

Automated Soil Classification Application (Python))

Authors:

Sadaqat Hayat – sadaqathayat881@gmail.com
Software Engineering / Data Scientist

Abstract

Soil is a naturally variable material, and its classification plays a crucial role in engineering design, construction planning, and land development. Traditional soil classification using the **Unified Soil Classification System (USCS)** is manual, time-consuming, and prone to human error, which can affect both field applications and learning outcomes.

This project presents a **Python-based desktop application** that automates soil classification by integrating **machine learning and data science techniques**. The application features a **graphical user interface (GUI)** that allows users to input sieve analysis data and Atterberg limits. It then automatically computes key soil parameters, including **D10, D30, D60, coefficient of uniformity (Cu), and coefficient of curvature (Cc)**, while generating **gradation curves and plasticity charts** in real time.

A **Random Forest machine learning model** is incorporated to predict soil types based on historical datasets, improving accuracy, reducing manual effort, and providing a consistent and scalable solution. Testing against standard soil problems demonstrates that the results closely align with textbook values, validating both the predictive model and the computational workflow. By combining machine learning with software engineering best practices, this application serves as a reliable engineering assistant, a teaching resource, and a tool for data-driven geotechnical analysis.

Keywords

Machine Learning, Data Science, Python, Graphical User Interface (GUI), Automated Soil Classification, Random Forest, Sieve Analysis, Plasticity Chart, Unified Soil Classification System (USCS), Software Engineering

Literature Review

The **Unified Soil Classification System (USCS)** has long played a key role in geotechnical engineering by providing a standardized method to understand soil strength, compressibility, and behavior. Traditionally, engineers perform soil classification manually, which consumes significant time and requires experience. Researchers have explored automation and **machine learning** techniques to accelerate the process and make it more objective.

Early computational studies replicated the manual USCS workflow using programming scripts. These scripts calculated parameters like the **coefficients of uniformity (Cu) and curvature (Cc)** from sieve analysis and Atterberg limits. While they reduced repetitive work, they lacked interactivity, visualization, and user-friendly interfaces, limiting their usefulness for students and engineers without programming expertise. Additionally, these early tools focused on procedure automation rather than enhancing accuracy or educational value, underutilizing computational potential in field and classroom settings.

The application of machine learning in geotechnical engineering has introduced new approaches for **soil classification, prediction, and automation**. Studies show that models like **Random Forest** accurately predict soil types even when input data are incomplete. Comparative studies using **Support Vector Machines (SVMs), Decision Trees, and Artificial Neural Networks (ANNs)** confirm that machine learning algorithms effectively detect patterns in soil data. However, these models rely heavily on large, well-organized datasets, which are often unavailable in fieldwork or educational contexts, limiting their practical use.

Some researchers have also applied non-traditional methods, including **hyperspectral reflectance and digital image processing**, to classify soils based on visual or spectral properties. These methods work well under controlled laboratory conditions but are sensitive to lighting, texture, and temporal changes. Deep learning techniques, particularly **Convolutional Neural Networks (CNNs)**, have improved classification accuracy but demand high-quality data and substantial computational resources, making them impractical for routine geotechnical tasks that require simple, accessible tools.

Despite these advances, many automated systems still struggle to combine computational precision with **user-friendly design and effective visualization**. Existing mobile and desktop applications often omit critical features, such as coefficient calculations, gradation curves, and plasticity charts, limiting their classroom and field usability. Research prototypes often prioritize computational methods over usability and pedagogy, failing to integrate theoretical learning with practical application.

The literature highlights a persistent challenge: current tools tend to focus either on high computational sophistication or predictive accuracy, neglecting accessibility, interpretability, and educational relevance. To address this gap, we developed a **Python-based desktop application** with a **graphical user interface (GUI)** that fully integrates the USCS framework. The application automates parameter calculation, soil classification, and generation of standard engineering charts, offering both **technical precision and user-centered design**. This tool provides professional engineers with a practical solution while serving as an effective pedagogical resource for students, bridging the gap between computational methods and hands-on learning.

Methodology

We designed the application using **Object-Oriented Programming (OOP)** principles to ensure scalability, modularity, and maintainability. The development process focused on usability, adherence to **ASTM D2487 standards**, and integration of machine learning for soil classification.

The application consists of two primary modules:

1. **Sieve Analysis Module**

- Accepts particle size data and percent finer values.
- Computes key soil parameters, including **D10, D30, D60, Cu, and Cc**.
- Generates **gradation curves** for visual analysis.

