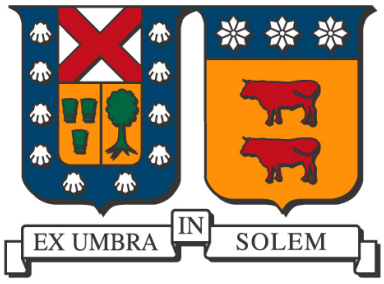


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE MATEMÁTICA

VALPARAÍSO - CHILE

Detección de fraudes en el consumo de agua potable

Memoria de Título presentada por

Francisco Alfaro Medina

como requisito parcial para optar al título de

Ingeniero Civil Matemático

Profesor Guía

Dr. Ronny Vallejos

Martes 22 de Agosto, 2017.

Índice general

Capítulo 1

Introducción

Desde la llegada de la modernidad, la dinámica de acceso a los servicios básicos se ha presentado de distintas formas, por ejemplo la provisión de agua para beber comenzó siendo responsabilidad individual de los usuarios, quienes acudían a fuentes diversas, posteriormente concurrían a pozos centralizados y regularizados, sin embargo, la llegada de la energía eléctrica domiciliaria cambió la forma de acceso, pues no era posible para un usuario individual generar su propia energía, teniendo que acudir a terceros que proveían el servicio, generándose un monopolio natural. El incremento de los requisitos básicos para el consumo de agua, junto con el aumento de la población en los núcleos urbanos, hizo necesario que se normalizara y centralizara el acceso, dando paso a las empresas de servicios sanitarios.

En Chile, la cobertura se ha incrementado de manera sostenida en el tiempo (ver Tabla C.1), llegando a un nivel de acceso del 99,9 % de la población (Superintendencia de Servicios Sanitarios, 2014), colocando a Chile por sobre el promedio latinoamericano, el cual es de un 93 % (Soulier Faure, M., Ducci, J., & Altamira, M., 2013), ubicándose a niveles comparables con países como Noruega o Dinamarca (OECD,2007).

En términos de los consumos y pérdidas, la Asociación Internacional de Agua (IWA) en conjunto con la Asociación Americana de Trabajos en Agua (AWWA), definen los tipos de consumo que puede seguir un cliente y los tipos de pérdidas que existen en los sistema de agua (American Water Works Association, 2012). Para comprender la relación entre estas terminologías, es necesario definir algunos conceptos previos, cuya relación se muestra en la Tabla C.2 (Ver Apéndice C).

- Suministro de Agua Potable: Corresponde al volumen anual de agua potable inyectada al sistema.

- Consumos autorizados: Corresponde al volumen anual de agua medida y/o no medida entregada a clientes autorizados.
- Pérdidas de Agua: Diferencia entre el agua inyectada al sistema y los consumos autorizados.
- Pérdidas aparentes: Consumos no autorizados, todo tipo de imprecisiones de medición, y errores sistemáticos de manejo de datos.
- Pérdidas Físicas: El volumen anual de pérdidas mediante todo tipo de filtraciones, fugas, roturas o rebalse en redes, almacenamiento o puntos de servicio, hasta el punto de medición del cliente.

De acuerdo a los datos contenidos en el Informe de Gestión del Sector Sanitario 2014, presentado por la Superintendencia de Servicios Sanitarios (SISS), en Chile las empresas sanitarias produjeron en su conjunto un total de 1.670 millones de metros cúbicos de agua potable de la cual un 33,65 % no fue facturada, de ese total estimaciones de la SISS establecen que el 74 % se originan por pérdidas físicas, mientras que el 26 % restante corresponde a pérdidas aparentes, esto equivale a cerca de 146 millones de metros cúbicos de agua potable. Si se estima que el metro cúbico de agua cuesta alrededor de $700 pesos chilenos (Superintendencia de Servicios Sanitarios, 2016), en total se estaría perdiendo más 100.000 millones de pesos por año, cifras alarmantes para las empresas sanitarias.

El objetivo principal de este trabajo se enfoca en desarrollar un modelo matemático capaz de identificar y predecir aquellos clientes que tienen un consumo anómalo. El tipo de problema abordado corresponde a un problema clásico de "Modelos de Detección de Fraude". Este tipo de problemas ha sido ampliamente estudiado en las matemáticas financieras, dado que es aquí donde las empresas tienen las mayores pérdidas económicas.

El fenómeno a estudiar tiene una dificultad adicional, la clasificación de los clientes se encuentra sesgada, es decir, existen clientes que presentan consumos anómalos que de momento no han sido detectados como tal, encontrándose registrados como clientes de consumo regular. Por tanto, se debe proponer una nueva metodología de investigación para dar solución a la problemática, tomando un enfoque diferente al usual abordado en este tipo de problemas.

La estructura de trabajo es la siguiente: En el Capítulo 2 se presenta una serie de definiciones y resultados que permiten definir de manera formal los modelos en este trabajo (Murphy, 2012). En el mismo capítulo se habla de enfoques alternativos encontrados en la literatura, desde cómo ha sido abordado el mismo problema en otras partes del mundo: Caso Kampala, Uganda

(Humaid & Barhoom, 2012) y Medidores inteligentes, hasta problemas similares abordados en este ámbito: Detección de crímenes y mapas de calor (PredPol), y aplicaciones de la ley de Benford para la detección de fraudes en contabilidad.

En el Capítulo 3 se presenta la metodología de investigación, divido en tres etapas: La primera etapa corresponde al análisis del conjunto de datos, seleccionando la información relevante para comprender el consumo de los clientes de la empresa ESVAL mediante gráficos y tablas. La segunda etapa corresponde a la modelación, donde se detallan los pasos para dar solución al problema. Para comprender el comportamiento del consumo de los clientes se ajustaran modelos de clasificación sobre el conjunto de datos, tales como: regresión logística, bosque aleatorio y máquinas de soporte vectorial. Los resultados de los modelos serán comparados por diferentes medidas de rendimiento, tales como: accuaracy, precision, recall y f-score. Por otro lado, se realiza pruebas de aleatoriedad espacial (basados en cuadrantes y en distancias) sobre las coordenadas espaciales de los clientes que siguen un comportamiento fraudulento, para concluir si el comportamiento de estos clientes sigue algún patrón o su comportamiento es completamente aleatorio.

Finalmente, en el Capítulo 4 se concluye el presente trabajo con algunos comentarios de los resultados obtenidos y se presenta una propuesta para continuar la investigación a modo de trabajos futuros.

Capítulo 2

Preeliminares

En este capítulo se introduce algunas definiciones y resultados básicos sobre los modelos ocupados en este trabajo. También, se realiza una investigación de cómo ha sido abordado este problema en diferentes partes del mundo y de cómo problemas similares a este, podrían dar una visión más amplia en la resolución del mismo.

2.1. Modelos de regresión

Para definir correctamente los modelos de regresión, es conveniente establecer la siguiente notación:

- $\boldsymbol{x}^{(i)} = (x_1^{(i)}, ..., x_k^{(i)})^\top$: Vector de entrada.
- $y^{(i)}$: Variable de salida.
- $\{(\boldsymbol{x}^{(i)}, y^{(i)});\ i = 1, ..., m\}$: Conjunto de entrenamiento.
- $\mathcal{X}$,$\mathcal{Y}$: Espacio de entrada y de salida, respectivamente.

Dado un conjunto de entrenamiento $\{(\boldsymbol{x}^{(i)}, y^{(i)});\ i = 1, ..., m\}$, un modelo de regresión consiste en estimar el valor de $\boldsymbol{y} = (y^{(1)}, ..., y^{(m)})^\top \in \mathcal{Y}$ mediante los valores de $\boldsymbol{X} = (\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(m)})^\top \in \mathcal{X}$.

Existen dos enfoque para resolver estos problemas: paramétrico y no-paramétrico. El **enfoque paramétrico** asume que los datos del conjunto de entrenamiento vienen de una población que sigue una distribución probabilística, basados en un sistema finitos de parámetros. Mientras que para el **enfoque**

no paramétrico, la estructura del modelo no se especifica a priori, sino que se determina a partir del conjunto de entrenamiento.

Dentro de los modelos de regresión, se encuentran los modelos de clasificación, en el cual la variable de salida $y^{(i)}$ solo puede tomar un número pequeño de valores discretos, es decir, $y^{(i)} \in \{0, 1, 2, ..., K\}$ para todo $i = 1, 2, ..., m$. Si $K = 2$, se tiene un problema de **clasificación binaria**, en donde la variable de salida puede tomar los valores 0 ó 1. Si $K > 2$, se tiene un problema de **clasificación multiclase**.

Este trabajo esta enfocado principalmente a los modelos de clasificación binaria. Si bien existe una variedad de este tipo de modelos (para más detalles (Murphy, 2012)), se da énfasis a tres modelos en particular: regresión logística, máquinas de soporte vectorial y bosque aleatorio.

2.1.1. Regresión logística

Corresponde al más simple de los modelos de clasificación binaria. El objetivo de este problema es realizar un símil con un problema de regresión lineal. Para definir adecuadamente el modelo, se considera los siguientes previos:

i) Se define la **función logística** o **función sigmoide** por:

$$g(z) = 1/(1 + e^{-z}).$$

Esta función cumple con las siguientes propiedades:

a) $g(z)$ tiende a 1 cuando $z \to \infty$ y $g(z)$ tiende a 0 cuando $z \to -\infty$.

b) $g'(z) = g(z)(1 - g(z))$.

ii) Sea $\boldsymbol{\theta} = (\theta_0, \theta_1, ..., \theta_k) \in \mathbb{R}^{k+1}$ un vector de parámetros. En base a la función sigmoide, se define la función:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) = g(\boldsymbol{\theta}^\top \boldsymbol{x}^{(i)}) = \frac{1}{1 + e^{\boldsymbol{\theta}^\top \boldsymbol{x}^{(i)}}} \quad , \quad \text{con} \quad \boldsymbol{\theta}^\top \boldsymbol{x}^{(i)} = \theta_0 + \sum_{j=1}^{k} \theta_j x_j^{(i)}$$

Luego, el modelo de regresión logística se define considerando algunos supuestos distribucionales sobre el conjunto de entrenamiento $\{(\boldsymbol{x}^{(i)}, y^{(i)});\ i = 1, ..., m\}$, estos son

$$p(y^{(i)} = 1 | \boldsymbol{x}^{(i)}; \boldsymbol{\theta}) = h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})$$
$$p(y^{(i)} = 0 | \boldsymbol{x}^{(i)}; \boldsymbol{\theta}) = 1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})$$

Es decir, $y^{(i)} \sim \text{Bernoulli}(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))$. Lo anterior, se escribe de forma compacta mediante

$$p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) = (h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))^{y^{(i)}}(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))^{1-y^{(i)}} \tag{2.1}$$

Suponiendo que las m variables son generadas independientemente, la función de verosimilitud queda expresada por

$$\begin{aligned} L(\boldsymbol{\theta}) &= p(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\theta}) \\ &= \prod_{i=1}^{m} p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) \\ &= \prod_{i=1}^{m} (h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})^{y^{(i)}}(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))^{1-y^{(i)}} \end{aligned} \tag{2.2}$$

Luego, la función de log-verosimilitud es

$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \log\, L(\boldsymbol{\theta}) \\ &= \sum_{i=1}^{n} y^{(i)} \log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) \end{aligned} \tag{2.3}$$

Derivando la ecuación (2.3) respecto a la j-ésima componente de $\boldsymbol{\theta}$, esto queda

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}_j}\ell(\boldsymbol{\theta}) &= (\boldsymbol{Y}\frac{1}{g(\boldsymbol{\theta}^\top \boldsymbol{X})} - (\mathbb{1} - \boldsymbol{Y})\frac{1}{g(\boldsymbol{\theta}^\top \boldsymbol{X})})\frac{\partial}{\partial \theta_j} g(\boldsymbol{\theta}^\top \boldsymbol{X}) \\ &= (\boldsymbol{Y}\frac{1}{g(\boldsymbol{\theta}^\top \boldsymbol{X})} - (\mathbb{1} - \boldsymbol{Y})\frac{1}{g(\boldsymbol{\theta}^\top \boldsymbol{X})}) g(\boldsymbol{\theta}^\top \boldsymbol{X})(\mathbb{1} - g(\boldsymbol{\theta}^\top \boldsymbol{X}))\frac{\partial}{\partial \theta_j}\boldsymbol{\theta}^\top \boldsymbol{X} \\ &= (\boldsymbol{Y}(\mathbb{1} - g(\boldsymbol{\theta}^\top \boldsymbol{X})) - (\mathbb{1} - \boldsymbol{Y}) g(\boldsymbol{\theta}^\top \boldsymbol{X})\boldsymbol{X}_{(j)} \\ &= (\boldsymbol{Y} - h_{\boldsymbol{\theta}}(\boldsymbol{X}))\boldsymbol{X}_{(j)} \end{aligned} \tag{2.4}$$

Existen varios métodos para estimar el valor de $\boldsymbol{\theta}$, los más usados se presentan a continuación: método de puntuación de Fisher, gradiente descendente, gradiente descendente estocástico, Newton-Rampson, entre otros métodos (Solomon, 2015).

2.1.2. Máquinas de soporte vectorial

El modelo de máquinas de soporte vectorial (abreviado con las siglas SVM), es un modelo de clasificación multiclase, sin embargo, en esta sección se

desarrolla la teoría para el problema de clasificación binaria. Para definir adecuadamente el modelo, se consideran los siguientes previos:

i) En el conjunto de entrenamiento $\{(\boldsymbol{x}^{(i)}, y^{(i)});\ i = 1, ..., m\}$, se considera que $y^{(i)} \in \{-1, 1\}$, para todo $i = 1, ..., m$.

ii) Un conjunto de entrenamiento $\{(\boldsymbol{x}^{(i)}, y^{(i)});\ i = 1, ..., m\}$ se dice que es **separable** si los conjuntos $\{\boldsymbol{x}^{(i)}, y^{(i)} = 1\}$ y $\{\boldsymbol{x}^{(i)}, y^{(i)} = -1\}$ pueden ser separados por un hiperplano:

$$H = \{\boldsymbol{x}\ ;\ R(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b = \sum_{j=1}^{k} w_j x_j + b\},$$

Es decir, $y^{(i)} R(\boldsymbol{x}^{(i)}) > 0$, para todo $i = 1, ..., m$.

Existen dos escenarios respecto al conjunto de entrenamiento: el caso separable y el caso no separable.

Caso Separable

Cuando el conjunto de entrenamiento es separable, el objetivo es encontrar un hiperplano $H(\boldsymbol{x})$ que separe correctamente el conjunto de entrenamiento, que al mismo tiempo maximice la distancia perpendicular entre dicho hiperplano con el punto más cercano al mismo.

Para comprender mejor el planteamiento del problema, se analiza un conjunto de entrenamiento en el espacio euclidiano (ver Figura 2.1). En este caso, el hiperplano $H(x)$ correspondería a una recta que separa el conjunto de entrenamiento en dos regiones: $\mathcal{R}_1$ y $\mathcal{R}_2$. Se dice que el punto $x \in \mathcal{R}_1$ si $R(x) > 0$, de lo contrario el punto pertenece a la región $\mathcal{R}_2$. Por otro lado, sea $\boldsymbol{x}$ un punto en el espacio, r la distancia de $\boldsymbol{x}$ al hiperplano $H(\boldsymbol{x})$ (el valor de r se denomina **margen**), $\boldsymbol{w}$ el vector normal de r y $\boldsymbol{x}_\perp$ la proyección ortogonal de $\boldsymbol{x}$. Entonces, $\boldsymbol{x}$ se puede expresar como

$$\boldsymbol{x} = \boldsymbol{x}_\perp + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$$

Por tanto, el hiperplano luce como en la Figura 2.1, quedando definido matemáticamente por la expresión

$$H(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b = (\boldsymbol{w}^\top \boldsymbol{x}_\perp + b) + r\frac{\boldsymbol{w}^\top \boldsymbol{w}}{\|\boldsymbol{w}\|}$$

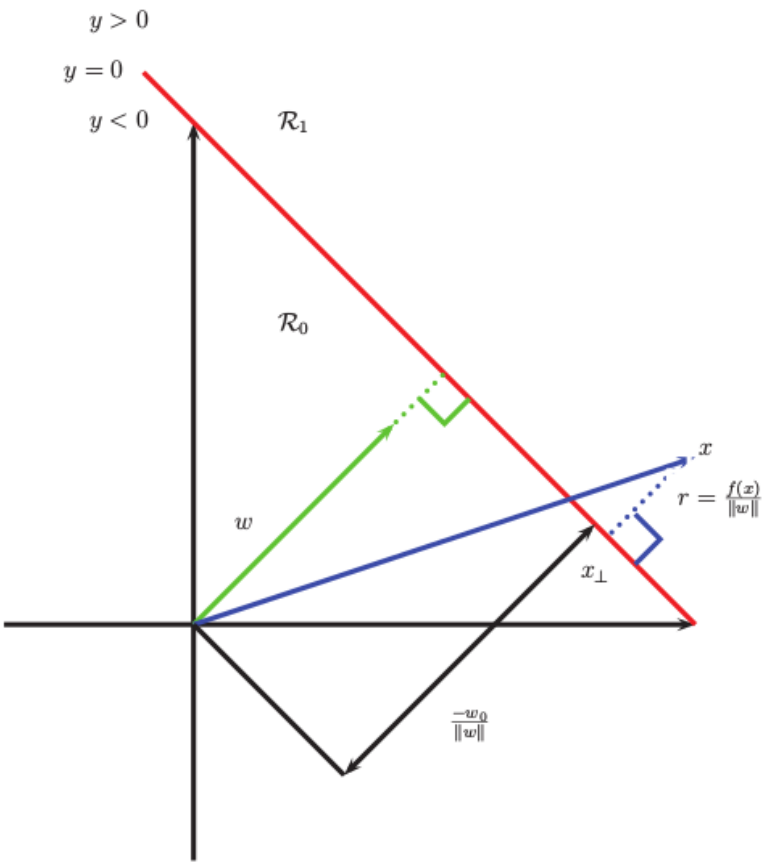


Figura 2.1: Esquema del modelo SVM en el plano Euclidiano.

Notar que: $H(\boldsymbol{x}_\perp) = 0$, así $\boldsymbol{w}^\top \boldsymbol{x}_\perp + b = 0$. Entonces, el hiperplano se puede escribir como: $H(\boldsymbol{x}) = r\frac{\boldsymbol{w}^\top \boldsymbol{w}}{\sqrt{\boldsymbol{w}^\top \boldsymbol{w}}}$ y $r = \frac{H(\boldsymbol{x})}{\|\boldsymbol{w}\|}$.
En este contexto, se estaría buscando que la distancia $r = H(\boldsymbol{x})/\|\boldsymbol{w}\|$ sea lo más grande posible, como se muestra en la Figura 2.2. En particular, podrían haber muchas líneas que separen perfectamente los datos de entrenamiento (especialmente si se trabaja en un espacio de alta dimensión), pero intuitivamente, el mejor a escoger es aquel que maximiza el margen, es decir, la distancia perpendicular al punto más cercano. Además, se busca que asegurar que cada punto se encuentre en la región correcta del hiperplano, es decir, se quiere que $H(x_i)y_i > 0$.

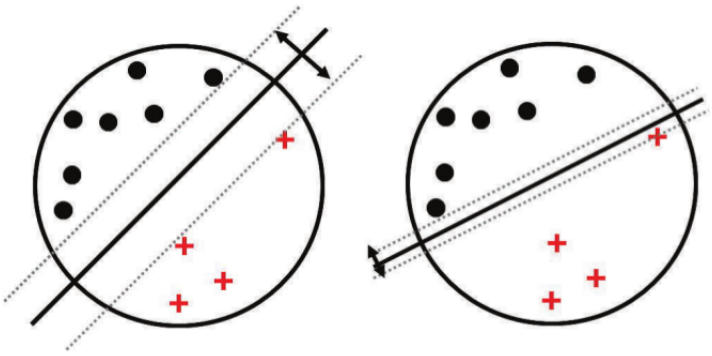


Figura 2.2: Ilustración del largo de los margenes principales. Izquierda: un hiperplano separador con un margen grande. Derecha: un hiperplano separador con un margen pequeño.

