National College of Ireland

A Transformer-Driven Framework for Accurate CO_2 Emission Modelling and Forecasting

MSc Research Project
MSc Data Analytics

Lokesh Musunuru
Student ID: 24130753

School of Computing
National College of Ireland

Supervisor: Dr. Vikas Tomer

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Lokesh Musunuru
Student ID:	24130753
Programme:	MSc in Data Analytics **Year:** 2025
Module:	Research Practicum Part 2
Lecturer:	Dr. Vikas Tomer
Submission Due Date:	28 January 2026
Project Title:	A Transformer-Driven Framework for Accurate CO_2 Emission Modelling and Forecasting
Word Count:	900 **Page Count:** 9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Lokesh Musunuru
Date:	28 January 2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A Transformer-Driven Framework for Accurate CO_2 Emission Modelling and Forecasting

Lokesh Musunuru
Student ID: 24130753

1. Introduction

This project builds a reproducible machine learning pipeline to predict vehicle CO_2 emissions (g/km) from structured vehicle specifications and fuel-consumption features. The task is challenging because emissions depend on complex, nonlinear interactions among engine size, transmission type, fuel category, and consumption behaviour, which limits the effectiveness of simple linear assumptions. The developed system standardizes data validation and preprocessing, trains models from classical baselines to deep learning and transformer-based architectures, and evaluates them consistently using MAE, MSE, RMSE, and R^2 to identify the most accurate and generalizable approach.

2. System Specifications

Hardware requirements (minimum and recommended)

- **CPU:** Minimum 4 cores; recommended 8+ cores for faster preprocessing, cross-validation, and boosting training.
- **GPU:** Optional; recommended NVIDIA GPU with 6 GB VRAM for deep models (MLP and FT-Transformer), particularly for larger datasets or longer training schedules.
- **RAM:** Minimum 4 GB; recommended 16 GB to support faster experimentation, multiple model training runs, and cross-validation.
- **Storage:** Minimum 5 GB free; recommended 7 GB to store datasets, notebooks, and model artifacts.

Software requirements

- **Operating System:** Windows 10/11, Ubuntu 20.04+, or macOS 12+ (Linux recommended for CUDA workflows).
- **Python version:** Python 3.9+ (recommended 3.10 or 3.11 for compatibility and performance).
- **Environment:** Conda or venv-based isolated environment with Jupyter support for reproducibility.
- **CUDA/cuDNN (if GPU training is used):** CUDA 11.8+ with a compatible cuDNN version aligned to the installed PyTorch build.

3. Required Libraries

The pipeline is based on a specialized collection of scientific and machine learning libraries to assist in data processing, modelling, maximization, and assessment. NumPy offers fast numerical functions and array manipulations in the entire procedure of pre-processing and computation of metrics. Pandas is applicable in loading CSV data, schema inspection, missing value checking, removing duplicates, and categorical transformations. Scikit-learn assists the classical baselines (Random Forest, Lasso, Ridge), preprocessing (scaling, train test split), and hyperparameters optimization through cross-validation. XGBoost has a high regression baseline that is capable of boosting nonlinear interactions in an efficient way. PyTorch uses Multilayer Perceptron (MLP) and the FT-Transformer architecture to provide the ability to use and train custom architectures with optional GPU acceleration. Training curves and predicted-vs-actual plots are illustrated with the help of Matplotlib as necessary to interpret and report.

Pip can be used in an isolated environment to provide a consistent way of installation to ensure replicability across machines.

1) **pip install numpy==2.3.5 pandas==2.3.3 matplotlib==3.10.7**
2) **pip install scikit-learn==1.7.2**
3) **pip install xgboost==3.1.2**
4) **pip install torch==2.9.1 torchvision==0.24.1 torchaudio==2.9.1**

4. Dataset description

The dataset in this project includes the vehicle-level data that is detailed, including both technical characteristics and fuel consumption features, which are essential predictors of CO 2 emissions. It contains such attributes like vehicle make and model, vehicle class, engine size (L), number of cylinders, transmission type, and fuel type. Such operational parameters as fuel consumption in city, highway, and combined driving conditions (L/100 km), and combined fuel efficiency (mpg) offer critical clues of real-world performance. The target variable, which is the CO 2 emissions (g/km), is also included in the dataset, which allows supervised learning to approach predictive modelling. The given sample includes a wide range of vehicles of different types compact cars, mid-size sedans, SUVs, sports cars, and luxury cars of such manufacturers as Acura, Alfa Romeo, Aston Martin, and Audi. This variety means that the FT-Transformer model is trained in a broad variety of engine types and driving behavior, enhancing its ability to generalize to predict CO 2 in different types of vehicles.

