

# Feature importance

Which features are the most predictive or have most impact?

Terence Parr  
MSDS program  
**University of San Francisco**

# What does the model say about the data?

- A good model is great, but we usually want to interpret the model
- Feature importance tells us about the business or application; e.g., what matters to people renting apartments or buying used bulldozers or identifying cancer or getting a bank loan?
- Importance scores themselves don't usually mean anything, and they are often normalized 0..1 anyway
- Only relative rank/magnitude matters for feature selection to drop irrelevant features

# Formal definition of feature importance?

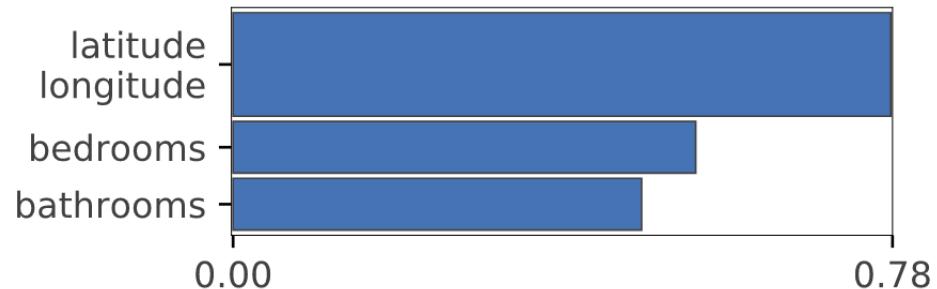
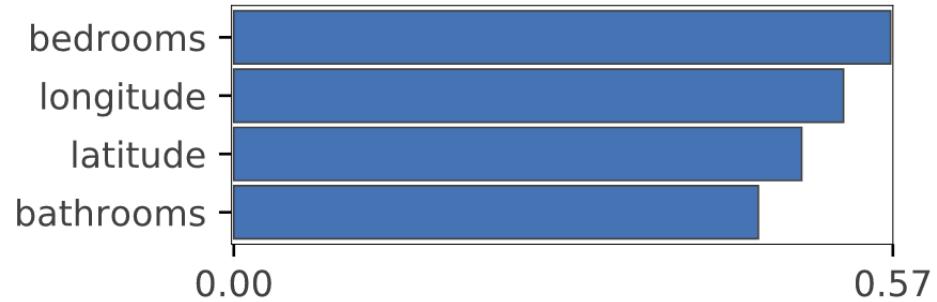
- Not sure there is an agreed-upon definition
- *Feature importance* gives (usually just) a relative ranking of the predictive strength among the features of the model; useful for simplifying models and possibly improving generality
- We'd also like a model-independent *feature impact* that identifies the effect of features on  $y$  without peering through lens of a model: "the amount of  $y$  variation attributed specifically to feature  $x_j$ " See [1]
- We want importance/impact to isolate impact of individual features
- Nonetheless, we'll focus on feature importance as proxy for impact

Coining new term to distinguish from feature importance

[1] <https://arxiv.org/abs/2006.04750> for detailed discussion

# Rent example

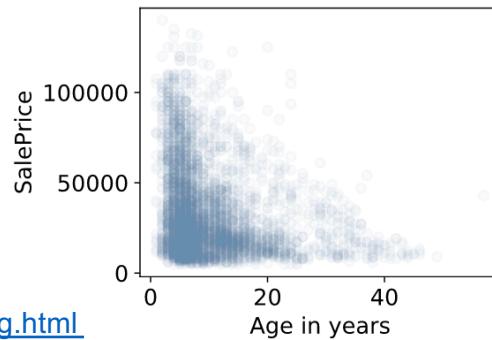
- Number of **bedrooms** appears to be the most important, meaning that it is the most predictive of rent price
- That tells us something about the market for New York City rent
- Combined, however, the location matters more by far; note: **longitude** and **latitude** are codependent features



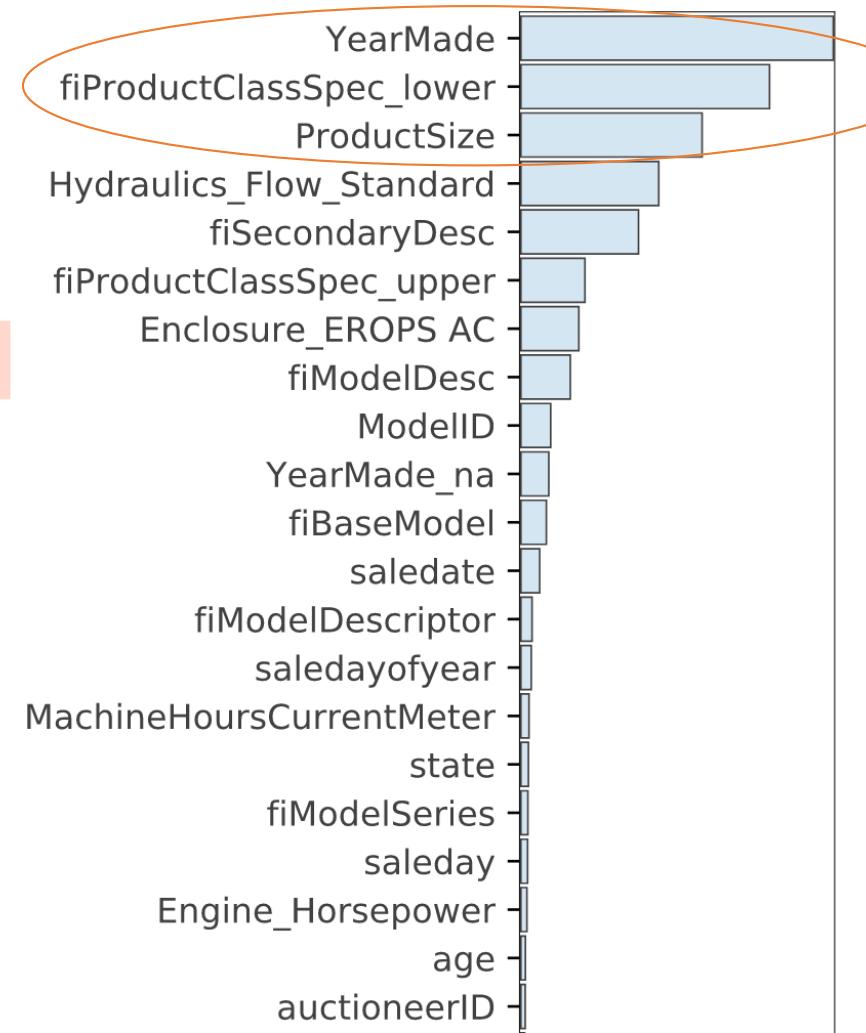
See <https://mlbook.explained.ai/first-taste.html>

