de la literatura

Paquete	Lenguaje de programación	¿Activo?	¿Es open source?	Preprocesado de datos	Conjuntos de elección individuales	Cálculo de indicadores	Modelos de elección discreta
Biogeme	Interfaz en Python 3 (implementado en C)	Sí	Sí	• Pandas • Funciones internas	Sí	WTP, VOT, cuotas de mercado, elasticidades	9 modelos
PyLogit	Python 2/3	Actualizado en 2017	Sí, BSD-3-Clause	• Pandas • Datos en formato largo	Sí	No	7 modelos
ChoiceModels	Python 2/3	Sí	Sí, BSD-3-Clause	• Pandas	Sí (limitado)	No	Sólo MNL
mlogit	R	Sí	Sí, GPLv2	• Datos en formato largo	No	No	3 modelos

Paquete	Lenguaje de programación	¿Activo?	¿Es open source?	Preprocesado de datos	Conjuntos de elección individuales	Cálculo de indicadores	Modelos de elección discreta
Biogeme	Interfaz en Python 3 (implementado en C)	Sí	Sí	• Pandas • Funciones internas	Sí	WTP, VOT, cuotas de mercado, elasticidades	9 modelos
PyLogit	Python 2/3	Actualizado en 2017	Sí, BSD-3-Clause	• Pandas • Datos en formato largo	Sí	No	7 modelos
ChoiceModels	Python 2/3	Sí	Sí, BSD-3-Clause	• Pandas	Sí (limitado)	No	Sólo MNL
mlogit	R	Sí	Sí, GPLv2	• Datos en formato largo	No	No	3 modelos

Paquete	Lenguaje de programación	¿Activo?	¿Es open source?	Preprocesado de datos	Conjuntos de elección individuales	Cálculo de indicadores	Modelos de elección discreta
Biogeme	Interfaz en Python 3 (implementado en C)	Sí	Sí	• Pandas • Funciones internas	Sí	WTP, VOT, cuotas de mercado, elasticidades	9 modelos
PyLogit	Python 2/3	Sí	Sí, 3-Clause	• Pandas • Datos en formato largo	Sí	No	7 modelos
ChoiceModels	Python 2/3	Sí	Sí, BSD-3-Clause	• Pandas	Sí (limitado)	No	Sólo MNL
mlogit	R	Sí	Sí, GPLv2	• Datos en formato largo	No	No	3 modelos

