Managing Used Car Auction Data for ML Models

By Chih-Hsien Lin

B.S. Chemical Engineering

M.S. Applied Data Science

December 2, 2021

Feel free to contact me via:

chihhsil@usc.edu

Link to Project Files:

https://drive.google.com/drive/folders/1R6-IE7DV2bE9bSSIs7yS5aT-P5s7Ru-o?usp=sharing

Link to Demo Video:

https://usc.zoom.us/rec/play/1ZqfRillYjov9OSw1TQIHUrrg21aE6EoBS8EVXiM2TPeNF-wH274c9AtztRHFjQpEQwEfJJqq-Np81Qx.bMWYnvGwd0TXfZ_F?continueMode=true

Background

Topic Background

Driving feels like a necessity in the United States. While anyone would love to sport a brand-new vehicle from the dealer, the reality is that most people, especially young adults, will have to rely on second-hand purchases or even hand-me-downs. Used cars are indeed very popular in the United States, so much so that the used car market is significantly larger than new car markets. According to Vox, "[a]bout 40 million used cars were sold in 2019, compared to about 17 million new vehicles" (Nguyen). With so many different car manufacturers and models, how would anyone know whether he or she is buying or selling a good deal?

One solution is to look at the Manheim Market Report (MMR), a widely trusted indicator of wholesale car value. The wholesale value of a car is the price a dealer pays a manufacturer to buy cars from the manufacturer (CarsDirect). The issue, however, is that the MMR is not indicative of what regular consumers will pay a dealer for a car since dealers will charge a premium in order to net some profit. My project aims to solve this issue by harnessing the power of machine learning to predict used car selling prices based on past car auction data so that users of my program will be able to find fair prices to buy and or sell cars at.

My Background

I have a Bachelors in Chemical Engineering, which may seem irrelevant to the field of data science. However, being able to process data is a core skill of any chemical engineer, albeit not at the scale of a data scientist. Although I have little programming experience when compared to computer science majors, I actually started programming with MATLAB during my freshman year. I soon learned JavaScript for front-end web development and a little bit of Python for data science on my own time. I did learn the basics of machine learning, specifically the KNN algorithm, but have never actually tried to implement it on my own. So for this project, as an Applied Data Science major, I aimed to do just that and more.

Description of Dataset

My project is based off of the “Used Car Auction Prices” dataset from Kaggle (Tunguz). The dataset includes sales transactions of over 500,000 cars with 16 columns of information about each car’s manufacture year, make, model, trim, condition, MMR, selling price, sale date, and etc. It ranges from cars in 1982 to cars in 2015. By extracting certain features from this dataset and feeding it into KNN and multiple regression algorithms, I was able to create tools to help users find similar cars and predict car prices of their choosing.

