

Robert Frisee

#Titel und

#Untertitel der Arbeit

BACHELOR’S THESIS

To obtain the academic degree of Bachelor of Science (B.Sc.)

Submitted to the

Institute of Rock Mechanics and Tunneling, Graz University of Technology, Graz, Austria, 8010 Graz, Rechbauerstraße 12

Primary Supervisor

Univ.-Prof. Dipl.-Ing. Dr.-Ing, Marcher, Thomas

Second Supervisor

BSc MSc, Unterlass, Paul Johannes

Graz, August 2025

**AFFIDAVIT**

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. The text document uploaded to TUGRAZonline is identical to the present bachelor’s thesis.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Date |  | Signature |

**Abstract**

#1(Sentence on Motive, Method,) +Key Results, +Conclusions, (don’t exceed 3)

This study presents an unsupervised machine learning approach for detecting rock mass anomalies in tunnel boring machine (TBM) data from the Brenner Base Tunnel. A K-means clustering method, using the Kneedle algorithm to determine the optimal number of clusters, is applied to identify significant changes in rock mass conditions. Additionally, a cluster-based local outlier detection approach is employed to identify major anomalies in the data.

**Table of Contents**

1. **Introduction** 3  
    1.1 Heading (Level 2) ..................................................... 3  
    1.2 Heading (Level 2) ..................................................... 3  
     #1.2.1 Heading (Level 3) ............................................. 3

**References** 3  
**#AppendixA** 3

**1 Introduction**   
#What is the Problem: (Outline)/ Who are main contributors/ what did they do/ what am I revealing?

**Problem:**  
Geological recording in TBM operations is predominantly based on subjective or semi-quantitative assessments, creating inconsistencies and underscoring the need for an objective, data-driven classification system.

**Main contributors and their work:**  
The Brenner Base Tunnel project team drove an exploration tunnel using a double-shield TBM to investigate geological conditions. Over a period of approximately four years, the TBM collected an extensive dataset, including operational parameters, geological observations, and geotechnical measurements.

**Relevance:**  
The availability of this unique, continuous dataset from a long TBM drive in an exploration tunnel offers an unprecedented opportunity to develop and test objective classification methods that can later be applied in the main tunnel excavation.

**This study reveals:**  
By applying an unsupervised machine learning approach—specifically the K-means clustering algorithm—it is possible to detect variations in rock mass conditions and identify anomalies within the dataset. This provides a basis for improving geological classification, enhancing personal and machine safety, reducing standstills during future excavation in the main tunnel, and anticipating potential challenges for the TBM.

**Method**  
#Computational Paper: Inputs, Tools, method/ Explain PP1&2, CLUSTA

#Preprocessor1: This script loads many raw Excel data files from a folder, merges them into one big table, optionally removes standstill periods, optionally checks for missing data, saves the processed data to a parquet file, and returns the processed dataset. Path to the directory where your raw TBM data files (Excel files) are stored. A flag to decide if periods where the machine is stopped should be removed. A flag to decide if you want to generate a plot of missing data values for analysis.  You call a method concat\_tables from your utils instance.

#This method:

* Reads all individual Excel files from the folder.
* Concatenates them into one big table (likely a pandas DataFrame).
* Drops standstill periods if drop\_standstills is True.
* Optionally checks and plots missing values if check\_for\_miss\_vals is True.

#Preprocessor2:

### Summary of preprocessor():

1. **Imports & Utilities Initialization:**  
   Loads pandas, numpy, matplotlib, and custom utility classes utilities and computation.
2. **Load Data:**  
   Reads a parquet file (01\_TBMdata\_BBT\_S.gzip) containing TBM (Tunnel Boring Machine) operational data.
3. **Select and Rename Columns:**  
   Keeps only relevant columns and renames them with more manageable names (e.g., tunnel distance, forces, speed, torque, timestamps, power, etc.).
4. **Handle Missing Values:**
   * Fills NaNs in some force and advance number columns with zero.
   * Combines three advance force columns into a single one by averaging or selecting the single non-zero value.
   * Drops rows still containing NaNs after that.
5. **Aggregate by Tunnel Distance:**  
   Groups data points with identical tunnel distances and takes the median, reducing data redundancy.
6. **Outlier Filtering:**  
   Uses a Mahalanobis distance-based method (from utils.filter\_outliers) to remove outliers based on key TBM operational parameters, considering a sliding window of previous 100 data points and threshold at 90th percentile.
7. **Linear Interpolation:**  
   Creates an equally spaced DataFrame based on the median tunnel distance step size, ensuring uniform sampling along the tunnel axis.
8. **Hard Limits Filtering:**  
   Drops rows exceeding physical machine limits (e.g., main drive torque > 14 MNm, advance force > 42750 kN).
9. **Derived Features Calculation:**
   * Computes Specific Penetration (penetration per MN of advance force) using a custom function comp.s\_pen.
   * Calculates Torque Ratio and Theoretical Torque based on cutter parameters and machine constants using comp.t\_ratio.
10. **Add Geological Information:**  
    Reads geological data from an Excel sheet (rock mass ratings, UCS, classes, etc.) and maps geological classes to integers.  
    Then, it fills geological attributes in the main DataFrame for ±10 m around each geological survey point.
11. **Add Fault Zones:**  
    Reads fault zone data and merges it with the main DataFrame by tunnel distance, propagating fault labels backward to fill missing values.
12. **Save Processed Data:**  
    Saves the final preprocessed DataFrame to a parquet file (02\_TBMdata\_BBT\_S\_preprocessed\_wlabels\_mahal\_90.gzip).
13. **Return:**  
    Returns the fully preprocessed DataFrame.

