Project Everest Report - Jakub Bartnik

Author: Jakub Bartnik

GitHub URL

https://github.com/jakubbartnik/UCDPA_jakubbartnik

Abstract

This review contains the summary about impact of introducing new safety gear for climbing versus expedition falling rate that climbed in area of Himalaya Nepal within last 100 years. The review will explain how all that was done by myself and what conclusion I made from results. Report content can be found in python notebook including the code so it is easier to follow steps. There is another python notebook in repository demonstrating simple webscrapping. File name is Web_scrapping_quotes.ipynb.

Introduction

I chose this topic for my Data Science project as I have a strong personal interest in climbing. The objective of this research is to examine the impact of safety equipment, such as steel carabiners and dynamic ropes and harnesses, on the survival rates of climbers in the Nepal Himalaya region over the years. For my analysis, I will be utilising a Mount Everest dataset that encompasses all expeditions from 1905 to 2019. To ensure a more structured and readable review, I like to set smaller goals that stem from my main objective.

Goal 1: is to investigate the impact of the introduction of the climbing carabiner in 1910 on the survival rate of expeditions. To achieve this, I will split the dataframe into two sets and compare the falls.

Goal 2: is to examine the effect of the implementation of dynamic rope and harness in the 1960s. By dividing the data into periods with and without these innovations and determine their impact on the falls prediction.

Goal 3: is to determine the optimum number of helpers (hired_staff) that an expedition should hire in order to increase their chances of success, and express this result as a percentage.

Dataset

The Himalayan Database is a remarkable resource that documents all expeditions in the Nepal Himalaya, including notable achievements by mountaineers, such as early attempts, first ascents, and major accidents. This invaluable resource spans a timeline from 1905 through Spring 2019, covering over 465 significant peaks in Nepal. The database was compiled from Elizabeth Hawley's expedition archives, supplemented by alpine journals, books, and contributions from Himalayan climbers. The dataset comprises of three CSV files: expeditions.csv (16 columns, 10364 rows), peaks.csv (8 columns, 467 rows), and members.csv (21 columns, 76519 rows). Surprisingly, the dataset contains about 280,630 empty values. The variables types summary shows

that the majority of columns (31) are string data type, followed by 7 boolean type columns, and 7 integer type columns.

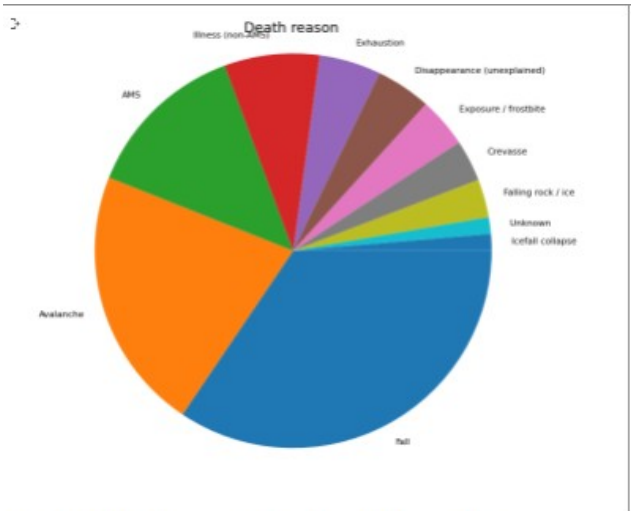
Implementation Process

Data cleaning

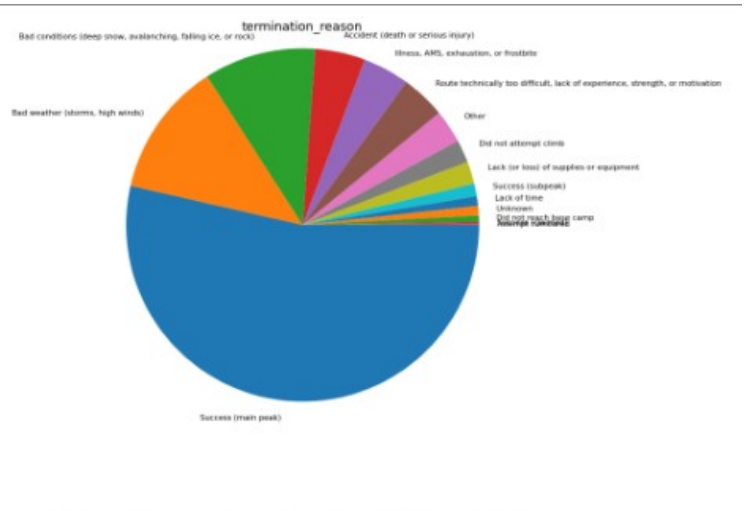
I started my data analysis process by manually reviewing the data. One of my objectives was to merge all of the csv files into a single one. To achieve this, I examined each csv file one by one and used the ".head()" function to view the first five rows but I limited that to 3 rows. Through this, I identified the "peak_id" column as the best one to merge the "expeditions.csv" file with the "peak.csv" file. I temporarily assigned this to "df1" and continued to merging it with the "members.csv" file based on the "expedition_id" column, which I assigned to the final dataframe as "df". I merged all three csv files together using the ".merge" command. Next, I searched for empty values in the "df" dataframe using the ".isna()" function. I examined the "death_cause" column more closely using the "df['death_cause'].unique()" function to gain a better understanding of its attributes. I noticed that the value "Other" did not add value, only noise; therefore, I replaced all "Other" entries with "nan". To limit the dataset's variables and avoid repetition, I decided to drop several columns that came after the merge, such as 'trekking_agency,' 'injury_type,' 'sex,' and 'age.' From "expeditions.csv", I kept "termination_reason," "expedition_id," and "year." From "peaks.csv," I kept "peak_name" and "climbing_status," while from "members.csv," I kept "death_cause." After trimming down the columns, I used the "expedition_id" to find any duplicated values that could be removed. Finally, I cleaned up the "expedition_id" column to remove excess strings using regular expressions and saved my clean data to a clean csv file called "clean_everest.csv."