Por lo tanto, la función objetivo de este problema queda formulada matemáticamente por

$$(\mathrm{P}) : \max_{\boldsymbol{w},b}\Big(\min_{i=1,\dots,m} \frac{y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^i + b)}{\|\boldsymbol{w}\|}\Big) \tag{2.5}$$

Re-escalando los parámetros por: $\boldsymbol{w} \to k\boldsymbol{w}$ y $b \to kb$, lo anterior no cambia la distancia de todos los puntos al hiperplano en cuestión, esto se debe a que el factor k se cancela cuando se divide por $\|\boldsymbol{w}\|$. Luego, se define el valor de escala $y^{(i)}H(\boldsymbol{x}^{(i)}) = 1$ (con $i = 1, ..., m$) para los puntos más cercanos al hiperplano.
Por lo tanto, la ecuación (2.5) se puede reescribir por

$$\text{(P):} \begin{cases} \min\limits_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.a.} : y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 \ , \ i = 1, ..., n \end{cases} \tag{2.6}$$

La restricción indica que todos los puntos que se encuentren en el lado correcto del hiperplano, tengan al menos un margen de 1. Por esta razón, se dice que un SVM es un ejemplo de **clasificador de grandes margenes**.

Caso No Separable

En primera instancia, la idea es llevar el problema del caso no separable a un problema del caso separable. Para ello, se busca una función $\phi : \mathbb{R}^k \to \mathbb{R}^K$ (usualmente $K \gg k$), tal que los conjuntos $\{\boldsymbol{z}^{(i)} = \phi(\boldsymbol{x}^{(i)}), y^{(i)} = 1\}$ y $\{\boldsymbol{z}^{(i)} = \phi(\boldsymbol{x}^{(i)}), y^{(i)} = -1\}$ sean linealmente separables.

Por otro lado, si no existen restricciones respecto a la asignación de ϕ, esto puede conducir a que el problema de hallar un clasificador lineal en $\mathbb{R}^K$, sea computacionalmente intratable.

La solución, es escoger funciones particulares de ϕ, denominadas funciones de **kernelización**. Para esto se escoge $\phi : \mathbb{R}^k \to \mathbb{R}^K$ tal que:

$$\langle \phi(x), \phi(y) \rangle = \kappa(x, y)$$

para algún kernel $\kappa : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$. Aquí, $\langle \cdot, \cdot \rangle$ denota el producto escalar en $\mathbb{R}^K$ (Para más detalles, consultar el Apéndice C) .

Si después de aplicar las funciones de kernelización sobre el conjunto de entrenamiento, este sigo siendo un conjunto no separable, se re-formula la ecuación (2.6), añadiendo un vector de holgura $\boldsymbol{\xi} = (\xi_1, ..., \xi_m)$, con $\xi_i \geq 0$, para todo $i = 1, ..., m$.

Se tiene que: $\xi_i = 0$, si el punto se encuentra en o dentro del borde marginal correcto, en caso contrario, $\xi_i = |y^{(i)} - H(\boldsymbol{x}^{(i)})|$. Si $0 < \xi_i \leq 1$, el punto se encuentra dentro del margen pero no en el lado correcto del hiperplano. Si $\xi_i > 1$, el punto se encuentra en el lado equivocado del hiperplano (ver Figura 2.3).

Ahora, se reemplaza la condición $y^{(i)}H(\boldsymbol{x}^{(i)} \geq 1$ por $y^{(i)}H(\boldsymbol{x}^{(i)}) \geq 1 - \xi_i$ (esta condición se conoce con el nombre de **condición de margen suave**). Por lo tanto, la nueva función objetivo es

$$\text{(P):} \begin{cases} \min\limits_{\boldsymbol{\xi},\boldsymbol{w},b} \ \dfrac{1}{2}\|w\|^2 + C\sum\limits_{i=1}^{n}\xi_i \\ \text{s.a. :} \quad \xi_{(i)} \geq 0 \ , \ i = 1, ..., n \\ \quad y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_{(i)} \ , \ i = 1, ..., n \end{cases} \tag{2.7}$$

donde el parámetro C corresponde a un parámetro de regularización, cuya función es controlar el número de errores que se está dispuesto a tolerar en el conjunto de entrenamiento. Comúnmente se define $C = 1/\nu n$, donde $0 \leq \nu \leq 1$ controla la fracción de los puntos mal clasificados permitidos.

La ecuación (2.7) corresponde a un problema con restricciones que puede ser resuelto usando los multiplicadores de Lagrange. La función objetivo adopta la forma de un problema tipo **QP**("Quadratic Program"), cuyo tiempo de ejecución es del orden de $\mathcal{O}(n^3)$. Existen algoritmos especializados que evitan soluciones genéricas QP, por ejemplo, el algoritmo SMO ("Sequential minimal optimization") (Platt, 1998), que en la práctica tiene un orden de $\mathcal{O}(n^2)$. No obstante, esto puede ser lento si n es demasiado grande, por lo que se frecuenta ocupar los SVM lineales, cuyo tiempo ejecuación es de $\mathcal{O}(n)$ (Joachims (2006) ; Bottouet al. (2007)).

Se puede probar que la solución de la ecuación (2.7) es:

$$\hat{\boldsymbol{w}} = \sum_i \alpha_i \boldsymbol{x}_i, \tag{2.8}$$

donde $\alpha_i = \lambda_i y_i$ y $\alpha = (\alpha_1, ..., \alpha_m)$ es vector con una gran cantidad de ceros. Los x_i para los cuales $\alpha_i > 0$ son llamados vectores de soporte. Estos puntos son clasificados incorrectamente o son clasificados correctamente pero se encuentran en o dentro del margen.

Por tanto, la predicción viene dada por:

$$\hat{y}(\boldsymbol{x}) = \text{sgn}(\hat{\boldsymbol{w}}^\top \boldsymbol{x} + \hat{b}). \tag{2.9}$$

2.1.3. Bosque aleatorio

Un árbol de decisión, se define como una partición recursiva del espacio de entrada R, definiendo con ello un modelo local en cada partición realizada.

Sea R_m es la m-ésima partición de la región, w_m es la respuesta media en aquella región y $\boldsymbol{v}_m$ codifica la elección de la variable a partir de la cual esta se dividirá, y el valor del umbral, es la trayectoria de la raíz a la m-ésima hoja. El modelo se escribe como:

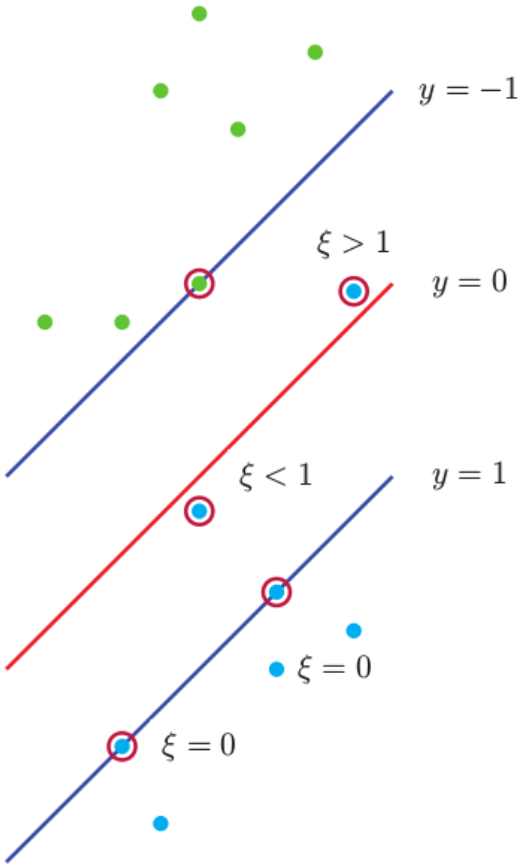


Figura 2.3: Ilustración del margen principal suave. Los puntos con círculos a su alrededor son vectores de soporte. También se indica el valor de las correspondientes variables de holgura. Basado en la Figura 7.3 de (Obispo 2006a).

$$f(x) = \mathbb{E}(y \mid \boldsymbol{x}) = \sum_{m=1}^{M} w_m \mathbb{1}_{\{\boldsymbol{x} \in R_m\}} = \sum_{m=1}^{M} w_m \phi(\boldsymbol{x} \ ; \ \boldsymbol{v}). \tag{2.10}$$

Se observa que un árbol de decisión queda definido por funciones de bases adaptativas. Por otro lado, estas estimaciones tienen un número considerables de variables, buscando alternativas para reducirlas. Un camino para solucionar este problema es considerar el promedio conjunto de muchas estimaciones. Por ejemplo, se puede entrenar M árboles diferentes (f_m) definidos por la ecuación (2.10) sobre subconjuntos diferentes de datos elegidos al azar con reemplazo, y luego calcular la función

$$f(x) = \sum_{m=1}^{M} \frac{1}{M} f_m(x) \tag{2.11}$$

Esta técnica es llamada "Bagging" (Breiman, 1996), lo que significa "bootstrap aggregating".

Las desventajas que posee este método es que al volver a ejecutar el mismo algoritmo de aprendizaje en diferentes subconjuntos de datos, puede resultar en predictores altamente correlacionados, lo que limita la reducción de la varianza.

Esta técnica es conocida como Bosque Aleatorio (Breiman, 2001). Este tipo de modelos suelen tener una precisión predictiva muy buena (Caruana & Niculescu-Mizil , 2001) y se han usado ampliamente en muchas aplicaciones, por ejemplo, para el reconocimiento de la pose del cuerpo usando el sensor del Kinect de Microsoft (Shotton et al., 2011).

También es posible desarrollar un enfoque bayesiano del aprendizaje de árbol. En particular (Wu et al., 2007), realizaron inferencia aproximada sobre el espacio de los árboles (tanto en la estructura como los parámetros) utilizando técnicas del tipo Markov Chain Monte Carlo (MCMC). Esta metodología redujo la varianza.

2.2. Medidas de rendimiento de los modelos

La **matriz de confusión** es una herramienta que permite la visualización del desempeño de los modelos de clasificación. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real.

Para el caso de los problemas de clasificación binaria, la matriz de confusión corresponde a una matriz cuadrada de 2×2. Las entradas de esta matriz se explican considerando el siguiente experimento: se tienen P instancias positivas y N instancias negativas. Por lo tanto, la matriz de confusión se define según la Figura 2.4, en donde:

- **Verdadero Positivo**(TP): Número de veces donde hubo una instancia positiva P, dado que hubo una instancia positiva P.
- **Falso Positivo**(FP): Número de veces donde hubo una instancia positiva P, dado que hubo una instancia negativa N.
- **Verdadero Negativo**(TN): Número de veces donde hubo una instancia una instancia negativa N, dado que hubo una instancia negativa N.
- **Falso Negativo**(FN): Número de veces donde hubo una instancia una negativa N, dado que hubo una instancia positiva P.

		Clase real	
		Clase referencia	***Clase no referencia***
Clase estimada	Clase referencia	TP	FP
	Clase no referencia	FN	TN

Figura 2.4: Matriz de confusión para la clasificación binaria.

A partir de los elementos de la matriz de confusión, se definen las siguientes medidas de rendimientos:

a) **accuracy**: En el campo de la recuperación de información, la medida "accuracy" corresponde al total de documentos recuperados en la consulta. Esta se define por:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.12}$$

b) **precision**: En el campo de la recuperación de información, la medida "presicion" corresponde a la fracción de documentos recuperados que son relevantes para la consulta. Esta se define por:

$$\text{precision} = \frac{TP}{TP + FP} \tag{2.13}$$

c) **recall**: En el campo de la recuperación de información, la medida "recall" corresponde a la fracción de los documentos relevantes que se recuperan con éxito. Esta se define por:

$$\text{recall} = \frac{TP}{TP + TN} \tag{2.14}$$

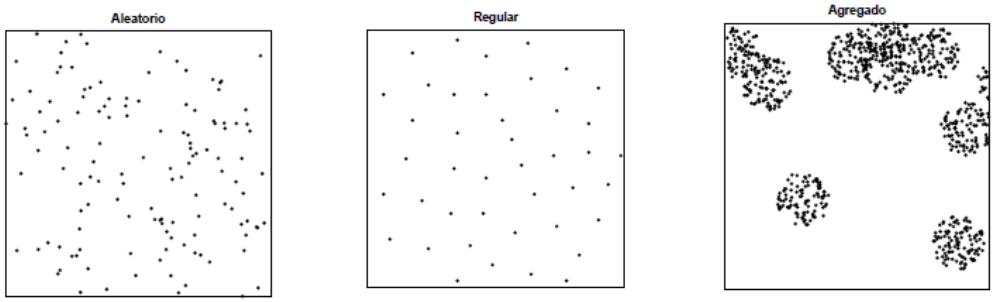
d) **f-score**: Corresponde a la media armónica entre las medidas "presicion" y "recall". Esta se define por:

$$\text{f-score} = 2\ \frac{\text{accuracy} \cdot \text{precision}}{\text{accuracy} + \text{precision}} \tag{2.15}$$

2.3. Análisis de patrones puntuales

Sea $\{Z(s) : s \in D \subset \mathbb{R}^d\}$ un proceso estocástico, en el que s es la ubicación en el espacio Euclidiano d-dimensional. Un **proceso puntual** se define como un arreglo o patrón de puntos en un conjunto aleatorio D. Estos puntos se denominan los eventos del proceso. Si los eventos son observados sólo parcialmente, el patrón se llama un patrón muestreado ("Sampled point mapped"). Cuando todos los eventos de la realización se registran, se dice que este es puntual.

Sea D un conjunto aleatorio, el experimento que genera una realización particular puede ser visto como un sorteo de lugares en D de los eventos que son observados. Los patrones puntuales son realizaciones de experimentos aleatorios y se distingue entre patrones (completamente) aleatorios, agrupados (especialmente agregados), y los regulares, lo cual no debe conducir a la falsa impresión de que los dos últimos tipos de patrones no contienen ningún tipo de aleatoriedad. Ejemplo de estos tres tipos de patrones aleatorios se observan en la Figura 2.5.



(a) Patrones puntuales con distribución aleatoria.

(b) Patrones puntuales regulares.

(c) Patrones puntuales agregados.

Figura 2.5: Tipo de patrones puntuales.

Un patrón puntual se llama completamente aleatorio si se cumplen los siguientes requisitos:

a) El promedio de eventos por unidad de área, la intensidad $\lambda(s)$, es homogénea a lo largo D.

b) El número de eventos en dos sub-regiones que no se solapan, A_1 y A_2 son independientes.

c) El número de eventos en cualquier sub-región sigue una distribución de Poisson.

Por lo tanto, los eventos se distribuyen uniforme e independiente a lo largo del dominio, proceso que se reconoce matemáticamente como un proceso Poisson homogéneo, y que sirve como hipótesis nula para muchas investigaciones estadísticas en los patrones puntuales.

Test de aleatoriedad

Método basado en cuadrantes

Otro tipo de métodos para probar la aleatoriedad espacial de un patrón son basados en la división del dominio D en sub-regiones no traslapadas (cuadrantes) $A_1, ..., A_k$ de igual tamaño. Sean $A_1, ..., A_k$ tal que $\cup_{i=1}^{k} A_i = D$, donde A_i y A_j no se traslapan.

Test de Bondad de ajuste Chi-Cuadrado

Esta prueba de ajuste estadístico, prueba la hipótesis nula, en que los n puntos estarían distribuidos uniformemente independientes a lo largo de D, o en otras palabras, los conteos de sitios por cuadrante son variables Poisson independientes con media común.
El estadístico de prueba esta dado por:

$$X^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \sim \chi^2_{(rc-1)} \tag{2.16}$$

Ahora, si $O_{ij} = n_{ij}$, $E_{ij} = \overline{n} = n/rs$, con r y s definidos, $\hat{\lambda} = n/|A|$, $N(A) \sim$ Poisson(λA). La ecuación (2.16) se puede escribir de la siguiente forma

$$X^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{(n_{ij} - \overline{n})^2}{n_{ij}} \sim \chi^2_{(rc-1)} \tag{2.17}$$

Índice de Dispersión

Si $X \sim$ Poisson(λ),$\mathbb{E}(X) = \mathbb{V}(X) = \lambda$. Entonces, el índice de dispersión se define por:

$$I = \frac{\mathbb{V}(N(A))}{\mathbb{E}(N(A))} = \frac{S^2}{\overline{n}}, \tag{2.18}$$

donde $S^2 = \sum\sum(n_{ij} - \overline{n})^2/rc - 1$. Considerando que $(I-1)$ indica el tamaño del cluster, se definen las siguientes reglas de decisión:

I) Si $(I-1) \approx 0$, el patrón es completamente aleatorio.

II) Si $(I-1) > 0$, el patrón es agregado.

III) Si $(I-1) < 0$, el patrón es regular.

Métodos basados en distancias

La elección de la forma y el número de cuadrantes para probar la hipótesis de aleatoriedad basados en conteos por áreas es un elemento que puede influenciar los resultados. Las pruebas estadísticas basadas en distancias entre eventos o entre puntos muestreados y eventos elimina estas características. En esta sección se nombran las pruebas que tienen en cuenta tales distancias entre eventos.

Función G(h)

Esta función tiene en cuenta la mínima distancia entre eventos(distancia al vecino mas cercano). Se define:

- d_i = Distancia mínima entre un evento y sus vecinos.
- d = Variable aleatoria de mínima distancia de un punto a un evento.
- $G(d)$ = Función de distribución de d.

La función de distribución empírica de $G(d)$ viene dada por:

$$\hat{G}(d) = \frac{\sum \mathbb{1}_{\{d_i \leq d\}}}{n} \tag{2.19}$$

Por otro lado, la distribución teórica de $G(d)$ viene dada por

$$\begin{aligned} G(d) &= \mathbb{P}((D \leq d)) \\ &= 1 - \mathbb{P}((D > d)) \\ &= 1 - \mathbb{P}(\text{No hay eventos en } A = \pi d^2). \end{aligned} \tag{2.20}$$

Si $N(A) \sim \text{Poisson}(\lambda\pi d^2)$, entonces $P(N(A) = 0) = e^{-\lambda\pi d^2}$. Por lo tanto:

$$G(d) = 1 - e^{-\lambda\pi d^2} \tag{2.21}$$

de donde

- Si $\hat{G}(d)$ crece rápidamente en distancias cortas, los eventos son agregados.
- Si los eventos son regularmente espaciados, $\hat{G}(d)$ crece lentamente hasta cierta distancia (espacio eventos) y después crece rápidamente.

Adicionalmente, se hace un cálculo de las bandas de confianza para la función en donde se calcula:

1) $\hat{G}_0(h)$ = Función $G(H)$ calculada con los datos observados.
2) $\hat{G}_1(h), ..., \hat{G}_g(h)$ = Función $G(h)$ calculada con g realizaciones de un proceso completamente aleatorio.
3) $G_L(h) = \min\limits_{i=1,...,g} \hat{G}_i(h)$ y $G_U(h) = \max\limits_{i=1,...,g} \hat{G}_i(h)$

Función F(h)

Tiene en cuenta la mínima distancia punto-evento, donde un punto es un evento escogido aleatoriamente dentro de la región de estudio. Para su construcción se siguen los siguientes pasos:

1) Seleccionar aleatoriamente m puntos $\{p_1, ..., p_m\}$.
2) Calcular $d_i = d(p_i, s_i)$, la distancia de cada punto escogido al sitio del evento más cercano.

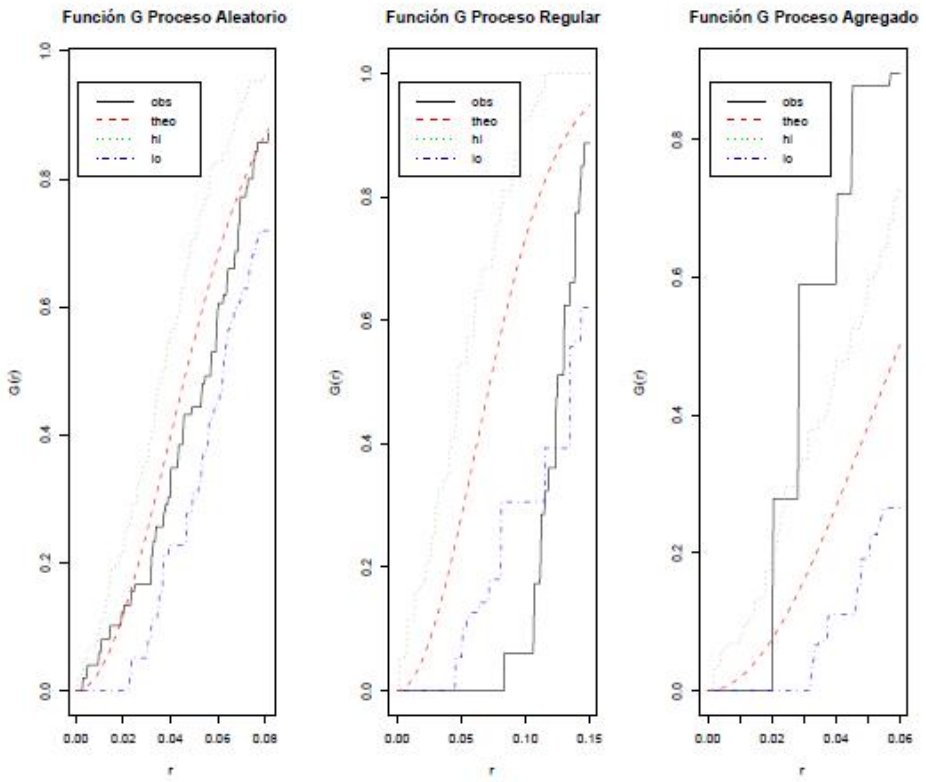
Sea d la variable aleatoria de mínima distancia de un punto a un evento, entonces:

$$F(d) = \mathbb{P}(D \leq d) \quad , \quad \hat{F}(d) = \frac{\sum \mathbb{1}_{\{d_i \leq d\}}}{m} \tag{2.22}$$

Luego, la función teórica es

$$F(d) = 1 - e^{-\lambda\pi d^2}. \tag{2.23}$$

de donde



(a) Patrones puntuales con distribución aleatoria.

