Craigslist Car & Truck Prices

Can Supervised Learning Predict Through the Scams?

Research Question

Can supervised learning techniques predict car & truck listing price on Craigslist?

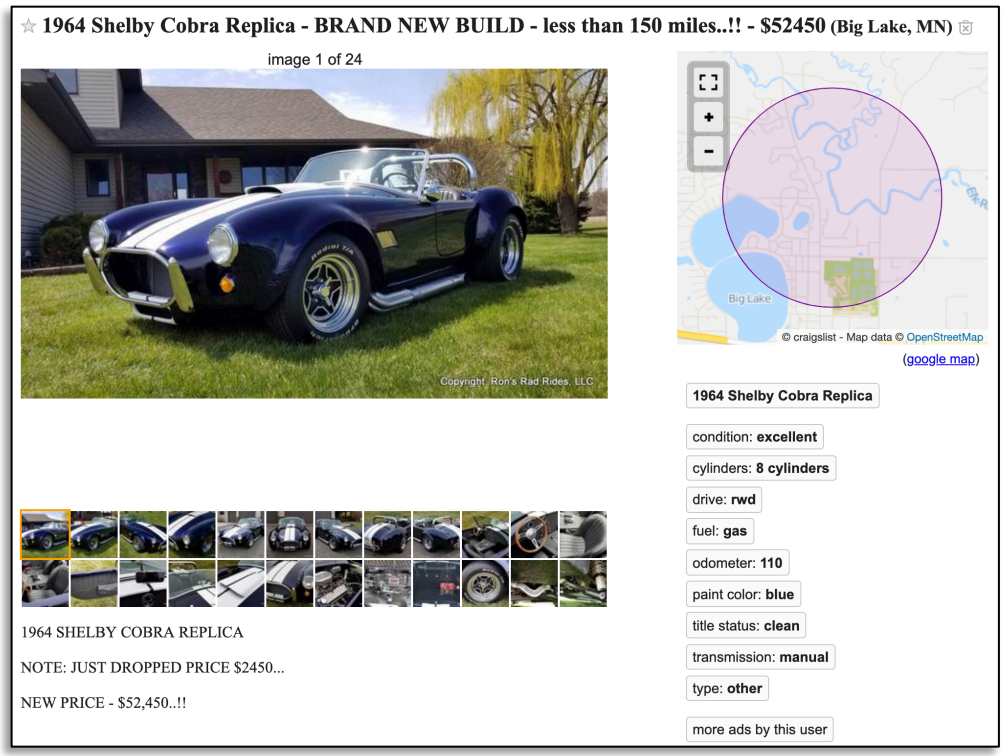
Project Purpose & Stakeholders

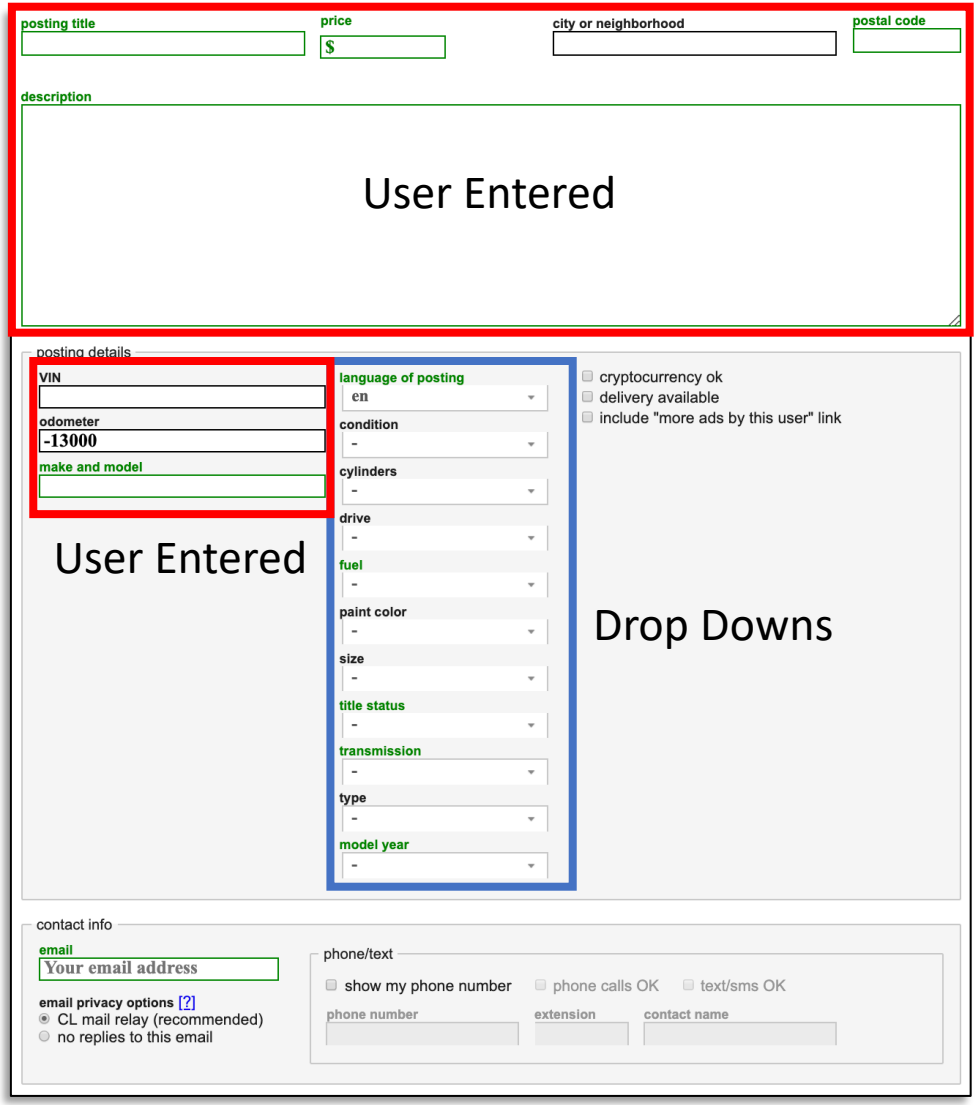
- Craigslist has become a crucial part of the new and used vehicle markets in North America.
- It has become such an important marketplace that car dealerships have begun posting new and used vehicle adds on the website to gain exposure to users who want an uncomplicated approach to car shopping.
- With the predominant number of postings being from non-industry linked users, it could be considered a crowdsourced pricing model for new and used cars.
- By analyzing the prices and descriptive characteristics of the vehicles we can build predictive models for future sellers or buyers to understand what a 'good price' is for a vehicle.

Table of Contents

Data Collection

- Data comes from Kaggle user @AustinReese
- Scraping was done over Oct. and Nov. of 2018
- Includes all North American car & truck postings





https://www.kaggle.com/austinreese/craigslist-carstrucks-data
https://github.com/AustinReese/craigslistFilter

Data Cleaning

- Most important data for the model is 100% user-input
  - Including price, car manufacturer, and car model
- Model and Manufacturer names were the most prone to typos and ‘over-shares’
- For example:
  - peterbilt379exhd
  - accord4doorsedan
  - Oddysey
- There were also a lot of missing state_codes (~58,000) which was also associated with some erroneous lats & longs

Data Cleaning: Manufacturer & Model

```python
def word_frequency(df, col):
    words = list(df.loc[:, col])

    word_count = {}

    for word in words:
        if word in word_count:
            word_count[word] += 1
        else:
            word_count[word] = 1

    word_df = pd.DataFrame.from_dict(
        word_count,
        orient='index',
        columns=['frequency']
    )
    word_df.sort_values(
        by='frequency',
        ascending=False,
        inplace=True
    )
    word_df = word_df.rename_axis('word').reset_index()

    return word_df
```

Convert the dataframe column into a list & generate dictionary to store word counts

Loop through list, if the word exists in the dict keys then add 1 to the value, if it doesn't, create an instance of the word and set the value to 1

Convert the dict keys and values into a dataframe and sort the dataframe by the frequency count

