Random Forest + PQG-ID3

Pedro Almagro Blanco

30 de marzo de 2017

1. Random Forest

Los *métodos combinados* (o *métodos ensemble*) utilizan múltiples algoritmos de aprendizaje para obtener un rendimiento predictivo que mejore el que podría obtenerse por medio de cualquiera de los algoritmos de aprendizaje individuales que lo constituyen.

La idea de los métodos combinados es considerar múltiples hipótesis simultáneamente para formar una hipótesis que, con suerte, se comporte mejor. El término de *métodos ensemble* se suele reservar para aquellas combinaciones que hacen uso de múltiples hipótesis pertenecientes a una misma familia, mientras que se usa el término más general de *sistemas de aprendizaje múltiple* cuando se permite que las hipótesis que se combinan provengan de diversas familias.

Debido a que los métodos combinados hacen uso de varias hipótesis simultáneas, se produce una elevación en los costes computacionales, por lo que suele ser habitual utilizar algoritmos rápidos como espacio de hipótesis base, como son los árboles de decisión. De forma genérica, se denominan *Random Forest* (Bosques Aleatorios) a los métodos agregados que hacen uso de árboles de decisión como elementos constitutivos básicos.

Una combinación de algoritmos de aprendizaje supervisados es un algoritmo de aprendizaje supervisado y puede ser entrenado y usado para hacer predicciones.

Se debe tener en cuenta que una combinación de hipótesis de una determinada familia no es necesariamente una hipótesis de la misma familia, por lo que podríamos obtener mejores resultados que con los elementos individuales de la familia, o a veces también peores si no se tienen ciertas precauciones.

En la práctica, la forma en que se seleccionan los modelos individuales que se combinan hacen uso de algunas técnicas que tienden a reducir los problemas relacionados con el exceso de ajuste de los datos de entrenamiento y mejoran la predicción conjunta.

Empíricamente, se ha comprobado que cuando existe una diversidad significativa entre los modelos individuales, las combinaciones tienden a obtener mejores resultados, por lo que muchos de los métodos existentes buscan promover la diversidad entre los modelos que se combinan, y ello provoca a veces que se usen como modelos aquellos que hacen un uso fuerte de la aleatoriedad, en vez de modelos más dirigidos y que funcionan mejor individualmente.

Los métodos de combinación más usuales son el *Bagging* (o *Agregación Bootstrap*), el *Boosting*, y los *Subespacios Aleatorios*, y todos ellos se pueden aplicar sobre árboles de decisión para conseguir Random Forest de manera eficiente.

1.1. Bagging

Meta-algoritmo diseñado para conseguir combinaciones de modelos a partir de una familia inicial, provocando una disminución de la varianza y evitando el sobreajuste. Aunque lo más común es aplicarlo con los métodos basados en árboles de decisión, se puede usar con cualquier familia.

La técnica consiste en lo siguiente:

- Dado un conjunto de entrenamiento, D, de tamaño n, el bagging genera m nuevos conjuntos de entrenamiento, D_i, de tamaño n', tomando al azar elementos de D de manera uniforme y con reemplazo.
- A partir de estos m nuevos conjuntos de entrenamiento se construyen m nuevos modelos de aprendizaje, y la respuesta final de la combinación se consigue por

medio de votación de las m respuestas (en caso de buscar una clasificación) o por la media de ellas (en caso de buscar una regresión).

Se ha probado que el bagging tiende a producir mejoras en los casos de modelos individuales inestables (como es el caso de las redes neuronales o los árboles de decisión), pero puede producir resultados mediocres o incluso empeorar los resultados con otros métodos, como el de los K vecinos más cercanos.

1.2. Boosting

A diferencia del bagging, en el boosting no se crean versiones del conjunto de entrenamiento, sino que se trabaja siempre con el conjunto completo de entrada y se manipulan los pesos de los datos para generar modelos distintos. La idea es que en cada iteración se incremente el peso de los objetos mal clasificados por el predictor en esa iteración, por lo que en la construcción del próximo predictor estos objetos serán más importantes y será más probable clasificarlos bien.

