Car Sale Data Analysis & Price Modeling Report

Zachary Burpee (zcb2110), Ming Liu (ml4802), Napasorn Phongphaew (np2839) - **202212-20**

Abstract

Our project provides valuable purchasing and selling information to customers who utilize online car selling platforms such as Craigslist.com [1] and AutoTrader.com [2]. The team evaluated and analyzed historical and current car sales data from those websites to predict future trends in car prices based on make, model, year of posting, and more. Additionally, we provide customers with more context of the larger car market to give them more tools in their decision making when buying or selling cars through our website, with the help of machine learning models. We were able to approach predictive power similar to that of Kelley Blue Book, the leading company in the industry of used car price prediction. In fact, in some commonly specified cars, we were able to predict the price of live listings more accurately than Kelly Blue Book - with an estimated difference of only around $400-$1,200 from the real listing price.

Introduction

Car price prediction is a longstanding existing problem tackled by companies like Kelley Blue Book (KBB) [3]. When purchasing or selling their used cars, consumers often judge listings based on similar offerings in the same time period. However this is a time consuming task to undertake for individuals, and usually many can't account for externalities that may affect a reasonable price of a car. Our goal is to provide customers with a resource displaying high quality information based on a large number of current listings and previously posted listings. To further this goal, we trained a MLP neural network in Tensorflow using a combined dataset from Craigslist (posted on Kaggle) and real-time scraped data from AutoTrader (a new/used car website that hosts listings both from third-party and certified dealerships). This data is augmented with census data to account for local price fluctuations based on affordability in the targeted region.

The scope of this problem pertains to a good majority of Americans, with an estimated 8M-10M new cars produced each year, and an average of 15M cars bought and sold each year in the United States [4]. If we are able to characterize a trend that takes into account the overall market share of cars bought and sold since the year 2000 (the approximate inception of the internet), we can provide a comprehensive buying and selling guide to allow consumers to make the most out of their investment–for something as simple as an everyday transportation vehicle. The guidance from our website will not only save customers money, but also save the customer's time, and could even provide a recommendation of what make/model to buy next based on the integrity of the trend in the recent years, or whether to hold on and wait for better situations to arise. We continuously keep the model updated with new parameters and freshly scraped data to help it adapt to market conditions (even in dire times such as the 2008 and 2020 market crashes), so we

know that the integrity of our model will withstand the test of time.

A high-level overview of the final architecture of our problem can be described as: data obtained and trained from Craigslist and Autotrader, which is sent for processing and then trained on a couple different models (MLP and decision trees), to determine the best classifiers of the data. Once the models are finalized with training data, we created a client-facing web application that will allow a consumer to look at overall market trends for an input vehicle, provide a comprehensive prediction of a car's estimated value at any given time, and display current listings off of Cars.com and AutoTrader.com; making our website an all-inclusive package for our consumers to buy and sell cars with confidence.

Our predicted results are qualitatively close to those of Kelly Blue Book, which give us a good indication that our models are predicting quite accurately - given that Kelly Blue Book is a widely utilized and industry-trusted resource for pricing used cars.

Related Work

The most prominent modern day implementation of car resale price analysis is Kelley Blue Book, a research company that publishes a website detailing fair pricing for cars, tips for evaluating price, and more insight of the car market. The company uses retail sale prices, real-world resale pricing, industry conditions, economic trends and location to help determine the price of a used vehicle. The company primarily sources its data from dealerships, auctions, and other forms of private sales to inform their predictive models. The benefits of this holistic approach is that it takes into account data from many sales forums, solving the problem of trying to sell a car for the maximum price possible (for the seller), and solving the issue of overpaying for a vehicle (for the buyer). Additionally, Kelley Blue Book can also look into market trends through its analysis of the entire market and its deep connections with nationwide dealerships and manufacturers.

Our approach is slightly different to KBB in that we are only directly analyzing peer-to-peer sales on alternative marketplaces (Craigslist and Autotrader) as compared to traditional used-car dealers, since we want to tailor our models to those specific use cases. We assume that traditional retail sales will have less of an effect on the pricing via these peer-to-peer websites. We're also interested in extracting time-based data (historical trends for car prices), instead of mainly just current fair pricing for sales. In terms of the final customer-facing portion of our project, we're taking a page out of Kelley's Blue Book, and creating a GUI where a customer can enter their make, model, year, and more information on their car, and receive an analysis of its pricing. We want the website to be easy to use, similar to KBB, and to abstract away complexity in the backend.

