A Comparison of Machine Learning Approaches to Housing Value Estimation

Orton Babb*
Department of Mathematics, George Mason University

Abstract

Housing value estimation relies on hedonic pricing models whereby price is determined by both internal characteristics (bedrooms, bathrooms, living area, etc.) as well as external characteristics (neighboring houses, ZIP code, etc.). While classical parametric models based on linear regression analysis have been well studied in this application, the theory of hedonic prices places no restrictions on the hedonic price functional form, and hence, more recent research has attempted to apply machine learning (ML) approaches such as K-Nearest Neighbors and Support Vector Machine Regression (SVR). Many of these ML methods are employed on the basis of their flexibility in terms of making less assumptions on the shape or distribution of the data. ML models are therefore used with the expectation of higher accuracy on predicting the final sale price of a house. In this study, we consider the combination of various pre-processing procedures and candidate models on a historical data set of house sales in King County, Washington. Different measures of accuracy are considered in interpreting model performance. The results suggest that while machine learning algorithms like SVR achieve top performance as measured by the adjusted R^2, classical parametric models can also achieve out-of-sample generalization nearing that of the more sophisticated ML models, with faster training times, no need for feature scaling and more easily interpreted parameters.

1 Introduction

In economic models of the real-estate market for residential homes, buyers and sellers engage in so called "arms-length transactions" whereby sellers attempt to maximize their sale price while buyers attempt to get the best home for some price within their budget. Most importantly, they do not have any relation to each other outside of this transaction. Following this assumption, housing value estimation relies on hedonic pricing models in which price is determined by both internal characteristics (bedrooms, bathrooms, living area, etc.) as well as external characteristics (neighboring houses, ZIP code, etc.). In this case, a house is a composite good which may be broken down into its constituent parts from which the contributory value of each part may be determined. In other words, characteristics of the house may be represented by a vector $x \in X \subseteq \mathbb{R}^n$ where the estimated price is given by $\hat{p}(x)$. Buyers of type z, then maximize utility, i.e. to solve $\max_x\{u(z, x, \hat{p}(x))\}$ subject to their budget constraints.

However, according to Nesheim [2006], the theory of hedonic prices places no restrictions on the hedonic price functional form. Different approaches have been taken including the use of linear, log-linear, Box-Cox and fixed effects models. It is suggested that there are no theoretical grounds for a restrictive parametric empirical model unless prior knowledge of the market and products support such restrictions. If such knowledge is not given, then nonparametric estimation is recommended, particularly when the number of examples is large relative to the dimension of x. When the goal is prediction out-of-sample, satisfaction of goodness of fit and stability criteria may be sufficient to justify the choice of functional form.

The online real estate database Zillow.com offers users a Zestimate® estimated market value for individual homes based on recorded sales for over 110 million homes and proprietary automated valuation models which are reported to have a median absolute percentage error of 4.3%.[1] This Zestimate®, while being exceptionally accurate on the median home, may be less accurate in certain areas where there is less availability of historical data or in luxury real estate markets where prices tend to be much higher. This is suggestive of real estate market data being heteroscedastic, i.e. the variance in prices is itself an increasing function of the features of the house. Therefore, robust regression methods may be useful in this application. The reported performance of these proprietary models appears to be the best evidence that automated valuation can achieve goodness of fit.

*obabb@gmu.edu
[1] https://www.zillow.com/zestimate/

Using learning algorithms for model selection in the functional form of a pricing model

Within the past decade, researchers have attempted to apply machine learning approaches in housing value estimation. Del Cacho [2010] compared various data mining methods (from linear regression to decision trees, neural networks and other ML methods) for real estate appraisal on a data set of about 25,000 properties in Madrid, Spain. It is noted that the rising interest in automated valuation models has been driven by the high availability of data accessible through the internet. Results showed that an ensemble of model trees exhibited the highest correlation rates between predicted and actual prices, and the lowest mean relative error— underscoring the suitability of ML models in appraisal schemes.

Gan et al. [2015] also applied decision trees and neural networks to the prediction of real estate price for the years 2012-2013 in King County, Washington using data from the Department of Assessments. The top and bottom 5% of houses (by price) were excluded, and 2012 data were used to train the models, while those from 2013 were used for testing. Estimates of the percentage price increase from 2012 to 2013 were used to create a post-prediction price adjustment leading to a mean absolute error of about \$100,000 for the decision trees and \$85,000 for neural networks. Results were also reported over 10 price categories demonstrating higher mean absolute errors when house values are higher for both models. It is noted that, the results conflict with that of Del Cacho [2010] as to the superiority of tree based models—casting doubt on the primacy of any single ML paradigm over others for this application.

Oladunni and Sharma [2016] conducted an empirical study of the suitability of learning algorithms including Support Vector Machine Regression (SVR), K-Nearest Neighbors (K-NN) and Principal Component Regression (PCR). This study was motivated by the fact that application of the hedonic theory using ordinary least squares (OLS) linear regression has been well studied, but other options for the functional form of these models should be considered. They applied feature scaling to the data,

$$x' = \frac{x - min(X)}{max(X) - min(X)}$$

where lowercase, x, refers to a single point and uppercase, X, refers to the entire data set. (More details later) Their results indicated that SVR had a Spearman's rho correlation coefficient of approximately 0.87, while PCR had a coefficient of 0.89. They suggest that these machine learning approaches to value estimation are superior to manual appraisal mechanisms which are open to the sort of price manipulation that may have contributed to the most recent financial crisis, and conclude that hedonic pricing theory is implementable using PCR, K-NN, and SVR.

While the results of Oladunni and Sharma, Del Cacho, and Gan et al. demonstrate the capacity of an ML approach, it is important to know how much SVR and other models outperform traditional approaches to hedonic pricing, how much of a difference the feature scaling makes, and which available accuracy measures are appropriate for measuring goodness of fit. In the category of ML models, we place extra emphasis on SVR due to its origins in statistical learning theory which is a useful explanatory framework both for non-parametric model selection when the functional form is unknown (with its ability to employ different kernels) as well as interpretation of stability (i.e. model variation as the input changes) in terms of so called "structural risk minimization" which protects against over-fitting. The regression tree also has flexibility in terms of the functions that can be approximated and allows for bounds on the tree growth that may also help with out-of-sample generalization. Many linear regression[2] models (with or without robustness or regularization) are considered as more standard approaches whose parameters may have more easily interpreted economic significance when coupled with theoretically informed structural equations regarding market demand and the utilities of agents (although such equations will not be covered). The models mentioned here were also chosen because implementations of efficient solvers were readily available in MATLAB®. Studied together, these models will give each other more context and help to determine whether ML models provide any advantages over simpler methods.

