# Proyecto Máquinas de Aprendizaje Computacional:
Predicción de Incendios Forestales utilizando técnicas de Máquinas de Aprendizaje

Daniel San Martín
daniel.sanmartinr@sansano.usm.cl

17 de abril de 2018

## 1. Introducción

Los incendios forestales son unos de los problemas medioambientales de mayor interés por el daño que generan en términos económicos, ambientales y por poner en peligro vidas humanas. Es por esto último que poder modelar estos fenómenos se convierte en una tarea de vital importancia.

Según (Preisler y Ager, 2013) los modelos de incendios se pueden agrupar en tres tipos: los modelos de riesgo, de propagación y de efecto. Los primeros están asociados a cuantificar la probabilidad y potenciales efectos ante posibles episodios de incendio. Según las variables utilizadas se pueden encontrar distintos trabajos como por ejemplo la estimación de la probabilidad de ocurrencia de un incendio en función de la ubicación y el día del año (Brillinger, Preisler, y Benoit, 2003; Hernandez-Magallanes, 2010), peligro de incendio e índices climáticos (Van Wagner, 1987; Burgan, 1988) entre otros (Braun, Jones, Lee, Woolford, y Wotton, 2010; Ager, Finney, Kerns, y Maffei, 2007; Calkin et al., 2011). El segundo grupo intenta modelar la propagación del fuego y existen desde enfoques físicos (Rothermel, 1972) hasta modelos de regresión para estimar la tasa de propagación (Sullivan, 2009). Generalmente este tipo de modelos asume que el combustible (lugar donde ocurre el incendio) puede ser teselado por una malla regular en donde cada celda tiene asociada una probabilidad de quemarse y que depende de las condiciones de las celdas vecinas. Existen distintas herramientas asociadas a este tipo de modelos y pueden ser revisados en (Andrews, 1986; Finney, 2006, 1998; Finney, McHugh, Grenfell, Riley, y Short, 2011; Finney, Grenfell, et al., 2011). Por último, los modelos de efecto son importantes para estudiar la administración y gestión de los combustibles, por ejemplo el control de mortalidad de árboles o análisis de ecosistemas, entre otros (Larkin et al., 2009; Reinhardt, 2003; Robichaud, Elliot, Pierson, Hall, y Moffet, 2007).

El desarrollo de este trabajo se basará en el artículo (Cortez y Morais, 2007) que realiza una comparación entre distintos modelos de *Aprendizaje de Máquinas* para predecir el comportamiento de un incendio dado datos meteorológicos. Además se espera incluir una propuesta de modelo utilizando un *Ensamblado de Máquinas* para intentar mejorar la calidad de los resultados.

### 1.1. Datos

Para el desarrollo de este proyecto se utilizará el *dataset* "Forest Fires Data Set" del *UCI Machine Learning Repository* (Cortez y Morais, 2007) y las características generales del *dataset* están descritas en el Cuadro 1.

Cuadro 1: Características de los datos

| | |
|---|---|
| Características de los datos | Multivariados |
| Características de los atributos | Reales y Etiquetas |
| Tareas asociadas | Regresión |
| Número de instancias | 517 |
| Número de atributos | 13 |
| Área | Física |
| Fecha donación | 2008-02-29 |

Los datos fueron extraídos del parque Montesinho (Portugal) y están relacionados con factores climáticos e índices/códigos basados en el sistema FWI (Canadian Forest Fire Weather Index). Estos códigos son FFMC (Fine Fuel Moisture Code), DMC (Duff Moisture Code), DC (Drought Code) e ISI (Initial Spread Index). Los detalles de cada atributo se presentan en el Cuadro 2.

