Predicting Quarter Mile Times Using Machine Learning

Author

Aniruth Narayanan

Abstract

The quarter mile time is an essential measure of how fast a vehicle is; a method to estimate this time during the building and planning process allows necessary adjustments before racing. Considering the variance and relatively outdated numbers used, this paper uses various machine learning techniques to determine a more precise way to find times, generalizable across many vehicles. The result was $ET = 5.8566\sqrt[3]{\frac{WT}{HP}}$, which supports the physics equations used to derive the ideal relationship while accounting for real-world air resistance and traction issues.

1 Introduction

Drag racing is a key measure for vehicle enthusiasts about performance, as it is standardized and over a significant distance with high popularity, especially in the United States. Many older vehicles from a handful of brands, such as Ford, Chevrolet, Dodge, and Chrysler are revamped with improved transmissions, engines, and wheels in the hopes of creating a fast vehicle. Shorthand for what it means for a vehicle to be fast is often linked to drag race times; a sub-10 vehicle is one that takes under 10 seconds to complete a quarter mile. When taking vehicles to the drag strip, however, it is often difficult to perform large scale work beyond maintenance. Therefore, a way to determine this time before transporting vehicles for testing proves invaluable.

All code is available at https://github.com/aniruthn/predicting-quarter-mile-times

2 Related Work

There is significant prior work dating back to the mid twentieth century relating to this problem. The old racers' rule of thumb used to relate miles per hour (at the end of the quarter mile) to the estimated time. This is not very useful, since finding the miles per hour is equally difficult to estimate.

Patrick Hale determined that the relationship was $ET = 5.825\sqrt[3]{\frac{WT}{HP}}$[1] in 1986 using a variety of variables, while others such as Geoffrey T. Fox found other relationships such as $ET = 6.269\sqrt[3]{\frac{WT}{HP}}$. At the very least, the variation occurs in the constant multiplied by the $\sqrt[3]{\frac{WT}{HP}}$ term. Thus, the goal of this paper is to find the coefficient value that best mirrors the real-world performance of vehicles using their weight and horsepower (Lucius 17)[2].

Regarding Hale's equation, there is a theoretical backing that was first published in HOT ROD magazine in November 1973, page 86. It calculates minimum horsepower for a vehicle to accelerate the weight of a vehicle over a quarter mile at a given time, where he finds the coefficient to come out to roughly 5.818[3].

3 Method

Considering there is a physical backing towards finding a linear relationship between time and $\sqrt[3]{\frac{WT}{HP}}$, it makes most sense to use this relationship for the modelling (the degree of the root will not be adjusted). The sklearn library in Python is used; more advanced techniques which can be classified as neural networks are not due to restrictions in the usable data that can be found online efficiently.

[1] ET refers to elapsed time across the quarter mile (in seconds), WT refers to weight (in pounds), and HP refers to horsepower.

[2] Much of this section is paraphrased from Lucius; in that, there is more work done to estimate various lines and curves of best fit which is mostly irrelevant beyond the physics equations for the purposes of this paper.

[3] The full equations can be found here: http://www.stealth316.com/misc/performance_mccaul_equations.doc

