Indian Institute of Technology, (B.H.U), Varanasi

EXPLORATORY PROJECT REPORT

Name – **Anmol Saluja**

Roll Number – **18065011**

Branch – **Civil Engineering**

Year – **2nd**

Semester – **4th**

Title – **Slope Stability Analysis using Machine Learning**

Performed Under – **Dr. Supriya Mohanty**

Slope Stability Analysis using Machine Learning

INDEX

Abstract

This project is focused on a performance comparison of four **supervised learning methods** for slope stability prediction. Based on characteristics of slope instability and analysis of data availability, six typical slope parameters—**the saturated unit weight, cohesion, internal friction angle, slope inclination** – were chosen to establish the evaluation index system. The **Multi-Layered Artificial Neural Network, Support Vector Machine, Random Forest Regression and Multiple Linear Regression** were used as the algorithms for the task. A data set of 65 observations (with varying parameters) was created using the software OptumG2 and was used to train and test the four classifiers, and then, key parameters of the four models were optimized by using the method of 5-fold cross validation (except for ANN). The prediction performances of the four supervised learning methods were compared and analysed. The results of various accuracy reveal that the ANN, RAF and SVM models can achieve satisfactory results.

Introduction and Theory

For any Civil or Geotechnical engineer, understanding and analysing the nature and stability of **slope** is necessary because of its application in a wide sphere of engineering application such as –

1. in the design of earth dams and embankments.

2. in the analysis of stability of natural slopes.

3. in the analysis of the stability of excavated slopes.

4. in the analysis of deep-seated failure of foundations and retaining walls.

5. in the appropriate design of foundations and other sub-structures in the areas prone to landslides.

A slope can occur as a **finite or infinite slope**.

A slope is said to be infinite, when the slope has no definite boundaries and soil under the free surface contains the same properties up to identical depths along the slope. The failure surface in this case is assumed to be parallel to the slope.

On the other hand, finite slopes are generally limited in terms of length and depth.

The basic purpose of slope stability analysis is determining a factor of safety against a potential failure, or landslide. This factor of safety is calculated by comparing the shear strength developed in the slope with the maximum shearing capacity or resistance of the slope. If this factor of safety is determined to be large enough, the slope is judged to be stable (safe).

Slope Stability analysis generally revolves around two things –

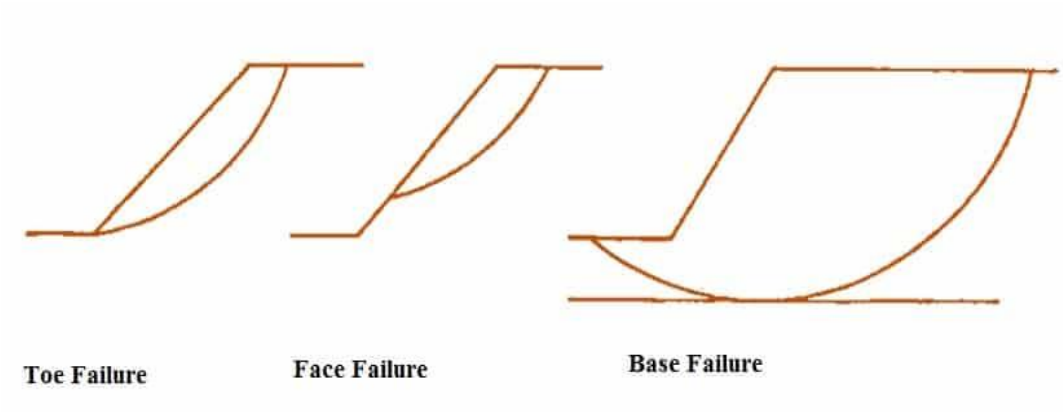
1. **Computing the Factor of Safety**
2. **Determining the failure type and failure surface**

There can be different ways in which a slope can undergo failure,

Translational Failure – Generally, happens in infinite slopes as their failure surface is parallel to their slopes.

Rotational Failure – Happens, in finite slopes when the failure occurs in rotation along a slip surface and a curved slip surface is obtained. It happens in 3 different ways depending on the soil properties beneath the slope –

i. Face failure – Occurs when the soil above the base of the soil is weaker compared to the soil at the base and below. Here, the slip surface intersects the slope.
ii. Toe failure – Occurs when the slip surface passes through the base or the toes of the slope.
iii. Base failure – Occurs when the soil at and beyond the base is weaker compared to the deeper soil strata. The slip surface overlaps the entire slope in this case.



Wedge Failure – This failure generates a slip plane which is inclined to the slope. This type of failure occurs when there are fissures, joints, or weak soil layers in slope, or when a slope is made of two different constituents.

Factor of Safety analysis is generally done by the following methods –

1. **Limit Equilibrium Method**
2. **Finite Element Analysis**

Limit Equilibrium Analysis revolves around searching the optimal failure plane by using the concepts of geometry and moment and force balancing. The model does not incorporate the displacement mechanics of the soil during the failure as it deals with the equilibrium states.

