FACULTAD DE COMERCIO Y TURISMO

Máster en Big Data y Data Science. Aplicaciones al Comercio, Empresa y Finanzas.

TRABAJO FIN DE MÁSTER

"Prevención de incendios: Clasificación de tipos de incendios en función de la superficie afectada. Una aproximación mediante técnicas de Machine Learning"

Por:

Ricardo Afonso Spinola

Nerea Gómez Miguel

Pablo Oliva Gómez

Samanta Palango Caiza

Celia Regueiro Emperador

Malena Rodriguez Carrancio

Madrid, 15 de septiembre de 2021

INDICE

TABLA DE ILUSTRACIONES

1. INTRODUCCIÓN

1.1 MOTIVACIÓN E HISTORIA

La idea de un verano en España sin ver noticias de bosques ardiendo, hidroaviones y bomberos forestales agotados y cubiertos de hollín está cada vez más alejada de la realidad. Anualmente se queman una media de 100.000 hectáreas en nuestro país y el cambio de las condiciones climáticas y la falta de políticas adecuadas de prevención nos hace visualizar un futuro con incendios cada vez más grandes y difíciles de extinguir. De hecho, ya nos empieza a ser familiar el concepto de incendios de sexta generación.

Cuando nos planteamos el tema sobre el que realizar el proyecto, ya comenzaban a aparecer las noticias de los primeros incendios del verano y nos pareció un problema interesante a tratar y del que aparentemente se podrían conseguir datos públicos con cierta facilidad.

En un primer momento surgió la idea de intentar predecir el riesgo de que ocurriera un incendio en un punto determinado, pero una lectura más amplia sobre el tema nos hizo descartar este aspecto ya que más del 80% de los incendios son provocados o debidos a la acción del hombre, más que por efectos externos o naturales. Nos pareció más razonable asumir que en determinado lugar habrá un incendio e intentar predecir cuántas hectáreas se quemarían de acuerdo a las condiciones meteorológicas previstas.

Revisando las herramientas desarrolladas relacionadas con este tema vimos que AEMET proporciona diariamente un mapa que muestra los niveles de riesgo de incendio previstos generado a partir de datos meteorológicos y de modelos numéricos, como se ve en la *ilustración 1*.

Ilustración 1. Mapa de previsión de incendios AEMET

Este mapa es una foto fija, sólo muestra un día y no permite interactuar con él, lo que nos llevó a plantearnos la posibilidad de crear una aplicación similar, pero prediciendo el número de hectáreas quemadas en cada municipio y para los próximos 7 días, que son los datos de predicción meteorológica que facilita AEMET.

1.2 OBJETIVO

El objetivo principal del estudio es, asumido que en determinado lugar se va a declarar un incendio, encontrar un modelo de clasificación multiclase que sea capaz de predecir el tipo de incendio (número de hectáreas quemadas) que ocurriría en base a una serie de características dadas.

Además, se pretende crear una aplicación en la que el usuario pueda solicitar la predicción de superficie quemada en una provincia para los próximos 7 días y visualice el resultado en un mapa interactivo.

1.3 DESCRIPCIÓN DEL TRABAJO

La mayor dificultad que nos hemos encontrado durante la ejecución del proyecto ha sido conseguir los datos con los que empezar a modelar, ya que hemos tenido que recurrir a diversas fuentes para lograr un conjunto de datos consistente y con variables diversas.

Nuestro punto de partida era el histórico de incendios ocurridos en España desde 2001 a 2015. Necesitábamos agregarle las mediciones meteorológicas reales que hubo en el momento del incendio. Para ello, primero hicimos una llamada a la API de AEMET y obtuvimos las medidas que tomaron durante esos 15 años, más de 1.2M de registros. Por otra parte, hicimos otra llamada a la API para obtener la recopilación de estaciones meteorológicas existentes.

Unir las estaciones con sus medidas fue sencillo, pero no había ningún tipo de identificador común con la base de datos de incendios. Para resolver esto creamos una función que calculaba las distancias entre cada incendio y todas las estaciones meteorológicas y le asignaba la más cercana. Así pudimos unir a cada incendio las variables medidas por su estación más cercana.

El inconveniente es que, de 82.600 incendios, 18.000 tenían datos nulos ya que su estación más cercana o no existía en el momento del incendio, o ese día no había tomado

datos. Para resolverlo creamos un conjunto de datos propio con todas las estaciones y las estaciones más próximas o similares a ellas. De este modo, si un incendio tenía datos nulos por su estación más próxima le asignábamos los de la siguiente.

Aprovechamos la nueva base de datos creada para añadir al dataset las variables de zonas meteorológicas y el código de municipio que el INE asigna a cada municipio. Esta variable nos ha permitido posteriormente unir y combinar todos los datos. Del Instituto Geográfico Nacional (IGN) hemos conseguido la información asociada a cada municipio, así como su representación geospacial, y los hemos unido también al dataset original. El objetivo era tener el mayor número de variables útiles.

Una vez que tuvimos un conjunto de datos que nos convencía, se llevó a cabo un análisis profundo de los datos y su posterior limpieza, así como la creación de nuevas variables que pudieran ser útiles en la creación del modelo. Tras obtener el conjunto de datos final, realizamos una selección de variables basándonos en la correlación con la variable objetivo y la colinealidad existente entre algunas de las variables predictoras y así continuar con la elaboración de un modelo de clasificación multiclase por el cuál podremos saber la superficie quemada dadas unas condiciones.

Por último, hemos automatizado el proceso para simular una aplicación real, mediante la cual hacemos una nueva llamada a la API de AEMET para obtener la predicción meteorológica de los próximos 7 días de los municipios deseados con el fin de obtener la superficie quemada para cada uno de ellos en cada día y poder visualizarlo e interactuar con los datos en Tableau.

En la *ilustración 2*, se puede ver el esquema de todo el proceso realizado para completar el proyecto.