# Bulldozer example

- **YearMade**, lower capacity spec, size matter most
- Note **age** appears unimportant but date-related features are highly correlated; other features can cover for codependent features; still looks predictive to me:

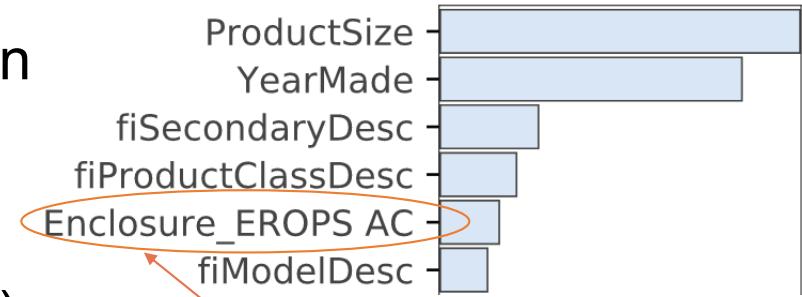


See <https://mlbook.explained.ai/bulldozer-feateng.html>

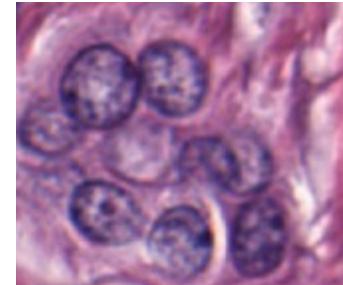


# Bulldozer one-hot example

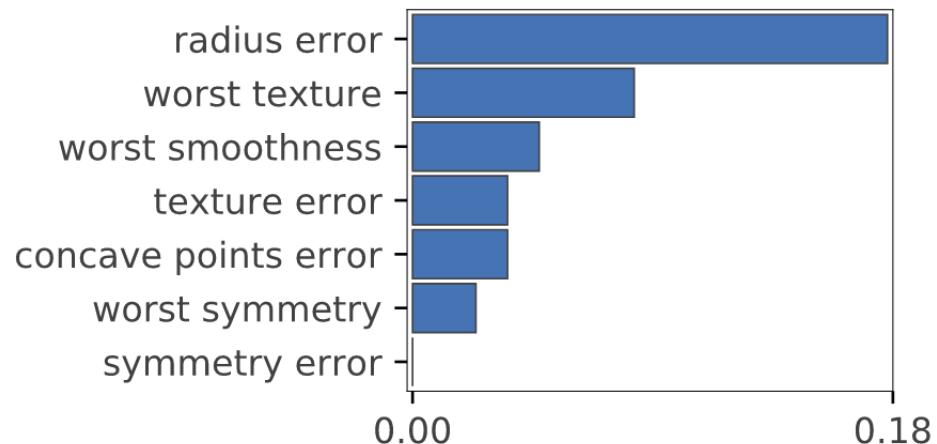
- One-hot'ing features temporarily and then examining feature importance can identify which category level is important vs just the variable
- E.g., one-hot **Enclosure** and we see that “EROPS AC” (air conditioned cab) is most important **Enclosure**, which is valuable marketing / business info (other levels are way down in the noise)
- Could be useful for improving model but definitely useful for business intelligence



# Breast cancer classification example



- Distinguishing between malignant versus benign masses: **radius error** is strongly predictive
- That is “*standard error for the mean of distances from center to points on the perimeter*” which could give an indication of mass edge irregularity
- **Worst texture** is “*largest mean value for standard deviation of gray-scale values*”



See <https://mlbook.explained.ai/first-taste.html>

# Common application: Tuning the model

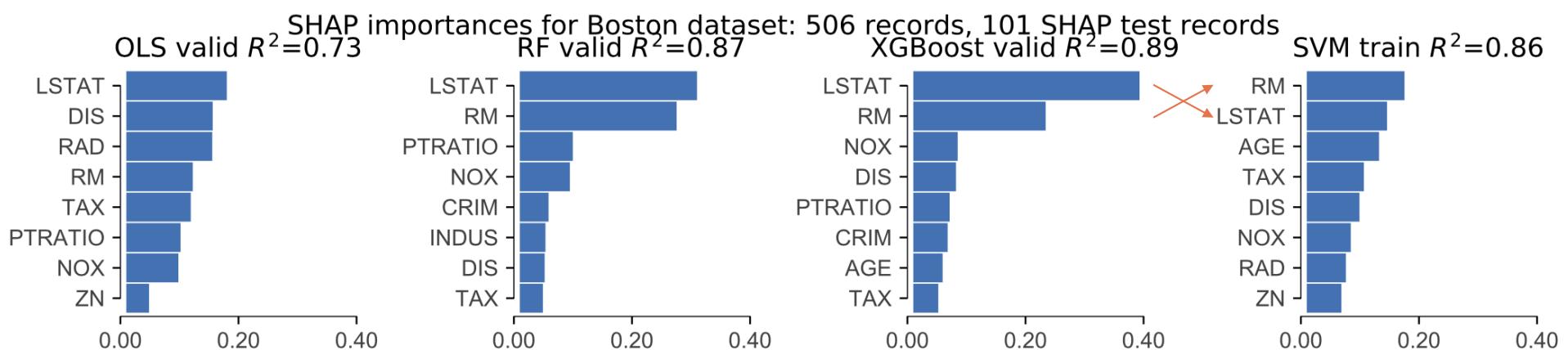
- Dropping unimportant features simplifies the model; fewer features make it easier to interpret/explain model behavior
- Often increases accuracy and generality because the model is not taking irrelevant features into consideration (e.g., noise vars)
- Simplifying models is a form of regularization
- Fewer features increases training and prediction speed
- Drop the lowest importance feature and retrain the model, redo validation metric; if validation metric worse, then we have dropped one too many features
- Codependencies between feature is why we recompute feature importances after dropping each feature; at minimum, rank of features can change dramatically after dropping a feature

# Beware!

- Can't trust feature importances from *weak* models
- Can't trust feature importances from *unstable* models
- Importance says how important a feature is to a specific model
- Sum of importances not usually meaningful; doesn't tell you how much of the overall prediction your model has covered
- When possible compute importances with validation not training set; we care about features predictive for generalization
- Even with good model, importances are a clue not gospel

# Importance is typically model-specific!

- Same method applied to same data can yield different importances; here's SHAP on Boston but different models



# Implementation of importance

# Importance directly from the data: Spearman's rank correlation

- Simplest technique for regression is to rank  $x_j$  features by their Spearman's rank correlation [1] with target  $y$  (doesn't assume linear relationship)
- The feature with largest coefficient is most important
- Measures *single-feature importance* and works well for independent features, but not good for codependent features
- Groups of features with similar relationships to the response variable receive the same or similar ranks, even though just one should be considered important
- Seems inappropriate for categorical variables

