# *MINERÍA DE DATOS APLICADA A LA DETECCIÓN DE CLIENTES CON ALTA PROBABILIDAD DE FRAUDES EN SISTEMAS DE DISTRIBUCIÓN*

**ANDRÉS FELIPE RIOS VILLEGAS**

**KEVIN ALEJANDRO URIBE AGUIRRE**

**DIRECTOR:**
**ING.GUSTAVO ANDRÉS BETANCOURT OROZCO**

**UNIVERSIDAD TECNOLÓGICA DE PEREIRA**
**FACULTAD DE INGENIERÍAS**
**PROGRAMA DE INGENIERÍA ELÉCTRICA**
**PEREIRA**
**2013**

# *MINERIA DE DATOS APLICADA A LA DETECCIÓN DE CLIENTES CON ALTA PROBABILIDAD DE FRAUDES EN SISTEMAS DE DISTRIBUCIÓN*

**ANDRÉS FELIPE RIOS VILLEGAS**

**KEVIN ALEJANDRO URIBE AGUIRRE**

**DIRECTOR:**
**ING.GUSTAVO ANDRÉS BETANCOURT OROZCO**

**PROYECTO DE GRADO PARA OPTAR POR EL TÍTULO DE INGENIERO ELECTRICISTA**

**UNIVERSIDAD TECNOLÓGICA DE PEREIRA**
**FACULTAD DE INGENIERÍAS**
**PROGRAMA DE INGENIERÍA ELÉCTRICA**
**PEREIRA**
**2013**

# DEDICATORIA

***A Dios.***
*Por brindarnos la sabiduría y fortaleza suficiente, para afrontar las dificultades que se presentan a diario.*

***A nuestros padres.***
*Por habernos apoyado incondicionalmente en aquellos momentos donde necesitábamos de un consejo y un aliciente para nuestro desarrollo personal y profesional.*

***A nuestros hermanos y familiares.***
*Por su acompañamiento y apoyo a través del tiempo.*

*Andrés Felipe Rios Villegas*
*Kevin Alejandro Uribe Aguirre*

# AGRADECIMIENTOS

*Agradecimientos sinceros al Ingeniero Gustavo Andrés Betancourt Orozco, por aceptarnos la propuesta de ser nuestro director de proyecto; por sus orientaciones y total disposición de ayudarnos en momentos críticos a lo largo del mismo.*

*Gratitud y agradecimiento, al Ingeniero Luis Hernando Martínez Rubio por asesorarnos en una etapa importante del proyecto.*

*A nuestros Padres, hermanos y Familiares que hicieron parte de este importante logro.*

*Andrés Felipe Rios Villegas*
*Kevin Alejandro Uribe Aguirre*

# *TABLA DE CONTENIDO*

# LISTA DE FIGURAS

# LISTA DE TABLAS

# *LISTA DE ABREVIATURAS*

| | |
|---|---|
| CREG | Comisión Reguladora de Energía y Gas |
| CV | Cross-Validation |
| IA | Inteligencia Artificial |
| IPI | Indice de Potencialidad de Infracción |
| KDD | Knowledge Discovery in Databases |
| KKT | Karush-Kuhn-Tucker |
| LIBSVM | Library for Support Vector Machine |
| NTL | Non-Technical Loss |
| QP | Quadratic Programming |
| RBF | Radial Basis Function |
| RNA | Redes Neuronales Artificiales |
| SMO | Sequential Minimal Optimization |
| SV | Support Vector |
| SVM | Support Vector Machine |
| UPME | Unidad de Planeamiento Minero – Energético |

# *CAPITULO 1 - INTRODUCCIÓN*

## *1.1 PRELIMINARES*

Las pérdidas no técnicas de energía eléctrica, son una problemática que las empresas distribuidoras y/o comercializadoras de energía enfrentan a diario. Es importante que cada empresa comercializadora de energía eléctrica cuente con un programa de recuperación de pérdidas que permita reducir al máximo este tipo de anomalías, en el sector comercializador de energía.

Resolver este problema no es para nada fácil, sí tenemos en cuenta que toda la energía disponible para la venta en un sistema eléctrico regulado, debe contar con indicadores de consumo para cada usuario residencial que compone un sistema eléctrico de distribución. De hecho, muchos países en el mundo se ven obligados a investigar nuevas metodologías que permitan una reducción de dicho fenómeno.

Uno de los inconvenientes más importantes con los que cuentan las empresas distribuidoras de energía en los planes de reducción de pérdidas, es el soporte financiero; por tal motivo, se buscan planes más eficientes y económicos que garanticen una recuperación notable en dichos planes, con resultados cada vez más satisfactorios.

En los sistemas de distribución de energía que se presentan actualmente, es de suma importancia contar con metodologías de clasificación, basadas en la extracción de patrones característicos capaces de hacer frente a grandes cantidades de datos, donde se busca clasificar y caracterizar el comportamiento de los consumidores de electricidad.

Utilizar métodos basados en minería de datos, es el primer paso para la solución y recuperación de pérdidas no técnicas en sistemas de distribución de energía, buscando así resultados más confiables y satisfactorios, para la construcción de nuevos métodos con mayores convergencias.

Esta investigación, tiene como enfoque principal la utilización de minería de datos y como herramienta de extracción de características, las SVMs (Support vector machines) para la detección de clientes fraudulentos en sistemas de distribución residenciales de energía, donde el fraude es una de las principales causas de pérdidas de ingresos para muchas empresas comercializadoras. Dicho problema se centra en la identificación del fraude.

## *1.2 DESCRIPCIÓN DEL PROYECTO*

El fraude en el ámbito del campo de la electricidad, se define como la acción ilegal llevada a cabo por el usuario con el fin de reducir la cantidad de energía consumida, que infiere en una facturación errónea de dicho consumo. El fraude eléctrico está asociado a las actividades de pérdidas no técnicas.

La minería de datos es un proceso que tiene como propósito fundamental analizar, extraer y almacenar información relevante de amplias bases de datos, las cuales contienen historiales de consumo de energía por medio de herramientas estadísticas y probabilísticas. En este proyectos se empleará la máquina de soporte vectorial (SVM), la cual es una herramienta muy poderosa para la clasificación de datos y elección con una alta probabilidad, qué datos pueden ser importantes para la decisión que se encuentra en proceso de análisis.

Esta investigación utiliza como fuente de información perfiles de carga de usuarios residenciales con el fin de construir y caracterizar dichos comportamientos, por medio de patrones de consumo reales que se pueden presentar para un periodo de tiempo determinado y permitir su representación cuantificada, por medio de herramientas computacionales que definen su estado (normal o sospechoso). Es aquí donde la SVM se encarga de utilizar los perfiles de carga y clasificarlos según su comportamiento.

Las SVMs se basan principalmente en aprendizajes supervisados que requieren de etapas previas de entrenamiento, y es aquí donde se buscan modelos de clasificación más eficientes y confiables, que garanticen una alta precisión sin importar el número de perfiles de carga que se desea estudiar.

## *1.3 OBJETIVOS DE LA INVESTIGACIÓN*

### *1.3.1 Objetivo general*

Proponer e implementar una metodología para el estudio de clientes con alta probabilidad de fraude, en sistemas de distribución por medio de la minería de datos, aplicando técnicas inteligentes como herramienta de clasificación y extracción de características.

### *1.3.2 Objetivos específicos*

- Desarrollar una metodología que permita la simulación de perfiles de carga, teniendo en cuenta patrones de referencia reales.

- Implementar una metodología de pre-procesamiento de datos, que logre un mejor desempeño para el modelo desarrollado.

- Desarrollar un modelo de clasificación de datos basado en técnicas computacionales.

## *1.4 APORTES DE LA INVESTIGACIÓN*

- Implementación de un modelo ajustado a la identificación de perfiles de carga sospechosos utilizando métodos computacionales, eficientes y económicos para empresas distribuidoras de energía eléctrica.

- Reducción de los costos operacionales de visitas a campo, para las cuadrillas encargadas de lectura y normalización.

- El modelo desarrollado proporciona un sistema de herramientas con información útil para las empresas distribuidoras de energía, enfocado a la reducción de pérdidas no técnicas.

- Se propone una metodología que logra asimilar gran cantidad de datos, en situaciones donde el tiempo es un factor fundamental para el estudio de perfiles de carga pertenecientes a un determinado sistema de distribución residencial.

# *CAPITULO 2 - REVISIÓN LITERARIA*

En los sistemas de alimentación regulada la información de consumo del cliente es importante para la gestión de la demanda de energía y la planificación del sistema o la definición de mejores tarifas. En los mercados eléctricos regulados el conocimiento de los patrones de consumo del cliente (***perfil de carga***), es extremadamente importante para el logro de un acuerdo en el precio de la electricidad entre consumidores y proveedores, y la definición de políticas de mercado y servicios, al igual que el conocimiento de las necesidades de sus clientes [1].

Una de las herramientas importantes utilizando estos datos son los ***perfiles de carga*** para las clases de consumidores, y con los cuales se busca que las nuevas herramientas de clasificación de clientes sean capaces de tratar con grandes cantidades de datos y con todos los problemas asociados a las bases de datos reales, es decir, el ruido, los valores perdidos y daños al equipo, entre otros [1].

Teniendo en cuenta que el aumento de pérdidas no técnicas para las empresas distribuidoras de energía generan grandes pérdidas financieras, la CREG como ente regulador implementó una metodología para establecer planes de reducción de pérdidas no técnicas en los sistemas de distribución local, en la resolución CREG 172 de 2011 [2].

Uno de los inconvenientes más importantes con los que cuenta la empresa distribuidora de energía en los planes de reducción de pérdidas es el soporte financiero; por tal motivo se buscan planes más eficientes y económicos que generen una recuperación notable en dichos planes.

## *2.1 CONCEPTO DE DEMANDA Y PERFIL DE CARGA*

Se entiende como ***demanda*** a la cantidad de potencia que un consumidor utiliza en cualquier momento (variable en el tiempo). Dicho de otra forma: la demanda de una instalación eléctrica en los terminales receptores, tomada como un valor medio en un intervalo determinado. El período durante el cual se toma el valor medio se denomina intervalo de demanda [3].

La variación de la demanda en el tiempo para una carga dada, origina el ciclo de carga que es un ***perfil de carga*** (demanda vs tiempo).

Un ***perfil de la carga*** puede ser definido, como un patrón de demanda de electricidad de un consumidor, o un grupo de consumidores, durante un período determinado de tiempo [1].

Un ***perfil de carga mensual*** está formado por la cantidad de potencia obtenida en intervalos mensuales. El perfil de carga mensual, brinda una indicación de las características de consumo de un usuario que pertenece a un sistema. Su análisis debe conducir a conclusiones similares a las obtenidas con curvas de carga anual, pero proporcionan mayores detalles sobre la forma en que han venido variando durante el periodo histórico y constituye una base para determinar las tendencias predominantes de las cargas del sistema y predecir un comportamiento de consumo normal o anormal, dado el caso.

La mayoría de las compañías de electricidad distinguen el comportamiento de la demanda sobre una base de clases, caracterizando cada clase con un ***perfil de carga*** típico para clientes con un perfil normal de consumo, el cual representa el patrón de uso esperado o promedio de la carga para un cliente de esa clase. Tales perfiles describen los aspectos más importantes de la energía total consumida y su comportamiento dentro del sistema.

## *2.2 MINERÍA DE DATOS*

La minería de datos es una combinación de bases de datos y tecnologías de inteligencia artificial (IA) [4]. La minería de datos es el proceso de planteamiento de búsqueda y extracción de patrones, a menudo desconocido de grandes cantidades de datos mediante comparación de patrones u otras técnicas de razonamiento [5]. De hecho, muchos expertos creen que la minería de datos es el tercer campo más atractivo en la industria detrás de la Internet y almacenamiento de datos [4].

La minería de datos es un concepto que está despegando en el sector comercializador de energía eléctrica como un medio para encontrar información útil de amplias bases de datos, que contiene historiales de consumo de clientes que componen un determinado sistema eléctrico.

A través de los tiempos, varios métodos se han desarrollado para hacer frente a grandes volúmenes de datos, muchos de los cuales eran vistos como pasos importantes para las matemáticas de la época. Algunos de estos métodos incluyen transformadas rápidas de Fourier, análisis multivariado de regresión, así como toda una gama de métodos estadísticos. Más recientemente la visualización ha

sido ampliamente adoptada por los científicos como un medio de estudio a grandes grupos de datos cada vez mayores [4].

Antes de seguir hablando de minería de datos, es de suma importancia contar con algunos términos [4].

- ***DATO:*** Son algunos hechos, números o texto que pueden ser procesados por un ordenador. Hoy en día, las organizaciones están acumulando cada vez mayores cantidades de datos en diferentes formatos y bases de datos diferentes. Esto incluye:

  ***Los datos operacionales o transaccionales***, tales como, ventas, costos, inventarios, nómina y contabilidad.

  ***Datos no operacionales,*** como las ventas de la industria y datos de pronóstico.

  ***Los datos meta***, como el diseño lógico de bases de datos.

- ***INFORMACIÓN:*** Asociaciones o relaciones entre todos los datos de estudio que pueden proporcionar información. Por ejemplo el análisis de posibles clientes fraudulentos, que se ajustan a un perfil de carga determinado.

- ***CONOCIMIENTO:*** La información puede ser convertida en conocimiento de los patrones históricos y las tendencias futuras.

Con las tendencias actuales en la centralización de datos de una organización en grandes bases de datos, especialmente en un entorno comercial, el proceso de ***extracción*** de información útil se ha vuelto más formal. El proceso de ***extracción*** de información previamente desconocida, comprensible y procesable, permite a partir de grandes bases de datos utilizarla para tomar decisiones cruciales en el campo de investigación que se está estudiando.

La minería de datos es un proceso que puede tomar diferentes enfoques dependiendo del tipo de datos en cuestión y los objetivos deseados. Como esto sigue siendo en gran medida una disciplina en evolución, es mucho lo que se está llevando a cabo para determinar los procesos estándar para ambientes variados [4]. El proceso de minería de datos no es un procedimiento simple, porque a menudo involucra una variedad de ciclos de retroalimentación, ya que si se aplica una técnica en particular, el usuario puede determinar que los datos seleccionados son de mala calidad o que las técnicas aplicadas no produjeron los resultados

esperados. En tales casos, el usuario tiene que repetir y refinar los pasos anteriores; posiblemente reiniciar todo el proceso desde el principio.

Todo proyecto de minería de datos debe cumplir con una serie de procedimientos [4].

- ***EXPLORACIÓN DE DATOS:*** Es el proceso de recolección y exploración de datos. También se identifican los problemas de calidad de los datos y donde la definición del problema es vital. En la fase de exploración de datos, las herramientas tradicionales de análisis de datos, por ejemplo las estadísticas, se utilizan para explorar los datos.
- ***PREPARACIÓN DE DATOS:*** Se busca construir el modelo de datos para el proceso de modelado. Recoger, limpiar y dar formato a los datos, ya que algunas de las funciones de minería aceptan datos sólo en un determinado formato. En la fase de preparación de datos los datos se verifican varias veces sin ningún orden prescrito. Preparación de los datos para la herramienta de modelado mediante la selección de tablas, registros y atributos, son tareas típicas de esta fase.
- ***MODELAMIENTO:*** Expertos en minería de datos seleccionan y aplican las diversas funciones de minería, porque se pueden utilizar diversas funciones de minería de datos para un solo problema. Los expertos en minería de datos, deben evaluar cada modelo (*Máquina de soporte vectorial, lógica difusa, lógica borrosa*).
- ***EVALUACIÓN:*** Se busca evaluar el modelo. Si el modelo no satisface las expectativas, se regresa a la fase de modelado y reconstrucción del modelo cambiando sus parámetros, hasta que los valores óptimos se logren.
- ***DESPLIEGUE:*** Los resultados obtenidos son enviados a bases de datos o en otras aplicaciones. Los productos de Intelligent Miner le ayudará a seguir este proceso.

En resumen: La minería de datos es solo una etapa del proceso de extracción de conocimiento a partir de datos. Este proceso consta de varias fases como la preparación de datos (selección, limpieza y transformación), su exploración y auditoria, minería de datos propiamente dicha (desarrollo de modelos y análisis de datos), evaluación, difusión y utilización de modelos (Output). Además, el proceso de extracción incorpora diferentes técnicas (arboles de decisión, regresión lineal, redes neuronales artificiales, técnicas bayesianas, Máquinas de soporte vectorial) [6].

## *2.3 PERFILES DE CARGA CON TÉCNICAS EN MINERÍA DE DATOS*

Muchas investigaciones siguen en curso para estudiar herramientas de clasificación y caracterizar el comportamiento de los consumidores de electricidad. El perfil de carga ha sido identificado como un método adecuado de tratamiento de clientes sin equipo de medición, en intervalos de tiempo. Además los perfiles de carga también sirven como una herramienta para que las empresas de distribución puedan mejorar sus estrategias de mercado y ofrecer nuevos servicios, así como para desarrollar nuevas tarifas en el mercado regulado. Muchas técnicas diferentes, que van desde métodos convencionales a métodos más sofisticados, se han utilizado para el modelado de *perfiles de carga* [7].

Los *perfiles de carga* se utilizaron antes de que los consumidores fueran clientes regulados, para la formulación de las tarifas. Sin embargo, desde la regulación, la presión se intensifica y la necesidad de *perfiles de carga* de los clientes de electricidad es cada vez más importante, ya que el conocimiento adquirido a partir de la descripción del perfil de carga, se puede utilizar para diversos propósitos. Básicamente el modelo de *perfiles de carga* puede estar basado en la ubicación geográfica o tipo categórico; sin embargo, esto también puede depender de los diversos factores que contribuyen al estudio.

La mayor parte de la investigación realizada, fue la de captar las necesidades derivadas de la regulación de clientes, pero ha habido muy poco enfoque en la detección y predicción de pérdidas no técnicas en los servicios públicos de energía [7]. El fraude es una de las principales causas de la pérdida de ingresos en muchas áreas de negocio. Entre ellos, tarjeta de crédito, teléfono celular y el seguro, son los más destacados. Por lo tanto, una gran cantidad de trabajos de investigación han hecho frente al problema de la identificación del fraude. Al igual que en otras áreas de negocio, las empresas de distribución de electricidad, también pueden sufrir fraude de sus clientes. En Brasil, como en muchos otros países, las pérdidas de ingresos de las empresas de electricidad debido a los fraudes, puede ir tan alto como el 3% [8].

Las técnicas de minería de datos utilizados en este tipo de estudio, básicamente se dividen en cinco fases: planeamiento del problema, la recopilación de datos, los datos de pre-procesamiento, minería de datos y representación del conocimiento.

La fase de minería de datos se centra en el módulo de perfiles de carga que clasifica a los clientes en función de su comportamiento.

En [7] se analizan tres métodos de agrupamiento con el fin de clasificar a los clientes de electricidad en varios grupos. La técnica de agrupamiento también se

conoce, como aprendizaje no supervisado porque no hay ninguna clase de ser predicha. Los métodos son los siguientes:

- k-Means
- EM method
- COBWEB method

## *2.4 PÉRDIDAS DE ENERGÍA*

Pérdidas de potencia o pérdida de energía, se definen como la diferencia entre la energía suministrada o entregada y la energía facturada o consumida por el cliente, como se ilustra en la siguiente ecuación [9].

$$E_{LOSS} = E_{DELIVERED} - E_{SOLD} \quad \textbf{(2.1)}$$

Donde:

$E_{LOSS}$ : $Es\ la\ cantidad\ de\ energía\ perdida.$
$E_{DELIVERED}$ : $Representa\ la\ cantidad\ de\ energía\ suministrada.$
$E_{SOLD}$ : $Representa\ la\ cantidad\ de\ energía\ registrada\ o\ vendida.$

Fundamentalmente las pérdidas de energía se pueden dividir en dos áreas principales que son: ***pérdidas técnicas*** y ***pérdidas no técnicas*** [9]***.***

### *2.4.1 Pérdidas técnicas*

Las pérdidas técnicas se producen durante la transmisión y distribución, e involucran a la subestación, transformador y las pérdidas en las líneas de distribución. Estos incluyen las pérdidas resistivas en los alimentadores primarios de distribución, pérdidas al interior del transformador (perdidas resistivas en los devanados y pérdidas en el núcleo), las pérdidas resistivas en las redes secundarias y las pérdidas en la medición de energía [9]. Por lo tanto:

$$C_{LOSS} = U_{ElectricityCost} \quad x \quad E_{LOSS} + M_{MaintenanceCost} \quad \textbf{(2.2)}$$

Donde:

$C_{LOSS}$ : $la\ pérdida\ de\ ingreso\ debido\ a\ pérdidas\ técnicas.$
$U_{ElectricityCost}$ : $Representa\ el\ costo\ unitario\ de\ electricidad.$
$E_{LOSS}$ : $Representa\ la\ cantidad\ de\ energía\ pérdida.$
$M_{MaintenanceCost}$ : $Representa\ el\ mantenimiento\ y\ gastos\ operacionales$

### *2.4.2 Pérdidas no técnicas*

Las pérdidas no técnicas están relacionadas principalmente con el robo de electricidad y procesos administrativos defectuosos. Estos medios incluyen la manipulación de medidores para grabar menores tasas de consumo, la organización de lecturas falsas por medio de sobornos, o conexiones ilegales, donde dicha conexión no pasa por el medidor y la organización irregular de facturas por parte de empleados de entidades comercializadoras de energía, extendiendo facturas a más bajo costo, el ajuste de la posición del punto decimal en las facturas, o simplemente, haciendo caso omiso de las facturas pendientes de pago [9]. De acuerdo a esto, se tiene que:

$$C_{NTL} = C_{LOSS} - C_{TechnicalLoss} \quad \textbf{(2.3)}$$

Donde

$C_{NTL}$: $Es\ la\ componente\ de\ costo\ NTL.$
$C_{LOSS}$ : $Es\ la\ pérdida\ de\ ingreso\ debido\ a\ pérdidas\ técnicas.$
$C_{TechnicalLoss}$: $Representa\ la\ componente\ de\ costo\ de\ pérdida\ técnica.$

Aunque algunas pérdidas eléctricas de potencia son inevitables, se pueden tomar medidas para minimizar dicho fenómeno. Varias medidas han sido aplicadas a este fin, incluidos los basados en la tecnología y los que se basan en el esfuerzo y el ingenio humano. La reducción de *pérdidas no-técnicas,* es crucial para las empresas distribuidoras de energía. A medida que estas pérdidas se concentran en la red de baja tensión, sus orígenes se extienden a lo largo de todo el sistema y son más críticos en niveles residenciales bajos y pequeños comerciales; su recuperación es esencial y requiere de una inversión significativa [9].

## *2.5 IMPACTO DE LAS PÉRDIDAS DE ENERGÍA*

Las empresas distribuidoras de energía se ven fuertemente afectadas por las *pérdidas no- técnicas* debidas al fraude, imprecisión de la medición, errores de lectura o mala gestión de los sistemas de recogida de datos [10]. Las compañías eléctricas pierden grandes cantidades de dinero cada año debido al fraude por consumidores de electricidad y es difícil distinguir entre los clientes honestos y fraudulentos [11].

Las investigaciones se llevan a cabo por las empresas de servicios eléctricos para evaluar el impacto de las pérdidas técnicas en las redes de generación, transmisión y distribución, y el rendimiento global de las redes de energía. Las *pérdidas no- técnicas* constituyen una de las preocupaciones más importantes de los servicios públicos de distribución de electricidad en todo el mundo. En el año

2004, Tenaga Nasional Berhad (TNB) Sdn. Bhd, el único proveedor de electricidad en la península de Malasia, registró pérdidas de ingresos tan altos como EE.UU. $ 229 millones al año a consecuencia del robo de electricidad, medición defectuosa y errores de facturación. Las *pérdidas no-técnicas* que enfrentan las empresas de servicios eléctricos en los Estados Unidos se estiman entre el 0,5% y el 3,5% de los ingresos brutos anuales, que es relativamente bajo en comparación con las pérdidas que enfrentan las compañías de electricidad en los países en vía de desarrollo como Bangladesh, India y Pakistán [11].

Ahora, se analiza el comportamiento de las pérdidas de energía (técnicas y *no-técnicas*) en Colombia, que se ilustra en la ***figura 2.1***, se puede apreciar el comportamiento de las pérdidas del sistema de distribución, vista desde las ventas y de la demanda. La evolución histórica de las pérdidas en los sistemas de distribución, muestra una notable disminución en la última década, llegando a casi la mitad de su valor en porcentaje. De esta revisión, se aprecia que las pérdidas se estiman de manera preliminar en el 2011, en un 13% vista desde la demanda y en un 15.7% vistas desde la ventas [12]. Sin embargo estos porcentajes de pérdidas, siguen siendo considerables para las empresas distribuidoras de energía.

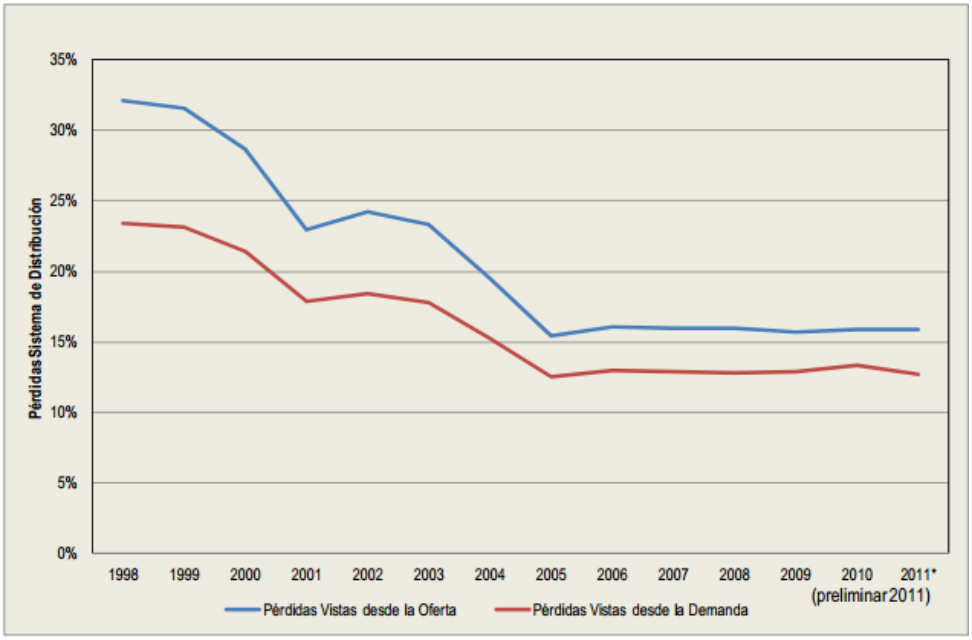



***Figura 2.1*** *Comportamiento Histórico de las Pérdidas de Distribución de Energía Eléctrica.* (Fuente: UPME, "Proyección de demanda de energía eléctrica y potencia máxima", 2012, p. 13)

La empresa Cedenar (Centrales Eléctricas de Nariño) [13], de cada 1000 GWh / año, 200 GWh / año, son asignados a perdidas en el sistema, donde el 98% corresponde a usuarios residenciales. Solo 800 GWh / año, son reconocidos por la empresa de energía.