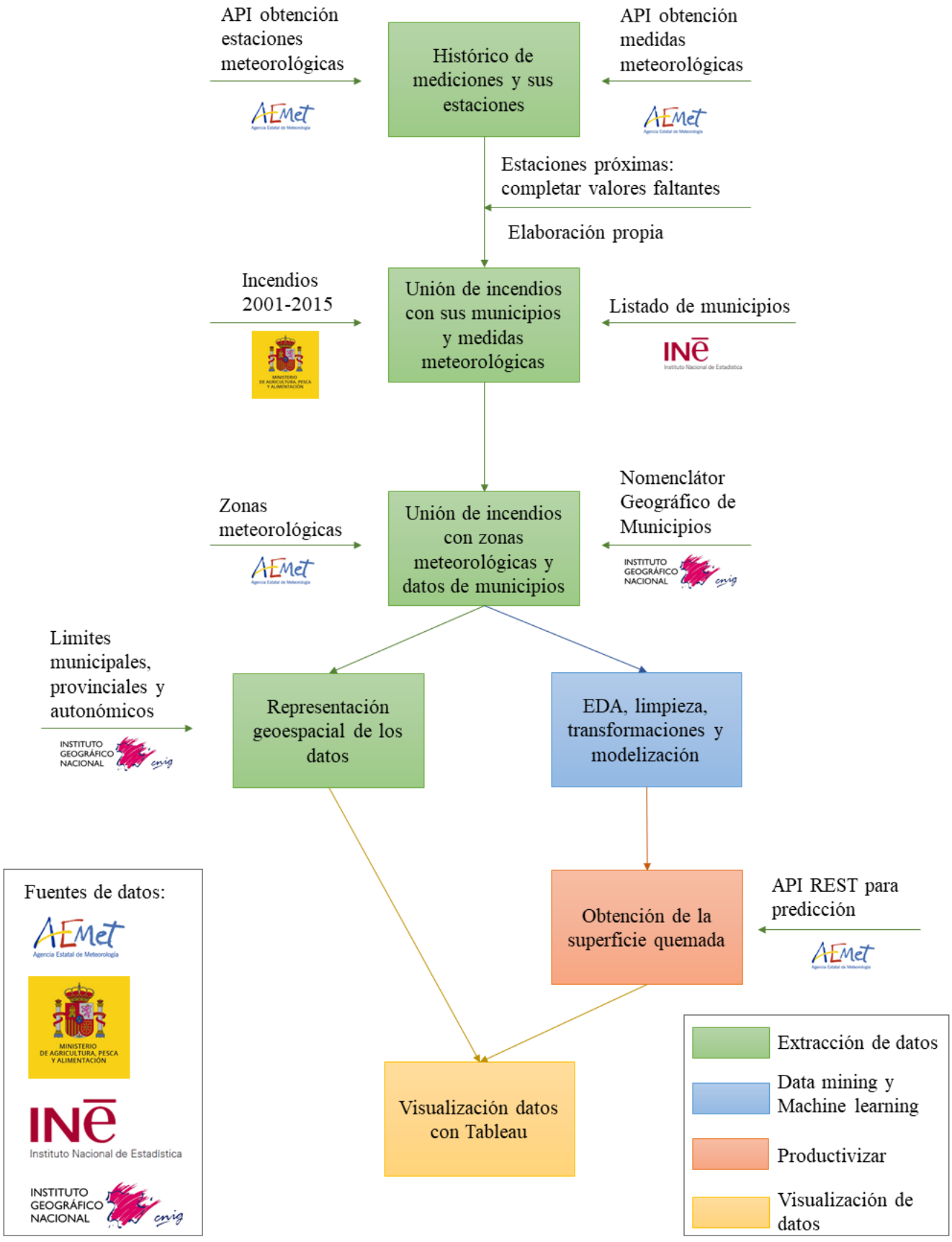

Ilustración 2. Flujo de trabajo

2. EXTRACCIÓN, PREPARACIÓN Y UNIÓN DE DATOS DE DISTINTAS FUENTES

Durante toda esta etapa hemos tenido muy presente las continuas advertencias que nos hacían los profesores en clase de que la mayor parte de la tarea y tiempo de trabajo la ocupaba la obtención y limpieza de los datos, ya que como comentábamos antes es donde mayores dificultades hemos encontrado. Sin embargo, creemos que podemos estar muy satisfechos por habernos enfrentado a un caso que podría ser un trabajo real, en el que hemos tenido que conseguir datos de distintas fuentes y formatos, y hemos logrado resolverlo con éxito.

Los datos del histórico de incendios en España entre el año 2001 y el 2015 los hemos conseguido de la página web Datos Civio[1]. Se trata de la Estadística General de Incendios Forestales (EGIF) que elabora el CCINIF a partir de la información anual suministrada por las CCAA al Ministerio a partir de los partes de incendio que rellenan los agentes forestales. La EGIF se actualiza y publica cada 5 años, pero aún no están disponibles los datos hasta 2020, quizá debido a la pandemia.

Esta base contiene información sobre 82.640 incendios y las distintas variables que describen los mismos, entre ellas la superficie quemada en hectáreas, que será nuestra variable objetivo. Se comprobó que existían incendios mal ubicados, con sus coordenadas geográficas en América o la Antártida, por lo que se corrigieron sustituyéndolas por las coordenadas del municipio donde sucedió el incendio.

En segundo lugar, queríamos extraer de una fuente fiable las mediciones meteorológicas reales que se dieron en las fechas de dichos incendios para poder entender la influencia que puede tener el clima en ellos. Para ello, decidimos utilizar la *API* de AEMET[2] que nos permite recoger la información meteorológica de los años que requeríamos, mediante un llamado a su *API REST* a través de código realizado en *Python* (*ver anexo 1*), donde se obtiene un estándar *http* por mes de búsqueda consultado. Este código se ejecuta tantos años como hay hasta obtener la información de los 15 años requeridos. Los datos se obtienen en formato *JSON*, por lo que los transformamos a data frame para poder tratarlos (*ver anexo 2*). Finalmente, creamos una lista que contiene todos los data frames elaborados, obteniendo así todas las mediciones realizadas de 2001 a 2015, más de 1.2M de registros (*ver anexo 3*).

El problema es que no había cómo unir estas mediciones a los incendios. Observamos que AEMET también permite descargar a través de su *API* el listado de estaciones meteorológicas que existen incluyendo sus coordenadas geográficas y ubicación. Se descargaron más de 290 estaciones. El tratamiento de esta información es igual que en el caso anterior, obtenemos un estándar *http* y los datos en formato *JSON* que transformamos a data frame para su posterior uso y lo unimos a los datos medidos a través de la variable indicativo que identifica a las estaciones meteorológicas.

Para incorporar a cada incendio las condiciones climatológicas que se dieron ese día, necesitamos saber cuál es la estación meteorológica más cercana a dicho incendio ya que será la que nos aporte los datos con mayor precisión. En este punto es de gran utilidad la longitud y la latitud tanto de los incendios como de las estaciones, pero el formato de ambas variables en las distintas bases es distinto por lo que tuvimos que unificar criterios. Además de esto, tuvimos que eliminar las estaciones creadas en fechas posteriores a 2015 debido a que no existe información meteorológica para estas estaciones antes de este año y podría darse el caso de relacionar estas nuevas estaciones a algún incendio dado en esa zona, obteniendo de esta forma valores nulos y no la información necesaria.

Gracias a la creación de una función (*ver anexo 4)* que calcula cuál es la distancia entre las coordenadas del incendio dado y las coordenadas de todas las estaciones meteorológicas, podemos encontrar la más cercana a cada incendio y asignarle el indicativo de esa estación para unir, a través de la fecha y del indicativo, las estaciones, sus mediciones y los incendios.

Al comprobar la base de datos resultante, observamos que hay más de 18.000 filas con valores nulos en todas las variables referidas a las medidas meteorológicas. Esto sucede porque hay estaciones que han desaparecido o se han creado en una fecha posterior al incendio con el que se ha relacionado. Para solucionar esto, hemos creado una base de datos que contenga todas las estaciones meteorológicas y sus dos estaciones más cercanas en función de las zonas meteorológicas que utiliza AEMET para la predicción. Estas zonas agrupan municipios con similares características climáticas. El criterio seguido es:

- Si la estación está en la misma zona que la estación más cercana, incluimos esta.
- Si no hay otra estación en su zona, buscamos otra cercana que tenga características similares. Por ejemplo, que sea costera, de montaña…

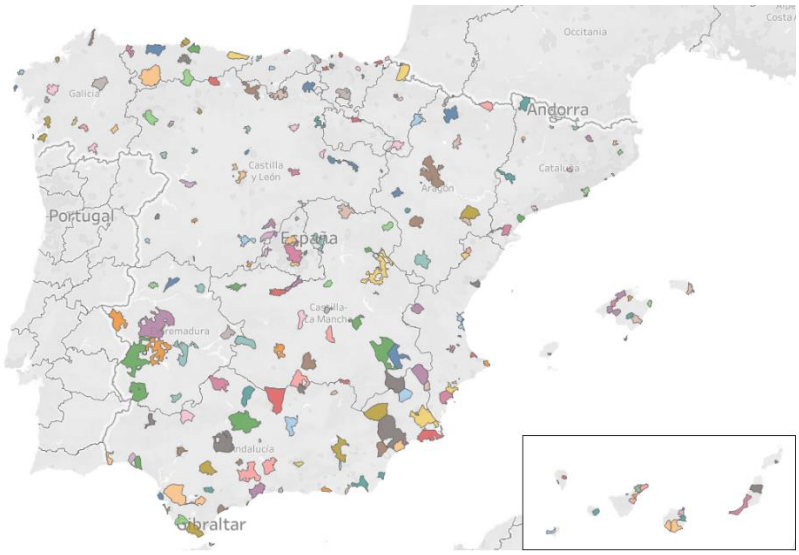

Ilustración 3. Municipios en los que están ubicadas las estaciones meteorológicas.