El método de boosting más famoso es el que se conoce como AdaBoost que consta de los siguientes pasos:

1. Inicialmente a todos los datos del conjunto de entrenamiento se les asigna un peso idéntico, $w_i = \frac{1}{n}$, donde n es el tamaño del conjunto de datos.
2. Se entrena el modelo usando el conjunto de entrenamiento.
3. Se calcula el error del modelo en el conjunto de entrenamiento, se cuentan cuántos objetos han sido mal clasificados y se identifican cuáles son.
4. Se incrementan los pesos en aquellos casos de entrenamiento en los que el modelo anterior ha dado resultados erróneos.
5. Se entrena un nuevo modelo usando el conjunto de pesos modificados.
6. Volver al punto 3 (y se repite el proceso hasta el número de iteraciones fijadas inicialmente)

7. El modelo final se consigue por votación ponderada usando los pesos de todos los modelos.

Concretamente, para modificar los pesos tras el cálculo del predictor M_t con los pesos en el tiempo t, $w_{i,t}$ se utilizan las siguientes ecuaciones:

$$e_t = \frac{\sum_{i=1}^{n} w_{i,t} y_t M_t(x_i)}{\sum_{i=1}^{n} w_{i,t}}, \qquad \alpha_t = \frac{1}{2} \ln(\frac{1 - e_t}{1 + e_t})$$

donde x_i es el vector de entrada del objeto, y_i es la clase del objeto i-ésimo, y $M_t(x_i)$ es la predicción del modelo para la entrada x_i.

Tras esto, los pesos son actualizados de la siguiente forma:

$$w_{i,t+1} = c \cdot w_{i,t} e^{-\alpha_t y_i M_t(x_i)}$$

donde c es una constante de normalización elegida de forma que $\sum_{i=1}^{n} w_{i,t+1} = 1$.

La combinación construida clasifica por medio del voto de la mayoría, ponderando cada modelo M_t por medio de α_t. Es decir:

$$M(x_i) = sign(\sum_{i=1}^{t_{max}} \alpha_t M_t(x_i))$$

Normalmente, se espera una mejora significativa sobre la clasificación producida por cada uno de los modelos individuales, pero la convergencia no está garantizada y el rendimiendo podría degradarse tras un cierto número de pasos.

Estos pesos se utilizan para modificar el entrenamiento de los modelos individuales, por ejemplo, los árboles de decisión se pueden construir de manera que favorecen la división conjuntos de muestras con pesos altos.

1.3. Subespacios Aleatorios

En este método cada modelo se entrena con todos los ejemplos, pero solo considera un subconjunto de los atributos. El tamaño de estos subconjuntos es el parámetro del

método, y de nuevo el resultado es el promedio o votación de los resultados individuales de los modelos.

Sea N el tamaño del conjunto de entrenamiento, D el número de atributos de los datos y L el número de clasificadores individuales del conjunto. La combinación se construye siguiendo el siguiente algoritmo:

1. Para cada clasificador individual, l, tomamos d_l con $d_l < D$, el número de variables de entrada de l. Suele ser común usar el mismo valor de d_l para todos los clasificadores individuales.
2. Para cada clasificador individual, l, se crea un conjunto de entrenamiento seleccionado al azar d_l atributos distintos de los datos de entrada, y se entrena con ellos al clasificador.
3. Para clasificar un nuevo objeto, se combinan las salidas de los L clasificadore individuales por medio del voto de la mayoría o por combinación ponderada.

2. PQG

Definición 1. *Un* Grafo *es una tupla* $G = (V, E, \mu)$ *donde:*

- V *y* E *son conjuntos, que llamaremos, respectivamente,* conjunto de nodos *y* conjunto de aristas *del grafo.*
- μ *es una relación (no necesariamente funcional) que asocia a cada nodo o arista en el grafo su conjunto de propiedades, es decir,* $\mu : (V \cup E) \times R \to S$*, donde* R *representa el conjunto de posibles* claves *para dichas propiedades, y* S *el conjunto de posibles* valores *asociados a las mismas.*

Definición 2. *Un* Property Query Graph (PQG) *sobre* L *es un grafo con propiedades sobre* L*,* $Q = (V_Q, E_Q, \mu_Q)$*, donde existen* α *y* θ*, propiedades destacadas en* μ_Q*, tales que:*

- $\alpha : V_Q \cup E_Q \to \{+, -\}$ *total. Notaremos* x^+*, respectivamente* x^-*, para indicar que* $\alpha(x) = +$*, respectivamente* $\alpha(x) = -$*.*

- $\theta : V_Q \cup E_Q \to FORM(L)$ *asocia un predicado binario,* θ_x*, a cada elemento* x *de* $V_Q \cup E_Q$*. Si para un elemento* x*,* θ_x *no está explícitamente definida, supondremos que* θ_x *es una tautología, que podemos denotar en general por* T*.*

Escribiremos $Q \in PQG(L)$ para denotar que Q es un Property Query Graph sobre L (si el lenguaje está prefijado y no hay posibilidad de confusión, escribiremos simplemente $Q \in PQG$).