Finite Element Analysis is a computer-based detection of slope failure plane and subsequent displacement of the slope. This will take the centre stage for the project as software based FEA is used to generate data for the machine learning models.

With the advent of greater computational capacities and technological developments and its better visualization power, Finite Element Analysis has become an attractive choice for researchers to study the slope developments around the world.

However, Limit Equilibrium still remains the easiest and highly accurate method of FOS calculation.

In Limit Equilibrium Analysis, methods investigate the equilibrium of a soil mass tending to slide down under the influence of gravity. Transitional or rotational movement is considered on an assumed or known potential slip surface below the soil or rock mass. All these methods are based on the comparison of forces, moments, or stresses resisting movement of the mass with those that can cause failure motion (destabilizing forces). The output of the analysis is a factor of safety, defined as the **ratio of the shear strength/capacity to the shear stress** required for equilibrium. If the value of factor of safety is less than 1.0, the slope is deemed unstable.

<u>Factor of Safety with respect to Shear Strength</u>

Commonly, Factor of Safety is determined as the ratio of the shear strength to the mobilised shear stress along the failure surface.

Therefore, $FOS = s/\tau_m$

where s is the shear strength, τ_m is the mobilised shear stress and FOS is the factor of safety.

This can further be reduced to the following,

$$FOS = F = (c + \bar{\sigma} * \tan(\Phi))/(cm + \bar{\sigma} * \tan(\Phi m))$$

where cm is the mobilised cohesion, σ is the effective stress and Φm is the mobilised friction angle.

Therefore,

$$(cm + \bar{\sigma} * \tan(\Phi m)) = (c + \bar{\sigma} * \tan(\Phi))/F$$

On comparing both the sides,

$cm = c/F$; this indicates the Factor of Safety with respect to cohesion

$\tan(\Phi m) = \tan(\Phi)/F$; this indicates the Factor of Safety with respect to Friction Angle

Methods employed under Limit Equilibrium Analysis –

1. Swedish Circular Arc Method
2. Ordinary Method of Slices
3. Modified Bishop's method of analysis
4. Taylor's Stability Number

Limit Equilibrium models are mostly governed by the linear **Mohr-Coulomb** based relationships of shear and normal stresses of soils.

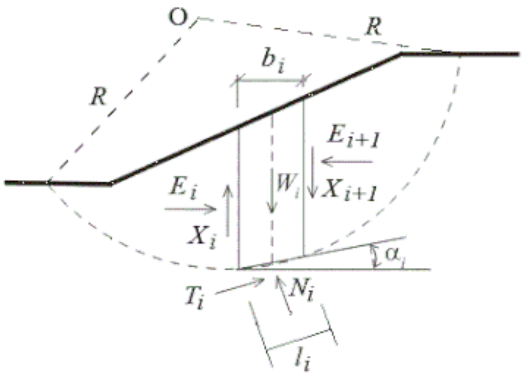
The following relation between shear strength, and parameters like **Cohesion and Friction Angle** of the soil is used extensively in method of slices -

$$\tau = \sigma' \tan \phi' + c'$$

where **τ** is the shear strength of the interface, σ' is the effective stress, Φ' is the effective friction angle of the soil and c' is the effective cohesion.

Swedish Circular Method

The Swedish Slip Circle method assumes that the friction angle of the soil or rock is equal to zero, i.e. $\tau = c'$. When the friction angle is considered to be zero, the effective stress term goes to zero, thus equating the shear strength to the cohesion parameter of the given soil. The Swedish slip circle method assumes a circular failure interface, and analyses stress and strength parameters using circular geometry and statics. The moment caused by the internal driving forces of a slope is compared to the moment caused by forces resisting slope failure. If resisting forces are greater than driving forces, the slope is assumed stable.

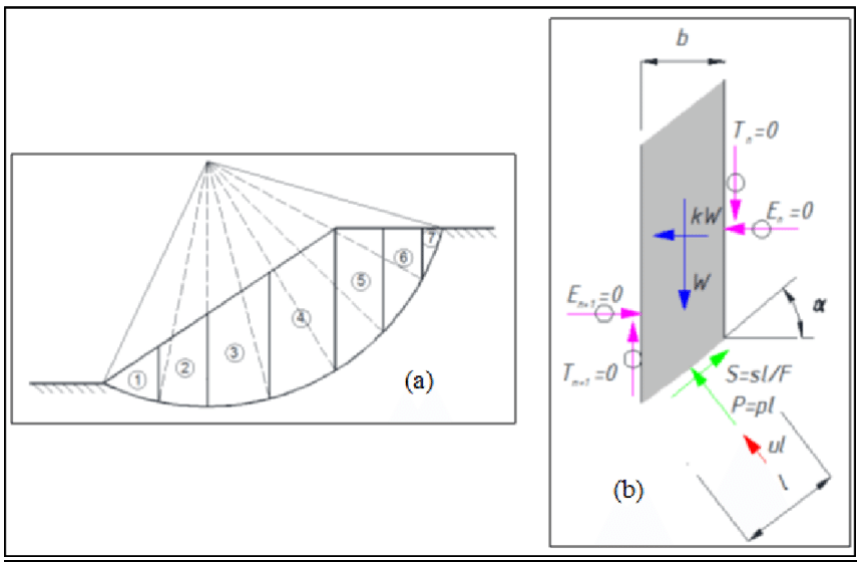


An illustration of Swedish Circle Method.