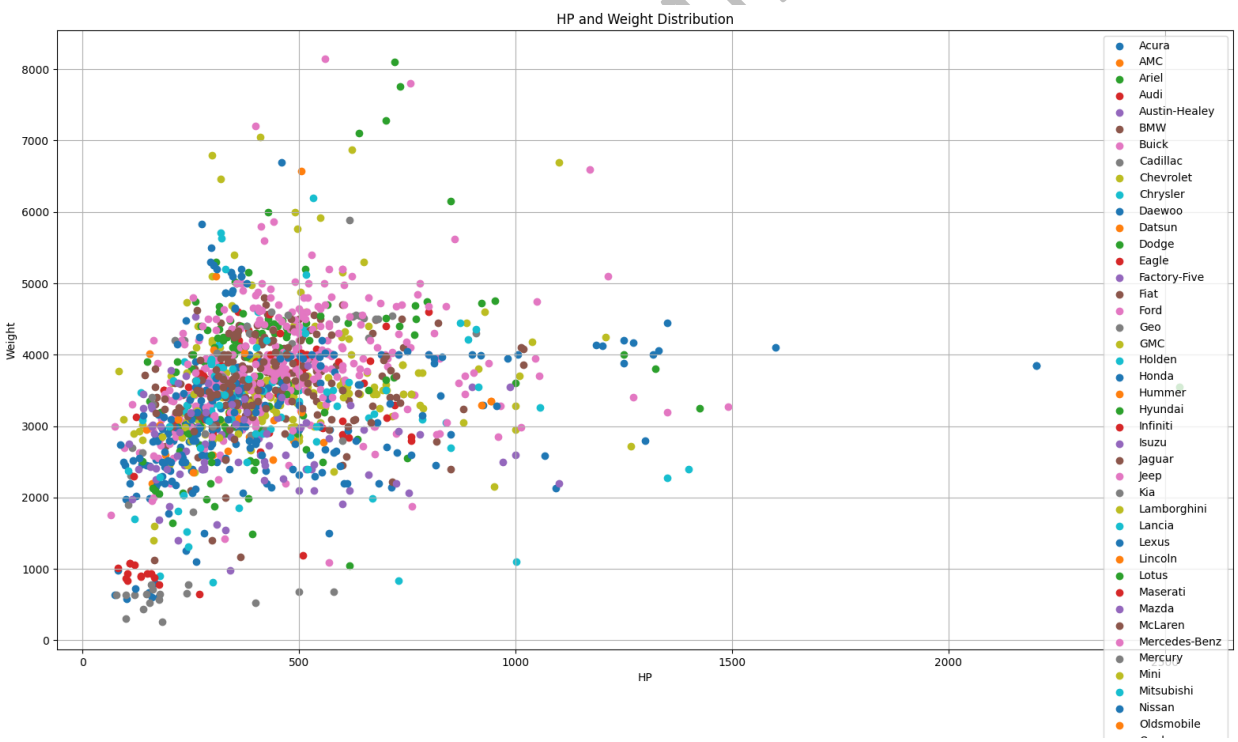
The data is scraped from an online database of quarter and eighth mile timeslips (dragtimes.com). I created a web scraper using Python (Selenium and BeautifulSoup4) which finds timing information, vehicle information, and the crucial weight and horsepower information for training the models.

4 Creating Dataset and Feature Engineering

The dataset is created using Selenium's Webdriver which gets various links to be parsed and manipulated by the BeautifulSoup library. The full code is available at the GitHub link from the introduction. As a quick summary, it first gets every brand of vehicle on the site, transforming this into a list of records for every vehicle timeslip within each brand. Lastly, it opens a .csv to add the data for every timeslip (all 27,000+ timeslips on the website) contingent on the timeslip having information for horsepower and weight. Of the many vehicles, only 1,660 timeslips had this information; thus, the dataset was of a moderate size, but not quite large.

Next, the data is cleaned based on the feature data. Vehicles using nitrous and those that have been modified to a significant degree or do not reflect the average vehicle (such as motorcycles, snowmobiles, electric vehicles) are dropped. Next, the data is restricted to be within two standard deviations of the mean for both horsepower and weight.

The horsepower and weight of the vehicles remaining classified by make is represented in the graph below:



There is some minor clustering by make of horsepower and weight, but not very significant (some makes have the same or very close color due to the high number of different makes).

Feature engineering used a lambda function to transform the horsepower and weight information into a new column, 'Converted', which was then restricted to be within two standard deviations of the mean. This feature represents $\sqrt[3]{\frac{WT}{HP}}$ and is key since it correlates with the ET[4].

However, there are a few outliers that need to be manually removed which was determined through usage of the matplotlib library. These are removed for cars that are too slow, too fast, or too heavy for their given converted feature.

The models are variations on the linear model, namely linear regression, Bayesian Ridge, Lasso, ElasticNetCV, and a Decision Tree Regressor to make comments about the linear model in general. The linear models were chosen with linear regression serving as the classic algorithm, Bayesian Ridge chosen for performance, and Lasso along with ElasticNetCV chosen based on sklearn's recommendations for models to try regarding the data size and regression.

