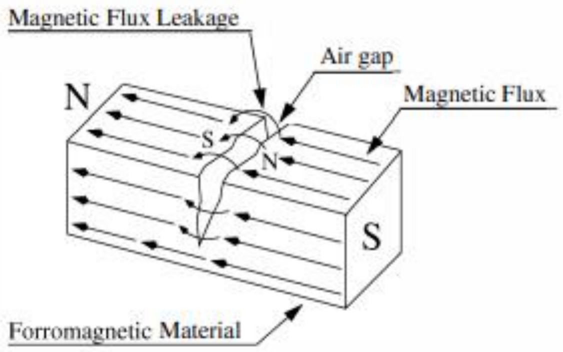
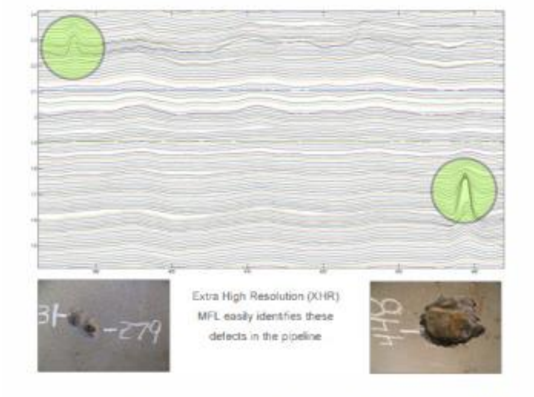
Background: ILI (In-Line Inspection) inspections involve using specialized tools, often referred to as "pigs," to assess the condition of pipelines from the inside. These inspections help detect anomalies such as corrosion, cracks, dents, or metal loss, ensuring the integrity and safety of the pipeline. ILI tools gather high-resolution data while traveling through the pipeline, enabling pipeline operators to make informed maintenance decisions.

Aim: The aim of the program is to develop a machine learning-based solution for pipeline defect analysis by leveraging historical inspection data. It facilitates automated defect classification in current inspection datasets, improving efficiency, consistency, and accuracy in identifying pipeline issues while reducing manual effort.

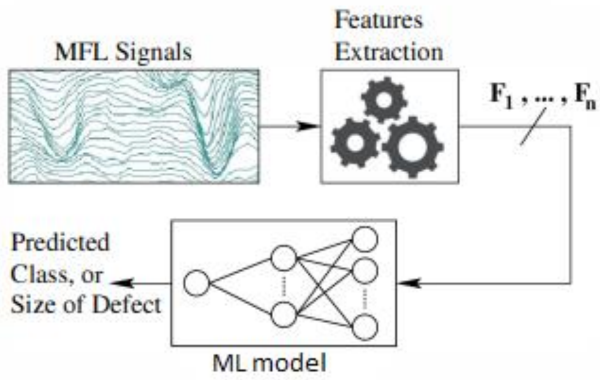


Magnetic Flux Leakage around a metal loss feature



Sample MFL measurements with corresponding metal loss defects (Source: Pure Technologies Ltd., www.puretechltd.com)

MFL-based pipeline inspection tools (Source: Baker Hugues Inc., www.bakerhughes.com)



Predicting defect class and size from MFL signals

Results: Two prototype Python programs were developed: one to train a machine learning model using historical pipeline inspection data, and another to classify defects from current inspection data based on the trained model. The programs handle data preprocessing, model training, prediction, and defect classification, saving results and metadata for reproducibility.

Content:

1. 12301.dbf - Historical inspection data file used to train the machine learning model.
2. 2_random_forest_model.joblib - The saved trained Random Forest model for predicting defects.
3. label_mapping.json - a mapping of defect types to numeric labels, created during training for encoding categorical target values.
4. reverse_mapping.json - the inverse of label_mapping.json, used during prediction to decode numeric labels back to defect types.
5. training_columns.json - stores the list of training feature columns to ensure consistent preprocessing for new inspection data.
6. 12602.dbf - current inspection data file used for prediction and defect classification.
7. predicted_results.csv - the output file containing the original inspection data with added columns for predictions and classified defect types.