

# Model 01

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data = pd.read_excel('Data_model1/gmi-methane-data-epa.xlsx')
data.head()
```

```
0      Afghanistan      NaN      NaN      No      Biogas      Manure Management      CH4      1990      MMTCO2e      1.146776      Agriculture      Livestock      Manure
1      Afghanistan      NaN      NaN      No      Biogas      Manure Management      CH4      1991      MMTCO2e      1.224627      Agriculture      Livestock      Manure
2      Afghanistan      NaN      NaN      No      Biogas      Manure Management      CH4      1992      MMTCO2e      1.302478      Agriculture      Livestock      Manure
3      Afghanistan      NaN      NaN      No      Biogas      Manure Management      CH4      1993      MMTCO2e      1.380330      Agriculture      Livestock      Manure
4      Afghanistan      NaN      NaN      No      Biogas      Manure Management      CH4      1994      MMTCO2e      1.458181      Agriculture      Livestock      Manure
```

```
data.tail()
```

```
237895  Zimbabwe      NaN      NaN      No      Other (Non-GMI)      Waste: Other      CH4      2046      MMTCO2e      0.0      Waste      OtherWaste      NaN
237896  Zimbabwe      NaN      NaN      No      Other (Non-GMI)      Waste: Other      CH4      2047      MMTCO2e      0.0      Waste      OtherWaste      NaN
237897  Zimbabwe      NaN      NaN      No      Other (Non-GMI)      Waste: Other      CH4      2048      MMTCO2e      0.0      Waste      OtherWaste      NaN
237898  Zimbabwe      NaN      NaN      No      Other (Non-GMI)      Waste: Other      CH4      2049      MMTCO2e      0.0      Waste      OtherWaste      NaN
```

Image 1.1

## Data Preparation

```
...      ...      ...
237895      OtherWaste      NaN
237896      OtherWaste      NaN
237897      OtherWaste      NaN
237898      OtherWaste      NaN
237899      OtherWaste      NaN

[237900 rows x 13 columns]
```

```
data.shape
```

```
(237900, 13)
```

```
data.describe()
```

```
Subregion      Region      year      value
count      0.0      0.0      237900.000000      237900.000000
mean      NaN      NaN      2020.000000      2.110528
std      NaN      NaN      17.606854      15.984733
min      NaN      NaN      1990.000000      0.000000
25%      NaN      NaN      2005.000000      0.000000
50%      NaN      NaN      2020.000000      0.006072
75%      NaN      NaN      2035.000000      0.304760
max      NaN      NaN      2050.000000      662.636621
```

