CS361: Individual Carbon Footprint

Mohit Kumar-210101068 Sahil Jaiswal-210101091 Anup Kumar-210101019
Sachin Gautam-210101090 Avinav Yadav-210101025

Abstract

Climate change is universally recognized as a critical global environmental issue, with human activities identified as significant contributors due to increased anthropogenic emissions of greenhouse gases, primarily CO2. To effectively address this, quantifying individual carbon emissions is essential. However, existing carbon footprint calculators often require detailed data inputs, hindering accessibility for users. In this study, we propose estimating individual carbon footprints through a simplified questionnaire based on lifestyle choices readily available to users. By assigning emission values to various activities and providing personalized feedback and recommendations, this approach aims to empower individuals to reduce their carbon footprint and adopt more environmentally sustainable behaviors.

1. INTRODUCTION

Climate change is now universally acknowledged as a critical global environmental challenge, largely driven by human activities and the resulting increase in anthropogenic emissions of greenhouse gases, predominantly CO2. Given the substantial contribution of individual actions to these emissions, quantifying and reducing personal carbon footprints are crucial steps towards mitigating climate change. However, existing carbon footprint calculators often present barriers to accessibility due to their reliance on extensive data inputs, such as electricity usage, natural gas consumption, and more. In response to this challenge, this research proposes a novel methodology for estimating individual carbon footprints through a simplified questionnaire focused on lifestyle choices that are readily accessible to users. By assigning emission values to various activities and providing personalized feedback and recommendations, this approach aims to empower individuals to actively reduce their carbon footprint and adopt more environmentally sustainable behaviors.

The motivation for this project stems from the urgent need to address carbon emissions and mitigate climate change on a global scale. Initiatives like the Paris Agreement highlight the critical importance of effective carbon management in limiting global warming and minimizing the associated risks of climate change. In this context, the primary objective of this project is to develop machine learning models capable of predicting carbon emissions based on various lifestyle attributes. Leveraging datasets encompassing attributes such as body type, diet, energy usage, waste production, and more, the project aims to construct predictive models that can accurately estimate individual carbon footprints. Ultimately, the overarching goal is to provide individuals with actionable insights for assessing their carbon footprint, enabling them to make informed lifestyle decisions and tailor their habits to minimize potential carbon emissions. Through this endeavor, the project aims to contribute to the advancement of sustainable living practices, climate change mitigation initiatives, and overall environmental conservation.

2. DATASET

WE have used the Individual Carbon Footprint Calculation Dataset from Kaggle. The training set consisted of 20 basic features regarding lifestyle choices like Diet, How Often Shower, etc.

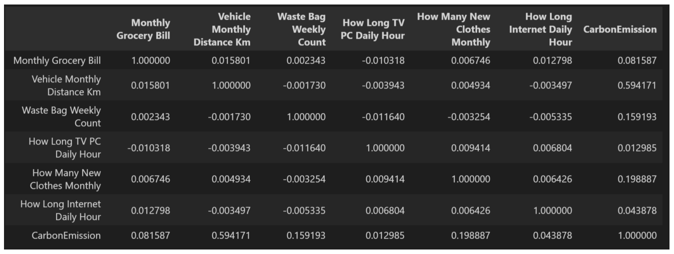
3. Data Cleaning and Exploratory Data Analysis (EDA)

3.1. Data Cleaning

Before we can analyze the data, we need to ensure that it is clean. This involves handling missing values, removing duplicates, and checking for inconsistencies in the data.

After exploring the data, we found that the 'Vehicle Type' column had 6721 null values. To handle this, we imputed these null values with 'no vehicle' as if a person left it blank we assume that he/she does not own a vehicle. We also checked for duplicate rows but all entries were different.

Furthermore, we encountered two columns, 'Cooking With' and 'Recycling', which contained arrays of strings as their values, indicating multiple values for each entry. To facilitate algorithm training, we employed a one-hot encoder to transform these columns into binary features, with each unique value in the arrays represented by a separate binary feature.