Una vez creada esta base hacemos un bucle y le decimos que, si incendios tiene los datos nulos en las condiciones meteorológicas, los rellene con los de la estación más cercana; y, si también son nulos, con la siguiente estación. De esta manera lo conseguimos reducir a 1.500 nulos que decidimos eliminar ya estimamos que el coste de rellenarlos era mayor que el beneficio. A su vez, aprovechamos para unir al dataset la variable zona meteorológica.

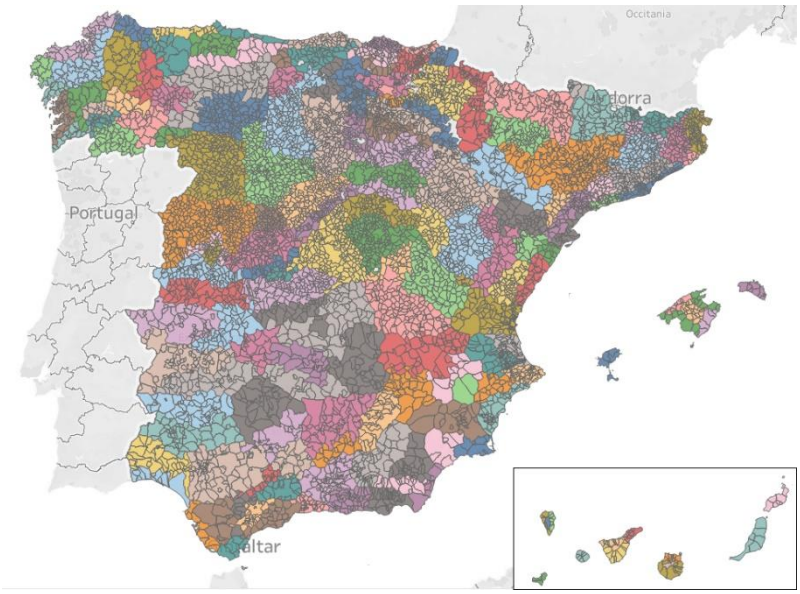

Ilustración 4. Zonas meteorológicas definidas por AEMET

Por último, descargamos de la página del IGN[3] tanto el Nomenclátor Geográfico de Municipios, que nos permitió añadir más variables a los incendios (la superficie del

municipio o su población), así como los ficheros en formato *shapefile* de límites municipales, que nos permitirá posteriormente representar gráficamente los datos. Se facilita un fichero para la península y Baleares y otro para Canarias, por lo que hubo que unirlos y además agregarles la información de los municipios con el fin de poder dibujarlos.

3. METODOLOGÍA

3.1 VISUALIZACIÓN PREVIA DE DATOS

Contamos con las siguientes variables en nuestra base de datos final:

- **id:** Asignación de identificador único para las observaciones.
- **superficie:** Superficie forestal quemada en hectáreas. Variable objetivo.
- **tipoincendio:** Tipo de incendio por tamaño. Conato, Incendio y Gran Incendio.
- **fecha:** Fecha de detección del incendio (yyyy-mm-dd).
- **latitude:** Latitud geográfica del origen del incendio.
- **longitude:** Longitud geográfica del origen del incendio.
- **comunidad:** Nombre de CC.AA.
- **provincia_x:** Nombre de provincia.
- **idmunicipio:** Identificador de municipio.
- **municipio:** Nombre de municipio.
- **zona:** Zonas de España.
- **causa:** Causas de incendio.
- **time_ctrl:** Tiempo transcurrido hasta entrar en fase de control del incendio (minutos).
- **time_ext:** Tiempo transcurrido hasta la extinción del incendio (minutos).
- **personal:** Número de personas que han participado en la extinción del incendio.
- **medios:** Número de medios terrestres y aéreos que han participado en la extinción del incendio.
- **idcomunidad:** Identificador de CC.AA.
- **idprovincia:** Identificador de provincia.
- **indicativo:** Identificador de estación meteorológica.
- **distanciaMin:** Distancia del incendio a estación meteorológica.
- **nombre:** Municipio estación meteorologica.
- **horatmax:** Hora temperatura máxima (horas).
- **dir:** Dirección media del viento.
- **velmedia:** Velocidad media del viento (km/h).
- **racha:** Velocidad máxima del viento (km/h).
- **horaracha:** Hora de la velocidad máxima del viento (horas).
- **presMax:** Presión máxima atmosférica (hPa).
- **horaPresMax:** Hora de la presión máxima atmosférica (horas).
- **presMin:** Presión mínima atmosférica (hPa).
- **horaPresMin:** Hora de la presión mínima atmosférica (horas).

- **latitud:** Latitud geográfica de estación meteorológica.
- **longitud:** Longitud geográfica de estación meteorológica.

3.2 ANÁLISIS EXPLORATORIO DE LOS DATOS

Una vez hemos obtenido la base con los datos necesarios para poder trabajar, iniciamos una primera fase analítica conocida como Análisis Exploratorio de los Datos o *EDA*. Que con la ayuda de paquetes como *DataExplorer y visdat*, entre otros, nos permitirán visualizar los datos para poder sacar las primeras conclusiones.

El objetivo es detectar y entender los datos y su estructura, así como detectar nulos y variables mal codificadas. Antes de realizar el *EDA* ha sido necesario recodificar algunas de las variables que presentaban los nulos como "". Una vez corregidos estos valores, llevamos a cabo una primera visualización del dataset, que nos permite observar las columnas discretas (un 26.09% del total de los datos), columnas continuas (73.91%), el porcentaje de filas completas (74.47%) y los datos faltantes (4.9%). Se pueden comprobar estos valores gráficamente en la *ilustración 5*.

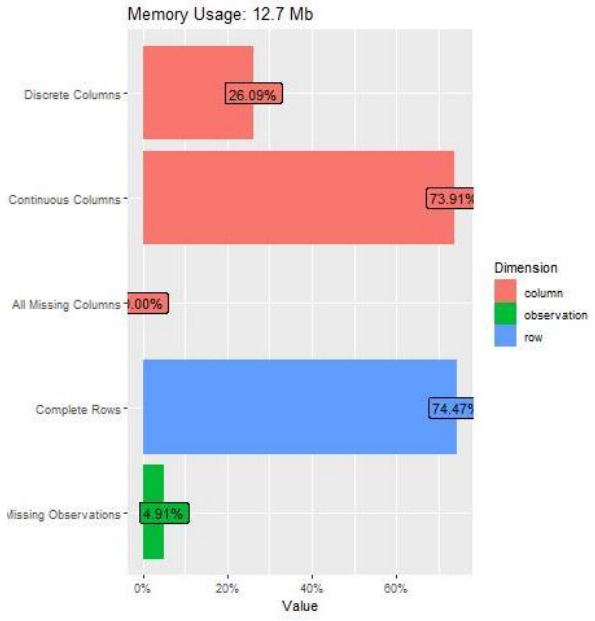

Ilustración 5. Visión global del conjunto de datos

Tras esta visualización inicial de los datos podemos hacer un estudio en más profundidad de las características de las variables, para ello hacemos un análisis individual y colectivo de las mismas.

En el estudio individual podemos observar las variables categóricas y numéricas. En el primer caso, mediante un histograma, se visualizan las variables numéricas (*ilustración 6*), se pueden ver las distribuciones de los valores de las variables. Mientras que para las variables categóricas hemos analizado la frecuencia de los diferentes niveles de las mismas (*ilustración 7*), siendo **fecha** y **municipio** las variables que presentan más niveles de categorías.