```
data.columns
```

```
Index(['country', 'Subregion', 'Region', 'GMI Country', 'GMI Sector',
       'GMI Category', 'gas', 'year', 'unit', 'value', 'EPA-Report-sector',
       'EPA-Report-source', 'EPA-Report-subsource'],
      dtype='object')
```

Image 1.2

Data Analysis:

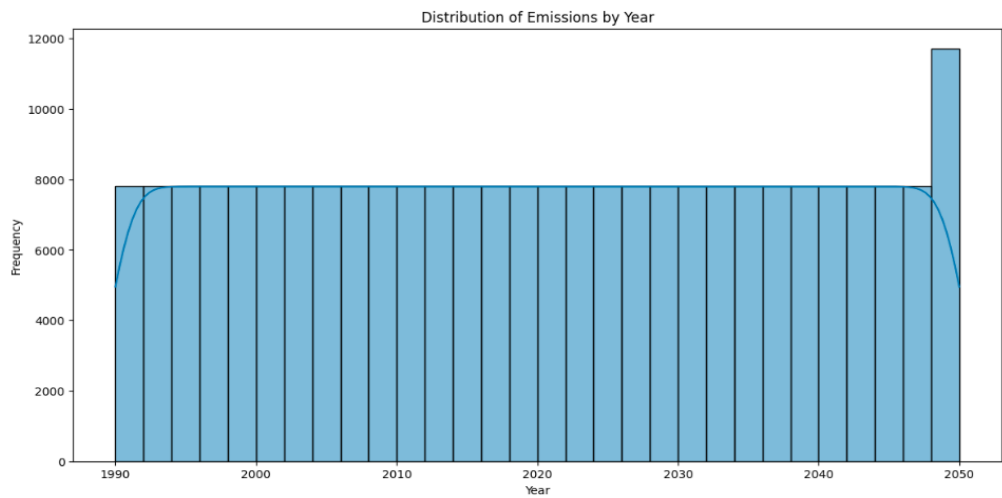


Image 1.3

Emissions by Country (top 10 countries by emissions):

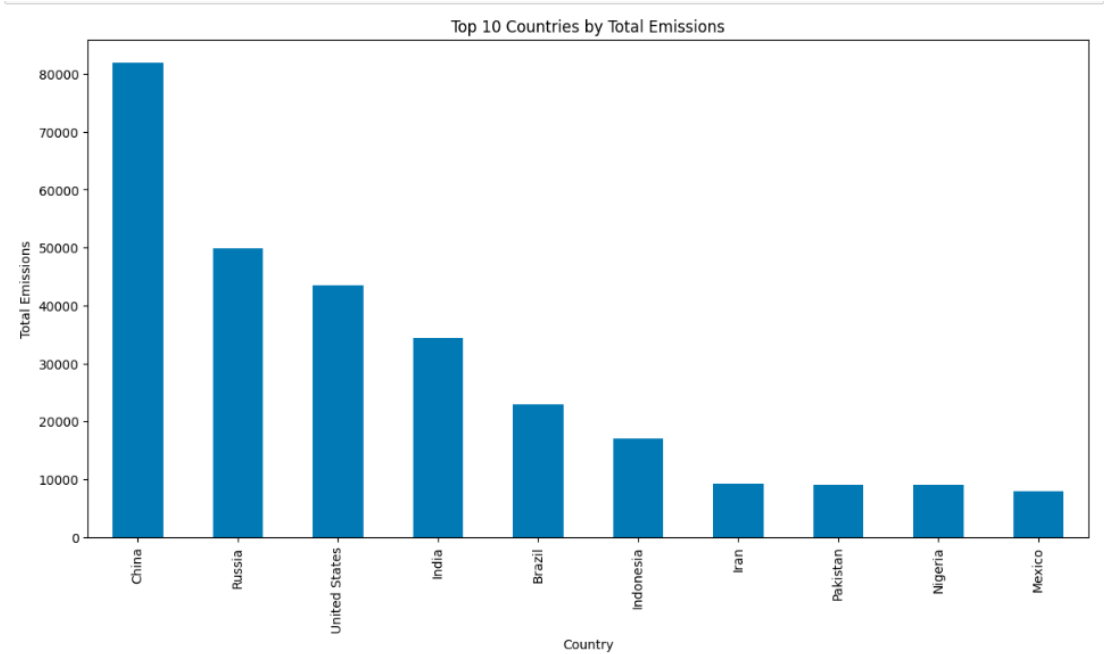


Image 1.4

Emissions by GMI Sector:

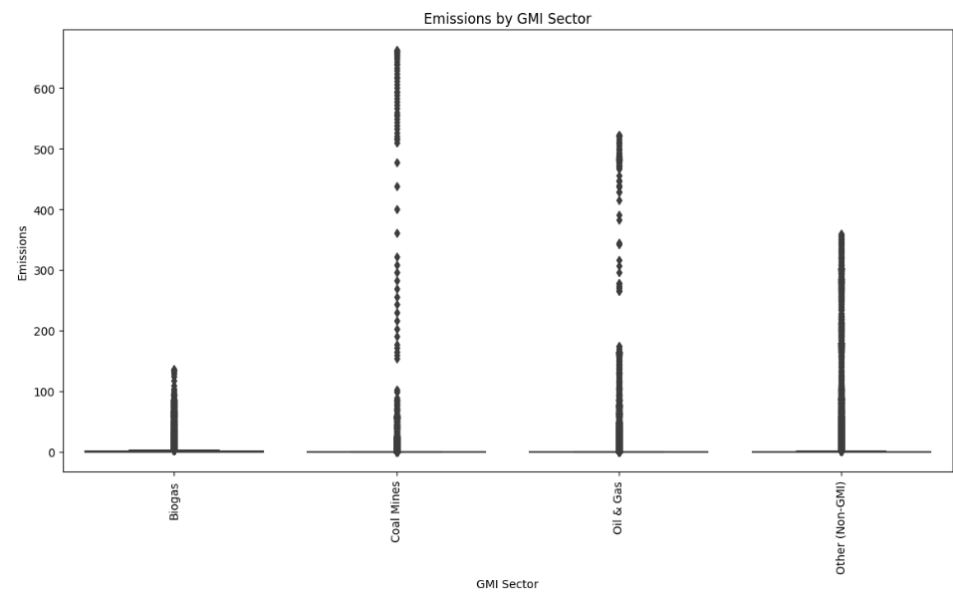


Image 1.5

Model Evaluation

Model 1 - Ancillary Variable Integration