	Monthly Grocery Bill	Vehicle Monthly Distance Km	Waste Bag Weekly Count	How Long TV PC Daily Hour	How Many New Clothes Monthly	How Long Internet Daily Hour	CarbonEmission
Monthly Grocery Bill	1.000000	0.015801	0.002343	-0.010318	0.006746	0.012798	0.081587
Vehicle Monthly Distance Km	0.015801	1.000000	-0.001730	-0.003943	0.004934	-0.003497	0.594171
Waste Bag Weekly Count	0.002343	-0.001730	1.000000	-0.011640	-0.003254	-0.005335	0.159193
How Long TV PC Daily Hour	-0.010318	-0.003943	-0.011640	1.000000	0.009414	0.006804	0.012985
How Many New Clothes Monthly	0.006746	0.004934	-0.003254	0.009414	1.000000	0.006426	0.198887
How Long Internet Daily Hour	0.012798	-0.003497	-0.005335	0.006804	0.006426	1.000000	0.043878
CarbonEmission	0.081587	0.594171	0.159193	0.012985	0.198887	0.043878	1.000000

Figure 1: Univariate Analysis

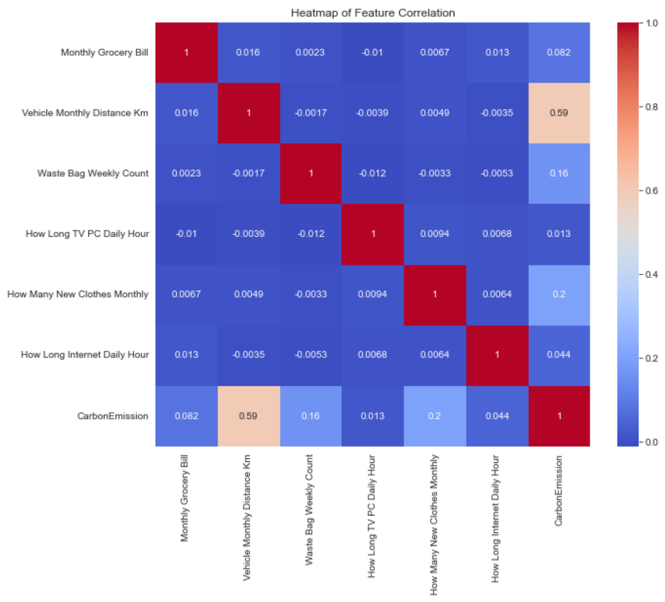


Figure 2: Correlation Matrix

3.2. Univariate Analysis

This involves analyzing each variable individually. We used the describe() function to see the distribution of numerical features of the dataset. We observed that the mean and median for almost all features are equal except for Vehicle Monthly Distance Km which is due to some people (67%) not having a vehicle to drive. Further, we used a box plot to see the distribution of numerical features and found out same as we saw in df.describe().

For categorical features, we used bar plots to see the distribution of data across different categories and found out that data is well distributed with every category having almost equal share in total data.

3.3. Bivariate Analysis

Here, we analyze the relationship between two variables. Scatterplots, correlation matrices, and cross-tabulations are common techniques used.

The heatmap reveals relationships among numerical variables like 'Vehicle Monthly Distance Km', 'Waste Bag Weekly Count', and 'How many New Clothes monthly' with 'Carbon Emission'. A high positive correlation suggests that as one increases, so does carbon emission. We can see a high correlation between 'Vehicle Monthly Distance Km' and Carbin Emission as evident from univariate analysis. However, low correlation doesn't imply less helpful as correlation only infers a linear relationship and the features may share a complex relationship.

Similarly, to see the bivariate relationship of categorical features with Carbon Emission we employed a bar plot to display the relationship average carbon emission of each category of the column. From the graphs we can infer that the mean Carbon Emission for a category in a feature follow a order. As we go to a higher order category of that feature the mean Emssion increases. Thus indicating us to use a ordinal encoding to encode our data.