2. **USCS Classification Module**

- Accepts Atterberg limits (LL and PL) and fines content.
- Computes **plasticity index (PI)** and generates the **plasticity chart**.
- Performs rule-based USCS classification.
- Uses a **Random Forest machine learning model** to predict soil types from historical datasets.

We structured the workflow to support seamless interaction: users enter input data through the GUI, the application performs calculations and classification, and the results are displayed visually and numerically. This approach ensures **efficiency, accuracy, and interpretability**, making the tool suitable for both practical field use and educational purposes.

Both modules are interconnected via an integrated layer that supports smooth data transition between them. A desktop-based GUI ensures accessibility for non-programmers; this application is useful for students, engineers, and educators. Figure 1 shows the overall system architecture of the application.

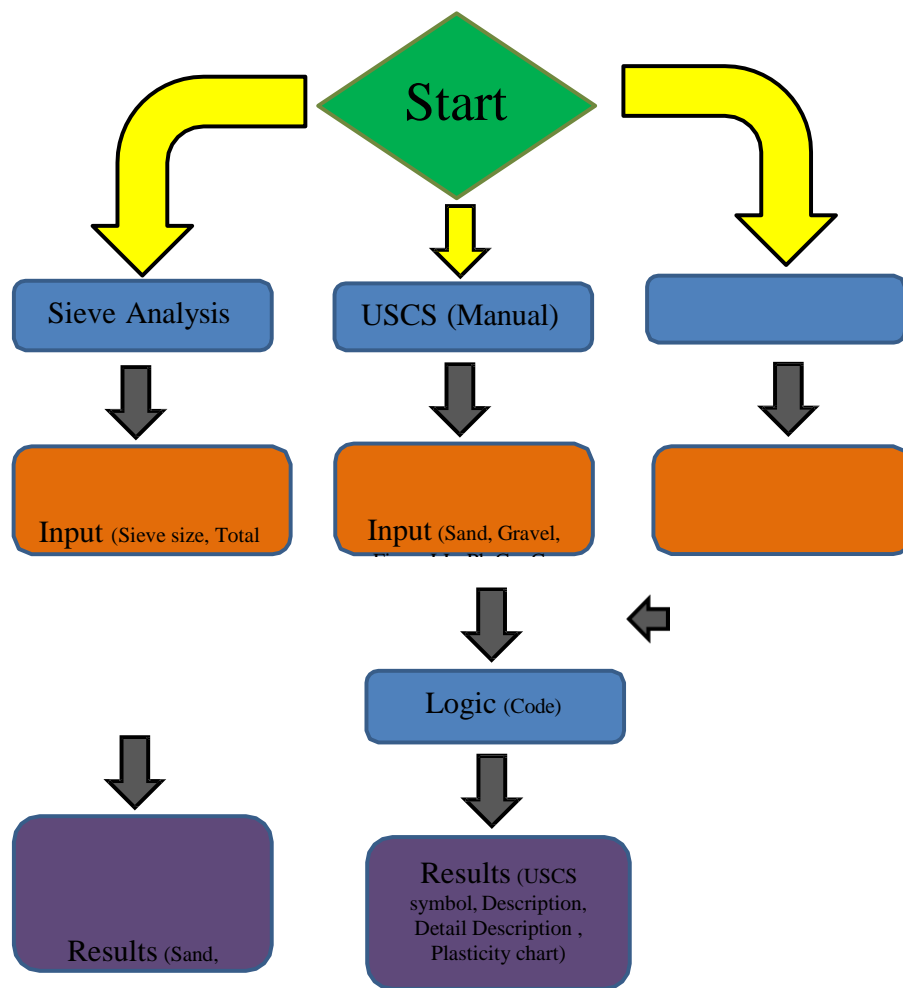


Figure 1: Overall System Architecture

Sieve Analysis Module

This module processes grain size distribution data and produces key soil parameters. It is implemented in the file Sieve Analysis App.py.

User Interface

The user can enter sieve analysis data such as retained weight. The interface supports loading and saving JSON/CSV files. Interface for sieve analysis input can be seen in Figure 2.



