Prediction Models and Learning from Data

Data 102: Data, Inference, and Decisions

Sandrine Dudoit

Department of Statistics, UC Berkeley

Fall 2019

Outline

Examples of Prediction Problems

- Many Data Science (DS) problems involve prediction, which entails learning about the conditional distribution of an outcome given covariates.
- Predicting rent from housing listings. Using data from listings scraped from websites such as Craiglist, predict rent based on rental features such as square footage, number of bedrooms, number of bathrooms, latitude, and longitude.
- Handwriting recognition. Predict a digit (0 through 9) based on pixel values from images of handwritten digits. E.g. MNIST (Modified National Institute of Standards and Technology) database of handwritten digits (`http://yann.lecun.com/exdb/mnist/`).

Examples of Prediction Problems

- Patient diagnosis and prognosis based on genomic data.
  Predict patient cancer type, response to treatment, or survival based on genome-wide expression measures from high-throughput sequencing assays.

Predicting Rent Using Craigslist Data

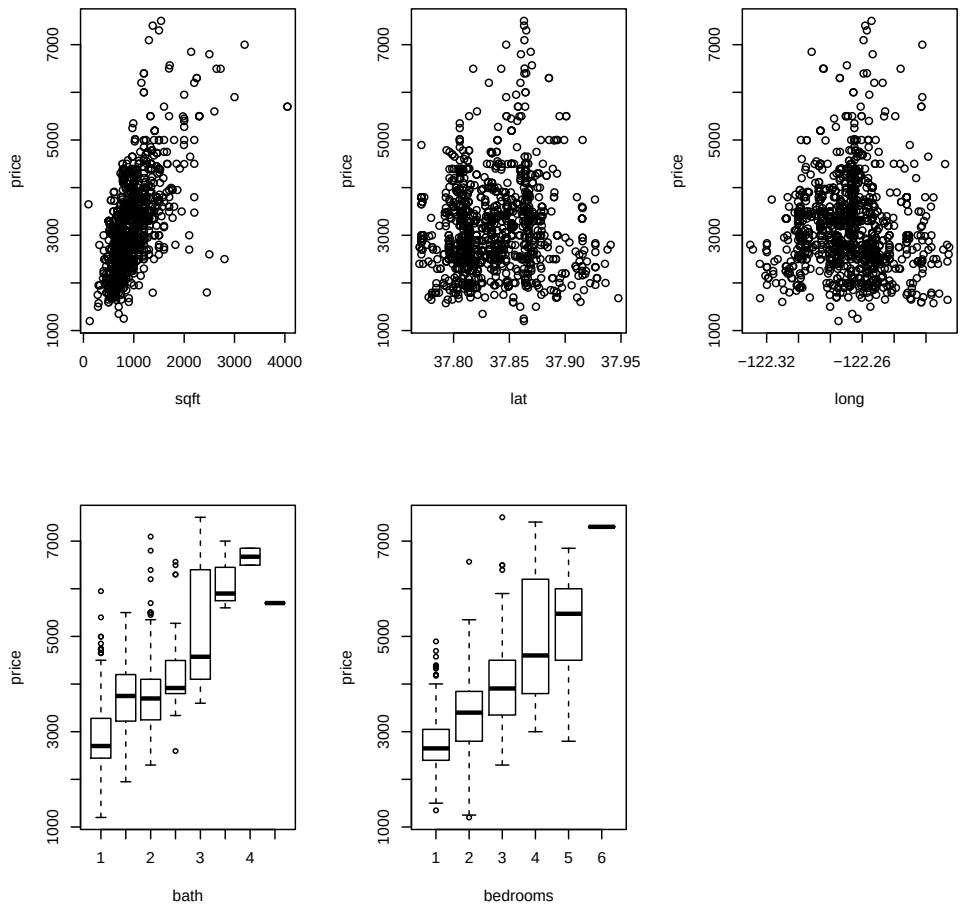


Figure 1: *Craigslist.* Plots of rent vs. five covariates ($n = 1271$).

MNIST Handwritten Digit Recognition

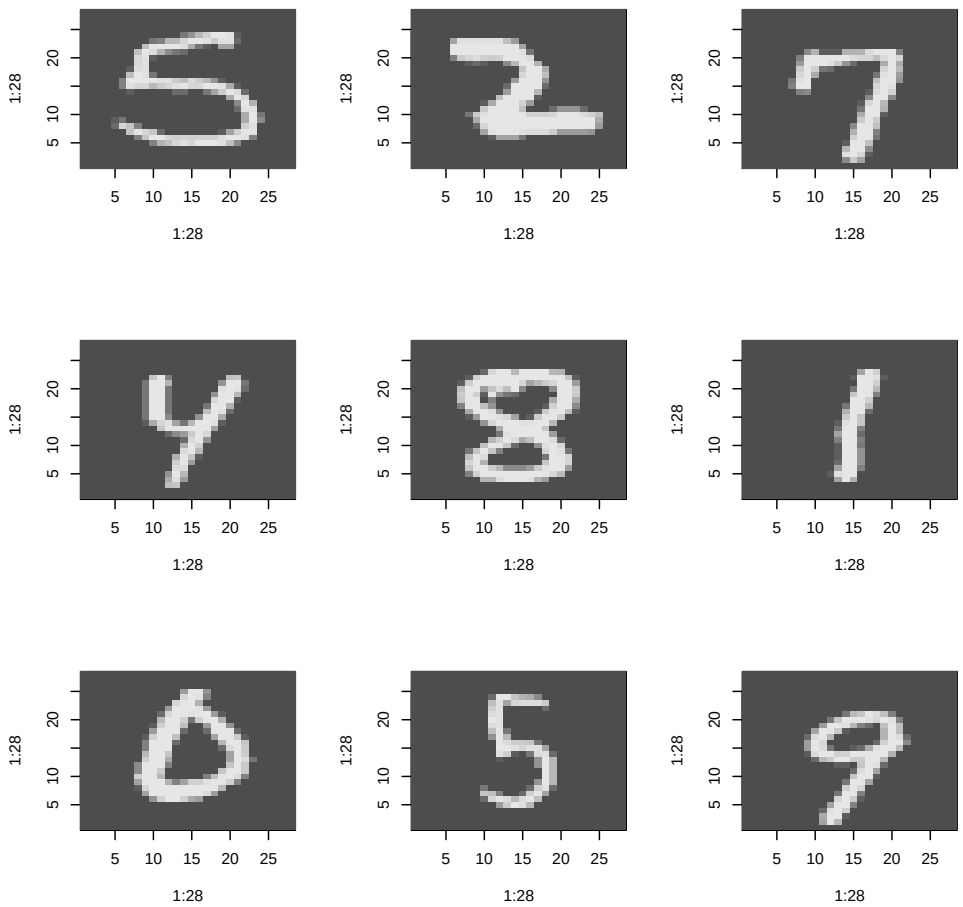


Figure 2: *MNIST digits.* Random sample of 9 images from the MNIST learning set, 28×28 pixels, $[0, 2^8 - 1]$.

Prediction: Classification and Regression

- The above examples all involve predicting an outcome/response Y given a possibly high-dimensional vector of covariates/features/explanatory variables X, i.e., finding a function of X, $\theta(X)$, that will be "close" to the actual values of Y.

$$\begin{aligned} \widehat{\text{price}} &= \theta(\text{sqft}, \text{lat}, \text{long}, \text{bath}, \text{bedrooms}) \\ \widehat{\text{digit}} &= \theta(\text{px}_1, \text{px}_2, \ldots, \text{px}_{784}). \end{aligned}$$

- The outcome for Craigslist is quantitative (i.e., rent), while that for MNIST is qualitative (i.e., one of ten labels corresponding to the digits 0 through 9).

Prediction: Classification and Regression

- The terms classification and regression are often used to refer to the prediction of qualitative and quantitative outcomes, respectively.
- Although different types of predictors are typically used in classification and regression, there are commonalities between the two problems.
- Classification and regression can be handled within the general unified framework of risk optimization, with different loss functions for the different types of outcomes.
- A loss function measures how "close" the predicted values $\hat{Y} = \theta(X)$ are to the actual values Y.
  - Squared or absolute error loss function in regression: $(\hat{Y} - Y)^2$ or $|\hat{Y} - Y|$.
  - Indicator/zero-one loss in classification: $\mathrm{I}(\hat{Y} \neq Y)$.

Prediction: Classification and Regression

- Loss functions can be used for the selection of an optimal predictor as well as for performance assessment of the resulting predictor.

Regression

- In the context of regression, the data structure is (X, Y), where $X \in \mathbb{R}^J$ is a J-dimensional column vector of covariates and $Y \in \mathbb{R}$ a quantitative outcome.
- An intuitive choice for the prediction function or regression function is the conditional expected value of the outcome given the covariates

$$\theta(X) = \mathrm{E}_P[Y|X], \tag{1}$$

where P is the typically unknown data generating (population) distribution for (X, Y).

- The covariates can be either qualitative or quantitative, but often need to be transformed or imputed prior to fitting the regression function, e.g., dummy variables/one-hot encoding for categorical covariates.

Regression

- Note that, in principle, the regression function could be any function from $\mathbb{R}^J$ to $\mathbb{R}$, e.g., it need not be a linear function of the covariates X.
- The parameter space Θ is the set of all possible regression functions $\theta : \mathbb{R}^J \to \mathbb{R}$.
- Subsets $\tilde{\Theta}$ of the parameter space Θ correspond to models for the regression function, i.e., sets of distributions.
- Models involve assumptions about the data generating distribution P.
- A natural loss function for regression is the squared error or L_2 loss function

$$L((X, Y), \theta) = (Y - \theta(X))^2. \tag{2}$$

Regression

- The regression function (an unknown parameter) minimizes risk, i.e., mean squared error (MSE), computed with respect to the unknown distribution P,

$$\theta(X) = \mathrm{E}_P[Y|X] = \mathrm{argmin}_{\theta' \in \Theta} \mathrm{E}_P[(Y - \theta'(X))^2]. \quad (3)$$

- In practice, the population distribution P is unknown and one needs to estimate the regression function θ based on a learning set, $\mathcal{L}_n = \{(X_i, Y_i) : i = 1, \ldots, n\}$.

Regression

We are faced with two main questions.

- What is a suitable model for the regression function? E.g. What type of function $\theta : \mathbb{R}^J \to \mathbb{R}$ would you envisage using for each of the examples above?
- How can we use the data $\mathcal{L}_n$ to learn a "good" estimator of θ?

Regression Models

- Linear regression model.

$$\theta(X) = X^\top \beta = \sum_{j=1}^{J} X_j \beta_j, \tag{4}$$

where β_j are the regression coefficients (parameters).

- Do we use all X_j's?
- Do we include powers of X_j?
- Do we consider interactions, i.e., include products $X_j X_{j'}$?
- How do we estimate β_j given data $\mathcal{L}_n$?

Regression Models

- Regression tree model.

$$\theta(X) = \sum_{k=1}^{K} \beta_k \, \mathsf{I}(X \in \mathcal{A}_k), \tag{5}$$

where the sets $\mathcal{A}_k$ form a partition of the covariate space $\mathcal{X}$, i.e., $\cup_k \mathcal{A}_k = \mathcal{X}$ and $\mathcal{A}_k \cap \mathcal{A}_{k'} = \emptyset$.

  - What types of partitions $\mathcal{A}_k$ should we consider, e.g., linear boundaries?
  - How fine a partition?
  - How do we estimate $\mathcal{A}_k$ and β_k given data $\mathcal{L}_n$?

- What are other types of regression functions do you know? What are issues when learning these functions from data?

Regression Models

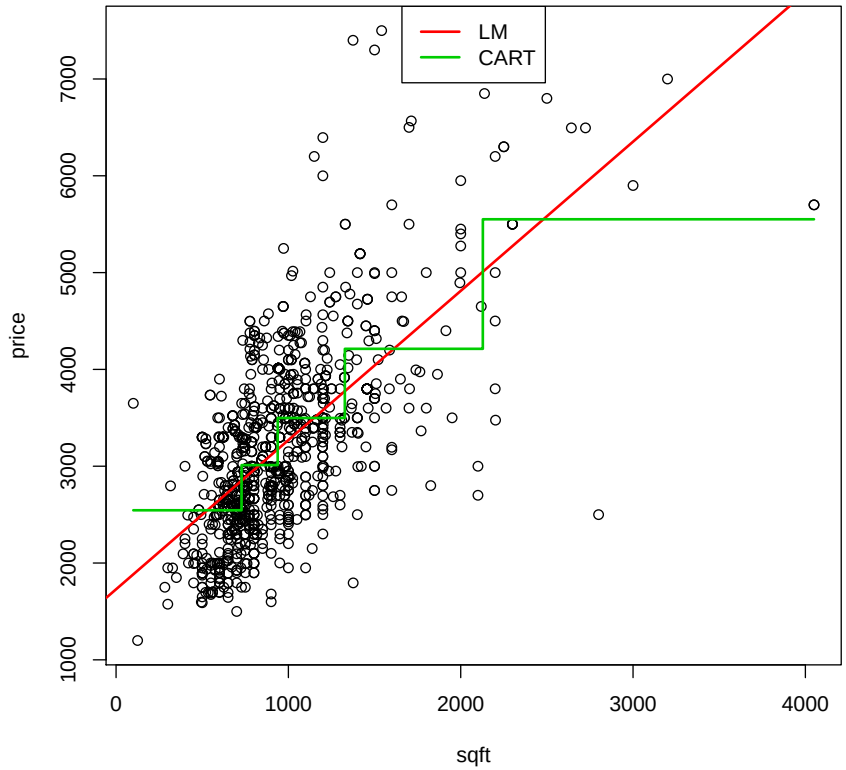


Figure 3: *Craigslist: Linear and tree-based regression.* Regression function of rent on "sqft", linear regression (red) and regression tree (green).