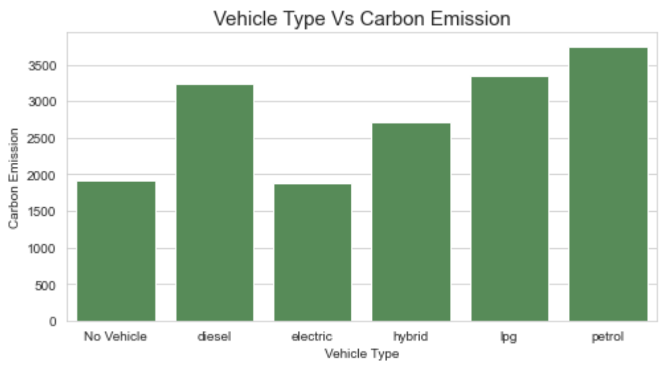


Figure 3: Petrol vs. Electric Vehicles

By Figure **??**, we can see that petrol-consuming vehicles produce more carbon than electric vehicles.

By Figure **??**, we can see that obese people have a higher carbon footprint.

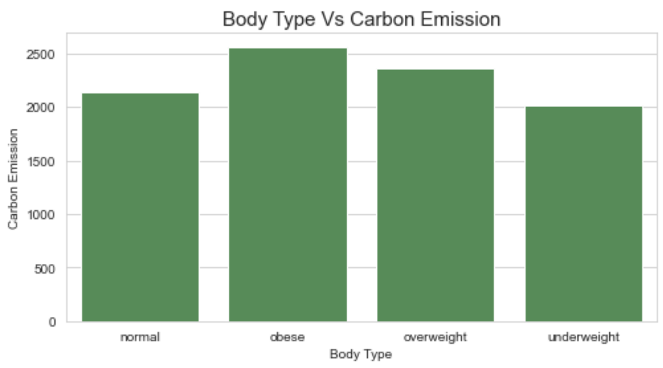


Figure 4: Obesity and Carbon Footprint

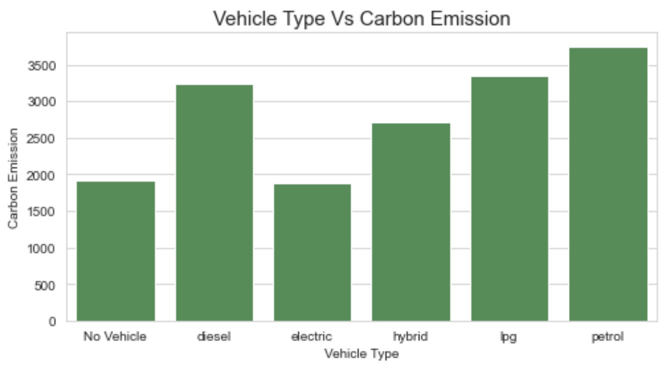


Figure 5: Petrol vs. Electric Vehicles

By Figure **??**, we can see that petrol-consuming vehicles produce more carbon than electric vehicles.

3.4. Multivariate Analysis

This involves analyzing more than two variables at the same time to understand the complex relationships between them.

For example, in Figure **??**, we have plotted a scatter plot showing the relation between multiple features. We found that males and people who frequently use air travel produce more carbon.

4. Data Preprocessing/ Feature Engineering

The original dataset contained 20 features and 6 features were numerical. To convert the rest of the features to numerical we used various preprocessing techniques inferred from the EDA we did previously. So to convert categorical features to numerical we employed a custom ordinal encoder that encoded our data to numerical. For this, we provided the encoder with a custom order which

