Predicting Risk to Human Life Due to Landslides

Ashley Kleen

Loyola University Maryland

DS796, Spring 2025

Table of Contents

Landslide Risk Classification of Reported Data 0

Abstract 4

Motivation and Background 5

Motivation 5

Background and Objective 5

Dataset 6

Project Plan and Research Platforms 6

Project Plan 6

Jupyter Notebooks 7

Python Packages Utilized 8

Tableau 8

Literature Review 9

Data Analysis and Discussion 10

Exploratory Data Analysis 10

Data Pre-processing 14

Model Analysis 15

Random Forest Model and Analysis 16

CART Model and Analysis 17

SVC Model and Analysis 19

SVCLinear Model and Analysis 20

Hypothesis Conclusions 22

Threats to the Validity 22

Problems with Reported Data 22

Assumptions Made 24

Missing Data 24

Future Work 24

Reflections 25

References 27

List of Figures

Figure 1 *Map of Landslides and Risk* 11

Figure 2 *Map of Risk in Himalayan Mountains and South-East Asia* 12

Figure 3 *Count of Landslides by Season and Cause* 13

Figure 4 *Count of Landslides by Size and Classification* 14

Figure 5 *Permutation Importance for Final Random Forest Model* 16

Figure 6 *Confusion Matrix for Final Random Forest Model* 17

Figure 7 *Permutation Importance for Final CART Model* 18

Figure 8 *Confusion Matrix for Final CART Model* 19

Figure 9 *Confusion Matrix for Final SVC Model* 20

Figure 10 *Confusion Matrix for Final CART SVCLinear Model* 21

Figure 11 *Count of Reports by Year and Colored by Injury/Fatality* 23

List of Tables

Table 1: *Python Packages Utilized* 8

Table 2: *Top 10 Important Features for Final SVC Model* 20

Table 3: *Top 10 Important Features for Final SVCLinear Model*..... 21

Abstract

The main focus of this study is to determine whether or not reported data on landslides can be used to accurately predict risk to human life, to reveal factors that go into the determination, and to investigate which model is best (Random Forest, classification and regression trees (CART), Support Vector Classification (SVC), or SVCLinear). The dataset selected contains reports from all over the world. First, an exploratory data analysis is conducted in Tableau to show where the data is being reported, where the worst landslides take place, when the most landslides occur, and what the most common sizes/types of landslides are. Next, Python was utilized to clean the data and to fill in or remove any NAN data, new variables are created for ‘risk’, ‘biome’, and ‘month’, ‘year’, and ‘season’. SelectKBest feature selection and removal of correlated data with correlation score greater than .80 was completed next. Fourth, the data was split into 70/30 train and test subsets and the four different models were created with hyperparameters that were selected using GridSearchCV. Last, the models were validated using both precision, recall, and F1-scores and correlation matrices. The Random Forest model with a training accuracy of 0.998 and test accuracy of 0.957 was selected as the best possible model for this dataset with the CART model a close second choice.

Motivation and Background

Motivation

Natural disasters can be devastating, so studying them and their effects is beneficial to human life. Studying earthquakes and tornadoes has led to advanced building architecture and structural planning; studying at-risk areas for fires and flooding has led to better decision-making for construction placements; and studying the ocean's patterns and force has led to better, stronger ships and submersibles that can cross great distances and withstand outstanding pressures. Normally, we take in alerts of risk at face value and do not stop and think about what factors go into determining the risk. Therefore, in this project, we will investigate landslides and which factors go in to determining high-risk areas so we can better understand the world around us.

Background and Objective

The current trend in studying rain-triggered landslides is using live or semi-live satellite data to track risk and patterns, but what did we do before this, how accurate was it, and what can it be used for? Because although the future is to use satellite feeds, we must also understand past events so that we can plan for the future. The data uncovered in this report could be used for city-planning, construction placement and protocols, and in staffing of search-and-rescue teams among other things.

This study aims to uncover the main risk-determining factors for landslides. We will develop a factor of risk using the reported number of injuries and fatalities, and then this risk-factor will be put to the test in both linear and non-linear machine learning models to see if the reported data is enough to predict outcomes.

Dataset

NASA's Global Landslide Catalog (GLC) dataset consists of reports of rain-triggered landslides across the world coming from media, scientific reports, disaster databases, and other sources from its start in 2007 – March 2016. It has 31 columns and 11033 observations describing the source, data/time, news title, location, size, and fatality & injury counts. Due to its nature of being reported data, there were quite a few pieces of missing data that had to be accounted for. A reverse lookup of latitude and longitude was completed to fill in for country and city data. Unreported injuries and fatalities was filled with zeros, and then summed for a combined risk. At total of 4 levels was determined when it was discovered that some events reported over 3,000 injuries/fatalities. To improve the dataset, new factors for season and biome was added using the geographical data provided.

Project Plan and Research Platforms

Project Plan

The approach to this study can be divided into 5 different phases including:

1. Selecting the dataset, cleaning the data, and creating new variables that would add value. This took a significant amount of time due to missing values and the need to use reverse-lookups to obtain city and biome data using the provided latitude and longitude;
2. Reviewing similar projects to confirm thesis direction and selection of machine learning models;
3. Selecting features using SelectKBest and checking for multicollinearity. This proved to be quite a challenge. The initial 12 features increased to 4715 features after using

get_dummies to ensure all non-numerical data was converted into numerical data. SeleckKBest was run and the dataset filtered by score, then evaluated each time by running a default SVC model until a final decision was made;

4. Running models using SVC, Random Forest, and CART models and determining the best possible hyperparameters using GridSearchCV. This also took a significant amount of the time. GridSearchCV is a great indicator, but is slow to run on the amount of data, and the SVC models made this process even slower; and
5. Running validation tests and determining the best possible model.

Jupyter Notebooks

All data processing, cleaning, and modeling was completed using Python in Jupyter Notebooks. A total of 3 notebooks were created: The first notebook was created for processing the data, cleaning the data, creating new variables, and creating basic, preliminary visualizations, the second notebook was created for model pre-processing (feature selection and correlation removal), and the third notebook was created primarily for modeling.

Python Packages Utilized

Python packages and their uses are described in the below table.

Table 1:
Python Packages Utilized

Package	Use
Pandas	View and manipulate the data
scikit-learn (StandardScaler , LogisticRegression, SelectKBest, train_test_split, SVM, RandomForestClassifier, and DecisionTreeClassifier)	Create training and test subsets, feature selection, and run modeling tests
geopy, geocoder, and shapely	Retrieve country and biome data from the given latitude and longitude
Matplotlib	Create basic, preliminary visualizations in the process of data exploration

Tableau

Tableau offers a unique way to see patterns in the data. Users can explore and view the data and its complexities using clean, interactive dashboards. The map shows where the reported landslides are located and the damage to human-life the landslide had on the population. Examining when landslides take place and what types are prevalent is essential in planning for disaster-relief funding and safe evacuation routes/locations. Finally, using a graph of when the reported landslides took place, we can understand where the data is coming from and which years were the least and most catastrophic.

Literature Review

The research of landslides allows us to identify areas of risk and possible hazards. Predicting landslide risk aids in making better decisions regarding construction, road closures during natural disasters, and preparing for rescue operations. Data gathering, analysis tools, and verification are essential to predicting landslide risk.

The methodology for data gathering varies depending on the research question, data availability, and scope of research. Most data comes from field surveys, local datasets on soil and rock compositions, or remote sensing data. While gathering this information is relatively easy for small research areas, larger areas present significant challenges. Large areas can be divided into grid cells, mapped to 0 and 1 depending on past presence of landslides, and then analyzed by logistic regression, but (Van Den Eeckhaut et al., 2006) found that this does not account for areas where landslides occur frequently versus infrequently. Identifying landforms is also a challenge because landforms to do not have clear boundaries, but machine learning can be used with fuzzy k-means clustering (Gorsevski et al., 2003) on remote sensing data and/or Geographic Information Systems (GIS) data to classify landform types.

Methods for analyzing landslide hazard risk include Random Forest algorithms (Frodella et al., 2022; Chen et al., 2017), stepwise logistic regression in SAS (Van Den Eeckhaut et al., 2006), CART (Zhao et al., 2016; Chen et al., 2017), Support Vector Machine (SVM) models (Zhao et al., 2016), Bayesian models (Gorsevski et al., 2003), and logistic model trees (LMT) (Chen et al., 2017). In a study of model selection between LMT, Random Forest, and CART models (Chen et al., 2017), it was found that while all did well, the Random Forest model offered the best prediction capability. In a study predicting groundwater levels of landslides, an SVM model was compared to a CART model (Zhao et al., 2016). The study suggested that while

both models were effective, the CART model was ultimately selected due to it using less computational cost and its performance in terms of fitting, generalization, and variable selection.

Verification methods used were uniform when mentioned. Van Den Eeckhaut et al. (2006) used Akaike's information criterion (AIC), the negative of twice the log likelihood (− 2 log L), area under the ROC curve (AUC), and Cohen's Kappa index. Chen et al. (2017) also used ROC curves and general predictive accuracy (ACC) methods.

Building upon these studies, it is important to examine if one of these model types is effective in predicting key areas around the world that are most at-risk.

Data Analysis and Discussion

Research has been conducted and proved on the likelihood of occurrence of a landslide, but what factors are contributing to the likelihood of a landslide causing harm to humans? Is there a specific region on earth that is inherently more dangerous than others, and can we predict the risk to human life based on this dataset? The expectation is that highly populated and mountainous areas will the most dangerous and that most landslides will occur during the rainy season, but are there other factors?

In the following subsections, will begin by exploring the data in Tableau, review the steps taken for data pre-processing, investigate and select the best possible model, and review what we can conclude from our models.

Exploratory Data Analysis

An initial exploration of the dataset was completed in Tableau. A map of the landslides was created. My initial hypothesis that there are certain areas of the world that are more dangerous is confirmed. The map in Figure 1 details the landslides reported in the dataset: the

circle size indicates the count of injured/deceased, and the color indicates the level “Risk” which is calculated from that count. As shown, there is a higher propensity for landslides to occur near major mountain ranges. We can interpret that the highest number of injured/deceased occurred in the Pakistan-India region of the Himalayan Mountains (Figure 2).

Figure 1

Map of Landslides and Risk

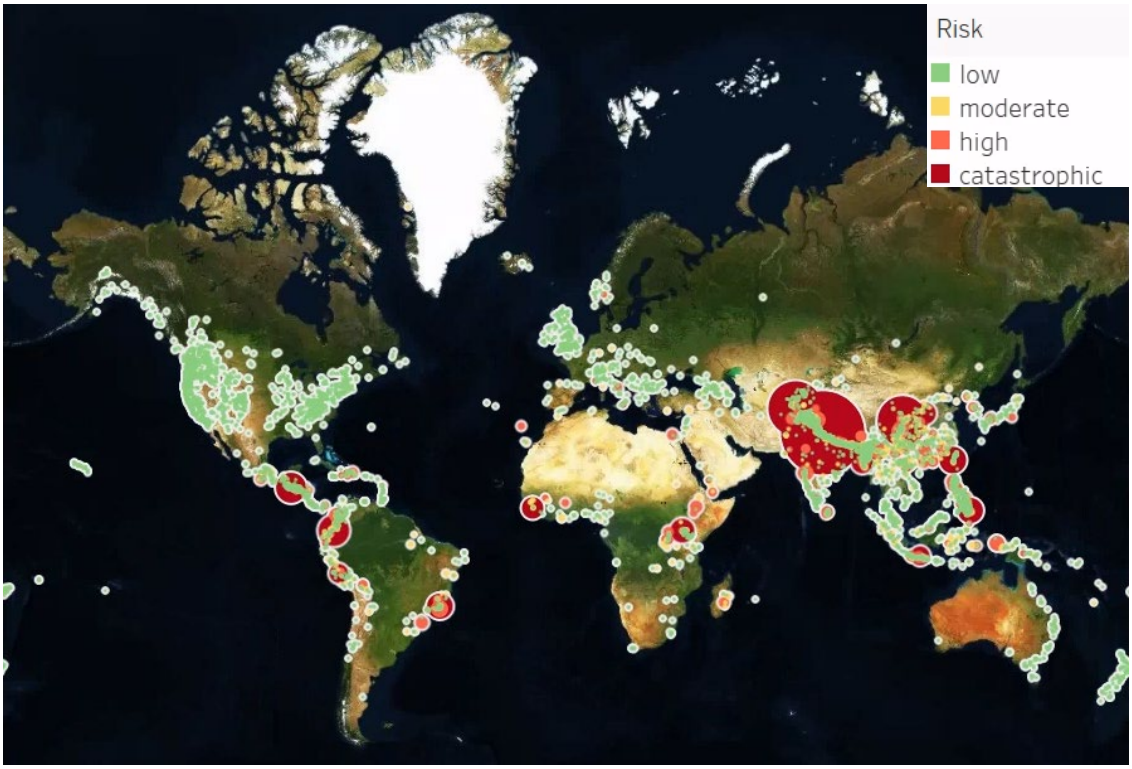
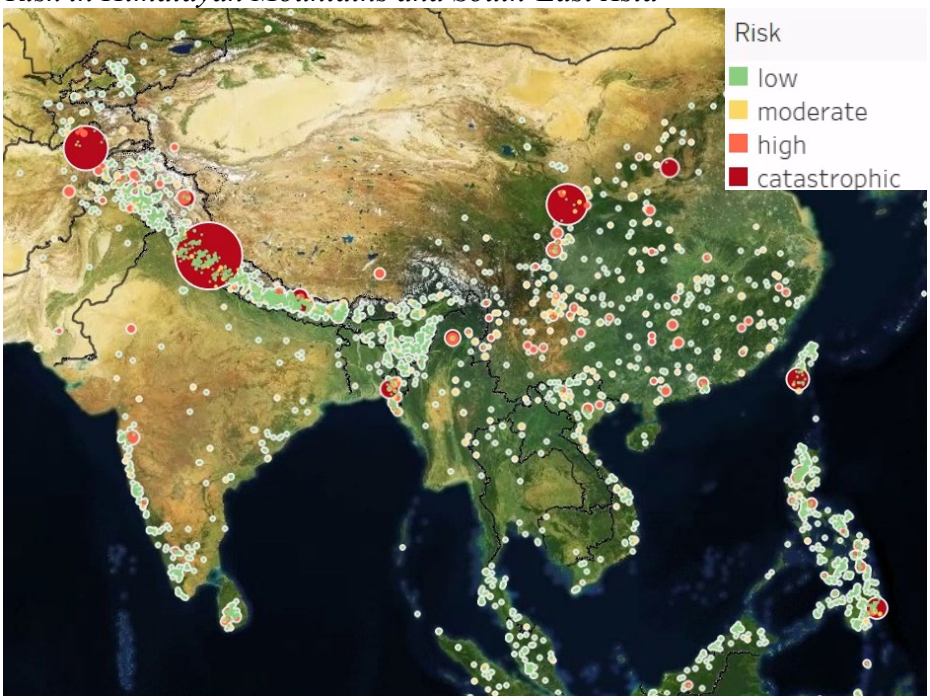


Figure 2

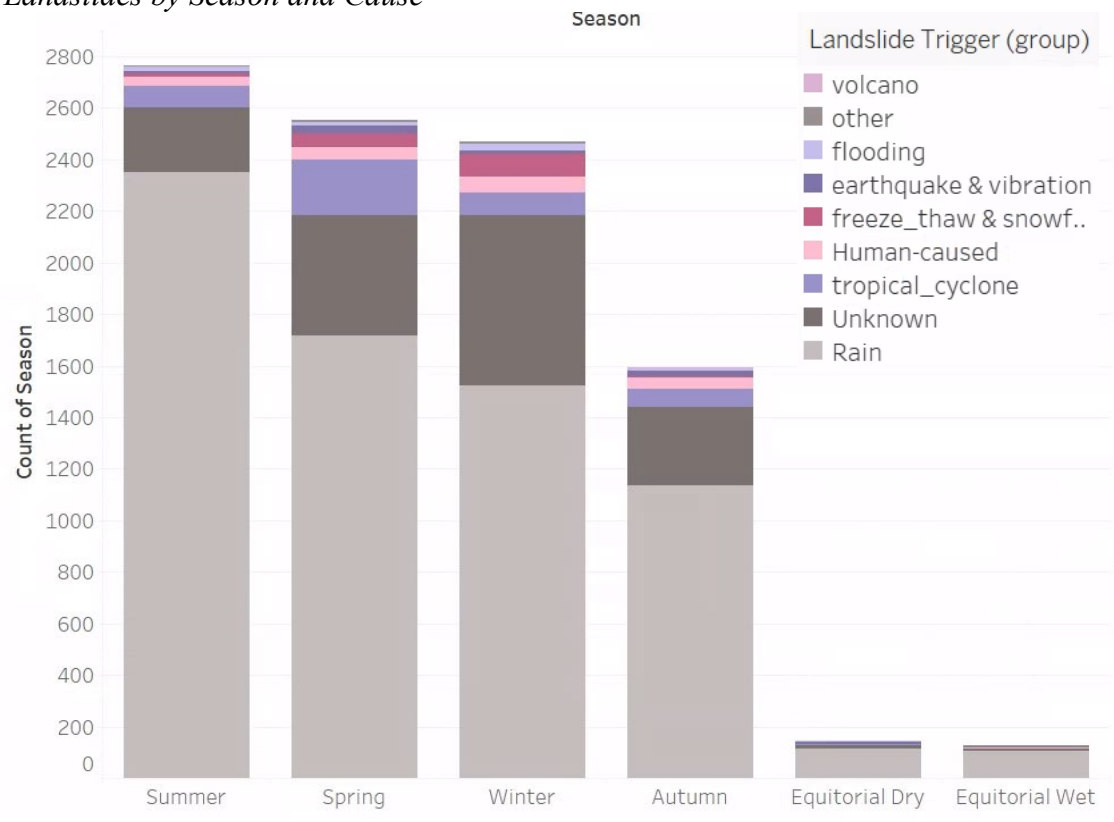
Map of Risk in Himalayan Mountains and South-East Asia



An examination of when the landslides occurred and the types revealed that summer is when the majority of landslides occur, and in the equatorial region, the dry season saw slightly more occurrences. The top 4 causes of landslides include rain, an unknown reason, tropical cyclones, and human-caused (either by mining or other construction).

