Predictive Analytics Lecture 6

Adam Kapelner

Stat 422/722
at The Wharton School of the University of Pennsylvania

February 21 & 22, 2017

Modeling Framework Refresher

Recall the general regression model:

$$Y = f(x_1, \ldots, x_p) + \mathcal{E}$$

A couple lectures ago, we made the parametric assumption that:

Modeling Framework Refresher

Recall the general regression model:

$$Y = f(x_1, \ldots, x_p) + \mathcal{E}$$

A couple lectures ago, we made the parametric assumption that:

$$Y = s(x_1, \ldots, x_p; \theta_1, \ldots, \theta_\ell) + \tilde{\mathcal{E}}$$

where the $\tilde{\mathcal{E}}$ term now includes the previous $\mathcal{E}$ plus $f - s$, the misspecification error.

Modeling Framework Refresher

Recall the general regression model:

$$Y = f(x_1, \ldots, x_p) + \mathcal{E}$$

A couple lectures ago, we made the parametric assumption that:

$$Y = s(x_1, \ldots, x_p; \theta_1, \ldots, \theta_\ell) + \tilde{\mathcal{E}}$$

where the $\tilde{\mathcal{E}}$ term now includes the previous $\mathcal{E}$ plus $f - s$, the misspecification error. The parametric model s we employed was the linear model and the θ's we called β's:

Modeling Framework Refresher

Recall the general regression model:

$$Y = f(x_1, \ldots, x_p) + \mathcal{E}$$

A couple lectures ago, we made the parametric assumption that:

$$Y = s(x_1, \ldots, x_p; \theta_1, \ldots, \theta_\ell) + \tilde{\mathcal{E}}$$

where the $\tilde{\mathcal{E}}$ term now includes the previous $\mathcal{E}$ plus $f - s$, the misspecification error. The parametric model s we employed was the linear model and the θ's we called β's:

$$Y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \tilde{\mathcal{E}}$$

Last lecture, we started adding interactions and polynomials (as well as other transformations e.g. log which we did not cover).

Modeling Framework Refresher

Recall the general regression model:

$$Y = f(x_1, \ldots, x_p) + \mathcal{E}$$

A couple lectures ago, we made the parametric assumption that:

$$Y = s(x_1, \ldots, x_p; \theta_1, \ldots, \theta_\ell) + \tilde{\mathcal{E}}$$

where the $\tilde{\mathcal{E}}$ term now includes the previous $\mathcal{E}$ plus $f - s$, the misspecification error. The parametric model s we employed was the linear model and the θ's we called β's:

$$Y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \tilde{\mathcal{E}}$$

Last lecture, we started adding interactions and polynomials (as well as other transformations e.g. log which we did not cover). This was a means of "expanding" the feature set "visible" to the model using "derived" features:

Modeling Framework Refresher

Recall the general regression model:

$$Y = f(x_1, \ldots, x_p) + \mathcal{E}$$

A couple lectures ago, we made the parametric assumption that:

$$Y = s(x_1, \ldots, x_p; \theta_1, \ldots, \theta_\ell) + \tilde{\mathcal{E}}$$

where the $\tilde{\mathcal{E}}$ term now includes the previous $\mathcal{E}$ plus $f - s$, the misspecification error. The parametric model s we employed was the linear model and the θ's we called β's:

$$Y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \tilde{\mathcal{E}}$$

Last lecture, we started adding interactions and polynomials (as well as other transformations e.g. log which we did not cover). This was a means of "expanding" the feature set "visible" to the model using "derived" features:

$\{x_1, \ldots, x_p\} \Rightarrow \{x'_1, \ldots, x'_{p'}\}$ where $p' > p$ and maybe much, much greater.

"Non-parametric" Linear Regression

Once we expand this feature set, we can now fit a larger linear model:

$$Y = \beta_0 + \beta_1 x_1' + \ldots\ldots\ldots\ldots\ldots\ldots + \beta_p x_{p'}' + \tilde{\mathcal{E}}$$

Given more degrees of freedom with this expanded feature set allows the linear model to fit more complicated real-world functions.

"Non-parametric" Linear Regression

Once we expand this feature set, we can now fit a larger linear model:

$$Y = \beta_0 + \beta_1 x_1' + \ldots\ldots\ldots\ldots\ldots\ldots + \beta_p x_{p'}' + \tilde{\mathcal{E}}$$

Given more degrees of freedom with this expanded feature set allows the linear model to fit more complicated real-world functions. This is essentially a means of doing non-parametric parametric modeling (it's oxymoronic). It's technically parametric but conceptually it's non-parametric since we don't have our parametric benefits: parsimony, inference nor interpretation. Hopefully $\tilde{\mathcal{E}}$ will be close to $\mathcal{E}$, the irreducible noise.

"Non-parametric" Linear Regression

Once we expand this feature set, we can now fit a larger linear model:

$$Y = \beta_0 + \beta_1 x_1' + \ldots\ldots\ldots\ldots\ldots\ldots + \beta_p x_{p'}' + \tilde{\mathcal{E}}$$

Given more degrees of freedom with this expanded feature set allows the linear model to fit more complicated real-world functions. This is essentially a means of doing non-parametric parametric modeling (it's oxymoronic). It's technically parametric but conceptually it's non-parametric since we don't have our parametric benefits: parsimony, inference nor interpretation. Hopefully $\tilde{\mathcal{E}}$ will be close to $\mathcal{E}$, the irreducible noise.

Back to our problem... we can curb overfitting by ...

"Non-parametric" Linear Regression

Once we expand this feature set, we can now fit a larger linear model:

$$Y = \beta_0 + \beta_1 x_1' + \ldots\ldots\ldots\ldots\ldots\ldots + \beta_p x_{p'}' + \tilde{\mathcal{E}}$$

Given more degrees of freedom with this expanded feature set allows the linear model to fit more complicated real-world functions. This is essentially a means of doing non-parametric parametric modeling (it's oxymoronic). It's technically parametric but conceptually it's non-parametric since we don't have our parametric benefits: parsimony, inference nor interpretation. Hopefully $\tilde{\mathcal{E}}$ will be close to $\mathcal{E}$, the irreducible noise.

Back to our problem... we can curb overfitting by ... using 3-way split oos validation but we need to select good models... how to do so?

"Non-parametric" Linear Regression

Once we expand this feature set, we can now fit a larger linear model:

$$Y = \beta_0 + \beta_1 x_1' + \ldots\ldots\ldots\ldots\ldots\ldots + \beta_p x_{p'}' + \tilde{\mathcal{E}}$$

Given more degrees of freedom with this expanded feature set allows the linear model to fit more complicated real-world functions. This is essentially a means of doing non-parametric parametric modeling (it's oxymoronic). It's technically parametric but conceptually it's non-parametric since we don't have our parametric benefits: parsimony, inference nor interpretation. Hopefully $\tilde{\mathcal{E}}$ will be close to $\mathcal{E}$, the irreducible noise.

Back to our problem... we can curb overfitting by ... using 3-way split oos validation but we need to select good models... how to do so? One approach is termed **subset selection methods**.

Stepwise Regression

First we expand the feature set from $\{x_1, \ldots, x_p\} \Rightarrow \left\{x'_1, \ldots, x'_{p'}\right\}$.

Stepwise Regression

First we expand the feature set from $\{x_1, \ldots, x_p\} \Rightarrow \left\{x'_1, \ldots, x'_{p'}\right\}$. Then we attempt to find the "best" model consisting of a subset of these features. However there are $2^{p'}$ possible models. For $p' = 20$ that's about 1,000,000.

Stepwise Regression

First we expand the feature set from $\{x_1, \ldots, x_p\} \Rightarrow \left\{x'_1, \ldots, x'_{p'}\right\}$. Then we attempt to find the "best" model consisting of a subset of these features. However there are $2^{p'}$ possible models. For $p' = 20$ that's about 1,000,000. So we try to find a model *close* to the optimal using a "heuristic" (a rule of thumb that seems to generally be useful).

Stepwise Regression

First we expand the feature set from $\{x_1, \ldots, x_p\} \Rightarrow \left\{x'_1, \ldots, x'_{p'}\right\}$. Then we attempt to find the "best" model consisting of a subset of these features. However there are $2^{p'}$ possible models. For $p' = 20$ that's about 1,000,000. So we try to find a model *close* to the optimal using a "heuristic" (a rule of thumb that seems to generally be useful).

That heuristic is called **stepwise** model construction.

Stepwise Regression

First we expand the feature set from $\{x_1, \ldots, x_p\} \Rightarrow \left\{x'_1, \ldots, x'_{p'}\right\}$. Then we attempt to find the "best" model consisting of a subset of these features. However there are $2^{p'}$ possible models. For $p' = 20$ that's about 1,000,000. So we try to find a model *close* to the optimal using a "heuristic" (a rule of thumb that seems to generally be useful).

That heuristic is called **stepwise** model construction. We begin with **forward stepwise** model construction:

Stepwise Regression

First we expand the feature set from $\{x_1, \ldots, x_p\} \Rightarrow \left\{x'_1, \ldots, x'_{p'}\right\}$. Then we attempt to find the "best" model consisting of a subset of these features. However there are $2^{p'}$ possible models. For $p' = 20$ that's about 1,000,000. So we try to find a model *close* to the optimal using a "heuristic" (a rule of thumb that seems to generally be useful).

That heuristic is called **stepwise** model construction. We begin with **forward stepwise** model construction:

1. Find the "best" feature from the list of expanded features.

Stepwise Regression

First we expand the feature set from $\{x_1, \ldots, x_p\} \Rightarrow \left\{x'_1, \ldots, x'_{p'}\right\}$. Then we attempt to find the "best" model consisting of a subset of these features. However there are $2^{p'}$ possible models. For $p' = 20$ that's about 1,000,000. So we try to find a model *close* to the optimal using a "heuristic" (a rule of thumb that seems to generally be useful).

That heuristic is called **stepwise** model construction. We begin with **forward stepwise** model construction:

1. Find the "best" feature from the list of expanded features.
2. Find the "next best" feature from the remaining expanded features.

Stepwise Regression

First we expand the feature set from $\{x_1, \ldots, x_p\} \Rightarrow \left\{x'_1, \ldots, x'_{p'}\right\}$. Then we attempt to find the "best" model consisting of a subset of these features. However there are $2^{p'}$ possible models. For $p' = 20$ that's about 1,000,000. So we try to find a model *close* to the optimal using a "heuristic" (a rule of thumb that seems to generally be useful).

That heuristic is called **stepwise** model construction. We begin with **forward stepwise** model construction:

1. Find the "best" feature from the list of expanded features.
2. Find the "next best" feature from the remaining expanded features.
3. Repeat step 2 until you believe you are overfitting.