(b) Patrones puntuales regulares.

(c) Patrones puntuales agregados.

Figura 2.6: Gráficas del la función G para cada tipo de proceso.

- Si $\hat{F}(d)$ crece lentamente al comienzo y rápidamente para distancia largas, el patrón es agregado.
- Si $\hat{F}(d)$ crece rápido al comienzo, el patrón es regular.

Por otro lado, sea x la distancia mínima de un punto a un evento. Para un patrón completamente aleatorio se tiene que:

$$F(x) = 1 - e^{-\lambda \pi x^2}, \tag{2.24}$$

donde

$$\mathbb{E}(x) = \frac{1}{2\sqrt{\lambda}} \quad , \quad \mathbb{V}(x) = \frac{4-\pi}{4\pi\lambda}. \tag{2.25}$$

Ahora, si se tiene $x_1, ..., x_m$:

$$\mathbb{E}(\overline{x}) = \frac{1}{2\sqrt{\lambda}} \quad , \quad \mathbb{V}(\overline{x}) = \frac{4-\pi}{4\pi\lambda m}. \tag{2.26}$$

Luego, si se define $G(y) = 1 - e^{-\lambda\pi y^2}$, con y = distancia mínima al evento más cercano. Dado $y_1, ..., y_m$, se observa que

$$\mathbb{E}(\overline{y}) = \frac{1}{2\sqrt{\lambda}} \quad , \quad \mathbb{V}(\overline{y}) = \frac{4-\pi}{4\pi\lambda m}. \tag{2.27}$$

De la ecuación (2.26) y la ecuación (2.27), se concluye que bajo aleatoriedad $G(y)$ y $F(y)$ son iguales.

Función K(h) de Ripley

La función K de Ripley (1976) es una función de λ_2 para procesos estacionarios e isotrópicos. También es conocida como la medida reducida del segundo momento (Cressie, 1998), y es la función reducida del segundo momento.

Sea $\mathcal{A}$ = Número de eventos adicionales a una distancia d de un evento elegido aleatoriamente. La función K de Ripley se define por

$$K(d) = \frac{\mathbb{E}(\mathcal{A})}{\lambda}. \tag{2.28}$$

Sea $\hat{\lambda} = n/A$, con A = Área de la región D, d_{ij} = distancia entre el i-ésimo y j-ésimo evento y $I_d(d_{ij}) = \begin{cases} 1 \text{ si } d_{ij} < 0 \\ 0 \text{ en otro caso.} \end{cases}$.

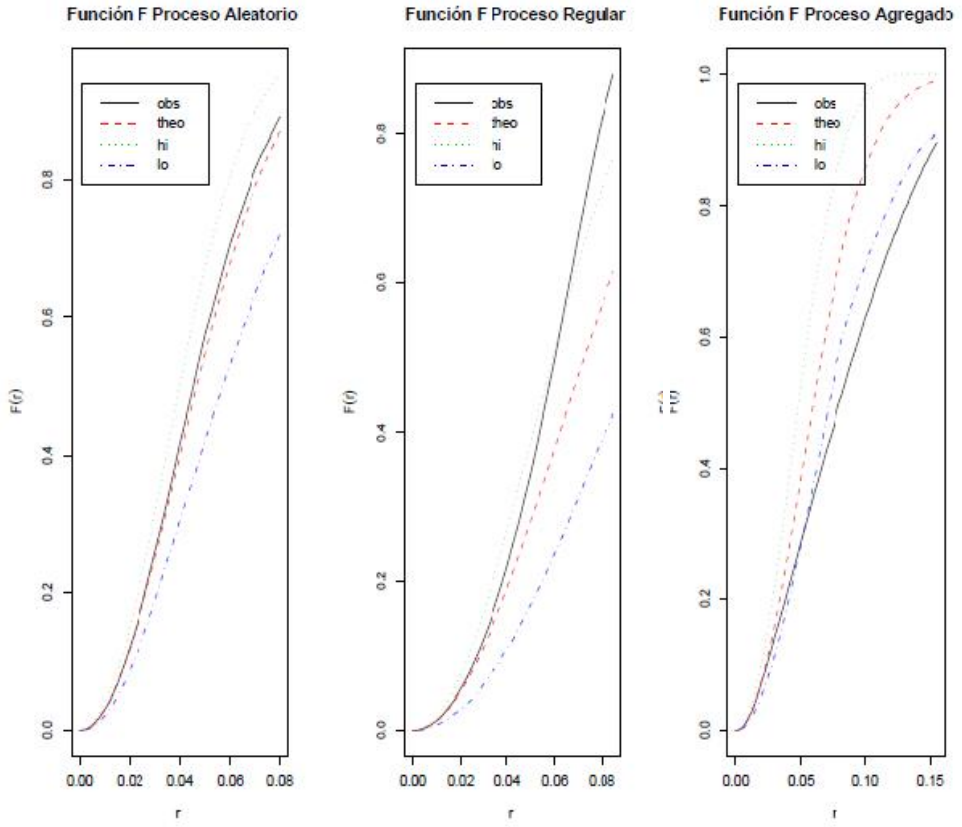
Luego, la función K de Ripley estimada, esta dada por

$$\hat{K}(d) = \frac{\sum\sum_{i \neq j} I_d(d_{ij})}{\hat{\lambda}}. \tag{2.29}$$

Bajo aleatoriedad:

$$K(d) = \lambda\pi d^2, \tag{2.30}$$

en donde: 1) si $\hat{K}(d) < \lambda\pi d^2$, hay regularidad, 2) si $\hat{K}(d) > \lambda\pi d^2$, hay agregación.



(a) Patrones puntuales con distribución aleatoria.

(b) Patrones puntuales regulares.

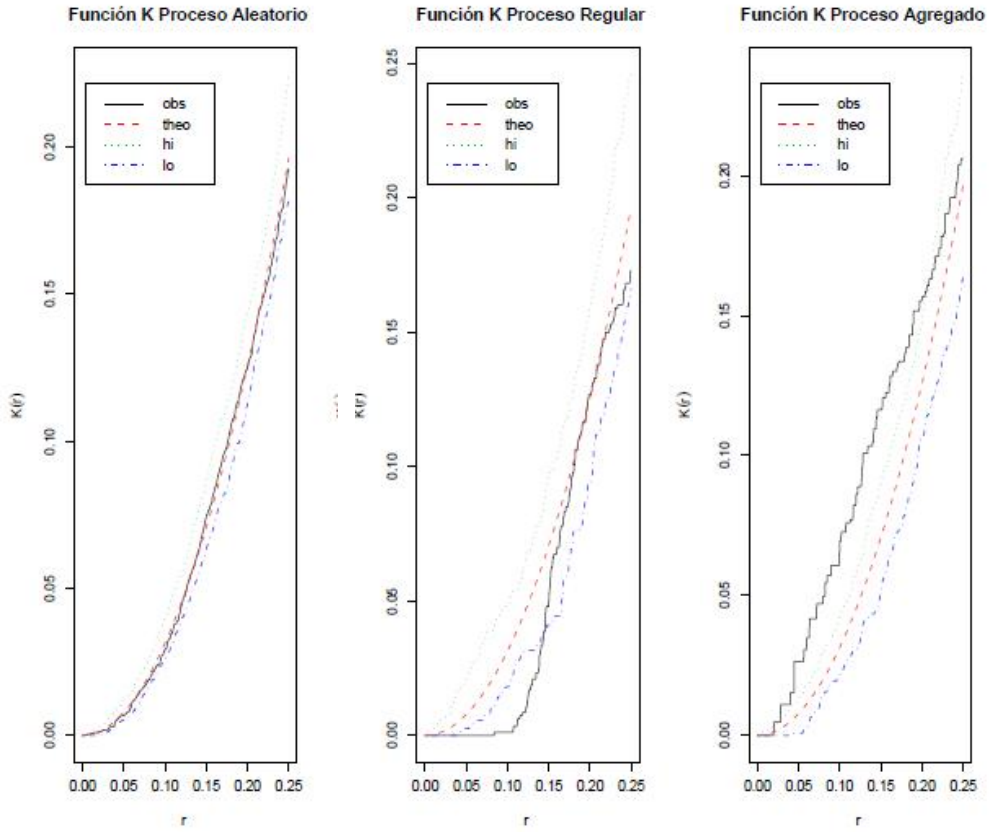
(c) Patrones puntuales agregados.

Figura 2.7: Gráficas del la función F para cada tipo de proceso.

2.4. Estudios alternativos

En términos generales el problema, tanto la cuantificación como la detección de pérdidas en sistemas de agua potable ha sido estudiado de manera consistente, sin embargo, en términos mayoritarios el enfoque investigativo se ha inclinado por la detección de pérdidas físicas y en particular en lo referido a las filtraciones en líneas de distribución, de esta forma es posible encontrar estudios enfocados en la detección mediante el análisis de vibraciones en tuberías soterradas, utilización de algoritmos genéticos, etc. Sin embargo el estudio de las pérdidas aparentes resulta ser mucho más fragmentario.

En esta sección se estudia cómo ha sido abordado el problema, tanto a nivel local (enfocados solo en el ámbito de agua potable) como a nivel global



(a) Patrones puntuales con distribución aleatoria.

(b) Patrones puntuales regulares.

(c) Patrones puntuales agregados.

Figura 2.8: Gráficas del la función K de Ripley para cada tipo de proceso.

(modelos de detecciones de fraudes en otros ámbitos).

2.4.1. Caso Kampala, Uganda

Manteniendo siempre en consideración que los valores particulares en torno a las pérdidas no son necesariamente extrapolables entre poblaciones de características disímiles, resulta valioso considerar el enfoque de cuantificación de pérdidas aparentes o comerciales utilizado en la ciudad de Kampala, Uganda (Humaid & Barhoom, 2012).

Se proponen separar el análisis en base a grupos causales de pérdidas:

i) **Medidores imprecisos**: bajo la premisa de que, al igual que cualquier dispositivo mecánico, los medidores se ven sometidos al desgaste, su-

friendo, como consecuencia, una merma en su precisión, de esta manera, y siguiendo métodos de muestreo estadístico previamente establecidos, se consideraron categorías de medidores, de acuerdo a su antigüedad, y se realizaron pruebas con distintos niveles de flujo.

ii) **Errores en la lectura**: Dado que en Kampala, al igual que en Chile, la recolección de datos de consumo, se realiza de manera manual, por inspectores relativamente poco capacitados, existiendo errores humanos inherentes en la lectura.

iii) **Manejo de datos y errores de facturación**: al comparar los datos entregados por los inspectores al realizar la lectura se detectan nuevos errores.

iv) **Consumos no autorizados**: se considera la alteración de medidores, la instalación de bypass, reposición no autorizada de servicio y conexiones no autorizadas a la red. Una vez detectados, se calcula el volumen de consumo no autorizado, como un promedio de los consumos mensuales previos a la falta, lo anterior en base a la detección mediante auditorías en terreno.

La estimación de la desagregación de las pérdidas aparentes se expresan por:

i) **Medidores imprecisos**: $22 \pm 2\,\%$ de la facturación.

ii) **Errores en la lectura**: $1,4 \pm 1\,\%$ de la facturación.

iii) **Manejo de datos y errores de facturación**: $3,5 \pm 0,5\,\%$ de la facturación.

iv) **Consumos no autorizados**: $10 \pm 2\,\%$ de la facturación.

Para la solución del problema, se propone una estrategia de modelación, desde la consistencia de los datos hasta la propuesta de los modelos con sus respectivos resultados. Los modelos abordados son: máquinas de soporte vectorial y redes neuronales.

2.4.2. Medidores inteligentes de agua potable

Un enfoque alternativo al desarrollo clásico de este tipo de problemas, es reemplazar los antiguos medidores de agua potable por medidores inteligentes. Las ventaja que se tiene sobre los antiguos medidores de agua potables son múltiples, en los que se destacan:

i) **Información**: Existe un sensor remoto que se utiliza para recepcionar la información del cliente a 100 metros de distancia del medidor inteligente. Con esto se evita errores de medición humano cometidos por los inspectores que van a tomar el estado del agua a terreno.

ii) **Manipulación**: Se tiene conocimiento si el medidor ha sido dañado o intervenido por terceros.

iii) **Cortes**: EL medidor inteligente cuenta con una válvula de corte de agua. Esto evita el rompimiento del suelo para intervenir la matriz, evitando costos a las empresas y malos ratos a los clientes.

Este tipo de medidores está muy lejos de ser implementados por las empresas distribuidoras de agua en Chile, debido a los enormes costos asociados.

2.4.3. Fraude en el sistema financiero

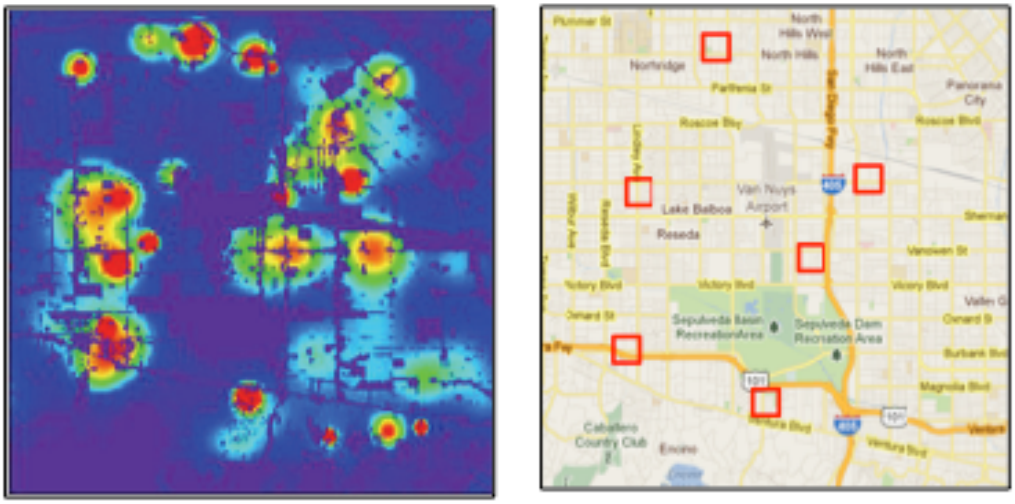
El fraude es un problema serio que enfrentan los emisores de tarjetas de crédito. Las transacciones con tarjeta de crédito tuvieron una pérdida total de 800 millones dólares de fraude en los Estados Unidos el 2004. En Reino Unido, en el mismo año, la pérdida causada por el fraude con tarjeta de crédito fue de 425 millones de libras (aproximadamente 750 millones de dólares EE.UU.). EL retraso de la gestión del riesgo se convierte en uno de los mayores obstáculos para el crecimiento de la rentabilidad. Por tanto, la gestión de riesgos de tarjetas de crédito se convierte en uno de los temas más importantes para los investigadores en el sector financiero privado.

Debido a la importancia del asunto, se ha desarrollado una basta teoría de modelos de predicción en el fraude financiero. La mayoría de estas investigaciones va enfocada en modelos estudiados en la rama de estadística y la computación. Es en esta última rama donde se han enfocado con mayor énfasis, debido al gran impacto que han causado la sub-ramas de la programación como lo son el aprendizaje de máquinas y la minería de datos.

Para más información, ver algunas de las siguientes referencias: Quah & M. Sriganesh (2008), Duman & Ozcelik (2011), Ngai et al. (2011).

2.4.4. PredPol

Predpol (en inglés “Predictive Policing”), es un software que tiene como objetivo mostrar las zonas geográficas en donde es más probable que se cometa un crimen dado que se dio un crimen en cierto espacio-tiempo. El algoritmo que utiliza PredPol necesita tres entradas: tipo de crimen, lugar donde



(a) Mapa de Calor. **(b)** Lugares más probables de crimenes.

Figura 2.9: Esquema del software PredPol.

se cometió y fecha exacta cuando tuvo lugar. Una vez ingresada esta información, el software muestra dos imágenes. La primera imagen (Figura 2.9a) corresponde un mapa de calor, en el cual se señala donde es más probable que suceda un crimen en las proximas horas. La segunda imagen (Figura 2.9b) corresponde a un mapa satelital, en el cual se señala con sectores rectangulares los lugares donde deben visitar los agentes de turnos.

Este software ha sido implementado en varias oficinas de los Estados Unidos: Florida, Miami, Atlanta, Los Angeles, entre otros estados. Por ejemplo, en el año 2013 , en el estado de Atlanta, el número de crímenes disminuyó en un 13 %.

Por otro lado, no existen publicaciones científicas que detallen los modelos ocupados en el algoritmo, limitándose los alcance del método.

Capítulo 3

Modelamiento y resultados

En esta sección se detallan los pasos a seguir en el modelamiento y resolución del problema. Al final de esta sección se presentan los resultados obtenidos durante la investigación.

3.1. Análisis de los datos

El conjuntos de datos recabados corresponden a los clientes de la Empresa de Servicios Sanitarios de Valparaíso (ESVAL). La información proporcionada se encuentra segregada en los siguientes conjuntos de datos.

a) **Datos de consumo**: Información del consumo mensual de los clientes entre los años 2010 al 2014.

b) **Datos de localía**: Información de la localía (o comuna) a la cual pertenece el cliente.

c) **Datos de localización espacial**: Información de la ubicación del cliente en coordenadas espaciales.

d) **Datos de clientes fraudulentos**: Información de la fecha y tipo de fraude cometidos por el cliente.

3.1.1. Preparación de los datos

Del conjunto de datos disponibles, se crea una nueva base de datos con toda la información proporcionada. En la tabla Tabla 3.1 se definen las variables del modelo.

Tabla 3.1: Definición de las Variables

Variable	Tipo	Definición
id cliente	Caracter	Identificador único del cliente
localidad	Caracter	Indica la localidad a la cual pertenece el cliente
coordenadas	Numérico	Coordenadas espaciales del cliente
fecha fraude	Caracter	Indica la fecha del fraude
tipo fraude	Caracter	Indica el tipo de fraude
indicador fraude	Numérico	1: Se comete fraude durante el 2010-2014, 0: e.o.c.
consumo	Numérico	Consumo del cliente en el mes yy durante el año xx.

3.1.2. Análisis exploratorio de los datos

En esta sección se presenta un resumen de las principales características del conjunto de datos, mediante tablas y/o gráficos. El fin del análisis exploratorio es comprender la naturaleza de la problemática, y con ello establecer los criterios necesarios para dar paso a la etapa de la modelación.

a) Información general del conjunto de datos

El conjunto de datos contiene la información de aproximadamente 850 mil clientes de la empresa ESVAL, estos se encuentran repartidos en 54 comunas de la V región. Respecto a los datos de consumo, se tiene el consumo mensual de los clientes entre los años 2010 y 2014, correspondientes a un total de 60 meses de consumo.

Por otro lado, existen clientes que presentan información incompleta en su consumo, localía y/o ubicación. , es decir, se está en presencia de un problema de **datos perdidos**. Dado que no se asume de momento ningún supuesto distribucional, se opta por separar el conjunto de datos en dos conjuntos:

- **Información Real**: En ella se tiene la información de todos los clientes.
- **Información Efectiva**: En ella se tiene la información de todos los clientes que no presentan problemas de datos perdidos.

En la Tabla 3.2 se compara la información disponibles en ambos conjuntos.

De aquí en adelante, se considera el conjunto de datos correspondiente a la información efectiva disponible.

Tabla 3.2: Comparación entre la información real y la información efectiva.