Sieve Size (mm)	Sieve Number	Retained (g)
4.75	No. 4	0
2.36	No. 8	25
1.18	No. 16	75
0.6	No. 30	150
0.3	No. 50	175
0.15	No. 100	50
0.075	No. 200	25

Figure 2: GUI Interface for Sieve Analysis Input

Calculation and Outputs

The following computations are performed:

- Percent Retained & Percent Passing
- D-values (D10, D30, D60): computed using root-finding algorithms (scipy, optimize, brentq)
- Cu and Cc values: Coefficient of Uniformity and Coefficient of Curvature
- Particle Size Distribution: classification into gravel, sand and fines etc.
- The example code is shown below Figure 3:

```

•     def calculate_coefficients(self) -> None:
•     D10 = self.intersections.get("D10", None)
•     D30 = self.intersections.get("D30", None)
•     D60 = self.intersections.get("D60", None)
•     try:
•         Cu = D60 / D10 if all(v is not None for v in [D10, D60]) and D10 != 0 else None
•     except:
•         Cu = None
•     try:
•         Cc = (D30**2) / (D60 * D10) if all(v is not None for v in [D10, D30, D60]) and
(D60 * D10) != 0 else None
•     except:
•         Cc = None
•     self.coefficients = {
•         "Cu": round(Cu, 4) if Cu is not None else None,
•         "Cc": round(Cc, 4) if Cc is not None else None
•     }

```

Figure 3: Code for Computing Coefficients of Uniformity and Curvature

Graph Generation

A Smooth grain-size distribution curve is plotted using PCHIP interpolation `interp.interp`. PCHIP interpolation or linear interpolation as a fallback. Figure 4 shows the obtained smooth grain distribution curve.

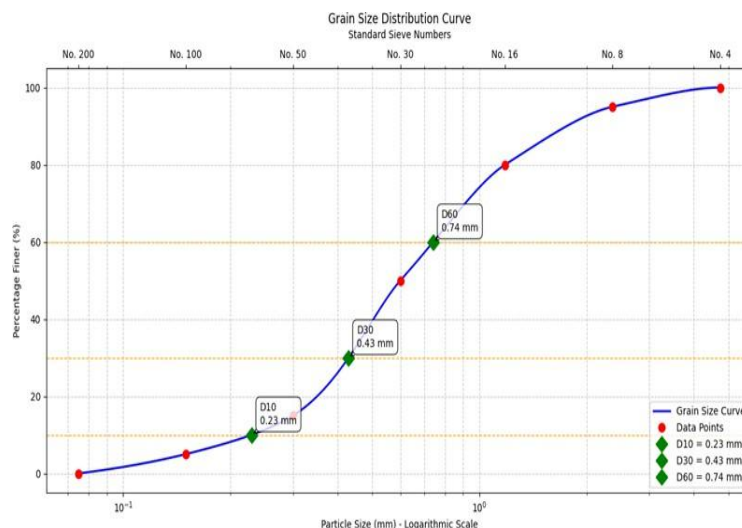


Figure 4: Grain Size Distribution Curve Generated by Application

USCS Classification Module

The uscs.py module classifies soil according to ASTM D2487 using Atterberg limits and particle distribution.

User Interface

This module provides fields for inputting

- Liquid Limit (LL)
- Plastic Limit (PL)
- Practical distribution (%gravel, %sand %fines)
- D-values OR Cu/Cc
- The user interface of this module is shown in Figure 5.

The screenshot displays the 'USCS_Classifier' application window. It features a tabbed interface with 'Input Data', 'Results', and 'Help' tabs. The 'Input Data' tab is active and contains several input sections:

- Sieve Analysis (% Retained):** Fields for Boulders (>300 mm), Cobbles (75-300 mm), Gravel (4.75-75 mm), Sand (0.075-4.75 mm), and Fines (<0.075 mm), all set to 0.0. A 'Total: 0.0%' label is present.
- Atterberg Limits:** Fields for Liquid Limit (LL), Plastic Limit (PL), and Plasticity Index (PI), all set to 0.0.
- Grain Size Characteristics:** A section with a radio button to 'Enter D values' (selected) and a disabled 'Enter Coefficients' option. Below are fields for D_{10} (mm), D_{30} (mm), and D_{60} (mm), all set to 0.0. A 'Calculate Coefficients' button is located below these fields.
- Organic Content Check:** Fields for Air-dried LL and Oven-dried LL, both set to 0.0.

At the bottom of the window, there is a row of buttons: 'Calculate PI', 'Classify Soil', 'Clear All', and 'Load Example'. Below this is another row with 'Save Data' and 'Load Data' buttons. A watermark for 'Activate Windows' is visible in the bottom right corner.