Cuadro 2: Información de los atributos

| Nombre | Significado | Valores |
|---|---|---|
| X | Coordenada del eje $x$ | 1 a 9 |
| Y | Coordenada del eje $y$ | 2 a 9 |
| month | Mes del año | 'jan' a 'dec' |
| day | Día de la semana | 'mon' a 'sun' |
| FFMC | Código FFMC | 18.7 a 96.20 |
| DMC | Código DMC | 1.1 a 291.3 |
| DC | Código DC | 7.9 a 860.6 |
| ISI | Índice ISI | 0.0 a 56.10 |
| temp | Temperatura en °C | 2.2 a 33.30 |
| RH | Humedad relativa en % | 15.0 a 100 |
| wind | Velocidad del viento en km/h | 0.40 a 9.40 |
| rain | Lluvia en $mm/m^2$ | 0.0 a 6.4 |
| area | Área quemada (en ha) | 0.00 a 1090.84 |

En el trabajo desarrollado por la publicación guía del trabajo, se establece al atributo *area* como el *target* o variable dependiente.

#### 1.1.1. Pre-procesamiento

Para trabajar con el *dataset* fue necesario hacer algunas transformaciones. A las variables categóricas como el mes y día, se hizo la transformación numérica respectiva. Se realizó un breve análisis de los datos de donde notamos que existe mayor cantidad de incendios "pequeños", es por esto que se sugiere aplicar la transformación logarítmica $y = ln(x + 1)$ para incluir un poco de simetría y además escalar los datos. El resultado puede apreciarse en la Figura 1.

## 2. Desarrollo

Como se mencionó anteriormente, el enfoque de este proyecto será probar distintos modelos para intentar obtener resultados mejores o comparables al trabajo realizado por la publicación guía. Para apoyar el trabajo experimental se entrega un breve marco teórico sobre los modelos y métricas utilizadas, para posteriormente detallar el proceso experimental y resultados del proyecto.

### 2.1. Modelos

A continuación se presenta una breve descripción de los modelos utilizados en el desarrollo de este trabajo.

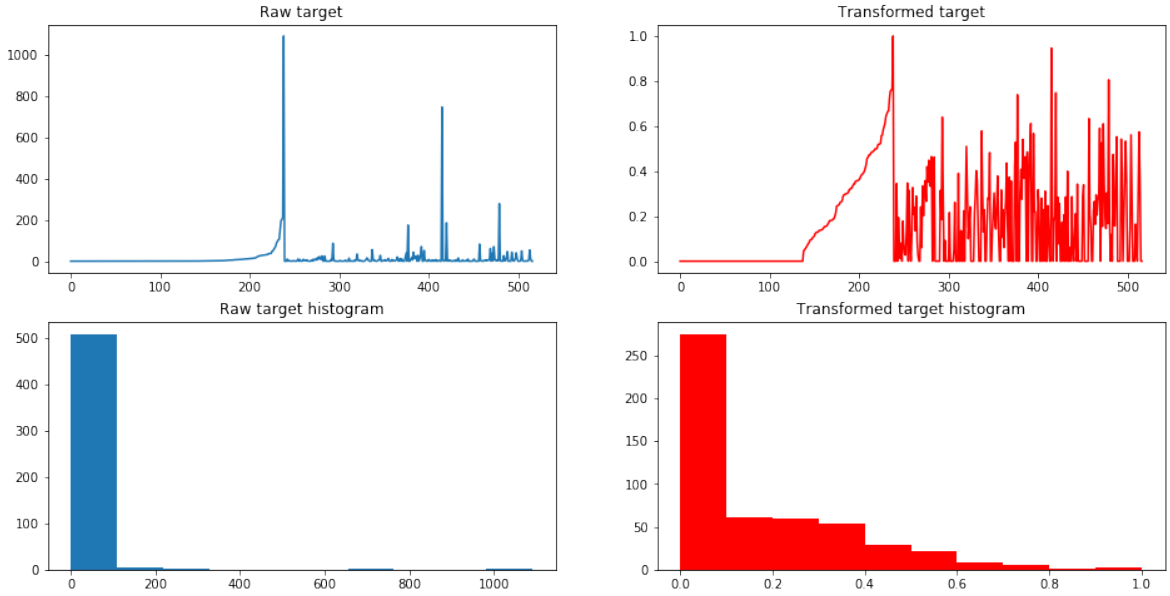


Figura 1: Transformación del *target*.

