Lecture Four: A Very Short Introduction to Machine Learning

Charles Rahal

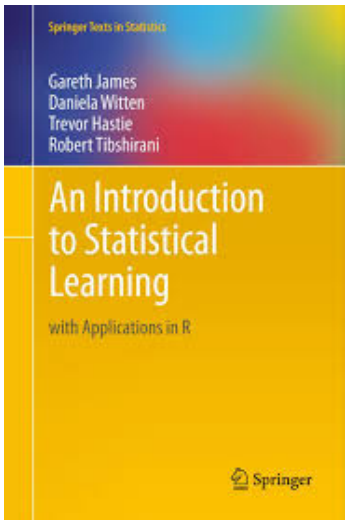
Slides, Notebooks and Markdown at: https://crahal.github.io

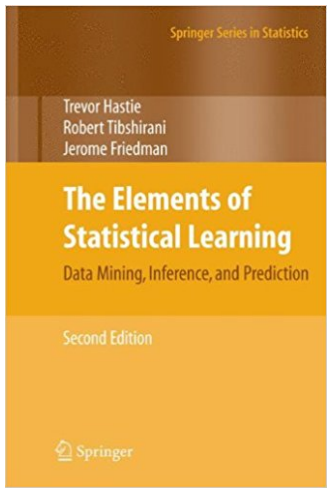
What will we cover?

- We'll cover all basics, give two examples, then host a prediction competition!

1. The key concepts:
   - 1.1 The general form of the problem.
   - 1.2 Conceptual differences between inference and prediction.
   - 1.3 Various types of prediction problem (and why we care).
   - 1.4 Fundamentals of feature engineering.
   - 1.5 Fundamentals of cross validation.
   - 1.6 Linear Model Selection and Regularization.
   - 1.7 An introduction to tree based methods.
   - 1.8 An introduction to support vector machines.
2. Example of supervised regression problem (predicting house prices).
3. Example of supervised classification (whether a person survives the titanic).

Two Highly Recommended Books





- Left ('ISLR') is an introduction to applied statistical learning ('Anyone who wants to intelligently analyze complex data should own this book'). This lecture **closely** follows Chapters 1-6, 8 and 9. 15 hours of lectures/full e-book available for free!
- Right ('ESL') is a more advanced introduction to the technical aspects of statistical learning for students with a technical background.

Differences between Machine and Statistical Learning

When you go to a party and
Someone says, 'What do you do?'
When you say, 'I'm a statistician',
They run for the door.
But nowadays, we can say, well,
'We do machine learning.'
And they still run for the door,
but just take longer to get there.

Machine Learning (ML)	Statistical Learning (SL)
weights	parameters
supervised learning	classification, regression
unsupervised learning	clustering
large grant = \$1m	large grant = \$50k
learning	fitting

- Boundaries between ML and SL increasingly blurred!
- ML: greater emphasis on large scale applications and predictive accuracy.
- SL: greater emphasis on models, interpret-ability, precision and uncertainty

Differences between Machine and Statistical Learning

When you go to a party and
Someone says, 'What do you do?'
When you say, 'I'm a statistician',
They run for the door.
But nowadays, we can say, well,
'We do machine learning.'
And they still run for the door,
but just take longer to get there.

Machine Learning (ML)	Statistical Learning (SL)
weights	parameters
supervised learning	classification, regression
unsupervised learning	clustering
large grant = \$1m	large grant = \$50k
learning	fitting

- Boundaries between ML and SL increasingly blurred!
- ML: greater emphasis on large scale applications and predictive accuracy.
- SL: greater emphasis on models, interpret-ability, precision and uncertainty

Differences between Machine and Statistical Learning

When you go to a party and
Someone says, 'What do you do?'
When you say, 'I'm a statistician',
They run for the door.
But nowadays, we can say, well,
'We do machine learning.'
And they still run for the door,
but just take longer to get there.

Machine Learning (ML)	Statistical Learning (SL)
weights	parameters
supervised learning	classification, regression
unsupervised learning	clustering
large grant = \$1m	large grant = \$50k
learning	fitting

- Boundaries between ML and SL increasingly blurred!
- ML: greater emphasis on large scale applications and predictive accuracy.
- SL: greater emphasis on models, interpret-ability, precision and uncertainty

A General Form

- Suppose a quantitative response Y and p different predictors, X_1, X_2,..., X_p.
- Assume some relationship between Y and $X = (X_1, X_2, \ldots, X_p)$:

$$Y = f(X) + \varepsilon \tag{1}$$

- Where f is some fixed but unknown function.
- ε is a random error term.
- Statistical/Machine Learning refers to a set of approaches for estimating f.

A General Form

- Consider two classic social science examples:
  - Regression: modeling years of education and income.
  - Classification: predicting whether somebody will default on their loan.

We're likely familiar with basic multivariate linear regression form:

$$y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \ldots + \varepsilon_i \tag{2}$$

Possibly in linear-log form, possibly with some polynomials:

$$y_i = \beta_0 + \beta_1 ln(x_{1,i}) + \beta_2 x_{2,i}^2 + \ldots + \varepsilon_i \tag{3}$$

The World Is Not Flat

A General Form

- Consider two classic social science examples:
  - Regression: modeling years of education and income.
  - Classification: predicting whether somebody will default on their loan.

We're likely familiar with basic multivariate linear regression form:

$$y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \ldots + \varepsilon_i \tag{2}$$

Possibly in linear-log form, possibly with some polynomials:

$$y_i = \beta_0 + \beta_1 ln(x_{1,i}) + \beta_2 x_{2,i}^2 + \ldots + \varepsilon_i \tag{3}$$

The World Is Not Flat

Why Estimate f? Prediction

- $\hat{f}$ represents our estimate for f, $\hat{Y}$ the prediction of Y:

$$\hat{Y} = \hat{f}(X) \tag{4}$$

- $\hat{f}$ is often treated as a black box (no concern for form).
- The accuracy of $\hat{Y}$ depends on reducible and irreducible error.
- As $\hat{f}$ will not be a perfect estimate for f, this introduces reducible error.
- Even if it were possible to form a perfect estimate ($\hat{Y} = f(X)$), we would still have some error (ε: perhaps a function of unobserved variables/mis-measurement):

$$\begin{aligned} E(Y - \hat{Y})^2 &= E[f(X) + -\hat{f}(X)]^2 \\ &= \underbrace{[f(X) - \hat{f}(X)]^2}_{\text{Reducible}} + \underbrace{Var(\varepsilon)}_{\text{Irreducible}} \end{aligned} \tag{5}$$

- We term this irreducible error.

Why do we care about prediction?

- Increasing computational power, infinite data pipelines and algorithmic advances force us to reconsider what's possible.
- 'Future of mankind is reformulating existing challenges as prediction problems'.
- Where does prediction impact our lives?:
  1. Naive Bayes classifiers: protect your emails from spam.
  2. Recommender Systems: what to watch (Netflix) or buy (Amazon)
  3. Deep Convolutional Neural Networks: facial recognition (Facebook).
  4. Recurrent Neural Network: Speech recognition (Siri, Alexa, etc).
  5. Reinforcement learning: AlphaGo teaches itself how to win without data.
- Can you think of any other popular or commonplace algorithms?