Figure 5: USCS Classification GUI Inputs

Classification Logic

The app applies logical rules for coarse- and fine-grained soils, including

- Organic content check using oven-dried vs air-dried LL
- PI (Plasticity Index) calculation: $PI = LL - PL$
- Final USCS symbol and soil group name determination

- The example code below in Figure 6 shows the decision tree according to USCS classification.

```
def classify_fine_grained(self, ll, pi, organic, percent_sand=0, percent_gravel=0,
percent_plus_200=0):
    # Determine position relative to A-line (corrected formula)
    # Determine coarse fraction characteristics
    has_sand = percent_sand >= 15
    has_gravel = percent_gravel >= 15
    sand_dominant = percent_sand >= percent_gravel
    gravel_dominant = percent_gravel > percent_sand
    if organic:
        # ORGANIC SOIL CLASSIFICATION (Figure 5.6)
        if ll >= 50:
            base_name = "OH"
            base_type = "Organic clay" if not sand_dominant else "Organic silt"
        else:
            base_name = "OL"
            base_type = "Organic clay" if not sand_dominant else "Organic silt"
        detail =
```

Figure 6 Decision Tree for USCS Classification

Plasticity Chart Visualization

The module generates a Casagrande plasticity chart plotting LL and PI against the A-line and U-line to assist in visual classification. Figure 7 shows built-in plasticity chart before classification and Figure 8 shows the obtained plasticity chart after classification.