Therefore, the study considers the following three objectives:

i *Compare the performance of Support Vector Machine Regression (SVR) using a variety of kernels, Regression Trees, Lasso Regression, Ridge Regression, Ordinary Least Squares (OLS) and robust linear regression in the housing value estimation problem*

ii *Assess the impact of pre-processing procedures on model performance*

This paper is organized as follows: First, we give a brief description of the data set used. Next, in the methodology section, the pre-processing procedures, models and accuracy measures tested are defined. Finally, the most salient aspects of the empirical results are presented in the order that they appear in the machine learning pipeline (pre-processing, models, and accuracy measures). Key findings point to the utility of pre-processing and corroborate the effectiveness of machine learning, as in Oladunni and Sharma, but also point to a similar effectiveness of much less sophisticated models for goodness of fit as measured by the adjusted R^2.

[2]We use the term "linear regression" to also include polynomial regression models which are linear in their parameters.

2 Data

We consider data from the "eRealProperty" catalogue of the King County Department of Assessments over the period from May 2014 to May 2015 in King County, Washington. The cumulative file includes 21,613 observations and was retrieved from Kaggle.com[3], an online platform for predictive modeling and analytics competitions. The feature descriptions were taken from the King County Department of Assessments eSales Residential Glossary of Terms.[4] Tables 1 and 2 and Figure 1 summarize the features and target.

Type	**Feature/Target**	**Description**
Ordinal	bedrooms	The number of bedrooms
	bathrooms	The number of bathrooms, reported on quarter point scale
	floors	The number of floors, reported on a half point scale between '1' and '3.5'
	waterfront	Whether property has a waterfront, '1' for yes, '0' for no
	view	Rating of the view from '1' to '4', and '0' if no view
	condition	Rating of the condition from '1' to '5'
	grade	Rating of the construction grade from '1' to '13', i.e. from poor to excellent
	yr_built	The year the house was initially built
	yr_renovated	The year the house was renovated, '0' if never renovated
Categorical	zipcode	The ZIP Code that the house falls in
Continuous	sqft_living	The area of the living area (not to be confused with living room area) in sq. ft.
	sqft_living15	The average living area, in sq. ft., of the 15 nearest houses
	sqft_lot	The lot area in sq. ft.
	sqft_lot15	The average lot area, in sq. ft., of the 15 nearest houses
	sqft_above	The living area, in sq. ft., above the basement
	sqft_basement	The area, in sq. ft., of the basement
	lat	The latitude of house's location on a map
	long	The longitude of house's location on a map
	price (target)	The final sale price of the house in dollars

Table 1: Feature and target descriptions for the King County, WA House Sales Data Set

Features/Target	**Mean**	**Std. Dev.**	**Minimum**	**Median**	**Maximum**
bedrooms	3.37	0.93	0	3	33
bathrooms	2.11	0.77	0	2.25	8
floors	1.49	0.54	1	1.5	3.5
waterfront	0.01	0.09	0	0	1
view	0.23	0.77	0	0	4
condition	3.41	0.65	1	3	5
grade	7.66	1.18	1	7	13
yr_built	1 971.01	29.37	1900	1975	2015
yr_renovated	84.40	401.68	0	0	2015
sqft_living	2 079.90	918.44	290	1 910	13 540
sqft_living15	1 986.55	685.39	399	1 840	6 210
sqft_lot	15 106.97	41 420.51	520	7 618	1 651 359
sqft_lot15	12 768.46	27 304.18	651	7 620	87 1200
sqft_above	1 788.39	828.09	290	1 560	9 410
sqft_basement	291.51	442.58	0	0	4 820
lat	47.56	0.14	47.1559	47.5718	47.7776
long	-122.21	0.14	-122.5190	-122.2300	-121.3150
price	540 088.14	367 127.20	75 000	450 000	7 700 000

Table 2: Data summary (excluding 'zipcode')

[3]https://www.kaggle.com/harlfoxem/housesalesprediction
[4]https://info.kingcounty.gov/assessor/esales/Glossary.aspx