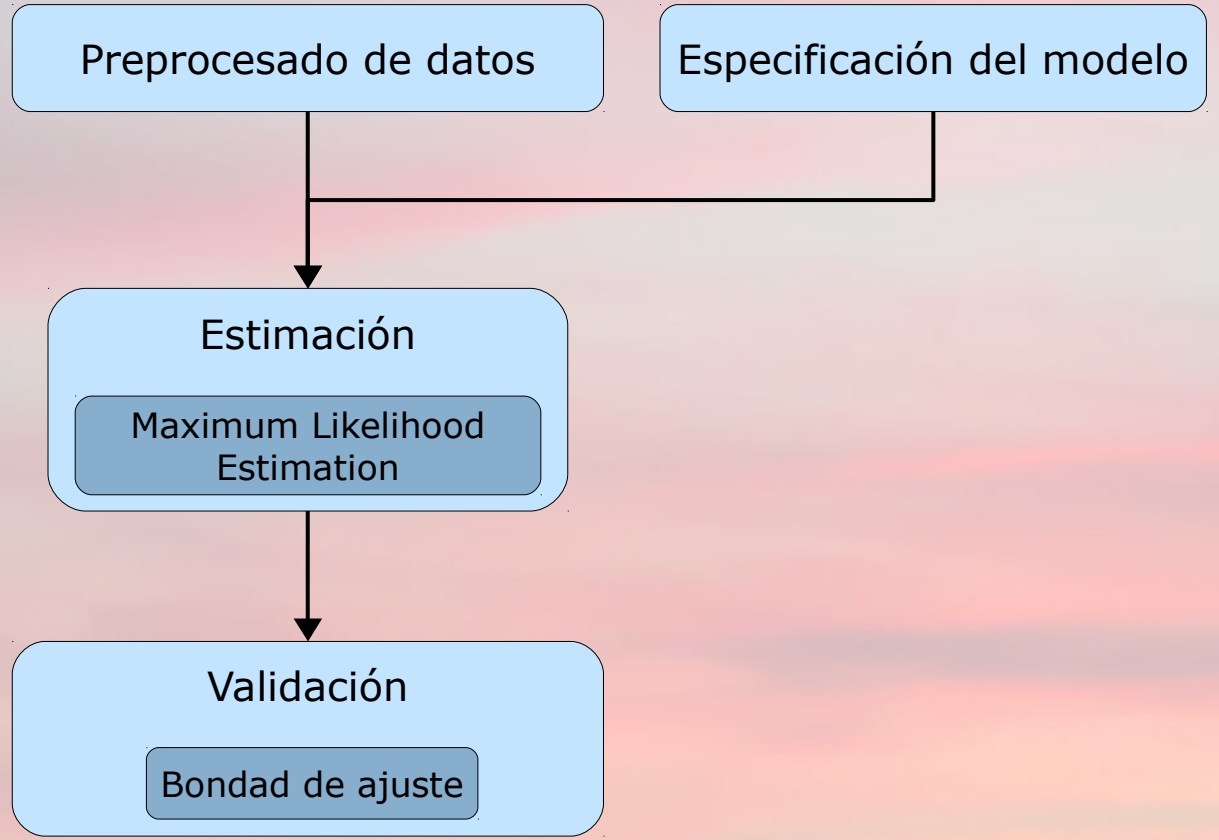
Timothy Brathwaite

University of Berkeley, California

2018


PKL
PyKernelLogit

PyLogit


Preprocesado de datos
Especificación del modelo
Estimación
Maximum Likelihood Estimation
Validación
Bondad de ajuste

Especificación de un modelo Kernel

RUM

$$V_{in} = \boldsymbol{\beta}_i^\top \mathbf{x}_{in} = \sum_{k=1}^{K} \beta_{ik} x_{ink}$$

KLR

$$V_i(\mathbf{x}_{in'}, \boldsymbol{\alpha}_i) =$$ **RKHS**

RUM

$$V_{in} = \boldsymbol{\beta}_i^\top \mathbf{x}_{in} = \sum_{k=1}^{K} \beta_{ik} x_{ink}$$

KLR

$$V_i(\mathbf{x}_{in'}, \boldsymbol{\alpha}_i) = \sum_{n=1}^{N} \alpha_{in} k_i(\mathbf{x}_{in}, \mathbf{x}_{in'})$$

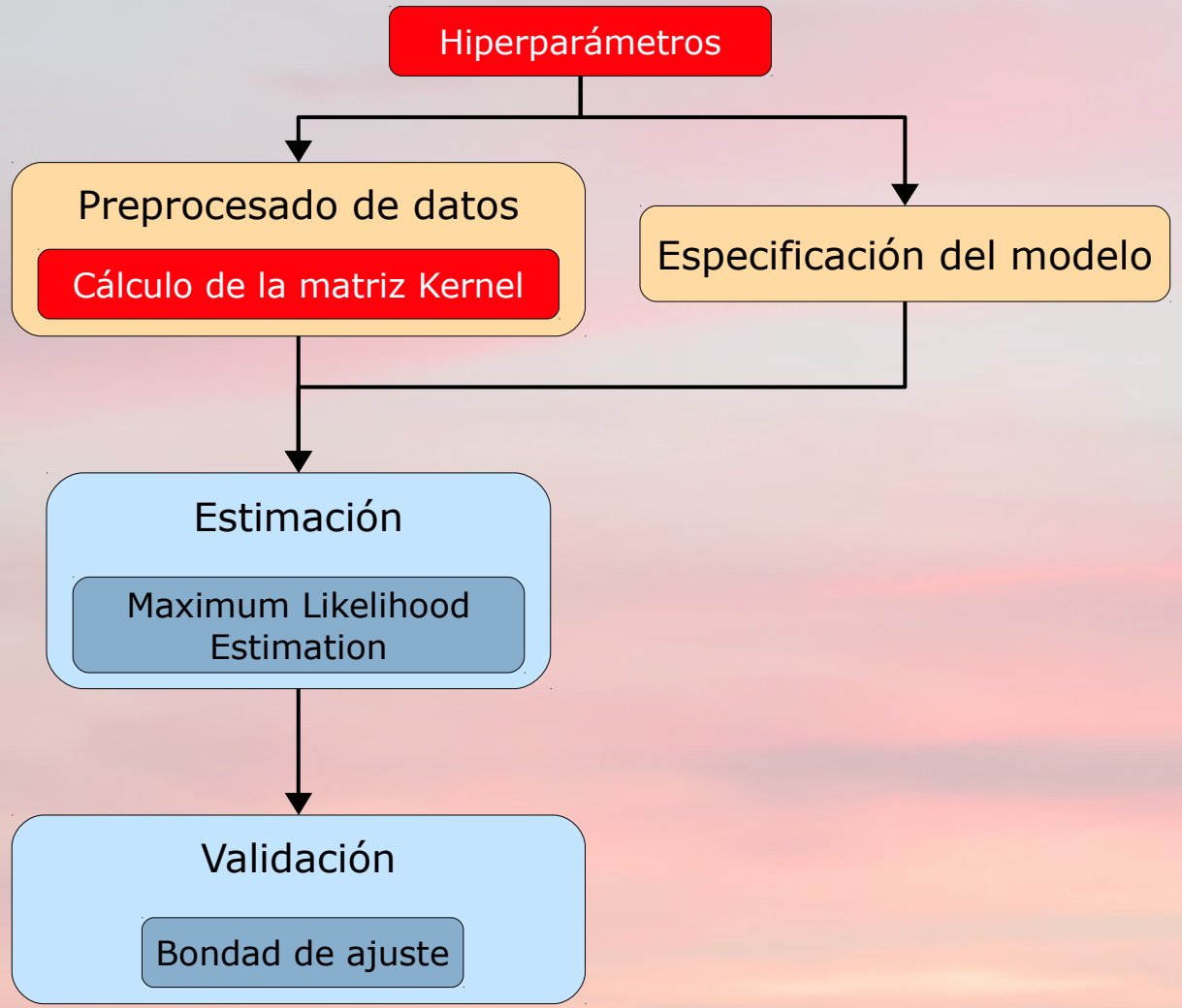
$$\underbrace{\mathbf{K}^{(i)}_{n',n}} = \mathrm{k}_i(\mathbf{x}_{in'}, \mathbf{x}_{in})$$

```
K = pkl.long_format_with_kernel_matrix(dataset,
                                       ...,
                                       variables,
                                       kernel_type = "RBF")
```