Data exploration and preparation for goals

Importing libraries for exploration mainly for visual representation of results. Importing cleaned data as ‘dfc’. Features termination_reason and death_cause are nominal and need to be converted to dummy variables. I am doing it using dataframe option ‘.get_dummies’.
Showing overall procentage of reasons to die in Himalaya based on all 114 years of experience using pie chart and then doing it again for termination reason.



Graph 1. Death reason based on 114 years data

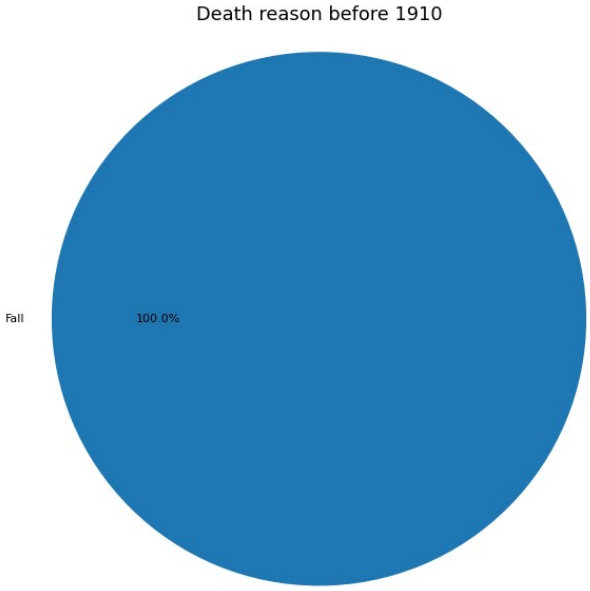


Graph 2. Termination reason based on 114 years data

It seems that avalanches, AMS(Alternated Mind status caused by trauma or intoxication) and Illnes are the main reasons for death after falling.

Exploration and preparation for Goal 1.

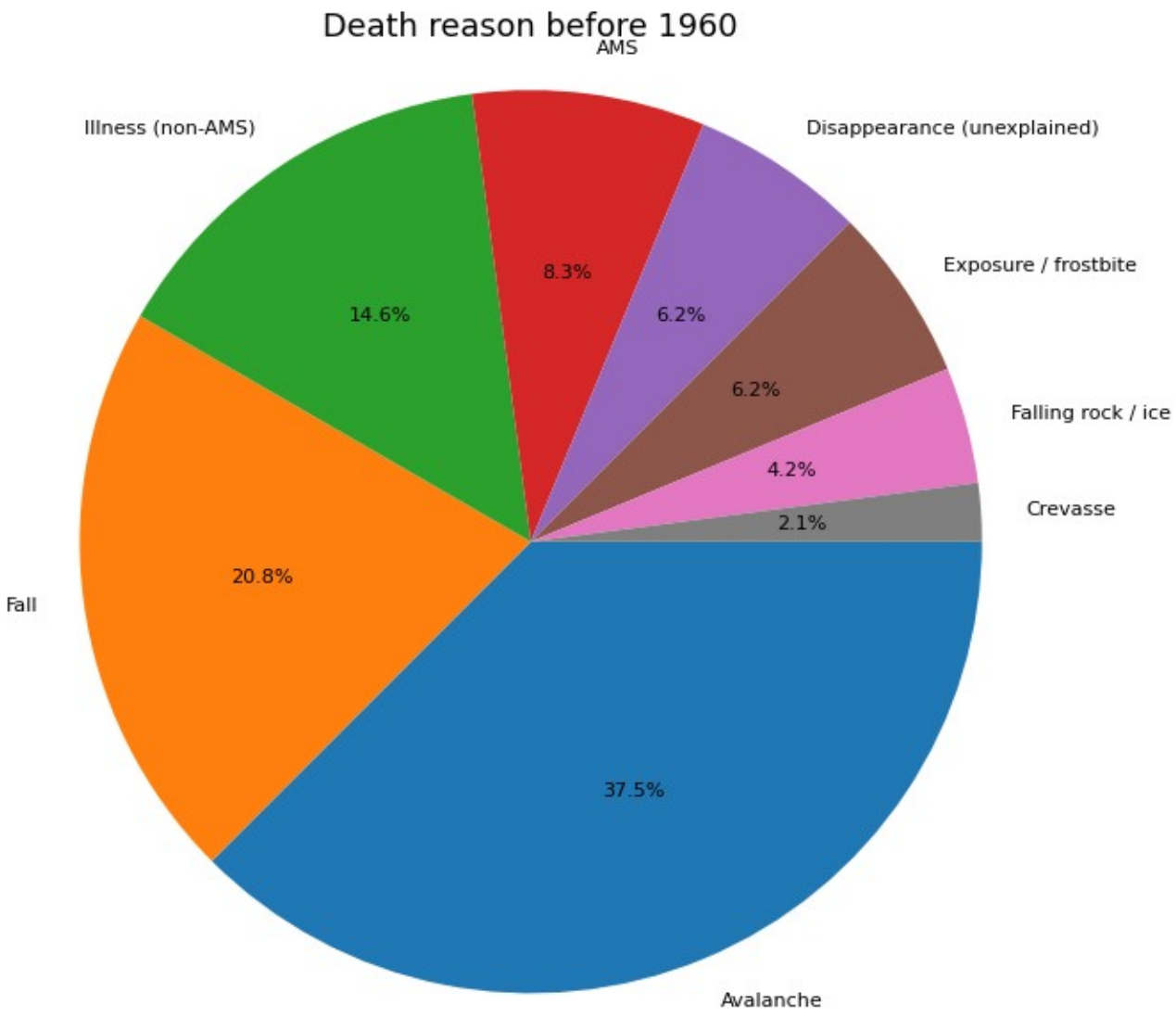
Splitting data for goal 1 and assign it to dataframe ‘before1910’ and ‘after1910’.



