Facultad de Ingeniería, Universidad de la República
Reconocimiento de Patrones

Detección de consumos anómalos de energía eléctrica

Elaborado por:

Diego Acuña
Lucía Korenko

Tutores:

Alicia Fernández
Ignacio Ramirez

Versión	Fecha	Detalle
Avance	21/10/2014	Random Forest
Final	12/12/2014	Clustering

Índice

1. Introducción

2. Descripción del problema

2.1. Introducción del problema

En el suministro de energía eléctrica se entiende por pérdidas a la energía que es entregada a la red y que no es facturada por la empresa suministradora . Existen dos tipos de pérdidas, en primer lugar se encuentran las pérdidas técnicas las cuales están asociadas al hecho de la distribución de energía eléctrica no es completamente eficiente, e.g. existen pérdidas Joule en lo cables y transformadores.

Por otro lado están las pérdidas no técnicas, las cuales pueden deberse a defectos en los equipos de medida de consumo, manipulación fraudulenta de los mismos o errores en el proceso de facturación. Este es el tipo de irregularidades son las que se desean detectar. Se excluyen las pérdidas no técnicas en las zonas carenciadas dado que estas ya se encuentran plenamente identificadas.

La detección de estas irregularidades en el consumo de energía eléctrica es un asunto de interés para las empresas suministradoras dado las pérdidas económicas que estas producen. En el Uruguay el porcentaje de pérdidas es elevado. Se estima que las pérdidas totales se ubican en el $19,3\,\%$, si se excluyen las pérdidas técnicas y las correspondientes a las zonas carenciadas se estima las pérdidas son aproximadamente del $4\,\%$. Por esta razón UTE trabaja en la detección de estas irregularidades.

El método tradicional de detección de irregularidades utilizado en UTE, es mediante inspección visual de la curva de consumos, es decir una persona experta inspecciona la curva de consumos históricos de los clientes y determina si el mismo es sospechoso o no de fraude. Luego se realiza una inspección de estos clientes sospechosos para determinar si efectivamente existe una irregularidad.

La cantidad de clientes en el mercado eléctrico uruguayo es muy elevada y el recurso de personas expertas para realizar las inspecciones visuales es limitado. Esto imposibilita la inspección del total de los clientes y es necesario limitarse a un subconjunto de clientes, llamado clientes estratégicos. Los clientes estratégicos son aquellos clientes con mayor potencia contratada y clientes comerciales. Otra desventaja que presenta este método es que la tarea consume mucho tiempo y resulta monótona, lo cual conlleva a que luego de un tiempo de estar analizando consumos la atención de la persona decaiga reduciéndose así la efectividad de la inspección.

Esto motiva a desarrollar una herramienta que permita clasificar de manera automática si el consumo es sospechoso de fraude o no. De esta manera se podría inspeccionar al total de los clientes así como destinar el recursos humano de experto en el tema en tareas más productivas, como inspeccionar solamente a los clientes previamente etiquetados como sospechosos.

2.2. Antecedentes

Tal como fue descrito la empresa suministradora de energía eléctrica UTE ha demostrado gran interés en la implementación de un algoritmo automático de detección. Por esta razón ya se han realizado varios estudios sobre este tema en la facultad.

El problema fue introducido por primera vez en el 2008 por Juan Pablo Kosut y Diego Alcetegaray, en el marco del proyecto final del curso 'Introducción al reconocimiento de patrones' [RP2008] . En el mismo se trabajó con muestras de consumo de clientes pertenecientes a los rubros almacenes y autoservicios en un periodo de 3 años. Estas muestras se encontraba etiquetadas como consumos sospechosos y no sospechosos en base a los resultados obtenidos a través de inspección visual . Se propuso utilizar la técnica One Class SVM, en la que se consideró al problema como de una sola clase, los consumos normales, mientras que los consumos anormales son considerados como outliers.

En el 2011, se amplió el estudio del tema en el proyecto de fin de carrera carrera realizado por Federico Decia, Matías Di Martino y Juan Molinelli [Deca2011]. En el mismo se amplío la base de datos, utilizando muestras de consumo de clientes estratégicos. Se utilizaron diferentes técnicas de clasificación como One Class SVM, Cost Sensitive SVM, Optimum Path Forest y C4.5. También fue tenido en consideración el desbalance de clases que presenta el problema a tratar.

Tambin en 2011, Diego Introini y Daniel Lena[RP2011], analizaron el tema desde otra perspectiva. En este caso se buscó identificar clusters dentro de los consumidores y analizar si en estos clusters se podían asociar al rubro de los consumidores.

En el 2012, Fernanda Rodriguez y Sebastián Castro [RP2012] propusieron la inclusión de nuevas característica no extraídas de los consumo para el diseño del clasificador. Algunas de las características que fueron tenidas en consideración son potencia contratada, zona, si el cliente ya cometió alguna irregularidad, entre otras.

Finalmente en el 2014, Fernanda Rodríguez, Federico Lecumberry y Alicia Fernandez realizaron el estudio *Proyecto CSIC Sector Productivo Modalidad I - UTE Detección de registros de consumo anómalos*[CSIC-UTE2014], en el cual se evaluaron nuevas técnicas. Una de las bases de datos que utilizadas en el estudio es la misma que la que se utilizó en este proyecto por lo cual fue de gran utilizada comparar los resultados obtenidos con los resultados presentados en el estudio.

2.3. Descripción de los datos

2.3.1. Base de datos utilizada:

Para trabajar en este proyecto se disponen de las siguientes bases de datos:

BASE 6: 2058 curvas de consumo de clientes estratégicos con un registro de 36 meses con simple etiquetado de fraude/no fraude.

BASE 6+: para los datos de la base anterior se posee una serie de nuevas características no basadas en las curvas de consumo de los clientes, las cuales se acoplaron para analizar el impacto en el desempeño de los clasificadores.

Esta base se utilizó para entrenar y para evaluar los clasificadores. Para ello se dividió la misma en dos subconjunto aleatorios. El primer subconjunto, el cual contiene el 75 % de las muestra, fue utilizado entrenar y buscar los parámetros óptimos de los clasificadores, al cual llamaremos conjunto de entrenamiento. Una vez se terminó con el entrenamiento de los clasificadores se evaluó el desempeño de los mismos, utilizando muestras del conjunto de test.

2.3.2. Caracterísiticas utilizadas

Características extraídas en los consumos:

De acuerdo a lo visto en la sección 2.2 el problema de detección de fraudes ya fue tratado con anterioridad, y parte de los resultados obtenidos en los mismos fueron utilizados en este trabajo. En particular se utilizaron las características propuestas en estudios anteriores. En [Deca2011] se utilizó un serie de características calculadas a partir de las curvas de consumo. A continuación se encuentra una breve descripción de las mismas y la numeración utilizada. Para una descripción más detalla de las características consultar[Deca2011].

1. Relación entre el consumo medio y el promedio de los consumos de los últimos 3 meses.
2. Relación entre el consumo medio y el promedio de los consumos de los últimos 6 meses.
3. Relación entre el consumo medio y el promedio de los consumos de los últimos 12 meses.
4. Norma de la diferencia entre del consumo real y la proyección del consumo.
5. Diferencia entre el primer coeficiente de Fourier del último año y el de los años anteriores.
6. Diferencia entre el segundo coeficiente de Fourier del último año y el de los años anteriores.
7. Diferencia entre el tercer coeficiente de Fourier del último año y el de los años anteriores.
8. Característica extraída de los coeficientes de Wavelet
9. Característica extraída de los coeficientes de Wavelet
10. Característica extraída de los coeficientes de Wavelet
11. Característica extraída de los coeficientes de Wavelet
12. Característica extraída de los coeficientes de Wavelet
13. Característica extraída de los coeficientes de Wavelet
14. Característica extraída de los coeficientes de Wavelet
15. Diferencia entre el primer coeficiente del polinomio de grado cuatro que mejor se ajusta los consumos del último año y el de los años anteriores.

16. Diferencia entre el segundo coeficiente del polinomio de grado cuatro que mejor se ajusta los consumos del último año y el de los años anteriores.
17. Diferencia entre el tercer coeficiente del polinomio de grado cuatro que mejor se ajusta los consumos del último año y el de los años anteriores.
18. Diferencia entre el cuarto coeficiente del polinomio de grado cuatro que mejor se ajusta los consumos del último año y el de los años anteriores.
19. Diferencia entre el quinto coeficiente del polinomio de grado cuatro que mejor se ajusta los consumos del último año y el de los años anteriores.
20. Norma de la diferencia mes a mes del consumo del cliente y el consumo promedio de todos los clientes.
21. Cociente entre la varianza del consumo del último año respecto a los años anteriores.
22. Cociente entre la varianza del consumo del último año respecto a la varianza promedio de todos los consumos.
23. Norma del primer coeficiente de Fourier.
24. Norma del segundo coeficiente de Fourier.
25. Norma del tercer coeficiente de Fourier.
26. Norma del cuarto coeficiente de Fourier.
27. Norma del quinto coeficiente de Fourier.
28. Pendiente de la recta que mejor se ajusta a la curva de consumos.

Características no extraídas de los consumos:

En [CSIC-UTE2014] se tuvieron en cuentas otras características no extraídas de los registros de consumo de los clientes, que conforman la *Base 6+*. Estas características son:

29. Proporción de lecturas reales (real).
30. Potencia Contratada (entero).
31. Cantidad de irregularidades (entero).
32. Días de última inspección (entero).
33. Días de renovado (entero).
34. Mora (entero).

Cabe destacar que se tiene un gran número de características, por lo cual es necesario realizar un proceso de selección y/o extracción de características para determinar cuáles son las que maximizan el desempeño de los clasificadores.

3. Marco teórico

3.1. Random Forest

El método Random Forest fue propuesto en el 2001 por Breiman en el paper [Breiman2001]. En el mismo se propone explotar la idea de que la combinación de varios clasificadores tiene un mejor desempeño que cada uno individualmente. Esta misma idea es la que utilizan los métodos de boosting y bagging.

En el caso de Random Forest se combinan los resultados de varios árboles de decisión. Para cada nueva muestras que se desea clasificar, primero se obtiene el resultado de clasificación de cada uno de los árboles y luego se le asigna la muestra la clase mayoritaria dentro de los resultados obtenidos.

Supongamos se dispone de un conjunto de datos con N muestras de entrenamiento y M características, el proceso de construcción de cada árbol es el siguiente:

1. Se toman N muestras al azar con remplazamiento del conjunto de entrenamiento.
2. En cada nodo se toman m características ($m < M$) al azar. Dentro de este subconjunto de características se elige para realizar la siguiente partición aquella que reduce en mayor medida la impureza. Para medir la impureza se utiliza la función de Gini.
3. Cada árbol se construye de forma maximal y no se utilizan algoritmos de poda.

Este método presenta las siguientes ventajas. Como se utilizan árboles de decisión es sencillo utilizar características nominales lo cual va a ser de utilidad para incorporar las nuevas características no asociadas a los consumos (*Base 6+*). Por otro lado, Random Forest al promediar los resultados de un gran número de clasificadores no presenta sobre-entrenamiento [Breiman2001]. Además en la práctica este método a demostrado tener muy buen desempeño.

