Vehicle Price Gouging

**Due Date**: July 14th, 2022

**Contributors**: Daniel Czornyj, Robert Gonzalez, Robert Lane, Cody Sultenfuss, Krystal Lewis

# Objective

Economic inflation impacts on auto sales; purchasing a new vehicle posed the question “Is it really inflation or are the dealerships price gouging Americans?”. Our website serves to explore the variance in MSRP & listed/selling price for two different sample sizes of vehicles in the DFW, Houston, San Antonio, and Austin Texas areas. We also utilized data from Illinois to reference other historical data.

# Data Sources and Cleanup

* kaggle: [Large Car Dataset](https://www.kaggle.com/datasets/cisautomotiveapi/large-car-dataset)
  + CSV file reduced from 478.94 MB to 20.26 MB via Jupyter
* Web-scraping algorithm on new car listings from [cargurus.com](https://www.cargurus.com/)
  + Resulted in 6 separate CSV files later merged via Jupyter
* Kaggle and web-scrape files were imported into PostgreSQL and converted to SQLite.
* After web-scrape not all data elements were supplied for the same columns. With more time we would have used another API to fill in those elements using another web site (https://www.edmunds.com/) for a fuller data set.

# Challenges

* The kaggle data file was too large initially, there were roughly 1 million rows. To effectively run the file in Jupyter, we cut the data in half by filtering vehicle types: Pickup and Passenger Vehicle.
* The web-scraping had to be broken up by query ranges to prevent the website from blocking us and preventing further scraping.
* Postgres was useful for manipulating the data to develop the database, but it was easier to import to the web using SQLite. SQLite had some bad data from JSON which held up the plotting of the markers on the map since we hadn’t cleaned up the inputs upon entry – we deleted the one vehicle causing issues and then it worked very well.
* Utilizing the API’s, whenever we got over 1000 entries the performance started suffering. To remedy this, we ran summary information on the backend on some of the API’s instead.
* No real challenges with Heroku, we linked it to our Main branch.
  + Postgres would’ve limited us to 10k rows. With Postgres we couldn’t implement Views during our query which was another reason we moved to SQLite.

# Analysis Methods

* Utilized bootstrap to initially create HTML/CSS/JS coded site
* Created API built into Flask for plotly and leaflet charts and graphs for visualization by end user
  + **kaggleMakeUrl** = api/v1.0/kaggle/
  + **gougeapi** = api/v1.0/scraped/
* JS library not covered and used = jQuery used for the About section (“about.html”, “where.html”, “why.html”)

# User Driven Interaction

https://gouge-data.herokuapp.com/#home

* Menu Options: Home, About, Map, Visualizations, Meet the Team!
  + Meet the team provides photos, GitHub profile link and LinkedIn link.
* Cars by Dealer (map) allows end user to view dealers on the map by their Vehicle Make. Selecting a dealer on the map shows Avg MSRP vs Avg Dealer Pricing (Gouge Score).
  + Gouge Score based on if Dealer Price is greater than, equal to, or less than MSRP.
* Two interactive grouped Bar Charts allow end user to filter by Vehicle Make and compare the various models and their pricing side by side.

# Project Requirements

1. Your visualization must include a Python Flask–powered API, HTML/CSS, JavaScript, and at least one database (PostgreSQL, MongoDB, SQLite, etc.).
2. Your project should fall into one of the below four tracks:

* A combination of web scraping and Leaflet or Plotly
* A dashboard page with multiple charts that update from the same data
* Should be deployed to Heroku

1. Your project should include at least one JS library that we [did not cover](#_Analysis_Methods).
2. Your project must be powered by a data set with at least 500 records.
3. Your project must include some level of user-driven interaction (e.g., dropdowns, textboxes).
4. Your final visualization should ideally include at least three views.