Classification of Landslide Triggers with Random Forest Classifier and Support Vector Machines

Kayla Bachler[1], Dong Si[2]
[1]Computing and Software Systems
University of Washington Bothell
Bothell, WA. USA
bachlerk@uw.edu, dongsi@uw.edu

Abstract. There has been a significant amount of research previously done for forecasting potential landslide occurrences based on the soil and slope of terrain, and even recent research with the use of machine learning to predict landslide susceptibility [7]. However, there has yet to be research regarding the prediction of landslide triggers. Our research focuses on extending the existing research by determining the primary triggers for landslides in various locations around the globe. We believe that by considering the initial cause of a landslide we can extend research to better predict the likelihood of a landslide occurrence. Using a dataset of landslide incidences and their associated triggers this work summarizes two methods of implementation, Random Forest Classifier (RFC) and Support Vector Machines (SVM). We found through these methods that the proposed dataset in imbalanced and lacks features causing poor model performance. With most rainfall-triggered landslides and a shortage of data for non-rainfall triggers we implemented ADASYN on our RFC model and SMOTE with our SVM model to handle data imbalance. Our results for this multi-class classification problem show that with oversampling we were able to improve our model performance, but due to lack of data for minority classes our overall performance for both models was subpar.

Keywords: Landslide Trigger, Machine Learning, Random forest, Support vector machine, Classification, SMOTE, ADASYN, Class imbalance.

1 Introduction

Landslides are a dangerous natural occurrence that happen often in areas where there are high natural slopes, loose terrain and rainfall. Scientific research on landslide susceptibility is nothing new and machine learning has been used extensively to predict landslide occurrences based on the location and type of terrain [1]. However, this current research could be extended to further consider classification of landslide triggers in order to better predict the chances of a landslide occurring. Our research aims to focus on the primary cause of a landslide, or its trigger, in order to determine the likelihood of a landslide occurring again or in the future.

The dataset used for this project is the Global Landslide Catalog (GLC) export published by NASA [2]. This dataset provides several landslide incidences each with a corresponding trigger. There are 22 features in the dataset, 11 of which we chose to use for this project (see Table 1). The landslide_trigger feature is the class label. After preprocessing this set includes 9133 landslide triggers with 7966 being rain, 562 tropical cyclone, 133 snow, 129 monsoon, 93 mining, 89 earthquake, 86 construction, and 74 flooding.

Table 1. Global Landslide Catalog data features representation.

Feature	Description
event_month	The calendar month of the landslide incident
event_time	Time at which the landslide event took place
landslide_category	The type of landslide movement – slide (rock slide, debris slide, earth slide), creep, debris_flow, earth_flow, snow_avalanche, lahar, rock_fall, earth_fall, complex (combination of two or more of the categories)
landslide_size	The general size of the landslide (small, medium, large, and very_large)
fatality_count	The number of fatalities due to the landslide
injury_count	The number of injuries that took place due to the landslide
country_name	Name of the country where the landslide occurred
population	The population count of the location at which the landslide took place
longitude	Exact longitude of the landslide location
latitude	Exact latitude of the landslide location
landslide_trigger	The trigger that caused the landslide – rain, construction, earthquake, flooding, freeze-thaw, mining, monsoon, snow, and tropical cyclone

2 Methods

Since this research is essentially a new extension on the existing research for the prediction of landslide susceptibility, there were no previous works to reference while implementing our problem. We followed the workflow process seen in figure 1 in order to implement our idea. We knew that our problem involved classifying landslide triggers using supervised learning. Since we had a multi-class problem with 8 landslide triggers, we chose to use classification methods Random Forest Classifier (RFC) and Support Vector Machines (SVM). Our reasoning for selecting these classifier methods was due to their ability to train multiple classes while handling the issue of class imbalance. With the problem established and the methods of choice selected, our next step was to begin pre-processing our dataset.