Figure 3

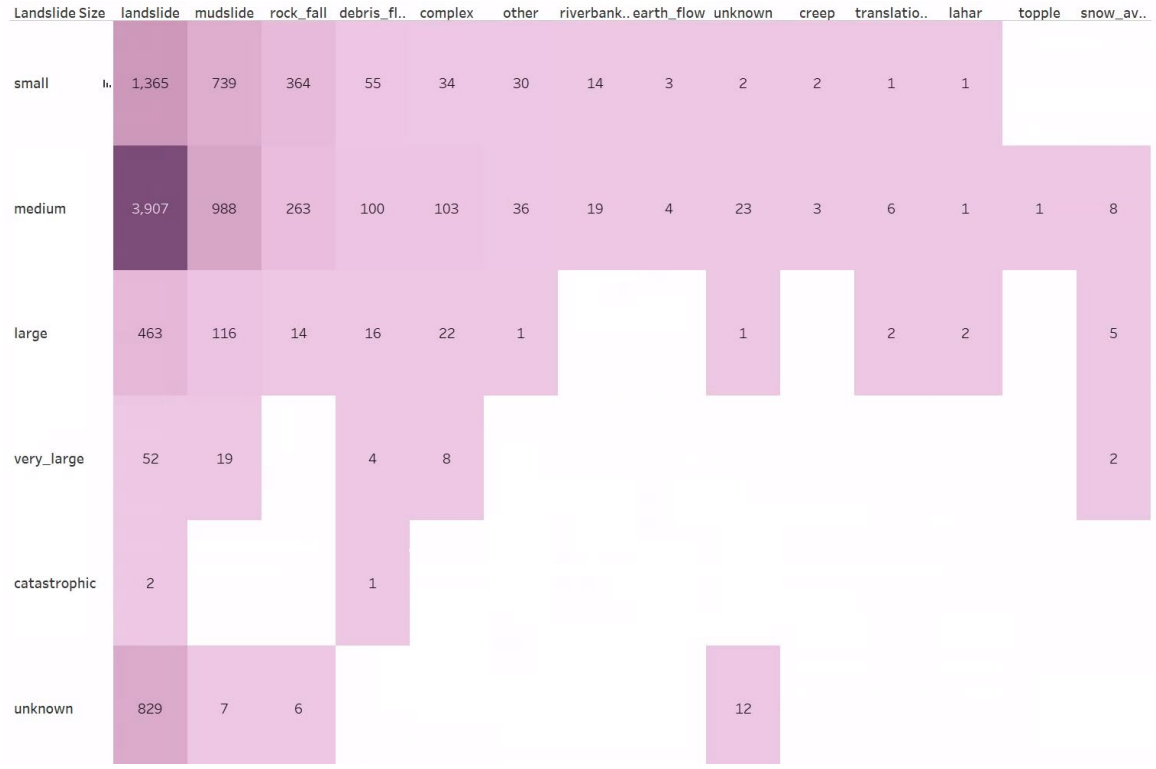
Count of Landslides by Season and Cause



Taking a closer look at landslide types shows that the majority of landslides are categorized as either small or medium in size, and are classified as generic landslides, as mudslides, or as rock falls, as shown in Figure 4.

Figure 4

Count of Landslides by Size and Classification



Data Pre-processing

To investigate the likelihood of a landslide causing human harm, a target variable 'risk' has been created with four levels based on the number of reported injured or deceased due to the disaster. To investigate if there are other factors involved, a new variable to season was created and determined based on longitude, and new variable for biome was created using the provided latitude and longitude and the World Wildlife Federation's dataset on world biomes.

The initial dataset has 11,033 observations and 31 features, and after removing unnecessary features such as storm name, event/image source links, or other IDs, 13 features remained. Missing data was filled in wherever possible using the provided latitude and longitude, or event title. Fatality count and injury count was combined and used to create a 'risk' variable,

and missing values were filled in with 0. Before converting non-numerical data, 9,656 observations and 12 features remained.

Conversion of nun-numerical data was completed using pd. get_dummies. This increased the total features to 4,715. Feature selection was completed using SelectKBest, removing correlated data with a correlation score of 0.80 or more, and checking training/test accuracy with SVC and Random Forest. Due to initial oversight, a check of class imbalance was completed after this step and data was upsampled to the highest risk factor. The final dataset contains 35,044 observations and 4,691 features.

Four different predictive models were chosen for evaluation to determine the best possible model: Random Forest, CART, SVC, and SVCLinear models.

Model Analysis

To begin analysis, the data was split 70-30 training and test sets, scaled using sc.transform due to the different types of data presented, random_state was set to 1, and best parameters were determined using GridSearchCV. All models are then cross-validated using confusion matrices and determining the precision, recall, and F1-scores.

In comparing the precision and recall scores, and the confusion matrices for all models, although the CART model did similarly well, the Random Forest model is the best choice for this dataset. In comparison, the SVC models did not do as well in classification with consistently lower test and accuracy scores, and precision and recall scores. When comparing the two SVC model types, the SVCLinear is outperforms the SVC model, but both have significantly lower scores than the Random Forest or CART models.

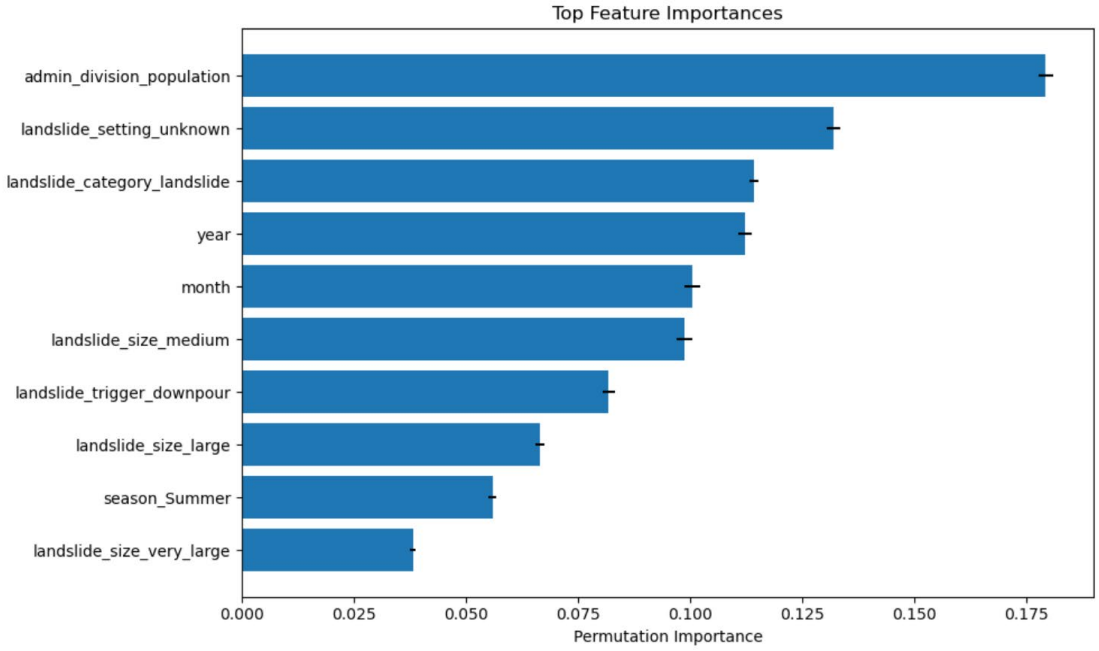
Random Forest Model and Analysis

The parameters chosen for the final Random Forest model included the following: 'criterion' = 'gini', 'max_depth' = None, 'max_features' = 11, 'n_estimators' = 40, and 'n_jobs' = 1. The training accuracy reported was 0.998 and the testing accuracy reported was 0.957. A test of precision, recall, and F1 score resulted in 0.957 for all; this is due to the nature of a classification tree model.

Feature importance for a Random Forest Model must be reviewed with caution. The following are actually permutation importance which means they do not reflect the predictive value, but how important the feature is to the model. As shown in Figure 5, the population holds the highest importance in this model which partially supports the hypothesis that highly populated mountainous areas are inherently more dangerous, but there is no high importance on any one geographical type that might be more affected. From this graph, we can begin to infer that perhaps the general size, trigger, and season are more influential.

Figure 5

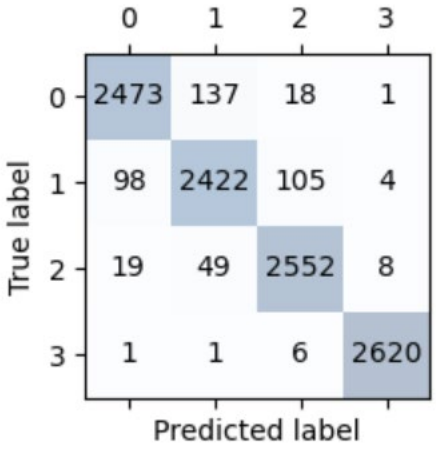
Permutation Importance for Final Random Forest Model



Because the precision, recall, and F1-scores are the same for a Random Forest model (0.957), it is important to further verify with a confusion matrix as shown in Figure 6. Investigation shows us that only 6.4% of class 0 is mislabeled and only 8.5% of class 1 is mislabeled while neither class 2 or 3 are mislabeled more than 3% of the time. If we look at it as a whole, only 4.5% is mislabeled. We can conclude, then, that this model is very well.

Figure 6

Confusion Matrix for Final Random Forest Model



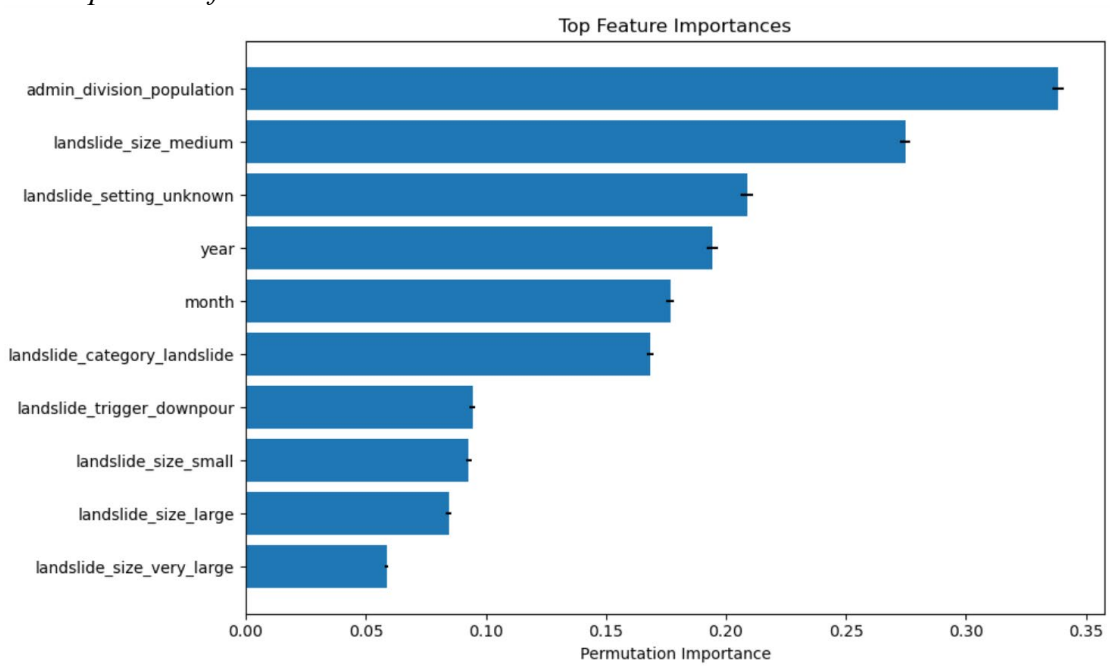
CART Model and Analysis

The parameters chosen for the final CART model included the following: 'criterion' = 'log_loss', 'max_depth' = 61, 'min_weight_fraction_leaf' = 0.0, and 'splitter' = 'best'. The training accuracy reported was 0.998 and the testing accuracy reported was 0.951. A test of precision, recall, and F1 score resulted in 0.951 for all; this is due to the nature of a classification tree model.

Like a Random Forest model, feature importance for a CART Model must be reviewed with caution. The following are actually permutation importance which means they do not reflect the predictive value, but how important the feature is to the model. See Figure 7 for a complete list and the importance of the top ten features in this model. The features used in this model are

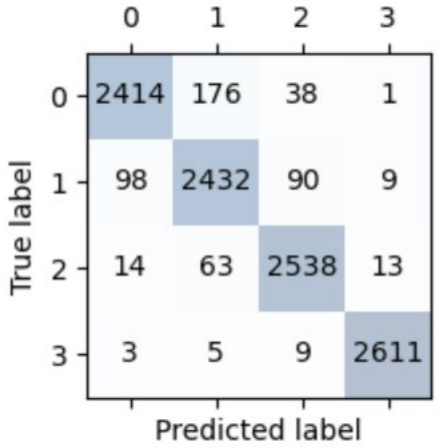
nearly identical to the Random Forest model with population, size, and trigger among the highest used, but we should take notice that season is no longer on this list. Perhaps season is not as important as we might think.

Figure 7
Permutation Importance for Final CART Model



Because the precision, recall, and F1-scores are the same for a CART model (0. 951), it is important to further verify with a confusion matrix as shown in Figure 8Figure 6. Investigation shows us that only 8.9% of class 0 is mislabeled and only 8.1% of class 1 is mislabeled while neither class 2 or 3 are mislabeled more than 3% of the time. If we look at it as a whole, only 5.2% is mislabeled. We can conclude, then, that this model is also predicting very well, but not as well as the Random Forest.

Figure 8
Confusion Matrix for Final CART Model



SVC Model and Analysis

The parameters chosen for the final SVC model included the following: 'kernel' = linear, 'decision_function_shape' = ovr, and C was determined best as the default, 1.0. The training accuracy reported was 0.801 and the testing accuracy reported was 0.714. A test of precision, recall, and F1 score resulted in 0.736 for precision, 0.713 for recall, and an F-score of 0.718.

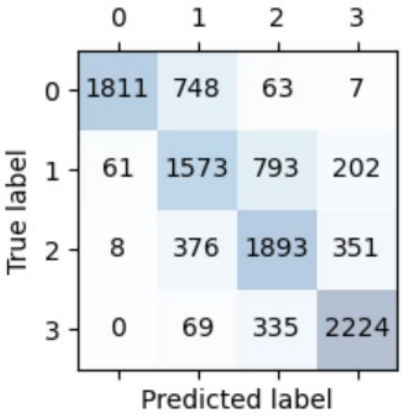
The top ten most important features for the SVC model are listed below in Table 2. Again, we see that landslide size, trigger, category, and the unknown setting being high contributors. We now see, though, the addition of country names and biomes in the list.