A continuación analizaremos el porcentaje de pérdidas para este caso.

$$\%\ pérdida = \frac{1000\ {}^{Gwh}/_{año} - 800\ {}^{Gwh}/_{año}}{1000\ {}^{Gwh}/_{año}}\ x\ 100 = 20\% \qquad \textbf{(2.4)}$$

Se puede determinar que el índice de pérdidas es alto, lo cual se traduce en pérdidas económicas para la empresa distribuidora de energía.

Para EPM (Empresas Públicas de Medellín), el mantenimiento de pérdidas no-técnicas, asciende al orden de $ 15.000 millones. Tales costos se dividen en contratos de obra y compra de materiales [14].

Para CENS, (Centrales Eléctricas de Norte de Santander S.A), el costo de reducción de pérdidas no-técnicas, es de 13.088 millones de pesos en programas de apoyo a la operación, programas complementarios, etapas de macromedición y etapas de normalización [15].

Para ELECTRICARIBE, los costos de inversiones ejecutadas entre los años 2001 y 2007 ascienden al orden de 541.433 millones de pesos corrientes [16].

Lo anterior determina un amplio panorama de como las pérdidas no-técnicas repercuten en el comportamiento económico, social y administrativo de las empresas distribuidoras de energía.

## *2.6 SOLUCIONES ACTUALES PARA LA DISMINUCIÓN DE PÉRDIDAS NO TÉCNICAS*

El método más eficaz para reducir las pérdidas no-técnicas en su totalidad, es mediante el uso de contadores electrónicos inteligentes y sofisticados que hacen que las actividades fraudulentas sean más difíciles, y además más fáciles de detectar [11].

En los últimos años varios estudios de minería de datos y de investigación sobre la identificación de fraude y técnicas de predicción, se han llevado a cabo en el sector distribuidor de electricidad. Estos métodos estadísticos incluyen árboles de decisión, redes neuronales artificiales (RNA), descubrimiento de conocimiento en

bases de datos (KDDs) y clasificadores múltiples usando cross-validation y esquemas de votación. Entre estos métodos los perfiles de carga, son uno de los métodos más utilizados que se define como el patrón de consumo de electricidad de un cliente o grupo de clientes, en un período de tiempo [11].

En la actualidad el enfoque principal es la construcción de redes inteligentes que permitan la detección de clientes fraudulentos por medio del historial de consumo, perteneciente al sistema eléctrico que se esté evaluando.

En Colombia la aplicación de metodologías se realiza en busca de índices de potencialidad de infracción – IPI. Este procedimiento se basa en teorías probabilísticas, más específicamente, en la teoría de bayes, lo cual abre caminos para el estudio de dichos fenómenos aplicados a empresas comercializadoras de energía eléctrica [17], [18].

Actualmente, se espera contar con más información de las pérdidas en el sistema de distribución con la implementación del plan de reducción de pérdidas no técnicas propuesto por la CREG y la información consolidada en el estudio del sector de distribución y comercialización de electricidad, realizado por Asocodis y la Upme [12].

Algunas empresas distribuidoras de energía en Colombia buscan implementar una metodología novedosa para la reducción de pérdidas no-técnicas llamada ***sistema VECO*** [13], donde los medidores de cada usuario y sus acometidas se indagan durante 7 días. Basado en que este periodo, se normalizan los consumos. Tiene la potencialidad de medida centralizada y se instalan equipos con la mínima intervención del usuario.

## *2.7 MÉTODOS PARA EL ANÁLISIS DE CONSUMOS*

### *2.7.1 Por críticas de los consumos*

La mayoría de las comercializadoras de energía se limitan solo a este método; es decir, al análisis de las variaciones significativas contra el consumo histórico, generalmente en cuentas que han tenido una disminución del 20% o el 30%; cuando el consumo aumenta, esto no se realiza [37].

### *2.7.2 Por consumo estimado*

El requisito indispensable para hallar el consumo estimado es haber realizado el censo o aforo de carga y conocer el factor de utilización (F.U) del sector económico al que pertenece el cliente [37]. Estimar quiere decir aproximar; este consumo en kW-h/mes sale de la fórmula:

$$C.E = kW_{inst}\ x\ 720\ x\ F.U \quad \textbf{(2.5)}$$

Donde:
$C.E = Consumo\ Estimado.$
$kW_{inst} = Kilovatios\ instalados\ encontrados\ en\ el\ censo\ o\ aforo\ de\ carga.$
$720 = Horas\ al\ mes.$
$F.U = Factor\ de\ utilización.$

El factor de utilización es el porcentaje utilizado en equipos eléctricos que se tienen disponibles.

### *2.7.3 Por consumo proyectado*

Para este método de análisis, se debe tener una lectura actualizada y con la última lectura facturada, se obtiene el consumo proyectado de la siguiente forma:

$$Consumo\ Proyectado\left({}^{kW}/_{mes}\right) = \frac{(LR - Ulf)}{NDT} * 30 \quad \textbf{(2.6)}$$

Donde:
$LR = Lectura\ el\ día\ de\ la\ revisión.$
$Ulf = Última\ lectura\ facturada.$
$NDT = Número\ de\ dias\ transcurridos\ entre\ la\ ultima\ lectura\ y\ el\ dia\ de\ la\ revisión.$
$30 = Periodo\ de\ facturación\ de\ 30\ dias. Tambien\ puede\ ser\ de\ 60\ dias\ cuando\ el\ periodo\ es\ de\ facturación\ bimestral.$

### *2.7.4 Por capacidad del transformador de distribución*

Cuando se encuentran transformadores de distribución particulares o también llamados exclusivos, se pueden estimar los consumos, pero necesariamente, se debe conocer el factor de utilización. Se emplea un método similar cuando se conoce los kW instalados, pero en la fórmula se reemplaza por los multiplicados por el factor de potencia [37].

$$C.E = kVA * Cos(\varphi)\ x\ 720\ x\ F.U \quad \textbf{(2.7)}$$

Donde:
$C.E = Consumo\ Estimado.$
$kVA = Potencia\ nominal\ del\ transformador\ exclusivo\ de\ distribución.$
$Cos(\varphi) = Factor\ de\ potencia, se\ asume\ 0.9$
$720 = Horas\ al\ mes.$
$F.U = Factor\ de\ utilización.$

### *2.7.5 A partir del factor multiplicador*

Se parte de que en la medición semidirecta el factor multiplicador (FM) es igual a la relación de transformación de corriente (RTC); en indirecta, el factor multiplicador es igual a la relación de transformación de corriente (RTC), multiplicada por la relación de transformación de potencial (RTP) [37].

### *2.7.6 Por caracterización de consumo*

La tecnología CIF[1] por medio de un software especializado, permite caracterizar el consumo de cada cliente y determinar cuál es su comportamiento. Cuando la cuenta analizada se sale de este rango, el sistema por medio de gráficas muestra el comportamiento anómalo y emite una señal para que se genere una inspección. En la actualidad, este método ha demostrado ser uno de los más eficientes en los programas de recuperación de pérdidas de energía [37].

## *2.8 ACCIONES TÍPICAS SOBRE LOS COMPONENTES DE LAS PÉRDIDAS NO TÉCNICAS DE ENERGÍA*

En [38], el autor expone las siguientes acciones desde el punto de vista de las empresas comercializadoras de energía:

- En cuanto a las pérdidas no técnicas, no se miden de forma correcta; la selección no es confiable, se detectan y se corrigen a medias, pero no se controlan.

- Las pérdidas administrativas, solo se detectan por casualidad o cuando se presentan casos puntuales, pero no son fruto de programas definidos para ubicarlas. No se realiza un amarre real de los clientes, identificación de cuentas que no están en el sistema, críticas a la lectura y facturación y en general, auditorias al sistema de gestión comercial ya que por lo general se encuentra desactualizada.

- Las pérdidas por otros comercializadores, medianamente se miden, no se seleccionan; se detectan y se corrigen, pero no hay seguimiento.

Todos los componentes anteriores conducen al logro de la meta y la función y peso relativo de cada componente, determinan su incidencia sobre el resultado final. Como se puede observar, solo se logra en empresas que trabajan en todos los frentes que tienen las pérdidas de energía.

[1] CIF (Centro Internacional de Física), Entidad privada sin ánimo de lucro, fundada en diciembre del año 1986. Su misión es promover la investigación básica y aplicada para Colombia, algunos países de la región andina y el Caribe.

## *2.9 ETAPAS DE LA REDUCCIÓN DE PÉRDIDAS*

En [38] el autor expone las siguientes etapas de reducción de pérdidas de energía:

### *2.9.1 Medir*

Para determinar si existen pérdidas, la primera etapa es medir, determinando el monto y el lugar para así tomar decisiones. Al no tener mediciones completas, se incurre en estimativos que quitan precisión a los cálculos.

### *2.9.2 Seleccionar*

Una vez medido el circuito, se determina el volumen de las pérdidas, el lugar donde se encuentran y se hacen los análisis para la toma de decisiones.

### *2.9.3 Detectar*

Una vez se han seleccionado los circuitos a intervenir, le corresponde al equipo operativo detectar el terreno de las causas. Tradicionalmente se revisa la acometida en forma parcial; se le realiza al medidor la prueba de tiempo potencia con carga resistiva y algunos realizan la prueba de registro cuando el medidor tiene decimales.

### *2.9.4 Normalizar*

Medianamente se detectan las pérdidas o se hacen bien, pero a costos muy altos y las correcciones se hacen a medias, interviniendo recursos sólo en revisar varias veces. Cuando se tienen pérdidas de energía y no se corrigen de forma adecuada, es igual a cuando un médico tiene un paciente y le diagnostica una enfermedad y le da solo calmantes para su mal y no se dedica a curarlo definitivamente.

### *2.9.5 Controlar*

Una vez disminuidas las pérdidas se deben controlar; esto mediante vigilancia especial por medio del sistema, para luego enviar a revisar al terreno las que verdaderamente se salen de los rangos y resultan sospechosos. Esto garantiza un mejor acierto a menores costos y sin incomodar a los clientes que no manipulan su equipo de medidas.

# *CAPITULO 3 - Marco Teórico*

## *3.1 INTRODUCCIÓN*

Muchos de los problemas de tratamiento estadístico de información, se pueden agrupar en dos conjuntos: los problemas de clasificación o decisión y los problemas de regresión.

En los de clasificación, el interés se centra en saber si una determinada muestra en función de las características que presenta, pertenece a una u otra clase. En regresión, por el contrario, lo que se pretende es estimar una variable desconocida a partir de observaciones que guardan algún tipo de relación con ella. Tanto la decisión como la regresión, pueden verse como casos particulares de un problema de aproximación de funciones [19]

La búsqueda de una regla de decisión adecuada, puede interpretarse como la estimación de una función $g$, la cual asigna a cada punto del espacio de observación, un punto en el espacio de las clases (-1 y +1 en el caso binario). Esta búsqueda se lleva a cabo, usando un conjunto de *datos de entrenamiento* formado por *N* pares de muestras etiquetadas, distribuidas generalmente mediante una distribución de probabilidad desconocida *P(**x**, y)*.

Lo que se busca con la función *g* es que generalice; es decir, que clasifique correctamente un conjunto de datos que no pertenezcan al conjunto de entrenamiento. Esta función suele tomar la forma:

$$g(x) = sign\left(f(x)\right) \tag{3.1}$$

Así pues, suponiendo el caso binario, las muestras nuevas se asignarán a la clase +1 si $f(x) > 0$ o a la clase -1 $f(x) < 0$ . Al valor de $f(x)$ se le denomina *salida blanda*; mientras que a $sign\,(f(x))$, *salida dura*.

La clasificación, en el ámbito del Aprendizaje Máquina, consiste en encontrar una regla de decisión tal, que dada una muestra externa, ésta sea asignada a su clase correspondiente [19].

SVMs (Support Vector Machines) son una técnica muy útil para la clasificación de datos. Aunque la SVM se considera más fácil de usar que las redes neuronales, los usuarios que no están familiarizados con ella, a menudo obtienen resultados satisfactorios. Una tarea de clasificación generalmente consiste en separar los

datos en conjuntos de entrenamiento y prueba. Cada instancia en el conjunto de entrenamiento contiene un "valor objetivo" (es decir, las etiquetas de clase) y varios "atributos" (es decir, las características o variables observadas). El objetivo de la SVM, es producir un modelo (basado en los datos de entrenamiento) que predice el valor objetivo de los datos de prueba, dadas solamente los atributos de los datos de prueba [20]. Las SVMs son un conjunto de métodos novedosos de máquinas de aprendizaje, utilizadas para la clasificación, y se han convertido recientemente en un área activa de investigación intensa, con extensión a regresión. En SVM, la formación se realiza de una manera tal, de obtener un problema de programación cuadrática (QP) [21].

La SVM fue propuesta por Vapnik [22] y recientemente se ha aplicado en una amplia gama de problemas, que incluyen el reconocimiento de patrones, la bioinformática y la categorización de texto. Cuando se utiliza SVM, dos cuestiones deben resolverse: cómo elegir el subconjunto de entrada característica óptima para SVM, y cómo establecer los mejores parámetros del kernel. Tradicionalmente, los dos problemas se resuelven por separado, haciendo caso omiso de sus estrechas relaciones. Estos dos problemas son cruciales, ya que la elección del subconjunto de característica, influye en los parámetros apropiados en el kernel y viceversa [23].

## *3.2 COMPONENTES DE UN SISTEMA DE CLASIFICACIÓN*

En la resolución de problemas de toma de decisiones, la primera parte de la tarea consiste en clasificar el problema o la situación, para después, aplicar la metodología correspondiente que en buena medida, dependerá de esa clasificación.

Cuando se diseña un sistema de clasificación, es necesario definir cada uno de sus componentes. Los principales componentes de un sistema de clasificación general son [19]:

- **Clase**: Conjunto de entidades que comparten alguna característica que las diferencia de otras.
- **Clase de rechazo**: Conjunto de entidades que no se pueden etiquetar como ninguna de las clases del problema.
- **Extractor de características**: Subsistema que extrae información relevante para la clasificación, a partir de las entidades cuantificables.
- **Clasificador**: Subsistema que utiliza un vector de características de la entidad cuantificable y lo asigna a una de las *M* clases.

- **Evaluación de error de clasificación**: Error que se comete al realizar la clasificación.

Es de suma importancia en esta etapa de desarrollo del proyecto, establecer una serie de herramientas y procedimientos para determinar el conjunto de características cuantificables y el conjunto de clases a la que pertenece cada característica.

El paso siguiente, es la búsqueda del clasificador y tratar de enfocar un buen algoritmo que nos permitan encontrar resultados satisfactorios. Las etapas son [19]:

- Elección de un modelo eficiente.
- Aprendizaje o entrenamiento del clasificador.
- Verificación de resultados.

## *3.3 INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL*

Uno de los signos más característicos del progreso de la humanidad, ha sido el diseño de máquinas con que el ser humano ha tratado de suplir y atenuar el esfuerzo físico, allí donde fuera necesario.

Los investigadores Warren McCulloch y Walter Pitts propusieron en 1943, la primera red neuronal artificial, naciendo así una de las tareas que con el tiempo se convertirían en una de las más características y prometedoras de la Inteligencia Artificial. En los años 60, se consigue el primer programa de IA. El programa llamado *Logic Theorist* y desarrollado por Newell, Shaw y Simon, era capaz de demostrar teoremas en el área de la lógica proporcional [24].

Entre otros principios generales, los científicos pioneros en IA están de acuerdo en que uno de los objetivos principales, debe ser el diseño de programas de ordenador que exhiban una de las cualidades más características de la inteligencia humana y animal: *la facultad de aprendizaje.*

Aunque los sistemas expertos y las redes neuronales artificiales, representan sin lugar a dudas los prototipos de sistemas inteligentes más populares de la IA, otras técnicas y metodologías como la ***lógica difusa, lógica borrosa o fuzzy Logic,*** merecen ser mencionados dentro y fuera de la IA [24].

Aunque en la actualidad, hay descritas un número ingente de reglas de *aprendizaje,* éstas pueden ser clasificadas en dos grupos o categorías, atendiendo

a la presencia o no de un agente externo o supervisor, que guie a la red neuronal durante el proceso de aprendizaje

### *3.3.1 Aprendizaje supervisado*

Es cuando dicho elemento supervisor está presente durante el proceso de aprendizaje. Se caracteriza por que la regla de aprendizaje, incorpora un agente externo o supervisor que evalúa a la red neuronal, enseñándole cómo debería corregir su comportamiento dinámico durante el aprendizaje y requiere de una fase de entrenamiento [24].

### *3.3.2 Aprendizaje no supervisado*

Es cuando el aprendizaje tiene lugar en ausencia de un elemento supervisor. Se basa en la auto-organización de la información contenida en los datos y por tanto, en los vectores de entrada, detectando la red neuronal sus propiedades colectivas, hecho que es reflejado con posterioridad en la salida generada por la red. Esta clase se inspira en la capacidad del cerebro humano, de aprender sin necesidad de que un sujeto está presente dirigiendo el aprendizaje [24].

## *3.4 MÁQUINA DE SOPORTE VECTORIAL*

### *3.4.1 Problema linealmente separable*

La solución a este problema QP es global y única. Para los datos empíricos, se dispone de N datos de entrenamiento llamados patrones, de la forma $(X_1,\ Y_1) \dots (X_n,\ Y_n)\ donde\ X \in R^n$. Cada escalar $Y_i$ (llamada Etiqueta) corresponderá a una de dos clases que se identificarán $\{-1, +1\}$. Se llamará vector de etiquetas al vector $\boldsymbol{Y} = (Y_1, Y_2, \dots, Y_n)$.

En un problema linealmente separable, existen muchos hiperplanos que pueden clasificar los datos. Pero las SVMs, no hayan uno cualquiera de estos hiperplanos, sino el único que maximiza la distancia entre él y el dato más cercano de cada clase (Figura 3.1). Esta distancia es llamada margen; y el hiperplano que la maximiza se llama *hiperplano máximo de margen* o de *Separación Optima* [25].

El hiperplano separador, está dado manera general por:

$$(\boldsymbol{W} \cdot \boldsymbol{X}) + b = 0 \; donde \; W, X \in R^n, b \in R \tag{3.2}$$

Donde el trabajo consiste en hallar el vector $\boldsymbol{W}$ de pesos, que contiene la ponderación de cada atributo, indicando qué tanto aportan en el proceso de clasificación; en tanto que $b$ define el umbral de decisión. La función discriminante (distancia) $\boldsymbol{d}$ será la función:

$$d(\boldsymbol{X}, \boldsymbol{W}, b) = \frac{|(\boldsymbol{W} \cdot \boldsymbol{X}) + b|}{\|\boldsymbol{W}\|} \quad con \; \|\boldsymbol{W}\| = \sqrt{(\boldsymbol{W} \cdot \boldsymbol{W})} \tag{3.3}$$

Donde $\|\boldsymbol{W}\|$ es la norma asociada al producto escalar en $\boldsymbol{R^n}$. Como se trata de patrones linealmente separables, se puede reescalar $\boldsymbol{W} \; y \; b$, de tal manera que:

$$d(\boldsymbol{X}, \boldsymbol{W}, b) = \frac{1}{\|\boldsymbol{W}\|} \; por \; lo \; tanto \quad |(\boldsymbol{W} \cdot \boldsymbol{X}) + b| = 1 \tag{3.4}$$

Así se obtiene el *hiperplano máximo de margen* canónico, en el cual los patrones de entrenamiento más cercanos al plano, tienen distancia normalizada $d(\boldsymbol{X}, \boldsymbol{W}, b) = 1 \; con \; d(\boldsymbol{X}, \boldsymbol{W}, b) \geq 1$ para los demás patrones.

Hallar el mejor hiperplano separador es un clásico problema de maximización con restricciones de la ecuación $(\mathbf{3.4})$**,** el cual el problema se puede convertir en un problema de optimización más sencillo, utilizando el principio de dualidad, quedando como:

$$\begin{aligned} &\mathbf{min} \quad \frac{\|W\|^2}{2}, \\ &\boldsymbol{sujeto\ a}: \; Yi\,[(W \cdot X) + b\,] \geq 1 \; para \; i = 1,2,\ldots,N \end{aligned} \tag{3.5}$$

Aplicando los multiplicadores de Lagrange $(\alpha i)$ la ecuación $(\mathbf{3.5})$ se formula como:

$$L(\boldsymbol{W}, b, \alpha) = \frac{\|\boldsymbol{W}\|^2}{2} - \sum_{i=1}^{N} \alpha i[yi[(\boldsymbol{W} \cdot \boldsymbol{X}) + b] - 1] \quad \textbf{(3.6)}$$

Para hallar el punto de silla $(\boldsymbol{Wo}, bo, \alpha o)$, se debe minimizar $L(\boldsymbol{W}, b, \alpha)$ con respecto a $\boldsymbol{W}$ $y$ $b$ y maximizar con respecto a $\alpha \geq 0$ , la cual representa una solución en el espacio primal. El problema puede solucionar en el espacio dual, para lo cual se replantea como:

$$\textbf{min} : L_d(\alpha) = \sum_{i=1}^{N} \alpha i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha i \alpha j \; yiyj \left(x_i^T \cdot x_j\right) \quad \textbf{(3.7)}$$

$$\boldsymbol{Sujeto\ a}: \quad \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \alpha iyi = 0 \quad y \quad \alpha i \geq 0, \;\; i \in \{1, \dots N\} \quad \textbf{(3.8)}$$

Donde $L_d$ es la forma dual de $L$ , la cual depende de los multiplicadores de Lagrange. Esta presentación es perfecta, ya que $Ld$ depende únicamente del producto escalar de los patrones de entrada $x$ lo cual implica sus cálculos.

Ésta formulación del problema, satisface las condiciones de *Karush-Kuhn-Tucker (KKT)* y por tanto se tiene las condiciones necesarias y suficientes, para que un valor extremo exista. Además, la solución siempre conduce al mismo vector normal al hiperplano separador. Tomando la representación matricial de las ecuaciones $(\mathbf{3.7})$ $y$ $(\mathbf{3.8})$ se obtiene:

$$\begin{aligned} \boldsymbol{min}: \quad & L_d(\alpha) = \frac{1}{2}(\alpha)^T H\alpha - f^T \alpha \\ \boldsymbol{Sujeto\ a}: \quad & y^T \alpha = 0, \qquad \alpha \geq 0 \end{aligned} \quad \textbf{(3.9)}$$

Donde se utiliza el vector unitario $f = [1,1 \dots 1]^T$ y mediante cualquier método de optimización, se halla el vector de multiplicadores $\alpha_0 = (\alpha_1^0, \alpha_2^0, \dots, \alpha_N^0)$, para determinar finalmente, el vector normal $W_0$ y el $b_0$ del *hiperplano Máximo de Margen.*

$$\mathbf{W_0} = \sum_{i=1}^{N} y_i \, \alpha_i^0 x_i \quad y \quad b_0 = -\frac{1}{2} W_0 \cdot [\, \boldsymbol{x_r} + \boldsymbol{x_s}] \tag{3.10}$$

Esto indica que $W_0$ se puede expresar como combinación lineal de los N vectores de entrada. De las combinaciones de los KKT se desprende que gran parte de los $\alpha_i^0$ son 0; y solo una menor cantidad $N_{sv}$ de vectores del conjunto de entrada, participan en la combinación lineal que originan a $W_0$: $son$ $\boldsymbol{los\ vectores\ de\ soporte}$ $(SV)$. En la ecuación $(\mathbf{3.10})$ $\boldsymbol{x_r}$ y $\boldsymbol{x_s}$ son un par de vectores de soporte, uno de cada clase [25].

El clasificador buscado finalmente, se puede escribir como:

$$f(x) = Sign(W_0 \cdot x \ + \ b_0 \,) = Sign\left( \sum_{i \in SV} y_i \, x_i^0 (x_i \, x) \ + \ b_0 \right) \tag{3.11}$$

Donde el signo resultante, indica a qué clase pertenece un dato determinado. Esta expresión lleva gran economía computacional, ya que la sumatoria no se realiza sobre todos los puntos de entrenamiento, únicamente los que son vectores de soporte y el número $N_{sv}$ de $SV$ puede ser muy pequeño, siendo en general mucho menor que $N$ [25].

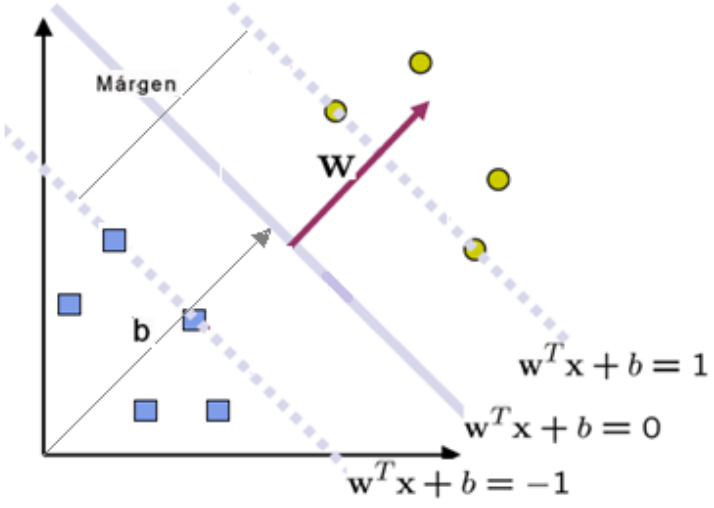



***Figura 3.1.*** *Hiperplano de separación óptimo* $(\boldsymbol{W^T} \cdot \boldsymbol{X}) + b = 0$ *para el caso bidimensional*

### *3.4.2 Problema linealmente no separable*

Si el conjunto $S = (X1,\ Y1)\ \ldots (Xn,\ Yn)$ no es linealmente separable, violaciones a la clasificación deben ser permitidas en la formulación de la SVM.
Para tratar con datos que no son linealmente separables, el análisis previo puede ser generalizado introduciendo algunas variables no-negativas $\xi_i \geq 0$ de tal modo que $(\boldsymbol{W} \cdot \boldsymbol{X}) + b \geq 1$ es modificado a

$$\boldsymbol{Yi}\big((\boldsymbol{W} \cdot \boldsymbol{X_i}) + b\big) \geq 1 - \xi_i \qquad donde \quad i = 1,2, \ldots, N \tag{3.12}$$

Recordemos que en la mayoría de los casos, la búsqueda de un hiperplano adecuado en un espacio de entrada, es demasiado restrictiva para ser de uso práctico. Una solución a esta situación, es mapear el espacio de entrada en un espacio de características de una dimensión mayor y buscar el hiperplano óptimo allí. Sea $Z = \varphi(x)$, la notación del correspondiente vector en el espacio de características con un mapeo $\varphi\ de\ R^n$ a un espacio de características $Z$.