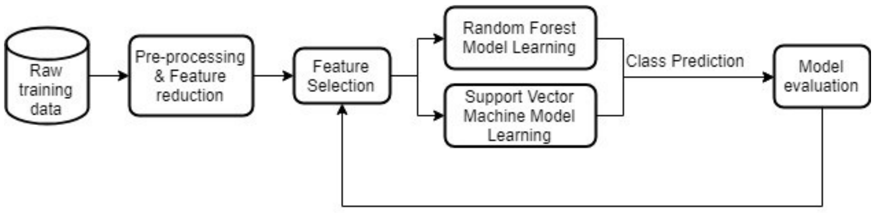

Raw training data
Pre-processing & Feature reduction
Feature Selection
Random Forest Model Learning
Support Vector Machine Model Learning
Class Prediction
Model evaluation

Fig. 1 Workflow of the proposed system

2.1 Data Preprocessing

As mentioned above, we used the Global Landslide Catalog (GLC) dataset in order to train our model. Prior to preprocessing, the dataset consisted of just over 11,000 landslide occurrences from 1988 to 2017. When NASA released the dataset online they stated that it “was developed with the goal of identifying rainfall-triggered landslide events around the world, regardless of size, impacts or location”[2]. Because of this a large portion of our data was related to rainfall triggers, thus causing our data imbalance. The original dataset consisted of 17 landslide trigger labels, some with less than five occurrences. After completing data preprocessing, we narrowed our class labels down to a total of 8 unique landslide triggers which are:

- Rain
- Construction
- Earthquake
- Flooding
- Mining
- Monsoon
- Snow
- Tropical Cyclone

Our dataset contained multiple class labels associated with rainfall, so we decided to combine them all into one majority class, rain. Our reasoning for this was primarily because we wanted individual class labels to be unique from each other and felt that having more than one rain related class would create noise within our dataset. When pre-processing was complete we found that our data was imbalanced, with 'rain' as our majority class and all other classes as minorities. Figure 2 shows the data distribution for each class label after preprocessing was completed.

