Car Attribute Prediction Model

By: Justin D, Abdullah Q, & Krishna S

Intro to Data Science

Instructor: Naina Chaturvedi

Abstract:

This project takes a large car model dataset, cleans it, retrieves what qualities a user wants for a car, predicts and returns the rest of the attributes of a car given what the user wants. It does this using a random forest model to account for the different features. Before giving the result, the project will return many values to see how accurate the predicted value is. This paper will go over the intricacies of each method used, and reveal how each method and decision reveals how using a random forest model works.

Problem:

When designing or customizing a car, it's hard to know how certain features affect other parts of the vehicle. For example, if a user wants a car with a large fuel tank and high acceleration, what should the trunk size or weight be? There isn't an easy way to figure this out. We wanted to create a tool that can help predict missing car details based on the features a person already knows or wants.

Connection to Course Material:

- Data Cleaning and Preprocessing: Methods to manage missing values, tokenization, and Regex to clean and format data.
- Numpy and Pandas: Dataframe usage and manipulation was vital throughout the entirety of the project.
- Random Forest: The Random Forest model as explained and demonstrated in class and the homework labs.
- Graphical Representation: Utilization of Matplotlib and Seaborn to represent statistics and indicate data relationships found between features. The code for these models is included but commented out in the project code for clarity when running the model.
- MAPE, R^2, Accuracy: Many analysis statistics were used or attempted in order to determine the accuracy and usefulness of our model.

Proposed Solution:

Our project is a machine learning model that predicts the missing attributes of a car when the user enters at least 6 known ones. We use a Random Forest model that can handle both categorical (like fuel type) and continuous values (like mileage and weight). The model gives estimated values for the unknown features based on patterns it learned from real car data.

Project Importance:

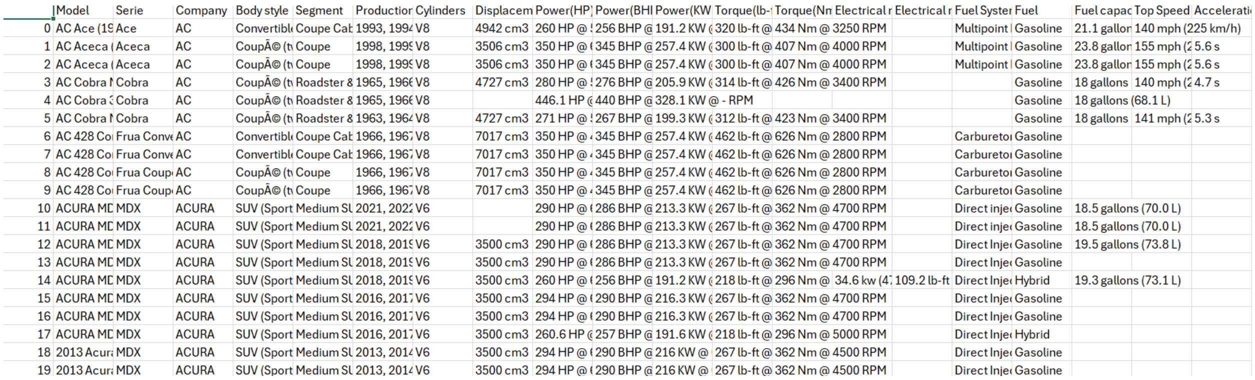
This project is important because it helps people understand how different car features relate to each other. It could be useful for manufacturers designing new cars or car buyers who want to imagine a car with custom features. It also shows how machine learning can be used for practical, real-world problems — not just for classification or recommendations, but for generating realistic, missing values.

Existing Questions/prior related works:

There is little to no public work focused on predicting multiple missing car attributes from partial input, especially using multi-output models. However, there are several models that predict a single attribute after being given many other values, such as the car price prediction model and the Vehicle type classifications.

Dataset:

We used a car-specification dataset from Kaggle , which contains technical and market information about a wide range of cars. The data is tabular, with each row representing a vehicle model and each column representing a car attribute. Of the 57 columns in the dataset, our model makes use of the 21 columns with the least null values and the most value to our model.
The features in this dataset used for our project include:
 top speed (mph), acceleration 0-62 mph (0-100 kph), fuel capacity (gallons), length (in), width (in), height (in), Ground Clearance (in), Cargo Volume (cuFT), Unladen Weight (lbs), Gross Weight Limit (lbs), combined mpg, city mpg, highway mpg, displacement (cm3), power(hp), power(rpm), torque (lb-ft), body style, segment, fuel, drive type