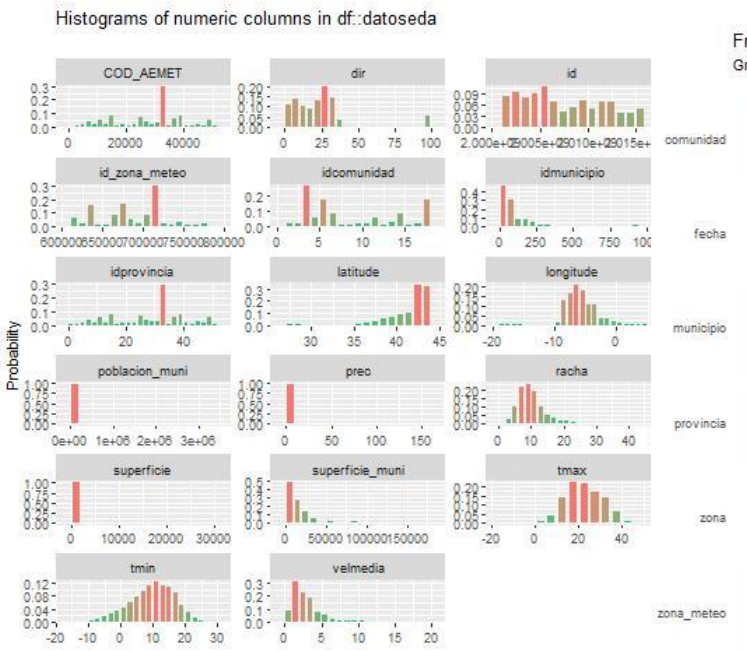

Ilustración 7. Frecuencia de las variables numéricas

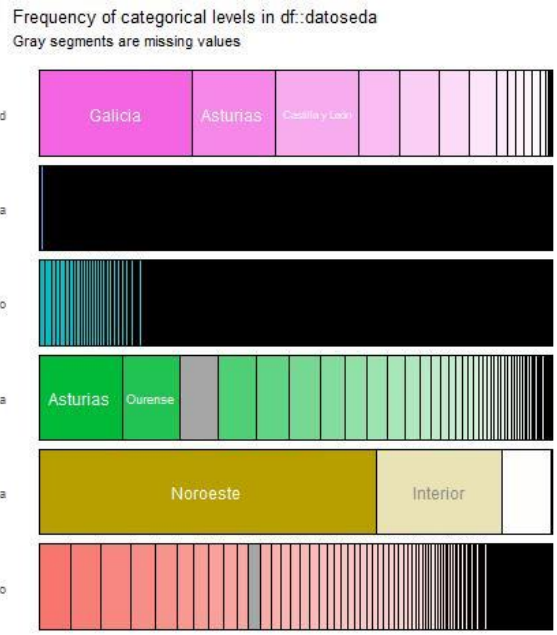

Ilustración 6. Distribución de las variables categórica.

Para el análisis en conjunto o análisis multi-variante, hemos calculado la correlación entre las variables a través de la matriz de correlación. En esta matriz podemos observar su valor para cada uno de los elementos (*ilustración 8*). A primera vista podemos afirmar que los elementos con mayor correlación son las temperaturas; la racha, velocidad y dirección del viento y las variables relacionadas con el espacio geográfico del incendio.

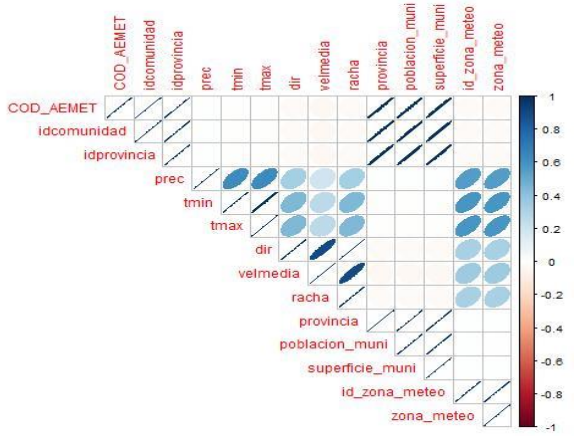

Ilustración 8. Matriz de correlación.

3.3 DEPURACIÓN DE LA BASE DE DATOS Y TRATAMIENTO DE DATOS NULOS

Se requiere de una fase previa de depuración de datos para poder estudiarlos de forma adecuada, con la finalidad de modificar de forma correcta nuestros datos. La depuración, o limpieza de datos, nos permite detectar aquellos datos incorrectos, incompletos o que son irrelevantes para nuestro estudio, y corregirlos.

En esta fase tomaremos decisiones a la hora de rellenar datos nulos o eliminar variables sin importancia para nuestro estudio. En este apartado, también se utilizarán técnicas con el fin de tener una buena codificación de los datos, gracias a la imputación de valores nulos, categorización de valores numéricos y transformación de variables, entre otras.

Es necesario determinar si existen variables que tenemos que eliminar en el caso de que estas sean innecesarias para nuestro trabajo. Por ejemplo, algunas de las variables eliminadas de entrada son **horaPresMax, horaPresMin, horaracha, PresMax y PresMin** con escaso interés para nuestro trabajo, así como la variable **sol** que presenta más de 50% de datos nulos. En la *ilustración 9* se pueden observar el resto de los datos que presentan nulos, junto con su frecuencia relativa.

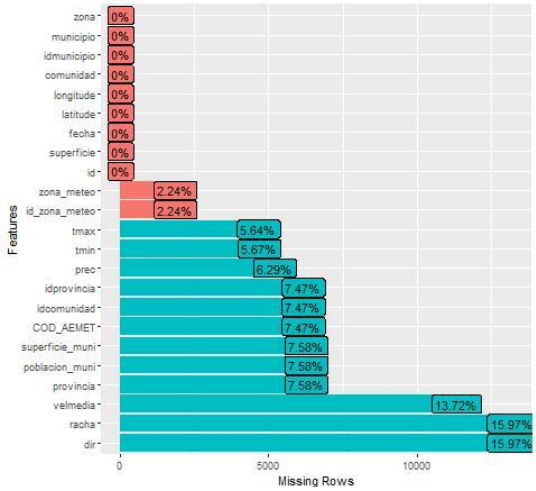

Ilustración 9. Tabla de variables que presentan valores nulos

En primer lugar, para el tratamiento de algunos de los datos faltantes, hemos optado por llevar a cabo una imputación con sus medias, en función del mes y de la provincia correspondiente, en lugar de calcular su media de forma general, lo que nos lleva a conseguir observaciones más exactas para imputar los datos faltantes. Las variables que se han imputado con este método son las temperaturas (**tmin** y **tmax**) y la precipitación (**prec**).

Antes de imputar la variable **racha**, y teniendo en cuenta la variable **velmedia**, asumimos que, si el viento medio es cero, la racha también será cero. Así mismo, para imputar esta variable, se ha optado por hacerlo de forma aleatoria entre la velocidad media de cada registro y racha máxima registrada en nuestro conjunto de datos.

Por otro lado, para las variables **idprovincia** e **idcomunidad**, se han rellenado los nulos asignado su valor correspondiente, el cual es conocido, pues se tratan de identificadores únicos.

El resto de nulos se han imputado de forma aleatoria.

3.3 TARGET ENGINEERING

El objetivo de este trabajo es conseguir, como se ha mencionado anteriormente, predecir la superficie quemada en hectáreas en un posible incendio en función de las variables que se encuentran en la base de datos final. Esta, por tanto, será la variable objetivo del modelo.

Originalmente esta variable de superficie es una variable numérica que recoge las hectáreas alcanzadas por el incendio. Debido a su distribución sesgada a la izquierda, se decidió categorizar la variable, tramificándola en 6 grupos diferentes.