Ordinary Method of Slices

In the method of slices, the sliding mass above the failure surface is divided into a number of slices of different length and width. The forces acting on each slice are obtained by considering the force and moment equilibrium for the slices. Each slice is considered on its own and force interactions between slices are neglected because the resultant forces are parallel to the base of each slice.

This method allows for a simple static equilibrium calculation, considering only soil weight, along with shear and normal stresses along the failure plane. Both the friction angle and cohesion can be considered separately for each slice. This method gives a very conservative and often less accurate FOS as it doesn't consider the interslice normal and shear forces.



An illustration of Method of Slices

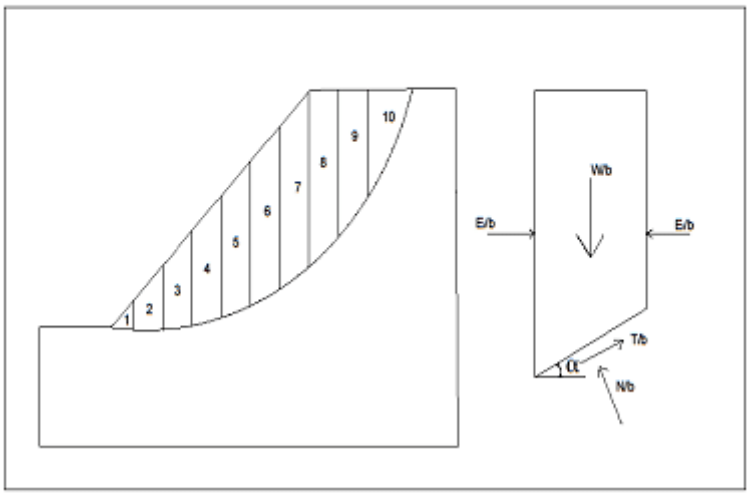
Factor of safety for the slice is equal to the ratio of the resisting moment (Mr) and the overturning moment (Mo).

The formula for Factor of Safety after derivation using Method of Slices turns out to be this –

$$FS = \frac{c'L_a + \tan\phi' \sum (W\cos\alpha - ul)}{\sum W \sin\alpha}$$

Modified Bishop's method of analysis

This method is slightly different from the ordinary method of slices. In this normal interaction forces between adjacent slices are assumed to be collinear and the resultant inter slice shear force is zero. This method satisfies vertical force equilibrium for each slice and overall moment equilibrium about the centre of the circular trial surface.



Taylor's Stability Number

If the slope angle β, height of embankment H, the effective unit weight of material γ, angle of Internal friction φ ', and unit cohesion c' are known, the factor of safety may be determined. In order to make unnecessary the more or less tedious stability determinations, Taylor (1937) conceived the idea of analysing the stability of a large number of slopes through a wide range of slope angles φ' and angles of internal friction, and then representing the results by an abstract number which he called the "stability number". This number is designated as Ns.

$$Ns = c' / (Fc * \gamma * H)$$

Taylor published his results in the form of curves which give the relationship between Ns. However, it is important to note that the stability numbers are obtained for factors of safety with respect to cohesion **by keeping the factor of safety with respect to friction ($F_\emptyset$) equal to unity**. Following, figure shows the stability number (N_s) v/s slope angle (β) distribution for different values of friction angle (Φ). However, these curves may be used without serious error for slopes down to β = 14°. The stability number Ns for the case when Φ= 0 is greatly dependent on the position of the ledge.

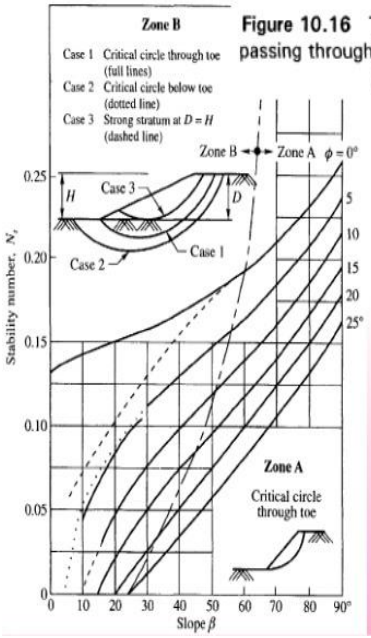


Figure 10.16 Taylor's stability numbers for circles passing through the toe and below or above the toe (after Taylor, 1937)

Project Review

In recent years, data-mining techniques and intelligent evaluation models have been widely accepted in mining and geotechnical applications. Moreover, many supervised machine learning algorithms have been successfully been used in predicting factor of safety for slope stability analysis and similar tasks, as the quality and availability of data has increased. With the rise of Finite Element Analysis software like OptumG2, which provide a tremendously detailed visualization and a high degree of user adaptability, coupled with rising computational capacities of modern computers has aided geotechnical engineers in closely analysing the failure pattern of natural structures and predictive analysis for the same.