	Model	Serie	Company	Body style	Segment	Productior	Cylinders	Displacem	Power(HP)	Power(BHI	Power(KW	Torque(lb-f	Torque(Nm	Electrical r	Electrical r	Fuel Syster	Fuel	Fuel capac	Top Speed	Accelerati
0	AC Ace (19	Ace	AC	Convertibl	Coupe Cab	1993, 1994	V8	4942 cm3	260 HP @	256 BHP @	191.2 KW (	320 lb-ft @	434 Nm @ 3250 RPM			Multipoint	Gasoline	21.1 gallon	140 mph (225 km/h)	
1	AC Aceca (	Aceca	AC	CoupÃ© (tv	Coupe	1998, 1999	V8	3506 cm3	350 HP @	345 BHP @	257.4 KW (	300 lb-ft @	407 Nm @ 4000 RPM			Multipoint	Gasoline	23.8 gallon	155 mph (2	5.6 s
2	AC Aceca (	Aceca	AC	CoupÃ© (tv	Coupe	1998, 1999	V8	3506 cm3	350 HP @	345 BHP @	257.4 KW (	300 lb-ft @	407 Nm @ 4000 RPM			Multipoint	Gasoline	23.8 gallon	155 mph (2	5.6 s
3	AC Cobra	Cobra	AC	CoupÃ© (tv	Roadster &	1965, 1966	V8	4727 cm3	280 HP @	276 BHP @	205.9 KW (	314 lb-ft @	426 Nm @ 3400 RPM				Gasoline	18 gallons	140 mph (2	4.7 s
4	AC Cobra 3	Cobra	AC	CoupÃ© (tv	Roadster &	1965, 1966	V8		446.1 HP @	440 BHP @	328.1 KW @ - RPM						Gasoline	18 gallons (68.1 L)		
5	AC Cobra	Cobra	AC	CoupÃ© (tv	Roadster &	1963, 1964	V8	4727 cm3	271 HP @	267 BHP @	199.3 KW (	312 lb-ft @	423 Nm @ 3400 RPM				Gasoline	18 gallons	141 mph (2	5.3 s
6	AC 428 Co	Frua Conve	AC	Convertibl	Coupe Cab	1966, 1967	V8	7017 cm3	350 HP @	345 BHP @	257.4 KW (	462 lb-ft @	626 Nm @ 2800 RPM			Carburetor	Gasoline			
7	AC 428 Co	Frua Conve	AC	Convertibl	Coupe Cab	1966, 1967	V8	7017 cm3	350 HP @	345 BHP @	257.4 KW (	462 lb-ft @	626 Nm @ 2800 RPM			Carburetor	Gasoline			
8	AC 428 Co	Frua Coup	AC	CoupÃ© (tv	Coupe	1966, 1967	V8	7017 cm3	350 HP @	345 BHP @	257.4 KW (	462 lb-ft @	626 Nm @ 2800 RPM			Carburetor	Gasoline			
9	AC 428 Co	Frua Coup	AC	CoupÃ© (tv	Coupe	1966, 1967	V8	7017 cm3	350 HP @	345 BHP @	257.4 KW (	462 lb-ft @	626 Nm @ 2800 RPM			Carburetor	Gasoline			
10	ACURA MD	MDX	ACURA	SUV (Sport	Medium SU	2021, 2022	V6		290 HP @	286 BHP @	213.3 KW (	267 lb-ft @	362 Nm @ 4700 RPM			Direct inje	Gasoline	18.5 gallons (70.0 L)		
11	ACURA MD	MDX	ACURA	SUV (Sport	Medium SU	2021, 2022	V6		290 HP @	286 BHP @	213.3 KW (	267 lb-ft @	362 Nm @ 4700 RPM			Direct inje	Gasoline	18.5 gallons (70.0 L)		
12	ACURA MD	MDX	ACURA	SUV (Sport	Medium SU	2018, 2019	V6	3500 cm3	290 HP @	286 BHP @	213.3 KW (	267 lb-ft @	362 Nm @ 4700 RPM			Direct Inje	Gasoline	19.5 gallons (73.8 L)		
13	ACURA MD	MDX	ACURA	SUV (Sport	Medium SU	2018, 2019	V6	3500 cm3	290 HP @	286 BHP @	213.3 KW (	267 lb-ft @	362 Nm @ 4700 RPM			Direct Inje	Gasoline			
14	ACURA MD	MDX	ACURA	SUV (Sport	Medium SU	2018, 2019	V6	3500 cm3	260 HP @	256 BHP @	191.2 KW (	218 lb-ft @	296 Nm @	34.6 kw (47	109.2 lb-ft	Direct Inje	Hybrid	19.3 gallons (73.1 L)		
15	ACURA MD	MDX	ACURA	SUV (Sport	Medium SU	2016, 2017	V6	3500 cm3	294 HP @	290 BHP @	216.3 KW (	267 lb-ft @	362 Nm @ 4700 RPM			Direct Inje	Gasoline			
16	ACURA MD	MDX	ACURA	SUV (Sport	Medium SU	2016, 2017	V6	3500 cm3	294 HP @	290 BHP @	216.3 KW (	267 lb-ft @	362 Nm @ 4700 RPM			Direct Inje	Gasoline			
17	ACURA MD	MDX	ACURA	SUV (Sport	Medium SU	2016, 2017	V6	3500 cm3	260.6 HP @	257 BHP @	191.6 KW (	218 lb-ft @	296 Nm @ 5000 RPM			Direct Inje	Hybrid			
18	2013 Acura	MDX	ACURA	SUV (Sport	Medium SU	2013, 2014	V6	3500 cm3	294 HP @	290 BHP @	216 KW @	267 lb-ft @	362 Nm @ 4500 RPM			Direct Inje	Gasoline			
19	2013 Acura	MDX	ACURA	SUV (Sport	Medium SU	2013, 2014	V6	3500 cm3	294 HP @	290 BHP @	216 KW @	267 lb-ft @	362 Nm @ 4500 RPM			Direct Inje	Gasoline			