The data is split 75% training and 25% testing; validation data is not necessary to prevent overfitting in this case due to the physical certainty behind the correlation present.

Each model is retrained with random allocations of data 1000 times, with the best accuracy and information saved for future information. Additionally, on the linear models, the intercept is set to 0 to mirror the original physics equation (which has no separate constant term).

5 Results

The results of each model are displayed below:

Model Name	Accuracy	Coefficient
Linear Regression	0.8684	5.8566
Bayesian Ridge	0.8680	5.8620
Lasso	0.8640	5.8360
Elastic Net CV	0.8559	5.8243
Decision Tree	0.7699 (max_depth=2)	N/A[5]

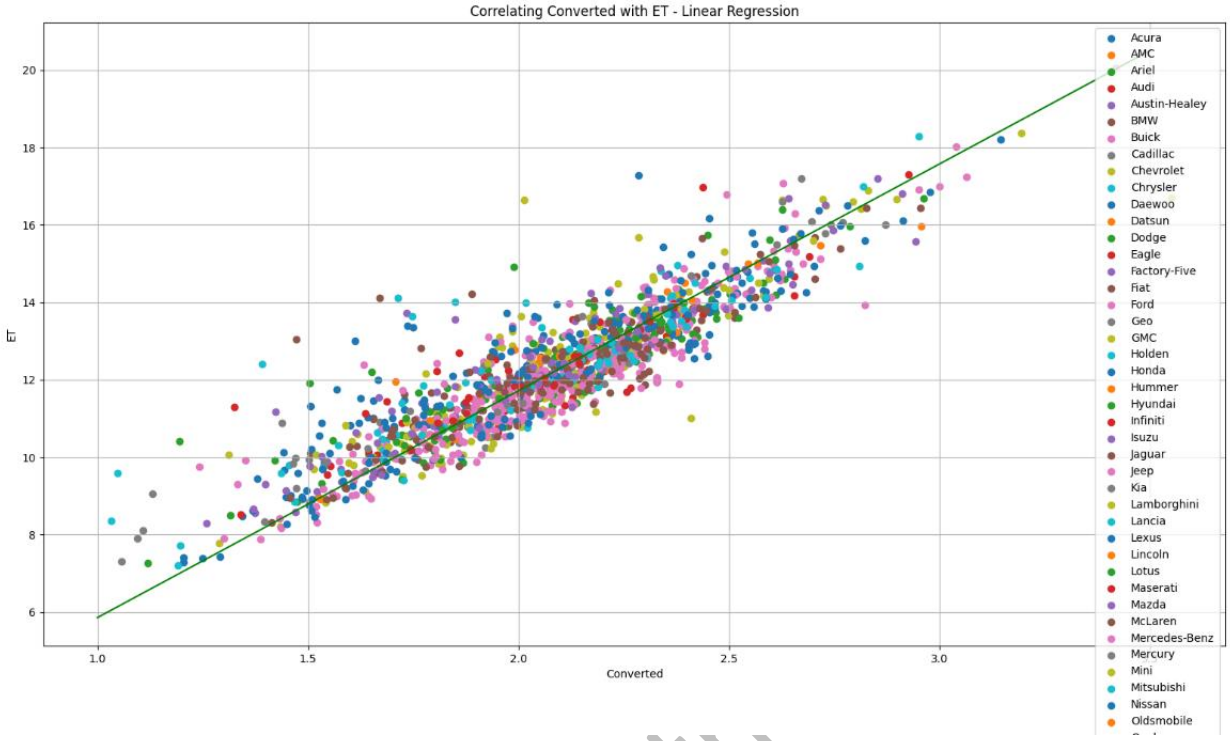
The most accurate value (albeit marginally) is about 5.8566.