The most basic approach to this problem is using the linear regression model to predict the selling prices based on dependent variables such as brand, model, year, etc. However, the drawbacks of this approach are the sensitivity to the outliers when it

comes to complex data and its vulnerability to underfitting or overfitting. Apart from the data from the listing, our project takes the average income into account which varies throughout the states and also reflects the economy of the U.S. over the years. This introduces additional dimensions to the dataset. Another interesting model we experimented with was the decision tree. This model, which is a supervised nonlinear algorithm, proved that it can tackle the outliers and is viable at extracting complex, non-linear relationships - according to “Cars Selling Price Prediction using Random Forest Machine Learning Algorithm" [5]. The resource mentioned aimed to predict the selling price of the used cars based on the dataset provided on Kaggle. The algorithm in general results in little to no overfitting, takes less training time, and has high accuracy even with an extensive amount of missing data.

Another resource we used was a paper named “A Case Study on Car Evaluation and Prediction: Comparative Analysis using Data Mining Models” - which presented the decision tree model as one of the various data classification models to evaluate the price of the cars. They claimed that decision trees are the most suitable for car dataset with 91.1% accuracy [6]. As a supervised learning algorithm, the decision tree is a tree-like graph that consists of a root node representing the entire dataset, branches representing the decision rules, internal nodes representing the features of the dataset and leaf nodes representing the outcomes. It has the advantage over other models as it is easier to interpret and be able to handle categorical data as shown in our cars dataset. Its objective is to predict the class of the data based on several input attributes. We can see that the algorithm of this decision tree is similar to human’s decision making process when it comes to a decision on buying or selling a car. It should be comprehensive enough and be beneficial to our training models.

With all of these sources in mind, we decided to approach this problem with both Decision Trees and Artificial Neural Networks by using Multi-layer Perceptron (MLP), a feedforward step function consisting of fully connected layers. We chose to classify the data with a MLP as well because of the size and the complexity of the dataset and to create a bounding of the price between two different, but highly efficient models. While we expected a more accurate result from MLP since the algorithm traditionally has the learning ability to distinguish data that is not linear, we were surprised to see that the decision tree was more accurate by almost 10% for all models tested.

Data

Data Collection - Previous Datasets

As a baseline for our model, we sought out a highly dense dataset from Kaggle which included 426K observations and over 16 dimensions that sourced cars from Craigslist. Additionally, we sourced from an online AutoTrader database with previous listings that included 80K observations and 5 dimensions. Also, to provide a level of discretion between each state (since income varies greatly between states), we sourced

the average income over the years for each state to qualify our results. The income dataset had 50 observations over 20 years to total 1000 observations with 2 dimensions.

Data Collection - Live Scraped Data

To compare our model with live listings from AutoTrader.com and Cars.com - and to provide a liquidity to update our model as time progresses - we built a custom web scraper to get the first 100 listings that pertain to the search query of the customer. These results included 100 observations and 5 dimensions for each set, for a total of 200 observations and 5 dimensions per query. As our customers utilize our website, the data returned is appended to the model. We made the system passive to not abuse DOS pings to AutoTrader.com and Cars.com in case of commercialization issues.

Data Pre-Processing

Each previously sourced dataset was collected in the form of a CSV that was delimited by commas for ease of processing. Each scraped set of listings is in the form of a python dictionary for ease of object consistency.

We pre-processed the data by cleaning any empty cells from the columns, which would be from incomplete listings online. We then reformatted the resulting dense data into different tables for feature distinction and dictionaries for quick access of an internal database. Finally, we combined into a master dataset of car sales and listings of past and present. Additionally, the master dataset includes state information with the average income for each state.

During the cleaning of our Kaggle dataset, we eliminated NANs, blanks, empty strings, duplicate listings, and invalid pricings (prices that were above $100,000 and above $1) - leaving us with a dense dataset with 85K observations. We deemed this size was acceptable for our purposes. For the cleaning of the AutoTrader dataset, the data did not have to be cleaned at all since all the columns were filled.

