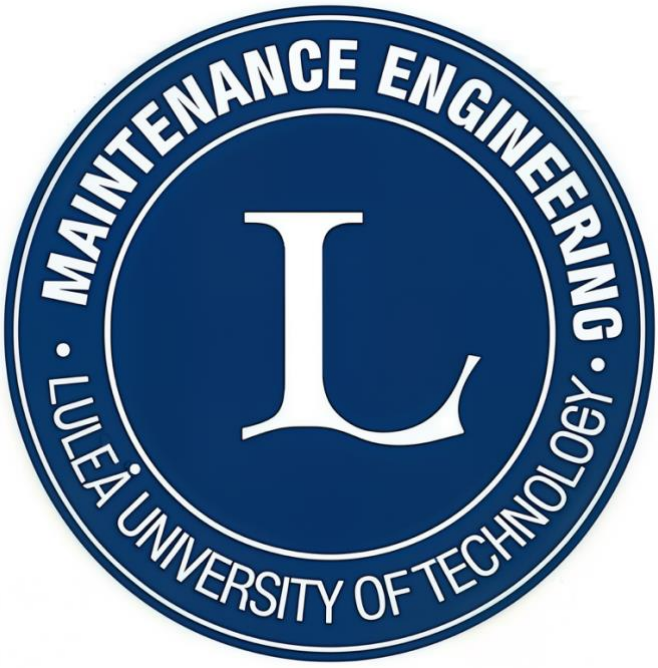

Luleå University of Technology

[Department of Civil, Environmental and Natural Resources Engineering]

Course: D7015B – Industrial AI & eMaintenance

Assignment 3

"Analysis of Local Track Discontinuities"

Submitted by:
Akbar Ali Khan
Student ID: 1158006726

Supervisor: Mohammed Amin Adoul

Date 22 February 2026

Table of Contents

Objective

The objective of this assignment is to analyse railway track acceleration features and develop a machine learning model to detect local track discontinuities. The goal is to distinguish between normal track conditions and defects using feature extraction and classification techniques.

Data Description

The provided dataset consists of extracted acceleration features from multiple railway track tests. Each row represents a measurement window with statistical and spectral features such as mean, standard deviation, kurtosis, crest factor and dominant frequency.

Data Preparation

Data Loading & Preprocessing

Python Code:

```
import pandas as pd

# load trail datasets
df1 = pd.read_csv("Trail1_extracted_features_acceleration_m1ai1-1.csv")
df2 = pd.read_csv("Trail2_extracted_features_acceleration_m1ai1.csv")
df3 = pd.read_csv("Trail3_extracted_features_acceleration_m2ai0.csv")

# combine into one dataframe
df = pd.concat([df1, df2, df3], ignore_index=True)

# quick checks
print("Shape:", df.shape)
print("Columns:", df.columns.tolist())
print("Event labels:", df["event"].unique())
print(df.head())


# remove non-feature columns
df = df.drop(columns=["start_time", "axle", "cluster", "tsne_1", "tsne_2"])

# convert event labels to binary
df["event"] = df["event"].apply(lambda x: 0 if x == "normal" else 1)

# check result
print("\nAfter cleaning:")
print("Shape:", df.shape)
print("Event value counts:\n", df["event"].value_counts())


# separate features and target
X = df.drop("event", axis=1)
y = df["event"]

print("\nFeature shape:", X.shape)
print("Target shape:", y.shape)
```

Summary:

After cleaning, the dataset contained 150 samples and 17 columns. The target variable distribution shows 60 normal cases and 90 defect cases.

Feature Scaling

Feature Normalization

Python Code:

```
from sklearn.preprocessing import StandardScaler

# normalize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("\nAfter normalization:")
print("First row (scaled):\n", X_scaled[0])
```

Summary:

Feature scaling was applied using StandardScaler to make sure that all features are contributing equally to the model and also to improve classification performance.

Training & Testing Split

Train-Test Split

Python Code:

```
from sklearn.model_selection import train_test_split

# split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)

print("\nTraining samples:", X_train.shape[0])
print("Testing samples:", X_test.shape[0])
```

Summary:

The dataset was split into training (80%) and testing (20%) sets to evaluate model performance on unseen data.

Model Training (SVM)

SVM Model Training

Python Code:

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
```

```
# train SVM model
svm_model = SVC(kernel="rbf", random_state=42)
svm_model.fit(X_train, y_train)
```

Summary:

The SVM model was trained to distinguish between normal and defective track conditions based on the extracted features.

Model Evaluation

Evaluation

Python Code:

```
# predictions
y_pred = svm_model.predict(X_test)

# evaluation
accuracy = accuracy_score(y_test, y_pred)
print("\nSVM Accuracy:", accuracy)

print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Summary:

The SVM model achieved an accuracy of 97% (approx.) on the test dataset. The model correctly identified most defects.