[1] [https://en.wikipedia.org/wiki/Spearman%27s\\_rank\\_correlation\\_coefficient](https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient)

# Importance directly from the data: PCA

- Principle component analysis (PCA) operates on just the  $X$  explanatory matrix; limited to linear relationships and "most variance" is not always the same as "most important"
- PCA transforms data into a new space characterized by eigenvectors of  $X$  and identifies features that explain the most variance in the new space
- If the first principal component covers a large percentage of the variance, the “loads” associated with that component can indicate importance of features in the original  $X$  space
- Seems inappropriate for categorical variables

# Importance directly from the data: mRMR

- To deal with codependencies, rank not just by *relevance* but also by *redundance*; mRMR: minimal-redundancy maximal-relevance

$$J_{\text{mRMR}}(x_k) = I(x_k, y) - \frac{1}{|S|} \sum_{x_j \in S} I(x_k, x_j)$$

- Redundance is the amount of information shared between codependent features
- $I(x_k, x_j)$  is some measure of mutual information between  $x_k$  and  $x_j$ ,  $S$  is the growing set of selected features, and  $x_k$  is the candidate feature; can use Spearman's for  $I(x_k, x_j)$
- Only considers single-feature relationships with  $y$ , limited to classification, what is  $I(x_k, x_j)$  for categorical variables?

# Importance via linear models

- For suitably prepared data without missing variables and other appropriate assumptions, linear model coefficients give the amount of  $y$  variance explained for each  $x_j$  variable
- But, linear models are often too weak in practice for modeling, which renders coefficient interpretation highly suspect
- More on difficulties of interpreting regression coefficients by Breiman <https://projecteuclid.org/euclid.ss/1009213726>

# So how do we really get feature importance?

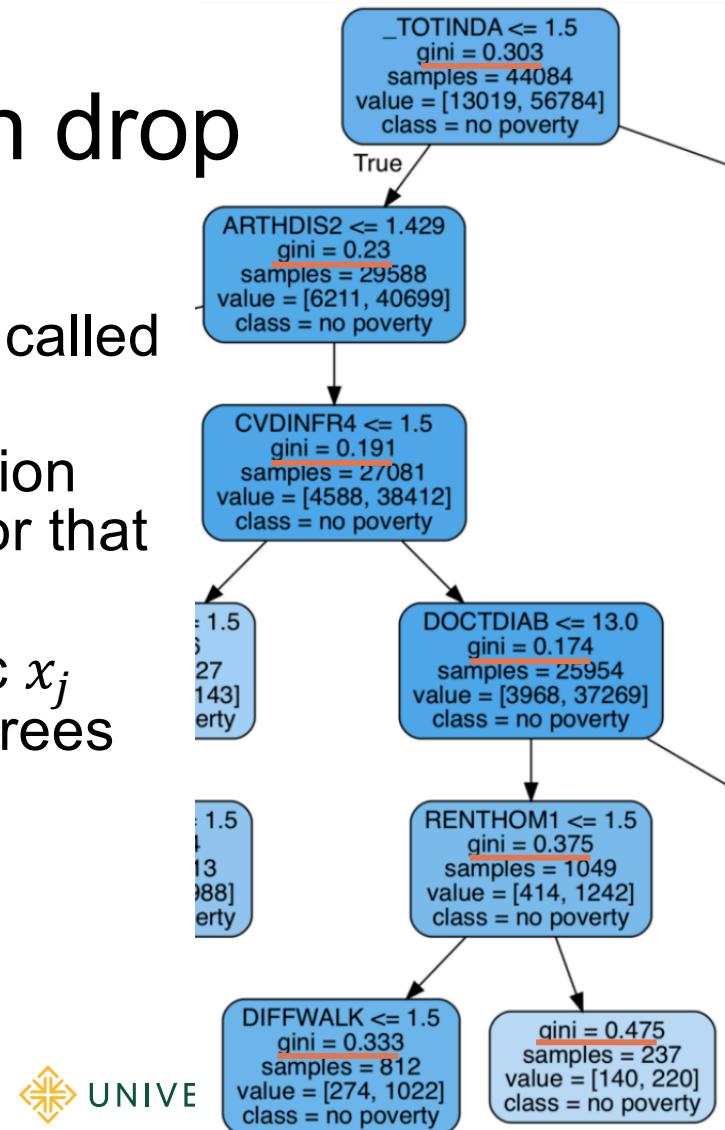
- RF specific and scikit-learn default: gini/MSE drop (I'd avoid)
- Drop-column importance
- Permutation importance

```
from sklearn.inspection import permutation_importance
```
- scikit-learn added after we exposed the weakness of gini drop  
<https://explained.ai/rf-importance/index.html> and see rfpimp [1]
- SHAP (<https://github.com/slundberg/shap>) likely most accurate so far (w/amazing library) but I find it unbearably slow

[1] <https://github.com/parrt/random-forest-importances/blob/master/README.md>

# Random Forest loss function drop

- Random Forests have a built-in mechanism called “gini drop” (for regression, it’s “MSE drop”)
- The idea is to track how much the loss function drops from a decision node to its children (for that split variable)
- The average loss function drop for a specific  $x_j$  across decision nodes for  $x_j$  and across all trees gives the feature importance
- Unfortunately, this gives biased feature importances



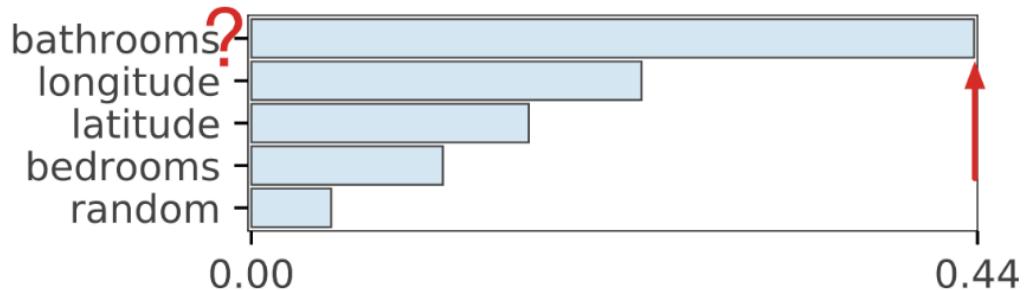
# Random Forest gini/MSE drop issues

- Exhaustively testing every unique  $x_j$  value when finding decision node splits increases likelihood of finding a  $x_j$  value that, purely by chance, happens to predict  $y$  well
- That increases the likelihood that variable  $x_j$  will appear more often in the trees, which leads to inflated/biased importance
- It's likely that extremely random trees, that pick a random split value between  $\min(x_j)$  and  $\max(x_j)$  would not suffer from this bias; I haven't tried this theory out, but it makes sense
- Breiman: "*adding up the gini decreases for each individual variable over all trees in the forest gives a **fast** variable importance that is **often very consistent** with the permutation importance measure.*"