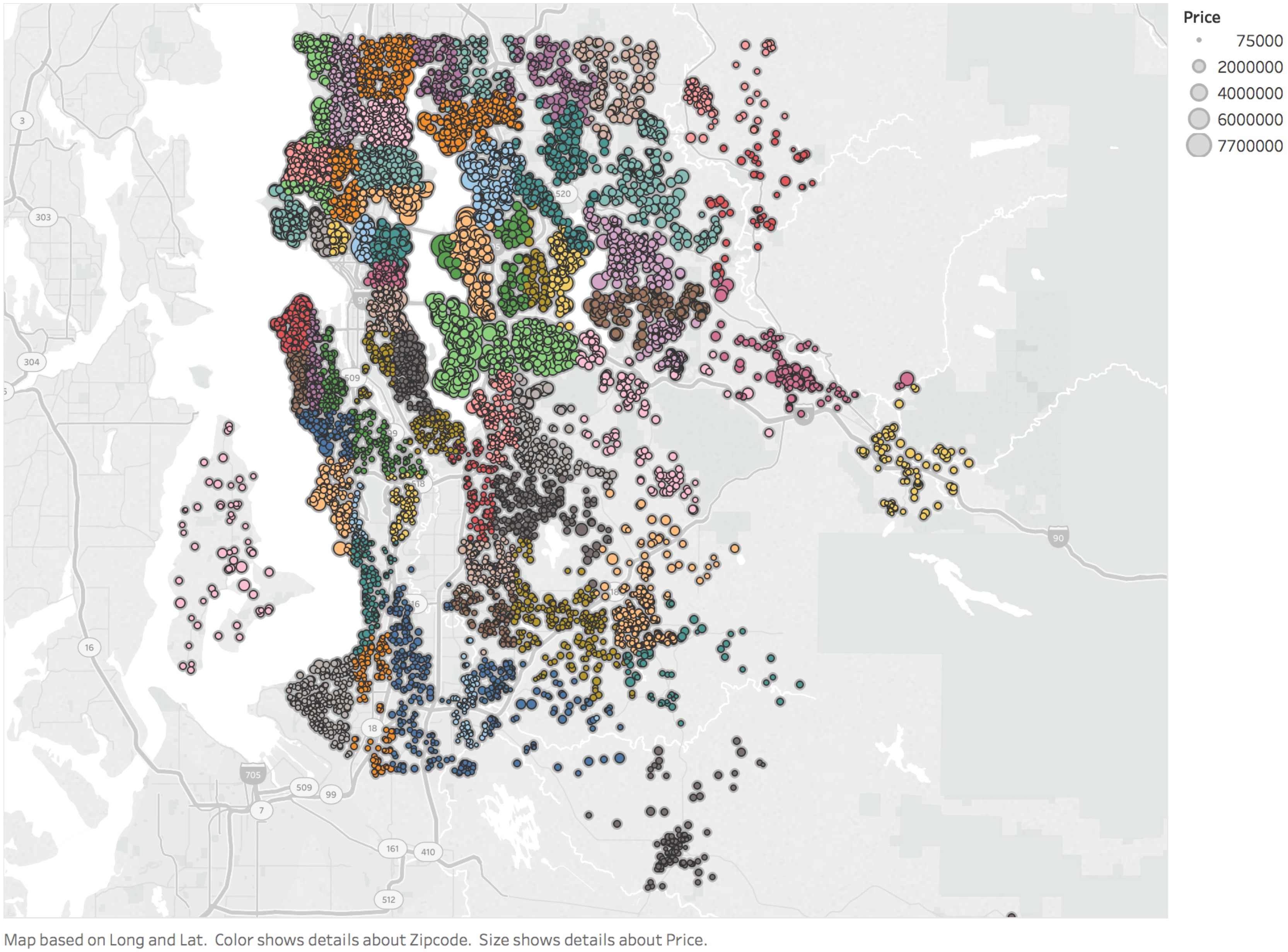

Price
75000
2000000
4000000
6000000
7700000
Map based on Long and Lat. Color shows details about Zipcode. Size shows details about Price.

Figure 1: Price data in King County, Washington plotted proportional to the area of the markers (Visualized in Tableau Desktop 10.3)

3 Methodology

The early stage feature engineering involved several steps. Firstly, histograms and bar charts were produced to inspect for outliers. There was a house which had 33 bedrooms but associated with an unlikely lot size of 6,000 sq. ft and only a single floor. Hence, the record was treated as faulty and removed from the data set. Next, the single categorical variable ('zipcode') which had 70 possible values was transformed to produce 70 numerical one-hot encoded variables that took on a value of '1' when the house corresponded to a certain ZIP Code and '0' otherwise. There was an 'id' attribute which was excluded since it didn't carry any information about the house itself but is used for internal purposes. The 'yr_built' feature was converted to a feature called 'age' according to the formula $age = 2017 - yr\_built$. Likewise, the 'yr_renovated' feature was converted into a new feature called 'effective_age' according to the formula:

$$effective\_age = \begin{cases} age & yr\_renovated = 0 \\ 2017 - yr\_renovated & yr\_renovated > 0. \end{cases}$$

For the training and testing phases, the data were sorted by date and split with the first 80% (earliest in time) for training and the remaining 20% (latest in time) for testing. As seen in Figure 1, the price as a function of location varied a lot across King County. Since the models considered do not all accommodate spatial variables, we exclude the latitude and longitude as features and rely on the ZIP Code features for location. As a result, the input variable, x, consisted of all features mentioned in the data summary with the exception of the target variable ('price'), the latitude and longitude. Together, there was a total of 13 raw features, 2 engineered features ('age' and 'effective_age') and 80 one-hot encoded features (derived from 'zipcode'), making a total of 85. Different forms of pre-processing on x were considered, and model performance was measured on both the training and testing set using various accuracy measures. In the following sub-sections, we will specify the types of pre-processing, models and measures of accuracy used.

Feature Pre-processing

By feature pre-processing, we mean any transformation done to the data to put the features on a common scale. Table 3 introduces the pre-processing procedures. Each pre-processing relies on parameters, derived from the training set, which are stored and used to also process the testing data before input into the trained model. In the following notation, we have a vector of pre-processing parameters with entries for each feature such as $min(X)$ (the minima), $max(X)$ (the maxima), μ_X (averages), and σ_X (the std. deviations). Pre-processings were not applied to the one-hot encoded variables.

Pre-processing	**Formula**
Unprocessed	$x' = x$
Feature Scaling	$x' = \frac{x - min(X)}{max(X) - min(X)}$
Standardization	$x' = \frac{x - \mu_X}{\sigma_X}$

Table 3: Definitions for each form of pre-processing

Models

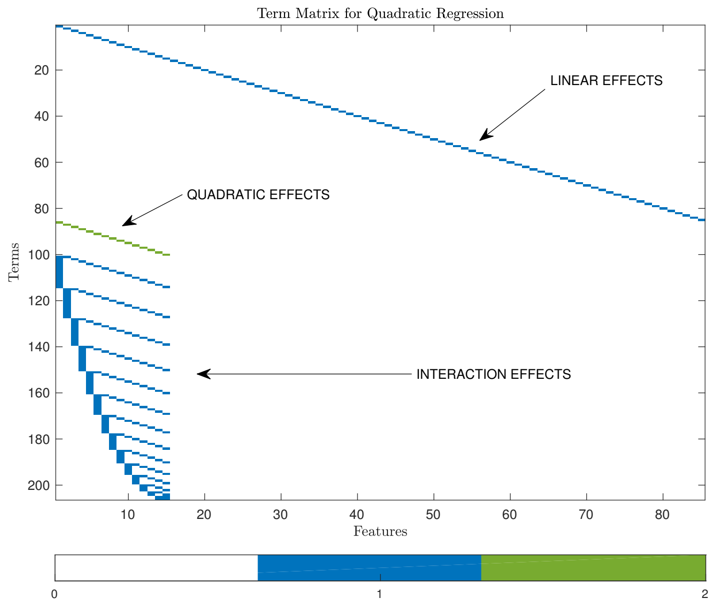
We consider a model to be a combination of functional form, loss function and risk minimization principle. After pre-processing, various instances of these combinations were trained and used to make predictions for the target variable on the testing set. Of the models in Table 4, 'lasso', 'ridge', 'svr-linear', 'svr-polynomial-order2', 'svr-gaussian' and the 'regressiontree' required hyperparameter tuning. MATLAB®'s built in Bayes Optimizer was used. According to Snoek et al. [2012], this sequential design strategy can optimize black-box functions without the use of derivatives and also reduce the amount of objective function evaluations. A treatment of Bayesian model selection and adaptation of hyperparameters along with cross validation is given in Chapter 5 of Rasmussen and Williams [2006]. For a detailed account of robust loss functions and their motivation, see Huber [1981], DuMouchel and O'Brien [1989], Holland and Welsch [1977], Street et al. [1988], or a more recent treatment in Chapter 15 of Montgomery et al. [2013]. MATLAB® includes many robust methods for re-weighting model parameters in linear regression, but preliminary results suggested that it was sufficient to consider only the 'bisquare' iterative re-weighting as a representative for this class of robust options both since it is the default and since there is a lack of *a priori* knowledge about the underlying joint distribution between house features and price.