Table 2:
Top 10 Important Features for Final SVC Model

Feature	Coefficient
landslide_size_medium	1.237586
country_name_China	1.020584
landslide_size_small	0.952286
landslide_category_landslide	0.924603
gazeteer_closest_point_unknown	0.823382
landslide_setting_unknown	0.801074
landslide_trigger_downpour	0.679744
biome_Serra do Mar coastal forests	0.676831
landslide_setting_above_road	0.652958
country_name_United States	0.636420

A review of the confusion matrix in Figure 9 confirms that the model is not doing particularly well. Many items are being misclassified, so we should not use this model for our final prediction.

Figure 9
Confusion Matrix for Final SVC Model



SVCLinear Model and Analysis

The parameters chosen for the final SVCLinear model included the following: 'dual' = False, 'penalty' = l1, 'max_iter' = 20000, 'loss' = squared_hinge, and 'C' = 0.02. The training accuracy reported was 0.801 and the testing accuracy reported was 0.714. A test of precision, recall, and F1 score resulted in 0.765 for precision, 0.764 for recall, and an F-score of 0.762.

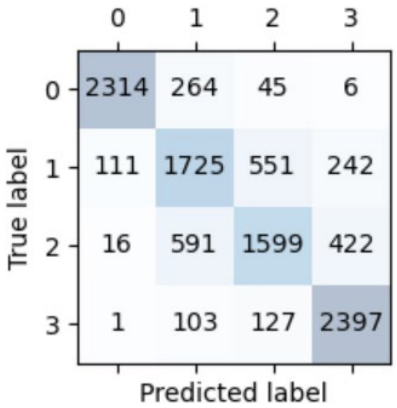
The top ten most important features for the SVCLinear model are listed below in Table 3. This shows us that the landslide size, trigger, category, and the unknown setting are still high contributors, but the countries and biome chosen differ among the SVC type models.

Table 3:
Top 10 Important Features for Final SVCLinear Model

Feature	Coefficient
country_name_United States	0.583482
landslide_size_medium	0.349732
landslide_size_small	0.297659
landslide_trigger_downpour	0.205534
landslide_setting_above_road	0.175938
landslide_setting_unknown	0.170389
country_name_Canada	0.155604
country_name_United Kingdom	0.149071
landslide_category_landslide	0.116058
biome_Western Himalayan broadleaf forests	0.109023

A closer inspection of the confusion matrix as shown in Figure 10 further confirms that while still not performing well, the SVCLinear model is performing slightly better than the base SVC model.

Figure 10
Confusion Matrix for Final CART SVCLinear Model



Hypothesis Conclusions

From the features found to be important, we can conclude that local population, size, category, and trigger are all contributors to the risk of human life in a landslide. We have been successful in being able to predict risk to human life based on this dataset within 99% accuracy. There is no one particular place that is inherently more dangerous, but the maps consistently show a higher number of reports from mountainous areas. We may be able to infer that larger landslides occur with larger mountains, but further research and more data is needed to confirm this hypothesis.

It is difficult to determine how impactful the season may be since it was only included in the top 10 for the Random Forest model, however, since month was still used in both the Random Forest and CART models, it can be concluded that is has some level of impact. Landslide setting was not used as much in the classification trees, but was used in both SVC models, so there may be some impact from setting as well.

It is extremely important to note that the predictions for risk to human life in this dataset are predicting past events, meaning that since the models are heavily reliant on the size of the landslide and its triggers, we cannot predict future risk because we do not yet know the size or location of the next landslide to occur. This data can be used directly after an event to predict need for hospitals or search and rescue events, but cannot be used to be pro-active.

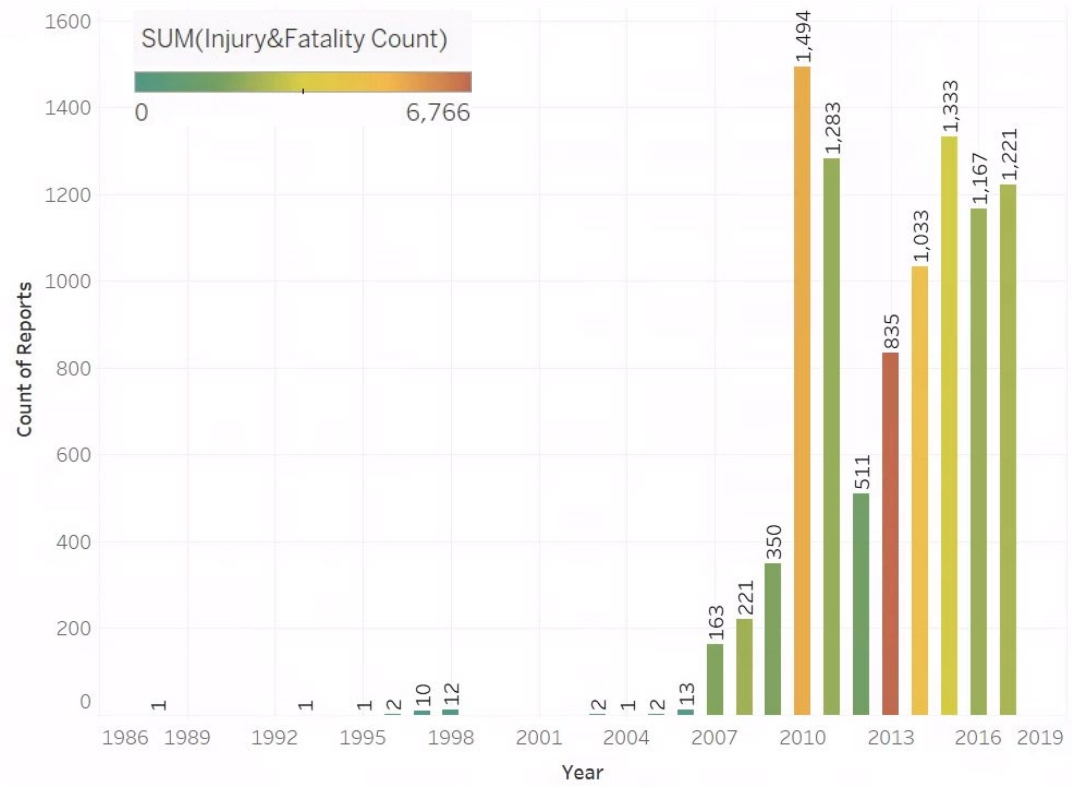
Threats to the Validity

Problems with Reported Data

Some level of caution must be used while using similar datasets due to the number of events being reported and by whom. As shown in Figure 11, there are many years with few

reported events and some years with many events. This is likely not due to the actual number of events being lower, but due to awareness of reporting sites and general reporting practices. It could also be due to news sources that are in English vs another language.

Figure 11
Count of Reports by Year and Colored by Injury/Fatality



Another inherent problem with reported data is that we have to assume that all facts listed are true and complete, and that these are the final reports. It was highly suspicious that exactly 5,000 fatalities occurred in a landslide in India with 0 reported as injured and that 2,100 fatalities occurred in a landslide in Afghanistan also with 0 reported as injured. We have seen in many reports over the last few years that the initial reports and the final reports are, at times, weeks apart, so it must be assumed that fallacies are going to exist.

Assumptions Made

It is also important to note that the true number of injured and deceased may differ due to non-reporting – it was assumed in this project that if none were listed in the dataset, then the count is zero. It was also assumed that biome determination would be more important than country/city, so the country/city information was removed in the correlation removal process.

Missing Data

This dataset often had similar columns for event title, event description, notes, and others where data was either repeated, misplaced, or left completely blank. Landslide setting is a great example of this since the 'unknown' setting was the most often used in the models - the information was likely included, but since it was not written in the correct column, our research ultimately suffers from it.

Data that could be useful, but is not included in this dataset is the general geographical data where the landslides take place. Knowledge of elevation, mountain/hill steepness, and composition (rocky, silt, or forested) could have greatly improved the data so that prediction of future catastrophes could be predicted. Average rainfall, drought status, and other factors could also improve this dataset but are unlikely to be included.

Future Work

Related Projects

One related project that immediately comes to mind is using location data to predict where landslides are most likely to occur. Researchers have used other methods for doing similar tasks, so it would be good to try other machine learning models.

Improvement Projects

This project can be improved upon in several ways that were not done due to time constraints. A researcher can likely use webscraping to find the missing values of population by year and location, or gathering more resources by searching for “landslide” in other languages; investigate the ‘risk’ level more closely and see if removing outliers and reducing the number of risk levels improves the result; verifying whether upsampling vs downsampling for the class imbalance is best; removing all NAN values vs making assumptions; trying other methods besides GridSearchCV for tuning models; or spending more time for correlation are all valid reasons to revisit this project.

Reflections

I learned that I thoroughly enjoy data science through this project. Data cleaning is the most complex part and takes the most time, but the reward is akin to cleaning a very messy room to complete organization. Certainly, changes can always be made for improvement and you may go back and alter things, but it’s the backbone to a good result. Feature selection is a close second for me. It can vastly alter your result and must be done carefully – having a list of things to consider would have been beneficial and I would highly recommend putting together a simple reminder list of what to look for based on the model-type (classification, regression, etc).

Investigating parameters and seeing that your choices made a difference in the outcome is likely the most rewarding and most challenging for me. Each run of GridSearch took a significant amount of time, so much so that at one point, I had my laptop running one, and my desktop running two instances at the same time just to be able to get results. Knowing that the time and effort had an effect is a great relief.

The other part that was most challenging is not having someone there to remind you of key items or assure you that you're doing something correctly. Something that I deeply regret not being able to do is going back to feature selection after I realized that I forgot to take class imbalance into consideration. With practice and experience, this feeling of low confidence will dissipate, but it was challenging to overcome. I am proud to have gotten this far and am generally happy with the results.

A suggestion for someone doing a future project is that once you've selected your dataset, immediately start in on the literature review. It will give you ideas and direction of where to take your research, as well as narrow down possible machine learning models or methods you might want to try. Once you have that, review your notes and research what steps you need to take and what considerations you might need to have in mind when cleaning your data.

References

Chen, W., Xie, X., Wang, J., Pradhan, B., Hong, H., Bui, D. T., Duan, Z., & Ma, J. (2017). A comparative study of logistic model tree, random forest, and classification and regression tree models for spatial prediction of landslide susceptibility. *CATENA*, *151*, 147–160. https://doi.org/https://doi.org/10.1016/j.catena.2016.11.032

Frodella, W., Rosi, A., Spizzichino, D., Nocentini, M., Lombardi, L., Ciampalini, A., Vannocci, P., Ramboason, N., Margottini, C., Tofani, V., & Casagli, N. (2022). Integrated approach for landslide hazard assessment in the High City of Antananarivo, Madagascar (UNESCO tentative site). *Landslides*, *19*(11). https://doi.org/10.1007/s10346-022-01933-4

Global Climates and Seasons Precipitation. Journey North: Global Climate and the seasons. (n.d.). https://journeynorth.org/weather/ClimateTempPrecipAns2.html#:~:text=This%20tropical%20rain%20belt%20runs,sink%20back%20to%20the%20surface

Gorsevski, P. v., Gessler, P. E., & Jankowski, P. (2003). Integrating a fuzzy k-means classification and a Bayesian approach for spatial prediction of landslide hazard. *Journal of Geographical Systems*, *5*(3). https://doi.org/10.1007/s10109-003-0113-0

NASA Public Data. (n.d.). *Global landslide catalog export*. NASA. https://data.nasa.gov/Earth-Science/Global-Landslide-Catalog-Export/dd9e-wu2v/about_data

National Geographic Society. (n.d.). Equator. Education. https://education.nationalgeographic.org/resource/equator/

Terrestrial ecoregions of the world | publications | WWF. World Wildlife Federation. (n.d.). https://www.worldwildlife.org/publications/terrestrial-ecoregions-of-the-world

Van Den Eeckhaut, M., Vanwalleghem, T., Poesen, J., Govers, G., Verstraeten, G., & Vandekerckhove, L. (2006, June 30). *Prediction of landslide susceptibility using rare events logistic regression: A case-study in the Flemish Ardennes (Belgium)*. ScienceDirect. https://www.sciencedirect.com/science/article/pii/S0169555X05003788

Zhao, Y., Li, Y., Zhang, L., & Wang, Q. (2016). Groundwater level prediction of landslide based on classification and regression tree. *Geodesy and Geodynamics*, *7*(5). https://doi.org/10.1016/j.geog.2016.07.005

Appendix

Source code begins on the next page.

Data Cleaning Source Code

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import geopy
import geocoder
from geopy.geocoders import Nominatim
from geopy.extra.rate_limiter import RateLimiter
from geopy.geocoders import get_geocoder_for_service
get_geocoder_for_service("nominatim")
import datetime
import geopandas as gpd
from shapely.geometry import Point
from geopy.extra.rate_limiter import RateLimiter
import osmnx as ox
import seaborn as sns
```

Check for and fill in NANs wherever possible

In [19]:
```
path = "C:/Users/ashkl/Documents/Loyola/Capstone/Global_Landslide_Catalog.csv"
df = pd.read_csv(path)
df.head
```

Out[19]:
```
<bound method NDFrame.head of              source_name
source_link  \
0                    AGU  https://blogs.agu.org/landslideblog/2008/10/14...
1              Oregonian  http://www.oregonlive.com/news/index.ssf/2009/...
2               CBS News  https://www.cbsnews.com/news/dozens-missing-af...
3                Reuters  https://in.reuters.com/article/idINIndia-41450...
4            The Freeman  http://www.philstar.com/cebu-news/621414/lands...
...                  ...                                                ...
11028   The Jakarta Post  http://www.thejakartapost.com/news/2017/04/02/...
11029    Greater Kashmir  http://www.greaterkashmir.com/news/jammu/lands...
11030          NBC Daily  http://www.nbcdaily.com/separate-landslides-ki...
11031  AGU Landslide Blog  http://blogs.agu.org/landslideblog/2017/05/02/...
11032  The Times of India  https://timesofindia.indiatimes.com/city/hyder...

       event_id              event_date  event_time  \
0           684  08/01/2008 12:00:00 AM         NaN
1           956  01/02/2009 02:00:00 AM         NaN
2           973  01/19/2007 12:00:00 AM         NaN
3          1067  07/31/2009 12:00:00 AM         NaN
4          2603  10/16/2010 12:00:00 PM         NaN
...         ...                     ...         ...
11028     11109  04/01/2017 01:34:00 PM         NaN
11029     10845  03/25/2017 05:32:00 PM         NaN
11030     10973  12/15/2016 05:00:00 AM         NaN
11031     10901  04/29/2017 07:03:00 PM         NaN
11032     10949  03/13/2017 02:32:00 PM         NaN

                                          event_title  \
0          Sigou Village, Loufan County, Shanxi Province
1                                     Lake Oswego, Oregon
2      San Ramon district, 195 miles northeast of the...
3                                       Dailekh district
4                        sitio Bakilid in barangay Lahug
...                                                 ...
11028                        Major landslide in Banaran
11029                           Barnari Sigdi Landslide
11030                     Landslide at Pub Sarania Hill
11031                    Mayor landslide at Ayu village
11032  Kondapur Commercial Complex Construction Mudslide

                                       event_description  \
0      occurred early in morning, 11 villagers buried...
1      Hours of heavy rain are to blame for an overni...
2      (CBS/AP) At least 10 people died and as many a...
3      One person was killed in Dailekh district, pol...
4      Another landslide in sitio Bakilid in barangay...
...                                                  ...
11028  Landslide exacerbated by deforestation and bad...
11029  Two teenage girls died after they were buried ...
11030  An octogenarian was killed when a sudden lands...
11031  Landslide triggered by heavy rainfall buried 1...
11032  A mudslide at an under-construction commercial...

