ANEXO E

Diseño del modelo y resultados

La arquitectura del modelo se puede observar en la Figura 1 se muestra cómo están implementados y la interacción de cada uno de los componentes. Para los datos de entrada y el preprocesamiento de los datos se aplicó la preparación de los datos de la metodología analítica. Los demás componentes que serán explicados a detalle en los siguientes apartados están contemplados en el desarrollo de una librería en *Python* la cual puede ser encontrada en el siguiente repositorio https://github.com/JuanCa11/RMFAR.

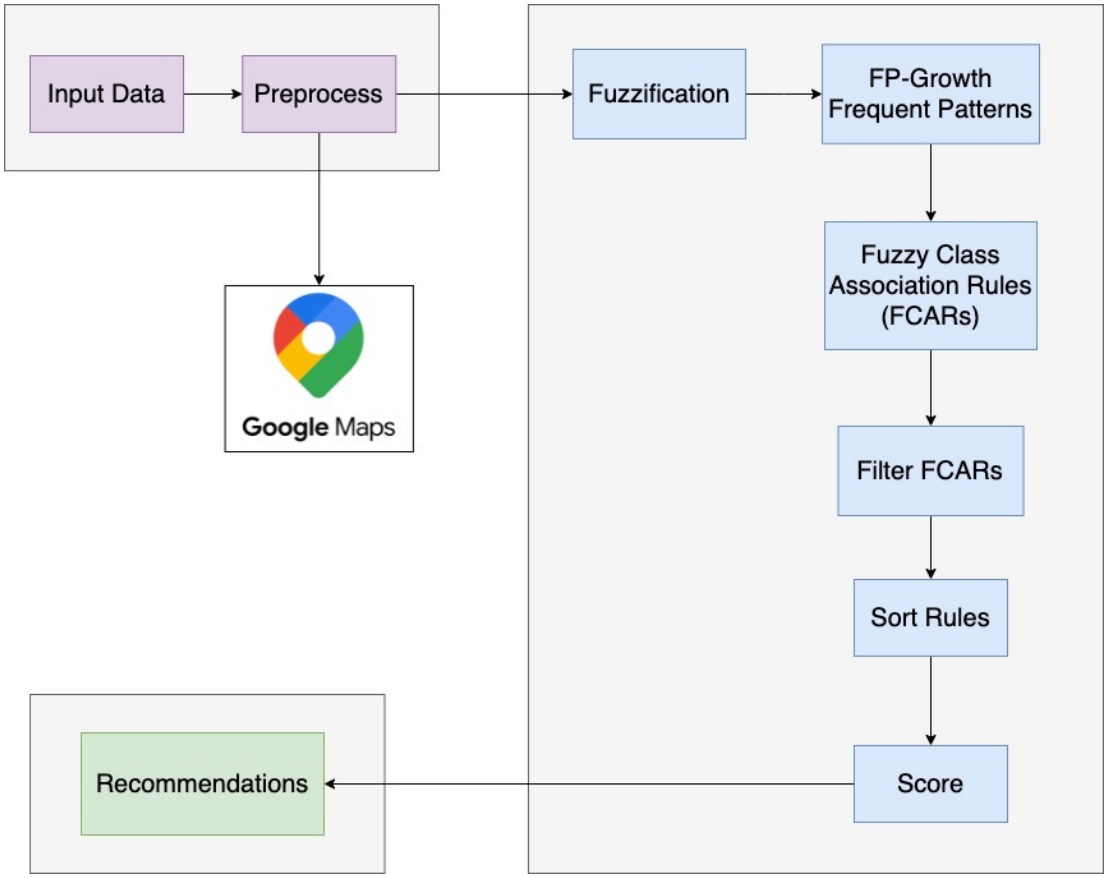

Figura 1: Arquitectura del modelo de recomendaciones RMFAR.

