Predicting Lemons at Wholesale Auto Auction

A Machine Learning Approach

Alex Abraham

Why Should You Care?

- Used car market running hot mid-COVID
  - Spring 2020 shutdowns froze new car production
  - "Sales of used cars soared last year … A used car bought a year ago is worth more now" (*WSJ*, 1/10/21)
  - Carvana - online car retailer - stock price up 800% since 2020 trough
- Wholesale auto auction purchasing decisions are uncertain, demanding
  - On average, there's 1 lemon in 10 used vehicles sold at wholesale auction
  - 30 pieces of information may be available about a prospective purchase
- Data analytics can help auction buyers avoid future lemons
  - Recommendations may be custom-tuned to individual risk tolerance/search preference
- Early success: my prototype model places 90th percentile on global competition leaderboard

Project Context

- Around 2012, Carvana – car retailer – posted wholesale auto auction transaction data for a prediction competition
- Instruction: "predict if a car purchased at auction is a lemon"
- Carvana defines a lemon as a car with “serious issues that prevent it from being sold to customers:
    - tampered odometers
    - mechanical issues the dealer is not able to address
    - issues with getting the vehicle title from the seller"

About the Data

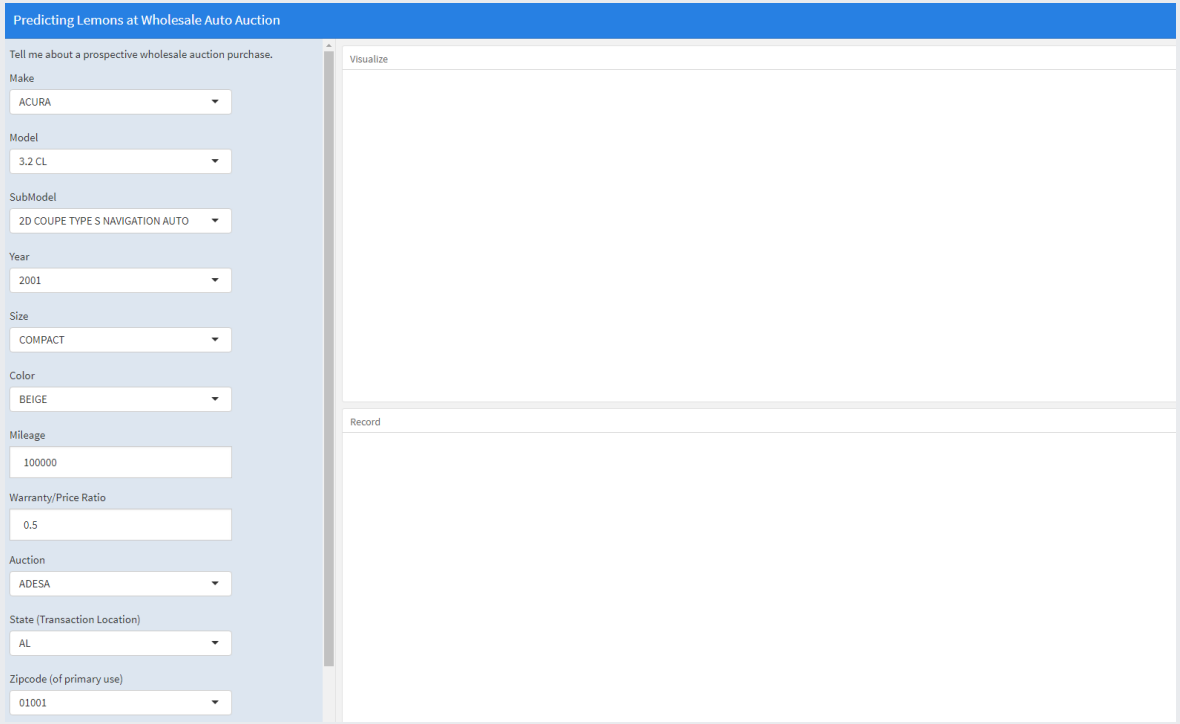
How do these data look in a spreadsheet?

- 70,000 rows (one row represents one vehicle purchased at auction)
- 30 columns (vehicle characteristics; potential predictors)
  - Auction platform, Make/Model/SubModel/Year, Mileage, State/Zip of transaction ...
  - A true/false indicator: does the vehicle turn out a lemon?

Objective

Support wholesale auto auction purchasing decisions, with a software interface to the predictive model

- Input: prospective purchase characteristics
- Output: predicted lemon probability



About the Predictive Model

Natural opening for machine learning - why?

- Theory does not illuminate the underlying relationships
- With many predictors, need automated search for important ones
- Underlying relationships likely complex -- *interplay* between predictors' effects

About the Predictive Model

Relatedly, how does a machine learning model yield my 90th percentile solution?

- Model flexibly learns complex relationships, with little guidance from me
    - Structurally, add together thousands of shallow decision trees
        - Hierarchy of "if-thens" yields a lemon probability
        - "eXtreme Gradient Boosting"
    - Intuitively, this method constructs custom predictors from initial set
- *Incorporate **many** predictors' influences, without memorizing historical data*
    - Model's input data contain hundreds of predictors (which I've specified)
    - Most have a degree of influence on lemon probability

About the Predictive Model Performance

- Prototype is tuned to the analytics competition scoring metric
    - Not directly true positive rate, false positive rate, etc
- Prototype delivers ~25% true positive rate, and < 1% false positive rate
- May be custom-tuned to meet user's purchasing profile!

About the Predictive Model Interpretation

- Broadly -- trade-off between model interpretability and predictive power
- With one predictor's influence depending on other predictors' values, tricky to deliver one-line interpretations
    - Visualizations especially helpful here
- Some (arcane) interpretability tools offer quick predictor importance bits:
    - Particularly influential predictor is 'Wheel type' -- *a data quirk* in this prototype setting
    - Intuitively, Make/Model/Year
    - Car's zipcode matters -- likely a proxy for SES characteristics

About the Overall Deliverable

- Model development is the quick and easy part
- Data preparation *pipeline* is the hard part
- My product entails:
    - Automated data prep pipeline
    - Homemade subroutines bundled into a reusable library/package
        - Example of a superb library: Python's 'scikit-learn'

Supporting Evidence

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
xgboost1.csv	a minute ago	1 seconds	1 seconds	0.24764

Complete

48	▼ 12	Wall-E		0.24810	11	9y
49	▲ 10	occupy		0.24763	6	9y

(Over 500 total competitors)