



Kristianstad
University
Sweden

Kristianstad University
SE-291 88 Kristianstad
+46 44-250 30 00
www.hkr.se

Machine Learning – DA380A

2022 National Bridge Inventory Data

Group 27
Ibrahim Mohamed
Anne-Claire Koch

Content

1. Project Description	3
1.1 Dataset Description	3
2. Problem Definition and Motivation.....	3
3. Approach.....	4
3.1 Data Exploratory Analysis	4
3.2 Preprocessing Techniques	4
3.3 Machine Learning Process	5
3.4 Challenges	6
3.5 Model Results	6
4. Discussion.....	8
4.1 Important Challenges in Machine Learning.....	8
4.2 Lessons Learned.....	8
4.3 Potential Improvements	9
.5. Documentation	9
5.1 Jupyter Notebook	9
5.2 Word Document of all Features	10
5.3 Model Deployment.....	10

1. Project Description

1.1 Dataset Description

Our dataset was directly sourced from the National Bridge Inventory from <https://infobridge.fhwa.dot.gov/>. Our dataset had 143 features with a dataset of approximately 15,000 data cells. The main feature categories cover general bridge information (identification and location), construction details (year built, repairs, materials, design), and size and dimensions (height, weight, width, spans). Traffic information is included with daily use, projections, and special classifications. Maintenance and inspection data provide inspection frequency, condition ratings, and load limits, while operational status addresses traffic management and accessibility. Weather and climate features report average temperatures, rainfall, and humidity, and legislation information includes administrative details like responsible agencies and districts.

2. Problem Definition and Motivation

The primary objective of this project is to develop a machine learning model to predict the "Total Bridge Cost" for accurate budgeting and planning in bridge construction projects. We chose a regression model (a supervised learning approach) due to the continuous nature of the "Total Bridge Cost" feature.

To motivate this approach, we conducted an initial data analysis to understand relationships between features like bridge length, material types, and environmental factors, which could impact construction costs. Through cluster analysis, we visualized the data to identify groups of closely correlated features, helping us select variables with the strongest association with cost. This confirmed that a regression model would be effective in capturing these relationships.

We then developed three regression models:

- K-Nearest Neighbors (KNN) Regression

- Decision Tree Regression
- Random Forest Regression

For each model, we identified the top 10 important features that support predictive power. These features consist of quantitative data (continuous or ordinal), aligning well with regression analysis. This feature selection enabled the models to learn patterns in the data and produce accurate numerical predictions of bridge costs.

3. Approach

3.1 Data Exploratory Analysis

To better understand the structure and relationships within our dataset, we performed a series of exploratory data analysis steps before selecting features for our model. This process included:

Data Cleaning and Preprocessing: We inspected for missing values, inconsistencies, and irrelevant columns, which were either removed or imputed to improve data quality and streamline our analysis based on our target variable *Total Bridge Cost*.

Descriptive Statistics and Visualizations: We calculated basic statistics (mean, median, standard deviation) and used pair plots and histograms to identify correlations, potential outliers, and columns needing transformations or scaling.

K-Means Clustering: After preprocessing, we applied K-Means clustering to detect patterns, redundant features, and feature groupings. Visualizing clusters provided insights into segments within the data that we could leverage in our regression model.

3.2 Preprocessing Techniques

During preprocessing, we used pipelines for encoding, imputation, and scaling, combining numerical and categorical pipelines with a ColumnTransformer for a streamlined process.

Numerical Data Pipeline: We applied SimpleImputer with median imputation (robust to outliers) and MinMaxScaler to scale features between 0 and 1. This scaling ensures consistency in feature ranges, enhancing model performance, especially in regression.

Categorical Data Pipeline:

- **One-Hot Encoding (OHE):** For non-ordinal categories. This allowed converting each of our category into a binary column to treat them as distinct without implying hierarchy.
- **Label Encoding:** We used this for our ordinal features, mostly features relating to 'ratings' or 'appraisals', which allowed for assigning a unique integer to each value, preserving order when relevant and improving efficiency

3.3 Machine Learning Process

To build a reliable model for predicting *Total Bridge Cost*, we tested three machine learning algorithms: KNN Regressor, Decision Tree Regressor, and Random Forest Regressor.