```
from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Use only 'year' as the feature for simplicity
features_model1 = ['year']
X = data[features_model1]
y = data['value']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predict on the test set
y_pred_model1 = rf_model.predict(X_test)

# Evaluate the model
mse_model1 = mean_squared_error(y_test, y_pred_model1)
mae_model1 = mean_absolute_error(y_test, y_pred_model1)
r2_model1 = r2_score(y_test, y_pred_model1)

print(f"Model 1 Mean Squared Error (MSE): {mse_model1}")
print(f"Model 1 Mean Absolute Error (MAE): {mae_model1}")
print(f"Model 1 R-squared (R²): {r2_model1}")

Model 1 Mean Squared Error (MSE): 269.106453300848
Model 1 Mean Absolute Error (MAE): 3.466042432432984
Model 1 R-squared (R²): -0.0002374094818515804
```

Image 1.6

## Results

```
# Evaluate the model
mse_model1 = mean_squared_error(y_test, y_pred_model1)
mae_model1 = mean_absolute_error(y_test, y_pred_model1)
r2_model1 = r2_score(y_test, y_pred_model1)

print(f"Model 1 Mean Squared Error (MSE): {mse_model1}")
print(f"Model 1 Mean Absolute Error (MAE): {mae_model1}")
print(f"Model 1 R-squared (R²): {r2_model1}")