Donde $\boldsymbol{W} \in Z$ y $b \in R$. Más precisamente, el conjunto *S* se dice que es linealmente separable si existe (***W***, *b*) tal que las inecuaciones

$$\begin{cases} \big((\boldsymbol{W} \cdot \boldsymbol{Z_i}) + b\big) \geq 1, & Y_i = 1 \\ \big((\boldsymbol{W} \cdot \boldsymbol{Z_i}) + b\big) \leq 1 & Y_i = -1 \end{cases} \tag{3.13}$$

Los $\xi_i \neq 0$ en $(\boldsymbol{3.12})$ son aquellos para los cuales el punto $x_i$ no satisface $(\boldsymbol{3.13})$**.** Entonces el término $\sum_{i=1}^{N} \xi_i$ puede ser tomado como algún tipo de medida del error en la clasificación.

El problema del hiperplano óptimo es entonces redefinido como la solución al problema

$$\begin{gathered} \boldsymbol{min}\ \left\{\frac{\boldsymbol{1}}{\boldsymbol{2}}\boldsymbol{w} \cdot \boldsymbol{w} + \boldsymbol{C}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \xi_i\right\} \\ \boldsymbol{Sujeto\ a}\colon \big((\boldsymbol{W} \cdot \boldsymbol{Z_i}) + b\big) \geq 1 - \xi_z \\ i = 1,2, \ldots . N \quad \xi_i \geq 0, \end{gathered} \tag{3.14}$$

Donde $\boldsymbol{C}$ es una constante. El parámetro $\boldsymbol{C}$ puede ser definido como un parámetro de regularización. Este es el único parámetro libre de ser ajustado en la formulación de la SVM. El ajuste de éste parámetro puede hacer un balance entre la maximización del margen y la violación a la clasificación [26].

Buscando el hiperplano óptimo en $(\mathbf{3.14})$ es un problema QP, que puede ser resuelto construyendo un Lagrangiano y transformándolo en el dual

$$\begin{aligned} \boldsymbol{Max}\text{:}\ \ & W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_J\, y_i y_j z_i \cdot z_j \\ \boldsymbol{Sujeto\ a}\text{:}\ \ & \sum_{i=1}^{N} \alpha_i y_i = 0, \quad 0 \le \alpha_i \le C, \quad i = 1, \dots, N \end{aligned} \tag{3.15}$$

Donde $\alpha = \alpha_1, \dots \alpha_N$ es una vector de multiplicadores de Lagrange positivos asociados con las constantes en $(\mathbf{3.12})$.
El teorema de Khun-Tucker juega un papel importante en la teoría de las SVM. De acuerdo a este teorema, la solución $\overline{\alpha_i}$ del problema $(\mathbf{3.15})$ satisface:

$$\overline{\alpha_i}\left(y_i\left(\overline{w}\cdot z_i + \bar{b}\right) - 1 + \bar{\xi}i\right) = 0, \quad i = 1, \dots, N \tag{3.16}$$

$$(C - \overline{\alpha_i})\xi_i = 0, \quad i = 1, \dots, N \tag{3.17}$$

De esta igualdad se deduce que los únicos valores $\overline{\alpha_i} \neq 0$ **(3.17)** son aquellos que para las constantes en **(3.12)** son satisfechas con el signo de igualdad. El punto $x_i$ corresponde con $\overline{\alpha_i} > 0$ es llamado vector de soporte. Pero hay dos tipos de vectores de soporte en un caso no separable. En el caso $0 \le \bar{\alpha}_i \le C$, el correspondiente vector de soporte $x_i$ satisface las igualdades $y_i\left(\overline{w}\cdot z_i + \bar{b}\right) = 1\ y\ \xi_i = 0$. En el caso $\overline{\alpha_i} = C$, el correspondiente $\xi_i$ es diferente de cero y el correspondiente vector de soporte no satisface. Nos referimos a estos vectores de soporte como errores. El punto $x_i$ correspondiente con $\overline{\alpha_i} = 0$ es clasificado correctamente y está claramente alejado del margen de decisión [26].

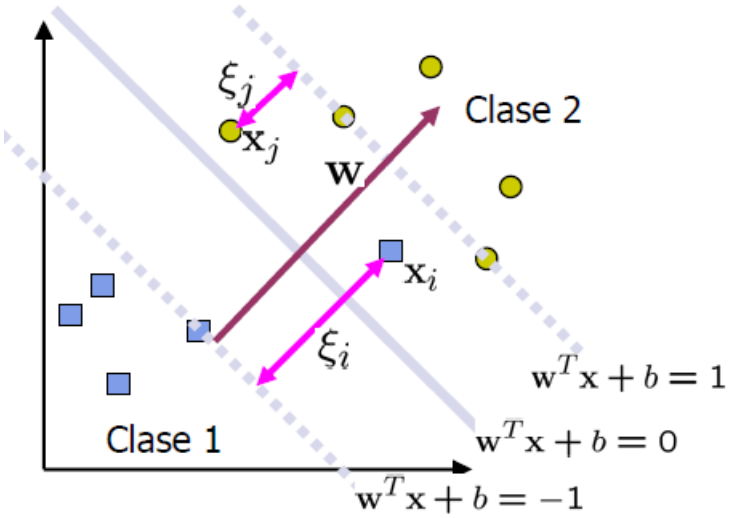



***Figura 3.2.*** *Aparición del parámetro de error* $\xi_i$ *es el error de clasificación* [26]

Para construir el hiperplano óptimo $\bar{w} \cdot z + \bar{b}$, se utiliza

$$\bar{w} = \sum_{i=1}^{N} \overline{\alpha_i}\, y_i z_i \qquad \textbf{(3.18)}$$

Y el escalar *b* puede ser determinado de las condiciones de Kuhn-Tucker

Recordemos que

$$f(x) = sign(w \cdot z_i + b) = \begin{cases} 1 & y_i = 1 \\ -1 & y_i = -1 \end{cases} \qquad \textbf{(3.19)}$$

Podremos establecer la relación de la ecuación $(\mathbf{3.11})$ en su forma generalizada, encontrando un hiperplano no lineal como solución óptima.

### *3.4.3 Truco del Kernel para el caso no linealmente separable*

Cuando no existe una apropiada superficie lineal de decisión en el espacio de entrada, se considera el mapeo del vector de entrada $x$ en un espacio de mayor dimensión $R^c$ llamado ***espacio de características*** $\boldsymbol{\tau}$, que está dotado de producto escalar. Eligiendo el $\boldsymbol{\tau}$ apropiado, se utiliza el mapeo y se busca el hiperplano óptimo de separación siguiendo la mecánica expuesta anteriormente y que será lineal en $R^c$, pero representa un hiperplano no lineal en el espacio de entrada $R^n$ [25].

Supóngase que este mapeo se realiza mediante una función no lineal de la forma

$$\phi(x) \;:\; R^n \longmapsto R^c, \qquad c > n$$
$$\phi(x) \;:\; R^n \longmapsto \phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_c(x))$$

Se puede observar una dificultad práctica para hallar el hiperplano óptimo de separación en $\boldsymbol{\tau}$, ya que se debe solucionar el problema planteado en la ecuación $(\mathbf{3.7})$, replanteada ahora como la minimización de $L_d(\alpha) = \sum_{i=1}^{N} \alpha i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha i \alpha j \, yiyj \left(\phi(x_i) \cdot \phi(x_j)\right)$ donde los vectores de entrenamiento, sólo aparecen en la forma de producto escalar, pero definido en el espacio de características. El cálculo de $\phi(x_i) \cdot \phi(x_j)$ es demasiado exigente computacionalmente, ya que C es mucho más grande que $n$. La solución de éste grave inconveniente viene de la mano de las funciones del kernel [25].

Una función Kernel es una función es una función $K : R^n \; x \; R^n \longmapsto R \; tal \; que$

$$K : (x_i, x_j) \longmapsto \phi(x_i) \cdot \phi(x_j) \qquad \textbf{(3.20)}$$

Que representa el producto punto en un espacio de características de dimensión arbitraria. Al utilizar la función Kernel, se puede obtener el producto escalar en el espacio de características, pero realizando el cálculo en el espacio de entrada cuya complejidad es mucho menor. Más aún, con el método del kernel no es necesario conocer explícitamente la función $\phi$ [25].

A pesar de los nuevos núcleos que están siendo propuestos por investigadores, los principiantes en dicho campo, podrán encontrar en los libros de SVMs los siguientes cuatro núcleos básicos [20]:

- $Linear: \quad K(x_i, x_j) = x_i^T x_j$
- $Polynomial: \quad K(x_i, x_j) = (\gamma x_i^T x_j + r)^d \quad \gamma > 0.$
- $Radial\ Basis\ Function\ (RBF): \quad K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \quad \gamma > 0.$
- $Sigmoid: \quad K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Donde $\gamma, r\ y\ d$ son parámetros del Kernel.

## *3.5 MÉTODOS MULTICLASE PARA MÁQUINAS DE SOPORTE VECTORIAL (SVMs)*

Las SVMs fueron originalmente creadas para clasificación binaria. En la actualidad, las investigaciones se orientan a problemas multiclase, principalmente en dos tipos de enfoques para SVMs. Uno, es mediante la construcción y la combinación de varios clasificadores binarios; mientras que el otro método, es considerando todos los datos en una formulación de optimización.

La formulación para resolver problemas multiclase SVM en un solo paso, tiene variables proporcionales al número de clases. Por lo tanto, los métodos multiclase SVM, ya sea de varios clasificadores binarios, tiene que ser construidos a un problema de optimización más grande. Por consiguiente, es computacionalmente más costoso para resolver un problema multiclase, que un problema binario con el mismo número de datos. Hasta ahora, experimentos se limitan a pequeños grupos de datos [27].

A continuación, mostraremos dos métodos basados en clasificación binaria:

- "one-against-all" (Uno vs todos)
- "one-against-one" (Uno vs uno)

### *3.5.1 "one-against-all"*

La primera implementación para la clasificación multiclase SVM, es probablemente el método "one-against-all". Se construyen $K$ modelos de SVM donde $K$ es el número de clases. La $i - SVM$ es entrenada con todos los ejemplos de la clase $i$, con etiquetas positivas y todos los demás ejemplos con etiquetas negativas [27].

Dado $l$ datos de entrenamiento:

$(x_1,\ y_1), \dots, (x_l,\ y_l),\ donde\ x_i \in R^n \quad i = 1,2, \dots, l \quad y \quad y_i \in \{1, \dots, k\}\ es\ la\ clase\ de\ x_i$

La $i - SVM$ resuelve el siguiente problema:

$$\begin{gathered} \underset{w^i, b^i, \xi^i}{min} \quad \frac{1}{2}(w^i)^T w^i + C\sum_{j=1}^{l} \xi_j^i\,(w^i)^T \\ (w^i)^T \phi(x_j) + b^i \geq 1 - \xi_j^i, \quad si \quad y_j = i \\ (w^i)^T \phi(x_j) + b^i \leq -1 + \xi_j^i, \quad si \quad y_j \neq i \\ \xi_j^i \geq 0, \quad j = \{1, \dots, l\} \end{gathered} \tag{3.21}$$

Donde los datos de entrenamiento $x_i$ son mapeados en un espacio de dimensión mayor por medio de la función $\phi$ y $C$, como parámetro de penalización.

Después de resolver $(\mathbf{3.20})$, hay $K$ funciones de decisión:

$$\begin{gathered} (w^1)^T \phi(x) + b^1 \\ \vdots \\ (w^k)^T \phi(x) + b^k. \end{gathered}$$

Se dice que $x$ pertenece a la clase donde la función tiene el valor mayor de decisión.

$$class\ of\ x = \underset{i\ =\ 1, \dots, k}{argmax} \left( \left(w^i\right)^T \phi(x) + b^i \right) \tag{3.22}$$

Prácticamente, se resuelve el problema dual de $(\mathbf{3.21})$ cuyo número de variables es el mismo que el número de datos en $(\mathbf{3.21})$. Por lo tanto, $Kl - variables$ son resueltas por medio de programación cuadrática.

### 3.5.2 “one-against-one”

Estos clasificadores entrenan cada uno de los datos de dos clases. Para los datos de entrenamiento de $i\ y\ j\ clases$, resolveremos el siguiente problema binario de clasificación [27]:

$$\begin{aligned} \min_{w^{ij}, b^{ij}, \xi^{ij}} \quad & \frac{1}{2}(w^{ij})^T w^{ij} + C\sum_t \xi_t^{ij}\,(w^{ij})^T \\ & (w^{ij})^T \phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij}, \quad si \quad y_t = i \\ & \left(w^{ij}\right)^T \phi(x_t) + b^{ij} \leq -1 + \xi_t^{ij}, \quad si \quad y_t = j \\ & \xi_t^{ij} \geq 0, \end{aligned} \tag{3.23}$$

En la clasificación, se utiliza una estrategia de votación: cada clasificación binaria es considerada para ser una votación donde los votos, pueden ser emitidos por todos los puntos de datos $x$. Al final, un punto es designado para estar en una clase con el máximo número de votos [20].

Si $sign\left(\left(w^{ij}\right)^T \phi(x) + b^{ij}\right)$ dice que $x$ está en la clase $i$, entonces el voto a favor de la clase $i$ se añade en uno. De lo contrario, la $clase\ j$ se incrementa en uno. Entonces podemos predecir que $x$ está en la clase con el mayor número de votos. El enfoque de votación descrito anteriormente, también se llama estrategia "MaxWins". En caso de que dos clases tienen votos iguales, pensamos que puede no ser una buena estrategia, ya que sólo tiene que seleccionar el que tiene el índice más pequeño [27].

Se resuelve el problema dual $(\mathbf{3.23})$**,** cuyo número de variables es el mismo que el número de datos de dos clases.

## 3.6 SEQUENTIAL MINIMAL OPTIMIZATION (SMO)

Es un algoritmo simple que rápidamente puede resolver problemas de programación cuadrática (QP) para SVMs. SMO descompone el problema QP global a sub-problemas QP, usando el teorema de Osuna para asegurar la convergencia, eligiendo resolver el problema de optimización más pequeño posible en cada paso.

En 1997, Osuna, probó un teorema que sugiere un nuevo conjunto de algoritmos QP para SVMs. El teorema demuestra, que el problema QP grande, puede ser dividido en una serie de pequeños QP sub-problemas [28].

La ventaja de SMO reside en el hecho de que la solución durante dos multiplicadores de Lagrange, se puede hacer analíticamente. Por lo tanto, la optimización numérica QP se evita completamente. A pesar de que más sub-problemas de optimización, se resuelven en el desempeño del algoritmo; cada sub-problema es tan rápido, que el problema QP general, se resuelve rápidamente.

Además, SMO no requiere de un almacenamiento matricial adicional en absoluto. Así, los problemas de formación SVM muy grandes, pueden caber dentro de la memoria de un ordenador personal. Debido a que no se utilizan algoritmos matriciales en SMO, que es menos susceptible a los problemas de precisión numérica [29].

Hay dos componentes para SMO: un método analítico para la solución de los dos multiplicadores de Lagrange, y una heurística para elegir qué multiplicadores optimizar.

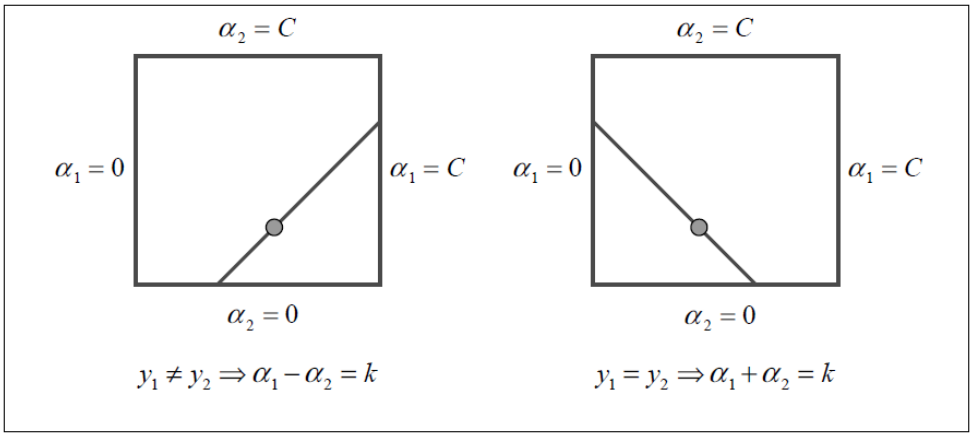



***Figura 3.3.*** Restricciones de desigualdad que causan los multiplicadores de Lagrange. (Fuente: J. C. Platt, 1998, p. 6, adaptado)

### *3.6.1 Resolviendo para dos multiplicadores de Lagrange.*

Con el fin de resolver para los dos multiplicadores de Lagrange, SMO primero calcula las limitaciones de estos multiplicadores y luego resuelve para el mínimo restringido. Por comodidad, todas las cantidades que se refieren al primer multiplicador, tendrá un subíndice 1; mientras que todas las cantidades que se refieren al segundo multiplicador, tendrá un subíndice 2. Debido a que sólo hay dos multiplicadores, las restricciones pueden ser fácilmente mostradas en dos dimensiones. Las restricciones unidas hacen que los multiplicadores de Lagrange se encuentren dentro de un cuadro; mientras que la restricción de igualdad lineal hace que los multiplicadores de Lagrange, se acuesten sobre una línea diagonal.

Así, el mínimo restringido de la función objetivo, debe situarse sobre un segmento de línea diagonal como se muestra en la figura. Esta limitación explica por qué dos, es el número mínimo de los multiplicadores de Lagrange que pueden ser optimizados: si SMO es optimizado solamente por un multiplicador, no se podría cumplir la restricción de igualdad lineal en cada paso [29].

Los extremos del segmento de línea diagonal puede expresarse de manera simple. Sin perder la generalidad, el algoritmo calcula en primer lugar el segundo multiplicador de Lagrange $\alpha_2$ y calcula los extremos del segmento de la línea diagonal en términos de $\alpha_2$. Si el objetivo $y_1$ no es igual al objetivo $y_2$, entonces los límites siguientes se aplican a $\alpha_2$:

$$L = \max(0, \alpha_2 - \alpha_1), \qquad H = \min(C, C + \alpha_2 - \alpha_1) \tag{3.24}$$

Si el objetivo $y_1$ igual al objetivo $y_2$, a continuación, los siguientes límites se aplican a $\alpha_2$:

$$L = \max(0, \alpha_2 + \alpha_1 - C), \qquad H = \min(C, \alpha_2 + \alpha_1) \tag{3.25}$$

La segunda derivada de la función objetivo a lo largo de la línea diagonal se puede expresar como:

$$\eta = k(\vec{x}_1, \vec{x}_1) + k(\vec{x}_2, \vec{x}_2) + 2k(\vec{x}_1, \vec{x}_2) \tag{3.26}$$

En circunstancias normales, la función objetivo será definida positiva; habrá un mínimo a lo largo de la dirección de la restricción de igualdad lineal, y $\eta$ es mayor que cero. En este caso, SMO calcula el mínimo a lo largo de la dirección de la restricción:

$$\alpha_2^{new} = \alpha_2 + \frac{y_2(E_1 - E_2)}{\eta} \tag{3.27}$$

Donde $E_i = u_i - y_i$ es el error sobre $i - esimo$ ejemplo entrenado. Como paso siguiente, el mínimo se encuentra restringido por el mínimo, sin restricciones a los extremos del segmento de línea:

$$\alpha_2^{new,clipped} \begin{cases} H & si \quad \alpha_2^{new} \geq H \\ \alpha_2^{new} & si \quad L < \alpha_2^{new} < H \\ L & si \quad \alpha_2^{new} \leq L \end{cases} \tag{3.28}$$

Ahora, $S = y_1\, y_2$. El valor de $\alpha_1$ se calcula a partir del nuevo $\alpha_2$.

$$\alpha_1^{new} = \alpha_1 + s\left(\alpha_2 - \alpha_2^{new,clipped}\right) \tag{3.29}$$

### *3.6.2 Heurística para elegir qué multiplicadores optimizar*

Con el fin de acelerar la convergencia, SMO utiliza heurística que selecciona los dos multiplicadores de Lagrange para optimizar conjuntamente.

Hay dos elecciones heurísticas separadas: La elección de la primera heurística, proporciona el lazo externo del algoritmo SMO. El lazo exterior, primero se repite en el conjunto de entrenamiento y busca determinar si cada ejemplo viola las condiciones KKT. Si un ejemplo viola las condiciones KKT, entonces es elegible para la optimización. Después de un ciclo a través del conjunto de capacitación, el bucle externo se repite en todos los multiplicadores de Lagrange cuyos ejemplos no son C ni 0. Después de que el conjunto de capacitación obedece a las condiciones KKT, el algoritmo termina.

El segundo multiplicador de Lagrange, se elige de modo que el tamaño del paso dado durante la optimización conjunta se maximiza, lo que resulta en un gran aumento del objetivo dual [29].

## 3.7 ESTIMACIÓN DE PROBABILIDADES EN SVMs

Dependiendo de la solución multiclase que se adopte, las SVMs binarias pueden comparar dos clases entre sí o bien, una de ellas con todas las demás. No obstante, por lo general, el proceso de obtención de probabilidades a posteriori a

partir de las salidas de las SVMs consta en ambos casos de dos pasos: 1) obtener la probabilidad de que la muestra pertenezca a cada clase en todas las SVMs binarias, y 2) transformar estas probabilidades binarias a probabilidades multiclase. En la aproximación 1-contrael resto, este último paso puede reducirse a una simple normalización para que la suma de las probabilidades a posteriori sea uno, ya que el número de SVMs binarias coincide con el número de clases [39].

La forma más comúnmente empleada para transformar la salida de una SVM en probabilidades binarias, consiste en el uso de una función sigmoide, donde se propone ajustar mediante sendas gaussianas, las funciones de densidad de probabilidad condicional de la salida de la SVM $(p(f \mid y = +1))$ y $(p(f \mid y = -1))$. Aplicando la regla de Bayes:

$$P(1 \mid f) = \frac{p(f \mid y = 1)\, P(y = 1)}{\sum_{i=\pm 1} p(f \mid y = i)\, P(y = i)} \qquad \textbf{(3.30)}$$

y sustituyendo dichas expresiones para las funciones de densidad de probabilidad condicional se llega a:

$$P(y = 1 \mid f) = \frac{1}{1 + \exp(af^2 + bf + c)} \qquad \textbf{(3.31)}$$

Para simplificar esta función y evitar que no sea monótona, se asume que las gaussianas mencionadas están centradas en los márgenes (±1) y tienen la misma varianza, la cual debe estimarse. En este caso, la expresión de la probabilidad a posteriori se simplifica en una sigmoide, cuya pendiente en la zona lineal se calcula a partir de la varianza de las gaussianas [39]. El sesgo se calcula de forma que $P(1 \mid f) = 0.5$ en $f = 0$.
Basándose en este trabajo y asumiendo que en la zona comprendida entre los márgenes las funciones de densidad de probabilidad condicional son aproximadamente exponenciales, Platt propone un modelo paramétrico para la probabilidad binaria a posteriori.

$$P(y = 1 \mid f) = \frac{1}{1 + \exp(Af + B)} \quad \textbf{(3.32)}$$

La diferencia respecto al trabajo anterior, consiste en que los parámetros $A \ y \ B$ se estiman de manera discriminativa, maximizando la verosimilitud. Esta expresión proporciona directamente probabilidades multiclase en el caso 1-contra-el resto. Si se tienen k clases distintas, la probabilidad a posteriori de la clase i-ésima se puede obtener como [39]:

$$P(y = i \mid \mathbf{x}) = \frac{1}{1 + \exp(A_i f_i(\mathbf{x}) + B_i)} \quad \textbf{(3.33)}$$

Siendo $f_i(\mathbf{x})$ la salida de la SVM binaria que clasifica la clase i contra el resto. En este caso, no se garantiza que la suma de las probabilidades sea uno; una forma de obtener probabilidades a posteriori normalizadas consiste en usar la función softmax:

$$P(y = i \mid \mathbf{x}) = \frac{\exp(\gamma f_i(\mathbf{x}))}{\sum_{j=1}^{k} \exp(\gamma f_j(\mathbf{x}))} \quad \textbf{(3.34)}$$

Donde el parámetro $\gamma$ se estima maximizando la verosimilitud.

En el caso 1-contra-1, en primer lugar se debe calcular la probabilidad de Platt para la muestra de entrada en cada SVM binaria $(i, j)$ , $\forall i, j \ \in \ \{1, \dots, k\}$. La probabilidad de Platt de que $\mathbf{x}$ pertenezca a la clase i en la SVM binaria $(i, j)$ se calcula como:

$$\gamma_{ij} = P(y = i \mid y = j \text{ ó } j, \mathbf{x}) = \frac{1}{1 + \exp(A_{ij} f_{ij}(\mathbf{x}) + B_{ij})}$$
$$\gamma_{ji} = P(y = j \mid y = i \text{ ó } j, \mathbf{x}) = 1 - \gamma_{ij} \quad \textbf{(3.35)}$$

Siendo $f_{ij}(\mathbf{x})$ la salida de la SVM binaria $(i, j)$.

A continuación, se emplea una modificación del método de Refregier-Vallet, para transformar estas probabilidades binarias $\gamma_{ij} \ \forall \ i, j$ en probabilidades multiclase.

$P(y = i \mid \mathbf{x}) \quad \forall\, i$. Se propone resolver un sistema lineal formado por k − 1 ecuaciones del tipo:

$$\gamma_{ji} P(y = i \mid \mathbf{x}) = \gamma_{ij} P(y = j \mid \mathbf{x}) \tag{3.36}$$

Junto con otra que fuerce que la suma de las probabilidades sea uno. Se señala que la solución que se obtiene, depende en gran medida de las ecuaciones seleccionadas, por lo que se propone como alternativa el siguiente problema de minimización, que considera todas las ecuaciones posibles [39]:

$$\min_{P} \quad \frac{1}{2} \sum_{i=1}^{k} \sum_{j:j\neq i}^{k} \left( \gamma_{ji} P(y = i \mid \mathbf{x}) - \gamma_{ij} P(y = j \mid \mathbf{x}) \right)^2$$

$$Sujeto\ a: \sum_{i=1}^{k} P(y = i \mid \mathbf{x}) = \mathbf{1} \tag{3.37}$$

$$P(y = i \mid \mathbf{x}) \geq 0 \ \forall\, i$$

Con $P(\mathbf{x}) = [P(y = 1 \mid \mathbf{x}), \ldots, P(y = k \mid \mathbf{x})]$.