Particionamiento difuso

El componente o módulo de fuzzificación del modelo de recomendaciones se encarga de recibir la información pre procesada. Consiste en funciones para normalizar los datos, agrupar por k-medias y función de pertenencia al conjunto difuso. El número de agrupaciones realizadas corresponde al número de conjuntos difusos (ver Tabla 1). La entrada del Algoritmo 1. es el número de *clústers* El algoritmo selecciona puntos aleatorios como los primeros 4 grupos. Asigna repetidamente nuevos clústeres promediando los puntos asignados hasta que los puntos no cambian a un nuevo clúster. Después de agrupar, los puntos ahora tienen sus respectivos grupos. La salida es un dato difuso que es la conversión de los puntos a conjuntos difusos de pertenencia basados en la función triangular y trapezoidal usando los clústeres.

Variable	Conjunto difuso 1	Conjunto difuso 2	Conjunto difuso 3	Conjunto difuso 4
SCA_AREA	MUY PEQUEÑO	PEQUEÑO	MEDIANO	GRANDE
DENSIDAD_NOCTURNA	MUY BAJA	BAJA	MEDIANO	ALTA
DENSIDAD_BANCARIA	MUY BAJA	BAJA	MEDIANO	ALTA
DENSIDAD_TIENDAS	MUY BAJA	BAJA	MEDIANO	ALTA
DENSIDAD_PARQUES	MUY BAJA	BAJA	MEDIANO	ALTA
ILUMINACIÓN	MUY BAJA	BAJA	MEDIANO	ALTA
PERSONAS	MUY BAJA	BAJA	MEDIANO	ALTA
SEGURIDAD	MUY BAJA	BAJA	MEDIANO	ALTA
SENDERO	MUY BAJA	BAJA	MEDIANO	ALTA
TRANSPORTE	MUY BAJA	BAJA	MEDIANO	ALTA

Tabla 1: Definición de conjuntos difusos. Fuente Elaboración propia.

Algorithm 1 k-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4: **expectation:** Assign each point to its closest centroid.
5: **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

[1]

Clasificación basada en reglas de asociación

En esta sección se describe todo lo relacionado al componente de clasificación del modelo de recomendaciones *RMFAR* como se observa en el Algoritmo 2, dentro de este componente en la librería de *Python* creada se hace uso de la librería *mlxtend* para aplicar los algoritmos que se mencionan en los siguientes apartados.

Algorithm 2 Clasificación basada en reglas de asociación

Input: rowData
Output: classification

1: frequentItemsets = generateFrequentItemsets(minSupport)
2: CARules = CARs(frequentItemsets, minConfidence)
3: candidates = candidateRules(CARules, data)
4: rules = sortRules(candidates, c1, c2, c3, c4)
5: **return** rules[0].consequent

Extracción de patrones frecuentes

El algoritmo seleccionado para la extracción de patrones frecuentes es el algoritmo de *FP-Growth* según lo revisado en el marco teórico y estado del arte, internamente utiliza una estructura de datos llamada *FP-tree (árbol de patrones frecuentes)* sin generar los conjuntos candidatos de forma explícita, lo que lo hace particularmente atractivo para grandes conjuntos de datos[2]. Así las búsquedas son mucho menores en comparación a otros algoritmos como el *A priori*.

[1] *K-Means Clustering in Python: A Practical Guide – Real Python*. (n.d.). Retrieved June 17, 2021, from https://realpython.com/k-means-clustering-python/

[2] *Fpgrowth - mlxtend*. (n.d.). Retrieved May 19, 2021, from http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/fpgrowth/

En la generación de patrones frecuentes se definió como *threshold* o criterio de selección el *mínimo soporte* permitido de 0.014. El soporte es calculado como las transacciones donde se encuentra el ítem sobre el total de transacciones. Se definió este soporte para obtener la mayoría de las reglas posibles que pueden tener un soporte bajo, pero pueden tener otras métricas bastante buenas que permitan que sean bastante interesantes.

