GEO-AI Challenge for Landslide Susceptibility Mapping report

- Marcos Tidball (@zysymu), University of Padua, Italy

Introduction

Landslides are a very common occurrence in the world, especially in the Italian Alps due to the region's steep slopes, heavy rainfall, and geological instability. They pose a significant hazard, causing damage to infrastructure, property and human life.

In this competition organized by ITU, participants are tasked with creating a landslide susceptibility model that is able to map areas that are potentially unstable and prone to landslides. Such mapping is crucial in making informed decisions and implementing preventive measures such as monitoring, early warning, evacuation, stabilisation, and restoration.

Data

Datasets

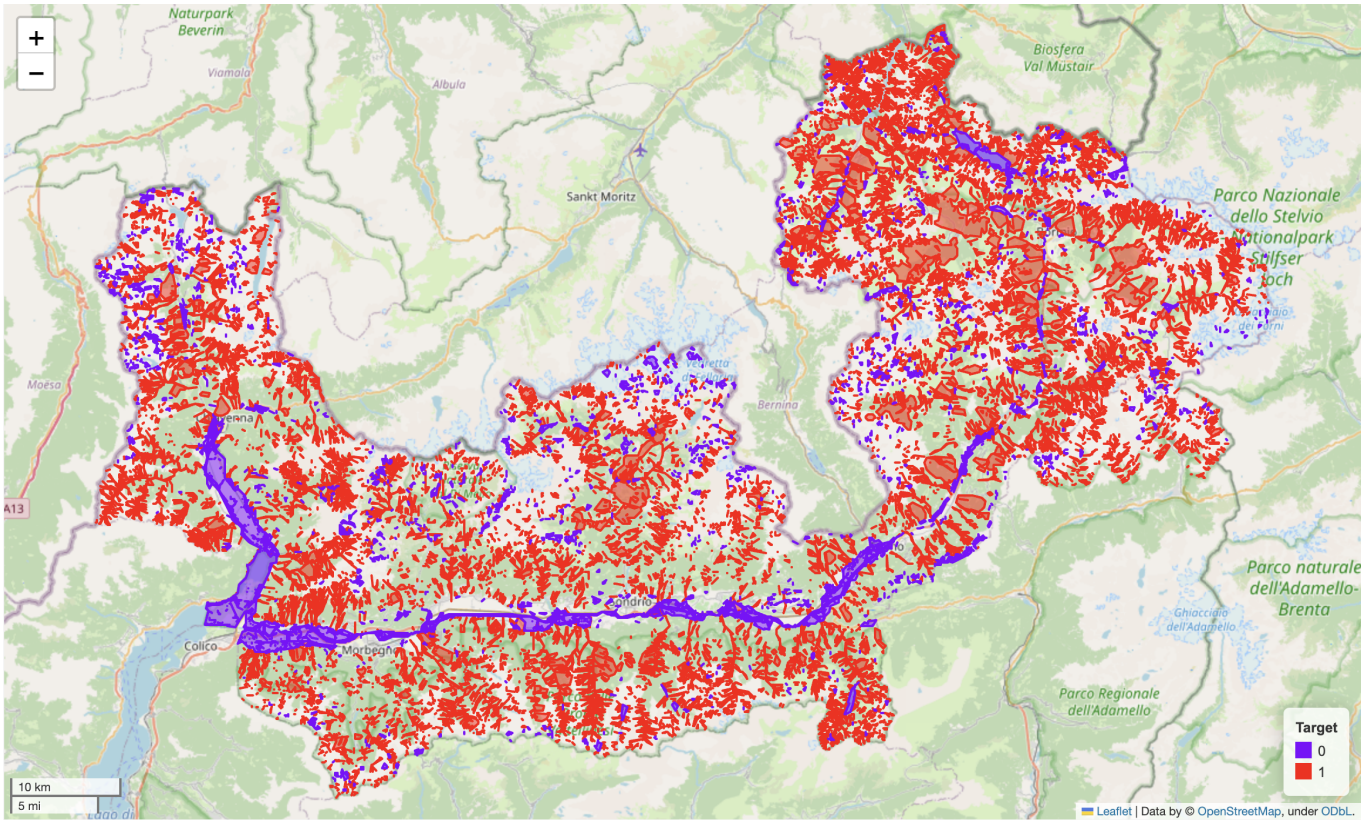
We used only the official datasets provided by the competition hosts; we used both the geopackage datasets:

- Road network at a 1:10,000 scale.
- River network at a 1:10,000 scale.
- Geological fault zones map at a 1:10,000 scale.
- Land use/land cover map at a 1:10,000 scale.