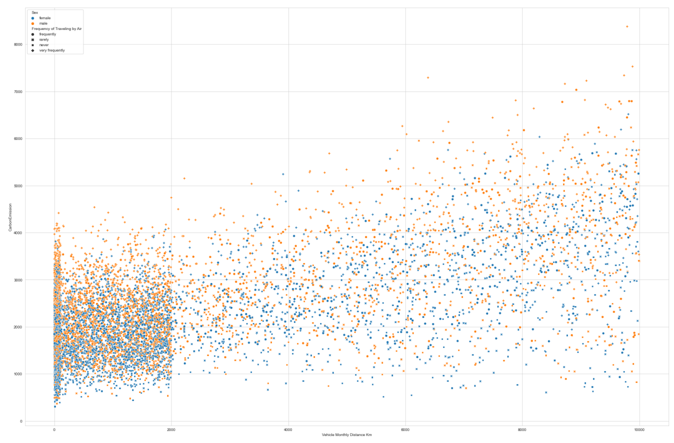


Figure 6: Example of Multivariate Analysis

we derived from the EDA part(by giving a higher number to the category whose mean Carbon Emission was higher). In this way, we converted our all features to numerical except 2(number of items recycled and types of cookware used) which were of string type and contained a list of items. To deal with them we first tried to make dummies for each item on the list but the results we not up to the mark. So, we added the total items in each list and used that as an order. Upon this, we also normalized a few of the features and applied our algorithm directly to the raw data, encoded data, and normalized data.

5. Metrics Used

5.1. Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is a metric used to measure the average magnitude of errors between predicted and actual values. It is calculated as the average of the absolute differences between predicted and actual values. MAE is relevant in regression models because it provides a straightforward interpretation of prediction accuracy in the original units of the target variable. It is less sensitive to outliers compared to other error metrics.

5.2. Mean Squared Error (MSE)

Mean Squared Error (MSE) is a metric used to measure the average squared differences between predicted and actual values. It penalizes larger errors more heavily than smaller ones, making it sensitive to outliers. MSE is commonly used in regression models as it provides a measure of the average squared deviation of predictions from the actual values. It is useful for evaluating the overall model performance.

5.3. Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is the square root of the mean squared error. It represents the standard deviation of the residuals and is measured in the same units as the target variable. RMSE is beneficial because it provides a more interpretable measure of error compared to MSE. It indicates the average magnitude of errors in the original units of the target variable, making it easier to understand and compare across different models.

5.4. Coefficient of Determination (R^2)

The Coefficient of Determination (R^2) is a statistical measure that represents the proportion of variance in the dependent variable that is explained by the independent variables in the model. It ranges from 0 to 1, where 0 indicates that the model does not explain any variability in the target variable, and 1 indicates perfect prediction. R^2 is relevant in regression models as it provides insights into the goodness of fit of the model. It helps assess how well the independent variables explain the variability observed in the dependent variable.

6. Methods

We have implemented these methods:

6.1. Linear Regression

6.1.1. Why Linear Regression

We employed linear regression exclusively on numerical columns within our dataset. This method allows for predicting a continuous target variable based on the linear relationship with one or more independent variables.

6.1.2. Algorithm Details

Algorithm 1 Linear Regression

```
Input: data $x_i$, target $y_i$, size $m$, learning rate $\alpha$, number of iterations $N$
Initialize weights $w$ and bias $b$
for $k = 1$ to $N$ do
    for $i = 1$ to $m$ do
        Compute predicted output $\hat{y}_i = w^T x_i + b$
        Compute loss $L_i = 1/2(\hat{y}_i - y_i)^2$
        Compute gradients $\nabla_w L_i = (\hat{y}_i - y_i)x_i$ and $\nabla_b L_i = \hat{y}_i - y_i$
        Update weights $w \leftarrow w - \alpha \nabla_w L_i$ and bias $b \leftarrow b - \alpha \nabla_b L_i$
    end for
end for
```

6.1.3. Challenges Faced and Solutions

In our original dataset, all columns were not numerical except 6. So first we applied linear regression considering only these 6 columns. After that, we preprocessed our dataset in 2 folds such that all features were numerical. After that we obtained our results on both the encoded data and normalized data and the results were the same but better than on the original dataset and also better than the mid-evaluation results. This was due to the better preprocessing techniques used this time.