Creación de reglas de asociación de clase

En la creación de las reglas de asociación se define como *threshold* o criterio de selección la mínima confianza de 0.5 como se puede observar en el Algoritmo 3, la confianza de una regla se calcula como el soporte del *itemset* formado por todos los *items* que participan en la regla, dividido por el soporte del *itemset* formado por los *items* del antecedente. Posteriormente, se realiza el filtrado de las reglas de asociación de clase (CARs *Class Association Rules*), es decir, las reglas que tienen como consecuente el valor de la variable target, para este proyecto el tipo de delito de hurto.

Algorithm 3 CARs

```
Input: frequentItemsets, minConfidence
Output: CARs
CARs = []
for rule in rules do
    if rule.consequent is in classItems then
        CARs.append(rule)
    end if
end for
return CARs
```

Reglas candidatas

Para saber cuáles de las reglas se deben aplicar, normalmente una clasificación basada en reglas de asociación busca que cuando se va a clasificar un registro, este contenga todos los ítems de una regla, en caso contrario la regla es descartada, este proceso es netamente *CRISP*. Aprovechando el proceso de particionamiento *fuzzy* en el que se calculó el $\boldsymbol{\mu(m)}$ (función de pertenencia) para cada conjunto difuso. Se calculó el $\boldsymbol{\mu(m)}$ de la regla a través de la operación *AND* utilizando el operador básico del mínimo sobre los $\boldsymbol{\mu(m)}$ del registro en cada uno de los conjuntos difusos de la regla. Si el $\boldsymbol{\mu(m)}$ de la regla es mayor que 0 la regla debe ser considerada, en caso contrario que sea igual a 0 la regla es descartada.

Algorithm 4 Reglas candidatas

```
Input: CARules, dataMemberships
Output: candidateRules
candidateRules = []
for rule in rules do
    minimum = 0
    for item in rule.antecedent do
        minimum = min(minimum, dataMemberships[item])
    end for
    if min> 0 then
        rule.miu = minimum
        candidateRules.append(rule)
    end if
end for
return candidateRules
```

Ordenamiento de reglas

Posteriormente, se pondero cada una de las reglas para saber qué tan buena es una regla y establecer la prioridad teniendo en cuenta 4 criterios: soporte de la regla, confianza de la regla, cantidad de ítems en el antecedente, número de comodines (ver Tabla 2). El comodín se interpreta como aquel ítem/valor que puede ser cambiado en pro de generar una recomendación, por ejemplo, el género de la víctima de un delito de hurto no puede ser cambiado, pero el día en que una la victima estuvo en el lugar de los hechos si puede ser cambiado. Por defecto, los coeficientes de ponderación son los mismos para cada uno de los 4 criterios; sin embargo, también se realizaron varios experimentos variando los factores de ponderación, los resultados pueden verse en la siguiente sección.

ID Criterio	Criterio de ponderación
C1	Soporte de la regla
C2	Confianza de la regla
C3	Tamaño del antecedente
C4	Número de comodines

Tabla 2: Criterios de ponderación.

Por último, para la clasificación se calculó la factibilidad en adelante nombrada como *puntuación (score)* del delito de hurto, mediante la ponderación del $\boldsymbol{\mu(m)}$ de la regla y la ponderación de la regla realizada mediante los 4 criterios. Los coeficientes de ponderación son los mismos por defecto. Se ordenaron las reglas por este puntaje y la primera regla es la aplicada para clasificar el registro.