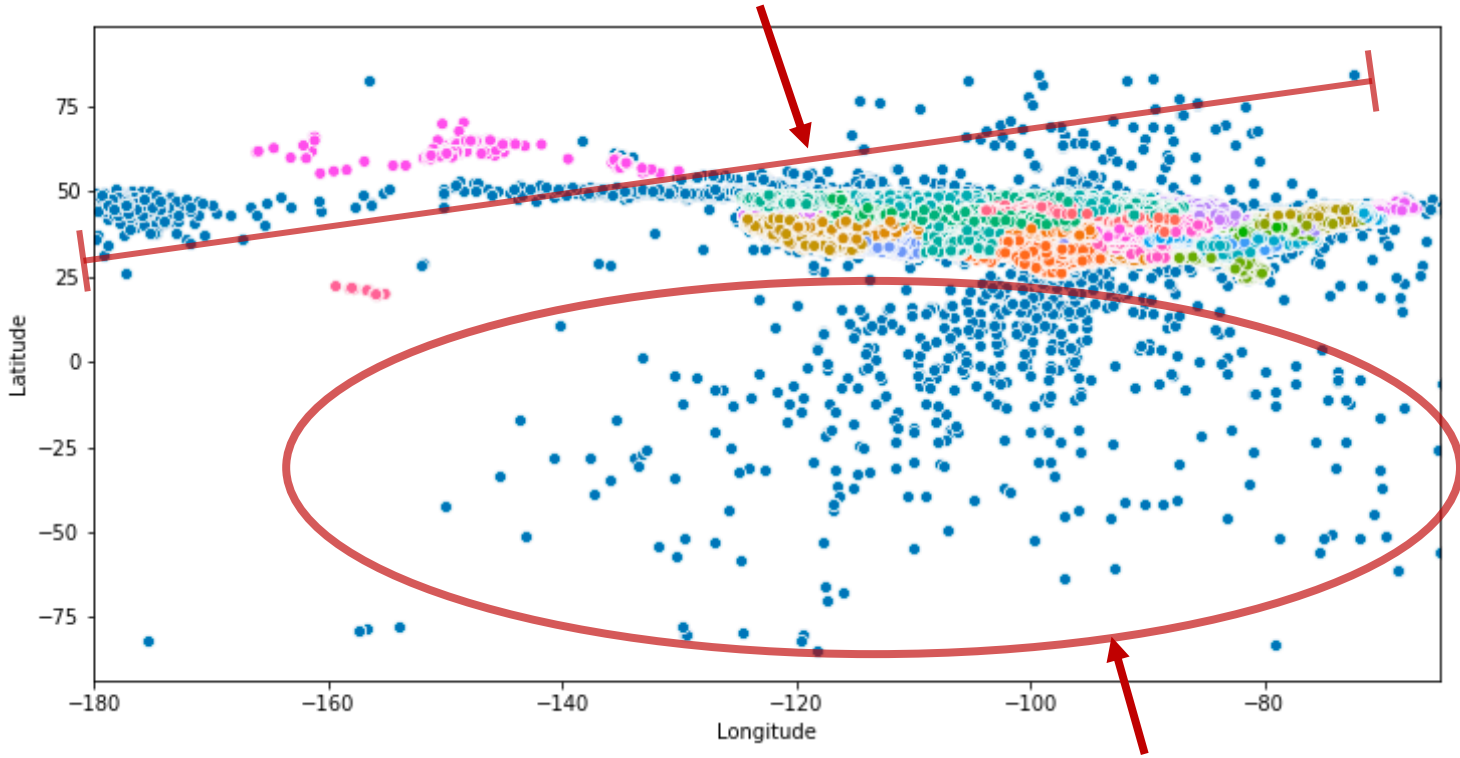
Return the dataframe that was created for frequency analysis and imputation

- This was done in two separate iterations with some typo data cleaning on the most common names
- Initial model count: **88,260**
- Frequency model count: **514**
- **94%** of the rows accounted for that initially had model names
- Raw model column contained ~60,000 nulls

Data Cleaning: State Codes & lat/long

- Turns out most of the missing state codes (~56,000) were from cities in Canada
- The other 2000 missing codes also had erroneous lat & long
- Dark blue points are null state_codes

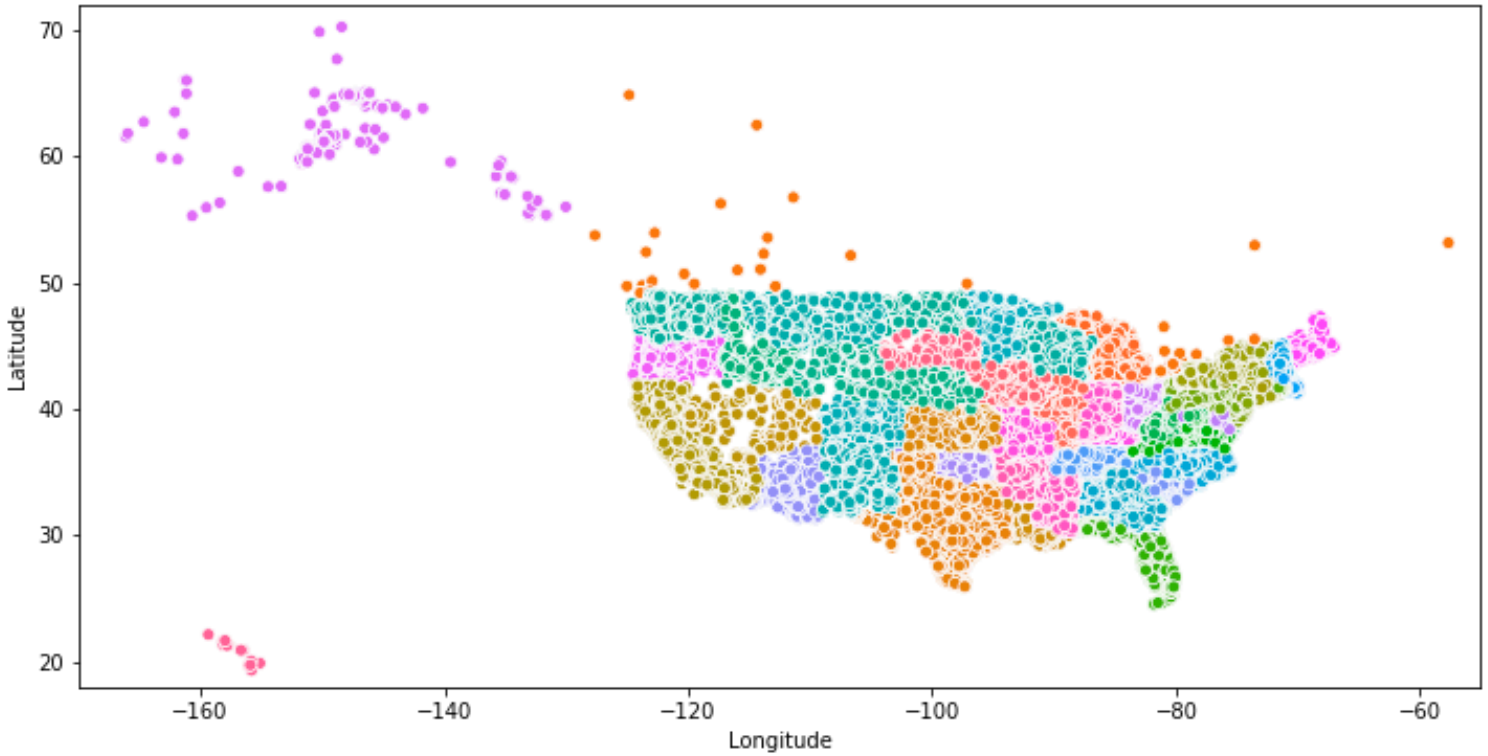
Canadian nulls in Russia and China?



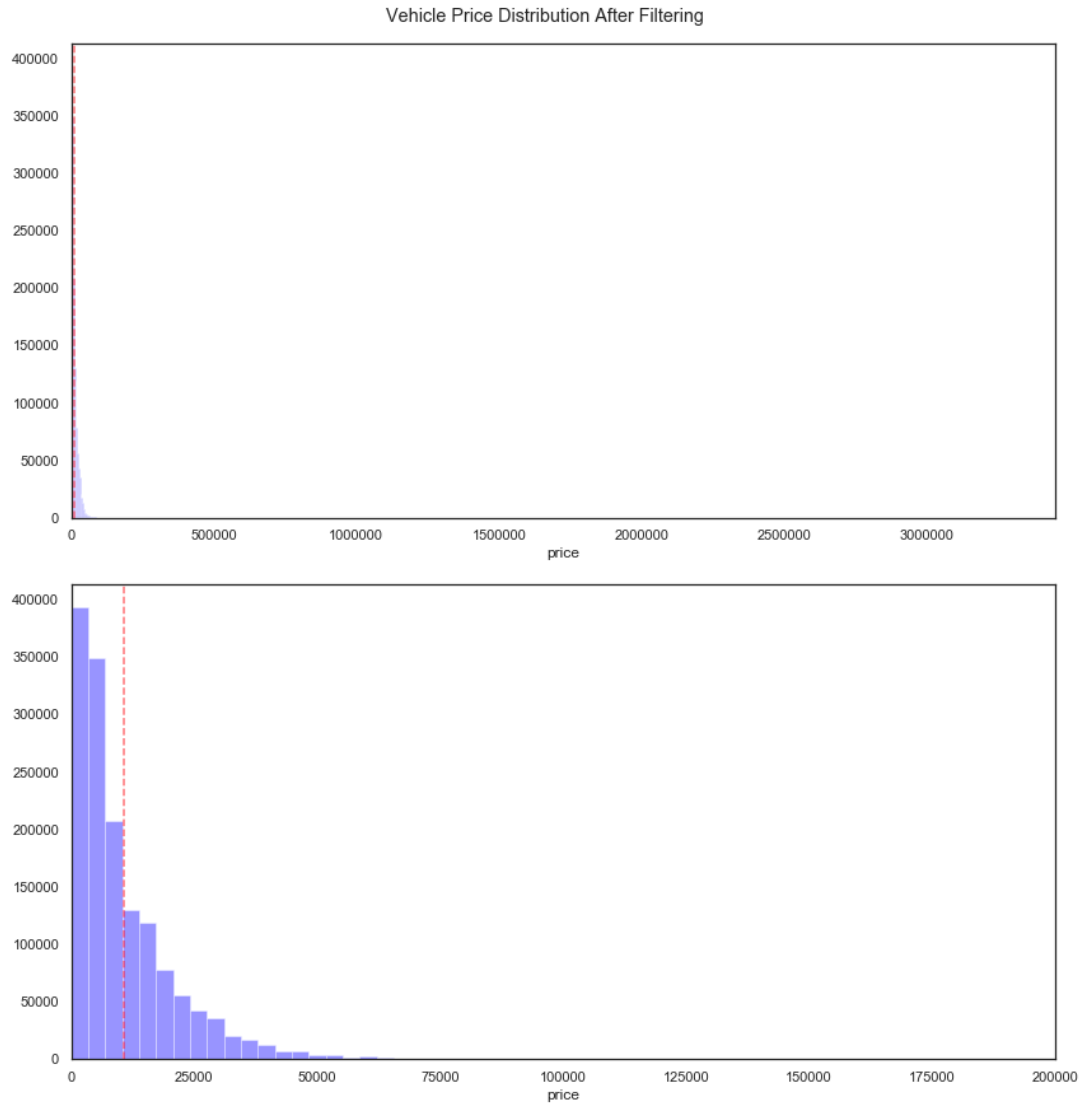
other nulls

Data Cleaning: State Codes & lat/long

- Final solution involved:
  - Assigning Canadian cities the state code ‘CAN’ and their city center lat & long
  - U.S. cities → state frequency imputation
  - Using states avg. lat & long to correct rows with null state code’s position