6.1.4. Preliminary Result

After applying linear regression only on available numerical features we get the below errors and scores:

```
Mean Absolute Error: 605.164
Mean Squared Error: 595954.463
Root Mean Squared Error: 771.980
R-squared (R²): 0.413
```

After preprocessing and normalising the dataset we got the below errors and scores:

```
Mean Absolute Error: 444.6818998678045
Mean Squared Error: 281435.00806897157
Root Mean Squared Error: 530.5044844946851
R-squared (R²): 0.5903597901553821
```

6.1.5. Conclusion

The preliminary analysis involved initially applying linear regression solely to numerical columns, yielding a Mean Absolute Error of approximately 605.16 and an R-squared value of 0.41. Subsequent preprocessing of the data resulted in a notable enhancement in model performance, with the Mean Absolute Error decreasing to around 444.68 and the R-squared value improving to 0.59. This improvement underscores the significance of capturing additional information from non-numeric columns, leading to a more accurate representation of the target variable. Also, the results were the same on encoded data and normalized data and this was because linear regression doesn't get affected by linear changes in features because it adjusts its parameters according to it. The reduction in error from last time was due to better preprocessing techniques used. Also the MAE of 444 can be explained because of high bias of Linear Regression.

6.2. Decision Tree

6.2.1. Why Decision Tree

Decision trees are a great choice for their simplicity, interpretability, and ability to handle non-linear relationships in data without complex preprocessing. They're robust to outliers and missing values, scalable, and provide insights into feature importance. Thus after trying linear regression non-linear Decision tree is a good choice and also our Distance travelled in km had outliers so we can bridge the gap here. This makes them valuable for both understanding data and building predictive models.

6.2.2. Algorithm Details

Algorithm 2 Decision Tree Regressor

```
Input: Training data $\{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$, maximum depth $D$
Initialize decision tree $T$
Grow decision tree $T$ recursively:
    - Find the best feature $f$ and split point $v$ using a split criterion (e.g., mean squared error or variance reduction)
    - Split the data into two subsets $S_{\text{left}}$ and $S_{\text{right}}$ based on $f$ and $v$
    - If maximum depth $D$ is reached or stopping criteria met, stop recursion
    - Otherwise, recursively apply steps 3-5 to $S_{\text{left}}$ and $S_{\text{right}}$
Predict the output for each input $x$:
    - Traverse the decision tree $T$ to find the leaf node for $x$
    - Assign the predicted value as the average of target values in the leaf node
```

6.2.3. Challenges Faced and Solutions

Though a decision tree can handle numerical and categorical data, the categorical data must be encoded into numbers. This we tried first on our original dataset and then encoded it and normalised(though it has no effect on decision tree). After that we obtained our results on both the encoded data and normalized data and the results were the same but better than the mid-evaluation results and original dataset. This was due to the better preprocessing techniques used this time and the model inferred more from more features.

6.2.4. Preliminary Result

After applying decision tree regressor only on available numerical column we get the below errors and scores:

```
Mean Absolute Error: 669.0105410385177
Mean Squared Error: 748810.9937768794
Root Mean Squared Error: 865.3386584319918
R-squared (R²): 0.263559914755803
```

After preprocessing and normalising the dataset we got the below errors and scores:

```
Mean Absolute Error: 625.8340633612471
Mean Squared Error: 659017.8862397395
Root Mean Squared Error: 811.7991661979825
R-squared (R²): 0.040772407587459925
```

6.2.5. Conclusion

In conclusion, decision trees offer a compelling solution for predictive modeling due to their simplicity, interpretability, and robustness to nonlinear relationships in data. Despite facing challenges such as the need to encode categorical data and the potential presence of outliers, decision trees can still deliver reliable results when coupled with effective preprocessing techniques. Our experimentation revealed that even with minimal preprocessing, decision tree regression yielded promising results, showcasing its potential for understanding data and making accurate predictions.

6.3. Random Forest

6.3.1. Why Random Forest

We employed a random forest regressor on our dataset. Random forest is well suited as it can handle datasets with a large number of features with both numerical and categorical data. It also overcomes the issue of high variance of decision trees with techniques like bagging in which we ensemble results from multiple decision trees.