                                    location_description location_accuracy  \
0          Sigou Village, Loufan County, Shanxi Province           unknown
1                                     Lake Oswego, Oregon              5km
2      San Ramon district, 195 miles northeast of the...              10km
3                                       Dailekh district           unknown
4                        sitio Bakilid in barangay Lahug              5km
...                                                  ...               ...
```

```
11028          Banaran, Ponorogo, Jawa Timur, Indonesia              5km
11029  Barnari Sigdi area, Tehsil Mughalmaidan, Kisht...             5km
11030          Pub Sarania Hill, Guwahati, Assam, India              1km
11031                      Ayu, Ozgon, Osh, Kyrgyzstan               1km
11032                 Hyderabad, Rangareddy, Telangana               1km

       landslide_category  ... country_code admin_division_name  \
0               landslide  ...           CN             Shaanxi
1                mudslide  ...           US              Oregon
2               landslide  ...           PE               Junín
3               landslide  ...           NP         Mid Western
4               landslide  ...           PH     Central Visayas
...                   ...  ...          ...                 ...
11028           landslide  ...          NaN                 NaN
11029           landslide  ...          NaN                 NaN
11030           landslide  ...          NaN                 NaN
11031  translational_slide  ...          NaN                 NaN
11032            mudslide  ...          NaN                 NaN

      admin_division_population  gazeteer_closest_point  gazeteer_distance  \
0                           0.0                Jingyang           41.02145
1                       36619.0             Lake Oswego            0.60342
2                       14708.0               San Ramón            0.85548
3                       20908.0                 Dailekh            0.75395
4                      798634.0               Cebu City            2.02204
...                         ...                     ...                ...
11028                      NaN                     NaN                NaN
11029                      NaN                     NaN                NaN
11030                      NaN                     NaN                NaN
11031                      NaN                     NaN                NaN
11032                      NaN                     NaN                NaN

              submitted_date            created_date        last_edited_date  \
0      04/01/2014 12:00:00 AM  11/20/2017 03:17:00 PM  02/15/2018 03:51:00 PM
1      04/01/2014 12:00:00 AM  11/20/2017 03:17:00 PM  02/15/2018 03:51:00 PM
2      04/01/2014 12:00:00 AM  11/20/2017 03:17:00 PM  02/15/2018 03:51:00 PM
3      04/01/2014 12:00:00 AM  11/20/2017 03:17:00 PM  02/15/2018 03:51:00 PM
4      04/01/2014 12:00:00 AM  11/20/2017 03:17:00 PM  02/15/2018 03:51:00 PM
...                       ...                     ...                    ...
11028  07/28/2017 01:34:00 PM  12/19/2017 09:42:00 PM  02/15/2018 03:51:00 PM
11029  09/21/2017 05:32:00 PM  12/05/2017 06:45:00 PM  02/15/2018 03:51:00 PM
11030  07/26/2017 01:22:00 PM  12/08/2017 08:37:00 PM  02/15/2018 03:51:00 PM
11031  07/14/2017 07:03:00 PM  12/07/2017 09:19:00 PM  02/15/2018 03:51:00 PM
11032  10/05/2017 02:32:00 PM  12/08/2017 07:57:00 PM  02/15/2018 03:51:00 PM

        longitude   latitude
0      107.450000  32.562500
1     -122.663000  45.420000
2      -75.358700 -11.129500
3       81.708000  28.837800
4      123.897800  10.333600
...           ...        ...
11028  111.679944  -7.853409
11029   75.680611  33.403080
11030   91.772042  26.181606
11031   73.472379  40.886395
11032   78.356505  17.465630

[11033 rows x 31 columns]>
```

In [3]:
```
df.shape
```

Out[3]:
```
(11033, 31)
```

In [3]:
```
df.columns
```

Out[3]:
```
Index(['source_name', 'source_link', 'event_id', 'event_date', 'event_time',
       'event_title', 'event_description', 'location_description',
       'location_accuracy', 'landslide_category', 'landslide_trigger',
       'landslide_size', 'landslide_setting', 'fatality_count', 'injury_count',
       'storm_name', 'photo_link', 'notes', 'event_import_source',
       'event_import_id', 'country_name', 'country_code',
       'admin_division_name', 'admin_division_population',
       'gazeteer_closest_point', 'gazeteer_distance', 'submitted_date',
       'created_date', 'last_edited_date', 'longitude', 'latitude'],
      dtype='object')
```

In [5]:
```
#drop any that are not needed
df.drop(['photo_link', 'source_link', 'storm_name', 'submitted_date', 'submitted_date'
         'event_time', 'source_name', 'notes', 'event_import_id', 'event_import_source
         'event_description', 'location_description', 'admin_division_name','gazeteer_
         'location_accuracy'], axis=1, inplace=True)
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
event_date                    0
event_title                   0
landslide_category            1
landslide_trigger            23
landslide_size                9
landslide_setting            69
fatality_count             1385
injury_count               5674
country_name               1562
admin_division_population  1562
gazeteer_closest_point     1563
longitude                     0
latitude                      0
dtype: int64
```

In [7]:
```
df.shape
```

Out[7]:
```
(11033, 13)
```

In [10]:
```
#combine lat and long to get a column we can use for reverse lookup of contries
df["coordinates"] = df["latitude"].apply(str)+ ", " +df["longitude"].apply(str)
```

In [11]:
```
#create a subset of only those that have no country name
dfCountry = pd.DataFrame(df.loc[df['country_name'].isna()])
```

In [13]:
```
#Reverse lookup of country name using the coorinates
geolocator = Nominatim(user_agent="AK", timeout= 10)
rgeocode = RateLimiter(lambda coord: geolocator.reverse(coord, language="en"), min_del

dfCountry["country_name"] = dfCountry["coordinates"].apply(lambda coord: rgeocode(coor

# print(dfCountry.head(10))
```

In [14]:
```
#do the same thing for the closest point (to the landslide)
dfnas = pd.DataFrame(df.loc[df['gazeteer_closest_point'].isna()])
dfnas["gazeteer_closest_point"] = dfnas["coordinates"].apply(lambda coord: rgeocode(co
```

In [15]:
```
#fill in the original dataset with the new information for closest point
df['gazeteer_closest_point'] = df['gazeteer_closest_point'].fillna(dfnas['gazeteer_clo
df.isna().sum()
```

Out[15]:
```
event_date                    0
event_title                   0
landslide_category            1
landslide_trigger            23
landslide_size                9
landslide_setting            69
fatality_count             1385
injury_count               5674
country_name               1562
admin_division_population  1562
gazeteer_closest_point     1207
longitude                     0
latitude                      0
coordinates                   0
dtype: int64
```

In [16]:
```
#fill in the original dataset with the new information for county name
df['country_name'] = df['country_name'].fillna(dfCountry['country_name'])
df.head
```

Out[16]:

```
<bound method NDFrame.head of                   event_date  \
0      08/01/2008 12:00:00 AM
1      01/02/2009 02:00:00 AM
2      01/19/2007 12:00:00 AM
3      07/31/2009 12:00:00 AM
4      10/16/2010 12:00:00 PM
...                       ...
11028  04/01/2017 01:34:00 PM
11029  03/25/2017 05:32:00 PM
11030  12/15/2016 05:00:00 AM
11031  04/29/2017 07:03:00 PM
11032  03/13/2017 02:32:00 PM

                                           event_title   landslide_category  \
0          Sigou Village, Loufan County, Shanxi Province            landslide
1                                    Lake Oswego, Oregon             mudslide
2      San Ramon district, 195 miles northeast of the...            landslide
3                                       Dailekh district            landslide
4                         sitio Bakilid in barangay Lahug            landslide
...                                                   ...                  ...
11028                         Major landslide in Banaran            landslide
11029                            Barnari Sigdi Landslide            landslide
11030                      Landslide at Pub Sarania Hill            landslide
11031                     Mayor landslide at Ayu village  translational_slide
11032  Kondapur Commercial Complex Construction Mudslide             mudslide

      landslide_trigger landslide_size landslide_setting  fatality_count  \
0                  rain          large              mine            11.0
1              downpour          small           unknown             0.0
2              downpour          large           unknown            10.0
3               monsoon         medium           unknown             1.0
4      tropical_cyclone         medium           unknown             0.0
...                 ...            ...               ...             ...
11028              rain         medium     natural_slope            27.0
11029             other          small     natural_slope             2.0
11030           unknown          small             urban             1.0
11031          downpour          large     natural_slope            24.0
11032      construction          small             urban             2.0

       injury_count   country_name  admin_division_population  \
0               NaN          China                        0.0
1               NaN  United States                    36619.0
2               NaN           Peru                    14708.0
3               NaN          Nepal                    20908.0
4               NaN    Philippines                   798634.0
...             ...            ...                        ...
11028           0.0      Indonesia                        NaN
11029           0.0          India                        NaN
11030           0.0          India                        NaN
11031           NaN     Kyrgyzstan                        NaN
11032           0.0          India                        NaN

      gazeteer_closest_point   longitude   latitude               coordinates
0                   Jingyang  107.450000  32.562500           32.5625, 107.45
1                Lake Oswego -122.663000  45.420000           45.42, -122.663
2                  San Ramón  -75.358700 -11.129500        -11.1295, -75.3587
3                    Dailekh   81.708000  28.837800           28.8378, 81.708
4                  Cebu City  123.897800  10.333600          10.3336, 123.8978
...                      ...         ...        ...                        ...
11028                   None  111.679944  -7.853409  -7.853409423, 111.6799444
```

```
11029                  None   75.680611  33.403080   33.40307977, 75.68061125
11030               Guwahati   91.772042  26.181606      26.181606, 91.772042
11031                  None   73.472379  40.886395   40.88639497, 73.47237853
11032             Hyderabad   78.356505  17.465630   17.46562972, 78.35650524

[11033 rows x 14 columns]>
```

In [17]:
```
df.isna().sum()
```

Out[17]:
```
event_date                    0
event_title                   0
landslide_category            1
landslide_trigger            23
landslide_size                9
landslide_setting            69
fatality_count              1385
injury_count                5674
country_name                   0
admin_division_population   1562
gazeteer_closest_point      1207
longitude                      0
latitude                       0
coordinates                    0
dtype: int64
```

In [18]:
```
#convert date to date format
df['event_date'] = pd.to_datetime(df['event_date'], format='%m/%d/%Y %I:%M:%S %p')
```

In [141…
```
#exporting to csv so i don't have to run the above again
df.to_csv(r'C:\Users\ashkl\Documents\Loyola\Capstone\dfCountries.csv', index=False)
```

In [3]:
```
#re-import for quick access as needed
path = "C:/Users/ashkl/Documents/Loyola/Capstone/dfCountries.csv"
df2 = pd.read_csv(path)
df2.head(4)
```

Out[3]:

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting	fatali
0	2008-08-01 00:00:00	Sigou Village, Loufan County, Shanxi Province	landslide	rain	large	mine	
1	2009-01-02 02:00:00	Lake Oswego, Oregon	mudslide	downpour	small	unknown	
2	2007-01-19 00:00:00	San Ramon district, 195 miles northeast of the...	landslide	downpour	large	unknown	
3	2009-07-31 00:00:00	Dailekh district	landslide	monsoon	medium	unknown	

In [4]:
```
#redo dateformat if re-importing
df2['event_date'] = pd.to_datetime(df2['event_date'])
print(df2.dtypes)
```

```
event_date                  datetime64[ns]
event_title                         object
landslide_category                  object
landslide_trigger                   object
landslide_size                      object
landslide_setting                   object
fatality_count                     float64
injury_count                       float64
country_name                        object
admin_division_population          float64
gazeteer_closest_point              object
longitude                          float64
latitude                           float64
coordinates                         object
dtype: object
```

Adding new variables

Season variable

In [5]:
```
#create function to obtain seasons. Keep in mind that near the equator, there are only
def get_season(latitude, date=None):

    if date is None:
        date = 'unknown'

    month = date.month
    northern_hemisphere = latitude > 23.5
    southern_hemisphere = latitude < -23.5
    eq_north = latitude >= 0 and latitude <= 23.5
    eq_south = latitude < 0 and latitude <= -23.5

    if eq_north:
        if (month >= 4 or month < 10):
            season = "Equitorial Wet"
        else:
            season = "Equitorial Dry"
    if eq_south:
        if (month >= 10 or month < 4):
            season = "Equitorial Wet"
        else:
            season = "Equitorial Dry"
    elif northern_hemisphere:
        if (month == 12 or month < 3):
            season = "Winter"
        elif (month < 6):
            season = "Spring"
        elif (month < 9):
            season = "Summer"
        else:
            season = "Autumn"
    else:  # Southern Hemisphere
        if (month == 12 or month < 3):
```

```
            season = "Summer"
        elif (month < 6):
            season = "Autumn"
        elif (month < 9):
            season = "Winter"
        else:
            season = "Spring"

    return season
```

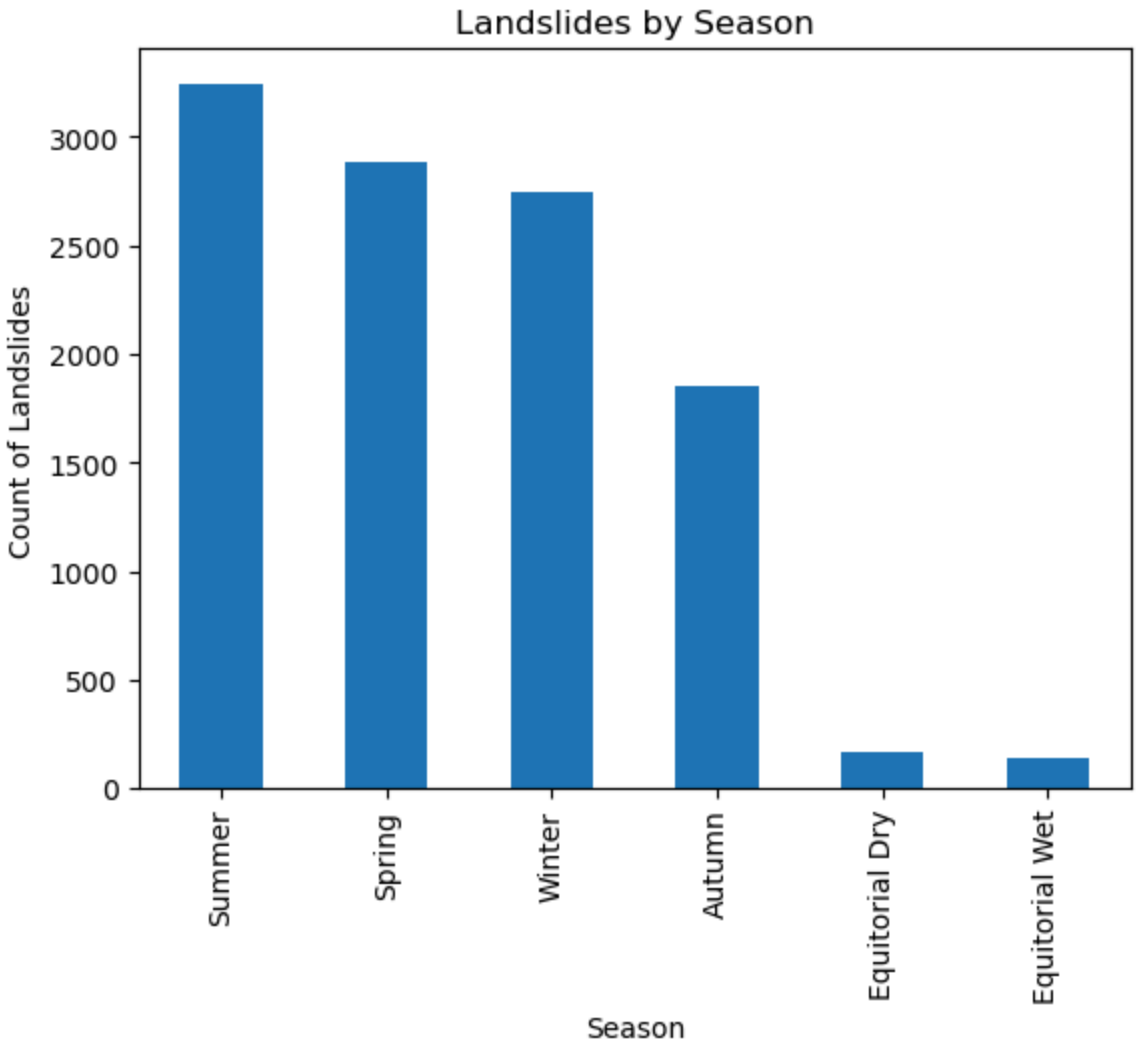
In [6]:
```
#Create season column!
df2['season'] = df2.apply(lambda row: get_season(row['latitude'], row['event_date']),
```

In [146…
```
df2.head(4)
```

Out[146]:

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting	fatali
0	2008-08-01 00:00:00	Sigou Village, Loufan County, Shanxi Province	landslide	rain	large	mine	
1	2009-01-02 02:00:00	Lake Oswego, Oregon	mudslide	downpour	small	unknown	
2	2007-01-19 00:00:00	San Ramon district, 195 miles northeast of the...	landslide	downpour	large	unknown	
3	2009-07-31 00:00:00	Dailekh district	landslide	monsoon	medium	unknown	

In [17]:
```
#first look at seasons
ax = df2.season.value_counts().plot(kind='bar')
ax.set_title('Landslides by Season')
ax.set_xlabel('Season')
ax.set_ylabel('Count of Landslides');
```



Biome variable

In [148…
```
# Load the shapefile for getting biome data
biome_shapefile = "C:/Users/ashkl/Downloads/6kcchn7e3u_official_teow/official/wwf_terr
biomes = gpd.read_file(biome_shapefile)
```

In [149…
```
def get_biome(latitude, longitude):

    point = Point(longitude, latitude)