Model 1 Mean Squared Error (MSE): 269.106453300848
Model 1 Mean Absolute Error (MAE): 3.466042432432984
Model 1 R-squared (R²): -0.0002374094818515804
```

Image 1.7

### 1) Graphs:

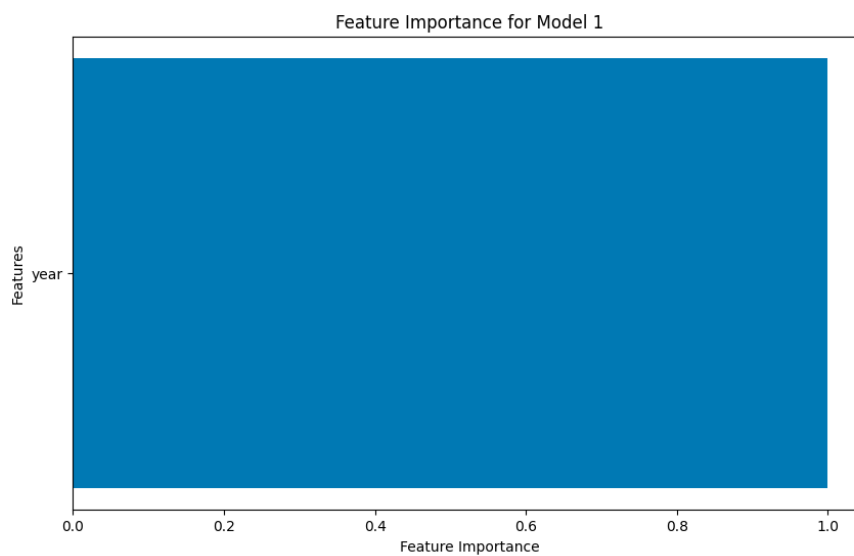


Image 1.8

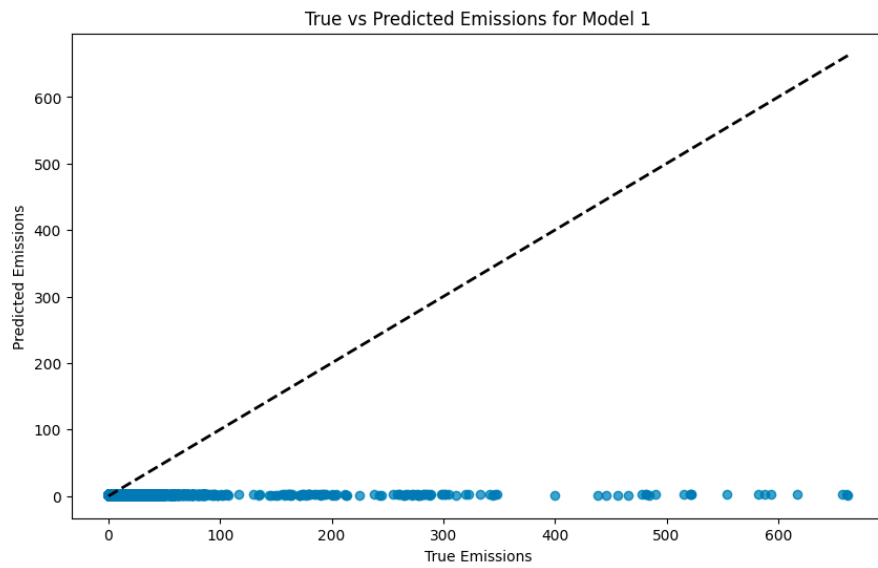


Image 1.9

Time Series of Predicted Emissions:

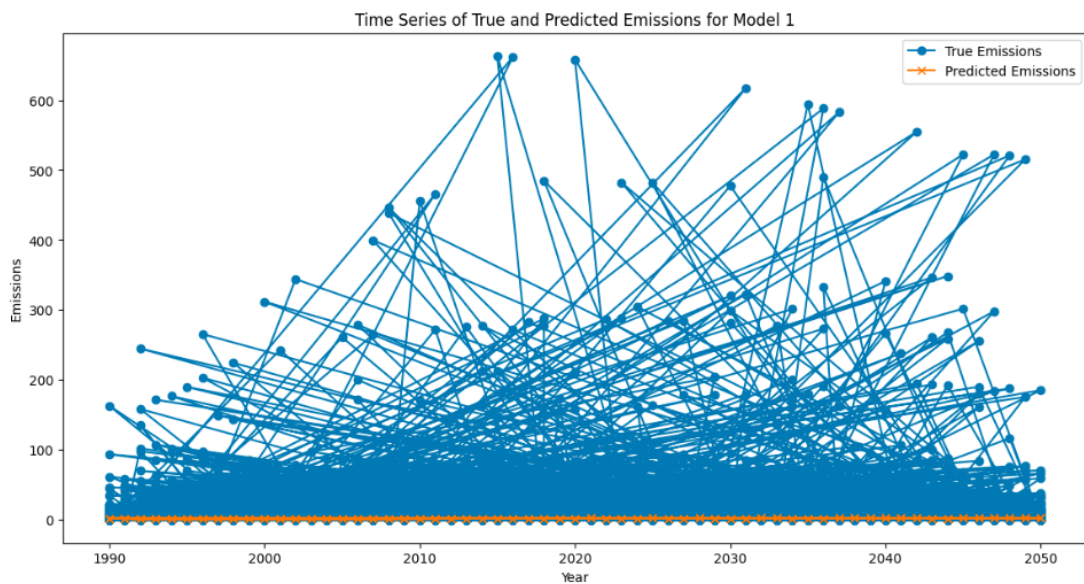


Image 1.10

## Model 02

### Model Evaluation