Descripcion	Informacion total	Informacion efectiva	% de uso
Total clientes	843.833	259.796	30.79
Total clientes fraudulentos	59.692	25.793	43.21
Total clientes fraudulentos esp.	11.932	3.915	32.81

En la Tabla C.3 (ver apéndice C) se muestra la información de los tipos de clientes y los porcentajes relativos de clientes fraudulentos correspondiente a cada comuna. Al considerar solo los clientes fraudulentos especiales, el porcentaje relativo de estos no supera el 5 % del total de la población. Cabe recordar que según la Superintendencia de Servicios Sanitarios alrededor de un 5 % a un 10 % corresponde a clientes fraudulentos, por lo tanto existen clientes fraudulentos que no han sido detectados.

b) Tipo de cliente

Los clientes son clasificados en dos categorías basado en su consumo histórico: clientes con un consumo normal y clientes con un consumo fraudulento.

El perfil de un cliente que sigue un tipo de consumo normal, es caracterizado por tener una curva de consumo variable mes a mes, debido a los diversos factores que influyen en el consumo de un persona, por ejemplo: efectos estacionales, abandono del hogar, fugas, aumento del número de integrantes entre otros factores. Existen casos donde se mantiene una tendencia en su consumo anual, por ejemplo, los clientes de la comuna de Cartagena mantienen un consumo elevado durante el periodo de verano (Diciembre, Enero y Febrero), mientras que el resto del año su consumo baja considerablemente. En la Figura 3.1, se muestra el consumo mensual entre los años 2010 al 2014 de un cliente de la comuna de Cartagena. Por otro lado, existen casos donde los clientes tienden a mantener un consumo anual más caótico, donde no es posible definir claramente algún tipo de tendencia, por ejemplo, en la comuna de Valparaíso, el consumo anual de los clientes no mantiene ninguna tendencia. En la Figura 3.2, se muestra el consumo mensual entre los años 2010 al 2014 de un cliente de la comuna de Valparaíso.

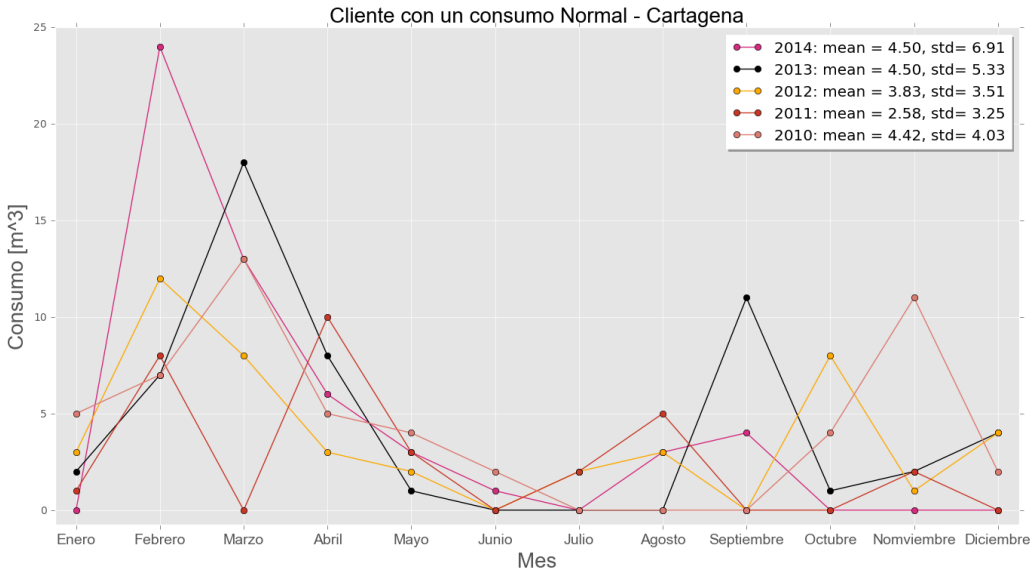


Figura 3.1: Perfil de un cliente Normal con tendencia en su consumo anual.

La característica esencial de que un cliente sigue un consumo normal es su curva de consumo promedio anual, ya que dicho consumo se mantiene relativamente estacionario año a año (ver Figura 3.3).

Por otro lado, el perfil de un cliente que sigue un tipo de consumo fraudulento, es caracterizado por la existencia de periodos donde el consumo se mantiene relativamente constante y significativamente bajo respecto al promedio anual (ver Figura 3.4). Cuando se analiza el consumo promedio anual, se observa que la curva ya no es estacionaria como el caso de un cliente que sigue un consumo normal (ver Figura 3.5).

3.2. Modelación

En esta sección se establece los pasos a seguir para dar solución al problema. En primer lugar, se toma en cuenta el conjunto de datos que no presenta problemas de datos perdidos. En segundo lugar, debido a la magnitud del conjunto de datos, se opta por trabajar el problema por comunas. Esto disminuye el gasto computacional y se logra realizar un estudio más especializado por sector.

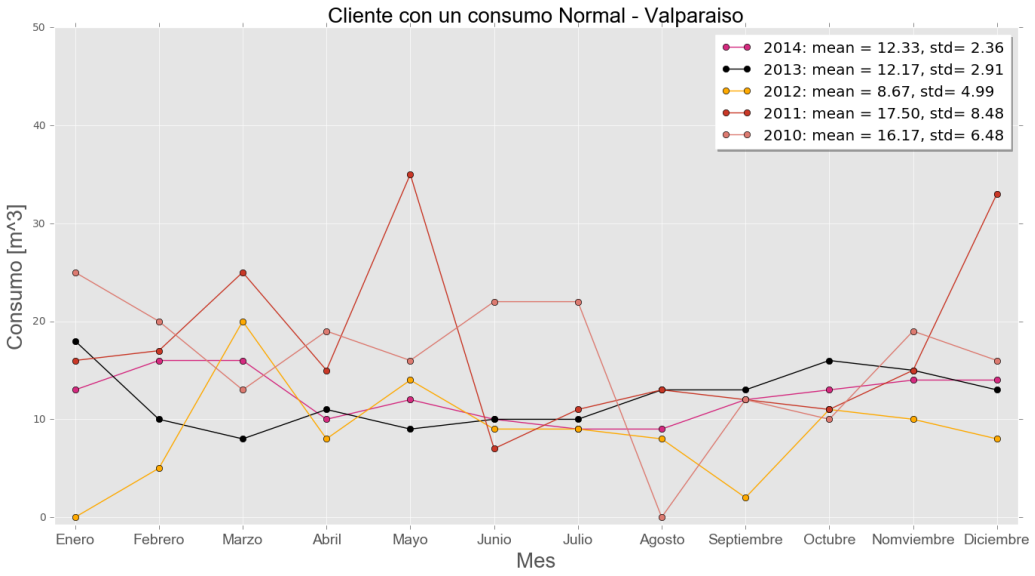


Figura 3.2: Perfil de un cliente Normal sin tendencia en su consumo anual.

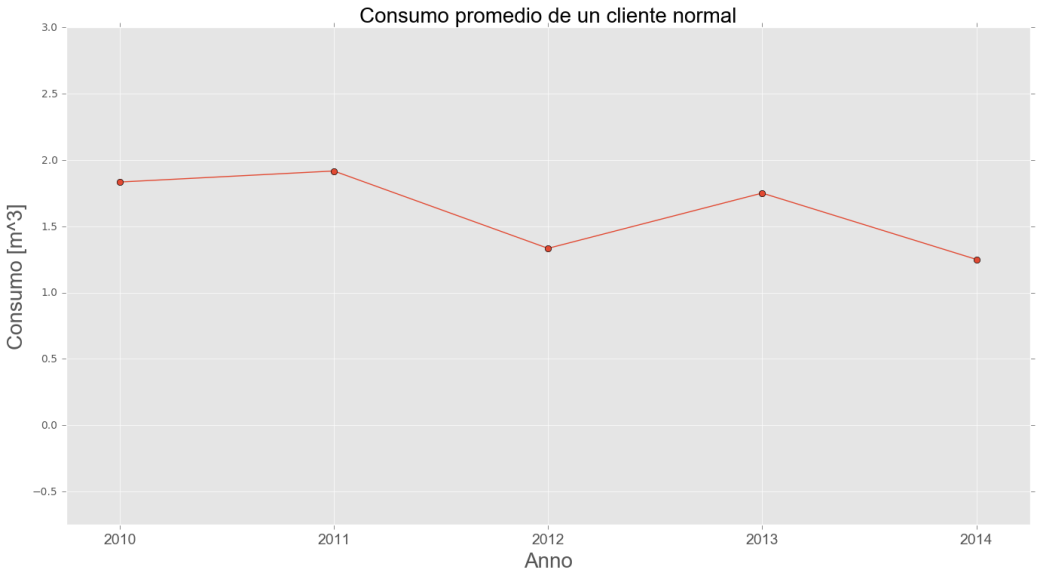


Figura 3.3: Perfil del consumo promedio de un cliente normal.

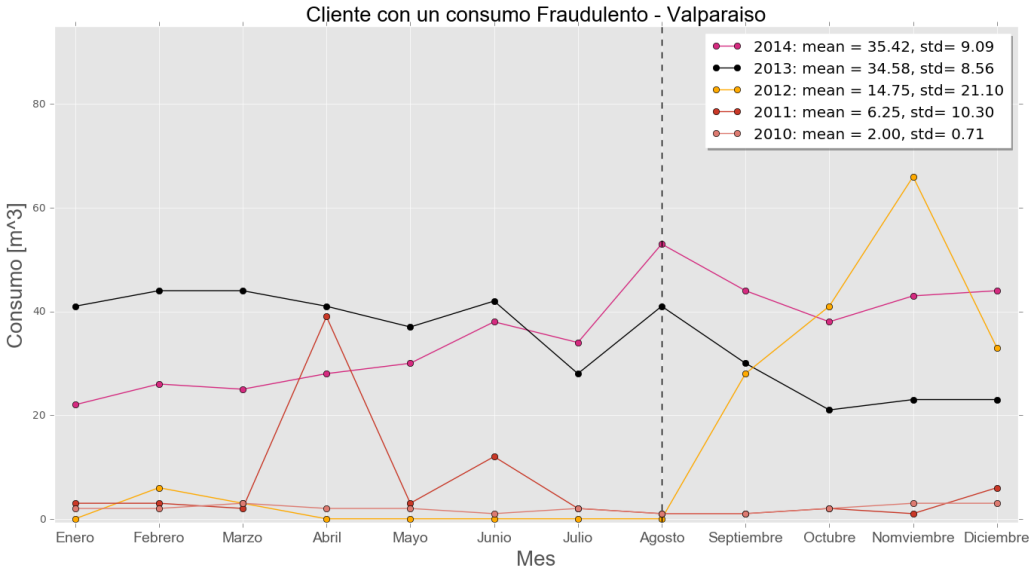


Figura 3.4: Perfil de un cliente fraudulento.

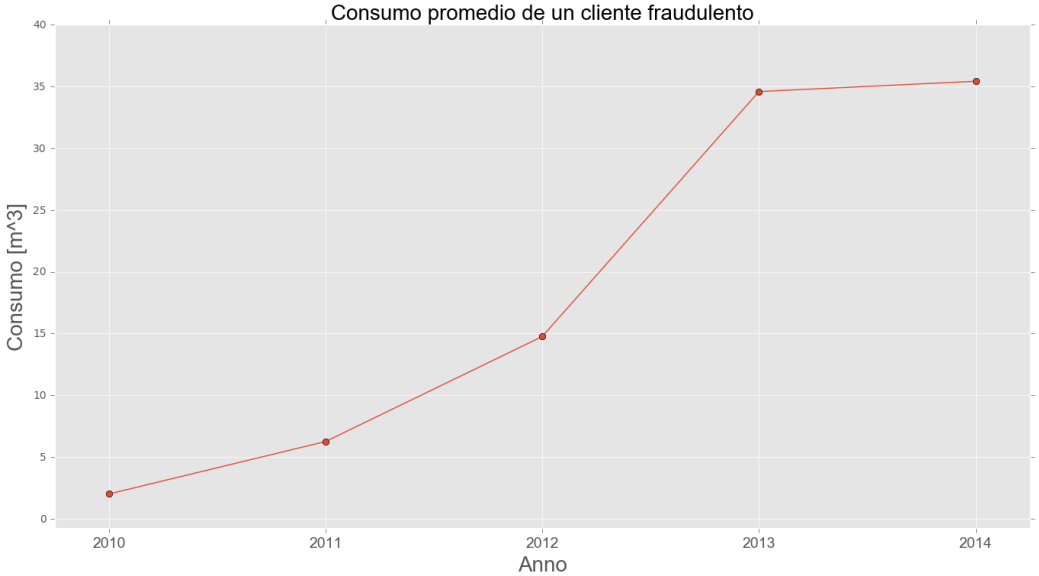


Figura 3.5: Perfil del consumo promedio de un cliente fraudulento.

3.2.1. Aplicación e interpretación de los modelos de regresión

El conjunto de entrenamiento se define por:

- $\boldsymbol{x}^{(i)} = (x_1^{(i)}, ..., x_{60}^{(i)})$: Consumo mensual del cliente i durante los años 2010 y 2014.
- $y^{(i)}$: 1, si el cliente i cometió algún tipo de fraude durante los años 2010 y 2014. 0, en otro caso.

Definido el conjunto de entrenamiento, se procede a aplicar los modelos descritos en la Sección 2.1: regresión logística (lr), bosque aleatorio (rf) y máquinas de soportes vectorial (svm). Para comparar los resultados de los distintos modelos, se ocupan las medidas de rendimiento descritos en la Sección 2.2: accuracy, precision, recall y f-score. Para obtener una banda de confiabilidad de las medidas de rendimientos se calculan intervalos de confianza asintóticos, para ello el conjunto de entrenamiento se divide aleatoriamente en dos conjuntos: "training dataset" y "testing dataset", cada uno tiene el 70 % y 30 % de la información del conjunto de datos, respectivamente. El procedimiento es el siguiente: se ajustan los modelos sobre el conjunto "training dataset", luego se procede a predecir los resultados ocupando el conjunto "testing dataset", con estas predicciones se calculan las medidas de rendimiento. El procedimiento anterior se repite 100 veces. Con esta información se calcula el promedio y la desviación estándar de las medidas de rendimientos y se da paso para encontrar los intervalos de confianza asintóticos del promedio mediante la ley de los grandes números. Se debe tener cuidado con la interpretación de las medidas de rendimiento, debido a que el conjunto de datos se encuentra desproporcionado (aproximadamente se tiene proporciones de 95 % de clientes normales y 5 % de clientes fraudulentos sobre el total de clientes). Estos resultados serán buenos a medida que el total verdaderos positivos sea alto , es decir, el total de verdaderos positivos sea lo más cercano al total de clientes fraudulentos, mientras que el número de falsos positivos sea lo más bajo posible (idealmente igual a cero).

Aplicados los diferentes modelos, se aborda el problema de los clientes fraudulentos que no fueron detectados como tal en el conjunto de datos. En este caso, un falso positivo corresponde a un cliente que sigue un consumo normal pero que el modelo detecto como cliente fraudulento y considerando el hecho que aproximadamente entre un 5 % a 10 % de la población del país corresponde a clientes fraudulentos, es probable que al menos uno de estos falsos positivo corresponda efectivamente a un cliente fruadulento (si es que

el número total de clientes fraudulentos es menor al 5 % de la población total de la comuna). Por tanto, estos falsos positivos serán clasificados como **clientes sospechosos**. Se definen tres tipos de clientes sospechosos:

- **Cliente sospechoso A**: Cliente detectado como un falso positivo por un modelo solamente.
- **Cliente sospechoso B**: Cliente detectado como un falso positivo por dos modelos simultáneamente.
- **Cliente sospechoso C**: Cliente detectado como un falso positivo por los tres modelos simultáneamente

Otro aspecto importante es que la suma de los clientes sospechosos más los clientes fraudulentos no debe superar el 10 % de la población.

El objetivo de este trabajo es encontrar aquellos clientes fraudulentos que no han sido detectados por las empresas de servicios de agua potable. En este caso, los candidatos a ser este tipo de clientes serán los clientes detectados como sospechosos. Para verificar si un cliente sospechoso corresponde efectivamente a un cliente fraudulento, las empresas deben mandar inspectores a las direcciones de los clientes detectados como sospechosos.

3.2.2. Análisis espacial de los clientes fraudulentos

Finalizada la parte de clasificación, se procede a realizar una análisis espacial de los datos. Aquí, se estudia la aleatoriedad de las coordenadas espaciales de lo clientes fraudulentos mediantes los test basados en cuadrantes (Test Xi-cuadrado) y en distancia (Función G,F,K-de Ripley), descritos en la sección 2.3.

Si los test indican que los clientes fraudulentos no siguen un patrón completamente aleatorio, se podría definir sectores específicos donde los inspectores puedan realizar la búsqueda de clientes fraudulentos, ahorrando tiempo y dinero a las empresas de servicios básicos de agua. La investigación realizada da los primeros pasos para el desarrollo de un software capaz de indicar las zonas geográficas que deben visitar en la búsqueda de los nuevos clientes fraudulentos, sin embargo, esto queda como trabajo a futuro.

3.3. Evaluación y pruebas

La comuna para desarrollar la metodología descrita es Cartagena. Esta comuna registra 1573 clientes con consumo continuo durante los años 2010 al

2014, con un total de 66 clientes fraudulentos (correspondiente a un 4.20 % de la población). El porcentaje de clientes fraudulentos se encuentra por debajo del intervalo límite establecido de clientes fraudulentos estimados (de alrededor de un 5 % a un 10 % por comuna).

3.3.1. Modelos de clasificación

En la Ecuación (3.1) se muestran los resultados para los distintos modelos vía la matriz de confusión. El número de verdaderos positivos de cada modelo (49 para el modelo de lr, 50 para el modelo de rf y 62 para el modelo de svm) esta cercano al número de clientes fraudulentos (66). Por otro lado, si se considera que a los más el número de clientes fraudulentos estaría rondando los 157 clientes, este número se encuentra por debajo del total de clientes fraudulentos detectado por cada modelo. Esto se debe al gran número de falsos positivos detectados en cada modelo.

$$CM_{lr} = \begin{pmatrix} 49 & 17 \\ 321 & 1186 \end{pmatrix} \ , \ CM_{rf} = \begin{pmatrix} 50 & 16 \\ 272 & 1235 \end{pmatrix} \ , \ CM_{svm} = \begin{pmatrix} 62 & 4 \\ 170 & 1337 \end{pmatrix} \tag{3.1}$$

En la Tabla 3.3 se muestran los intervalos de confianzas asintóticos de las medidas de rendimiento para los distintos modelos. Los resultados de medida de rendimiento "Accuracy" de los modelos son bastante similares entre si (alrededor de un 80 %), esto se debe a la cantidad de clientes normales que tiene el conjunto de entrenamiento. Por otro lado, en las otras tres medidas de rendimiento, el modelo SVM es levemente mejor que el modelo de LR y bastante mejor que el modelo RF.

Tabla 3.3: Intervalo de Confianza para el promedio de las medidas de rendimiento a nivel 0,95.

Medidas de Rendimiento	Regresión Logística	Bosque Aleatorio	Máquina de Vectores de Soporte
Accuracy	[77.94, 78.81]	[80.75, 82.79]	[80.75, 81.73]
Precision	[37.42, 41.78]	[18.48, 22.02]	[34.91, 39.50]
Recall	[7.68, 8.55]	[5.35, 6.55]	[8.42, 9.45]
F-score	[12.68, 14.06]	[8.07, 9.68]	[13.48, 15.04]

En lo que respecta al tipo de cliente, en la Tabla 3.4 se muestran los resultados obtenidos de los modelos ajustados. Cabe destacar que la suma de los clientes fraudulentos más los clientes sospechosos tipo C suman un total de 88 clientes (un 5.6 % de la población de Cartagena), es decir, si todos los clientes detectados como sospechoso tipo C fuesen fraudulentos, este número de clientes estaría dentro de los márgenes estipulados de clientes fraudulentos por comuna.

Tabla 3.4: Clasificación del tipo de cliente en la comuna de Cartagena.

Tipo de Cliente	Total
Normal	1065
Sospechoso A	284
Sospechoso B	136
Sospechoso C	22
Fraudulento	66

3.3.2. Análisis espacial

En la Figura 3.6 se muestra un mapa satelital de la comuna de Cartagena con la distribución de los clientes. Luego, en la Figura 3.7 se muestra un mapa satelital solo de los clientes fraudulentos. Ahora se procede a aplicar los test de aleatoriedad sobre las coordenadas espaciales de los clientes fraudulentos.

a) **Test Basado en los Cuadrantes**: Para este caso solo se considera el test Chi-cuadrado. Los resultados de la Tabla 3.5 indican que al aplicar el test se obtiene un p-valor < 0.05. Por lo tanto se concluye que las coordenadas espaciales de los clientes fraudulentos siguen un patrón puntual agrupado.