Why Estimate f? Inference

- This is about the way Y is affected as $X_1, \ldots, X_p$ change.
- f cannot be treated as a black box, because we need to know its exact form.
- We might be interested in, for example:
  1. Which predictors are associated with the response?
  2. What is the relationship between the response and each predictor?
  3. Can we summarize with a linear equation, or is it more complicated?
- Depending on whether our goal is prediction, inference, or both, different methods for estimating f may be appropriate:
  - Linear models allow for relatively simple interpretation, but predict poorly.
  - Highly non-linear approaches often predict well, but are less interpret-able.
- We're set to focus on prediction for the remainder of this class.

Class Quiz: Which is inference, which is prediction?

- Given the age of a passenger, did they survive the titanic?
- What is the effect of gender on surviving the titanic?

- How does the local crime rate change house prices?
- Given a set of variables, what will a specific house sell for?

- How does an additional pound of income affect the probability of default?
- Will a specific loan applicant default on their loan?

- Will Oxford United F.C. win the first game of the new season?
- Is a team more likely to win if they shoot towards their own fans in the 2nd half?

Supervised vs. Unsupervised Learning

- Two main types of machine learning problems: supervised and unsupervised:

1. Supervised problems: make predictions based on a set of examples (training data).
    - We can further split supervised problems into two categories:

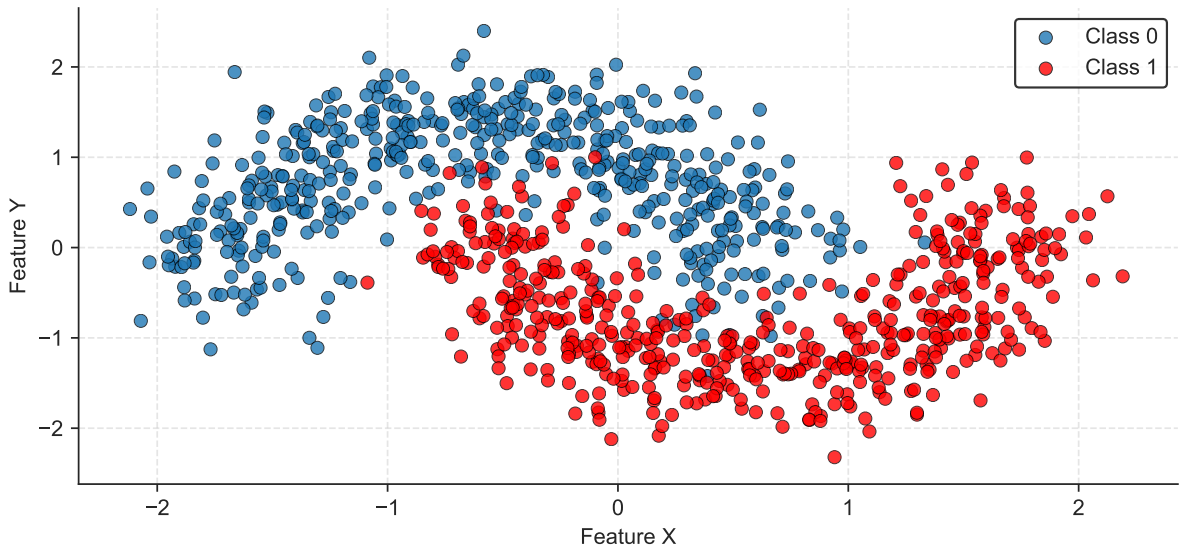
    1.1 Classification: predict which category something falls into (a discrete set of values). Examples include: Naive Bayes, Supper Vector Machines, Logistic Regressions, Neural Networks.

    1.2 Regression: make a prediction on a continuous scale. Examples could be predicting the stock price of a company or predicting the temperature tomorrow. Examples of techniques include OLS, Nonlinear Regression and Bayesian Linear Regression.

2. Unsupervised problems: where our data does not have a defined set of categories, but instead we are looking for the machine learning algorithms to help us organize the data (i.e. to look for hidden features and cluster it a way that makes sense). Examples: K-means clustering, Neural Networks, Principal component analysis

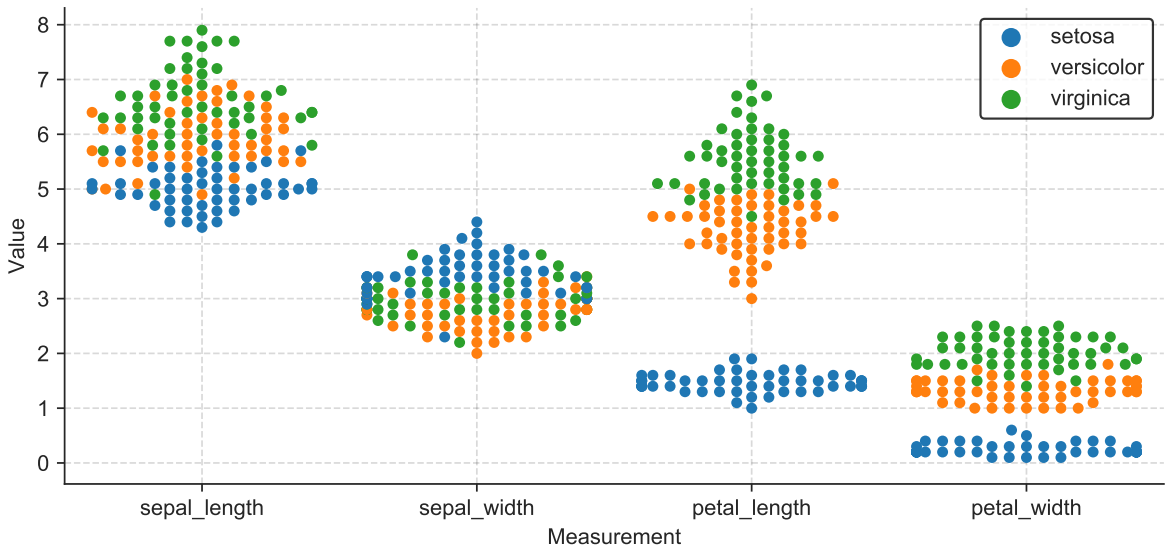
- Lets now look at examples of each of these three in turn!

A Simulated Binary Classification Example



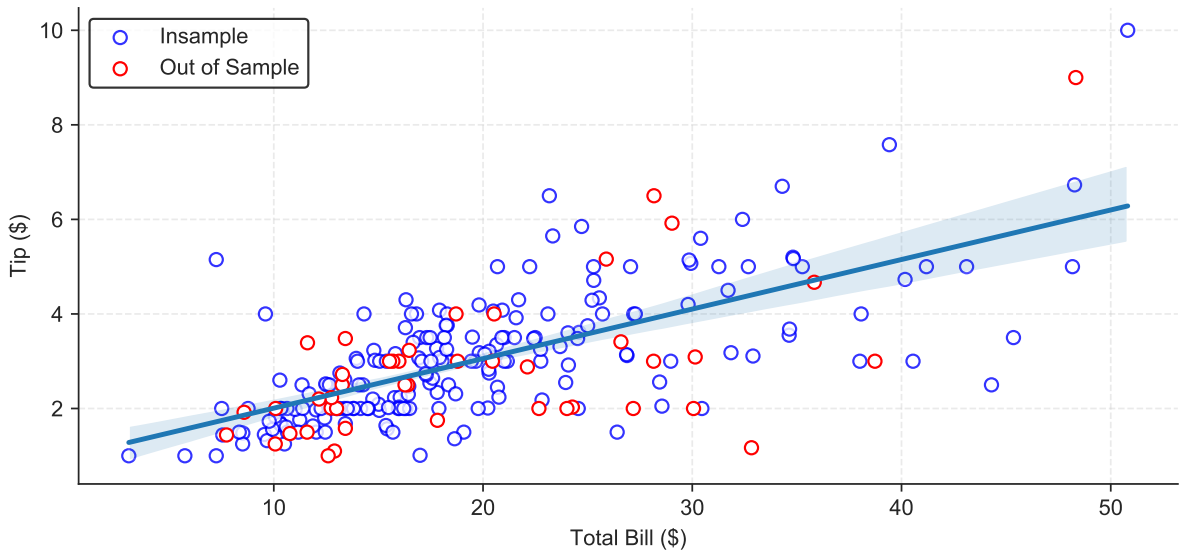
- The objective: given X and Y, can we create a model which predicts Class 1 or 0?
- Binary classification conceptually/analytically/computationally simpler than multi-class.