Raw Dataset Snippet

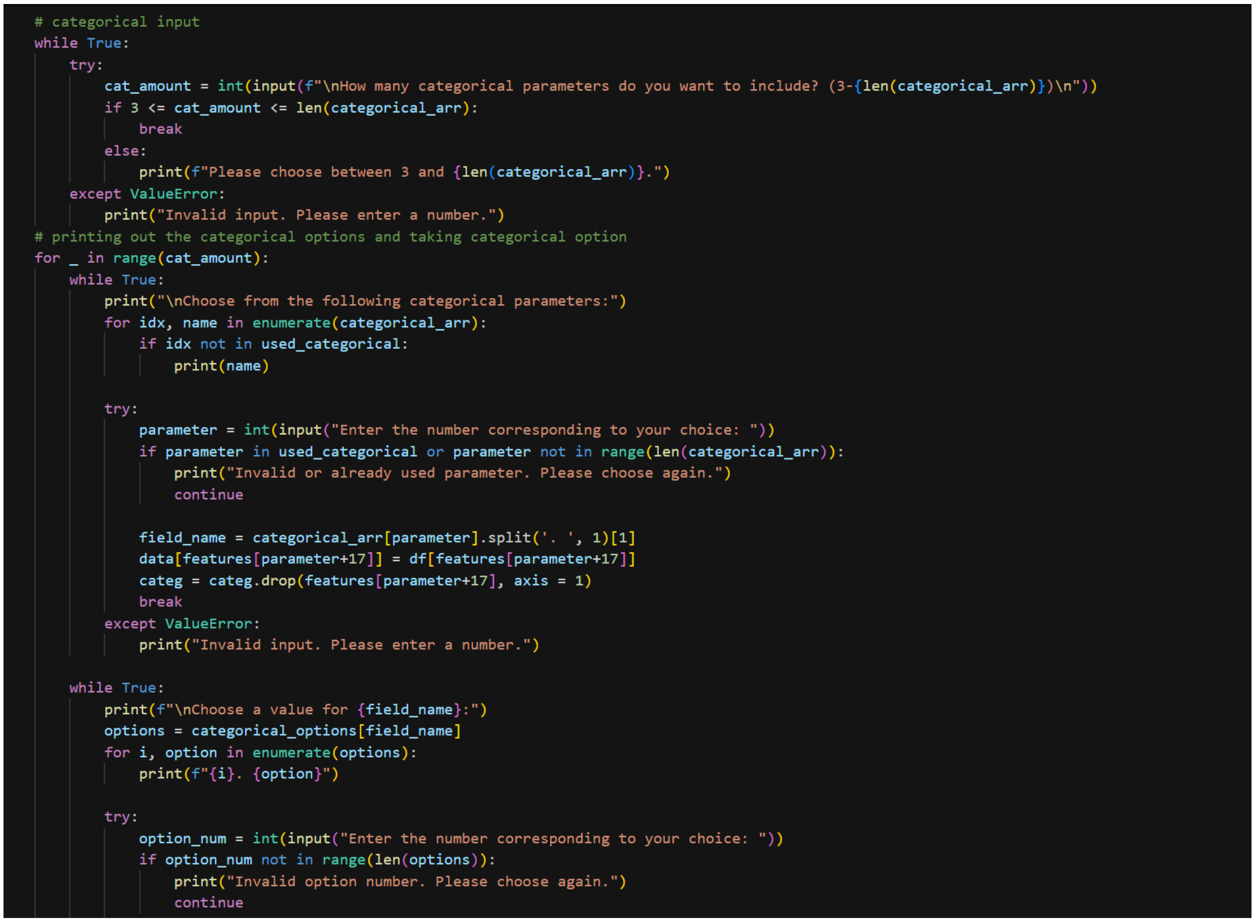
Model:

Our project utilizes a random forest model in order to predict the attributes of a car based on pre-selected features from the user. Specifically, we use the RandomForestClassifier, RandomForestRegressor, MultiOutputClassifier, and MultiOutputRegressor from the sklearn.ensemble package.