# *CAPITULO 4 - DESARROLLO DEL MODELO*

## *4.1 PRUEBAS REALIZADAS*

### *4.1.1 Introducción*

En este capítulo, se implementa una metodología basada en la minería de datos y tiene como fin la detección de clientes "sospechosos" pertenecientes a un sistema eléctrico de distribución y cuya característica fundamental es el estudio de sus perfiles de carga.

Para obtener resultados satisfactorios es necesario que la metodología de búsqueda cumpla con dos etapas fundamentales:

- Pre-procesamiento de datos.
- Selección del modelo

En el *pre-procesamiento* se ilustran técnicas de minería de datos basada en la búsqueda y construcción de características, que determinan el comportamiento de los usuarios en un periodo determinado.

En la *selección del modelo* se analiza las etapas de entrenamiento con datos de prueba, selección de parámetros y etapas de validación, las cuales determinan un buen modelo de clasificación.

### *4.1.2 Marco metodológico*

A continuación, se propone un marco metodológico con el fin de desarrollar un sistema que permita la detección de clientes y que cumplan un perfil de carga determinado. La metodología correspondiente se ilustrará en la ***Figura 4.1.***

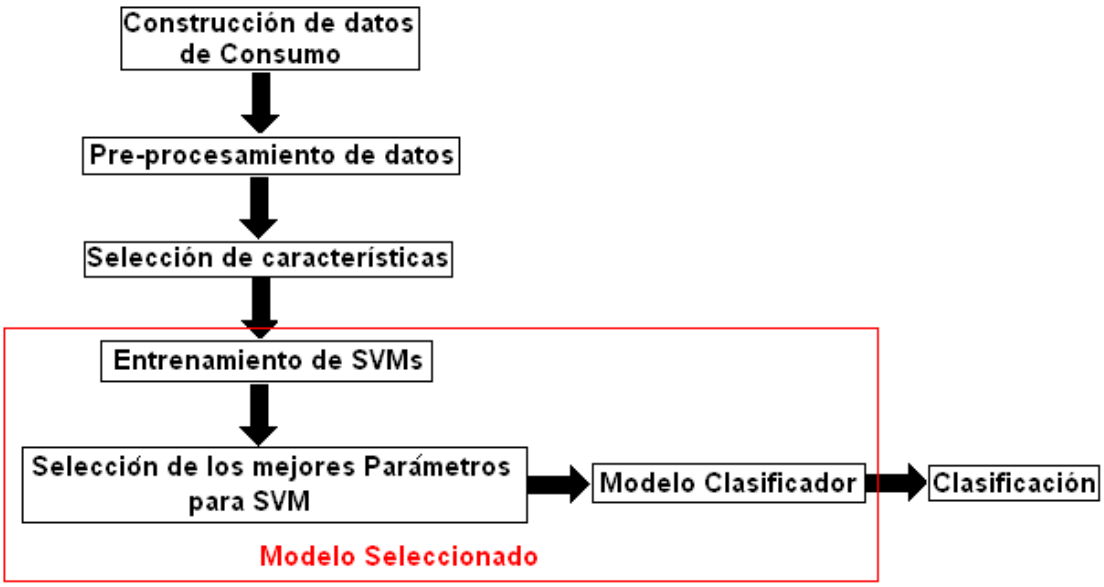



***Figura 4.1.*** Diagrama propuesto para la clasificación de clientes con probabilidad de fraude

Para el diagrama propuesto en la ***Figura 4.1,*** se utilizaron datos de consumo construidos para un período de 25 meses, los cuales brindan un panorama del comportamiento de la carga a lo largo del tiempo y permiten que estos comportamientos sean aprendidos por la máquina, para obtener óptimas clasificaciones a futuro.

### *4.1.3 Construcción de datos de consumo*

La construcción de datos de consumo se basa principalmente en crear perfiles de carga, los cuales cumplan todos los requerimientos y características que un perfil de carga real pueda generar. Consumos que van dentro de una franja a lo largo del tiempo, dependiendo del estrato social al que pertenece cada carga y cambios notoriamente significativos en el consumo de un usuario, son algunos parámetros importantes con los que cuentan los datos creados para este proyecto.

Para construir los perfiles de carga, fue necesario consultar modelos ya inspeccionados y estudiados [20], [15], [13] para un comportamiento determinado, sea normal o anormal y contribuir con otros perfiles de carga, que siendo normales, pueden llevar a un entendimiento erróneo del problema. La construcción de los perfiles de carga, están acompañados de aquellos valores característicos, en donde la variación de consumo de energía cambia según el estrato social. Estos datos fueron consultados en el *Sistema Único de Información* de servicios públicos, de la República de Colombia, donde muestran de manera detallada el consumo promedio (kW-h) por estrato durante el año [30]. Partiendo de este recurso técnico, se pueden establecer modelos de perfiles de carga muy ajustados

a la realidad y cuyos modelos de selección, sean muy aproximados y de bastante utilidad.

Teniendo en cuenta los parámetros de construcción de los perfiles de carga, se elaboraron cuatro perfiles diferentes para evaluar el modelo de clasificación. Se cuentan con cinco carpetas, donde cada una de ellas contiene los programas desarrollados en Matlab R2008a como se muestra en la ***figura 4.2***.

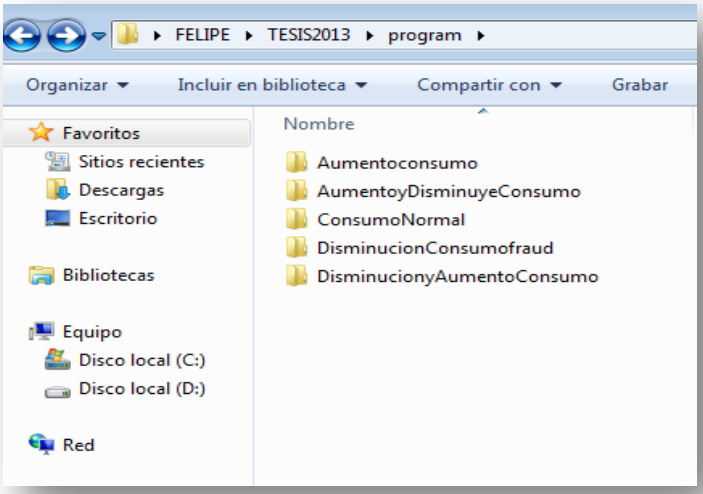



***Figura 4.2*** Carpetas que contienen los diferentes tipos de perfiles de carga

Cada carpeta contiene tres programas, los cuales cumplen una función específica. Se muestra el contenido de cada carpeta en la ***figura 4.3***.

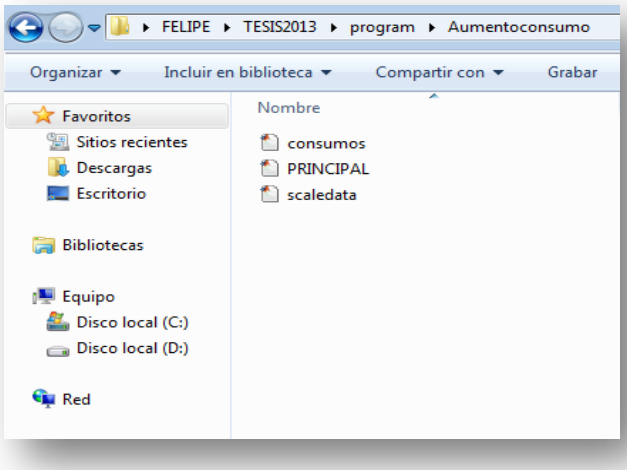



***Figura 4.3***. Programas para la construcción de perfiles de carga

A continuación, se explica el comportamiento que tiene cada uno de estos programas para la generación de datos. Es preciso recordar que cada carpeta representa un perfil de carga diferente y está compuesta por: ***consumos, PRINCIPAL*** y ***scaledata.***

- ***Consumos:*** Es una función que tiene como propósito fundamental, crear franjas de consumo en donde el perfil de carga del usuario, va a fluctuar entre valores máximos y mínimos de consumo. Dichos valores de consumo, fueron consultados en el Sistema Único de Información[2]. Para los estratos 1,2 y 3, la gráfica tendrá un comportamiento de +/- 5kW-h teniendo como base, un valor de consumo obtenido aleatoriamente dentro de los rangos permitidos por estrato. Para los estratos 4,5 y 6, el comportamiento de la gráfica será de +/- 5kW-h. En el siguiente seudocódigo, mostraremos la función correspondiente a la construcción de franjas de consumo en un período determinado.

- **Seudocódigo para la función *consumos*.**

**Datos:** meses, usuario,estrmin, estrmax, rangus, graph

Función ***consumos*** (meses, usuario,estrmin, estrmax, rangus, graph)

```
mes ← meses;
us ← usuarios;
ermin ← strmin + rangus;
ermax ← strmax – rangus;
fus ← rangus;
ploteo ← graph;

Para i ← 1 Hasta mes Con Paso 1 Hacer
        Para j ← 1 Hasta us Con Paso 1 Hacer
        u(j , i) ← Crear Matriz franjas de consumo
        Fin Para
Fin Para

Para k ← 1 Hasta us Con Paso 1 Hacer
        e1(k) ← Escalar datos de cada usuario a una franja en particular dado por cada uno
                de los u(j , i) datos de e1.
Fin Para
Para i ← 1 Hasta us Con Paso 1 Hacer
```

[2] La Superintendencia de Servicios Públicos tiene la responsabilidad de establecer, administrar, mantener y operar el sistema único de información para los servicios públicos, SUI, de conformidad con lo establecido en la Ley 689 de 2001.

u2(i , :) ← scaleData(u( i : ), e1(i) – fus, e1(i) + fus)

// Comentario: ***Crear una Matriz que contiene los datos escalados de consumo para una franja determinada.***

Fin Para

Si ploteo = = 1 entonces

Asignar color a cada cadena de caracteres

Para j ← 1 Hasta *us* Con Paso 1 Hacer

Plot(1: mes, u2(j : ), cstring(mod( j, 7) +1))

// Comentario: ***Trazar los perfiles de carga correspondiente al número de usuarios Asignados.***

Fin Para

Fin Si

datos ← u2

***Fin Función.***

- ***PRINCIPAL:*** Es el programa principal. En este programa fijamos valores como: ***meses, usuario, estrmin, estrmax, rangus, graph.***

- ***Meses:*** Es un número entero y determina el período de tiempo que tiene una carga en relación con su consumo de energía.
- ***Usuario:*** Es el número de usuarios requeridos para la construcción de su correspondiente perfil de carga.
- ***Estrmin:*** Es el mínimo valor de consumo (kW-h) que se presenta en un estrato determinado.
- ***Estrmax:*** Es el máximo valor de consumo (kW-h) que se presenta en un estrato determinado.
- ***Rangus:*** Este valor determina las franjas sobre un valor de consumo de energía, escogido de forma aleatoria entre ***estrmin y estrmax.*** Para los estratos 1,2 y 3 es +/- 5kW-h y para estratos 4,5 y 6 es +/- 10kW-h.
- ***Graph:*** Permite la generación gráfica de perfiles de carga de acuerdo a los parámetros vistos anteriormente.

- **Seudocódigo *PRINCIPAL*. Para usuarios con "consumos normales"**

INICIO

//Comentario: ***Ingresamos los valores requeridos para la construcción***
***De perfiles de carga y ejecución de la función "consumos".***

**Datos:** meses, usuario, estrmin, estrmax, rangus, graph

Función ***consumos*** (meses, usuario, estrmin, estrmax, rangus, graph)

uinicio ← datos

FINAL

- ***scaledata:*** Este programa, permite escalar los valores de una matriz a partir de un mínimo y un máximo especificado. Hablamos de mínimo y de máximo, a valores de consumo de energía en kW-h.

- **Seudocódigo para la función *scaleData***

Función ***scaleData*** (datain, minVal, maxVal)

u( i : ) ← datain
e1(i) – fus ← minVal
e1(i) + fus ← maxVal

// Comentario: *La función* ***scaledata***[3] *está contenida en el algoritmo de la función* ***consumos****. Por lo tanto, los valores mostrados en esta función se pueden encontrar en la función* ***consumos.*** *Realizaremos el algoritmo en su forma original.*

dataout ← datain – min(datain(:))
dataout ← (dataout/range(dataout(:))) * (maxVal – minVal)
dataout ← dataout + minVal

***Fin Función.***

Esta etapa basada en la *construcción de datos de consumo,* es la etapa más importante del proyecto, ya que nos permite encontrar los datos ajustados para la extracción y estudio de características. Si los datos son de buena calidad, libre de datos erróneos, el modelo de clasificación se comportará de tal forma, que las características esperadas, nos arrojen resultados muy positivos.

[3] Fuente: Programa escrito por: Aniruddha Kembhavi, julio 11, 2007

En la ***Figura 4.4,*** se mostrarán cinco perfiles de carga, cuyo comportamiento lo podemos denominar como "perfiles de carga normal", correspondientes a cinco usuarios pertenecientes a algún sistema eléctrico de distribución. Estos perfiles de carga, fueron construidos por los programas mostrados anteriormente.

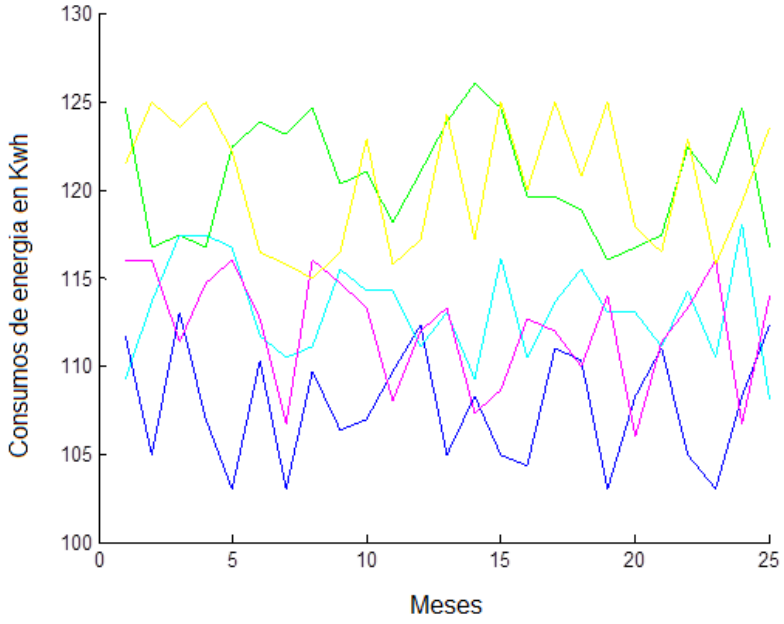



***Figura 4.4*** Perfiles de carga con un comportamiento "Normal"

En la ***Figura 4.5*** se muestran cinco perfiles de carga, cuyo comportamiento se puede denominar como "*perfiles de carga sospechosos*", correspondientes a cinco usuarios pertenecientes a algún sistema eléctrico de distribución. Estos perfiles de carga fueron construidos por medio de las funciones ***consumos, scaledata*** y el programa ***PRINCIPAL*** que se mostrará a continuación.

Cabe resaltar que el programa principal para "clientes normales" y el programa principal para "clientes anormales", tiene diferencias significativas en su planteamiento.

- **Seudocódigo *PRINCIPAL*. Para usuarios con "consumos sospechosos"**

INICIO
//Comentario: ***Ingresamos los valores requeridos para la construcción***
***De perfiles de carga y ejecución de la función "consumos".***

**Datos:** meses, usuario, estrmin, estrmax, rangus, graph.

Función ***consumos*** (meses, usuario, estrmin, estrmax, rangus, graph)
uinicio ← datos
Para i ← 1 Hasta *usuarios* Con Paso 1 Hacer
mesin( i ) ← round( 8+ (16-8) * rand)
//Comentario: ***Genera el mes aleatorio donde empieza el fraude***.
uinicio( i, [ mesin( i ) : size (uinicio, 2)]) ← 0
//Comentario: ***Valor de cero en los meses siguientes después del fraude***.

Fin Para

//Comentario: ***Ingresamos los valores requeridos para la construcción***
***De perfiles de carga para los meses siguientes después de***
***Ocurrido el fraude.***

**Datos:** estrmin, estrmax, graph.

Función ***consumos*** (meses, usuario, estrmin, estrmax, rangus, graph)

Para i ← 1 Hasta *usuarios* Con Paso 1 Hacer
uinicio( i, [ mesin( i ) : meses] ) ← datos( i, [ mesin( i ) : meses] )
//Comentario: ***Generación de consumos fraudulentos después del aleatorio***
***donde empieza el fraude***.
Fin Para
Si ploteo = = 1 entonces
Asignar color a cada cadena de caracteres
Para j ← 1 Hasta *us* Con Paso 1 Hacer
Plot(1: mes, u2(j : ), cstring(mod( j, 7) +1))
Fin Para
Fin Si
FINAL

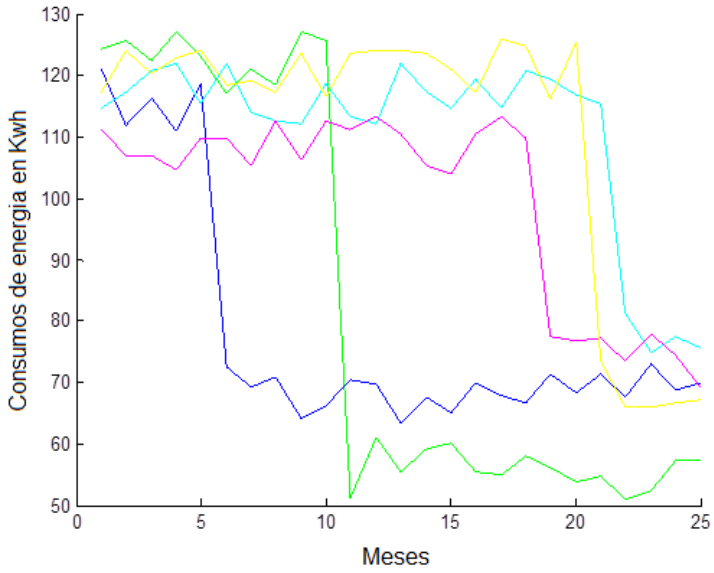



***Figura 4.5*** Perfiles de carga con un comportamiento "sospechoso"

La mayoría de las empresas comercializadoras de energía, se limitan sólo al método basado en ***críticas de consumos*** [37]; es decir, al análisis de las variaciones significativas contra el consumo histórico, generalmente en cuentas que han tenido una disminución del 20% o 30%. En este proyecto, este recurso es de vital importancia para la construcción de clientes sospechosos, como el mostrado en la ***Figura 4.5.***

### *4.1.3 Pre-procesamiento de datos*

En el mundo real, los datos tienden a ser ruidosos e incoherentes. Por lo tanto, para superar estos inconvenientes, las técnicas de minería de datos basada en métodos estadísticos, se pueden llevar a cabo. Para este proyecto, la base de pre-procesamiento consiste en la *normalización de los datos de consumo* y *métodos de estimación de precisión.*

- ***Normalización de datos.***

La normalización es importante antes de aplicar la SVM. La principal ventaja, es evitar los atributos en mayores rangos numéricos que dominan los de pequeño rango. Otra ventaja, es evitar dificultades numéricas durante la clasificación y del desempeño del kernel, escogido para un análisis determinado [1].

Se recomienda que la normalización de los datos, estén entre un rango de [0,1].

Los datos de consumo necesitan ser representados mediante una escala normalizada para el clasificador SVM. Por lo tanto, los datos de consumo se normalizaron de la siguiente manera [1]:

$$NL = \frac{L - \min(L)}{\max(L) - \min(L)} \quad \textbf{(4.1)}$$

Donde

$L$ : Representa el valor de consumo actual en kW-h del cliente.

$\min(L)$ : Representa el mínimo valor de consumo en kW-h del cliente.

$\max(L)$: Representa el máximo valor de consumo en kW-h del cliente.

- ***MÉTODO DE ESTIMACIÓN DE PRESICIÓN***

La estimación de precisión de un clasificador por algoritmos de aprendizaje supervisado, es importante, no sólo para predecir su precisión a futuro, sino también, la elección de un clasificador (modelo de selección) adecuado [31].

La precisión de un clasificador $\mathcal{C}$, es la probabilidad de clasificar correctamente una instancia seleccionada al azar; por ejemplo, $acc = \Pr(\mathcal{C}(v) = y)$ para una instancia seleccionada al azar $(v, y) \in \mathcal{X}$, donde $\mathcal{X}$ es el espacio de instancias etiquetadas, $v$ es el espacio de los casos no etiquetados y $y$ el conjunto de posibles etiquetas [31].

- ***CROSS-VALIDATION: En búsqueda de mejores Parámetros para el desempeño del modelo***

En $fc - fold$ *CROSS-VALIDATION,* a veces llamado "estimación de rotación", el conjunto de datos $v$, es aleatoriamente dividido en dos subgrupos mutuamente excluyentes (los $folds$) $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_h$ de tamaño aproximadamente igual [31].

Cross-validation se utiliza principalmente en entornos, cuyo objetivo es la predicción y la clasificación de datos; poder calcular la precisión de un modelo que se esté trabajando y encontrar parámetros de ajuste, para algún método seleccionado durante el entrenamiento.

Para nuestro caso de estudio, Cross-validation tiene como propósito fundamental, dividir los datos seleccionados para el entrenamiento del modelo en $fc - fold$, donde $fc$ representa el número de conjuntos que contienen aproximadamente el mismo número de datos de entrenamiento. Cabe recordar, que dichos datos están representados por perfiles de carga pertenecientes a un conjunto de usuarios determinado.

Antes de realizar cualquier procedimiento de búsqueda de parámetros, es importante enfatizar en el tipo de Kernel que se desea estudiar. Cada Kernel contiene parámetros diferentes; pero la filosofía de búsqueda para obtener los mejores parámetros, en la misma. Para cada instancia entre el cruce de cada uno de los conjuntos seleccionados, obtendremos una precisión dada en porcentaje. Este porcentaje de salida calculado por el modelo de búsqueda, está acompañado por los mejores parámetros según el kernel escogido.

Uno de los factores importantes para obtener los mejores parámetros, es la carga computacional que estos originan. Por tal motivo, estudiaremos hasta qué punto es necesario tener en cuenta este procedimiento, que en la mayoría de los casos es necesario.

### *4.1.4 Selección de características*

La selección de características se basa principalmente en la selección de datos, con el fin de extraer información útil y necesaria, para entrenar el modelo de clasificación. Como el parámetro para establecer la clasificación depende del perfil de carga, es muy importante que estos datos cumplan todas las características mencionadas y estudiadas anteriormente, para que el modelo escogido efectúe un buen desempeño en la clasificación de aquellos datos, en los que se necesita establecer alguna característica determinada.

En nuestro caso de estudio, buscamos establecer el mejor desempeño de la máquina de soporte, teniendo en cuenta un aumento secuencial de los datos para un solo estrato. Esto se realiza, con el fin de obtener alguna idea en el comportamiento de su procesamiento para obtener un modelo clasificatorio, que contenga los diferentes perfiles de carga para cada estrato. En este sentido, se busca realizar cuatro pruebas diferentes 1.000, 10.000, 100.000 y 600.000 datos para un solo estrato. El modelo clasificatorio, estará estructurado con el mayor número de perfiles de carga conforme sea el desempeño de las pruebas anteriores. Cada etapa de los datos, es dividida en 70% para datos de

entrenamiento y 30% para datos de validación. El propósito de dicha partición, es establecer la precisión del modelo a medida que aumenta el número de datos.