Architecture

UI: Jupyter Notebook

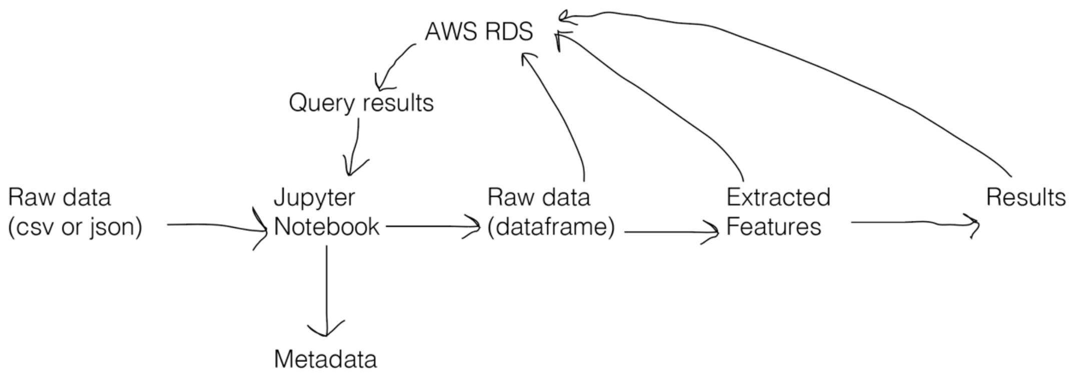
Cloud database: AWS RDS

Back-end: Python

Libraries used: see below

Library	**Purpose**	**Files that use it**
ipywidgets	Create interactable UI	Jupyter Notebook
numpy	Use numpy arrays and functions	Almost all files
pandas	Use dataframes	Almost all files
seaborn	Data visualization	plot.py and Jupyter Notebook
qgrid	Create interactable dataset explorer	Jupyter Notebook
matplotlib	Data visualization	Jupyter Notebook
scipy	Statistics calculations and compressed sparse row matrix creation	transform.py, knn.py, regression.py
sklearn	KNN and multiple regression implementation	knn.py, regression.py
pymysql	Connect to AWS RDS to load and query tables	load.py, query.py, regression.py, getTables.py, clearDatabase.py
sqlalchemy	Connect to AWS RDS to load and query tables	load.py, query.py, regression.py, getTables.py, clearDatabase.py

Pipeline



The general flow of data goes like this. The client first loads a “csv” or “json” file in Jupyter Notebook, which reads it into a Pandas dataframe using its Python backend. The dataframe is immediately loaded onto the AWS RDS database. The Jupyter Notebook then accesses the AWS RDS database for the raw data to perform feature extraction, transformation, and cleaning. The resulting dataframe is then stored onto the AWS RDS database under a different table. Finally, the Jupyter Notebook accesses the database for the extracted features to perform KNN and or multiple regression algorithms and stores those results under their respective tables in the database.

If all features of the Jupyter Notebook are used, there should be 4 tables in the database, namely “raw data”, “features”, “recommendations”, and “predictions”. These can be accessed at any time throughout using the program via the data exploration UI part of the Jupyter Notebook (details provided later). The data exploration UI will output an interactive KDE plot along with interactive tables to explore the relevant data as well as its metadata.

Files and Components

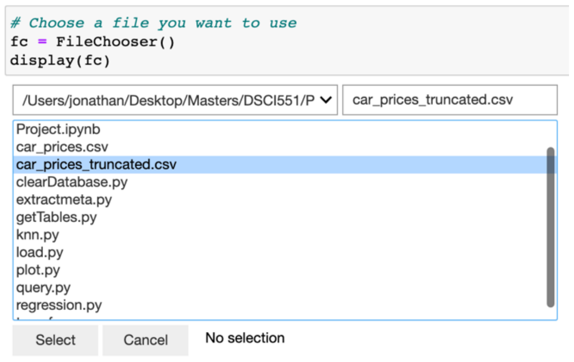
File	Purpose
Project.ipynb	This is the UI of my project, which allows users to upload their own datasets, extract features from their datasets, and perform KNN and multiple regression on the features to recommend similar cars or predict price of a car. At each step, each output dataset is loaded into the AWS RDS database so that users can explore these datasets at anytime.
load.py	Introduces loadData(data, table) function. Loads a dataframe to a specified table in the AWS RDS database.
getTables.py	Introduces getTables() function. Runs "SHOW TABLES" query on AWS RDS database to return a list of tables available in the database.
query.py	Introduces queryData(table) function. Runs "SHOW * FROM [table]" query on AWS RDS to return all data from a specified table in the database.
extractmeta.py	Introduces extractMeta(dataframe) function. Extracts metadata from a dataframe — specifically, number of rows and columns, data types of each column, existing and missing values of each column, and memory size of dataframe.
plot.py	Introduces plotData(data) function. Plots a KDE plot (density of selling price) of a dataframe. Also draws the average line as well as shades in the standard deviation.
transform.py	Introduces transformData(data, type, outlier) function. First, extracts features from raw data. Then, depending on "type", may remove rows with missing data or replace missing data with the mode of each column since there are quite a few categorical variables. Next, standardizes common representations of certain words. For instance, "Mercedes-B" becomes "Mercedes-Benz".

	Lastly, depending on “outlier”, may or may not remove rows with outliers.
knn.py	Introduces recommend(features) function. Runs KNN algorithm on features dataset to return the N-most similar cars. Specifics on the KNN algorithm will be discussed later.
regression.py	Introduces predict(features) function. Runs a KNN algorithm to increase precision of sample before running a multiple regression algorithm to predict price of a user-specified car. Specifics of the multiple regression algorithm will be discussed later.
car_prices.csv	Original dataset from Kaggle. Includes over 500,000 rows and 16 columns. Due to its size, it is recommended to use the truncated version of this dataset for demonstrations. Otherwise it will take too long to load the data into AWS RDS.
car_prices_truncated.csv	Truncated version of original dataset. It is simply the first 1000 rows of the original dataset. Much faster to load into AWS RDS.