Read more <https://explained.ai/rf-importance/index.html>

# Trouble in paradise (regression)

- Don't trust default ("gini drop") importances for RF in sklearn
- Here we see the unlikely idea that New Yorkers care most about bathrooms and much more than location or bedrooms
- New Yorkers can be weird, but they can't be this fixated on bathrooms
- Random noise column is last (which it should be)

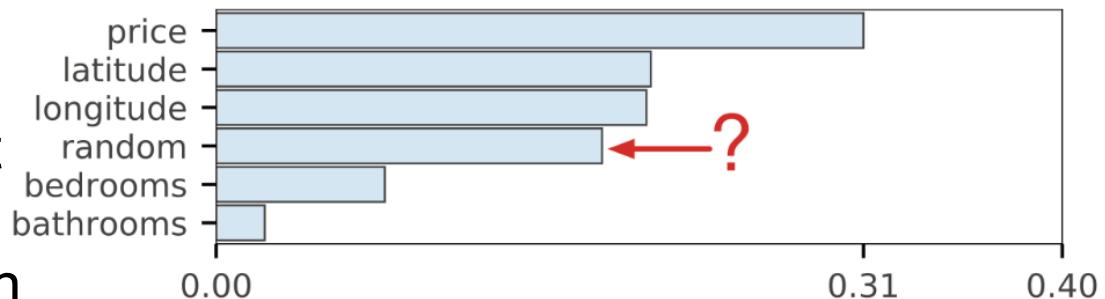


Read more <https://explained.ai/rf-importance/index.html>

# Trouble in paradise

(classification: predicting interest in apt web page)

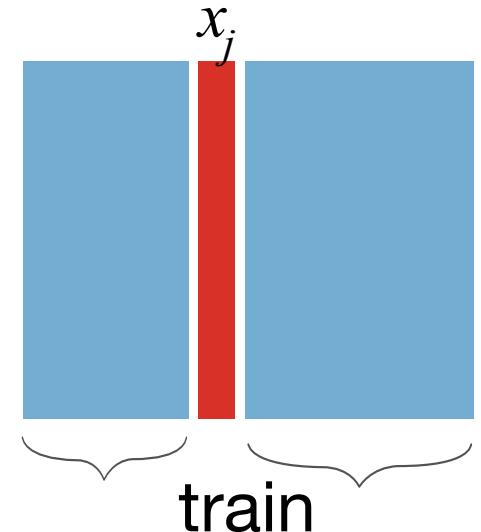
- Same data but classifying interest in apartments but price is now feature not target
- Random noise column is now somehow more important than bedrooms and bathrooms?
- Somethin ain't right
- For more on “gini drop”, see:  
<https://stackoverflow.com/questions/15810339/how-are-feature-importances-in-randomforestclassifier-determined>



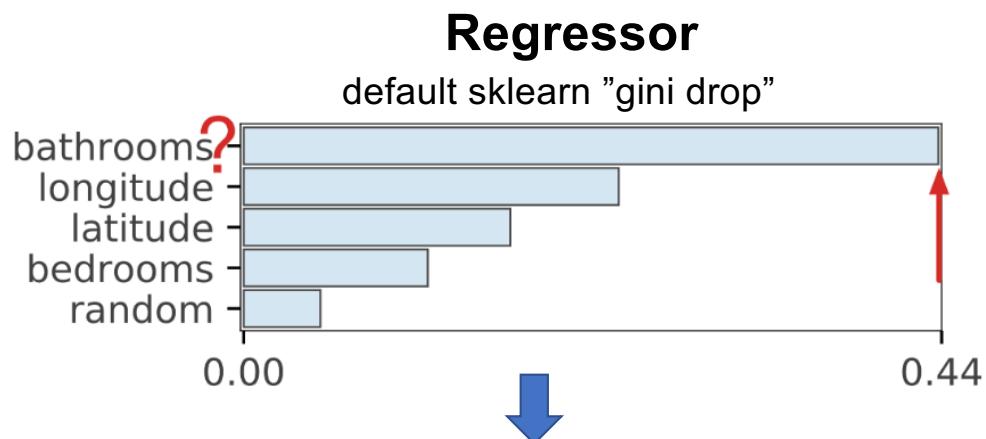
Hmm... what can we do instead?

# Drop-column importance

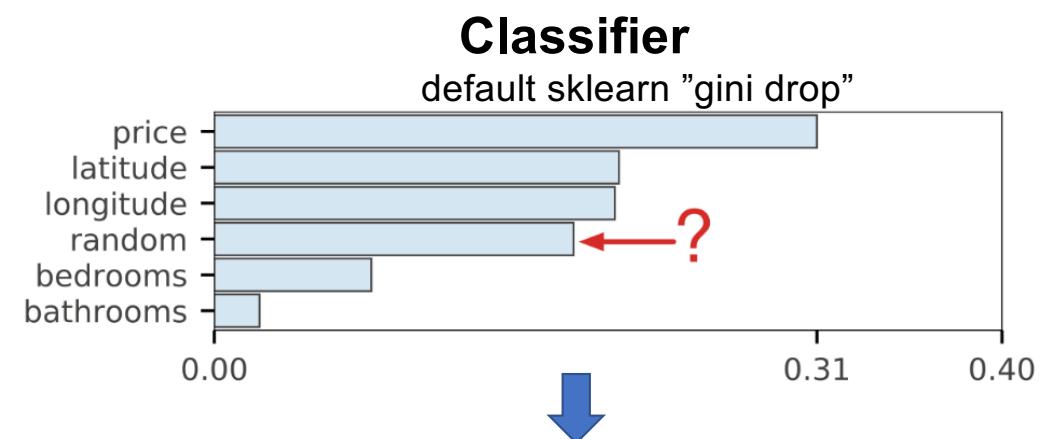
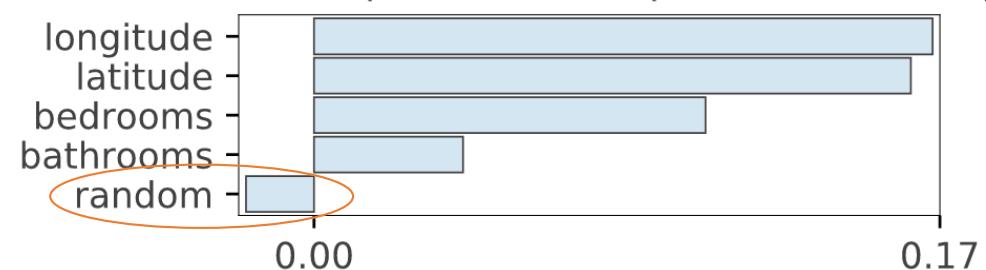
- Brute force mechanism to examine importance of any feature or combination of features
- Procedure:
  1. Compute validation metric for model trained on all features
  2. Drop column  $x_j$  from training set
  3. Retrain model
  4. Compute validation metric set
  5. Importance score is the change in metric
- Answers the question of how loss of a feature affects overall model performance (which might not be actual importance)