And while FE softwares are definitely the most accurate and powerful mechanisms for Geotechnical researchers, they are equally complex and costly for a comprehension. Moreover, they require a significant computational backup and are immobile (not suitable for field analysis).

Therefore, to strike a balance between the geotechnical requirements and economics of time and money, an ML based FOS predicting approach has been explored. **This approach aims to provide a fairly accurate Factor of Safety for given field conditions by providing the required input parameters.** This approach will work on Supervised Learning Models like SVM and Multi-Layers Perceptron/ANN. These models can be trained on large and diverse datasets to achieve high accuracies. Furthermore, the performance of these models can be judged by computing parameters like **R^2, Adjusted R^2 or Root Mean Squared Error**.

The model was built and implemented completely using Python 3.4 on a Google Collab Notebook (the same has been attached with the report).

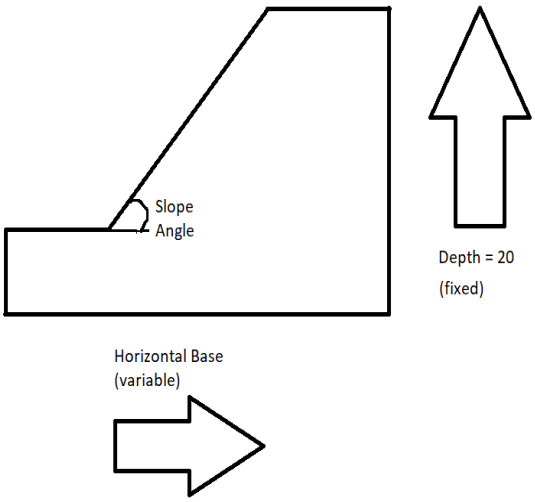
Ultimately this model can be deployed onto a mobile based application making it much easier to use in the field and in the labs for preliminary analysis of slopes.

Data Collection

The dataset was generated using the FEA software OptumG2. OptumG2 allows the user to design a soil structure with user defined material properties, loading characteristics and geometric designs. After designing the required slope, the user can run failure analysis using methods like Strength Reduction or Limit Analysis and the software's algorithm predicts the failure plane as well as the displacement of the slope during failure.

A dataset of 65 observations with varying input parameters was generated. **It is to be noted that, slope dataset was based on the Mohr-Coulomb materials. Also, the slope designs were undrained with water table assumed to be much below the base of the slope and the slopes were made of a uniform material. Moreover, the slope failure conditions were analysed on the Standard Fixities (no external load on the slope structure apart from the natural forces of gravitation).**

The data set was created taking the maximum height/depth of as constant (equal to 20 units), while the horizontal distance between the head and the toe of the slope was varied for different angles.