Classification

- In classification, the outcome Y is qualitative, i.e., takes on values arbitrary labeled as $\{1, \ldots, K\}$. The covariates X can be either qualitative or quantitative and may need to be transformed or imputed.
  E.g. Digit in MNIST dataset.
- A classification function or classifier θ generates a partition of the covariate space $\mathcal{X}$ into K disjoint and exhaustive subsets, $\mathcal{C}_1, \ldots, \mathcal{C}_K$, such that for an observation with covariates $X \in \mathcal{C}_k$ the predicted class is k. That is,

$$\theta(X) = \sum_{k=1}^{K} k \, \mathrm{I}(X \in \mathcal{C}_k). \tag{6}$$

Classification

- It is intuitive to predict the outcome based on the conditional class probabilities $\Pr(Y = k|X)$, $k = 1, \ldots, K$.
- A natural loss function in classification is the indicator/zero-one loss

$$L((X, Y), \theta) = \mathrm{I}(Y \neq \theta(X)). \tag{7}$$

- The optimal classifier, i.e., risk minimizer, for the indicator loss function yields the class with maximum posterior probability given the covariates X and is known as the Bayes classifier

$$\theta(X) = \mathrm{argmax}_k \Pr(Y = k|X). \tag{8}$$

Classification

- In practice, however, the class posterior probabilities are unknown and one relies on the learning set to build a classifier $\hat{\theta}$ that is as close as possible to the Bayes classifier in terms of risk.

Classification Models

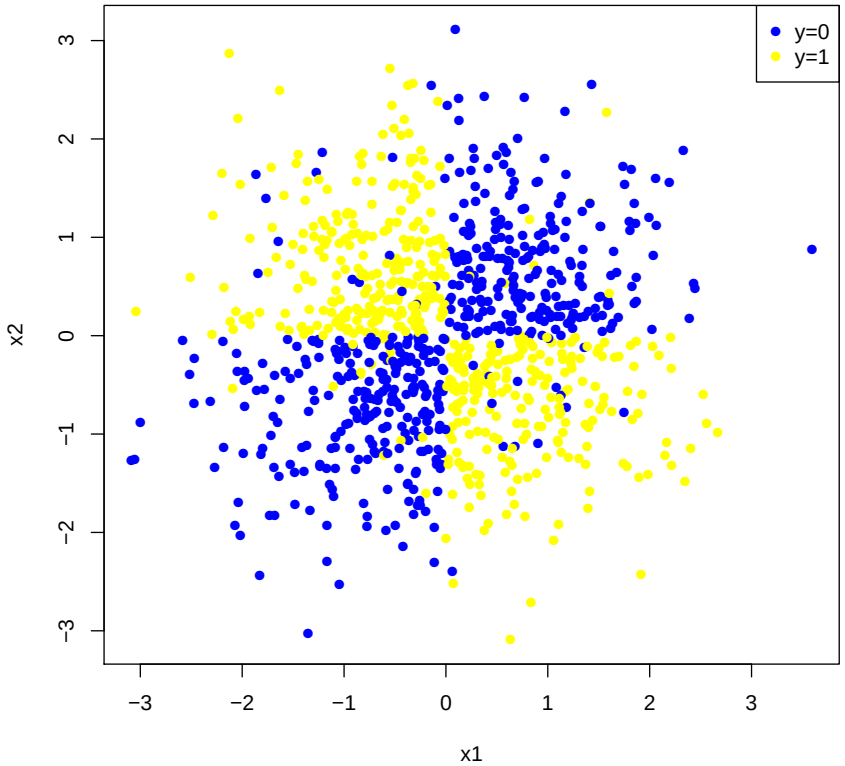


Figure 4: *Classification: Simulated two-class dataset.* $(X_{i,1}, X_{i,2}) \in \mathbb{R}^2$ and $Y_i \in \{0, 1\}$, $i = 1, \ldots, 500$. The class of each observation is indicated by color.

Classification Models

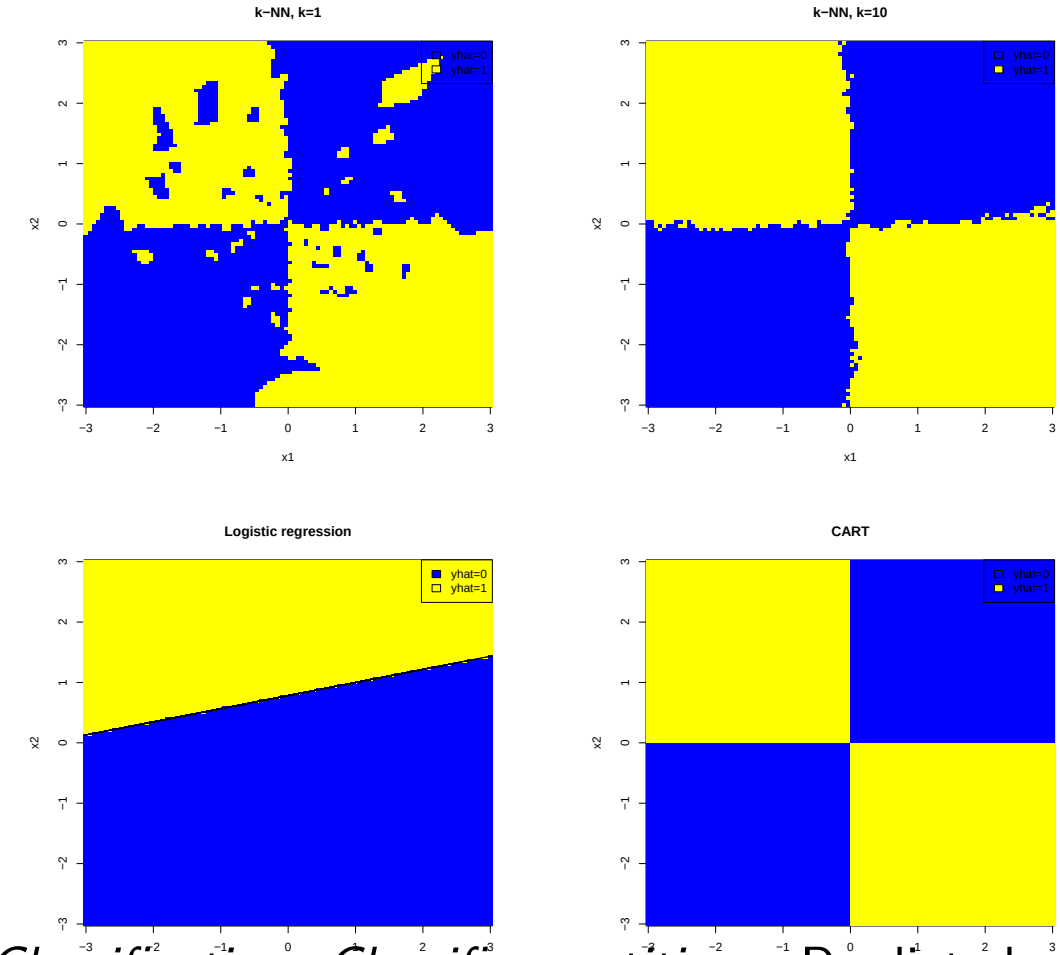


Figure 5: *Classification: Classifier partitions.* Predicted class indicated by color.

How "Parametric" a Model?

- Models such as the linear regression model are typically referred to as parametric models, in the sense that the regression function has a very specific form indexed by a parameter β of regression coefficients.
- By contrast, non-parametric models place few, if any, restrictions on the form of the regression function (e.g., continuity) and let the data determine the function.
  E.g. In robust local regression (loess, lowess), there is no closed-form expression for the regression function, which is obtained by fitting weighted linear regression functions to covariate neighborhoods.
  E.g. Likewise, there is no closed-form expression for k-nearest neighbor classifiers.

How "Parametric" a Model?

- The distinction between parametric and non-parametric inference is blurry. There is no clear dichotomy, but rather a continuum, in the degree of "parametricity" of distributions/models and methods.
- The distinction may have been more relevant historically.
- Idem for the terms "model-based" and "model-free".
- There is always a model. What varies are the characteristics of the model, e.g., its "complexity", its "size", how restrictive it is, its underlying assumptions.

Model Complexity

The complexity of a model or estimator can be measured in various ways.

- The number of covariates for a regression function.
- The polynomial degree for a regression function.
- The number of leaf nodes (i.e., sets in the partition of the covariate space) for a classification or regression tree.
- The span for robust local regression (i.e., loess) and the bandwidth for kernel density estimation, i.e., how "local" a smoother is.
- The penalty parameter for regularized regression, e.g., ridge regression.
- The number of input nodes and layers for a neural network.

Fitting Models to Data

- Given a model, the main task is to estimate or learn the classification/regression function θ from the learning set $\mathcal{L}_n = \{(X_i, Y_i) : i = 1, \ldots, n\}$.
- It is common to estimate θ by minimizing the learning set or empirical risk over the subset $\tilde{\Theta} \subseteq \Theta$ of the parameter space corresponding to the model.
- In the case of regression, on seeks to find the predictor that minimizes MSE on the learning set

$$\begin{aligned} \hat{\theta}(X) &= \text{argmin}_{\theta' \in \tilde{\Theta}} \, \mathrm{E}_{P_n}[(Y - \theta'(X))^2] \qquad (9) \\ &= \text{argmin}_{\theta' \in \tilde{\Theta}} \, \frac{1}{n} \sum_{i=1}^{n} (Y_i - \theta'(X_i))^2, \end{aligned}$$

where P_n is the empirical distribution corresponding to the learning set.

Fitting Models to Data

- What popular methods use this approach?
- Any problems with this approach?

Model Complexity and Bias-Variance Trade-Off

- In some cases, we may choose a model that is too simple to represent the underlying data generation mechanism, i.e., misses the signal in the learning data.
  E.g. Fitting a constant regression function, when there is in fact a non-linear relationship between the outcome and the covariates.
- In others, we may choose a model that is too complex, i.e., fits the noise in the learning data.
  E.g. Fitting a regression function that is a high-degree polynomial of the covariates, when there is in fact a simple linear relationship between the outcome and the covariates.
- These two situations are referred to, respectively, as underfitting and overfitting the learning data.
- The phenomenon of overfitting/underfitting is related to

Model Complexity and Bias-Variance Trade-Off

  - the bias of an estimator, i.e., how close its average is to the parameter of interest, and
  - its variance or precision, i.e., how variable it is around its expected value (not necessarily the parameter, unless the estimator is unbiased).

- Ideally, we'd like to minimize both bias and variance.
- However, this is not possible, as there is a trade-off between bias and variance: Decreasing bias is typically associated with an increase in variance and vice versa.
- In general, the more complex a model, the less biased and more variable an estimator.
- Note also that, in general, variance decreases with increasing sample size, but not bias.
  One can become more and more precise about a completely wrong answer!