UI and Functions

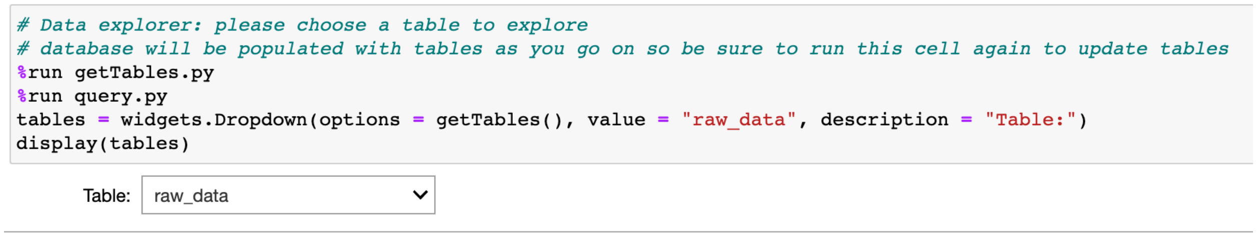
The following is a list of functions the UI has that users can take use intuitively (make sure to check out my demo to see how to use my notebook):

1. **File Chooser**: This allows users to choose their files. In the backend, this stores the absolute filepath into a variable that is used later on to read the file into a dataframe.



```
# Choose a file you want to use
fc = FileChooser()
display(fc)
```

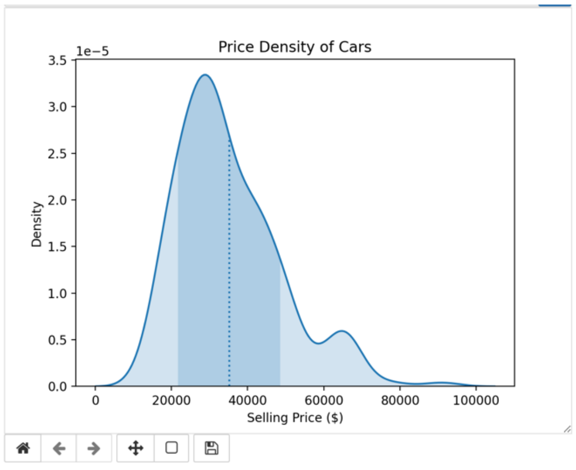
2. **Table Selector** (for viewing in the next cell): By running getTables(), this cell retrieves the available tables in the database and lets users select which tables they want to view in the next cell via a dropdown box.

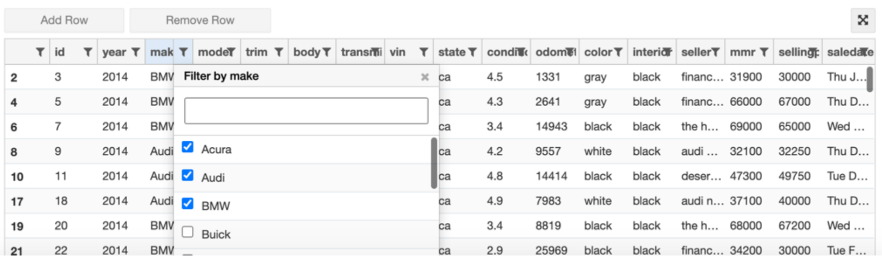


```
# Data explorer: please choose a table to explore
# database will be populated with tables as you go on so be sure to run this cell again to update tables
%run getTables.py
%run query.py
tables = widgets.Dropdown(options = getTables(), value = "raw_data", description = "Table:")
display(tables)
```

Table: raw_data

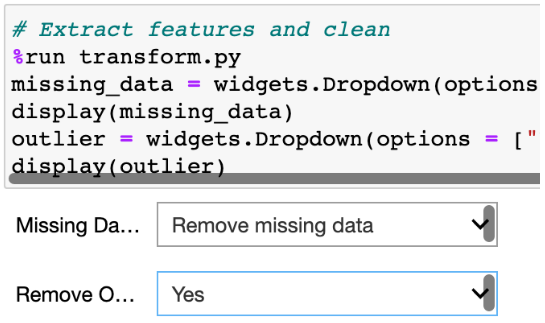
3. **Data Explorer**: The Data Explorer displays a KDE plot of the selling price (with mean and standard deviation) of whichever table is selected in the Table Selector. The interactive qgrid below allows users to filter through the table, which will update the KDE plot above based on the filter. Metadata is displayed under the qgrid, which the user can also explore (not shown in image below).