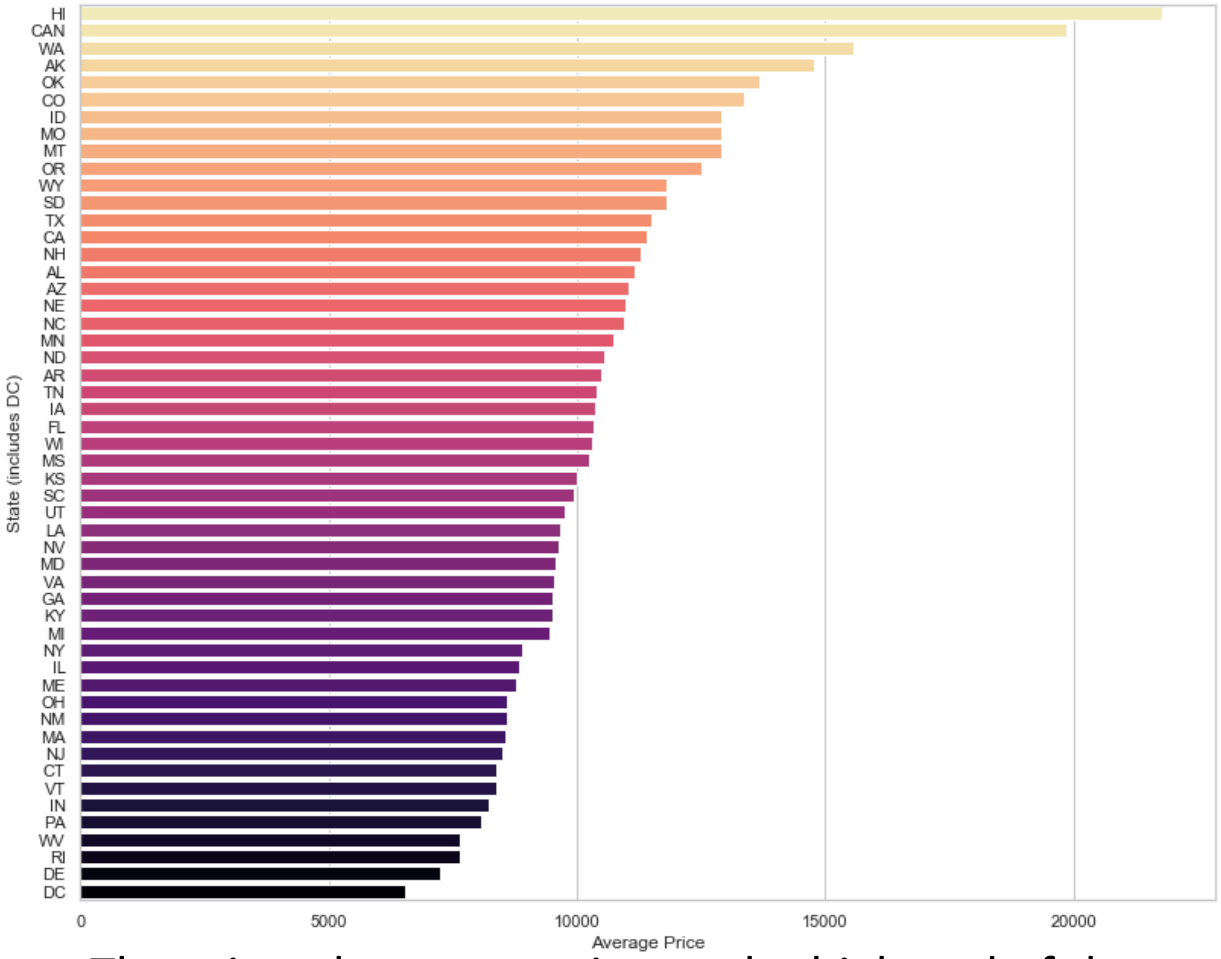
Exploratory Data Analysis

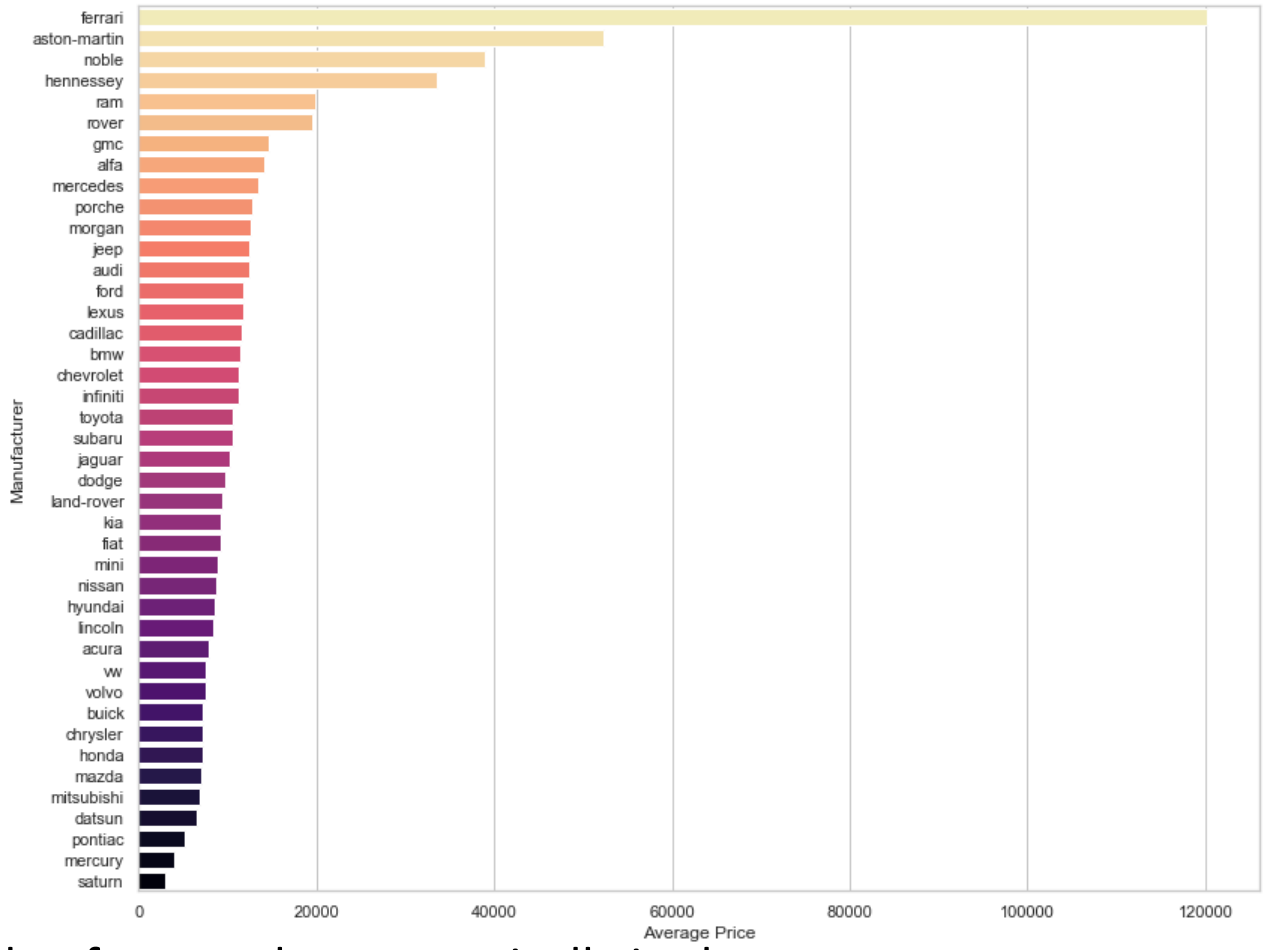


Craig DataFrame	Value
raw unique rows	1,509,089
columns	24
rows removed on price[1]	15,644
motorcycles removed	731
odometer's > 1,000,000	1908