El sentido de usar predicados binarios es que en la semántica asociada a un PQG, que vamos a definir formalmente a continuación, usaremos la primera entrada de estos predicados para poder hablar de condiciones de pertenencia sobre subgrafos de G (el grafo general sobre el que estamos evaluando las consultas), mientras que la segunda esperará recibir como entrada elementos adecuados al tipo de elemento al que está asociado. Así, si S es un subgrafo y $a \in V_Q$ entonces $\theta_a(., S) \in FORM_V$, y si $e \in E_Q$ entonces $\theta_e(., S) \in FORM_{\mathcal{P}}$. Por ejemplo:

$$\theta_a(v, S) = \exists z \in S \ (z \rightsquigarrow v)$$
$$\theta_e(\rho, S) = \exists y, z(y \overset{\rho}{\rightsquigarrow} z \wedge y \notin S \wedge z \in S)$$

El primer predicado tendrá sentido para nodos, y se verificará cuando exista un camino en G que conecta un nodo de S (el subgrafo que estamos evaluando) con v, el nodo de entrada sobre el que se evalúa. El segundo predicado tendrá sentido para caminos, y se verificará cuando el camino evaluado, ρ, conecta S con su complementario (en G).

Definición 3. *Dado* $Q = (V_Q, E_Q, \mu_Q)$ *un PQG, el conjunto de* Q*-predicados asociados a* Q *es (si* $e = (u, v)$ *entonces* $e^o = u$ *y* $e^i = v$*):*

1. *Para cada arista,* $e \in E_Q$*, definimos el* Q*-predicado asociado como:*

$$Q_{e^o}(v, S) = \exists \rho \in \mathcal{P}_v^o(G) \ \left(\theta_e(\rho, S) \wedge \theta_{e^o}(\rho^o, S) \wedge \theta_{e^i}(\rho^i, S)\right)$$

$$Q_{e^i}(v, S) = \exists \rho \in \mathcal{P}_v^i(G) \ \left(\theta_e(\rho, S) \wedge \theta_{e^o}(\rho^o, S) \wedge \theta_{e^i}(\rho^i, S)\right)$$

En general, escribiremos $Q_{e^*}(v, S)$*, donde* $* \in \{o, i\}$*, y notaremos:*

$$Q_{e^*}^+ = Q_{e^*}, \qquad Q_{e^*}^- = \neg Q_{e^*}$$

2. Para cada nodo, $n \in V_Q$, definimos el Q-predicado asociado como:

$$Q_n(S) = \exists v \in V \left(\bigwedge_{e \in \gamma^o(n)} Q_{e^o}^{\alpha(e)}(v, S) \wedge \bigwedge_{e \in \gamma^i(n)} Q_{e^i}^{\alpha(e)}(v, S) \right)$$
$$= \exists v \in V \left(\bigwedge_{e \in \gamma^*(n)} Q_{e^*}^{\alpha(e)}(v, S) \right)$$

Y que podemos escribir en general como:

$$Q_n(S) = \exists v \in V \left(\bigwedge_{e \in \gamma(n)} Q_e^{\alpha(e)}(v, S) \right)$$

ya que para cada nodo no hay posibilidad de confusión. Además, notaremos:

$$Q_n^+ = Q_n, \qquad Q_n^- = \neg Q_n$$

A partir de estas notaciones, podemos definir formalmente cuándo un subgrafo verifica un PQG determinado:

Definición 4. *Dado un subgrafo S de un grafo con propiedades, $G = (V, E, \mu)$, y un Property Query Graph, $Q = (V_Q, E_Q, \mu_Q)$, ambos sobre el lenguaje L, diremos que S* verifica *Q, y lo denotaremos $S \vDash Q$, si se verifica la fórmula:*

$$Q(S) = \bigwedge_{n \in V_Q} Q_n^{\alpha(n)}(S)$$

En caso contrario, escribiremos: $S \nvDash Q$.