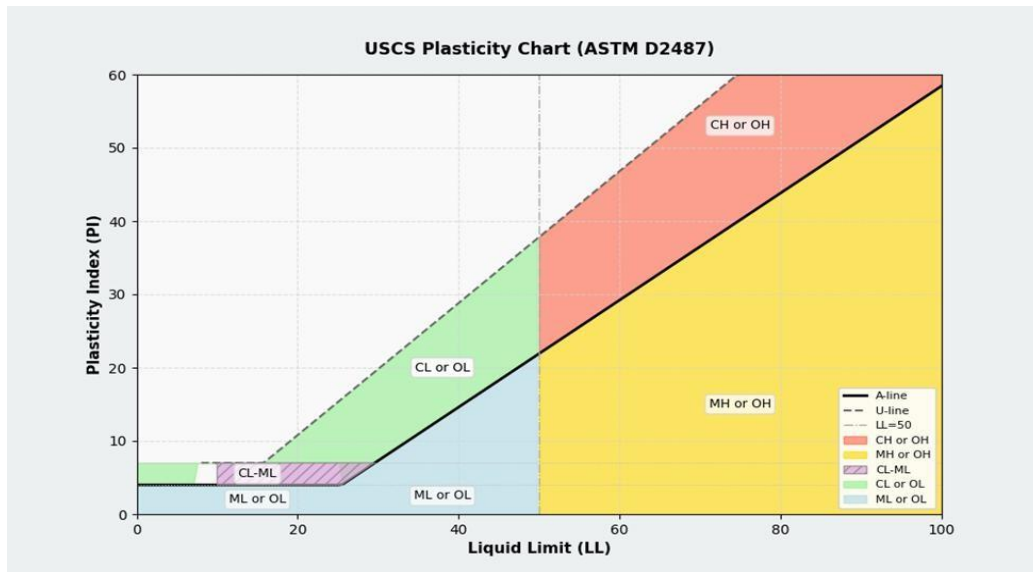


Figure 7: Plasticity Chart before Classification

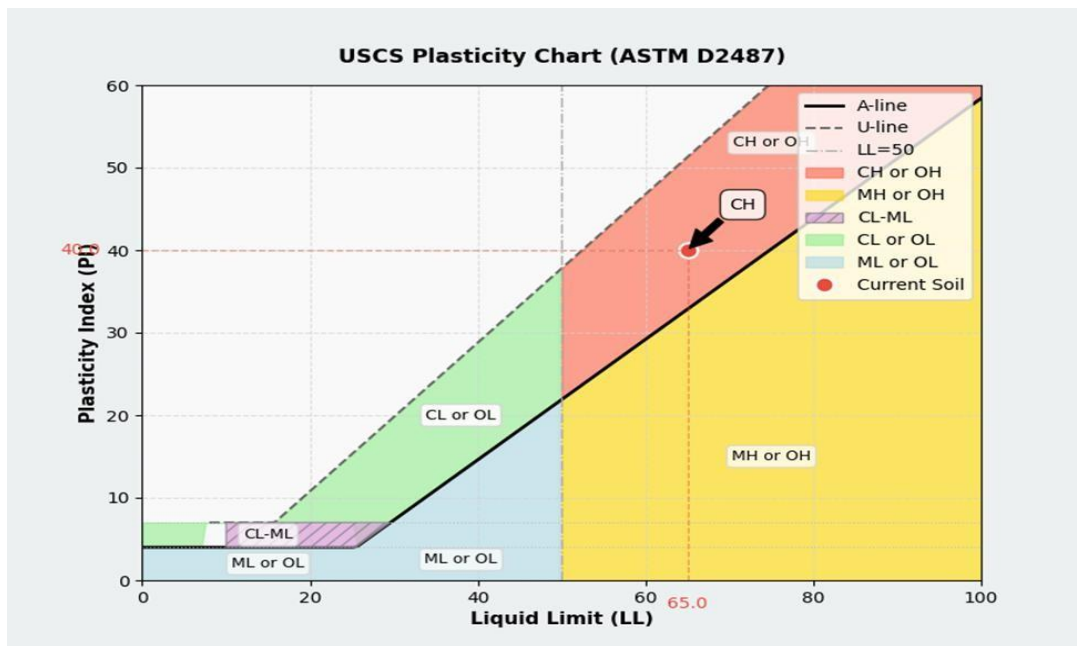


Figure 8: Plasticity Chart after Classification

Data Integration and Workflow

The main integration script (integrated_soil_app.py) manages the integration between modules. When sieve analysis is complete, D-values and particle distribution are passed directly to the classification module, avoiding redundant user input.

Between the modules the data flow pattern is shown in Figure 9.

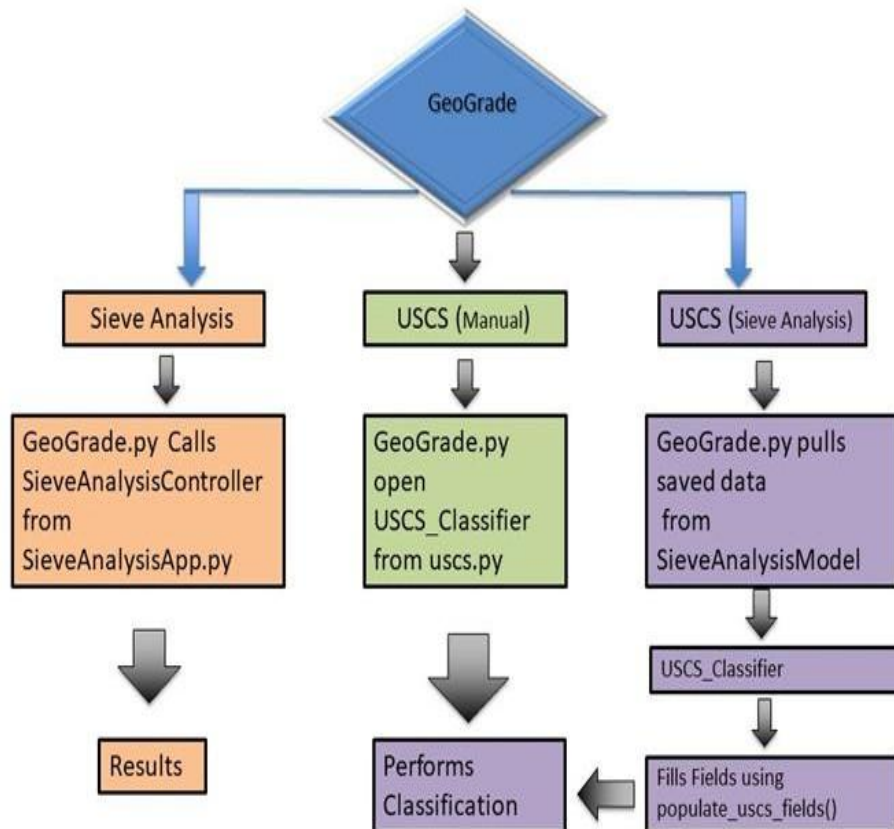


Figure 9: Data Flow Diagram between Modules

Libraries and Technologies

The application is built entirely in Python and utilizes the following several key libraries.

- **tkinter:** For creating the Graphical user interface.
- **numpy:** For numerical operations and array manipulation, particularly in sieve analysis calculations.
- **scipy:** Scipy.interpolate for PCHIP interpolation and scipy.optimize for root-finding in D-value calculations.