We used scikit-learn for implementation due to its built-in tools for preprocessing, model tuning, and evaluation. We also utilized GridSearchCV for hyperparameter tuning.

K-Nearest Neighbors (KNN): KNN was used as a baseline due to its simplicity and interpretability, to assess if a distance-based approach could capture the patterns in our data.

Decision Tree: Decision Tree was chosen for its interpretability and ability to handle nonlinear relationships and feature interactions in smaller datasets.

Random Forest: Finally, Random Forest was chosen for its ensemble method, which reduces overfitting by combining multiple decision trees, making it robust to noise and suitable for complex relationships.

We performed GridSearchCV for hyperparameter tuning on each model. However, we saw a performance decrease after tuning, likely due to overfitting, where the model better fit the training data but generalized poorly to new data. As a result, we retained the untuned models for better generalization.

3.4 Challenges

One major challenge was maintaining data quality throughout the project. With 143 features, processing took considerable time, and we had to frequently reassess our approach to ensure effective data handling.

Deciding which features to keep, combine, or drop was especially challenging due to our limited domain knowledge on bridges. To address this, we researched each feature to understand its relevance, though this was time-consuming. Ensuring only meaningful features were included was essential for optimal model performance.

Balancing model learning without overfitting was a constant focus. We aimed for a model that could generalize well to new data without being overly influenced by specific patterns in the training data.

Some fields contained engineering terms or abbreviations that were unclear, complicating decisions on feature inclusion. After researching the data source website, we opted to drop features that lacked clear meaning, as they could negatively impact model accuracy.

3.5 Model Results

In terms of quantitative results (Figure 1), the Decision Tree model performs the best overall, with the lowest Mean Absolute Error (MAE) and Mean Squared Error (MSE). It also has the highest R^2 , meaning that it provides the most accurate predictions among the three models. The Random Forest model also performs quite well, though it is a bit less accurate than the Decision Tree. In contrast, the K-Nearest Neighbors (KNN) model has significantly higher error rates and a low R^2 making it the least effective model.

	Model	Mean Absolute Error (MAE)	Mean Squared Error (MSE)	R ² Score
0	Random Forest	0.000753	0.000242	0.768161
1	Decision Tree	0.000442	0.000198	0.810064
2	K-Nearest Neighbors	0.002916	0.000784	0.247049

Figure 1- Quantitative results

Regarding the qualitative results, we observe how each model's predictions align with actual values. In Figure 2, the Random Forest model shows a relatively good alignment between predicted and actual values, although it is slightly less accurate than the Decision Tree. Figure 3 illustrates the performance of the Decision Tree model, which, as our best performer quantitatively, shows data points closely following the ideal line on the scatter plot, confirming its accuracy for predictions. Finally, in Figure 4, we see the KNN model struggling to accurately predict values compared to the other models, as evidenced by its lower R²