Algorithm 5 Ordenamiento de reglas

Input: candidates, c1, c2, c3, c4
Output: rules

```
for rule in rules do
    rule.weight = rule.support*c1 + rule.confidence*c2
     + rule.antecedents*c3 + rule.wildcards*c4
    rule.score = rule.weight*0.5 + rule.miu*0.5
end for
sortedRules = rules.sortedBy(score)
return sortedRules
```

Recomendaciones

Las recomendaciones son elaboradas mediante el algoritmo 5, en el cual para cada comodín de la regla se cambia el valor actual por los demás valores que puede tomar y verificar si existe una nueva regla con estos ítems (por ejemplo, si el valor actual es NOCHE los nuevos valores que podría tomar serian DIA, MEDIO DIA y TARDE). Si la regla existe se calcula la diferencia entre el puntaje de la regla actual y el puntaje de la nueva regla. Se ordenan las reglas por esta diferencia en orden descendente, para que la primera regla sea la que más minimiza el puntaje de ser víctima de delito de hurto.

Algorithm 6 Recomendaciones

Input: sortedRules
Output: recomendaciones

```
recomendations = []
for rule in sortedRules do
    for consequent in rule.consequents do
        for newConsequent in consequentsGroups[consequent.group] do
            if newConsequent != consequent then
                newRule = rule
                newRule.consequents[consequent.index] = newConsequent
                if newRule in sortedRules then
                    diffScore = rule.score - newRule.score
                    recomendations.append(newRule, diffScore)
                end if
            end if
        end for
    end for
end for
return recommendations
```

Un ejemplo de una recomendación podría ser dado los parámetros de entrada (ver Figura 2):

- Genero: Femenino
- Barrio: Pasadena
- Fecha y hora: 7 de diciembre de 2021 a las 9:00 pm
- Sitio de la actividad: Local comercial

Si la persona cambia el estar en horas de la noche por estar en horas de la mañana, la factibilidad o puntaje de ser victima del delito hurto a personas podría bajar a 0.25. Si la persona cambio el mes de diciembre por el mes de noviembre, la factibilidad o puntaje de ser victima del delito hurto a personas podría bajar a 0.4.

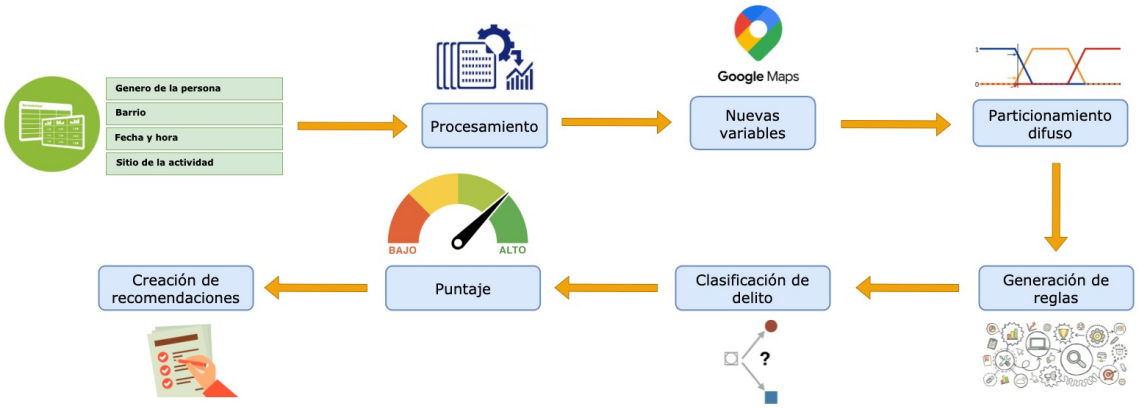

Figura 2: Flujo de recomendaciones del modelo RMFAR.

Experimentación y comparación

Métricas

Se realizaron 12 experimentos en los cuales se varió el porcentaje del *dataset* para entrenamiento y para pruebas, además para el modelo de *DT* se realizó un experimento con los hiperparámetros