5. Models Implemented

The modelling process is well defined with a flow starting with the basic techniques up to sophisticated deep architectures. It starts with classical regression baselines like Ridge and Lasso to find regularized linear regression point of reference and continues with nonlinear ensemble learners like Random Forest and XGBoost to find feature interactions without hand engineering. Then, Multilayer Perceptron (MLP) is introduced as a main deep-learning baseline that is used to train nonlinear mappings between scaled tabular inputs. Last but not

the least, an FT-Transformer is the most modern architecture that builds upon self-attention instead of feature representations to capture cross-feature dependencies, which is the last stage of the baseline → deep learning → transformer-based hybrid progression.

6. Code snippets and plots:

1) Data Info

```
# Check for missing values and data types
print("\nDataset Info:")
print(df.info())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7385 entries, 0 to 7384
Data columns (total 12 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Make                              7385 non-null   object
 1   Model                             7385 non-null   object
 2   Vehicle Class                     7385 non-null   object
 3   Engine Size(L)                    7385 non-null   float64
 4   Cylinders                         7385 non-null   int64
 5   Transmission                      7385 non-null   object
 6   Fuel Type                         7385 non-null   object
 7   Fuel Consumption City (L/100 km)  7385 non-null   float64
 8   Fuel Consumption Hwy (L/100 km)   7385 non-null   float64
 9   Fuel Consumption Comb (L/100 km)  7385 non-null   float64
 10  Fuel Consumption Comb (mpg)       7385 non-null   int64
 11  CO2 Emissions(g/km)               7385 non-null   int64
dtypes: float64(4), int64(3), object(5)
memory usage: 692.5+ KB
None
```

2) Missing Values

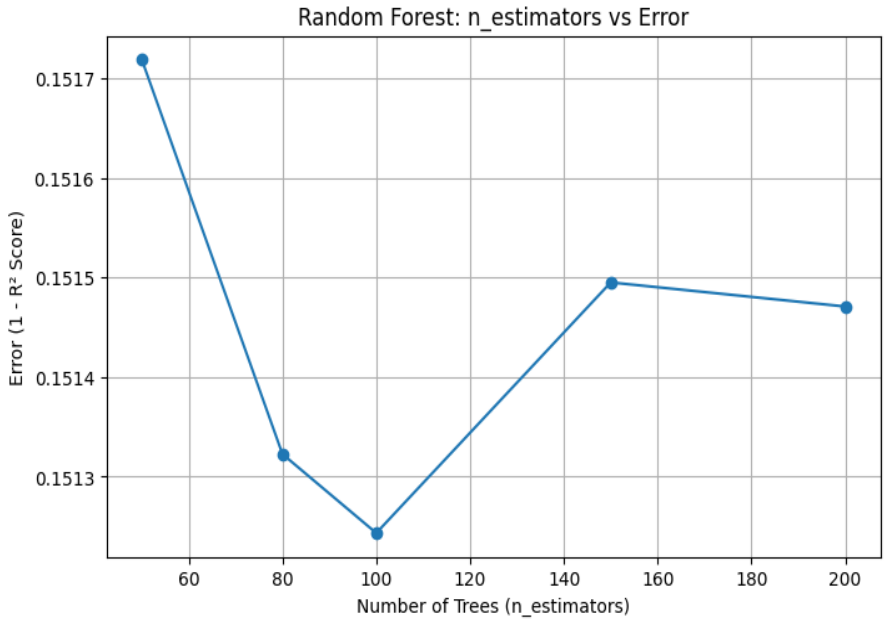
```
print("\nMissing values per column:")
print(df.isnull().sum())
```

```
Missing values per column:
Make                              0
Model                             0
Vehicle Class                     0
Engine Size(L)                    0
Cylinders                         0
Transmission                      0
Fuel Type                         0
Fuel Consumption City (L/100 km)  0
Fuel Consumption Hwy (L/100 km)   0
Fuel Consumption Comb (L/100 km)  0
Fuel Consumption Comb (mpg)       0
CO2 Emissions(g/km)               0
dtype: int64
```

3) Splitting data

```
import torch
import random
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

# Set consistent random seeds
seed = 42
np.random.seed(seed)
random.seed(seed)
torch.manual_seed(seed)
if torch.cuda.is_available():
    torch.cuda.manual_seed_all(seed)


y = df['CO2 Emissions(g/km)']
X = df.drop('CO2 Emissions(g/km)', axis=1)

# Scale features for neural models
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y.values.reshape(-1, 1))

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0.2, random_state=seed)
```