Model Implementation Details

In Table 4 (pp. 6-7), we introduce the models considered, all of which are available in the MATLAB® Statistics and Machine Learning Toolbox Documentation, and their settings. The notation $\hat{p}(x)$ refers to the estimated pricing function and $\vec{p}_{train}$ refers to the vector of true prices in the training set.

Model	Details
linear-ols	Linear combination of features plus intercept term $\hat{p}(x) = \beta_0 + \sum_{j=1}^{85} \beta_j x_j$ (where x_j refers to the value of the j-th feature for the input and not j-th data point) and least squares loss function

linear-bisquare	Linear combination of features plus intercept term with the robust regression derived from iteratively re-weighted least squares, where the functional form is the same as 'linear-ols' with weightings given by:
	
	 `w` = `(abs(r) < 1).*(1-r.^2).^2`
	 `r` = `resid/(tune*s*sqrt(1-h))`
	 `s` = `MAD/0.6745` where `MAD` is the median abs. deviation of residuals.
	 `tune` = `4.685`
	
	 Further, `resid` is the vector of residuals from the previous iteration, `h` is the vector of leverage values from a least-squares fit, and `s` is an estimate of the standard deviation of the error term and finally, `w` is the weighting vector so that if W is the corresponding diagonal matrix then at each iteration $\beta = (\mathbf{X}^t W \mathbf{X})^{-1} \mathbf{X}^t W \vec{p}_{train}$ where $\mathbf{X}$ includes a column of 1's for the intercept. The default tunings are specific to MATLAB but the general formulation of this algorithm is covered in Holland and Welsch [1977].

quadratic-ols	Quadratic regression (including linear effects, quadratic effects, interaction effects between non-encoded variables and intercept) for a total of 206 terms with least squares loss function:  These types of models are discussed in Chapter 7 of Montgomery et al. [2013].

Model	Details
quadratic-bisquare	Same as 'quadratic-ols' but with bisquare weighting function as described for 'linear-bisquare'
lasso	Ordinary least squares linear regression with L_1 regularization, model parameters solved with stochastic gradient descent, and regularization hyperparameter, $\lambda \in (5.784e-10, 5.784)$, optimized with Bayes Optimizer using 3-fold cross validation and 27 allotted objective function evaluations. See Tibshirani [1996] for details on lasso regression.
ridge	Same settings as 'lasso' except with L_2 regularization. See Marquardt and Snee [1975] for more details about the formulation of the ridge regression.
regressiontree	Partition of the feature space through a binary tree structure which maximizes the information of each partition, and determines output as a piecewise constant function (average *price* values) for all houses within that partition: $\hat{p}(x) = \frac{1}{\lvert Partition(x)\rvert} \sum_{c \in Partition(x)} p_c$ Trained using CART algorithm for determining the best split predictor at each node and maximum number of splits (randomly searched between 1 and 17,290) and minimum leaf size (randomly searched between 1 and 8,645) optimized with Bayes Optimizer over 3-fold Cross Validation alotted 27 objective function evaluations. See Breiman et al. [1984], Loh [2002], or more recently Chapter 15 of Montgomery et al. [2013] for more details.
svr-linear	Linear combination of features with intercept term and using ϵ-insensitive loss function and structural risk minimization (akin to regularization). Tuned using Bayes Optimizer with an allotment of 27 objective function evaluations on 3-fold Cross Validation over the parameter space of $\epsilon \in (\frac{1}{100}, 2000)$ and $C \in (10, 5e+6)$. See Vapnik [1998] for details of the SVR formulation and Platt [1998] for a description of the SMO solver.
svr-polynomial-order2	Combination of features using polynomial kernel with intercept term, ϵ-insensitive loss function and structural risk minimization, with function form: $$\hat{p}(x) = \beta_0 + \sum_{i=1}^{N} \beta_i (x_i \cdot x + 1)^2$$ where x_i refers to the i-th data point. (Same settings as 'svr-linear')
svr-gaussian	Combination of features using Gaussian kernel with intercept term and using ϵ-insensitive loss function and structural risk minimization with functional form: $$\hat{p}(x) = \beta_0 + \sum_{i=1}^{N} \beta_i \exp\{-\gamma \lVert x_i - x \rVert^2\}$$ (Same settings as 'svr-linear' in addition to kernel scale settings, $\gamma \in [1, 5e+6]$)

Table 4: Definitions and settings for each model.

Accuracy

We use several, common accuracy measures since they each tell us something different about goodness of fit and allow for comparison with the reported results of others. Table 5 introduces each these measures. In the following notation, $p(x_i)$ refers to the true price for the i-th house; $\hat{p}(x_i)$ refers to the estimated price for the i-th house; p_{avg} refers to the average true price within the given split of the data set (training or testing); N refers to the number of houses in a given split; n refers to the sample size (size of the training set); k refers to the number of features.

The RMSE can be understood as an unsigned average error, but this value is susceptible to outliers. This motivates the use of some median based measures such as MAE and MAPE. However, the MAE has the disadvantage that the reported error is not placed into relation with the true house price. Since a slightly larger error is expected as the house price increases, the MAPE is typically a more useful, relative measure. Recall also that this is the main value reported by Zillow.com. Next, the adjusted R^2 is also used, as is done in many Kaggle kernels. The adjustment to the regular R^2 is required since, as is well known, this measure is biased towards overestimation when the number of features increases. While the adjustment allows for negative values, the R^2 measure has the advantage of interpretability, roughly on a scale from 0 to 1, as a measure of how well the variability of the dependent variable has been accounted for in terms of the independent variable, and is also a relative measure as the sum of the squared errors is scaled by the sum of the squared deviation of prices from their mean. These advantages make R^2 a promising accuracy measure for this data, because we are interested in the overall performance, including at the extrema of the price range, and the $[0, 1]$ scaling abstracts away the initial scaling in terms of price units to a scaling more suitable to interpretation of accuracy. In other words, R^2 combines the best features of RMSE and MAPE for our purposes. Finally, to evaluate how well models generalize, we take these measurements on both the training and the testing set.