Tabla 3.5: Resultados del test de cuadratura en la comuna de Cartagena.

Test	χ^2	df	p-valor
Chi-Cuadrado	17.938	5	0.006052

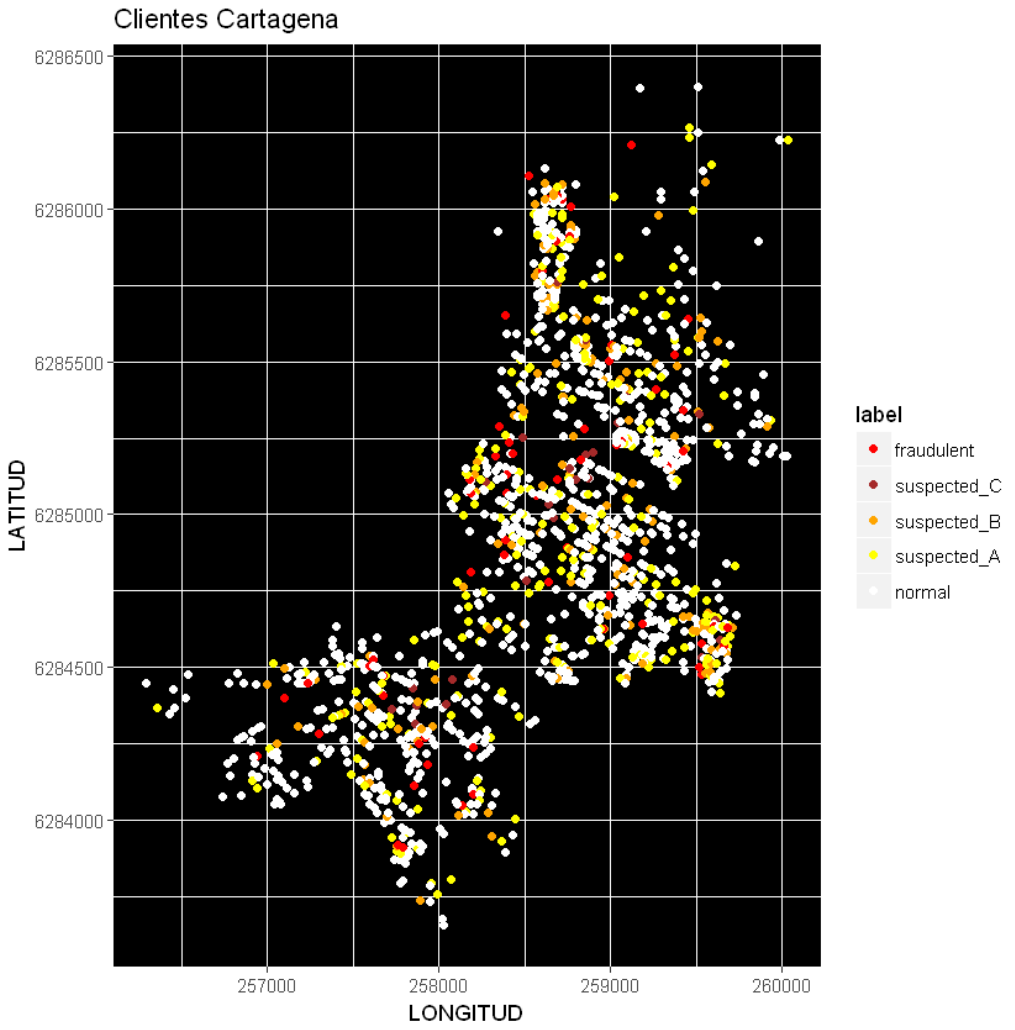


Figura 3.6: Mapa de los clientes de Cartagena clasificado según el tipo de cliente.

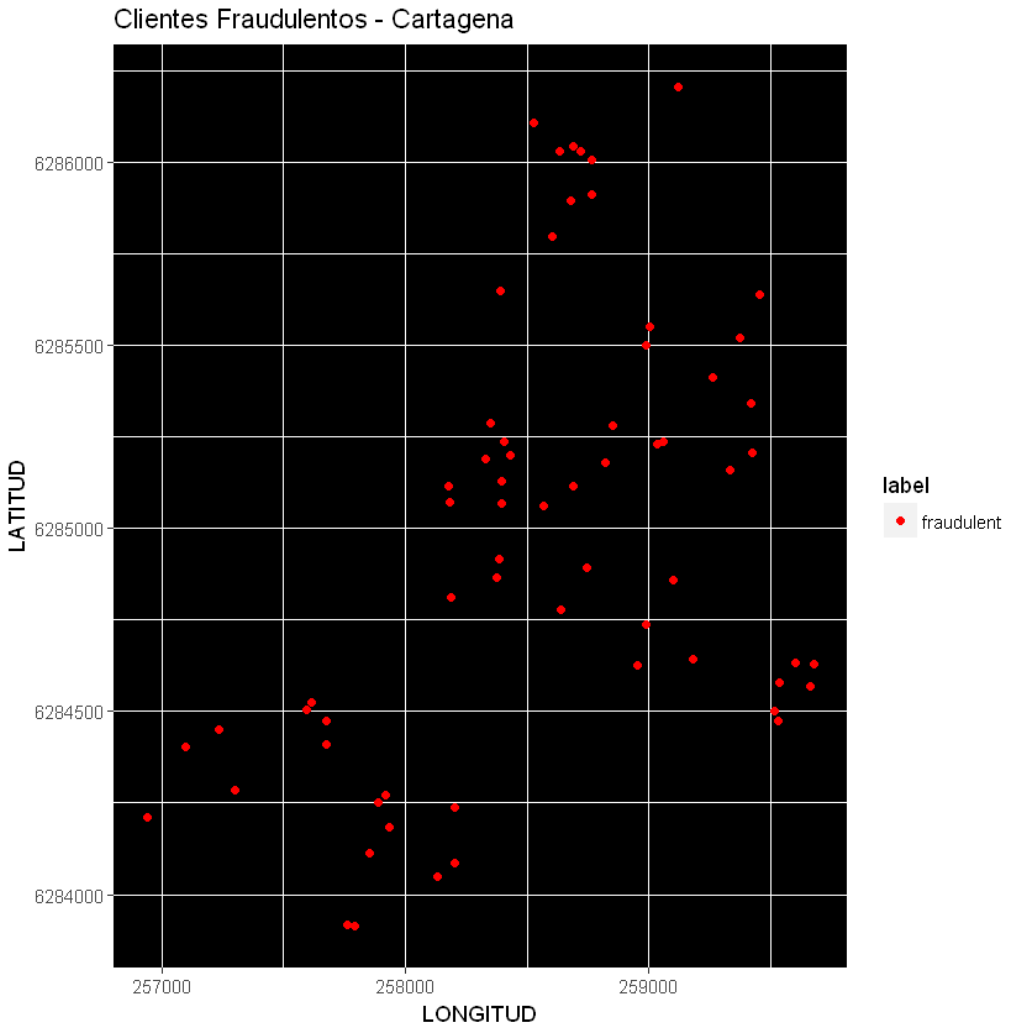
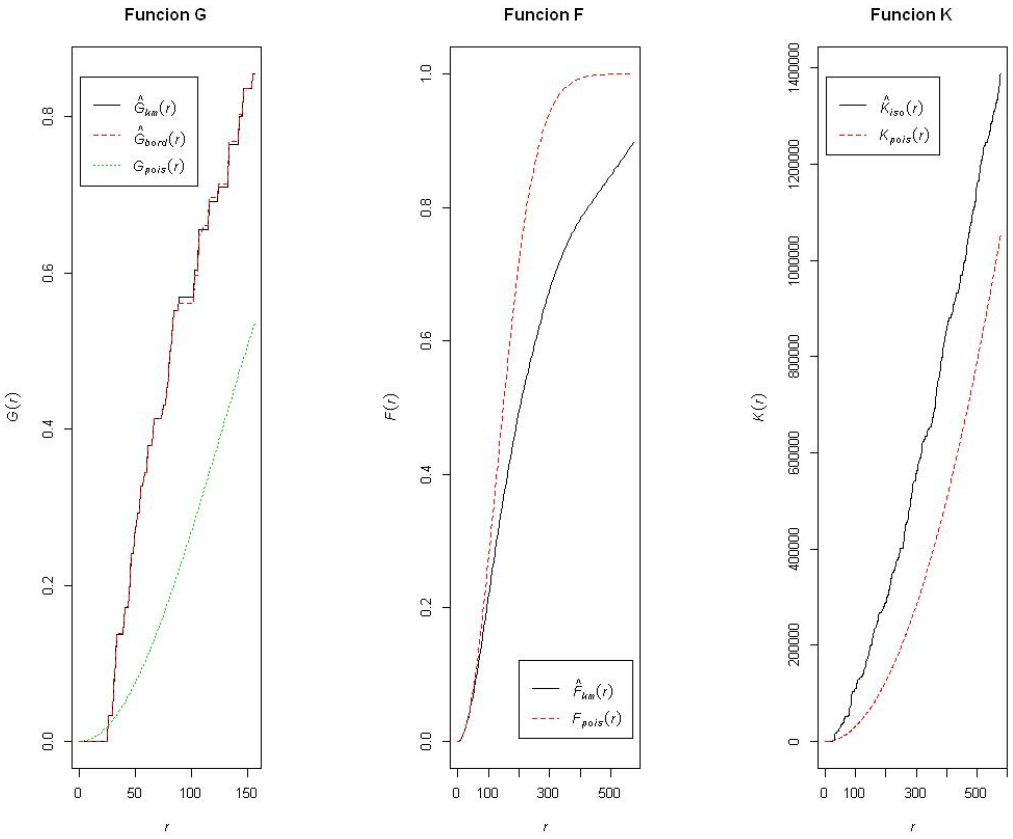


Figura 3.7: Mapa de los clientes de fraudulentos de Cartagena. Los test de aleatoriedad espacial indican que este patrón sigue el comportamiento de un patrón puntual agrupado.

b) **Test Basado en las distancias**: Para este caso se considera las funciones G, F y K de Ripley. En la Figura 3.8 se obtienen las gráficas de aplicar las distintas funciones de distancias. Según la forma que adoptan las gráficas, se concluye que las coordenadas espaciales de los clientes fraudulentos siguen un patrón puntual agrupado.



(a) Función G. **(b)** Función F. **(c)** Función K.

Figura 3.8: Resultados de los test basados en la distancia.

De los distintos test de aleatoriedad, se concluye que las coordenadas espaciales de los clientes fraudulentos siguen un patrón puntual agrupado. En la Figura 3.9 como estos clientes fueron agrupados mediante 4 grupos diferentes.

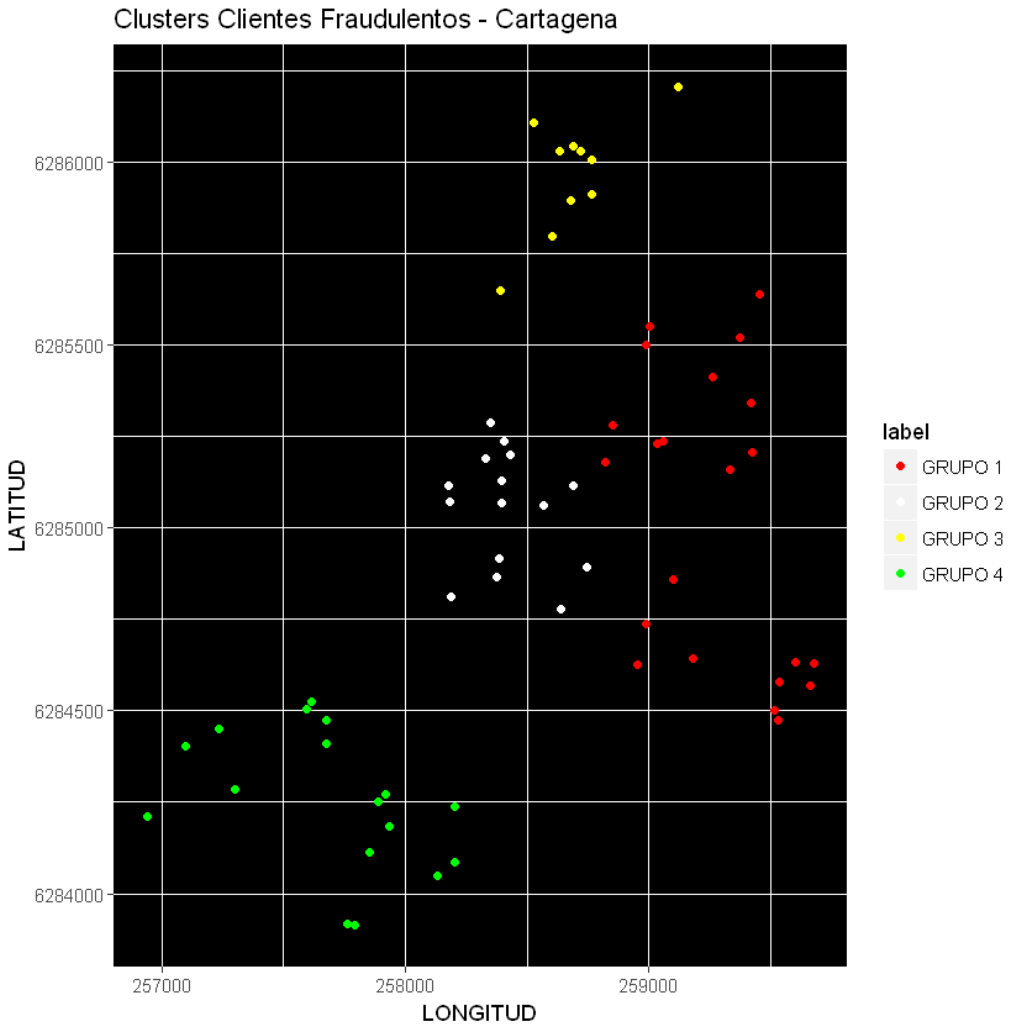


Figura 3.9: Mapa geográfico de los clientes fraudulentos de Cartagena separados por grupos.

Capítulo 4

Conclusiones

Los principales resultados obtenidos durante la investigación fueron:

a) En lo que respecta al análisis de datos, se establece las principales diferencias entre los consumos históricos de los clientes normales y los clientes fraudulentos. Además, se logra identificar patrones o tendencias en el consumo anual de ciertas comunas, por ejemplo, en la comuna de Cartagena los clientes durante el periodo de verano mantienen consumos más elevado en comparación al resto del año.

b) La metodología propuesta arroja resultados satisfactorios, los modelos propuestos se ajustan bastante bien al conjunto de entrenamiento, y se desarrolla un procedimiento para identificar los clientes fraudulentos que no han sido detectados, cumpliendo con el objetivo del problema. Sin embargo, estos resultados no fueron comprobados en terrenos por los inspectores, debido a la poca interacción que hubo con la empresa ESVAL.

c) Se realiza un primer avance respecto al análisis espacial del problema. El estudio enfocado a las coordenadas de los clientes fraudulento basado en las distintas pruebas de aleatoriedad indicaron que este proceso espacial sigue un patrón agrupado. La información que se puede rescatar del análisis espacial puede ser de utilidad para el desarrollo de un software capaz de identificar las zonas donde es más probable encontrar un cliente fraudulento. Además, se buscaría desarrollar una interfaz gráfica que muestre estos resultados, similar a la visualización del software Predpol (descrito en la Sección 2). Esto se deja como un eventual trabajo futuro.

Trabajos de investigación futuro corresponden a añadir nuevas variables de decisión al conjunto de entrenamiento, por ejemplo, la información socio-económica. Esta información serviría para contrastar la hipótesis si los clientes tienden a encontrarse en un estrato en particular o no. Adicionalmente, es de interés validar en terreno los resultados obtenidos durante la investigación, con el fin de contrastar la efectividad teórica respecto a la efectividad real.

Considerando que la metodología propuesta se enfoca principalmente en el consumo de sus clientes, esta se podría replicar sin problemas a otros servicios básicos, por ejemplo, servicios de luz, telefonía, cable, gas, entre otros.

Apéndice A

Diferenciación vectorial y matricial

En esta sección se introduce algunos resultados básicos de la derivación vectorial y matricial. Las definiciones y resultados presentados en esta sección, serán la base para poder desarrollar toda la teoría de las secciones siguientes.

Para efecto de esta sección, se considera que todas las derivadas existen y son continuas. (Para Más detalles, consultar en (Magnus & Neudecker, 1999))

Definición A.1. Sea $\boldsymbol{x} = (x_1, x_2, ..., x_k) \in \mathbb{R}^k$ un vector k-dimensional y sea $f : \mathbb{R}^k \to \mathbb{R}$ una función escalar de $\boldsymbol{x}$. La primera derivada parcial de f respecto a $\boldsymbol{x}$ queda definida por un vector k-dimensional de las derivadas parciales $\partial f/\partial x_i$:

$$\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}) = \begin{pmatrix} \dfrac{\partial f}{\partial x_1} \\ \vdots \\ \dfrac{\partial f}{\partial x_k} \end{pmatrix} \quad ; \quad \frac{\partial f}{\partial \boldsymbol{x}^\top}(\boldsymbol{x}) = (\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_k})$$

Definición A.2. La segunda derivada parcial de f respecto a $\boldsymbol{x}$ queda definida por una matriz de $k \times k$ de las derivadas parciales $\partial^2 f/\partial x_i \partial x_j$:

$$\frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{x}^\top}(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_k} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_k} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_k} & \frac{\partial^2 f}{\partial x_2 \partial x_k} & \cdots & \frac{\partial^2 f}{\partial x_k^2} \end{pmatrix}$$

Resultado A.1. Sean f y g dos funciones escalares del vector k-dimensional $\boldsymbol{x}$, y sea a, b dos constantes reales. Entonces:

a) $\partial(af + bg)/\partial x_j = a\partial f/\partial x_j + b\partial g/\partial x_j$

b) $\partial(fg)/\partial x_j = f\partial g/\partial x_j g + g\partial f/\partial x_j$

c) $\partial(f/g)/\partial x_j = (1/g^2)(g\partial f/\partial x_j + f\partial g/\partial x_j)$

Definición A.3. Sea $\boldsymbol{A}$ una matriz de $m \times n$ y sea $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ una función escalar de $\boldsymbol{A}$. La primera derivada parcial de f respecto a $\boldsymbol{A}$ se define como una matriz de $m \times n$ de las derivadas parciales de $\partial f/\partial a_{ij}$:

$$\begin{aligned}\frac{\partial f(\boldsymbol{A})}{\partial \boldsymbol{A}} &= \{\partial f/\partial a_{ij}\} \ , \ \forall i = 1, ..., m \ , \ \forall j = 1, ..., n \\ &= \begin{pmatrix} \partial f/\partial a_{11} & \cdots & \partial f/\partial a_{1n} \\ \vdots & \ddots & \vdots \\ \partial f/\partial a_{m1} & \cdots & \partial f/\partial a_{mn} \end{pmatrix}\end{aligned}$$

Resultado A.2. Sea $\boldsymbol{x}$ un vector n-dimensional y sea $\boldsymbol{A}$ una matriz de $m \times n$. Entonces:

$$\frac{\partial \boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}^\top} = A \quad ; \quad \frac{\partial \boldsymbol{x}^\top \boldsymbol{A}^\top}{\partial \boldsymbol{x}} = A^\top$$

Resultado A.3. Sea $\boldsymbol{x}$ un vector n- dimensional y sea $\boldsymbol{A}$ una matriz de $m \times n$. Entonces:

a) $$\frac{\partial \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}} = \begin{cases} (\boldsymbol{A} + \boldsymbol{A}^\top)\boldsymbol{x} & , \text{ si } \boldsymbol{A} \text{ no es simétrica} \\ 2\boldsymbol{A}\boldsymbol{x} & , \text{ si } \boldsymbol{A} \text{ es simétrica} \end{cases}$$

b) $$\frac{\partial \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}^\top} = \begin{cases} \boldsymbol{x}^\top(\boldsymbol{A} + \boldsymbol{A}^\top) & , \text{ si } \boldsymbol{A} \text{ no es simétrica} \\ 2\boldsymbol{x}^\top \boldsymbol{A} & , \text{ si } \boldsymbol{A} \text{ es simétrica} \end{cases}$$

c) $$\frac{\partial \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}\partial \boldsymbol{x}^\top} = \begin{cases} \boldsymbol{A} + \boldsymbol{A}^\top & , \text{ si } \boldsymbol{A} \text{ no es simétrica} \\ 2\boldsymbol{A} & , \text{ si } \boldsymbol{A} \text{ es simétrica} \end{cases}$$