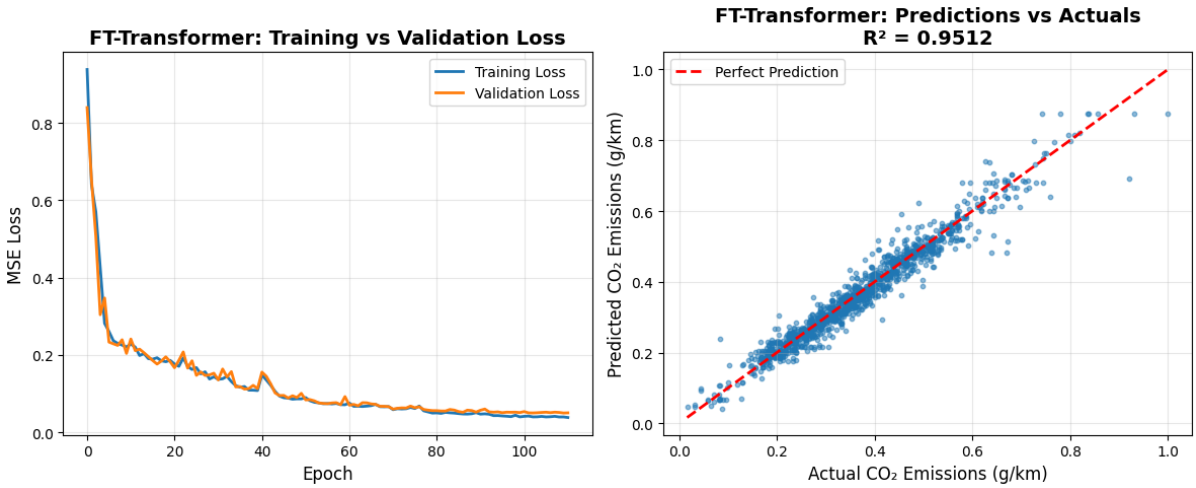
4) Random Forest hyperparameter plot



5) FT-Transformer

```
# Feature Tokenizer Layer
class FeatureTokenizer(nn.Module):
    def __init__(self, n_features, d_token):
        super().__init__()
        self.linear = nn.Linear(1, d_token)
        self.bias = nn.Parameter(torch.zeros(n_features, d_token))

    def forward(self, x):

        x = x.unsqueeze(-1)
        tokens = self.linear(x) + self.bias
        return tokens

# FT-Transformer Architecture
class FTTransformer(nn.Module):
    def __init__(self, n_features, d_token=64, n_heads=4, n_layers=3,
                 d_ffn=256, dropout=0.1):
        super().__init__()

        # Feature Tokenization
        self.feature_tokenizer = FeatureTokenizer(n_features, d_token)

        # CLS token for aggregation
        self.cls_token = nn.Parameter(torch.zeros(1, 1, d_token))

        # Transformer Encoder
        encoder_layer = nn.TransformerEncoderLayer(
            d_model=d_token,
            nhead=n_heads,
            dim_feedforward=d_ffn,
            dropout=dropout,
            activation='gelu',
            batch_first=True)
```

6) Loss plot of FT-Transformer



7) Comparison of evaluation metrics

Table 1. Model Performance Comparison Across Regression Metrics

Model	MAE	MSE	RMSE	R2 Score
Random Forest	0.0401	0.0029	0.0538	0.8543
Lasso	0.0495	0.0042	0.0649	0.7883
Ridge	0.0495	0.0042	0.0649	0.7885
XGBoost	0.0388	0.0028	0.0525	0.8614
MLP	18.2487	563.4910	23.738	0.8439
FT-Transformer	0.0219	0.0010	0.0312	0.9512

7. Conclusion:

The project provides a reproducible CO 2 -emission prediction pipeline, which normalizes data checks, preprocessing, model training, and evaluation across a variety of model families. The pipeline allows making a fair comparison and ensuring reliable reproducibility due to the transition to deep learning and the use of a transformer-based tabular model as the next stage of classical baselines. The FT-Transformer meant the most due to its performance since it can better capture interactions between the features than linear models and standard ensembles due to its attention mechanism. The system is extensible for future datasets, additional vehicle attributes, and deployment-oriented improvements such as explainability and model serving.

References:

1. https://numpy.org/doc/stable/
2. https://pandas.pydata.org/docs/
3. https://matplotlib.org/stable/
4. https://scikit-learn.org/stable/
5. https://docs.pytorch.org/docs/stable/