A Classic (Multi-Class) Classification Example



- R.A. Fisher (1936): 'The Use of Multiple Measurements in Taxonomic Problems':
- 50 samples from each of Iris setosa, Iris virginica and Iris versicolor.
- 4 features measured: length and width of sepals and petals (in cm).
- Objective: if we measure a new plant, can we accurately predict the species?

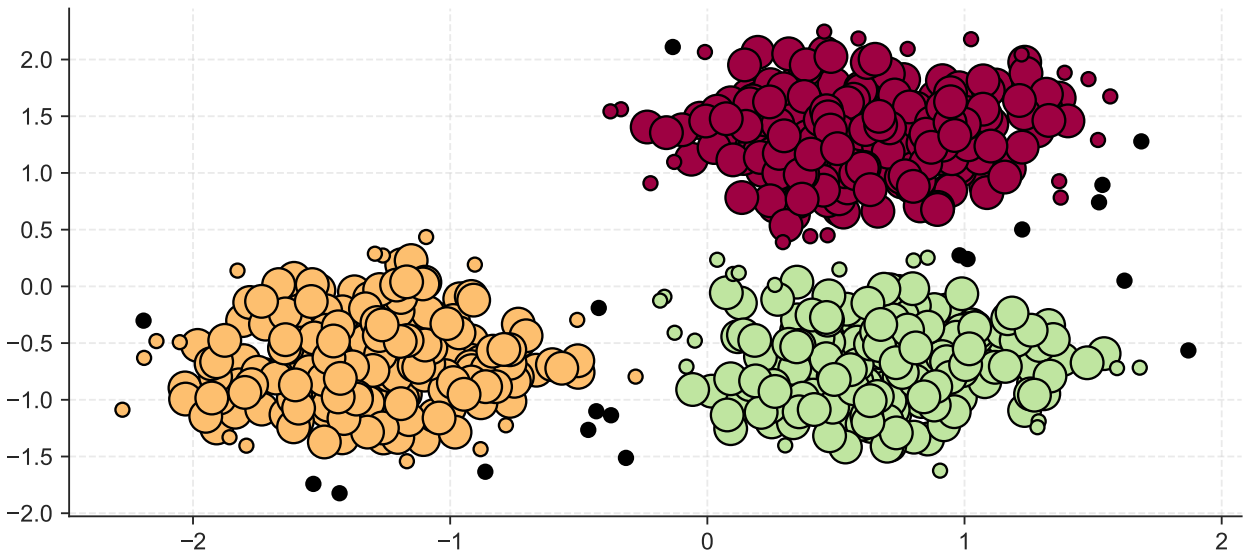
A Less Classic Regression Example



- As mentioned above, the objective of regression is to predict continuous variables.
- Food servers' tips in restaurants may be influenced by many factors.
- Blue dots are part of the training set, the red are the test set (more shortly).
- Given data on the total bill, can we predict how much somebody will tip?

An Unsupervised Learning Example

- Algorithm decides clusters/categories: no external classification.
- Most famous algorithm of this kind is called 'k-means clustering'.



- Example: density-based spatial clustering of applications with noise (DBSCAN)
- More familiar? Latent Dirichlet allocation of topics in a (series of) document(s).

Class Quiz: Which is classification, which is regression?

Explain whether each scenario is a classification or regression problem, and indicate whether we are most interested in inference or prediction. What are n and p for each?

1. We collect a set of data on the top 500 firms in the US. For each firm we record profit, number of employees, industry and the CEO salary. We are interested in understanding which factors affect CEO salary.
2. We are considering launching a new product and wish to know whether it will be a success or a failure. We collect data on 20 similar products that were previously launched. For each product we have recorded whether it was a success or failure, price charged for the product, marketing budget, competition price, and ten other variables
3. We are interested in predicting the % change in the USD/Euro exchange rate in relation to the weekly changes in the world stock markets Hence we collect weekly data for all of 2012. For each week we record the % change in the USD/Euro, the % change in the US market, the % change in the British market, and the % change in the German market.

Introducing Feature Engineering

- The question of how to best represent your data is known as 'feature engineering'.
- It can have a critical influence on the performance of your prediction.
- The 'model selection' stage determines which features to actually include.

1. Categorical Variables:

- What about a variable such as colour? One hot encoding:

Color	red	green	blue	yellow
red	1	0	0	0
green	0	1	0	0
red	1	0	0	0
yellow	0	0	0	1

- Note: categorical variables in survey data-sets are often encoded using integers.

Feature Engineering Cont:

2. **Binning (or 'Discretization'):**
   - Binning continuous data **and** using one-hot can often improve performance.
   - Example: Bin ages into deciles, then one-hot encode into a bunch of dummies.
3. **Interactions and Polynomials:**
   - Often used in statistical modeling, but also useful in machine learning.
   - Polynomials can substantially increase performance and reduce under-fitting.
   - $Y = \beta_0 + \beta_1 X_1$ transforms to quadratic: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2$ (still linear).
   - We can also remove the additive assumption: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$.
4. **Uni-variate Non-linear transformations:**
   - Functions like `log` and `exp` help by adjusting relative scales of the data so they can be better modelled using linear regression or neural networks.
   - Most models work best when each feature is loosely Gaussian: log and exp are a hacky but effective way to achieve this (to deal with non-constant variance).

Introducing Cross-Validation

- Until now, we haven't adequately discussed training and test sample splits.
- If we go back to the slide titled 'Regression', some dots are red, some are blue.
- 'In-sample' prediction is an unrealistic representation of a typical forecasting exercise in practice: it gives the model data which is then use to create the predictions which itself generates...
- We need to split our data into two (or three):
  1. Training: fit the data to the model.
  2. (Optional) Grid-search: tune hyper-parameters associated with model.
  3. Test: how well this fit (and parameter set) applies to unseen observations.
- This is 'out-of-sample' forecasting or 'cross-validation' of the model.

k-fold Cross-Validation

- Randomly splitting into a train and test sample set is not the most robust solution.
- It might provide a disproportionately easy test set.
- k-fold cross-validation involves partitioning the data into k folds....
- then training the model on all but one of the folds, and testing it on the remainder, and cycling through.
- e.g.: 5-fold cross-validation – train on fold 1-4, test on fold 5. Then, train on fold 1,2,3 and 5, test on 4, and so on.
- This gives us k test sub-samples to evaluate the stability of our model across (i.e. analyze the spread, mean of evaluation metrics): but computationally costly.
- *Stratified* k-fold cross-validation ensures an even split of case/controls in each fold.
- *Leave-one-out* cross-validation (a specific example of k-fold, where k=n-1) trains on all data apart from one observation, testing on the one.
- For many algorithms, we need a third split for tuning parameters (as we'll see).