Definición 5. *Dado $Q \in PQG$. Diremos que $R \subseteq PQG$ es un* conjunto de refinamiento *de Q en G si verifica:*

1. $\forall\ Q' \in R\ (Q' \to Q)$
2. $\forall\ S \subseteq G\ (S \vDash Q \Rightarrow \exists!\ Q' \in R\ (S \vDash Q'))$

Teorema 1 (Añadir nodo nuevo a Q)**.** *Dado $Q \in PQG$ y $m \notin V_Q$, entonces el conjunto $Q + \{m\}$, formado por:*

$$Q_1 = (V_Q \cup \{m\},\ E_Q,\ \alpha_Q \cup (m, +),\ \theta_Q \cup (m, T))$$
$$Q_2 = (V_Q \cup \{m\},\ E_Q,\ \alpha_Q \cup (m, -),\ \theta_Q \cup (m, T))$$

es un conjunto de refinamiento de Q en G.

En la Figura 1 se muestra un diagrama explicativo de este primer conjunto de refinamiento.

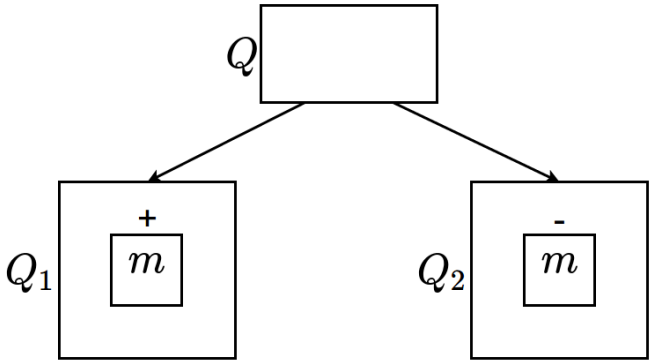

Figura 1: Refinamiento $Q + \{m\}$.

Teorema 2 (Añadir arista nueva entre nodos positivos de Q)**.** *Dado $Q \in PQG$ y $n, m \in V_Q^+$, entonces $Q + \{n^+ \xrightarrow{e^*} m^+\}$ ($* \in \{+, -\}$), formado por (donde $Q' = Cl_Q^{\{n,m\}}$):*

$$Q_1 = (V_{Q'},\ E_{Q'} \cup \{n^+ \xrightarrow{e^*} m^+\},\ \theta_{Q'} \cup (e, T))$$
$$Q_2 = (V_{Q'},\ E_{Q'} \cup \{n^+ \xrightarrow{e^*} m^-\},\ \theta_{Q'} \cup (e, T))$$
$$Q_3 = (V_{Q'},\ E_{Q'} \cup \{n^- \xrightarrow{e^*} m^+\},\ \theta_{Q'} \cup (e, T))$$
$$Q_4 = (V_{Q'},\ E_{Q'} \cup \{n^- \xrightarrow{e^*} m^-\},\ \theta_{Q'} \cup (e, T))$$

es un conjunto de refinamiento de Q en G.

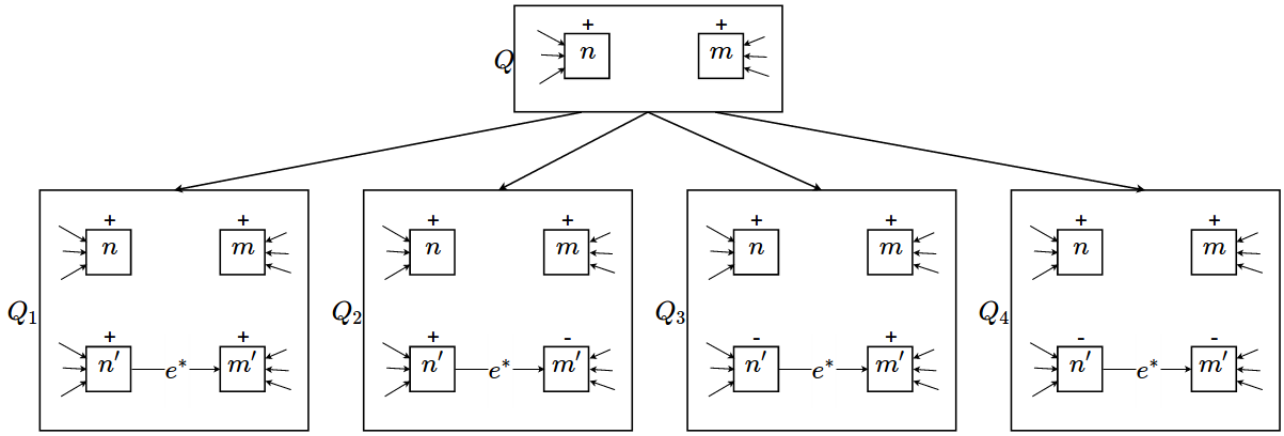

Figura 2: Refinamiento $Q + \{n^+ \xrightarrow{e^*} m^+\}$ $(* \in \{+, -\})$.