[1]Rows where price == $1 and price >$200,000 that were not supercars

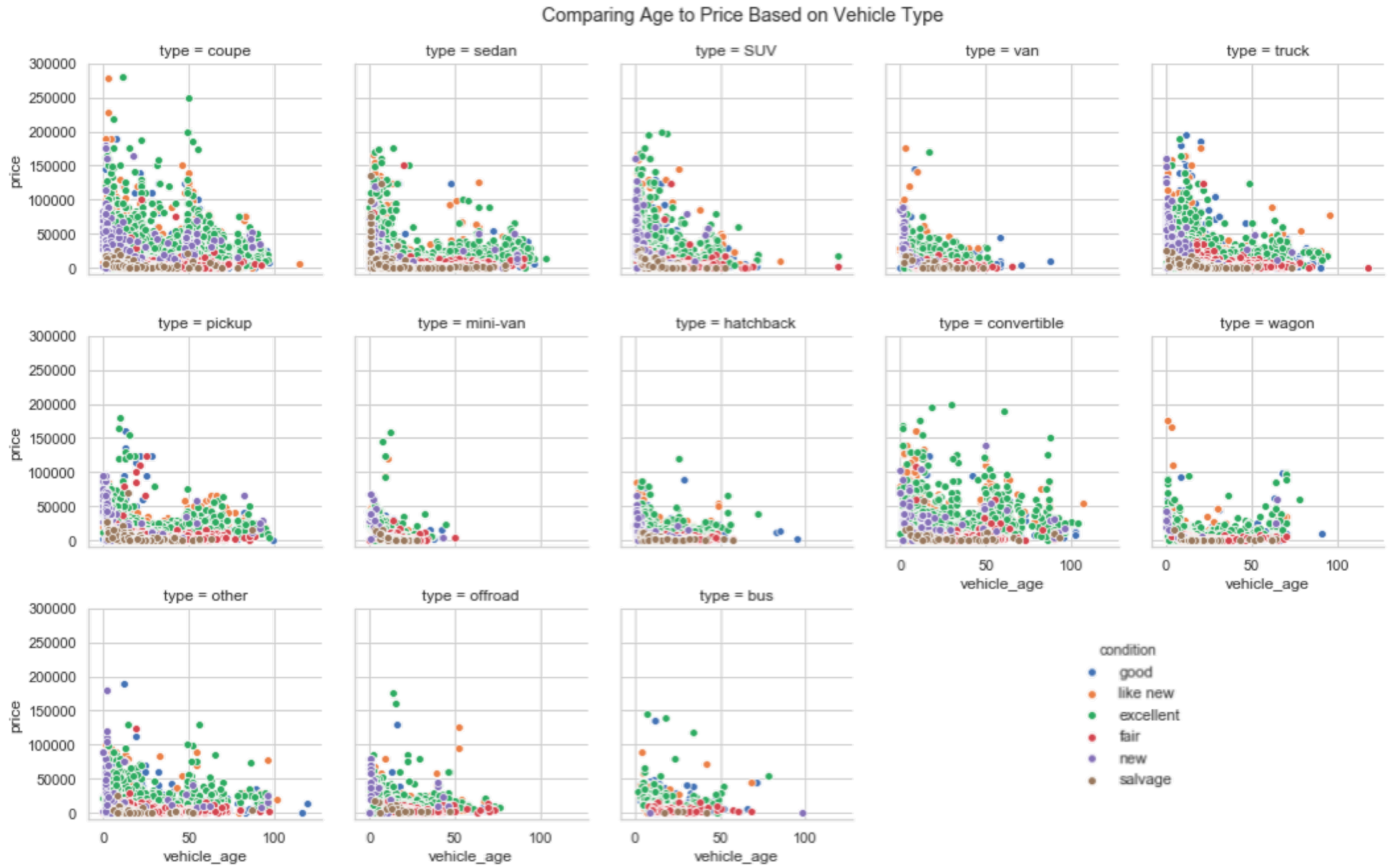
Exploratory Data Analysis





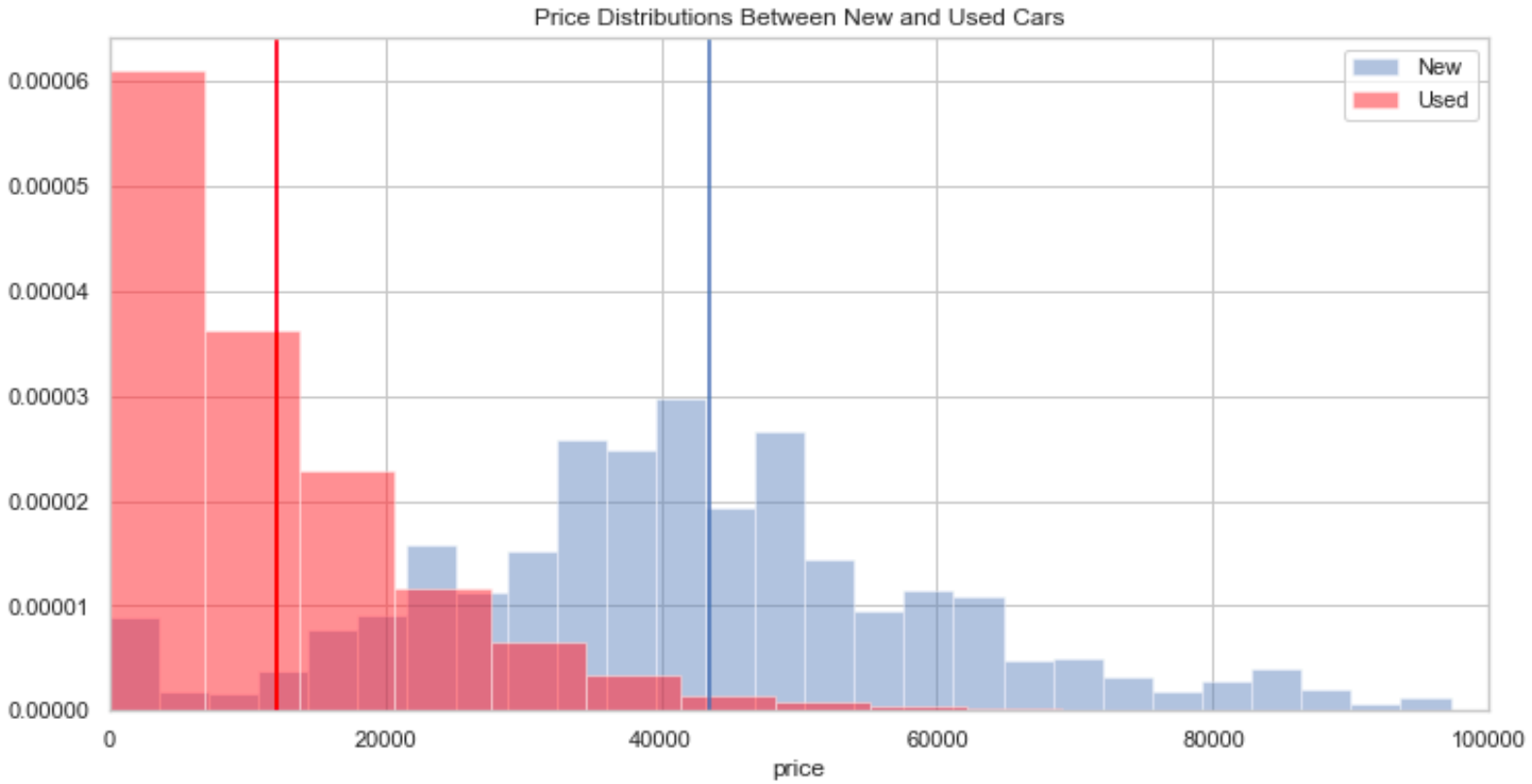
- There is a clear separation at the high end of the market for cars that are typically in the super category
- Hawaii and Canada are the top ‘states’ in the database along with Washington and Alaska
  - Canadian postings could be in CAD which could explain the higher list price (1:1.3 us:can at time of scraping)

Exploratory Data Analysis



- The price of the vehicles decreases as they get older in all but one case
- Wagons have a resurgence in price after they are 50 years old
- Convertibles and coupes seem to retain their value with age better than the rest of the dataset