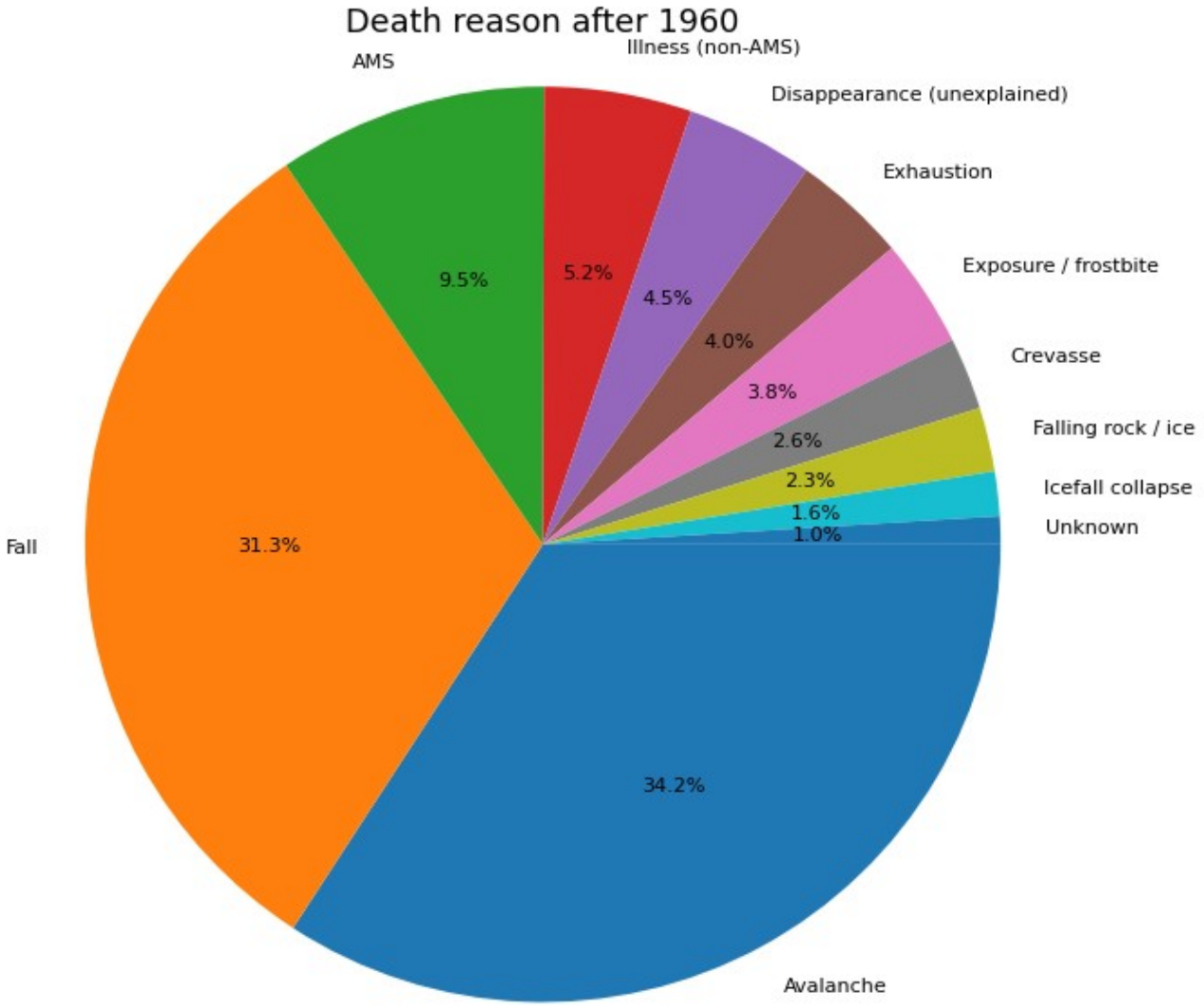
By shape and unique values of before1910 I can see that its really small and does contain only one record of falling. Therefore goal 1 will not be continued, there is not enough observation in data. Sugesstions how this can be fixed I provided in secetion Results.

Exploration and preparation for Goal 2.