A unique attribute of our dataset included string values such as paint color, fuel type, drive type, and sizing; which we chose to one-hot encode relevant features to train our model with. We originally one-hot encoded each feature, which did not yield very accurate results, so we applied exploratory data analysis (EDA) to our data to find the most relevant features. We started out with the natively numeric values, which we applied to a correlation matrix shown in Figure 1 below.

	price	year	odometer	income_this_year
price	1.00	-0.00	0.00	-0.00
year	-0.00	1.00	-0.08	-0.00
odometer	0.00	-0.08	1.00	-0.02
income_this_year	-0.00	-0.00	-0.02	1.00

Figure 1. Correlation Matrix of Numeric Values in Kaggle Dataset

As we can see, the natively relevant values were all completely uncorrelated - which means they were excellent classifiers for our data.

Next, we moved onto our string variables, which we one-hot each feature and applied a lasso and ridge regression model in R to get feature complexity and R^2 values. We decided to use R since R has intuitive data cleaning and processing capabilities. The

variables that we kept include fuel type (gasoline, electric, etc), make (Toyota, Ford), model (Corolla, Camry, etc), cylinders, and more commonly used pieces of data. We also kept the state that each row was sold from, and attached the average state income to each listing. This was to account for regional differences in car pricing, as cars may be more expensive in an expensive state. Then, we took a look at which variables were categorical (Brand, Fuel type etc), and which were numerical (Odometer, etc). We then converted each categorical variable into a one-hot representation, which greatly increased the number of columns in our dataset. This was another motivation to reduce the number of unique categories per variable, to reduce the width of the training dataset and improve generalized fitting. Once one-hotted, we ended with about 70 columns for our dataset, and this was used for our one-hotted dataset. Overall, we were able to trim the number of features to 17 from 26. With the rectified features selected, we decided to perform a skew analysis of our results to see what features might be skewed for our training. A few of the figures which show skew values of selected features are shown below in Figures 2-3.

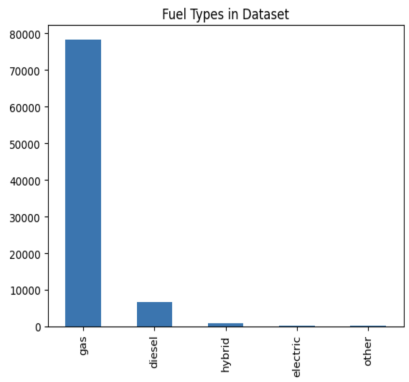

Figure 2. Top Fuel Types in Dataset.

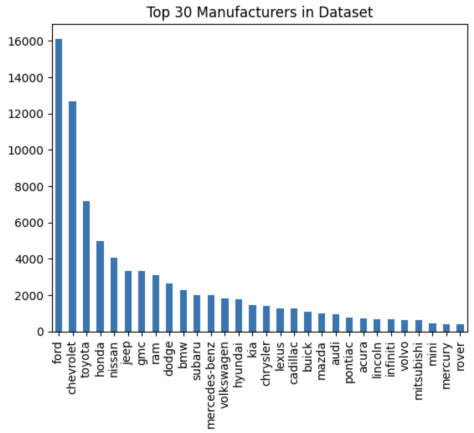

Figure 3. Top Makes in Dataset

As we can see, there is a significant skew for Ford as a gas car. We kept this in mind when we trained our set since queries outside the natural skew will likely have a higher error. In light of our EDA, we applied random sampling and batch processing to train our models.

Methods

Modeling the Data

We first fed our processed data into simple linear regression models to get an overview of the distribution of the data and determine whether there was a linear relationship between the price of a used car and the specific attributes we chose. A depiction of the linear regression of the price over time is shown below in Figure 4.

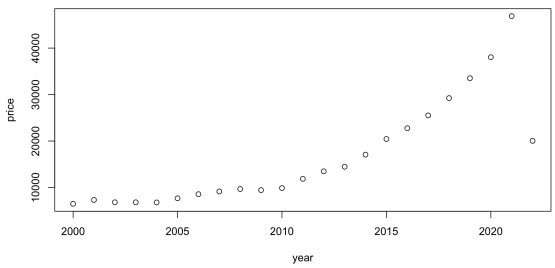

Figure 4. Regression of Price vs. Year

The linear regression models proved to be accurate for predicting very long periods of time, but did not give an accurate enough gradient down to a more fine time period. We noticed that the results of the simple linear regression were limited to a year-over-year schedule, so we needed to find a new model that could provide more integrity on a month-to-month or week-by-week basis.