6.3.2. Algorithm Details

Algorithm 3 Random Forest Regressor Training

```
  **Input:** Training data $X$, target values $y$, number of estimators $n_{\text{estimators}}$, maximum depth maxdepth, minimum samples split minsamples_split
  Initialize an empty list of trees: trees = []
  **for** $t = 1$ to nestimators **do**
      Initialize a Decision Tree Regressor: tree$_t$ with maximum depth maxdepth and minimum samples split minsamples_split
      Randomly select indices with replacement: indices $\leftarrow$ Randomly choose $n$ indices from $[1, |X|]$ with replacement
      Train the $t$-th tree using the selected indices: tree$_t$.fit($X$[indices], $y$[indices])
      Append the trained tree to the list of trees: trees.append(tree$_t$)
  **end for**
  **Output:** Trained Random Forest model with nestimators trees
```

6.3.3. Challenges Faced and Solutions

The challenges faced were almost same as Decision Tree except the results of random forest can be less interpretable compared to simpler models. Understanding and interpreting the results of a random forest model required additional effort. Training random forests was also computationally intensive and time-consuming, especially with a large number of trees and a high-dimensional feature space.

6.3.4. Preliminary Result

After applying random forest regressor only on available numerical column we get the below errors and scores:

```
Mean Absolute Error: 625.7381501237061
Mean Squared Error: 628943.3310387341
Root Mean Squared Error: 793.0594750954899
R-squared (R²): 0.3814472755164351
```

After preprocessing and normalizing the dataset we got the below errors and scores:

```
Mean Absolute Error: 592.7102322343035
Mean Squared Error: 584825.2536326956
Root Mean Squared Error: 764.7386832328384
R-squared (R²): 0.14876283066455565
```

6.3.5. Conclusion

After implementing both Decision Tree and Random Forest regression models, we observed a slight improvement in error metrics compared to the initial results. The Random Forest model, which is an ensemble method built upon Decision Trees, showed a slightly lower error rate, indicating better performance in predicting the target variable due to less variance. We can see random forest capture much more variance. This improvement suggests that the Random Forest model better captures the underlying patterns in the data and provides more accurate predictions.

6.4. XGBoost

6.4.1. Why XGBoost

XGBoost, or Extreme Gradient Boosting, excels in predictive modeling due to its ability to handle complex datasets and deliver superior performance compared to traditional decision trees. It harnesses the power of ensemble learning, particularly gradient boosting, to sequentially improve model predictions. XGBoost offers several advantages:

-Enhanced Predictive Power: By combining multiple weak learners, typically decision trees, XGBoost builds a robust predictive model that effectively captures complex relationships within the data.

-Scalability: Despite its sophisticated algorithms, XGBoost is highly scalable and can efficiently handle large datasets with millions of observations and features.

-Regularization: XGBoost incorporates regularization techniques to prevent overfitting, ensuring the model's generalization capability and robustness.

-Feature Importance: XGBoost provides insights into feature importance, allowing users to identify the most influential variables driving predictions.

Algorithm 4 XGBoost Training

```
**Input:** Training data $X$, target values $y$, number of boosting rounds $n_{\text{rounds}}$, learning rate $\eta$, maximum tree depth maxdepth, regularization parameters $\lambda$ and $\gamma$
Initialize model with initial prediction: $\hat{y}_0 = 0$
**for** $t = 1$ to $n_{\text{rounds}}$ **do**
    Compute the negative gradient: $g_t = -\frac{\partial}{\partial \hat{y}} L(y, \hat{y})|_{\hat{y}=\hat{y}_{t-1}}$
    Compute the Hessian: $h_t = \frac{\partial^2}{\partial \hat{y}^2} L(y, \hat{y})|_{\hat{y}=\hat{y}_{t-1}}$
    Fit a decision tree to the negative gradients: $\text{tree}_t = \text{DecisionTreeRegressor}(X, -g_t, \text{maxdepth}, \lambda)$
    Update the model: $\hat{y}_t = \hat{y}_{t-1} + \eta \cdot \text{tree}_t(X)$
**end for**
**Output:** Trained XGBoost model
```

6.4.2. CHALLENGES FACED AND SOLUTIONS

While XGBoost offers remarkable performance, it also presents challenges that need to be addressed:

-Data Preprocessing: Similar to decision trees, XGBoost requires preprocessing steps such as encoding categorical variables and handling missing values to optimize model performance.