RandomForestClassifier determined the value of categorical values by creating predictions drawn from taught data patterns, while RandomForestRegressor created regression models based on the same trained data in order to determine continuous values. Both models are built upon the random forest model, which works by first finding the most important feature through methods such as gini indexes, and then deducing a split point that best evenly splits the dataset. The decision tree then repeats this process for all the features in the dataset until any given prediction can be funneled into these decision trees, giving a predicted value. The random forest model then creates the optimal number of decision trees based on a mathematical model, and then runs the prediction on all of the trees. The most common answer for categorical, or the average answer for regression, will be the final predicted value of the model. Due to how our model predicts multiple continuous values and could have predicted multiple categorical values, we utilized MultiOutputClassifier and MultiOutputRegressor to return multiple predicted values at once.

Feature Space:

Our project presents the user with 21 possible features, 4 categorical and 17 continuous. Through analysis, we determined that 6 features, 3 categorical and 3 continuous, were the minimum number of required values that needed to be given to the model in order to accurately predict the remaining 15 values. As such, the feature space our model is trained on changes based on the users requirements, requiring a higher degree of modularity than typical Random Forest Models.



```
# categorical input
while True:
    try:
        cat_amount = int(input(f"\nHow many categorical parameters do you want to include? (3-{len(categorical_arr)})\n"))
        if 3 <= cat_amount <= len(categorical_arr):
            break
        else:
            print(f"Please choose between 3 and {len(categorical_arr)}.")
    except ValueError:
        print("Invalid input. Please enter a number.")
# printing out the categorical options and taking categorical option
for _ in range(cat_amount):
    while True:
        print("\nChoose from the following categorical parameters:")
        for idx, name in enumerate(categorical_arr):
            if idx not in used_categorical:
                print(name)

        try:
            parameter = int(input("Enter the number corresponding to your choice: "))
            if parameter in used_categorical or parameter not in range(len(categorical_arr)):
                print("Invalid or already used parameter. Please choose again.")
                continue

            field_name = categorical_arr[parameter].split('. ', 1)[1]
            data[features[parameter+17]] = df[features[parameter+17]]
            categ = categ.drop(features[parameter+17], axis = 1)
            break
        except ValueError:
            print("Invalid input. Please enter a number.")

    while True:
        print(f"\nChoose a value for {field_name}:")
        options = categorical_options[field_name]
        for i, option in enumerate(options):
            print(f"{i}. {option}")

        try:
            option_num = int(input("Enter the number corresponding to your choice: "))
            if option_num not in range(len(options)):
                print("Invalid option number. Please choose again.")
                continue
```

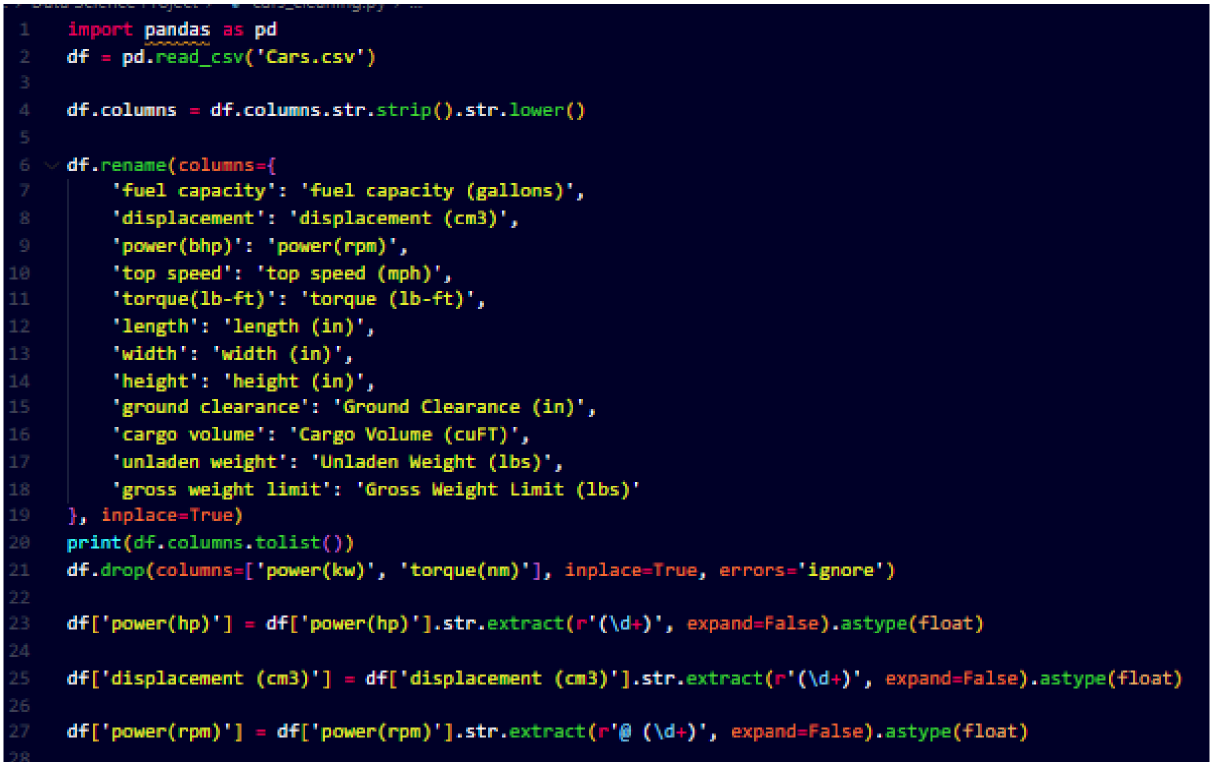
Feature Space Determination Code for Categorical Values