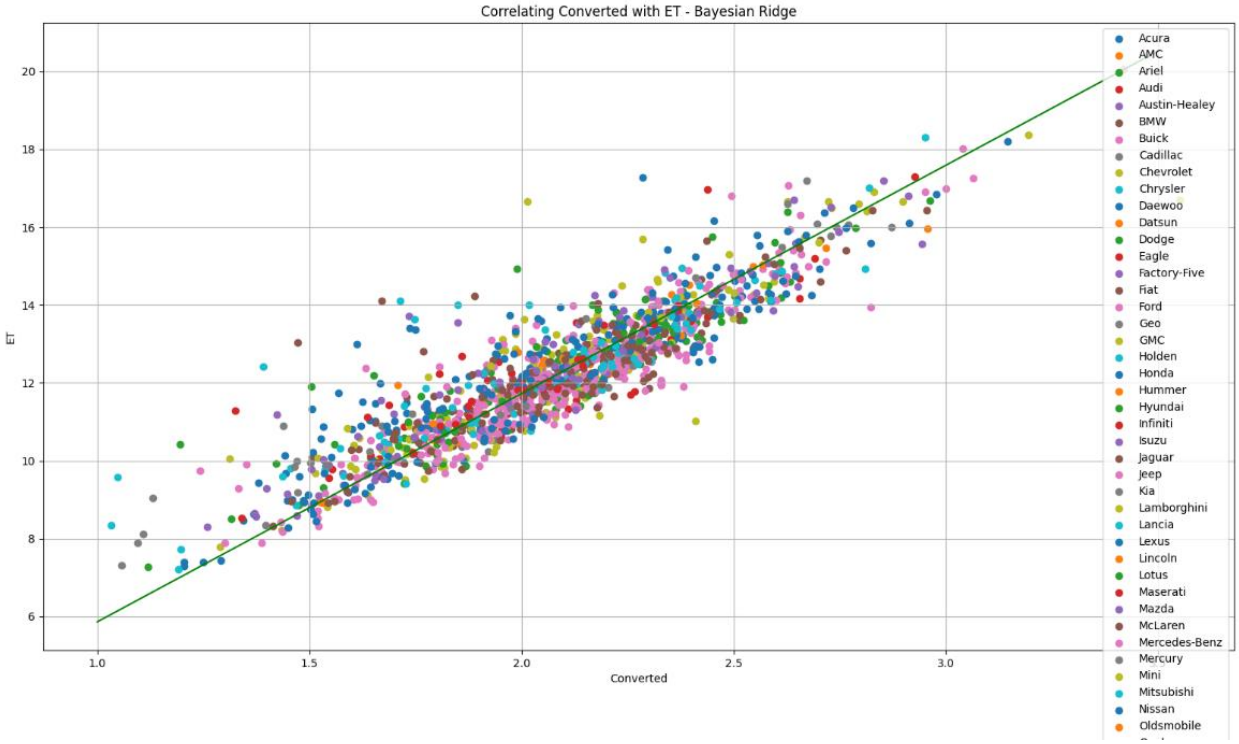
The graphs for each model, in the same order as in the table, show the clustering of the data using different colors correlating to different makes, with the line representing the model's correlation.[6]