Stepwise Regression

First we expand the feature set from $\{x_1, \ldots, x_p\} \Rightarrow \left\{x'_1, \ldots, x'_{p'}\right\}$. Then we attempt to find the "best" model consisting of a subset of these features. However there are $2^{p'}$ possible models. For $p' = 20$ that's about 1,000,000. So we try to find a model *close* to the optimal using a "heuristic" (a rule of thumb that seems to generally be useful).

That heuristic is called **stepwise** model construction. We begin with **forward stepwise** model construction:

1. Find the "best" feature from the list of expanded features.
2. Find the "next best" feature from the remaining expanded features.
3. Repeat step 2 until you believe you are overfitting.

Estimating Overfitting (again)

If you choose the feature to give you the best in-sample R^2, you will eventually take all the features (until $n = p + 1$) and you will get $R^2 = 100\%$. We need a metric to tell us when we may be overfitting and halt at that moment.

Estimating Overfitting (again)

If you choose the feature to give you the best in-sample R^2, you will eventually take all the features (until $n = p + 1$) and you will get $R^2 = 100\%$. We need a metric to tell us when we may be overfitting and halt at that moment. Here are a few:

1. oos RMSE (keep a holdout set and quit when this starts increasing)

Estimating Overfitting (again)

If you choose the feature to give you the best in-sample R^2, you will eventually take all the features (until $n = p + 1$) and you will get $R^2 = 100\%$. We need a metric to tell us when we may be overfitting and halt at that moment. Here are a few:

1. oos RMSE (keep a holdout set and quit when this starts increasing)
2. Only include a variable if its t stat (or partial F stat) is significant

Estimating Overfitting (again)

If you choose the feature to give you the best in-sample R^2, you will eventually take all the features (until $n = p + 1$) and you will get $R^2 = 100\%$. We need a metric to tell us when we may be overfitting and halt at that moment. Here are a few:

1. oos RMSE (keep a holdout set and quit when this starts increasing)
2. Only include a variable if its t stat (or partial F stat) is significant
3. Use *AICc*.

$$-AIC = 2\ell\left(\hat{\beta}; \boldsymbol{y}, \boldsymbol{x}\right) - 2p$$

The first component (the log-likelihood) represents in-sample fit.

Estimating Overfitting (again)

If you choose the feature to give you the best in-sample R^2, you will eventually take all the features (until $n = p + 1$) and you will get $R^2 = 100\%$. We need a metric to tell us when we may be overfitting and halt at that moment. Here are a few:

1. oos RMSE (keep a holdout set and quit when this starts increasing)
2. Only include a variable if its t stat (or partial F stat) is significant
3. Use *AICc*.

$$-AIC = 2\ell\left(\hat{\beta}; \boldsymbol{y}, \boldsymbol{x}\right) - 2p$$

The first component (the log-likelihood) represents in-sample fit. $\ell()$ is like R^2 though... as the fit gets closer to the points, the likelihood goes to 1 (and the log likelihood goes to 0).

Estimating Overfitting (again)

If you choose the feature to give you the best in-sample R^2, you will eventually take all the features (until $n = p + 1$) and you will get $R^2 = 100\%$. We need a metric to tell us when we may be overfitting and halt at that moment. Here are a few:

1. oos RMSE (keep a holdout set and quit when this starts increasing)
2. Only include a variable if its t stat (or partial F stat) is significant
3. Use $AICc$.

$$-AIC = 2\ell\left(\hat{\beta}; \boldsymbol{y}, \boldsymbol{x}\right) - 2p$$

The first component (the log-likelihood) represents in-sample fit. $\ell()$ is like R^2 though... as the fit gets closer to the points, the likelihood goes to 1 (and the log likelihood goes to 0). The $2p$ term is a reality check. If you have more features, you are going to overfit. So each additional feature must be justified in terms of the increase in log-likelihood.

Estimating Overfitting (again)

If you choose the feature to give you the best in-sample R^2, you will eventually take all the features (until $n = p + 1$) and you will get $R^2 = 100\%$. We need a metric to tell us when we may be overfitting and halt at that moment. Here are a few:

1. oos RMSE (keep a holdout set and quit when this starts increasing)
2. Only include a variable if its t stat (or partial F stat) is significant
3. Use $AICc$.

$$-AIC = 2\ell\left(\hat{\beta}; \boldsymbol{y}, \boldsymbol{x}\right) - 2p$$

The first component (the log-likelihood) represents in-sample fit. $\ell()$ is like R^2 though... as the fit gets closer to the points, the likelihood goes to 1 (and the log likelihood goes to 0). The $2p$ term is a reality check. If you have more features, you are going to overfit. So each additional feature must be justified in terms of the increase in log-likelihood. Thus, good models maximize $-AIC$ (i.e. minimize AIC).

AICc for linear models

For linear regression under OLS, we can calculate the log-likelihood explicitly (we approximately did this in Lecture 2) to obtain:

$$AIC = n \ln\left(RMSE^2\right) + 2p$$

AICc for linear models

For linear regression under OLS, we can calculate the log-likelihood explicitly (we approximately did this in Lecture 2) to obtain:

$$AIC = n \ln\left(RMSE^2\right) + 2p$$

So once again, we want this to be small. If we decrease *RMSE* by adding a feature, it needs to counteract an increase of 2 by $p \to p + 1$. If it can't, we're probably overfitting.

AICc for linear models

For linear regression under OLS, we can calculate the log-likelihood explicitly (we approximately did this in Lecture 2) to obtain:

$$AIC = n \ln\left(RMSE^2\right) + 2p$$

So once again, we want this to be small. If we decrease *RMSE* by adding a feature, it needs to counteract an increase of 2 by $p \to p+1$. If it can't, we're probably overfitting. *AIC* works well with large sample sizes. For small sample sizes, we use a corrected version *AICc* defined as:

$$AICc = AIC + \frac{2p(p+1)}{n-p-1}$$

Needless to say, this is all approximate since we are assuming OLS and a whole bunch of other things (beyond scope of course).

AICc for linear models

For linear regression under OLS, we can calculate the log-likelihood explicitly (we approximately did this in Lecture 2) to obtain:

$$AIC = n \ln\left(RMSE^2\right) + 2p$$

So once again, we want this to be small. If we decrease *RMSE* by adding a feature, it needs to counteract an increase of 2 by $p \to p+1$. If it can't, we're probably overfitting. *AIC* works well with large sample sizes. For small sample sizes, we use a corrected version *AICc* defined as:

$$AICc = AIC + \frac{2p(p+1)}{n-p-1}$$

Needless to say, this is all approximate since we are assuming OLS and a whole bunch of other things (beyond scope of course). Note: there are also BIC and Mallow's C_p which are similar metrics, but we will not cover them.

Stepwise Linear & Logistic Regr. Demos

- white wine on AICc

Stepwise Linear & Logistic Regr. Demos

- white wine on AICc
- white wine oos validation R^2 and test

Stepwise Linear & Logistic Regr. Demos

- white wine on AICc
- white wine oos validation R^2 and test
- white wine $K-$fold on AICc (why is this not a great idea?)
- telecom all 1st order interactions with oos validation on AICc
- telecom all 1st order interactions with oos validation on min logistic R^2

More About Stepwise Linear Regression

Backward selection begins with all features and then deletes one for each step until no more can justifiably be cut out.

More About Stepwise Linear Regression

Backward selection begins with all features and then deletes one for each step until no more can justifiably be cut out. Backward selection has a major weakness: it cannot be run on dataframes where the extended feature set is more than the number of rows (only forward or forward with mixed works there).

More About Stepwise Linear Regression

Backward selection begins with all features and then deletes one for each step until no more can justifiably be cut out. Backward selection has a major weakness: it cannot be run on dataframes where the extended feature set is more than the number of rows (only forward or forward with mixed works there). **Mixed selection** begins with either none or all and then looks for both good additions and good subtractions.

More About Stepwise Linear Regression

Backward selection begins with all features and then deletes one for each step until no more can justifiably be cut out. Backward selection has a major weakness: it cannot be run on dataframes where the extended feature set is more than the number of rows (only forward or forward with mixed works there). **Mixed selection** begins with either none or all and then looks for both good additions and good subtractions.

Simple case where stepwise doesn't work?

More About Stepwise Linear Regression

Backward selection begins with all features and then deletes one for each step until no more can justifiably be cut out. Backward selection has a major weakness: it cannot be run on dataframes where the extended feature set is more than the number of rows (only forward or forward with mixed works there). **Mixed selection** begins with either none or all and then looks for both good additions and good subtractions.

Simple case where stepwise doesn't work? How about three features where x_1 is most correlated but x_2 and x_3 together are the best model but there is high collinearity between x_2 and x_3? What happens?

More About Stepwise Linear Regression

Backward selection begins with all features and then deletes one for each step until no more can justifiably be cut out. Backward selection has a major weakness: it cannot be run on dataframes where the extended feature set is more than the number of rows (only forward or forward with mixed works there). **Mixed selection** begins with either none or all and then looks for both good additions and good subtractions.

Simple case where stepwise doesn't work? How about three features where x_1 is most correlated but x_2 and x_3 together are the best model but there is high collinearity between x_2 and x_3? What happens? Forward: the model enters x_1 and then x_2 but it doesn't see x_3 as a worthy addition. Backward: the model can nuke x_2 or x_3 since its p-value or F test is poor.

Conceptual Review of What We Just Did

Once again,

$$Y = f(x_1, \ldots, x_p) + \mathcal{E}$$

and we made the parametric assumption that:

Conceptual Review of What We Just Did

Once again,

$$Y = f(x_1, \ldots, x_p) + \mathcal{E}$$

and we made the parametric assumption that:

$$Y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \tilde{\mathcal{E}}$$

then we take our "raw" features and create "derived features"

$\{x_1, \ldots, x_p\} \Rightarrow \{x'_1, \ldots, x'_{p'}\}$ where $p' > p$ and maybe much, much greater.

then we allowed for a large linear model:

$$Y = \beta_0 + \beta_1 x'_1 + \ldots\ldots\ldots\ldots\ldots\ldots + \beta_p x'_{p'} + \tilde{\mathcal{E}}$$

of which we took a subset by "stepping through" and reaching a local optimum:

$$Y = \beta_0 + \beta_1 x'_{(1)} + \ldots\ldots\ldots\ldots + \beta_p x'_{(p')} + \tilde{\mathcal{E}}$$

Binning

Our derived predictors

$$\{x_1, \ldots, x_p\} \Rightarrow \{x'_1, \ldots, x'_{p'}\}$$

still may not be that "flexible" as "bases" for the $\hat{f}$ (whiteboard demo).