Model Complexity and Bias-Variance Trade-Off

- Instead of attempting to simultaneously minimize both bias and variance, one seeks to maximize accuracy/minimize risk, i.e., minimize the average "distance" between an estimator and the parameter of interest.
- Risk for the squared error loss function, i.e., mean squared error (MSE), can be decomposed in terms of bias and variance components. That is, given an estimator $\hat{\theta}$ of a parameter θ,

$$\begin{aligned} \mathrm{MSE}_P[\hat{\theta}, \theta] &= \mathrm{E}_P[(\hat{\theta} - \theta)^2] \\ &= \mathrm{E}_P[(\hat{\theta} - \mathrm{E}_P[\hat{\theta}])^2] + (\mathrm{E}_P[\hat{\theta}] - \theta)^2 \\ &= \mathrm{Var}_P[\hat{\theta}] + (\mathrm{Bias}_P[\hat{\theta}, \theta])^2. \end{aligned} \tag{10}$$

Model Complexity and Bias-Variance Trade-Off

In short,

$$\text{MSE} = \text{Variance} + \text{Bias}^2.$$

Proof.

$$\begin{aligned}
\mathrm{E}_P[(\hat{\theta} - \theta)^2] &= \mathrm{E}_P[(\hat{\theta} - \mathrm{E}_P[\hat{\theta}] + \mathrm{E}_P[\hat{\theta}] - \theta)^2] \\
&= \mathrm{E}_P[(\hat{\theta} - \mathrm{E}_P[\hat{\theta}])^2] + \mathrm{E}_P[(\mathrm{E}_P[\hat{\theta}] - \theta)^2] \\
&\quad +2\,\mathrm{E}_P[(\hat{\theta} - \mathrm{E}_P[\hat{\theta}])(\mathrm{E}_P[\hat{\theta}] - \theta)] \\
&= \mathrm{Var}_P[\hat{\theta}] + (\mathrm{E}_P[\hat{\theta}] - \theta)^2 \\
&\quad +2(\mathrm{E}_P[\hat{\theta}] - \theta)\,\mathrm{E}_P[(\hat{\theta} - \mathrm{E}_P[\hat{\theta}])] \\
&= \mathrm{Var}_P[\hat{\theta}] + (\mathrm{Bias}_P[\hat{\theta}, \theta])^2,
\end{aligned}$$

where the third equality follows by noting that $\mathrm{E}_P[\hat{\theta}] - \theta$ is a constant and the fourth by $\mathrm{E}_P[\hat{\theta} - \mathrm{E}_P[\hat{\theta}]] = \mathrm{E}_P[\hat{\theta}] - \mathrm{E}_P[\hat{\theta}] = 0$. □

Bias, Variance, and Accuracy

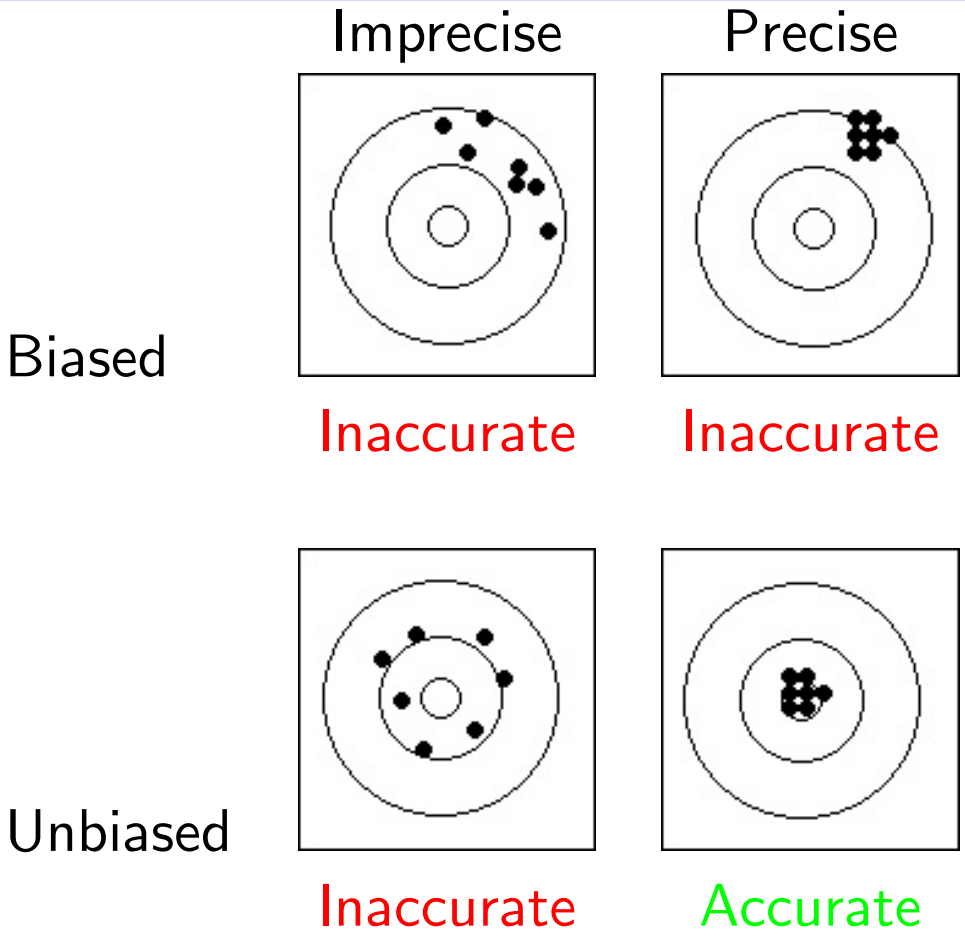


Figure 6: *Bias, variance, and accuracy.*

Model Complexity and Bias-Variance Trade-Off

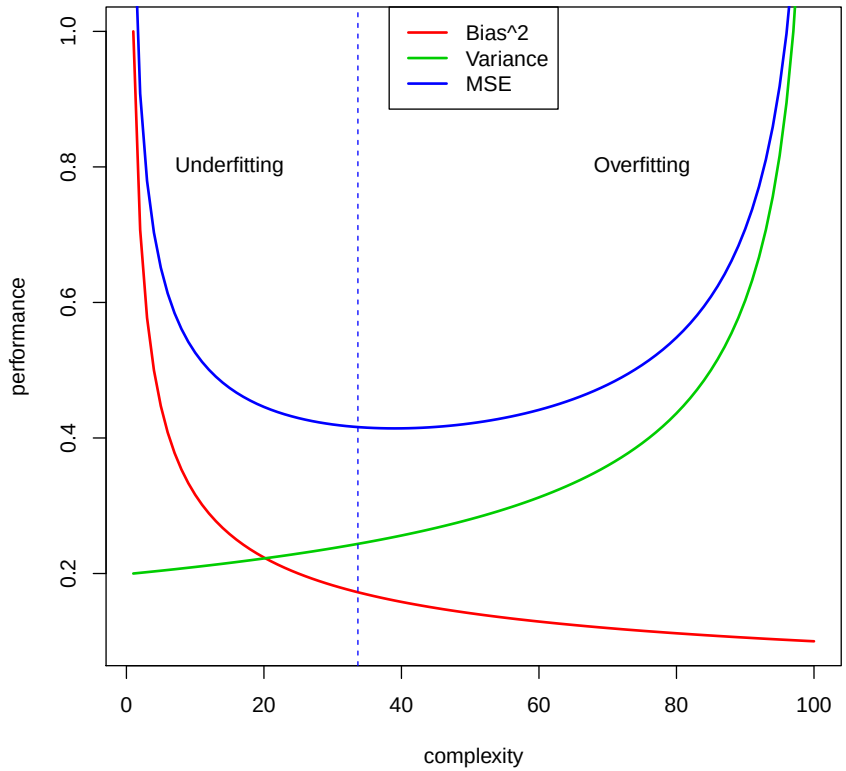


Figure 7: *Bias-variance trade-off.* Schematic representation of bias-variance trade-off as a function of model complexity.

Model Complexity and Bias-Variance Trade-Off

Table 1: *Bias-variance trade-off.* Effect of model complexity and of sample size on bias and variance.

	Bias	Variance
Complexity ↑	↓	↑
Sample size ↑	?	↓

Model Complexity and Bias-Variance Trade-Off

Figure 7 illustrates the bias-variance trade-off as it relates to model complexity. It is an idealized representation of this phenomenon.

- The term "complexity" is vague and needs to be precisely defined. Complexity means different things depending on the type of model/estimator, e.g., polynomial degree for linear regression, smoother span for loess.
- In practice, bias and variance can be on very different scales.
- In practice, the decay/increase of bias/variance with complexity is not always smooth.

Cross-Validation

- Minimizing risk with respect to the learning set can lead to overfitting, especially for large/complex models.
- This translates into poor generalization error, i.e., risk on an independent test set from the same population as the learning set.
- Instead, we can cleverly divide the learning set into data for training estimators and data for validating their performance, i.e., computing risk.
- This is the main idea behind cross-validation (CV):
  - Partition the available learning set into two sets: A training set and a validation set.
  - Observations in the training set are used to compute, or train, estimators.

Cross-Validation

  - Observations in the validation set are used to assess the risk of, or validate, the estimators.
- One of the most common forms of cross-validation is K-fold cross-validation.
  - Randomly partition the learning set into K mutually exclusive and exhaustive sets of approximately equal size.
  - Use each of the K sets in turn as a validation set to assess risk for estimators computed using the union of the remaining $(K-1)$ sets as a training set.
  - The cross-validated risk estimator is the average of the K validation set risks.
  - Smaller values of the number of folds K tend to lead to lower variance (larger validation set), but higher bias (smaller training set) in risk estimation.
  - Common choices for the tuning parameter K are between 5 and 10.

Cross-Validation

- Another common type of cross-validation is Monte-Carlo cross-validation, where the learning set is repeatedly randomly partitioned into a training set comprising $(1 - \kappa)100\%$ of the learning set and a validation set comprising the remaining observations. Common values for κ are between 0.05 and 0.20.
- When using cross-validation for model selection, e.g., selecting the degree of a polynomial or features to include in a regression model, we select the model with lowest cross-validated risk.
- When a test set is available, one can assess the performance of the selected predictor by computing its risk on the test set.

Cross-Validation

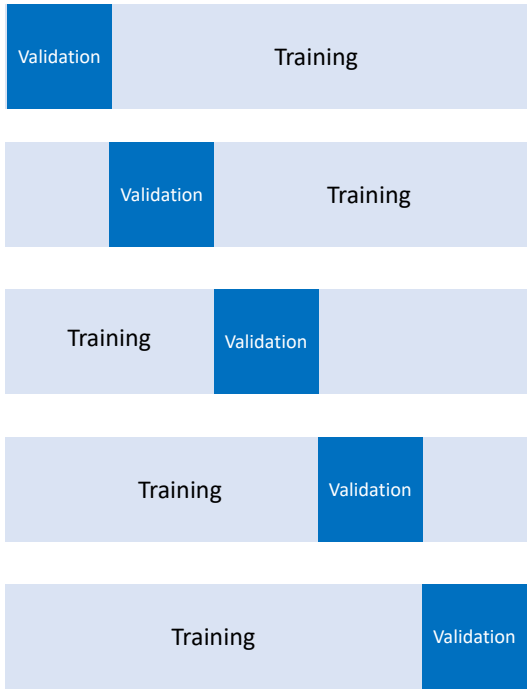


Figure 8: *Five-fold cross-validation.*

Regularization

- Another general approach for model/variable selection and for preventing overfitting is regularization, also known as shrinkage.
- The main idea is to introduce additional modeling assumptions or impose constraints on the estimators, usually through a penalty for complexity in the loss function.
- For linear regression, with the squared error/L_2 loss function, common regularization approaches involve "penalizing" covariates with "large" regression coefficients.
  - Ridge regression: Penalty based on sum of squares (Euclidean/L_2 norm) of regression coefficients.
  - Least absolute shrinkage and selection operator or LASSO: Penalty based on sum of absolute values (L_1 norm) of regression coefficients.

Regularization

- Elastic net: Both L_1 and L_2 penalties.

$$\hat{\beta}^{\text{enet}} = \text{argmin}_{\beta \in \mathbb{R}^J} \sum_{i=1}^{n} \left(Y_i - \sum_{j=1}^{J} \beta_j X_{i,j} \right)^2 + \lambda_1 \sum_{j=1}^{J} |\beta_j| + \lambda_2 \sum_{j=1}^{J} \beta_j^2. \quad (11)$$

- Regularization techniques may themselves require another layer of model selection, corresponding to the tuning of complexity parameters used to penalize the loss function. Cross-validation may be used for this purpose.

Model Trade-Offs

- Slightly different models could lead to different fits on the same data, i.e., results are model-driven vs. data-driven.
- Small perturbation of the data could lead to different fits for the same model, i.e., results are data-driven vs. model-driven.
  E.g. Regression trees are sensitive to perturbations of the data.
- The previous two issues concern robustness/stability to the model and data, respectively.
- Many models can lead to the same fit. For instance, highly-parametric linear regression and lowly-parametric loess can lead to virtually identical fits and prediction accuracies.

Model Trade-Offs

- In addition to prediction accuracy, one should consider computability and interpretability when selecting a prediction model.
  E.g. Loess no closed-form expression for regression function vs. linear regression simple interpretable regression function.
- One should also consider the plausibility of assumptions for the domain context.
- Pre-processing steps (e.g., dimensionality reduction, data transformation/normalization, data imputation) can have a larger impact on the results than the choice of prediction function.
- Breiman (2001). *Statistical modeling: The two cultures*.

Model Trade-Offs

  - "The data modeling culture". Focus on model for data generating mechanism.
  - "The algorithmic modeling culture". Treat data generating model as black box and focus on prediction accuracy.
- Yu and Kumbier (2019). *Three principles of data science: predictability, computability, and stability (PCS)*.
- The goals of the study, i.e., the question, should guide how we negociate the trade-offs related to the choice of a model.