	id	year	mak	model	trim	body	transmi	vin	state	condi	odom	color	interior	seller	mmr	sellingp	saledate
2	3	2014	BMV						ca	4.5	1331	gray	black	financ…	31900	30000	Thu J…
4	5	2014	BMV						ca	4.3	2641	gray	black	financ…	66000	67000	Thu D…
6	7	2014	BMV						ca	3.4	14943	black	black	the h…	69000	65000	Wed …
8	9	2014	Audi						ca	4.2	9557	white	black	audi …	32100	32250	Thu D…
10	11	2014	Audi						ca	4.8	14414	black	black	deser…	47300	49750	Tue D…
17	18	2014	Audi						ca	4.9	7983	white	black	audi n…	37100	40000	Thu D…
19	20	2014	BMV						ca	3.4	8819	black	black	the h…	68000	67200	Wed …
21	22	2014	BMV						ca	2.9	25969	black	black	financ…	34200	30000	Tue F…

4. **Cleaning Options**: The program allows users to choose how they clean their data during feature extraction. Options include "remove missing data" or "replace missing data [with mode]" and "remove outliers" or "keep outliers".

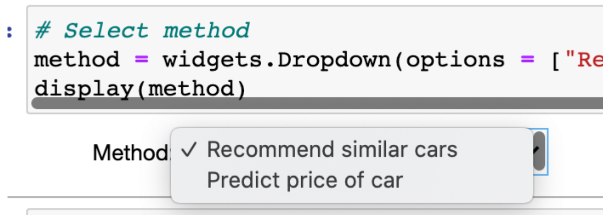


```
# Extract features and clean
%run transform.py
missing_data = widgets.Dropdown(options
display(missing_data)
outlier = widgets.Dropdown(options = ["
display(outlier)
```

Missing Da... Remove missing data

Remove O... Yes

5. **Method Selection**: This is the most important part of the project as it allows users to choose what type of algorithm they want to use. "Recommend similar cars" uses the KNN algorithm to identify the N-most similar cars based on user input. "Predict price of car" uses multiple regression to predict price of a user-specified car. Details of the algorithms will be explained next.



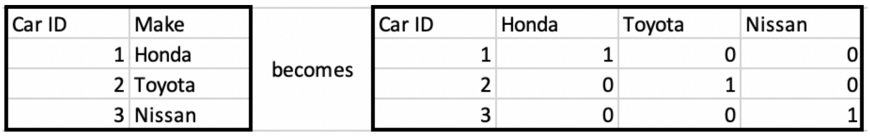
```
# Select method
method = widgets.Dropdown(options = ["Re
display(method)
```

Method ✓ Recommend similar cars
Predict price of car

Machine Learning Algorithms

Recommend Similar Cars

In order to recommend similar cars to the user, the KNN algorithm was used. Categorical variables from the extracted features had to be transformed into indicator variables in order to meaningfully calculate the dissimilarity between different categorical variables (see example below).



Car ID	Make
1	Honda
2	Toyota
3	Nissan

becomes

Car ID	Honda	Toyota	Nissan
1	1	0	0
2	0	1	0
3	0	0	1

The table becomes very sparse by creating indicator variables due to the large amounts of 0s in the table. For this reason, a compressed sparse row matrix had to be used for the KNN model to fit more efficiently. Before the KNN algorithm is used, various adjustments to the weightings of each attribute had to be made to ensure the recommendations were as relevant as possible. For instance, the model of the car had the highest weighting since there is nothing more similar to a car than a car that is the exact same model. Cosine similarity is used to calculate distances between each car.

Predict Price of Car

To predict price of a car, the KNN algorithm above was used to narrow down the dataset to a more precise sample. Afterwards, the multiple regression algorithm is used on the continuous variables of the resulting dataset. This allows users to predict the price of a user-defined car based on his or her provided dataset. The accuracy of this algorithm is not as high as hoped, but was a lot better than when it was first implemented due to many changes to the algorithm (see Learning Experiences section).