Binning

Our derived predictors

$$\{x_1, \ldots, x_p\} \Rightarrow \{x'_1, \ldots, x'_{p'}\}$$

still may not be that "flexible" as "bases" for the $\hat{f}$ (whiteboard demo).

What if we derived predictors that split up the space of raw predictors into tiny regions. In one dimension, we can make some new derived variables:

$$\mathbb{1}_{x\in[0,1]}, \mathbb{1}_{x\in[1,2]}, \mathbb{1}_{x\in[2,3]}, \mathbb{1}_{x\in[3,4]}, \ldots \mathbb{1}_{x\in[9,10]}$$

and assign a different β parameter (fit to be the average y) for each of these 10. This is called "binning" and allows for flexible, non-parametric fits. Why?

Binning

Our derived predictors

$$\{x_1, \ldots, x_p\} \Rightarrow \{x'_1, \ldots, x'_{p'}\}$$

still may not be that "flexible" as "bases" for the $\hat{f}$ (whiteboard demo).

What if we derived predictors that split up the space of raw predictors into tiny regions. In one dimension, we can make some new derived variables:

$$\mathbb{1}_{x\in[0,1]}, \mathbb{1}_{x\in[1,2]}, \mathbb{1}_{x\in[2,3]}, \mathbb{1}_{x\in[3,4]}, \ldots \mathbb{1}_{x\in[9,10]}$$

and assign a different β parameter (fit to be the average y) for each of these 10. This is called "binning" and allows for flexible, non-parametric fits. Why? Did you do any binning on the project?

Does Binning Breakdown?

In multiple dimensions, you can get nice bins by doing interactions between each dimensions' bins. For example in two dimensions, you can bin both into $B = 10$ bins. Crossing the bins makes 100 square bins. Each square gets its own β parameter (fit to be the average y).

Does Binning Breakdown?

In multiple dimensions, you can get nice bins by doing interactions between each dimensions' bins. For example in two dimensions, you can bin both into $B = 10$ bins. Crossing the bins makes 100 square bins. Each square gets its own β parameter (fit to be the average y).

In the white wine data, we have 11 predictors and $B = 10$ bins per continuous predictor, that's $B^{11} = 100$ billion "hyper-square" bins. Problem?

Does Binning Breakdown?

In multiple dimensions, you can get nice bins by doing interactions between each dimensions' bins. For example in two dimensions, you can bin both into $B = 10$ bins. Crossing the bins makes 100 square bins. Each square gets its own β parameter (fit to be the average y).

In the white wine data, we have 11 predictors and $B = 10$ bins per continuous predictor, that's $B^{11} = 100$ billion "hyper-square" bins. Problem? (1) $n > p$ and you can't use the linear model,

Does Binning Breakdown?

In multiple dimensions, you can get nice bins by doing interactions between each dimensions' bins. For example in two dimensions, you can bin both into $B = 10$ bins. Crossing the bins makes 100 square bins. Each square gets its own β parameter (fit to be the average y).

In the white wine data, we have 11 predictors and $B = 10$ bins per continuous predictor, that's $B^{11} = 100$ billion "hyper-square" bins. Problem? (1) $n > p$ and you can't use the linear model, (2) most bins are empty how do you take the average of nothing?) and (3) a lot of bins will have similar averages — useless bins.

Better Bins?

Instead of just binning each predictor into B bins, why not try to make "custom" or "smart" bins in the regions of f that give the **most** increase in fit (i.e. higher R^2, lower SSE or $RMSE$). So you "split" the data into bins that are now "hyper-rectangle" shaped.

Better Bins?

Instead of just binning each predictor into B bins, why not try to make "custom" or "smart" bins in the regions of f that give the **most** increase in fit (i.e. higher R^2, lower SSE or $RMSE$). So you "split" the data into bins that are now "hyper-rectangle" shaped.

Demo JMP white wine.

Better Bins?

Instead of just binning each predictor into B bins, why not try to make "custom" or "smart" bins in the regions of f that give the **most** increase in fit (i.e. higher R^2, lower SSE or $RMSE$). So you "split" the data into bins that are now "hyper-rectangle" shaped.

Demo JMP white wine. Within each hyperrectangle ("split" or "partition"), you can split again by taking the best split, and you can keep going. Because the splits are **binary** (i.e. you split one space into two spaces), and they are hierarchical (you can trace the splits back generations), they look like "trees".

Better Bins?

Instead of just binning each predictor into B bins, why not try to make "custom" or "smart" bins in the regions of f that give the **most** increase in fit (i.e. higher R^2, lower SSE or $RMSE$). So you "split" the data into bins that are now "hyper-rectangle" shaped.

Demo JMP white wine. Within each hyperrectangle ("split" or "partition"), you can split again by taking the best split, and you can keep going. Because the splits are **binary** (i.e. you split one space into two spaces), and they are hierarchical (you can trace the splits back generations), they look like "trees". JMP.

Better Bins?

Instead of just binning each predictor into B bins, why not try to make "custom" or "smart" bins in the regions of f that give the **most** increase in fit (i.e. higher R^2, lower SSE or $RMSE$). So you "split" the data into bins that are now "hyper-rectangle" shaped.

Demo JMP white wine. Within each hyperrectangle ("split" or "partition"), you can split again by taking the best split, and you can keep going. Because the splits are **binary** (i.e. you split one space into two spaces), and they are hierarchical (you can trace the splits back generations), they look like "trees". JMP. How to predict?

Better Bins?

Instead of just binning each predictor into B bins, why not try to make "custom" or "smart" bins in the regions of f that give the **most** increase in fit (i.e. higher R^2, lower SSE or $RMSE$). So you "split" the data into bins that are now "hyper-rectangle" shaped.

Demo JMP white wine. Within each hyperrectangle ("split" or "partition"), you can split again by taking the best split, and you can keep going. Because the splits are **binary** (i.e. you split one space into two spaces), and they are hierarchical (you can trace the splits back generations), they look like "trees". JMP. How to predict? "Drop the observation down "and follow instructions all the way until the end.

Better Bins?

Instead of just binning each predictor into B bins, why not try to make "custom" or "smart" bins in the regions of f that give the **most** increase in fit (i.e. higher R^2, lower SSE or $RMSE$). So you "split" the data into bins that are now "hyper-rectangle" shaped.

Demo JMP white wine. Within each hyperrectangle ("split" or "partition"), you can split again by taking the best split, and you can keep going. Because the splits are **binary** (i.e. you split one space into two spaces), and they are hierarchical (you can trace the splits back generations), they look like "trees". JMP. How to predict? "Drop the observation down "and follow instructions all the way until the end. Does this make them interpretable?

Better Bins?

Instead of just binning each predictor into B bins, why not try to make "custom" or "smart" bins in the regions of f that give the **most** increase in fit (i.e. higher R^2, lower SSE or $RMSE$). So you "split" the data into bins that are now "hyper-rectangle" shaped.

Demo JMP white wine. Within each hyperrectangle ("split" or "partition"), you can split again by taking the best split, and you can keep going. Because the splits are **binary** (i.e. you split one space into two spaces), and they are hierarchical (you can trace the splits back generations), they look like "trees". JMP. How to predict? "Drop the observation down "and follow instructions all the way until the end. Does this make them interpretable? YES. Simple?

Better Bins?

Instead of just binning each predictor into B bins, why not try to make "custom" or "smart" bins in the regions of f that give the **most** increase in fit (i.e. higher R^2, lower SSE or $RMSE$). So you "split" the data into bins that are now "hyper-rectangle" shaped.

Demo JMP white wine. Within each hyperrectangle ("split" or "partition"), you can split again by taking the best split, and you can keep going. Because the splits are **binary** (i.e. you split one space into two spaces), and they are hierarchical (you can trace the splits back generations), they look like "trees". JMP. How to predict? "Drop the observation down "and follow instructions all the way until the end. Does this make them interpretable? YES. Simple? Maybe if they are not too complex.

Decision Trees

These are known as **decision trees** since you can imagine examine when you predict, you follow the "decisions" (i.e. the **split rules**). There is some terms of anatomy to know:

- The top is the "root node"
- Nodes that split have "children" and are "inner nodes" or "split nodes".
- Nodes that do not split are called "leaves" or "terminal nodes".
- "Depth" indicates the maximum number of generations in the tree (the root has zero depth).
- Nodes that have a split must contain a splitting rule e.g. $x_3 < 14.56$ (or equivalently $x_3 \geq 14.56$) and x_3 is called the "split variable" and 14.56 is called the "split value".

Decision Trees

These are known as **decision trees** since you can imagine examine when you predict, you follow the "decisions" (i.e. the **split rules**). There is some terms of anatomy to know:

- The top is the "root node"
- Nodes that split have "children" and are "inner nodes" or "split nodes".
- Nodes that do not split are called "leaves" or "terminal nodes".
- "Depth" indicates the maximum number of generations in the tree (the root has zero depth).
- Nodes that have a split must contain a splitting rule e.g. $x_3 < 14.56$ (or equivalently $x_3 \geq 14.56$) and x_3 is called the "split variable" and 14.56 is called the "split value".

For the purposes of this class, there are two types of decision trees.

Decision Trees

These are known as **decision trees** since you can imagine examine when you predict, you follow the "decisions" (i.e. the **split rules**). There is some terms of anatomy to know:

- The top is the "root node"
- Nodes that split have "children" and are "inner nodes" or "split nodes".
- Nodes that do not split are called "leaves" or "terminal nodes".
- "Depth" indicates the maximum number of generations in the tree (the root has zero depth).
- Nodes that have a split must contain a splitting rule e.g. $x_3 < 14.56$ (or equivalently $x_3 \geq 14.56$) and x_3 is called the "split variable" and 14.56 is called the "split value".

For the purposes of this class, there are two types of decision trees.

- Regression (for continous responses)
- Classification (for categorical responses)

Decision Trees

These are known as **decision trees** since you can imagine examine when you predict, you follow the "decisions" (i.e. the **split rules**). There is some terms of anatomy to know:

- The top is the "root node"
- Nodes that split have "children" and are "inner nodes" or "split nodes".
- Nodes that do not split are called "leaves" or "terminal nodes".
- "Depth" indicates the maximum number of generations in the tree (the root has zero depth).
- Nodes that have a split must contain a splitting rule e.g. $x_3 < 14.56$ (or equivalently $x_3 \geq 14.56$) and x_3 is called the "split variable" and 14.56 is called the "split value".

For the purposes of this class, there are two types of decision trees.

- Regression (for continous responses)
- Classification (for categorical responses)

Making Splits

There are basically three things to decide in the decision tree algorithm:

- How to make the split? What metric to optimize? What splits to check?