Sin embargo, este método pierde una de las ventajas que presenta los árboles de decisión que es la fácil interpretación de cómo es el proceso de clasificación. Por otro lado también se tiene un costo computacional mayor a la hora de realizar la clasificación, dado que es necesario evaluar las muestras con múltiples árboles.

En Random Forest es necesario elegir convenientemente el valor de dos parámetros, la cantidad m de característica a sortear en cada nodo y la cantidad de árboles a utilizar.

A la hora de elegir la cantidad de características que se sortean en cada nodo del árbol hay que tener en consideración dos aspectos importantes. El primer aspecto es la fuerza de cada árbol, la cual esta asociado a qué tan buen clasificador es cada árbol de manera independiente. Cuanto mayor es el número de características utilizadas en cada nodo mayor va a ser la información que se dispone en la construcción de cada árbol y por lo tanto la fuerza de los árboles aumenta. Por lo que cuanto mayor es la fuerza de los árboles mejor es el desempeño de Random Forest. Por otro lado si la cantidad de características utilizada es grande los árboles están muy correlacionados entre sí, porque estos son construidos con prácticamente la misma información. Y en la práctica se ha observado que cuanto mayor es la correlación entren los árboles peor es el desempeño del clasificador [Oshiro2012]. Por lo tanto a la hora de elegir la cantidad de características a sortear en cada nodo existe un compromiso entre las fuerza de ls árboles y la correlación de los mismos la cual debe ser tenida en consideración. En el trabajo original Breiman propone utilizar $m = log_2(M)$ [Breiman2001].

En la práctica se ha observado que cuanto mayor es el número de árboles utilizados mejor es el desempeño del clasificador. Pero al aumentar la cantidad de árboles aumenta el costo computacional de entrenamiento del clasificador y también el de clasificación de cada muestras [Oshiro2012]. Por lo cual a la hora de elegir el número árboles existe un compromiso entre cuánto mejora el desempeño con cada árbol que se agrega y cual es el costo computacional que esto implica. En

el caso de tener recursos computacionales suficientes, es conveniente tomar el valor para el cual continuar agregando árboles no presenta un aumento significativo en el desempeño del clasificador.

3.2. Clases desbalanceadas

Se dice que un problema de clasificación es desbalanceado, cuando se tiene una gran diferencia entre la cantidad de muestras pertenecientes a cada clase . En un problema de dos clases el desbalance implica que hay un *clase mayoritaria* y una *clase minoritaria*, y por lo general la clase minoritaria es la clase que se desea detectar.

En el problema de detección de consumos fraudulentos, existe un gran desbalance de clases dado que los clientes fraudulentos son muchos menos que los no fraudulentos. Para determinar si un cliente es fraudulento es necesario realizar una inspección de su instalación por lo cual la chance de obtener muestras fraudulentas está sesgada a la efectividad de la selección de clientes a inspeccionar. Por esta razón UTE desea tener una herramienta automatizada que permita detectar clientes en condición de conexión anómala para detectar la mayor cantidad de fraudes con la menor cantidad de inspecciones posible. Como notación se consideró como positivas los clientes fraudulentos, ya que son los que se desean detectar.

En la base de datos *BASE 6* se tienen 1864 consumos normales y 194 fraudulentos, por lo que se observa que la clase mayoritaria es casi 10 veces más frecuente que la minoritaria.

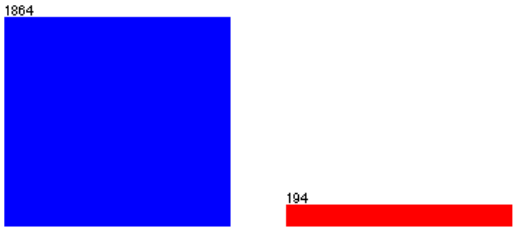


Figura 2: Distribución de consumos fraudulentos (en rojo) y consumos normales (en azul) en la base de datos.

3.2.1. Medidores de desempeño

Existen distintos medidores desempeño de un clasificador, en el caso de presentarse problemas desbalanceados se debe tener ciertos recaudos a la hora de seleccionarlos.

En este caso no es conveniente utilizar como medida del desempño la tasa de error del clasificador. Para fijar ideas supongamos el siguiente ejemplo; se tiene un desbalance tal que la clase minoritaria tiene 5 % del total, un clasificador que clasifique a todas las muestras como normales tendrá un desempeño del 95 %. Si bien podría decirse que el clasificador tiene una baja tasa de error no se está detectando a ninguna muestra positiva lo cual en nuestro caso es lo que se desea. En cambio otro clasificador que tenga un 90 % de tasa de acierto puede estar detectando muestras positivas, si se guiase sólo por la tasa de acierto se estaría eligiendo el clasificador que no detecta lo que se desea.

Por esta razón se tuvieron en consideración otras medidas de desempeño de los clasificadores. Previa a la definición de los medidores es necesario definir los siguientes términos:

- *TP* o 'true positive' refiere a las muestras que fueron clasificadas como fraudulentas cuando efectivamente su etiqueta indica que son fraude.
- *TN* o 'true negative' refiere a las muestras que fueron clasificadas como no fraudulentas cuando su etiqueta indica que son clientes con una consumo normal.

- *FP* o 'false positive' refiere a las muestras que fueron clasificadas como fraudulentas cuando su etiqueta indica que son son clientes con una consumo normal.
- *FN* o 'true negative' refiere a las muestras que fueron clasificadas como no fraudulentas cuando su etiqueta indica fraude.

En función de lo anterior se definen los siguientes medidores:

$$Recall^p = \frac{TP}{TP + FN} \tag{1}$$

El *Recall* es una medida de cuantos del total de clientes fraudulentos son detectados por el clasificador. En algunas literaturas se puede encontrar con el nombre de sensibilidad.

$$Precision^p = \frac{TP}{TP + FP} \tag{2}$$

El *Precision* es una medida de pureza en mi clasificación, en otras palabras es la porción de los clientes etiquetados como fraudulentos que efectivamente lo son.

$$F - value^p = \frac{(1 + \beta^2) \times Recall \times Precision}{\beta^2 \times Recall + Precision} \tag{3}$$

El *F-value* integra ambos medidores anteriores, variando β se modifica el peso que se le da a cada uno de los medidores. Un β grande da más peso al *Precision* y un β pequeño le da más peso al *Recall*. En este caso se utilizó $\beta = 1$ con el fin de darle el mismo peso al *recall* y al *precision*.

En particular a partir de las definiciones de los términos anteriores se pueden obtener los medidores habituales, el porcentaje de acierto y de error.

$$acc = \frac{(TP + TN)}{TP + TN + FP + FN} \tag{4}$$

$$err = \frac{(FP + FN)}{TP + TN + FP + FN} \tag{5}$$

Una forma complementaria de presentar los resultados es la matriz de confusión, en la cual se aprecia claramente el error que se está cometiendo. En este trabajo se utilizó la siguiente notación para las clases y la asignación de etiquetas:

- ω_0 clase de consumos normales.
- ω_1 clase de consumos fraudulentos.
- α_0 asignar la etiqueta de normal a un patrón.
- α_1 asignar la etiqueta de fraudulento a un patrón.

Bajo esta notación la matriz de confusión se compone de la siguiente manera:

$\omega \vert \alpha$	α_0	α_1
ω_0	TN	FP
ω_1	FN	TP

(6)

3.2.2. Técnicas de balanceo

Existen distintos algoritmos que intentan balancear los datos con el fin de obtener clasificadores que permitan determinar la presencia de elementos de la clase minoritaria. No existe una técnica mejor que la otra por lo cual se deben probar desempeño aplicando varias de ellas.

3.2.2.1. Sobremuestreo

Las técnicas de sobremuestreo permiten balancear las clases realizando un sobremuestreo de las muestras de la clase minoritaria.

Existen algoritmos de sobremuestreo en los cuales simplemente se replican muestras de la clase minoritaria. Esto permite obtener clases más balanceadas, pero hace que las fronteras de decisión se centren alrededor de las muestras generadas, y por lo tanto se corre un mayor riesgo de sobre-entrenamiento.

Otra técnica muy popular de sobremuestreo, la cual es la que se utilizó ne este trabajo es SMOTE (*Synthetic Minority Over Sampling Technique*. Esta es una herramienta que realiza el sobremuestreo creando muestras sintéticas de la clase minoritaria. Las nuevas muestras se obtienen a partir de interpolación de las muestras originales, de esta manera se asegura que los casos sintéticos sean cercanos a los originales, y se reduce en cierta el riesgo de sobre-entrenamiento.

3.2.2.2. Matriz de costos

El método de la matriz costos consiste en asignar un costo más alto a clasificar una muestra positiva como negativa. Es decir se premian las muestras de la clase minoritaria a fin de que las mismas tengan un mayor peso. Este método es muy utilizado y ha demostrados tener un buen desempeño en problemas de dos clases.

En el artículo [ChaoChen] se propone el método *Weighted Random Forest* para mejorar el desempeño del clasificador ante clases desbalanceadas. Este consiste en aplicar costos en la etapa de aprendizaje, de manera de beneficiar la detección de elementos de la clase minoritaria. En el paper se indica que los costos se pueden aplicar en dos momentos, en la etapa de construcción de los árboles (en el caso de RandomForest ponderar los costos en la medida de impureza de los nodos por Gini) o luego de tener todos los árboles construidos premiar a los arboles que clasifiquen a un patrón como perteneciente a la clase minoritaria.

El costo se expresa en forma matricial, utilizando la siguiente notación.

- $\lambda(\alpha_1|\omega_0) = \lambda_{FP}$ costo de asignar la etiqueta de fraudulento a una muestra normal.
- $\lambda(\alpha_0|\omega_1) = \lambda_{FN}$ costo de asignar la etiqueta de normal a una muestra fraudulenta.

Sea Λ la matriz de costos la misma se define de la siguiente manera:

$$\Lambda = \begin{pmatrix} 0 & \lambda_{FP} \\ \lambda_{FN} & 0 \end{pmatrix} \tag{7}$$

Se asume que se tiene costo cero realizar la asignación de clase de manera acertada. En este caso si se desea que un clasificador tenga mayor tendencia a clasificar una muestra como positiva, se debe subir el costo de asignar la etiqueta de normal a una muestra fraudulenta, es decir λ_{FN}.