Model Trade-Offs

"All models are wrong, but some are useful." (G.E.P. Box, 1976)

Nearest Neighbor Classifiers

- Nearest neighbor predictors are based on a measure of distance between observations, such as the Euclidean distance or one minus the correlation between two covariate vectors X.
- The k-nearest neighbor rule (k-NN), due to Fix and Hodges (1951), classifies a test case (i.e., a new observation) with covariates X as follows.
  - Find the k observations, or neighbors, in the learning set that are closest to the observation to be classified.
  - Predict the class of the test case by majority vote, i.e., choose the class that is most common among those k neighbors.

Nearest Neighbor Classifiers

- For a large enough number of neighbors k, k-nearest neighbor classifiers suggest simple estimators of the class posterior probabilities: The proportion of votes for each class.
- The vote proportions may also be used to measure confidence for individual predictions.
- The one-nearest neighbor partition (i.e., $k = 1$ case) of the covariate space corresponds to the Dirichlet tessellation of the learning set.

Nearest Neighbor Classifiers: Selecting k

- In general, the number of neighbors k can be chosen by cross-validation (CV).
- For a given partition of the learning set into a training set and a validation set, perform the following steps for a range of values of k.
  - For each observation in the validation set, identify its k nearest neighbors in the training set. Classify this observation by the nearest neighbor rule.
  - Compute the classification error rate for the validation set by comparing the actual classes to the predicted classes.
- The cross-validation error rate (i.e., risk for the indicator loss function) is the average of the validation set error rates.
- Select the value of k which minimizes the cross-validated risk.

Tree-Structured Predictors

- Tree-structured predictors can be used for predicting either qualitative or quantitative outcomes, i.e., for either classification or regression.
- Tree-structured predictors are constructed by repeated splits of subsets of the covariate space $\mathcal{X}$, or nodes, into descendant subsets, starting with $\mathcal{X}$ itself.
- Each terminal node, or leaf, is assigned a fitted value and the resulting partition of $\mathcal{X}$ corresponds to the predictor.

Tree-Structured Predictors

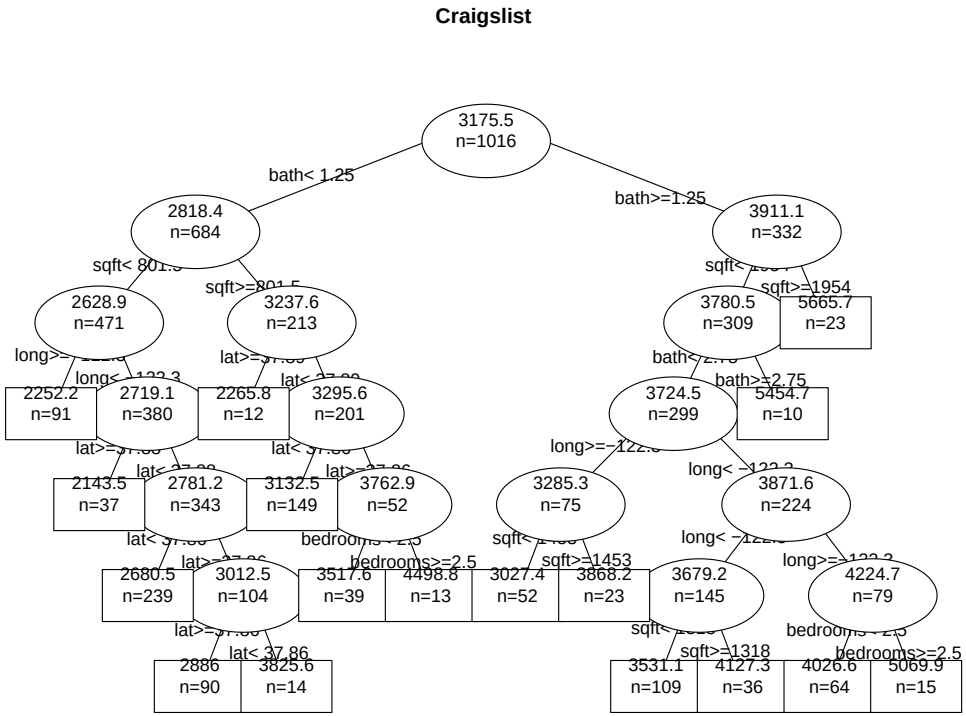


Figure 9: *Craigslist: Regression trees.* Decision tree for regression of rent on all 5 covariates.

Tree-Structured Predictors

- For a tree, the classification/regression function has the form

$$\theta(X) = \sum_{k=1}^{K} \beta_k \, \mathrm{I}(X \in \mathcal{A}_k), \tag{12}$$

  where the sets $\mathcal{A}_k$ form a partition of the covariate space and β_k is the predicted outcome for an observation with covariates in $\mathcal{A}_k$.
- There are three main aspects to tree construction:
  1. the selection of the splits;
  2. the decision to declare a node terminal or to continue splitting;
  3. the assignment of a fitted value for each terminal node.

Tree-Structured Predictors

- Different tree predictors use different approaches to deal with these three issues. Here, we consider classification and regression trees or, in short, CART (Breimn et al., 1984).
- Other tree predictors are C4.5, FACT, and QUEST; an extensive comparison study is found in Lim et al. (2000).

Classification and Regression Trees

1. Node-splitting rule. At each node, choose the split that maximizes the decrease in empirical risk.
   - Classification. Various loss functions, or impurity measures, have been proposed, e.g., Gini index, entropy, and twoing rule.
   - Regression. The most common loss function is the squared error loss function. One could also consider the absolute or Huber loss functions.
2. Split-stopping rule. Obtaining the "right-sized" tree and accurate estimators of risk can be achieved as follows.
   - Grow a large tree, selectively prune the tree upward, getting a decreasing sequence of subtrees.
   - Use cross-validation to identify the subtree having the lowest risk, i.e., classification error (in classification) or mean squared error (in regression).

Classification and Regression Trees

3. Fitted values. For each terminal node, choose the value that minimizes the empirical risk.
   - Classification. The predicted outcome is the most common class in the leaf, cf. majority vote.
   - Regression. The predicted outcome is the average outcome for all the observations in the leaf.

Classification and Regression Trees

- Classification and regression trees have many tuning parameters/inputs, as well as output values in addition to the tree itself and fitted values at the leaves.
- There are also differences in implementation across software packages. Make sure to consult the documentation to understand how the trees are built and how to interpret the results.
- Trees yield a number of useful by-products, including surrogate splits/variables and variable importance measures.
- A surrogate split is a split based on another variable (surrogate) than the primary variable used for splitting a node, but that partitions the data in a "similar" way.

Classification and Regression Trees

- Surrogate variables are helpful for handling missing values, as the surrogate can be used to split a node when an observation has a missing value for the primary variable.
- An overall variable importance measure can be defined based on the decreases in empirical risk for each node for which the variable is used for either a primary or a surrogate split.

Classification and Regression Trees

- Pros.
  - Applicable to both classification and regression.
  - Can handle categorical covariates naturally.
  - Can handle highly non-linear interactions and classification boundaries.
  - Perform automatic variable selection.
  - Can handle missing values through surrogate variables.
  - Easy to interpret if the tree is small. The picture of the tree can give valuable insights into which variables are important and where.
  - Computationally simple and quick to fit, even for large problems.
- Cons.
  - Unstable, i.e., small changes in the learning set can lead to large changes in the tree. This makes interpretation not as straightforward as it first appears.

Classification and Regression Trees

- Often outperformed in terms of accuracy by methods such as support vector machines (SVM) or even classical linear discriminant analysis or k-nearest neighbors.

Ensemble Methods

- A single classification or regression tree can be unstable, i.e., vary greatly with small changes in the learning set.
- Averaging is a natural way to reduce variability.
- This is the main idea behind Random Forests and, more generally, ensemble methods.
- An ensemble predictor can be built by combining the results of
  - the same predictor (e.g., tree) applied to multiple versions of the learning set (e.g., bootstrap samples) or
  - multiple predictors applied to the original learning set.
- In regression, predictions are aggregated by averaging and in classification they are aggregated by voting.

Ensemble Methods

- In bagging (*b*ootstrap *agg*regat*ing*), one aggregates the same predictor built on multiple bootstrap samples of the learning set.
- In boosting, one aggregates the same predictor built on data obtained by repeated adaptive resampling of the learning set, where sampling weights are increased for observations with large prediction errors.

Random Forests

- In Random Forests, one aggregates a forest of many trees, each built on distinct bootstrap samples of the learning set and where subsets of covariates are randomly selected for consideration at each node (`https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`).
- Specifically, for each bootstrap sample of the learning set (typically 500), grow a tree as follows.
  - At each node, select a random subset of J' covariates out of all J covariates and find the best split on these selected variables.
  - Grow the trees to maximum depth.
  - Obtain predicted outcomes by voting/averaging over all trees.

Random Forests

- Random Forests yield a number of useful by-products, including variable importance measures, observation proximity measures, and risk estimates.
- The out-of-bag (OOB) observations, i.e., observations not in a bootstrap sample, can be used to obtain risk estimates: For each bootstrap sample, run OOB observations down the corresponding tree and compute empirical risk for that tree, then average empirical risk over all trees.
- There are two main types of variable importance measures for Random Forests: (1) Based on the decreases in empirical risk for splitting over a variable (aggregated over all internal nodes and trees); (2) based on the differences in risk for out-of-bag observations when permuting the values of the variable (aggregated over all trees).

Random Forests

Predicting Rent Using Craigslist Data

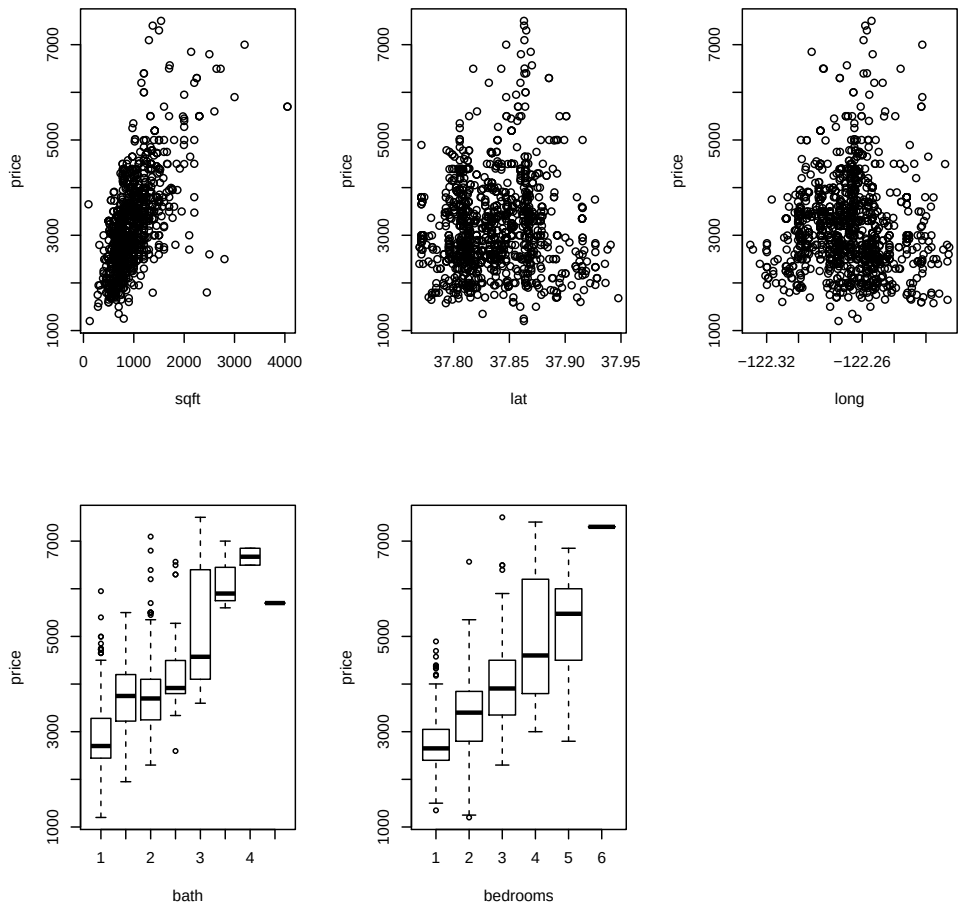


Figure 10: *Craigslist.* Plots of rent vs. five covariates ($n = 1271$).