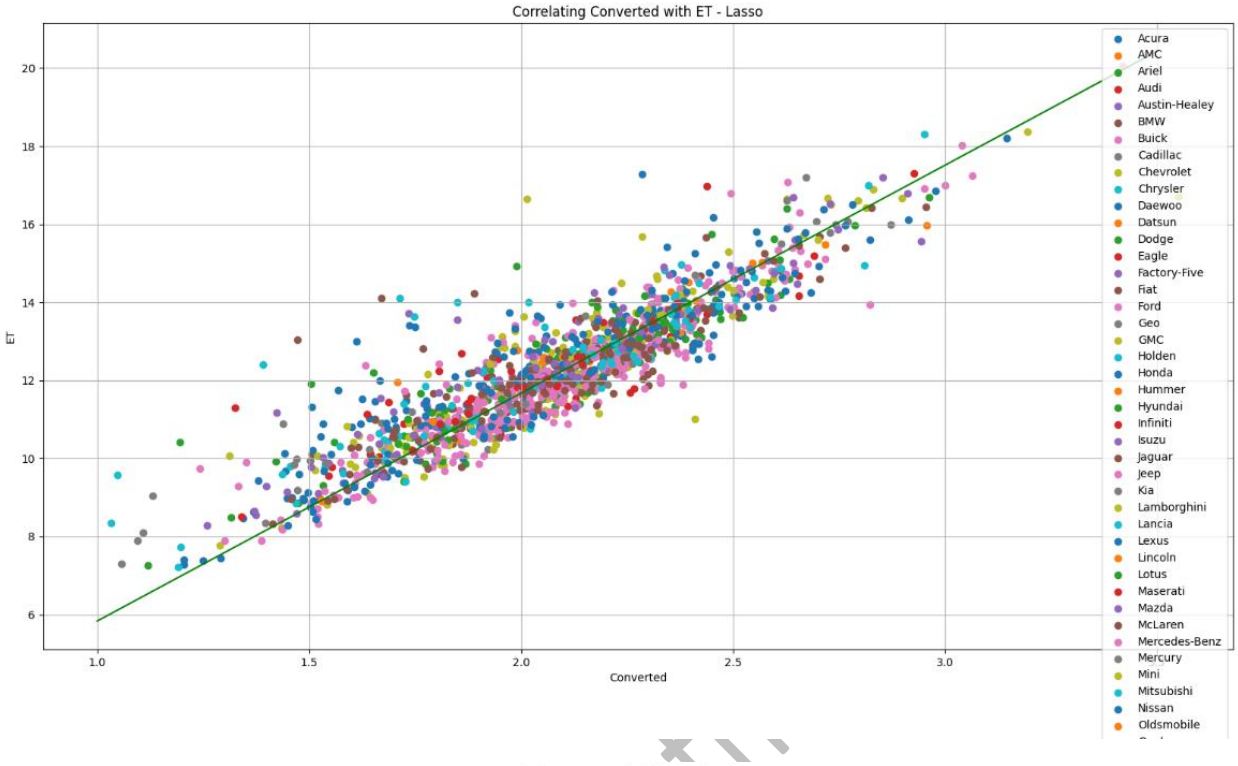
[4] This engineered feature is referred to as 'converted feature' for the rest of the paper.

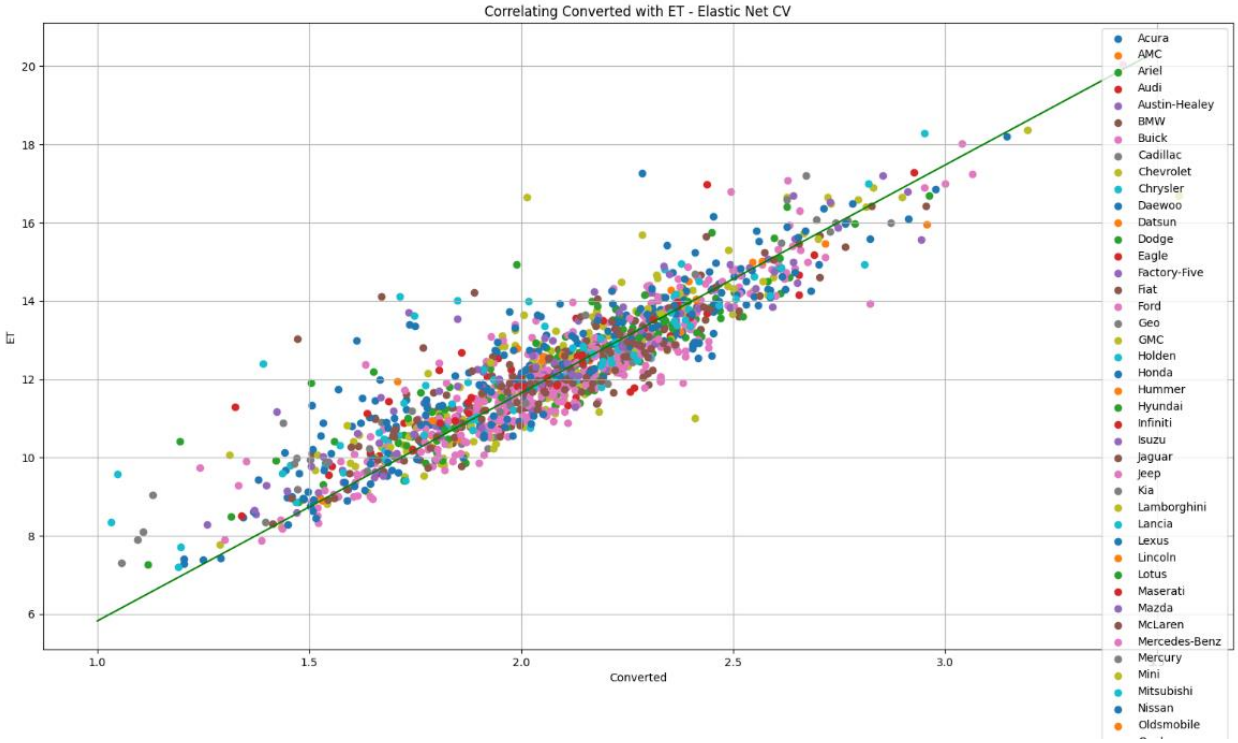
[5] Decision trees do not follow the coefficient pattern as the linear models do.

