

CODER HOUSE

PREDICCIÓN DE *incendios en*

LI GA CIA



AUTORES

- X*** ALEJANDRO NUÑEZ
- X*** CAROLINA VINAGRE
- X*** CLAUDIA COURAU
- X*** FRANCISCO GUITIERREZ

TUTOR

- X*** JOSE IGNACIO LOPEZ SAEZ

Situación

Los incendios forestales son uno de los mayores problemas ambientales y producen un daño ecológico, económico y humano, irreparables. El calentamiento global aumenta la frecuencia e intensidad de los mismos. Por tal motivo es de suma importancia sistematizar modelos de acción según la ubicación (municipio, latitud y longitud) y factores ambientales (temperaturas, precipitaciones, presión atmosférica, vientos) para estimar y optimizar los recursos (medios, personal) utilizados a fin de minimizar tiempos de control y evitar heridos o pérdidas humanas.

Objetivo

En este proyecto realizamos tareas de Data Acquisition, Data Wrangling y EDA que nos guiarán a la obtención de un modelo de machine learning fiable predictor de la variable causa en función de las variables latitud y longitud geográfica, temperatura media, racha, sol, trimestre y año.

Con nuestro modelo pretendemos predecir cómo se relacionan las causas de incendios con las distintas variables para poder generar planes eficientes de control del fuego: brindar un sistema de soporte de decisión a la planeación estratégica de recursos destinados a incendios forestales.

Crear un modelo que ayude en la predicción de desarrollo de incendios a partir de sus causas haría mucho más eficiente la distribución de los recursos necesarios para la extinción y ayudaría también en la reducción de costes, daños y pérdidas. Aún teniendo en cuenta la gran dificultad que presenta el desarrollo de un modelo de predicción de incendios (por su complejidad y ser altamente no lineal debido a la incertidumbre asociada al comportamiento humano en relación al fuego), se intentará desarrollar un modelo basándonos en el histórico de datos de la Región de Galicia que es la de mayor afluencia de incendios en España y de la cual tenemos acceso a un dataset con abundantes registros.

Descripción de los datos

| VARIABLE | DESCRIPCIÓN |
|---------------|---|
| id | Identificador del incendio |
| superficie | Superficie forestal quemada en hectáreas |
| fecha | Fecha de detección del evento (yyyy-mm-dd) |
| lat | Latitud geográfica del origen del evento |
| lng | Longitud geográfica del origen del evento |
| idprovincia | Identificador de provincia |
| idmunicipio | Identificador de municipio |
| causa | Identificador de causa del incendio |
| muertos | Número de fallecidos en el incendio |
| heridos | Número de heridos en el incendio |
| time_ctrl | Tiempo que tomó entrar en fase de control del incendio (en minutos) |
| time_ext | Tiempo transcurrido hasta la extinción del incendio (en minutos) |
| personal | Número de personas que participaron en la extinción del incendio |
| medios | Número de medios terrestres y aéreos participantes en la extinción del incendio |
| gastos | Gastos asociados a la extinción del incendio |
| ALTITUD | Altitud sobre el nivel del mar (en metros) |
| TMEDIA | Temperatura media (°C) |
| PRECIPITACION | Precipitación diaria (en mm) |
| TMIN | Temperatura mínima (°C) |
| TMAX | Temperatura máxima (°C) |
| VELMEDIA | Velocidad media (m/s) |

| | |
|------------|---------------------------------------|
| RACHA | Velocidad ráfagas viento (en m/s) |
| SOL | Horas de Sol (horas) |
| Trimestre | Trimestre en que acontece el incendio |
| Mes | Mes en que acontece el incendio |
| Año | Año en que acontece el incendio |
| DIR_VIENTO | Dirección del viento |
| PRES_RANGE | Rango de presión atmosférica |

Exploratory Data Analysis

Llevamos a cabo una investigación inicial de los datos en búsqueda de patrones y anomalías mediante el cálculo de medidas estadísticas básicas y gráficas simples.