En la Figura 2 se muestra un diagrama explicativo de este conjunto de refinamiento. Si $n = m$ (la arista añadida es un lazo), entonces el conjunto de refinamiento anterior queda reducido a dos PQG, los equivalentes a Q_1 y Q_4.

Teorema 3 (Añadir predicado a arista positiva entre nodos positivos de Q)**.** *Dado $Q \in PQG$ $n, m \in V_Q^+$, con $n^+ \xrightarrow{e^+} m^+$, y $\varphi \in FORM(L)$, el conjunto $Q + \{n^+ \xrightarrow{e \wedge \varphi} m^+\}$ formado por (donde $Q' = Cl_Q^{\{n,m\}}$):*

$$Q1 = (V_{Q'},\ E_{Q'} \cup \{n^+ \xrightarrow{e'} m^+\},\ \theta_{Q'} \cup (e', \theta_e \wedge \varphi))$$
$$Q2 = (V_{Q'},\ E_{Q'} \cup \{n^+ \xrightarrow{e'} m^-\},\ \theta_{Q'} \cup (e', \theta_e \wedge \varphi))$$
$$Q3 = (V_{Q'},\ E_{Q'} \cup \{n^- \xrightarrow{e'} m^+\},\ \theta_{Q'} \cup (e', \theta_e \wedge \varphi))$$
$$Q4 = (V_{Q'},\ E_{Q'} \cup \{n^- \xrightarrow{e'} m^-\},\ \theta_{Q'} \cup (e', \theta_e \wedge \varphi))$$

es un conjunto de refinamiento de Q en G.

En la Figura 3 se muestra un diagrama explicativo de este conjunto de refinamiento.

Teorema 4 (Añadir predicado a nodo positivo con entorno positivo en Q)**.** *Dado $Q \in PQG$, $n \in V_Q^+$, con $\mathcal{N}_Q(n) \subseteq V_Q^+$, y $\varphi \in FORM(L)$. Definimos el conjunto $Q + \{n \wedge \varphi\}$*

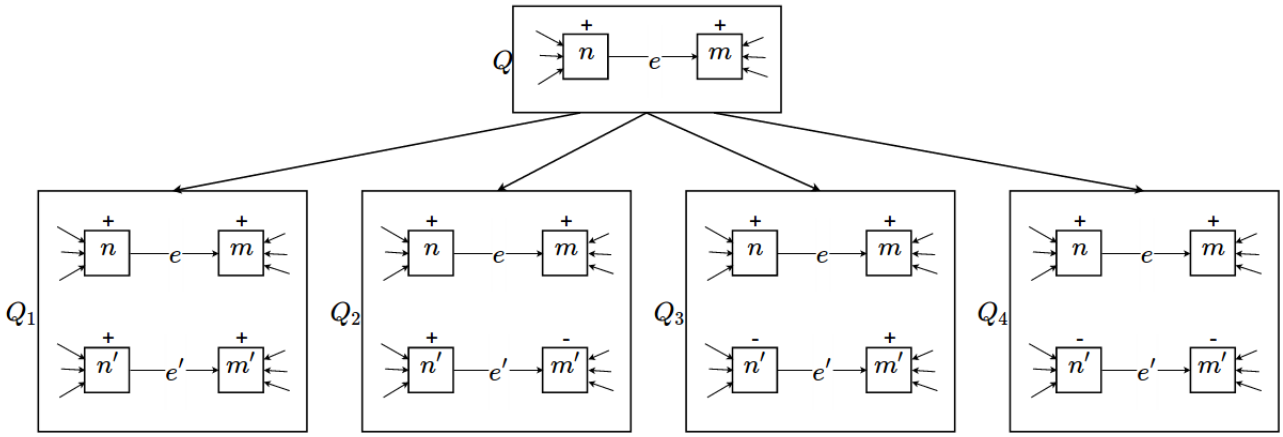

Figura 3: Refinamiento $Q + \{n^+ \xrightarrow{e \wedge \varphi} m^+\}$.

