Corey Kozlovski
12/18/19
CICS 397A
Final Project

Introduction: Car Shopping is a Draining Process

Purchasing a new car is one of the most impactful decisions an adult has to make at any stage of their lives. It is costly, generally requiring you to take out a loan either through a dealership or a bank, can potentially cost upwards of $30,000, and is something that would more than likely be kept until it is broken down. These factors put pressure on the buyer to get the right car, as they would not want to invest a large sum of money in a product they would not be happy with, as it is used nearly everyday and will likely last a long time. In deciding what car to purchase, it is apparent that there are dozens upon dozens of options in between all the brands, and even within the same brand, that offer different options, trim levels and styling that can often confuse the buyer.

Furthermore, when talking to other people about car recommendations, it is very apparent that people shop on cars based on marketing, brand loyalty, and word of mouth almost solely. Obviously, the car has to have the features they desire, but people seem to tend towards brands and models that are recommended through these options. I wanted to create a tool that expanded a person's shopping scope by suggesting cars that are similar in traits to what the person is already considering.

Getting the Dataset:

I took a look at Kaggle for inspiration on this project, and decided to combine one of my personal interests, cars, with this project. I found this dataset that outlined different Car Features alongside their MSRP, and decided I wanted to use it for this project.

In regards to preprocessing, pandas was able to handle the majority of it as it was a CSV download. Besides what read_csv() did, I personally decided to remove all N/A values as I saw that method to be the most fit, as it did not make sense to fill out N/A values with other data points, as I would essentially be creating new cars/trim levels that would not exist, which would be detrimental to the purpose of my project. Furthermore, in the dataset, I primarily focused on using Year, Horsepower, Number of Doors, and MSRP in my suggestions. I found these traits to encapsulate the majority of what someone would look for in a car while also being feasible in my knowledge of python/predictive analytics. There was some more preprocessing that I considered, but I will expand on that in the Facing Challenges portion of the write-up.

Task Writeup & Code Overview:

I created a program that was able to recommend cars based on either a given car model and a model year or given a year, horsepower, number of doors, and MSRP. To do this, I used NearestNeighbors through sklearn. NearestNeighbors is essentially the first half of k-Nearest Neighbors, which is one of the predictive analytics methods we learned in class. In KNN, we have a training and testing dataset where we essentially calculate the distance of any given data point to the predicted data point, and use the k-nearest neighbor's class value in a majority-vote style method in determining what the predicted data point should be. When it comes to NearestNeighbors by itself, it is different in the sense that it is unsupervised and is not trying to predict a specific class attribute. Here, it is trying to find which K number of neighbors are closest to the inputted values, and simply returning those values. I chose to use this method as I determined of all the methods we learned in class, this seems to be the best way of going about suggestions, as recommendations are generally best when they are close to a desired product, but not exactly that product. If this program works perfectly, the outputs would essentially be each brand's competitors for a given model.

Looking at the code itself, the most important method was suggest_model_given_attributes(). This method essentially took in the inputs of year, hp, doors, msrp, suggestion_count, and dataset and re-created a dataset that focused only on our four factors, made sure all the values were of the same datatype, scaled the values, and finally running NearestNeighbors() and returning an array of those neighbors' indexes. The other method, suggest_model_given_model(), essentially used the method above but added a portion that would "search" for the desired model and year and use those search results as the inputs for suggest_model_given_attributes(). Table_output() would then take advantage of the returned indexes, and return the entire "car" as a recommendation to the user.

Facing Challenges:

I faced a lot of challenges throughout the project that did change both the direction my project was going and how accurate my suggestions were intended to be. I think with more experience in both python/predictive analytics and with more time, as I've been in overdrive the last two weeks, would have let my project grow into something more in depth and interesting as a whole.