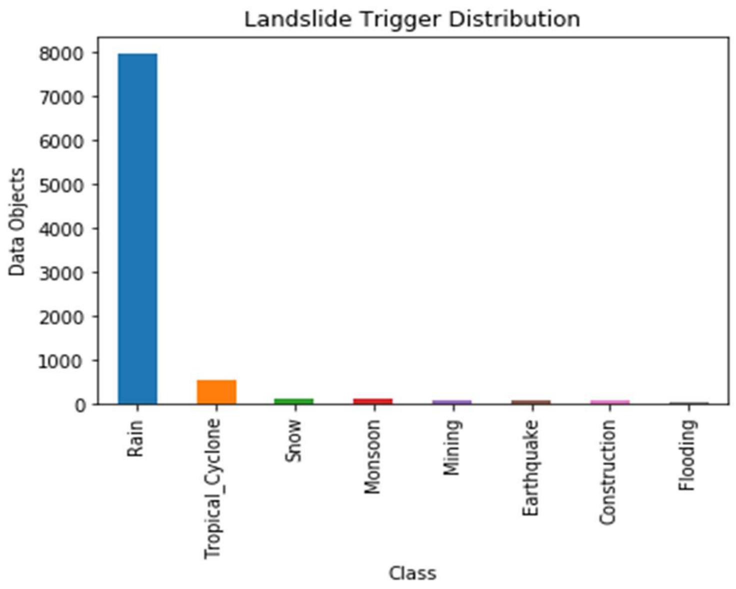


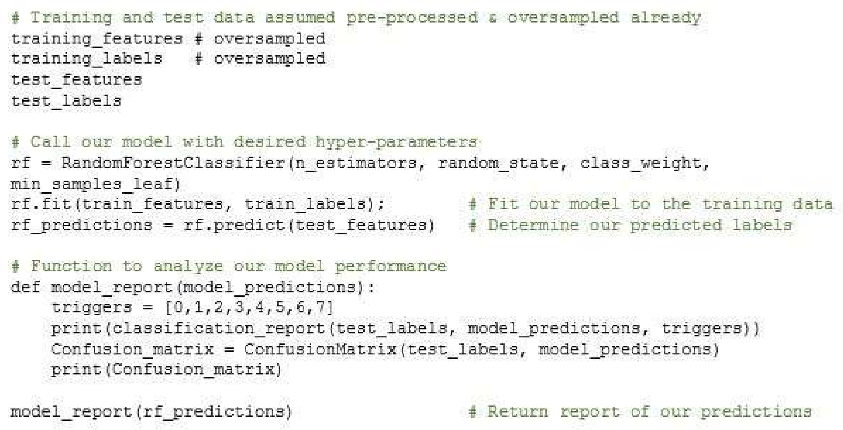
Fig 2. Data distribution for landslide triggers within our dataset

Class Imbalance. Since our dataset contained a large class imbalance with the rain label making up 87% (7966/9133) of the total data, we used oversampling methods of ADASYN, and SMOTE to rebalance our dataset. Our first method, Adaptive Synthetic Sampling Approach (ADASYN), uses a weighted distribution for different minority class instances according to their level of difficulty in learning, which leads to more synthetic data generated for minority class examples that are harder to learn [3]. Similarly, Synthetic Minority Over-sampling Technique (SMOTE) oversamples the minority class by taking each minority class sample and introducing synthetic examples along the line segments joining the minority class' k-nearest neighbors (kNN)[1] [4].

2.2 Methods

After data preprocessing and oversampling for minority classes was complete, we selected Random Forest Classifier and Support Vector Machines to predict the landslide trigger. Both of these methods are used commonly with supervised learning classification problems that have class imbalance. We chose to implement both RFC and SVM in order to compare the two methods and determine which preformed the best.

Random Forest Classifier (RFC). The random forest classifier is ensemble algorithm that creates a set of decision trees from our training dataset. Using majority votes from each decision tree the random forest decides the class label of the test data object. With the RandomForestClassifier library provided by Scikits-learn we trained our model with our oversampled training data from ADASYN. Pseudo-code for the core part of the method can be seen in Figure 3 below.



```
# Training and test data assumed pre-processed & oversampled already
training_features # oversampled
training_labels   # oversampled
test_features
test_labels

# Call our model with desired hyper-parameters
rf = RandomForestClassifier(n_estimators, random_state, class_weight,
min_samples_leaf)
rf.fit(train_features, train_labels);        # Fit our model to the training data
rf_predictions = rf.predict(test_features)   # Determine our predicted labels

# Function to analyze our model performance
def model_report(model_predictions):
    triggers = [0,1,2,3,4,5,6,7]
    print(classification_report(test_labels, model_predictions, triggers))
    Confusion_matrix = ConfusionMatrix(test_labels, model_predictions)
    print(Confusion_matrix)

model_report(rf_predictions)                 # Return report of our predictions
```

Fig 3. Pseudo-code for core function of Random Forest Classifier

Support Vector Machines (SVM). Support Vector Machines (SVM) is a discriminative classifier that separates classes through linear separators. Although SVM was originally designed for binary classification, it can also be used to solve multi-class classification problems like ours. For our method we chose to implement one-vs-one which trains a separate classifier for each different pair of labels. In turn we get,

$$\text{Classifiers} = N\,(N-1)/\,2 \qquad (1)$$

[1] In k-nearest neighbor(kNN) classification is achieved by identifying the nearest neighbors to a query example and using those neighbors to determine the class of the query [5].

where N is the number of different class labels. This method was chosen due to its insensitivity towards problems of imbalanced datasets but has a tradeoff of being computationally expensive. We oversampled with SMOTE for our SVM model.

2.3 Strategy

Determining the best hyper-parameters for both of our models proved to be our most challenging task. Our approach consisted of a combination of trial and error and the use of Scikit-learn's GridSearchCV tool. When we received our performance scores we evaluated them, adjusted our hyper-parameters as needed and trained our models again. This process was completed a number of times. With GridSearchCV we obtained a starting point for declaring the hyper-parameters for our RFC and SVM methods. When running our SVM model through GridSearchCV to calculate the C parameter we received the results seen in figure 4.