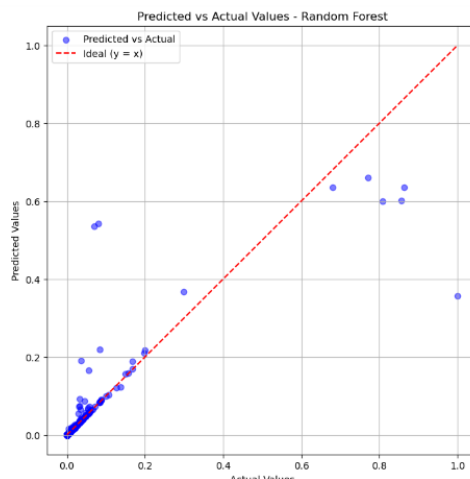


Figure 3 - Random Forest Predictions vs Actual

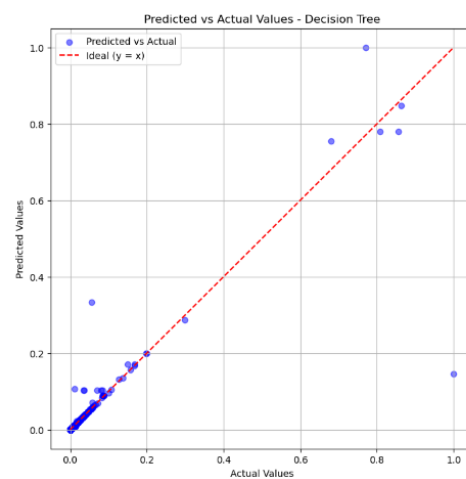


Figure 2- Decision Tree Predictions vs Actual

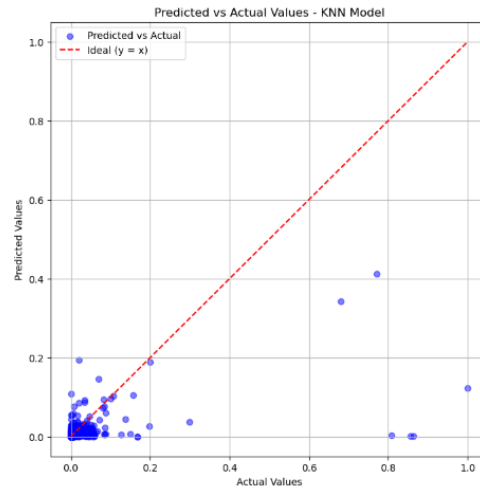


Figure 4 - KNN Predictions vs Actual

4. Discussion

4.1 Important Challenges in Machine Learning

One of the biggest challenges in ML projects is dealing with messy or incomplete data. Cleaning data, handling missing values, and encoding categorical variables are time-consuming but essential for model accuracy.

Selecting the most relevant features is challenging, as including too many irrelevant features can lead to overfitting while excluding important features can reduce model accuracy. We quickly realized that it takes knowing how to balance both to get the best model accuracy possible.

4.2 Lessons Learned

Data exploration proved important, as taking the time to explore our data thoroughly helped us uncover patterns and identify areas for improvement. This step provided a solid understanding of the data's structure and relationships, which was crucial for the following machine-learning tasks.

Thoughtful feature engineering significantly boosted model accuracy, emphasising the impact of well-prepared data and how important it was to work on clean data.

The process of preprocessing and model building was extensive, requiring multiple rounds of experimentation, testing, and refinement to achieve optimal results. This iterative approach taught us the importance of patience and adaptability, as each iteration offered a new opportunity for improvement.

Our preprocessing and model-building process required multiple rounds of experimentation, testing and refinement. This required a lot of patience and organisation.

Documenting each step allowed us to track decisions, and we recognised the need for realistic timelines to manage complex tasks efficiently. With a project of this complexity, it was easy to spend excessive time on certain task. Moving forward, a structured project plan will help balance our work and meet objectives.

4.3 Potential Improvements

With more time, there are several improvements we would have liked to work on. Advanced feature engineering, such as polynomial features or targeted transformations, could have allowed us to reveal more complex relationships within the data that was missed.

More time would have also allowed us to experiment with alternative algorithms, like ensemble methods or neural networks, which might have given better results.

While we did manage to deploy the model to the web, we would have liked to enhance the deployment by presenting the results in a more user-friendly format, allowing users to interpret predictions more easily.

5. Documentation

5.1 Jupyter Notebook

Our Jupyter Notebook, titled “ML Group Project” in the “Jupyter Notebook” folder, provides a comprehensive, step-by-step record of the project. It includes markdown cells explaining each phase, from data exploration to model training, with detailed explanations of data preprocessing, feature engineering, and model selection. Each code block has comments to explain our reasoning.