Craigslist: Regression Trees

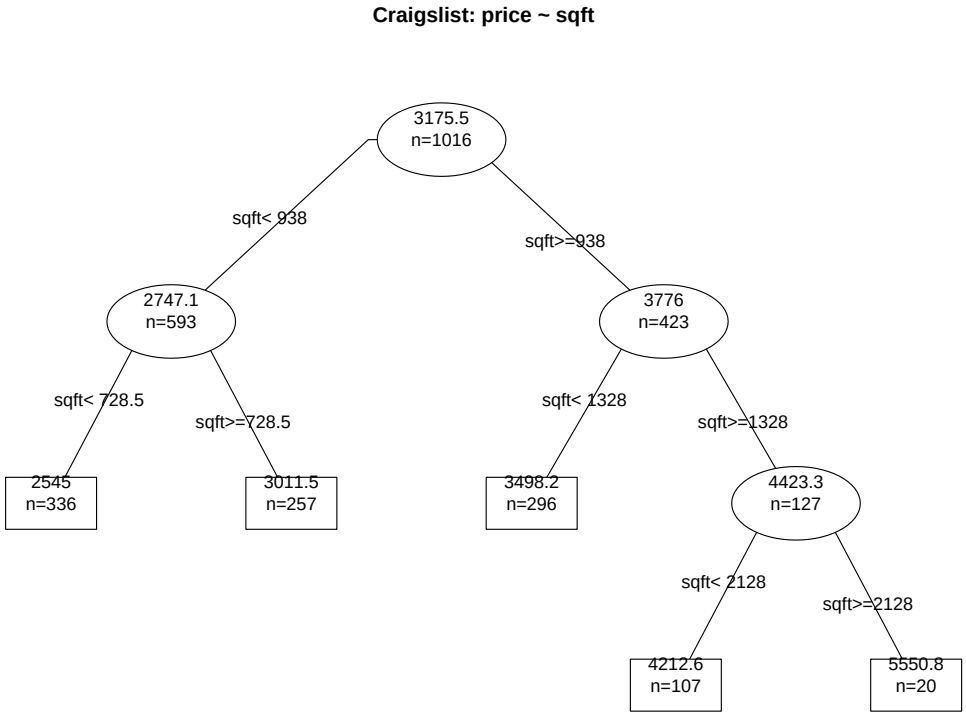


Figure 11: *Craigslist: Regression trees.* Decision tree for regression of rent on "sqft".

Craigslist: Linear and Tree-Based Regression

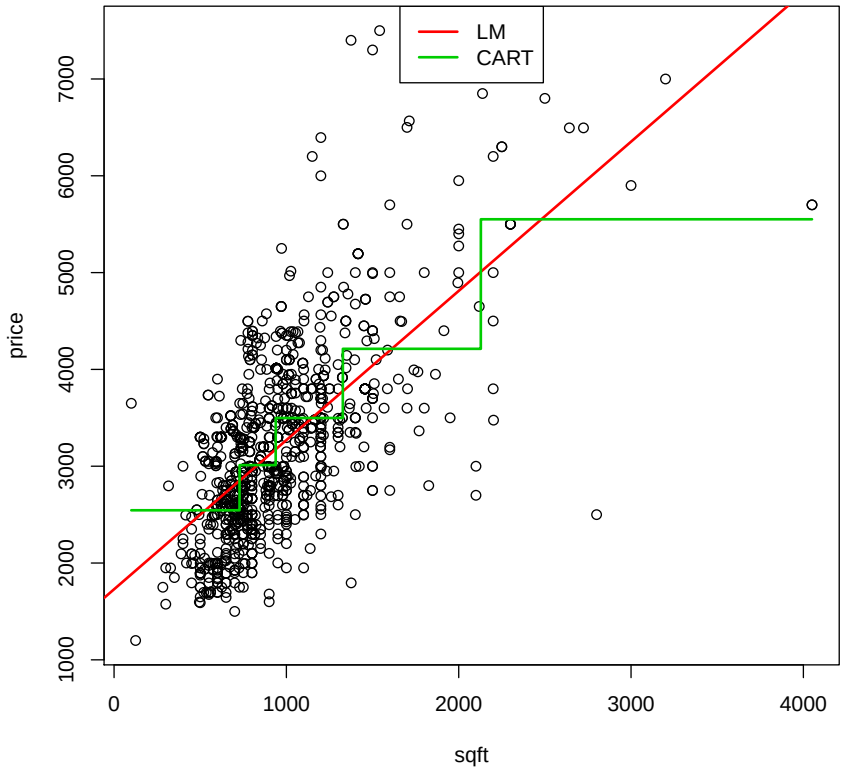
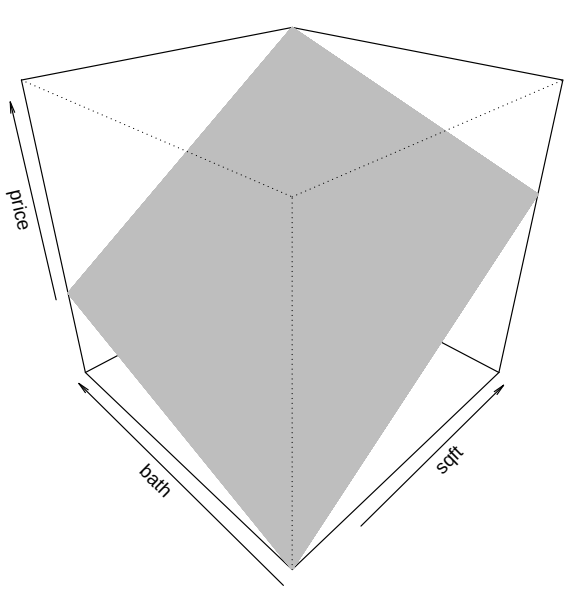


Figure 12: *Craigslist: Linear and tree-based regression.* Regression function of rent on "sqft", linear regression (red) and regression tree (green).

Craigslist: Linear Regression



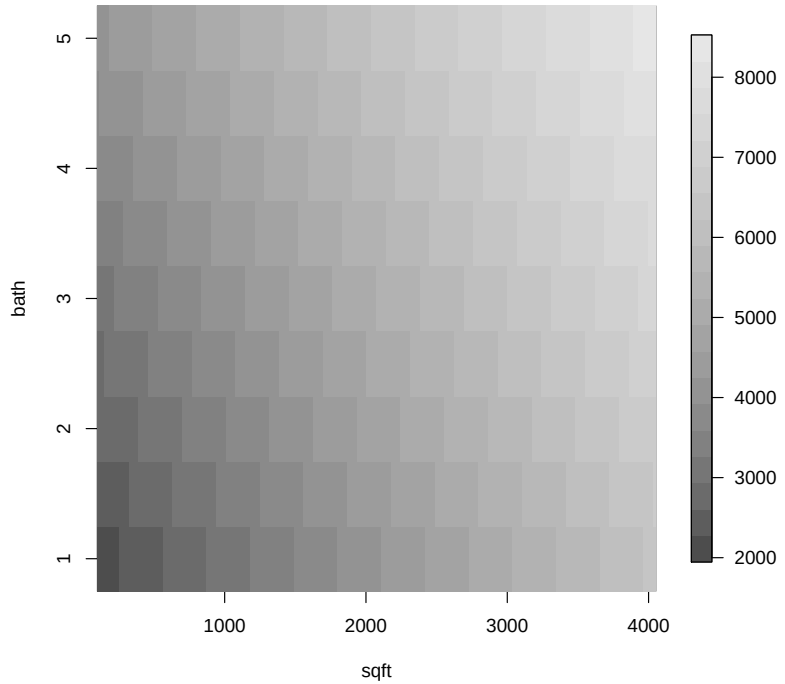


Figure 13: *Craigslist: Linear regression.* Regression function of rent on "sqft" and "bath".

Craigslist: Regression Trees

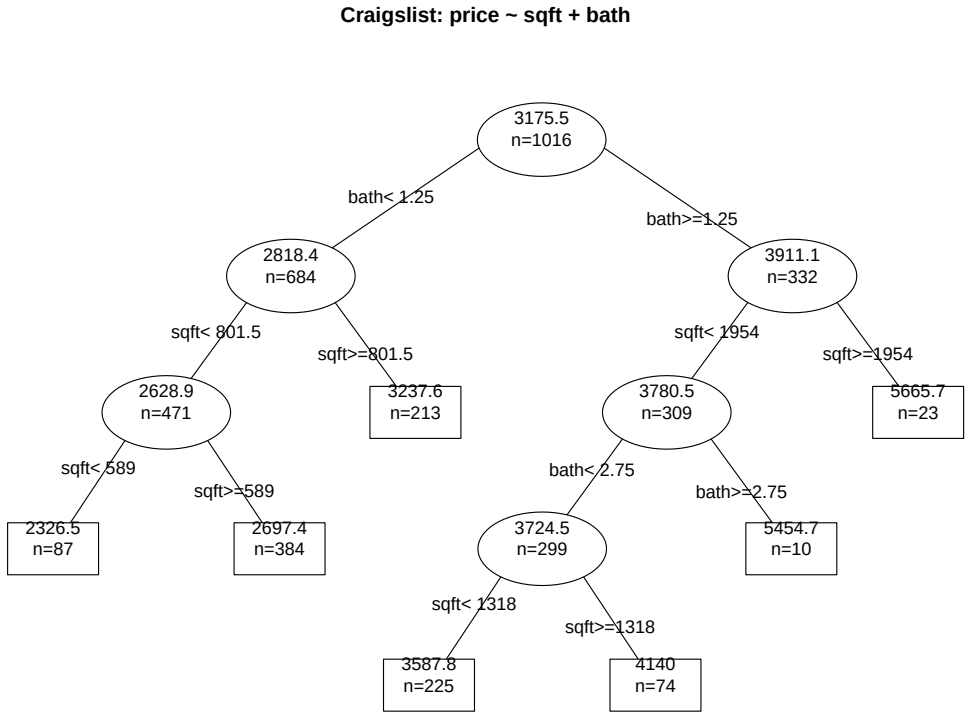
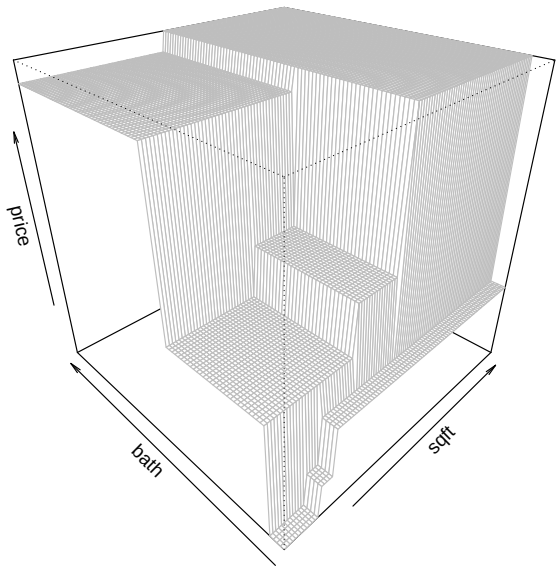


Figure 14: *Craigslist: Regression trees.* Decision tree for regression of rent on "sqft" and "bath".

Craigslist: Regression Trees



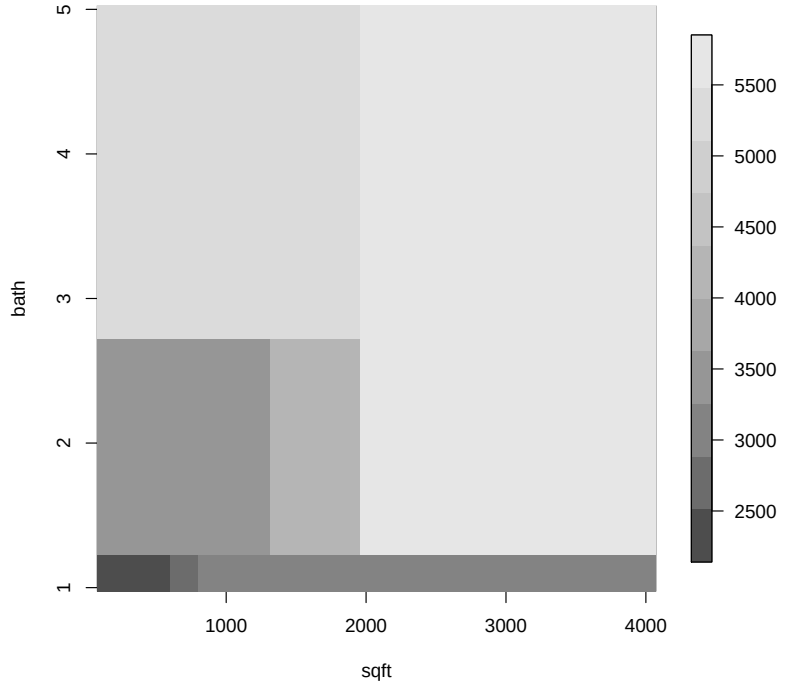


Figure 15: *Craigslist: Regression trees.* Regression function of rent on "sqft" and "bath".