After our analysis of linear regression, we decided to implement a more advanced Deep Learning model called a Multi-layer Perceptron (mentioned above) that could characterize more features and process bigger models with more accuracy than simple linear regression. We determined that the results were more accurate than the linear regression, however we are still training the model for accurate parameters since the MLP is much more complex to tune. We decided to once again try another type of model, and eventually landed on Decision Trees (DT), due to their previous success in papers we've seen before.

After comparing the outputs in the DT with the MLP, we came to the conclusion that the DT gave more accurate predictions (to online searches of the query), and was less likely to give odd predictions when presented with less common categories than MLP. We did also notice that the neural network tended to high-ball the prediction, whereas our DT was usually more centered in its prediction. Thus, we decided to bound our predicted price and move forward with our two models into our visualization and presentation solution. A sample image of the predicted price over time from the DT is shown below in Figure 5.

Figure 5. Decision Tree Price vs. Model Year of a Toyota Corolla

Overall, our MLP model was able to predict the price of a car with $1,200 on approximately a $12,000 car. This gives us an error of about 10%. However, for our DT model, we were able to price within $400 on a $12,000 car. This accuracy is much closer to 4%. As a benchmark, Kelly Blue Book gave the same price range of $1,000 on a $12,000 car, so both our models qualify with industry standards for the most commonly skewed data. We classified these training accuracies as a success since we matched and mostly improved the price prediction compared to our primary competitor.

For context on why we chose a $12,000 car was for two reasons. We chose cars that were positively skewed from the EDA performed above, and we took a look at the average price of cars sold in each state so we get a broad range of results. A depiction of the number of sales, with the average price, versus each state is shown below in Figure 6.

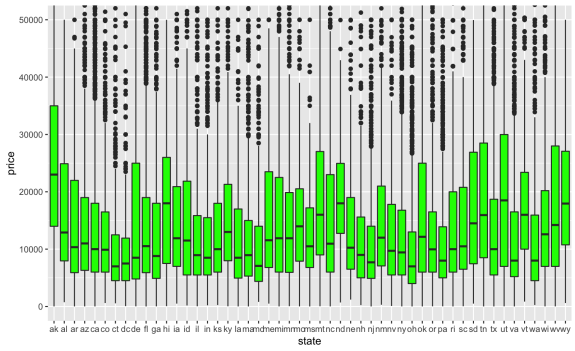

Figure 6. Average Price with Number of Sales vs. Each State

As we can see from the figure above, an estimation of $12,000 for the average price of a car is a good benchmark for the average consumer in the U.S.

Visualization & Client-Facing System

As for our client interface (front-end), we wanted to present the data in an intuitive and highly informative fashion for our consumers. We wanted to mimic the familiarity of commonly used websites, such as AutoTrader.com, Cars.com, and KBB.com, to have our input search query be intuitive to use. The landing page of our website queries them for some information about a prospective car's information. Our client-facing system is shown below in Figure 7.

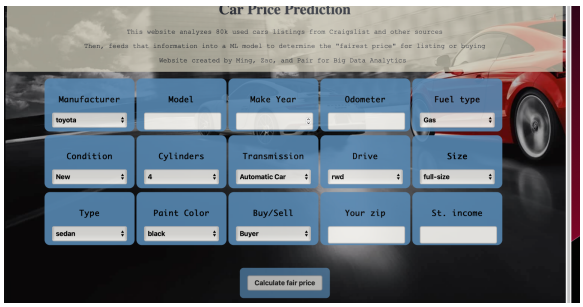

Figure 7. Our Client Facing Search Query

We wanted to make our website aesthetically pleasing, while also giving it a sense of professionalism. We implemented simple HTML backgrounds, input search queries, and a clear overall description of the website purpose.