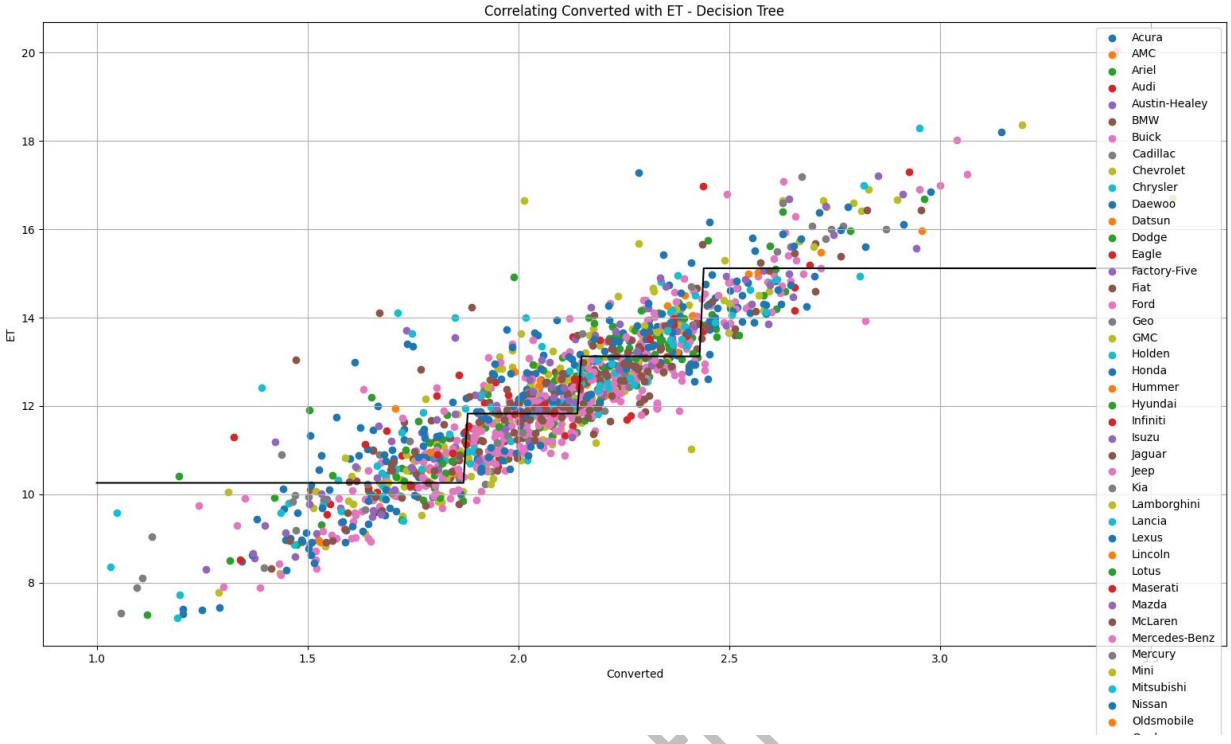
[6] Each graph is shown separately as individual lines are difficult to distinguish when graphing them together.