### 2.1.1. Regresión Lineal

La Regresión Lineal (*Linear Regression* LR) es un modelo matemático que expresa el valor predicho como la combinación lineal de los atributos de entrada (Seal, 1967). La notación matemática para un modelo de $I$ atributos es

$$\hat{y}(w, x) = \sum_{i=0}^{I} w_i x_i = Xw \tag{1}$$

con $x_0 = 1$. La forma más común para resolver este problema es utilizando mínimos cuadrados ordinarios

$$\min_{w} ||Xw - y||_2^2 \tag{2}$$

### 2.1.2. Arboles de Decisión

Los árboles de decisión (*Decision Tree* DT) (F. Breiman y Friedman, 1984) son un tipo de modelos construidos en base a reglas lógicas que permiten representar y categorizar una serie de condiciones utilizadas para resolver un problema. Se formalizan de la siguiente manera, dado un vector de entrenamiento $x_i \in R^n$, $i = 1, ..., l$ y un vector de etiquetas $y \in \mathbb{R}^l$, un árbol de decisión particiona recursivamente el espacio de tal forma que las muestras con las mismas etiquetas sean agrupadas conjuntamente.

Sea $Q$ el dato contenido en el nodo $m$. Para cada candidato a división $\theta = (j, t_m)$ consistente de una característica $j$ y un umbral $t_m$, particiona los datos en $Q_{\text{izq}}(\theta)$ y $Q_{\text{der}}(\theta)$ subconjuntos

$$Q_{\text{izq}}(\theta) = (x, y)|x_j \leq t_m \tag{3}$$

$$Q_{\text{der}}(\theta) = Q \setminus Q_{left}(\theta) \tag{4}$$

La impureza en $m$ es calculada utilizando una función de impureza $H()$, decisión que depende del tipo de problema (clasificación o regresión)

$$G(Q,\theta) = \frac{n_{\text{izq}}}{N_m} H(Q_{\text{izq}}(\theta)) + \frac{n_{\text{der}}}{N_m} H(Q_{\text{der}}(\theta)). \tag{5}$$

Los parámetros que minimizan la impureza se seleccionan de la siguiente forma

$$\theta^* = \text{argmin}_\theta \, G(Q,\theta). \tag{6}$$

El procedimiento se repite recursivamente para los subconjuntos $Q_{\text{izq}}(\theta^*)$ y $Q_{\text{der}}(\theta^*)$ hasta que la profundidad máxima definida haya sido alcanzada, $N_m < \text{mín}_{\text{muestras}}$ o $N_m = 1$.

Si el *target* es un valor continuo (problemas de regresión), el nodo $m$ representa una región $R_m$ con $N_m$ observaciones, los criterios utilizados puede ser el MSE, que minimiza el error L2 utilizando la media de valores en los nodos terminales, o el MAE, que minimiza el error L1 utilizando la mediana de los nodos terminales. De esta forma el MSE se define como

$$c_m = \frac{1}{N_m} \sum_{i \in N_m} y_i \tag{7}$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y_i - c_m)^2 \tag{8}$$

y el MAE como,

$$\bar{y_m} = \frac{1}{N_m} \sum_{i \in N_m} y_i \tag{9}$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} |y_i - \bar{y_m}| \tag{10}$$

donde $X_m$ es el conjunto de entrenamiento en el nodo $m$.

### 2.1.3. Máquinas de Soporte Vectorial

Las Máquinas de Soporte Vectorial (*Support Vector Machines* SVM) (Cortes y Vapnik, 1995) son una familia de modelos que construyen uno o varios hiperplanos separadores en un espacio de alta dimensión para resolver problemas de clasificación o regresión.

Dado un conjunto de vectores de entrenamiento $x_i \in \mathbb{R}^p, i = 1, ..., n$, y un vector $y \in \mathbb{R}^n$, $\varepsilon$-SVR resuelve el siguiente problema primal:

$$\begin{aligned} \min_{w,b,\zeta,\zeta^*} \; & \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\zeta_i + \zeta_i^*) \\ \text{sujeto a } & y_i - w^T \phi(x_i) - b \leq \varepsilon + \zeta_i, \\ & w^T \phi(x_i) + b - y_i \leq \varepsilon + \zeta_i^*, \\ & \zeta_i, \zeta_i^* \geq 0, i = 1, ..., n \end{aligned} \tag{11}$$