Implementation:

The Car Attribute prediction model has 6 distinct steps. Step 1 involves downloading all packages and data structures utilized in the project, such as the Pandas, Numpy, Seaborn, and the models from sklearn.ensemble.

In Step 2, we read the dataset included in the project folder and clean it by:

- Renaming the columns for legibility.
- Adding measurement units (like hp, lbs, inches) to column names for clarity and precision.
- Removed units from individual rows, keeping the numbers.
- Deleting duplicate columns.
- Switching to imperial units for consistency.



```python
import pandas as pd
df = pd.read_csv('Cars.csv')

df.columns = df.columns.str.strip().str.lower()

df.rename(columns={
    'fuel capacity': 'fuel capacity (gallons)',
    'displacement': 'displacement (cm3)',
    'power(bhp)': 'power(rpm)',
    'top speed': 'top speed (mph)',
    'torque(lb-ft)': 'torque (lb-ft)',
    'length': 'length (in)',
    'width': 'width (in)',
    'height': 'height (in)',
    'ground clearance': 'Ground Clearance (in)',
    'cargo volume': 'Cargo Volume (cuFT)',
    'unladen weight': 'Unladen Weight (lbs)',
    'gross weight limit': 'Gross Weight Limit (lbs)'
}, inplace=True)
print(df.columns.tolist())
df.drop(columns=['power(kw)', 'torque(nm)'], inplace=True, errors='ignore')

df['power(hp)'] = df['power(hp)'].str.extract(r'(\d+)', expand=False).astype(float)

df['displacement (cm3)'] = df['displacement (cm3)'].str.extract(r'(\d+)', expand=False).astype(float)

df['power(rpm)'] = df['power(rpm)'].str.extract(r'@ (\d+)', expand=False).astype(float)
```

Dataset cleaning code snippet

Following this, step 3 has the project ask the user for their attribute values. The features corresponding to the attributes from the dataset in step 2 are then put in a training dataset, with the remaining features separated based on if they are categorical or continuous for prediction and testing purposes.

Step 4 is where the model is finally trained. Of the 3 data frames created from step 3, the training data and the categorical predictors are encoded from string objects into distinct integers. In addition, these 3 datasets undergo an 80:20 split into training and testing data, with the testing data used in step 5. The training data instead trains both the classifier and the regression model, but the classifier is given the categorical predictors and the regression model is given the continuous predictors.

This leads into step 5, where we test and analyze the accuracy of the model. Due to how the feature space is highly mutable, model accuracy can vary greatly, which is why after significant analysis 6 given attributes were considered the minimum to have a decently accurate output, with more given attributes improving the model accuracy. In this step the model predicts the value of the testing data, after which the predicted values are compared against the actual values in the testing categorical predictors and the testing continuous predictors, with the accuracy of the categorical predicted values and the MAPE of the continuous values being stated to the user. This lets the user fine tune what values they give to the model in following queries, allowing them to make full use of the car attribute predictor as a tool to determine a hypothetical car's specifications.

Step 6 is the final step, and returns the predicted attributes based on the user's given values.

General Evaluation:

This model was the right pick for what was intended, however finding values and parameters that correlate was key in revealing accuracy. For what we were trying to accomplish, this gave great insight into what goes into making a car, and why using a prediction model to construct values for a car may not be the best. This model tried to use different values such as R^2, mean squared error, or variance. These values did not reveal any information about the data worthwhile, so we settled on values such as accuracy, giving a 10% margin of error, or mean absolute percentage error.