- Observamos las primeras y últimas 5 filas del dataset para tener un vistazo de los datos
- Ejecutamos códigos para obtener tamaño y para tipo de dato de cada columna.
- Buscamos datos nulos.
- La función *"describe"* nos muestra descriptores estadísticos básicos de nuestras columnas *"df.describe().round()"*

Hacer un análisis exploratorio de datos nos permite tener una idea de la distribución de variables en su conjunto (forma, sesgo, etc) . Utilizamos histogramas, gráficas de línea y boxplot en búsqueda de:

- Relaciones entre variables.
- Valores atípicos o puntos inusuales que puedan indicar problemas de calidad de los datos o conducir a descubrimientos interesantes.
- Patrones temporales

Data Wrangling

Ejecutamos códigos para ordenamiento y limpieza de nuestra *raw data* con el fin de detectar y eliminar errores de registro.

- Eliminamos columnas innecesarias.
- Verificamos que todos los datos cuenten con el mismo formato:
Transformamos la columna 'fecha' a tipo numérico para luego dividir mejor las variables

Una vez establecida la variable causa como *variable target* de nuestro proyecto, procedimos a la identificación de las variables más relevantes relacionadas con la primera.

- Agrupamos dentro del dataset según la variable **causa** -> **`df.groupby('causa').size()`**
- Mapeamos las variables categóricas que tienen un orden para que sean fácilmente adaptables a los modelos.
- Dado que la variable id municipio contiene gran cantidad de posibilidades, que decidimos dropear la columna para no agregar varianza a los datos. Latitud y longitud brindan ya la información de ubicación.
- Dividimos el dataset en dos (datos categóricos y numéricos) para optimizar su manejo.
- Utilizamos catboost para entender la importancia de las variables a la hora de predecir la causa. Observamos que a la hora de elegir una variable temporal, nos resulta conveniente inclinarnos por '**Trimestre**' ya que es mejor predictor por sobre '**mes**'.

Modelos utilizados

En este apartado vamos a comentar la implementación de diferentes modelos de machine learning de tipo supervisado (se conoce "ex ante" la variable respuesta) y enfocado en (nuestro) problema de clasificación.

Los valores a predecir (causas de incendio) están predefinidos y se les ha asignado un valor numérico para poder procesar la información:

- Negligencia: 1
- Intencionado: 2
- Causa desconocida: 3
- Rayo: 4
- Fuego reproducido: 5

Esto culminará con un análisis de la performance de cada uno y así poder optimizar la elección del modelo que se enviará a producción.

En todos los modelos, se dividió el dataset en dos: 70% de los datos fueron utilizados para entrenamiento y el 30% restante para testeo.

- **ÁRBOL DE DECISIÓN:** se trata de un modelo predictivo que plantea el problema desde distintas perspectivas de acción. Nos permite analizar de manera completa todas las posibles soluciones y provee de un diagrama para cuantificar el costo del resultado y su probabilidad de uso.

Una clara desventaja de éste modelo es que tiende a aprender muy bien los datos de entrenamiento pero su performance no es óptima a la hora de generalizar o predecir resultados sobre datos que no vio el árbol. Es por ello que aplicamos Random Forest.

- **RANDOM FOREST:** Es un método que ensambla varios árboles de decisión por lo que resulta en un modelo robusto para predecir nuevos datos. En otras palabras, todos los árboles son entrenados con muestras de datos distintas lo que permite obtener mejores predicciones combinando sus resultados y compensando errores.

Nuestro modelo mostró una mejora en las métricas con éste ensamble como veremos en el próximo apartado.

- **KNN:** Es un algoritmo de aprendizaje supervisado que clasifica cada dato nuevo en el grupo que corresponda, según tenga k vecinos más cerca de un grupo o de otro. Es decir, calcula la distancia del elemento nuevo a cada uno de los existentes, y ordena dichas distancias de menor a mayor para ir seleccionando el grupo al que pertenece. Este grupo será, por tanto, el de mayor frecuencia con menores distancias.