Su representación dual está dada por

$$\begin{aligned} \min_{\alpha,\alpha^*} \; & \frac{1}{2} (\alpha - \alpha^*)^T Q (\alpha - \alpha^*) + \varepsilon e^T (\alpha + \alpha^*) - y^T (\alpha - \alpha^*) \\ \text{sujeto a } & e^T (\alpha - \alpha^*) = 0 \\ & 0 \leq \alpha_i, \alpha_i^* \leq C, i = 1, ..., n \end{aligned} \tag{12}$$

donde $e$ es un vector de unos, $C > 0$ es la cota superior, $Q$ es una matriz semidefinida positiva de dimensión $n \times n$, $Q_{ij} \equiv K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ es el kernel. Implícitamente los vectores de entrenamiento son mapeados en un espacio de mayor dimensionalidad (posiblemente infinita) por la función $\phi$.

La función de decisión está definida por:

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x_i, x) + \rho \tag{13}$$

donde $\alpha_i - \alpha_i^*$ representa la diferencia de los vectores de soporte y $\rho$ es el término independiente o intercepto.

Las funciones de kernel utilizados en este tipo de problemas pueden ser los siguientes:

- Lineal: $\langle x, x' \rangle$.
- Polinomial: $(\gamma \langle x, x' \rangle + r)^d$. $d$ corresponde al grado del polinomio, $\gamma$ y $r$ son parámetros.
- RBF: $\exp(-\gamma \|x - x'\|^2)$. $\gamma \geq 0$ es un parámetros.
- Sigmoide: $\tanh(\gamma \langle x, x' \rangle + r)$, con $\gamma$ y $r$ parámetros.

### 2.1.4. Perceptrón Multicapa

El Perceptrón Multicapa (*Multi-layer Perceptron* MLP) (Rumelhart, Hinton, y Williams, 1986) es un algoritmo de aprendizaje supervisado que intenta aprender una función $f(\cdot) : R^m \to R^o$ mediante el entrenamiento sobre un *dataset*, donde $m$ es la dimensión de la entrada y $o$ es la dimensión de la salida. Dado un conjunto de características $X = x_1, x_2, ..., x_m$ y su respectivo *target* $y$, este algoritmo puede aprender una aproximador no lineal ya sea para problemas de clasificación o regresión.

Dado un conjunto de entrenamiento $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ donde $x_i \in \mathbb{R}^n$ e $y_i \in \mathbb{R}$, una MLP de una capa oculta y una neurona aprende la función $f(x) = W_2\ g(W_1^T x + b_1) + b_2$ donde $W_1 \in \mathbb{R}^m$ y $W_2, b_1, b_2 \in \mathbb{R}$ son parámetros del modelo. $W_1, W_2$ representan los pesos de la capa de entrada y oculta respectivamente; y $b_1, b_2$ representan los sesgos añadidos a las capas ocultas y de salida respectivamente. $g(\cdot) : \mathbb{R} \to \mathbb{R}$ es la función de activación de la capa oculta. Generalmente en problemas de regresión, la función de activación utilizada en la capa de salida es la función identidad.

Siguiendo en problemas de regresión, el MLP utiliza como función de costo el Error Cuadrático definido como,

$$Loss(\hat{y}, y, W) = \frac{1}{2}||\hat{y} - y||_2^2 + \frac{\alpha}{2}||W||_2^2. \tag{14}$$

Iniciando de un conjunto de pesos aleatorios el MLP minimiza la función de pérdida mediante la actualización repetida de estos pesos. Luego de calcular la pérdida, se realiza un paso de retropropagación desde la capa de salida hacia las capas anteriores, entregando a cada peso un valor que corresponda a disminuir el valor de la función de pérdida en el proceso de actualización.

En el gradiente descendente, el algoritmo de optimización generalmente utilizado, el gradiente $\nabla Loss_W$ de la pérdida con respecto a los pesos, es calculada y derivada desde $W$. Formalmente esto se expresa como,

$$W^{i+1} = W^i - \epsilon \nabla Loss_W^i \tag{15}$$

donde $i$ es el paso de la iteración, y $\epsilon > 0$ es la tasa de aprendizaje. El algoritmo se detiene una vez que alcanza el número máximo de iteraciones establecido o cuando se alcanza una cierta tolerancia del error.