Results:

The entire list of statistics used for the results found in this section can be found here: https://docs.google.com/document/d/1tJxRJTKWSxhuz681w3JnWBhNpqS06zV3WcthEzw8EOI/edit?usp=sharing

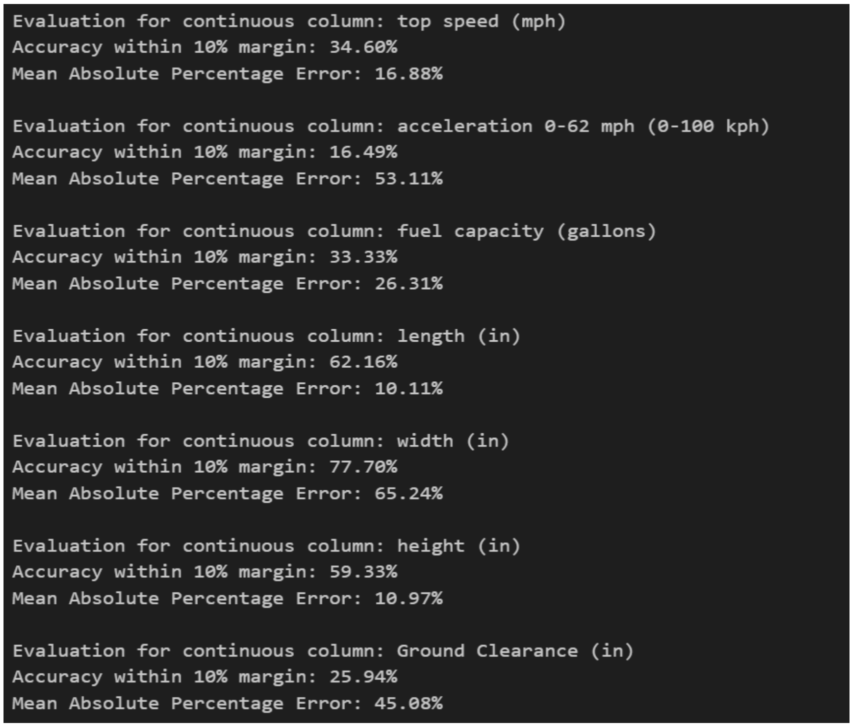
Our model worked best using certain parameters and a certain number of parameters, depending on how accurate the result is or how fast you want the model to run. Our testing helps show that the more input that is given to the model, the more accuracy increases. An example of this testing was how inputting 13 continuous or categorical values gives about an overall accuracy of 88%, compared to inputting 3 continuous variables or categorical values achieving an overall accuracy of 62%.

Analysis of performance metrics

The test offers valuable insights into the accuracy of the random forest model across different continuous and categorical variables. We decided to use a mean absolute percentage error and the accuracy within a 10% margin of error, due to how these values can reveal how accurate we are at predicting a result. Some other values that we tried to use included a R^2 value, however having some values such as torque and width, that hardly correlate to any other values that are used in this project.

Overall performance patterns

Our model demonstrates highly inconsistent performance across different vehicle parameters, however when examining on closer inspection and comparing it to the dataset, it would make sense why these values, such as width, are so inaccurate. Most of our data that resulted in a smaller mean absolute percentage error has data that correlates with other values, however a parameter such as width would score a higher mean absolute percentage error due to cars always maintaining the same width to fit within the space needed for a road. However a mean absolute percentage error such as acceleration and ground clearance would score a smaller value around 15% ranging to 27%.