Una desventaja del modelo es la sensibilidad frente a gran cantidad de dimensiones o variables. Decidimos realizar un PCA, que es un método para poder “lidiar” con el problema mencionado; decidimos tomar la cantidad (19) de columnas que explican el 90% de la variabilidad de los datos, y así reducir la cantidad de variables.

- **CATBOOST:** modelo de regresión basado en potenciación del gradiente. Una de las grandes ventajas de éste modelo permite utilizar tanto variables categóricas como numéricas, es decir, no necesitamos transformar las variables categóricas en numéricas para el procesamiento.

Además en líneas generales, éste modelo no tiende a memorizar los datos y por lo tanto es bueno generalizando el aprendizaje.

- **MÁQUINA DE SOPORTE VECTORIAL:** permite encontrar la forma óptima de clasificar entre varias clases. La clasificación óptima se realiza maximizando el margen de separación entre las clases. Los vectores que definen el borde de esta separación son los vectores de soporte. Una de las desventajas que presenta es que, generalmente, podemos encontrarnos con un problema de superposición de clases que no permitiría una óptima separación de clases. Por lo tanto aplicamos el “truco” del kernel para añadir una nueva dimensión y solucionar este problema.

Técnicas de balanceo

```
dataset.groupby('causa').size()
```

| causa | |
|-------|-------|
| 1 | 534 |
| 2 | 11293 |
| 3 | 830 |
| 4 | 104 |
| 5 | 215 |

Debido al desbalanceo que muestra el dataset, vamos a aplicar técnicas algunas técnicas de “**Sampling**”:

- **Oversampling:** consiste en balancear la distribución de las clases añadiendo ejemplos de las clases minoritarias. Como ventaja se puede mencionar que no presenta pérdida de información pero puede repetir o crear muestras con ruido, además de aumentar el tiempo de procesamiento del conjunto de datos. Las técnicas aplicadas fueron:
 - **Random Over Sampling (ROS):** duplica al azar ejemplos de las clases minoritarias.
 - **Synthetic minority over-sampling technique (SMOTE):** como su nombre lo indica, crea nuevas muestras sintéticas de las clases minoritarias, utilizando la técnica de “vecinos más cercanos”.

Cabe aclarar que ambos métodos solo fueron aplicados sobre el “train” para no alterar los datos del test (ni su distribución).

- **Undersampling:** de manera inversa la técnica anterior, elimina muestras de la clase mayoritaria para balancear el dataset. En nuestro caso, se hizo manualmente una submuestra aleatoria de la clase mayoritaria para luego añadirle las clases minoritarias y obtener el dataset a procesar por los diferentes modelos. Como ventaja podemos decir que acorta los tiempos de procesamiento de datos pero hay una pérdida de información.