Our framework for the construction of this website implemented HTML, Javascript, and D3 framework - all while hosted on a python Flask server. We wrote an HTML script that would take a customer query of a car consisting of: manufacturer, model, year, odometer, fuel type, condition, cylinders, transmission, drive, size, type, paint color, buyer or seller, zip code, and state income. To help improve the user experience, we provided comprehensive drop-down menus to minimize typing by the user. Some fields are numeric, so we allowed those to be text-enterable.

Once the user fills in all the fields and clicks the submit button, it will redirect to another HTML page showing a web scraped picture of the car (for easy identification) along with the predicted price of its value currently - from both the MLP and decision tree models. Also, we depicted an expected price of the different model years of the same car to give an estimation of what the price will be in the future, what it was in the past, what the overall trend might be, and whether a buyer/seller might want to wait to buy or sell their queried car. As a benchmark, our competitor Kelly Blue Book only provides a range of the predicted price currently. We also provide this information - to a possibly better accuracy - and provide a comprehensive price prediction of that car in the future.

In addition to the predicted price and the image of the car, we believe our most useful feature is the table showing live listings results on other car valuation websites - including AutoTrader.com and Cars.com. With this live implementation, customers can redirect themselves to the scraped URLs from those websites to directly start browsing for their car, determine the current pricing to sell their car, and compare the listing prices from those two websites.

With the inclusion of all of these features on our client-facing website, we have provided an all-inclusive interface for customers to identify, price, buy, sell, and predict over time any car ever produced!

System Overview

A holistic look at our system is shown below in Figure 8.

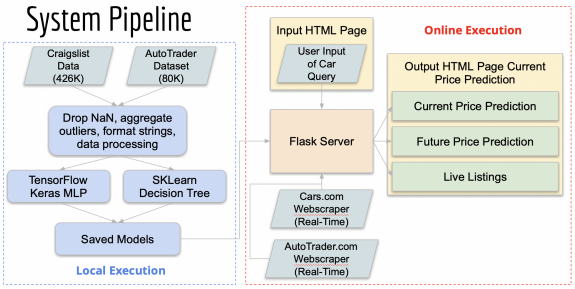

Figure 8. Overall System Pipeline

Starting at the top left of our implemented system, we gathered existing dataset from Craigslist and AutoTrader, then created a custom web scraper of Cars.com and Autotrader.com to get real-time datasets. We utilized Python as the language for the web scraping software since our team was most familiar with its capabilities and libraries available for this task. More specifically, we implemented the BeautifulSoup (bs4) library for the web scraping since it extracts the XML elements from any website in an intuitive fashion, where we only have to identify the XPATHs of our target values. An XPATH is essentially a key to a map that contains values such as JSON data or source image data to extract. XPATHs are not exclusively tailored to each element independently, so we were able to identify the JSON data to each car price and hyperlink with a few lines. Additionally, we web scraped the first relevant image of any car query with the same method of XPATH identification.

After data was extracted from all sources, we performed exploratory data analysis (EDA) to determine distribution and correlation to give us more insight of the available data. As a summary from the above section, we dropped all NaNs values, eliminated outliers, one-hotted relevant categorical variables and reclassified our data. Following the data cleaning, we did preliminary R regression models (which is described in more detail in the methods section above).

After our models with linear regression, we then fit our data to ML models where we were utilizing the TensorFlow with Keras and Dense libraries. These libraries specialize in the deep learning portion of our classification since Dense Layers are primarily used for MLP building of models. Each of these layers is formally known as a fully-connected layer. We used a total of 10 fully-connected Dense layers - totaling approximately 3.1 million parameters to

train. A summary of each layer and parameter is shown below in Figure 9.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 64)]              0

 layer_normalization (LayerN  (None, 64)               128
 ormalization)