Algunas funciones de activación típicas son:

- Identidad: $f(x) = x$
- Logística: $f(x) = \frac{1}{1+e^{-x}}$
- Tangente hiperbólica: $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

- Rectificador: (*Rectified linear unit* ReLU): $f(x) = \begin{cases} 0 & \text{para } x < 0 \\ x & \text{para } x \geq 0 \end{cases}$

### 2.1.5. Bagging

El Bootstrap Aggregating (*Bagging*) (L. Breiman, 1996) es un algoritmo que busca mejorar los resultados de predicción mediante la combinación de estimadores sobre datos de entrenamiento obtenidos de forma aleatoria utilizando *Bootstrap*. Formalmente podemos definir el método de la siguiente forma, sea un conjunto de entrenamiento $S = \{(x_1, y_1), ..., (x_m, y_m)\}$, para $n = \{1, ..., N\}$ se genera una muestra *Bootstrap* $S^n$ del conjunto $S$ y se entrena un estimador $f_n$ utilizando esta muestra. Finalmente la función hipótesis del ensamblado queda definida como:

$$F(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} f_n(\mathbf{x}). \tag{16}$$

### 2.1.6. Random Forest

*Random Forest* (RF) (L. Breiman, 2001) es una adaptación del algoritmo *Bagging* en el cual se promedia una gran cantidad de árboles de-correlacionados. Dado un conjunto de entrenamiento $S = \{(x_1, y_1), ..., (x_m, y_m)\}$, para cada $n = \{1, ..., N\}$ se entrena un árbol $T_n$ utilizando una muestra *Bootstrap* $S^n$, repitiendo los siguientes pasos:

1. Seleccionar $j$ variables aleatoriamente desde los atributos $I$.
2. Entrenar un árbol de clasificación o regresión (según corresponda) utilizando las $j$ variables seleccionadas en (1).

Esto se realiza para cada nodo terminal del árbol hasta alcanzar el número mínimo de nodos $n_{min}$. Finalmente la hipótesis del ensamblado queda definida como:

$$F(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} T_n(\mathbf{x}). \tag{17}$$

Además del RF, también se probará una variante de este método denominada *Extremely Randomized Tree* (ET) (Geurts, Ernst, y Wehenkel, 2006), en donde la regla de división de los árboles se construye de forma aleatoria.

### 2.1.7. Boosting

Los algoritmos de tipo *Boosting* (Freund y Schapire, 1997) intentan mejorar la predicción de un modelo mediante la iteración sobre *weak learners* para luego, mediante la agregación de estos, construir un modelo más robusto. La implementación utilizada en este trabajo es el algoritmo *AdaBoost*, específicamente *AdaBoost.R2* (Drucker, 1997) el cual funciona de la siguiente forma, inicializar los pesos $w_i = 1,\ i = 1, ..., N_1$, para cada muestra de entrenamiento. Repetir los siguientes pasos hasta que el promedio de la función de pérdida sea menor a 0,5:

1. Obtener una muestra de $N_1$ ejemplos (con reemplazo) para generar el conjunto de entrenamiento. La probabilidad de que un ejemplo $i$ esté dentro del conjunto de entrenamiento es $p_i = \frac{w_i}{\sum w_i}$
2. Construir un estimador para generar la hipótesis $h_t : x \to y$.
3. Entrenar el estimador para obtener la predicción $y_i^{(p)}(\mathbf{x}_i),\ i = 1, ..., N$.
4. Calcular el error $L_i$ para cada ejemplo de entrenamiento.
5. Calcular el error promedio $\bar{L} = \sum_{i=1}^{N_1} L_i p_i$.

6. Definir $\beta = \frac{\bar{L}}{1-\bar{L}}$, donde $\beta$ es una medida de confianza del predictor. Un bajo $\beta$ indica alta confianza.