4. Entrenamiento del clasificador

En esta sección se desea mostrar cómo fue el proceso de entrenamiento del clasificador seguido en el proyecto, de manera de obtener el mejor clasificador posible. También se busca analizar cómo es la evolución del clasificador durante el proceso de entrenamiento. Para ello se evaluó el desempeño de Random Forest en las siguientes instancias:

1. Clasificación sin selección de características.
2. Selección de características, para ellos se evaluaron las siguientes técnicas:
   - *PCA*
   - *CsfSubsetEval*
   - *Wrapper*
3. Aplicación de técnicas de balanceo, la técnicas utilizadas fueron:
   - *SMOTE*
   - *Matriz de costos*
4. Desnormalización de los datos
5. Utilización de nuevas características

Para los resultados que se muestran en este sección se utilizaron los medidores de desempeño *recall*, *precision* y *F-value* (con $\beta = 1$) para la clase positiva definidos en la sección 3.2. La decisión de adoptar β unitario hace que se de la misma importancia al *recall* y al *precision*. También en todos los casos se realizó validación cruzada con 10-folds.

4.1. Entrenamiento de clasificadores sin selección de características

En una primera instancia se analizó el desempeño de los clasificadores sin utilizar técnicas de selección ni extracción de características. Esto nos dió una primera aproximación al clasificador y un punto de partida para evaluar cómo mejora el desempeño del clasificador durante el entrenamiento.

Para el diseño de un clasificador *Random Forest* es necesario determinar:

- N: cantidad de árboles a utilizar.
- m: cantidad de muestras que se toman al azar en cada nodo.

Según lo analizado en la sección 3.1 es conveniente tomar una gran cantidad de árboles de decisión cuidando que el costo computacional no sea excesivo. Tomando esto en consideración se optó tomar $N = 200$.

Luego se evaluó cómo varía el desempeño del clasificador al variar m la cantidad de muestras a tomar en cada nodo. En la tabla 1 se pueden observar los resultados de este proceso. Cabe destacar que los resultados mostrados corresponden a un valor promedio valores de *recall*, *precision* y *F-value* obtenidos en las distintas repeticiones. De acuerdo a los criterios de evaluación establecidos, deseamos maximizar el *F-value* por lo que el mejor desempeño se obtuvo con $m = 10$ características.

m	$Precision$	$Recall$	$F - value$
5	33,83 %	3,29 %	5,74 %
6	34,00 %	3,62 %	6,28 %
7	39,14 %	3,90 %	6,65 %
8	38,67 %	3,60 %	6,27 %
9	37,50 %	4,49 %	7,60 %
10	**48,01** %	**5,11** %	**8,07** %
11	37,33 %	4,21 %	7,28 %
12	31,31 %	3,60 %	6,02 %
13	40,48 %	4,80 %	8,13 %
14	41,67 %	4,82 %	8,30 %
15	38,07 %	4,10 %	7,10 %

Cuadro 1: Resultados para RandomForest con 200 árboles variando la cantidad de características

Para complementar el resultado obtenido se incluye la matriz de confusión para este valor de m para una de las corridas:

$\omega \vert \alpha$	α_0	α_1
ω_0	1367	9
ω_1	160	7

Cabe destacar que el desempeño del clasificador es muy bajo. Observando la matriz de confusión y los valores de *recall* y *precision*, se pude observar que ninguno de los dos valores son buenos. En primer lugar el *precision* obtenido es en promedio 48 % lo cual indica que de menos de la mitad de la muestras etiquetadas como fraude efectivamente los son.

Por otro lado también se observa que un porcentaje muy chico de los consumos que son fraudulentos, lo cual se traduce en un bajo valor de *recall*. La razón de esto es el gran desbalance entre las clases, lo cual lleva a que la mayoría de los arboles del clasificador asignan a las muestras la clase mayoritaria.

Para mejorar el desempeño del clasificador es necesario reducir la dimensionalidad del problema, aplicando algún método de selección de caracterísiticas y aplicando algún método de balanceo de clases para mejorar el *recall* del clasificador.

4.2. Selección y extracción de características

Si bien a priori podría pensarse que cuantas más características se utilicen mejor será el desempeño del proceso de clasificación, porque que se dispone de más información, esto no sucede en la práctica. Se ha observado que para obtener un buen desempeño la cantidad de características no debe ser comparable con la cantidad de muestras de cada clase. Una regla de buena práctica que se utiliza habitualmente es la relación $\frac{n}{d} > 10$, donde d es la cantidad de características y n la cantidad de muestras de la clase.

En este problema se dispone de 28 características, y en se tiene 167 instancias de la clase minoritaria por lo que la relación es:

$$\frac{n}{d} = \frac{167}{28} = 5,96$$

Por lo cual se puede decir que la cantidad de características utilizadas es excesiva. Esto produce sobre-entrenamiento de los clasificadores y por lo tanto compromete el desempeño de los mismos. Por esta razón es conveniente realizar un proceso de selección/extracción de características para reducir la dimensionalidad del problema.

4.2.1. Análisis de componentes principales (PCA)

En primer lugar se aplicó el método de extracción de características de análisis de componentes principales (PCA) con el conjunto de referencia. Este método busca las direcciones del espacio en la cual los datos presentan mayor varianza. Una forma de evaluar cuantas componentes se deben tomar es ver cómo se distribuye la varianza acumulada en función de la cantidad de componentes.

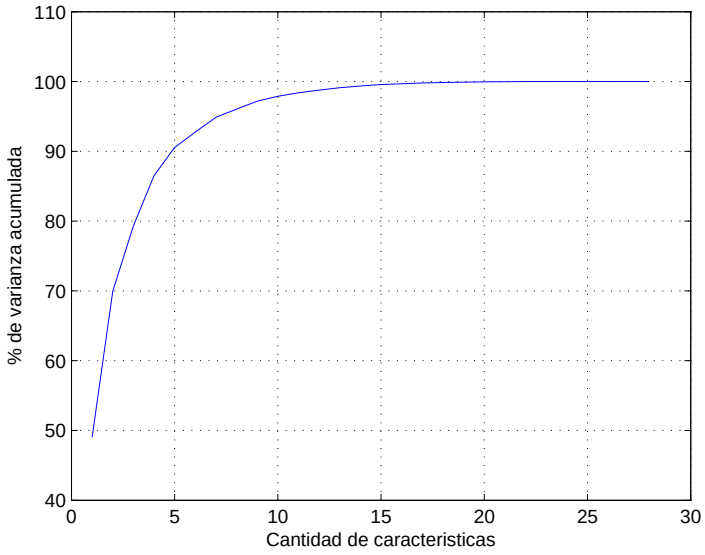


Figura 3: Varianza acumulada con PCA

En la figura 3 se observa que en las primera 10 componentes principales se acumula más del 95 % de la varianza, por lo cual se tomó este número como la cantidad máxima de componentes a considerar. Para evaluar cómo son las características obtenidas se realizó una gráfica de cómo se distribuyen las muestras en función de las primeras tres componentes, ver figura 4. Se observa que las muestras de las distintas clases se encuentran muy 'entreveradas' entre sí. Por lo cual a

priori no se espera que el método de extracción PCA ofrezca grandes mejoras en el desempeño de los clasificadores.

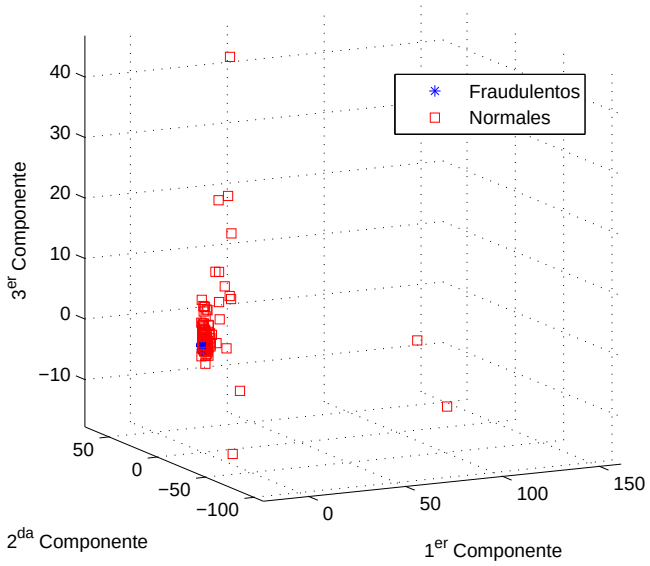


Figura 4: Tres características principales del PCA

Para determinar cuál es el impacto de esta técnica de extracción de características en el clasificador *Random Forest* se evaluó el desempeño del mismo variando la cantidad de componentes PCA (tomando como máximo 10 componentes) y el valor de m. Para ello se realizó validación cruzada con 10 folds.

Con el fin de no saturar el texto con resultados, se adjuntan sólo los mejores resultados para las 3, 5 y 10 primeras componentes de PCA.

Comp. PCA	m	*Precision*	*Recall*	$F - value$
3	3	$30,83\,\%$	$3,96\,\%$	$6,85\,\%$
5	3	$19,17\,\%$	$2,24\,\%$	$3,95\,\%$
10	9	$16,78\,\%$	$2,39\,\%$	$4,11\,\%$

Tal como se puede observar en la tabla, el mejor resultado se obtuvo tomando las primeras 3 componentes principales y tomando $m = 3$. El desempeño fue inferior al obtenido sin selección de características, el que se encuentra en la tabla 1. Por lo cual se concluye que el método de extracción de características PCA no resulta efectivo en la mejora del desempeño de *Random Forest*.

4.2.2. Criterio de evaluación de subgrupos basado en correlación (CfsSubsetEval)

El criterio de correlación busca encontrar el subconjunto de características que presenta una mayor correlación entre las clases y que a la vez las muestras del subconjunto estén lo menos correlacionadas entre sí posible. Para ellos busca el subgrupo que maximiza la siguiente función de mérito:

$$J_{cor}(S) = \frac{\sum_i U(x_j, C)}{\sqrt{\sum_i \sum_i U(x_j, x_i)}} \quad \forall i, j \in S \quad con \quad U(X,Y) = \frac{2\,(H(X) - H(X|Y))}{H(X) + H(Y)} \tag{8}$$

El denominador representa la correlación media entre las características de S y las clases, mientras que el denominador es la intercorrelación entre las características de S.

Existen diversas maneras de buscar cuáles son los mejores subconjuntos. En primer lugar está la búsqueda exhaustiva en la cual se evalúa el desempeño de todos los subconjuntos de tamaño d y se selecciona el mejor de todos. Este es el método de búsqueda óptimo porque asegura que se obtiene el mejor subconjunto. Sin embargo, este método de búsqueda presenta un gran costo computacional por lo cual no siempre puede ser utilizado.

En este caso el costo computacional de evaluar cada subconjunto es pequeño dado que este es un método de filtrado, por lo cual es posible realizar una búsqueda exhaustiva. De esta manera se obtuvo un conjunto de 6 características, las características seleccionadas son:

Método	Características
CfsSubsetEval	[1 2 3 18 20 25]

Cuadro 2: Características elegidas por *CfsSubsetEval*

A continuación se evalúa el desempeño de los clasificadores con este conjunto reducido de características. De los resultados obtenidos en la tabla 3 el mejor fue para 5 características tomadas

m	*Precision*	*Recall*	$F - value$
1	$45,56\,\%$	$6,38\,\%$	$11,0\,\%$
2	$43,17\,\%$	$6,59\,\%$	$11,14\,\%$
3	$51,78\,\%$	$7,60\,\%$	$12,8\,\%$
4	$46,78\,\%$	$7,60\,\%$	$12,63\,\%$
5	**45,17** %	**7,82** %	**12,81** %
6	$4,50\,\%$	$7,61\,\%$	$12,59\,\%$

Cuadro 3: Resultados de *Random Forest* con *CfsSubsetEval* para distinta cantidad de características

para el proceso de construcción del árbol. Este valor de $F - value$ es superior al obtenido sin selección de características. Por lo cual se considera que la elección de características mejoran el desempeño del clasificador.