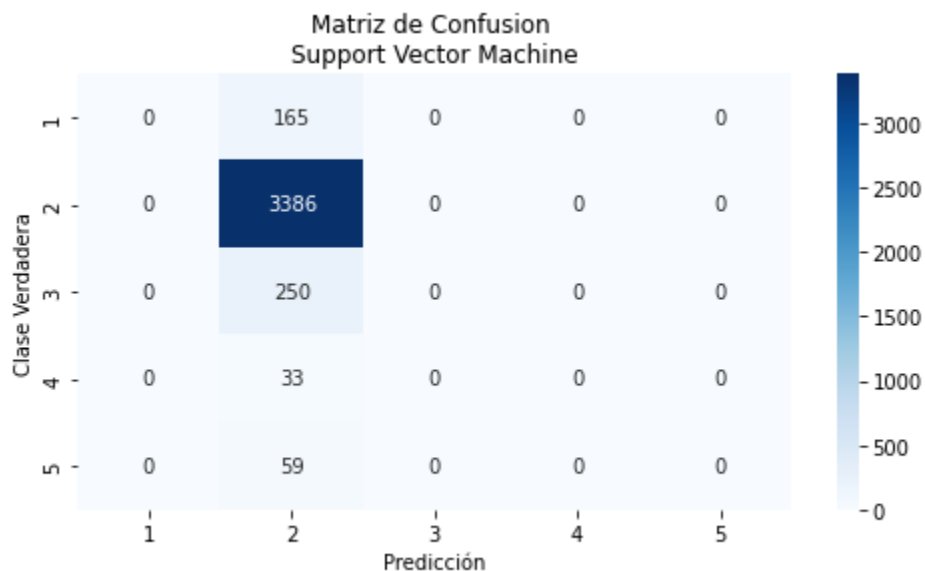
Métricas y Conclusiones

Vamos a comenzar describiendo las métricas utilizadas para culminar en los resultados y conclusiones que pudimos obtener.

- **Matriz de confusión:** en sí mismo no es una medida de desempeño como tal, pero casi todas las métricas de desempeño se basan en ella. Es una tabla que describe el rendimiento de un modelo supervisado de Machine Learning en los datos de prueba.

Se busca que los “falsos positivos” y “falsos negativos” sean los mínimos posibles; en nuestro caso particular el error del modelo a la hora de clasificar tiene el mismo peso relativo (en caso de enfermedades por ejemplo, sería más grave un falso negativo ya que podría no tratarse un caso a tiempo).

Fue importante realizar cada una de las matrices ya que en un golpe de vista pudimos observar, por ejemplo, que en el modelo “Support Vector Machine” predecía solamente una clase (mayoritaria). Con lo cual, no tendría ningún tipo de valor un modelo de éstas características a pesar de poseer resultados de métricas “aceptables”. Además nos permitió ver la posibilidad de aplicar técnicas de balanceo que intentaron paliar esta situación (a pesar de que los resultados no fueron óptimos para este modelo).



- **Accuracy:** Es el porcentaje total de elementos clasificados correctamente. Es una medida sencilla e intuitiva pero que tiene una gran desventaja: no es buen parámetro para datasets desbalanceados.
- **Recall o exhaustividad:** Es el número de elementos identificados correctamente como positivos del total de positivos verdaderos. Buscamos que sea un 100% (independientemente de la métrica precisión) cuando queremos minimizar los falsos negativos.

- **Precisión:** Es el número de elementos identificados correctamente como positivo de un total de elementos identificados como positivos. Inversamente a la métrica anterior, busquemos que sea un 100% cuando queremos minimizar los falsos positivos.
- **F1-Score:** El valor F1 se utiliza para combinar las medidas de precisión y recall en un sólo valor, a través de la media armónica entre ambos valores. **Creemos que es la métrica ideal para nuestro problema ya que, como mencionamos antes, le damos la misma importancia a la precisión y a la exhaustividad.**
- **ROCAUC:** la curva AUC - ROC es una medida de rendimiento para los problemas de clasificación en varias configuraciones de umbral. ROC es una curva de probabilidad y AUC representa el grado o medida de separabilidad. Indica cuánto es capaz el modelo de distinguir entre clases. Un modelo excelente tiene AUC cerca de 1, lo que significa que tiene una buena medida de separabilidad. Un modelo pobre tiene un AUC cercano a 0, lo que significa que tiene la peor medida de separabilidad. Y cuando AUC es 0.5, significa que el modelo no tiene capacidad de separación de clases.
- **Tiempo:** permite optimizar la elección del modelo al mostrar el costo (o tiempo/recursos) que toma implementar técnicas o modelos más sofisticados frente al beneficio obtenido en mejora de métricas.