    # Find the biome containing the point
    biome = biomes[biomes.contains(point)]

    return biome["ECO_NAME"].values[0] if not biome.empty else None
```

In [150…
```
df2.loc[:, 'biome'] = df2.apply(lambda row: get_biome(row['latitude'], row['longitude'
```

In [151…
```
df2.head(5)
```

Out[151]:

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting	fatali
0	2008-08-01 00:00:00	Sigou Village, Loufan County, Shanxi Province	landslide	rain	large	mine	
1	2009-01-02 02:00:00	Lake Oswego, Oregon	mudslide	downpour	small	unknown	
2	2007-01-19 00:00:00	San Ramon district, 195 miles northeast of the...	landslide	downpour	large	unknown	
3	2009-07-31 00:00:00	Dailekh district	landslide	monsoon	medium	unknown	
4	2010-10-16 12:00:00	sitio Bakilid in barangay Lahug	landslide	tropical_cyclone	medium	unknown	

Combine injury and fatality count

In [ ]:
```
df2.dropna(subset=['injury_count', 'fatality_count'], how= 'all', inplace=True)
```

In [153...
```
df2['injury_count'] = df2['injury_count'].fillna(0)
```

In [154...
```
df2['fatality_count'] = df2['fatality_count'].fillna(0)
```

In [155...
```
df2['injury&fatality_count'] = df2['injury_count'] + df2['fatality_count']
```

In [156...
```
df2.isna().sum()
```

Out[156]:

```
event_date                    0
event_title                   0
landslide_category            1
landslide_trigger            22
landslide_size                7
landslide_setting            65
fatality_count                0
injury_count                  0
country_name                  0
admin_division_population  1517
gazeteer_closest_point     1176
longitude                     0
latitude                      0
coordinates                   0
season                        0
biome                       278
injury&fatality_count         0
dtype: int64
```

In [157…

```
df2.head(30)
```

Out[157]:

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting
0	2008-08-01 00:00:00	Sigou Village, Loufan County, Shanxi Province	landslide	rain	large	mine
1	2009-01-02 02:00:00	Lake Oswego, Oregon	mudslide	downpour	small	unknown
2	2007-01-19 00:00:00	San Ramon district, 195 miles northeast of the...	landslide	downpour	large	unknown
3	2009-07-31 00:00:00	Dailekh district	landslide	monsoon	medium	unknown
4	2010-10-16 12:00:00	sitio Bakilid in barangay Lahug	landslide	tropical_cyclone	medium	unknown
5	2012-02-16 00:00:00	Paguite, Abuyog, Leyte	landslide	downpour	medium	unknown
6	2012-03-30 00:00:00	Pend Oreille County, State Route 20 near Usk, OR	mudslide	downpour	small	unknown
7	2007-09-02 00:00:00	3 killed in Acapulco	complex	tropical_cyclone	medium	unknown
9	2008-11-01 00:00:00	Lincang City, Yunnan, Yunnan-Tibet No. 214 hig...	complex	downpour	medium	unknown
10	2008-11-01 00:00:00	Kunming, Yunnan	complex	downpour	medium	unknown
11	2014-12-21 08:10:00	US 30, milepost 29	landslide	unknown	unknown	above_road
12	2009-01-01 22:24:00	South side of Highway 26 between Cherryville H...	mudslide	rain	small	unknown
13	2017-07-03 14:33:00	Landslides block roads	landslide	downpour	small	above_road
14	2009-01-02 20:30:00	Gadgaron village of	landslide	downpour	medium	unknown

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting
		Matnog town, Sorsogon				
15	2009-01-03 00:00:00	Suburban Portland, Oregon.	mudslide	rain	small	unknown
16	2009-01-07 01:41:00	Wildcat Run Road off Dogwood Drive in Maggie V...	landslide	rain	small	unknown
17	2009-01-07 00:00:00	Rochester along the Cowlitz and Lewis rivers, ...	landslide	rain	small	unknown
18	2009-01-07 00:00:00	Sixth Place South in Normandy Park , Washington	mudslide	rain	small	unknown
19	2009-01-07 00:00:00	Concrete, 70 miles northeast of Seattle,Washin...	landslide	downpour	medium	unknown
20	2009-01-07 00:00:00	Acme area, Whatcom County, Washington	mudslide	rain	small	unknown
21	2009-01-08 00:00:00	Burcham Street, Kelso, Washington	mudslide	unknown	small	unknown
22	2009-01-08 01:00:00	9700 block of Rainier Avenue South, Seatle, Wa...	complex	rain	small	unknown
23	2009-01-08 06:00:00	Mount Baker Highway at about milepost 16, Wash...	mudslide	rain	small	unknown
24	2009-01-13 00:00:00	Chehna in Jijel province, some 360 km (225 mil...	landslide	downpour	medium	unknown
25	2009-01-13 08:50:00	SR 522, near Bothell, Washington	mudslide	rain	small	unknown
26	2009-01-16 13:00:00	Sekolah Kebangsaan Sungai Arip, Mukah about 60...	landslide	rain	medium	unknown

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting
27	2009-01-19 23:00:00	Buwung Mas Sekotong, about 1,060km east of the...	landslide	rain	medium	unknown
28	2009-08-13 00:00:00	Freetown	landslide	downpour	medium	unknown
29	2009-01-22 00:00:00	Muara District	landslide	downpour	medium	unknown
30	2009-01-22 00:00:00	Mexico City	landslide	downpour	medium	unknown

In [158…
```
df2['biome'].nunique()
```

Out[158]:
```
449
```

Fill in remaining NAN data if possible

In [159…
```
tempCat = pd.DataFrame(df2.loc[df2['landslide_category'].isna()])
tempCat.head(1)
```

Out[159]:

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting	t
10102	2016-09-11 12:55:00	Landslide kills Army colonel, wife on Badrinat...	NaN	NaN	large	above_road	

In [160…
```
tempTrig = pd.DataFrame(df2.loc[df2['landslide_trigger'].isna()])
tempTrig.head(22)
```

Out[160]:

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting
138	2017-07-03 18:04:00	Landslide in Bhojpur	landslide	NaN	medium	NaN
10015	2016-09-11 12:55:00	Landslide on State Highway 6	landslide	NaN	large	above_road
10084	2016-12-15 14:22:00	Rockslide on HWY 299	rock_fall	NaN	large	NaN
10102	2016-09-11 12:55:00	Landslide kills Army colonel, wife on Badrinat...	NaN	NaN	large	above_road
10166	2016-09-17 14:54:00	Mudslide suspends train operations in SW China	mudslide	NaN	medium	above_road
10202	2016-09-12 12:55:00	Rasuwa landslide obstructs vehicular movement,...	landslide	NaN	medium	above_road
10305	2016-11-10 14:00:00	Landslide in St-Luc-de-Vincennes	landslide	NaN	medium	above_river
10307	2016-11-09 15:00:00	Rockslide on Tyee Road	rock_fall	NaN	medium	above_road
10318	2016-11-04 14:00:00	Rockfall on Kault Hill	rock_fall	NaN	small	above_road
10384	2016-09-20 14:24:00	Landslide in Shantipur	landslide	NaN	medium	natural_slope
10385	2016-09-21 14:24:00	Landslide in Dhankuta	landslide	NaN	small	NaN
10525	2016-09-16 14:54:00	Train Derails after hitting landslide	landslide	NaN	small	above_road
10560	2017-07-10 12:55:00	Cars crushed by deadly landslide on Chongey ro...	rock_fall	NaN	medium	NaN

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting
10577	2016-09-17 14:24:00	Landslide in Yuanmou	landslide	NaN	medium	natural_slope
10596	2016-11-09 05:00:00	Landslide near Zuvand Village	landslide	NaN	medium	NaN
10598	2017-02-16 19:25:00	Castelnuovo di Campli Landslide Evacuates Homes	landslide	NaN	NaN	below_road
10767	2017-04-02 12:57:00	Landslide on Doda-Kishtwar Highway	landslide	NaN	medium	above_road
10855	2016-09-14 14:54:00	Mudslide on Alaska Railroad tracks	mudslide	NaN	medium	above_road
10869	2016-09-17 14:54:00	Landslide in Mizoram	landslide	NaN	NaN	NaN
10934	2016-11-03 14:00:00	Rockslide on Malahat Drive	rock_fall	NaN	small	above_road
10980	2017-07-12 14:00:00	Landslide in Doda	landslide	NaN	medium	natural_slope
10995	2016-11-06 15:00:00	Mudslide near Bayview	mudslide	NaN	small	bluff

In [161…

```
tempSize = pd.DataFrame(df2.loc[df2['landslide_size'].isna()])
tempSize.head(22)
```

Out[161]:

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting
10078	2016-10-31 14:24:00	Landslide at Mt. Hagen	landslide	rain	NaN	NaN
10211	2016-09-12 12:55:00	Nepal-China Gyirong highway closed after lands...	landslide	downpour	NaN	above_road
10257	2016-09-12 12:55:00	Landslide blocks Siddhicharan highway	landslide	continuous_rain	NaN	above_road
10389	2016-10-30 14:24:00	Mudslide on Hwy 49 in Mokelumne Hill	mudslide	rain	NaN	above_road
10422	2016-09-14 14:54:00	Landslide on the Great Ocean Road	landslide	flooding	NaN	NaN
10598	2017-02-16 19:25:00	Castelnuovo di Campli Landslide Evacuates Homes	landslide	NaN	NaN	below_road
10869	2016-09-17 14:54:00	Landslide in Mizoram	landslide	NaN	NaN	NaN

In [162…

```
#craete values as needed
values = {"landslide_category": 'landslide', "landslide_trigger": 'unknown', "landslid
          "landslide_setting": 'unknown', 'admin_division_population': 0, 'gazeteer_cl
         'biome': 'unknown'}
df2.fillna(value=values, inplace=True)
```

In [163…

```
df2.isna().sum()
```

Out[163]:
```
event_date                  0
event_title                 0
landslide_category          0
landslide_trigger           0
landslide_size              0
landslide_setting           0
fatality_count              0
injury_count                0
country_name                0
admin_division_population   0
gazeteer_closest_point      0
longitude                   0
latitude                    0
coordinates                 0
season                      0
biome                       0
injury&fatality_count       0
dtype: int64
```

In [164…
```
df2.shape
```

Out[164]:
```
(9656, 17)
```

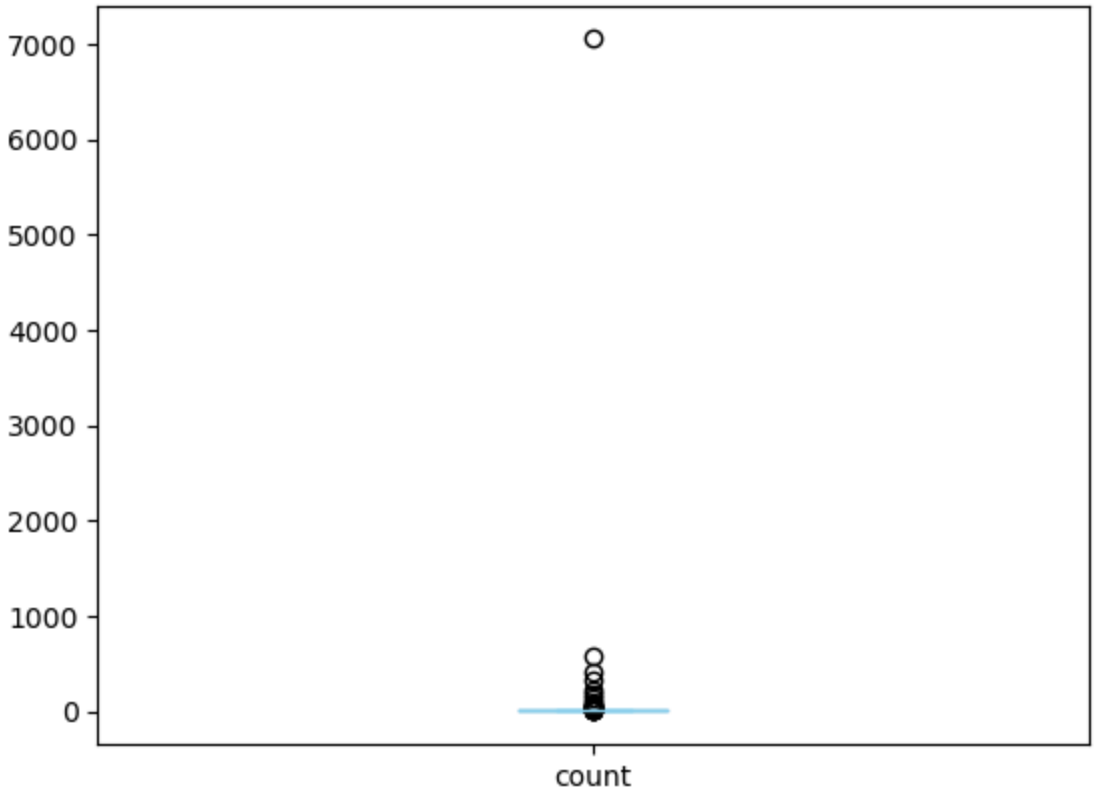
In [165…
```
print(df2.dtypes)
```

```
event_date                  datetime64[ns]
event_title                         object
landslide_category                  object
landslide_trigger                   object
landslide_size                      object
landslide_setting                   object
fatality_count                     float64
injury_count                       float64
country_name                        object
admin_division_population          float64
gazeteer_closest_point              object
longitude                          float64
latitude                           float64
coordinates                         object
season                              object
biome                               object
injury&fatality_count              float64
dtype: object
```

Check injury and fatality count and create risk variable

In [166…
```
counts = df2['injury&fatality_count'].value_counts()

# Plot value counts
counts.plot(kind='box', color='skyblue')
```

Out[166]:
```
<Axes: >
```



In [167…

```
x = df2.sort_values('injury&fatality_count', ascending = False)
x
```

Out[167]:

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting
5694	2013-06-16 19:30:00	Kedarnath (Hindu Shrine), Near Kedarnath Dome ...	debris_flow	downpour	very_large	unknown
5927	2014-05-02 00:00:00	Abi Barik Village, Badakhshan	landslide	continuous_rain	very_large	natural_slope
8173	2010-08-07 23:00:00	Zhugqu urban center and Beijie village, Bailon...	landslide	downpour	very_large	unknown
10190	2017-03-25 17:32:00	Deadly Mocoa Landslide	debris_flow	downpour	catastrophic	above_river
9097	2015-08-02 00:00:00	Guatamala	landslide	downpour	large	unknown
...	...	...	...	...	...	...
5362	2013-06-18 22:00:00	Tamti, Jumla District, Mid-Western Region	landslide	downpour	medium	unknown
5361	2014-03-10 00:00:00	Jammu and Kashmir	landslide	snowfall_snowmelt	medium	above_road
1354	2010-10-18 00:00:00	Labey(?), Bokod, Panganan(?), Benguet province	landslide	tropical_cyclone	medium	unknown
5359	2011-07-05 02:30:00	Intersection of state Route 92 and state Route...	landslide	downpour	medium	unknown
6729	2014-11-05 00:00:00	A47 outside Lyon	mudslide	downpour	medium	above_road

9656 rows × 17 columns

In [168…

```
import numpy as np

# Define conditions
conditions = [
    (df2['injury&fatality_count'] <= 5),
    (df2['injury&fatality_count'] > 5) & (df2['injury&fatality_count'] <= 20),
    (df2['injury&fatality_count'] > 20) & (df2['injury&fatality_count'] <= 200),
    (df2['injury&fatality_count'] > 200)
]