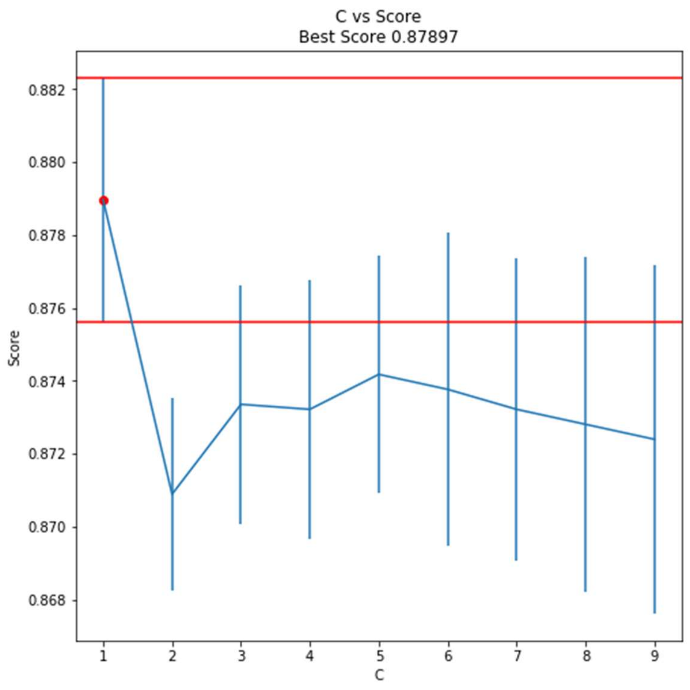


Fig 4. Grid search results for Support Vector Machines hyperparameter C

The C parameter tells the SVM optimization how much we want to avoid misclassifying each training example. In our case the best score came from a C parameter of 1. We also used GridSearchCV to help us identify the parameters for our RFC which gave us the following best scores: n_estimators=1500, random_state=50, min_samples_leaf=75 and class_weight = 'balanced'. We encorperated each of these hyperparameters into our models and found that overall performance increased.

2.3 Evaluation Metrics

We used precision, recall and f1-score to evaluate the performance of our model. These performance metrics allowed us to view how each class in our model was reacting to the hyperparameters that we set.

Precision. The portion of positive class predictions that were correct. This helped us determine when the costs of false positive values were high.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2}$$

Recall. The portion of actual true positive values that were correct. We looked at this value the most when analyzing our model performance.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{3}$$

F1-Score. Measure used when we need to search for a balance between precision and recall while having an uneven class distribution.

$$\text{F1-Score} = 2 \text{ x } \frac{Precision \text{ x } Recall}{Precision + Recall} \tag{4}$$

3 Results

During our experiment we could immediately see that after oversampling our minority classes with ADASYN and SMOTE, we still had a bias towards our majority class. Due to the imbalanced class labels our accuracy score would be misleading to the true performance of our model. For this reason, we decided to evaluate of model's performance based off the precision, recall, and f1-score. Although oversampling provided us with an even distribution of training data, when predicting the class label with our test data we had relatively low performance scores.

2.3 Improving Performance

In an attempt to improve our performance, we revisited the original dataset and removed noisy data. By filtering each attribute by unique values we removed data that occurred infrequently and imputed missing values with the median of that attribute. Feature creation was used in order to split the event_date attribute into an event_month column, removing the event_date from our dataset. After these procedures we saw a slight improvement in our models (see Table 2).