An illustration of the geometric considerations for the slope samples

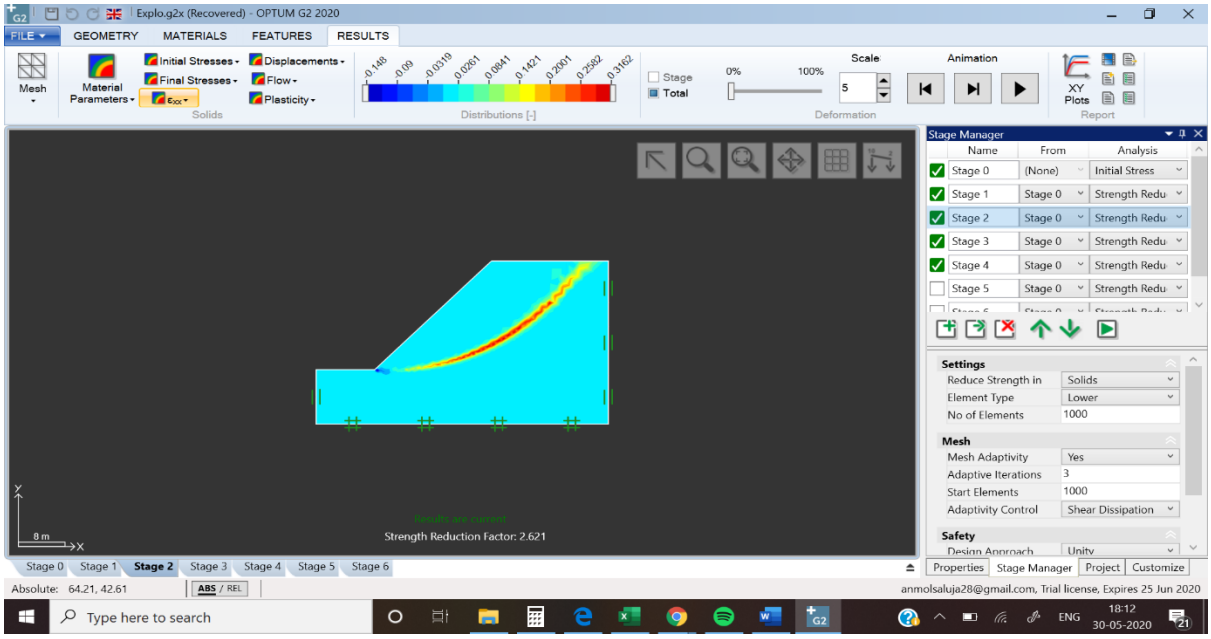
Parameter Selection –

After running sensitivity tests, 5 parameters for model training were selected. These are –

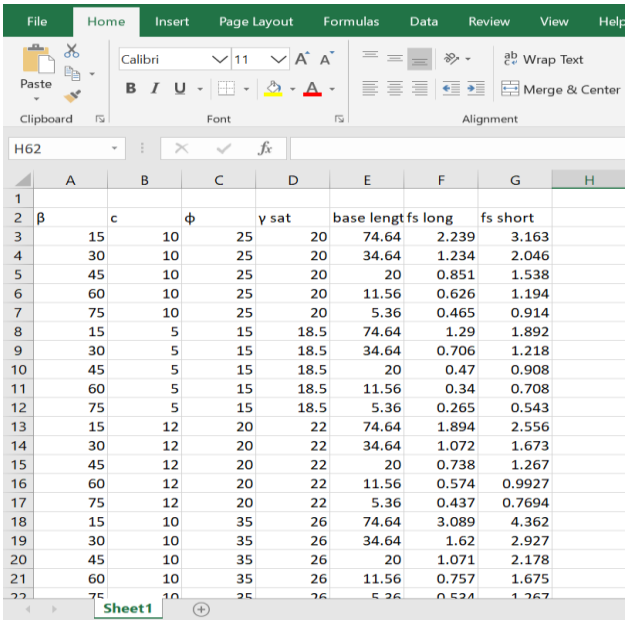
- Angle of Slope (β)
- Cohesion of Soil (c)
- Friction Angle of Soil (Φ)
- Saturated Unit Weight (γ sat)
- Horizontal distance between head and toe of the slope (b)

Meanwhile, 2 kinds of target parameters were considered –

- Long Term Factor of Safety
- Short Term Factor of Safety



A snippet of OptumG2



	A	B	C	D	E	F	G
1							
2	β	c	ϕ	γ sat	base lengt	fs long	fs short
3	15	10	25	20	74.64	2.239	3.163
4	30	10	25	20	34.64	1.234	2.046
5	45	10	25	20	20	0.851	1.538
6	60	10	25	20	11.56	0.626	1.194
7	75	10	25	20	5.36	0.465	0.914
8	15	5	15	18.5	74.64	1.29	1.892
9	30	5	15	18.5	34.64	0.706	1.218
10	45	5	15	18.5	20	0.47	0.908
11	60	5	15	18.5	11.56	0.34	0.708
12	75	5	15	18.5	5.36	0.265	0.543
13	15	12	20	22	74.64	1.894	2.556
14	30	12	20	22	34.64	1.072	1.673
15	45	12	20	22	20	0.738	1.267
16	60	12	20	22	11.56	0.574	0.9927
17	75	12	20	22	5.36	0.437	0.7694
18	15	10	35	26	74.64	3.089	4.362
19	30	10	35	26	34.64	1.62	2.927
20	45	10	35	26	20	1.071	2.178
21	60	10	35	26	11.56	0.757	1.675

A snippet of the generated dataset

Training and Test Spilt –

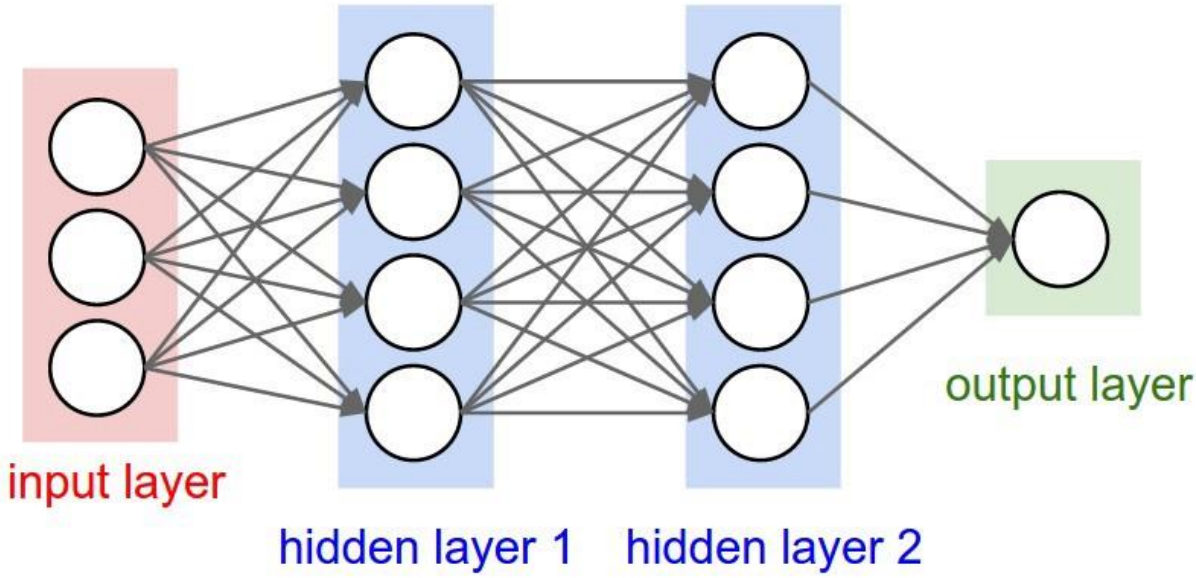
The dataset was split into a 70% training and 30% validation set. Python's scikit learn library was used for the same.