Accuracy Measure	**Formula**
Root Mean Squared Error (RMSE)	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(p(x_i)-\hat{p}(x_i))^2}$
Median Abs. Error (MAE)	$\text{median}(\ \{\lvert p(x_i)-\hat{p}(x_i)\rvert\}_{i=1}^{N})$
Median Abs. Percentage Error (MAPE)	$\text{median}(\ \{\lvert\frac{p(x_i)-\hat{p}(x_i)}{p(x_i)}\rvert\}_{i=1}^{N})\ \times 100\%$
Adjusted R^2	$1-(1-R^2)\frac{n-1}{n-k-1}$ where $R^2 = 1-\frac{\sum_{i=1}^{N}(p(x_i)-\hat{p}(x_i))^2}{\sum_{i=1}^{N}(p(x_i)-p_{avg})^2}$

Table 5: Formulas for accuracy measures used

4 Results and Discussion

There are several takeaways that can be made from the results. We organize these results in the order which they appear in the ML pipeline, first with the effect of pre-processing, then the aspects of models that contributed to higher performance, and finally, we compare the rankings of MAPE vs. adj. R^2. Given the 3 pre-processings and 10 models, there was a total of 30 pre-processing-model combinations. Training for each was timed and these accuracy measures were computed on both training and testing sets.[5] Of these 30 combinations, 6 lacked convergence including all 3 'lasso' cases, and the versions of 'ridge', 'svm-linear' and 'svm-polynomial-order2' which used unprocessed data. This lack of convergence is evidenced by their longer training time and exceptionally low accuracy. All analyses were performed using the available toolboxes and scripts written by the author in MATLAB® 2017a on a personal computer. A complete tabulation of the numerical results can be found in Table 6 in the appendix.

Feature Pre-processing

We found that feature pre-processing improved model performance. More precisely, an improvement in performance was registered in all cases with the only exceptions being 'linear-ols' and 'linear-bisquare', which were unaffected (with respect to both MAPE and adj. R^2). Although both 'lasso' and 'ridge' suffered from lack of convergence, it is noteworthy that in the one case when 'ridge' successfully converged and achieved comparable performance to 'linear-ols', it was on feature scaled data, as seen in Figure 2 on Page 9. While the quadratic and tree regression models only saw modest improvements due to pre-processing, these pre-processings proved very important for SVR. The Support Vector Regressions exhibited poor performance on unprocessed data and, in particular, failed to converge in the case

[5]The value of n in the adj. R^2 is kept constant when calculating accuracy across training and testing.

of ‘svr-linear’ and ‘svr-polynomial-order2’ on unprocessed data. While standardization and feature scaling were both reliable, the feature scaling provided a minor, albeit practically insubstantial, advantage for the ‘svr-gaussian’ and ‘svr-polynomial-order2’, and standardization, a minor advantage for ‘regressiontree’ with respect to their MAPE. Since these pre-processings have the same time complexity, $O(Nk)$, and provide practically equivalent improvements to model performance, they are equally suitable for use in normalizing the features of a house for value estimation.

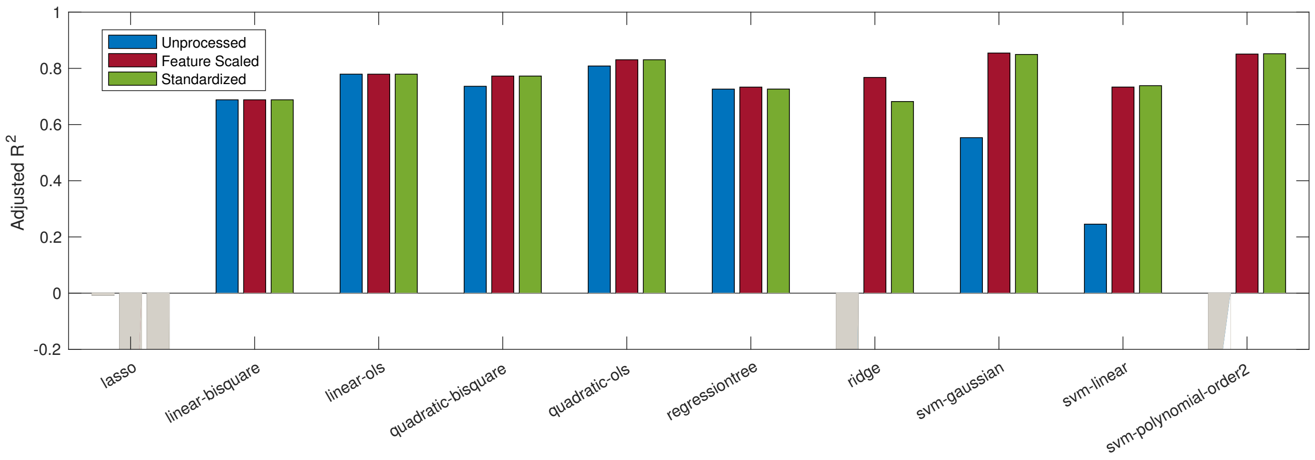


Figure 2: Bar chart of adj. R^2 for each Model-Pre-processing Combination (Testing Set)

The need for pre-processing comes from a variety of factors. One possible explanation may be the numerical conditioning of the problem. For example, in the OLS and bisquare quadratic regression models, we expect that the performance of an exact solution should not depend on the relative scaling of features since there is always a suitable scaling of the parameters to adjust for differently weighted inputs. However, the empirical results for ‘quadratic-ols’ show slight improvements due to the pre-processing (testing adj. R^2 of 0.81 for unprocessed data and 0.83 for both feature scaled and standardized data), i.e. the weightings did not adjust in the expected way. It may be the case that when the data is unprocessed, there are relatively higher rounding errors or similar effects acting as a bottleneck to optimal model performance.

The second need arises in the cases where the model includes regularization. Regularization assumes a prior on the model. In the case of Ridge Regression’s L_2 regularization, there is an assumption of a Gaussian prior with equal variance across the features. Other forms of regularization make similar assumptions regarding the variance. While ‘lasso’ and ‘ridge’ had problems with convergence more generally (possibly related to the stochastic gradient descent solver), there was a unique lack of convergence, in the cases of ‘svr-linear’, and ‘svr-polynomial-order2,’ whenever this assumption was violated, as in when the features were unprocessed.

Finally, SVRs are a special case where the construction of the kernel matrix depends on computing pairwise distances between points and thereby requiring one to find the sum of squares of the difference between those points. When the data are not pre-processed, features which naturally take on a higher order of magnitude such as the area of the lot, living space or basement will dominate lower order features, such as the number of bedrooms or bathrooms, in the Gaussian kernel. The same may be said of the polynomial kernel which is based on the dot product—changes in lower order features do not register as strongly as those in the higher order ones. This is problematic since the human readable units on higher magnitudes do not necessarily translate into greater importance for the model. One way to ameliorate the problem could be to use a more sophisticated kernel which while resembling the standard Gaussian kernel or polynomial kernel nonetheless has an individual “bandwidth” parameter for each feature, but since there are over 80 features and we assume no *a priori* domain knowledge of the real estate market, one would have to resort to numerically optimizing these prohibitively expensive hyperparameters. Hence, the approach of bringing features on to the same scale is most practical.

