Predicting The Price To Get Into #vanlife

Abstract

The goal of this project was to create the most accurate and well-fit regression model possible in order to predict the sale price of a campervan with a given set of features. I worked with data scraped from vanlifetrader.com, leveraging many techniques to clean the scraped data and engineer categorical, binary, and numeric features as necessary. I achieved moderate success with my model, finding it to be well fit and capable of predicting prices to a reasonable, if improvable, accuracy.

Design

This project originates from a pragmatic need of mine to establish a notion of the true market value of my own campervan as I move forward with the process of selling it. The data was acquired via a web scraping pipeline that I created to collect specific information on vans currently for sale in North America. A number of scraping and cleaning functions have been written by design to make this an expandable project as more data continues to become available in the future and market trends change. Predicting the sale price of a campervan accurately would allow me to (potentially) identify the best time to sell my own van, as well as what features I should consider adding, fixing, or altering in order to achieve the highest possible benefit for myself.

Data

My dataset currently contains 593 individual vans, collectively representing 56 original features and an additional 23 that I engineered. Of the original data, 40 features are binary, 7 are categorical, and 9 are numeric, including the target (sales price). The features available to me include everything from vehicular considerations like the mileage, make, model, model year, and drivetrain of the van to whether or not the van includes a toilet, an oven, an awning, and a wide array of other gadgets and comfort features. The binary features were grouped into more generalized categories ("Plumbing", "Gadgets", and "Creature Comfort") so that they could be scored and ordinated, while categorical features like selling location and drivetrain were transformed into dummy variables. Though this data certainly warrants further exploration and engineering, at current, 11 features have emerged as significant predictors of the target and have been applied to the created model.

Algorithms

Web Scraping

1. Utilization of two selenium drivers; one to access an entire page of listings and the second to crawl through each listing on that page. Inclusion of a 30-second pause when the listings are loaded, and 10 seconds between the scraping of each individual post.
2. Data scraped from three specific areas of a post are appended to a dictionary (k:v = feature:detail) which is then pre-processed to clean strings and convert string numbers to integers or floats as necessary.
3. When a whole page is scraped, the dictionary is appended to a list, and a brief pause of a random length is undertaken before loading a new page of listings
4. When all posts on the site are scraped, the list is converted to a Pandas dataframe.

Data Cleaning & Feature Engineering

1. Columns with significant amounts of missing data were located and the appropriate value (mean, median, or mode) imputed to fill NaNs.
2. *Conditional: if vanlifetrader is scraped on multiple different days, the expanding dataframes can have duplicate rows dropped where the Price, Location, and Make/Model are all duplicates*
3. Data was split into boolean and non-boolean features
4. Boolean features had their NaN values filled in with 'False', converted to numeric binary, and grouped according to their functional class within a campervan
5. Non-boolean features were further engineered by creating dummy variables for categoricals
6. Split data was recombined
7. Further EDA to find one-off errors in various columns, and appropriate cleaning methods applied (converting negative values, dealing with case-specific NaNs, parsing strings)
8. A great deal of feature engineering via binning, dummy variable creation, and ordination

Models

- Single-feature linear regression analyses between numeric features for sanity checking
- Data split 80/20 train/test
- statsmodels OLS summaries leveraged for preliminary feature selection
- Lasso and Ridge visualizations created to graphically explore significant features
- Preliminary final model generated via lasso regularization of features, which then underwent further modification via p-value elimination

Model Evaluation

- Once the final feature set was selected, the dataset was again split on an 80/20 train/test basis
- R^2 scores were computed for both the training and test sets to determine the R^2-delta:
    - R^2 for the training data is 0.745
    - R^2 for the test data is 0.715
    - The difference is 0.03
- MAE and RMSE values calculated for the final model, with RMSE being normalized
    - The Mean Absolute Error is 16962.4
    - The Root Mean Squared Error is 21896.4
    - The normalized RMSE is 0.123
- VIF scores were calculated for each feature in the final model, with outputs showing VIFs falling firmly between 1.00 and 1.69.

Tools

- Selenium and BeautifulSoup for web scraping and data ingestion
- Numpy Pandas, Scipy, and Math for data cleaning and manipulation
- Scikit-learn and statsmodels for modeling
- Matplotlib and Seaborn for visualization

Communication

In addition to the slides and visuals presented live, Vanlife Regression will be available on my personal GitHub, as will all future updates to this project.