```
# # Features and target
X = data[['year']]
y = data['value']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale the numerical features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Build the neural network model
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)))
model.add(Dense(32, activation='relu'))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
history = model.fit(X_train_scaled, y_train, epochs=5, validation_split=0.2, verbose=1)

# Predict on the test set
y_pred_model2 = model.predict(X_test_scaled)

# Evaluate the model
mse_model2 = mean_squared_error(y_test, y_pred_model2)
mae_model2 = mean_absolute_error(y_test, y_pred_model2)
r2_model2 = r2_score(y_test, y_pred_model2)
```

### Image 2.1

### Results

```
Epoch 1/5
4758/4758 [=====] - 49s 10ms/step - loss: 247.5354 - val_loss: 270.8434
Epoch 2/5
4758/4758 [=====] - 49s 10ms/step - loss: 247.4794 - val_loss: 270.8197
Epoch 3/5
4758/4758 [=====] - 52s 11ms/step - loss: 247.4409 - val_loss: 270.8072
Epoch 4/5
4758/4758 [=====] - 51s 11ms/step - loss: 247.4371 - val_loss: 270.8458
Epoch 5/5
4758/4758 [=====] - 40s 8ms/step - loss: 247.4194 - val_loss: 270.8667
1487/1487 [=====] - 16s 11ms/step
Model 2 Mean Squared Error (MSE): 269.06670573356627
Model 2 Mean Absolute Error (MAE): 3.2765413742411855
Model 2 R-squared (R²): -8.967239400625715e-05
```

### Image 2.2

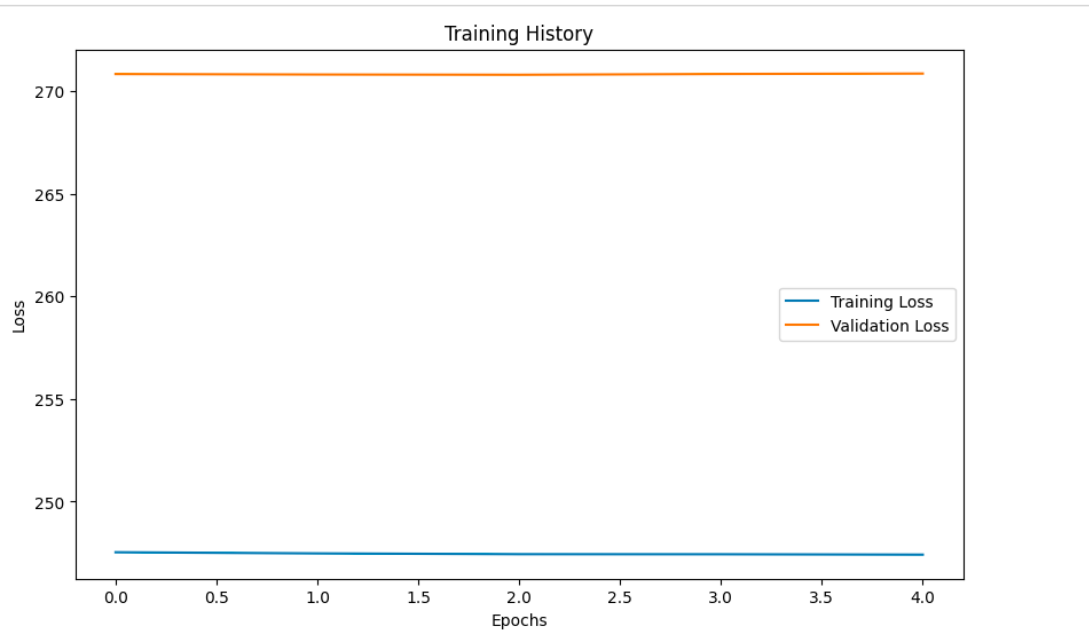


Image 2.3

True vs Predicted Emissions:

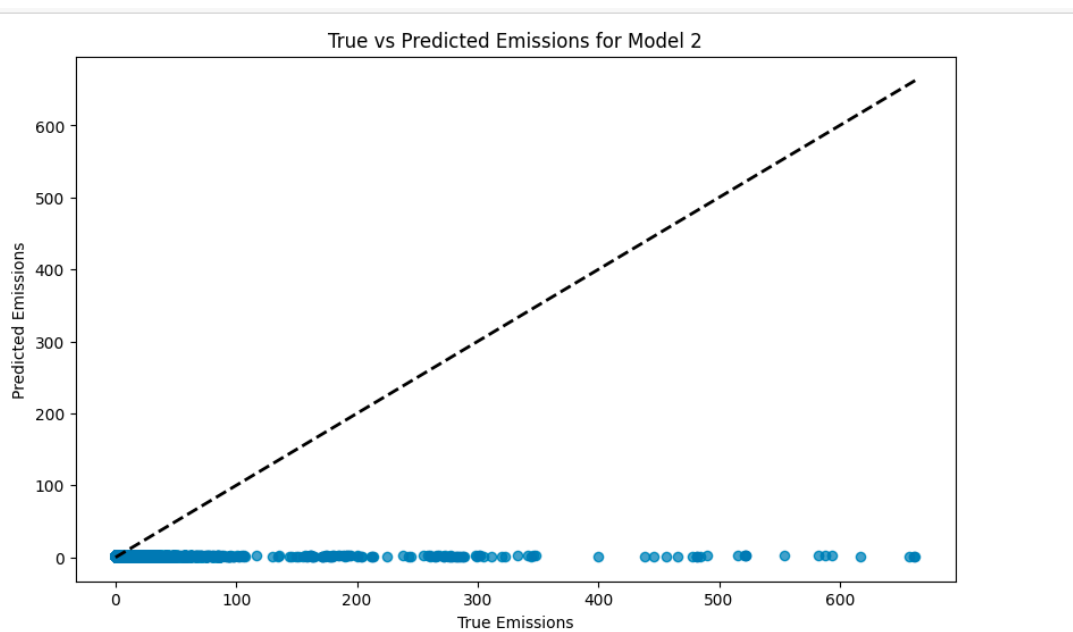


Image 2.4

Time series of emissions:

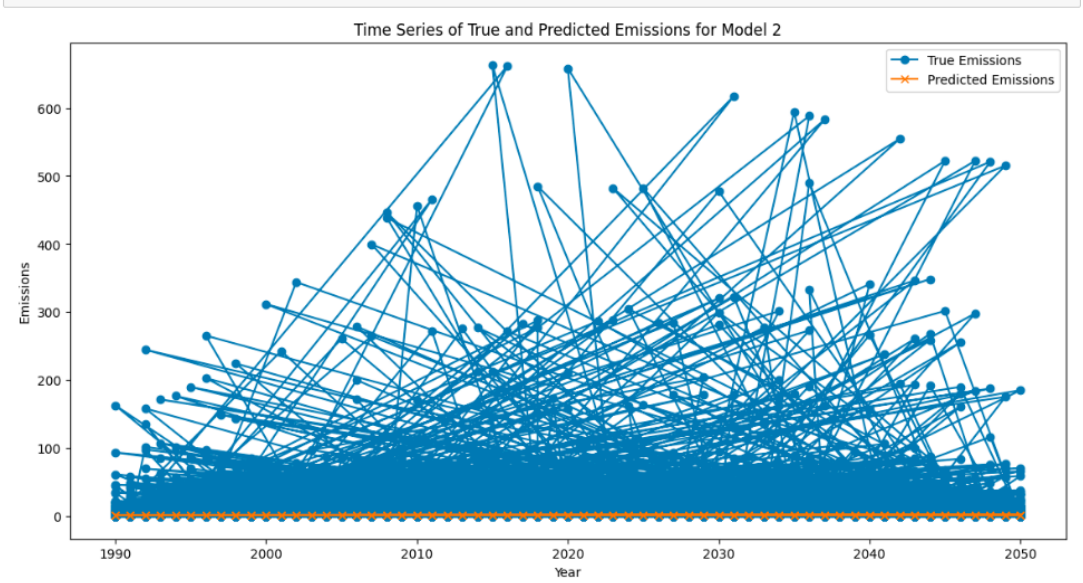


Image 2.5