Making Splits

There are basically three things to decide in the decision tree algorithm:

- How to make the split? What metric to optimize? What splits to check?
- Leaf assignment? What should the $\hat{y}$ be if the observation "falls" into a given terminal node?
- When to stop splitting? Should we split all the way down so each "terminal" partition (node) has one data point?

Making Splits

There are basically three things to decide in the decision tree algorithm:

- How to make the split? What metric to optimize? What splits to check?
- Leaf assignment? What should the $\hat{y}$ be if the observation "falls" into a given terminal node?
- When to stop splitting? Should we split all the way down so each "terminal" partition (node) has one data point?

The split rule is determined by...

Making Splits

There are basically three things to decide in the decision tree algorithm:

- How to make the split? What metric to optimize? What splits to check?
- Leaf assignment? What should the $\hat{y}$ be if the observation "falls" into a given terminal node?
- When to stop splitting? Should we split all the way down so each "terminal" partition (node) has one data point?

The split rule is determined by...

Look at all possible splits ($\approx n \times p$ splits) **greedily** and take the minimum total SSE for continuous and total entropy for categorical. JMP is non-standard; it takes the maximum log worth (which is something I've never heard of) for continuous and the maximum log likelihood for categorical. My bet is the performance will be similar as it's really the same concept: get the best, most homogeneous, split!

Leaf Assignment & Stopping / Pruning Rule

The leaf assignment is determined by...

Leaf Assignment & Stopping / Pruning Rule

The leaf assignment is determined by...

The sample average $\bar{y}$ among observations in the node for continuous and the most-represented (modal) class in the node for categorical.

Leaf Assignment & Stopping / Pruning Rule

The leaf assignment is determined by...

The sample average $\bar{y}$ among observations in the node for continuous and the most-represented (modal) class in the node for categorical.

The rule that controls whether to stop splitting (and ship the tree) is...

Leaf Assignment & Stopping / Pruning Rule

The leaf assignment is determined by...

The sample average $\bar{y}$ among observations in the node for continuous and the most-represented (modal) class in the node for categorical.

The rule that controls whether to stop splitting (and ship the tree) is...

There is no standard stopping / pruning rule. JMP stops when the next 10 splits below does not create a better cross-validated R^2. Standard software usually allows for a minimum leaf size or allows you to keep a oos validation set.

Leaf Assignment & Stopping / Pruning Rule

The leaf assignment is determined by...

The sample average $\bar{y}$ among observations in the node for continuous and the most-represented (modal) class in the node for categorical.

The rule that controls whether to stop splitting (and ship the tree) is...

There is no standard stopping / pruning rule. JMP stops when the next 10 splits below does not create a better cross-validated R^2. Standard software usually allows for a minimum leaf size or allows you to keep a oos validation set.

What would happen if there is no stopping rule?

Leaf Assignment & Stopping / Pruning Rule

The leaf assignment is determined by...

The sample average $\bar{y}$ among observations in the node for continuous and the most-represented (modal) class in the node for categorical.

The rule that controls whether to stop splitting (and ship the tree) is...

There is no standard stopping / pruning rule. JMP stops when the next 10 splits below does not create a better cross-validated R^2. Standard software usually allows for a minimum leaf size or allows you to keep a oos validation set.

What would happen if there is no stopping rule? It would fit until there is one point in each node i.e. ...

Leaf Assignment & Stopping / Pruning Rule

The leaf assignment is determined by...

The sample average $\bar{y}$ among observations in the node for continuous and the most-represented (modal) class in the node for categorical.

The rule that controls whether to stop splitting (and ship the tree) is...

There is no standard stopping / pruning rule. JMP stops when the next 10 splits below does not create a better cross-validated R^2. Standard software usually allows for a minimum leaf size or allows you to keep a oos validation set.

What would happen if there is no stopping rule? It would fit until there is one point in each node i.e. ... it would very much overfit $\mathcal{E}$.

Leaf Assignment & Stopping / Pruning Rule

The leaf assignment is determined by...

The sample average $\bar{y}$ among observations in the node for continuous and the most-represented (modal) class in the node for categorical.

The rule that controls whether to stop splitting (and ship the tree) is...

There is no standard stopping / pruning rule. JMP stops when the next 10 splits below does not create a better cross-validated R^2. Standard software usually allows for a minimum leaf size or allows you to keep a oos validation set.

What would happen if there is no stopping rule? It would fit until there is one point in each node i.e. ... it would very much overfit $\mathcal{E}$.

More on interpretation: AND rules?

Leaf Assignment & Stopping / Pruning Rule

The leaf assignment is determined by...

The sample average $\bar{y}$ among observations in the node for continuous and the most-represented (modal) class in the node for categorical.

The rule that controls whether to stop splitting (and ship the tree) is...

There is no standard stopping / pruning rule. JMP stops when the next 10 splits below does not create a better cross-validated R^2. Standard software usually allows for a minimum leaf size or allows you to keep a oos validation set.

What would happen if there is no stopping rule? It would fit until there is one point in each node i.e. ... it would very much overfit $\mathcal{E}$.

More on interpretation: AND rules? Overall effect of y on x?

Leaf Assignment & Stopping / Pruning Rule

The leaf assignment is determined by...

The sample average $\bar{y}$ among observations in the node for continuous and the most-represented (modal) class in the node for categorical.

The rule that controls whether to stop splitting (and ship the tree) is...

There is no standard stopping / pruning rule. JMP stops when the next 10 splits below does not create a better cross-validated R^2. Standard software usually allows for a minimum leaf size or allows you to keep a oos validation set.

What would happen if there is no stopping rule? It would fit until there is one point in each node i.e. ... it would very much overfit $\mathcal{E}$.

More on interpretation: AND rules? Overall effect of y on x? No ... not so clear. Categorical variable splitting?

Leaf Assignment & Stopping / Pruning Rule

The leaf assignment is determined by...

The sample average $\bar{y}$ among observations in the node for continuous and the most-represented (modal) class in the node for categorical.

The rule that controls whether to stop splitting (and ship the tree) is...

There is no standard stopping / pruning rule. JMP stops when the next 10 splits below does not create a better cross-validated R^2. Standard software usually allows for a minimum leaf size or allows you to keep a oos validation set.

What would happen if there is no stopping rule? It would fit until there is one point in each node i.e. ... it would very much overfit $\mathcal{E}$.

More on interpretation: AND rules? Overall effect of y on x? No ... not so clear. Categorical variable splitting? Very nice interpretation.

Advantages of Decision Trees

- Automatically knows which variables to include and which to not include — if they help performance, include otherwise not. Linear models include everything or try to ration during a stepwise procedure.

Advantages of Decision Trees

- Automatically knows which variables to include and which to not include — if they help performance, include otherwise not. Linear models include everything or try to ration during a stepwise procedure.
- Easily models curves and non-linearities

Advantages of Decision Trees

- Automatically knows which variables to include and which to not include — if they help performance, include otherwise not. Linear models include everything or try to ration during a stepwise procedure.
- Easily models curves and non-linearities
- Easily models interactions due to the AND intersection from parent node to child node

Advantages of Decision Trees

- Automatically knows which variables to include and which to not include — if they help performance, include otherwise not. Linear models include everything or try to ration during a stepwise procedure.
- Easily models curves and non-linearities
- Easily models interactions due to the AND intersection from parent node to child node
- Thus it is non-parametric. However...

Advantages of Decision Trees

- Automatically knows which variables to include and which to not include — if they help performance, include otherwise not. Linear models include everything or try to ration during a stepwise procedure.
- Easily models curves and non-linearities
- Easily models interactions due to the AND intersection from parent node to child node
- Thus it is non-parametric. However... Low-depth trees have both interpretability and parsimony! (but no inference possible to my knowledge) In fact, the interpretation in trees likely are more similar to how we create models. If income is low and down payment is low ... a default is more likely.

Advantages of Decision Trees

- Automatically knows which variables to include and which to not include — if they help performance, include otherwise not. Linear models include everything or try to ration during a stepwise procedure.
- Easily models curves and non-linearities
- Easily models interactions due to the AND intersection from parent node to child node
- Thus it is non-parametric. However... Low-depth trees have both interpretability and parsimony! (but no inference possible to my knowledge) In fact, the interpretation in trees likely are more similar to how we create models. If income is low and down payment is low ... a default is more likely.
- You no longer need to worry about binning! The tree will create the bins for you via splitting.

Advantages of Decision Trees

- Automatically knows which variables to include and which to not include — if they help performance, include otherwise not. Linear models include everything or try to ration during a stepwise procedure.
- Easily models curves and non-linearities
- Easily models interactions due to the AND intersection from parent node to child node
- Thus it is non-parametric. However... Low-depth trees have both interpretability and parsimony! (but no inference possible to my knowledge) In fact, the interpretation in trees likely are more similar to how we create models. If income is low and down payment is low ... a default is more likely.
- You no longer need to worry about binning! The tree will create the bins for you via splitting.
- You no longer need to worry about transformations such as logs since the split variables will take care of optimal splits.

Disadvantages of Decision Trees

- Non-parametric modeling does not give us inference. Can we ask how important alcohol is in determining wine quality? It looks important since it's the root, but we can't get a reassurance that it will stand up to sampling error formally as a p-value.

Disadvantages of Decision Trees

- Non-parametric modeling does not give us inference. Can we ask how important alcohol is in determining wine quality? It looks important since it's the root, but we can't get a reassurance that it will stand up to sampling error formally as a p-value.
- Trees have difficulty capturing actually linear or near linear relationships.

Disadvantages of Decision Trees

- Non-parametric modeling does not give us inference. Can we ask how important alcohol is in determining wine quality? It looks important since it's the root, but we can't get a reassurance that it will stand up to sampling error formally as a p-value.
- Trees have difficulty capturing actually linear or near linear relationships.
- Trees are high variance.

Disadvantages of Decision Trees

- Non-parametric modeling does not give us inference. Can we ask how important alcohol is in determining wine quality? It looks important since it's the root, but we can't get a reassurance that it will stand up to sampling error formally as a p-value.
- Trees have difficulty capturing actually linear or near linear relationships.
- Trees are high variance. This means that with different samples (different $\mathcal{E}$ values), the tree splits can vary wildly the lower in depth you go.
- But most important, tree predictive performance is not great. They are called "weak learners". Why?