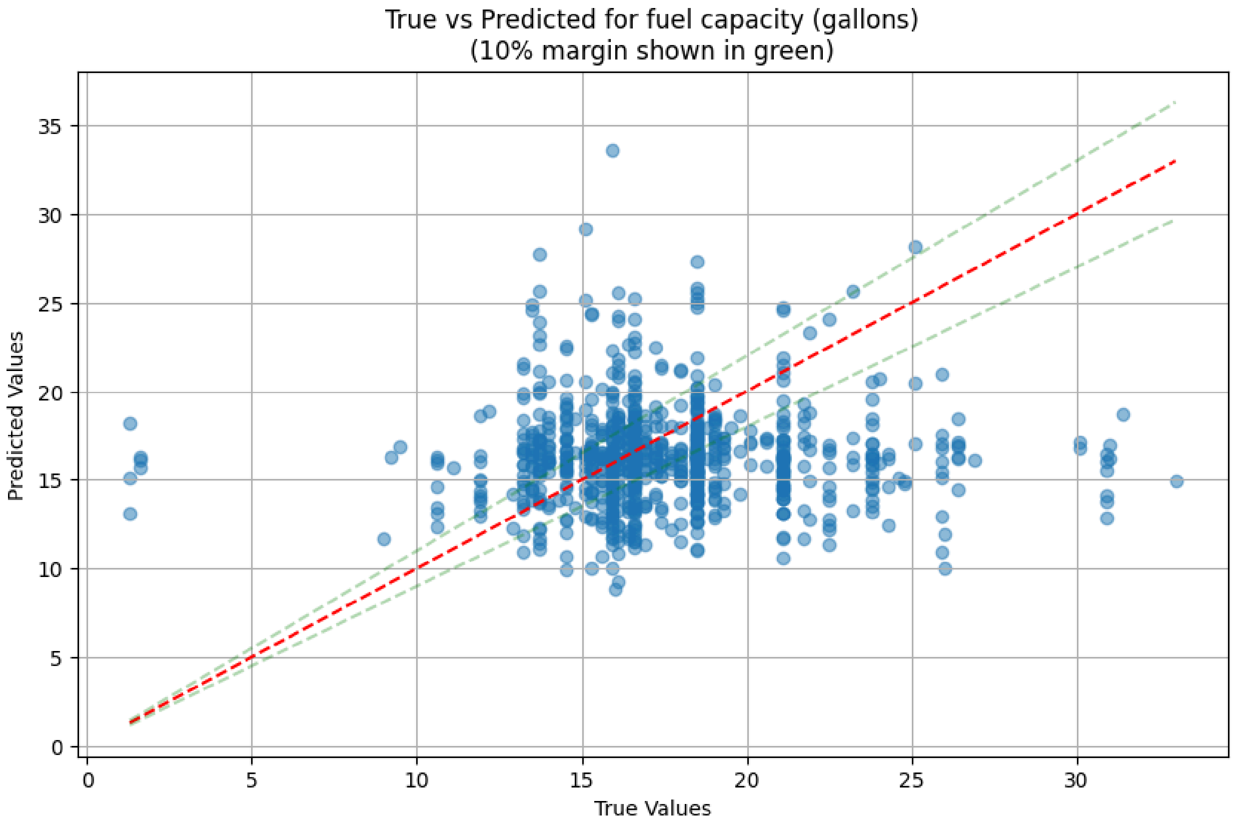


```
Evaluation for continuous column: top speed (mph)
Accuracy within 10% margin: 34.60%
Mean Absolute Percentage Error: 16.88%

Evaluation for continuous column: acceleration 0-62 mph (0-100 kph)
Accuracy within 10% margin: 16.49%
Mean Absolute Percentage Error: 53.11%

Evaluation for continuous column: fuel capacity (gallons)
Accuracy within 10% margin: 33.33%
Mean Absolute Percentage Error: 26.31%

Evaluation for continuous column: length (in)
Accuracy within 10% margin: 62.16%
Mean Absolute Percentage Error: 10.11%

Evaluation for continuous column: width (in)
Accuracy within 10% margin: 77.70%
Mean Absolute Percentage Error: 65.24%

Evaluation for continuous column: height (in)
Accuracy within 10% margin: 59.33%
Mean Absolute Percentage Error: 10.97%

Evaluation for continuous column: Ground Clearance (in)
Accuracy within 10% margin: 25.94%
Mean Absolute Percentage Error: 45.08%
```

Example: MAPE Statistics

Comparing our method

This project seems to be one of a kind, as there do not seem to be any other projects predicting the parameters of a car, so we do not have any other things to compare it to. This method does not seem usable, as we do not account for many parts of a car such as the aerodynamics and cost. When comparing our results to real cars put out on the market, our cars would seem to have many defects if the predicted results actually constituted a car. For example our predicted amount of fuel capacity could drastically change the workings of the car if we are supposed to store more than it can hold, yet our predictions do not seem to be close.

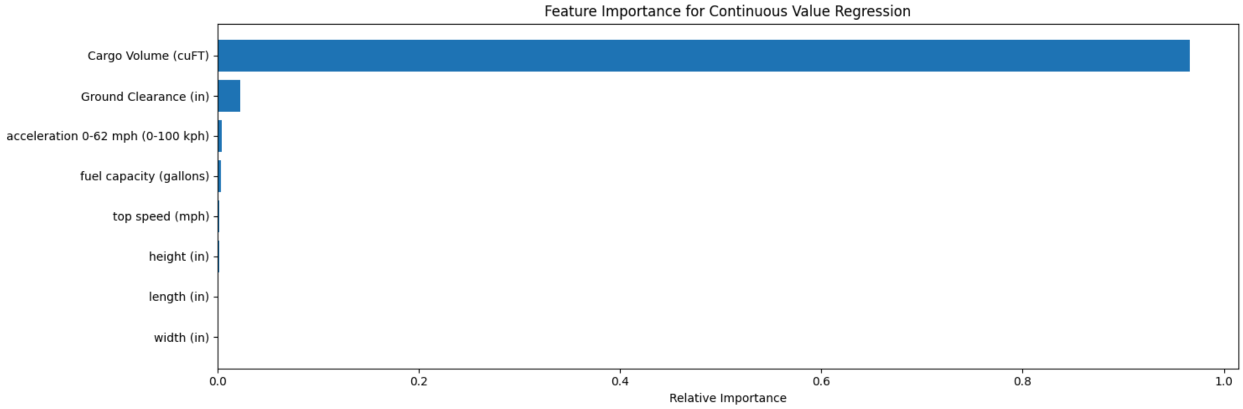


Specific Analysis:

Going into further detail about the best number of predictors, what we found is that the more predictors the model is given, the higher the accuracy is. This is due to how it tends to eliminate variance due to having more trees to predict the result. Having more predictors gives the trees more options to split on, and also by allowing a new predictor, this gives more options if the model is too simple, or if it is underfitting.