And the raster datasets:

- Digital Terrain Model (DTM) at a 5 m/pixel scale.
- Interpolated yearly averaged hour precipitation for the year of 2020.
- 90th percentile of the hour precipitation for the year of 2020.

We also used only the training dataset provided by the competition, which is landslide inventory comprising the boundaries of translational, rotational shallow landslides, and debris flows at a 1:10,000 scale in vector format, marked as a value of 1. The zones marked with a value of 0 are areas with low probability of shallow landslide:



Data processing

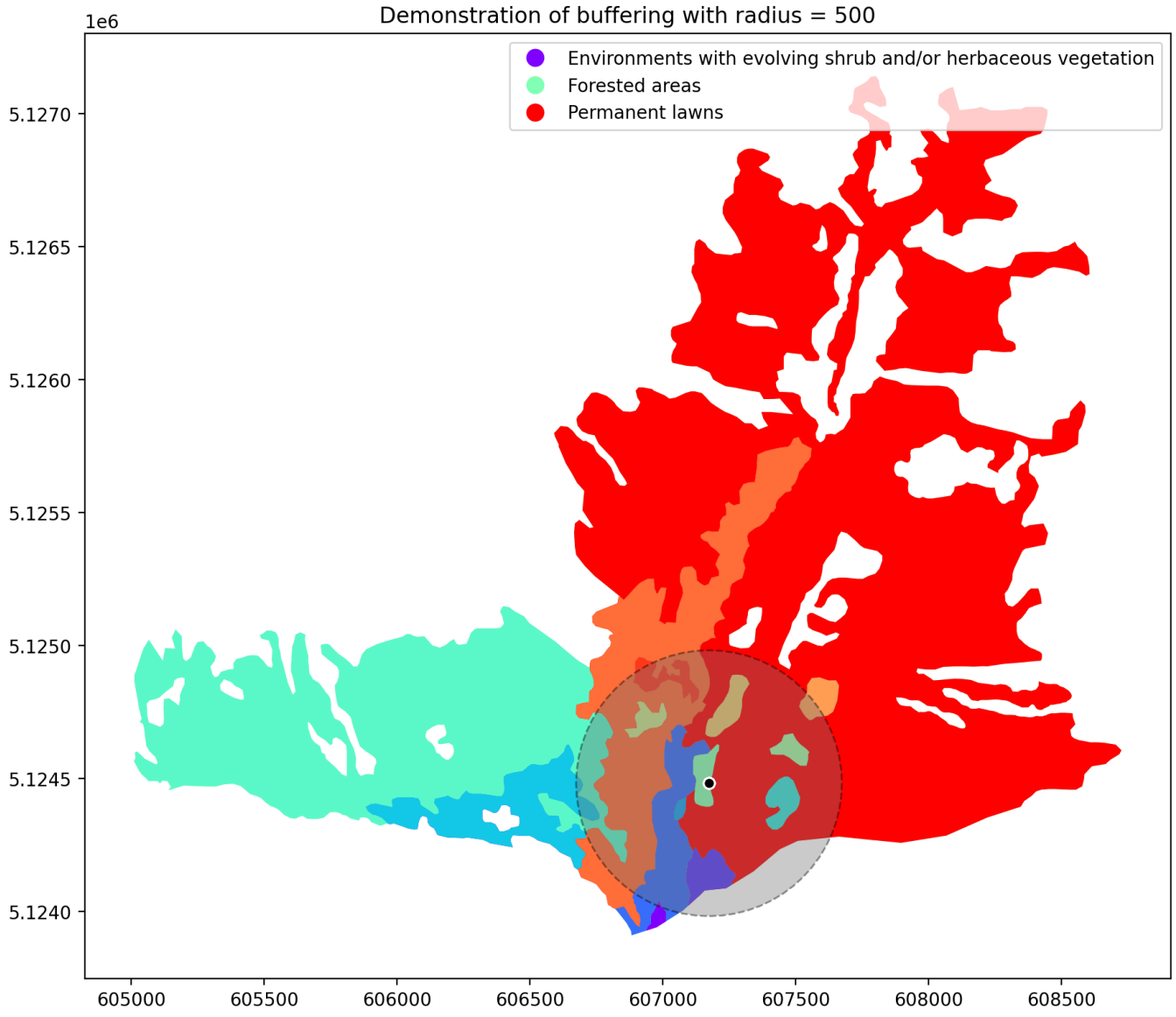
Since the test data has a `POINT` format, the first step in our data processing pipeline was to convert the `MULTIPOLYGON` geometry of the training dataset into `POINT` geometries by considering only the centroids of the `MULTIPOLYGON`s. This approach is very simple but already outputs promising results. If necessary, it would be possible to split the larger areas into multiple points in order to have a larger collection of examples in our training dataset.