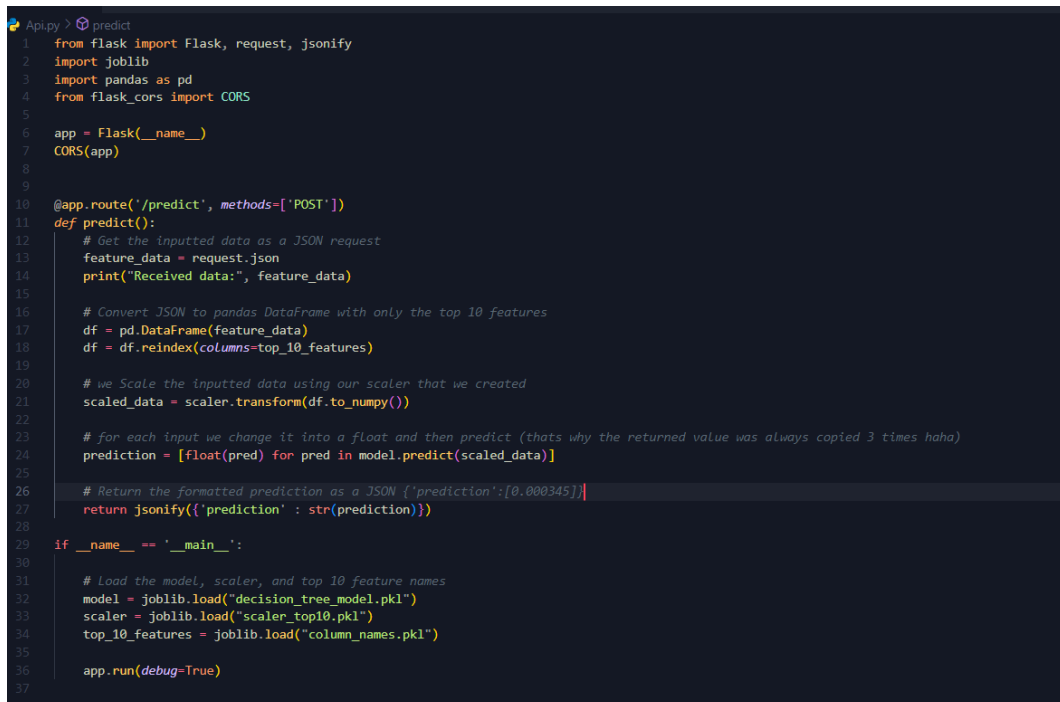
We documented all our steps within our Jupyter Notebook, so that others could follow and understand every aspect of the work.

5.2 Word Document of all Features

To better understand our dataset, we created a separate document called “Feature Details” where each feature is researched individually. This Word document served as a reference guide for clarifying feature meanings, especially when feature relevance was in question.

5.3 Model Deployment

We deployed our model to allow interaction with users. Figure 5, shows a screenshot of our Api file after using Flask. Detailed comments of how it works can be found within the file directly.



```

1  from flask import Flask, request, jsonify
2  import joblib
3  import pandas as pd
4  from flask_cors import CORS
5
6  app = Flask(__name__)
7  CORS(app)
8
9
10 @app.route('/predict', methods=['POST'])
11 def predict():
12     # Get the inputted data as a JSON request
13     feature_data = request.json
14     print("Received data:", feature_data)
15
16     # Convert JSON to pandas DataFrame with only the top 10 features
17     df = pd.DataFrame(feature_data)
18     df = df.reindex(columns=top_10_features)
19
20     # We Scale the inputted data using our scaler that we created
21     scaled_data = scaler.transform(df.to_numpy())
22
23     # for each input we change it into a float and then predict (thats why the returned value was always copied 3 times haha)
24     prediction = [float(pred) for pred in model.predict(scaled_data)]
25
26     # Return the formatted prediction as a JSON {'prediction':[0.000345]}
27     return jsonify({'prediction' : str(prediction)})
28
29 if __name__ == '__main__':
30
31     # Load the model, scaler, and top 10 feature names
32     model = joblib.load("decision_tree_model.pkl")
33     scaler = joblib.load("scaler_top10.pkl")
34     top_10_features = joblib.load("column_names.pkl")
35
36     app.run(debug=True)
37

```

Figure 5 - Api file for web deployment