# Define category choices
choices = ['low', 'moderate', 'high', 'catastrophic']

# Apply the conditions
df2['risk'] = np.select(conditions, choices, default='unknown')
```

In [169…

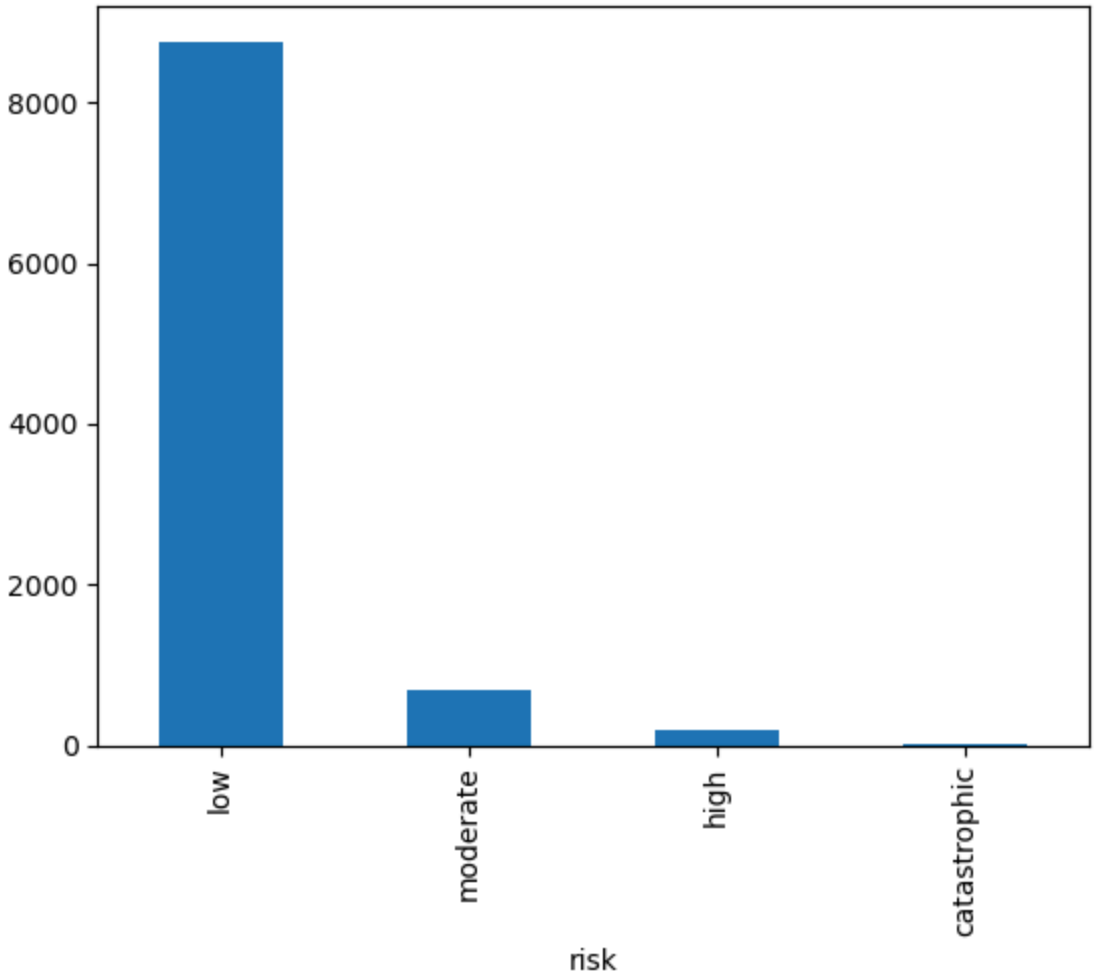
```
df2.head(5)
```

Out[169]:

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting	fatali
0	2008-08-01 00:00:00	Sigou Village, Loufan County, Shanxi Province	landslide	rain	large	mine	
1	2009-01-02 02:00:00	Lake Oswego, Oregon	mudslide	downpour	small	unknown	
2	2007-01-19 00:00:00	San Ramon district, 195 miles northeast of the...	landslide	downpour	large	unknown	
3	2009-07-31 00:00:00	Dailekh district	landslide	monsoon	medium	unknown	
4	2010-10-16 12:00:00	sitio Bakilid in barangay Lahug	landslide	tropical_cyclone	medium	unknown	

In [170…

```
df2.risk.value_counts().plot(kind='bar');
```



create month and Year variables

In [171…

```
df2['year'] = df2['event_date'].dt.year
df2['month'] = df2['event_date'].dt.month
df2.head(5)
```

Out[171]:

	event_date	event_title	landslide_category	landslide_trigger	landslide_size	landslide_setting	fatali
0	2008-08-01 00:00:00	Sigou Village, Loufan County, Shanxi Province	landslide	rain	large	mine	
1	2009-01-02 02:00:00	Lake Oswego, Oregon	mudslide	downpour	small	unknown	
2	2007-01-19 00:00:00	San Ramon district, 195 miles northeast of the...	landslide	downpour	large	unknown	
3	2009-07-31 00:00:00	Dailekh district	landslide	monsoon	medium	unknown	
4	2010-10-16 12:00:00	sitio Bakilid in barangay Lahug	landslide	tropical_cyclone	medium	unknown	

In [172…
```
#drop unneeded features
df2.drop(['coordinates', 'event_title', 'event_date'], axis=1, inplace=True)
```

In [173…
```
#exporting clean version to csv
df2.to_csv(r'C:\Users\ashkl\Documents\Loyola\Capstone\landslides_clean.csv', index=Fal
```

In [127…
```
df2.columns
```

Out[127]:
```
Index(['landslide_category', 'landslide_trigger', 'landslide_size',
       'landslide_setting', 'fatality_count', 'injury_count', 'country_name',
       'admin_division_population', 'gazeteer_closest_point', 'longitude',
       'latitude', 'season', 'biome', 'injury&fatality_count', 'risk', 'year',
       'month'],
      dtype='object')
```

Feature Selection Source Code

In [ ]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.svm import SVC
```

In [3]:
```
path = "C:/Users/ashkl/Documents/Loyola/Capstone/landslides_clean.csv"
df_clean = pd.read_csv(path)
```

In [4]:
```
df_clean.drop(['injury&fatality_count', 'fatality_count', 'injury_count', 'latitude',
```

In [5]:
```
#change risk levels to numerical levels
risk_mapping = {'low': 1, 'moderate':2, 'high':3, 'catastrophic':4}
df_clean['risk'] = df_clean['risk'].map(risk_mapping)
```

In [6]:
```
df_clean.shape
```

Out[6]:
```
(9656, 12)
```

In [7]:
```
dfsvm = pd.get_dummies(df_clean, dtype= int, drop_first=True)
dfsvm.shape
```

Out[7]:
```
(9656, 4715)
```

In [ ]:
```
X = df.drop(['risk'], axis=1)
y = df.risk
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

svc_model = SVC(kernel = 'linear', random_state= 1, decision_function_shape= 'ovr')
svc_model.fit(X_train_std, y_train)

print('Training accuracy:', svc_model.score(X_train_std, y_train))
print('Test accuracy:', svc_model.score(X_test_std, y_test))
```

Check for high correlations and remove >0.80

In [ ]:
```
corr = dfsvm.corr(method = 'pearson')
```

In [24]:
```
corr.head(5)
```

Out[24]:

	admin_division_population	risk	year	month	landslide_categc
admin_division_population	1.000000	0.046650	-0.103509	-0.000814	-
risk	0.046650	1.000000	-0.115840	0.030428	-
year	-0.103509	-0.115840	1.000000	-0.136251	
month	-0.000814	0.030428	-0.136251	1.000000	-
landslide_category_creep	-0.003576	-0.006741	0.007491	-0.004702	

5 rows × 4715 columns

In [59]:
```
corr = corr.replace(1.0, np.nan)
corrFiltered = corr[(corr > 0.9) | (corr < -0.9)].dropna(how='all', axis=0).dropna(how
```

In [7]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [65]:
```
corrFiltered
#chande mudslide to landslide in landslide category
```

Out[65]:

	country_name_Sierra Leone	country_name_South Korea	country_name_Taiwan	bic
country_name_Bangladesh	NaN	NaN	NaN	
country_name_Jamaica	NaN	NaN	NaN	
country_name_Sierra Leone	NaN	NaN	NaN	
country_name_South Korea	NaN	NaN	NaN	
country_name_Taiwan	NaN	NaN	NaN	
gazeteer_closest_point_Freetown	0.912824	NaN	NaN	
gazeteer_closest_point_Jayapura	NaN	NaN	NaN	
gazeteer_closest_point_Myitkyina	NaN	NaN	NaN	
biome_Central Korean deciduous forests	NaN	0.957377	NaN	
biome_Jamaican moist forests	NaN	NaN	NaN	
biome_Lower Gangetic Plains moist deciduous forests	NaN	NaN	NaN	
biome_Northern New Guinea montane rain forests	NaN	NaN	NaN	
biome_Northern Triangle subtropical forests	NaN	NaN	NaN	
biome_Taiwan subtropical evergreen forests	NaN	NaN	0.946568	

In [8]:
```
dfsvm.drop(['country_name_Bangladesh', 'country_name_Jamaica', 'country_name_Sierra Le
```

In [13]:
```
corr = dfsvm.corr(method = 'pearson')
corr = corr.replace(1.0, np.nan)
corrFiltered = corr[(corr > 0.8) | (corr < -0.8)].dropna(how='all', axis=0).dropna(how
```

In [9]:
```
dfsvm.drop(['country_name_Brazil', 'country_name_East Timor', 'country_name_Fiji', 'ga
```

In [18]:
```
corr = dfsvm.corr(method = 'pearson')
corr = corr.replace(1.0, np.nan)
corrFiltered = corr[(corr > 0.8) | (corr < -0.8)].dropna(how='all', axis=0).dropna(how
```

In [19]:
```
corrFiltered
```

Out[19]: —

In [10]:
```
dfsvm.to_csv(r'C:\Users\ashkl\Documents\Loyola\Capstone\dfClean.csv', index=False)
```

In [7]:
```
path = "C:/Users/ashkl/Documents/Loyola/Capstone/dfClean.csv"
dfsvm = pd.read_csv(path)
```

Run initial test

In [8]:
```
#split the data into a train and test set
X = dfsvm.drop(['risk'], axis=1)
y = dfsvm.risk
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=1, stratify=y)
```

In [9]:
```
#scale the data
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

In [10]:
```
svc_model = SVC(kernel = 'linear')
svc_model.fit(X_train_std, y_train)

print('Training accuracy:', svc_model.score(X_train_std, y_train))
print('Test accuracy:', svc_model.score(X_test_std, y_test))
```

```
Training accuracy: 0.9835774522858411
Test accuracy: 0.8463928201587849
```

Select Best Features

In [11]:
```
bestfeatures = SelectKBest(score_func=f_regression, k=10)
fit = bestfeatures.fit(X_train,y_train)

dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(dfsvm.columns)

featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Feature','Score']
print(featureScores.nlargest(40,'Score'))
```

	Feature	Score
33	landslide_trigger_volcano	924.687436
37	landslide_size_unknown	356.992734
73	country_name_Chile	346.155861
172	country_name_United Kingdom	319.757332
39	landslide_setting_above_river	195.743401
35	landslide_size_medium	173.171098
1	risk	94.665843
49	landslide_setting_retaining_wall	82.937060
4517	biome_Northern Thailand-Laos moist deciduous f...	80.854906
4687	biome_Yarlung Tsangpo arid steppe	75.037968
45	landslide_setting_engineered_slope	74.489456
132	country_name_Myanmar	68.130249
30	landslide_trigger_tropical_cyclone	60.018608
1065	gazeteer_closest_point_Dadeldhurā	53.590627
1411	gazeteer_closest_point_Ghumārwīn	53.590627
3997	gazeteer_closest_point_Welch	53.590627
4200	gazeteer_closest_point_Zhong'an	53.590627
140	country_name_Norway	49.665012
4431	biome_Jamaican moist forests	47.227708
177	country_name_Vietnam	46.415238
4553	biome_Qin Ling Mountains deciduous forests	45.868177
104	country_name_India	43.434576
36	landslide_size_small	42.792279
4552	biome_Pyrenees conifer and mixed forests	39.498329
123	country_name_Macedonia	37.744822
4315	biome_Central British Columbia Mountain forests	36.642567
4321	biome_Central Mexican matorral	35.356463
4527	biome_Northwestern thorn scrub forests	34.516282
4396	biome_Guinean forest-savanna mosaic	33.787192
98	country_name_Guernsey	33.020599
4539	biome_Pantanos de Centla	33.020599
17	landslide_trigger_continuous_rain	30.079591
12	landslide_category_rock_fall	30.011329
248	gazeteer_closest_point_Alpine	28.519431
883	gazeteer_closest_point_Chihuahua	28.519431
1605	gazeteer_closest_point_Hiroshima	28.519431
4453	biome_Luzon tropical pine forests	28.519431
29	landslide_trigger_snowfall_snowmelt	27.620954
74	country_name_China	26.806406
4269	biome_Bahia coastal forests	25.044019

In [12]:
```
# Remove kbest with scores less than 1 (689 features left)
filtered_df = featureScores[featureScores['Score'] > 20]
filtered_df.shape
```

Out[12]:
```
(129, 2)
```

In [14]:
```
#filter df for kscores >1
df_high_Kscore01 = dfsvm.iloc[:, filtered_df.index]

#split to train and test data
X = df_high_Kscore01.drop(['risk'], axis=1)
y = df_high_Kscore01.risk
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=1, stratify=y)

#scale the data:
sc.fit(X_train)
```

```
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

In [15]:
```
svc_model = SVC(kernel = 'linear')
svc_model.fit(X_train_std, y_train)

print('Training accuracy:', svc_model.score(X_train_std, y_train))
print('Test accuracy:', svc_model.score(X_test_std, y_test))
```

```
Training accuracy: 0.9094540612516645
Test accuracy: 0.90714532274767
```

In [16]:
```
coefficients = pd.DataFrame({'Variable': X_train.columns, 'Coefficient': svc_model.coe
coefficients
```

Out[16]:

	Variable	Coefficient
0	landslide_category_rock_fall	-3.813674e-06
1	landslide_trigger_continuous_rain	1.145044e-05
2	landslide_trigger_snowfall_snowmelt	3.730659e-06
3	landslide_trigger_tropical_cyclone	5.464325e-07
4	landslide_trigger_volcano	0.000000e+00
...	...	...
123	biome_South Central Rockies forests	6.968400e-06
124	biome_Southwest Iberian Mediterranean scleroph...	4.053799e-07
125	biome_Western Himalayan broadleaf forests	8.771862e-06
126	biome_Wyoming Basin shrub steppe	1.813975e-07
127	biome_Yarlung Tsangpo arid steppe	2.664535e-15

128 rows × 2 columns

In [17]:
```
coefficients.sort_values('Coefficient', ascending=False).head(10)
```

Out[17]:

	Variable	Coefficient
101	biome_Angolan scarp savanna and woodlands	0.000012
1	landslide_trigger_continuous_rain	0.000011
125	biome_Western Himalayan broadleaf forests	0.000009
63	gazeteer_closest_point_La Trinidad	0.000009
30	gazeteer_closest_point_Bellingham	0.000009
91	gazeteer_closest_point_Thị Trấn Mường Tè	0.000009
36	gazeteer_closest_point_Chihuahua	0.000008
6	landslide_size_small	0.000007
123	biome_South Central Rockies forests	0.000007
18	country_name_Nigeria	0.000007

In [18]:
```
coefficients.sort_values('Coefficient', ascending=True).head(10)
```

Out[18]:

	Variable	Coefficient
14	country_name_Guinea	-0.048640
115	biome_Madagascar subhumid forests	-0.034393
82	gazeteer_closest_point_Shangjin	-0.034390
42	gazeteer_closest_point_Datarpasang	-0.024325
96	gazeteer_closest_point_Xinjing	-0.024321
118	biome_Northern Thailand-Laos moist deciduous f...	-0.024318
83	gazeteer_closest_point_Shuangxi	-0.024317
38	gazeteer_closest_point_Commune Arcahaie	-0.024317
60	gazeteer_closest_point_Kahama	-0.024317
100	season_Summer	-0.000004

Analysis Source Code

In [1]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC, LinearSVC
from sklearn.feature_selection import SelectKBest, f_regression
import numpy as np
from sklearn.pipeline import make_pipeline
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix


pd.options.display.max_columns = None
pd.options.display.max_rows = None
```

In [2]:
```
path = "C:/Users/Kaelu/Documents/Loyola/DataScience Project/dfClean.csv"
df = pd.read_csv(path)
df.head(4)
```

Out[2]:

	admin_division_population	risk	year	month	landslide_category_creep	landslide_categc
0	0.0	2	2008	8	0	
1	36619.0	1	2009	1	0	
2	14708.0	2	2007	1	0	
3	20908.0	1	2009	7	0	

In [3]:
```
df.shape
```

Out[3]: (9656, 4691)

Class imbalance discovery and steps to overcome

In [5]:
```
print(df['risk'].value_counts())
```