Los incendios se suelen clasificar en tres grupos según su tamaño en función de las hectáreas quemadas: conato (inferior a una hectárea), incendio (entre 1 y 500 hectáreas) y gran incendio (superior a 500 hectáreas).

Debido a que con esta clasificación los datos no estaban balanceados, se decidió dividir la categoría incendio en cuatro nuevas: incendio entre 1 y 10 hectáreas (Incendio < 10 ha), incendio entre 10 y 30 hectáreas (Incendio < 30 ha), incendio entre 30 y 100 hectáreas (Incendio < 100 ha) e incendio entre 100 y 500 hectáreas (Incendio < 500 ha). Hemos decidido dividirlo de esta manera, dado que los recursos necesarios para extinguir un incendio de entre 1 y 10 hectáreas no son los mismos a los necesarios en un incendio entre 100 y 500 hectáreas.

3.4 FEATURE ENGINEERING

Una vez preparada la variable objetivo, se han realizado transformaciones sobre las variables independientes y se han creado nuevas variables a partir de las mismas. Realizar *feature engineering* a las variables puede mejorar significativamente el rendimiento de los modelos[4].

Para la mayoría de los modelos, es necesario transformar todas las variables categóricas a numéricas para que funcionen.

Para la transformación de las variables categóricas presentes en el dataset, se ha utilizado *label encoding*. El *label encoding* es un proceso de conversión numérica de los niveles de cada una de las variables categóricas[5]. La variable **prec** se ha transformado en binaria en función de si hay lluvia (prec > 0) o no (prec = 0). La variable **dir** se ha pasado a categórica teniendo en cuenta la orientación con respecto a los puntos cardinales de la rosa de los vientos.

Para las variables continuas hemos realizado las siguientes transformaciones:

- **Fecha**: se han obtenido la variable **mes** (haciendo referencia al número de este), **trimestre** (ya que hemos observado que en los meses de invierno y verano se producen más incendios), y una variable *dummy* refiriéndose a si es **verano** o no ya que queremos comprobar si en esta estación, al tener temperaturas más altas, se ve afectado el número de incendios.
- **Longitude/Latitude:** se ha obtenido la **distancia** entre puntos mediante la siguiente fórmula: $dist = \sqrt{latitude^2 + longitude^2}$.
- **Poblacion_muni/Superficie_muni:** al no ser capaces de acceder a una variable tipo suelo, a través de la **densidad de población** podríamos deducir si esa superficie poblacional está más o menos edificada. La hemos calculado a través de la fórmula: $densPob = \frac{poblacion\_muni}{superficie\_muni}$.
- **Racha** y **Vel_media:** se ha obtenido la **escala de Beaufort**, aquella que clasifica el viento en función de los km/h. Esta variable se ha calculado tanto con la racha como la vel_media y se han creado variables nuevas tanto numéricamente como categóricamente[6].
- **Tmin/Tmax**: se ha calculado la **temperatura media** mediante la siguiente fórmula: $tmed = \frac{tmax+tmin}{2}$.
- **Racha/tmax**: se ha obtenido la **sensación térmica** usando la fórmula: $sensaTerF = 13.1267 + (0.6215 * tmax) - (11.37 * racha)^{0.16} + (0.3956 * tmax * racha)^{0.16}$
- **Tmin, Tmax y Tmed**: se ha calculado el **tipo de temperatura**, en un rango de calor extremo a frío extremo, tanto categóricamente como numéricamente.

- Otras transformaciones:
  - Combinaciones de variables que puedan resultar en **condiciones óptimas** para darse un incendio. Proponemos un ejemplo de los múltiples que hemos llevado a cabo: alta velocidad viento – temperatura alta - poca precipitación. Como finalmente ninguna de las combinaciones resultaba significativa, hemos decidido descartarlas del modelo.
  - **Humedad relativa**: se ha intentado calcular la humedad relativa, pero para ello es necesario tener datos de la presión atmosférica (de los que sí disponíamos) y de la presión punto de rocío, para la cual se necesitan o bien la presión del vapor o la presión saturada, variables que no se encuentran en el dataset y que no pueden calcularse a partir de otras.

3.5 NORMALIZACIÓN

Dado que la escala en la que se miden las variables difiere, se ha decidido optar por la estandarización de estas, mediante la técnica *min_max_scaler* (*ver anexo 5*), la cual escala cada variable a un rango dado. El estandarizado de las características implica que las variables numéricas cuenten con una media de cero y una varianza unitaria, mediante técnicas de centralización y escalado. Lo que se persigue es una unidad de medida común y comparable entre las distintas variables del dataset[7].

4. MODELIZACIÓN DE LOS DATOS

4.1 SELECCIÓN DE VARIABLES

Antes de lanzar modelos, es necesario seleccionar las variables que se van a incluir en los mismos tanto para evitar colinealidad y correlación como para eliminar aquellas variables que no aportan información.

Uno de los métodos usados es el análisis univariante mediante la prueba de *Chi-Cuadrado*. Esta prueba estadística se basa en seleccionar aquellas variables con una relación más fuerte con la variable objetivo[8].

Para continuar con la selección, se ha procedido a calcular las correlaciones de los predictores con la variable objetivo y las correlaciones por pares. Fijándonos en los resultados de la *ilustración 10*, comprobamos que ninguno de los predictores mantiene una relación fuerte con la variable objetivo. Además, la correlación entre las variables

independientes es muy elevada. Por ende, sospechamos que estas variables no serán de ayuda a la predicción correcta del modelo.

	supQuemada
supQuemada	1.000000
id	0.108105
tmax	0.076227
tmaxCat	0.071207
tmin	0.059425
tminCat	0.057226
trimest	0.046444
fe_month	0.046090
idcomunidad	0.040371
NATCODE	0.040113
id_zona_meteo	0.037435
racha	0.031525
densPob	0.024698
beauRacha	0.022456
beauVelmed	0.022456
velmedia	0.022364
superficie_muni	0.022020
altitud	0.019399
dir	0.016808
COD_AEMET	0.015719
idprovincia	0.015621
latitude	0.015533
dist	0.014955
idmunicipio	0.013779
prec	0.011315
poblacion_muni	0.007793
longitude	0.002051
zona	0.000321

Ilustración 10. Correlación con la variable objetivo

Dada la imposibilidad de incluir otras variables como, por ejemplo, el tipo de suelo, debido a la complejidad de la obtención de estos datos, ya sea por la accesibilidad a los mismos o la dificultad a extraerlos, no se ha podido mejorar la calidad del dataset y se ha decidido optar por continuar con este conjunto de variables.

Siguiendo los resultados obtenidos en los estudios realizados con anterioridad, se han probado varios conjuntos de variables, en diferentes modelos de clasificación multiclase, los que se pueden consultar en los anexos (enlace a GitHub) y se ha llegado a la conclusión de que la mejor combinación de predictores para el modelo elegido es la siguiente (*ilustración 11*), atendiendo a las condiciones que pueden influir en un incendio, consideramos que estas variables tienen sentido.

```
variables = ['altitud', 'racha', 'densPob', 'idcomunidad',
             'superficie_muni', 'prec', 'tmaxCat', 'trimest', 'dist']
```

Ilustración 11. Variables predictoras del modelo

4.2 BALANCEO DE LOS DATOS

La desigualdad en la variable objetivo persiste, esto provocará que el modelo siempre clasifique la predicción a la clase más común, obteniendo así una tasa de precisión ilusoria[9]. Como solución a este problema, se ha recurrido a la técnica de *over-sampling* del paquete *SMOTE* (*ver anexo 6*), dedicado a la clasificación desbalanceada, añadiendo observaciones a las clases minoritarias con el fin de conseguir una variable objetivo con clases balanceadas como se observa a continuación (*ilustración 12*):

```
Distribution of class labels before resampling Counter({2: 36062, 3: 17056, 1: 8273, 4: 2410, 5: 527, 6: 287})
Distribution of class labels after resampling Counter({2: 36062, 1: 36062, 3: 36062, 5: 36062, 4: 36062, 6: 36062})
```

Ilustración 12. Categorías de la variable objetivo antes y después del resampling

4.3 SELECCIÓN DE MODELO

Una vez realizado el *feature engineering* y seleccionadas las variables, se han lanzado una selección de modelos de clasificación, (incluyendo *KNeighborsClassifier*, *RandomForestClassifier*, *GradientBoostingClassifier* y *XGBClassifier*, entre otros); utilizando la técnica de *cross-validation* (*ver anexo 7)*. Esta técnica es utilizada para aumentar la precisión y evitar el sobreajuste, es un método que consigue comparar y seleccionar el mejor modelo predictivo[10].