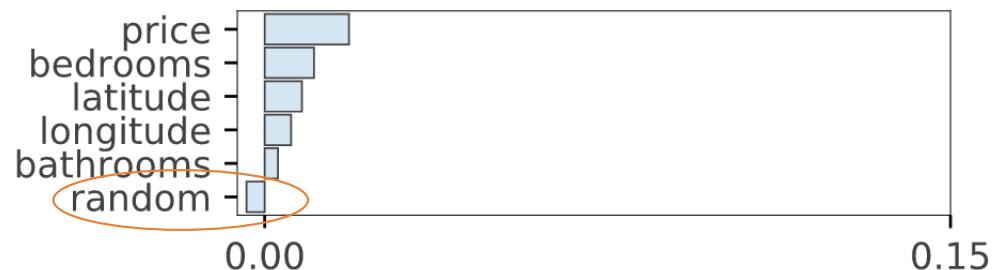
# Compare drop-column to gini/MSE drop



Feature importance via drop in model accuracy



Feature importance via drop in model accuracy



What does negative importance mean? Means dropping improves metric!

# Drop-column implementation

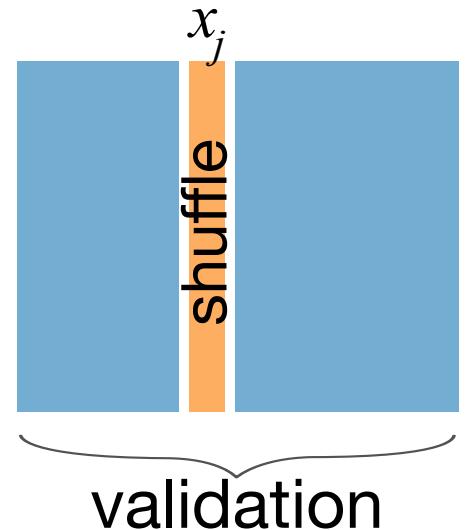
```
def dropcol_importances(model,
                        X_train, y_train, X_valid, y_valid):
    model.fit(X_train, y_train)
    baseline = metric(y_valid, model.predict(X_valid))
    imp = []
    for col in X_train.columns:
        X_train_ = X_train.drop(col, axis=1)
        X_valid_ = X_valid.drop(col, axis=1)
        model_ = clone(model)
        model_.fit(X_train_, y_train)
        m = metric(y_valid, model_.predict(X_valid_))
        imp.append(baseline - m)
    return imp
```

# Drop-column pros/cons

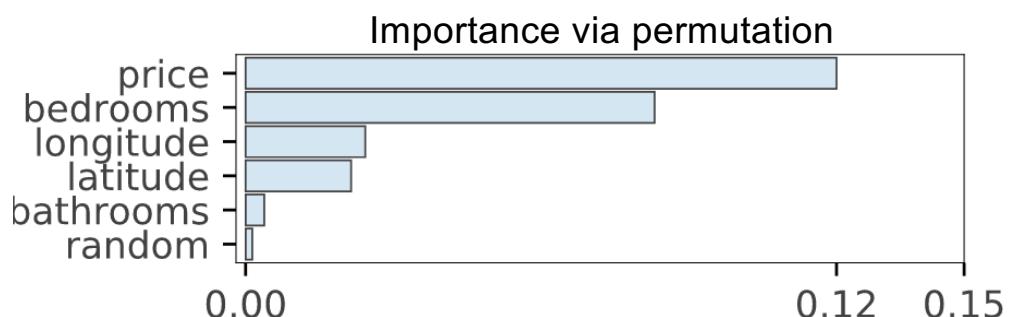
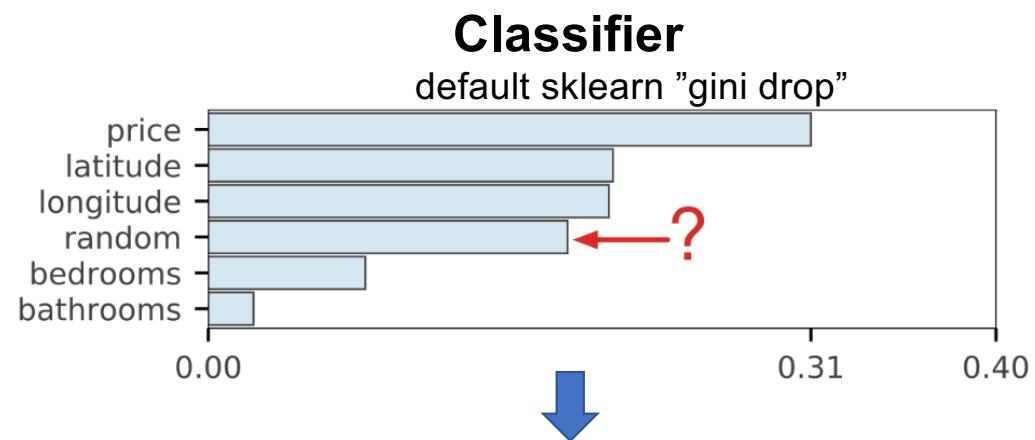
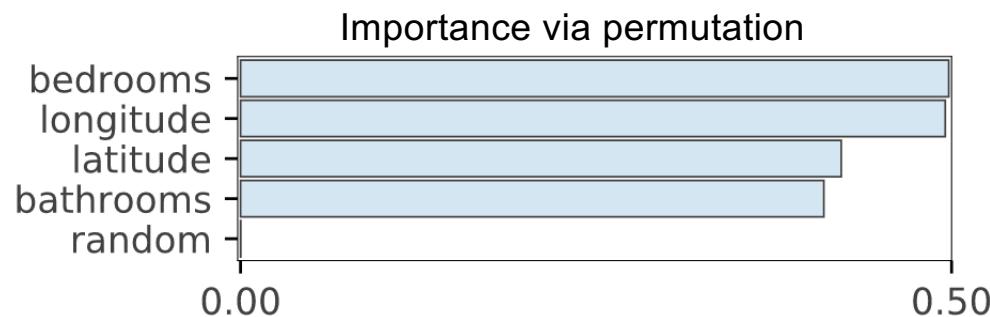
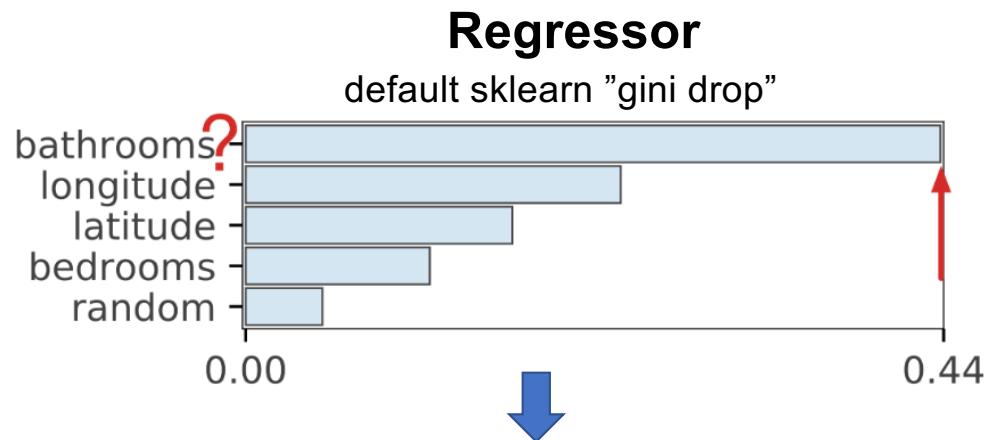
- Easy to understand
- Simple to implement
- Very direct means of measuring importance
- Works for any machine learning model
- BUT, very expensive because it means retraining the model  $p$  times for  $p$  features; try on a subset of the data for speed
- Codependent features often result in 0 or very low importance

