**INFORME PARA LA SIGUIENTE REUNIÓN**

1. Se ha probado a transformar la variable CONSTRUCTIONYEAR a una ANTIQUITY, de año de construcción a antigüedad “ANTIQUITY = 2020 – CONSTRUCTIONYEAR”. Las métricas de *accuracy* y *recall* son las mismas para ambos modelos (XGBoost y Random Forest) que las de la entrega de la Fase I.
2. En la variable MAXBUILDINGFLOOR, cuando era un valor nulo, se transformaba a -1 y ahora a 0. Las métricas de *accuracy* y *recall* son las mismas para ambos modelos (XGBoost y Random Forest) que las de la entrega de la Fase I.

CONCLUSIÓN: si estos cambios mejoran o empeoran la precisión muy ligeramente, es prácticamente imposible detectarlo.

1. Se ha probado a transformar la variable CADASTRALQUALITY de tipo categórica a one-hot. Para ello se ha eliminado esta y añadido las 13 one-hot correspondientes. Se ha generado un fichero respuesta con 100 iteraciones con este cambio y los dos anteriores. Dicho fichero es muy similar al de la entrega de la fase I, aunque contiene menos muestras predichas a la clase RESIDENTIAL.

CONCLUSIÓN: Posiblemente estos cambios mejores muy ligeramente la precisión.

1. Se ha entrenado un modelo Support Vector Machine sin realizar PCA a los datos, pero con los cambios anteriores realizados. La precisión se reduce al 23% y el *recall* al 14%.
2. Se están haciendo pruebas con MLBox (técnica de auto machine learning) tanto en un ordenador personal como en Google Colab pero de momento no muestra nada (parece estar bloqueado)
3. Se ha probado el modelo Extra Trees (con entre 200 – 400 estimadores), da resultados ligeramente peores que el Random Forest (tanto en *accuracy* como en *recall*)
4. Se ha realizado Grid Search para Random Forest y XGBoost pero nos encontramos con un problema. Estos algoritmos esperan un conjunto de datos equilibrados (cosa que realizamos mediante *under sampling* de RESIDENTIAL) y tras lanzar varias ejecuciones, los hiperparametros que nos devuelven son diferentes y distan entre sí. Esto es debido a que como las muestras añadidas de RESIDENTIAL son diferentes cada vez la búsqueda que hace cambia. Nos hemos quedado con una estimación de los hiperparametros que más salían ganadores.

CONCLUSIÓN: n\_estimators = 750, criterion = ‘gini’,

1. Se ha entrenado a un modelo Random Forest con los datos tras aplicarles SMOTE, obteniendo una precisión del 97%. Claramente, overfitting.
2. Se ha añadido un archivo visualizer.py para visualizar PCA, t-SNE, histogramas y pie chart. Por otro lado, el archivo sampling.py implementa todas las técnicas de over y under sampling, además de las combinaciones SMOTE + ENN y SMOTE + Tomek Links.
3. Paper OAA-DB: pensamos que puede ser prometedor. Lo dividiremos en las siguientes fases:
   1. Obtener los datasets:
      1. X, y originales
      2. X\_train\_random, y\_train\_random y los tests random, utilizando en principio un 20% de las muestras para test.
      3. X\_train\_notrandom, y\_train\_notrandom y los tests NO random. Nos será más representativo, pues intentamos que la clase residential sea mucho mayor y así podemos observar cómo va prediciendo el resto de clases minoritarias.
   2. Primer clasificador XGBClassifier.
   3. Primer OneVS(…)
      1. Rest
      2. One (según el paper, más favorable para datasets grandes)
   4. Balanceo de datos con técnicas:
      1. Oversampling + undersampling:
         1. SMOTE + ENN
         2. SMOTE + Tomek Links
      2. SMOTE + CMTNN → En el paper habla de dos redes neuronales complementarias feedforward con backpropagation. Una TruthNN y otra FalsityNN. No sabemos bien cómo o con qué librerías podríamos realizar lo que en el paper se muestra.
   5. OneVS(…) con el nuevo dataset balanceado.
   6. Resultado final obtenido.