Craigslist: Regression Trees

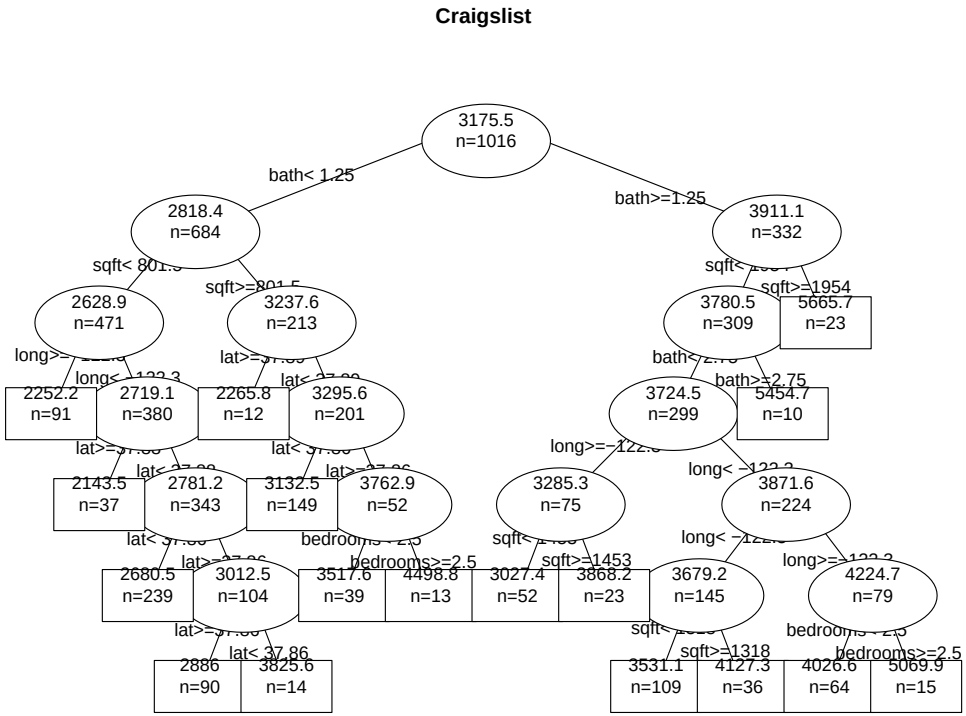


Figure 16: *Craigslist: Regression trees.* Decision tree for regression of rent on all 5 covariates.

Craigslist: Random Forests

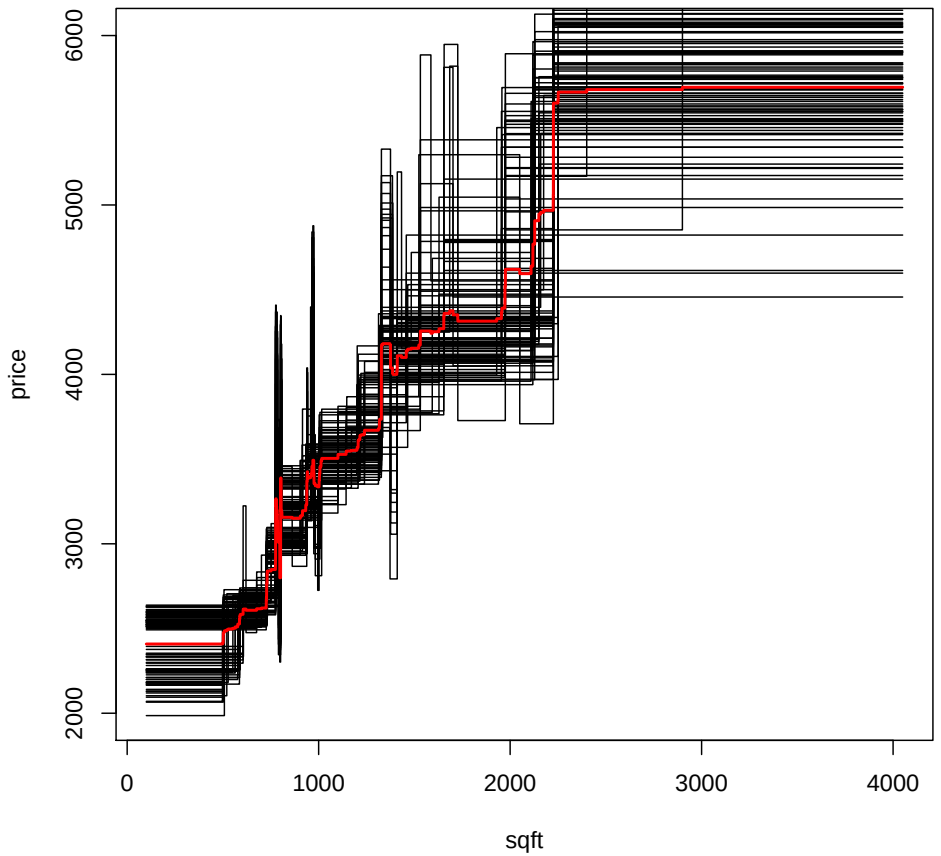
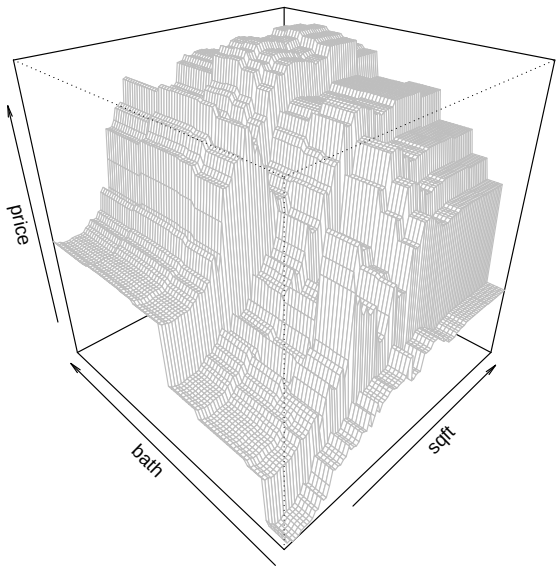


Figure 17: *Craigslist: Random Forests.* Regression function of rent on "sqft" for bootstrap samples of the learning set. Red curve is average.

Craigslist: Random Forests



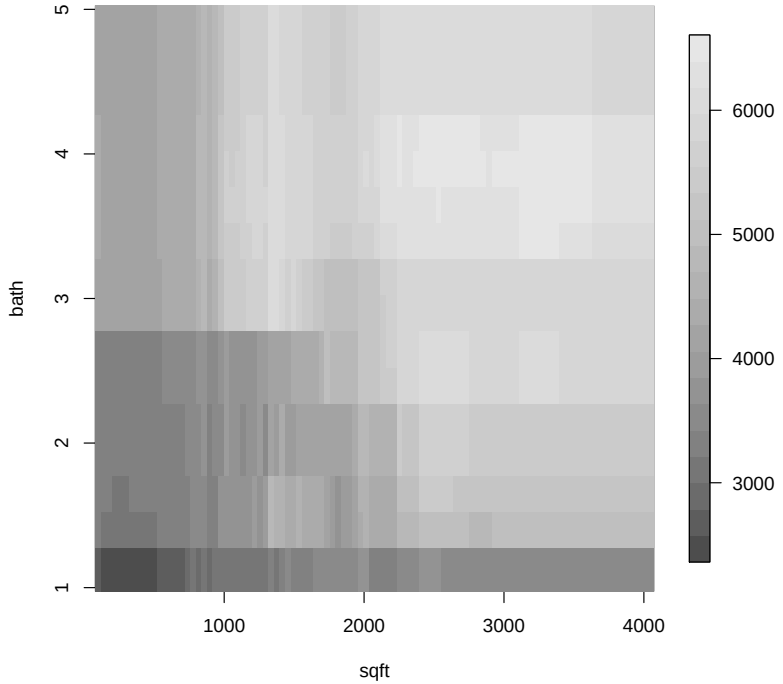
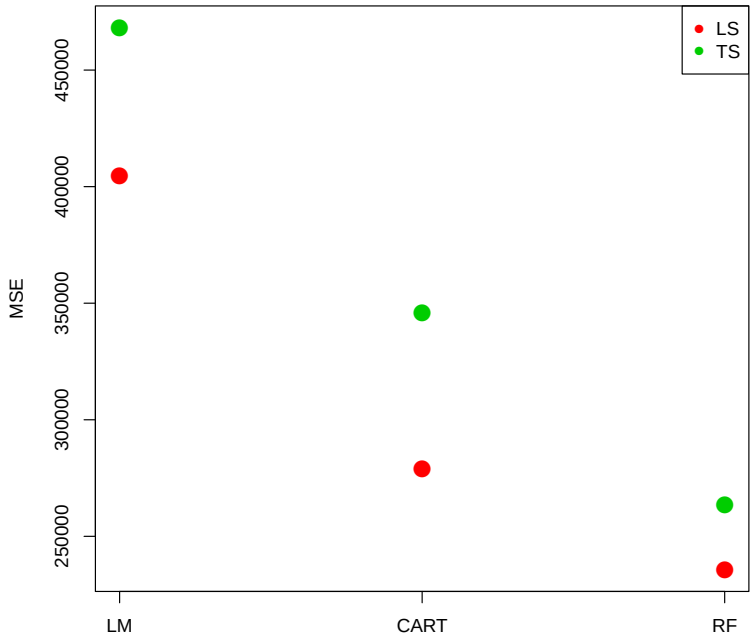


Figure 18: *Craigslist: Random Forests.* Regression function of rent on "sqft" and "bath".

Craigslist



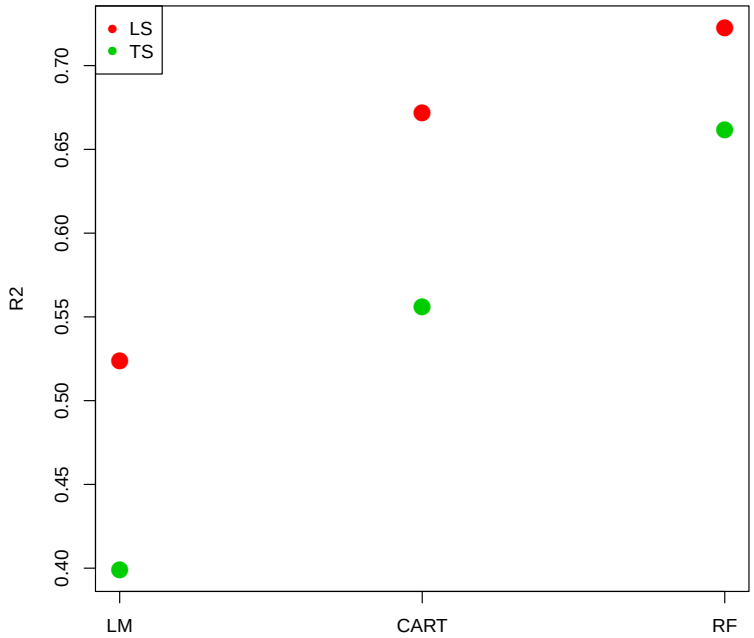


Figure 19: *Craigslist: Linear regression, CART, and Random Forests.* MSE and R^2 on learning and test sets (80-20% random split of dataset) for regression of rent on all 5 covariates.

MNIST Handwritten Digit Recognition

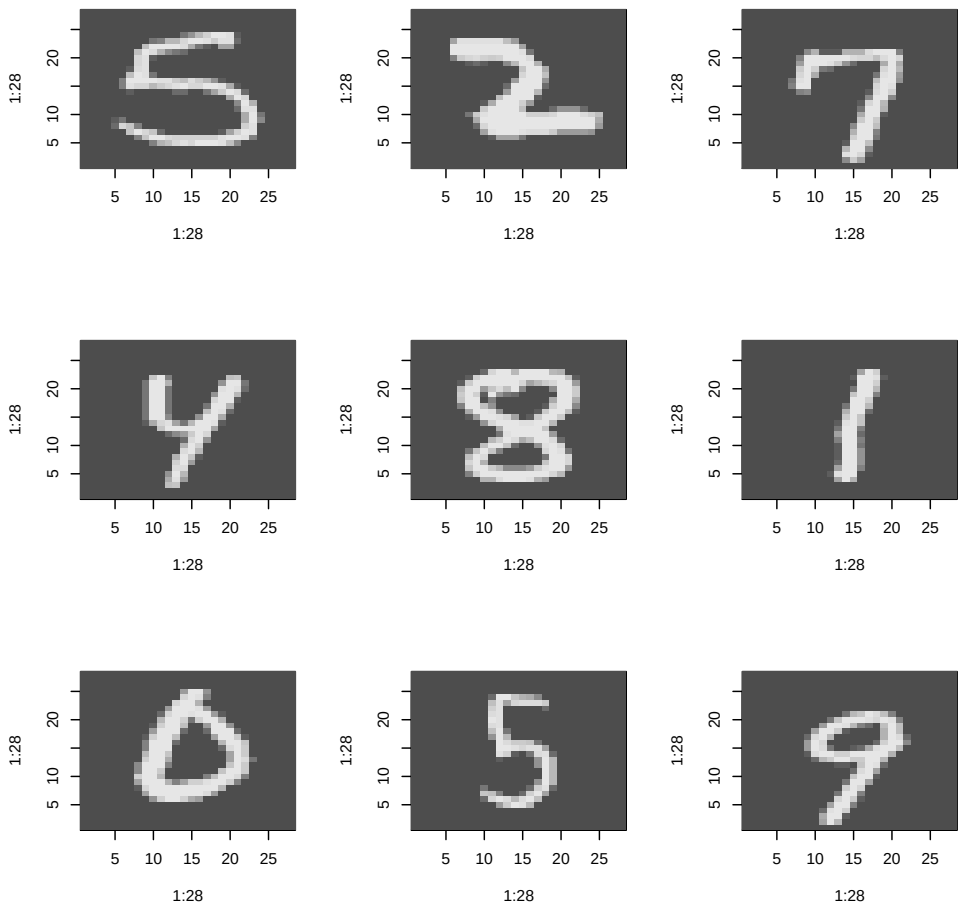


Figure 20: *MNIST digits.* Random sample of 9 images from the MNIST learning set, 28×28 pixels, $[0, 2^8 - 1]$.