```
risk
1    8761
2     679
3     196
4      20
Name: count, dtype: int64
```

In [6]:
```python
#upsample the 'risk' category to remove class imbalance
X = df.drop(['risk'], axis=1)
y = df.risk

# Initialize SMOTE and apply to the data
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Convert to df
upsampled_df = pd.DataFrame(X_resampled, columns=X.columns)
upsampled_df['risk'] = y_resampled

# verify it worked
print(upsampled_df['risk'].value_counts())
```

```
risk
2    8761
1    8761
3    8761
4    8761
Name: count, dtype: int64
C:\Users\Kaelu\AppData\Local\Temp\ipykernel_12884\2128993842.py:11: PerformanceWarni
ng: DataFrame is highly fragmented.  This is usually the result of calling `frame.in
sert` many times, which has poor performance.  Consider joining all columns at once
using pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe = fram
e.copy()`
  upsampled_df['risk'] = y_resampled
```

In [7]:
```python
upsampled_df.shape
```

Out[7]:
```
(35044, 4691)
```

In [9]:
```python
print(upsampled_df['risk'].value_counts())
```

```
risk
2    8761
1    8761
3    8761
4    8761
Name: count, dtype: int64
```

Re-run initial tests

In [11]:
```python
#split the data into a train and test set
X = upsampled_df.drop('risk', axis=1)
y = upsampled_df['risk']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=1, stratify=y)
```

```
#scale the data
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

In [ ]:
```
#Run Linear SVC model

svm_model = SVC(kernel='linear')
svm_model.fit(X_train_std, y_train)

print('Training accuracy:', svm_model.score(X_train_std, y_train))
print('Test accuracy:', svm_model.score(X_test_std, y_test))
```

Run Select Best Features again with balanced class

In [ ]:
```
bestfeatures = SelectKBest(score_func=f_regression, k=10)
fit = bestfeatures.fit(X_train_std,y_train)

dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(upsampled_df.columns)

featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Feature','Score']
print(featureScores.nlargest(40,'Score'))
```

In [ ]:
```
# Remove kbest with scores less than 1
filtered_df = featureScores[featureScores['Score'] > 1]
filtered_df.shape
```

In [ ]:
```
#filter df for kscores >1
df_high_Kscore01 = upsampled_df.iloc[:, filtered_df.index]
```

In [ ]:
```
selected_features =  filtered_df['Feature'].tolist()
selected_features.append('risk')

df_high_Kscore01 = upsampled_df[selected_features]
```

In [ ]:
```
print(df_high_Kscore01['risk'].value_counts())
```

In [ ]:
```
#split to train and test data
X = df_high_Kscore01.drop(['risk'], axis=1)
y = df_high_Kscore01.risk
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=1, stratify=y)

#scale the data:
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

In [ ]:
```
#Linear regression on features > 1

#fit the data and get scores:

svm_model = SVC(kernel='linear')
svm_model.fit(X_train_std, y_train)

print('Training accuracy:', svm_model.score(X_train_std, y_train))
print('Test accuracy:', svm_model.score(X_test_std, y_test))
```

In [ ]:
```
X_train_df = pd.DataFrame(X_train_std, columns = X.columns)
```

In [ ]:
```
coefficients = pd.DataFrame({'Variable': X_train_df.columns, 'Coefficient': svm_mod
#coefficients
```

In [ ]:
```
coefficients.sort_values('Coefficient', ascending=False).head(10)
```

In [ ]:
```
coefficients.sort_values('Coefficient', ascending=True).head(10)
```

Random Forest: Training accuracy: 0.998 || Test accuracy: 0.957

In [ ]:
```
forest = RandomForestClassifier(criterion='gini',
                                n_estimators=25,
                                random_state=1,
                                n_jobs=2)
forest.fit(X_train_std, y_train)

print('Training accuracy:', forest.score(X_train_std, y_train))
print('Test accuracy:', forest.score(X_test_std, y_test))
```

In [28]:
```
n_features = len(df_high_Kscore01.columns)
```

In [31]:
```
param_grid = {'n_estimators': [10, 20, 50],
              'max_features':[int(np.floor(np.sqrt(n_features))), int(np.floor(np.l
              'max_depth':[None, 3, 10],
              'criterion': ["gini", "entropy", "log_loss"],
              'random_state': [1],
              'n_jobs': [2, 5, 20]
             }

grid_search = GridSearchCV(RandomForestClassifier(),
                           param_grid=param_grid)
grid_search.fit(X_train_std, y_train)
print(grid_search.best_params_)
```

```
{'criterion': 'gini', 'max_depth': None, 'max_features': 11, 'n_estimators': 50, 'n_
jobs': 2, 'random_state': 1}
```

In [32]:
```
param_grid = {'n_estimators': [40, 50, 70],
              'max_features':[11],
              'max_depth':[None],
```

```
              'criterion': ["gini"],
              'random_state': [1],
              'n_jobs': [1, 2, 3]
             }

grid_search = GridSearchCV(RandomForestClassifier(),
                           param_grid=param_grid)
grid_search.fit(X_train_std, y_train)
print(grid_search.best_params_)
```

```
{'criterion': 'gini', 'max_depth': None, 'max_features': 11, 'n_estimators': 40, 'n_
jobs': 1, 'random_state': 1}
```

In [33]:
```
param_grid = {'n_estimators': [35, 40, 45],
              'max_features':[11],
              'max_depth':[None],
              'criterion': ["gini"],
              'random_state': [1],
              'n_jobs': [1]
             }

grid_search = GridSearchCV(RandomForestClassifier(),
                           param_grid=param_grid)
grid_search.fit(X_train_std, y_train)
print(grid_search.best_params_)
```

```
{'criterion': 'gini', 'max_depth': None, 'max_features': 11, 'n_estimators': 40, 'n_
jobs': 1, 'random_state': 1}
```

In [34]:
```
forest = RandomForestClassifier(criterion='gini',
                                n_estimators=40,
                                max_depth= None,
                                max_features = 11,
                                random_state=1,
                                n_jobs=1)
forest.fit(X_train_std, y_train)

print('Training accuracy:', forest.score(X_train_std, y_train))
print('Test accuracy:', forest.score(X_test_std, y_test))
```

```
Training accuracy: 0.9976763147166735
Test accuracy: 0.9574852577515693
```

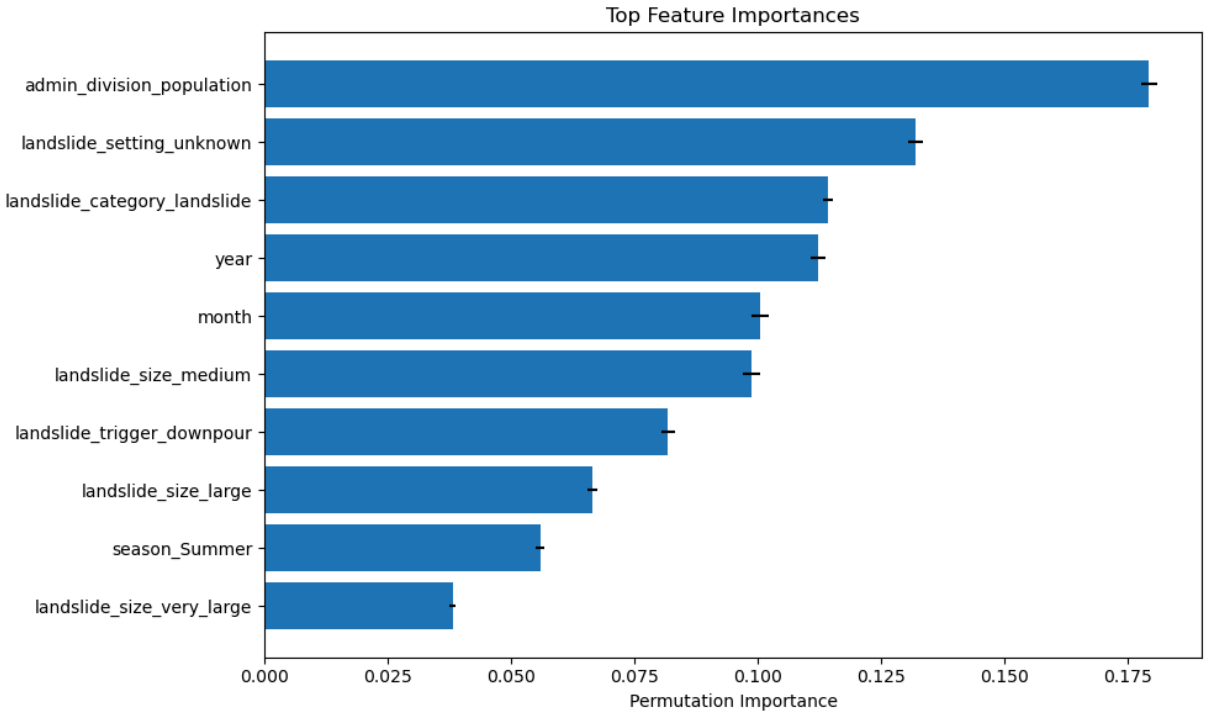
In [109…
```
from sklearn.inspection import permutation_importance
result = permutation_importance(forest, X_train_std, y_train, n_repeats=30, random_
```

In [110…
```
X_train_df = pd.DataFrame(X_train_std, columns = X.columns)
```

In [111…
```
importances_df = pd.DataFrame({
    'feature': X_train_df.columns,
    'importance_mean': result.importances_mean,
    'importance_std': result.importances_std
}).sort_values(by='importance_mean', ascending=False)
```

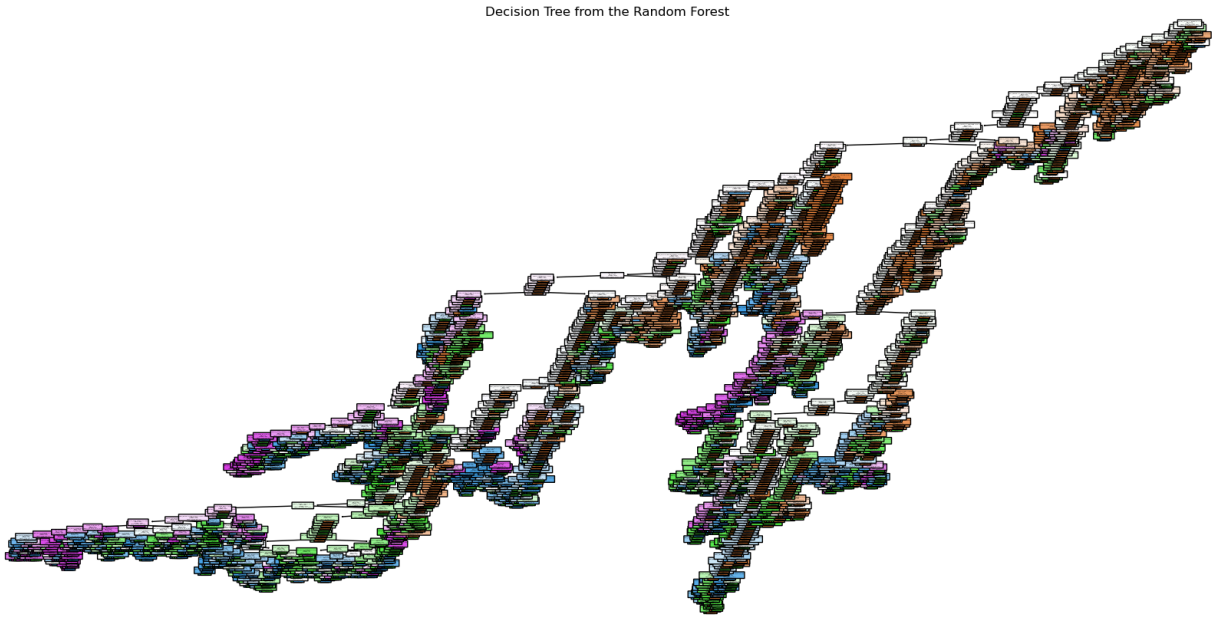
In [112…
```
top_n = 10
plt.figure(figsize=(10, 6))
```

```
plt.barh(importances_df['feature'][:top_n][::-1],
         importances_df['importance_mean'][:top_n][::-1],
         xerr=importances_df['importance_std'][:top_n][::-1])
plt.xlabel("Permutation Importance")
plt.title("Top Feature Importances")
plt.tight_layout()
plt.show()
```



In [187…

```
# for fun, let's see how the tree is looking
from sklearn.tree import plot_tree

tree = forest.estimators_[0]
plt.figure(figsize=(20,10))
plot_tree(tree,
          feature_names=X_train_df.columns,
          class_names=['1', '2', '3', '4'],
          filled=True,
          rounded=True)
plt.title("Decision Tree from the Random Forest")
plt.show()
```



CART Model Training accuracy: 0.998 || Test accuracy: 0.951

Training accuracy: 0.9977170811251529 Test accuracy: 0.9506372455773254

In [126…
```
cart = DecisionTreeClassifier(criterion='gini', random_state=1)
cart.fit(X_train_std, y_train)

print('Training accuracy:', cart.score(X_train_std, y_train))
print('Test accuracy:', cart.score(X_test_std, y_test))
```

```
Training accuracy: 0.9911229471815357
Test accuracy: 0.8726268553676216
```

In [35]:
```
param_grid = {'splitter': ["best", "random"],
              'max_depth':[30, 50, 80],
              'criterion': ["gini", "entropy", "log_loss"],
              'min_weight_fraction_leaf': [0.0, .2, .5]
             }

grid_search = GridSearchCV(DecisionTreeClassifier(random_state= 1),
                           param_grid=param_grid)
grid_search.fit(X_train_std, y_train)
print(grid_search.best_params_)
```

```
{'criterion': 'entropy', 'max_depth': 30, 'min_weight_fraction_leaf': 0.0, 'splitte
r': 'best'}
```

In [36]:
```
param_grid = {'max_depth':[70, 80, 90],
              'min_weight_fraction_leaf': [0.0, 0.01, 0.02]
             }

grid_search = GridSearchCV(DecisionTreeClassifier(random_state= 1, splitter= 'best'
                           param_grid=param_grid)
```

```
grid_search.fit(X_train_std, y_train)
print(grid_search.best_params_)
```

```
{'criterion': 'log_loss', 'max_depth': 80, 'min_weight_fraction_leaf': 0.0, 'splitte
r': 'best'}
```

In [37]:
```
param_grid = {'max_depth':[62, 63, 64],
              }

grid_search = GridSearchCV(DecisionTreeClassifier(random_state= 1, splitter= 'best'
                                                  min_weight_fraction_leaf=n 0.0),
                           param_grid=param_grid)
grid_search.fit(X_train_std, y_train)
print(grid_search.best_params_)
```

```
{'criterion': 'log_loss', 'max_depth': 63, 'min_weight_fraction_leaf': 0.0, 'splitte
r': 'best'}
```

In [38]:
```
param_grid = {'max_depth':[61, 62, 63]
              }

grid_search = GridSearchCV(DecisionTreeClassifier(random_state= 1, splitter= 'best'
                                                  min_weight_fraction_leaf=n 0.0),
                           param_grid=param_grid)
grid_search.fit(X_train_std, y_train)
print(grid_search.best_params_)
```

```
{'criterion': 'log_loss', 'max_depth': 61, 'min_weight_fraction_leaf': 0.0, 'splitte
r': 'best'}
```

In [35]:
```
cart = DecisionTreeClassifier(criterion='log_loss',
                              splitter = 'best',
                              max_depth = 62,
                              min_weight_fraction_leaf = 0.0,
                              random_state=1)
cart.fit(X_train_std, y_train)

print('Training accuracy:', cart.score(X_train_std, y_train))
print('Test accuracy:', cart.score(X_test_std, y_test))
```