 dense (Dense)               (None, 1024)              66560

 dense_1 (Dense)             (None, 1024)              1049600

 dropout (Dropout)           (None, 1024)              0

 dense_2 (Dense)             (None, 1024)              1049600

 dropout_1 (Dropout)         (None, 1024)              0

 dense_3 (Dense)             (None, 512)               524800
...
Total params: 3,128,449
Trainable params: 3,128,449
Non-trainable params: 0
```

Figure 9. MLP Layers Detail

For the Decision Tree classification, we opted to implement the SKLearn library for this task since we are most familiar with this library from class homework. We would have liked to show the classification of the models here as a visual image of the different classes, however the dimensionality being 72 different columns with limitless classes was not feasible.

We used a local execution of our models because we already had the libraries installed on our computers. However, after we got our models from the local execution, we hosted our model and HTML pages with the Flask server. We utilized the Flask framework (referenced from online resources) to host a simple website that takes the input from a user query and sends that information to the python file running the model. After the model outputs its results, the page is redirected to another HTML page that utilizes D3 and JavaScript to display the current price prediction, future price prediction, and live listings. We opted to use D3, JavaScript, and HTML since we learned these frameworks and libraries extensively during class.

Experiments

Refined Model Training

To refine our model training, we played around with the number of categorical variables we used in our data in order to refine the MLP and Decision tree model, in an attempt to achieve an accuracy that can be closely compared to the current prices that Kelley Blue Book has posted. Originally, our goal was to sample numerous car queries and compare the predicted values of our model with Kelley Blue Book, and consider success if the samples reach a certain reasonable tolerance. If we are able to gain a correct model of the prices over time, this signals our project is a success. Over the course of several iterations of what data we would feed into our models, we found that the original dataset had far too many categories. Some car brands only had two or three entries in the entire dataset, but would take up an entire column when one-hotted. This would skew the results to be highly unpredictable. Thus, we took a look at the most common categories, kept those, and put all the other special categories into “other”, so as to not have to delete rows that elsewise had usable data.

One thing that we adjusted and tweaked during training was the pricing range of the rows we used in training. We figured that some listings for “supercars” wouldn't be applicable to regular users, and thus we

restricted the upper bounds of the rows allowed in the dataset to be below a certain amount, which later helped our model with more accurate predictions for common car models but slightly hurt prediction accuracy for speciality cars or car models with fewer entries.