Disadvantages of Decision Trees

- Non-parametric modeling does not give us inference. Can we ask how important alcohol is in determining wine quality? It looks important since it's the root, but we can't get a reassurance that it will stand up to sampling error formally as a p-value.
- Trees have difficulty capturing actually linear or near linear relationships.
- Trees are high variance. This means that with different samples (different $\mathcal{E}$ values), the tree splits can vary wildly the lower in depth you go.
- But most important, tree predictive performance is not great. They are called "weak learners". Why? Basically, we have traded interpretability and simplicity for performance.

The Tree's History and Improving the Tree

Trees were conceived in 1963 and were made rigorous in the mid-1980's with the book "Classification and Regression Trees". The 1990's saw a few huge advances:

1. Bagging (bootstrap aggregation) in 1994 (Breiman).
2. Boosting (finding errors and reweighting to fix them) in 1995 (Freund and Shapire).
3. Sampling predictors in 1997 (Amit and Geman).

These were three historical ideas which gave birth to Leo Breiman's Random Forests idea in 2001.

The Tree's History and Improving the Tree

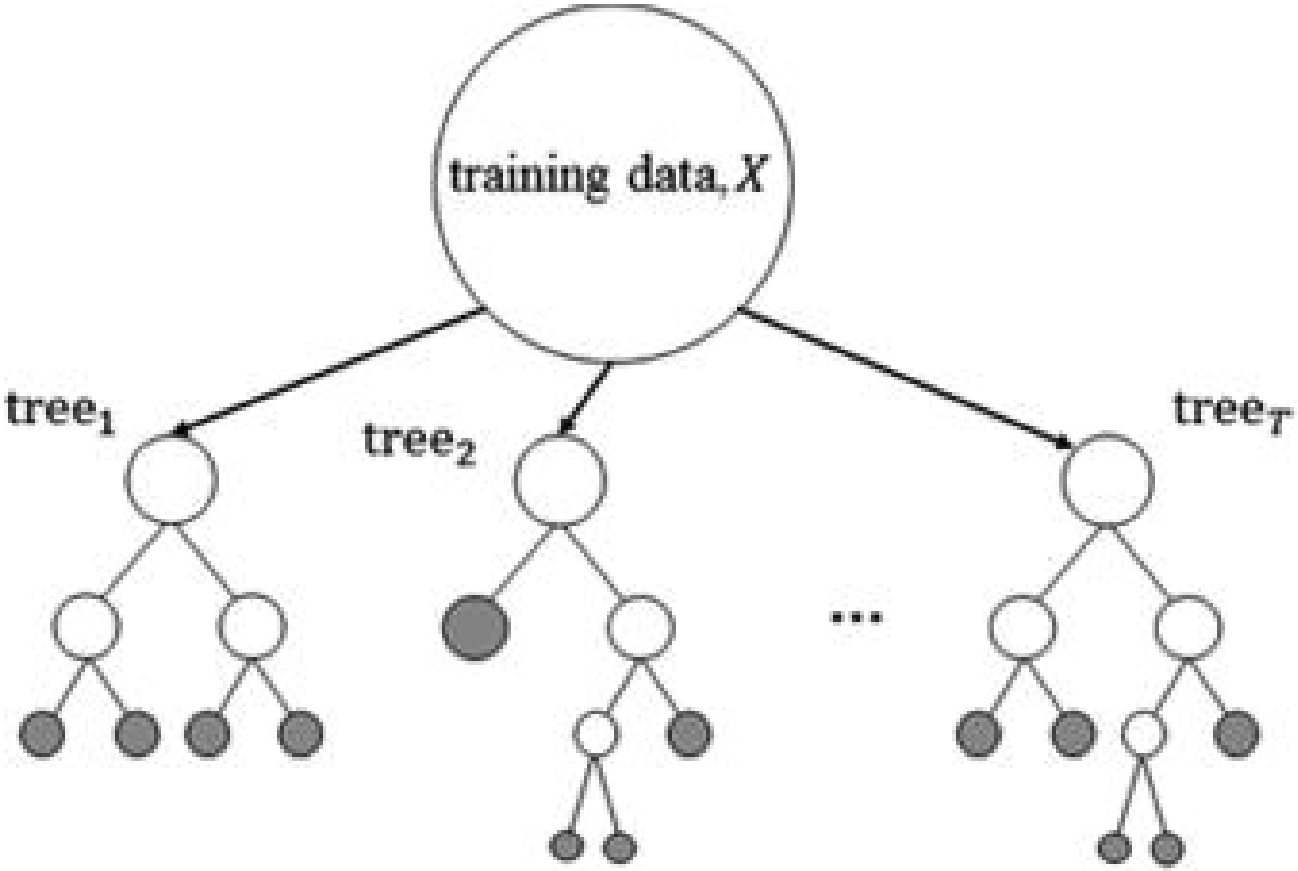
Trees were conceived in 1963 and were made rigorous in the mid-1980's with the book "Classification and Regression Trees". The 1990's saw a few huge advances:

1. Bagging (bootstrap aggregation) in 1994 (Breiman).
2. Boosting (finding errors and reweighting to fix them) in 1995 (Freund and Shapire).
3. Sampling predictors in 1997 (Amit and Geman).

These were three historical ideas which gave birth to Leo Breiman's Random Forests idea in 2001.

Many, Many Trees Together

Each of the above combines many decision trees together (T trees instead of one tree). Predictions are generated by the average leaf $\hat{y}$ (during regression) or the modal class (during classification).



What is Bagging?

"Bagging" is a portmanteau of "bootstrap" + "aggregation" which are...

What is Bagging?

"Bagging" is a portmanteau of "bootstrap" + "aggregation" which are...

Bootstrap (in our case, the "nonparametric bootstrap")

What is Bagging?

"Bagging" is a portmanteau of "bootstrap" + "aggregation" which are...

Bootstrap (in our case, the "nonparametric bootstrap")

Sample the observations $1, \ldots, n$ with replacement.

What is Bagging?

"Bagging" is a portmanteau of "bootstrap" + "aggregation" which are...

Bootstrap (in our case, the "nonparametric bootstrap")

Sample the observations $1, \ldots, n$ with replacement. You get only about $2/3$ of the unique n observations since you double and triple up.

What is Bagging?

"Bagging" is a portmanteau of "bootstrap" + "aggregation" which are...

Bootstrap (in our case, the "nonparametric bootstrap")

Sample the observations $1, \ldots, n$ with replacement. You get only about $2/3$ of the unique n observations since you double and triple up.

What it means in this context is that each of the T trees are built with a bootstrap sample.

What is Bagging?

"Bagging" is a portmanteau of "bootstrap" + "aggregation" which are...

Bootstrap (in our case, the "nonparametric bootstrap")

Sample the observations $1, \ldots, n$ with replacement. You get only about $2/3$ of the unique n observations since you double and triple up.

What it means in this context is that each of the T trees are built with a bootstrap sample. Each tree itself is a "weak learner" but many slightly different trees together is strong since they reduce variance from overfitting (beyond scope of course).

What is Bagging?

"Bagging" is a portmanteau of "bootstrap" + "aggregation" which are...

Bootstrap (in our case, the "nonparametric bootstrap")

Sample the observations $1, \ldots, n$ with replacement. You get only about $2/3$ of the unique n observations since you double and triple up.

What it means in this context is that each of the T trees are built with a bootstrap sample. Each tree itself is a "weak learner" but many slightly different trees together is strong since they reduce variance from overfitting (beyond scope of course).

Aggregation

What we said before: "aggregate" the $\hat{y}$'s from the T trees together via average (regression) or majority vote (classification).

What is Bagging?

"Bagging" is a portmanteau of "bootstrap" + "aggregation" which are...

Bootstrap (in our case, the "nonparametric bootstrap")

Sample the observations $1, \ldots, n$ with replacement. You get only about $2/3$ of the unique n observations since you double and triple up.

What it means in this context is that each of the T trees are built with a bootstrap sample. Each tree itself is a "weak learner" but many slightly different trees together is strong since they reduce variance from overfitting (beyond scope of course).

Aggregation

What we said before: "aggregate" the $\hat{y}$'s from the T trees together via average (regression) or majority vote (classification).

Bagging beats single trees... but once we bag, no interpretability anymore! Complete black box due to that function $\hat{f}$ being so complicated!!

Random Forests (RF)

To reduce variance between the trees, we have to bust up the correlation structure between each tree.

Random Forests (RF)

To reduce variance between the trees, we have to bust up the correlation structure between each tree. The last piece of the puzzle Breiman cracked in 2001: each tree only sees a sample of the features.

Random Forests (RF)

To reduce variance between the trees, we have to bust up the correlation structure between each tree. The last piece of the puzzle Breiman cracked in 2001: each tree only sees a sample of the features. Each tree only sees $p^* < p$.

Random Forests (RF)

To reduce variance between the trees, we have to bust up the correlation structure between each tree. The last piece of the puzzle Breiman cracked in 2001: each tree only sees a sample of the features. Each tree only sees $p^* < p$.

This was we do this, the predictive accuracy then beats bagging (no one uses just bagging anymore). **These many trees taken together as a unit do not overfit!!!**

Random Forests (RF) Algorithm

Random Forests (RF)

To reduce variance between the trees, we have to bust up the correlation structure between each tree. The last piece of the puzzle Breiman cracked in 2001: each tree only sees a sample of the features. Each tree only sees $p^* < p$.

This was we do this, the predictive accuracy then beats bagging (no one uses just bagging anymore). **These many trees taken together as a unit do not overfit!!!**

Random Forests (RF) Algorithm

- Build many overfit trees
- In each tree use a sample with replacement of the rows
- In each tree use a sample of the availabe features

Random Forests (RF)

To reduce variance between the trees, we have to bust up the correlation structure between each tree. The last piece of the puzzle Breiman cracked in 2001: each tree only sees a sample of the features. Each tree only sees $p^* < p$.

This was we do this, the predictive accuracy then beats bagging (no one uses just bagging anymore). **These many trees taken together as a unit do not overfit!!!**

Random Forests (RF) Algorithm

- Build many overfit trees
- In each tree use a sample with replacement of the rows
- In each tree use a sample of the availabe features

RF "Tuning Parameters"

- Number of trees — we want a lot to reduce variance (default is 500 in R)
- Number of splitting features for a individual tree — we want less than p to decorrelate (default $p/3$ for regression and $\sqrt{p}$ for classification)
- Number of observations in smallest split node — we want to overfit (default is 5 in regression, 1 in classification)

The amazing thing is ...