Models Used –

1. **3 Layered Artificial Neural Network –** ANNs constitute the bed rock of the field of Deep Learning. Artificial neural networks are a computational model

whose architecture is modelled after the brain. In other words, it is a computational system inspired by the structure, processing method and learning ability of a biological brain. They typically consist of many hundreds of simple processing units or **Full Connected Layers** which are wired together in a complex communication network. Each unit or node is a simplified model of a real neuron which fires (sends off a new signal) if it receives an input signal from the other nodes to which it is connected.

Here we have employed a 3 layered neural network with fairly basic approach since the variance in our dataset is not high. A logical representation for the same is this –



An illustration of a 3 Layered Neural Network

Our model structure is the following -

- **An input layer of 5 (apart from the bias) parameters or neurons.**
- **2 subsequent fully connected hidden layers of 64 units or neurons each with "relu" activation.** "relu" is a non-linearity operator which is helpful for the model to capture the non-linearity of the data.
- **An output layer of a single output unit.** This unit will output the predicted target variable for input observations.

The **loss function,** which highlights the difference between our predicted value and the actual value for an observation, for this model was the **Mean Squared Error and Mean Absolute Error** (separately) and it was minimized or **optimized** using the **Adam optimization** technique.

The model was performed using TensorFlow v.2.10, a numerical Python library which is vastly used for its simple and effective ML tools.

```
model = Sequential()
model.add(Dense(64,activation='relu',input_dim = 4))
model.add(Dense(64,activation = 'relu'))
model.add(Dense(1))
model.compile(loss = 'mean_squared_error',optimizer = 'Adam')
history = model.fit(X_train,y_train,validation_data = (X_test,y_test),epochs = 500)

model_2 = Sequential()
model_2.add(Dense(64,activation='relu',input_dim = 4))
model_2.add(Dense(64,activation = 'relu'))
model_2.add(Dense(1))
model_2.compile(loss = 'mean_absolute_error',optimizer = 'Adam')
history_2 = model_2.fit(X_train,y_train,validation_data = (X_test,y_test),epochs = 500)
```

```
Epoch 1/500
25/25 [==============================] - 0s 3ms/step - loss: 35.7495 - val_loss: 25.9986
Epoch 2/500
25/25 [==============================] - 0s 226us/step - loss: 21.3060 - val_loss: 14.8225
Epoch 3/500
25/25 [==============================] - 0s 174us/step - loss: 12.2234 - val_loss: 6.8265
Epoch 4/500
25/25 [==============================] - 0s 159us/step - loss: 5.6941 - val_loss: 2.1095
```

A snippet of ANN model

2. **Support Vector Machine -** Support vector machine (SVM) is one of the widely-employed machine learning algorithms which are aimed at detecting the decision boundary in order to separate different classes. Due to their outstanding presentation to deal with examples of the non-separable and high-dimensional data sets, SVMs have been effectively applied as a reliable solution for many classification problems. Here we will be using Support Vector Regression (SVR). During SVR learning, it is assumed that there is a unique relationship between each set of input-target pairs. Grouping and classifying the relation of these predictors will conduce to produce the system outputs (e.g., slope safety factor in this study). Unlike many predictive models that try to minimize the calculated error (i.e., the difference between the target and system outputs), SVR aims to improve its performance by optimizing and altering the generalization bounds for a regression.

   SVR model has been implemented using a scikit learn, another famous mathematical library used for its ML tools.