Learning Experiences

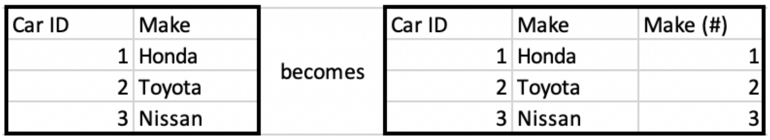
Lessons

The biggest lesson I learned from this project is to dedicate time to learning new programs, software, methods, etc. as well as to dedicate time to debug code. I initially thought certain portions of my project would be a breeze. However, things don't always go the way I intend them to go. Allocating time to these potential setbacks is very important when managing projects. Although I had enough time to complete my project, I wish I had allocated more time to these setbacks so that I could focus on improving other parts of my project or even learn how to implement a web-based UI.

Challenges

There are three main challenges that I faced throughout my project. The first one is loading a large dataset onto a cloud database. My initial approach with Firebase failed to work and or was extremely slow and inefficient. I had to learn how to use AWS RDS to store my data. Although my current method of loading large datasets onto AWS RDS is by no means fast, it at least works. For instance, the "car_prices.csv" file, which is 89.1 MB in size and contains 500,000 rows of data, can take up to 10 minutes to load onto AWS RDS (which is why I didn't use it for my demo although doing so will yield much higher accuracy on my multiple regression model).

The second challenge I faced was implementing a KNN model that made sense. My initial approach to KNN was to convert categorical variables to numbers like so:

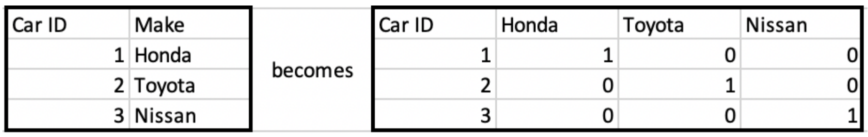


Car ID	Make
1	Honda
2	Toyota
3	Nissan

becomes

Car ID	Make	Make (#)
1	Honda	1
2	Toyota	2
3	Nissan	3

That, however, did not make any sense since it implied that Honda (1) is more similar to Toyota (2) than Nissan (3) when using cosine similarity. In reality, the similarity should be binary — it either is or isn't. For this reason, I had to find a different approach, which was to use indicator variables as explained before:



Car ID	Make
1	Honda
2	Toyota
3	Nissan

becomes

Car ID	Honda	Toyota	Nissan
1	1	0	0
2	0	1	0
3	0	0	1

By doing so, similarity now becomes binary when dealing with categorical variables.

The final challenge I faced was having to adjust weightings of variables throughout the project. For instance, the "odometer" values range from 0 to 999,999 whereas the indicator variables for "make" range from 0 to 1. Therefore, if the user searched for a "Honda with 10,000 as its odometer value", the KNN model would "disregard" the "make" and return any car that has "odometer" values close to 10,000. In other words, the "odometer" value dominated the KNN model. For this reason, I played around with the weightings of each attribute and I believe I found a good balance.

Experiences

This project allowed me to have hands-on experience in working with machine learning in a very holistic manner. It also required lots of creative thinking to on my part to create a UI that will provide a great user experience. On top of that, it also required lots of tech-related problem solving. However, the most valuable experience to me is the opportunity to see what others have done and potentially learn from their projects.

Skills Gained

Throughout this project, I gained many skills. First of all, I learned how to use ipywidgets to create an interactive UI. I also learned how to use AWS RDS and how to connect to AWS RDS via Python. By getting hands-on experience with KNN and multiple regression, I was also able to gain a deeper understanding of how these machine learning algorithms work. Perhaps the greatest skill gained is the ability to troubleshoot errors with code — in particular, knowing what to ask and to search for when I need help.

Works Cited

CarsDirect. "How to Get Wholesale Price Cars." *CarsDirect*, CarsDirect, 19 Mar. 2012, https://www.carsdirect.com/car-pricing/how-to-get-the-wholesale-price-on-cars#:~:text=The%20price%20of%20wholesale%20cars,can%20get%20to%20this%20price.

Nguyen, Terry. "Why Are Used Cars so Expensive Right Now?" *Vox*, Vox, 14 Oct. 2020, https://www.vox.com/the-goods/21507739/coronavirus-car-market-used-expensive.

Tunguz, Bojan. "Used Car Auction Prices." *Kaggle*, 18 May 2021, https://www.kaggle.com/tunguz/used-car-auction-prices.