Los resultados que se obtienen sobre el entorno de entrenamiento balanceado resultan sorprendentemente precisos teniendo en cuenta la calidad de los datos incluidos en ellos, como se ha observado en secciones anteriores. Con lo cual se procede a realizar los algoritmos por separado, sin la técnica de *cross-validation*, descubriendo que, aunque en el entorno de entrenamiento balanceado se obtengan resultados buenos, en el entorno de validación no se supera el 50% de precisión en ningún caso. Esto es debido a la mala calidad de los datos, y por mucho que se aumente el entorno de entrenamiento, reduciendo así el de validación, no mejoran los resultados.

Habiendo expuesto todo esto, hemos concluido que el modelo que mejor puede predecir nuestra variable objetivo es un *XGBoost*, este mejora el rendimiento en comparación con otros algoritmos y minimiza la velocidad de ejecución, importante cuando se trabaja con tantos datos. Además de incluir los hiperparámetros clásicos de *boosting* y del aprendizaje basado en árboles, incluye propios que ayudan a reducir el sobreajuste y aumentar la precisión[11.]

Este es el algoritmo con más precisión en el entorno de validación (*ilustración 13*). Aun así, la máxima puntuación obtenida sigue sin superar el percentil del 50, con lo cual se puede deducir que la predicción es aleatoria y no es una buena estimación para la variable superficie quemada.

```
xgb = XGBClassifier(n_estimators = 300)
xgb.fit(X_train_res, y_train_res)
print(xgb.score(X_train_res, y_train_res))
print(xgb.score(X_test, y_test))
```

```
0.7901761780637051
0.5040856753745202
```

Ilustración 13. Resultado del modelo

Para intentar mejorar el modelo, se han tuneado los hiperparámetros del *XGBoost* mediante la técnica de *grid search*. El tuneado de hiperparámetros mediante *grid search* ayuda a encontrar la mejor combinación de los mismos y, por tanto, a conseguir un rendimiento mayor en el modelo[12]. Debido a la alta capacidad computacional que requiere esta técnica, se han usado únicamente una serie de hiperparámetros reducidos, haciendo referencia sobre todo al número de parámetros y a la profundidad máxima. Por esta razón, el modelo anterior no fue superado, con lo cual esta opción se descartó. El código se puede encontrar en los anexos de GitHub.

Una vez elegido el modelo, se ha procedido, de nuevo, a la técnica de *cross-validation* para comprobar si se diera el caso de *over-fitting*. Con una profundidad de 10, podemos afirmar que es un resultado sin *over-fitting*, dado que los valores son muy parecidos, contando con una media de 0.7078 y una desviación estándar de 0.0591, *ilustración 14*.

```
cross_val_xgb = cross_val_score(xgb, X_train_res, y_train_res, cv=10)
print(cross_val_xgb)
print(cross_val_xgb.mean())
print(cross_val_xgb.std())
```

```
[0.55758388 0.63892227 0.70430281 0.72311319 0.73147849 0.74515876
 0.74631418 0.75001155 0.73623885 0.74511254]
0.707823651586881
0.05914240607468446
```

Ilustración 14. Cross-validation sobre el modelo elegido

5. API DE PREDICCIÓN Y VISUALIZACIÓN DE LOS DATOS

Una vez que hemos obtenido el modelo ya podemos predecir el rango de hectáreas quemadas en uno o varios municipios. AEMET proporciona la predicción meteorológica para los próximos 7 días para todos los municipios categorizados como tal por el INE, por lo que durante todo el proyecto nos hemos limitado a trabajar con esos municipios y unir todos los ficheros con el mencionado código. Para obtener los datos de predicción meteorológica hay que pasarle a la API, uno a uno, el código del municipio que nos interese.

Para ello, una vez más interactuamos con la API REST de AEMET y observamos que a las 25 solicitudes de información el servidor nos cortaba la conexión, por lo que incluimos un time.sleep de 60 segundos cada 25 solicitudes y solventamos este problema.

Como vimos anteriormente, los resultados de estas llamadas a la API devuelven un estándar *http* y los datos en un formato anidado entre listas y diccionarios. Desanidar todos estos datos para conseguir variables legibles fue otra de las grandes dificultades que nos encontramos durante el proyecto.

Finalmente solventamos el problema creando dos data frame por separado. Uno con listas anidadas al que le hicimos *explode*, y otro con diccionarios con el que utilizamos *json_normalize*. Luego unimos ambos resultados y transformamos dicho data frame para poder meterlo en el modelo.

Para cumplir con nuestro segundo objetivo de crear una aplicación y visualizar el resultado decidimos crear un notebook donde te solicita que introduzcas una provincia y automáticamente se le haga un llamado a la API REST de AEMET para descargar la predicción de los municipios de esa provincia, se generen los cambios pertinentes en las columnas así como la creación de nuevas variables que hubiéramos utilizado anteriormente para el modelo, y finalmente obtener un data frame del mismo número de columnas y variables que el data frame utilizado por el modelo.

En este notebook cargamos un *pickle* con el modelo que creamos anteriormente y le introducimos las variables de predicción que hemos transformado previamente, de esta manera obtenemos la variable objetivo superficie quemada para los próximos 7 días en cada uno de los municipios que hayamos indicado. Esta variable se une a los datos de los municipios y se guarda en un fichero *.csv* para que sea leído por Tableau y poder visualizarlo. También se cargan los datos geográficos con los límites municipales que tratamos anteriormente, de este modo podemos ir viendo la predicción para cada uno de los municipios los próximos 7 días.

Se adjuntan tanto el notebook como el archivo de Tableau necesarios para ejecutar la aplicación.

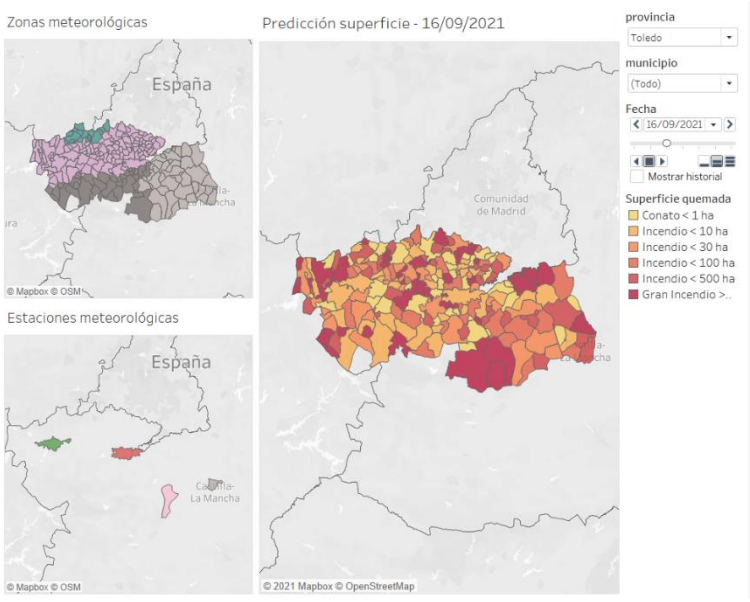

Ilustración 15. Ejemplo visualización con las predicciones

6. CONCLUSIONES

El objetivo principal del trabajo era encontrar un modelo de clasificación multiclase que lograse predecir el tamaño de un incendio empleando para ello variables meteorológicas y geográficas. Una vez observados los resultados obtenidos, podemos concluir que el modelo obtenido no es suficientemente bueno como para lograr el propósito con suficientes garantías.