Model Evaluation: Binary Classification

		Prediction Outcome	
		$\mathbf{p}$	$\mathbf{n}$
Actual Value	$\mathbf{p}'$	True Positive	False Negative
	$\mathbf{n}'$	False Positive	True Negative

- Using simple accuracy may not be advisable due to:
  1. Different types of errors not having symmetrical effect (e.g. FP vs TN).
  2. Unbalanced datasets (i.e. a many controls, few cases – e.g. click-through).
- Solution: the confusion matrix, summarized with precision, recall and f-scores:
- Precision $= \frac{\mathrm{TP}}{\mathrm{TP+FP}}$, Recall $= \frac{\mathrm{TP}}{\mathrm{TP+FN}}$. $\mathrm{F} = 2 \times \frac{\mathrm{precision} \times \mathrm{recall}}{\mathrm{precision+recall}}$
- Other common metrics include the Receiver Operating Characteristics (ROC) curve (FP rate vs TP rate) and the Area Under the [ROC] Curve (AUC).

Model Evaluation: Multi-class and Regression

Multi-class

- Multi-class evaluation for categorical problems expands naturally from the binary classification problem.
- e.g.: a 10×10 confusion matrix, with adjustments to f-score/other summaries.

Regression

- Regression problems need other metrics which measure the continuous distance from what was observed, such as the mean squared error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{6}$$

- or the mean absolute error (depending on how we want to penalize the distance):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}(|Y_i - \hat{Y}_i|) \tag{7}$$

- Implementations typically offer *a lot* of different types of 'loss functions'.

The Bias-Variance Trade-off

- We can show expected test MSE can always be decomposed into the 3 quantities:

$$E(y_0 - \hat{f}(x_0))^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\varepsilon) \tag{8}$$

- Here, $E(y_0 - \hat{f}(x_0))^2$ defines the expected test MSE.
- Note: expected test MSE can never lie below Var(ε) – the irreducible error.
- **Variance** refers to the amount by which $\hat{f}$ changes if estimated with a different training data set.
- **Bias** refers to the error that is introduced by approximating a real-life problem.
- Generally, as we use more flexible methods, the variance increases, bias decreases.
- Relative tradeoff between variance and bias determines whether test MSE $\uparrow$ or $\downarrow$.

Subset Selection: Best Subset Selection

- One way to increase predictive accuracy is to undertake *subset selection*.
- Identify a subset of p predictors related to Y (then fit with least squares).

Algorithm One: Best subset selection

Step 1: Let $\mathcal{M}_0$ denote the null model, which contains no predictors.
Step 2. For k= 1, 2, . . ., p.
    Step 2a. Fit all $\binom{p}{k}$ models that contain exactly k predictors
    Step 2b. Pick the best among these, and call it $\mathcal{M}_k$
Step 3: Select the single best model from among $\mathcal{M}_0$, . . . , $\mathcal{M}_k$ models.

- The objective is to select the 'best' model from the 2^p predictors.
- We then get a set of $p + 1$ models, which we can easily choose between.
- However, this is an extremely computationally demanding procedure.
- When p=20, there are over a million models to estimate....

Subset Selection: Forward Selection

- The best subset selection algorithm is often computationally infeasible.
- The 'best' model might not be parsimonious if p is large.

Algorithm Two: Forward step-wise selection

Step 1: Let $\mathcal{M}_0$ denote the null model, which contains no predictors.
Step 2. For k= 0, . . ., $p-1$.
    Step 2a. Consider all p-k models augmenting $\mathcal{M}_k$ (with one parametre)
    Step 2b. Pick the best among these, and call it $\mathcal{M}_{k+1}$
Step 3: Select the single best model from among M_0, . . . , $\mathcal{M}_p$ models.

- When $p = 20$, we need to estimate 1,048,576 models for best ubset selection...
- But with forward step-wise selection, we only need to fit 211!
- It is not guaranteed to find the best model (in the superset).

Subset Selection: Backward Selection

- An analogous strategy is 'Backward Selection'.
- It begins with the full least squares model containing all p predictors ...
- Then removes one at a time.

Algorithm Three: Backward stepwise selection

Step 1: Let $\mathcal{M}_p$ denote the model which contains all p predictors.
Step 2. For $k = p, p-1, \ldots, 1$.
    Step 2a. Consider all k models that contain all but one predictor
    Step 2b. Pick the best among these, and call it $\mathcal{M}_{k-1}$
Step 3: Select the single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ models.

- Like Forward Selection, we only need to estimate $\frac{1+p(p+1)}{2}$ models.
- Unlike Forward Selection, we cannot apply it when $n < p$.

Subset Selection: What is the 'Best' Model?

- How do we assess what represents the best model at each step?
- Each metric below has different properties, and can result in a different model.
- Note: adding variables causes RSS and R^2 to monotonically increase with k...

Method One: C_p, AIC, BIC and $\bar{R}$

$$C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2) \tag{9}$$

$$AIC = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + 2d\hat{\sigma}^2) \tag{10}$$

$$BIC = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + \log(n)d\hat{\sigma}^2) \tag{11}$$

$$\bar{R}^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)} \tag{12}$$

Method Two: Cross Validation

- More feasible now that available computational power has increased.

Shrinkage: Ridge Regressions

- Begin by recalling the least squares fitting procedure, which is to minimize:

$$\text{RSS} = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{i,j})^2 \tag{13}$$

- Ridge regression coefficient estimates ($\hat{\beta}_j^R$) are those which minimize:

$$\underbrace{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{i,j})^2}_{\text{RSS}} + \underbrace{\lambda\sum_{j=1}^{p}\beta_j^2}_{\text{ShrinkagePenalty}} = \text{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2 \tag{14}$$

- The aptly named shrinkage penalty shrinks estimates of β_j towards zero.
- Tuning parameter (λ – determined through CV) controls the relative impact.
- Note: we are not shrinking the intercept ($\hat{\beta}_0$)
- Note: We are no longer in a scale equivalent world! Standardization required.
- Ridge regressions outperform when LS estimates are highly variable (i.e. large p/n)

Shrinkage: The Lasso

- However, none of $\hat{\beta}_j^R$ are set to zero: large models remain uninterpreable.

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{i,j})^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \mathrm{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2 \tag{15}$$

- Some parameters forced to zero: essentially performing `variable selection`.
- This yields $sparser$ models than ridge regressions, and are easier to interpret.
- When $\lambda = 0$, then the model is equivalent to least squares ...
- When λ becomes sufficiently large, we are left with the null model.
- In contrast, ridge regression will always include all variables.
- λ can be determined through a C.V. based grid procedure.

Dimension Reduction

- Dimension reduction transforms predictors then uses them to fit models.
- Let Z_1, Z_2, ... ,Z_M represent $M < p$ linear combinations of p predictors.