Resultado A.4. Sea $\boldsymbol{A}$ una matriz de $m \times n$, sea $\boldsymbol{x}$ un vector m-dimensional y sea $\boldsymbol{y}$ un vector n-dimensional. Entonces:

$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y}}{\partial \boldsymbol{A}} = \boldsymbol{x}\boldsymbol{y}^\top$$

Resultado A.5. Sea $\boldsymbol{A}$ una matriz de $n \times n$. Entonces:

a) $$\frac{\partial \mathrm{tr}\boldsymbol{A}}{\partial \boldsymbol{A}} = \boldsymbol{I}_n$$

b) $\dfrac{\partial|\boldsymbol{A}|}{\partial \boldsymbol{A}} = [\text{adj}(\boldsymbol{A})]^\top$

Resultado A.6. Sea $\boldsymbol{A}$ una matriz de $n \times n$ con $|\boldsymbol{A}| > 0$. Entonces:

$$\frac{\partial \ln|\boldsymbol{A}|}{\partial \boldsymbol{A}} = (\boldsymbol{A}^\top)^{-1}$$

Resultado A.7. Sea $\boldsymbol{A}$ una matriz de $m \times n$ y Sea $\boldsymbol{B}$ una matriz de $n \times m$. Entonces:

$$\frac{\partial \text{tr}|\boldsymbol{AB}|}{\partial \boldsymbol{A}} = \boldsymbol{B}^\top$$

Resultado A.8. Sea $\boldsymbol{\Omega}$ una matriz simétrica de $n \times n$, sea $\boldsymbol{y}$ un vector n-dimensional, sea $\boldsymbol{\beta}$ un vector k-dimensional y sea $\boldsymbol{X}$ una matriz de $n \times k$. Entonces:

a) $\dfrac{\partial(\boldsymbol{y} - \boldsymbol{X\beta})^\top \boldsymbol{\Omega}(\boldsymbol{y} - \boldsymbol{X\beta})}{\partial \boldsymbol{\beta}} = -2\boldsymbol{X}^\top \boldsymbol{\Omega}(\boldsymbol{y} - \boldsymbol{X\beta})$

b) $\dfrac{\partial(\boldsymbol{y} - \boldsymbol{X\beta})^\top \boldsymbol{\Omega}(\boldsymbol{y} - \boldsymbol{X\beta})}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}^\top} = 2\boldsymbol{X}^\top \boldsymbol{\Omega X}$

Definición A.4. Sea $A : \mathbb{R}^{m \times n} \to \mathbb{R}$ una función escalar de θ. Entonces:

$$\begin{aligned}\frac{\partial \boldsymbol{A}}{\partial \theta} &= \{\partial a_{ij}/\partial\theta\} \ , \ \forall i = 1, ..., m \ , \ \forall j = 1, ..., n \\ &= \begin{pmatrix} \partial a_{11}/\partial\theta & \cdots & \partial a_{1n}/\partial\theta \\ \vdots & \ddots & \vdots \\ \partial a_{m1}/\partial\theta & \cdots & \partial a_{mn}/\partial\theta \end{pmatrix}\end{aligned}$$

Apéndice B

Kernels

Cuando se trabaja con modelos predictivos, se desea encontrar la mejor aproximación respecto a la variable observada, es decir, que la predicción y la variable observada sean los más similar posible, sin embargo, se debe establecer un criterio para medir esta similaridad. Esto criterio deben ser acorde al espacio donde se trabaja, por ejemplo, si se establece una medida para medir cuán similares son dos número o dos letras, seguramente este criterio no sirve para medir si dos funciones son iguales. Esto motiva a definir la funciones de kernel.

Se define una función kernel como una función real de dos argumentos, $\kappa(x, x) \in \mathbb{R}$. Esta función cumple con las propiedades de simetría ($\kappa(x, y) = \kappa(y, x)$), y no-negatividad ($\kappa(x, y) \geq 0$).

En base a su definición, existen una gran variedad de kernels en la literatura (Cristianini & Shawe-Taylor, 2000). En esta sección se estudian las funciones de kernels más relevantes para este trabajo.

a) **Kernel Lineal**: Es el más simple de todos los kernel, el cual se define por

$$\kappa(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{x}$$

Este tipo de Kernel es bastante útil si los datos son de altas dimensiones y si las características son individualmente informativas. Por supuesto, no todos los problemas de alta dimensiones son separables.

b) **Kernel Gaussiano o RBF**: El Kernel Gaussiano se define por

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{y})^\top \Sigma^{-1}(\boldsymbol{x} - \boldsymbol{y}))$$

Si Σ es diagonal, entonces puede ser escrito como

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = exp(-\frac{1}{2}\sum_{i=1}^{n}\frac{1}{\sigma_i^2}(x_i - y_i)^2)$$

Se puede interpretar los σ_i como la escala de longitud característica de la dimensión. Si $\sigma_i = \infty$, entonces la correspondiente dimensión es ignorada.

Si Σ es esférico, se obtiene el Kernel isotrópico

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = exp(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma})$$

c) **Kernel de Mercer**: Para definir este tipo de kernel, es necesario definir la matriz de Gram, la cual se define como

$$\boldsymbol{K} = \begin{pmatrix} \kappa(x_1, y_1) & \cdots & \kappa(x_1, y_n) \\ \vdots & \ddots & \vdots \\ \kappa(x_n, y_1) & \cdots & \kappa(x_n, y_n) \end{pmatrix}$$

Cuando la matriz de Gram es definida positiva para cualquier conjunto $\{x_i\}_{i=1}^n$, esta se conoce con el nombre de kernel de Mercer. Se puede mostrar que el kernel Gaussiano en un kernel de Mercer (Schoelkopf & Smola, 2002) como lo es el kernel de similaridad de coseno (Sahami& Heilman, 2006).

La importancia de los kernel de Mercer es el siguiente resultado, también conocido como el **Teorema de Mercer**. Si la matriz de Gram $\boldsymbol{K}$, es definida positiva, entonces la matriz $\boldsymbol{K}$ se puede estimar mediante su descomposición de vectores propios, es decir, existe una matriz ortogonal $\boldsymbol{U}$ y un matriz diagonal $\boldsymbol{\Lambda}$, con valores propios $\lambda_i > 0$, tal que la matriz $\boldsymbol{K}$ se puede escribir como

$$\boldsymbol{K} = \boldsymbol{U}^\top \boldsymbol{\Lambda} \boldsymbol{U}$$

Ahora, expresando los términos de $\boldsymbol{K}$ en función de su descomposición de vectores propios, se tiene que

$$k_{ij} = (\boldsymbol{\Lambda}^{1/2}\boldsymbol{U}_{:,i})^\top(\boldsymbol{\Lambda}^{1/2}\boldsymbol{U}_{j,:})$$

Se define $\phi(x_i) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{U}_{:,i}$. Entonces

$$k_{ij} = \phi(x_i)^\top \phi(y_j)$$

Lo anterior muestra que las entradas de un kernel de Mercer, pueden ser escritas como el producto interno de sus vectores de entrada, los cuales se encuentran definidos de forma implícita por lo valores propios de la matriz $\boldsymbol{U}$.

En general, si el kernel a trabajar es un Kernel de Mercer, entonces existe una función $\phi : \mathcal{X} \mapsto \mathbb{R}^n$ tal que:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{y})$$

donde ϕ depende de las funciones propias de κ (así $\mathbb{R}^n$ puede ser de dimensión infinita).

Debido a la ventaja de lo expuesto anteriormente, es que se busca probar que los kernel que se trabajan cumplan con la condición de ser un kernel de Mercer, sin embargo, muchas veces esto no es una tarea fácil y se requieren técnicas del análisis funcional. Por otro lado, es posible construir nuevos kernel de Mercer basado en un conjuntos de reglas estándares. Por ejemplo, si κ_1 y κ_2 son ambos Kernel de Mercer, entonces: $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \kappa_1(\boldsymbol{x}, \boldsymbol{y}) + \kappa_2(\boldsymbol{x}, \boldsymbol{y})$ también lo es (Schoelkopf & Smola, 2002).

Otros Kernels que pueden ser de interés para el lector encontrados a menudo en la literatura: Kernel de Mattern, Kernel de Fisher, "String Kernel", "Kernel Tricks", entre otros (Murphy, 2012).

Apéndice C

Tablas

Tabla C.1: Evolución de la cobertura de agua potable en Chile, Datos SISS

Año	Población Millones de habitantes	Cobertura Agua Potable Urbana (%)	Cobertura Alcance Urbano (%)
1965	5.85	53.5	25.4
1966	6.01	56.3	26.0
1967	6.18	59.1	26.8
1968	6.34	61.7	27.8
1969	6.51	64.1	29.5
1970	6.67	66.5	31.1
1971	6.86	67.2	33.0
1972	7.05	67.9	34.8
1973	7.24	68.6	36.5
1974	7.42	69.2	38.2
1975	7.62	77.4	43.5
1976	7.75	78.2	51.5
1977	8.21	85.6	55.9
1978	8.55	86.0	56.3
1979	8.70	90.1	62.4
1980	8.89	91.4	67.4
1981	9.07	91.5	68.2
1982	9.26	92.1	70.0
1983	9.55	92.7	70.6
1984	9.78	94.3	72.9

1985	9.66	95.2	75.1
1986	9.99	97.0	77.2
1987	10.32	97.2	78.8
1988	10.54	98.0	80.8
1989	10.76	98.2	81.5
1990	11.40	97.4	81.8
1991	11.67	97.6	83.6
1992	10.93	97.5	84.7
1993	11.31	98.0	86.4
1994	11.82	98.5	87.9
1995	11.96	98.6	89.4
1996	12.21	98.9	90.4
1997	12.46	99.3	91.0
1998	12.62	99.3	91.6
1999	12.72	99.2	92.1
2000	13.3	99.6	93.1
2001	13.6	99.7	93.6
2002	13.9	99.7	94.1
2003	14.1	99.8	94.7
2004	12.9	99.7	95.0
2005	13.3	99.8	94.9
2006	13,6	99,8	95,2
2007	14	99,9	95,2
2008	14,4	99,8	95,3
2009	14,8	99,8	95,6
2010	15,1	99,8	95,9
2011	15,4	99,8	96,1
2012	15,7	99,9	96,3
2013	16,1	99,9	96,5
2014	16,5	99,9	96,7

Tabla C.2: Componentes y Definiciones del Balance de Agua, IWA/AWWA

Suministro de Agua Potable	Consumos Autorizados	Consumos Autorizados Facturados	Consumos Medidos Facturados a Clientes Registrados
			Consumos No Medidos Facturados a Clientes Registrados
		Consumos Autorizados No Facturados	Medidos
			No Medidos
	Pérdidas de Agua	Pérdidas Aparentes	Consumos No Autorizados
			Consumos Con Medición Defectuosa
		Pérdidas Físicas	Fugas en Redes
			Fugas y Rebalses en Tanques de Almacenamiento

Tabla C.3: Información filtrada por Comuna

localidad	clientes	fraudes	fraudes especiales	% fraudes	% fraudes especiales
Algarrobo	2074	96	33	4.63	1.59
Algarrobo Norte	41	3	1	7.32	2.44
Artificio	1700	159	24	9.35	1.41
Brisas de Mirasol	72	4	0	5.56	0.00
Cabildo	2241	109	27	4.86	1.20
Cachagua	322	16	3	4.97	0.93
Calle Larga	1654	137	31	8.28	1.87
Cartagena	1573	264	66	16.78	4.20
Casablanca	3281	202	39	6.16	1.19
Catemu	1077	105	22	9.75	2.04
Chepical	109	13	5	11.93	4.59
Chincolco	176	14	0	7.95	0.00
Concon	6875	981	164	14.27	2.39
Curauma	2218	70	9	3.16	0.41
El Almendral	548	69	10	12.59	1.82
El Quisco	1084	91	24	8.39	2.21
El Tabo	432	34	7	7.87	1.62
Hijuelas	1060	102	12	9.62	1.13
Isla Negra	189	20	6	10.58	3.17
Las Cruces	428	31	12	7.24	2.80
La Calera	6310	537	57	8.51	0.90
La Cruz	2030	94	22	4.63	1.08
La Laguna	44	3	0	6.82	0.00
La Ligua	3006	131	24	4.36	0.80
Limache	5689	466	45	8.19	0.79
Llay Llay	2821	267	57	9.46	2.02
Los Andes	12827	1615	190	12.59	1.48
Mirasol	57	0	0	0.00	0.00
Nogales	1361	79	12	5.80	0.88
Papudo	674	35	9	5.19	1.34
Petorca	343	43	4	12.54	1.17
Placilla	2582	227	46	8.79	1.78
Placilla Ligua	756	29	4	3.84	0.53

Puchuncavi	593	41	6	6.91	1.01
Punta de Tralca	98	3	2	3.06	2.04
Punta Puyai	68	0	0	0.00	0.00
Putaendo	1632	149	25	9.13	1.53
Quillota	12733	784	79	6.16	0.62
Quilpue	27406	3213	443	11.72	1.62
Quintero	2610	274	30	10.50	1.15
Real Curimon	951	88	19	9.25	2.00
Renaca	4496	164	27	3.65	0.60
Rinconada	1211	146	38	12.06	3.14
Santa Maria	1070	124	24	11.59	2.24
San Antonio	17995	1557	440	8.65	2.45
San Esteban	2903	303	55	10.44	1.89
San Felipe	10829	1355	229	12.51	2.11
San Isidro	66	4	0	6.06	0.00
San Pedro	601	49	5	8.15	0.83
San Sebastian	300	38	13	12.67	4.33
Valparaiso	38113	4060	449	10.65	1.18
Villa Alemana	18795	2490	286	13.25	1.52
Viña del Mar	51256	4861	775	9.48	1.51
Zapallar	416	44	5	10.58	1.20

Apéndice D

Rutinas

I) Rutinas desarrolladas en Python

```
1
2  ## Librerias
3
4  import pandas as pd
5  import numpy as np
6  import matplotlib.pyplot as plt
7  import os
8  import time
9  import sklearn
10
11 from pandas import ExcelWriter
12 from numpy.linalg import inv
13 from sklearn import svm
14 from sklearn import preprocessing
15 from sklearn.metrics import confusion_matrix
16 from sklearn.cross_validation import train_test_split
17 from sklearn.linear_model import LogisticRegression
18 from sklearn.ensemble import RandomForestClassifier
19 from sklearn.svm import SVC
20
21 %matplotlib inline
```

Rutina D.1: Librerías necesarias para trabajar.

```
1
2  # Cargar archivos
3
4  file_names = []
5  for (dirpath, dirnames, filenames) in os.walk(my_data):
6      for file in filenames:
7          if file.endswith('.csv') or file.endswith('.xlsx'):
8              file_names.append(file)
9
10 file_names = sorted(list(set(file_names)))
11
12 # total clientes
13
```

```
14 all_clients = []
15
16 for name in file_names:
17
18     aux_df = pd.read_csv(my_data + "/" + name, sep = ";")
19     all_clients = all_clients + aux_df["id_client"].tolist()
20
21 all_clients = sorted(list(set(all_clients)))
22
23 # matriz de consumo
24
25 cons_df = pd.DataFrame(columns = ["id_client"])
26 cons_df["id_client"] = all_clients
27
28 # rellenar matriz de consumo
29
30 name_columns = ["id_client","month_01","month_02","month_03",
31                 "month_04","month_05","month_06","month_07","month_08",
32                 "month_09","month_10","month_11","month_12"]
33
34 for name in file_names:
35
36     year = name.split(".")[0]
37     aux_df = pd.read_csv(my_data +"/" + name, sep = ";")
38
39     aux_list = []
40
41     for i in range(len(name_columns)-1):
42         aux_list = aux_list + [name_columns[i+1] + " - " + year]
43
44     aux_list = ["id_client"]  + aux_list
45     aux_df.columns = aux_list
46
47     cons_df = cons_df.merge(aux_df,left_on="id_client", right_on="id_client"
           , how="left")
48
49 # agregar localidad
50
51 locality_df = pd.read_csv(otherDir +"/" +"Catastro_ESVAL.txt",sep="|")
52 locality_df = locality_df[["NRO_SUMINISTRO","DESCRIPCION_3"]]
53 locality_df.columns = ["id_client","locality"]
54 locality_df = locality_df.drop_duplicates('id_client', take_last=True)
55 locality_df.head()
56
57 # juntar informacion
58
59 cons_df = cons_df.merge(locality_df,left_on="id_client", right_on="id_client
       ", how="left")
60 cons_df = cons_df[["id_client","locality"] + cons_df.columns.tolist()[1:-1]]
61
62 # agregar cliente fraudulentos
63
64 fraudulent_df = pd.read_excel(otherDir +"/" + "/fraude.xlsx")
65
66 # crear df adecuado
67
68 list_01 = fraudulent_df["fec_creacion"].tolist()
69 date_list = []
70
71
72 for i in range(len(list_01)):
73     aux_str = str(list_01[i].year) + "/" + str(list_01[i].month) + "/" + str
```

```
        (list_01[i].day)
74      date_list.append(aux_str)
75
76  fraudulent_df["fec_creacion"] = date_list
77
78  list_01 = sorted(list(set(fraudulent_df["nro_suministro"])))
79
80  aux_list_01 = []
81  aux_list_02 = []
82  aux_list_03 = []
83
84
85  for i in range(len(list_01)):
86      aux_df = fraudulent_df[fraudulent_df["nro_suministro"] == list_01[i]]
87      aux_list_01 = aux_list_01 + [list_01[i]]
88      aux_list_02 = aux_list_02 + [" - ".join(aux_df["fec_creacion"].tolist())
            ]
89      aux_list_03 = aux_list_03 + [" - ".join(aux_df["cod_anomalia"].tolist())
            ]
90
91  new_fraud_df = pd.DataFrame(columns = ["id_client"])
92  new_fraud_df["id_client"]   = aux_list_01
93  new_fraud_df["fraud_date"]  = aux_list_02
94  new_fraud_df["anomaly_cod"] = aux_list_03
95
96  # juntar informacion
97
98  cons_df = cons_df.merge(new_fraud_df,left_on="id_client", right_on="id_
        client", how="left")
99  cons_df = cons_df[["id_client","locality","fraud_date","anomaly_cod"] + cons
        _df.columns.tolist()[2:-2]]
100
101 # agregar coordenadas
102
103 coord_df = pd.read_excel(otherDir + "/CLIENTES SIGEC TOTAL.xlsx")
104 coord_df = coord_df[["NUMERO","LONGITUD","LATITUD"]]
105 coord_df.columns = ["id_client","x_coord","y_coord"]
106
107 # juntar informacion
108
109 cons_df = cons_df.merge(coord_df,left_on="id_client", right_on="id_client",
        how="left")
110 cons_df = cons_df[["id_client","locality","x_coord","y_coord","fraud_date","
        anomaly_cod"] + cons_df.columns.tolist()[4:-2]]
```

Rutina D.2: Creación de la base de datos con la información de los clientes de la V región.

```
1  cons_df["locality"]    = cons_df["locality"].fillna("")
2  cons_df["fraud_date"]  = cons_df["fraud_date"].fillna("")
3  cons_df["anomaly_cod"] = cons_df["anomaly_cod"].fillna("")
4  cons_df["x_coord"]     = cons_df["x_coord"].fillna("")
5  cons_df["y_coord"]     = cons_df["y_coord"].fillna("")
6
7  all_clients    = cons_df["id_client"].tolist()
8  all_frauds     = sorted(list(set(cons_df["fraud_date"])))[1:]
9  all_localities = sorted(list(set(cons_df["locality"])))[1:]
10
11 print("\n")
12 print("Total de clientes             =  %d" %(len(all_clients)))
13 print("Total de clientes fraudulentos =   %d" %(len(all_frauds)))
```

```
print("Total de localidades = %d" %(len(all_localities)))
print("\n")
print("Total de clientes con consumo todo los annos = %d" %(len(cons_df.dropna() )))
print("Total de clientes que registran localidad = %d" %(len(cons_df[cons_df["locality"] != "" ])))
print("Total de clientes que cumplen ambas condiciones = %d" %(len(cons_df[cons_df["locality"] != "" ].dropna())))
```