Best Predictors

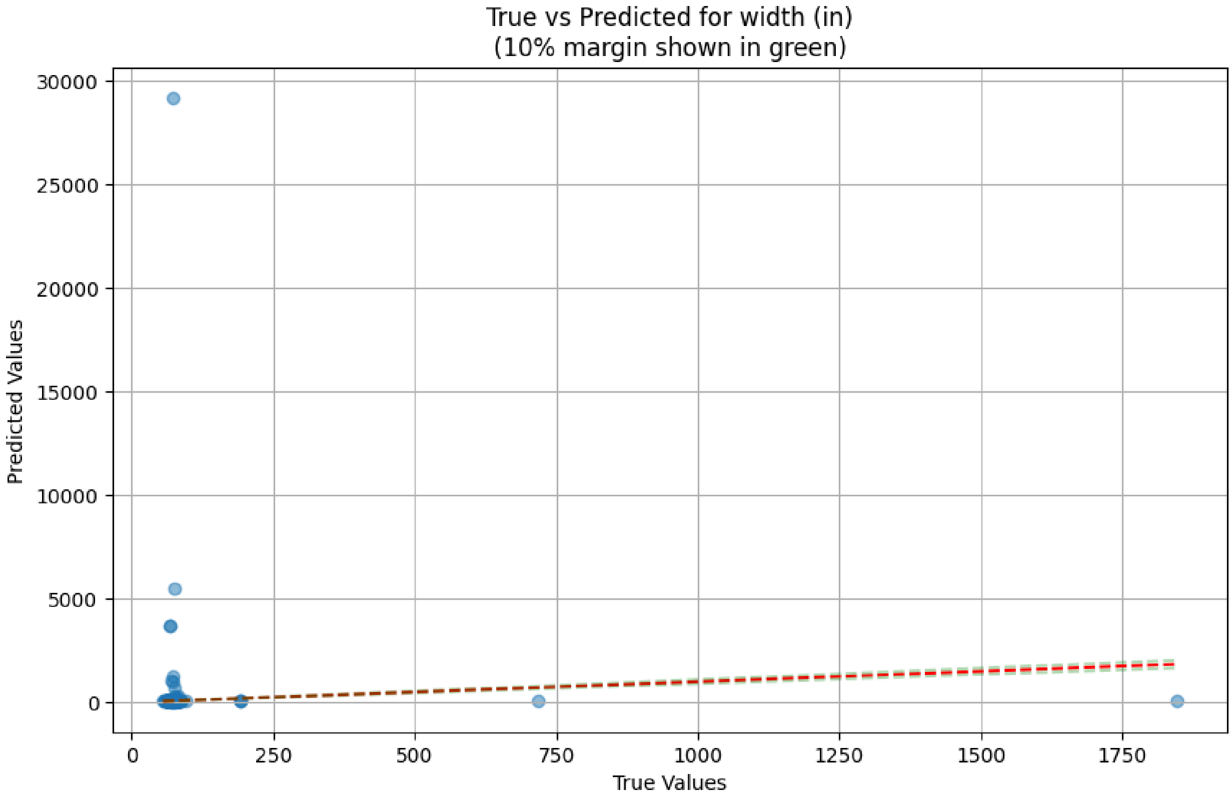
After long and extensive testing, the best continuous predictor to have would probably be cargo volume or unladen weight, due to how it correlates to many other qualities of a car. Since a lot of the weight depends on the other values, this gives a good framework for the model to base it off of, allowing for a more accurate prediction.



This graph indicates the overwhelming importance of cargo volume, while in other graphs using different features length and width were recorded as more important, making the above graph the most significant.

Worst predicted

The value that is the worst is the width. Since it hardly correlates with other parts of the car, this makes it hard for the random forest model to predict the width of a car. The model tries to find a value that would correlate with the data given, yet the width of the car is always consistent since they have to be to fit within a lane.



Project Pivot Explanation:

Our project at first was going to search through a database of cars to provide a user the ideal car given what qualities the user wants for a car. This was mainly due to our shared interest in cars and the desire to develop a practical tool that could assist potential car buyers in making informed decisions. However after consulting our professor, we realized that what we wanted to create was a database searching system, and that this would not align with what we learned in the class. After discussing as a team what we should change our project to, we decided that we should predict different qualities of a car given what qualities the user wants for the car. This way we could use a prediction model while still maintaining the same interest we all shared for cars.