The first major problem I ran into was mixing the categorical data alongside the numerical data in creating the suggestions for the user. Originally, the factors I wanted to include were year, horsepower, transmission, driving wheel (AWD, FWD, RWD), number of doors, market category (exotic, luxury, compact), style (coupe, sedan, hatchback, etc) and MSRP, but I had trouble converting these categorical variables into a form that would be usable with NearestNeighbors. While some of the options I did remove on purpose, as when it comes to User Experience Design oftentimes having very specific and a lot of questions can be harmful when it comes to recommendations, there are some I wish I had the knowledge of implementing correctly. My solution to this problem was to look at the numeric columns that I thought encapsulated the majority of the options, which lead me to picking just year, horsepower, number of doors, and MSRP. If this project was in R, a language I am personally a lot more comfortable with, I think I would have been able to implement more categorical variables as data points.

Another problem I ran into was with the data set itself. The data set includes a lot of duplicates, as very minor things can change between different years of a car or different trim levels of a car. For example, the base model Honda Civic 2015 has 143 HP, while the manual version of the same car has 205 HP. Both of these results are apparent in the dataset, causing some of the suggestions to recommend the car you searched for already. A prime example of this happening is when you use “python car_suggest.py WRX 2015 7”, and the first suggestion is a 2015 WRX, the same car you searched for. Since there are duplicates, it based the search off the automatic version of the WRX, leaving the manual version open as a suggestion. This is something that seemingly would take a lot of time to fix, but would definitely add a level of polish to the program as a whole.

Conclusions & Future Exploration:

I really enjoyed this project as a whole. I think the idea I have is an interesting one and definitely want to expand on it in the future as a personal project. As a whole, I enjoyed the freedom we all had in deciding what we wanted to accomplish with this project, but I really do wish this was structured differently. Obviously this is the first time this class has been offered so there are definitely improvements to come, but I think this project would have been perfect as a month-wide project. I think a lot of the ideas I had, as mentioned in both the problems section above and the paragraph following this, would have added a level of polish and depth to this project that would have really taken this to the next level. Overall, I am happy with my project. It does create recommendations that seem to be fairly close to what I would consider related cars, but given that it only looks at 4 specific attributes, I am satisfied with the outputted results it gives. There’s room to improve, which I will likely do on my own time as I really did enjoy this, but unfortunately my knowledge and the timing of this project was unfavorable.

One thing I would love to incorporate in the future for this project would be the ability to input values from every single potential column from the dataset. I think it would make the tool more useful as a whole, in addition to having ‘Any’ as a choice for any given category would improve the system dramatically. Adding upon that, being able to leave some categories blank and having the suggestions be based off the filled out ones would also be a good feature to implement in the future.

Another portion I would love to implement in the future would be adjusting the weights of each factor, and how that contributes to the results as a whole. For example, the difference between 300 HP and 310 HP isn't very significant if you are looking to purchase a car, but the difference between a 2-door and 4-door is. This is something my project overlooked, as I wasn't sure how to go about that using sklearn, but that would definitely be a must have feature moving forward.

Using the Program:

There are two ways of using this program, both through the terminal.

1. Getting Suggestions based on Model and Year:
    a. In the command line, type the following:
       python car_suggest.py model year suggestion_count

       Where:
        - Model = car's model, not including maker. (Ex: Civic, WRX, 240SX, 3000GT)
        - Year = desired car year, YYYY format.
        - Suggestion_count = How many recommendations you are looking for.
2. Getting Suggestions based on Year, Horsepower, Number of Doors, and MSRP:
    a. In the command line, type the following:

python car_suggest.py year horsepower number_doors msrp suggestion_count

Where:
- Year = desired car year, YYYY format.
- Horsepower = desired amount of horsepower.
- Number_doors = Number of doors desired, 2-4 inclusive. Other are accepted, but the dataset only encapsulates 2-4.
- MSRP = Price budget desired.

Some examples for testing:
- python car_suggest.py WRX 2015 7
- python car_suggest.py Civic 2017 10
- python car_suggest.py GT-R 2017 15
- python car_suggest.py 2003 200 2 20000 5
- python car_suggest.py 2017 500 2 100000 15