| Método | Accuracy | Recall | Precision | ROCAUC | F1-Score | Tiempo |
|--|----------|----------|-----------|----------|----------|------------|
| Random Forest - Random-Over-Sampling | 0.866684 | 0.866684 | 0.791670 | 0.748906 | 0.814361 | 51.045010 |
| Random Forest | 0.871821 | 0.871821 | 0.886013 | 0.741989 | 0.813550 | 3.499104 |
| Random Forest - SMOTE | 0.855381 | 0.855381 | 0.787795 | 0.753814 | 0.813520 | 65.163999 |
| k-Nearest-Neighbor | 0.862831 | 0.862831 | 0.789151 | 0.595602 | 0.810536 | 0.002956 |
| Support Vector Machine | 0.869766 | 0.869766 | 0.886727 | 0.418798 | 0.809185 | 5.872736 |
| Catboost | 0.826355 | 0.826355 | 0.790591 | 0.517953 | 0.806863 | 12.660552 |
| Arbol de Decision | 0.772669 | 0.772669 | 0.791550 | 0.570500 | 0.781815 | 0.171534 |
| Catboost - SMOTE | 0.703057 | 0.703057 | 0.798973 | 0.724738 | 0.744262 | 13.328406 |
| Support Vector Machine - SMOTE | 0.666838 | 0.666838 | 0.833735 | 0.501958 | 0.708688 | 755.390434 |
| Support Vector Machine - Random-Over-Sampling | 0.665810 | 0.665810 | 0.834127 | 0.463237 | 0.708251 | 728.200357 |
| Catboost - Random-Over-Sampling | 0.499101 | 0.499101 | 0.829066 | 0.744157 | 0.597705 | 8.424725 |
| Catboost - Random-Under-Sampling | 0.489593 | 0.489593 | 0.552612 | 0.751714 | 0.504501 | 6.245008 |
| Random Forest - Random-Under-Sampling | 0.556561 | 0.556561 | 0.497379 | 0.732526 | 0.463363 | 1.326793 |
| k-Nearest-Neighbor - Random-Under-Sampling | 0.507692 | 0.507692 | 0.457997 | 0.632401 | 0.460659 | 0.001743 |
| Arbol de decision - Random-Under-Sampling | 0.441629 | 0.441629 | 0.454825 | 0.574690 | 0.447603 | 0.047004 |
| Support Vector Machine - Random-Under-Sampling | 0.524887 | 0.524887 | 0.750619 | 0.473166 | 0.361346 | 1.151571 |

Elección del modelo y algunas reflexiones:

- En primer lugar notamos que la técnica Random Under Sampling fue la que arrojó los peores resultados en métricas. Con lo cual podemos afirmar que eliminar datos para intentar balancear las clases no es recomendable para nuestro problema.
- Es importante complementar las distintas métricas y visualizar la matriz de confusión ya que se puede obtener buenas métricas a pesar de que el modelo no sea útil (específicamente nos referimos al caso de la Máquina de Soporte Vectorial que mencionamos en párrafos anteriores).
- Random Forest es el modelo que mejor perfoma en la métrica F1 Score; de todas formas notamos que en el caso de aplicación de las técnicas ROS (mejor F1 observado) y SMOTE, el tiempo requerido es muy alto frente a otros modelos.
- Por lo tanto, podríamos elegir entre dos modelos ganadores: **KNN y Random Forest**. El primero tiene tiempos muy bajos; A pesar de ello, notamos que (si bien Random Forest no tiene el valor mínimo de tiempo, tampoco es demasiado costoso) **Random Forest le saca una leve ventaja en F1** (además de una mayor simpleza a la hora de construirlo) **y termina siendo el modelo elegido**

Mejoras:

A partir de la mirada integral que alcanzamos al realizar el proyecto, creemos conveniente implementar a futuro otras técnicas que busquen atacar el problema de desbalanceo presente en los datos. Sugerencias:

- a) se podría enfocar el problema como de clasificación binaria (dejando atrás las 5 clases de la variable target actual) y alcanzar un mejor balanceo en los datos.
- b) se podría realizar un approach al estilo divide and conquer, atacando el problema con un stackeo de modelos: el primero predice si la clase es mayoritaria o minoritaria (binario) y, de resultar clase minoritaria, el segundo modelo se encarga de decidir a cual clase pertenece (multiclase balanceada).
- c) Además se podría aplicar un Stratified K-Fold Cross-Validation.