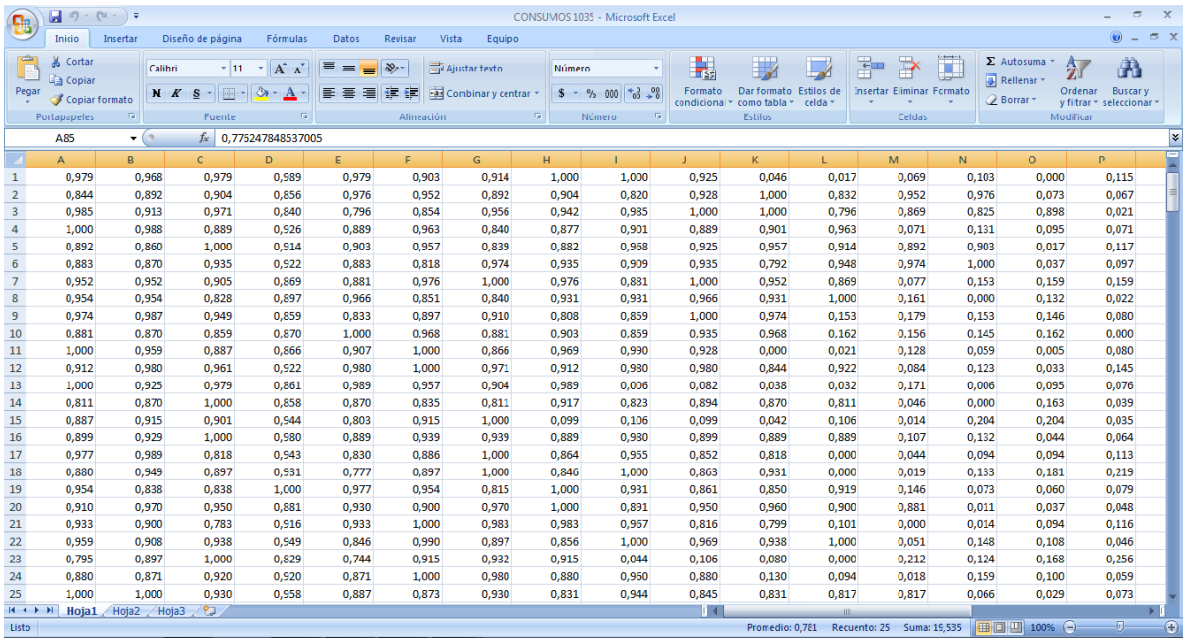



***Figura 4.6.*** 1035 usuarios seleccionados, con sus respectivos consumos normalizados para la etapa de entrenamiento

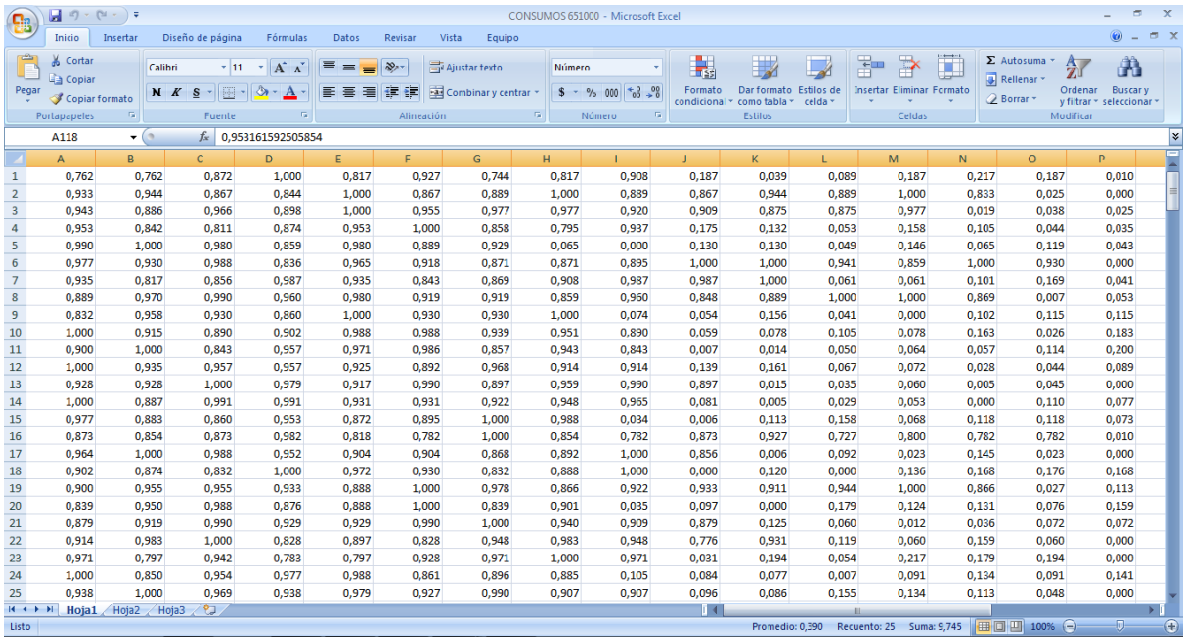
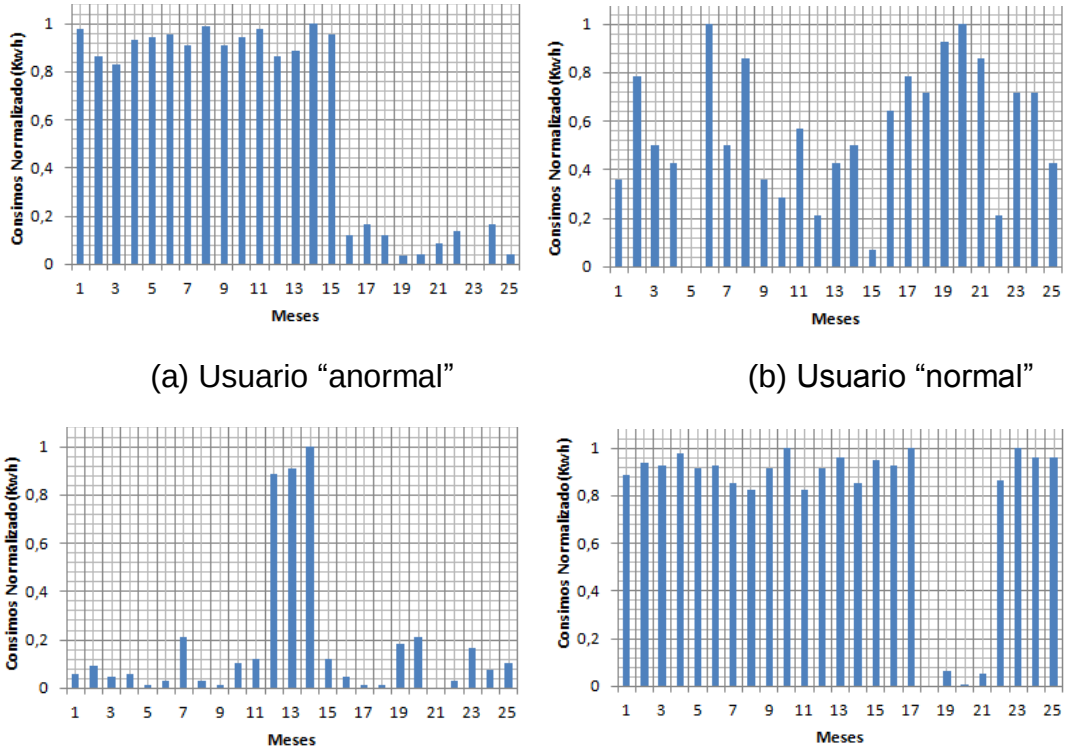



***Figura 4.7.*** 651.000 usuarios seleccionados, con sus respectivos consumos normalizados para la etapa de entrenamiento

Como se muestra en la ***Figura 4.6*** y ***Figura 4.7.*** Las etapas para 10.000 y 100.000 datos, también cuentan con tablas similares de datos de consumo normalizados, listos para el entrenamiento del modelo.

### *4.1.6 Selección de perfil de carga*

Con el propósito de construir el modelo de clasificación, es necesario establecer las características para cada perfil de carga y adecuarlo a una etiqueta determinada. El clasificador estudia 2-clases diferentes de perfiles de carga, donde la clase 1 está determinada por clientes "normales" y la clase 2 está determinada por clientes "anormales o sospechosos", donde su perfil de carga, muestra "cambios bruscos" de consumo durante intervalos de tiempo bastante prolongados. El hecho de que existan cambios bruscos de consumo en intervalos de tiempo pequeños, no significa que el usuario sea fraudulento. Este fenómeno puede ser el caso en que dicho usuario, no estuvo en su propiedad durante este tiempo. Por tal motivo, el consumo tiende a disminuir drásticamente respecto a su consumo normal. A continuación se muestran los perfiles de carga que serán estudiados en este proyecto.




(a) Usuario "anormal" (b) Usuario "normal"

(c) Usuario "normal" (d) Usuario "normal"

***Figura 4.8*** *Cuatro perfiles de carga en un periodo de 25 meses*

### *4.1.7 Entrenamiento del modelo (SVMs)*

El punto de partida de este proyecto se fundamenta en la utilización de algunas funciones contenidas en el toolbox Bioinformatic de Matlab R2008a, para la etapa de entrenamiento. Este toolbox contiene algunas funciones definidas para la aplicación de las SVMs en el proceso de clasificación. Se recuerda que las pruebas de entrenamiento tendrán dos fases: En la primera fase, construiremos modelos clasificatorios correspondientes a una cantidad determinada de perfiles de carga para un solo estrato. Según el desempeño del modelo obtenido, basado en la precisión y cantidad de datos, se procederá a la construcción del modelo final, el cual estará determinado por la totalidad de perfiles de carga para todos los estratos, siendo esta la segunda fase.

Para validar los resultados obtenidos en los modelos anteriores, fue necesario implementar LIBSVM 3.14 [29], el cual brinda un panorama más claro de los clasificadores que se implementar. El equipo utilizado para entrenar el modelo, es marca Samsung, con un sistema operativo Windows 7, Procesador Intel Pentium Inside, 2 GHz con 2 GB de RAM.

El funcionamiento de LIBSVM 3.14 requiere de la instalación de Microsoft Visual Studio 2012, ya que se requiere el empleo de funciones MEX, que permiten que una función en C sea llamada desde el entorno de Matlab.

Para trabajar con ficheros MEX, primero hay que seleccionar el compilador que se desea a utilizar.

A continuación, se muestra la estructura utilizada para obtener el "modelo" que contiene toda la información necesaria para la prueba de entrenamiento.

```
model= svmtrain (trainingtipconsu,trainingconsu,'-c 1  -s 0 -t 2 -b 1');
```

***Figura 4.9*** Función contenida en LIBSVM 3.14 para entrenamiento

En la ***Figura 4.9*** se pueden observar algunos parámetros necesarios para la etapa de entrenamiento, con valores por defecto, recordando que esta función internamente resuelve un problema de programación cuadrática. Este problema es resuelto por el método Sequential Minimal Optimization (SMO) [29], estudiado en capítulos anteriores. El vector ***trainingtipconsu***, representa las etiquetas o clases a la que pertenece cada perfil de carga de los datos de entrenamiento.

Contiene los valores 1 ó 2 dado el caso; ya que es un clasificador 2-clases. ***trainingconsu*** es una matriz que contiene consumos normalizados de n x 25. Donde n es el número de usuarios entre la clase 1 y clase 2. **– C** representa el parámetro de regularización que requiere el clasificador, **– S** representa el método de clasificación para los vectores de soporte, C-Support Vector Classification (C--SVC), **– t** representa el tipo de kernel, utilizado para la etapa de entrenamiento y **– b** representa la probabilidad estimada.

```
options:
-s svm_type : set type of SVM (default 0)
     0 -- C-SVC    (multi-class classification)
     1 -- nu-SVC  (multi-class classification)
     2 -- one-class SVM
     3 -- epsilon-SVR(regression)
     4 -- nu-SVR     (regression)
-t kernel_type : set type of kernel function (default 2)
     0 -- linear: u'*v
     1 -- polynomial: (gamma*u'*v + coef0)^degree
     2 -- radial basis function: exp(-gamma*|u-v|^2)
     3 -- sigmoid: tanh(gamma*u'*v + coef0)
     4 -- precomputed kernel (kernel values in training_set_file)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/num features)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR
    (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and
nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR
(default 0.1)
-m cachesize : set cache memory size in MB (default 100)
-e epsilon : set tolerance of termination criterion (default 0.001)
-m cachesize : set cache memory size in MB (default 100)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking : whether to use the shrinking heuristics, 0 or 1
  (default 1)
-b probability_estimates : whether to train a SVC or SVR
model for probability estimates, 0 or 1 (default 0)
-wi weight : set the parameter C of class i to weight*C, for
C-SVC (default 1)
-v n: n-fold cross validation mode
-q : quiet mode (no outputs)
```

***Figura 4.10*** Menú de opciones propuesto en LIBSVM 3.14 para entrenamiento

Con el propósito de obtener diferentes modelos de clasificación entrenados para cada una de las librerías utilizadas en este proyecto, se explican los parámetros de salida para la etapa de entrenamiento que compone la estructura *model* de LIBSVM 3.14 y *SVMStruct,* contenida en el toolbox Bioinformatic de Matlab.

Cuando se trabaja con la función ***svmtrain*** de la librería LIBSVM 3.14, esta retorna un modelo que se puede utilizar para la predicción futura. Se trata de una estructura conformada de la siguiente manera:

- ***Parameters:*** Contiene todos los parámetros necesarios para la construccion del modelo de clasificacion. Estos parámetros se pueden encontrar en la ***Figura 4.10.***
- ***nr_class:*** Número de clases del clasificador. En este caso se trabaja con un clasificador 2-clases.
- ***totalSV:*** Número total de vectores de soporte que contiene el clasificador.
- ***rho:*** Es el valor –b de la función decisión $wx + b$. Recordando que $b$ es la distancia perpendicular entre el origen y el hiperplano óptimo de separación.
- ***Label:*** Etiqueta de cada clase.
- ***ProbA y ProbB:*** Representación de la probabilidad estimada por parejas.
- ***nSV:*** Número de vectores de soporte para cada clase.
- ***sv_coef:*** Coeficientes para vectores de soporte en funciones de decisión.
- ***SVs:*** Vectores de Soporte contenidos en el clasificador.

En el siguiente paso, se ilustra la estructura que permite realizar la etapa de entrenamiento requerida para la validación del modelo, por medio del toolbox contenido en Matlab R2008a.

```
SVMStruct = svmtrain(trainingconsu,trainingtipconsu,'Kernel_Function'...
                     ,'rbf','RBF_Sigma',MejorSigma...
                     ,'BoxConstraint',MejorC,'method','SMO');
```

***Figura 4.11*** Función  contenida en el toolbox de Matlab para entrenamiento del modelo

La función svmtrain mostrada en la ***Figura 4.11,*** contiene una serie de parámetros que permiten la construcción de la estructura requerida para la etapa de validación. También, se muestran los valores de mejores parámetros calculados por medio de una red de búsqueda. A continuación, se ilustran los parámetros de salida para un modelo SVMStruct.

- ***SupportVectors:*** Matriz de datos en donde cada fila corresponde a un vector de soporte en el espacio de datos normalizados.

- ***Alpha***: Vector de ponderaciones para los vectores de soporte. El signo del peso es positivo para los vectores de soporte que pertenecen al primer grupo, y negativo para el segundo grupo.
- ***Bias:*** Intercepción del hiperplano que separa a los dos grupos en el espacio de datos normalizados.
- ***KernelFunction:*** Identifica la función que mapea los datos de entrenamiento en el espacio del kernel***.***
- ***KernelFunctionArgs:*** Serie de celdas de algunos argumentos requeridos para la función del Kernel.
- ***GroupNames:*** Vector categórico, numérico o lógico. También puede ser una matriz de caracteres con cada fila representada por una etiqueta de clase. Especifica los identificadores de grupo para los vectores de soporte.
- ***SupportVectorIndices:*** Vector de índices que especifican las filas de entrenamiento, los datos de entrenamiento, que fueron seleccionados como vectores de soporte después de que los datos fueron normalizados.
- ***ScaleData:*** Campo que contiene factores de normalización. Esta estructura contiene dos campos: shift y scaleFactor.
- ***FigureHandles:*** Vector de figura controlado creado por svmtrain cuando se utiliza el argumento “showplot”.

Es preciso recordar que en este caso de estudio, los datos para las pruebas de entrenamiento fueron aumentando secuencialmente. Por lo tanto, el número de estructuras SVMStruct y model, ambas estructuras de librerías diferentes, corresponden al número de pruebas realizadas para 1035, 10000, 100000 y 651000 perfiles de carga, respectivamente.

### *4.1.8 Evaluación del modelo (SVMs)*

Para la evaluación del modelo, es necesario el estudio y selección del clasificador. Dicha selección, se basa en la precisión con la que cuente el modelo. En la siguiente tabla, se mostrarán los historiales de consumo necesarios en el desarrollo de esta etapa. El número de perfiles de carga está distribuido de la siguiente manera:

***Tabla 4.1.*** Distribución de datos que componen la matriz “consumo”

| | Número Total de Perfiles de Carga | | | |
|---|---|---|---|---|
| | Normal | F.P1 | F.P2 | SOSP |
| Estrato 1 | 14000 | 3000 | 3000 | 300 |
| Estrato 2 | 14000 | 3000 | 3000 | 250 |
| Estrato 3 | 14000 | 3000 | 3000 | 200 |
| Estrato 4 | 14000 | 3000 | 3000 | 150 |
| Estrato 5 | 700 | 1500 | 1500 | 50 |
| Estrato 6 | 700 | 1500 | 1500 | 50 |
| | Total Perfiles = 101000 | | | |

Normal: Perfil de Carga Normal.

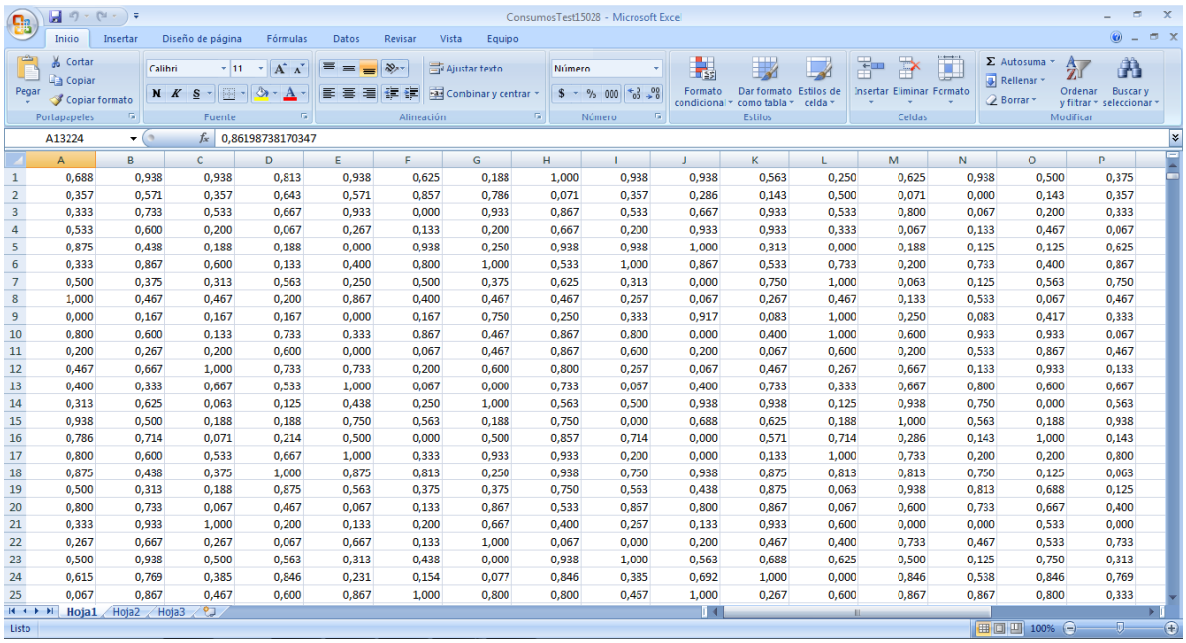
F.P1: Aumento y Disminución de consumo (Perfil Normal).

F.P2: Disminución y Aumento de consumo (Perfil Normal).

SOSP: Perfil de Carga Sospechoso.

Los datos de consumo agrupados en la ***Tabla 4.1,*** están construidos por los algoritmos mostrados al inicio de este capítulo.

Los tipos de perfiles de carga construidos para la evaluación del modelo, son aquellos datos de consumo, generalmente normalizados, que componen la siguiente tabla:




| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,688 | 0,938 | 0,938 | 0,813 | 0,938 | 0,625 | 0,188 | 1,000 | 0,938 | 0,938 | 0,563 | 0,250 | 0,625 | 0,938 | 0,500 | 0,375 |
| 2 | 0,357 | 0,571 | 0,357 | 0,643 | 0,571 | 0,857 | 0,786 | 0,071 | 0,357 | 0,286 | 0,143 | 0,500 | 0,071 | 0,000 | 0,143 | 0,357 |
| 3 | 0,333 | 0,733 | 0,533 | 0,667 | 0,933 | 0,000 | 0,933 | 0,867 | 0,533 | 0,667 | 0,933 | 0,533 | 0,800 | 0,067 | 0,200 | 0,333 |
| 4 | 0,533 | 0,600 | 0,200 | 0,067 | 0,267 | 0,133 | 0,200 | 0,667 | 0,200 | 0,933 | 0,933 | 0,333 | 0,067 | 0,133 | 0,467 | 0,067 |
| 5 | 0,875 | 0,438 | 0,188 | 0,188 | 0,000 | 0,938 | 0,250 | 0,938 | 0,938 | 1,000 | 0,313 | 0,000 | 0,188 | 0,125 | 0,125 | 0,625 |
| 6 | 0,333 | 0,867 | 0,600 | 0,133 | 0,400 | 0,800 | 1,000 | 0,533 | 1,000 | 0,867 | 0,533 | 0,733 | 0,200 | 0,733 | 0,400 | 0,867 |
| 7 | 0,500 | 0,375 | 0,313 | 0,563 | 0,250 | 0,500 | 0,375 | 0,625 | 0,313 | 0,000 | 0,750 | 1,000 | 0,063 | 0,125 | 0,563 | 0,750 |
| 8 | 1,000 | 0,467 | 0,467 | 0,200 | 0,867 | 0,400 | 0,467 | 0,467 | 0,257 | 0,067 | 0,267 | 0,467 | 0,133 | 0,533 | 0,067 | 0,467 |
| 9 | 0,000 | 0,167 | 0,167 | 0,167 | 0,000 | 0,167 | 0,750 | 0,250 | 0,333 | 0,917 | 0,083 | 1,000 | 0,250 | 0,083 | 0,417 | 0,333 |
| 10 | 0,800 | 0,600 | 0,133 | 0,733 | 0,333 | 0,867 | 0,467 | 0,867 | 0,800 | 0,000 | 0,400 | 1,000 | 0,600 | 0,933 | 0,933 | 0,067 |
| 11 | 0,200 | 0,267 | 0,200 | 0,600 | 0,000 | 0,067 | 0,467 | 0,867 | 0,600 | 0,200 | 0,067 | 0,600 | 0,200 | 0,533 | 0,867 | 0,467 |
| 12 | 0,467 | 0,667 | 1,000 | 0,733 | 0,733 | 0,200 | 0,600 | 0,800 | 0,257 | 0,067 | 0,467 | 0,267 | 0,667 | 0,133 | 0,933 | 0,133 |
| 13 | 0,400 | 0,333 | 0,667 | 0,533 | 1,000 | 0,067 | 0,000 | 0,733 | 0,057 | 0,400 | 0,733 | 0,333 | 0,667 | 0,800 | 0,600 | 0,667 |
| 14 | 0,313 | 0,625 | 0,063 | 0,125 | 0,438 | 0,250 | 1,000 | 0,563 | 0,500 | 0,938 | 0,938 | 0,125 | 0,938 | 0,750 | 0,000 | 0,563 |
| 15 | 0,938 | 0,500 | 0,188 | 0,188 | 0,750 | 0,563 | 0,188 | 0,750 | 0,000 | 0,688 | 0,625 | 0,188 | 1,000 | 0,563 | 0,188 | 0,938 |
| 16 | 0,786 | 0,714 | 0,071 | 0,214 | 0,500 | 0,000 | 0,500 | 0,857 | 0,714 | 0,000 | 0,571 | 0,714 | 0,286 | 0,143 | 1,000 | 0,143 |
| 17 | 0,800 | 0,600 | 0,533 | 0,667 | 1,000 | 0,333 | 0,933 | 0,933 | 0,200 | 0,000 | 0,133 | 1,000 | 0,733 | 0,200 | 0,200 | 0,800 |
| 18 | 0,875 | 0,438 | 0,375 | 1,000 | 0,875 | 0,813 | 0,250 | 0,938 | 0,750 | 0,938 | 0,875 | 0,813 | 0,813 | 0,750 | 0,125 | 0,063 |
| 19 | 0,500 | 0,313 | 0,188 | 0,875 | 0,563 | 0,375 | 0,375 | 0,750 | 0,553 | 0,438 | 0,875 | 0,063 | 0,938 | 0,813 | 0,688 | 0,125 |
| 20 | 0,800 | 0,733 | 0,067 | 0,467 | 0,067 | 0,133 | 0,867 | 0,533 | 0,857 | 0,800 | 0,867 | 0,067 | 0,600 | 0,733 | 0,667 | 0,400 |
| 21 | 0,333 | 0,933 | 1,000 | 0,200 | 0,133 | 0,200 | 0,667 | 0,400 | 0,257 | 0,133 | 0,933 | 0,600 | 0,000 | 0,000 | 0,533 | 0,000 |
| 22 | 0,267 | 0,667 | 0,267 | 0,067 | 0,667 | 0,133 | 1,000 | 0,067 | 0,000 | 0,200 | 0,467 | 0,400 | 0,733 | 0,467 | 0,533 | 0,733 |
| 23 | 0,500 | 0,938 | 0,500 | 0,563 | 0,313 | 0,438 | 0,000 | 0,938 | 1,000 | 0,563 | 0,688 | 0,625 | 0,500 | 0,125 | 0,750 | 0,313 |
| 24 | 0,615 | 0,769 | 0,385 | 0,846 | 0,231 | 0,154 | 0,077 | 0,846 | 0,385 | 0,692 | 1,000 | 0,000 | 0,846 | 0,538 | 0,846 | 0,769 |
| 25 | 0,067 | 0,867 | 0,467 | 0,600 | 0,867 | 1,000 | 0,800 | 0,800 | 0,457 | 1,000 | 0,267 | 0,600 | 0,867 | 0,867 | 0,800 | 0,333 |

***Figura 4.12*** Consumos de energía (kW-h) normalizados para la etapa de evaluación

En la siguiente imagen, mostraremos la función principal que permite la evaluación del modelo clasificatorio para un conjunto de perfiles de carga dado.

```
[predict_label, accuracy,prob_estimates] = svmpredict(tipconsumo...
                                          ,consumos, model,'-b 1');
```

***Figura 4.13*** Función contenida en LIBSVM 3.14 para evaluación

La estructura llamada ***svmpredict,*** está compuesta por un vector m x 1 etiquetas de predicción. Si este vector es desconocido, se pueden usar algunos valores aleatorios. Este vector lo representamos por medio de ***tipconsumo.*** La matriz ***consumos*** representa los perfiles de carga, que necesitan ser evaluados por medio del clasificador encontrado. Finalmente, el parámetro ***model*** representa las salidas de ***svmtrain.*** Usamos "- b 1" para optar por el cálculo de las probabilidades estimadas.

Los datos de salida de ***svmpredict*** proporcionan la siguiente información:

- ***predict_label:*** Es un vector de etiquetas asignado para la totalidad de perfiles de carga evaluados.
- ***Accuracy:*** Es un vector que incluye la precisión para la totalidad de perfiles de carga evaluados.
- ***Prob_estimates:*** Contiene los valores de las decisiones o las estimaciones de probabilidad según el perfil de carga.

En la ***Figura 4.14,*** *se* determina la estructura correspondiente a la etapa de evaluación por medio del toolbox contenido en Matlab.

```
Grouptipconsu = svmclassify(SVMStruct,consumos,'Showplot',true);
```

***Figura 4.14*** Función contenida en el toolbox de Matlab para evaluación

La ***Figura 4.14*** está determinada por la estructura "svmclassify". Esta función está compuesta por la estructura "SVMStruct", adquirida en la etapa de entrenamiento.

La matriz ***consumos,*** representa los perfiles de carga que necesitan ser evaluados por medio del clasificador encontrado.

"***Showplot***" es el parámetro que permite en algunos casos mostrar la clasificación de forma grafica.

Los datos de salida de esta función, están representados por las etiquetas estimadas para aquellos datos de consumos normalizados que necesitan ser evaluados.

## *4.2 RESULTADOS OBTENIDOS*

En esta etapa del proyecto se muestran los resultados obtenidos, sujetos a los datos previamente estudiados y analizados para un conjunto de pruebas realizadas en el modelo de clasificación y posteriormente su evaluación.

### *4.2.1 Resultados obtenidos para la etapa de entrenamiento y prueba*

A continuación se ilustra el conjunto de datos utilizado para entrenamiento y validación del modelo de clasificación, correspondiente a cada una de las pruebas realizadas.