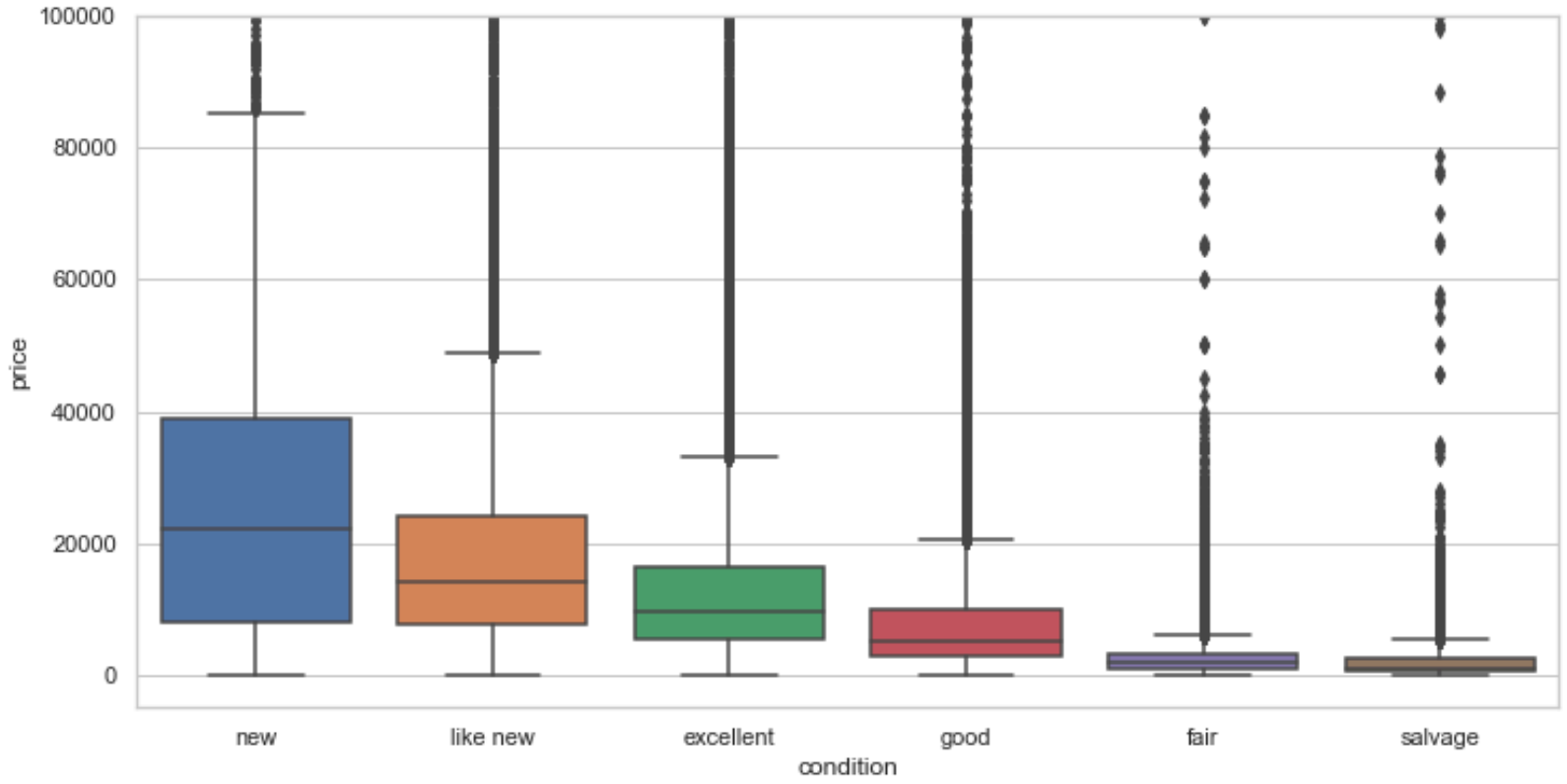
Exploratory Data Analysis



*New cars denoted by mileage < 300 and years 2018 or newer

- New vehicle prices are on average much higher than used, which is not much of a surprise, but will be helpful in modeling
- More importantly, new vehicle price distributions are normally distributed whereas used are log-normal

Exploratory Data Analysis



- Vehicle's condition that is excellent, like new, or new yields a much higher price on average than the rest of the data.
- A condition of excellent has a mean that is roughly equal to the rest of the dataset

Feature Engineering

- All plots from the EDA are in the notebook for feature justification
- Additional categorical encoding using vtreat was used on all applicable columns in the dataframe

Feature Names		
vehicle_age	is_new	Is_salvage
odometer	f250_pricey_state	no_vin
car_size_full_size	stang_pricey_state	well_used
is_supercar	pricey_model	quarter_dead
gross_color	is_canadian	silverfox_wagons
is_truck	is_hot_coupe	condition_is_fair
pricey_state	is_hot_convertible	condition_is_good
is_auto_fwd	part_of_a_car	

Craigslist Engineering

Sample Code

.loc[mask] assignment

```
Mask = (
    (df.col1 > #) &
    (df.col3 == 'text')
)
df.loc[:, 'feature'] = 0
df.loc[mask, 'feature'] = 1
```

Dummies

```
feature = pd.get_dummies(df.col1)
df = pd.concat([df, feature], axis=1)
```

Math

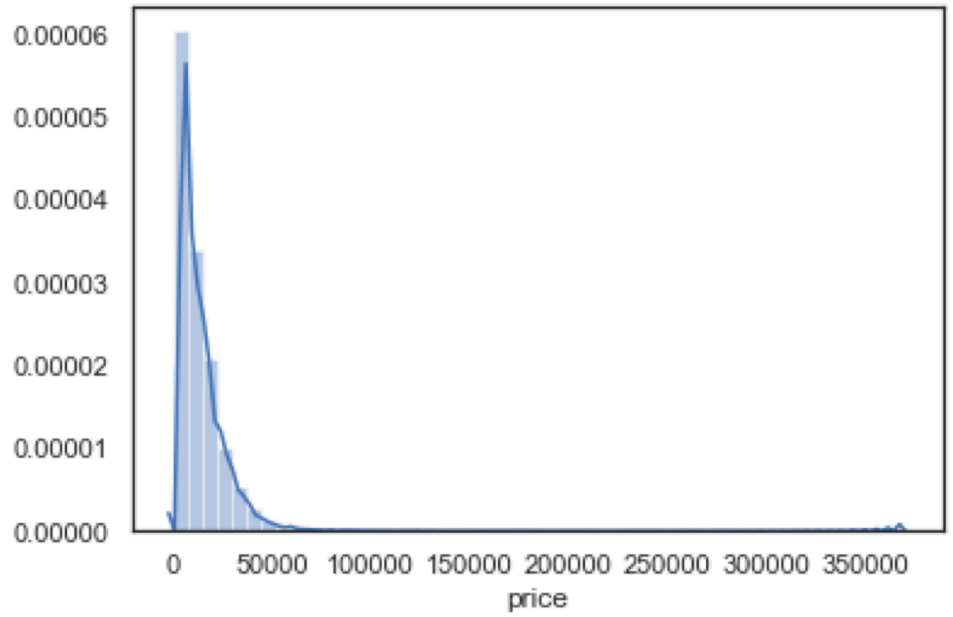
```
df.feature = df.col1 * df.col2
```

Train/Test Split & Data Handling

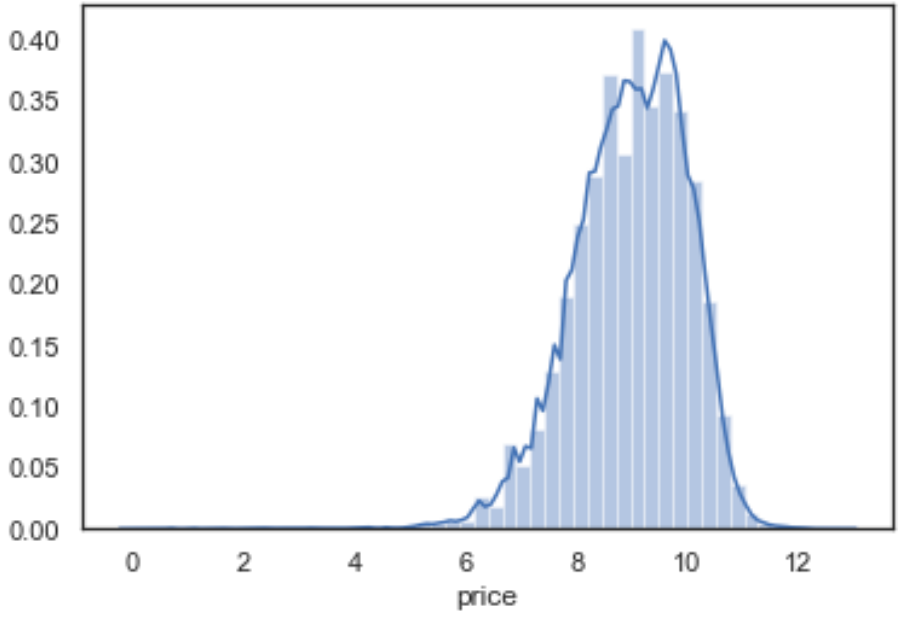
Group	# of Rows	% of full df
Training	631,329	64%
Validation	157,833	16%
Test	197,291	20%

- Transformed target variable and ran models pre and post transformation with mixed results
  - Y = df['price'].apply(np.log10)
  - Y_pred = df['Y_pred'].apply(lambda x: 10**x)

Price Distribution (original)



Price Distribution (post transformation)



Modeling: Random Forest

- Based on R^2 there are no overfit issues with the model
- RMSE is too high, the mean price of the df is ~$11,000
- Hyperparameters tuned with 5-fold CV optimizing for R^2 using skopt.BayesSearchCV()
- Model used was sklearn.ensemble.RandomForestRegressor()