Models

The best nonlinear models outperformed the best linear models. Linear least squares models had about 15.5% MAPE while the replacement of a robust loss (bisquare re-weighting or ϵ-insensitive) further reduced the MAPE to about 12.5%. However, this robustness corresponded to a trade-off on overall test set performance since ‘linear-ols’ achieved an adj. R^2 of 0.78 (the same as ‘regressiontree’) while ‘linear-bisquare’, only 0.69. These values applied across all forms of pre-processing. From here nonlinear models such as quadratic OLS regression, quadratic kernel SVR and Gaussian

SVR were able to approach an adj. R^2 above 0.80 on pre-processed data (both feature scaled and standardized). Due to its faster training time compared to 'svr-polynomial-order2' and high accuracy, 'svr-gaussian' is the best of the three.

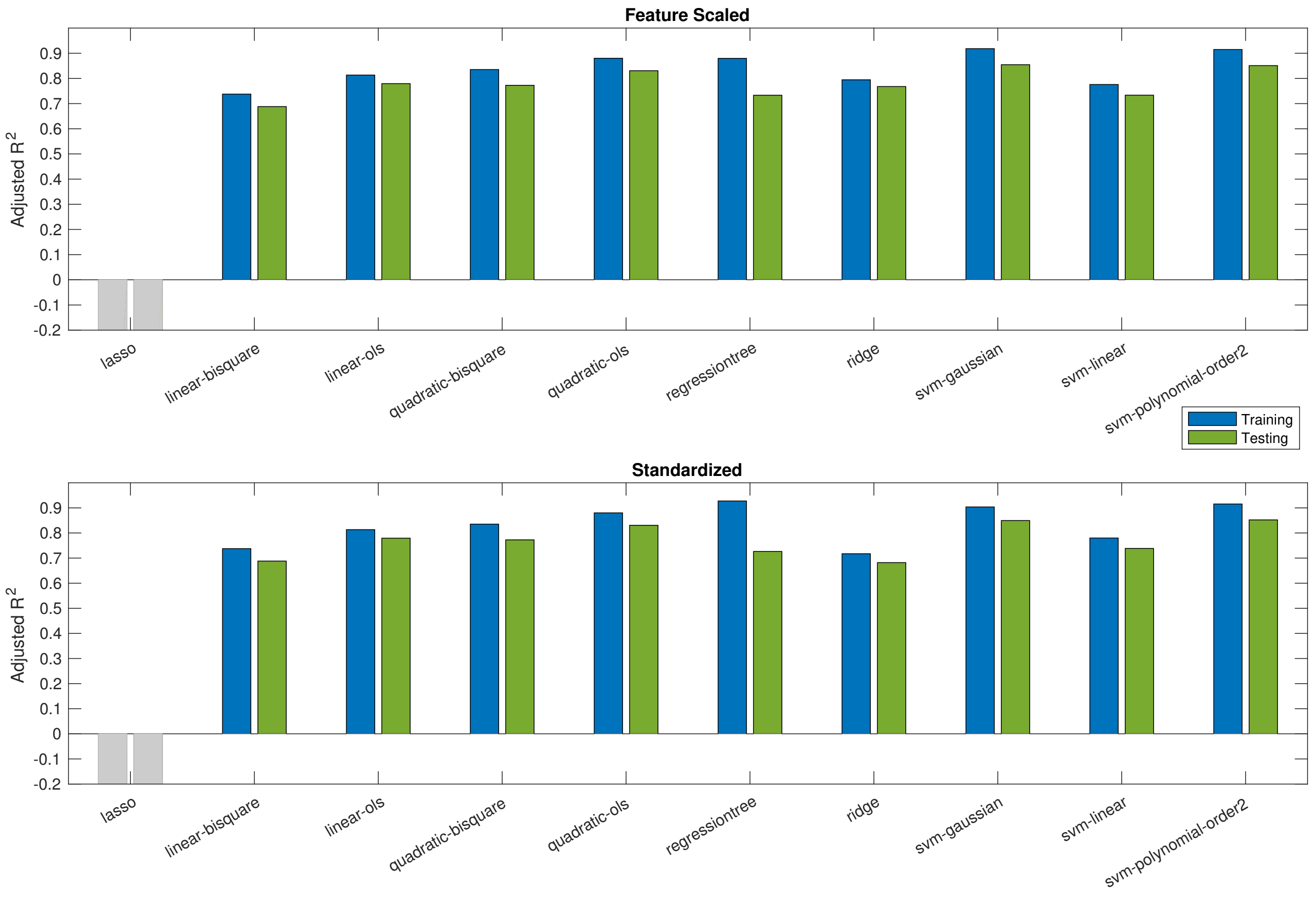


Figure 3: Bar chart of adjusted R^2 for the Training vs. Testing

All models saw a reduction in accuracy in generalizing to the testing set, but the effect was most pronounced for the regression trees. In Figure 3, we compare the training set and test set performance of each model. Even with hyperparameter optimization, the regression tree appeared to overfit, since on standardized data, it managed to achieve a training set MAPE of about 8%. Yet, this translated to about 15.5% on the testing set, and as shown, the adj. R^2 dropped from 0.93 down to 0.73. The 'quadratic-ols', and 'quadratic-bisquare' generalized relatively well. This may be attributable to our design of using fewer terms, relative to the combinatorial blow-up in parameters via inclusion of all the possible quadratic/interaction terms. This generalization may be contrasted with the quadratic kernel SVR which naturally involves more complicated variable interactions and saw a wider drop on feature scaled data of about 3% on MAPE and a 0.06 drop in adj. R^2. Similar remarks may be said of the Gaussian SVR on pre-processed data.

To compare the 'svr-gaussian' and 'quadratic-ols', Gaussian SVRs took 3 orders of magnitude longer to train while having comparable testing adj. R^2 to 'quadratic-ols' (0.85 vs. 0.83). However, once the costly, optimal hyperparameters have been found once, future retraining becomes simpler. One approach to narrowing the gap may be to estimate optimal hyperparameters on a tiny sampling of the training data, bringing the training time down to a more reasonable level. Another major advantage of 'quadratic-ols' is that with the relatively small set of estimated parameters and functional form, the parameters are easier to interpret compared to the Gaussian SVR which used a total of 17,242 support vectors (from the training set size of 17,289). As such, when timing or interpretability are paramount, the combination of high adj. R^2, fast training time and human-interpretable parameters confirm the attractiveness of quadratic OLS as a candidate model in hedonic pricing. However, if accuracy of prediction is the only concern, then the Gaussian SVR's more opaque functional form should not inhibit its use since, in principle, better hyperparameters could be found to improve our 'svr-gaussian' even more.