7. Actualizar los pesos $w_i \to w_i \beta^{1-L_i}$

Para una entrada $\mathbf{x}_i$ cada una de los $T$ estimadores generan una predicción $h_t,\ t = 1, ..., T$. La predicción acumulada $h_f$ utilizando $T$ estimadores se define como

$$h_f = \text{ínf} \left\{ y \in Y : \sum_{t:h_t \leq y} \log(1/\beta_t) \geq \frac{1}{2} \sum_t \log(1/\beta_t) \right\}. \tag{18}$$

Las funciones de pérdida $L_i$ a utilizar pueden ser:

- Lineal: $L_i = \frac{|y_i^{(p)}(\mathbf{x}_i) - y_i|}{D}$
- Cuadrática: $L_i = \frac{|y_i^{(p)}(\mathbf{x}_i) - y_i|^2}{D^2}$
- Exponencial: $L_i = 1 - \exp\left(\frac{-|y_i^{(p)}(\mathbf{x}_i) - y_i|}{D}\right)$

donde $D = \sup |y_i^{(p)}(\mathbf{x}_i) - y_i|,\ i = 1, ..., N_1$

### 2.1.8. Gradient Tree Boosting

*Gradient Tree Boosting* (GTB) (Friedman, 2001) es una generalización de la técnica *Boosting* pero que puede utilizar una función de costo diferenciable arbitraria. GTB es un método considerablemente más preciso y efectivo que puede ser utilizado para problemas de regresión y clasificación.

La formulación matemática de este modelo es la siguiente, GTB considera la suma de modelos de la siguiente forma:

$$F(x) = \sum_{m=1}^{M} \gamma_m h_m(x) \tag{19}$$

donde $h_m(x)$ son funciones base denominadas generalmente como *weak learners* en el contexto de *Boosting*. GTB utiliza árboles de decisión de tamaño fijo como *weak learners*. Los árboles de decisión tienen un número de habilidades que permiten que sean útiles para el *boosting*, por ejemplo la capacidad de manipular datos de distinto tipo además de modelar funciones de alta complejidad.

Al igual que otros algoritmos de *Boosting*, GTB construye modelos aditivos de forma secuencial:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x). \tag{20}$$

En cada etapa, el árbol de decisión $h_m(x)$ es escogido de forma que minimice la función de pérdida $L$ dado el modelo actual $F_{m-1}$ y su ajuste $F_{m-1}(x_i)$

$$F_m(x) = F_{m-1}(x) + \arg\min_h \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h(x)) \tag{21}$$

El modelo inicial $F_0$ es específico del problema, por ejemplo para una regresión por mínimos cuadrados, por lo general, se escoge la media de los valores de *target*.

GTB intenta resolver el problema de minimización numéricamente por medio de gradiente descendente: La dirección corresponde al gradiente negativo del la función de costo evaluada en el modelo actual $F_{m-1}$ el cual puede ser calculado para cualquier función de costo diferenciable de la siguiente forma:

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_F L(y_i, F_{m-1}(x_i)), \tag{22}$$

donde el tamaño del paso $\gamma_m$ se escoge utilizando una búsqueda lineal:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) - \gamma \frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}\right) \tag{23}$$

El algoritmo para regresión o clasificación solo se diferencian en la función de pérdida utilizada. Para el caso de regresión, algunas de las funciones de pérdida utilizadas son:

- Mínimos cuadrados $L = \sum_{i=1}^{n} (y_i - f(x_i))^2$.
- Mínimas desviaciones absolutas $L = \sum_{i=1}^{n} |y_i - f(x_i)|$.
- La función de Huber $L = \sum_{i=1}^{n} L_\delta(y_i, f(x_i)),\ L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{para } |y - f(x)| \leq \delta, \\ \delta\,|y - f(x)| - \frac{1}{2}\delta^2 & \text{en otro caso.} \end{cases}$

## 2.2. Métricas

Las métricas utilizadas para el análisis de los modelos fueron los siguientes:

### 2.2.1. Error Cuadrático Medio

El Error Cuadrático Medio (*Mean Squared Error* MSE) es una métrica que corresponde al valor esperado del error o pérdida al cuadrado. Se define de la siguiente forma, si $\hat{y}_i$ es el valor predicho de $i$-ésimo ejemplo, e $y_i$ corresponde al valor verdadero, entonces el MSE estimado sobre $N$ muestras está definido por

$$\text{MSE}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2. \tag{24}$$

Para comparar con los resultados expuestos en el artículo guía, se incluye también el $RMSE$ que es la raíz cuadrada del $MSE$.