Función Kernel (**hyperparámetro**)

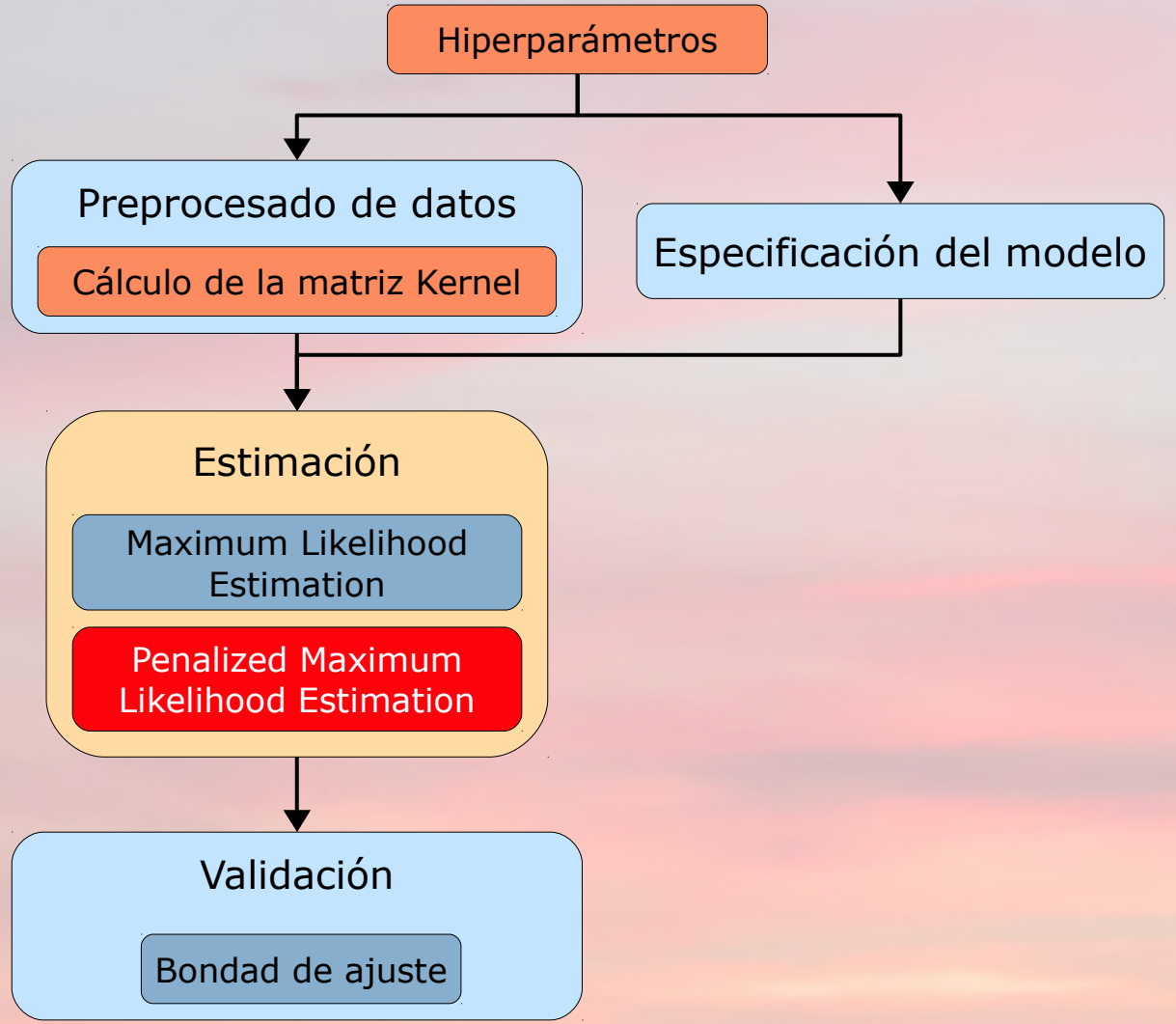
- Radial Basic Function (RBF)
- Matern
- RationalQuadratic
- Etc.