```
Training accuracy: 0.9977170811251529
Test accuracy: 0.9506372455773254
```

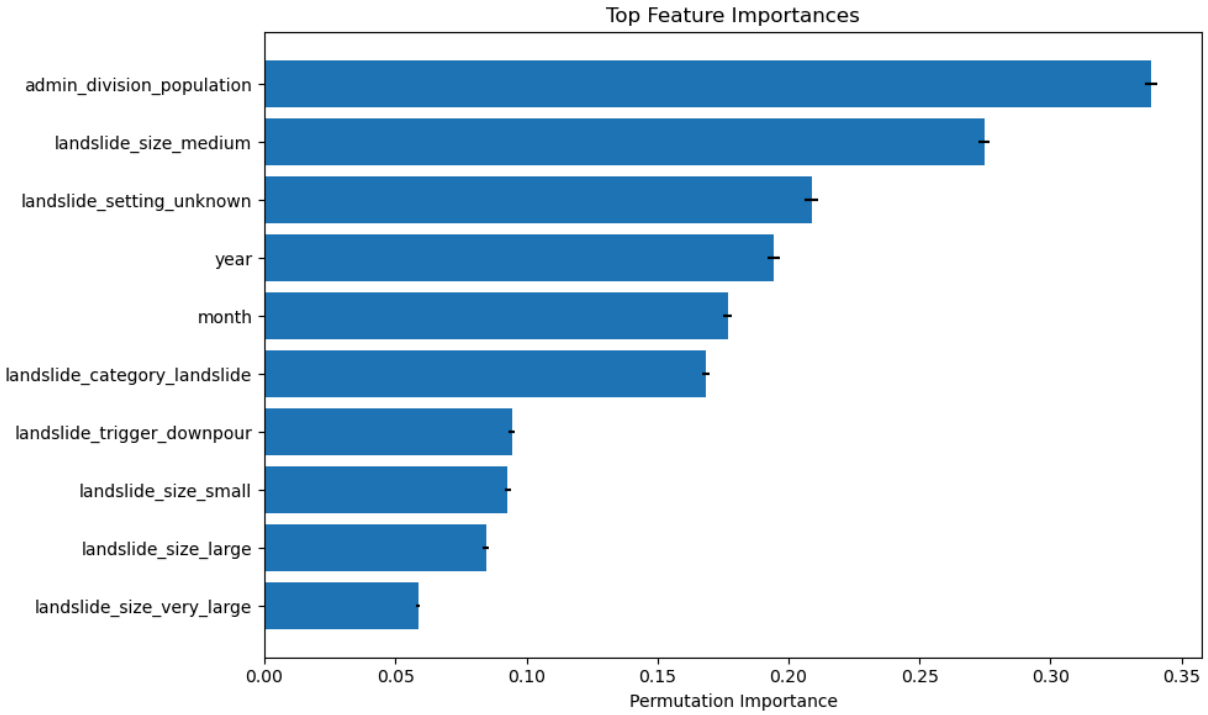
In [ ]:
```

```

In [99]:
```
from sklearn.inspection import permutation_importance
result = permutation_importance(cart, X_train_std, y_train, n_repeats=30, random_st
```

In [103…
```
X_train_df = pd.DataFrame(X_train_std, columns = X.columns)
```

In [105…
```
importances_df = pd.DataFrame({
    'feature': X_train_df.columns,
    'importance_mean': result.importances_mean,
    'importance_std': result.importances_std
}).sort_values(by='importance_mean', ascending=False)
```

In [107…
```
top_n = 10  # number of top features to show
plt.figure(figsize=(10, 6))
plt.barh(importances_df['feature'][:top_n][::-1],
         importances_df['importance_mean'][:top_n][::-1],
         xerr=importances_df['importance_std'][:top_n][::-1])
plt.xlabel("Permutation Importance")
plt.title("Top Feature Importances")
plt.tight_layout()
plt.show()
```



Linear SVC Training accuracy: 0.799 || Test accuracy: 0.764

In [ ]:
```
param_grid = {'penalty': ['l1', 'l2'],
              'C': [0.01, 0.05, 0.08, 0.1],
              'loss': ['hinge', 'squared_hinge']
             }

grid_search = GridSearchCV(LinearSVC(dual = False, max_iter = 20000, random_state =
                           param_grid=param_grid)
grid_search.fit(X_train_std, y_train)
print(grid_search.best_params_)
```

In [43]:
```
param_grid = {'C': [0.01, 0.02, 0.03]
             }

grid_search = GridSearchCV(LinearSVC(dual = False, penalty = 'l1', max_iter = 20000
                           param_grid=param_grid)
grid_search.fit(X_train_std, y_train)
print(grid_search.best_params_)
```

```
C:\Users\ashkl\anaconda3\Lib\site-packages\sklearn\svm\_base.py:1244: ConvergenceWar
ning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn(
{'C': 0.02}
```

In [36]:
```
svclinear_model = LinearSVC(dual = False,
                            penalty = 'l1',
                            max_iter = 20000,
                            loss = 'squared_hinge',
                            random_state = 1, C=0.02)
svclinear_model.fit(X_train_std, y_train)

print('Training accuracy:', svclinear_model.score(X_train_std, y_train))
print('Test accuracy:', svclinear_model.score(X_test_std, y_test))
```

```
Training accuracy: 0.7991031390134529
Test accuracy: 0.7642191363895758
```

In [ ]:
```
#train accuracy gets higher using default C, but train accuracy is lower
svclinear_model2 = LinearSVC(dual = False, penalty = 'l1', max_iter = 20000, loss =
svclinear_model2.fit(X_train_std, y_train)

print('Training accuracy:', svclinear_model2.score(X_train_std, y_train))
print('Test accuracy:', svclinear_model2.score(X_test_std, y_test))
```

In [83]:
```
X_train_df = pd.DataFrame(X_train_std, columns = X.columns)
coefficients = pd.DataFrame({'Variable': X_train_df.columns, 'Coefficient': svcline
coefficients.sort_values('Coefficient', ascending=False).head(10)
```

Out[83]:

	Variable	Coefficient
154	country_name_United States	0.583482
31	landslide_size_medium	0.349732
32	landslide_size_small	0.297659
16	landslide_trigger_downpour	0.205534
36	landslide_setting_above_road	0.175938
46	landslide_setting_unknown	0.170389
66	country_name_Canada	0.155604
153	country_name_United Kingdom	0.149071
6	landslide_category_landslide	0.116058
3433	biome_Western Himalayan broadleaf forests	0.109023

In [85]:
```
coefficients.sort_values('Coefficient', ascending=True).head(10)
```

Out[85]:

	Variable	Coefficient
68	country_name_China	-0.050608
2779	gazeteer_closest_point_Tura	-0.026842
1316	gazeteer_closest_point_Kakunodatemachi	-0.024407
685	gazeteer_closest_point_Cianjur	-0.019490
3275	biome_North Central Rockies forests	-0.019400
3097	biome_California interior chaparral and woodlands	-0.017320
82	country_name_Germany	-0.016991
3115	biome_Central American Atlantic moist forests	-0.015045
2435	gazeteer_closest_point_Semirara	-0.014113
1182	gazeteer_closest_point_Hiroshima-shi	-0.012367

SVC (kernel = Linear) Training accuracy: 0.801 || Test accuracy: 0.7139

In [ ]:
```
param_grid = {'C': [0.01, 0.05, 0.1, 0.08],
              'decision_function_shape': ['ovr', 'ovo']
             }

grid_search = GridSearchCV(SVC(kernel = 'linear', random_state= 1),
                           param_grid=param_grid)
grid_search.fit(X_train_std, y_train)
print(grid_search.best_params_)
```

In [ ]:
```
param_grid = {'C': [0.01, 0.02, 0.03]
             }

grid_search = GridSearchCV(SVC(kernel = 'linear', random_state= 1, decision_functio
                           param_grid=param_grid)
grid_search.fit(X_train_std, y_train)
print(grid_search.best_params_)
```

In [37]:
```
svc_model = SVC(kernel = 'linear', random_state= 1,
                decision_function_shape= 'ovr')
svc_model.fit(X_train_std, y_train)

print('Training accuracy:', svc_model.score(X_train_std, y_train))
print('Test accuracy:', svc_model.score(X_test_std, y_test))
```

```
Training accuracy: 0.8005707297187118
Test accuracy: 0.7134297127639339
```

In [79]:
```
X_train_df = pd.DataFrame(X_train_std, columns = X.columns)
coefficients = pd.DataFrame({'Variable': X_train_df.columns, 'Coefficient': svc_mod
```

```
coefficients.sort_values('Coefficient', ascending=False).head(10)
```

Out[79]:

	Variable	Coefficient
31	landslide_size_medium	1.237586
68	country_name_China	1.020584
32	landslide_size_small	0.952286
6	landslide_category_landslide	0.924603
3036	gazeteer_closest_point_unknown	0.823382
46	landslide_setting_unknown	0.801074
16	landslide_trigger_downpour	0.679744
3342	biome_Serra do Mar coastal forests	0.676831
36	landslide_setting_above_road	0.652958
154	country_name_United States	0.636420

In [81]:
```
coefficients.sort_values('Coefficient', ascending=True).head(10)
```

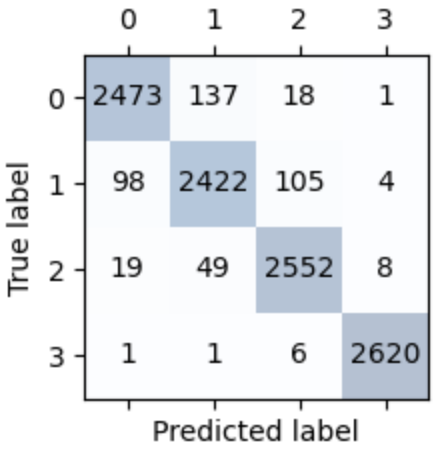
Out[81]:

	Variable	Coefficient
128	country_name_Philippines	-0.212029
3054	season_Spring	-0.089936
3121	biome_Central China loess plateau mixed forests	-0.053140
2779	gazeteer_closest_point_Tura	-0.048113
116	country_name_Myanmar	-0.046781
540	gazeteer_closest_point_Bāgh	-0.036212
92	country_name_Indonesia	-0.036062
44	landslide_setting_other	-0.031677
82	country_name_Germany	-0.017811
1251	gazeteer_closest_point_Islamabad	-0.016527

Confusion Matrices

In [40]:
```
#Random Forest Model
y_pred = forest.predict(X_test_std)
confmat = confusion_matrix(y_true=y_test, y_pred=y_pred)

fig, ax = plt.subplots(figsize=(2.5, 2.5))
ax.matshow(confmat, cmap=plt.cm.Blues, alpha=0.3)
for i in range(confmat.shape[0]):
```

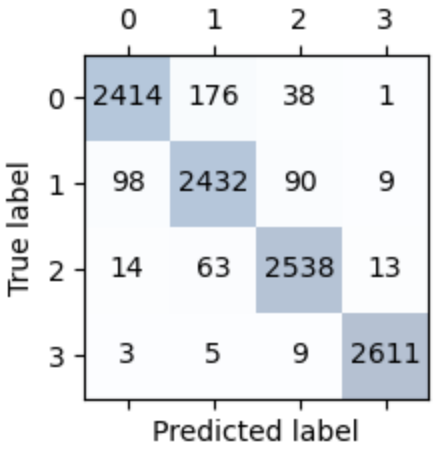
```
    for j in range(confmat.shape[1]):
        ax.text(x=j, y=i, s=confmat[i, j], va='center', ha='center')

plt.xlabel('Predicted label')
plt.ylabel('True label')

plt.tight_layout()
plt.show()
```



In [10]:
```
print('Class 0 mislabeled percentage: ',(137+18+1)/2437, '\n'
      'Class 1 mislabeled percentage: ',(98+105+4)/2422, '\n'
      'Class 2 mislabeled percentage: ',(19+49+8)/2552, '\n'
      'Class 3 mislabeled percentage: ',(1+1+6)/2620, '\n')
```

```
Class 0 mislabeled percentage:  0.06401313089864588
Class 1 mislabeled percentage:  0.08546655656482247
Class 2 mislabeled percentage:  0.029780564263322883
Class 3 mislabeled percentage:  0.0030534351145038168
```

In [41]:
```
#CART Model
y_pred = cart.predict(X_test_std)
confmat = confusion_matrix(y_true=y_test, y_pred=y_pred)

fig, ax = plt.subplots(figsize=(2.5, 2.5))
ax.matshow(confmat, cmap=plt.cm.Blues, alpha=0.3)
for i in range(confmat.shape[0]):
    for j in range(confmat.shape[1]):
        ax.text(x=j, y=i, s=confmat[i, j], va='center', ha='center')

plt.xlabel('Predicted label')
plt.ylabel('True label')

plt.tight_layout()
plt.show()
```



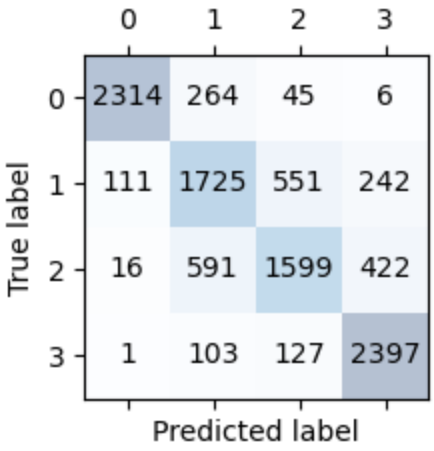
In [11]:
```
print('Class 0 mislabeled percentage: ',(176+38+1)/2414, '\n'
      'Class 1 mislabeled percentage: ',(98+90+9)/2432, '\n'
      'Class 2 mislabeled percentage: ',(14+63+13)/2538, '\n'
      'Class 3 mislabeled percentage: ',(3+5+9)/2611, '\n')
```

```
Class 0 mislabeled percentage:  0.08906379453189726
Class 1 mislabeled percentage:  0.08100328947368421
Class 2 mislabeled percentage:  0.03546099290780142
Class 3 mislabeled percentage:  0.0065109153581003444
```

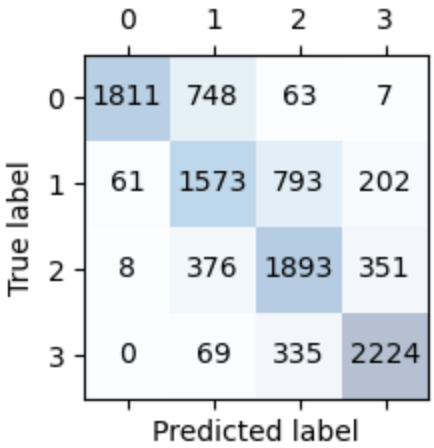
In [17]:
```
((176+38+1)+(98+90+9)+(14+63+13)+(3+5+9))/(2414+2432+2538+2611)
```

Out[17]:
```
0.05192596298149074
```

In [42]:
```
#LinearSVC Model
y_pred = svclinear_model.predict(X_test_std)
confmat = confusion_matrix(y_true=y_test, y_pred=y_pred)

fig, ax = plt.subplots(figsize=(2.5, 2.5))
ax.matshow(confmat, cmap=plt.cm.Blues, alpha=0.3)
for i in range(confmat.shape[0]):
    for j in range(confmat.shape[1]):
        ax.text(x=j, y=i, s=confmat[i, j], va='center', ha='center')

plt.xlabel('Predicted label')
plt.ylabel('True label')

plt.tight_layout()
plt.show()
```



In [43]:
```
#SVC(kernel = 'linear') Model
y_pred = svc_model.predict(X_test_std)
confmat = confusion_matrix(y_true=y_test, y_pred=y_pred)

fig, ax = plt.subplots(figsize=(2.5, 2.5))
ax.matshow(confmat, cmap=plt.cm.Blues, alpha=0.3)
for i in range(confmat.shape[0]):
    for j in range(confmat.shape[1]):
        ax.text(x=j, y=i, s=confmat[i, j], va='center', ha='center')

plt.xlabel('Predicted label')
plt.ylabel('True label')

plt.tight_layout()
plt.show()
```



Accuracy, precision, recall

In [45]:
```
from sklearn.metrics import precision_score, recall_score, f1_score
y_true=y_test
```

Random Forest

In [58]:
```
y_pred = forest.predict(X_test_std)

print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred, average='we
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred, average='weighted
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred, average='weighted'))
```

```
Precision: 0.957
Recall: 0.957
F1: 0.957
```

CART model

In [60]:
```
y_pred = cart.predict(X_test_std)

print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred, average='we
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred, average='weighted
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred, average='weighted'))
```

```
Precision: 0.951
Recall: 0.951
F1: 0.951
```

SVCLinear

In [63]:
```
y_pred = svclinear_model.predict(X_test_std)

print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred, average='we
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred, average='weighted
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred, average='weighted'))
```

```
Precision: 0.765
Recall: 0.764
F1: 0.762
```

SVC(kernel = 'linear')

In [66]:
```
y_pred = svc_model.predict(X_test_std)

print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred, average='we
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred, average='weighted
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred, average='weighted'))
```

```
Precision: 0.736
Recall: 0.713
F1: 0.718
```

In [ ]:
```

```