de la librería por defecto y otro experimento con ajuste de hiperparámetros. Para el modelo *RMFAR* se realizaron experimentos específicos variando los coeficientes de ponderación (ver Tabla 10). Los resultados pueden ser vistos en la Tabla 12, se puede observar que el *accuracy* de los modelos es mayor del 50%. El mejor *accuracy* (0.5735) para el modelo de Árbol de decisión se obtuvo mediante el experimento 11 con ajuste de hiperparametros y con 70% de los datos para entrenamiento y 30% para pruebas (ver Tabla 4). Por otro lado, el mejor *accuracy* (0.55) para el modelo *RMFAR* se obtuvo mediante el experimento 1 con los coeficientes de ponderación del mismo valor 0.25 (ver Tabla 3) para los criterios definidos anteriormente (ver Tabla 2) y con 70% de los datos para entrenamiento y 30% para pruebas (ver Tabla 4).
En cuanto al tiempo de ejecución, podemos observar una gran diferencia entre los 2 modelos en general, el mayor tiempo del modelo de árbol de decisión fue 4.81 milisegundos (experimento 10) y el menor fue de 3.1 milisegundos (experimento 11). Esto se debe a que el modelo de recomendaciones genera muchas reglas lo cual hace que su procesamiento se bastante demandante, como se puede ver el en experimento 5 en comparación al experimento 7, la cantidad de reglas generadas en el experimento 5 fue bastante lo que hace que su procesamiento sea mayor pero sus métricas de precisión aumenten (ver Figura 3). Teniendo en cuenta que la finalidad del modelo es generar recomendaciones basadas en las reglas de asociación difusas de clase, un modelo que contemple una gran cantidad de reglas permite generar recomendaciones más acertadas y más variedad de recomendaciones.

Modelo	Train	Test	C1	c2	c3	c4
RMFAR1	70	30	0.25	0.25	0.25	0.25
RMFAR2	70	30	0.15	0.15	0.35	0.35
RMFAR3	70	30	0.1	0.1	0.7	0.1
RMFAR4	70	30	0.1	0.1	0.1	0.7
RMFAR5	80	20	0.25	0.25	0.25	0.25
RMFAR6	80	20	0.15	0.15	0.35	0.35
RMFAR7	80	20	0.1	0.1	0.7	0.1
RMFAR8	80	20	0.1	0.1	0.1	0.7

Tabla 3: Descripción modelos RMFAR, coeficientes de ponderación.

#	MODELO	DELITO	Train	Test	Precision	Recall	F1-Score	Tiempo de ejecución	Accuracy
1	RMFAR1	HURTO A PATRIMONIO	70	30	0.53	0.56	0.55	4min 39s	0.55
		HURTO A PERSONAS	70	30	0.57	0.54	0.55		
2	RMFAR2	HURTO A PATRIMONIO	70	30	0.49	0.96	0.65	37.2 s	0.498
		HURTO A PERSONAS	70	30	0.64	0.96	0.65		
3	RMFAR3	HURTO A PATRIMONIO	70	30	0.49	0.96	0.65	38 s	0.498
		HURTO A PERSONAS	70	30	0.64	0.07	0.13		
4	RMFAR4	HURTO A PATRIMONIO	70	30	0.53	0.56	0.55	4min 40s	0.55
		HURTO A PERSONAS	70	30	0.57	0.54	0.55		
5	RMFAR5	HURTO A PATRIMONIO	80	20	0.53	0.56	0.55	4min 36s	0.55
		HURTO A PERSONAS	80	20	0.57	0.54	0.55		
6	RMFAR6	HURTO A PATRIMONIO	80	20	0.49	0.96	0.65	34.5 s	0.498
		HURTO A PERSONAS	80	20	0.64	0.07	0.13		
7	RMFAR7	HURTO A PATRIMONIO	80	20	0.49	0.96	0.65	32 s	0.498
		HURTO A PERSONAS	80	20	0.64	0.07	0.13		
8	RMFAR8	HURTO A PATRIMONIO	80	20	0.49	0.96	0.65	32.2 s	0.498
		HURTO A PERSONAS	80	20	0.64	0.07	0.13		
9	DTREE NAIVE	HURTO A PATRIMONIO	80	20	0.26	0.31	0.28	4.1 ms	0.23
		HURTO A PERSONAS	80	20	0.19	0.16	0.17		
10	DTREE NAIVE	HURTO A PATRIMONIO	70	30	0.30	0.34	0.32	4.81 ms	0.26
		HURTO A PERSONAS	70	30	0.23	0.19	0.21		
11	DTREE HYPER	HURTO A PATRIMONIO	80	20	0.59	0.45	0.51	3.1 ms	0.5713
		HURTO A PERSONAS	80	20	0.56	0.69	0.62		
12	DTREE HYPER	HURTO A PATRIMONIO	70	30	0.62	0.38	0.46	3.15 ms	0.5735
		HURTO A PERSONAS	70	30	0.55	0.78	0.65		