The next step was to clean the geopackage data, since a lot of the datasets have irrelevant or repeating columns. Refer to the Jupyter notebook for a thorough explanation of the process.

Feature engineering

The features used for the model revolve around the idea of buffering. A big part of the training data is made from relatively small geometries, and given that we want to model landslide susceptibility, the information present directly within the geometry that we're considering might not be enough. For instance, the fact that there's a river 50 m away from the geometry we're analyzing might make it more susceptible to a landslide, even though the river isn't crossing directly thorough the geometry. Thus, it'd be a good idea to consider also what is near our geometry.

We can use *buffering* as a way to enlarge our geometries by creating a buffer zone around them. We define buffer zones of circles of the following radii: 100 m, 500 m, 1 km, 2.5 km. Then, we gather information and relevant characteristics present in each of these areas in order to be able to extract what is around our geometry at different distances.



Here, the black geometry with a white outline is the original geometry and the greyed area with a dashed line is the buffered geometry. The other parts of the plot are the geometries that intersect with the buffered geometry, with different colors symbolizing different geometry instances.

Note that in this single example we were already able to get a much better understanding of the area surrounding the geometry. We are able to access a wider area containing a more varied geography that most certainly affects what happens on the original geometry we're looking at. This demonstration uses the land use dataset, but buffering is also very useful for the detection of nearby roads, rivers and geographic faults.

The difference is that in order to calculate our features we'll be calculating the overlays instead of just doing spatial joins. If we look at our example, this means that instead of getting the full geometries of the areas that intersect with our point, we only get the areas of these geometries that are inside the greyed area. This will be useful for us later on in order to calculate features related to density and also in order to get a more refined measure of what is actually part of the buffered area. In this example, for instance, the inclusion of the entire cyan geometry on the left wouldn't make much sense given the fact that only a fraction of it is intersecting our buffer area; instead, we include the information of this

geometry but only considering the smaller area in the intersection. We also implement a function that allows us to calculate the distance between the points in our training dataset and the geometries from the geopackage datasets that intersect with the buffer areas.

This process leaves us with a very large dataframe composed of all the geopackage characteristics found around different radii of our data. With this, we calculate features for combinations of different geopackage characteristics:

- Density of categorical feature by numerical feature, e.g. density of fault of type "faglia" over intersection length.
- Proportion of sum of numerical feature in relation to sum of other numerical feature. e.g. intersection area / total buffer area.
- Proportion of one categorical feature in relation to total number of categories of this feature, e.g. number of faults of type "faglia" / total number of faults.
- Statistics of distances between data point and geopackage characteristics, i.e. mean, std, min, max and median.

We calculate these features for a combination of different characteristics in the intersection between our buffer areas and the different geopackage datasets. A more detailed explanation of all of the calculated features is available on the Jupyter notebook. This leaves us with a dataframe with 1552 geopackage-based features. The number is very large and could be easily reduced with a dimensionality reduction algorithm, by exploring feature importance in the final model or by an expert (which would also be able to formulate more informative features).