1. Transform the predictors:

$$Z_m = \sum_{j=1}^{p} \phi_{j,m} X_j \tag{16}$$

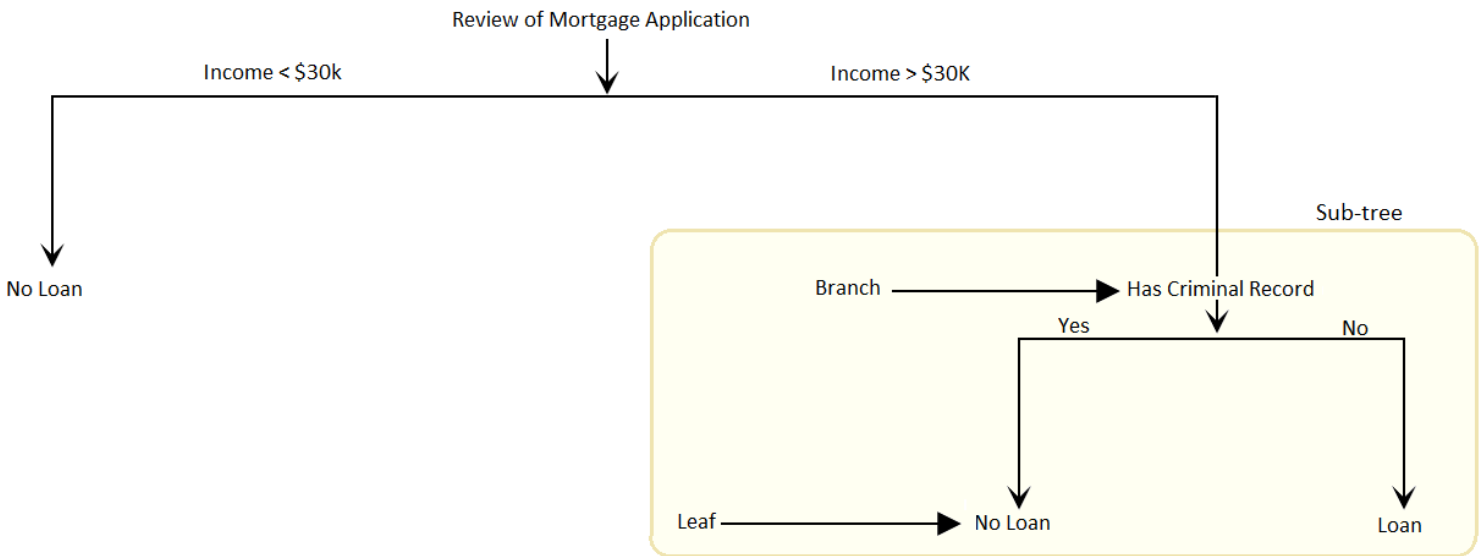
3. Fit the linear model:

$$y_i = \theta_0 + \sum_{j=1}^{p} \theta_m z_{i,m} + \varepsilon_i \tag{17}$$

- The dimension of the problem has been reduced from $p + 1$ to $M + 1$.
- Depending on how we choose $\phi_{1,m}$, ..., $\theta_{p,m}$, we can often outperform LS.

Introduction to Decision Trees

- Tree-based methods involve stratifying/segmenting the predictor space.
- Combining many trees can result in improvements in prediction accuracy.



- The 'internal nodes' which join the leaves are–naturally–called branches.
- The prediction is the mean of the response for training observations in that region.
- Considering all partitions is computationally infeasible: use a top-down, greedy approach: recursive binary splitting.

Introduction to Decision Trees: Cont

- The resulting tree is frequently too complex, over-fitting the data.
- Therefore, grow the full tree, then prune it back to get a subtree.
- Use *cost complexity pruning*: groups of subtrees with tuning parameter (α)
- α controls trade-off between and training data fit.

Algorithm Four: Building a Regression Tree

Step 1: Use recursive binary splitting to grow a large tree.
Step 2. Prune a large tree, obtain a sequence of best subtrees.
Step 3. Cross-validate to choose α, minimizing average error:
    Step 3a. Repeat Steps 1 and 2 on all but the kth fold.
    Step 3b. Evaluate MSE error on data in left-out k-fold.
Step 4: Select the single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ models.

- With classification we can no longer split with RSS (requires Gini coef. or entropy)
- Good: Trees are very easy to explain, and can be displayed graphically.
- Bad: generally poor performance, and sensitive to changes in data.

Bagging

- Bootstrap aggregation ('bagging') reduces high variance of decision trees.
- Build multiple models with re-sampled data and average resulting predictions.
- With B different bootstrapped training sets, train method on each b to get $\hat{f}^b(x)$:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x) \tag{18}$$

- These trees are grown deep, and are not pruned.
- Averaging these B trees reduces the variance
- In the classification realm, the decision function can be a *majority vote*.
- The number of trees B is not a critical parameter with bagging.
- We lose interpretability with bagging, but can still analyze the importance of a predictor through the amount by which the error is decreased due to splits over it.

Random Forests

- Random Forests improve on bagging by de-correlating the trees.
- As before, build a number of trees on bootstrapped training samples.
- Each time we split, use a random sample of m predictors as split candidates.
- Then, use only one one of them, where $m \approx \sqrt{p}$.
- Performance increased by removing the correlated bias between the predictors.
    - A random forest will be helpful when we have multiple correlated predictors.
- On average (p - m)/p of the splits will not even consider the strong predictor.
- If a random forest is built using $m = p$, then this is tantamount to bagging.

Boosting

- Like bagging, boosting can be applied to many statistical learning methods.
- Boosting grows trees sequentially using information from previous trees.
- Unlike a single tree (potentially over-fitting), boosting instead learns slowly.
- Given the current model, we fit a decision tree to the residuals from the model.
- The shrinkage (λ) slows the process down, allowing more shape to the trees.

Algorithm Five: Boosting for Regression Trees

Step 1: Set $\hat{f}(x)$ and $r_i = y_i$ for all i in training set.
Step 2. For $b = 1, 2, \ldots, B$, repeat:
  Step 2a. Fit tree $\hat{f}^b$ with d splits (d+1 terminal nodes).
  Step 2b. Update $\hat{f}$: $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$
  Step 2c. Update residuals: $r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$
Step 3: Output the boosted model: $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$