formado por:

$$\{Q_\sigma = (V_{Q'}, E_{Q'}, \alpha_{Q'} \cup \sigma, \theta_{Q'} \cup (n', \theta_n \wedge \varphi)) \ : \ \sigma \in \{+,-\}^{\mathcal{N}_Q(n)}\}$$

donde $Q' = Cl_Q^{\mathcal{N}_Q(n)}$, *y* $\{+,-\}^{\mathcal{N}_Q(n)}$ *es el conjunto todas las posibles asignaciones de signo a los elementos de* $\mathcal{N}_Q(n)$ *(el entorno, en* Q *del nodo* n*).*

Entonces $Q + \{n \wedge \varphi\}$ *es un conjunto de refinamiento de* Q *en* G*.*

En la Figura 4 se muestra un diagrama explicativo de este conjunto de refinamiento.

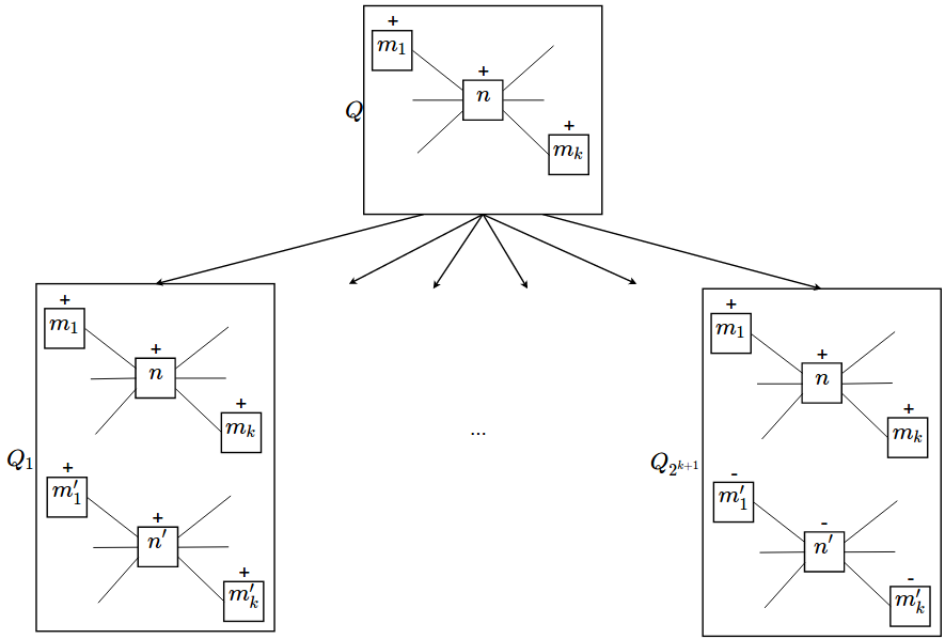

Figura 4: Refinamiento $Q + \{n \wedge \varphi\}$.

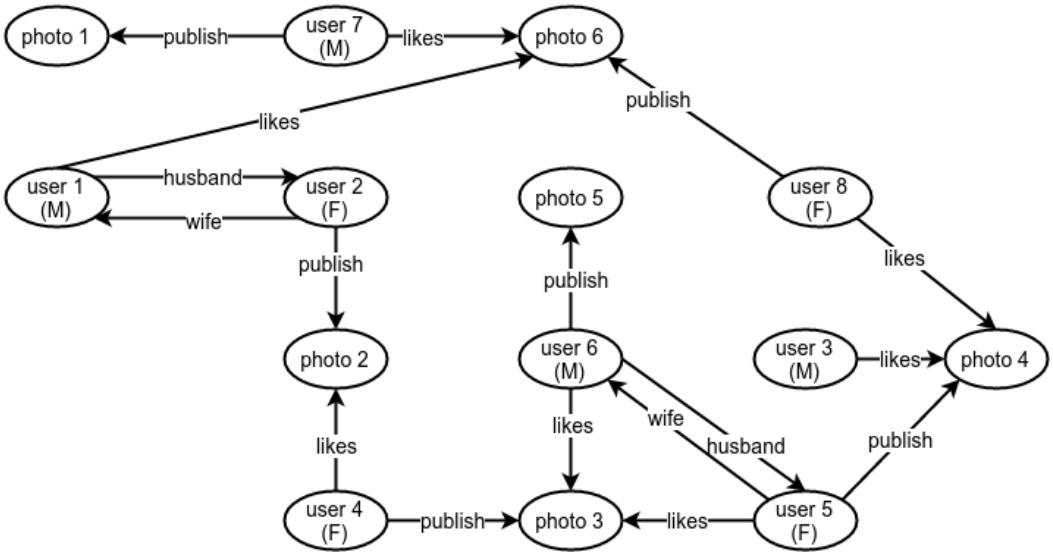

Figura 5: Grafo Social.