4.2.3. Wrapper

El método Wrapper es una técnica de selección de características de encapsulado. Este es un método supervisado en el que se evalúa cómo es el desempeño de cada subconjunto de características con un determinado clasificador.

Este método presenta la ventaja de que toma en consideración cuál es el clasificador que se va a utilizar luego en el proceso de clasificación. Por lo tanto permite hallar el mejor subconjunto de características para cada clasificador en particular.

Sin embargo, este método presenta como gran desventaja su costo computacional.En Wrapper se debe evaluar el desempeño del clasificador para cada subconjunto de características a considerar, lo cual hace que el costo computacional del mismo sea muy elevado. Esto imposibilita la búsqueda exhaustiva del mejor subconjunto de características, siendo necesario realizar búsquedas sub-óptimas las cuales no garantizan la obtención del mejor subconjunto. En este caso se utilizó como criterio de búsqueda el método Best First.

4.2.3.1. Wrapper - accuracy

El algoritmo Wrapper de Weka por defecto busca maximizar el accuracy del clasificador. En una primera instancia se probó este algortimo para observar cuál era el resultado del mismo. El

clasificador que se utilizó par evaluar el desempeño fue Random Forest con $N = 200$ y $m = 1$. El subconjunto de características obtenido consta de 6 características las cuales se encuentran en la tabla 4.

Método	Características
Wrapper -accuracy	[2 3 6 20 22 23]

Cuadro 4: Características elegidas por *Wrapper-accuracy*

Se evaluó el desempeño de *Random Forest* con el subconjunto de caracterísiticas obtenido.

m	Precision	Recall	F-value
1	$26,67\,\%$	$2,21\,\%$	$4,04\,\%$
2	$30,28\,\%$	$3,03\,\%$	$5,39\,\%$
3	$31,67\,\%$	$3,62\,\%$	$6,37\,\%$
4	$31,89\,\%$	$4,23\,\%$	$7,33\,\%$
5	$33,00\,\%$	$4,23\,\%$	$7,32\,\%$
6	**33,83** %	**4,82** %	**8,25** %

Cuadro 5: Resultados de *Random Forest* con *Wrapper -accuracy* para distinta cantidad de características

En la tabla 5 se puede ver que el desempeño de *Random Forest* con este subconjunto de características es levemente superior al obtenido sin selección de características pero inferior al obtenido con el método de selección CfsSubsetEval. Una razón que puede explicar esto es que el método Wrapper buscó maximizar el *accuracy* que no es lo que se desea maximizar en el clasificador final. Por esta razón fue necesario buscar la manera de poder realizar el método Wrapper maximizando el *F-value*.

4.2.3.2. Wrapper - *F-value*

Una de las posibles maneras de realizar la selección de características con *Wrapper* maximizando el *F-value* es combinando en Weka el método *Wrapper* con *ThresholdSelector* el cual permite seleccionar el *accuracy* como medida de desempeño.

Utilizando como clasificador a *Random Forest* con $m = 1$ y $N = 200$ se obtuvo el siguiente subconjunto de características:

Método	Características
Wrapper	[1 2 5 6 7 19 28]

Cuadro 6: Características elegidas por *Wrapper -F-value*

En los resultados obtenidos con este clasificadores, los cuales se encuentran en la tabla 7, se puede ver que al utilizar *F-value* para buscar el evaluar *Wrapper* se obtiene un **F-value** superior. Sin embargo el desempeño obtenido continúa siendo inferior al obtenido con *CfsSubsetEval*, lo cual no es lo que se esperaba.

A priori se esperaba que el subconjunto obtenido con *Wrapper* maximizando el *F-value* fuese mejor que el obtenido con *CfsSubsetEval*. Porque en el proceso de selección de características con *CfsSubsetEval* no tiene en cuenta el clasificador que se va a utilizar ni tampoco toma en consideración la etiqueta de las muestras.

Existen dos posibles explicaciones para eso, o bien la combinación en Weka de *Wrapper*+*ThresholdSelector* no fue bien aplicada. O bien el resultado no fue el óptimo porque en el proceso de selección de

m	Precision	Recall	F-value
1	$24,17\,\%$	$1,99\,\%$	$3,61\,\%$
2	$35,56\,\%$	$4,18\,\%$	$7,32\,\%$
3	$35,00\,\%$	$4,78\,\%$	$8,16\,\%$
4	$40,12\,\%$	$5,80\,\%$	$9,77\,\%$
5	$\mathbf{43,29}\,\%$	$\mathbf{6,78}\,\%$	$\mathbf{11,44}\,\%$
6	$38,67\,\%$	$6,40\,\%$	$10,68\,\%$

Cuadro 7: Resultados de *Random Forest* con *Wrapper -F-value* para distinta cantidad de características

características no se utilizó el mismo clasificador que en el proceso de clasificación (se utilizó otro valor de m).

Dado que el método de selección *Wrapper* no ofreció un mejor desempeño que *CfsSubsetEval* y el costo computacional del mismo es muy alto, se optó por utilizar de aquí en más el método *CfsSubsetEval*. [1]

4.2.4. Evaluación del criterio de elección de la cantidad de arboles

En esta etapa se decidió evaluar si la cantidad de árboles utilizada para la clasificación es la adecuada. De acuerdo a lo visto en la sección 3.1 el desempeño mejora con la cantidad de arboles a costa de un aumento del costo computacional. Por esta razón se realizó la gráfica de la figura 5 en la cual se aprecia que con los 200 arboles que se están manejando se está en la zona en la que el *F-value*se mantiene constante. Si bien se parece obtenerse el mismo resultado con una menor cantidad de arboles, por ejemplo 150, se tomó como medida conservadora continuar entrenando con esta cantidad de arboles $N = 200$.

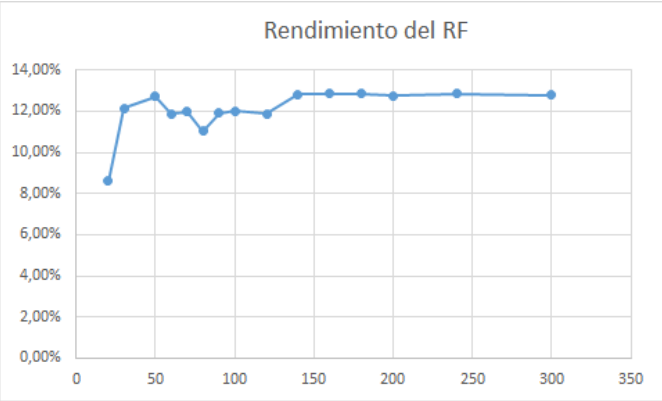


Figura 5: *F-value* en función de cantidad de arboles

[1] Esta decisión fue tomada teniendo en cuenta que el tiempo para realizar Wrapper era excesivo y no se pudo utilizar en el marco de este trabajo. Esto no quiere decir que se descarte este método para próximos estudios.

4.3. Técnicas de balanceo

Si se analizan los resultados obtenidos hasta el momento se observa que el término que más está afectando al *F-value* es el *recall*, el cual es está por debajo del 10 %. Esto quiere decir que se está detectando un porcentaje bajo de muestras fraudulentas. Una manera de obtener un mejor *recall* es aplicado técnicas de balanceo.

De acuerdo a lo visto en la sección 3.2 se utilizaron *SMOTE* y matrices de *costo* para mejorar el desempeño de los clasificadores. Ambas técnicas se aplicaron sobre el conjunto de características con las que se obtuvo el mejor resultado, es decir las que se encuentran en la tabla 2.

4.3.1. SMOTE

El conjunto inicial posee 1376 muestras de consumos normales (clase ω_0) y 167 muestras de fraudulentos (clase ω_1), por lo cual el porcentaje de desbalance está dado por:

$$\%desbalance = \frac{1376 - 167}{167} \times 100\,\% = 724\,\%$$

Con el algoritmo se buscó reducir esta brecha tomando distintos factores de crecimiento para la clase minoritaria.

En una primera instancia se probó aplicar SMOTE a la clase minoritaria de manera de aumentar la cantidad de muestras en un 700 % y luego realizar validación cruzada con estas muestras. De esta manera se obtuvo un $F - value^P = 88\,\%$ el cual es un resulta muy alentador. Sin embargo luego de analizar el procedimiento utilizado se observó que se estaba aplicando SMOTE sobre todo el conjunto de entrenamiento, y luego al realizar validación cruzada se tenían muestras sintéticas en el fold que se utiliza como test, lo cual no es correcto.

La manera correcta de aplicar SMOTE es aplicarlo solamente en conjunto con el cual se entrena el clasificador, y dejar incambiado el conjunto con el cual se valida el clasificador. Para poder tener un conjunto de prueba con muestras puras, se aplicó el algoritmo *meta.FilteredClassifier* de Weka, utilizando como método de filtrado el *SMOTE* y clasificador el *Random Forest*. El algoritmo realiza validación cruzada (se eligieron nuevamente 10-folds) aplicando el *SMOTE* al conjunto de entrenamiento y evaluando con un conjunto de test puro.

Para evaluar esta técnica de sobremuestreo se varió tanto el valor de m como la tasa de aumento de la clase minoritaria de *SMOTE*, tomando como valor límite el que produce que las clases tengan la misma cantidad de muestras.

m	$\%SMOTE$	$Precision^P$	$Recall^P$	$F - value^P$
6	300	23,8 %	25,4 %	24,3 %
4	500	21,5 %	31,2 %	25,3 %
4	**600**	**21,2** %	**34,4** %	**26,0** %
6	720	21,0 %	34,9 %	25,9 %

Cuadro 8: Resultados de *Random Forest* en validación cruzada con SMOTE

Los resultados obtenidos en los experimentos se pueden observar en la tabla 8. Los mejores resultados obtenidos fueron con un aumento del 600 % de la clase minoritaria en *SMOTE*, El valor obtenido supera el mejor resultado hasta el momento con *CfsSubsetEval*, se observa que tal como era de esperar el valor del *recall* sube considerablemente a costa de una disminución del *precision*. Por lo cual se puede decir que *SMOTE* resulta efectivo para elevar el desempeño de *Random Forest*.

4.3.2. Matriz de costos

Otra técnica que permite aumentar el *recall* del clasificador es asociar un mayor costo a clasificar las muestras fraudulentas como normales. Para esto se utilizó el algoritmo meta de Weka *CostSensitiveClasifier* el cual permite ingresar la matriz de costos de acuerdo a lo visto en la sección 3.2. Este es uno de los métodos sugeridos por [ChaoChen], se tomó como clasificador *Random Forest* con $m = 5$ y $N = 200$, utilizando las características que al momento han brindado mejor desempeño que fueron las de *CfsSubsetEval*.