PyKernelLogit



Estimación del modelo

PyKernelLogit



$$\underset{\boldsymbol{\alpha}}{\text{Minimize}} \sum_{n=1}^{N} \sum_{i=1}^{I} L(y_{in}, V_i(\mathbf{x}_{in}, \boldsymbol{\alpha}_i)) + \lambda \sum_{i=1}^{I} R_i(\alpha_i)$$

$$\underset{\boldsymbol{\alpha}}{\text{Minimize}} \sum_{n=1}^{N} \sum_{i=1}^{I} L(y_{in}, V_i(\mathbf{x}_{in}, \boldsymbol{\alpha}_i)) + \lambda \sum_{i=1}^{I} R_i(\alpha_i)$$

Función de perdida

$$-\sum_{n=1}^{N} \sum_{i=1}^{I} y_{in} \log P(i|\mathbf{x}_{in}, \boldsymbol{\alpha})$$

$$\underset{\boldsymbol{\alpha}}{\text{Minimize}} \sum_{n=1}^{N} \sum_{i=1}^{I} L(y_{in}, V_i(\mathbf{x}_{in}, \boldsymbol{\alpha}_i)) + \lambda \sum_{i=1}^{I} R_i(\alpha_i)$$

Función de perdida

$$-\sum_{n=1}^{N} \sum_{i=1}^{I} y_{in} \log P(i|\mathbf{x}_{in}, \boldsymbol{\alpha})$$

Función de penalización

- LASSO
- Ridge

$$\underset{\boldsymbol{\alpha}}{\text{Minimize}} \sum_{n=1}^{N} \sum_{i=1}^{I} L(y_{in}, V_i(\mathbf{x}_{in}, \boldsymbol{\alpha}_i)) + \lambda \sum_{i=1}^{I} R_i(\alpha_i)$$