Splited data before development of harnes and rope in 1960 and after 1961, assigned to no_harness_rope and yes_harness_rope. Ploted both dataframes for death_cause with pie chart.



Pie chart for data before 1960 indicate that avalanche and illness was the biggest reasons to die and falling going into third place.



Both graphs are very similar but its indicating that resons were changing along the years .

Created function that will save 1 when the fall occur and 0 if not. Stored result of categorise function in new feature. Then I decided to simplify my dataframe and use columns - 'year_x', 'members', 'member_deaths', 'hired_staff', 'hired_staff_deaths','died', 'falls'.

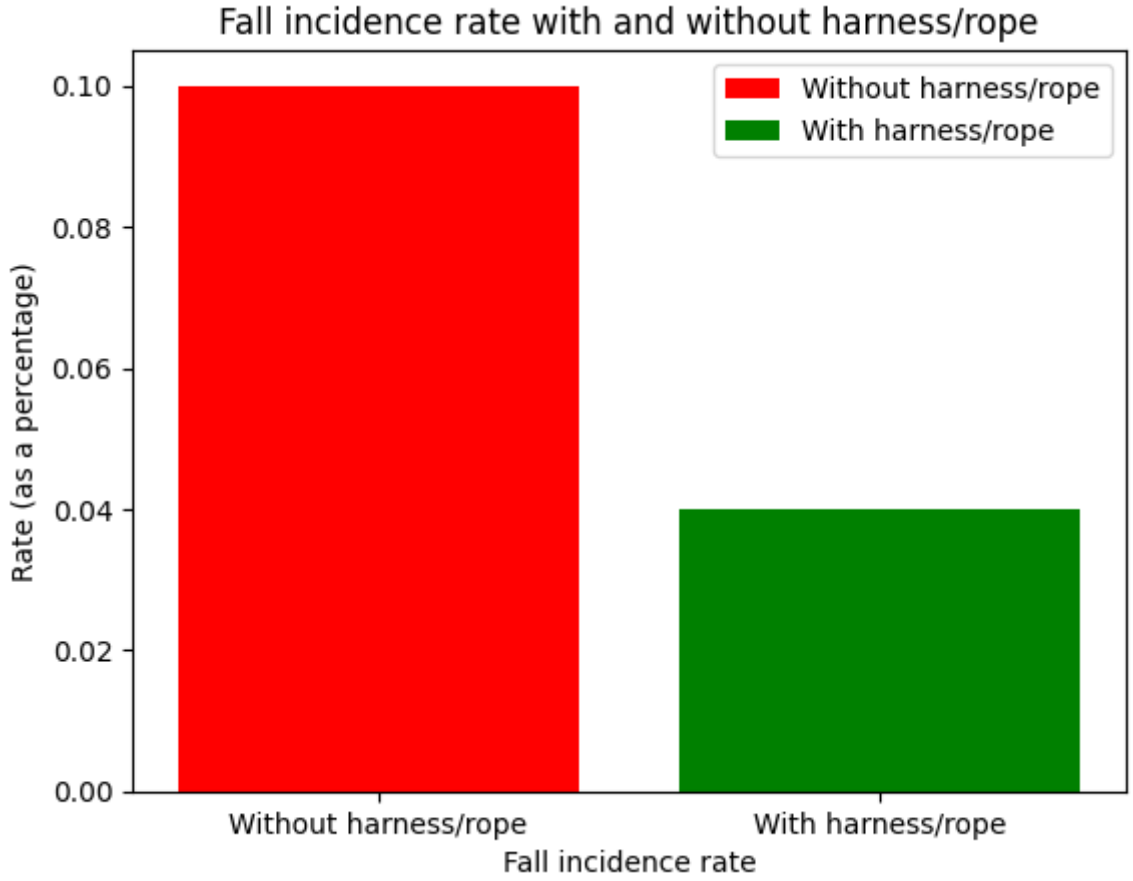
Analysis for Goal 2

Importing libraries for analysis for goeal2 - logistic regression and for goal3 RandomForestRegressor. GridSearchCV is used in both cases for Hyperparameter tuning.

Calculating the total number of falls and climbers in each group and then dividing the number of falls by the total number of climbers to find the fall incidence rate. Sliced dataframe for goal 2 for before and after dataframes and added new column 'harnes_used' where 0 mean no and 1 mean yes. This is to avoid 1 dimension data in model. Converted columns to a numeric type as per model requirement. Calculating the fall incidence rate for two different data periods one without harness and rope, and one with harness and rope.

Fall incidence rate without harness and rope: 0.10%
Fall incidence rate with harness and rope: 0.04%

This was visualised using boxplot:



The fall incidence rate without harness and rope is 0.10%, which means that out of all the climbers who did not use a harness and rope, 0.1% of them reported falls. The fall incidence rate with harness and rope is 0.04%, which means that out of all the climbers who used a harness and rope, 0.04% of them reported falls. Overall, using a harness and rope seems to be associated with a lower fall incidence rate compared to not using one.