Acccuracy Measures

Following the discussion of accuracy measures in the section on methodology, MAPE and adj. R^2 were the most useful for interpreting the results due to their properties. However, in the empirical results, these two measures did not agree on the ranking of models. While it is clear that the Gaussian SVR outperformed all other models with respect to each of the given measures, we narrow the scope here to a discussion of the two nonlinear models which demonstrate fast training time along with goodness of fit, 'quadratic-ols' and 'quadratic-bisquare', but for which there is a dispute about superiority of performance. The robust bisquare model had a lower error in terms of the MAPE (11.70 vs. 12.54), while the OLS model had a superior adj. R^2 (0.83 vs. 0.77). These values applied for both feature scaling and standardization. In Figure 4, we compare the two models by plotting the predicted house prices against the true price to visually inspect the spread of results from the line of perfect fit where $\hat{p}(x) = p(x)$. These plots indicate that the OLS model had a slightly tighter fit for the entire test set as predicted by the adj. R^2 which is a function of the mean squared error. There is further indication that, for the robust model, the magnitude of errors grew larger as the true price increased.

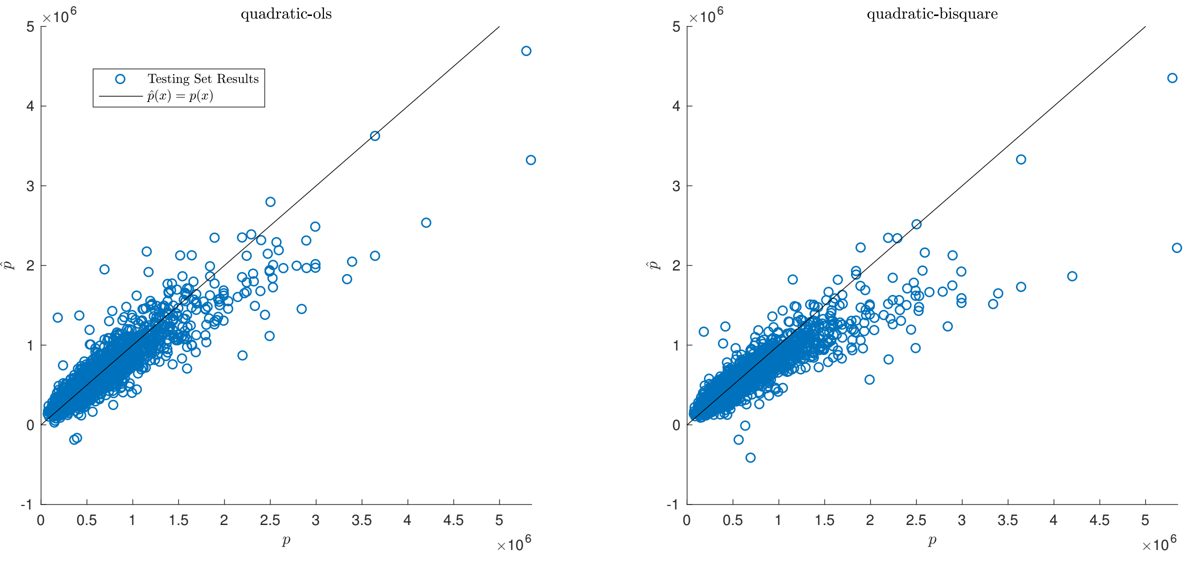


Figure 4: Comparison of quadratic regression models performance on the feature scaled testing set

For 'quadratic-bisquare', additional analyses showed that while the average house price for the testing set was about \$550,000, the average true house price for the worst 50% predictions (absolute errors above the median) was about \$650,000. This shows that the higher errors were for more expensive houses. On the other hand, 'quadratic-ols' had an average of \$630,000 on the worst 50% of predictions, which means that it performed slightly better than its robust counterpart on the higher end houses.

In the value estimation problem, we would ideally like to consider the entire distribution of errors. The analysis of the quadratic regression models shows empirically that accuracy measures based on the MSE such as RMSE and adj. R^2 would be the most appropriate measure by which to rank the models. As such, 'quadratic-ols' should be considered the better of the two models for our purposes.

When reporting the accuracy of a model, accessibility for a wide audience is important. However, the measures based on MSE may not satisfy this requirement. For example, Zillow reports the MAPE in addition to values such as the percentage of predictions falling within 5%, 10%, and 20% of the true price. These values are intuitive to understand for a home buyer who may not have domain expertise about housing value estimation. Therefore, it may be appropriate to select models based on the MSE, while reporting accuracy of prediction in percentiles.

5 Conclusion

The flexibility of machine learning models makes them applicable to a variety of use cases. Our findings indicated that for the housing value estimation problem, Gaussian SVR achieved top performance among candidate models when the King County, Washington housing sales data were suitably pre-processed. However, the results also suggested that classical parametric models such as the quadratic OLS regression can achieve out-of-sample generalization nearing that of more sophisticated machine learning models, with faster training times, insensitivity to the relative scaling of features and more easily interpreted parameters—confirming their attractiveness for use in hedonic pricing. Nonetheless, pre-processing of features, both feature scaling on to the range [0,1] and standardization, was helpful for numerical conditioning, hyperparameter optimization, and the use of kernel methods, across candidate models. The best nonlinear models outperformed the best linear models, in general, and robustness in the loss function or regularization on model parameters were not absolutely necessary for their accuracy nor out-of-sample generalization.

Limitations

The experiments relied on readily available data which may not be representative of all housing markets. Future work could employ additional procedures such as bootstrapping that would allow statistically significant comparisons. Nonetheless, the methodology taken should be useful for the task of comparing models on a large data set with comparably small number features.

Acknowledgments

I would like to thank my advisor, Dr. Igor Griva, for his mentorship during my research training in EXTREEMS-GMU 2017 and for his encouragement and guidance throughout the process of preparing this paper. I would also like to extend special thanks to faculty of the GMU Mathematics Department, Ms. Wendy Mann of GMU Data Services and the referees and editor for their constructive and insightful comments. This research was supported under NSF Grant DMS-1407087.

References

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Boca Raton, FL: CRC Press, 1984.

Carlos Del Cacho. A comparison of data mining methods for mass real estate appraisal. MPRA Paper 27378, University Library of Munich, Germany, December 2010. URL `https://ideas.repec.org/p/pra/mprapa/27378.html`.

W. H. DuMouchel and F. L. O'Brien. Integrating a robust option into a multiple regression computing environment. *Computer Science and Statistics: Proceedings of the 21st Symposium on the Interface*, 1989.