Costo	$Precision^P$	$Recall^P$	$F - value^P$
4	18,88 %	33,64 %	24,08 %
5	18,99 %	42,56 %	26,17 %
7	17,58 %	52,76 %	26,34 %
10	16,61 %	67,10 %	26,59 %
10,5	**17,00 %**	**70,91 %**	**27,37 %**
11	16,53 %	71,89 %	26,86 %
11,5	16,74 %	72,32 %	27,14 %
12	16,40 %	74,32 %	26,83 %
13	16,09 %	75,50 %	26,50 %
14	15,78 %	77,28 %	26,18 %
17	15,04 %	79,36 %	25,28 %
20	14,35 %	81,75 %	24,40 %
30	13,57 %	87,68 %	23,49 %
40	12,88 %	90,97 %	22,56 %
50	12,72 %	93,99 %	22,40 %
70	12,22 %	95,18 %	21,65 %
100	12,01 %	97,89 %	21,38 %

Cuadro 9: Resultados de *Random Forest* con características de *CfsSubsetEval* y costos

En la tabla 9 se tienen los principales resultados obtenidos con este método. Se observa que los mejores resultado se obtuvieron aplicando un costo de 10,5.

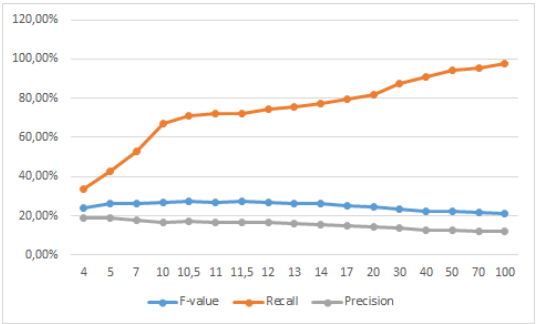


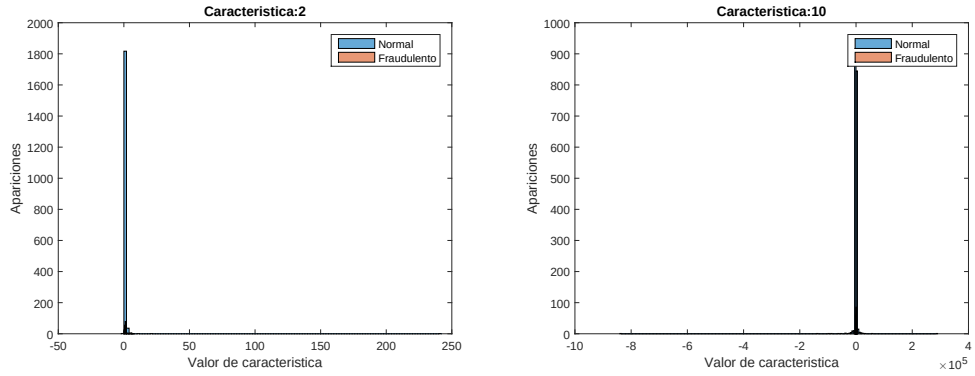
Figura 6: Resultados obtenidos al variar el costo con *Random Forest*

Para comprender de mejor manera cuál es el efecto de aplicar costos en el proceso de clasificación se construyó la gráfica de la figura 9. Se puede observar que el *recall* aumenta a medida que el costo aumenta, esto es lógico si se recuerda cómo es la definición del *recall* porque al aumentar el costo se tiene una mayor tendencia a clasificar las muestras como fraudulentas. Esta misma tendencia hace que también existan más muestras normales que son clasificadas como fraudulentas, lo cual hace disminuir el *precision*. Por lo cual se concluye que al aumentar el costo se aumenta el *recall* y se disminuye el *precision*, y por tanto el costo se debe elegir buscando un compromiso entre ambos.

4.4. Entrenamiento de clasificadores sin normalizar

El algoritmo implementado en [Deca2011], el cual extrae la características de las curvas de consumo, las devuelve normalizadas. Cuando se clasifica con arboles no es necesario realizar un proceso de normalización de los datos por lo cual se procedió a evaluar el desempeño con las características sin normalizar.

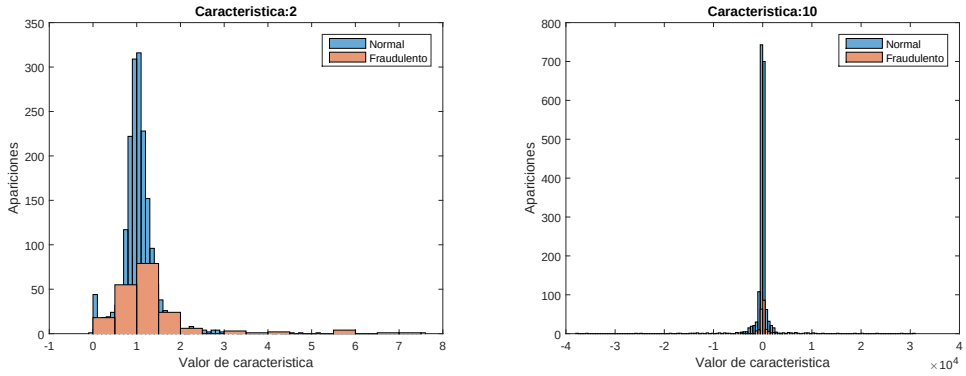
En este proceso se encontró que en los histogramas existían valores extremos muy alejados de la media. Por lo cual se procedió a estudiar los mismos a fin de determinar si estos valores son posibles outliers y así eliminarlos de los conjuntos de muestras.



(a) Problema con el máximo (b) Problema con el mínimo

Figura 7: Histograma con distribución de muestras por característica

En la figura 7 se aprecian dos casos en los cuales al tener un valor muy alejado el histograma queda alargado, al eliminar estos valores se obtuvieron histogramas más definidos como se puede observar en la figura 8. En total se extrajeron 8 muestras todas ellas pertenecientes a la clase mayoritaria.



(a) Problema con el máximo (b) Problema con el mínimo

Figura 8: Histograma con distribución de muestras por característica

Selección de características

Se realizó nuevamente el proceso de selección de características con el algoritmo *CfsSubsetEval* con el cual se obtuvieron las siguientes características.

Método	Características
CfsSubsetEval	[1 2 3 6 20 22 25]

Cuadro 10: Características elegidas por *CfsSubsetEval* para datos no normalizados

Utilizando estas características se obtuvo el resultado de la tabla 11. En cuanto a esta evaluación la normalización no afectó de manera significativa el desempeño del clasificador.

Datos	m	$F - value$
Normalizado	5	12,81 %
Sin normalizar	5	12,17 %

Cuadro 11: Mejor resultado con *CfsSubsetEval*

SMOTE

Una de las técnicas utilizadas para tratar con el desbalanceo fue SMOTE por lo cual se desea evaluar el impacto en el desempeño de este algoritmo con los datos sin normalizar. Los resultados se encuentran en la tabla 12, nuevamente no se encuentran diferencias tanto en el resultado final como en el porcentaje de aumento en el cual se da el máximo desempeño.

Datos	% SMOTE	$F - value$
Normalizado	600	26,00 %
Sin normalizar	570	25,92 %

Cuadro 12: Mejor resultado con *CfsSubsetEval* y SMOTE

Costos

Con la técnica de costos el mejor resultado obtenido fueron los que se encuentran en la tabla 13. En cuanto al desempeño es muy similar, la diferencia más notoria es el costo al cual se da el máximo valor de *F-value*. Para los datos normalizados fue necesario penalizar más el hecho de tener un FN.

Conclusiones de sin normalizar

Tanto los resultados como los parámetros de los clasificadores son muy similares. De acuerdo a estas pruebas realizadas se confirma que los arboles de decisión no son influenciados por la normalización de los datos.

Datos	Costo λ_{FN}	$F - value$
Normalizado	10,5	27,37 %
Sin normalizar	4,5	28,25 %

Cuadro 13: Mejor resultado con *CfsSubsetEval* y costos

4.5. Inclusión de nuevas características

Luego de terminado el proceso de entrenamiento del clasificador Random Forest utilizando las características extraídas del registro de consumo, se procedió a evaluar cuál es el impacto de incluir características no extraídas de los consumos. Dado que la adquisición de estas nuevas características requiere un trabajo de búsqueda es necesario evaluar si el impacto sobre del desempeño del clasificador justifica la inclusión de las mismas o no.

Cabe destacar que se continuó utilizando las características sin normalizar.

Selección de características

Se realizó nuevamente el proceso de selección de características con el algoritmo *CfsSubsetEval* con el cual se obtuvieron las siguientes características.

Método	Características
CfsSubsetEval	[1 3 22 31 33 34]

Cuadro 14: Características elegidas por *CfsSubsetEval* para datos con nuevas características

Se observa que el algoritmo seleccionó varias de las características no extraídas de los consumos, lo cual indica que existe una alta probabilidad de que mejoren los resultados al incluir las mismas.

Utilizando estas características se obtuvo el resultado de la tabla 15. Se observa que el desempeño del clasificador mejoró significativamente, mejorando tanto el *recall* como el *precision*. Incluso se obtuvieron mejores resultados que el clasificador final sin nuevas características el cual incluye técnicas de balanceo.

Datos	m	*Precision*	*Recall*	*F-value*
Sin nuevas características	5	45,2 %	7,8 %	12,8 %
Con nuevas características	3	79,5 %	24,0 %	36,8 %

Cuadro 15: Mejor resultado con *CfsSubsetEval* para las nuevas características

Matriz de costos

Con la técnica de costos el mejor resultado obtenido fueron los que se encuentran en la tabla 16, el desempeño con costos continúa siendo superior al sin nuevas características. Se observa que al aplicar costos disminuye el *precision* pero mejora el *recall*, por lo tanto aún así se observa una mejora en el *F-value* al utilizar la técnica de matriz de costos.

Dado que se obtuvieron resultados significativamente mejores al incluir las nuevas características, se puede concluir que es muy conveniente incluirlas en el análisis.

Datos	m	λ_{FN}	*Precision*	*Recall*	*F-value*
Sin nuevas características	5	10.5	17,00 %	70,91 %	27,37 %
Con nuevas características	3	2	54,5 %	32,9 %	41,0 %

Cuadro 16: Mejor resultado con costos para las nuevas características

5. Evaluación de los clasificadores

En esta sección se estudió cuál es el desempeño de los clasificadores obtenidos luego del proceso de entrenamiento. Si bien se realizó validación cruzada del clasificador con las muestras del conjunto de entrenamiento, es necesario validar los resultados obtenidos.

En primer lugar, se validó el mejor clasificador obtenido hasta el momento con el conjunto de muestras que fueron reservadas para test. El mejor clasificador corresponde a aquel que utiliza las nuevas características, utilizando *CfsSubsetEval* para la selección de características y aplicando la técnica de matriz de costos para balancear las clases.