### 2.2.2. Error Medio Absoluto

El Error Medio Absoluto (*Mean Absolute Error* MAE) es una métrica que corresponde al valor esperado de la pérdida de valor absoluto o norma $L1$. Se define de la siguiente manera, si $\hat{y}_i$ es el valor predicho del $i$-ésimo ejemplo, y $y_i$ es el valor real correspondiente, entonces el MAE estimado sobre $N$ muestras está definido por

$$\text{MAE}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|\,. \tag{25}$$

## 2.3. Experimentos

Para el desarrollo de este trabajo se dividió el *dataset* en un conjunto de entrenamiento y prueba, en la proporción 80 % y 20 % respectivamente. En el ajuste de los hiper-parámetros de cada modelo se utilizó una búsqueda exhaustiva implementada como *Grid Search* con *Cross-Validation* con $K = 5$. Entre cada familia de modelos se escogió el de menor valor de $MSE$ resultante del *Cross-Validation* sobre el conjunto de entrenamiento. Una vez seleccionado el modelo, se aplicaron las métricas sobre el conjunto de prueba.

## 2.4. Resultados

A continuación se presentan los resultados obtenidos luego de los experimentos.

Cuadro 3: Resultado de las méticas para cada modelo

| Model | MSE | RMSE | MAE |
|---|---|---|---|
| LR | 12069.92 | 109.86 | 19.58 |
| DT | 11575.84 | 107.59 | 15.02 |
| SVM | 12135.25 | 110.16 | 19.67 |
| MLP | 12079.37 | 109.91 | 19.66 |
| Bagging | 8978.24 | 94.75 | 14.38 |
| RF | 10293.97 | 101.46 | 15.25 |
| ET | 11764.26 | 108.46 | 18.04 |
| Adaboost | 11957.46 | 109.35 | 19.65 |
| GTB | 4255.87 | 65.24 | 8.13 |

De los resultados obtenidos podemos corroborar que son comparables a los expuestos en el artículo que se utilizó como base. Es importante mencionar que el resultado del $GTB$ fue mejor que los resultados del artículos en términos del $MAE$.

# 3. Conclusiones

Del desarrollo de este proyecto notamos la potencia de los modelos ensamblados, los cuales mejoran los resultados considerablemente en comparación a los modelos no ensamblados. El principal problema de estos modelos es que tienden a sobreajustar, pero gracias al uso de algunas herramientas (según el modelo) se puede mitigar este problema. Si bien, la publicación guía sugiere el uso de ciertas configuraciones de $X$, que podría ser vistos como una selección de características, el modelo $GTB$ demostró mejorar los resultados utilizando toda la información.

Por limitaciones computacionales no se realizaron más pruebas, pero como trabajo futuro se espera incluir técnicas de selección de características para así poder comparar resultados en iguales condiciones al artículo guía.

El modelo $GTB$ nos entregó resultados bastante interesantes sobre los modelos que se utilizan el artículo. Según la literatura es uno de los modelos más potentes pero que lamentablemente tiene asociada una alta exigencia computacional dada la estructura iterativa del algoritmo. De todas formas gracias a las capacidades computacionales actuales este podría no ser un inconveniente, siempre que el problema no implique re-entrenar el modelo muchas veces o quizás el uso de entrenamiento en tiempo real.

# Referencias


Ager, A. A., Finney, M. A., Kerns, B. K., y Maffei, H. (2007). Modeling wildfire risk to northern spotted owl (strix occidentalis caurina) habitat in central oregon, usa. *Forest Ecology and Management*, *246*(1), 45–56.

Andrews, P. L. (1986). Behave: fire behavior prediction and fuel modeling system-burn subsystem, part 1.

Braun, W. J., Jones, B. L., Lee, J. S., Woolford, D. G., y Wotton, B. M. (2010). Forest fire risk assessment: an illustrative example from ontario, canada. *Journal of Probability and Statistics*, *2010*.