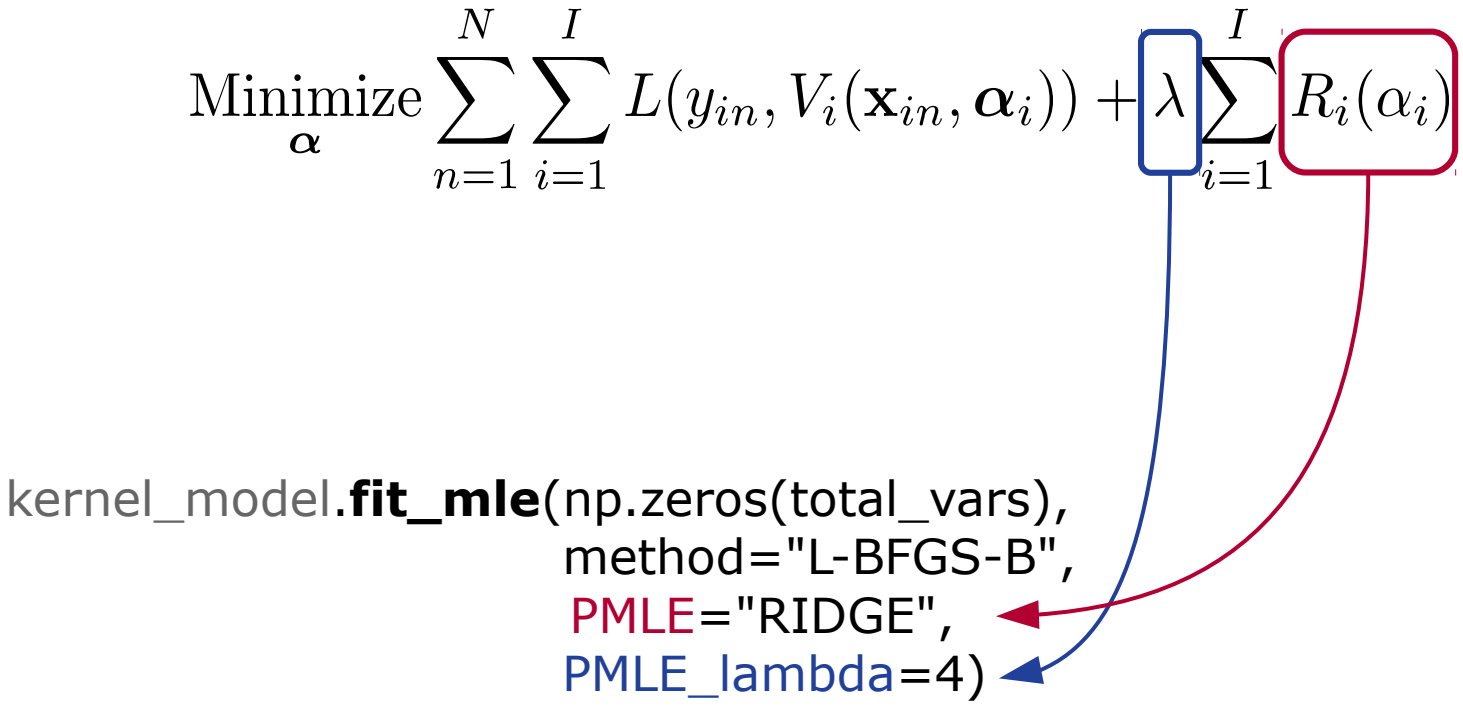
Parámetro de regularización

Función de perdida

Función de penalización

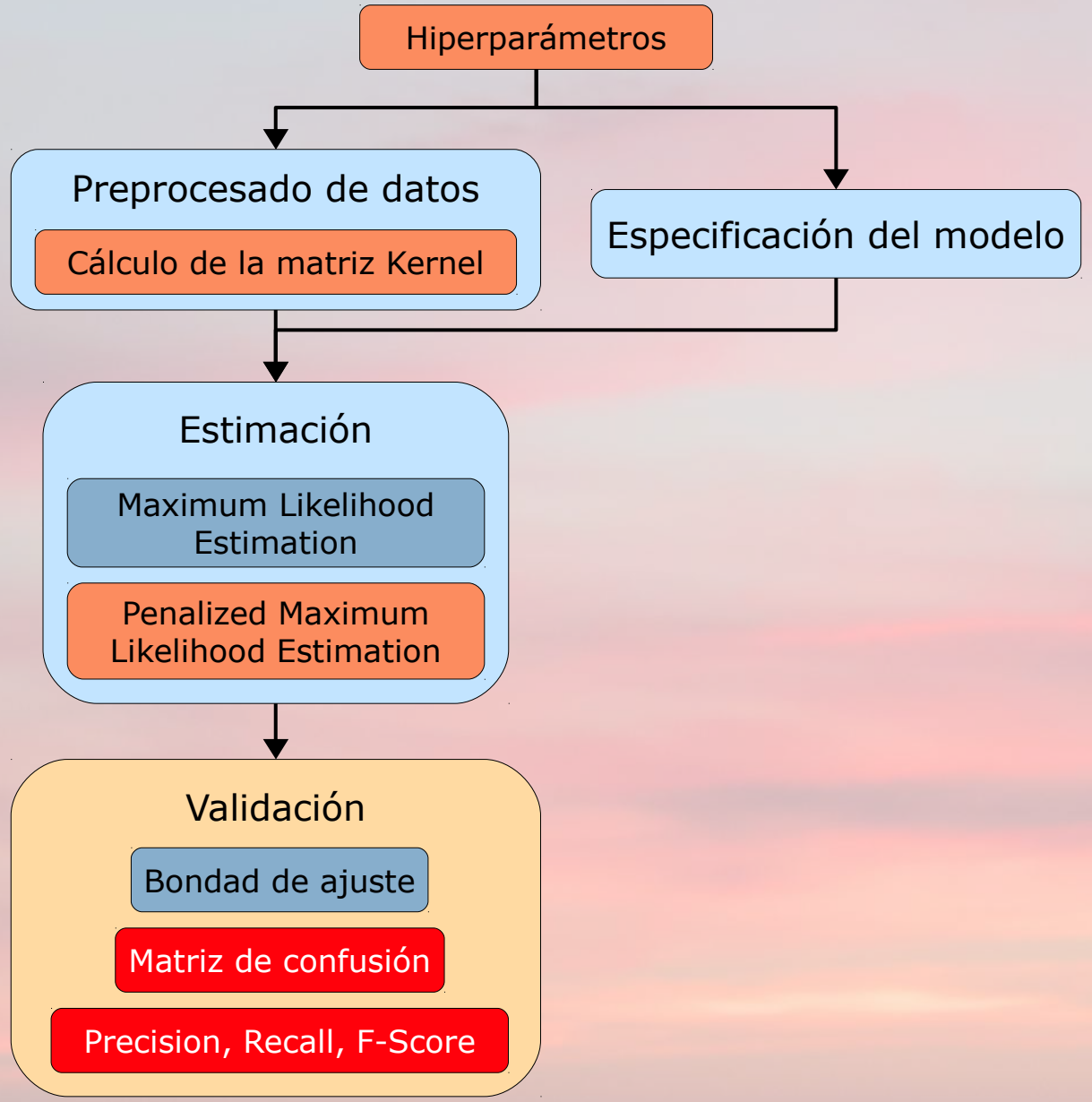
$$-\sum_{n=1}^{N} \sum_{i=1}^{I} y_{in} \log P(i|\mathbf{x}_{in}, \boldsymbol{\alpha})$$