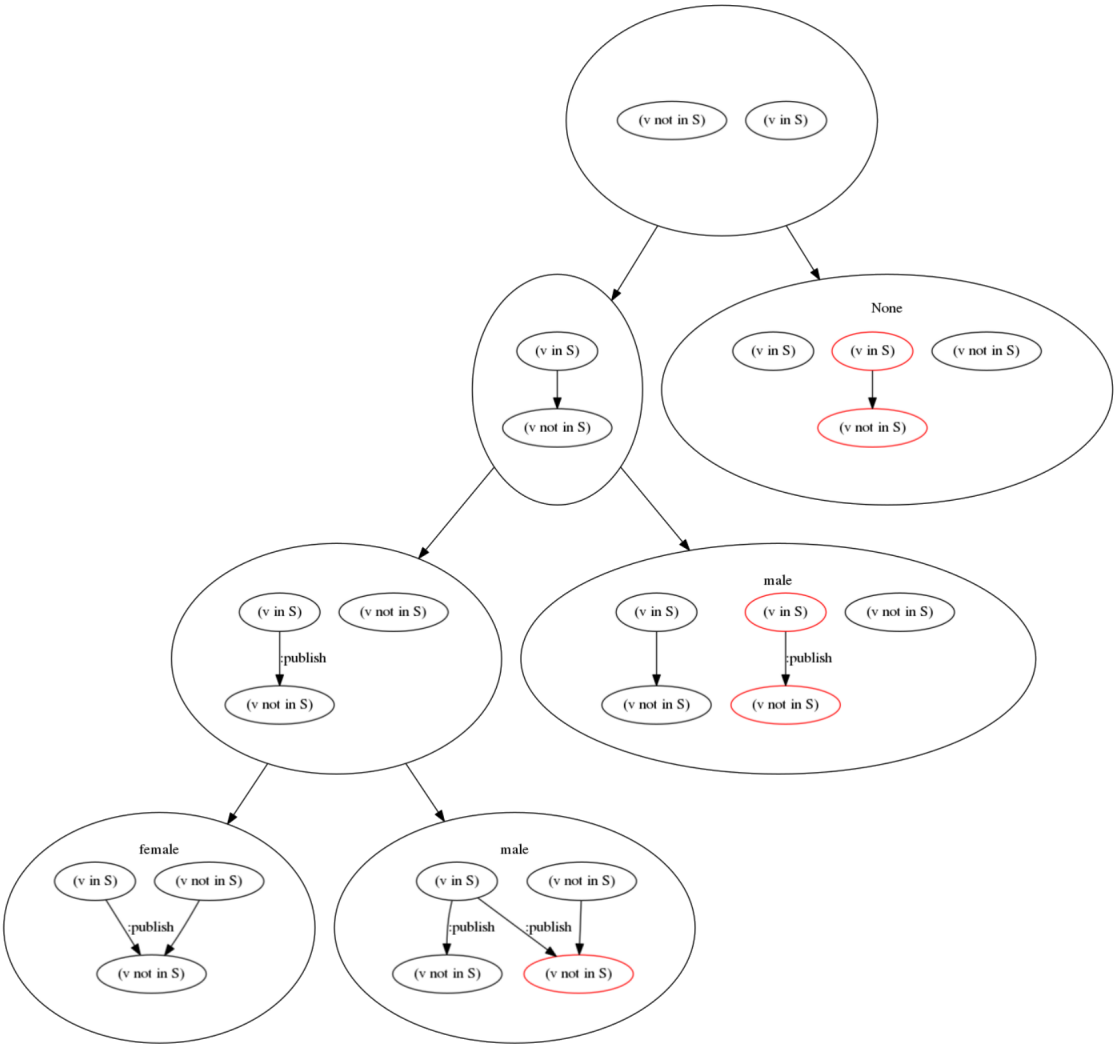

Figura 6: Árbol de decisión que clasifica correctamente a todos los nodos en el grafo 5 según su género.