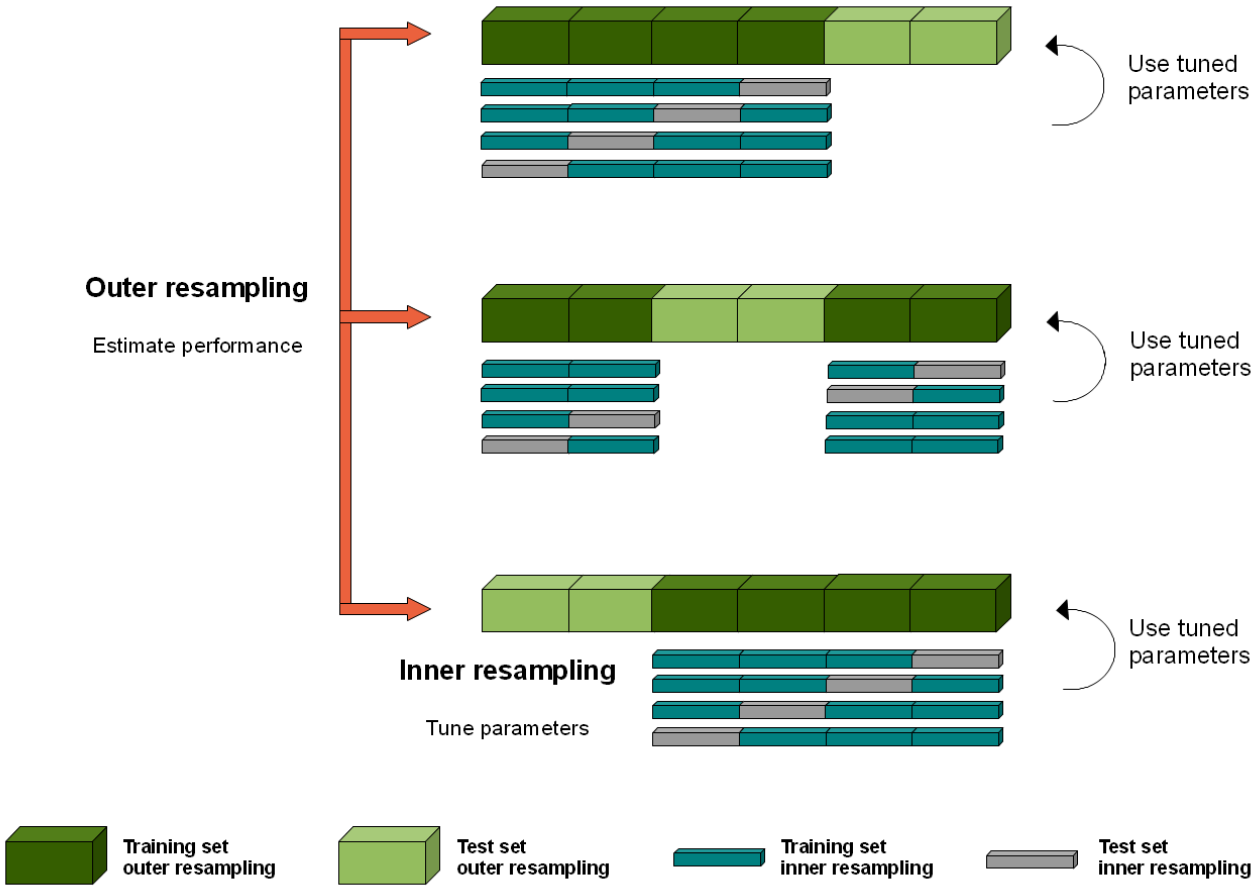
RF "Tuning Parameters"

- Number of trees — we want a lot to reduce variance (default is 500 in R)
- Number of splitting features for a individual tree — we want less than p to decorrelate (default $p/3$ for regression and $\sqrt{p}$ for classification)
- Number of observations in smallest split node — we want to overfit (default is 5 in regression, 1 in classification)

The amazing thing is ... the defaults work so well, you hardly ever need to change them!! **RF in my opinion is the best predictive model out-of-the-box.**

Tuning RF — Nested Resampling

Remember this from last class? Here's where it's used... But I do not recommend doing this!



A Nice Perk in RF — OOB Estimation

So RF doesn't overfit, but how do we assess this?

A Nice Perk in RF — OOB Estimation

So RF doesn't overfit, but how do we assess this? we can use oos validation (plain or with K-fold CV), but we don't have to! Why?

A Nice Perk in RF — OOB Estimation

So RF doesn't overfit, but how do we assess this? we can use oos validation (plain or with K-fold CV), but we don't have to! Why? Each tree was built with a bootstrap sample. This means $\approx 2/3$ of the n observations were used to build the tree and therefore $\approx 1/3$ of the n observations were not! Aren't those $\approx 1/3$ that were not oos?

A Nice Perk in RF — OOB Estimation

So RF doesn't overfit, but how do we assess this? we can use oos validation (plain or with K-fold CV), but we don't have to! Why? Each tree was built with a bootstrap sample. This means $\approx 2/3$ of the n observations were used to build the tree and therefore $\approx 1/3$ of the n observations were not! Aren't those $\approx 1/3$ that were not oos? YES!

A Nice Perk in RF — OOB Estimation

So RF doesn't overfit, but how do we assess this? we can use oos validation (plain or with K-fold CV), but we don't have to! Why? Each tree was built with a bootstrap sample. This means $\approx 2/3$ of the n observations were used to build the tree and therefore $\approx 1/3$ of the n observations were not! Aren't those $\approx 1/3$ that were not oos? YES! These $\approx 1/3$ are called "out of bag" (OOB). Over the many T trees, all n observations went out of bag many times giving us many trees to get oos average for each observation. OOB = K-fold CV (usually very close... you'll see).

A Nice Perk in RF — OOB Estimation

So RF doesn't overfit, but how do we assess this? we can use oos validation (plain or with K-fold CV), but we don't have to! Why? Each tree was built with a bootstrap sample. This means $\approx 2/3$ of the n observations were used to build the tree and therefore $\approx 1/3$ of the n observations were not! Aren't those $\approx 1/3$ that were not oos? YES! These $\approx 1/3$ are called "out of bag" (OOB). Over the many T trees, all n observations went out of bag many times giving us many trees to get oos average for each observation. OOB = K-fold CV (usually very close... you'll see).

(Note: this strategy can be used for any modeling procedure as an alternative to K-fold CV and MLR in R gives you this option... I'm still not sure why K-fold CV is the default).

A Nice Perk in RF — OOB Estimation

So RF doesn't overfit, but how do we assess this? we can use oos validation (plain or with K-fold CV), but we don't have to! Why? Each tree was built with a bootstrap sample. This means $\approx 2/3$ of the n observations were used to build the tree and therefore $\approx 1/3$ of the n observations were not! Aren't those $\approx 1/3$ that were not oos? YES! These $\approx 1/3$ are called "out of bag" (OOB). Over the many T trees, all n observations went out of bag many times giving us many trees to get oos average for each observation. OOB = K-fold CV (usually very close... you'll see).

(Note: this strategy can be used for any modeling procedure as an alternative to K-fold CV and MLR in R gives you this option... I'm still not sure why K-fold CV is the default).

However, RF does overfit in-sample. How?

A Nice Perk in RF — OOB Estimation

So RF doesn't overfit, but how do we assess this? we can use oos validation (plain or with K-fold CV), but we don't have to! Why? Each tree was built with a bootstrap sample. This means $\approx 2/3$ of the n observations were used to build the tree and therefore $\approx 1/3$ of the n observations were not! Aren't those $\approx 1/3$ that were not oos? YES! These $\approx 1/3$ are called "out of bag" (OOB). Over the many T trees, all n observations went out of bag many times giving us many trees to get oos average for each observation. OOB = K-fold CV (usually very close... you'll see).

(Note: this strategy can be used for any modeling procedure as an alternative to K-fold CV and MLR in R gives you this option... I'm still not sure why K-fold CV is the default).

However, RF does overfit in-sample. How? 2/3 of the T trees, it overfits and 1/3 of the T trees is OOB for every observation. Thus it is mostly overfit! Never trust in-sample R^2 from an RF! Only look at OOB!

Another Nice Perk: $\hat{p}$'s!

Another Nice Perk: $\hat{p}$'s!

In the classification case, there are T trees which each have a $\hat{y}$ of the class in the leaf. We can now go the reverse direction, aggregate the $\hat{y}$'s across all T trees and try to estimate the probability of the kth class for any observation $\boldsymbol{x}^*$

$$\hat{p}_k(\boldsymbol{x}^*) = \frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{\hat{y}_i(\boldsymbol{x}^*)=k}.$$

Another Nice Perk: $\hat{p}$'s!

In the classification case, there are T trees which each have a $\hat{y}$ of the class in the leaf. We can now go the reverse direction, aggregate the $\hat{y}$'s across all T trees and try to estimate the probability of the kth class for any observation $\boldsymbol{x}^*$

$$\hat{p}_k(\boldsymbol{x}^*) = \frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{\hat{y}_i(\boldsymbol{x}^*)=k}.$$

This sometimes works and sometimes doesn't... but many data scientists seem to use this procedure in practice.

Another Nice Perk: $\hat{p}$'s!

In the classification case, there are T trees which each have a $\hat{y}$ of the class in the leaf. We can now go the reverse direction, aggregate the $\hat{y}$'s across all T trees and try to estimate the probability of the kth class for any observation $\boldsymbol{x}^*$

$$\hat{p}_k(\boldsymbol{x}^*) = \frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{\hat{y}_i(\boldsymbol{x}^*)=k}.$$

This sometimes works and sometimes doesn't... but many data scientists seem to use this procedure in practice.

Note: there is a PhD student right now in this department exploring when this works and how much you can trust it! Bleeding edge of stats...

When does RF win / lose?

When does RF win / lose?

- Signal is high, noise is low (no one wins when signal is low, noise is high). I've seen the linear model and RF performing about equal in high noise cases.

When does RF win / lose?

- Signal is high, noise is low (no one wins when signal is low, noise is high). I've seen the linear model and RF performing about equal in high noise cases.
- When there truly are lots of interactions among the predictors and curvilinear relationships with the response.

When does RF win / lose?