Performing logistic regression model on 'before' and 'after' dataframes to predict falls and prints out the confusion matrix and classification report for both 'before' and 'after' dataframes. Model resulted in following:

```
Confusion Matrix for Before:
[[274   0]
 [  2   0]]
Classification Report for Before:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       274
           1       0.00      0.00      0.00         2

    accuracy                           0.99       276
   macro avg       0.50      0.50      0.50       276
weighted avg       0.99      0.99      0.99       276

Confusion Matrix for After:
[[14908     0]
 [   60     2]]
Classification Report for After:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     14908
           1       1.00      0.03      0.06        62

    accuracy                           1.00     14970
   macro avg       1.00      0.52      0.53     14970
weighted avg       1.00      1.00      0.99     14970
```

The results shows that my data might be unbalanced. I will oversample my data and try regression model again.

```
Confusion Matrix for BeforeRE:
[[250  17]
 [ 23 258]]
Classification Report for BeforeRE:
              precision    recall  f1-score   support

           0       0.92      0.94      0.93       267
           1       0.94      0.92      0.93       281

    accuracy                           0.93       548
   macro avg       0.93      0.93      0.93       548
weighted avg       0.93      0.93      0.93       548

Confusion Matrix for AfterRE:
[[13532  1231]
 [  717 14332]]
Classification Report for AfterRE:
              precision    recall  f1-score   support

           0       0.95      0.92      0.93     14763
           1       0.92      0.95      0.94     15049

    accuracy                           0.93     29812
   macro avg       0.94      0.93      0.93     29812
weighted avg       0.94      0.93      0.93     29812
```

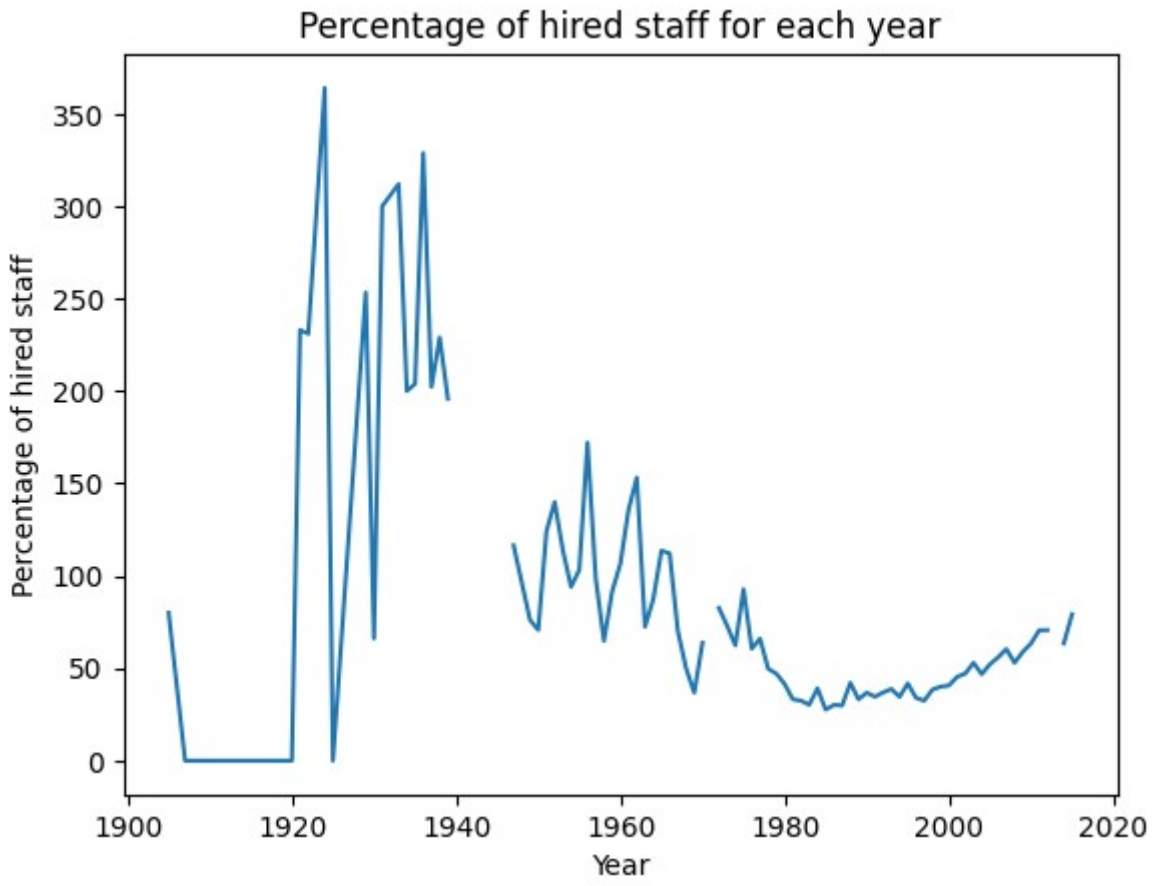
The first logistic regression model on before and after datasets resulted in very low recall values for the minority class(label 1 or falls; before dataset, recall=0.0 and after dataset, recall=0.03) where the oversampled models improved the recall values significantly (beforeRE dataset, recall=0.92 for label 1 and afterRE dataset, recall=0.95 for label 1). we can conclude that oversampling has significantly improved the logistic regression model's performance on the imbalanced datasets.

Hyperparameter tuning of logistic regression using GridSearchCV. GridSearchCV is searching through all possible combinations of 8 hyperparameter settings and using 5-fold cross-validation for each combination. The output indicates that the best set of hyperparameters is {'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}.