Entendemos que el problema a resolver guarda poca relación con las variables incluidas en el modelo y que habría que incluir otras características ya demostradas importantes como el tipo de suelo y vegetación, o la pendiente del terreno. Hemos logrado obtener y leer ficheros *shapefile* con las características del terreno, pero no hemos conseguido unir esa información a nuestros datos ya que exigen conocimientos avanzados en tratamiento de mapas y polígonos.

Por otra parte, consideramos que el trabajo creado puede ser una buena base para continuar y crear una herramienta útil para ayudar en la prevención de incendios, y que los organismos y empresas encargadas de la gestión forestal podrían utilizar la herramienta para decidir cómo distribuir sus recursos en función de la predicción.

Sí que estamos satisfechos por haber logrado el segundo objetivo del proyecto y disponer de una aplicación completa desde la solicitud de los datos hasta visualización. Con mejores variables y modelización ya podría ser útil.

A nivel personal, consideramos que la realización de este trabajo ha sido una muy buena oportunidad para poner en práctica todo lo aprendido ya que nos hemos enfrentado a una posible situación real tanto en la consecución de los datos, utilizando diversas fuentes, como durante la ejecución del proyecto. A pesar de no mostrar los resultados deseados, nos sentimos satisfechos por el gran esfuerzo y la gran dedicación mostrada por parte de todo el equipo.

Enlace a github: todos los códigos y ficheros para la implementación de este trabajo se encuentran en el siguiente enlace:

https://github.com/malenarodriguezcarrancio/TFM-Incendios

BIBLIOGRAFÍA

Apuntes de Minería de datos y modelización predictiva. Guillermo Villarino.

Apuntes de Machine Learning con R. Carlos Ortega.

Apuntes de Machine Learning con Python. José Ángel Carballo.

Apuntes de Business Intelligence con Tableau. Carlos Ortega.

Civio Dato. Recuparado el 7 de junio de 2021 en https://datos.civio.es/dataset/todos-los-incendios-forestales/

AEMET OpenData. Recuperado el 7 de junio de 2021 en https://opendata.aemet.es/centrodedescargas/inicio

Centro de Descargas. Organismo Autónomo Centro Nacional de Información Geográfica. Recuperado el 7 de junio de 2021 en http://centrodedescargas.cnig.es/CentroDescargas/catalogo.do?Serie=NGMEN

Hands-On Machine Learning with R. Recuperado el 14 de junio de 2021 en https://bradleyboehmke.github.io/HOML/engineering.html#numeric-feature-engineering

Hands-On Machine Learning with R. Recuperado el 14 de junio de 2021 en https://bradleyboehmke.github.io/HOML/engineering.html#label-encoding

AEMET. Escalas de viento y oleaje. Recuperado el 19 de junio de 2020 en https://www.aemet.es/documentos/es/conocermas/maritima/escalas_de_viento_y_oleaje.pdf

Hands-On Machine Learning with R. Recuperado el 29 de junio de 2021 en https://bradleyboehmke.github.io/HOML/engineering.html#standardization

Programado clic. Selección de funciones en el aprendizaje automático y sus ejemplos de Python. Recuperado el 30 de julio de 2021 en https://programmerclick.com/article/5690455654/

Data Science Team .Validación cruzada K-Fold. Recuperado el 4 de julio de 2021 en https://datascience.eu/es/aprendizaje-automatico/validacion-cruzada-de-k-fold/

Bradley Boehmke y Brandon Greenwell.Hands-On Machine Learning with R. 12.5 XGBoost. Recuperado el 25 de agosto de 2021 en https://bradleyboehmke.github.io/HOML/gbm.html#xgboost

GIANLUCA MALATO. Hyperparameter tuning. Grid search and random search.Your data teacher. Recuperado el 25 agosto de 2021 en https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search/

ANEXO

Enlace a github: todos los códigos y ficheros para la implementación de este trabajo se encuentran en el siguiente enlace:

https://github.com/malenarodriguezcarrancio/TFM-Incendios

```
url = "https://opendata.aemet.es/opendata/api/valores/climatologicos/diarios/datos/fechaini/{}T00%3A00%3A00UTC/fechafin
querystring = {"api_key":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJyYWZvbnNvc3Bpbm9sYUBnbWFpbC5jb20iLCJqdGkiOiJmNDhmMmIyMS00ZTU1
headers = {
    'cache-control': "no-cache"
    }

response = requests.request("GET", url, headers=headers, params=querystring)

print(response.text)
```

```
{
  "descripcion" : "exito",
  "estado" : 200,
  "datos" : "https://opendata.aemet.es/opendata/sh/3a1e5677",
  "metadatos" : "https://opendata.aemet.es/opendata/sh/b3aa9d28"
}
```

```
lista2001 = [
        "https://opendata.aemet.es/opendata/sh/c341925f",
        "https://opendata.aemet.es/opendata/sh/278bcfba",
        "https://opendata.aemet.es/opendata/sh/dcd856f9",
        "https://opendata.aemet.es/opendata/sh/9dfb77ab",
        "https://opendata.aemet.es/opendata/sh/6de945b5",
        "https://opendata.aemet.es/opendata/sh/99aafb39",
        "https://opendata.aemet.es/opendata/sh/89fa44a8",
        "https://opendata.aemet.es/opendata/sh/f408f225",
        "https://opendata.aemet.es/opendata/sh/f8067151",
        "https://opendata.aemet.es/opendata/sh/98102f73",
        "https://opendata.aemet.es/opendata/sh/3a1e5677",
        "https://opendata.aemet.es/opendata/sh/d541919c"
        ]
```

Anexo 1. Llamado a la API para la obtención de datos meteorológicos (01.datosAemet_01_15).

```
def obtenerDatos(x):
    r = requests.get(x)
    j = r.json()
    df = pd.DataFrame(j)
    return df
```

```
df = pd.DataFrame()
for x in lista2001:
    df = pd.concat([df,obtenerDatos(x)], ignore_index = True)
    df2001 = df
df2001.reset_index(drop=True, inplace=True)
df2001.shape
```

```
(70876, 20)
```

Anexo 2. Creación de DataFrames de los resultados obtenidos en ilustración 1 (01.datosAemet_01_15).

Todos los años

```
listaYears = [df2001, df2002, df2003, df2004, df2005, df2006, df2007, df2008, df2009,
              df2010, df2011, df2012, df2013, df2014, df2015]
datosAemet = pd.concat(listaYears)
datosAemet.reset_index(drop=True, inplace=True)
datosAemet.shape
```

```
(1219990, 20)
```

```
datosAemet.to_csv('datosAemet.csv')
datosAemet.to_csv('datosAemetNoIndex.csv', index = False)
```

Anexo 3. Unión de los resultados obtenidos en ilustración 2 (01.datosAemet_01_15).