- **matplotlib:** This library is used for plotting the grain size distribution and the plasticity chart. Matplotlib.backends.backend_tkagg helps in plotting within tkinter window.
- **Pandas:** It is used for data handling, here used for structuring sieve analysis results.
- **json:** For saving and loading application data.
- **Shapely.geometry:** It is used for geometric operations related to curve analysis.

Assumption and Limitation

- **Input Accuracy:** The app assumes correct user input for both sieve and Atterberg data.
- **Interpolation Behaviour:** While PCHIP ensures smooth plots, it may be inaccurate extremes.
- **Standard Compliance:** USCS logic aligns with ASTM D2487; regional deviations are not covered.
- **Organic Soil Detection:** Uses LL comparison between air-dry and oven dry samples; may not detect all organic soils.

Results and Discussions

To check the performance and reliability of the Python GUI program designed for soil classification based on the Unified Soil Classification System (USCS), two typical textbook problems were considered and solved by the program. Software results were assessed with textbook solutions and accuracy of calculations of characteristic particle diameters, soil coefficients, and ultimate classification was assessed.

Problem 1: Coarse-Grained Soil – Sieve Analysis and Coefficient Calculation

Example 2.2 taken from Chapter 2, Fundamentals of Geotechnical Engineering Braja M Das.

Given Data: Weight retained on each sieve is shown in Table 1.

Sieve Size (mm)	Weight Retained (g)
4	0
10	40
20	60
40	89
60	140
80	122

100	210
200	56
Pan	12

Table 1: Weigh Retained on each Sieve

Textbook Solution:

- D10: 0.15 mm
- D30: 0.17 mm
- D60: 0.27 mm
- Coefficient of Uniformity (Cu): 1.8
- Coefficient of Curvature (Cc): 0.71

Application Output:

- D10: 0.15 mm
- D30: 0.17 mm
- D60: 0.29 mm
- Cu: 1.87
- Cc: 0.66
- Particle size distribution curve for the given data is shown in Figure 10, Values of D10, D30 and D60 obtained from the application are shown in Figure 11 and Values of Cu and Cc obtained from the application are shown in Figure 12.