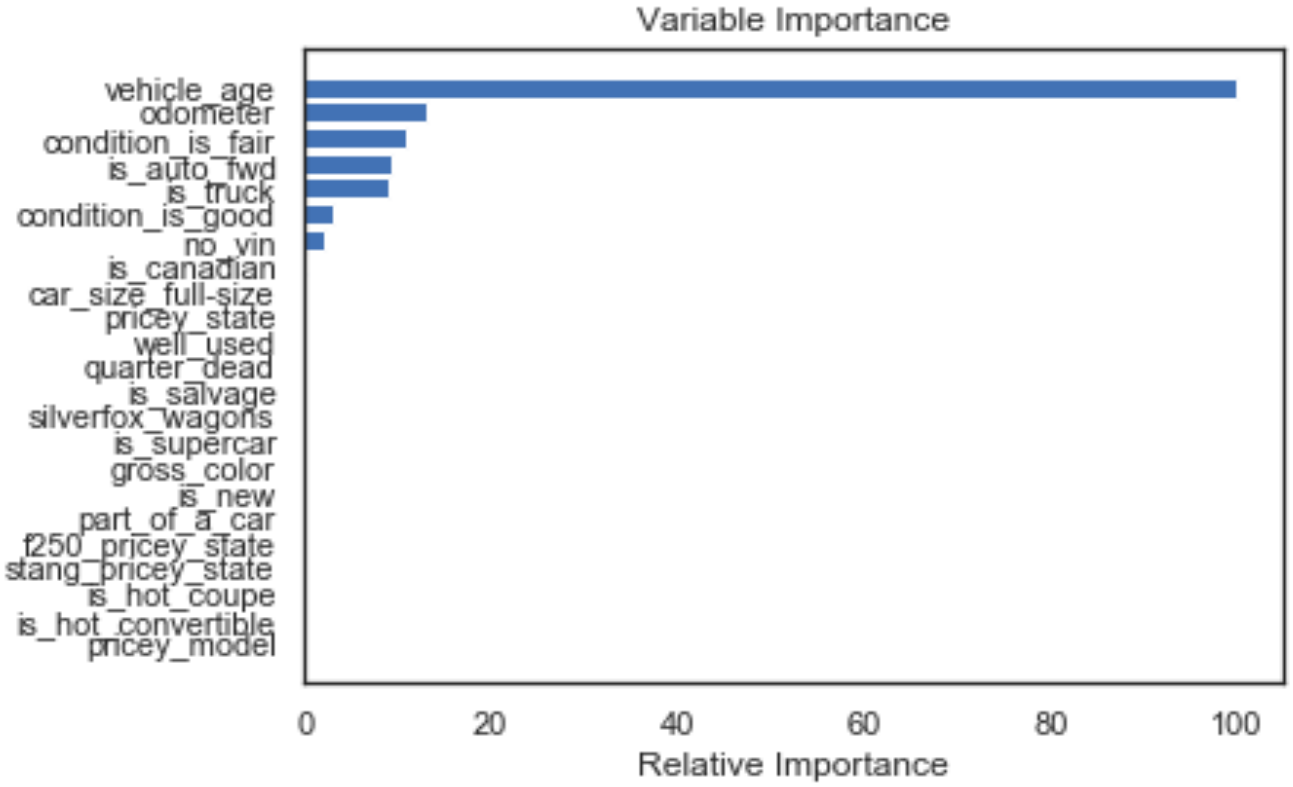
Hyperparameter Tuning

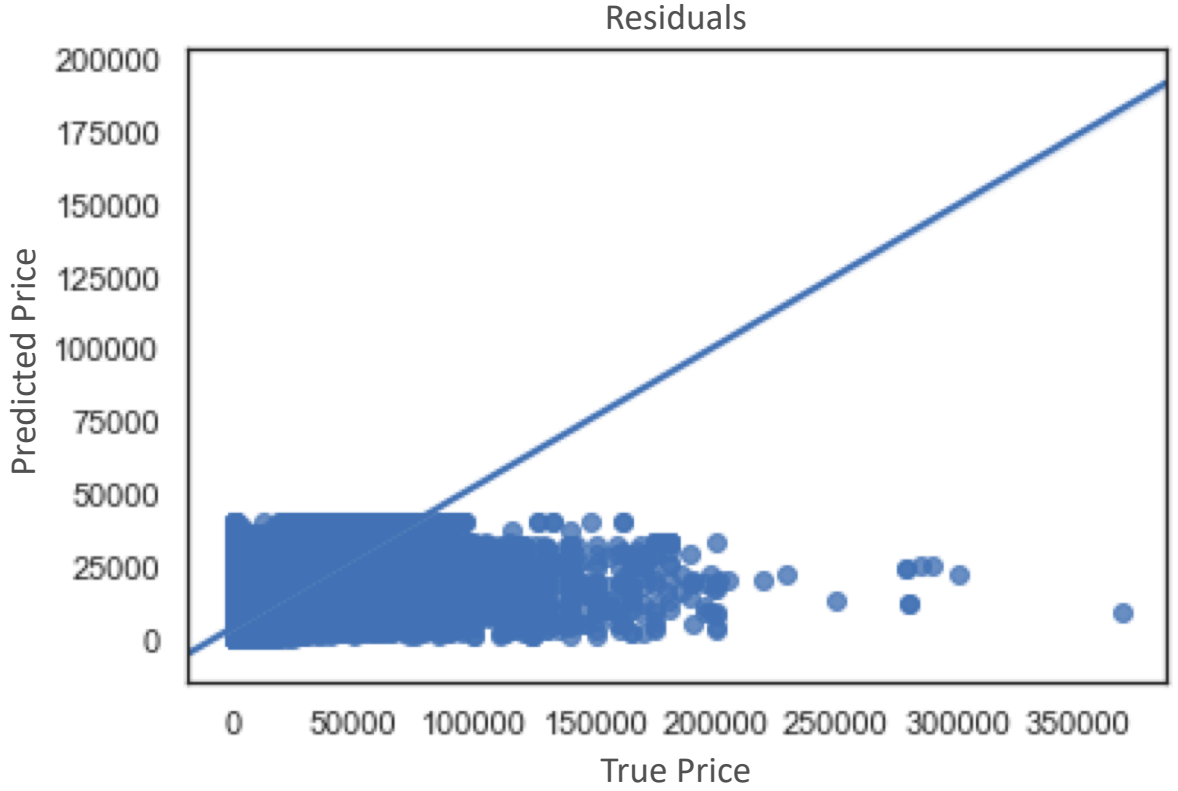
Params	Low	High	Best_param
n_estimators	200	1000	251
max_depth	2	10	9
min_samples_leaf	10	1000	987
max_features	1	23	21

Results

Data	R^2	RMSE
Training	0.58	8189.82
Validation	0.58	8183.02

Modeling: Random Forest





- Random Forest model only labelled 7 features as having any importance at all
- Model would not predict over $50,000, which is in part responsible for the poor performance
- 98% of the training set price is below $50k

Modeling: Gradient Boosting Regressor

- Gradient boosting hyperparameter tuning was not employed due to computational resources
- RMSE is much better than the Random Forest
- Small overfit issue between training and validation
- Model used was sklearn.ensemble.GradientBoostingRegressor()

Hyperparameter Tuning

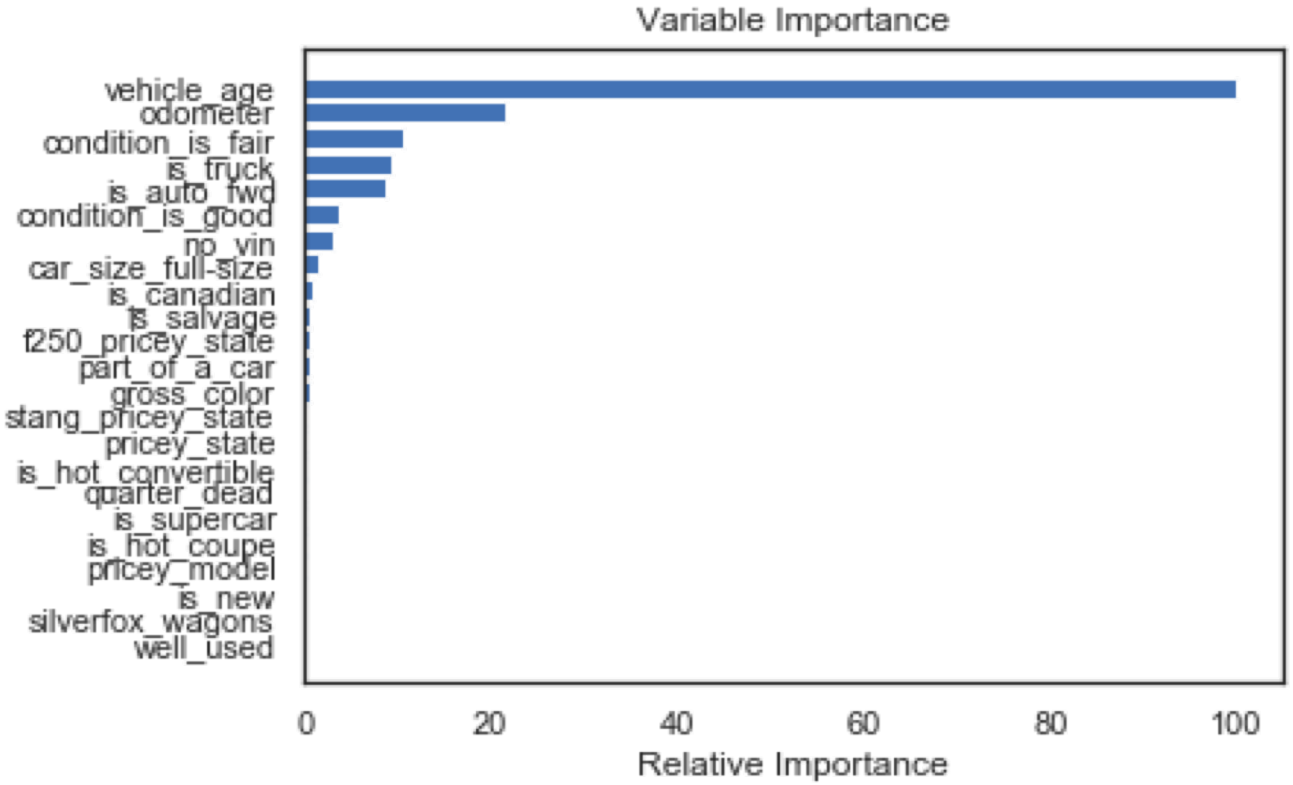
Params	Low	High	Best_param[1]
n_estimators	200	1000	300
max_depth	2	10	10
subsample	10	1000	0.8
max_features	1	23	23
criterion	[Friedman_mse, mse, mae]		mse