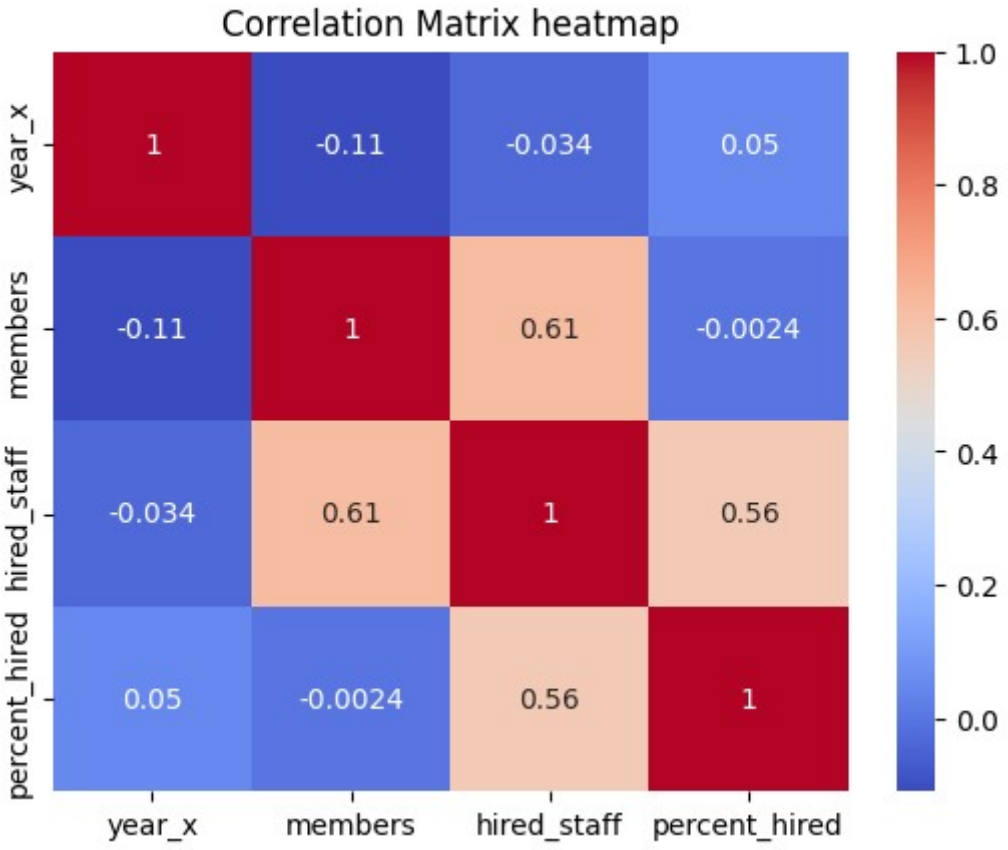
Exploration and preparation for Goal 3.

In interest of this goal is to limit dataset for only successful expeditions.
Using DataFrame.transform() method. I want to add percent of hired staff for amount of members to this data.

Visualise the percentage of hired staff for each year.



Data contained some inf and NaN values that were cleaned. Created heatmap of the correlation matrix to see related features.



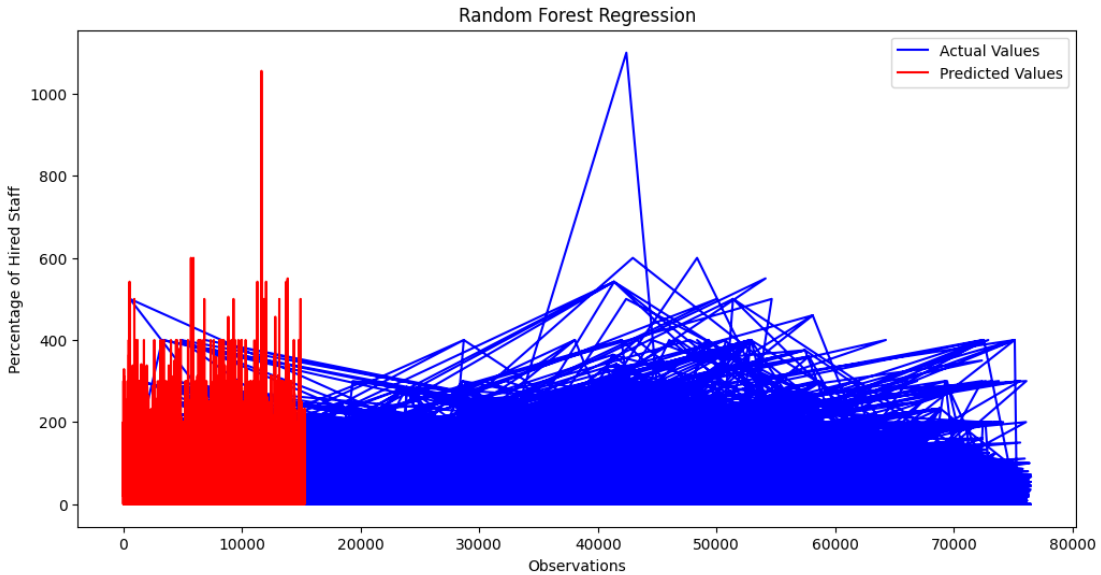
This indicated relationship on feaures like hired_staff with percent_hired and hired_staff with memebers.