MNIST Digits: Classification Trees

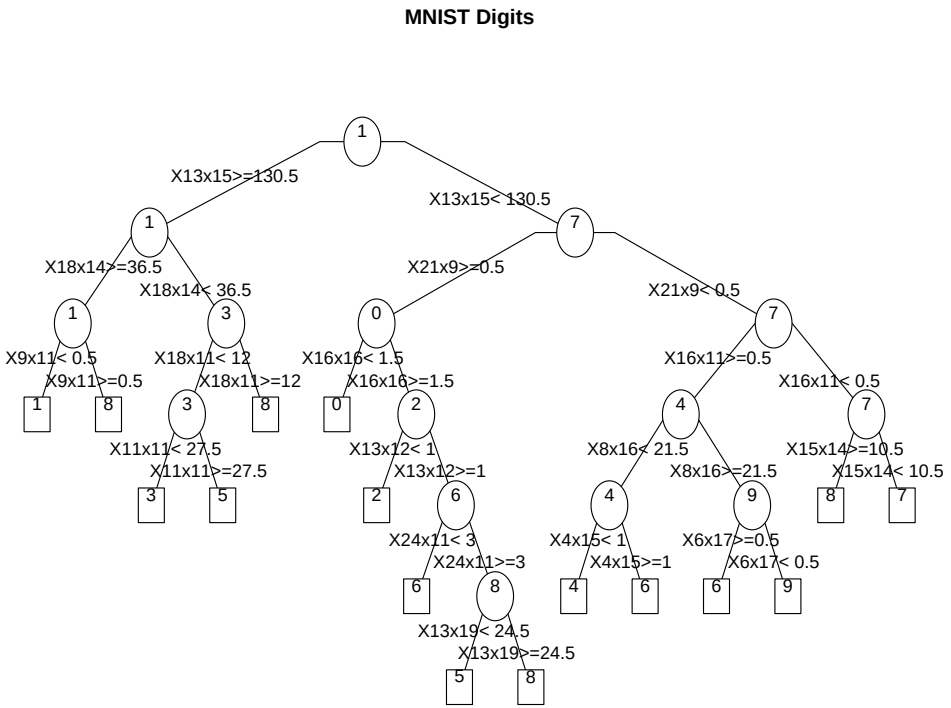


Figure 21: *MNIST digits: Classification trees.* Decision tree for predicting handwritten digits.

MNIST Digits: Random Forests

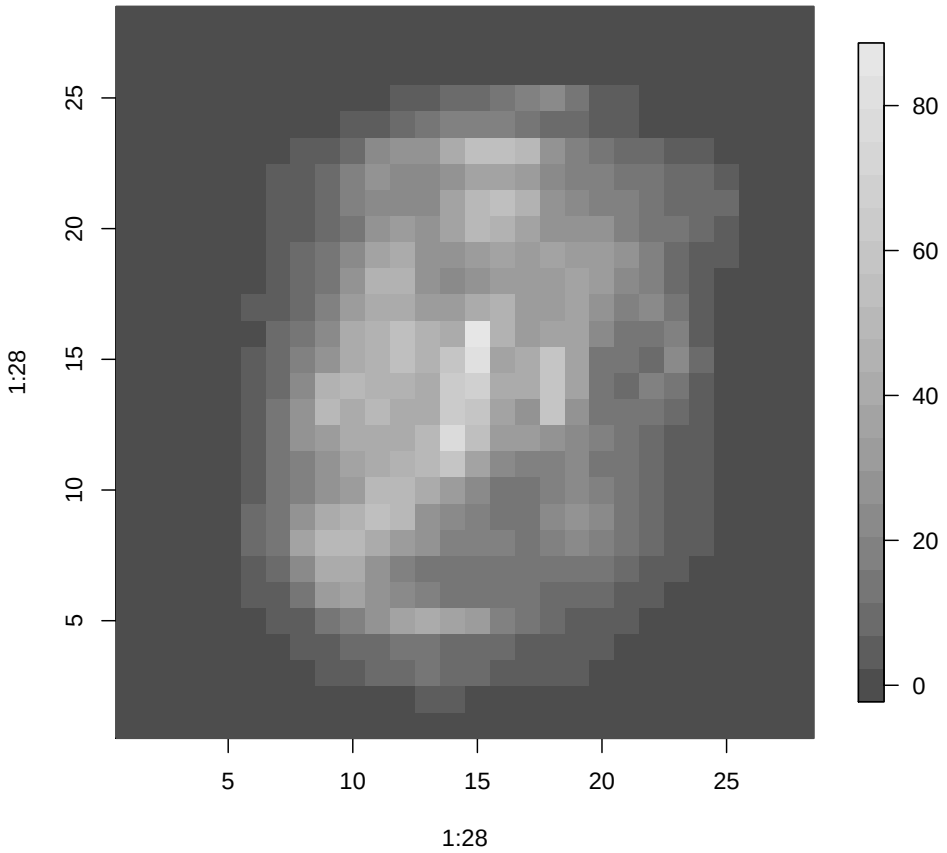


Figure 22: *MNIST digits: Random Forests.* Pseudo-color image of variable importance measures for learning set.

MNIST Digits

Table 2: *MNIST Digits: CART, Random Forests, and k-NN.* Classification error rates (%) on test set (subsets of MNIST LS and TS).

CART	35.6
RF	3.3
1-NN	4.2
5-NN	4.4
10-NN	4.8
25-NN	6.3
50-NN	8.4

Regression Example

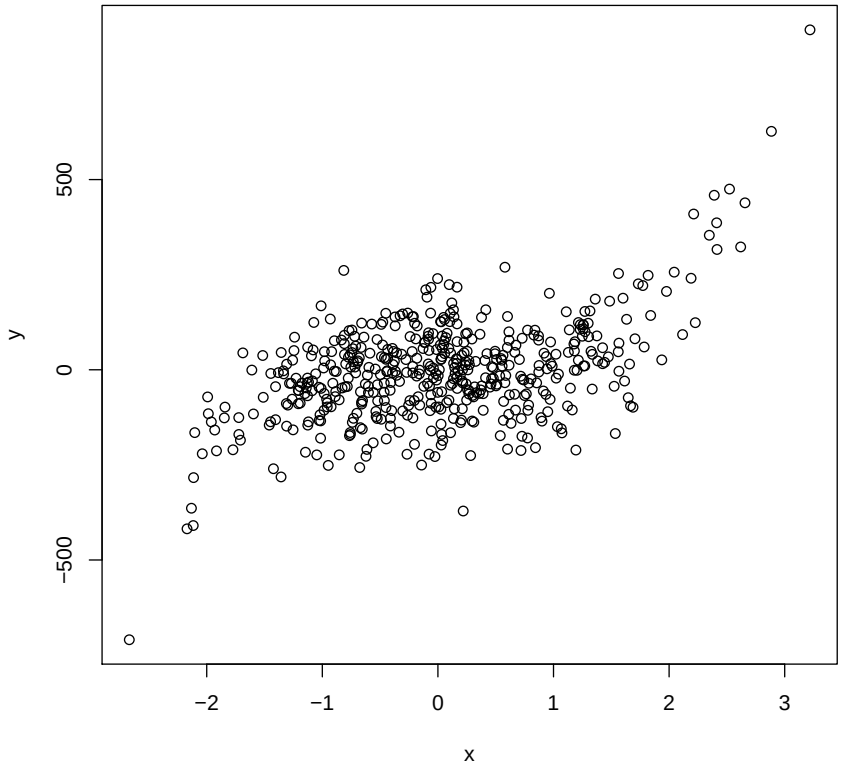


Figure 23: *Regression.* Scatterplot of 500 covariate-outcome pairs from an unknown data generating distribution. What is the regression function?

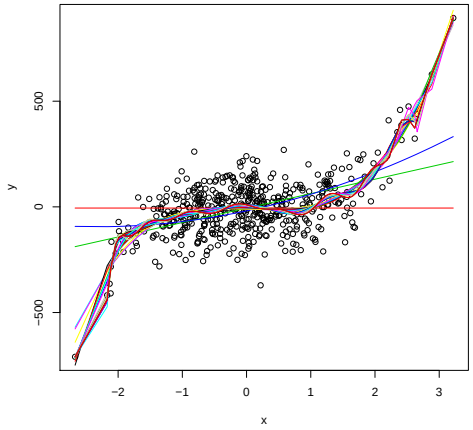
Regression Example

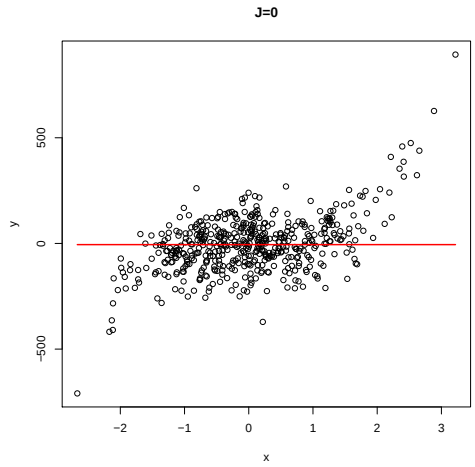
- Suppose we have a learning set $\mathcal{L}_n = \{(X_i, Y_i) : i = 1, \ldots, n\}$ of $n = 500$ independent and identically distributed (IID) covariate-outcome pairs from an unknown data generating distribution P.
- How can we use these data to estimate the regression function of Y on X: $\theta(X) = \mathrm{E}_P[Y|X]$?
- Based on the scatterplot of Y vs. X, it seems that the regression function is non-linear in X, i.e., a constant or linear (in X) regression function would be too simple to capture the patterns/trends suggested by the plot.
- We could try fitting polynomials in X of higher degrees. The higher the degree of the polynomial, the better the fit on the learning set.

Regression Example

- However, by arbitrarily increasing the polynomial degree, we risk fitting the noise, as opposed to the actual signal, in the learning data.

Regression Example: Model Complexity





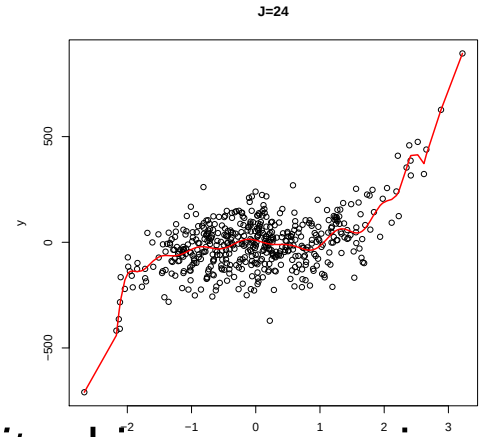
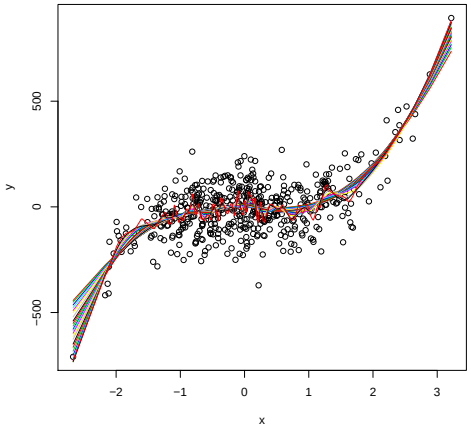
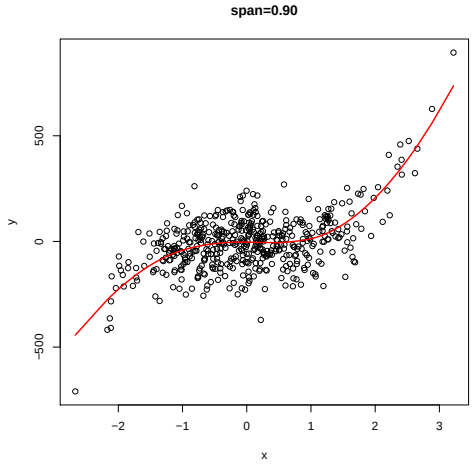


Figure 24: *Linear regression complexity.* Linear regression fits for polynomials of degree 0 to 24.