Conclusion

We compared our results with the real market listing prices and came to the conclusion that our predicted prices are within 12% of accuracy on average.

Due to our model taking car features as input queries and predicting the price based on users' selection, our model could be generalized and be applicable to other types of vehicles such as motorcycles as well. It would be an interesting area as we are still short on motorcycle evaluation websites. For future areas of research, we would like to try fitting various machine learning models to compare their performances with our dataset while minimizing the loss and optimizing the accuracy for the fairest price in the market. Apart from optimization, we still need solid evaluation metrics to strengthen the result of our findings. For further application, with an improvement on the front-end user interface, our website has a potential to become another option for a car evaluation site comparable to Kelly Blue Book, Autotrader.com, and Cars.com.

References

[1] "New York Jobs, Apartments, for Sale, Services, Community, and Events." *Craigslist*, https://newyork.craigslist.org/.

[2] "Used Cars for Sale." *Autotrader*, https://www.autotrader.com/cars-for-sale.

[3] *Kelley Blue Book | New and Used Car Price Values, Expert Car Reviews*. https://www.kbb.com/.

[4] M. Carlier, "U.S. vehicle sales 1976-2021," *Statista*, 14-Jul-2022. [Online]. Available: https://www.statista.com/statistics/199983/us-vehicle-sales-since-1951/. [Accessed: 01-Dec-2022].

[5] A. Pandey, V. Rastogi, and S. Singh, "Car's selling price prediction using random forest machine learning algorithm," *SSRN*, 01-Oct-2020. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3702236. [Accessed: 01-Dec-2022].

[6] Jain, Pravarti, and Santosh Kr. "A Case Study on Car Evaluation and Prediction: Comparative Analysis Using Data Mining Models." *International Journal of Computer Applications*, vol. 172, no. 9, 2017, pp. 21–25., https://doi.org/10.5120/ijca2017915205

[7] Burpee, Zachary "Big Data Analytics - Car Sale Data Analysis & Price Modeling Final Presentation Recording" *YouTube.com*. https://youtu.be/LPWYgyndh_8 [Accessed: 23-Dec-2022].

Name	Contribution
Ming Liu	40%
Zachary Burpee	40%
Napasorn Phongphaew	20%

Our YouTube Link to our Live Presentation is linked here and slides here.

Appendix

Sample comparison of our model output with KBB.com and depiction of live listing of AutoTrader.com and Cars.com

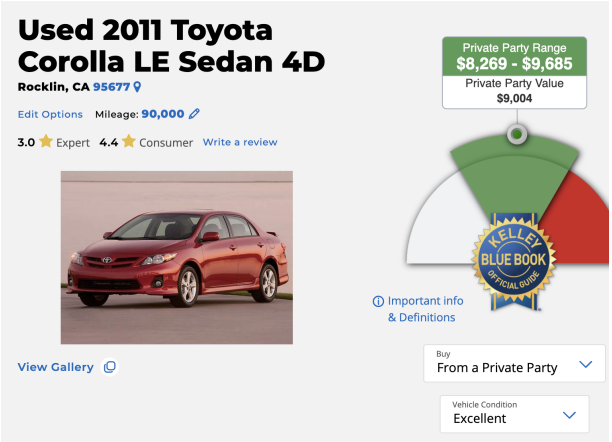

Figure 10. KBB Evaluation for Input Car

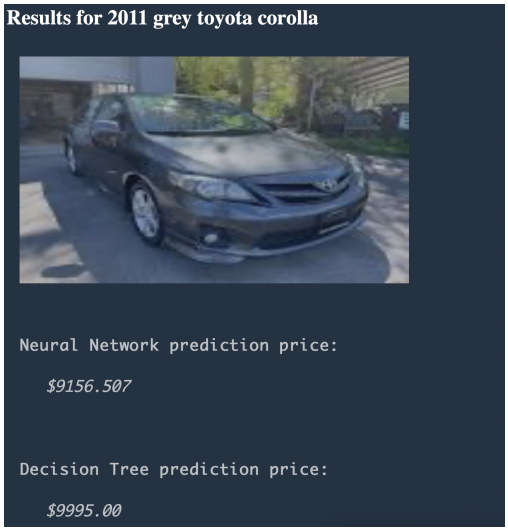

Figure 11. Our Result w/ Prices

Figure 12. Price Prediction for Model Year

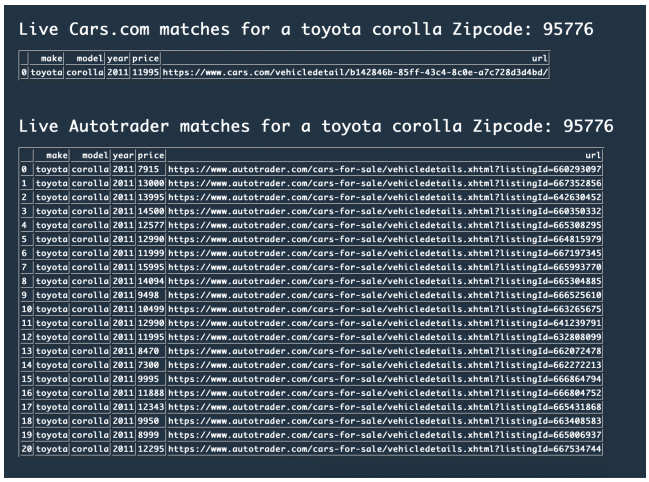

Figure 13. Live Listings on AutoTrader.com & Cars.com

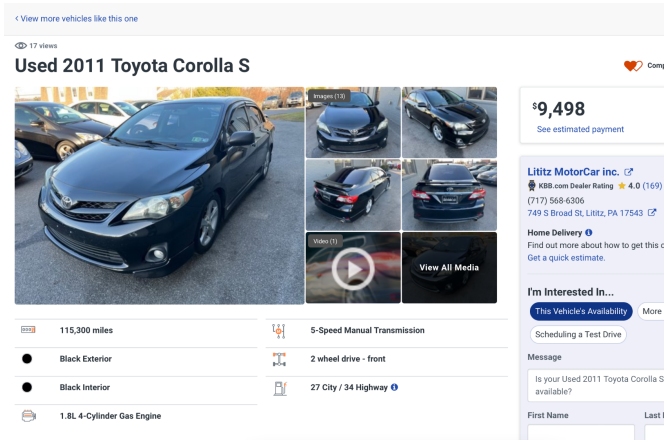

Figure 14. Live Comparison Listing scraped from AutoTrader.com

As we can see from our model outputs, the KBB price prediction, and the live listing from AutoTrader.com, we have successfully classified the price of cars simply based on their attribute features.