- LASSO
- Ridge


$\underset{\boldsymbol{\alpha}}{\text{Minimize}} \sum_{n=1}^{N} \sum_{i=1}^{I} L(y_{in}, V_i(\mathbf{x}_{in}, \boldsymbol{\alpha}_i)) + \lambda \sum_{i=1}^{I} R_i(\alpha_i)$
kernel_model.fit_mle(np.zeros(total_vars),
method="L-BFGS-B",
PMLE="RIDGE",
PMLE_lambda=4)

Validación del modelo

PyKernelLogit



Bondad de ajuste

McFadden R square

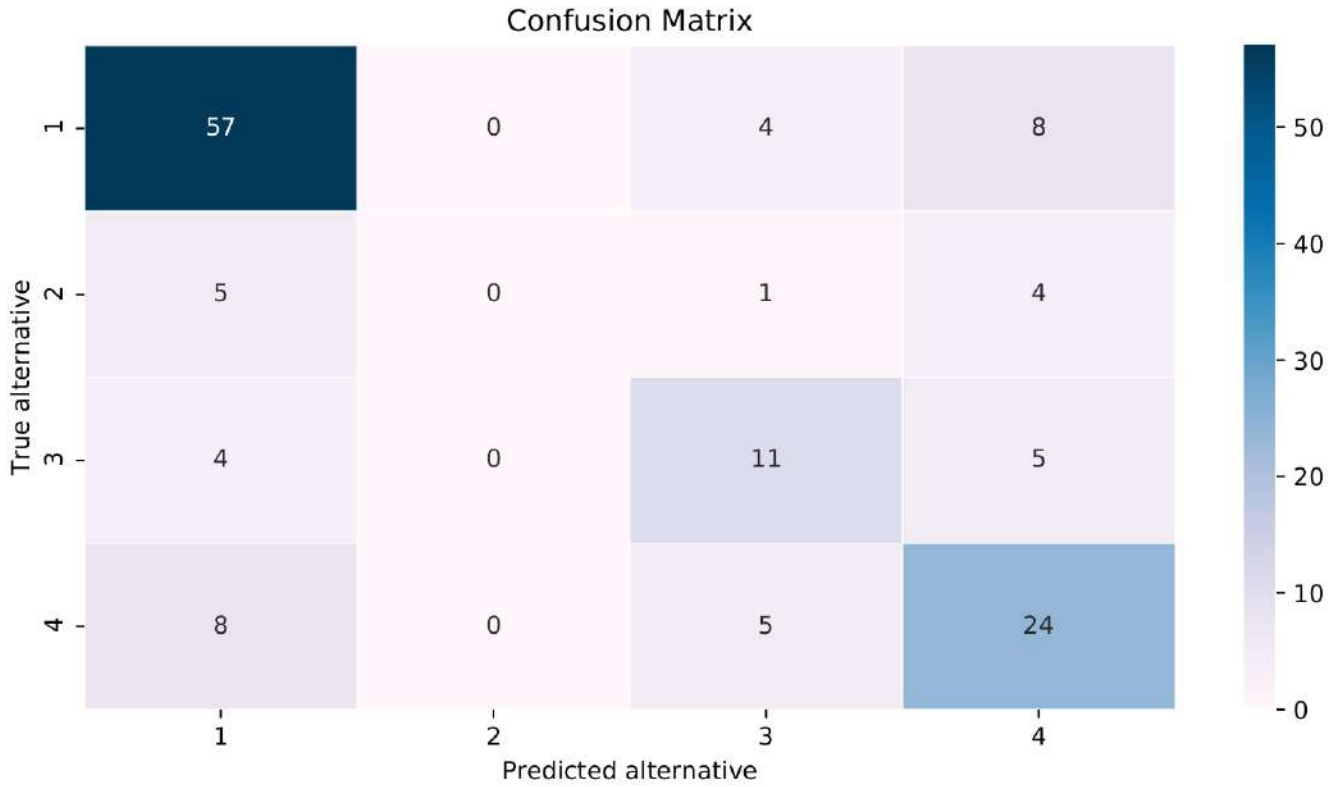
$$\rho^2 = 1 - \frac{LL(\hat{\boldsymbol{\beta}})}{LL(\mathbf{0})}$$

Multinomial Logit Model Regression Results

Dep. Variable:	choice	**No. Observations:**	317
Model:	Multinomial Logit Model	**Df Residuals:**	-3
Method:	MLE	**Df Model:**	320
Date:	Fri, 18 Oct 2019	**Pseudo R-squ.:**	0.355
Time:	10:55:14	**Pseudo R-bar-squ.:**	-0.373
AIC:	1,207.096	**Log-Likelihood:**	-283.548
BIC:	2,409.945	**LL-Null:**	-439.455

Matriz de confusión

kernel_model.**confusion_matrix**(K_test)