Correlating Converted with ET - Linear Regression
ET
Converted
Acura
AMC
Ariel
Audi
Austin-Healey
BMW
Buick
Cadillac
Chevrolet
Chrysler
Daewoo
Datsun
Dodge
Eagle
Factory-Five
Fiat
Ford
Geo
GMC
Holden
Honda
Hummer
Hyundai
Infiniti
Isuzu
Jaguar
Jeep
Kia
Lamborghini
Lancia
Lexus
Lincoln
Lotus
Maserati
Mazda
McLaren
Mercedes-Benz
Mercury
Mini
Mitsubishi
Nissan
Oldsmobile


Correlating Converted with ET - Bayesian Ridge
ET
Converted
Acura
AMC
Ariel
Audi
Austin-Healey
BMW
Buick
Cadillac
Chevrolet
Chrysler
Daewoo
Datsun
Dodge
Eagle
Factory-Five
Fiat
Ford
Geo
GMC
Holden
Honda
Hummer
Hyundai
Infiniti
Isuzu
Jaguar
Jeep
Kia
Lamborghini
Lancia
Lexus
Lincoln
Lotus
Maserati
Mazda
McLaren
Mercedes-Benz
Mercury
Mini
Mitsubishi
Nissan
Oldsmobile


Correlating Converted with ET - Lasso
ET
Converted
Acura
AMC
Ariel
Audi
Austin-Healey
BMW
Buick
Cadillac
Chevrolet
Chrysler
Daewoo
Datsun
Dodge
Eagle
Factory-Five
Fiat
Ford
Geo
GMC
Holden
Honda
Hummer
Hyundai
Infiniti
Isuzu
Jaguar
Jeep
Kia
Lamborghini
Lancia
Lexus
Lincoln
Lotus
Maserati
Mazda
McLaren
Mercedes-Benz
Mercury
Mini
Mitsubishi
Nissan
Oldsmobile


Correlating Converted with ET - Elastic Net CV
ET
Converted
Acura
AMC
Ariel
Audi
Austin-Healey
BMW
Buick
Cadillac
Chevrolet
Chrysler
Daewoo
Datsun
Dodge
Eagle
Factory-Five
Fiat
Ford
Geo
GMC
Holden
Honda
Hummer
Hyundai
Infiniti
Isuzu
Jaguar
Jeep
Kia
Lamborghini
Lancia
Lexus
Lincoln
Lotus
Maserati
Mazda
McLaren
Mercedes-Benz
Mercury
Mini
Mitsubishi
Nissan
Oldsmobile