Rutina D.3: Resumen de la información general de la base de datos.

```
# comunas

all_localities = sorted(list(set(cons_df["locality"])))[1:]

# sin contar consumo continuo

list_01 = []
list_02 = []
list_03 = []
list_04 = []
list_05 = []
list_06 = []

cont_1 = 0
cont_2 = 0

for locality in all_localities:

    aux_df     = cons_df[cons_df["locality"] == locality]
    aux_list_1 = aux_df["fraud_date"].tolist()
    aux_list_2 = aux_df["ind_fraud"].tolist()

    for i in range(len(aux_list_1)):
        if aux_list_1[i] != "":
            cont_1+=1
    for i in range(len(aux_list_2)):
        if aux_list_2[i] == 1:
            cont_2+=1

    aux_nm_1 = len(aux_df)
    aux_nm_2 = cont_1
    aux_nm_3 = cont_2
    aux_nm_4 = round(100.0*aux_nm_2/aux_nm_1,2)
    aux_nm_5 = round(100.0*aux_nm_3/aux_nm_1,2)

    list_01.append(locality)
    list_02.append(aux_nm_1)
    list_03.append(aux_nm_2)
    list_04.append(aux_nm_3)
    list_05.append(aux_nm_4)
    list_06.append(aux_nm_5)

    cont_1 = 0
    cont_2 = 0

locality_df = pd.DataFrame(columns = ["locality"])
locality_df["locality"]          = list_01
locality_df["nm_clients"]        = list_02
locality_df["nm_all_frauds"]     = list_03
locality_df["nm_special_frauds"] = list_04
```

```
locality_df["%all_frauds"]      = list_05
locality_df["%special_frauds"]  = list_06

locality_df = locality_df.sort(['%special_frauds'], ascending=[False])

# contando  consumo continuo

new_cons_df = cons_df.dropna()


list_01 = []
list_02 = []
list_03 = []
list_04 = []
list_05 = []
list_06 = []

cont_1 = 0
cont_2 = 0

for locality in all_localities:

    aux_df     = new_cons_df[new_cons_df["locality"] == locality]
    aux_list_1 = aux_df["fraud_date"].tolist()
    aux_list_2 = aux_df["ind_fraud"].tolist()

    for i in range(len(aux_list_1)):
        if aux_list_1[i] != "":
            cont_1+=1
    for i in range(len(aux_list_2)):
        if aux_list_2[i] == 1:
            cont_2+=1

    aux_nm_1 = len(aux_df)
    aux_nm_2 = cont_1
    aux_nm_3 = cont_2
    aux_nm_4 = round(100.0*aux_nm_2/aux_nm_1,2)
    aux_nm_5 = round(100.0*aux_nm_3/aux_nm_1,2)

    list_01.append(locality)
    list_02.append(aux_nm_1)
    list_03.append(aux_nm_2)
    list_04.append(aux_nm_3)
    list_05.append(aux_nm_4)
    list_06.append(aux_nm_5)

    cont_1 = 0
    cont_2 = 0

new_locality_df = pd.DataFrame(columns = ["locality"])
new_locality_df["locality"]           = list_01
new_locality_df["nm_clients"]         = list_02
new_locality_df["nm_all_frauds"]      = list_03
new_locality_df["nm_special_frauds"] = list_04
new_locality_df["%all_frauds"]        = list_05
new_locality_df["%special_frauds"]   = list_06
```

Rutina D.4: Resumen de los clientes según la comuna en estudio.

```
# filtro  por comuna

```

```
commune      = all_localities[all_localities.index("CARTAGENA")]

data_matrix = cons_df[cons_df["locality"] == commune]

# preparacion de datos para el algoritmo

## consumo continuo todo los annos

data_matrix = data_matrix.dropna()

## cambiar formato de los datos

X =  pd.DataFrame(data_matrix[data_matrix.columns[7:].tolist()])
y =  np.array(data_matrix["ind_fraud"], dtype=pd.Series).astype(int)

## normalizacion

X_norm = preprocessing.scale(X)
```

Rutina D.5: Preparación del conjunto de entrenamiento para aplicar los modelos (por comuna).

```

## regresion logistica

clf = LogisticRegression(penalty="l1",dual=False,C=1.0,fit_intercept=True,
    class_weight="balanced",random_state=int(time.clock()/10),verbose=10)
clf.fit(X_norm, y)


y_lr=clf.predict(X_norm)     # prediccion mediante regresion logistica
cm=confusion_matrix(y,y_lr,labels=[1,0 ])

Accuracy = float(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])

if (cm[0,0]+cm[0,1]):
    Precision = float(cm[0,0])/(cm[0,0]+cm[0,1])
else:
    Precision = 0

Recall = float(cm[0,0])/(cm[0,0]+cm[1,0])

if (cm[0,0]+cm[0,1]):
    F = (2*Precision*Recall)/(Precision + Recall)
else:
    F = 0

print("\n")
print("*** Matriz de confusion ***")
print(cm)
print("\n")
print("Accuracy:   %.2f %s" %(100*Accuracy,"%") )
print("Precision:  %.2f %s" %(100*Precision,"%") )
print("Recall:     %.2f %s" %(100*Recall,"%") )
print("F:          %.2f %s" %(100*F,"%") )

## bosque aleatorio

clf = RandomForestClassifier(n_estimators = 10, min_samples_split =350, min_
    samples_leaf = 15, class_weight= 'balanced' ,random_state= int(time.
    clock()/10))
```

```
clf.fit(X_norm, y)


y_rf=clf.predict(X_norm)   # prediccion mediante RF
cm=confusion_matrix(y,y_rf,labels=[1,0])

Accuracy = float(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])

if (cm[0,0]+cm[0,1]):
    Precision = float(cm[0,0])/(cm[0,0]+cm[0,1])
else:
    Precision = 0

Recall = float(cm[0,0])/(cm[0,0]+cm[1,0])

if (cm[0,0]+cm[0,1]):
    F = (2*Precision*Recall)/(Precision + Recall)
else:
    F = 0


print("*** Matriz de confusion ***")
print(cm)
print("\n")
print("Accuracy:  %.2f %s" %(100*Accuracy,"%") )
print("Precision: %.2f %s" %(100*Precision,"%") )
print("Recall:    %.2f %s" %(100*Recall,"%") )
print("F:         %.2f %s" %(100*F,"%") )

## maquina de soporte vectoriales

clf = svm.SVC(kernel = 'rbf', C = 300, gamma = 0.01, class_weight= 'balanced
    ')
clf.fit(X_norm, y)


y_svm=clf.predict(X_norm)   # prediccion mediante SVM
cm=confusion_matrix(y,y_svm,labels=[1,0])

Accuracy = float(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])

if (cm[0,0]+cm[0,1]):
    Precision = float(cm[0,0])/(cm[0,0]+cm[0,1])
else:
    Precision = 0

Recall = float(cm[0,0])/(cm[0,0]+cm[1,0])

if (cm[0,0]+cm[0,1]):
    F = (2*Precision*Recall)/(Precision + Recall)
else:
    F = 0


print("*** Matriz de confusion ***")
print(cm)
print("\n")
print("Accuracy:  %.2f %s" %(100*Accuracy,"%") )
print("Precision: %.2f %s" %(100*Precision,"%") )
print("Recall:    %.2f %s" %(100*Recall,"%") )
print("F:         %.2f %s" %(100*F,"%") )
```

Rutina D.6: Ajuste de los modelos sobre el conjunto de entrenamiento.

```
# eleccion del numero de repeteciones

num_rep = 100

# lista de entrenamiento

training_list = []

for i in range(num_rep):
    if i+1 < 10:
        training_list.append("training_0"+str(i+1))
    else:
        training_list.append("training_"+str(i+1))

## cross-validation regresion logistica ##

Accuracy_vect1  = np.zeros(num_rep)
Precision_vect1 = np.zeros(num_rep)
Recall_vect1    = np.zeros(num_rep)
Fscore_vect1    = np.zeros(num_rep)
cm_matrix1      = np.zeros((num_rep,4))

for i in range(num_rep):
    X_train, X_test, y_train, y_test = train_test_split(X_norm, y, test_size
        =0.3, random_state=(42+i))
    clf = LogisticRegression(penalty="l1",dual=False,C=1.0,fit_intercept=
        True,class_weight="balanced",random_state=int(time.clock()/10),
        verbose=10)
    clf.fit(X_train, y_train)
    y_lr = clf.predict(X_test)    # prediccion mediante regresion logistica
    cm = confusion_matrix( y_test,y_lr,labels=[1,0])

    Accuracy = float(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])

    if (cm[0,0]+cm[0,1]):
        Precision = float(cm[0,0])/(cm[0,0]+cm[0,1])
    else:
        Precision = 0

    Recall = float(cm[0,0])/(cm[0,0]+cm[1,0])

    if (Precision + Recall):
        F = (2*Precision*Recall)/(Precision + Recall)
    else:
        F = 0

    # rellenar vectores

    Accuracy_vect1[i]   = round(Accuracy * 100,2)
    Precision_vect1[i]  = round(Precision * 100,2)
    Recall_vect1[i]     = round(Recall * 100,2)
    Fscore_vect1[i]     = round(F * 100,2)
    cm_matrix1[i,0]     = cm[0,0]
    cm_matrix1[i,1]     = cm[0,1]
    cm_matrix1[i,2]     = cm[1,0]
    cm_matrix1[i,3]     = cm[1,1]

```

```
cv_1 = pd.DataFrame(columns = ["training"])
cv_1["training"]        = training_list
cv_1["true_positives"]  = cm_matrix1[:,0]
cv_1["false_negatives"] = cm_matrix1[:,1]
cv_1["false_positives"] = cm_matrix1[:,2]
cv_1["true_negatives"]  = cm_matrix1[:,3]
cv_1["accuracy"]        = Accuracy_vect1
cv_1["precision"]       = Precision_vect1
cv_1["recall"]          = Recall_vect1
cv_1["f_score"]         = Fscore_vect1

cv_1["frauds"]          = cv_1["true_positives"]  + cv_1["false_negatives"]
cv_1["no_frauds"]       = cv_1["false_positives"] + cv_1["true_negatives"]
cv_1["total_clients"]   = cv_1["frauds"] + cv_1["no_frauds"]

cv_1 = cv_1[["training","total_clients"] + cv_1.columns.tolist()[-3:-1] + cv
    _1.columns.tolist()[1:-3]]

## cross-validation  bosque aleatorio ##

Accuracy_vect2  = np.zeros(num_rep)
Precision_vect2 = np.zeros(num_rep)
Recall_vect2    = np.zeros(num_rep)
Fscore_vect2    = np.zeros(num_rep)
cm_matrix2      = np.zeros((num_rep,4))

for i in range(num_rep):

    X_train, X_test, y_train, y_test = train_test_split(X_norm, y, test_size
        =0.3, random_state=(42+i))
    clf = RandomForestClassifier(n_estimators = 10, min_samples_split =350,
        min_samples_leaf = 15, class_weight= 'balanced' ,random_state= int(
        time.clock()/10))
    clf.fit(X_train, y_train)
    y_rf = clf.predict(X_test)    # prediccion mediante RF
    cm = confusion_matrix( y_test,y_rf,labels=[1,0])



    Accuracy = float(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])

    if (cm[0,0]+cm[0,1]):
        Precision = float(cm[0,0])/(cm[0,0]+cm[0,1])
    else:
        Precision = 0

    Recall = float(cm[0,0])/(cm[0,0]+cm[1,0])

    if (Precision + Recall):
        F = (2*Precision*Recall)/(Precision + Recall)
    else:
        F = 0

    # rellenar vectores

    Accuracy_vect2[i]   = round(Accuracy * 100,2)
    Precision_vect2[i]  = round(Precision * 100,2)
    Recall_vect2[i]     = round(Recall * 100,2)
    Fscore_vect2[i]     = round(F * 100,2)
    cm_matrix2[i,0]     = cm[0,0]
    cm_matrix2[i,1]     = cm[0,1]
    cm_matrix2[i,2]     = cm[1,0]
```

```
113     cm_matrix2[i,3]      = cm[1,1]
114
115 cv_2 = pd.DataFrame(columns = ["training"])
116 cv_2["training"]         = training_list
117 cv_2["true_positives"]   = cm_matrix2[:,0]
118 cv_2["false_negatives"]  = cm_matrix2[:,1]
119 cv_2["false_positives"]  = cm_matrix2[:,2]
120 cv_2["true_negatives"]   = cm_matrix2[:,3]
121 cv_2["accuracy"]         = Accuracy_vect2
122 cv_2["precision"]        = Precision_vect2
123 cv_2["recall"]           = Recall_vect2
124 cv_2["f_score"]          = Fscore_vect2
125
126 cv_2["frauds"]           = cv_2["true_positives"]  + cv_2["false_negatives"]
127 cv_2["no_frauds"]        = cv_2["false_positives"] + cv_2["true_negatives"]
128 cv_2["total_clients"]    = cv_2["frauds"] + cv_2["no_frauds"]
129
130 cv_2 = cv_2[["training","total_clients"] + cv_2.columns.tolist()[-3:-1] + cv
    _2.columns.tolist()[1:-3]]
131
132 ## cross-validation  maquina de soportes vecttoriales ##
133
134 Accuracy_vect3  = np.zeros(num_rep)
135 Precision_vect3 = np.zeros(num_rep)
136 Recall_vect3    = np.zeros(num_rep)
137 Fscore_vect3    = np.zeros(num_rep)
138 cm_matrix3      = np.zeros((num_rep,4))
139
140 for i in range(num_rep):
141
142     X_train, X_test, y_train, y_test = train_test_split(X_norm, y, test_size
        =0.3, random_state=(42+i))
143     clf = svm.SVC(kernel = 'rbf', C = 100, gamma = 0.001, class_weight= '
        balanced')
144     clf.fit(X_train, y_train)
145     y_svm = clf.predict(X_test)    # prediccion mediante svm con kernel "rbf
        "
146     cm = confusion_matrix( y_test,y_svm,labels=[1,0])
147
148     Accuracy = float(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])
149
150     if (cm[0,0]+cm[0,1]):
151         Precision = float(cm[0,0])/(cm[0,0]+cm[0,1])
152     else:
153         Precision = 0
154
155     Recall = float(cm[0,0])/(cm[0,0]+cm[1,0])
156
157     if (Precision + Recall):
158         F = (2*Precision*Recall)/(Precision + Recall)
159     else:
160         F = 0
161
162     # rellenar vectores
163
164     Accuracy_vect3[i]    = round(Accuracy * 100,2)
165     Precision_vect3[i]   = round(Precision * 100,2)
166     Recall_vect3[i]      = round(Recall * 100,2)
167     Fscore_vect3[i]      = round(F * 100,2)
168
169     cm_matrix3[i,0]      = cm[0,0] # fraude detectado
170     cm_matrix3[i,1]      = cm[0,1] # fraude no detectado
```

```
    cm_matrix3[i,2]      = cm[1,0] # sospechoso
    cm_matrix3[i,3]      = cm[1,1] # normal detectado

cv_3 = pd.DataFrame(columns = ["training"])
cv_3["training"]         = training_list
cv_3["true_positives"]   = cm_matrix3[:,0]
cv_3["false_negatives"]  = cm_matrix3[:,1]
cv_3["false_positives"]  = cm_matrix3[:,2]
cv_3["true_negatives"]   = cm_matrix3[:,3]
cv_3["accuracy"]         = Accuracy_vect3
cv_3["precision"]        = Precision_vect3
cv_3["recall"]           = Recall_vect3
cv_3["f_score"]          = Fscore_vect3

cv_3["frauds"]           = cv_3["true_positives"]  + cv_3["false_negatives"]
cv_3["no_frauds"]        = cv_3["false_positives"] + cv_3["true_negatives"]
cv_3["total_clients"]    = cv_3["frauds"] + cv_3["no_frauds"]

cv_3 = cv_3[["training","total_clients"] + cv_3.columns.tolist()[-3:-1] + cv
    _3.columns.tolist()[1:-3]]

cv_3.head()
```

Rutina D.7: Creación del conjunto de datos para calcular los intervalos asintóticos de las medidas de rendimiento para los distintos modelos.

```
## Agregar informacion de los algoritmos

data_matrix["lr_pred"]  = y_lr
data_matrix["rf_pred"]  = y_rf
data_matrix["svm_pred"] = y_svm

# categorias

N = len(y)
categorias = np.zeros(N,dtype=int)

for i in range(N):

    if y[i]==0:

        if (y_lr[i] + y_svm[i] + y_rf[i]) == 1 :
            categorias[i]=1
        if (y_lr[i] + y_svm[i] + y_rf[i]) == 2:
            categorias[i]=2
        if (y_lr[i] + y_svm[i] + y_rf[i]) == 3:
            categorias[i]=3

    else:
        categorias[i] = 4

# agregar categorias

data_matrix["nm_category"] = categorias

## nombres a las catgorias

nm_category = data_matrix["nm_category"].tolist()
aux_list = []

sum_1=0
```

```
sum_2=0
sum_3=0
sum_4=0
sum_5=0

for i in range(len(nm_category)):
    if nm_category[i] == 0:
        aux_list.append("normal")
        sum_1+=1
    elif  nm_category[i] == 1:
        aux_list.append("suspected_A")
        sum_2+=1
    elif  nm_category[i] == 2:
        aux_list.append("suspected_B")
        sum_3+=1
    elif  nm_category[i] == 3:
        aux_list.append("suspected_C")
        sum_4+=1
    elif  nm_category[i] == 4:
        aux_list.append("fraudulent")
        sum_5+=1


data_matrix["ds_category"] = aux_list

columns_01 = ["id_client","locality","x_coord","y_coord","fraud_date","
    anomaly_cod","ind_fraud","lr_pred","rf_pred","svm_pred","nm_category","
    ds_category"]
columns_02 = data_matrix.columns.tolist()[7:-5]

data_matrix = data_matrix[columns_01+columns_02]

## INFORMACION ##

# Numero de clientes

print("CLIENTES NORMALES:                %d" %(sum_1))
print("CLIENTES SOSPECHOSOS_A:           %d" %(sum_2))
print("CLIENTES SOSPECHOSOS_B:           %d" %(sum_3))
print("CLIENTES SOSPECHOSOS_C:            %d" %(sum_4))
print("CLIENTES FRAUDULENTOS:             %d" %(sum_5))
print("-------------------------------------------")
print ("TOTAL:                            %d" %(sum_1+sum_2+sum_3+sum_4+sum_5))
```

Rutina D.8: Tipo de clientes basados en los resultados obtenidos de los modelos.

```

# graficos

%matplotlib inline
plt.style.use('ggplot')
#fig.set_size_inches(16.5, 8.5)
#fig, ax = plt.subplots()
my_xticks = ['Enero','Febrero','Marzo','Abril','Mayo',
             'Agosto', 'Junio','Julio','Agosto','Septiembre','Octubre','
                 Nomviembre']
#plt.xticks(x, my_xticks)

## grafico del consumo mensual durante los annos 2010 al 2014

```

```
a=range(1,13)

fig, ax = plt.subplots()
fig.set_size_inches(16.5, 8.5)

plt.xlim(1,12)
plt.plot(a,df5_plot,color='#D52B80',linestyle="-", marker='o',label='2014: mean = %.2f, std= %.2f' %(np.mean(df5_plot),np.std(df5_plot))) # amarillo 2014
plt.plot(a,df4_plot,color='#000000',linestyle="-", marker='o',label='2013: mean = %.2f, std= %.2f' %(np.mean(df4_plot),np.std(df4_plot))) # rojo 2013
plt.plot(a,df3_plot,color='#FFAA00',linestyle="-", marker='o',label='2012: mean = %.2f, std= %.2f' %(np.mean(df3_plot),np.std(df3_plot))) # azul 2012
plt.plot(a,df2_plot,color='#C93827',linestyle="-", marker='o',label='2011: mean = %.2f, std= %.2f' %(np.mean(df2_plot),np.std(df2_plot))) # verde 2011
plt.plot(a,df1_plot,color='#DD7B71',linestyle="-", marker='o',label='2010: mean = %.2f, std= %.2f' %(np.mean(df1_plot),np.std(df1_plot))) #negro 2010

plt.axvline(8, color='k',linestyle='--')
ax.set_xlim([0.75,12.25])
#ax.set_ylim([-0.75,95])

axis_font = {'fontname':'Arial', 'size':'20'}
axis_font_2 = {'fontname':'Arial', 'size':'14'}
plt.xlabel('Mes', **axis_font)
plt.ylabel('Consumo [m^3]', **axis_font)
#ll = plt.legend(loc='upper right')
#plt.title("* Consumo del cliente : %d *" %(int(ds_client)))

plt.title("Cliente con un consumo Normal" ,axis_font)

my_xticks = ['Enero','Febrero','Marzo','Abril','Mayo',
             'Junio','Julio','Agosto','Septiembre','Octubre','Nomviembre','
                 Diciembre']
plt.xticks(a, my_xticks, **axis_font_2)

legend = ax.legend(loc='upper right', shadow=True, fontsize='x-large')

# Put a nicer background color on the legend.
legend.get_frame().set_facecolor('white')

## grafico del consumo promedio anual durante los annos 2010 al 2014

year = [2010,2011,2012,2013,2014]
mean_vector = [np.mean(df1_plot),np.mean(df2_plot),np.mean(df3_plot),
               np.mean(df4_plot),np.mean(df5_plot)]
fig, ax = plt.subplots()
fig.set_size_inches(16.5, 8.5)
plt.xlim(2009.75,2014.25)
plt.plot(year,mean_vector, marker='o')
axis_font = {'fontname':'Arial', 'size':'20'}
axis_font_2 = {'fontname':'Arial', 'size':'14'}
plt.xlabel('Anno', **axis_font)
plt.ylabel('Consumo [m^3]', **axis_font)
#ll = plt.legend(loc='upper right')
#plt.title("* Consumo del cliente : %d *" %(int(ds_client)))
my_xticks = ['2010','2011','2012','2013','2014']
plt.xticks(year, my_xticks, **axis_font_2)
```

```
plt.title("Consumo promedio de un cliente fraudulento" ,axis_font)
ax.set_ylim([-0.75,40])
```

Rutina D.9: Gráfica de los consumos de los clientes.