Tabla 4: Resultados de experimentos RMFAR vs Decisión Tree.

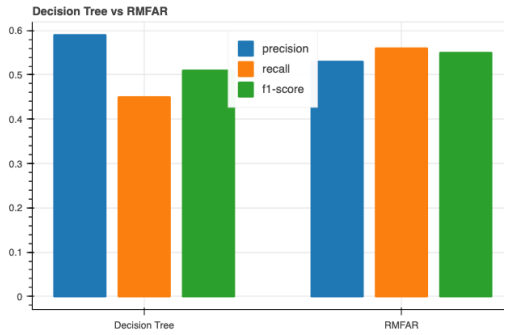

Figura 3: Comparación de métricas Decision Tree y RMFAR.

Reglas y decisiones

Otro punto de comparación son las reglas y decisiones generadas por los 2 modelos, por tal razón se eligió el modelo de *DT* puesto que su estructura está basada en reglas y/o decisiones lo cual también pueden ser utilizadas para su interpretación en incluso generar recomendaciones como lo hace *RMFAR*. Aunque el árbol de decisión permite ver sus reglas y/o decisiones (ver Figura 4) se puede observar que para las variables categóricas suele llegar a ser confuso por ejemplo, con la ACTIVIDAD el cual si la categoría es mayor o igual a 2.5 puede tomar un camino y otro. *RMFAR* genera reglas de asociación basadas en IF THEN. Lo que hace que estas puedan ser más entendibles y además crear mejores recomendaciones (ver Figura 5).

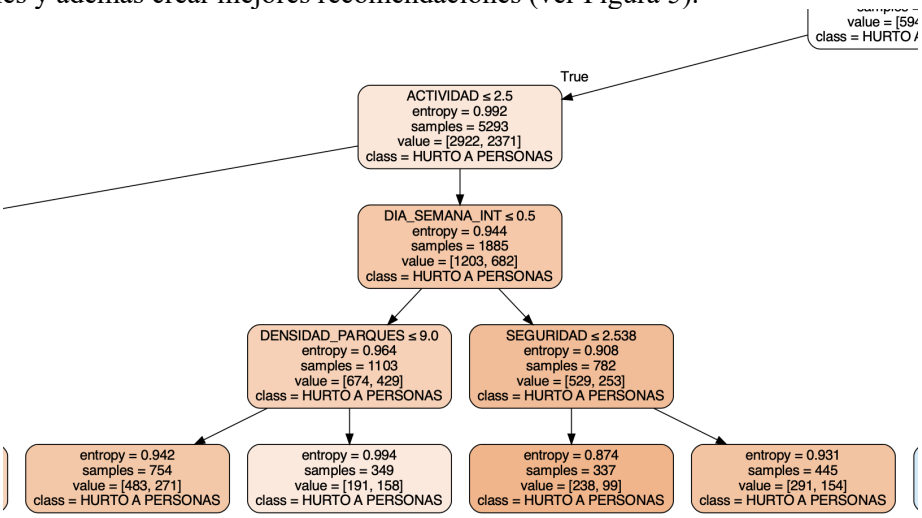

Figura 4: Reglas y/o decisiones del modelo de Árbol de Decisión.

```
In [17]: new_rules = recommender.get_new_rules(trigger_rules)
         new_rules
```