Analysis for Goal 3

Creating a random forest regressor model to predict the 'percent_hired' column based on the 'members' and 'hired_staff' columns. It then fits the model to the train data and makes predictions on the test data. Finally, it evaluates the performance of the model using the mean squared error (MSE) and R-squared score.
Results are:

```
MSE: 0.13759521565547536
R-squared: 0.9999625008645775
```

This was showed graphically:



Trying to adjust hyperparameter to improve model performance using a GridSearchCV approach which is a popular method for finding the optimal hyperparameters by testing every combination of values for the hyperparameters. This code-box took about 11 min to run.

The first set of results, MSE=0.1376 and R-square=0.99996, are the performance metrics of my random forest regressor model without any hyperparameter tuning. This model used n_estimators=100, which is the default value for this parameter in the RandomForestRegressor() model. The second set of results, MSE=0.2514 and R-square=0.99993, are the performance metrics of the same random forest regressormodel after hyperparameter tuning, with the following settings: max_depth=None, min_samples_leaf=1, min_samples_split=5, and n_estimators=50. The tuned model uses a smaller n_estimators than the default value, but other parameters are set manually. Overall, we can see that the second model performed worse in terms of both MSE and R-squared compared to the first model. This suggests that the default settings for the RandomForestRegressor() model works better. I am going to skip plotting this results.

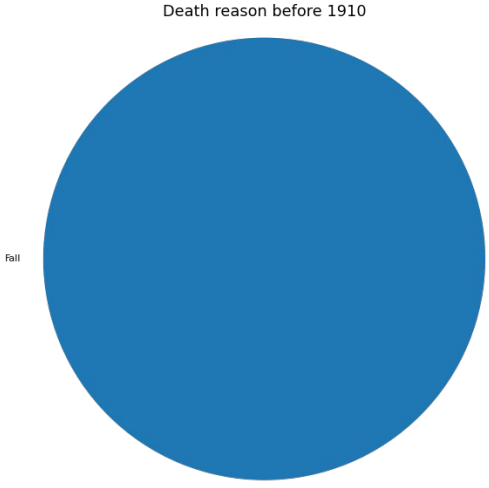
Observations for Goal 3

It looks like the mean squared error (MSE) is 0.1375 and the R-squared value is 0.9999, indicating that the random forest model fits thedata very well. This means that the model is able to make accurate predictions of the number of hired staff for future expeditions, based on the historical data that were fitted.

Results

Goal 1.

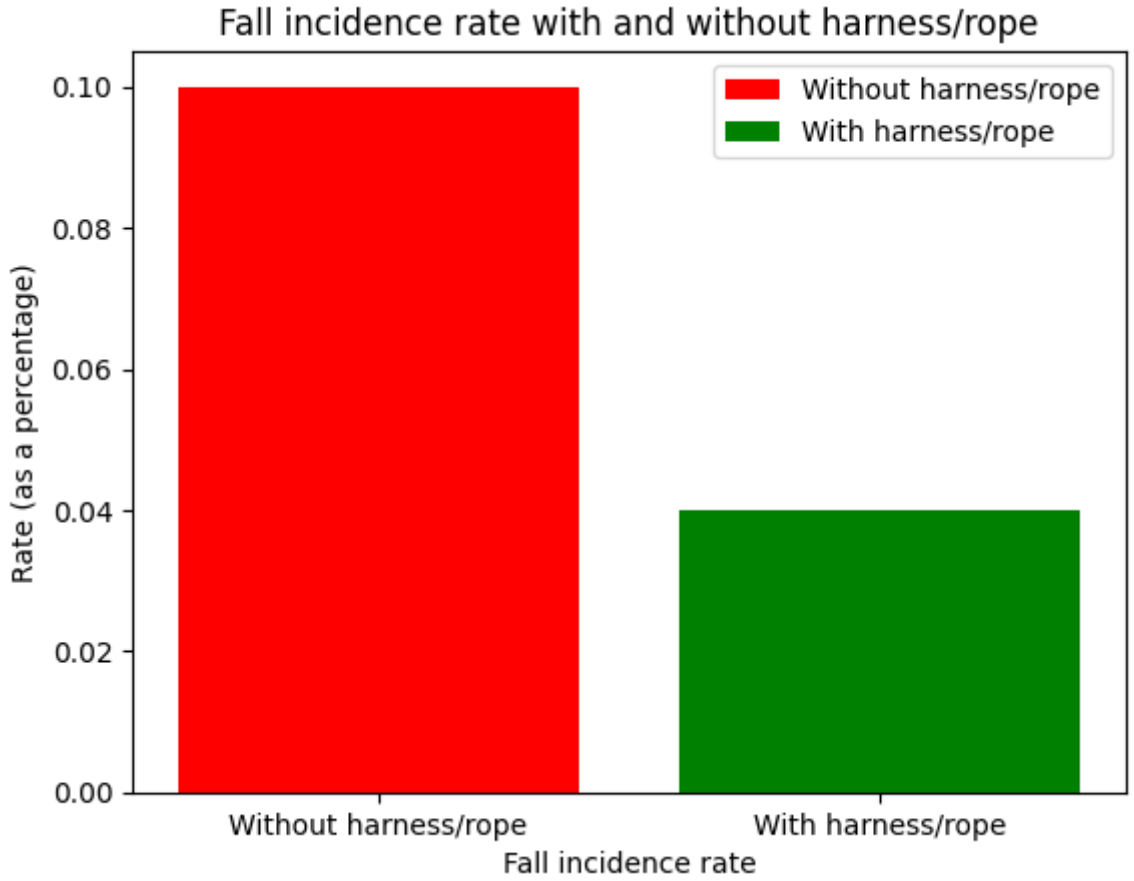
After exploring data on this paper it looks like goal 1 need to be skipped as there is not enough data to support this comparison. This is very clear when we see how many record are in splited dataframe for before 1910 period and from the pie chart.