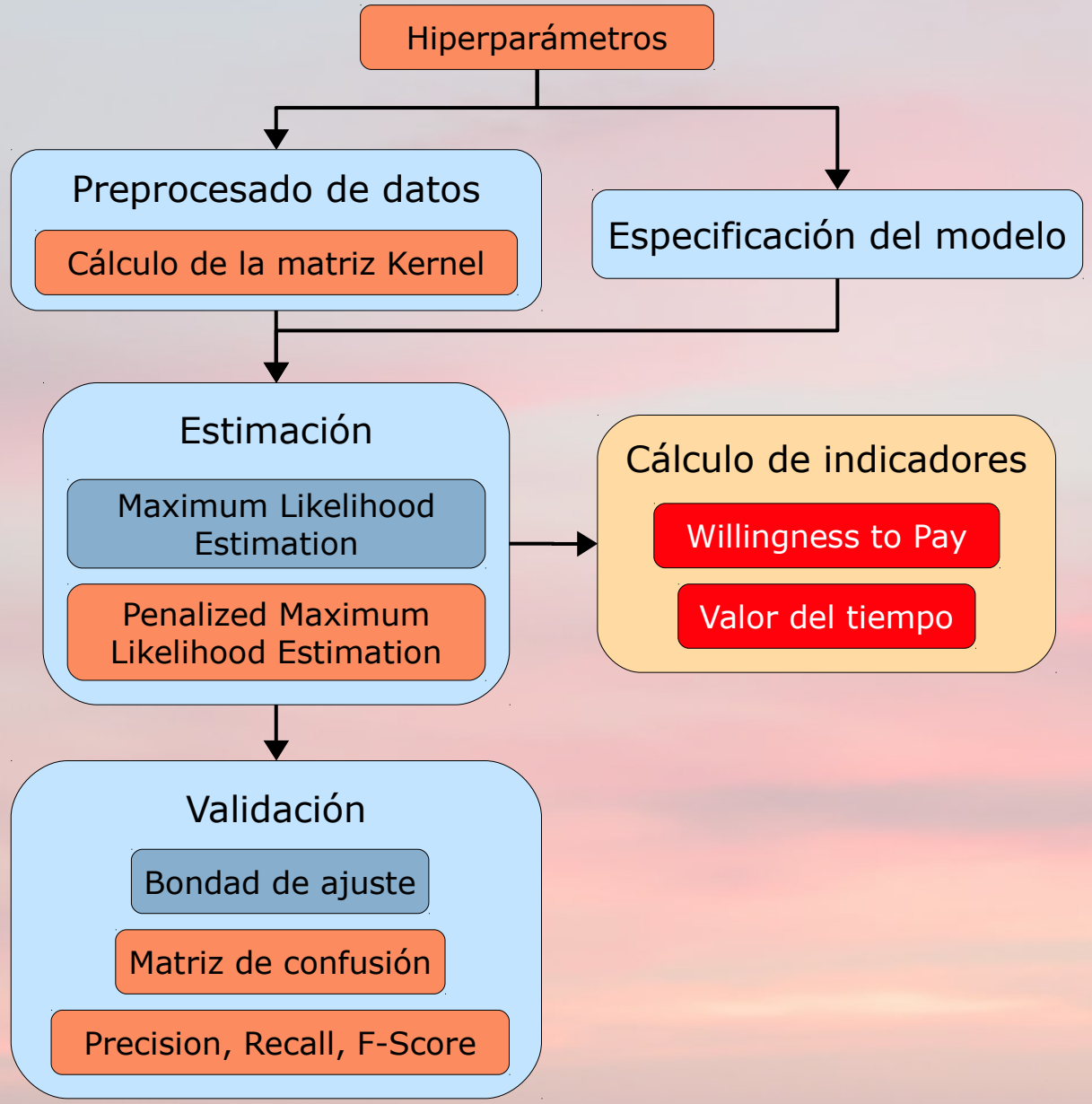


Precision, recall and F-Score

```
kernel_model.precision_recall_fscore(K_test,
                                     "micro",
                                     beta = 1)
```

Cálculo de indicadores

PyKernelLogit



Willingness to Pay

$$\mathrm{WTP}_{in} = \frac{\delta^c_{in}}{\delta^x_{in}} = -\frac{\left(\partial V_{in}/\partial x_{in}\right)\left(c_{in}, x_{in}\right)}{\left(\partial V_{in}/\partial c_{in}\right)\left(c_{in}, x_{in}\right)}$$

Valor del tiempo

$$\mathrm{VOT}_{in} = \frac{\delta^c_{in}}{-\delta^t_{in}} = \frac{\left(\partial V_{in}/\partial t_{in}\right)\left(c_{in}, t_{in}\right)}{\left(\partial V_{in}/\partial c_{in}\right)\left(c_{in}, t_{in}\right)}$$

	custom_id	mode_id	choice	cost	time
0	1.0	1.0	1.0	4.873487	37.044117
1	1.0	2.0	0.0	1.686348	39.771005
2	1.0	3.0	0.0	2.035669	39.923409
3	1.0	4.0	0.0	2.211683	39.504507