Datos	*Precision*	*Recall*	*F-value*
Conjunto de entrenamiento	54,5 %	32,9 %	41,0 %
Conjunto de test	51,9 %	29,8 %	37,8 %

Cuadro 17: Mejor resultado con costos para las nuevas características

El resultado de esta evaluación se encuentra en la tabla 17, se observa que el clasificador no presenta tan buen desempeño con el conjunto de test. Sin embargo esta diferencia no es muy grande por los que puede explicarse la misma a un leve sobre-entrenamiento del clasificador.

Para poder evaluar que tan bueno es los clasificadores obtenidos, es útil comparar el mismo con otro clasificador de referencia. Para ello es posible se comparar los resultados de Random Forest con los resultados obtenidos en [CSIC-UTE2014] para la misma base utilizando el clasificador C4.5 con AdaBoost, el cual presenta característcas muy similares a Random Forest.

El clasificador C4.5 es un árbol de decisión y la técnica de Adaboost permite combinar varios clasificadores C4.5 distintos para mejorar el desempeño del conjunto. Por lo cual se puede decir que Random Forest es muy similar al clasificador C4.5 + Adaboost.

Técnica	*Precision*	*Recall*	*F-value*
C4.5 sin nuevas caract.	17 %	55 %	26 %
RF sin nuevas caract.	17 %	70 %	27 %
C4.5 con nuevas caract.	35 %	43 %	39 %
RF con nuevas caract.	35 %	30 %	38 %

Cuadro 18: Mejor resultado con costos para las nuevas características

La tabla 18 es una tabla comparativa de los resultado obtenidos con Random Forest contra arboles C4.5 con AdaBoost. En la misma se puede notar que ambas técnicas devuelven valores muy similares. Esto era de esperar porque los dos clasificadores presentan muchas características en común.

En base a la comparación realizadas, se puede concluir los resultados Random Forest obtenidos en este trabajo estuvieron a la altura de los obtenidos en [CSIC-UTE2014] pero que no se obtuvo una mejora de performance. Esto indica que el proceso de entrenamiento de los clasificadores fue adecuado.

Esto también parece indicar que independientemente de las clasificador que se utilice, por más que se entrene el mismo de la mejor manera no es posible obtener muchas mejoras, sino se hace un cambio en cómo se enfrenta el problema.

6. Clustering

Luego de estudiar cómo es el desempeño de Random Forest en el problema de detección de consumos anómalos, se llegó a la conclusión de que Random Forest tiene un desempeño similar al resto de los clasificadores utilizados en estudios anteriores. Lo cual indica que en necesario hacer un cambio de enfoque para poder obtener mejores resultados en el problema.

Uno de los factores que hace que este problema sea difícil de tratar es la gran variabilidad que presentan las muestras, tanto normales como fraudulentas. Por lo cual en este sección se pretendió analizar la posibilidad de reducir esta diversidad utilizando técnicas de clustering. Se buscó separar las muestras en distintos clusters, y luego entrenar un clasificador diferente para cada uno de los clusters. Y se evaluó cual es el impacto en la detección de consumos anómalos el tener un clasificador especializado para cada cluster.

6.1. Detección de clusters

El problema que se desea atacar con las técnicas de clustering es reducir la diversidad de consumos que existe en la base de datos. Por lo cual se buscó identificar distintos clusters dentro del conjunto de consumos normales. El procedimiento que se siguió fue el siguientes:

1. Identificar clusters dentro de las muestras normales de entrenamiento.
2. Asociar las muestras fraudulentas de entrenamiento con alguno de los clusters formados.
3. Entrenar un clasificador para cada clusters.
4. Identificar las muestras del conjunto de test con alguno de los clusters.
5. Clasificar las muestras de test con el clasificador de su respectivo cluster.

Para detectar los clusters se utilizó la técnica de clustering k-means, la cual si bien es una técnica sencilla la misma es una técnica muy utilizada y la cual devuelve resultados razonablemente buenos.

El algoritmo de k-means independientemente de cómo se distribuyan las muestras y de cómo se configure el algoritmo siempre devuelve una partición del conjunto de datos en clusters, sin importan de si es una buena partición o no. Por esta razón es necesario que luego de encontrar una posible partición en de los datos en clusters, se realice una validación de los mismos para asegurarse de que se obtuvo una buena partición.

Existen dos enfoque de cómo validar una técnica de clustering, por una lado están las de validación externa y por otro las validación interna. En la validación externa se dispone a priori de las etiquetas de clusters para cada muestra, y se compara el resultado obtenido con la técnica de clustering con estas etiquetas. Por otro lado están la validación interna en la cual sólo se utiliza la información presente en las muestras, y se basan en medir cuan compactos son los datos dentro de un cluster y cuan separados están los clusters entre sí. Dado que no se dispone de información previa de cómo son los clusters de los consumos, es necesario realizar una validación interna de los clusters.

La validación interna de clusters generalmente se realiza mediante el cálculo de un índice el cual permite comparar distintas técnicas de clustering, o una mismas técnica con distintos parámetros. Existe una gran variedad de índices, los mismos generalmente consideran tanto la compaticidad como la separabilidad entre clusters. En este caso se utilizaron los índices Calinski-Harabasz y Davies-Bouldin.

Índice Calinski-Harabasz:

Este índice toma en consideración tanto la compacticidad como la separabilidad entre los clusters. Cuanto más alto es el valor del índice mejores son los clusters obtenidos.

$$iCH = \frac{\sum_{i=1}^{k} \frac{N_i d^2(\mu_i, \mu)}{k-1}}{\sum_{i=1}^{k} \sum_{x \in \mathcal{C}_i} \frac{d^2(x, \mu_i)}{N-k}} \tag{9}$$

Donde:

- k: cantidad de clusters.
- N: cantidad total de muestras.
- N_i: cantidad de muestras en el cluster i.
- μ: centro del total de muestras.
- μ_i: centro de las muestras del cluster i.

Índice Davies-Bouldin:

El índice Davies-Bouldin se calcula con la siguiente fórmula, cuanto menor es este índice mejores son los clusters.

$$iDB = \frac{1}{k} \sum_{i=1}^{k} \underset{j,j \neq i}{max} \left[\frac{\frac{1}{N_i} \sum_{x \in \mathcal{C}_i} d(x, \mu_i) + \frac{1}{N_j} \sum_{x \in \mathcal{C}_j} d(x, \mu_j)}{d(\mu_i, \mu_j)} \right] \tag{10}$$

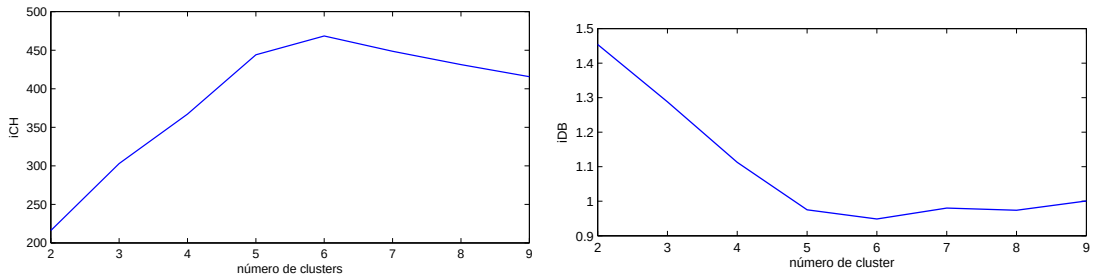
Se decidió que lo más conveniente era utilizar las características obtenidas con *CfsSubsetEval* para la base de datos con nuevas características. Sin embargo la característica número 32 es una característica nominal que presenta sólo tres valores distintos 0, 3 y 5, y como no es conveniente utilizar técnicas de clustering con características nominales se decidió realizar nuevamente el proceso de selección *CfsSubsetEval* eliminando esta característica. El resultado de esto fue el siguiente:

Método	Características
CfsSubsetEval	[1 3 22 32 33]

Lo primero a analizar para poder separar las muestras en distintos clusters es la cantidad k de clusters a utilizar. Como no se dispone de información previa de cuántos clusters hay en los datos, se procedió a aplicar el algoritmo k-means variando el número de clusters y evaluar la bondad de cada una de las particiones obtenidas. Sin embargo, si se conoce a priori que se debe mantener la cantidad de clusters en un valor bajo dado que la cantidad de muestras que se disponen es relativamente baja.

Se evaluó la bondad de las particiones variando k entre $k = 2, ..., 9$, se utilizó 9 como el número máximo de clusters a evaluar dada la limitación de la cantidad de muestras que se disponen en la base de datos. Para independizarnos de las posibles variaciones que se puedan dar por la aleatoriedad de la semilla que se utiliza en la técnica de clustering, se repitió el experimento 100 veces y se calculó el promedio de los índices.

En las figuras 9a y 9b se puede ver cómo varía los valores de los índices al variar el valor de k. Se puede observar que ambos índices coinciden en que la mejor partición se obtiene con un valor $k = 6$. Si bien era deseable que la cantidad de clusters fuese 2, 3 o a lo sumo 4 para poder obtener clusters con una cantidad de muestras razonables, se observa que los indices de bondad



(a) Índice Calinski-Harabasz (b) Índice Davies-Bouldin

Figura 9: Promedio de los índices obtenidos según la cantidad de clusters

de la partición no son buenos. Por lo cual se optó por tomar el valor $k = 6$ conociendo que luego se podrían tener inconvenientes dada que los clusters podrían tener un bajo número de muestras.

También se observó que tomando 6 como la cantidad de clusters se obtuvo una partición relativamente estable, dado que de las 100 repeticiones realizadas 66 devolvieron clusters exactamente iguales. Lo cual no sucedía con el resto de los valores de k.

Normalización de los datos:

Si bien no siempre es recomendable normalizar la muestras antes de hacer la detección de los clusters, en este caso si fue necesario volver a normalizar los datos. La razón de esto se explica a continuación.

En una primera instancia se probó realizar la búsqueda de los clusters sin normalizar los datos. Al calcular los índices para las particiones variando el valor de k se obtuvo que la mejor partición se daba con $k = 4$ y que el valor de los índices eran muy superiores a los que se encuentran en la figura 9. Esto en un principió parecía muy prometedor.

Sin embargo luego al analizar los clusters obtenidos se observó que la cantidad de muestras en cada cluster eran los siguientes:

$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$
3	5	8	1365

Lo cual pareció indicar que no se estaban detectando clusters de consumos sino que los clusters pequeños simplemente estaban detectando outliers. Por lo que luego al entrenar el cluster $\mathcal{C}_4$ se estaba volviendo al problema original a menos de algunas muestras que eran outliers. El objetivo buscado al aplicar la técnica de clustering es reducir la variabilidad de los tipos de consumo, por lo que la partición obtenida no lo cumple.

Se concluyó que el inconveniente se encontraba en que alguna de las características utilizadas presentan un rango de valores muy grande mientras que otras recorren un rango muy pequeño. Y por lo tanto las técnicas de clustering estaban centrando su atención en las características que presentan mayor variabilidad. Por lo cual se encontró que era conveniente normalizar los datos.