For the raster data features, we do a very similar process, also buffering over the same radii defined before. However, due to the nature of the data, we use zonal statistics for each buffer area: mean, std, min, max, sum, median, range, percentile_10, percentile_25, percentile_75 and percentile_90. These are calculated for each of the raster datasets we use. However we also create custom features based exclusively on the DTM raster. We calculate the slope of the terrain, the aspect of the slope, the terrain profile curvature (curvature of a land surface in the direction of the steepest slope; it quantifies how the slope changes along the slope direction) and the terrain plan curvature (curvature of a land surface perpendicular to the direction of the steepest slope; it quantifies how the slope changes in a direction orthogonal to the slope). More details are available on the Jupyter notebook.

This leaves us with a total of 1732 features. While the number is very large, due to the way the geopackage features are calculated, most of them are very sparse. The approach is based on creating a lot of features and then later removing unnecessary ones to speed up computation times, though during the competition all of the features were used and this analysis has not been made.

Model development

For the model we use LightGBM due to its performance, speed and probability-based predictions. In order to achieve the best performance we use Optuna to perform hyperparameter search. Also, for a more robust evaluation, we employ a 5-fold cross-validation for training (using 80% of data for training

and 20% of validation). We optimized based on the binary-logloss function, and this is the parameter grid used for optimization:

```
param_grid = {
    # fixed params
    "n_estimators": trial.suggest_categorical("n_estimators", [10000]),
    "objective": trial.suggest_categorical("objective", ["binary"]),
    "verbose": trial.suggest_categorical("verbose", [-1]),
    "random_state": trial.suggest_categorical("random_state", [42]),
    "metric": trial.suggest_categorical("metric", ["binary_logloss"]),

    # searchable params
    "is_unbalance": trial.suggest_categorical("is_unbalance", ["false", "true"]), # automatically balance class weights
    "learning_rate": trial.suggest_float("learning_rate", 0.01, 0.3),
    "num_leaves": trial.suggest_int("num_leaves", 20, 3000, step=20),
    "max_depth": trial.suggest_int("max_depth", 3, 100),
    "min_data_in_leaf": trial.suggest_int("min_data_in_leaf", 20, 1000, step=100),
    "lambda_l1": trial.suggest_int("lambda_l1", 0, 100, step=5),
    "lambda_l2": trial.suggest_int("lambda_l2", 0, 100, step=5),
    "min_gain_to_split": trial.suggest_float("min_gain_to_split", 0, 15),
    "bagging_fraction": trial.suggest_float("bagging_fraction", 0.2, 1.0, step=0.1),
    "bagging_freq": trial.suggest_int("bagging_freq", 0, 5, step=1),
    "feature_fraction": trial.suggest_float("feature_fraction", 0.2, 1.0, step=0.1),
}
```

Results

After 100 optimization trials we achieve an accuracy of 96.82% on the 5-fold cross validation, with the following parameters:

```
n_estimators: 10000
objective: binary
verbose: -1
random_state: 42
metric: binary_logloss
is_unbalance: false
learning_rate: 0.29173260941877943
num_leaves: 1040
max_depth: 20
min_data_in_leaf: 20
lambda_l1: 0
lambda_l2: 35
min_gain_to_split: 0.8067742150297849
bagging_fraction: 0.30000000000000004
bagging_freq: 0
feature_fraction: 1.0
```

After training the model with the best parameters on the entire training dataset, we achieve an accuracy of 94.2% on the public competition leaderboard. While for the competition we used our model for binary classification, LightGBM outputs probabilistic results, which allow us to easily tune it for either recall or precision and also to find areas that could have a higher chance of landslide occurrence, even if currently predicted as non-landslide areas.

Conclusion

In this report we explored the techniques and methods used for the GEO-AI Challenge for Landslide Susceptibility Mapping competition by ITU. Landslides are a global hazard that has been happening more frequently with time. While this competition focused on the North of Italy, particularly the Valtellina Valley, a region marked by steep slopes, heavy rainfall, and geological instability, the techniques and model developed in this solution could easily be applied to other areas in Italy and in the world, given there is data available for it.

We believe that one of the most valuable insights of this solution in particular is that it is possible to create very good probabilistic models by just using simple statistics of features found around different areas. This shows that machine learning models are very robust and, coupled with experts in the area, could predict landslides with high accuracy. However, in order to have more robust and general models that can be used in different areas, weathers and geographies, it is of upmost importance to have quality data available! The exploration of models focused on imbalanced data could also prove useful when trying to implement this approach for other locations where landslides are rare.