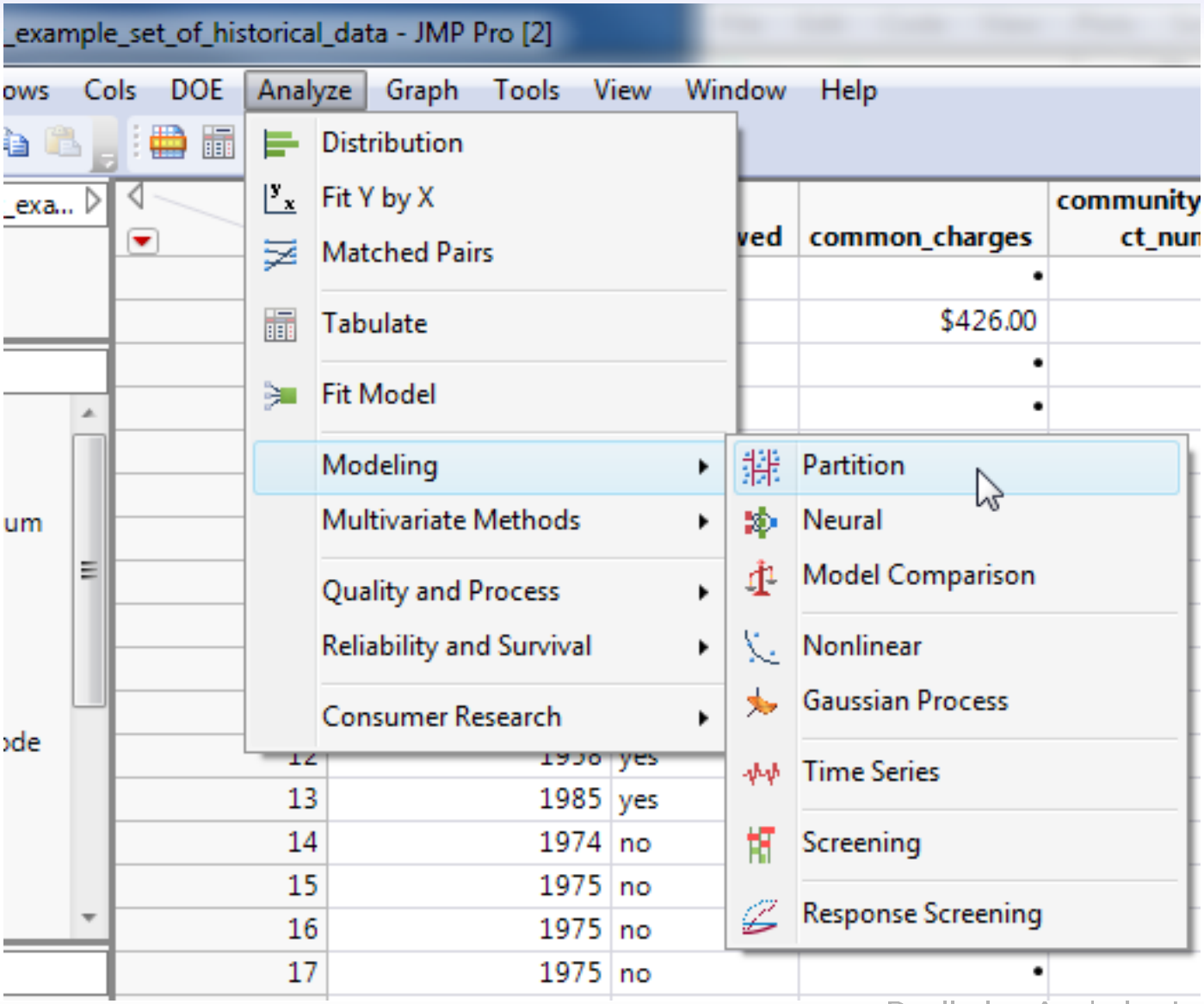
- Signal is high, noise is low (no one wins when signal is low, noise is high). I've seen the linear model and RF performing about equal in high noise cases.
- When there truly are lots of interactions among the predictors and curvilinear relationships with the response.
- When there are not "too many" features. $p > n$ is a problem for everyone. RF can find what it believes to be a nice predictor, but it's really fake and only idiosyncratically related to the response.

When does RF win / lose?

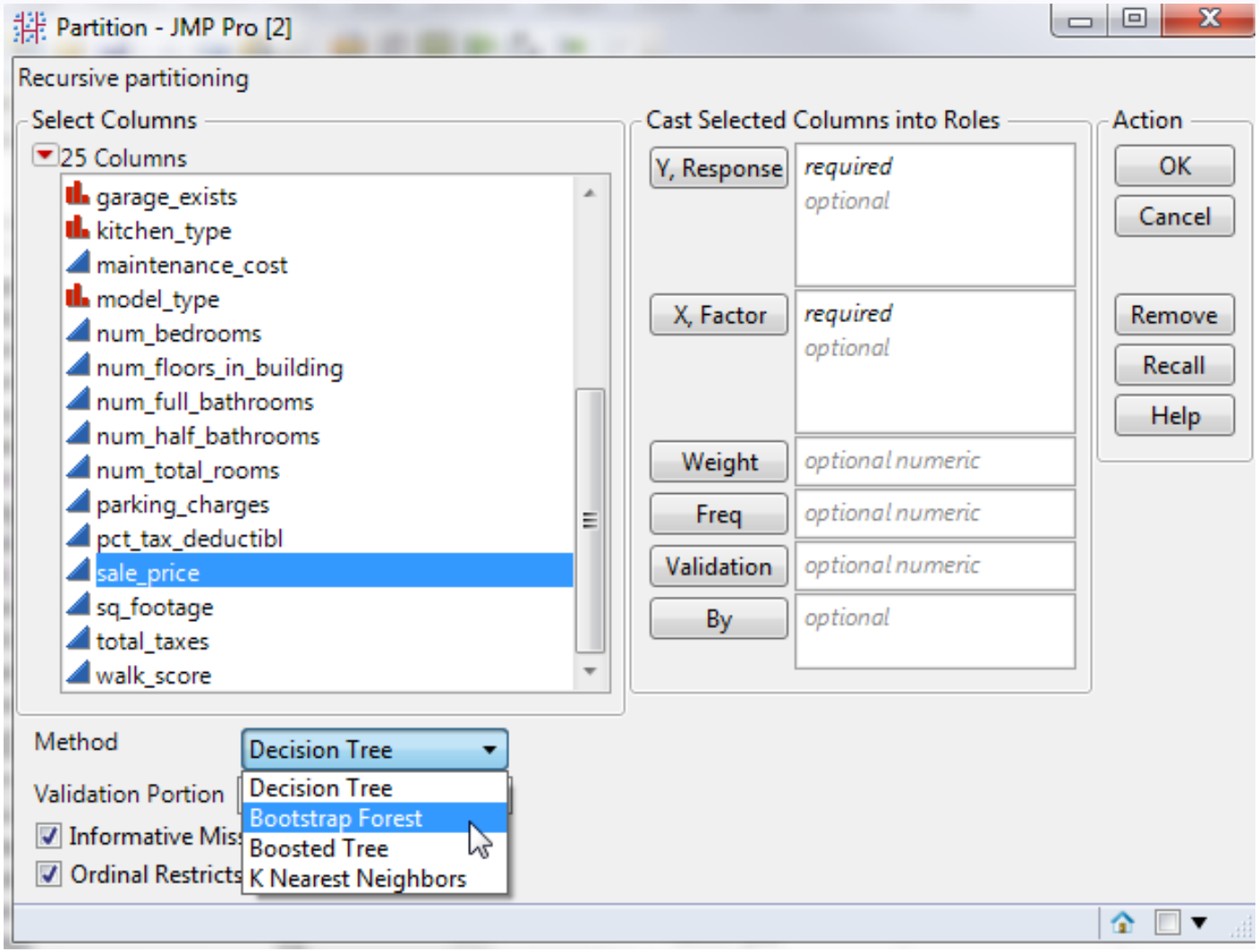
- Signal is high, noise is low (no one wins when signal is low, noise is high). I've seen the linear model and RF performing about equal in high noise cases.
- When there truly are lots of interactions among the predictors and curvilinear relationships with the response.
- When there are not "too many" features. $p > n$ is a problem for everyone. RF can find what it believes to be a nice predictor, but it's really fake and only idiosyncratically related to the response.
- When you need interpretation, RF cannot give you what you want! Note: many people working on this... it was actually part of my thesis! See package `ICEBox` in R.

Demo time...

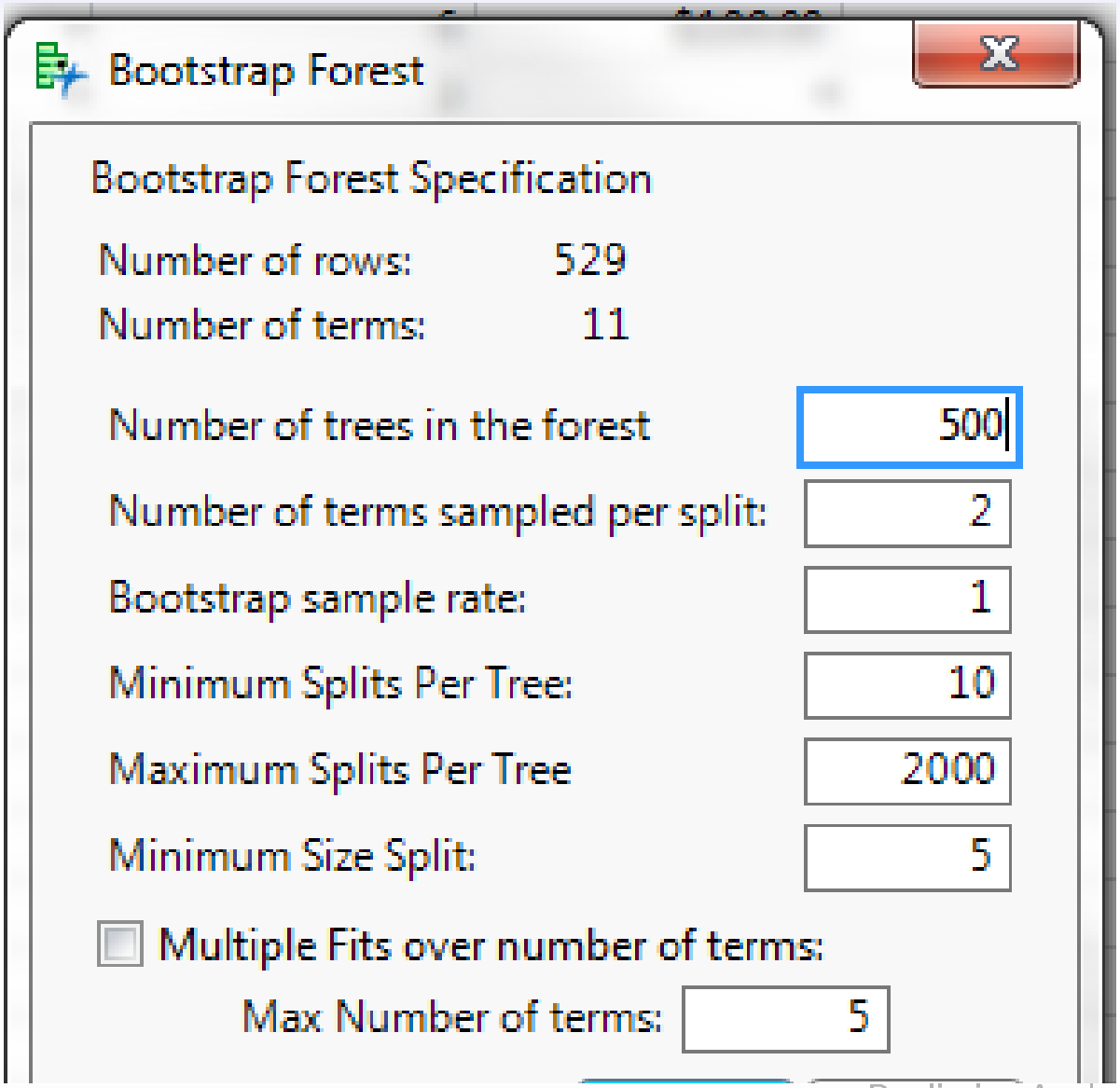
JMP 1/4


_example_set_of_historical_data - JMP Pro [2]
ows Cols DOE Analyze Graph Tools View Window Help
Distribution
Fit Y by X
Matched Pairs
Tabulate
Fit Model
Modeling
Multivariate Methods
Quality and Process
Reliability and Survival
Consumer Research
Partition
Neural
Model Comparison
Nonlinear
Gaussian Process
Time Series
Screening
Response Screening
common_charges
community ct_nun
$426.00
13 1985 yes
14 1974 no
15 1975 no
16 1975 no
17 1975 no