V. Gan, V. Agarwal, and B. Kim. Data mining analysis and predictions of real estate prices. *Issues in Information Systems*, 6:30–36, 2015.

P. W. Holland and R. E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics: Theory and Methods*, A6:813–827, 1977.

P. J. Huber. *Robust Statistics.* NJ: John Wiley Sons, Inc., 1981.

W.Y. Loh. Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, Vol. 12: 361–386, 2002.

D. W. Marquardt and R. D. Snee. Ridge regression in practice. *The American Statistician*, 29(1):3–20, 1975.

Douglas Montgomery, Elizabeth Peck, and G. Vining. *Introduction to Linear Regression Analysis.* Wiley, 5th edition, 2013.

L. Nesheim. Hedonic price functions. *Center for Microdata Methods and Practice (CEMMAP), University College London and Institute for Fiscal Studies*, 2006.

T. Oladunni and S. Sharma. Hedonic housing theory - a machine learning investigation. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 522–527, Dec 2016. doi: 10.1109/ICMLA.2016.0092.

John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *ArXiv e-prints*, June 2012.

J. O. Street, R. J. Carroll, and D. Ruppert. A note on computing robust regression estimates via iteratively reweighted least squares. *The American Statistician*, Vol. 42:152–154, 1988.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, (1):267–288, 1996.

V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

A Numerical Results

In Table 6, all the metrics for model performance are reported. Table cells are color coded so that for each column the better models are darker green. The scale ranges from white (meaning the worse model in this category) to the darkest shade of green (meaning the best for that category). There were exceptional cases with lack of convergence (all 3 'lasso'cases, and the unprocessed data versions of 'ridge', 'svm-linear' and 'svm-polynomial-order2') that had metrics which were too high to include in the color coding and so take the default of white.

Pre-processing	Model	Run-time (s)	Train RMSE	Train MAE	Train MAPE	Train Adj. R^2	Test RMSE	Test MAE	Test MAPE	Test Adj. R^2
unprocessed	linear-ols	0.27	1.58e+05	6.42e+04	14.18	0.81	1.72e+05	7.28e+04	15.38	0.78
feature-scaling	linear-ols	0.27	1.58e+05	6.42e+04	14.18	0.81	1.72e+05	7.28e+04	15.38	0.78
standard-scoring	linear-ols	0.27	1.58e+05	6.42e+04	14.18	0.81	1.72e+05	7.28e+04	15.38	0.78
unprocessed	linear-bisquare	2.70	1.87e+05	4.73e+04	11.02	0.74	2.05e+05	5.45e+04	12.42	0.69
feature-scaling	linear-bisquare	2.26	1.87e+05	4.73e+04	11.02	0.74	2.05e+05	5.45e+04	12.42	0.69
standard-scoring	linear-bisquare	2.23	1.87e+05	4.73e+04	11.02	0.74	2.05e+05	5.45e+04	12.42	0.69
unprocessed	lasso	199.05	3.67e+05	1.86e+05	37.54	-0.00	3.68e+05	1.80e+05	35.80	-0.01
feature-scaling	lasso	60.06	6.26e+05	4.44e+05	90.67	-1.92	6.24e+05	4.40e+05	88.02	-1.89
standard-scoring	lasso	34.52	1.33e+06	7.73e+05	171.49	-12.12	1.37e+06	7.98e+05	171.12	-12.97
unprocessed	ridge	109.12	6.90e+05	2.19e+05	46.49	-2.55	8.73e+05	2.13e+05	42.08	-4.65
feature-scaling	ridge	29.36	1.66e+05	5.99e+04	13.36	0.79	1.77e+05	6.56e+04	14.21	0.77
standard-scoring	ridge	29.44	1.95e+05	8.40e+04	18.42	0.72	2.07e+05	9.20e+04	19.63	0.68
unprocessed	regressiontree	66.75	9.71e+04	3.69e+04	7.97	0.93	1.92e+05	6.91e+04	15.31	0.73
feature-scaling	regressiontree	60.73	1.27e+05	5.52e+04	11.80	0.88	1.90e+05	7.75e+04	16.29	0.73
standard-scoring	regressiontree	64.05	9.86e+04	3.80e+04	8.23	0.93	1.92e+05	6.92e+04	15.32	0.73
unprocessed	quadratic-ols	0.93	1.42e+05	5.35e+04	12.01	0.85	1.61e+05	6.16e+04	13.62	0.81
feature-scaling	quadratic-ols	0.91	1.27e+05	4.83e+04	11.01	0.88	1.51e+05	5.57e+04	12.54	0.83
standard-scoring	quadratic-ols	0.99	1.27e+05	4.83e+04	11.01	0.88	1.51e+05	5.57e+04	12.54	0.83
unprocessed	quadratic-bisquare	7.39	1.74e+05	4.66e+04	10.82	0.77	1.88e+05	5.53e+04	12.43	0.74
feature-scaling	quadratic-bisquare	7.13	1.49e+05	4.32e+04	9.99	0.84	1.75e+05	5.17e+04	11.70	0.77
standard-scoring	quadratic-bisquare	7.67	1.49e+05	4.32e+04	9.99	0.84	1.75e+05	5.17e+04	11.70	0.77
unprocessed	svm-linear	31060.02	2.86e+05	1.14e+05	24.26	0.39	3.19e+05	1.18e+05	24.19	0.25
feature-scaling	svm-linear	1980.55	1.73e+05	4.96e+04	11.33	0.78	1.89e+05	5.67e+04	12.59	0.73
standard-scoring	svm-linear	18686.82	1.72e+05	5.05e+04	11.45	0.78	1.88e+05	5.79e+04	12.77	0.74
unprocessed	svm-polynomial-order2	28102.06	3.31e+13	3.21e+13	7.12e+09	-8.18e+15	3.97e+13	3.21e+13	6.81e+09	-1.17e+16
feature-scaling	svm-polynomial-order2	7790.06	1.07e+05	3.48e+04	7.93	0.91	1.42e+05	4.68e+04	10.62	0.85
standard-scoring	svm-polynomial-order2	17280.07	1.06e+05	3.47e+04	7.96	0.92	1.41e+05	4.62e+04	10.56	0.85
unprocessed	svm-gaussian	2328.67	2.00e+05	8.32e+04	18.86	0.70	2.45e+05	1.02e+05	22.03	0.55
feature-scaling	svm-gaussian	2231.27	1.05e+05	3.24e+04	7.42	0.92	1.40e+05	4.61e+04	10.34	0.85
standard-scoring	svm-gaussian	1773.68	1.14e+05	3.46e+04	7.94	0.90	1.42e+05	4.60e+04	10.53	0.85

Table 6: Complete results table for each pre-processing-model combination