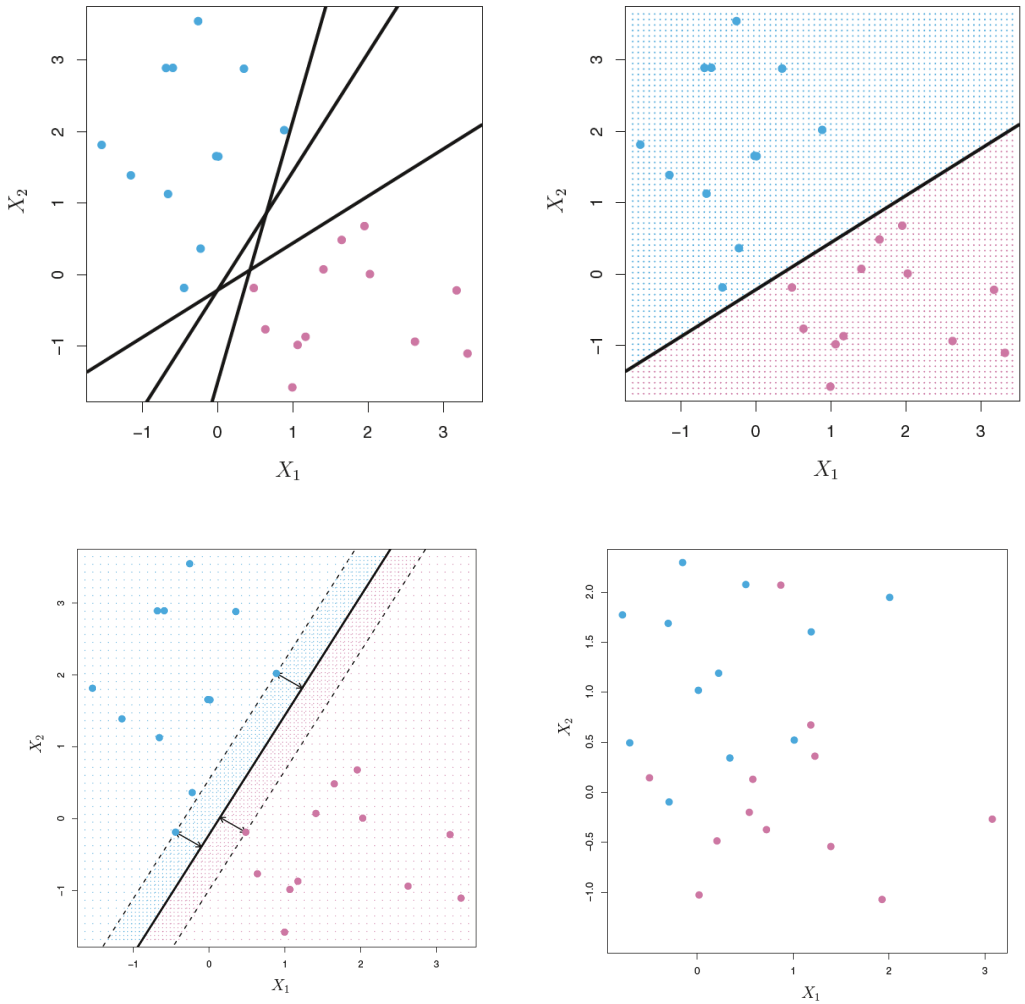
Introduction to SVMs

- Grown in popularity since 1990s, performing well in a variety of settings.
- The SVM is a generalization of a simple and intuitive maximal margin classifier.
- Maximal margin classifier requires the classes be separable by a linear boundary.
- They are based on the concepts of separating hyper-planes.
- In a p-dimensional space, a hyper-plane is a flat affine subspace of dimension $p - 1$.
- For example, in two dimensions, a hyper-plane is defined by the equation:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0 \tag{19}$$

- A natural choice is the maximal margin hyper-plane: separating hyper-plane where margin is largest (farthest minimum distance to training observations).
- This margin represents the mid-line of the widest "slab" that we can insert.
- The generalization of the maximal margin classifier to the non-separable case using soft margins is known as the support vector classifier.

Introducing Hyper-planes, Margins and Support Vectors



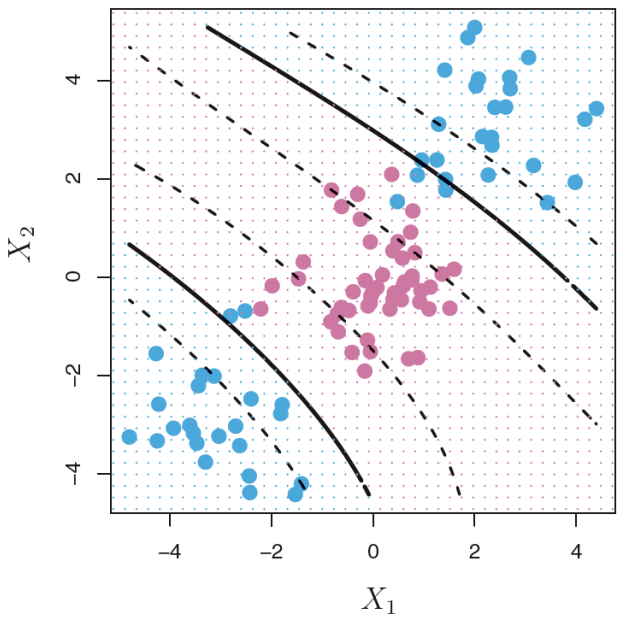
1. Top left subfigure shows that multiple hyper-planes can exist for one separable boundary
2. Top right subfigure shows how one of these separating hyperplanes would classify.
3. Bottom left shows the support vectors which maximize the distance and the maximal margin classifier.
4. Bottom right shows the non-separable case which requires soft margins.

Moving towards Support Vector Classifiers

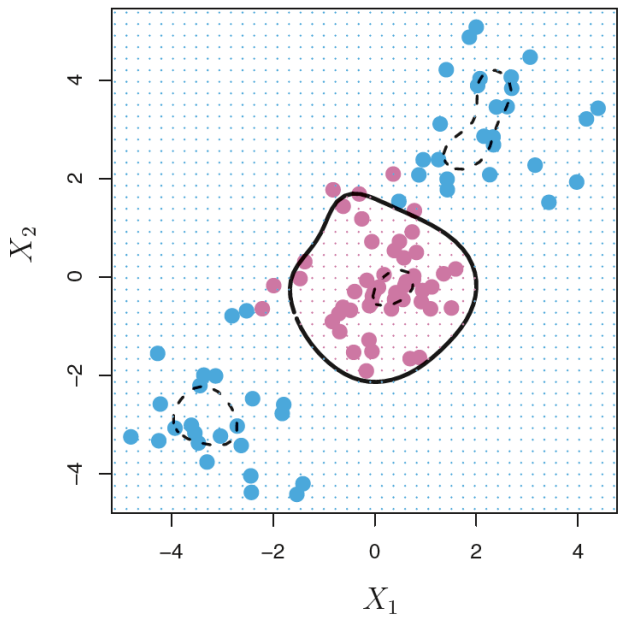
- What about if there is no obvious linear boundary?
- Consider a classifier based on a hyper-plane that doesn't perfectly separate two classes: could we mis-classify a few observations to better classify the rest?
- The support vector classifier (soft margin classifier) does exactly this.
- The margin is soft because it can be violated by some training observations.
- When there is no separating hyper-plane, such a situation is inevitable.
- So-called 'slack variables' within the maximization problem allow individual observations to be on the wrong side of the margin.
- Note: we still seek to make the margin as wide as possible.
- Observations which don't violate the margin don't affect the classifier.
- Changing the position of that observation would not change the classifier at all...:
    - As long as it remains on the correct side of the margin.

Moving towards Support Vector Machines

Polynominal kernel (d=3)



Radial kernel

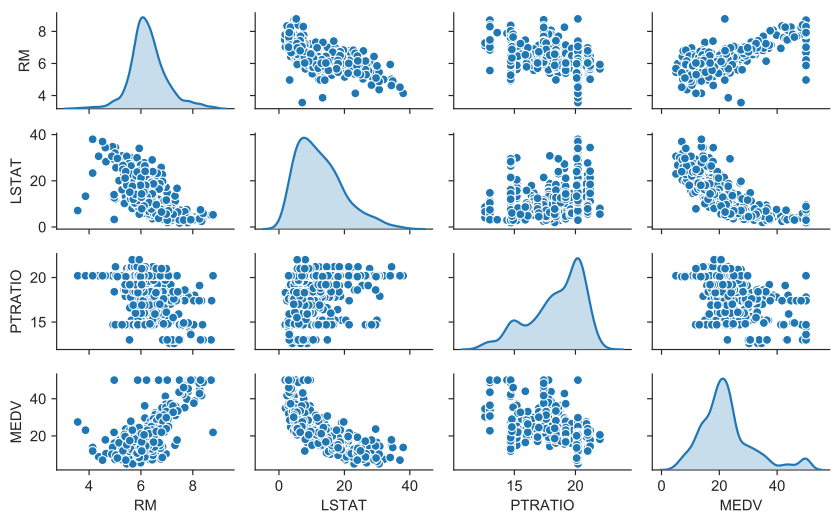


- The previous slide describes a support vector classifier.
- Moving to a support vector machine allows nonlinear decision boundaries.
- This is done by combining a support vector classifier with a non-linear kernel.
- A kernel is a weighting function used in non-parametric estimation techniques.
- Extended to multiple classes with ‘one-verses-one’ and ‘one-verses-all’ classification.

