Reto Minsait Land Classification

Equipo: Data Attack
Miembros: Vanessa Navarro
Centro: UCLM

1. Trabajo desarrollado

Para la realización del Datathon se ha utilizado la plataforma *Kaggle Notebooks* (ofrece 4 cores de CPU y 16GB de memoria RAM) y el lenguaje de programación Python 3.7. En un primer lugar se eligió *Jupyter Notebooks* como herramienta de desarrollo, pero se decidió cambiar debido a la baja potencia de nuestros equipos y al alto rendimiento requerido por algunos de los algoritmos de Machine Learning que se han testeado en este trabajo (concretamente el estimador *VotingClassifier*).

Lo primero que se hace en el cuaderno es almacenar los datos de *Modelado* en un dataframe de *Pandas*. Se hace lo mismo con el fichero de datos de *Estimación,* que se guardan en un dataframe (*test*). Este dataframe no lo tocaremos para nada, solo al final del proceso de entrenamiento de cada estimador, para predecir la clase del test y poder así enviar los resultados obtenidos.

Sobre el dataframe de modelado, se realiza un *holdout* del 25% de los datos para generar un pequeño conjunto de validación (*val*) para realizar predicciones y obtener los resultados. El resto de los datos (85%) conformarán el dataset de entrenamiento (*train*).

Para la realización del cuaderno se han seguido estos 5 pasos:

- Exploración de los datos y análisis de las variables y de la clase.
- Preprocesamiento de los datos.
- Selección de variables.
- Prueba de distintos estimadores.
- Elección del modelo final y resultados.

2. Análisis exploratorio

En primer lugar se realiza un análisis exploratorio de la variable clase y su distribución dentro del dataset de entrenamiento, donde se observa un claro desbalanceo de la clase *RESIDENTIAL* en relación con el resto de tipos de terreno (en total 7 tipos). Esto puede llegar a ser un problema, ya que según nos informaron, el conjunto de test no estaría tan desbalanceado como el de train.

Además de esto, también se observó la presencia de valores perdidos en el dataset. En concreto, había 20 filas con valores perdidos, los cuales se concentraban en las variables *MAXBUILDINGFLOOR* y *CADASTRALQUALITYID*, por lo que se decidió eliminar los 20 casos del dataset de entrenamiento.

Después se procede a analizar las variables del dataset, dividiéndolas en 2 tipos: numéricas (todas menos *CADASTRALQUALITYID*) y categóricas (*CADASTRALQUALITYID*). También se genera una matriz de correlación para ver posibles relaciones entre variables (ver conclusiones obtenidas en el cuaderno).

El último paso en la exploración fue visualizar con diferentes gráficas de tipo *kdeplot* la distribución de la clase para cada una de las variables del dataset. Las conclusiones que se obtuvieron fueron estas:

- Las variables de los canales indican mayoritariamente que, si su valor se acerca aproximadamente a la media de dicha variable, la clase será *RESIDENTIAL*, que es la clase mayoritaria del dataset, por lo que esto no nos dice mucho.
- Sobre las variables *X e Y*, podemos observar que si sus valores indican más o menos el centro de la zona del municipio de Madrid, el tipo de terreno será *OFFICE*, mientras que a las afueras tendremos terrenos de tipo *INDUSTRIAL* o *AGRICULTURE*.

- Respecto a la variable *AREA*, podríamos decir que si esta es muy grande, el tipo de terreno será casi con total seguridad, *RETAIL*, posiblemente debido a los centros comerciales de gran superficie.
- Sobre las variables *GEOM_R2* y *GEOM_R3*, si sus valores son muy bajos, se tratará de terreno de tipo *PUBLIC* o *OFFICE*, mientras que si es muy elevado, se tratará de terrenos de tipo *OTHER*.
- El año de construcción de los edificios del entorno (*CONTRUCTIONYEAR*) nos dice que, si su valor está entre los años 60 y 70, puede que se trate de terrenos de tipo *OTHER* o *RETAIL*. Los edificios más viejos están en terrenos de tipo *RETAIL* sobretodo.
- El número máximo de plantas de los edificios del entorno (*MAXBUILDINGFLOOR*) indica claramente que si su valor es cero, el terreno será *AGRICULTURE*, mientras que si su valor es muy elevado, se tratará de terrenos de tipo *RETAIL* o *PUBLIC*.
- Sobre *CADASTRALQUALITYID* se puede decir que las viviendas de mayor calidad serán sobretodo de tipo *PUBLIC* y *OFFICE*, mientras que las de menor calidad serán de tipo *INDUSTRIAL* en su mayoría.

3. Manipulación de variables

Para el preprocesamiento de las variables numéricas se ha utilizado un *Pipeline* que consta de 2 pasos:

- *SimpleImputer* para imputar los posibles valores perdidos entrantes con la media.
- *StandardScaler* para normalizar los valores de las variables a media 0 y desviación 1.

Para las variables categóricas se utilizó un Pipeline con un primer paso idéntico al del Pipeline anterior, y un segundo paso con *OneHotEncoder* para binarizar los diferentes valores de la variable categórica.

Una vez hecho esto, se agrupan estos 2 pipelines en un *ColumnTransformer* para procesar todas las variables, y finalmente obtener un conjunto de datos de train válido para el entrenamiento de los algoritmos de *Scikit-Learn* elegidos.

Cuando se terminó de entrenar y obtener las predicciones de todos los modelos, se decidió intentar una selección de un subconjunto de variables, para comprobar si una simplificación de los datos ayudaría a la mejora de los modelos. Para ello se utilizó *SelectKBest*, que devolvió un subconjunto de k=10 mejores variables (ver cuaderno para más info).

4. Selección del modelo

Se probaron diferentes algoritmos de la librería Scikit-Learn, utilizando de nuevo *Pipeline* para el preprocesamiento de los datos:

- *DecisionTreeClassifier* con *GridSearchCV* para encontrar la mejor combinación de sus hiperparámetros.
- *RandomForest*. Este modelo obtuvo buenos resultados, pero creemos que se trata de sobreajuste a los datos de train.
- *KNeighborsClassifier*. También dio buenos resultados tanto en train como en val.
- Regresión logística multinomial.
- *VotingClassifier*.

Cada uno de ellos fue probado tomando el conjunto entero de variables (54 más la clase) y también utilizando el subconjunto de las 10 mejores variables obtenido con *SelectKBest*. Las métricas utilizadas para medir el rendimiento de los modelos probados fueron *precision* (*weighted*) y *accuracy*.

El modelo que mejor resultados obtuvo fue *VotingClassifier* con el subconjunto de k=10 variables. Los resultados obtenidos con este estimador fueron:

- **Accuracy**: 0.9607 (train), 0.9461 (validación).
- **Precision**: 0.9627 (train), 0.9474 (validación).