   The model has used a **rbf kernel** and has been tuned on its hyperparameters, **C and gamma**, using the **Grid Search Cross-validation**.

```
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
parameters = [{'kernel': ['rbf'], 'gamma': [1e-4, 1e-3, 0.01, 0.1, 0.2, 0.5, 0.6, 0.9],'C': [1, 10, 100, 1000, 10000]}]
svr = GridSearchCV(SVR(epsilon = 0.01), parameters, cv = 5)
svr.fit(X, y)
means = svr.cv_results_['mean_test_score']
stds = svr.cv_results_['std_test_score']
print(svr.best_params_)
print(svr.best_score_)
```

A snippet of the SVR code

3. **Multiple Linear Regression –** The goal of a multiple linear regression (MLR) model is to establish a linear equation to the data samples to reveal the relationship between two or more independent (explanatory) variables and a dependent (response) variable. The overall structure of the MLR formula is shown by this equation:

$$y = \alpha_0 + \alpha_1 * x_1 + \alpha_2 * x_2 + \ldots + \alpha_s * x_s$$

In the above formula, y and x represent the dependent and independent variables, respectively. The terms α0, α1, . . . αs are indicative of MLR unknown parameters. The goal here is to find those parameters ($\alpha_0, \alpha_1 \ldots \alpha_s$) which are closest to predicting the actual FOS. We can optimize them by minimizing the squared difference between the predicted and actual target labels. This can be achieved by various optimization algorithms like Stochastic Gradient Descent or Adam Optimization.

```
[ ]  from sklearn.linear_model import LinearRegression
     reg = LinearRegression(normalize = True)
     reg.fit(X_train,y_train)
     y_pred = reg.predict(X_test)
     error = mean_absolute_error(y_test,y_pred)
     error
```

A snippet of the MLR code

4. **Random Forest Regression –** A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called **Bootstrap Aggregation**, commonly known as **bagging**. Bagging, in the Random Forest method, involves training each decision tree on a different data sample where sampling is done with replacement. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

Similar to SVR and MLR, Random Forest model was also built using the scikit learn library. Two major hyperparameters, **maximum depth of trees and number of estimators**, were optimized using Grid Search Cross-Validation.

```
from sklearn.ensemble import RandomForestRegressor
# Number of trees in random forest
n_estimators = [x for x in range(200,300,10)]
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)

parameters = {'n_estimators': n_estimators,
              'max_depth': max_depth,
              }

rf = RandomForestRegressor()
rf_random = GridSearchCV(rf,  random_grid, cv = 5)

# Fit the random search model
rf_random.fit(X,y)
```

A snippet of Random Forest Regression Code

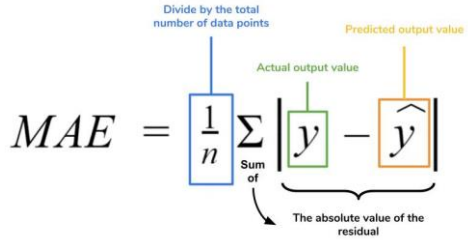
Model Performance Analysis

Each of the 4 models was judged on the basis of some important statistical metrics for regression models which are –

- **Mean Squared Error (MSE) –** It is simply the average of all the squared differences between the target values and the values predicted by the regression model. As it squares the differences and thus yielding always a positive value, it penalizes even a small error which leads to over-estimation of how bad the model is. It is preferred more than other metrics because it is differentiable and hence can be optimized better.

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

- **Mean Absolute Error (MAE) –** MAE is the absolute difference between the target value and the value predicted by the model. The MAE is more robust if the dataset contains outliers and does not penalize the errors as extremely as MSE. MAE is a linear score which means all the individual differences are weighted equally.



$$MAE = \frac{1}{n}\sum \left| y - \hat{y} \right|$$

- **Root Mean Squared Error (RMSE) –** It is the squared root of the averaged squared difference between the target value and the value predicted by the model. It is preferred more in some cases because the errors are first squared before averaging which poses a high penalty on large errors. This implies that RMSE is useful when large errors are undesired.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

- **R-squared (R^2) –** R-squared (R^2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. The metric helps us to compare our current model with a constant baseline and tells us how much our model is better. The constant baseline is chosen by taking the mean of the data and drawing a line at the mean.

$$R^2 = 1 - \frac{\text{MSE(model)}}{\text{MSE(baseline)}}$$

- **Adjusted R^2** - R^2 suffers from the problem that the scores improve on increasing the number of variables even though the model is not improving which may misguide in judging the performance. Adjusted R^2 is always lower than R^2 as it adjusts for the increasing predictors and only shows improvement if there is a real improvement. However, since we haven't changed the number of parameters, we won't be using this metric for the evaluation of our models.