Predicting House Prices: Exploratory Data Analysis

- Lets predict median housing values across census tracts in Boston from a data-set based on Harrison and Rubinfeld (1978).
- Use 5 features: `MEDV`: median value of homes, `RM`: average rooms, `LSTAT`: socioeconomic status (%), `PTRATIO`: pupil/teacher ratio, and a categorical `TOWN`.
- Always begin by examining the data ('exploratory data analysis'):

	RM	LSTAT	PTRATIO	MEDV
Count	506	506	506	506
Mean	6.28	12.65	18.46	22.53
Std	0.70	7.14	2.16	9.20
Min	3.56	1.73	12.60	5.00
25%	5.89	6.95	17.4	17.03
50%	6.21	11.36	19.05	21.20
75%	6.62	16.96	20.20	25.00
max	8.78	37.97	22.00	50.00



Predicting House Prices: Feature Engineering

- Lets engineer a new set of features through discretization (binning).
- Specifically, lets bin the 'PTRATIO' into three categories (low, medium and high):
- '`ptratio_low`' < 16, > 20 '`ptratio_medium`' ≥ 16, and '`ptratio_higher`' < 20.

In [4]:
```
boston_data['PTRATIO_low'] = np.where(boston_data['PTRATIO']<16, 1, 0)
boston_data['PTRATIO_medium'] = np.where((boston_data['PTRATIO']>=16) &
                                         (boston_data['PTRATIO']<=20), 1,0)
boston_data['PTRATIO_high'] = np.where(boston_data['PTRATIO']>20,1,0)
boston_data[['PTRATIO','PTRATIO_low','PTRATIO_medium','PTRATIO_high']].head(3)
```

	PTRATIO	ptratio_low	ptratio_medium	ptratio_high
0	15.3	1	0	0
1	17.8	0	1	0
2	17.8	0	1	0
⋮	⋮	⋮	⋮	⋮

Predicting House Prices: Two models, simple split

- Lets estimate and compare two models with a simple (3:1) sample split.
- First, lets estimate a (very) small model with no binning on PT_RATIO:

```
regr = linear_model.LinearRegression()
regr.fit(X_train[['RM','LSTAT','PTRATIO']], y_train)
y_pred = regr.predict(X_test[['RM','LSTAT','PTRATIO']])
print("Mean squared error of nobins model: %.2f" % mean_squared_error(y_test, y_pred))
print('Variance score of nobins model: %.2f' % r2_score(y_test, y_pred))
```

```
Mean squared error of nobins model: 35.89
Variance score of nobins model: 0.66
```

- Then, lets re-estimate the same model and CV split with PT_Ratio split into bins:

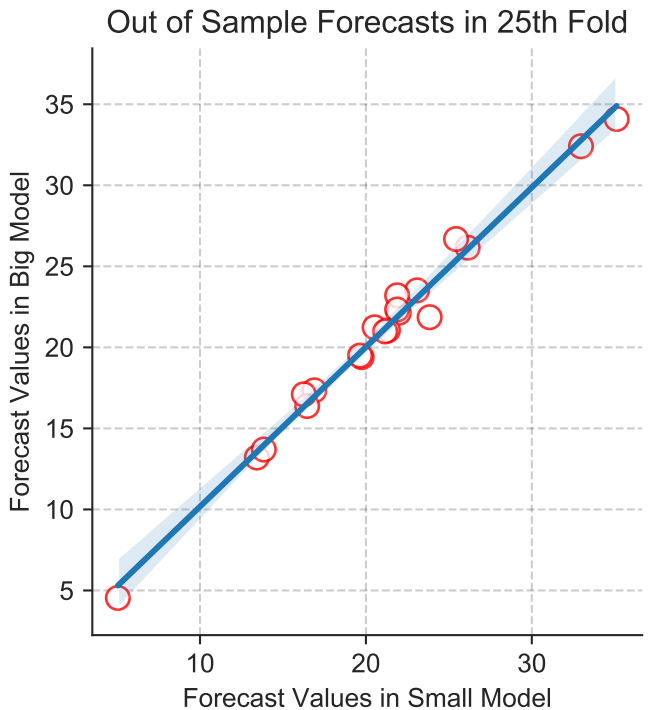
```
regr.fit(X_train.drop(columns=['PTRATIO','MEDV']), y_train)
y_pred = regr.predict(X_test.drop(columns=['PTRATIO','MEDV']))
print("Mean squared error of binned model: %.2f" % mean_squared_error(y_test, y_pred))
print('Variance score of binned model: %.2f' % r2_score(y_test, y_pred))
```

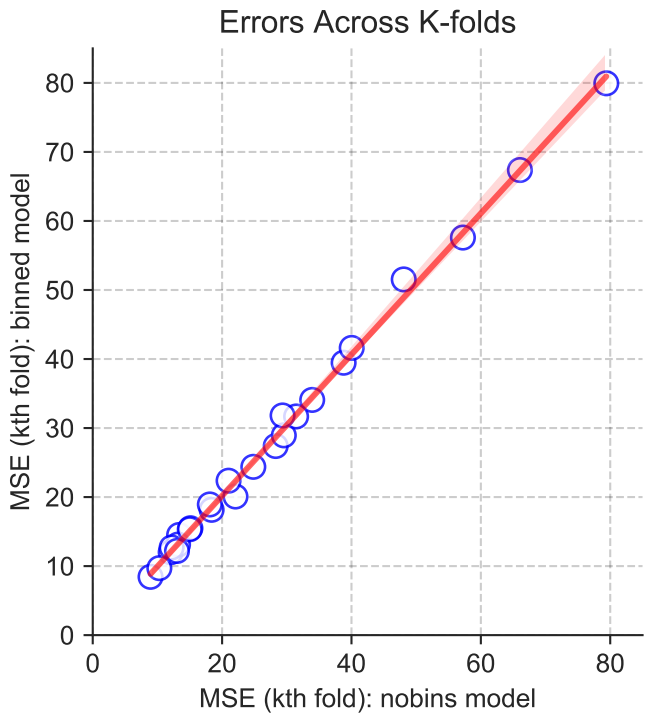
```
Mean squared error of binned model: 36.03
Variance score of binned model: 0.66
```

- What can we learn about which model is better, and by how much?
- How much difference does binning make in this example?