Definimos la función que calcula los km entre el incendio y la estacion mas cercana.

```
def haversine_vectorize(lon1, lat1, lon2, lat2):
    import numpy as np
    lon1, lat1, lon2, lat2 = map(np.radians, [lon1, lat1, lon2, lat2])

    newlon = lon2 - lon1
    newlat = lat2 - lat1

    haver_formula = np.sin(newlat/2.0)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(newlon/2.0)**2

    dist = 2 * np.arcsin(np.sqrt(haver_formula ))
    km = 6367 * dist #6367 for distance in KM for miles use 3958
    return km
```

Definimos la funcion distancias, que devolvera los km y el indicativo de la estacion:

```
def buscarDistancia5(latitud,longitud):
    conseguirDistancias = pd.read_csv('estacionesLatLong.csv')
    dataIncendio = [{'lat': latitud, 'lon': longitud}]
    incendio = pd.DataFrame(dataIncendio)
    incendioNuevo = pd.concat([incendio]*conseguirDistancias.shape[0])
    incendioNuevo.reset_index(drop=True, inplace=True)
    obtengoDistancias = pd.concat([conseguirDistancias, incendioNuevo], axis=1, ignore_index=True)
    obtengoDistancias.columns = ['provincia','altitud','indicativo','nombre','indsinop',
                                 'latitud','longitud','latIncendio','lonIncendio']
    for x in obtengoDistancias:
        obtengoDistancias['distancia'] = haversine_vectorize(obtengoDistancias['latitud'],
                                                             obtengoDistancias['longitud'],
                                                             obtengoDistancias['latIncendio'],
                                                             obtengoDistancias['lonIncendio'])
    resultado = obtengoDistancias.iloc[obtengoDistancias['distancia'].idxmin(),:]

    return resultado.indicativo, resultado.distancia
```

Aplicamos la funcion distancia:

```
lista = []
agregar = []
for i in incendiosLatLon.index:
    lista.append([incendiosLatLon["latitude"][i], incendiosLatLon["longitude"][i]])
    agregar.append(list(buscarDistancia5(float(lista[i][0]),float(lista[i][1]))))
```

Anexo 4. Definición y aplicación de la función distancia (03. UNION BBDD)

```
# Normalización:
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(datosML)
df_normalized = pd.DataFrame(np_scaled)
df_normalized.columns = datosML.columns.values
```

```
df_normalized.head()
```

	id	latitude	longitude	idcomunidad	idprovincia	COD_AEMET	zona	altitud	poblacion_muni	superficie_muni	...	fe_month	idmunicipio	dist	densPob	trimest	beauVelmed	beauRacha	tmaxCat	tminCat	supQuemada
0	0.137933	0.943081	0.699894	0.882353	0.0	0.00000	0.0	0.215703	0.000878	0.011078	...	0.727273	0.000000	0.877416	0.008649	0.666667	0.333333	0.333333	0.666667	0.75	3
1	0.000002	0.953245	0.680593	0.882353	0.0	0.00002	0.0	0.086957	0.003076	0.054829	...	0.636364	0.000176	0.892472	0.006247	0.666667	0.333333	0.333333	0.833333	0.75	3
2	0.206899	0.952685	0.677827	0.882353	0.0	0.00002	0.0	0.086957	0.003076	0.054829	...	0.727273	0.000176	0.892095	0.006247	0.666667	0.333333	0.333333	0.666667	0.75	2
3	0.068966	0.953245	0.680593	0.882353	0.0	0.00002	0.0	0.086957	0.003076	0.054829	...	0.000000	0.000176	0.892472	0.006247	0.000000	0.333333	0.333333	0.500000	0.50	2
4	0.000002	0.953245	0.680593	0.882353	0.0	0.00002	0.0	0.086957	0.003076	0.054829	...	0.636364	0.000176	0.892472	0.006247	0.666667	0.333333	0.333333	0.666667	0.75	2

Anexo 5. Función y resultado de la normalización

```
sm = SMOTE(random_state=2)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train)
print ("Distribution of class labels before resampling {}".format(Counter(y_train)))
print ("Distribution of class labels after resampling {}".format(Counter(y_train_res)))
```

```
Distribution of class labels before resampling Counter({2: 36062, 3: 17056, 1: 8273, 4: 2410, 5: 527, 6: 287})
Distribution of class labels after resampling Counter({2: 36062, 1: 36062, 3: 36062, 5: 36062, 4: 36062, 6: 36062})
```

Anexo 6. Función y resultado del balanceo

```
use_label_encoder=False
models = {
    "neighbors": KNeighborsClassifier(),
    "histboosting": HistGradientBoostingClassifier(),
    "histboosting0.2": HistGradientBoostingClassifier(learning_rate=0.25),
    "forest300,50": RandomForestClassifier(n_estimators=300, max_depth=10)
    "Gradientboosting" : GradientBoostingClassifier(),
    "XGBoost": XGBClassifier(n_estimators = 100)
}
for name, model in models.items():
    scores = cross_val_score(model, X_train_res, y_train_res, cv=10)
    print(f"{name} accuracy => {np.mean(scores)}")
    print(scores.mean())
    print(scores.std())
```

```
neighbors accuracy => 0.7351322946388008
0.7351322946388008
0.015157316978774024
histboosting accuracy => 0.6014832455776769
0.6014832455776769
0.03530683409834005
histboosting0.2 accuracy => 0.6637557434307847
0.6637557434307847
0.04236191580995601
forest300,50 accuracy => 0.5407818261368542
0.5407818261368542
0.012642421994648203
Gradientboosting accuracy => 0.4268992865556049
0.4268992865556049
0.014987314445927663
XGBoost accuracy => 0.6390298768080507
0.6390298768080507
0.04630573917910128
```

Anexo 7. Función y resultado de la combinación de modelos usando cross-validation

```
codigoValido = False

while codigoValido == False:
    try:
        buscarProvincia = input("Introduce una provincia: ")
        lista_prov = df_munic_merge['provincia'].tolist()
        if buscarProvincia in lista_prov:
            print("La provincia es correcta. Pedimos la predicción a AEMET.")
            codigoValido = True
            provincia = df_munic_merge[df_munic_merge['provincia'] == buscarProvincia]
            municipios = provincia['COD_AEMET'].tolist()
            predic = prediccion(municipios)
            for key,value in dict_Mapa.items():
                if buscarProvincia == key:
                    repeticiones = value
            print(predic)
        else:
            print("Disculpa, ese código no pertenece a ninguna provincia, prueba con otro.")
    except ValueError:
        print("Disculpa, ese código no pertenece a ninguna provincia, prueba con otro.")
        continue
```

```
Introduce una provincia: Segovia
La provincia es correcta. Pedimos la predicción a AEMET.
Ya se han descargado 24 municipios.
Ya se han descargado 48 municipios.
Ya se han descargado 72 municipios.
Ya se han descargado 96 municipios.
Ya se han descargado 120 municipios.
Ya se han descargado 144 municipios.
Ya se han descargado 168 municipios.
Ya se han descargado 192 municipios.
```

Anexo 8. Descarga de datos de predicción de la API REST de la AEMET para una provincia en específico

Y esta es su predicción de incendios:

	fecha	Tipo de Incendio	COD_AEMET	Municipio
0	2021-09-15	Incendio < 10 ha	40001	Abades
1	2021-09-16	Incendio < 10 ha	40001	Abades
2	2021-09-17	Incendio < 10 ha	40001	Abades
3	2021-09-18	Incendio < 10 ha	40001	Abades
4	2021-09-19	Incendio < 10 ha	40001	Abades
...	...	...	...	...
1458	2021-09-17	Incendio < 10 ha	40906	San Cristóbal de Segovia
1459	2021-09-18	Incendio < 10 ha	40906	San Cristóbal de Segovia
1460	2021-09-19	Incendio < 10 ha	40906	San Cristóbal de Segovia
1461	2021-09-20	Incendio < 10 ha	40906	San Cristóbal de Segovia
1462	2021-09-21	Incendio < 10 ha	40906	San Cristóbal de Segovia

1463 rows × 4 columns

Anexo 9: Predicción obtenida de los datos del anexo 8