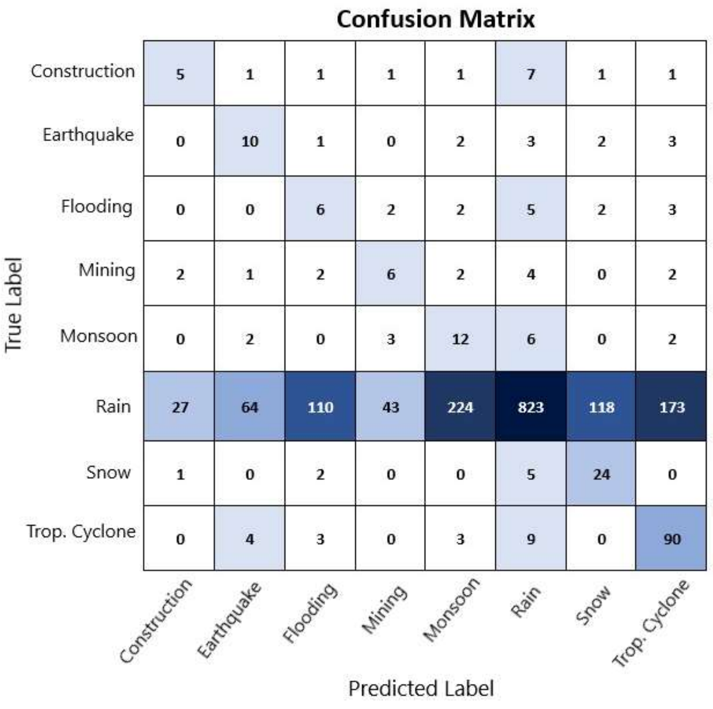
Table 2. Performance of RFC with ADASYN and SVM with SMOTE

Class	**Random Forest Classifier**			**Support Vector Machines**			
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Support
Construction	0.14	0.28	0.19	0.03	0.61	0.06	18
Earthquake	0.12	0.48	0.19	0.15	0.48	0.22	21
Flooding	0.05	0.30	0.08	0.07	0.15	0.09	20
Mining	0.11	0.32	0.16	0.04	0.11	0.06	19
Monsoon	0.05	0.48	0.09	0.08	0.24	0.12	25
Rain	0.95	0.52	0.67	0.95	0.58	0.72	1582
Snow	0.16	0.75	0.27	0.26	0.62	0.37	32
Tropical Cyclone	0.33	0.83	0.47	0.31	0.50	0.38	109

2.3 Model Evaluation

When evaluating our model performance's, we found that the majority class had the highest precision score for both RFC and SVM. However, this was not necessarily a bad thing since the support from the test dataset had 1582 instances of the 'Rain' class. When looking at the minority classes we could see that the support levels of the test dataset most likely caused the precision and f1-score to be so low. With almost all of our under-represented classes having less than 100 support instances there is little room for misclassification. However, the recall score for our minority classes score in both models held higher scores, which means that our models were able to get more actual true positive values correct.

Confusion Matrix. The precision and recall scores can also be visualized in a confusion matrix with the actual values and predicted labels. Our confusion matrix for the random forest classifier using ADASYN can be seen in figure 5, while the confusion matrix for support vector machines using SMOTE is seen in figure 6.



	Construction	Earthquake	Flooding	Mining	Monsoon	Rain	Snow	Trop. Cyclone
Construction	5	1	1	1	1	7	1	1
Earthquake	0	10	1	0	2	3	2	3
Flooding	0	0	6	2	2	5	2	3
Mining	2	1	2	6	2	4	0	2
Monsoon	0	2	0	3	12	6	0	2
Rain	27	64	110	43	224	823	118	173
Snow	1	0	2	0	0	5	24	0
Trop. Cyclone	0	4	3	0	3	9	0	90