***Tabla 4.2.*** Perfiles de carga para el entrenamiento del modelo

| | Total de Perfiles de Carga | Perfiles Entrenamiento | Perfiles Validación |
|---|---|---|---|
| Prueba 1 | 1035 | 724 | 310 |
| Prueba 2 | 10000 | 7000 | 3000 |
| Prueba 3 | 100000 | 70000 | 30000 |
| Prueba 4 | 651000 | 455700 | 195300 |

El total de perfiles de carga para la etapa de construcción del modelo de clasificación, serán divididos en 70% entrenamiento y 30% validación.

En las siguientes tablas, se ilustran los diferentes resultados para la etapa de entrenamiento. Se evaluaron parámetros como: el tiempo de compilación y la precisión del modelo, para un tipo de kernel determinado. El entrenamiento para cada modelo se realizó por medio del toolbox Bioinformatics, SVM de Matlab.

***Tabla 4.3.*** Tiempo de compilación para el entrenamiento del modelo

| | Tiempo de Compilación (s) | | | |
|---|---|---|---|---|
| | Número de Datos | Kernel RBF con "Mejores Parámetros" | Kernel RBF con "Parámetros por defecto" | Kernel Lineal con "Parámetros por defecto" |
| Prueba 1. | 1035 | 317.99 s | 0.8623 s | 0.4523 s |
| Prueba 2. | 10000 | N.A | 57.798 s | 2.463 s |
| Prueba 3. | 100000 | N.A | N.A | 19.99 s |
| Prueba 4. | 651000 | N.A | N.A | 79.14 s |

***Tabla 4.4.*** Precisión para el modelo de clasificación entrenado

| | Presición (%) | | | |
|---|---|---|---|---|
| | Número de Datos | Kernel RBF con "Mejores Parámetros" | Kernel RBF con "Parámetros por defecto" | Kernel Lineal con "Parámetros por defecto" |
| Prueba 1. | 1035 | 100% | 99.35% | 100% |
| Prueba 2. | 10000 | N.A | 99.90% | 100% |
| Prueba 3. | 100000 | N.A | N.A | 100% |
| Prueba 4. | 651000 | N.A | N.A | 100% |

Estos valores de tiempo y precisión, también son estudiados por medio de la librería LIBSVM 3.14. Antes de mostrar los resultados obtenidos, es necesario tener en cuenta los parámetros de desempeño del clasificador (c, g).

Para las pruebas 1 y 2, mostradas en la ***Tabla 4.5,*** los parámetros escogidos según la red de búsqueda para un conjunto de candidatos dado es [32]:

$$log_2 C = -1 : 3$$

$$log_2 g = -4 : 1$$

Para este caso se realiza Cross-validation igual a 5 (cv=5) en búsqueda de los mejores parámetros para el modelo. Los datos seleccionados, serán c= 8 y g= 0.5. El parámetro g, sólo es utilizado para el Kernel RBF. A continuación, se muestra la estructura para la búsqueda de estos parámetros.

- **Seudocódigo para obtener los mejores parámetros**

INICIO

bestcv = 0**.**

//Comentario: ***Inicializamos la variable bestcv = 0.***

Para -1 ← $\log_2 C$ Hasta 3 Con Paso 1 Hacer

Para -4 ← $\log_2 g$ Hasta 1 Con Paso 1 Hacer

cmd = [‘- v 5 -c’, num2str(2 ^ $\log_2 C$), ‘- g’, num2str(2 ^ $\log_2 g$)];

//Comentario: ***cmd es el vector que contiene los parámetros necesarios para el clasificador, dividiendo los datos de entrenamiento en 5 subconjuntos de igual cantidad de datos.***

cv = svmtrain (trainingtipconsu, Trainingconsu, cmd)

//Comentario: ***Se construyen diferentes pruebas de entrenamiento con cada Parámetro c y g.***

Si cv ≥ bestcv Entonces

cv = bestcv; bestc = 2 ^ $\log_2 C$; bestg = 2 ^ $\log_2 g$

//Comentario: ***Se obtiene la mejor precisión de todos los clasificadores Evaluados. La mejor precisión está acompañada de los Mejores parámetros.***

Fin si

Fin Para

Fin Para

FIN PROCEDIMIENTO

Para las pruebas 3 y 4, los parámetros de trabajo para el desarrollo del modelo de clasificación serán valores por defecto mostrados en el menú de la ***Figura 4.10.***

***Tabla 4.5.*** Tiempo de compilación para el entrenamiento del modelo (LIBSVM)

| | LIBSVM 3.14 | | | |
|---|---|---|---|---|
| | Tiempo de Compilación (s) | | | |
| | Número de Datos | Kernel RBF con "Mejores Parámetros" | Kernel RBF con "Parámetros por defecto" | Kernel Lineal con "Parámetros por defecto" |
| Prueba 1. | 1035 | 4.8564 s | 0.1602 s | 0.1259 s |
| Prueba 2. | 10000 | 211.2764 s | 0.76 s | 0.4171 s |
| Prueba 3. | 100000 | N.A | 14.8325 s | 5.7522 s |
| Prueba 4. | 651000 | N.A | 196.8255 s | 60.3274 s |

***Tabla 4.6.*** Precisión para el modelo de clasificación entrenado (LIBSVM)

| | LIBSVM 3.14 | | | |
|---|---|---|---|---|
| | Presición (%) | | | |
| | Número de Datos | Kernel RBF con "Mejores Parámetros" | Kernel RBF con "Parámetros por defecto" | Kernel Lineal con "Parámetros por defecto" |
| Prueba 1. | 1035 | 100% | 100% | 100% |
| Prueba 2. | 10000 | 100% | 100% | 100% |
| Prueba 3. | 100000 | N.A | 99.9967% | 99.9967% |
| Prueba 4. | 651000 | N.A | 100% | 100% |

### *4.2.2 Resultados obtenidos para la predicción de etiquetas de nuevos perfiles de carga*

Los resultados obtenidos en esta etapa del proceso, están determinados principalmente por los perfiles de carga expuestos en la ***Tabla 4.1***. Estos datos son construidos bajo patrones característicos, mencionados en el numeral ***4.1.3*** del presente capitulo.

Se recuerda que las pruebas realizadas para diferentes números de datos, contienen un modelo de clasificación. Por lo tanto, la etapa de predicción de nuevas etiquetas para un conjunto de perfiles de carga, estará determinada por la prueba que tenga el mayor número de muestras. En este sentido, trabajaremos con el modelo desarrollado para la prueba 4 correspondiente a la ***Tabla 4.6.***

Básicamente, este proceso es desarrollado para el Kernel RBF y para el Kernel Lineal. En el análisis de resultados, mostraremos algunas características relevantes de ambos métodos utilizados en el modelo de clasificación.

Para la evaluación del modelo con nuevos datos, la precisión obtenida es de 99.4782% usando el kernel RBF. Esta precisión en la clasificación, es calculada en base al vector de etiquetas ingresado en la estructura mostrada en la ***Figura 4.13.*** Es de suma importancia, resaltar que el vector de etiquetas ingresado no afecta el normal funcionamiento del clasificador; de hecho, si este vector es desconocido, se debe ingresar valores aleatorios de acuerdo al número de etiquetas requerido[4] [32].

De forma similar, para la precisión en la clasificación es de 99.4426% para la prueba realizada con el Kernel Lineal.

[4] Ver el README contenido en la librería LIBSVM 3.14. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

## *4.3 ANÁLISIS DE RESULTADOS OBTENIDOS*

En los siguientes numerales se buscan respuestas reales de acuerdo a cada uno de los datos numéricos obtenidos en pruebas realizadas. Se analizan los datos por separado, para las pruebas de entrenamiento y pruebas para la evaluación del modelo.

### *4.3.1 Análisis de resultados obtenidos para las pruebas de entrenamiento*

Los datos obtenidos en las ***Tablas 4.4*** y ***4.5,*** están representadas por el tiempo de compilación y la precisión del modelo, respectivamente. En este caso, el procedimiento se basa principalmente, en estudiar los valores de tiempo y precisión para un Kernel con parámetros de ajuste determinado. Podemos apreciar la aparición de la nomenclatura N.A, la cual representa el caso en que el programa, según el número de datos, no arrojó ningún tipo de respuesta para el desarrollo del modelo. En este sentido, podemos determinar que la carga computacional brindada por una red de búsqueda de mejores parámetros y el aumento en el número de muestras, afecta el normal funcionamiento para la etapa de entrenamiento del modelo. En ciertos casos, cuando trabajamos con el kernel lineal con parámetros de ajuste por defecto, el modelo entrenado arroja una precisión de clasificación del 100% para cada una de las pruebas realizadas. El tiempo de compilación es otro factor importante que se debe tener en cuenta. Por lo tanto, a mayor número de muestras, mayor será el tiempo para que la solución del problema de optimización, arroje la estructura necesaria para la clasificación de nuevos perfiles de carga. Esta etapa se realizó con el toolbox Bioinformatic de Matlab.

En la siguiente imagen mostraremos el error arrojado por Matlab, al término de la compilación del problema, para aquellos casos en los que aparece N.A en las tablas mencionadas anteriormente.

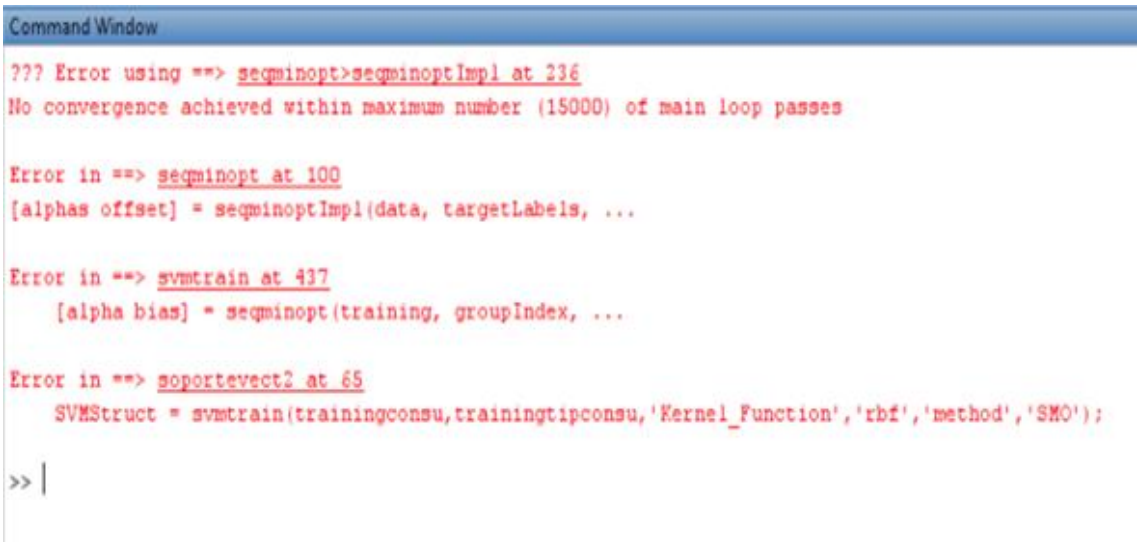



***Figura 4.15*** Error del programa al final de la compilación

En la ***Figura 4.15*** la función svmtrain, permite el desarrollo de la etapa de entrenamiento del modelo. Cuando se trabaja con un número de muestras considerable, es necesario escoger el método SMO para la solución del problema de optimización. Svmtrain cuenta con un conjunto de argumentos que entre ellos está el método SMO, utilizado para encontrar el hiperplano de separación. Cuando escogemos el método SMO, se crea una estructura statset, la cual contiene el máximo número de iteraciones predeterminado. Este valor es igual a 15000, equivalente al número máximo de iteraciones permitidas. Si se supera este límite antes de que converja el algoritmo, el algoritmo se detiene y devuelve un error como el mostrado en la figura [33].

Si se analiza la librería LIBSVM 3.14 con los resultados obtenidos anteriormente, esta librería es utilizada para aquellos problemas en el que el número de datos es considerable [34], [35]. Utiliza el método SMO como solución del problema [36].

Las ***Tablas 4.6 y 4.7,*** contienen aquellos valores de tiempo y precisión, calculados para cada una de las pruebas. Podemos determinar que el tiempo de compilación es menor respecto al toolbox, Bioinformatic de Matlab, utilizando el Kernel RBF para la totalidad de las pruebas en donde el desempeño para el modelo de clasificación mejora notablemente.

Finalmente, para la selección del modelo de clasificación, es necesario determinar la prueba que contiene el mayor número de perfiles de carga y cuya precisión, sea lo suficientemente satisfactoria, reflejada en un alto valor porcentual. En este caso, el modelo seleccionado será la estructura para los perfiles de carga de la Prueba 4, la cual contiene 651000 muestras de entrenamiento para una precisión del 100%.

Después de la selección del modelo clasificatorio, el cual contiene un número considerable de perfiles de carga para un solo estrato y la precisión del modelo, el paso siguiente es construir un nuevo modelo que especifique la cantidad de datos, teniendo en cuenta los diferentes perfiles de carga para todos los estratos.

En la siguiente tabla se ilustrará la distribución de los datos para el nuevo modelo.

***Tabla 4.7.*** Perfiles de carga para la construcción del nuevo modelo

| | Número total de Perfiles de Carga | | | |
|---|---|---|---|---|
| | Normal | F.P1 | F.P2 | SOSP |
| Estrato 1 | 126000 | 27000 | 27000 | 1800 |
| Estrato 2 | 84000 | 18000 | 18000 | 1500 |
| Estrato 3 | 84000 | 18000 | 18000 | 1200 |
| Estrato 4 | 63000 | 13500 | 13500 | 900 |
| Estrato 5 | 42000 | 9000 | 9000 | 300 |
| Estrato 6 | 21000 | 4500 | 4500 | 300 |
| | Total Perfiles = 606000 | | | |

Normal: Perfil de Carga Normal.

F.P1: Aumento y Disminución de consumo (Perfil Normal).

F.P2: Disminución y Aumento de consumo (Perfil Normal).

SOSP: Perfil de Carga Sospechoso.

Los resultados obtenidos en esta etapa después del entrenamiento realizado, se estima una precisión del 99.8333% con un tiempo de compilación de 2818s utilizando el kernel RBF. La figura que se mostrará a continuación, es el algoritmo compilado para obtener una estructura del nuevo modelo requerido.

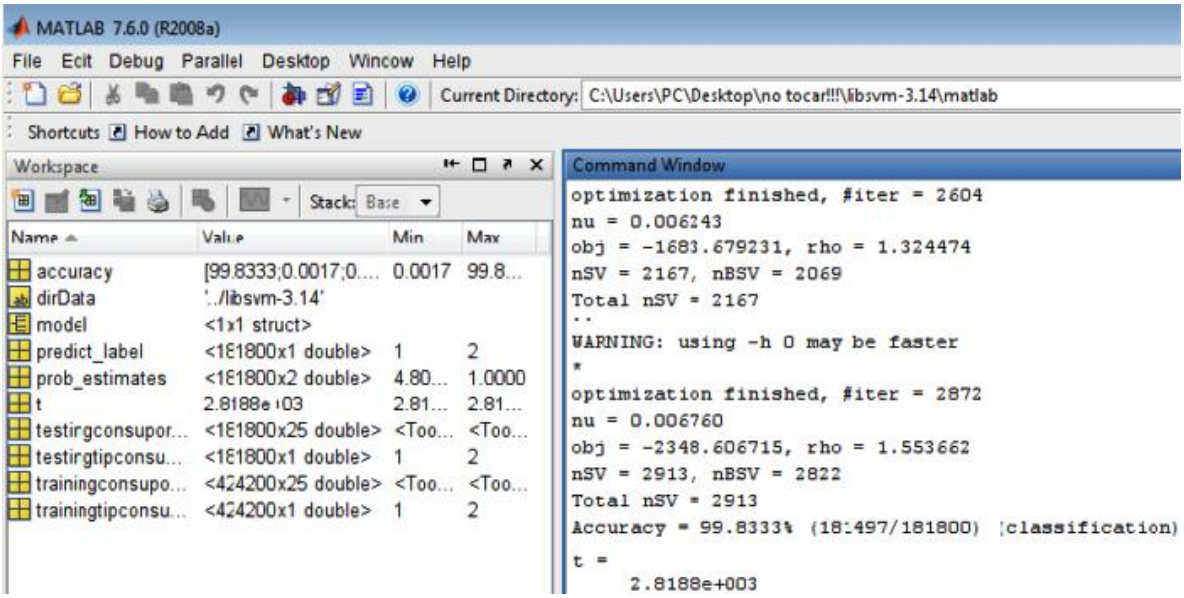



***Figura 4.16.*** Nuevo Modelo clasificador con Kernel RBF

Los resultados obtenidos en el nuevo modelo nos determinan que tanto el tiempo de compilación como la precisión, se ven afectados notoriamente respecto a los modelos obtenidos anteriormente para un solo estrato. En los análisis siguientes, determinaremos el grado de afectación que conlleva el cambio de dichos parámetros.

### *4.3.2 Análisis de resultados obtenidos para la predicción de etiquetas de nuevos perfiles de carga*

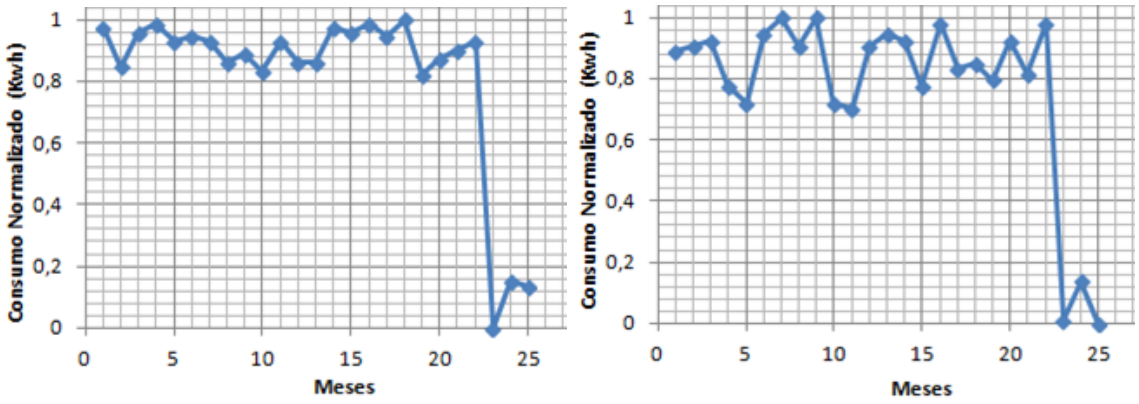
Para la predicción de nuevas etiquetas en el estudio de perfiles de carga, es necesario recurrir al estudio de dos modelos para establecer una adecuada separación de los datos. El primer modelo se basa principalmente, en escoger el Kernel RBF en busca de un modelo clasificador, que soporte el mayor número de datos posibles y que involucre el mayor número de características en todos los estratos. De igual manera, se selecciona el kernel Lineal para averiguar la variabilidad en la precisión de aquellos datos, que necesitan ser evaluados y clasificados correctamente.

En la predicción de nuevos datos obtenidos, podemos apreciar una variación en la precisión final de la clasificación. Esta precisión se obtiene por medio de aquellas etiquetas que componen el vector ***tipconsumo*** de la ***Figura 4.13,*** y las etiquetas asignadas a cada perfil de carga por el modelo clasificador, denominadas ***predict_label***. El hecho de que la clasificación final no tenga una precisión del 100%, no significa que el modelo realice una clasificación errónea de los nuevos datos, si no que algunos perfiles de carga que inicialmente fueron creados y caracterizados con algún comportamiento especifico en el consumo de energía, el clasificador acertó en el comportamiento real del perfil de carga, de acuerdo a parámetros entrenados previamente.

En este tipo de estudio es importante contar con algunos casos específicos, que se pueden presentar al momento de obtener la clasificación final. En nuestro caso, analizamos el comportamiento del clasificador cuando la construcción de un perfil de carga normal, coincidía con el perfil de carga de un usuario fraudulento o sospechoso. Recordemos que la característica principal de un cliente sospechoso, es un cambio brusco en el consumo de energía. Este comportamiento se puede presentar para un cliente normal que en un determinado periodo, su consumo disminuyó; característica propia de un consumo normal. Ahora nos hacemos la siguiente pregunta. ¿Qué sucede si la disminución o cambio en el consumo de energía se presentó al finalizar el periodo para ambos perfiles de carga?

Básicamente se busca que en el momento en que ambos perfiles de carga coincidan en el cambio de consumo para el último periodo, el clasificador por obligación debe etiquetar ambos perfiles de carga como sospechosos.

Es importante aclarar este tipo de características constructivas, ya que nuestro propósito fundamental al inicio de este trabajo, es construir perfiles de carga de acuerdo a ciertos parámetros que se pueden presentar en su comportamiento. En las siguientes imágenes, mostraremos el caso en que un perfil de carga construido como normal, coincide con un consumo fraudulento o sospechoso.




(a) (b)

***Figura 4.17.*** (a) Perfil de carga construido por Disminucionconsumofraud[5], (b) Perfil de carga construido por Disminucionyaumentoconsumo[6]

Teniendo en cuenta el problema presentado en la ***Figura 4.17,*** el clasificador inicial mostrado en las ***Tablas 4.6*** y ***4.7*** donde el tiempo de compilación para el desarrollo del modelo clasificador es de 196.8295s, precisión del 100%, para un Kernel RBF y 60.3274s, precisión del 100%, para el Kernel Lineal no cuentan con un clasificador adecuado, ya que el tipo de perfiles de carga mostrados en la ***Figura 4.17*** no son etiquetados correctamente.

[5] Programa para la construcción de perfiles de carga *sospechosos*. Ver ***Figura 4.2.***
[6] Programa para la construcción de perfiles de carga normales. Ver ***Figura 4.2.***

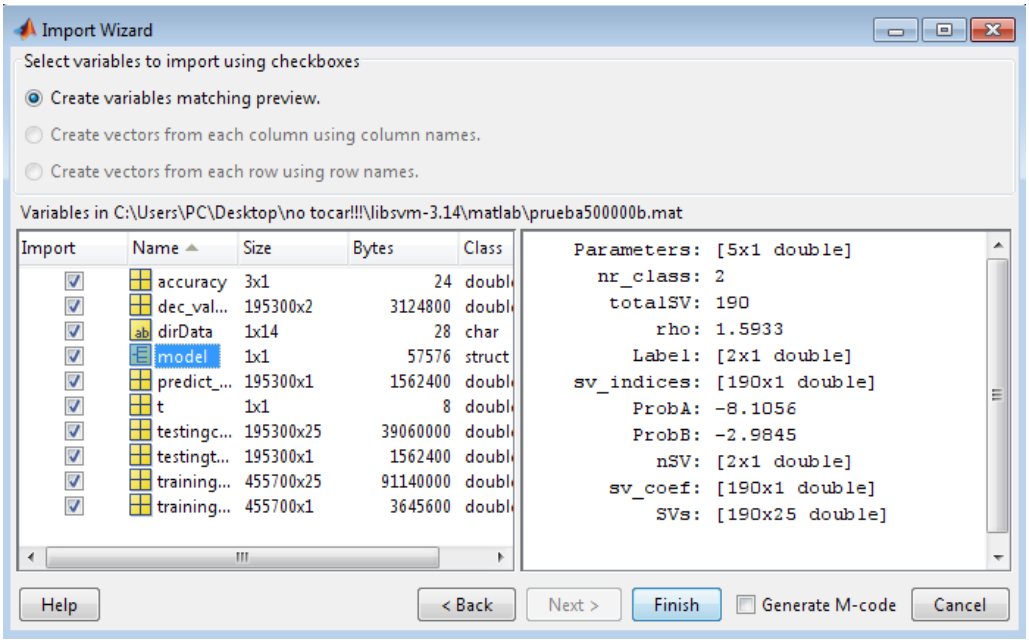



***Figura 4.18.*** Modelo clasificador inicial con Kernel RBF

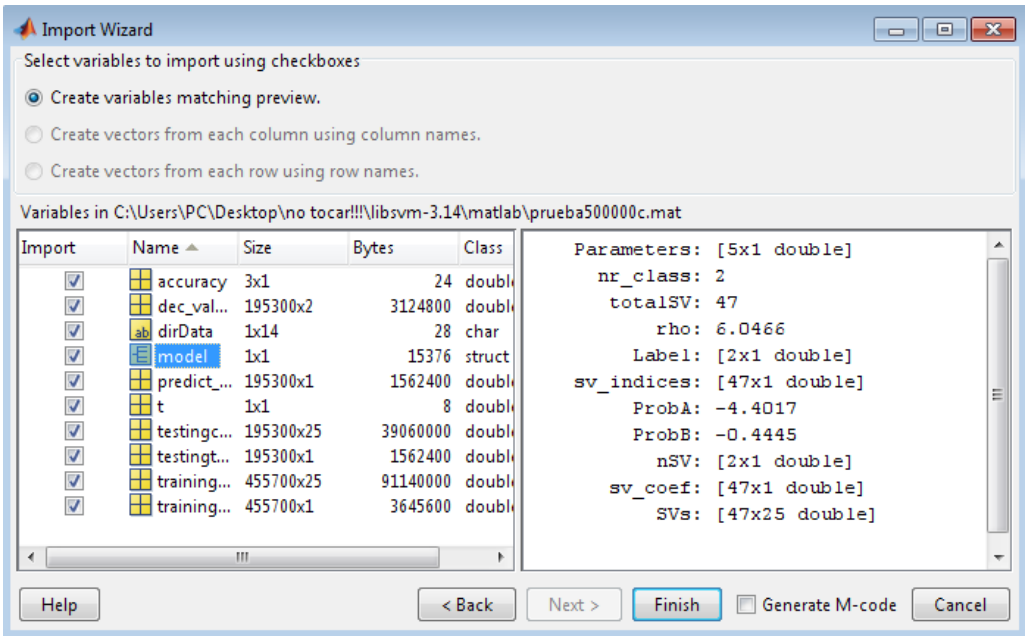



***Figura 4.19.*** Modelo clasificador inicial con Kernel Lineal

Teniendo en cuenta los modelos de clasificación mostrados anteriormente, es necesario buscar una solución óptima sin necesidad de crear un conflicto computacional para el problema. En la búsqueda de dicha solución, es necesario

analizar dos situaciones por separado. La primera sería, implementar una red de búsqueda de mejores parámetros que permita el buen funcionamiento del clasificador. La segunda opción sería, optimizar la construcción de los perfiles de carga. En este caso, ampliar el período donde se genera el fraude de forma aleatoria. Por lo tanto, es necesario, modificar el programa que permita la construcción de perfiles de carga sospechosos, generando el fraude entre el primer mes y el mes 24. Después de realizar la modificación en el programa destinado para la construcción de dichos perfiles, es necesario reentrenar nuevamente el modelo para obtener respuestas más contundentes y satisfactorias.