Luego de normalizar los datos y utilizando $k = 6$ las muestras se distribuyeron entre los clusters de la siguiente manera:

$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
11	38	80	131	555	566

Si bien algunos clusters presentan pocas muestras ninguno de ellos presenta la mayor parte de las mismas.

Asociación de las muestras fraudulentas a un cluster:

Dado que los clusters fueron formados utilizando sólo las muestras normales, luego fue necesario establecer algún criterio para asociar un cluster para cada una de las muestras fraudulentas. El criterio utilizado fue utilizar la regla del vecino más cercano tomando en consideración los 5 vecinos más cercanos.

Asociación de las muestras de test a un cluster:

Para la asociación de las muestras de test a un cluster se utilizó la misma estrategia que para las muestras fraudulentas.

6.2. Entrenamiento clasificadores de cada cluster

El proceso de entrenamiento consiste en entrenar un clasificador especifico para cada cluster buscando tener el máximo $F - value$ posible. De acuerdo a los resultados obtenidos hasta el momento se realizó una selección de características con *CfsSubsetEval* y para combatir el desbalance de clases se utilizó Matriz de Costos.

A continuación se muestra cómo se distribuyen las muestras de entrenamiento y test para cada uno de los clusters.

	Cantidad en Train		Cantidad en Test	
cluster	Normal	Fraudulentos	Normal	Fraudulentos
$\mathcal{C}_6$	566	52	249	11
$\mathcal{C}_5$	555	40	172	5
$\mathcal{C}_4$	131	12	36	3
$\mathcal{C}_3$	80	33	21	6
$\mathcal{C}_2$	38	9	9	2
$\mathcal{C}_1$	11	0	2	0

Al separar las muestras en cluster se redujo la cantidad de muestras disponibles para entrenar los clasificadores y para validar los resultados, por lo cual los resultados obtenidos están ligados a las pocas muestras que se tienen.

Una primera observación de la tabla permite detectar que el cluster $\mathcal{C}_1$ sólo presenta muestras normales dentro del conjunto de entrenamiento. Por esta razón no tiene sentido entrenar un clasificador para este cluster sino que basta con utilizar un clasificador trivial, en el cual todas las muestras sean clasificadas como normales.

Selección de características con CfsSubsetEval

Aplicando el algoritmo a cada uno de los conjuntos de entrenamiento asociados a los cluster se determinaron las características presentes en la tabla 19.

cluster	Características
$\mathcal{C}_6$	[3 30 31 33 34]
$\mathcal{C}_5$	[1 3 30 31 32 33 34]
$\mathcal{C}_4$	[30 31]
$\mathcal{C}_3$	[1 30 31]
$\mathcal{C}_2$	[1 30 33]
$\mathcal{C}_1$	–

Cuadro 19: Características elegidas por *CfsSubsetEval* por cluster

Estas características serán las utilizadas para entrenamiento y evaluación de los clasificadores. Se observa que si bien todos los subconjuntos de características son distintos, los mismos son muy similares entre sí. Algunas de las características que son seleccionadas por el algoritmo se repiten entre los clusters, las de mayor frecuencia son la 30 y 31, potencia contratada y número de irregularidades respectivamente. Otra observación es que la mayoría de las características seleccionadas son las de la *Base 6+* lo cual es consistente con el aumento de desempeño que tuvieron los clasificadores al agregar las nuevas características.

Resultados del entrenamiento por cluster

Se buscaron los mejores parámetros para *Random Forest* utilizando las técnicas estudiadas con anterioridad. A fin de definir una notación para los resultados obtenidos se utilizan las siguientes

abreviaciones.

- RF - *Random Forest*
- m - Cantidad de características a sortear en cada nodo
- N - Cantidad de árboles, se utilizaron 200 en todos los casos por lo cual para no saturar con números la notación no lo incluimos.
- Costo λ_{FN} - Costo asociado a clasificar como normal a un fraudulento. En todos los casos se utilizó como unitario el costo asociado a clasificar como fraudulento a un consumo normal.

Cluster	Clasificador	*F-value*
$\mathcal{C}_6$	RF - $m = 1$ - Costo $6,6$	25,40 %
$\mathcal{C}_5$	RF - $m = 2$ - Costo $4,6$	45,30 %
$\mathcal{C}_4$	RF - $m = 2$ - Costo $1,2$	25,00 %
$\mathcal{C}_3$	RF - $m = 1$ - Costo $1,0$	75,80 %
$\mathcal{C}_2$	RF - $m = 2$ - Costo $2,8$	47,60 %
$\mathcal{C}_1$	clasificador trivial	N.C.

Cuadro 20: Mejores resultados obtenidos en el proceso de entrenamiento con cluster

En la tabla 20 se encuentran los mejores resultados obtenidos con el clasificador diseñado específicamente para cada cluster[2].

Una verificación a realizar es que efectivamente el clasificador entrenado para un cluster está optimizado para clasificar los elementos pertenecientes al mismo cluster y que no es tan bueno para otros clusters. Con el fin de realizar esta verificación se utilizó cada uno de los 5 clasificadores entrenados paran las muestras de entrenamiento de los 4 clusters restantes, los resultados se encuentran en la tabla 21. Los resultados muestran que los clasificadores que mejor se desempeñan son los que fueron entrenados para el cluster específico, los cuales corresponden a la diagonal de la tabla. El único clasificador que no fue el mejor es el de $\mathcal{C}_2$, esto se debe a la poca cantidad de muestras en el cluster las cuales no permiten tener un buen entrenamiento de los arboles.

La efectividad del clasificador varía dependiendo del cluster, para evaluar el resultado global de la clasificación se debió considerar los TP, TN, FP y FN en cada uno de los conjuntos.

Datos / Clasificador	$\mathcal{C}_6$	$\mathcal{C}_5$	$\mathcal{C}_4$	$\mathcal{C}_3$	$\mathcal{C}_2$
$\mathcal{C}_6$	**25,40** %	21,95 %	25,40 %	16,51 %	18,75 %
$\mathcal{C}_5$	27,67 %	**45,28** %	37,21 %	37,68 %	28,13 %
$\mathcal{C}_4$	15,58 %	16,22 %	**25,00** %	22,73 %	0,00 %
$\mathcal{C}_3$	43,82 %	45,09 %	57,78 %	**75,76** %	45,21 %
$\mathcal{C}_2$	33,85 %	51,43 %	44,44 %	**58,33** %	47,62 %

Cuadro 21: Resultados de los clasificadores de los distintos cluster cruzados

En la tabla 19 se encuentra la clasificación por cluster y la global. El desempeño global es apenas superior al obtenido con *Random Forest* con las nuevas características. Una diferencia radical que en el caso anterior es que se pueden buscar clasificadores que mejoren el desempeño por cluster lo que deriva en un mejoramiento del caso global. Los cluster en los cuales tiene sentido buscar mejorar los clasificadores son $\mathcal{C}_5$ y $\mathcal{C}_6$ ya que son los que tienen más cantidad de muestras fraudulentas.

[2]Se observa que en el caso del cluster $\mathcal{C}_1$ no tiene sentido calcular el *F-value* porque el cluster no presenta muestras fraudulentas. De todas maneras este clasificador tiene un accuracy de 100 %, ya que clasifica a todas las muestras correctamente.

cluster	TP	TN	FP	FN	*Precision*	*Recall*	$F - value$
$\mathcal{C}_6$	24	453	113	28	17,52 %	46,15 %	25,40 %
$\mathcal{C}_5$	24	513	16	42	60,00 %	36,36 %	45,28 %
$\mathcal{C}_4$	4	115	16	8	20,00 %	33,33 %	25,00 %
$\mathcal{C}_3$	25	72	8	8	75,76 %	75,76 %	75,76 %
$\mathcal{C}_2$	5	31	4	7	55,56 %	41,67 %	47,62 %
$\mathcal{C}_1$	0	11	0	0	N.C.	N.C.	N.C.
Total	**82**	**1195**	**157**	**93**	**34,31** %	**46,86** %	**39,61** %

Cuadro 22: Resultados por cluster y resultado global

6.3. Optimización en clasificadores de cluster

Para aprovechar la posibilidad de buscar una optimización por cluster, se buscó cómo mejorar los clasificadores en cada uno de los casos.

Selección de características con Wrapper

Se utilizó el algoritmo encapsulado maximizando el *F-value* para obtener un subconjunto de características que mejore el desempeño del clasificador. A continuación se indican las características que con este método se obtuvieron mejores resultados a los vistos anteriormente.

Cluster	Características
C_6	[1 3 13 14 27 28 30 33]
C_5	[9 14 20 29 31 33 34]
C_4	[12 22 26 29 31]
C_2	[16 23 33]

Cuadro 23: Características elegidas por *Wrapper* por cluster

A diferencia que con el *CfsSubsetEval* las características seleccionadas con *Wrapper* no se repiten entre clusters. Esto se debe a que el *Wrapper* busca maximizar el *F-value* teniendo en cuenta tanto el clasificador como las etiquetas de las muestras. por lo cual es razonable que la elección varíe entres los distintos clusters. Se hace notar que a diferencia del método *CfsSubsetEval* se seleccionan más características derivadas de las curvas de consumo de los clientes.

Clasificadores mejorados

Utilizando *Wrapper* se lograron mejoras en los clasificadores de todos los cluster a excepción del $\mathcal{C}_3$, en la tabla 24 se encuentran los resultados que fueron mejorados.

cluster	Clasificador	$F - value$
$\mathcal{C}_6$	RF - $m = 7$ - Costo $4,0$	28,15 %
$\mathcal{C}_5$	RF - $m = 7$ - Costo $2,5$	47,27 %
$\mathcal{C}_4$	RF - $m = 6$ - Costo $2,7$	43,48 %
$\mathcal{C}_2$	RF - $m = 2$ - Costo $1,5$	60,00 %

Cuadro 24: Mejores resultados de entrenamiento con cluster con *Wrapper*

Un detalle que se observa claramente en la descripción del clasificador de la tabla es que se necesita un m mayor que en el caso del *CfsSubsetEval*. Esto tiene sentido ya que cuantas más características se usan en el proceso del crecimiento del árbol se tiene un clasificador más fuerte,

siempre y cuando las características en conjunto aporten información. En el caso del *CfsSubsetEval* el mejor resultado se obtiene con pocas características lo que quiere decir que para *Random Forest* no tenían tanto valor en conjunto. Al utilizar un método de encapsulado con ese clasificador parece lógico que más características sean necesarias para el proceso de crecimiento del árbol. Recordando lo visto en el marco teórico de este clasificador tener muchas características hace que los arboles estén correlacionados entre sí por lo cual en el proceso de selección de m se optó por el número más pequeño posible que aporte una mejora considerable en el desempeño.