JMP 2/4


Partition - JMP Pro [2]
Recursive partitioning
Select Columns
25 Columns
garage_exists
kitchen_type
maintenance_cost
model_type
num_bedrooms
num_floors_in_building
num_full_bathrooms
num_half_bathrooms
num_total_rooms
parking_charges
pct_tax_deductibl
sale_price
sq_footage
total_taxes
walk_score
Cast Selected Columns into Roles
Y, Response
required
optional
X, Factor
required
optional
Weight
optional numeric
Freq
optional numeric
Validation
optional numeric
By
optional
Action
OK
Cancel
Remove
Recall
Help
Method
Decision Tree
Validation Portion
Informative Miss
Ordinal Restricts
Decision Tree
Bootstrap Forest
Boosted Tree
K Nearest Neighbors

JMP 3/4


Bootstrap Forest
Bootstrap Forest Specification
Number of rows: 529
Number of terms: 11
Number of trees in the forest 500
Number of terms sampled per split: 2
Bootstrap sample rate: 1
Minimum Splits Per Tree: 10
Maximum Splits Per Tree 2000
Minimum Size Split: 5
Multiple Fits over number of terms:
Max Number of terms: 5

JMP 4/4

Bootstrap Forest for sale_price

Specifications

Target Column:	sale_price	Training rows:	529
		Validation rows:	0
Number of trees in the forest:	500	Test rows:	0
Number of terms sampled per split:	2	Number of terms:	11
		Bootstrap samples:	529
		Minimum Splits Per Tree:	10
		Minimum Size Split:	5

Overall Statistics

Individual Trees	RMSE
In Bag	1049084
Out of Bag	1415828

RSquare	RMSE	N
0.623	110008.48	529

Per-Tree Summaries

Actual by Predicted Plot

You don't need to do oos validation on Random Forests as the "out of bag" is as good as oos.

R 1/2

```
#load the data
X = read.csv(
 "stat_422_722_project_example_set_of_historical_data.csv")
#recode sale_price as a number
X$sale_price = as.numeric(gsub('[$,]','',
  as.character(X$sale_price)))
#install and load up the RF package
install.packages("randomForest")
library(randomForest)
#run the RF
rf_mod = randomForest(sale_price ~
coop_condo + num_bedrooms +
num_full_bathrooms + walk_score, X)
rf_mod #print out its output
```

R 2/2

```
> rf_mod

Call:
 randomForest(formula = sale_price ~ coop_condo + num_bed
rooms +      num_full_bathrooms + walk_score, data = X)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1

          Mean of squared residuals: 10620431253
                    % Var explained: 66.96
```

We Made It! Now a Summary and Review...

A quick review of the topics (and the topics that will not be covered on the final next week).

Lecture 1

- Definitions of prediction, model, response, feature / predictor, observation

We Made It! Now a Summary and Review...

A quick review of the topics (and the topics that will not be covered on the final next week).

Lecture 1

- Definitions of prediction, model, response, feature / predictor, observation
- Mathematizing English into measurements so we can create models

We Made It! Now a Summary and Review...

A quick review of the topics (and the topics that will not be covered on the final next week).

Lecture 1

- Definitions of prediction, model, response, feature / predictor, observation
- Mathematizing English into measurements so we can create models
- We talked about mathematical models, statistical models and the conditional expectation function f.

We Made It! Now a Summary and Review...

A quick review of the topics (and the topics that will not be covered on the final next week).

Lecture 1

- Definitions of prediction, model, response, feature / predictor, observation
- Mathematizing English into measurements so we can create models
- We talked about mathematical models, statistical models and the conditional expectation function f.
- Types of data: continuous, categorical. We talked about binning to convert continuous $\rightarrow$ categorical.

We Made It! Now a Summary and Review...

A quick review of the topics (and the topics that will not be covered on the final next week).

Lecture 1

- Definitions of prediction, model, response, feature / predictor, observation
- Mathematizing English into measurements so we can create models
- We talked about mathematical models, statistical models and the conditional expectation function f.
- Types of data: continuous, categorical. We talked about binning to convert continuous $\rightarrow$ categorical.
- Raw data represenations and deep learning, featurization to p measurable features

We Made It! Now a Summary and Review...

A quick review of the topics (and the topics that will not be covered on the final next week).

Lecture 1

- Definitions of prediction, model, response, feature / predictor, observation
- Mathematizing English into measurements so we can create models
- We talked about mathematical models, statistical models and the conditional expectation function f.
- Types of data: continuous, categorical. We talked about binning to convert continuous $\rightarrow$ categorical.
- Raw data represenations and deep learning, featurization to p measurable features
- Meehl's work on basic statistical models beating humans

We Made It! Now a Summary and Review...

A quick review of the topics (and the topics that will not be covered on the final next week).

Lecture 1

- Definitions of prediction, model, response, feature / predictor, observation
- Mathematizing English into measurements so we can create models
- We talked about mathematical models, statistical models and the conditional expectation function f.
- Types of data: continuous, categorical. We talked about binning to convert continuous $\rightarrow$ categorical.
- Raw data represenations and deep learning, featurization to p measurable features
- Meehl's work on basic statistical models beating humans
- Data collection: sampling and measurement to create a data frame

Review...

- Good machine learning (sampling and featurization) vs. bad machine learning (mindless data dumps of convenience)
- Parametric vs non-parametric worldview when modeling
- Definition of parameters
- Definition of parameter estimates (model fitting)
- Residuals and loss functions (**SSE** *, SAE)
- Measures of fit: R^2, SSE, $RMSE$ and their interpretations and their equivalence
- Empirical rule $\hat{y} \pm RMSE$

Review...

Lecture 2

- Graphical causal models with bubbles and arrows
- Definition of noise in statistical models
- Probabilistic Causation
- Three sources of noise (1) Irreducible Error (2) Model Misspecification due to a parametric assumption (3) Model Estimation Error
- The reasonable assumption of $\mathcal{E}$ distributed normally
- The four OLS assumptions and what this buys you
- What a likelihood is and what a log-likelihood is
- What maximum likelihood estimation is
- What the likelihood ratio is and when to reject (big values that exceed critical χ^2 values)

Review...

- OLS regression: Omnibus / global F-tests
- OLS regression: Partial F-test
- OLS regression: t-test
- OLS regression: how to predict ($\hat{y}$ calculations)
- Proper interpretation of $\hat{\beta}_j$
- Causation and correlation
- Lurking / confounding predictors
- Spurious (or coincidental) correlations
- Natural observation vs. manipulated measurements
- Randomized experiments can reveal additive causal effects
- Why "science is impossible"
- Why collinearity may invalidate $\hat{\beta}_j$ interpretation
- Why correlations are still useful

Review...

- Data dredging
- Multiple testing correction with Bonferroni (hedges against dredging)
- Generalizability of models

Lecture 3

- Extrapolation (a violation of generalizability)
- How different modeling strategies extrapolate differently
- Optimal design in linear models (over sample predictors' extreme values)
- Optimal design in highly non-parametric models (more towards even spacing throughout predictors' ranges)
- Odds, logistic link function and its sigmoidal shape, logit function (log odds)
- Fitting $\hat{\beta}_j$'s in logistic regression

Review...

- How to draw inference in logistic regression (likelihood ratio test)
- How to predict conditional probabilities $\hat{p}$'s in logistic regression
- How to use classification rules based on a threshold $\hat{p}_0$
- Confusion Matrices, FN, FP, TP, TN, sensitivity, specificity, FNR, FPR, FDR, FOR
- ROC Curve, ROC table and AUC metric

Lecture 4

- Asymmetric costs in classification, weighted misclassification error, minimizing this metric to find best p_0
- Survival Model — the response and why it's censored
- Interpreting $\hat{\beta}_j$ in an exponential survival model
- How to draw inference in survival regression (likelihood ratio test)
- Predictions in survival models

Review...

- Testing linearity for one covariate in linear regression
- Interpreting quadratic effects
- Interpreting interaction effects
- Overfitting and optimizing for the idiosyncratic $\mathcal{E}$ terms
- In-sample vs. oos metrics of fit (complexity-fit tradeoff curves)
- Single training-test splits for oos validation
- The test set as a lockbox that can be opened once and only once
- Tradeoffs for training-test split sizes

Lecture 5

- K-fold cross validation (CV)
- Advantage of doing K-fold CV
- Using three splits to perform model selection

Review...

- Missing data: selection models vs. pattern mixture models
- Mechanisms: MCAR, MAR, NMAR
- Why listwise deletion is nothing to write home about
- Imputation under missingness due to each of the three mechanisms
- Creating missingness features (the $\boldsymbol{m}$'s)
- Forward stepwise regression using t-test, AICc and R^2 in a oos set

Lecture 6

- Binning — advantages and disadvantages
- Better bins through greedy splitting (decision trees)
- Leaf assignment in decision trees
- Pruning / stopping the growth of decision trees
- Advantages and disadvantages of decision trees

Review...

- Model averaging over multiple trees
- Bagging (bootstrap and aggregation)
- Sampling predictors
- The 3 ideas above together = random forests (RF)
- OOB estimates as oos estimates
- When RF performs well and when it does not