Out[17]:

	rule	class	score	new_rule	new_class	new_score	wildcard	diff
0	{SENDERO PUBLICO, BARRIO PEQUEÑO, FIN DE SEMANA}	{HURTO A PERSONAS}	0.575257	{BARRIO PEQUEÑO, LOCAL COMERCIAL, FIN DE SEMANA}	{HURTO A PATRIMONIO}	0.122702	SENDERO PUBLICO	0.452555
1	{SENDERO PUBLICO, NO QUINCENA, BARRIO PEQUEÑO, FIN DE SEMANA}	{HURTO A PERSONAS}	0.598784	{BARRIO MEDIANO, NO QUINCENA, FIN DE SEMANA, SENDERO PUBLICO}	{HURTO A PERSONAS}	0.195657	BARRIO PEQUEÑO	0.403127
2	{ILUMINACION BAJA, DENSIDAD BANCARIA MUY BAJA, SENDERO BAJO, SENDERO PUBLICO}	{HURTO A PERSONAS}	0.501181	{ILUMINACION BAJA, DENSIDAD BANCARIA MEDIA, SENDERO BAJO, SENDERO PUBLICO}	{HURTO A PERSONAS}	0.145759	DENSIDAD BANCARIA MUY BAJA	0.355421
3	{ILUMINACION BAJA, NO QUINCENA, FIN DE SEMANA, SENDERO PUBLICO}	{HURTO A PERSONAS}	0.524167	{SENDERO PUBLICO, NO QUINCENA, ILUMINACION MEDIA, FIN DE SEMANA}	{HURTO A PERSONAS}	0.270782	ILUMINACION BAJA	0.253385
4	{SENDERO PUBLICO, SENDERO BAJO, FIN DE SEMANA}	{HURTO A PERSONAS}	0.478497	{SENDERO PUBLICO, SENDERO MEDIO, FIN DE SEMANA}	{HURTO A PERSONAS}	0.266447	SENDERO BAJO	0.212050
5	{DENSIDAD PARQUES MUY BAJA, FEMENINO, CANTIDAD DE PERSONAS BAJA}	{HURTO A PATRIMONIO}	0.297952	{DENSIDAD PARQUES MEDIA, FEMENINO, CANTIDAD DE PERSONAS BAJA}	{HURTO A PATRIMONIO}	0.110899	DENSIDAD PARQUES MUY BAJA	0.187053
6	{FEMENINO, TRANSPORTE BAJO, CANTIDAD DE PERSONAS BAJA, NOCHE}	{HURTO A PATRIMONIO}	0.321058	{NOCHE, FEMENINO, TRANSPORTE MUY BAJO, CANTIDAD DE PERSONAS BAJA}	{HURTO A PATRIMONIO}	0.139479	TRANSPORTE BAJO	0.181579
7	{CANTIDAD DE PERSONAS MEDIA, DENSIDAD PARQUES MUY BAJA, DENSIDAD BANCARIA MUY BAJA, FEMENINO}	{HURTO A PATRIMONIO}	0.447879	{DENSIDAD PARQUES MUY BAJA, DENSIDAD BANCARIA MUY BAJA, FEMENINO, CANTIDAD DE PERSONAS BAJA}	{HURTO A PATRIMONIO}	0.320953	CANTIDAD DE PERSONAS MEDIA	0.126927
8	{FIN DE SEMANA, SENDERO PUBLICO, SENDERO MEDIO, ILUMINACION MEDIA, NO QUINCENA}	{HURTO A PERSONAS}	0.295902	{SENDERO PUBLICO, SENDERO MEDIO, ILUMINACION MEDIA, NO QUINCENA, ENTRE SEMANA}	{HURTO A PERSONAS}	0.172612	FIN DE SEMANA	0.123290

Figura 5: Reglas y recomendaciones del modelo RMFAR.