En las siguientes figuras, se muestran los nuevos modelos entrenados para la solución del problema.

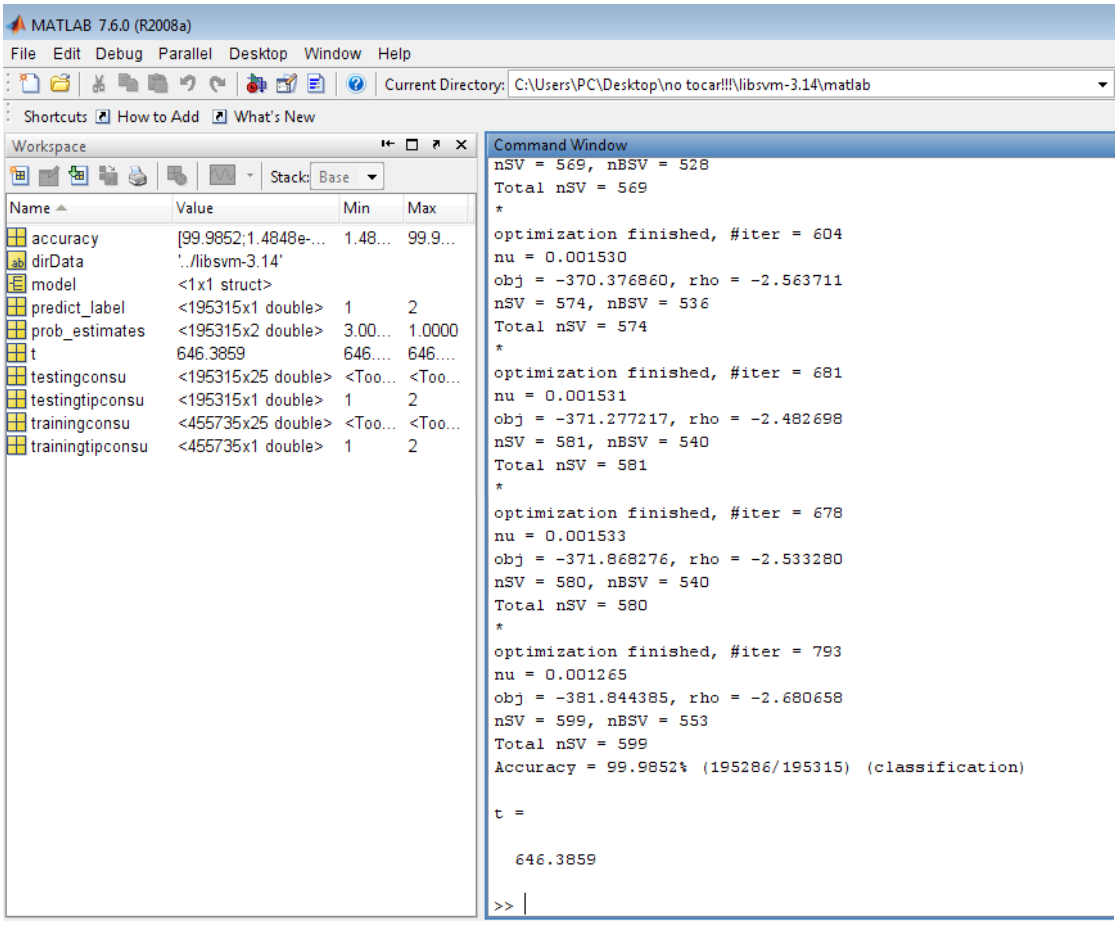



***Figura 4.20.*** Modelo clasificador reentrenado con Kernel RBF

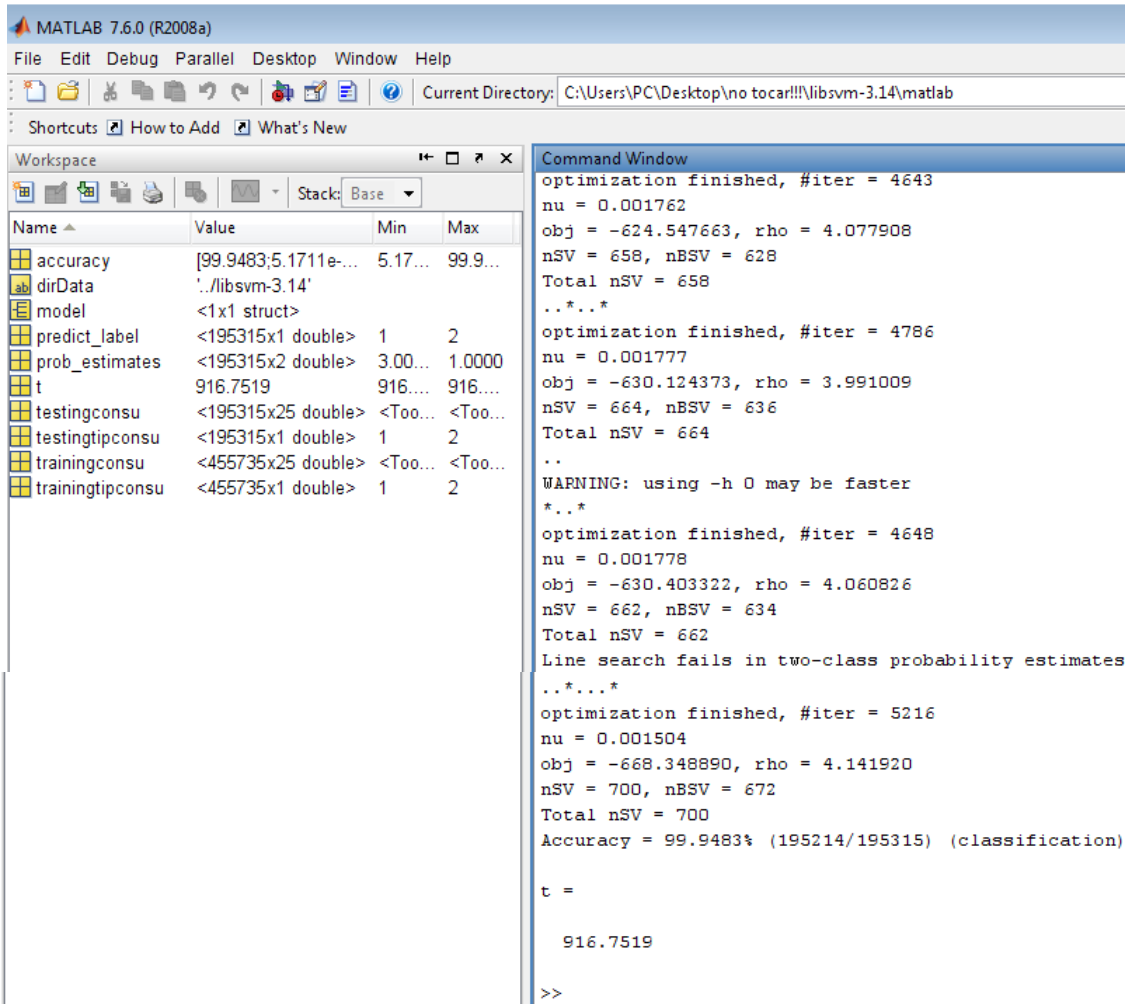



***Figura 4.21.*** Modelo clasificador reentrenado con Kernel Lineal

En las ***Figura 4.20*** y ***4.21*** se puede observar que el tiempo de compilación como la precisión del modelo, se ven afectados respecto a las estructuras inicialmente obtenidas. La característica principal del modelo reentrenado, radica en el aumento de vectores de soporte para un Kernel RBF; en este caso, 599 vectores de soporte serán utilizados para la clasificación de nuevos datos. Para el Kernel Lineal, serán utilizados 700 vectores de soporte. Se presenta un aumento significativo respecto a los modelos encontrados anteriormente.

Con estos nuevos modelos encontrados, se logra gratamente superar el conflicto mencionado en la ***figura 4.17.*** Para la evaluación de los nuevos perfiles de carga, introduciendo el modelo reentrenado con kernel RBF, se alcanza una precisión del 99.3653%. Utilizando el kernel lineal se obtiene una precisión de 99.6634%.

Por otro lado, el nuevo modelo clasificador de la ***Figura 4.16***, construido para mejorar el desempeño en la evaluación de nuevos perfiles de carga, teniendo en cuenta todos los estratos, no es lo suficientemente confiable en comparación con el modelo clasificador reentrenado con kernel RBF para un solo estrato, ya que presenta problemas en la clasificación de perfiles de carga para el caso mostrado en la ***figura 4.17.***

Finalmente, los resultados obtenidos para la predicción de etiquetas de nuevos perfiles de carga expuestos en el presente capítulo, son los resultados esperados en este trabajo de investigación. La mejor clasificación es desarrollada cuando se analiza el modelo clasificatorio, con datos para un solo estrato y con el Kernel RBF como parámetro seleccionado, ya que el tiempo de compilación y precisión en la clasificación, muestra un método más efectivo, práctico y confiable.

En la siguiente tabla, se resume de manera detallada la precisión obtenida por cada prueba realizada en busca de la evaluación y predicción de nuevos perfiles de carga:

***Tabla 4.8.*** Precisión para los nuevos perfiles de carga, evaluados con diferentes modelos clasificatorios

| | Presición en la evaluación | |
|---|---|---|
| Modelos Clasificatorios | Kernel RBF | Kernel Lineal |
| Prueba500000b | 99.4782% | |
| Prueba500000c | | 99.4426% |
| ModeloFinal500000b | 99.3653% | |
| ModeloFinal500000c | | 99.6634% |
| MODELO ESTRATOS ULTIMO | 99.8218% | |

# *CAPITULO 5 - CONCLUSIONES Y RECOMENDACIONES*

## *5.1 CONCLUSIONES*

- Teniendo en cuenta los resultados obtenidos en este trabajo de investigación, es de suma importancia contar con metodologías alternas que permitan la solución de problemas, para el caso en el que se tiene como objetivo principal, **la predicción del fraude de energía eléctrica en sistemas de distribución**. Por lo tanto, la construcción de perfiles de carga, los cuales cumplen todos los requerimientos característicos que se pueden presentar en un perfil de carga real, son avances significativos para obtener modelos clasificatorios, con altas precisiones y obtener resultados satisfactorios a futuro.

- La complejidad de un modelo de clasificación no garantiza el buen funcionamiento del mismo cuando se trabaja con perfiles de carga. Un buen modelo depende de los datos, que han sometido sometidos a un pre-procesamiento óptimo, con características claramente definidas en su diseño.

- El uso de herramientas computacionales son de bastante utilidad en problemas que requieren ser estudiados de forma clasificada. La librería LIBSVM 3.14 arroja resultados bastante positivos, sin tener en cuenta procedimientos complejos para el cálculo de parámetros de ajuste que son requeridos para el funcionamiento de las SVMs, ya que estos procedimientos, albergan una gran carga computacional, afectando el normal funcionamiento del modelo y cuyo tiempo de compilación, es bastante considerable en aquellos procedimientos donde el número de muestras es grande.

- Es de suma importancia resaltar que la característica principal de esta librería según los resultados obtenidos, es el tiempo de compilación del programa y la facilidad para reconocer gran cantidad de datos

- Si se utiliza C-SVM como parámetro de clasificación, mediante el uso de valores por defecto en el estudio de perfiles de carga, se pueden encontrar resultados importantes, aún cuando se trabaja con el Kernel RBF [40]. Esto demuestra que el método elegido es una manera sencilla y eficiente para clientes con alta probabilidad de fraude.

## *5.1 RECOMENDACIONES*

Debido al aumento poblacional que se presenta actualmente en los sistemas de distribución de energía eléctrica, las empresas comercializadoras están en la obligación de implementar planes de control en el consumo de energía para cada uno de los usuarios. Por tal motivo, se necesitan técnicas más confiables que puedan suplir esta necesidad.

Después del análisis desarrollado y el enfoque dado al proyecto, surgió una recomendación para posibles trabajos futuros:

- Construir una **interfaz gráfica** que permita a las empresas distribuidoras de energía consultar el comportamiento de los perfiles de carga asignado a cada usuario registrado, en el sistema de distribución que se tiene a cargo.

## *BIBLIOGRAFÍA*


[1] S. Ramos, Z. Vale, J. Santana and J. Duarte. "*Data Mining Contributions to Characterize MV Consumers and to Improve the Suppliers-Consumers Settlements*". IEEE Transactions on Power Systems. 2007. pp. 1-3.

[2] Comisión Reguladora de Energía y Gas. Metodología para establecer los planes de reducción de pérdidas no técnicas en los sistemas de distribución local. **Disponible en:** http://www.creg.gov.co/html/i_portals/index.php?p_origin=internal&p_name=content&p_id=MI-182&p_options=. **Consultado en septiembre de 2012.**

[3] S. Ramírez Castaño. "*Características de las Cargas*". Redes de Distribución de Energía. $3^{ra}\ Edición$. Manizales, Universidad Nacional de Colombia. pp. 22-23.

[4] L.S. Bora. "*DATA MINING AND WARE HOUSING*", Technical report, Department of Computer Science and Application, IEEE 2011 Chandrapur, India. 2011. pp 1-4

[5] B. Thuraisingham. "*Data Mining for Malicious Code Detection and Security Applications*", IEEE Conference on Web Intelligence and Intelligent Agent Technology-Workshop, 2009. pp 1.

[6] C. Pérez López, Minería de Datos. **Disponible en:** http://books.google.es/books?id=wzD_8uPFCEC&printsec=frontcover&dq=mineria+de+datos&hl=es&sa=X&ei=yq7sUPvQGo2s8QSK5oHwDA&ved=0CEAQ6AEwAA#v=onepage&q=mineria%20de%20datos&f=false. Pagina consultada 4. **Consultada en Diciembre 2012.**

[7] A. H. Nizar, Z. Y. Dong, J. H. Zhao. "*Load Profiling and Data Mining Techniques in Electricity Deregulated Market* ". presented at IEEE. 2006. pp.1-5.

[8] J. R Filho, E. M. Gontijo, A. C Delaiba, E. Mazina, J. E. Cabral, O. P. Pinto. "*Fraud Identification In Electricity Company Costumers Using Decision Tree*". IEEE Intemational Conference on Systems, Man and Cybernetics, 2004. pp 3730 – 3732

[9] A. H. Nizar, Z. Y. Dong, and Y. Wang, "*Power Utility Nontechnical Loss Analysis with Extreme Learning Machine Method*", IEEE Transactions on Power Systems, vol. 23, no. 3, Aug. 2008, pp. 946-955.

[10] R. Alves, P. Casanova, E. Quirogas, O. Ravelo, and W. Gimenez. "*Reduction of Non-Technical Losses by Modernization and Updating of Measurement Systems*", IEEE. 2006. pp. 2-5

[11] J. Nagi, K. S.Yap, S. K.Tiong, S. K. Ahmed, and M. Mohamad. "*Nontechnical Loss Detection for Metered Customers in Power Utility Using Support Vector Machines*", IEEE Transactions on power delivery, vol. 25, No. 2, APRIL 2010. pp. 1162-1170.

[12] UPME. "*PROYECCIÓN DE DEMANDA DE ENERGÍA ELÉCTRICA Y POTENCIA MÁXIMA*", Subdirección de planeación energética grupo de demanda de energía. Marzo de 2012. pp. 12-13

[13] CEDENAR S.A. E.S.P– Centrales Eléctricas de Nariño S.A. "*SISTEMA VECO UNA OPCIÓN NOVEDOSA PARA REDUCCIÓN DE PÉRDIDAS NO TÉCNICAS*". [Diapositivas]. Bogotá, Diciembre de 2011. 13 Diapositivas.

[14] EPM – Empresas Públicas de Medellín. "*PÉRDIDAS DE ENERGÍA: UNA VISIÓN DE LA INDUSTRIA*". [Diapositivas]. 22 Diapositivas.

[15] Centrales Eléctricas de Norte de Santander S.A. "*PLANEACION Y EJECUCION REDUCCION DE PÉRDIDAS*". [Diapositivas]. Octubre de 2007. 49 Diapositivas.

[16] ELECTRICARIBE. "*Estrategia operativa para el control de pérdidas en la Costa Caribe Colombiana".* [Diapositivas]. Octubre de 2007. 29 Diapositivas.

[17] J. P. Alzate Gallego, "*Comparación de Métodos Para Detección y Control de Fuentes de Perdidas No Técnicas en Sistemas de Distribución*". Informe Técnico, Universidad de los Andes. 2007.

[18] F. P. Gutiérrez, A. Chavarro Leal "*Aplicación de la Metodología del Índice de Potencialidad de Infracción (IPI) a los Usuarios de la Empresa de Energía De Arauca E.S.P*". Informe Técnico, Universidad de los Andes. Bogotá, *Colombia.*

[19] F. Valle Padilla. "*Implementación eficiente de clasificadores Prior-SVM para Matlab"*. Proyecto fin de carrera. Universidad Carlos III de Madrid. Madrid, España. 2010. pp. 15-16.

[20] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. "*A practical guide to support vector classification"*. Technical report, Dept. of Computer Science, National Taiwan University, 2010.

[21] J. Nagi, A. M. Mohammad, K. S. Yap, S. K. Tiong, S. K. Ahmed, "*Non-Technical Loss Analysis for Detection of Electricity Theft using Support Vector Machines*", $2^{nd}$ IEEE International Conference on Power and Energy (PECon 08), Johor Bahru, Malaysia. December 1-3, 2008. pp. 907-912.

[22] V. N. Vapnik, "The Nature of Statistical Learning Theory," Springer Verlag, New York, 2000.

[23] S. Ding, L. Chen, "*Intelligent Optimization Methods for High-Dimensional Data Classification for Support Vector Machines*",Technical report, College of Computer Science and Technology, Wuhan, China, *May* 4, 2010. pp. 161-168.

[24] R. Lahoz- Beltran, Aprendizaje Supervisado y Aprendizaje no supervisado, **Disponible en:** http://books.google.com.co/books?id=1Lxt1Eviy8cC&pg=PA417&lpg=PA417&dq=aprendizaje+supervisado&source=bl&ots=OqJt3x4bk6&sig=WZLvo9h0qZg5S6rTcs6QuNz5gYY&hl=es&sa=X&ei=TcrpULPoC4qY9QS154DAAQ&sqi=2&ved=0CGEQ6AEwCg#v=onepage&q=aprendizaje%20supervisado&f=false. **Consultado el 8 de noviembre de 2012.**

[25] L. Jiménez Moscovitz, P. Rengifo Rengifo, "*AL INTERIOR DE UNA MÁQUINA DE SOPORTE VECTORIAL*". Revista de Ciencias, Facultad de Ciencias Naturales y Exactas, Universidad del Valle Vol. 14, Colombia, octubre 19, 2010. pp. 73-78.

[26] G. A. Betancourt. "*LAS MÁQUINAS DE SOPORTE VECTORIAL (SVMs)", Revista* Scientia et Technica Año XI, Universidad Tecnológica de Pereira, No 27. Abril 2005. pp. 68-72.

[27] C. W Hsu, C. J Lin. "*A Comparison of Methods for Multiclass Support Vector Machines".* IEEE Transactions on neural networks, vol. 13, No. 2, March 2002. pp. 415- 417.

[28] Osuna E., Freund R., Girosi F. "Improved Training Algorithm for Support Vector Machines," *Proc. IEEE NNSP '97*, 1997.

[29] J. C. Platt, "*Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*", Technical Report MSR-TR-98-14. April 21, 1998. pp. 1-18.

[30] *Sistema Único de Información* de servicios públicos. Información Comercial - Empresa. **Disponible en:** http://reportes.sui.gov.co/fabricaReportes/frameSet.jsp?idreporte=ele_com_094. **Consultado en: Octubre de 2012.**

[31] R. Kohavi, "*A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*", in Proc. of the 14th International Joint Conference on Artificial Intelligence, 1995.

[32] C.-C. Chang and C.-J. Lin, 2005. LIBSVM: *A Library for Support Vector Machines*. http://www.csie.ntu.edu.tw/~cjlin/libsvm. **Consultado en**: octubre 28, 2012.

[33] MathWorks. Caja de herramientas Bioinformática. Svmtrain. **Disponible en**: http://www.mathworks.com/help/bioinfo/ref/svmtrain.html. **Consultado en**: Febrero de 2013.

[34] Jian-xiong Dong, Adam Krzyzak and Ching Y. Suen, "*Fast SVM Training Algorithm with Decomposition on Very Large Data Sets*", IEEE Transactions on pattern analysis and machine intelligence, Vol. 27, No. 4, April 2005, pp.603-618.

[35] Xun Liang, "*An Experimental Study on Number of Support Vectors in N-bit Parity Problem*", School of Information, Remin University of China, Beijing, China, 2010, pp. 1- 4.

[36] A. Barbero, J. R. Dorronsoro, "*Momentum Sequential Minimal Optimization: An accelerated method for Support Vector Machine training",* IEEE Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA, July 31 – August 5, 2011, pp. 370 - 376.

[37] Las Pérdidas de Energía, Enfoque Operativo, Hernando O. P. SELECCIONAR. $2^a\ edición$, ISBN: 978-958-44-3701-3, Colombia, Noviembre de 2010. pp. 74-87.

[38] Las Pérdidas de Energía, Enfoque Operativo, Hernando O. P. EL ENFOQUE DE LAS PÉRDIDAS DE ENERGÍA. $2^a\ Edición$, ISBN: 978-958-44-3701-3, Colombia, Noviembre de 2010. pp. 23 – 26.

[39] R. Solera Ureña, F. Pérez Cruz, F. Díaz de María. "*ESTIMACIÓN DE PROBABILIDADES A POSTERIORI EN SVMS MULTICLASE PARA RECONOCIMIENTO DE HABLA CONTINUA*", Reporte Técnico, Universidad Carlos III de Madrid, Avda. de la Universidad, 30, 28911-Leganés (Madrid), ESPAÑA. pp.2-3.

[40] Chu-Hsing Lin, Jung-Chun Liu, Chia-Han Ho, "*Anomaly Detection Using LibSVM Training Tools*", IEEE International Conference on Information Security and Assurance, 2008. p.p 170.

[41] Determine the Best Parameters via K-fold Crossvalidation. **Disponible en:** http://www.youtube.com/watch?v=MyywXRLuABg&list=UUYkNfnagLldTZgYBZtT3R3w&index=8. **Consultado en**: diciembre de 2012.

# ANEXOS

## *A1. PROGRAMAS UTILIZADOS EN EL PROYECTO*

### *A1.1. PROGRAMAS UTILIZADOS PARA MODELAR PERFILES DE CARGA*

#### *A1.1.1 Programas para perfiles de carga normal*

- **Programa PRINCIPAL**

```
clear all
close all
clc
meses=25;
usuarios=21000;
estrmin=347;%rango inial alto
estrmax=765;%rango inicial bajo
rangus=10;
graph=1;
datos=consumos(meses,usuarios,estrmin,estrmax,rangus,graph);
uinicio=datos;

save('C:\Users\PC\Desktop\FELIPE\TESIS2013\SVMsTesis\entrenost6normal.mat
','uinicio');
ruta=['C:\Users\PC\Desktop\no tocar!!!\libsvm-3.14\matlab'
'\consumonormalst6.mat'];
save(ruta,'uinicio');
```

***Nota***: Los valores numéricos que aparecen en este programa son sujetos a cambios durante la construcción de los perfiles de carga.

- **Programa *Función consumos***

```
function datos=consumos(meses,usuarios,estrmin,estrmax,rangus,graph)

mes=meses;
us=usuarios;
ermin=estrmin+rangus;
ermax=estrmax-rangus;
fus=rangus;
ploteo=graph;

for i=1:mes
    for j=1:us
        u(j,i)=round(ermin+(ermax-ermin).*rand);
    end
end
```

```
% se generan [us] datos que serán las franjas de cada uno de los usuarios
%e1: vector con los centros de las franjas

for k=1:us
    e1(k)=round(ermin+(ermax-ermin).*rand);
end

%Se necesita escalar los datos de cada usuario a una franja en particular
%dada por cada uno de los [us] datos de e1
% u2 es la matriz que contiene los datos escalados en franjas de cada
% usuario

for i=1:us
    u2(i,:)=scaledata(u(i,:),e1(i)-fus,e1(i)+fus);
end

if(ploteo==1)
    cstring='rgbcmyk'; % color string
hold on
for j=1:us
    plot(1:mes,u2(j,:),cstring(mod(j,7)+1))

end
end

datos=u2;
```

- **Programa *Función scaledata***

```
function dataout = scaledata(datain,minval,maxval)
%
% Program to scale the values of a matrix from a user specified minimum
to a user specified maximum
%
% Usage:
% outputData = scaleData(inputData,minVal,maxVal);
%
% Example:
% a = [1 2 3 4 5];
% a_out = scaledata(a,0,1);
%
% Output obtained:
%            0    0.1111    0.2222    0.3333    0.4444
%       0.5556    0.6667    0.7778    0.8889    1.0000
%
% Program written by:
% Aniruddha Kembhavi, July 11, 2007

dataout = datain - min(datain(:));
dataout = (dataout/range(dataout(:)))*(maxval-minval);
dataout = dataout + minval;
```

### *A1.1.2 Programa para perfiles de carga normales con aumento de consumo en un corto periodo de tiempo.*

Es importante recordar que los programas creados como funciones (**Programa *Función scaledata*** y **Programa *Función consumos***) son los mismos para este programa. Por tal motivo es importante tenerlos en cuenta.

- **Programa PRINCIPAL**

```
clear all
close all
clc
meses=25;
usuarios=4500;
% genero un vector aleatorio q me determina el tim

dale=round(1+(3-1).*rand(1,usuarios));
estrmin=347;%rango final bajo
estrmax=765;%rango final alto
rangus=10;
graph=0;

datos=consumos(meses,usuarios,estrmin,estrmax,rangus,graph);
uinicio=datos;
for i=1:usuarios
mesin(i)=round(3+(20-3).*rand);% me genera el mes aleatorio donde se
genera el fraude
end

estrmin=387;%rango inicial alto
estrmax=810;%rango inicial bajo
graph=0;
datos=consumos(meses,usuarios,estrmin,estrmax,rangus,graph);

for i=1:usuarios
dmeses(i)= mesin(i)+dale(i);
uinicio(i,[mesin(i):dmeses(i)])=datos(i,[mesin(i):dmeses(i)]);
% genero los datos fraudulentos después del aleatorio donde empieza el
fraude
end

cstring='rgbcmyk';
hold on
for j=1:usuarios
    plot(1:meses,uinicio(j,:),cstring(mod(j,7)+1))
end

save('C:\Users\PC\Desktop\FELIPE\TESIS2013\SVMsTesis\entrenost6FP1.mat','
uinicio');
ruta=['C:\Users\PC\Desktop\no tocar!!!\libsvm-3.14\matlab'
'\consumonormal1st6.mat'];
save(ruta,'uinicio');
```

### *A1.1.3 Programa para perfiles de carga normales con disminución de Consumo en un corto periodo de tiempo.*

Es importante recordar que los programas creados como funciones (**Programa *Función scaledata*** y **Programa *Función consumos***) son los mismos para este programa. Por tal motivo es importante tenerlos en cuenta.