Fig 5. Confusion Matrix for Random Forest Classifier (RFC) with ADASYN

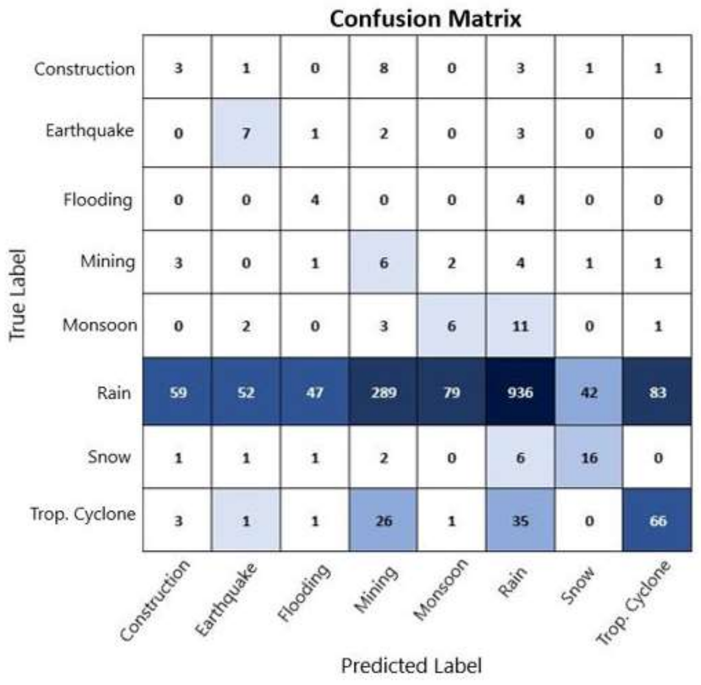


Fig 6. Confusion Matrix for Support Vector Machines (SVM) with SMOTE

Related Work. With no related work to our problem we didn't have another experiment to compare our results to. However, after analyzing our results we can see that the reason no related work has been completed may be due to the difficulty of the problem. This makes our problem interesting because although previous works with predicting landslide susceptibility provides insight on the possibility of a landslide, our work focuses on the reasons what causes landslides occur in the first place.

4 Conclusion

The methods used to solve our problem provided insight on the relationship between the landslide triggers and their features. With the sampling techniques of ADASYN and SMOTE we were able to balance the dataset through oversampling. However, we still experienced below average performance with both our models. Random forest classifier performed slightly better than SVM with an average mean absolute error of 0.86, where SVM had an average of 1.03. Although RFC appears to be the better model for our problem, they both are susceptible to producing both false positives and false negatives. It appears that the current challenge of this work simply is that the data is not very good. The dataset required an extensive amount of preprocessing and lacked attributes that could help us solve this problem.

For future work we would look to solve this problem again using a larger dataset possibly with additional geographical attributes directly relating to the landslide. Some attributes of suggestion would be the type of soil, the duration of the landslide event, or the elevation level at which the landslide took place. Another future work would be to implement another method such as Artifical Neural Networks (ANN).

References

1. Binh, P., Biswajeet, P., Dieu, B., Indra P.: A comparative study of different machine learning methods for landslide susceptibility assessment: A case study of Uttarakhand area (India), In: Environmental Modelling & Software, vol. 84, pp. 240-250, ISSN 1364-8152, (2016). https://doi.org/10.1016/j.envsoft.2016.07.005.
2. N. Global Landslide Catalog Export. NASA. (2016, March 7). https://data.nasa.gov/Earth-Science/Global-Landslide-Catalog-Export/dd9e-wu2v
3. Haibo H., Yang B., E. A. Garcia and Shutao L., "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, Hong Kong, 2008, pp. 1322-1328.
4. Chawla, N. Bowyer, K., Hall, L., Kegelmeyer, W.: SMOTE: Synthetic Minority Over-sampling Technique. JAIR. (2002). https://jair.org/index.php/jair/article/view/10302
5. Cunningham, P., Delany, S.: k-Nearest neighbour classifiers. (2007). Mult Classif Syst. https://www.researchgate.net/publication/228686398_k-Nearest_neighbour_classifiers
6. Provost, Floriane & Hibert, Clément & Malet, J.-P. (2016). Automatic classification of endogenous landslide seismicity using the Random Forest supervised classifier: Seismic sources automatic classification. Geophysical Research Letters. 44. 10.1002/2016GL070709.
7. Micheletti, Natan & Foresti, Loris & Robert, Sylvain & Leuenberger, Michael & Pedrazzini, Andrea & Jaboyedoff, Michel & Kanevski, Mikhail. (2013). Machine Learning Feature Selection Methods for Landslide Susceptibility Mapping. Mathematical geosciences. 46. 10.1007/s11004-013-9511-0.