II) Rutinas desarrolladas en R

```
## Librerias

install.packages("spatstat",repos="http://cran.us.r-project.org")
install.packages("ggmap",repos="http://cran.us.r-project.org")
install.packages("sp",repos="http://cran.us.r-project.org")
install.packages("readxl",repos="http://cran.us.r-project.org")
install.packages("maptools",repos="http://cran.us.r-project.org")
install.packages("ggplot2",repos="http://cran.us.r-project.org")
install.packages("DCluster",repos="http://cran.us.r-project.org")
install.packages("NbClust",repos="http://cran.us.r-project.org")

library(spatstat)
library(ggmap)
library(sp)
library(readxl)
library(maptools)
library(ggplot2)
library(DCluster)
library(NbClust)
```

Rutina D.10: Librerías necesarias para trabajar.

```
## intervalo de confianza para muestras grandes

norm_interval <- function(data, conf.level = 0.95) {
    z        <- qnorm((1 - conf.level)/2, lower.tail = FALSE)
    xbar     <- mean(data)
    variance <- var(data)
    sdx      <- sqrt(variance/length(data))
    return(c(xbar - z * sdx, xbar + z * sdx))
 }

## informacion de las muestras

info_ci <- function(data, conf.level = 0.95){
    nm_sample <- length(data)
    mean <- round(mean(data),digits=2)
    std  <- round(sqrt(var(data)),digits=2)
    ci_lower <- round(norm_interval(data, conf.level = 0.95)[1],digits=2)
    ci_upper <- round(norm_interval(data, conf.level = 0.95)[2],digits=2)

    return (c(nm_sample,mean,std,ci_lower,ci_upper))
}

# data frame con la informacion de los estimadores por cada  dataframe

ci_df <- function(data_1,data_2,data_3, conf.level = 0.95) {
    columns_stimator <- c("accuracy","precision","recall","f_score")
    aux_matrix <- matrix(0, 4*3, 5)
```

```
28      aux_name   <- c()
29      cont <-1
30      for (i in 1:4){
31
32          aux_list_1 <- info_ci(c(sapply(data_1[columns_stimator[i]], as.
                numeric)),conf.level)
33          aux_list_2 <- info_ci(c(sapply(data_2[columns_stimator[i]], as.
                numeric)),conf.level)
34          aux_list_3 <- info_ci(c(sapply(data_3[columns_stimator[i]], as.
                numeric)),conf.level)
35          aux_matrix[cont,] <- aux_list_1
36          aux_name[cont] <- paste(c(columns_stimator[i],"_lr"), collapse = "")
37          cont <-cont +1
38          aux_matrix[cont,] <- aux_list_2
39          aux_name[cont] <- paste(c(columns_stimator[i],"_rf"), collapse = "")
40          cont <-cont +1
41          aux_matrix[cont,] <- aux_list_3
42          aux_name[cont] <- paste(c(columns_stimator[i],"_svm"), collapse = ""
                )
43          cont <-cont +1
44      }
45
46      stimator  <-aux_name
47      nm_sample <- aux_matrix[,1]
48      mean      <- aux_matrix[,2]
49      std       <- aux_matrix[,3]
50      CI_lower  <- aux_matrix[,4]
51      CI_upper  <- aux_matrix[,5]
52
53
54      return (data.frame(stimator,nm_sample, mean, std ,CI_lower,CI_upper))
55  }
56
57  # varios graficos al mismo tiempo
58
59  multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
60    library(grid)
61
62    # Make a list from the ... arguments and plotlist
63    plots <- c(list(...), plotlist)
64
65    numPlots = length(plots)
66
67    # If layout is NULL, then use 'cols' to determine layout
68    if (is.null(layout)) {
69      # Make the panel
70      # ncol: Number of columns of plots
71      # nrow: Number of rows needed, calculated from # of cols
72      layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
73                       ncol = cols, nrow = ceiling(numPlots/cols))
74    }
75
76   if (numPlots==1) {
77      print(plots[[1]])
78
79    } else {
80      # Set up the page
81      grid.newpage()
82      pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))
83
84      # Make each plot, in the correct location
85      for (i in 1:numPlots) {
```

```
86        # Get the i,j matrix positions of the regions that contain this
              subplot
87        matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))
88
89        print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
90                                        layout.pos.col = matchidx$col))
91      }
92    }
93  }
94
95  # qq-plot
96
97  gg_qq <- function(x, distribution = "norm", ..., line.estimate = NULL, conf
        = 0.95,
98                    labels = names(x),name){
99    q.function <- eval(parse(text = paste0("q", distribution)))
100   d.function <- eval(parse(text = paste0("d", distribution)))
101   x <- na.omit(x)
102   ord <- order(x)
103   n <- length(x)
104   P <- ppoints(length(x))
105   df <- data.frame(ord.x = x[ord], z = q.function(P, ...))
106
107   if(is.null(line.estimate)){
108     Q.x <- quantile(df$ord.x, c(0.25, 0.75))
109     Q.z <- q.function(c(0.25, 0.75), ...)
110     b <- diff(Q.x)/diff(Q.z)
111     coef <- c(Q.x[1] - b * Q.z[1], b)
112   } else {
113     coef <- coef(line.estimate(ord.x ~ z))
114   }
115
116   zz <- qnorm(1 - (1 - conf)/2)
117   SE <- (coef[2]/d.function(df$z)) * sqrt(P * (1 - P)/n)
118   fit.value <- coef[1] + coef[2] * df$z
119   df$upper <- fit.value + zz * SE
120   df$lower <- fit.value - zz * SE
121
122   if(!is.null(labels)){
123     df$label <- ifelse(df$ord.x > df$upper | df$ord.x < df$lower, labels[ord
          ],"")
124     }
125
126   p <- ggplot(df, aes(x=z, y=ord.x)) +
127     geom_point() +
128     geom_abline(intercept = coef[1], slope = coef[2]) +
129     geom_ribbon(aes(ymin = lower, ymax = upper), alpha=0.2) +
130     labs(title = paste0("Normal QQ-Plot - ", name)) +
131     labs(x = "",y=name)
132   if(!is.null(labels)) p <- p + geom_text( aes(label = label))
133   return (p)
134
135 }
136
137 multy_hist <- function(data){
138
139     columns_stimator <- c("accuracy","precision","recall","f_score")
140     par(mfrow=c(2,2))
141
142     p1 <- ggplot(data=data, aes(data$accuracy)) +
143       geom_histogram(aes(y=..density..),binwidth=.5,colour="black", fill="
          white") +
```

```
144         geom_density(alpha=.2, fill="#FF6666") +
145         labs(title=paste0("Histogram - ", columns_stimator[1])) +
146         labs(x="percentage", y="Count")
147 
148       p2 <- ggplot(data=data, aes(data$precision)) +
149         geom_histogram(aes(y=..density..),binwidth=.5,colour="black", fill="
              white") +
150         geom_density(alpha=.2, fill="#FF6666") +
151          labs(title=paste0("Histogram - ", columns_stimator[2])) +
152         labs(x="percentage", y="Count")
153 
154       p3 <- ggplot(data=data, aes(data$recall)) +
155         geom_histogram(aes(y=..density..),binwidth=.5,colour="black", fill="
              white") +
156         geom_density(alpha=.2, fill="#FF6666") +
157          labs(title=paste0("Histogram - ", columns_stimator[3])) +
158         labs(x="percentage", y="Count")
159 
160       p4 <- ggplot(data=data, aes(data$f_score)) +
161         geom_histogram(aes(y=..density..),binwidth=.5,colour="black", fill="
              white") +
162         geom_density(alpha=.2, fill="#FF6666") +
163          labs(title=paste0("Histogram - ", columns_stimator[4])) +
164         labs(x="percentage", y="Count")
165 
166 
167     return (multiplot(p1, p2, p3, p4, cols=2))
168 }
169 
170 multy_qq <- function(data){
171 
172     p1 <- gg_qq(data$accuracy,name = "accuracy")
173     p2 <- gg_qq(data$precision,name = "precision")
174     p3 <- gg_qq(data$recall,name = "recall")
175     p4 <- gg_qq(data$f_score,name = "f_score")
176 
177     return (multiplot(p1, p2, p3, p4, cols=2))
178 }
179 
180 multy_nt <- function(data){
181 
182     aux_01 <- shapiro.test(data$accuracy)
183     aux_02 <- shapiro.test(data$precision)
184     aux_03 <- shapiro.test(data$recall)
185     aux_04 <- shapiro.test(data$f_score)
186 
187     w_statistic <- c(aux_01$s,aux_02$s,aux_03$s,aux_04$s)
188     w_statistic <- round(w_statistic, 3)
189     p_value     <- c(aux_01$p,aux_02$p,aux_03$p,aux_04$p)
190     p_value     <- round(p_value, 3)
191 
192     evaluation  <- c()
193 
194     for(i in 1:4){
195 
196         if(p_value[i]>0.05){
197             evaluation  <- c(evaluation,"accepting null hypothesis")
198         }
199 
200         else{
201             evaluation  <- c(evaluation,"reject null hypothesis")
202         }
```

```
203
204     }
205
206     stimator     <- c("accuracy","precision","recall","f_score")
207
208     return (data.frame(stimator, w_statistic,p_value))
209
210 }
211
212 multy_test <- function(data){
213
214     p1 <-multy_hist(data)
215     p2 <- multy_qq(data)
216     p3 <- multy_nt(data)
217
218
219     return(c(p1,p2))}
```

Rutina D.11: Definición de algunas funciones que serán de utilidad en otras rutinas.

```
1
2 # cargar datos
3
4 commune <- "\\CARTAGENA"
5
6 big_df <- list()
7 for (i in 1:3) {
8   df <- read_excel(paste(c(results_Dir,commune,"\\cross_validation.xlsx"),
        collapse = ""), sheet = i)
9   assign(paste0("cv_", i), df)
10  big_df[[i]] <- df
11 }
12
13 #  logistic regresion
14 head(cv_1)
15
16 p1 <-multy_hist(cv_1)
17 p2 <-multy_qq(cv_1)
18 p3 <- multy_nt(cv_1)
19
20 #  random forest
21 head(cv_2)
22 p1 <-multy_hist(cv_2)
23 p2 <-multy_qq(cv_2)
24 p3 <-multy_nt(cv_2)
25
26 #  svm kernel rbf
27 head(cv_3)
28
29 p1 <-multy_hist(cv_3)
30 p2 <-multy_qq(cv_3)
31 p3 <-multy_nt(cv_3)
32
33 # informacion de los estimadores
34
35 ci_df(cv_1,cv_2,cv_3)
```

Rutina D.12: Tablas y gráficos de los intervalos de confianza de las medidas de rendimientos para los disntintos modelos.

```
# cargar datos

commune <- "CARTAGENA"

data_df <- read.csv(paste(c(results_Dir,"\\",commune,"\\summary_",commune,".csv"), collapse = ""),sep = ";")


## filtro ##

mask_zeros <- data_df$x_coord >0
mask_f1    <- data_df$x_coord < 300000
mask_f2    <- data_df$y_coord < 6286500
mask_f3    <- data_df$y_coord > 6282000

data_df <- data_df[mask_zeros & mask_f1 & mask_f2 & mask_f3,]

## Grafico ##

LONGITUD <- data_df$x_coord
LATITUD  <- data_df$y_coord
LABEL    <- data_df$ds_category
label <- factor(LABEL, levels = c("fraudulent", "suspected_C", "suspected_B","suspected_A","normal"))

p <- qplot(LONGITUD,LATITUD, data =  data_df,colour = label)
p +
  #labs(title =  paste(c(commune ,"CLIENTS"), collapse = " ") )+
  labs(title =  "Clientes Cartagena" )+
  theme(panel.background = element_rect(fill = "black"))+
  scale_colour_manual(values = c("red", "brown","orange","yellow","white"))

## Grafico ##

data_1 <- data_df[data_df$nm_category >3,]

LONGITUD <- data_1$x_coord
LATITUD  <- data_1$y_coord
LABEL    <- data_1$ds_category
label <- factor(LABEL)


p <- qplot(LONGITUD,LATITUD, data =  data_1,colour = label)
p +
  #labs(title =  paste(c(commune ,"CLIENTS"), collapse = " ") )+
  labs(title =  "Clientes Fraudulentos - Cartagena" )+
  theme(panel.background = element_rect(fill = "black"))+
  scale_colour_manual(values = c("red"))

## Trasformar datos a formato ppt ##

data_1 <- data_df[data_df$nm_category >3,]

LONGITUD <- data_1$x_coord
LATITUD  <- data_1$y_coord
LABEL    <- data_1$ds_category
#label <- factor(LABEL, levels = c("fraudulent", "suspected_C", "suspected_B","suspected_A","normal"))


delta <- 10
```

```
x_min <- min(LONGITUD) -delta
x_max <- max(LONGITUD) + delta
y_min <- min(LATITUD) - delta
y_max <- max(LATITUD) + delta
w_1 <- convexhull.xy(LONGITUD,LATITUD)
w_2 <-owin( c(x_min,x_max), c(y_min,y_max))
Coordinates <-ppp(LONGITUD, LATITUD,w_2)

par(mfrow = c(1,1))
plot(Coordinates)
plot(w_1,add=TRUE)

Coordinates <-unmark(as.ppp(Coordinates))

# Test de aleatoriedad

##  metodo cuadratura ##

quadrat.test(Coordinates, nx = 3, ny = 2, method = "Chisq")

## Test de Clark-Evans ##

clarkevans.test(Coordinates, correction = "Donnelly")

## Funciones K de Ripley y G de Digglet ##

coordK <- Kest(Coordinates, correction = "best")
coordG <- Gest(Coordinates, correction = "best")
coordF <- Fest(Coordinates, correction = "best")
par(mfrow = c(1,3))
plot(coordG, main = "Funcion G", legendpos = "float")
plot(coordF, main = "Funcion F", legendpos = "float")
plot(coordK, main = "Funcion K", legendpos = "float")

p <- 0.05
n <-100
valor <- (p * (n + 1))



bandcoordK <- envelope(Coordinates, Kest, nsim = n, nrank = valor, global =
    TRUE )
#bandcoordG <- envelope(Coordinates, Gest, nsim = n, nrank = valor)
#plot(bandcoordK, ylab = expression(hat("K")), xlab = "Distancia",
 #     main = paste0("Simulacion usando Monte-Carlo\n n = 100, banda con p =
     ",p))
#plot(bandcoordG, ylab = expression(hat("G")), xlab = "Distancia",
 #     main = paste0("Simulacion usando Monte-Carlo\n n = 100, banda con p =
     ",p))


## clusters


cdata <- data_1
cluster<-NbClust(cdata[ , 3:4], distance = "euclidean", min.nc = 2, max.nc =
     4,
                 method = "complete", index = "ch")
cluster$All.index
cluster$Best.nc
cluster$Best.partition

```

```
117 label=factor(cluster$Best.partition)
118
119 cluster_data <- data.frame(label, LONGITUD, LATITUD)
120 LONGITUD <- cluster_data$LONGITUD
121 LATITUD  <- cluster_data$LATITUD
122 LABEL    <- cluster_data$label
123 label    <- factor(LABEL)
124
125 p <- qplot(LONGITUD,LATITUD, data = cluster_data,colour = label)
126 p +
127   labs(title = "Clusters Clientes Fraudulentos - Cartagena")+
128   theme(panel.background = element_rect(fill = "black"))+
129   scale_colour_manual(labels = c("GRUPO 1", "GRUPO 2", "GRUPO 3","GRUPO 4"),
130                       values = c("red", "white","yellow","green"))
```

Rutina D.13: Análisis espacial del conjunto de datos.

Bibliografía

American Water Works Association. (2012). "IWA/AWWA Water Audit Method".

Bottou, L., O. Chapelle, D. DeCoste, and J. Weston (Eds.) (2007). " Learning with large datasets (nips tutorial)".

Breiman, L. (1996) " Machine Learning - Bagging predictors". *Springer.*

Breiman, L. (2001) " Machine Learning - Random forests". *Springer.*

Caruana, R. and A. Niculescu-Mizil. (2006) " An empirical comparison of supervised learning algorithms". *In Intl. Conf. on Machine Learning.*

Cressie, N. (1998) " Fundamentals of spatial statistics.". *Collecting Spatial Data: Optimun Design of Experiments for Random Field W.G Muller, 9–33.*

Cristianini & Shawe-Taylor . (2000). "An Introduction to Support Vector Machines and Other Kernel-based Learning Methods". *Cambridge University Press.*

Duman & Ozcelik (2011). " Detecting credit card fraud by genetic algorithm and scatter search". *Expert Systems with Applications.*

Hastie, T., R. Tibshirani, & J. Friedman (2009). " The Elements of Statistical Learning". *Springer. 2nd edition.*

Humaid, E., & Barhoom, T. (2012). " Data Mining Based Fraud Detection Model for Water Consumption Billing System in MOG". *Islamic University of Gaza.*

Joachims, T. (2006). " Training Linear SVMs in Linear Time". *In Proc. of the Int'l Conf. on Knowledge Discovery and Data Mining.*

Magnus & Neudecker. (1999). " Matrix Differential Calculus with Applications in Statistics and Econometrics". *Wiley, New York*.

Murphy,Kevin. (2012). "Machine Learning-A Probabilistic Perpective". *The MIT Press,Massachusetts*.

Ngai, Wong, Yong, Chen & Sun. (2011). " The application of data mining techniques in financial fraud detection: a classification framework and an academic review of literature". *Decision Support Systems*.

OECD. (2007). "Financing water supply and sanitization in EECCA countries ans progress in achieving the water related millenium development goals".

Platt, J. (1998). " Using analytic QP and sparseness to speed training of support vector machines". *In NIPS*.

Platt, J. (2000). " Probabilities for sv machines. In A. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans(Eds.), Advances in Large Margin Classifiers ". *MIT Press*.

Quah & M. Sriganesh (2008). " Real-time credit card fraud detection using computational intelligence". *Expert Systems with Applications*.

Sahami, M. & T. Heilman (2006). " A Web-based Kernel Function for Measuring the Similarity of Short Text Snippets ". *In WWW conferenec*.

Schoelkopf, B. & A. Smola (2002). " Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond". *MIT Press*.

Shotton, J., A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake (2011). " Real-time human pose recognition in parts from a single depth image". *In CVPR*.

Solomon, J. (2015). " Numerical Algorithm - Method for Computer vision, Machine Learning, and Graphics". *CRC Press*.

Soulier Faure, M., Ducci, J., & Altamira, M. (2013). " Agua Potable, Saneamiento y los Objetivos del Milenio en América Latina y el Caribe". *Banco Interamericano de Desarrollo*.

Superintendencia de Servicios Sanitarios. (2014). "Informe de Gestión del Sector Sanitario".

Superintendencia de Servicios Sanitarios. (2016). “Tarifas vigentes”. Recuperado de: `http://www.siss.gob.cl/577/w3-propertyvalue-3512.html`

Tipping, M. (2001). “ Sparse bayesian learning and the relevance vector machine. J. of Machine Learning”. *Research 1, 211–244*

Wu, Y., H. Tjelmeland, and M. West (2007). “ Bayesian CART: Prior structure and MCMC computations”. *J.of Computational and Graphical Statistics 16(1), 44–66*