# Permutation importance

- Similar to drop column, but permute  $x_j$  instead of dropping it from the model
- Keeps same  $x_j$  distribution but breaks relationships
- Procedure:
  1. Compute validation metric for model trained on all features
  2. Permute column  $x_j$  in validation set
  3. Compute validation metric set
  4. Importance score is the change in metric



# Compare permutation to gini/MSE drop



# Permutation implementation

```
def permutation_importances(model, X_valid, y_valid):
    baseline = metric(y_valid, model.predict(X_valid))
    imp = []
    for col in X_valid.columns:
        save = X_valid[col].copy()
        X_valid[col] = np.random.permutation(X_valid[col])
        m = metric(y_valid, model.predict(X_valid))
        X_valid[col] = save
        imp.append(baseline - m)
    return imp
```

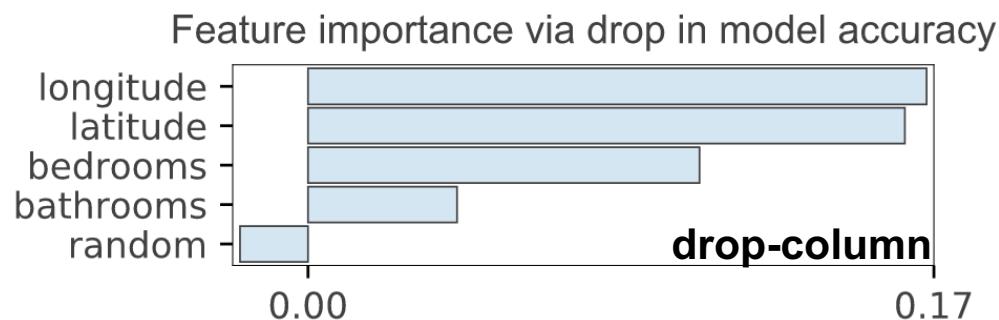
# Permutation importance pros/cons

- Easy to understand
- Simple to implement
- Works for any machine learning model
- No need to retrain the model so much more efficient than drop column importance
- Can create nonsensical records through permutation, such as pregnant male, which makes the results suspect
- Codependent features often share importance, such as longitude and latitude
- Strobl et al “*permutation importance over-estimates the importance of correlated predictor variables*”

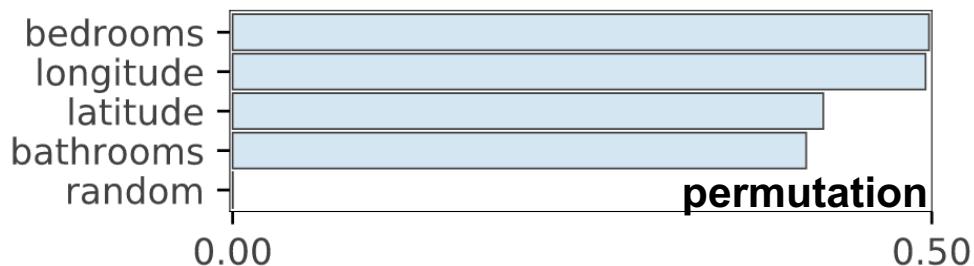
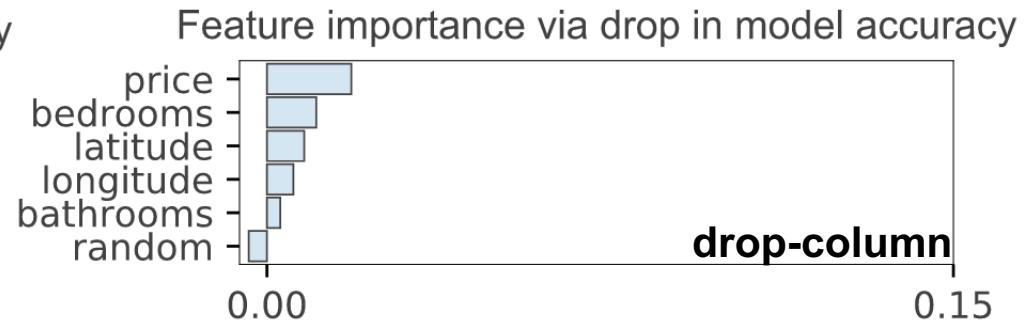
<https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-307>

# Compare drop column to permutation

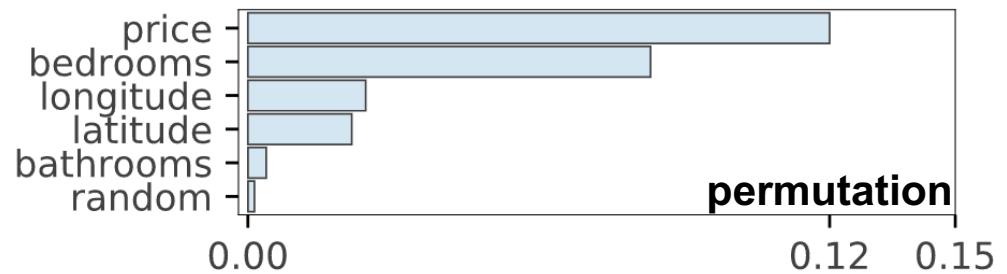
**Regressor**



**Classifier**



(fairly different due to codependence; permutation tends to share importance scores for codependent features)