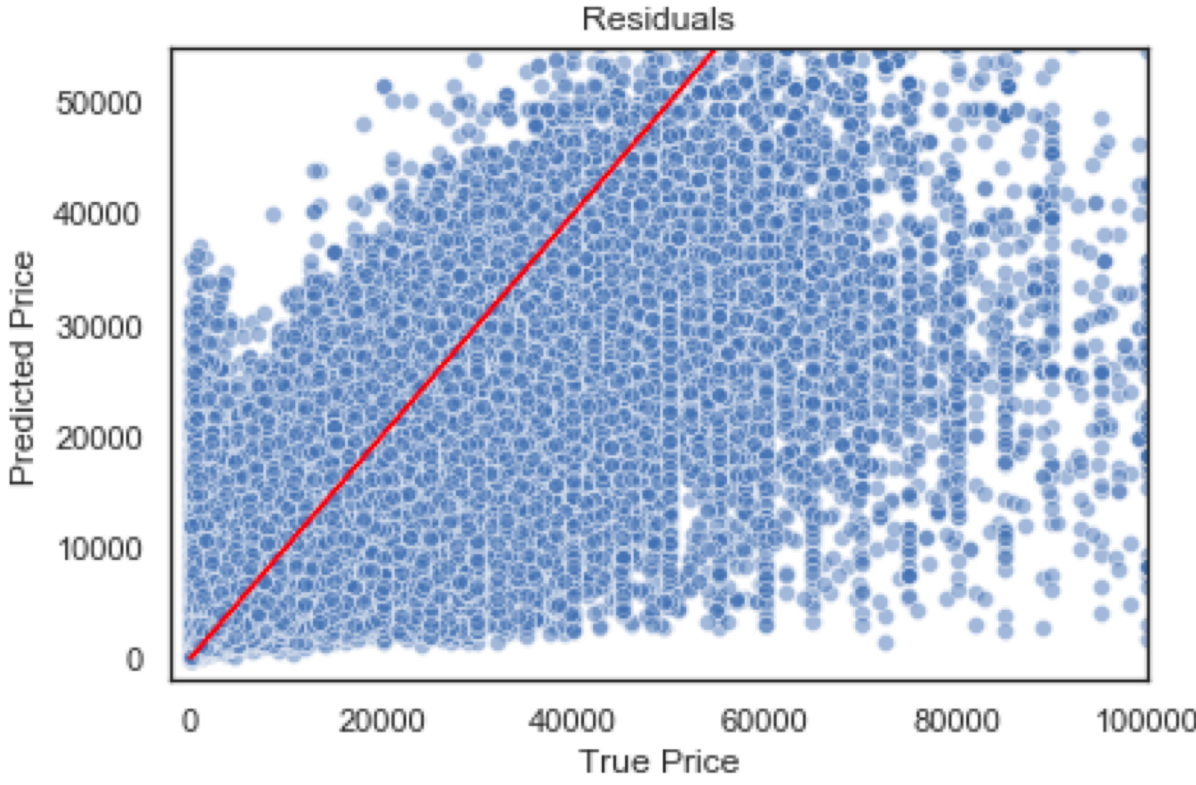
[1]Estimates of best parameters

Results

Data	R^2	RMSE
Training	0.68	7056.14
Validation	0.62	7466.98

Modeling: Gradient Boosting Regressor





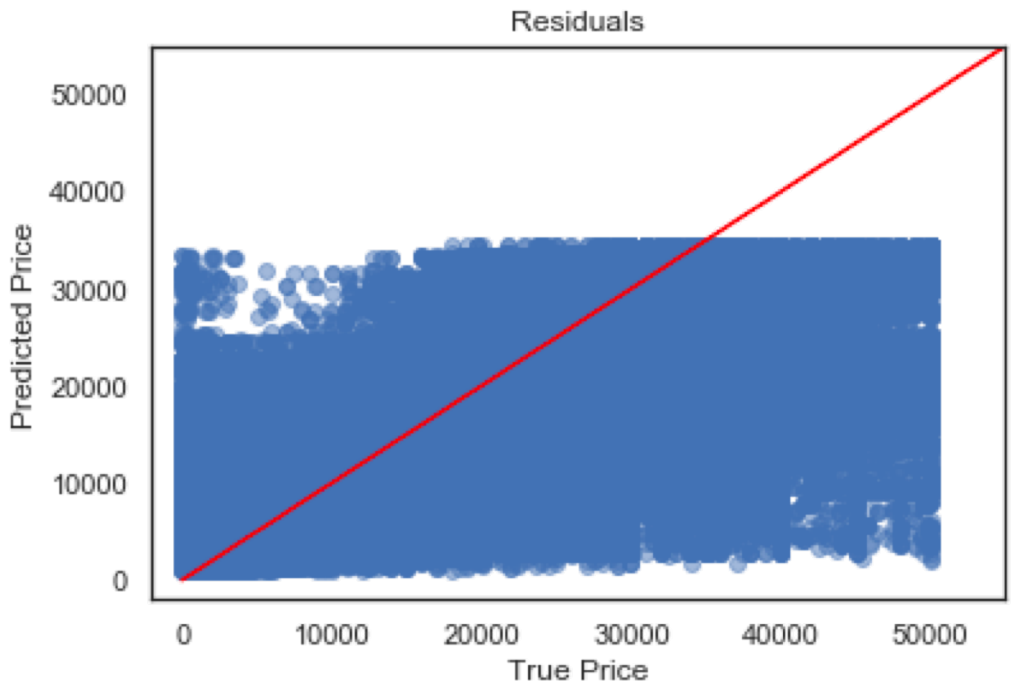
- Gradient boosting includes 13 features as being important
- Model performance is still low given the mean of the dataset, but it at least will predict all ranges of price
- RMSE is being influenced by large errors on abnormally large prices

Modeling: RF price <= $50k

- Used the same params that were already tuned to the data
- RMSE is much lower compared to previous model
- Overfit is still not a concern here and R^2 is slightly worse
- Now the model will not predict over $35k
  - The log transformation and removal of “outlier” parts of the price did not solve our prediction problem
- Feature importance remained the same between the two models

Results

Data	R^2	RMSE
Training	0.57	6227.86
Validation	0.57	6243.52

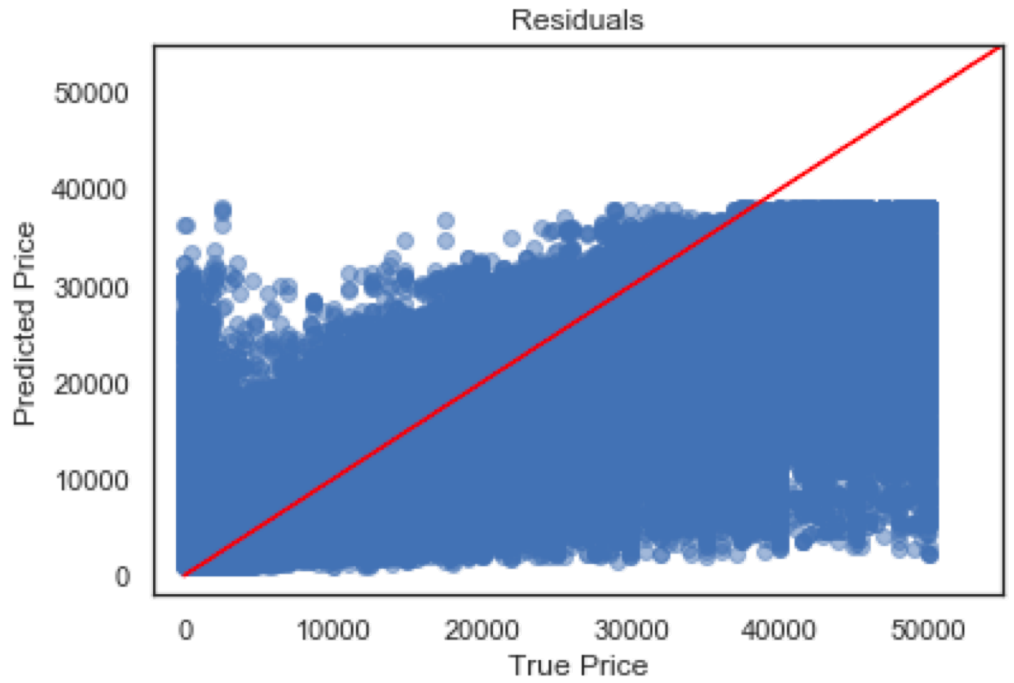


Modeling: RF price <= $50k & vtreat

- Used the same params that were already tuned to the data
- RMSE is still improving
- Starting to see some effects of overfitting
  - Could need to re-tune hyperparameters with larger feature set
- Now the model will not predict over $35k
  - The log transformation and removal of “outlier” parts of the price did not solve our prediction problem

Results

Data	R^2	RMSE
Training	0.67	5245.65
Validation	0.66	5270.66



Modeling: GBR < $50k & vtreat

- Iteration includes limited price dataset and introduces all categorical variables using vtreat module
- Performance is better on all fronts
- Model performance could be enhanced by more rigorous null handling & parameter tuning
- Small degree of overfit between training and validation