-Computational Complexity: XGBoost's computational complexity increases with the size of the dataset and the number of iterations. Efficient implementation and optimization strategies are crucial to mitigate this challenge.

6.4.3. Preliminary Result

After applying random forest regressor only on available numerical column we get the below errors and scores:

```
Mean Absolute Error: 617.0839588629144
Mean Squared Error: 616021.5027636262
Root Mean Squared Error: 784.8703732232643
R-squared (R²): 0.39415562568158613
```

After preprocessing and normalising the dataset we got the below errors and scores:

```
Mean Absolute Error: 268.9725907868724
Mean Squared Error: 108791.65224106541
Root Mean Squared Error: 329.8357958758652
R-squared (R²): 0.84164928322474
```

6.4.4. CONCLUSION

We saw that XGBoost gave us the best result. This can be explained due to the boosting it uses. It used multiple trees bu unlike random forest, it doesn't ensemble the results from all(bagging) but it keeps on improving results by boosting technique and this can be seen in results.

7. Conclusion

Linear Regression initially yielded a Mean Absolute Error (MAE) of 605.16 and an R-squared value of 0.41. However, after implementing feature engineering techniques, such as preprocessing and encoding categorical variables, there was a notable improvement in performance. The MAE decreased to 444.68, indicating a more accurate estimation of individual carbon footprints, while the R-squared value increased to 0.59, signifying a better fit of the model to the data.

In contrast, Decision Tree Regression displayed a slightly lower performance compared to Linear Regression, especially when incorporating categorical variables. When considering only numerical columns, the Decision Tree Regression yielded a MAE of 669.70 and an R-squared value of 0.26. However, upon including categorical variables, the model's performance slightly improved, with a MAE of 625.72 and a lower R-squared value of 0.14. This suggests that the model used the categorical values effectively.

Random Forest Regression, similar to Linear Regression, performed relatively better when considering only numerical columns. It exhibited a lower MAE of 629.43 and a higher R-squared value of 0.375. However, when categorical columns were included after label encoding, the model's performance slightly improved, with a MAE of 592.72 and a lower R-squared value of 0.17. This suggests that the model decreased the high variance of decision tree by using bagging ensemble technique.

Additionally, XGBoost demonstrated its prowess in predictive modeling, leveraging gradient boosting to achieve superior performance and scalability. Initially, without using label encoding, XGBoost yielded a MAE of 617.08 and an R-squared value of 0.39. However, after applying label encoding to categorical columns, the MAE significantly improved to 268.97, and the R-squared value increased to 0.84. This showcases the effectiveness of XGBoost, especially when combined with feature engineering techniques.

Overall, feature engineering, particularly preprocessing and encoding techniques, played a crucial role in enhancing the performance of Linear Regression, while

Decision Tree and Random Forest Regression models were more sensitive to the inclusion of categorical variables, with their performance varying accordingly. XGBoost, on the other hand, offered superior performance and scalability, making it a preferred choice for building accurate predictive models.

8. References

The dataset for our project is available on Kaggle under the title mydarkblueIndividual Carbon Footprint Calculation.

The code for our project is available on GitHub under the title mydarkblueCarbon Emission.

Sevde USTUN, Hanife BUYUKGUNGOR .*Calculation AND Evaulation of Individual Carbon Footprint.*

Loyarte-López, E., Barral, M., Morla, J. C. (2020). *An Automated Personal Carbon Footprint Calculator for Estimating Carbon Emissions from Transportation Use.*

Girish Bekaroo, Divesh Roopowa, Chandradeo Bokhoree.*Mobile-Based Carbon Footprint Calculation: Insights from a Usability Study.*

Chiamaka Ann Marie Ajufo, Girish Bekaroo. *Methodology for Carbon Footprint Calculation Towards Sustainable Innovation in Intangible Assets.* mydarkblue Sustainability, 12(4), 1629.

mydarkblueComplex Carbon Footprint Calculator