(very similar)



UNIVERSITY OF SAN FRANCISCO

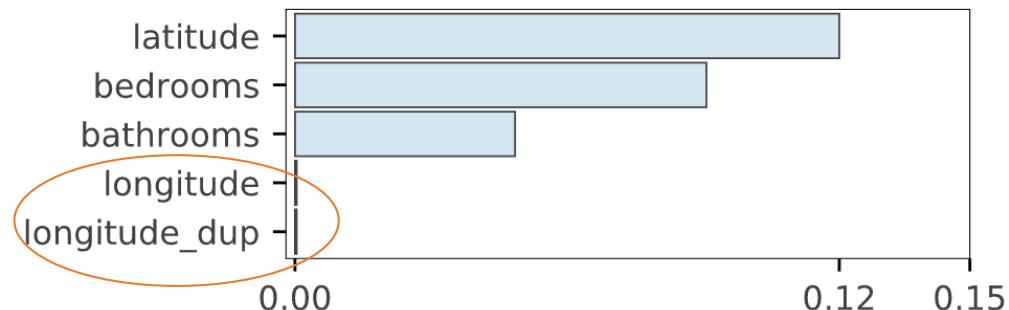
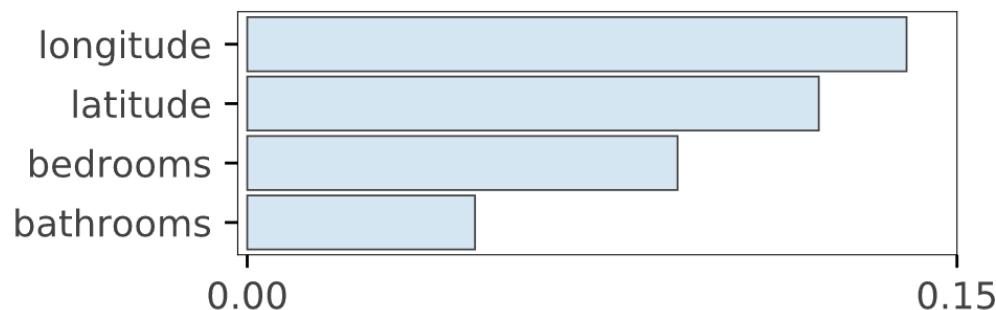
# Interpreting importance results

# Codependent features

- Drop column and permutation importance consider each feature individually, though my **rfpimp** package lets you consider multiple features together
- If all features are totally independent, then computing feature importance individually is no problem
- If, however, two or more features are *codependent* (correlated in some way but not necessarily with a strictly linear relationship) computing feature importance individually can give unexpected results
- Drop-column tends to show low importance scores and permutation tends to share importance scores for codependent features

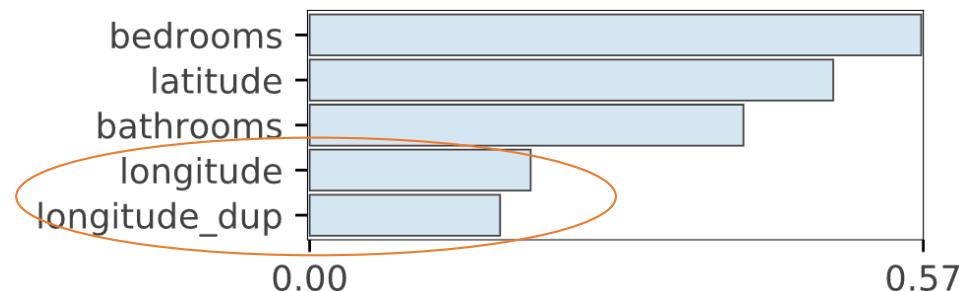
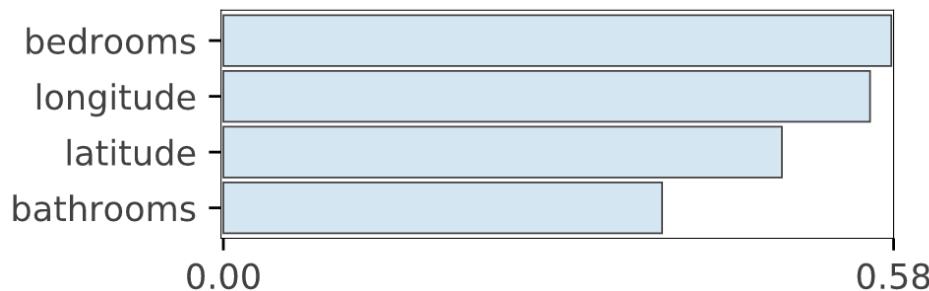
# Effect of duplicated columns on drop column importance

- Compare feature importances for original regressor model and another with duplicated longitude
- Shocking to see BOTH longitude and duplicated longitude both go to zero importance but we measure as drop in accuracy
- Dropping one doesn't affect accuracy; other column covers for it



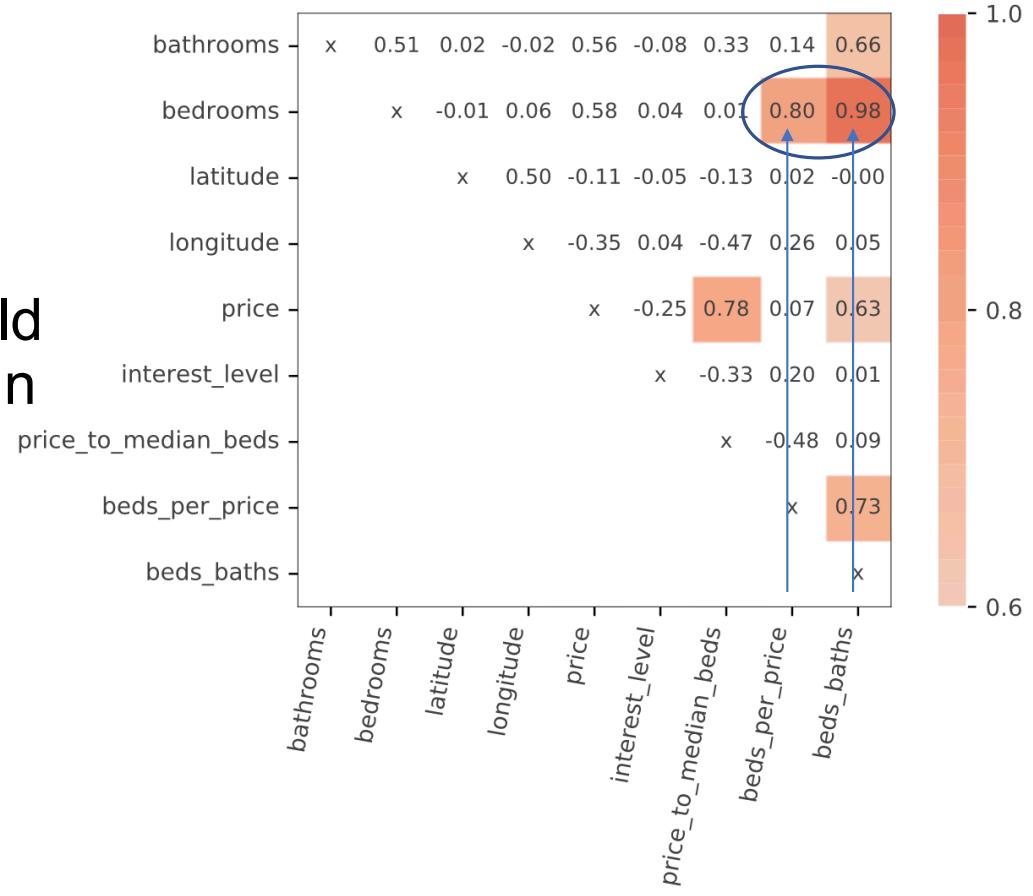
# Effect of duplicated columns on permutation importance