Predicting House Prices: 25-fold Cross Validation

- Lets estimate and compare two models with 25-fold CV in the simple linear model:

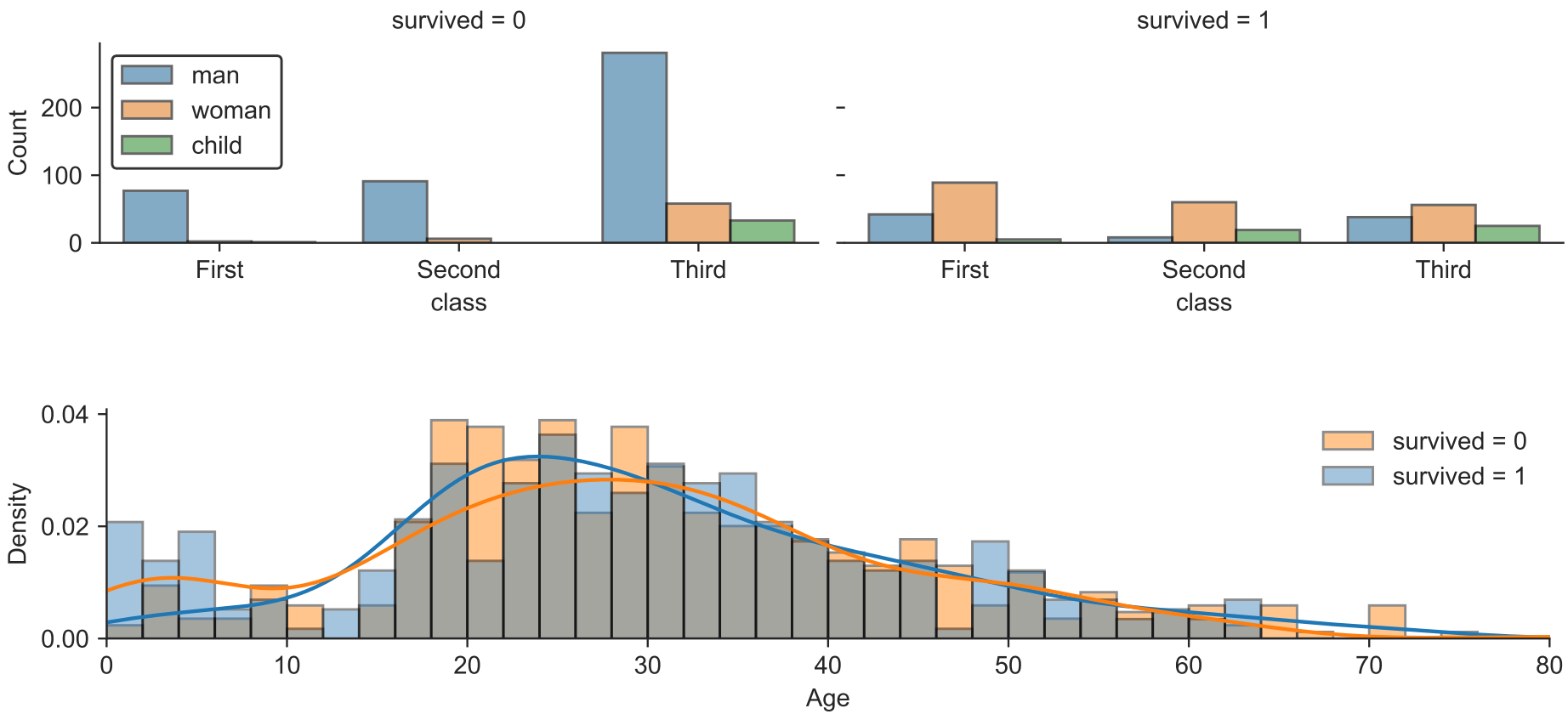




- Which model is better here? (hint: $\beta > 1$ or $\beta < 1$?)

Predicting Titanic Survival: Exploratory Data Analysis

- Example 2: predicting survival on the Titanic (a binary classification problem).
- Lets again begin with some EDA:



Predicting Titanic Survival: Feature Engineering

- Again engineer features and preview the dataframe to be used for prediction:
  1. Bin the age into three categories (young, middle and old).
  2. One-hot encode the cabin class (first, second, third).
  3. Generate a gender dummy (male=0, female=1).

```
titanic['young_age'] = np.where(titanic['age']<18, 1,0)
titanic['middle_age'] = np.where((titanic['age']>=18) &
                                 (titanic['age']<=60), 1,0)
titanic['old_age'] = np.where(titanic['age']>60, 1,0)
titanic = titanic.join(pd.get_dummies(titanic['class']))
titanic['gender'] = np.where(titanic['sex']=='female', 1,0)
titanic[['survived', 'gender', 'young_age',
         'middle_age','old_age','First','Second',
         'Third','sibsp','parch']].head(3)
```

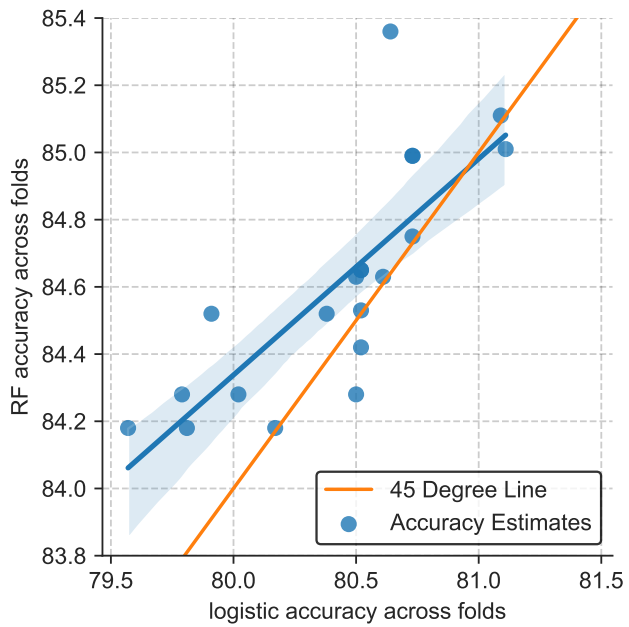
	survived	gender	young_age	middle_age	old_age	First	Second	Third	sibsp	parch
0	0	0	0	1	0	0	0	1	1	0
1	1	1	0	1	0	1	0	0	1	0
2	1	1	0	1	0	0	0	1	0	0

Predicting Titanic Survival: Multiple Models

- Instead of estimating two different specifications, lets instead estimate a bunch of different algorithms: 'Support Vector Machines', 'KNN', 'Logistic Regression', 'Random Forest', 'Naive Bayes', 'Stochastic Gradient Decent'.
- This is an extremely common strategy: Estimate a large model space, search over grids, and then pick the model with the best cross-validation results for production.
- Evaluation here with 'accuracy', although should be exploring confusion matrices, precision and recall. Can you expand the accompanying jupyter notebook to do so?

```
results = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'Stochastic Gradient Decent'],
    'Score': [np.mean(acc_svm), np.mean(acc_knn), np.mean(acc_log),
              np.mean(acc_RF), np.mean(acc_gau), np.mean(acc_sgd)],
    'Std': [np.std(acc_svm), np.std(acc_knn), np.std(acc_log),
            np.std(acc_RF), np.std(acc_gau), np.std(acc_sgd)]})
result_df = results.sort_values(by='Score', ascending=False)
result_df
```

	Model	Score	Std
3	Random Forest	84.5955	0.270804
1	KNN	82.9520	0.495476
2	Logistic Regression	80.6200	0.309548
0	Support Vector Machines	80.2240	0.349276
4	Naive Bayes	78.2020	0.560121
5	Stochastic Gradient Decent	72.2865	8.792160



Conclusion

- The machine learning pipeline and how to approach your own problems:
  1. Obtain data-sets.
  2. Wrangle data into the right shape.
  3. Engineer features.
  4. Define an array of algorithms and specifications.
  5. Estimate and cross-validate.
  6. Evaluate the performance, implement into practice as required.

.

- Things we didn't cover, which a longer class would have done:
  1. We didn't *really* talk about grid searches across parameter/specification space.
  2. We didn't begin to discuss deep learning models.
  3. We didn't begin to talk about big data architectures/infrastructures.
  4. We didn't talk about DAGs and 'causal learning/discovery'