To get goal 1 accomplished it would be much better to analyse climbing data from another part of the world where there is more observations. Unfortunately observations made in pre-computerised years would be spread across various personal diaries so it would be very hard to group it.

Goal 2.

It seems that falls were more likely to occur before 1960. This can be inferred from the fact that the 'recall' score for the '1' label ('falls') was higher in the 'before' dataset (0.00) than in the 'after' dataset (0.03), indicating that the model was better able to identify true positive 'falls' in the 'after' dataset than in the 'before' dataset.However after oversampling my data I got my conclusion completly oposite. That might mean that maybe Logistic regression is not good choice for this problem and I should search for another algorithm. I will leave it here as there is still another goal to archive. Without oversampling data I would be happy with the results of ML as it confirming Fall incidence rate calculation.



Goal 3.

From the graph I can read that hiring 20% to 40% of helping staff looks like perfect percentage for expedition to be successful.

Insights

Goal 1.

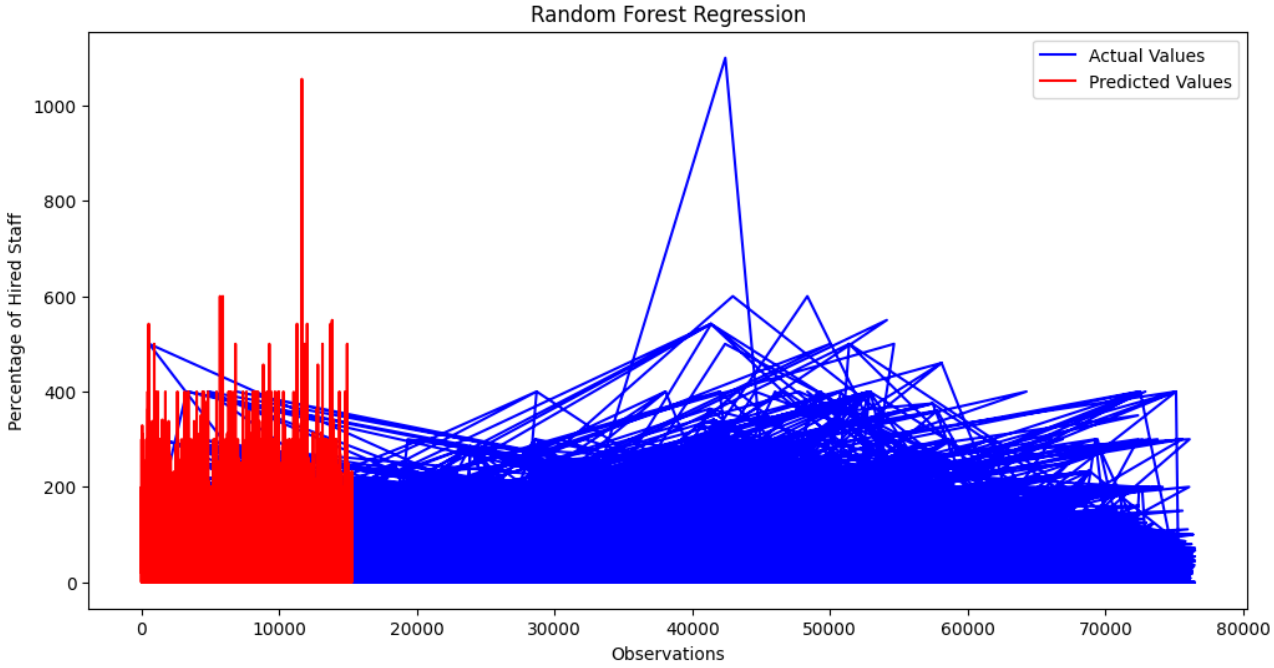
There is not enough observations to provide definite answer if steel carabiner made any difference on climbing safety

Goal 2.

Death due to fall is actually extreme as climbers are falling on route attempt basis. Not all falls are dangerous. In dataset fall is listed only as a reason for death but there is no count how many time climbers has fall before they die. One of the reason to die due to fall is not due to break of the rope but most often due to bad protection placing in the rock. As in Nepal we are dealing with Ice climbing it could be due to loose ice.

Goal 3.

After performing random forest regressor model on both type of data I got really good visualisation for predicted percentage for future expeditions.



From which I can read that hiring 20% to 40% of helping staff looks like perfect percentage for expedition to be successful. Where 20% seems to be minimum.

References

Dataset: https://www.kaggle.com/datasets/majunbajun/himalayan-climbing-expeditions
Convert nominal to numerical values: https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/
Generating more samples: https://machinelearningmastery.com/generate-test-datasets-python-scikit-learn/
Comparing: https://towardsdatascience.com/how-to-compare-two-or-more-distributions-9b06ee4d30bf
Scrapping webpage url: https://quotes.toscrape.com
Book - https://ethanweed.github.io/pythonbook
Supervised learning - https://campus.datacamp.com/courses/supervised-learning-with-scikit-learn/