 # Ejemplo: VOT de 30 minutos de viaje

```
VOT = -30 * pkl.calculate_WTP_kernel(WTP_cost_column = 'cost',
                                     WTP_x_column = 'time',
                                     ...)
```

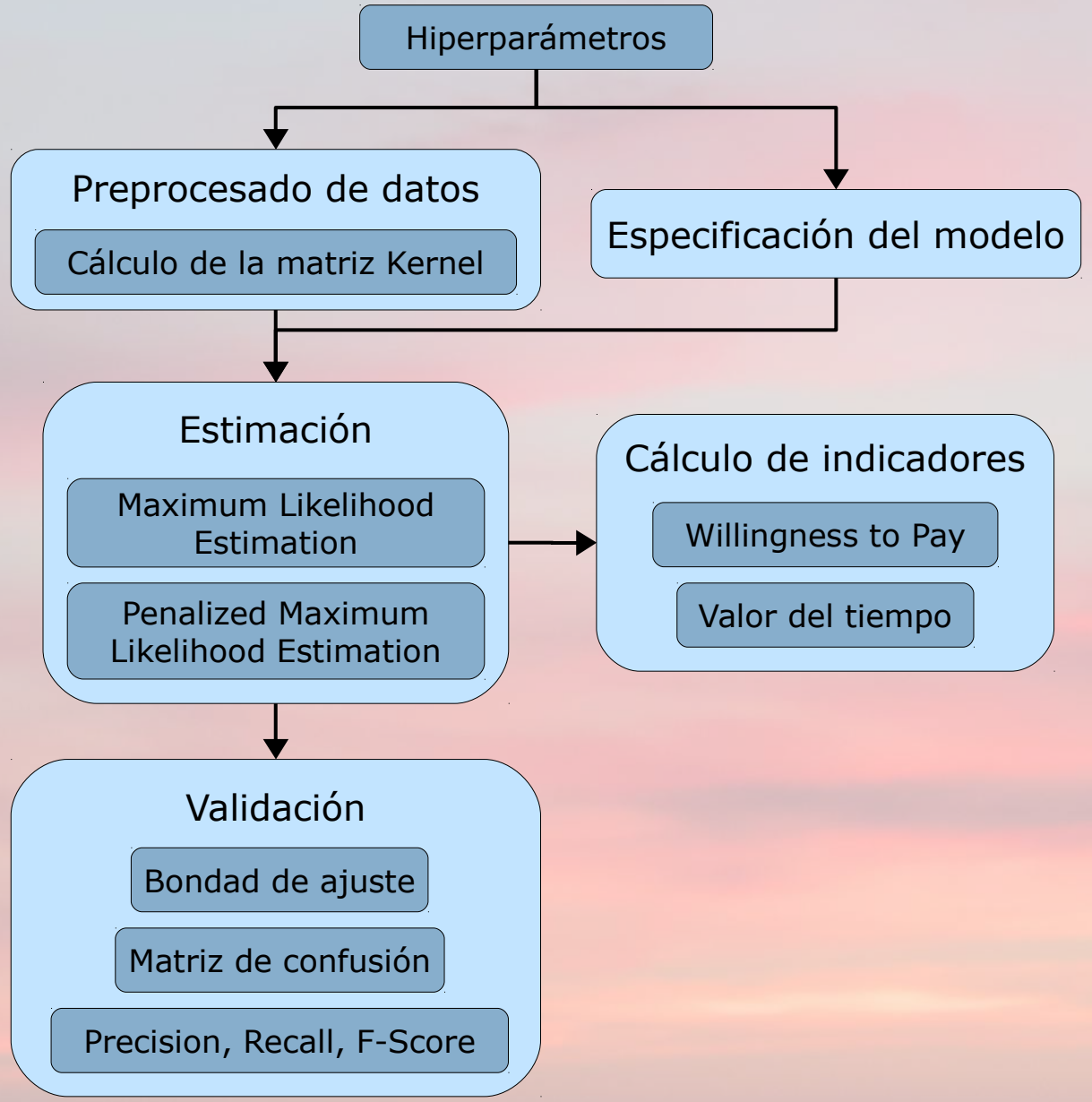
$ 3.8

PyKernelLogit

https://bit.ly/PyKernelLogit

```
$ pip install pykernellogit
```

PyKernelLogit



PyKernelLogit

Random Utility Model (RUM)

Kernel Logistic Regression (KLR)

Muchas gracias

por vuestra atención

José Ángel Martín-Baos
Ricardo García-Ródenas
Luis Rodriguez-Benitez

JoseAngel.Martin@uclm.es

https://joseangelmartinb.github.io

UCLM Universidad de Castilla-La Mancha

esi Escuela Superior de Informática