Cluster	TP	TN	FP	FN	*Precision*	*Recall*	$F - value$
$\mathcal{C}_6$	19	502	64	33	22,89 %	36,54 %	28,15 %
$\mathcal{C}_5$	13	553	2	27	86,67 %	32,50 %	47,27 %
$\mathcal{C}_4$	5	125	6	7	45,45 %	41,67 %	43,48 %
$\mathcal{C}_3$	25	72	8	8	75,76 %	75,76 %	75,76 %
$\mathcal{C}_2$	6	33	5	3	54,55 %	66,67 %	60,00 %
$\mathcal{C}_1$	0	11	0	0	NC	NC	NC
Total	**68**	**1285**	**85**	**78**	**44,44 %**	**46,58 %**	**45,48 %**

Cuadro 25: Resultados finales del conjunto de entrenamiento

Al comparar los resultados obtenidos en la tablas 25 y 22 se observa una mejora del $5,87\,\%$ del conjunto original. Las optimizaciones estudiadas fueron realizar un proceso de selección de características diferente y aplicar *SMOTE* para combatir el desbalance. Este último resultó peor en todos los caso en comparación a *Matriz de Costos* por ello no se encuentran los resultados con este algoritmo en el informe.

Ahora bien surge la disyuntiva si el *F-value* es una medida adecuada para el problema propuesto, para entender un poco más esto analicemos los resultados globales obtenidos en las tablas 25 y 22.

	Optimizado	Inicial
TP	68	82
TN	1285	1184
FP	85	186
FN	78	64
Precision	44,44 %	34,31 %
Recall	46,58 %	46,86 %
$F - value$	45,48 %	39,61 %

Cuadro 26: Resultados globales

En la tabla 26 se encuentra la comparación de los resultados obtenidos antes y después de la optimización. Según el criterio de evaluación establecido (el mejor es el de mayor *F-value*), los clasificadores optimizados son mejores que los iniciales, ¿pero necesariamente en un problema de detección de fraudes este indicador es el mejor?. La respuesta a esta pregunta es discutible, para comprender un poco más esta afirmación observemos los resultados obtenidos. Los TP en el conjunto inicial son más que en el optimizado, lo que quiere decir que se están detectando menos fraudulentos. Lo que hace que el conjunto optimizado sea mejor es que tiene menos FP lo que implica que le estoy acertando más a las muestras que clasifico como fraudes.

Dependiendo de los recursos disponibles (en este caso los recursos serían la cantidad de inspecciones) puede ser más conveniente detectar más fraudes que detectar menos. Con el evaluador que se está utilizando el compromiso de 'encontrar más' a costa de 'equivocarse más se puede obtener modificando el valor de β de forma de dar mayor peso al *Recall* que al *Precision*.

Un análisis alternativo que se puede utilizar es encontrar un evaluador que muestre de forma explícita lo que se desea en el problema de detección de fraudes: dada una cierta cantidad

de recursos (cantidad máxima de inspecciones a realizar) cómo se debe modificar al clasificador para encontrar la mayor cantidad posible de fraudes. Si bien la idea intuitiva es sencilla con los evaluadores clásicos no se considera este enfoque, por lo cual habría que realizar modificaciones a los algoritmos clásicos de búsqueda de parámetros óptimos para incluir este estudio.

6.4. Evaluación desempeño clusters

Se evaluó el desempeño de los conjuntos de clasificadores inicial y optimizado utilizando las muestras de test de cada cluster. Los resultados se encuentran en la tabla 27.

Se observa que el resultado con los conjuntos de test son sensiblemente inferiores a los obtenidos con los de entrenamiento. Esto llama la atención no sólo por la diferencia de los valores, sino porque hasta ese momento no se habían detectado diferencias tan grandes entre los resultados obtenidos sobre el conjunto de entrenamiento realizando validación cruzada y el conjunto de test.

	Optimizado			Inicial		
Cluster	*Precision*	*Recall*	$F - value$	*Precision*	*Recall*	$F - value$
$\mathcal{C}_6$	4,49 %	100,00 %	8,59 %	6,86 %	63,64 %	12,39 %
$\mathcal{C}_5$	26,32 %	100,00 %	41,67 %	9,26 %	100,00 %	16,95 %
$\mathcal{C}_4$	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %
$\mathcal{C}_3$	22,22 %	66,67 %	33,33 %	22,22 %	66,67 %	33,33 %
$\mathcal{C}_2$	0,00 %	0,00 %	0,00 %	100,00 %	50,00 %	66,67 %
$\mathcal{C}_1$	NC	NC	NC	NC	NC	NC
Total	**7,09** %	**74,07** %	**12,94** %	**10,11** %	**62,07** %	**17,39** %

Cuadro 27: Resultados de clustering sobre conjunto de test

Los malos resultados se pueden deber a dos razones, sobre-entrenamiento o la escasa cantidad de muestras disponibles para test. Con la cantidad de muestras disponibles, cuando el clasificador se equivoca en una sola muestra se traduce en un drástico cambio en los porcentajes obtenidos. Por lo cual no se puede determinar a ciencia cierta si los clasificadores están sobre-entrenados o si el conjunto de test es tan reducido que no es representativo del problema real.

Curiosamente el conjunto optimizado tuvo menor desempeño que el conjunto inicial, esto podría indicar que al utilizar *Wrapper* se sobreajustó al *Random Forest* a las muestras de entrenamiento pero con la poca cantidad de muestras no se puede asegurar que dicho efecto también se observe en un proceso de clasificación masivo.

7. Conclusiones

En el caso de estudio propuesto se pudieron implementar muchas de las herramientas vistas en el curso de *Reconocimiento de patrones 2014*, clasificadores, clustering y técnicas de balanceo de clases. Las características del problema fueron tratadas en reiteradas ocasiones en el curso, dado que es un problema de dos clases con aprendizaje supervisado, por lo tanto se pudo tener una implementación real de lo visto durante el semestre.

El principal problema con el que se tuvo que lidiar fue el desbalance de clases, dado que los clientes en condiciones anómalas de conexión son un escaso porcentaje del total de consumidores. Por lo cual se buscaron técnicas que permitieran equilibrar el problema, para efectivamente encontrar los fraudes. Otra de las dificultades que se encontró con este problema es la variedad de consumos que se tiene, lo cual dificulta enormemente la posibilidad de encontrar clientes fraudulentos ya que hay mucha variabilidad dentro los consumos normales.

Los datos con los que se trabajó son el resultado de inspecciones realizadas a una muestra de total de clientes por lo cual los resultados están sesgados a la elección clientes a inspeccionar. Es sabido que dependiendo de la zona, tipo de cliente, etc. los fraudes que se pueden encontrar son diferentes por lo cual es de esperar que a la hora de implementar masivamente un clasificador entrenado con este conjunto reducido se encuentren fraudes en el resto de los clientes que son similares a las muestras de entrenamiento.

Las características propuestas tanto en [Deca2011], como las nuevas incluidas en la *BASE 6+* permitieron detectar en el mejor de los caso 70 % de los fraudulentos, el cual viene acompañado de un gran número de falsos positivos. A lo largo de los diversos estudios realizados sobre este tema se utilizaron muchas técnicas, no sólo las realizadas en este trabajos sino también las citadas en la bibliografía. Si bien los resultados han ido mejorando con el tiempo se encontró mucha consistencia en los diferentes clasificadores y técnicas utilizadas, lo que indica que se está alcanzando el límite de los resultados esperables con el enfoque que se le esta dando al problema. Por lo que se puede concluir que si se desea mejorar efectivamente los resultados obtenidos se deben buscar enfoques alternativos.

Se encontró que el enfoque por clustering, intentando atacar el problema de la variabilidad de los consumos normales para facilitar el trabajo de los clasificadores, puede derivar en mejores resultados. Pero con la cantidad de datos con los que se dispone no se puede afirmar que efectivamente lleve a resultados notoriamente diferentes. De disponer de un conjunto de muestras más grande, con los criterios de elección de cantidad de cluster propuestos se podría llegar a atacar el problema desde diferentes ángulos ya que es posible encontrar técnicas para cada uno de los agrupamientos de consumos.

En el marco de este trabajo una limitante que se tuvo son los costos computacionales en el proceso, tanto de entrenamiento como los de obtención de los 'mejores parámetros' para cada clasificador. Al aumentar la cantidad de muestras con las que se trabaja este problema también se verá acentuado pero una vez se puedan determinar cluster de clientes resulta sencillo subdividir el problema y atacar a cada cluster con una técnica distinta.

Para obtener mejores resultados se podría probar con otro enfoque más especifico como lo puede ser simular los fraudes que se comenten, encontrar las características que presentan cada uno y encontrar un método para detectar estos fraudes específicos. Esto es un giro importante en el enfoque actual del problema, uno de los cambios sería que el problema pase a tener más clases, una asociada a cada uno de los fraudes que se desea detectar.

Un punto pendiente es buscar otro evaluador de desempeño de los clasificadores el cual este más adaptado a la realidad del problema. En un caso real se dispone de un recurso fijo, que en este caso serían la cantidad de inspecciones que se pueden realizar y lo que se busca maximizar es la cantidad de fraudes que se pueden encontrar dentro de los recursos disponibles.

Referencias

[RP2008] Kosut, Juan Pablo y Alcegaray, Diego (2008), *One Class SVM para detección de fraudes en el uso de energía eléctrica*. Proyecto de fin de curso de Introducción al Reconocimiento de Patrones, Facultad de Ingeniería, Universidad de la República.

[Deca2011] Decia, Federico; Di Martino, Matías y Molinelli Juan (2011), *Detección de consumos anómalos (De-Ca)*. Proyecto de grado, Facultad de Ingeniería, Universidad de la República.

[RP2011] Introini, Diego y Lena, Daniel (2011), *Proyecto detección de clusters*. Proyecto de fin de curso de Introducción al Reconocimiento de Patrones, Facultad de Ingeniería, Universidad de la República.

[RP2012] Castro, Sebastián y Rodríguez, Fernanda (2012) *Detección de consumos anómalos de energía eléctrica utilizando nuevas características.*. Proyecto de fin de curso de Introducción al Reconocimiento de Patrones, Facultad de Ingeniería, Universidad de la República.

[CSIC-UTE2014] Rodr´guez, Fernanda; Lecumberry, Federico y Fernández, Alicia (2014), *Proyecto CSIC Sector Productivo Modalidad I - UTE, Detección de registros de consumo anómalos*. Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República.

[Breiman2001] Breiman, Leo (2001), *Random Forest* Statistics Department, Universidad de California.

[Oshiro2012] Oshiro, Thais Mayumi; Perez, Pedro Santoro y Baranauskas, José Augusto. *How Many Trees in a Random Forest?* Department of Computer Science and Mathematics, Faculty of Philosophy, Sciences and Languages at Ribeirao Preto University of Sao Paulo

[ChaoChen] Chao, Chen;Andy, Liaw y Leo, Breiman. *Using Random Forest to Learn Imbalanced Data* Department of Statistics UC Berkeley

[ClVAlid] Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, Junjie Wu, *Understanding of Internal Clustering Validation Measures* 2010 IEEE International Conference on Data Mining.