- **Programa PRINCIPAL**

```
clear all
close all
clc
meses=25;
usuarios=1000;
% genero un vector aleatorio que me determina el tim

dale=round(1+(3-1).*rand(1,usuarios));
estrmin=102;%rango inial alto
estrmax=128;%rango inicial bajo
rangus=10;
graph=0;
datos=consumos(meses,usuarios,estrmin,estrmax,rangus,graph);
uinicio=datos;

for i=1:usuarios
mesin(i)=round(3+(22-3).*rand);%me genera el mes aleatorio donde se
genera el fraude
end
estrmin=50;%rango final bajo
estrmax=80;%rango final alto
graph=0;
datos=consumos(meses,usuarios,estrmin,estrmax,rangus,graph);

for i=1:usuarios
dmeses(i)= mesin(i)+dale(i);
uinicio(i,[mesin(i):dmeses(i)])=datos(i,[mesin(i):dmeses(i)]);% genero
los datos fraudulentos despues del aleatorio donde empieza el fraude
end

cstring='rgbcmyk';
hold on
for j=1:usuarios
    plot(1:meses,uinicio(j,:),cstring(mod(j,7)+1))
end

save('C:\Users\PC\Desktop\FELIPE\TESIS2013\SVMsTesis\entrenost6FP2.mat','
uinicio');
ruta=['C:\Users\PC\Desktop\no tocar!!!\libsvm-3.14\matlab'
'\consumonormal2st6.mat'];
save(ruta,'uinicio');
```

### *A1.1.4 Programa para perfiles de carga sospechosos*

Es importante recordar que los programas creados como funciones (**Programa *Función scaledata*** y **Programa *Función consumos***) son los mismos para este programa. Por tal motivo es importante tenerlos en cuenta.

- **Programa PRINCIPAL**

```
clear all
close all
clc
meses=25;
usuarios=300;
estrmin=347;%rango inial alto
estrmax=765;%rango inicial bajo
rangus=10;
graph=0;
datos=consumos(meses,usuarios,estrmin,estrmax,rangus,graph);
uinicio=datos;

for i=1:usuarios
mesin(i)=round(1+(24-1).*rand);% me genera el mes aleatorio donde se
genera el fraude
uinicio(i,[mesin(i):size(uinicio,2)])=0;% ponemos en cero los meses
siguientes donde empieza el fraude
end

estrmin=314;%rango final bajo
estrmax=489;%rango final alto
graph=0;
datos=consumos(meses,usuarios,estrmin,estrmax,rangus,graph);
for i=1:usuarios
uinicio(i,[mesin(i):meses])=datos(i,[mesin(i):meses]);% genero los datos
fraudulentos despues del aleatorio donde empieza el fraude
end

cstring='rgbcmyk';
hold on
for j=1:usuarios
    plot(1:meses,uinicio(j,:),cstring(mod(j,7)+1))
end

save('C:\Users\PC\Desktop\FELIPE\TESIS2013\SVMsTesis\entrenost6sosp.mat',
'uinicio');
ruta=['C:\Users\PC\Desktop\no tocar!!!\libsvm-3.14\matlab'
'\consumofraudest6.mat'];
save(ruta,'uinicio');
```

## *A1.2. PROGRAMAS UTILIZADOS PARA ENTRENAMIENTO DE MODELOS CLASIFICATORIOS*

### *A1.2.1 Programa para el entrenamiento del modelo en un solo estrato Utilizando el toolbox Bioinformatic de Matlab.*

```
clear
clc
close all

load ('consumosfraud.mat');
x=uinicio; clear 'uinicio'
clase1([1:size(x,1)],1)=2;
load ('consumosnormvac.mat');
y=uinicio; clear 'uinicio'
clase2([1:size(y,1)],1)=1;
load ('consumosvac2.mat');
z=uinicio; clear 'uinicio'
clase3([1:size(z,1)],1)=1;
load ('consumosnorm.mat');
w=uinicio; clear 'uinicio'
clase4([1:size(w,1)],1)=1;

consumos=[x;y;z;w]

for i=1:size(consumos,1)  % normaliza los consumos fila por fila.
   minimo= min(consumos(i,:));
   maximo= max(consumos(i,:));

   consumos(i,:)=(consumos(i,:)- minimo)/(maximo-minimo);
end

tipconsumo=[clase1;clase2;clase3;clase4];

[Train, Test]= crossvalind('Resubstitution',tipconsumo,[0.3,0.7]);

Trainingconsu = consumos(Train,:);
Trainingtipconsu = tipconsumo(Train,1);
Testingconsu = consumos(Test,:);
Testingtipconsu = tipconsumo(Test,1);

%numfold=10;
%Indices = crossvalind('Kfold', trainingtipconsu,numfold);
%c=2.^(-5:2:15);
%sigma=2.^(-15:2:3);

%[MejorSigma,MejorC]= MejoresParametros(trainingconsu,trainingtipconsu,…

                     numfold,Indices,sigma,c);

%SVMStruct = svmtrain(trainingconsu,trainingtipconsu,'method','SMO');
```

```
%SVMStruct = svmtrain (trainingconsu,trainingtipconsu,'Kernel_Function'…
    'rbf','RBF_Sigma',MejorSigma,'BoxConstraint',MejorC,'method','SMO');

SVMStruct = svmtrain(trainingconsu,trainingtipconsu,'Kernel_Function'…
    'rbf','method','SMO');

Grouptipconsu = svmclassify(SVMStruct,testingconsu,'Showplot',true);

acc=sum(Grouptipconsu==testingtipconsu)/sum(Test)
```

El programa anterior representa la etapa de entrenamiento para un clasificador determinado. Los datos que inicialmente son cargados al problema cambian, dependiendo del modelo que se esté desarrollando. Se puede observar que este programa cuenta con tres estructuras **`SVMStruct.`** La primera estructura determina un modelo clasificatorio con parámetros por defecto. La segunda estructura determina un modelo por medio de mejores parámetros de ajuste para el kernel RBF. Finalmente la ultima estructura, la cual se encuentra activada en este caso, determina un modelo clasificatorio utilizando el kernel RBF y con parámetros por defecto. Es importante aclarar que solo se utiliza una estructura de las tres mencionadas anteriormente.

- **Programa *Función MejoresParámetros* utilizando el toolbox Bioinformatic de Matlab [41].**

```
function [MejorSigma, MejorC]=MejoresParametros(trainingconsu,…
,trainingtipconsu,numfold,Indices,sigma,c)

 for j=1:length(c)
  for k=1:length(sigma)
   for i=1:numfold

foldtestingconsu=trainingconsu(Indices==i,:);
foldtrainingconsu=trainingconsu(Indices~=i,:);
foldtrainingtipconsu=trainingtipconsu(Indices~=i,:);

SVMStruct= svmtrain(foldtrainingconsu,foldtrainingtipconsu,…
'Kernel_Function','RBF','RBF_Sigma',sigma(k),'BoxConstraint',c(j),…
'method','SMO');

Grouptipconsu(Indices==i,1) = svmclassify(SVMStruct,foldtestingconsu,…
'Showplot',0);

end

foldpresicion(j,k)=sum(Grouptipconsu==trainingtipconsu)/length(trainingti
pconsu);
  end
end
```

```
[maxpresicion,Ipresicion] = max(foldpresicion,[],1);
[b,Ic] = max(maxpresicion,[],2);
MejorSigma=sigma(Ic);
MejorC=c(Ipresicion(Ic));
```

### *A1.2.2 Programa general para el entrenamiento del modelo utilizando LIBSVM 3.14* [32].

```
clear
clc
close all

tic

addpath('../libsvm-3.14/matlab');
dirData = '../libsvm-3.14';
addpath(dirData);

load ('entrenartipconsumo.mat');
load ('entrenarconsumo.mat');
load ('evaluartipconsumo.mat');
load ('evaluarconsumo.mat');

%bestcv = 0;
%for log2c = -1:3,
 % for log2g = -4:1,
  % cmd = ['-v 5 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
   % cv = svmtrain(trainingtipconsu,trainingconsu, cmd);
    %if (cv >= bestcv),
     % bestcv = cv; bestc = 2^log2c; bestg = 2^log2g;
    %end
    %fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv,
bestc, bestg, bestcv);
 %end
%end

model= svmtrain (trainingtipconsuporestrato,trainingconsuporestrato,
'-c 1  -s 0 -t 2 -b 1');

[predict_label, accuracy, prob_estimates] = svmpredict(testingtipconsu,
testingconsu, model,'-b 1');

t=toc
```

- ***Programa utilizado para la partición de datos de entrenamiento para un solo estrato***

Con este programa se busca que la partición de los datos de entrenamiento sea lo más precisa posible en búsqueda de la preparación de un modelo clasificatorio. El programa es el siguiente:

```
clear
clc
close all

load ('consumosfraud.mat');
x=uinicio; clear 'uinicio'
clase1([1:size(x,1)],1)=2;
load ('consumosnormvac.mat');
y=uinicio; clear 'uinicio'
clase2([1:size(y,1)],1)=1;
load ('consumosvac2.mat');
z=uinicio; clear 'uinicio'
clase3([1:size(z,1)],1)=1;
load ('consumosnorm.mat');
w=uinicio; clear 'uinicio'
clase4([1:size(w,1)],1)=1;

consumos=[x;y;z;w]

for i=1:size(consumos,1)  % normaliza los consumos fila por fila.
   minimo= min(consumos(i,:));
   maximo= max(consumos(i,:));

   consumos(i,:)=(consumos(i,:)- minimo)/(maximo-minimo);
end

tipconsumo=[clase1;clase2;clase3;clase4];

[Train, Test]= crossvalind('Resubstitution',tipconsumo,[0.3,0.7]);

Trainingconsu = consumos(Train,:);
Trainingtipconsu = tipconsumo(Train,1);
Testingconsu = consumos(Test,:);
Testingtipconsu = tipconsumo(Test,1);

save('C:\Users\PC\Desktop\no tocar!!!\libsvm-
3.14\LIBSVMTesis\entrenarconsumo.mat','trainingconsu');

save('C:\Users\PC\Desktop\no tocar!!!\libsvm-
3.14\LIBSVMTesis\entrenartipconsumo.mat','trainingtipconsu');

save('C:\Users\PC\Desktop\no tocar!!!\libsvm-
3.14\LIBSVMTesis\evaluarconsumo.mat','testingconsu');

save('C:\Users\PC\Desktop\no tocar!!!\libsvm-
3.14\LIBSVMTesis\evaluartipconsumo.mat','testingtipconsu');
```

- ***Programa utilizado para la partición de datos de entrenamiento para todos los estratos***

```
clear
clc
close all
```

```
load ('entrenost1sosp.mat');
x1=uinicio; clear 'uinicio'
clase1([1:size(x1,1)],1)=2;
load ('entrenost2sosp.mat');
x2=uinicio; clear 'uinicio'
clase2([1:size(x2,1)],1)=2;
load ('entrenost3sosp.mat');
x3=uinicio; clear 'uinicio'
clase3([1:size(x3,1)],1)=2;
load ('entrenost4sosp.mat');
x4=uinicio; clear 'uinicio'
clase4([1:size(x4,1)],1)=2;
load ('entrenost5sosp.mat');
x5=uinicio; clear 'uinicio'
clase5([1:size(x5,1)],1)=2;
load ('entrenost6sosp.mat');
x6=uinicio; clear 'uinicio'
clase6([1:size(x6,1)],1)=2;

load ('entrenost1normal.mat');
x7=uinicio; clear 'uinicio'
load ('entrenost2normal.mat');
x8=uinicio; clear 'uinicio'
load ('entrenost3normal.mat');
x9=uinicio; clear 'uinicio'
load ('entrenost4normal.mat');
x10=uinicio; clear 'uinicio'
load ('entrenost5normal.mat');
x11=uinicio; clear 'uinicio'
load ('entrenost6normal.mat');
x12=uinicio; clear 'uinicio'

clase7([1:size(x7,1)],1)=1;
clase8([1:size(x8,1)],1)=1;
clase9([1:size(x9,1)],1)=1;
clase10([1:size(x10,1)],1)=1;
clase11([1:size(x11,1)],1)=1;
clase12([1:size(x12,1)],1)=1;

load ('entrenost1FP1.mat');
x13=uinicio; clear 'uinicio'
load ('entrenost2FP1.mat');
x14=uinicio; clear 'uinicio'
load ('entrenost3FP1.mat');
x15=uinicio; clear 'uinicio'
load ('entrenost4FP1.mat');
x16=uinicio; clear 'uinicio'
load ('entrenost5FP1.mat');
x17=uinicio; clear 'uinicio'
load ('entrenost6FP1.mat');
x18=uinicio; clear 'uinicio'

clase13([1:size(x13,1)],1)=1;
clase14([1:size(x14,1)],1)=1;
clase15([1:size(x15,1)],1)=1;
clase16([1:size(x16,1)],1)=1;
```

```
clase17([1:size(x17,1)],1)=1;
clase18([1:size(x18,1)],1)=1;

load ('entrenost1FP2.mat');
x19=uinicio; clear 'uinicio'
load ('entrenost2FP2.mat');
x20=uinicio; clear 'uinicio'
load ('entrenost3FP2.mat');
x21=uinicio; clear 'uinicio'
load ('entrenost4FP2.mat');
x22=uinicio; clear 'uinicio'
load ('entrenost5FP2.mat');
x23=uinicio; clear 'uinicio'
load ('entrenost6FP2.mat');
x24=uinicio; clear 'uinicio'

clase19([1:size(x19,1)],1)=1;
clase20([1:size(x20,1)],1)=1;
clase21([1:size(x21,1)],1)=1;
clase22([1:size(x22,1)],1)=1;
clase23([1:size(x23,1)],1)=1;
clase24([1:size(x24,1)],1)=1;

consumosporestratos=[x1;x2;x3;x4;x5;x6;x7;x8;x9;x10;x11;x12;x13;x14;x15;
x16;x17;x18;x19;x20;x21;x22;x23;x24];

for i=1:size(consumosporestratos,1)

   minimo= min(consumosporestratos(i,:));
   maximo= max(consumosporestratos(i,:));

consumosporestratos(i,:)=(consumosporestratos(i,:)-minimo)/(maximo-
minimo);

end

tipconsumoporestratos=[clase1;clase2;clase3;clase4;clase5;clase6;clase7;
clase8;clase9;clase10;clase11;clase12;clase13;clase14;clase15;clase16;
clase17;clase18;clase19;clase20;clase21;clase22;clase23;clase24];

[Train, Test] = crossvalind('Resubstitution',tipconsumoporestratos,
[0.3,0.7]);

Trainingconsuporestrato = consumosporestratos(Train,:);
Trainingtipconsuporestrato = tipconsumoporestratos(Train,1);
Testingconsuporestrato = consumosporestratos(Test,:);
Testingtipconsuporestrato = tipconsumoporestratos(Test,1);

save('C:\Users\PC\Desktop\no tocar!!!\libsvm-
3.14\LIBSVMTesis\entrenarconsumoporestrato.mat','trainingconsuporestrato'
);
save('C:\Users\PC\Desktop\no tocar!!!\libsvm-
3.14\LIBSVMTesis\entrenartipconsumoporestrato.mat','trainingtipconsupores
trato');
```

```
save('C:\Users\PC\Desktop\no tocar!!!\libsvm-
3.14\LIBSVMTesis\evaluarconsumoporestrato.mat','testingconsuporestrato');

save('C:\Users\PC\Desktop\no tocar!!!\libsvm-
3.14\LIBSVMTesis\evaluartipconsumoporestrato.mat','testingtipconsuporestr
ato');
```

### *A1.3. PROGRAMAS UTILIZADOS PARA LA EVALUACIÓN DE NUEVOS PERFILES DE CARGA*

```
clear all
clc
close all

load('MODELO ESTRATOS ULTIMO.mat', 'model');

load('consumofraudest1.mat','uinicio');
consumofraude11=uinicio; clear uinicio
clase1([1:size(consumofraude11,1)],1)=2;
for i=1:size(consumofraude11,1)
   minimo= min(consumofraude11(i,:));
   maximo= max(consumofraude11(i,:));
   consumofraude11(i,:)=(consumofraude11(i,:)- minimo)/(maximo-minimo);
   end

load('consumofraudest2.mat','uinicio');
consumofraude22=uinicio; clear uinicio
clase2([1:size(consumofraude22,1)],1)=2;
for i=1:size(consumofraude22,1)
   minimo= min(consumofraude22(i,:));
   maximo= max(consumofraude22(i,:));
   consumofraude22(i,:)=(consumofraude22(i,:)- minimo)/(maximo-minimo);
   end

load('consumofraudest3.mat','uinicio');
consumofraude33=uinicio; clear uinicio
clase3([1:size(consumofraude33,1)],1)=2;
for i=1:size(consumofraude33,1)
   minimo= min(consumofraude33(i,:));
   maximo= max(consumofraude33(i,:));
   consumofraude33(i,:)=(consumofraude33(i,:)- minimo)/(maximo-minimo);
   end

load('consumofraudest4.mat','uinicio');
consumofraude44=uinicio; clear uinicio
clase4([1:size(consumofraude44,1)],1)=2;
for i=1:size(consumofraude44,1)
   minimo= min(consumofraude44(i,:));
   maximo= max(consumofraude44(i,:));
   consumofraude44(i,:)=(consumofraude44(i,:)- minimo)/(maximo-minimo);
end
```

```
load('consumofraudest5.mat','uinicio');
consumofraude55=uinicio; clear uinicio
clase5([1:size(consumofraude55,1)],1)=2;
for i=1:size(consumofraude55,1)
   minimo= min(consumofraude55(i,:));
   maximo= max(consumofraude55(i,:));
   consumofraude55(i,:)=(consumofraude55(i,:)- minimo)/(maximo-minimo);
end

load('consumofraudest6.mat','uinicio');
consumofraude66=uinicio; clear uinicio
clase6([1:size(consumofraude66,1)],1)=2;
for i=1:size(consumofraude66,1)
   minimo= min(consumofraude66(i,:));
   maximo= max(consumofraude66(i,:));
   consumofraude66(i,:)=(consumofraude66(i,:)- minimo)/(maximo-minimo);
end

load('consumonormal1st1.mat','uinicio');
consumonormala1=uinicio; clear uinicio
clase7([1:size(consumonormala1,1)],1)=1;
for i=1:size(consumonormala1,1)
   minimo= min(consumonormala1(i,:));
   maximo= max(consumonormala1(i,:));
   consumonormala1(i,:)=(consumonormala1(i,:)- minimo)/(maximo-minimo);
end

load('consumonormal1st2.mat','uinicio');
consumonormala2=uinicio; clear uinicio
clase8([1:size(consumonormala2,1)],1)=1;
for i=1:size(consumonormala2,1)
   minimo= min(consumonormala2(i,:));
   maximo= max(consumonormala2(i,:));
   consumonormala2(i,:)=(consumonormala2(i,:)- minimo)/(maximo-minimo);
end

load('consumonormal1st3.mat','uinicio');
consumonormala3=uinicio; clear uinicio
clase9([1:size(consumonormala3,1)],1)=1;
for i=1:size(consumonormala3,1)
   minimo= min(consumonormala3(i,:));
   maximo= max(consumonormala3(i,:));
   consumonormala3(i,:)=(consumonormala3(i,:)- minimo)/(maximo-minimo);
end

load('consumonormal1st4.mat','uinicio');
consumonormala4=uinicio; clear uinicio
clase10([1:size(consumonormala4,1)],1)=1;
for i=1:size(consumonormala4,1)
   minimo= min(consumonormala4(i,:));
   maximo= max(consumonormala4(i,:));
   consumonormala4(i,:)=(consumonormala4(i,:)- minimo)/(maximo-minimo);
end
```

```
load('consumonormal1st5.mat','uinicio');
consumonormala5=uinicio; clear uinicio
clase11([1:size(consumonormala5,1)],1)=1;
for i=1:size(consumonormala5,1)
   minimo= min(consumonormala5(i,:));
   maximo= max(consumonormala5(i,:));
   consumonormala5(i,:)=(consumonormala5(i,:)- minimo)/(maximo-minimo);
end

load('consumonormal1st6.mat','uinicio');
consumonormala6=uinicio; clear uinicio
clase12([1:size(consumonormala6,1)],1)=1;
for i=1:size(consumonormala6,1)
   minimo= min(consumonormala6(i,:));
   maximo= max(consumonormala6(i,:));
   consumonormala6(i,:)=(consumonormala6(i,:)- minimo)/(maximo-minimo);
end

load('consumonormal2st1.mat','uinicio');
consumonormalb1=uinicio; clear uinicio
clase13([1:size(consumonormalb1,1)],1)=1;
for i=1:size(consumonormalb1,1)
   minimo= min(consumonormalb1(i,:));
   maximo= max(consumonormalb1(i,:));
   consumonormalb1(i,:)=(consumonormalb1(i,:)- minimo)/(maximo-minimo);
end

load('consumonormal2st2.mat','uinicio');
consumonormalb2=uinicio; clear uinicio
clase14([1:size(consumonormalb2,1)],1)=1;
for i=1:size(consumonormalb2,1)
   minimo= min(consumonormalb2(i,:));
   maximo= max(consumonormalb2(i,:));
   consumonormalb2(i,:)=(consumonormalb2(i,:)- minimo)/(maximo-minimo);
end

load('consumonormal2st3.mat','uinicio');
consumonormalb3=uinicio; clear uinicio
clase15([1:size(consumonormalb3,1)],1)=1;
for i=1:size(consumonormalb3,1)
   minimo= min(consumonormalb3(i,:));
   maximo= max(consumonormalb3(i,:));
   consumonormalb3(i,:)=(consumonormalb3(i,:)- minimo)/(maximo-minimo);
end

load('consumonormal2st4.mat','uinicio');
consumonormalb4=uinicio; clear uinicio
clase16([1:size(consumonormalb4,1)],1)=1;
for i=1:size(consumonormalb4,1)
   minimo= min(consumonormalb4(i,:));
   maximo= max(consumonormalb4(i,:));
   consumonormalb4(i,:)=(consumonormalb4(i,:)- minimo)/(maximo-minimo);
end
```

```
load('consumonormal2st5.mat','uinicio');
consumonormalb5=uinicio; clear uinicio
clase17([1:size(consumonormalb5,1)],1)=1;
for i=1:size(consumonormalb5,1)
   minimo= min(consumonormalb5(i,:));
   maximo= max(consumonormalb5(i,:));
   consumonormalb5(i,:)=(consumonormalb5(i,:)- minimo)/(maximo-minimo);
end

load('consumonormal2st6.mat','uinicio');
consumonormalb6=uinicio; clear uinicio
clase18([1:size(consumonormalb6,1)],1)=1;
for i=1:size(consumonormalb6,1)
   minimo= min(consumonormalb6(i,:));
   maximo= max(consumonormalb6(i,:));
   consumonormalb6(i,:)=(consumonormalb6(i,:)- minimo)/(maximo-minimo);
end

load('consumonormalst1.mat','uinicio');
consumonormal1=uinicio; clear uinicio
clase19([1:size(consumonormal1,1)],1)=1;
for i=1:size(consumonormal1,1)
   minimo= min(consumonormal1(i,:));
   maximo= max(consumonormal1(i,:));
   consumonormal1(i,:)=(consumonormal1(i,:)- minimo)/(maximo-minimo);
end

load('consumonormalst2.mat','uinicio');
consumonormal2=uinicio; clear uinicio
clase20([1:size(consumonormal2,1)],1)=1;
for i=1:size(consumonormal2,1)
   minimo= min(consumonormal2(i,:));
   maximo= max(consumonormal2(i,:));
   consumonormal2(i,:)=(consumonormal2(i,:)- minimo)/(maximo-minimo);
end

load('consumonormalst3.mat','uinicio');
consumonormal3=uinicio; clear uinicio
clase21([1:size(consumonormal3,1)],1)=1;
for i=1:size(consumonormal3,1)
   minimo= min(consumonormal3(i,:));
   maximo= max(consumonormal3(i,:));
   consumonormal3(i,:)=(consumonormal3(i,:)- minimo)/(maximo-minimo);
end

load('consumonormalst4.mat','uinicio');
consumonormal4=uinicio; clear uinicio
clase22([1:size(consumonormal4,1)],1)=1;
for i=1:size(consumonormal4,1)
   minimo= min(consumonormal4(i,:));
   maximo= max(consumonormal4(i,:));
   consumonormal4(i,:)=(consumonormal4(i,:)- minimo)/(maximo-minimo);
end
```

```
load('consumonormalst5.mat','uinicio');
consumonormal5=uinicio; clear uinicio
clase23([1:size(consumonormal5,1)],1)=1;
for i=1:size(consumonormal5,1)
   minimo= min(consumonormal5(i,:));
   maximo= max(consumonormal5(i,:));
   consumonormal5(i,:)=(consumonormal5(i,:)- minimo)/(maximo-minimo);
end

load('consumonormalst6.mat','uinicio');
consumonormal6=uinicio; clear uinicio
clase24([1:size(consumonormal6,1)],1)=1;
for i=1:size(consumonormal6,1)
   minimo= min(consumonormal6(i,:));
   maximo= max(consumonormal6(i,:));
   consumonormal6(i,:)=(consumonormal6(i,:)- minimo)/(maximo-minimo);
end


consumos=[consumofraude11;consumofraude22;consumofraude33;consumofraude44
;consumofraude55;consumofraude66;consumonormala1;consumonormala2;
consumonormala3;consumonormala4;consumonormala5;consumonormala6;
consumonormalb1;consumonormalb2;consumonormalb3;consumonormalb4;
consumonormalb5;consumonormalb6;consumonormal1;consumonormal2;
consumonormal3;consumonormal4;consumonormal5;consumonormal6];

tipconsumo=[clase1;clase2;clase3;clase4;clase5;clase6;clase7;clase8;
clase9;clase10;clase11;clase12;clase13;clase14;clase15;clase16;clase17;
clase18;clase19;clase20;clase21;clase22;clase23;clase24];

[predict_label,accuracy,prob_estimates] = svmpredict(tipconsumo,consumos,

model,'-b 1');
```