Hyperparameters

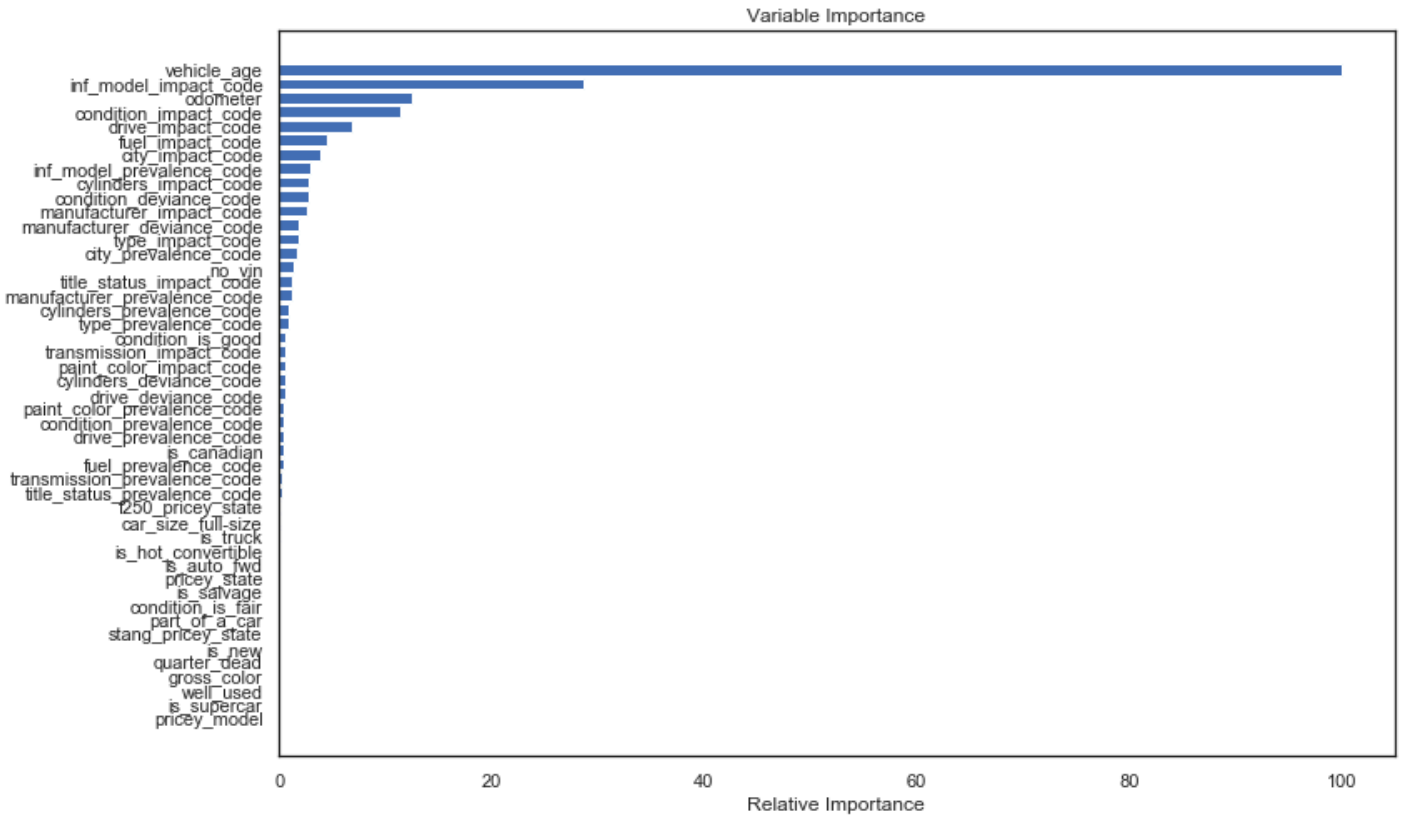
Params	Values[1]
n_estimators	20
max_depth	12
subsample	1
max_features	47
criterion	mse

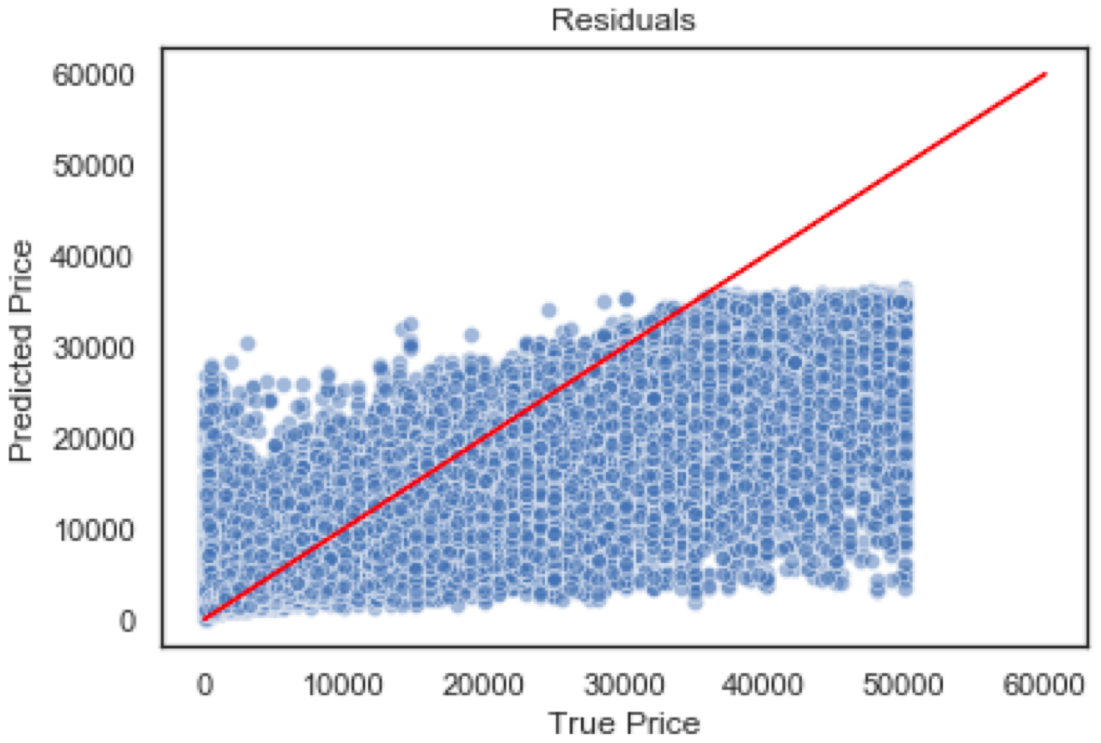
[1]Parameters were updated to reflect larger feature set and adjust for overfitting as well as limited resources

Results

Data	R^2	RMSE
Training	0.79	4531.05
Validation	0.73	4799.27

Modeling: GBR < $50k & vtreat





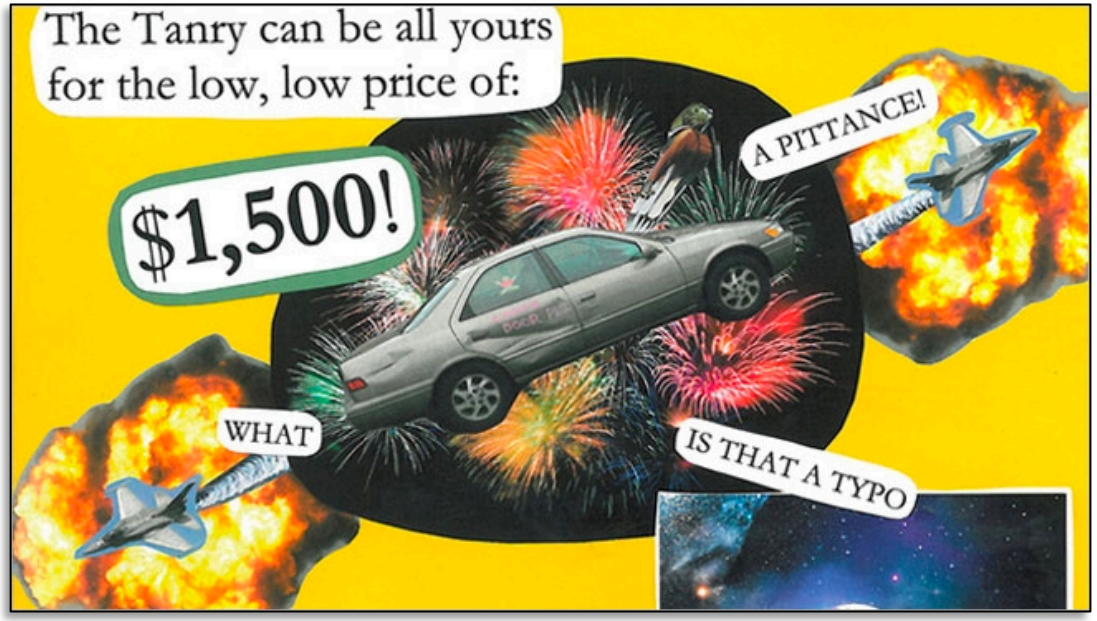
- vtreat categorical encoding provided a lot of predictive power, all features were more important than the discreetly engineered ones
- Model performance is the best here, but the residuals plot tells a similar story to the others

Model Selection

- Based on RMSE and R^2, gradient boosting with vtreat categorical encoding is the best model to predict price in this application
- There is something to be said for how the prices are generated
  - User has complete control on what is entered
  - No sold price is included, which would likely show stronger correlations to vehicle characteristics and price (i.e. the price someone is willing to pay, not the price it is listed at).
  - Usability of the model might be limited to sellers during the posting process

Selected Model Results

Data	R^2	RMSE
Training	0.79	4531.05
Validation	0.73	4799.27
Test	0.73	4764.02



Future Considerations: Data Collection

- Re-write scraper to collect posted date/time
- Include Descriptions with each posting
- Collect the length of “more adds by user” to denote dealerships or to classify if they are active/experienced user
- Track postings with continual scraping to evaluate when posts appear and disappear
  - Could help identify legitimate offers

Future Considerations: Data Cleaning

- Impute odometer values based on age of the car
  - Maybe use this to infer a year?
- Break out model name, but retain all other information in the cell
  - Use leftover text as column on trim level
  - Would need to do similar frequency based cleaning tactics
- Use model names to impute missing manufacturer names

Future Considerations: Modeling

- Allow for more computational resources to more effectively tune models
- Try adding more features
- Re-fit models with features assigned to importance
- Try additional models

Questions & Acknowledgements

Thanks to my mentor Brett Nebeker

&

My family