Breiman, F., y Friedman, J. (1984). *Classification and regression trees.*

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Brillinger, D. R., Preisler, H. K., y Benoit, J. W. (2003). Risk assessment: a forest fire example. *Lecture Notes-Monograph Series*, 177–196.

Burgan, R. E. (1988). *1988 revisions to the 1978 national fire-danger rating system* (Inf. Téc.). US Department of Agriculture, Forest Service, Southeastern Forest Experiment Station.

Calkin, D. E., Ager, A. A., Thompson, M. P., Finney, M. A., Lee, D. C., Quigley, T. M., ... Gilbertson-Day, J. M. (2011). A comparative risk assessment framework for wildland fire management: the 2010 cohesive strategy science report.

Cortes, C., y Vapnik, V. (1995, 01 de Sep). Support-vector networks. *Machine Learning*, *20*(3), 273–297. Descargado de https://doi.org/10.1007/BF00994018 doi: 10.1007/BF00994018

Cortez, P., y Morais, A. (2007). A data mining approach to predict forest fires using meteorological data. En J. Neves, M. F. Santos, y J. Machado (Eds.), *New trends in artificial intelligence, proceedings of the 13th epia 2007 - portuguese conference on artificial intelligence* (p. 512-523). Guimaraes, Portugal. (Available at: http://www.dsi.uminho.pt/~pcortez/fires.pdf)

Drucker, H. (1997). Improving regressors using boosting techniques. En *Icml* (Vol. 97, pp. 107–115).

Finney, M. A. (1998). *Farsite, fire area simulator–model development and evaluation* (Vol. 3). US Department of Agriculture, Forest Service, Rocky Mountain Research Station Ogden, UT.

Finney, M. A. (2006). An overview of flammap fire modeling capabilities.

Finney, M. A., Grenfell, I. C., McHugh, C. W., Seli, R. C., Trethewey, D., Stratton, R. D., y Brittain, S. (2011). A method for ensemble wildland fire simulation. *Environmental Modeling & Assessment*, *16*(2), 153–167.

Finney, M. A., McHugh, C. W., Grenfell, I. C., Riley, K. L., y Short, K. C. (2011). A simulation of probabilistic wildfire risk components for the continental united states. *Stochastic Environmental Research and Risk Assessment*, *25*(7), 973–1000.

Freund, Y., y Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*(1), 119–139.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Geurts, P., Ernst, D., y Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, *63*(1), 3–42.

Hernandez-Magallanes, I. (2010). *Integrating grouped and ungrouped data: the point process case* (Tesis Doctoral no publicada). PhD Dissertation, Statistics Department, University of California, Berkeley.

Larkin, N. K., O'Neill, S. M., Solomon, R., Raffuse, S., Strand, T., Sullivan, D. C., ... Ferguson, S. A. (2009). The bluesky smoke modeling framework. *International Journal of Wildland Fire*, *18*.

Preisler, H. K., y Ager, A. A. (2013). Forest-fire models. *Encyclopedia of Environmetrics*.

Reinhardt, E. D. (2003). Using fofem 5.0 to estimate tree mortality, fuel consumption, smoke production and soil heating from wildland fire. En *Proceedings of the 2nd international wildland fire ecology and fire management congress and 5th symposium on fire and forest meteorology* (pp. 16–20).

Robichaud, P., Elliot, W., Pierson, F., Hall, D., y Moffet, C. (2007). Predicting postfire erosion and mitigation effectiveness with a web-based probabilistic erosion model. *Catena*, *71*(2), 229–241.

Rothermel, R. C. (1972). A mathematical model for predicting fire spread in wildland fuels.

Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533.

Seal, H. L. (1967). Studies in the history of probability and statistics. xv the historical development of the gauss linear model. *Biometrika*, *54*(1-2), 1–24.

Sullivan, A. L. (2009). Wildland surface fire spread modelling, 1990–2007. 3: Simulation and mathematical analogue models. *International Journal of Wildland Fire*, *18*(4), 387–403.

Van Wagner, C. (1987). Development and structure of the canadian forest fireweather index system. En *Can. for. serv., forestry tech. rep.*