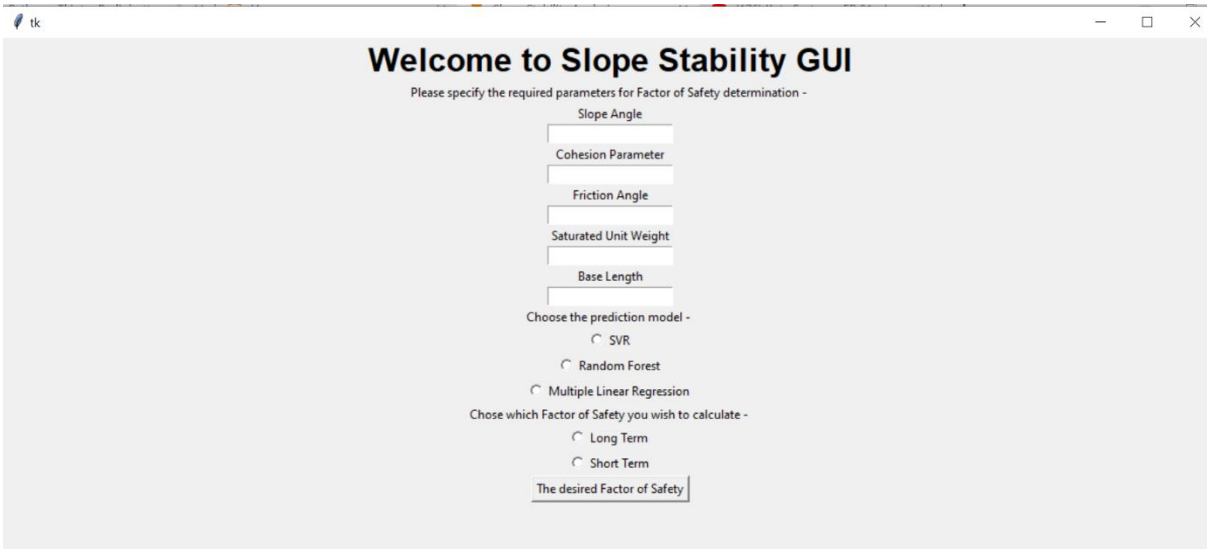
Output Metrics

For **long term** Factor of Safety evaluation on the **test set** -

Model	MSE	MAE	RMSE	R^2
SVR	0.190	0.250	0.436	0.776
MLR	1.170	0.857	1.082	-0.376
RAF	0.547	0.430	0.740	0.355
ANN (loss=mae)	0.013	0.018	0.116	0.678
ANN (loss=mse)	0.004	0.010	0.068	0.889

GUI Development

In order to make this ML approach more handy and easier to understand and use, a Graphical User Interface (GUI) has been created where the user can input parameters and receive the desired output. This interface has been created using Python's Tkinter GUI library.



A snippet of the Python GUI

Conclusion

In this project, I explored the use of 4 supervised machine learning algorithms (Multi-layered ANN, SVR, MLR and RAF) in predicting the Factor of Safety, a metric used extensively in analysing the stability of natural and artificial slopes. A dataset of 65 observations was generated using FEA software OptumG2. Four input parameters were chosen for training and validation. The performance of each model was evaluated using different accuracy metrics. Finally, a GUI was created for better user interaction.

Future Use Cases

Machine Learning has been a pioneering development of computer science since the last 2 decades. And with the rise of modern and powerful computers, its ability to learn from a pre-existing data resource and use it for unknown or future analysis will only rise. As we can observe, even with a relatively small dataset the models were able to extract valuable insights from the it and were able to predict the Factor of Safety with great accuracy. This mechanism, if converted into a mobile application, can be used by the on-field engineers for preliminary assessment of the slope.

Moreover, by predicting the FOS using just 5 parameters is merely scratching the surface. With much larger datasets and more variables, one can predict more

properties (like effective-stress at any point, etc.) of the slopes with a high accuracy in limited time.

In future, one can even go to the extent of predicting failure of slopes just by analysing the images of the slope structure. This can be achieved using the modern Image Processing Techniques of Object Detection using Convolutional Neural Networks (CNNs) and is a great path for future research.

All in all, the use of Artificial Intelligence and ML can simplify the complex and mechanical analysis and can pave way for highly accurate results in a fraction of time.

References

- *Prediction of Slope Stability Using Four Supervised Learning Methods by YUN LIN , KEPING ZHOU , AND JIELIN LI* https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8383970
- *Predicting Slope Stability Failure through Machine Learning Paradigms by Dieu Tien Bui, Hossein Moayedi, Mesut Gör, Abolfazl Jaafari and Loke Kok Foong*
- *The Stability of Slopes* https://people.eng.unimelb.edu.au/stsy/geomechanics_text/Ch11_Slope.pdf
- https://www.slideshare.net/PallaviBadry/slope-stability-74922154
- *An Overview on Methods for Slope Stability Analysis by Mr. Digvijay P. Salunkhe, Ms. Rupa N. Bartakke, Assist. Prof. Guruprasd Chvan, Ms. Pooja R Kothavale*
- *Slope Stability Prediction using Artificial Neural Network (ANN) by Arunav Chakraborty and Dr. Diganta Goswami*
- https://medium.com/@gkadusumilli/gui-programming-using-python-tkinter-package-8e7f67fa4637
- *Regression: An Explanation of Regression Metrics And What Can Go Wrong* https://towardsdatascience.com/regression-an-explanation-of-regression-metrics-and-what-can-go-wrong-a39a9793d914
- *SOIL AND FOUNDATION ENGINEERING BY Dr. K.R ARORA*