Confusion Matrix

Python Code:

```
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

# plot confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_test, y_pred),
                              display_labels=["Normal", "Defect"])
disp.plot()
plt.title("Confusion Matrix – SVM Track Defect Detection")
plt.tight_layout()
plt.show()
```

Plot:

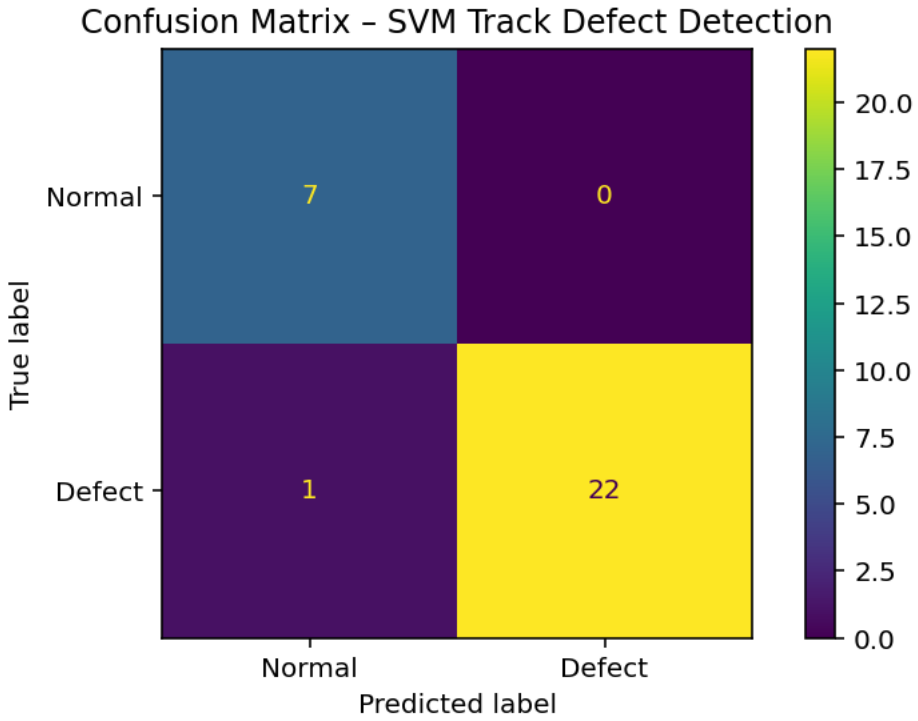

Figure 1. Confusion Matrix for Track Defect Detection

Summary:

The confusion matrix shows that the model correctly classified 29 out of 30 test samples. Only one defect was misclassified and no false alarms were generated. This indicates strong reliability for maintenance decision-making.

Cross-Validation

Cross Validation

Python Code:

```
print("\nClassification Report:\n", classification_report(y_test, y_pred))

from sklearn.model_selection import cross_val_score

# 5-fold cross-validation
cv_scores = cross_val_score(svm_model, X_scaled, y, cv=5)

print("\nCross-validation scores:", cv_scores)
print("Mean CV accuracy:", cv_scores.mean())
```

Summary:

Cross-validation produced a mean accuracy of 94.7% (approx.) which indicates that the model performs consistently across different data splits.

Feature Selection (ANOVA)

Feature Selection

Python Code:

```
from sklearn.feature_selection import SelectKBest, f_classif

# select top 5 features using ANOVA F-test
selector = SelectKBest(score_func=f_classif, k=5)
X_selected = selector.fit_transform(X_scaled, y)

# get selected feature names
selected_features = X.columns[selector.get_support()]

print("\nSelected features (ANOVA):", list(selected_features))
```

Summary:

The most significant features for defect detection were:

- Max
- Min
- Range
- Kurtosis
- Crest-Factor

These features capture peak values and help in identifying signal shape characteristics that indicate track irregularities.

Model with Selected Features

SVM with Selected Features

Python Code:

```
# use only selected features
X_selected = df[selected_features]

# normalize selected features
X_selected_scaled = scaler.fit_transform(X_selected)

# train-test split again
X_train_s, X_test_s, y_train_s, y_test_s = train_test_split(
    X_selected_scaled, y, test_size=0.2, random_state=42
)

# train SVM on selected features
svm_selected = SVC(kernel="rbf", random_state=42)
svm_selected.fit(X_train_s, y_train_s)

# evaluate
y_pred_s = svm_selected.predict(X_test_s)
accuracy_selected = accuracy_score(y_test_s, y_pred_s)
print("\nSVM Accuracy with selected features:", accuracy_selected)
```

Summary:

The model maintained similar accuracy using only selected features which indicates that these features capture the most relevant information.

Conclusion

This study demonstrates that machine learning can effectively detect railway track discontinuities using acceleration-based features. The SVM model achieved high accuracy and showed consistent performance through cross-validation. Feature selection identified key indicators such as peak values and signal shape characteristics, which are strongly associated with the track defects.

References

- Lecture materials and assignment instructions provided for Assignment 3 – Analysis of Local Track Discontinuities.
- Python libraries documentation (Numpy, Pandas, Matplotlib, Sciklit-learn) were used for data analysis and plotting functions.