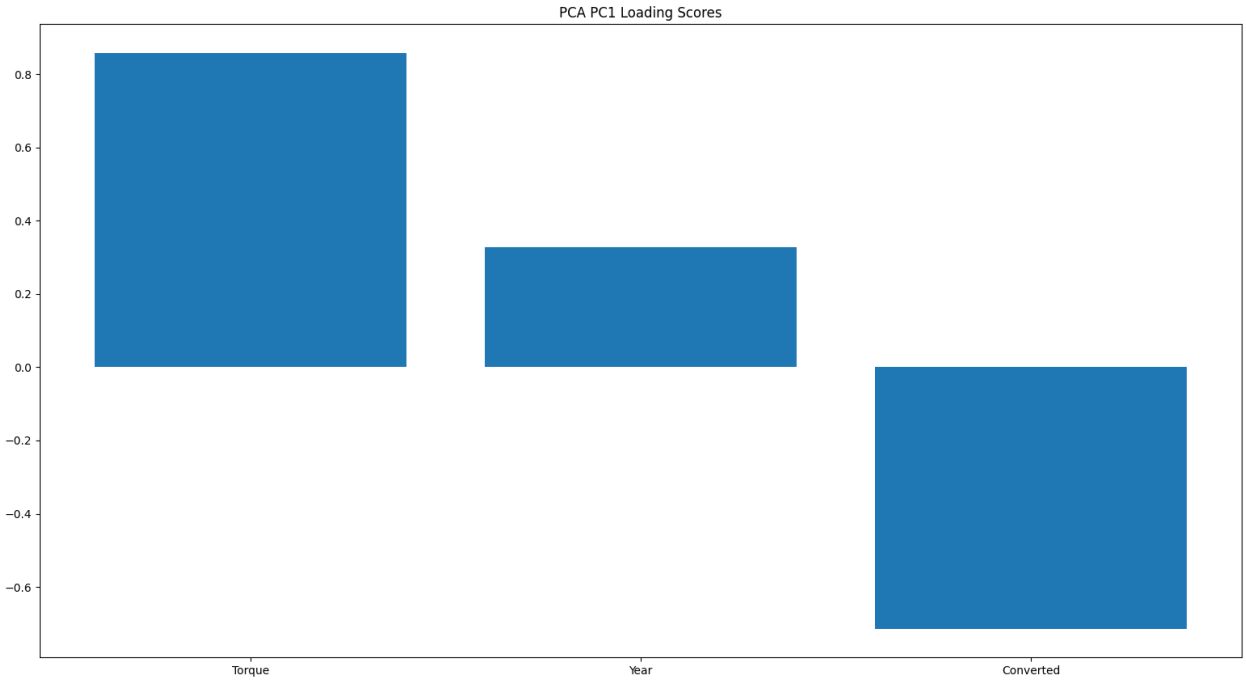
Each graph looks quite similar except for the final decision tree, which appears to be choosing different boundaries and assigning one ET accordingly. The accuracy for it is quite low, which can be artificially raised by using higher depth – but it leads to overfitting issues.

The decision trees are quite arbitrary, however – with some experimentation regarding the degree of the root for the converted feature, the linear models fail compared to the decision tree.

6 PCA and Feature Importance

The dataset minimally requires HP, WT, and ET; for more complex analysis that may not have physics backing, it is useful to determine which features can be removed in the dataset to improve training.

PCA, or Principal Component Analysis, analyzes data to determine where variation occurs. By finding the span of vectors (which can loosely be thought of as a line) that is perpendicular to the variation, change of basis to this new span encapsulates the original variation with fewer dimensions (features). More in-depth explanation of the underlying mathematics can be found online, but for the purposes here, it will be used to assist in feature importance along with coefficient values. The results of PCA testing will be used to confirm the theoretical relationship as the engineered feature from HP and WT being the only required features (the minimum dimension) for the analysis.



The PCA scores for PC1 numerically are 0.858991, 0.328019, and -0.713745 for Torque, Year, and Converted, respectively.

Retraining the linear regression model (as it had highest accuracy) to include torque, year, and converted yields -7.7114×10^{-4}, 1.5910×10^{-3}, and 4.5509, as the coefficients, respectively. This improves accuracy up to 0.89404, although the change in coefficients indicates possible overfitting. While the coefficient for the converted feature is less than it otherwise would be without the other features, the other coefficients are very, very small by orders of magnitude.

Thus, while the variation of torque mirrors that of the converted feature in terms of the first principal component, it is still not a very relevant feature; an interesting proposition that the torque of a vehicle has little if any implication upon a vehicle's ET.

7 Conclusion

$ET = 5.8566\sqrt[3]{\frac{WT}{HP}}$ is the equation with the best coefficient that accurately reflects the real-world behavior of vehicles with varying weight and horsepower from a variety of training techniques. Of the models used, the linear regression model performed best.

One important caveat to this conclusion is the validity of the data. The site I used, dragtimes.com, is not an official measurement of vehicle timings; it could be the case that values entered are incorrect, although that is highly unlikely considering this result supports previous work.

8 References

Drag Racing 1/4 Mile times. (n.d.). Retrieved from http://dragtimes.com/

Lucius, J. (2017, January 11). Calculators for 1/4-Mile ET & MPH vs. HP and Weight. Retrieved from http://www.stealth316.com/2-calc-hp-et-mph.htm

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.