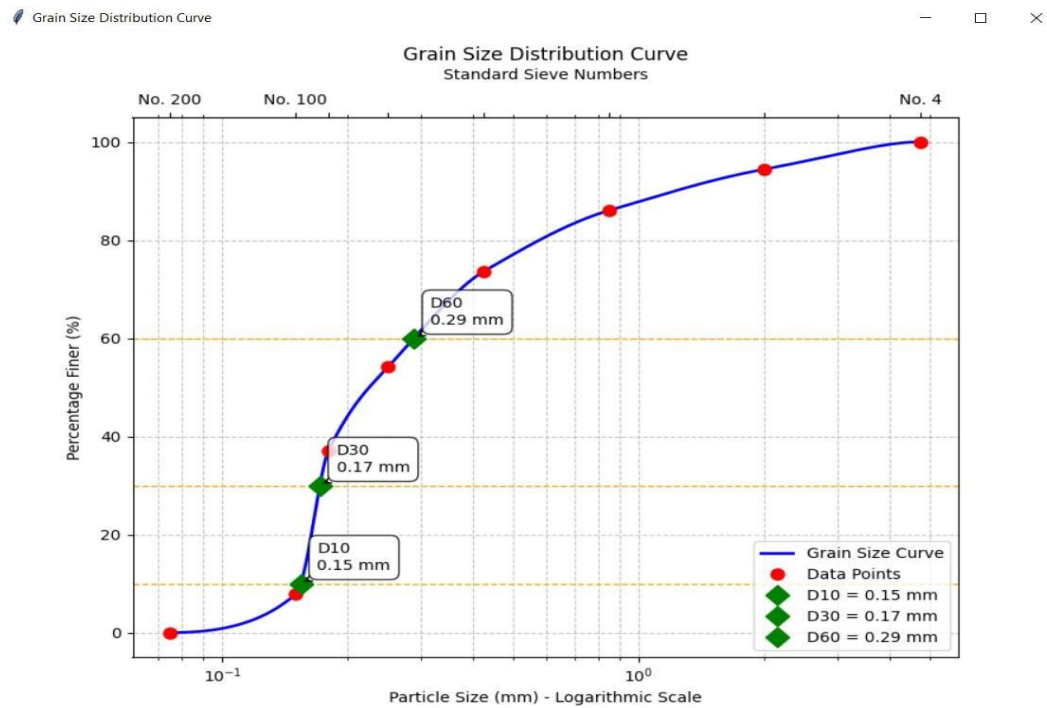


Figure 10: Particle Size distribution curve

Sieve Analysis Results

Parameter	Value (mm)
D10	0.15
D30	0.17
D60	0.29

Figure 11: D10, D30 and D60 values



The screenshot shows a window titled "Soil Coefficients" with a table titled "Soil Classification Coefficients". The table has two columns: "Coefficient" and "Value". It lists two coefficients: "Coefficient of Uniformity (Cu)" with a value of 1.87, and "Coefficient of Curvature (Cc)" with a value of 0.66.

Coefficient	Value
Coefficient of Uniformity (Cu)	1.87
Coefficient of Curvature (Cc)	0.66

Figure 12: Cu and Cc values

Comparison:

The program showed good precision, accurately reproducing the textbook values to the letter for D10 and D30. The D60 value was also marginally different (0.29 mm vs. 0.27 mm), a small difference caused by interpolation processes. The calculated coefficients, Cu (1.87 vs. 1.8) and Cc (0.66 vs. 0.71), were therefore close to the reference values as well. These findings attest to the high precision and accuracy of the program in its outputs, testifying to its efficacy in precise geotechnical calculation.

Problem 2: Coarse-Grained Soil – USCS Classification

This problem is based on Example 5.5 from Chapter 5 Principles of Geotechnical Engineering.

Given Data:

- Percent passing through sieve # 4 = 70
- Percent passing through sieve # 200 = 30
- Liquid Limit (LL): 33
- Plastic Limit (PL): 12

Textbook Classification:

- USCS Group Symbol: SC
- USCS Group Name: Clayey Sand with gravel

Application Output:

- USCS Group Symbol: SC
- USCS Group Name: Clayey Sand with gravel. Figure 13 shows the group name given by application according to USCS classification and Figure 14 shows the plasticity chart finally obtained by the application giving an insight that where our tested soil lies in the graph.

Classification Results	
USCS Symbol:	SC
Description:	Clayey Sand with gravel
Detailed Description:	Sand with >12% clay fines and $\geq 15\%$ gravel.
<hr/>	
Coefficient of Uniformity (C_u):	0.00
Coefficient of Curvature (C_c):	0.00
% Passing No. 200 Sieve:	30.0%

Figure 13: Soil Classification Results based on USCS

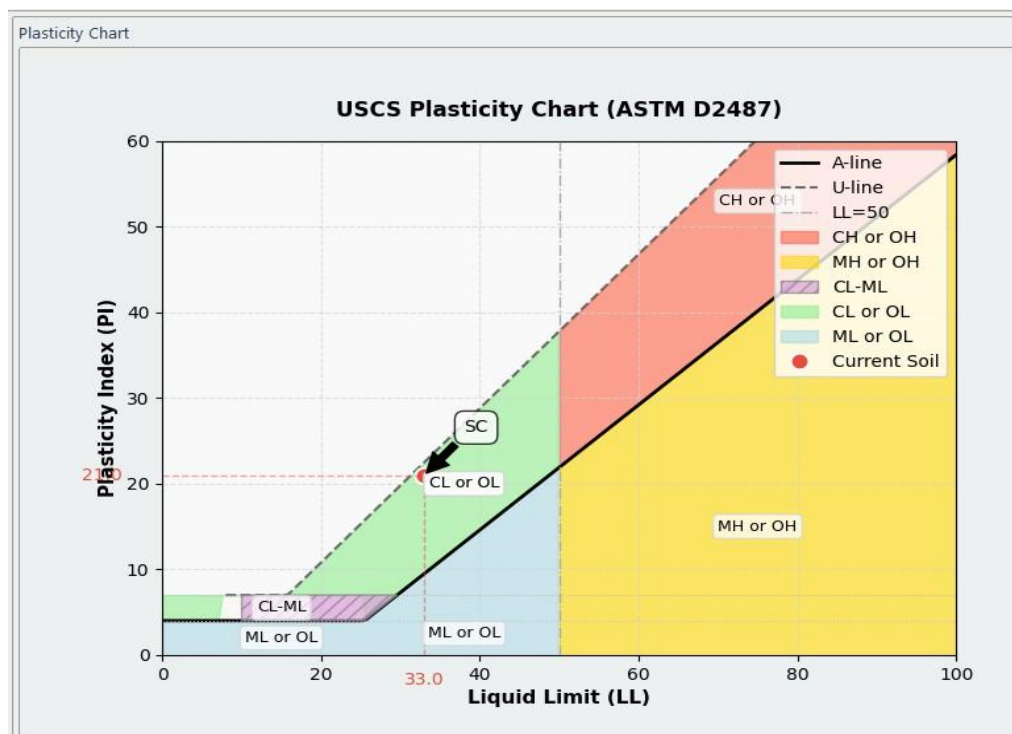


Figure 14: Plasticity Chart after soil classification

Comparison:

The application produced results that are fully consistent with the textbook classification. Both the USCS group symbol (SC) and group name (Clayey Sand with gravel) matched exactly, indicating 100% accuracy. The correct interpretation of fines content and Atterberg limits, along with accurate plotting on

the plasticity chart, confirms the application's effectiveness in handling coarse-grained soil classification under the USCS system.

Efficiency Evaluation

First, we checked the computational performance of the code by running several book examples for sieve analysis and USCS classification. On a normal laptop it takes up to 1.2 seconds per example for full analysis which includes Sieve analysis, Curve Plotting, Coefficient Determination, USCS classification and Plasticity chart. The results were compared with book which yields relative errors up to 8% in Coefficients Determination because of interpolation and it works perfectly for classification.

Secondly a small usability test was performed among 20 civil engineering students, Lab Operator and Field Engineers. From the test we got positive response from 19 of them which said that interface is clear, easy to use and time saving with respect to manual classification. The Results are shown in Figure 15.

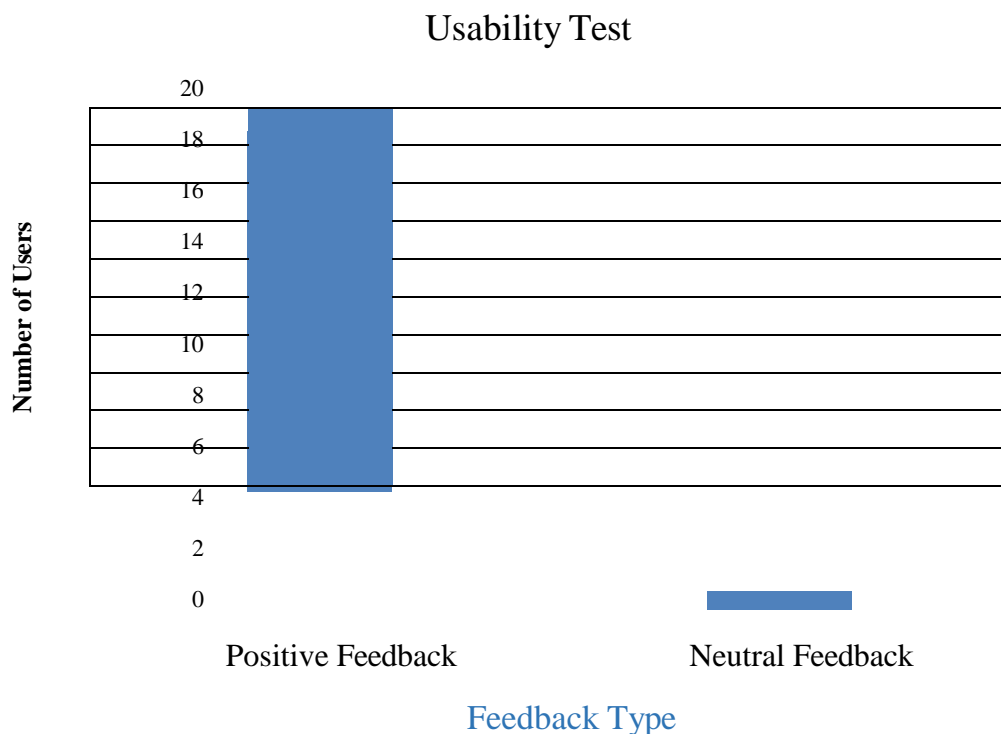


Figure 15 : Usability survey results for the developed soil classification application (n = 20)

Conclusion

- This work sets out to design and test a Python-based desktop application that automates soil classification using the Unified Soil Classification System (USCS).
- The tool is used to process the common laboratory inputs of sieve analysis and Atterberg limits and then automatically calculate parameters such as D₁₀, D₃₀, D₆₀, C_u, and C_c. In addition, it generates gradation curves and plasticity charts in real time.