Regression Example: Model Complexity





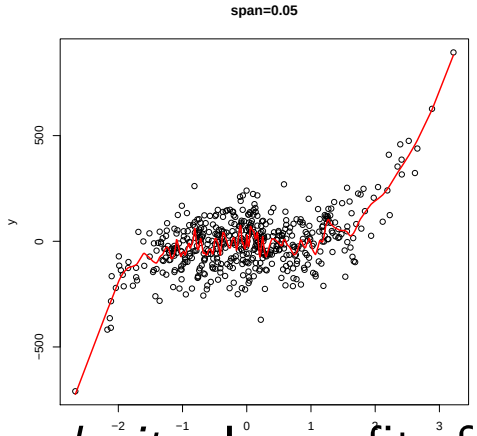


Figure 25: *Robust local regression complexity.* Loess fits for spans ranging from 0.05 to 0.90.

Regression Example: Model Complexity

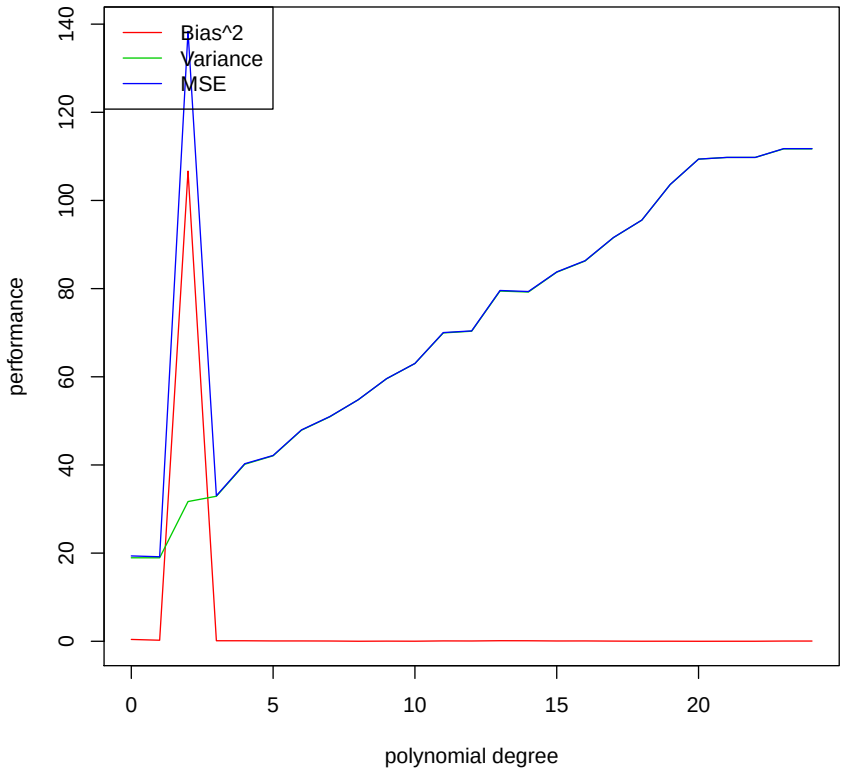


Figure 26: *Bias-variance trade-off: Linear regression.* Bias, variance, and MSE for linear regression fits for polynomials of degree 0 to 24.

Regression Example: Model Complexity

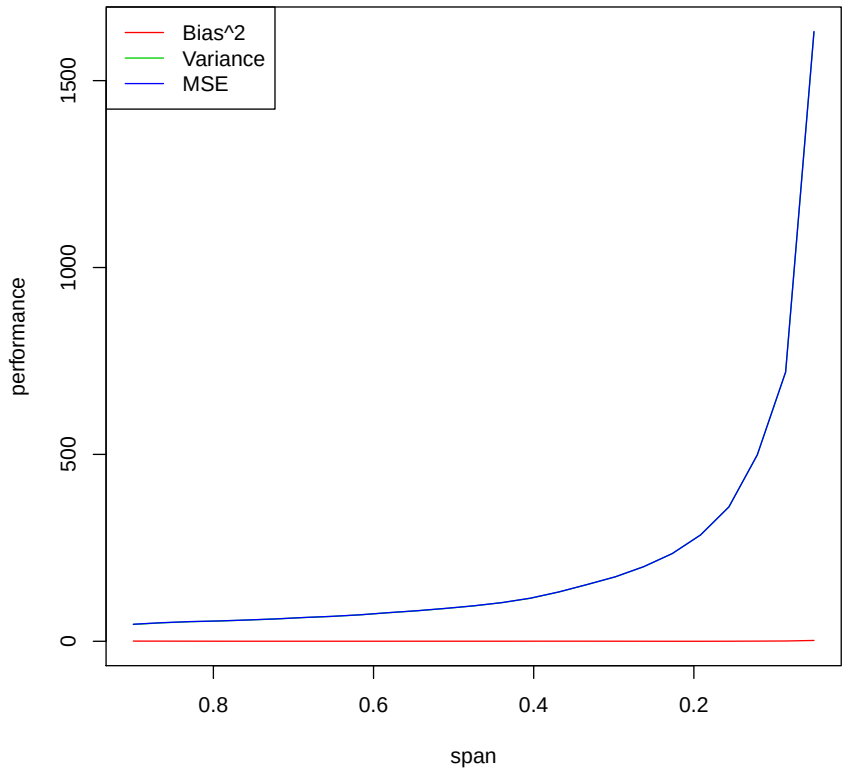


Figure 27: *Bias-variance trade-off: Robust local regression.* Bias, variance, and MSE for loess fits for spans ranging from 0.05 to 0.90.

Regression Example: Sample Size

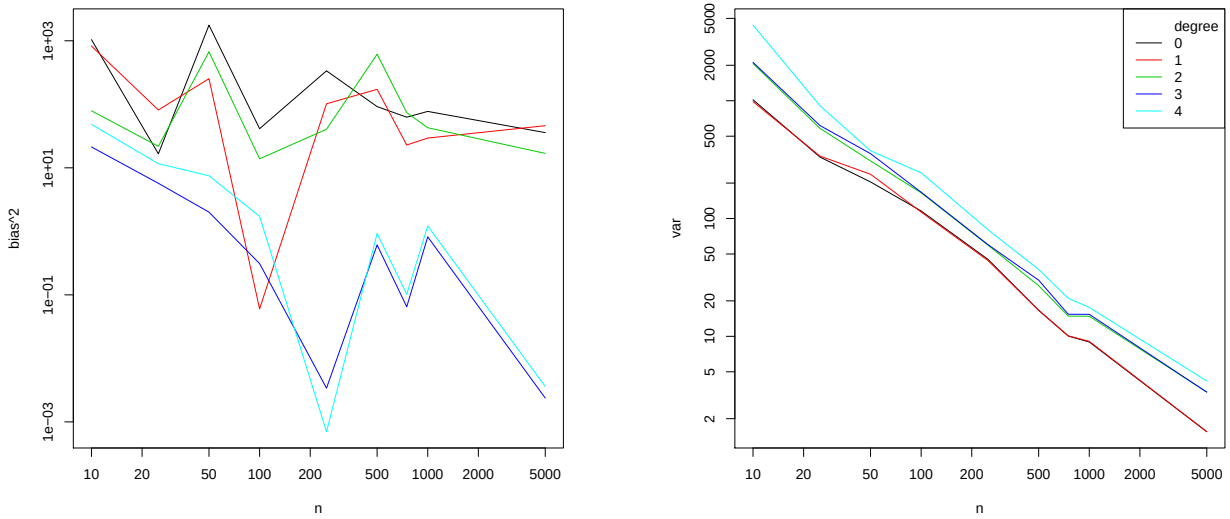


Figure 28: *Effect of sample size on bias and variance: Linear regression.* Bias and variance for linear regression fits vs. sample size n, for polynomials of degree 0 to 4.

Regression Example: Sample Size

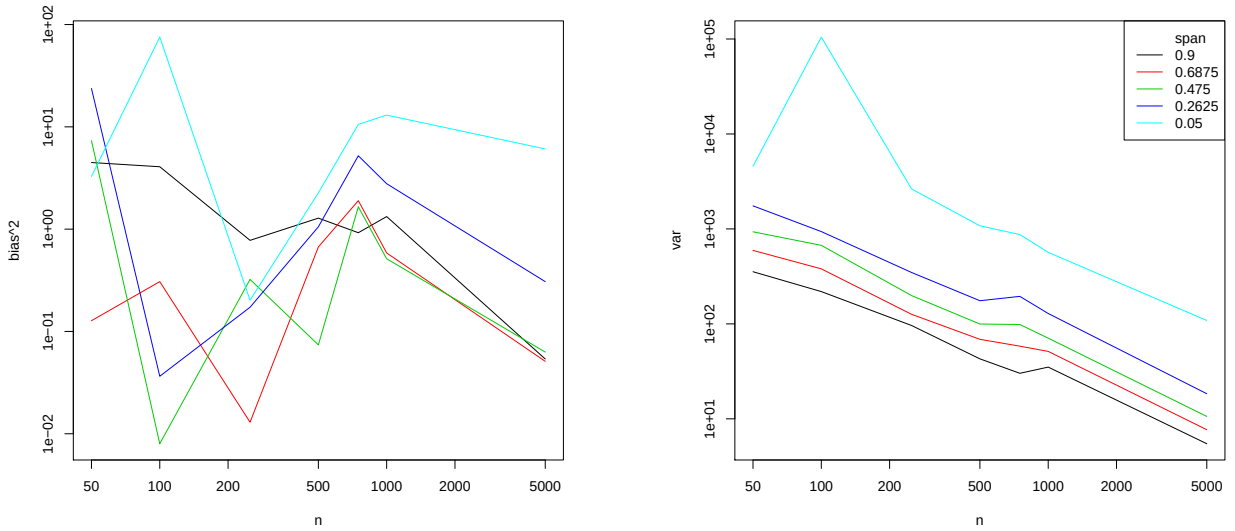


Figure 29: *Effect of sample size on bias and variance: Robust local regression.* Bias and variance for loess fits vs. sample size n, for spans ranging from 0.05 to 0.90.

Regression Example: True Regression Function

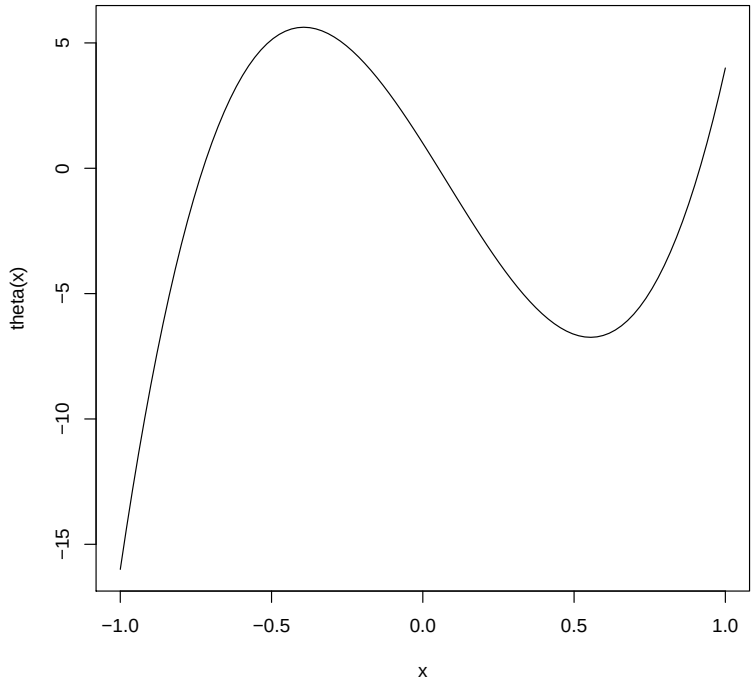


Figure 30: *Regression.* True regression function $\theta(x) = \mathrm{E}_P[Y|X = x] = 1 - 19x - 7x^2 + 29x^3$. $\mathrm{Var}_P[Y|X] = \sigma^2 = 100^2$. $X \sim \mathrm{N}(0, 1)$.

References

L. Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16 (3):199–215, 2001.

L. Breimn, J. H. Friedman, R. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman & Hall/CRC, Boca Raton, FL, 1984.

E. Fix and J. Hodges. Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical report, Randolph Field, Texas: USAF School of Aviation Medicine, 1951.

T-S Lim, W-Y Loh, and Y-S Shih. A comparison of prediction accuracy, complexity, and training time of thiry-three old and new classification algorithms. *Machine Learning*, 40:203–229, 2000.

B. Yu and K. Kumbier. Three principles of data science: predictability, computability, and stability (PCS). *arXiv*, 2019.