In short, this function **cleans, merges, filters, interpolates, enriches, and prepares** TBM operational data with geological and fault info, making it ready for further analysis or modeling.

### #Plot: Summary of plot() function:

* **Purpose:**  
  Visualizes key operational metrics of the Tunnel Boring Machine (TBM) over a specified tunnel distance range (FROM to TO), smoothing data with a rolling window of size WINDOW.
* **Input:**
  + df: Preprocessed TBM data DataFrame.
  + FROM and TO: Numeric range of tunnel distance (meters) to plot.
  + WINDOW: Window size for rolling mean smoothing.
* **Functionality:**  
  Creates a figure with 7 vertically stacked subplots, each showing a different TBM-related variable plotted against tunnel distance:
  + **Advance Force [kN]**  
    Raw data in grey, rolling mean in black, horizontal lines at ±2 standard deviations.
  + **Torque Ratio**  
    Same style as above, limited y-axis from 0 to 1.2.
  + **Specific Penetration [mm/rot/MN]**  
    Drill penetration per rotation per MN of advance force.
  + **Penetration [mm/rot]**  
    Raw penetration depth per rotation.
  + **Main Drive Torque [MNm]**  
    Torque applied to the main cutter head.
  + **Specific Energy [MJ/m³]**  
    Energy consumption normalized by volume of excavated material.
  + **Belt Scale 1 [t]**  
    Material transported metric tonnage on conveyor belt.
* **Additional Features:**
  + Each subplot has grid, axis labels, limits, and legends showing ±2 standard deviation bands.
  + X-axis range zoomed to [FROM, TO].
  + The layout is tight and saved as a high-resolution PNG file named by the distance range.

In essence, this function helps **explore and analyze trends and anomalies in TBM performance data** over a specific tunnel segment with smooth visualization and statistical context.

#CLUSTA! (Where the magic happens)

## #Give sufficient Detail that the reader can reproduce CLUSTAMAGIC

This Python script uses clustering (KMeans) on tunneling data to detect fault zones (anomalies) based on distance from cluster centroids. It:

* Loads preprocessed tunneling data.
* Filters the dataset to a specific tunnel distance range.
* Selects features relevant to rock cutting mechanics.
* Standardizes these features.
* Uses the elbow method (inertia + KneeLocator) to find the optimal number of clusters.
* Performs KMeans clustering.
* Calculates distances from each data point to its nearest cluster centroid.
* Flags points farthest from centroids as potential faults (outliers).
* Visualizes results with rolling means, outliers, mapped fault zones, and clustered fault zones.
* Extracts and saves clustered fault zones as start/end intervals.

## Detailed Explanation

### 1. Data Preparation and Filtering

* Data is loaded from a Parquet file.
* Only rows within a tunnel distance range (FROM=14650 to TO=15150) are retained.
* Features chosen for clustering are:
  + Specific Energy [MJ/m³]
  + Spec. Penetration [mm/rot/MN]
  + torque ratio

### 2. Scaling Features

* Features are standardized using StandardScaler to zero mean and unit variance — necessary for KMeans to work well.

### 3. Determining Optimal Clusters with Elbow Method

* For 1 to 10 clusters, KMeans inertia (sum of squared distances of samples to their closest cluster center) is calculated.
* The KneeLocator finds the "elbow" point where adding more clusters yields diminishing returns.
* This optimal cluster count is used for final clustering.

### 4. Clustering and Distance Calculation

* KMeans is run with the optimal cluster number.
* Each sample’s distance to its closest centroid is calculated.
* The top outlier\_fraction (1%) farthest points are flagged as anomalies/fault zones.
* These are added as a binary column Fault Zone Cluster and a continuous score column cluster\_prob\_score in the dataframe.

### 5. Visualizing Results

* Plots show rolling means and raw data for the three features, highlighting fault zones.
* Two types of fault zones are visualized:
  + **Mapped fault zones** from labeled data (Fault column).
  + **Clustered fault zones** detected by the clustering algorithm.
* Fault zones are highlighted as vertical lines and shaded regions.
* Clustered fault zones are detected by sliding a 3m window along the tunnel distance and counting fault hits exceeding a threshold (15 hits), then merging nearby zones.
* Legends explain colors and markers.

### 6. Saving Clustered Fault Zones

* A function extracts the start/end of clustered fault zones using the sliding window and threshold.
* Adjacent zones separated by less than 3m are merged.
* The result is saved as a text file listing all clustered fault zones with their tunnel start and end distances.

## Additional Notes

* The code includes commented out logic for detecting fault zones by consecutive rows flagged as faults instead of sliding windows.
* The color maps and overlays are adapted based on the number of rock classes for better visualization.
* The rolling window smooths data trends to highlight anomalies better.

**Results**

#Present output of the experiment, model or computation

#(dont mix Results with Discussion)

**Discussion**

#Extract principles, relationship, generalizations

#Present analysis, model or theory  
#Show relationship between the results analysis, model or theory

**Conclusion**  
#Draw together the most important results and their consequences

#List any reservations or limitations

**Acknowledgements**  
#Thank people who have helped you with ideas, technical assistance, material

**References**

#cite significant previous work

#cite sources of theories, data, anything else I took from elsewhere

#References: name, initials, year, title, journal, volume, start-page and finish-page

**Figures**

#Graphs plot data

**,**

#Essential material that would interrupt the flow of the main text  
#must have purpose: place for tedious derivations