- Consider RFs; during training, node splitting should choose equally important variables roughly 50-50
- Permuting a duplicated column should still allow prediction to be half supported by the other identical column
- That's what we see in practice for duplicated columns; has the effect of pulling down the perceived importance of the original



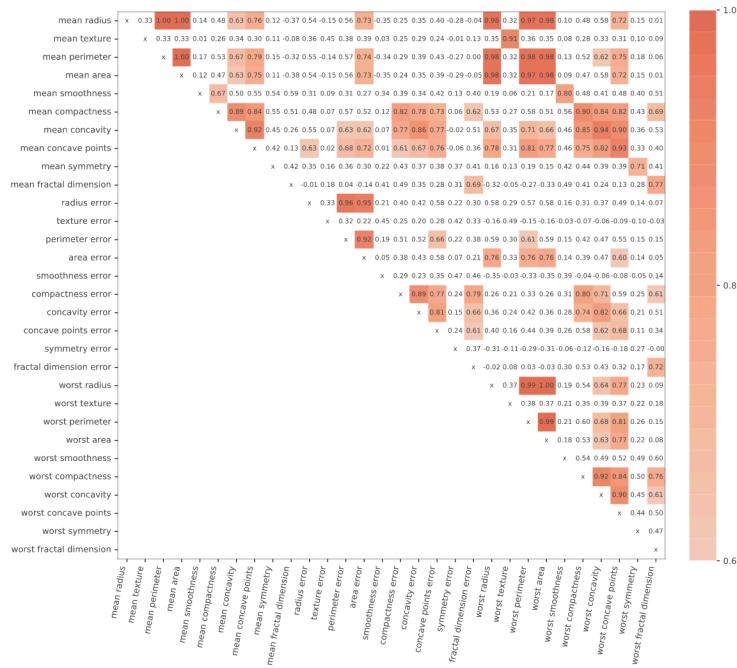
# Spearman's feature heat map

- Spearman's correlation is same as converting two variables to rank values and running standard correlation
- Highly-correlated features should be dropped/permuted together in feature importance metrics to decrease confusion; E.g., as **bedrooms**, **beds\_baths**, and **beds\_per\_price**

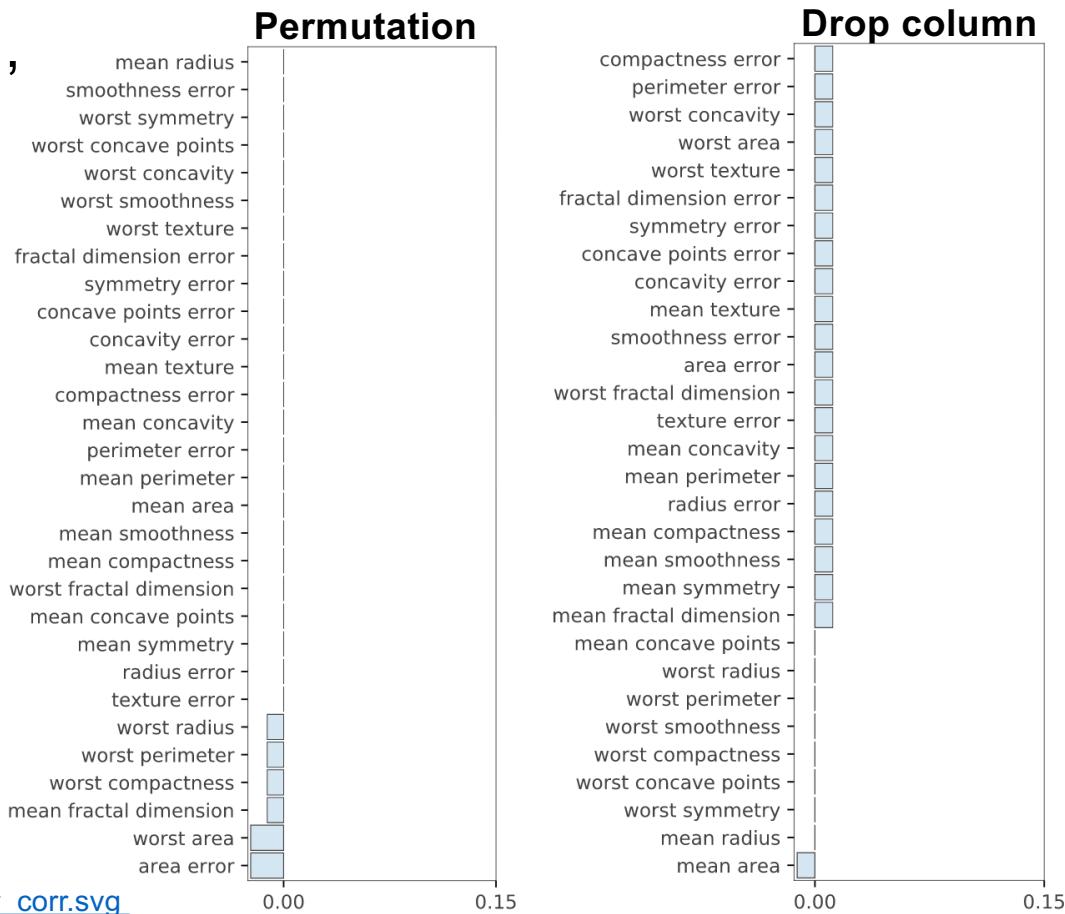


# Ex: Breast cancer correlation matrix

- Features are super correlated, yielding useless feature importances



Larger correlation image [https://explained.ai/rf-importance/images/cancer\\_corr.svg](https://explained.ai/rf-importance/images/cancer_corr.svg)

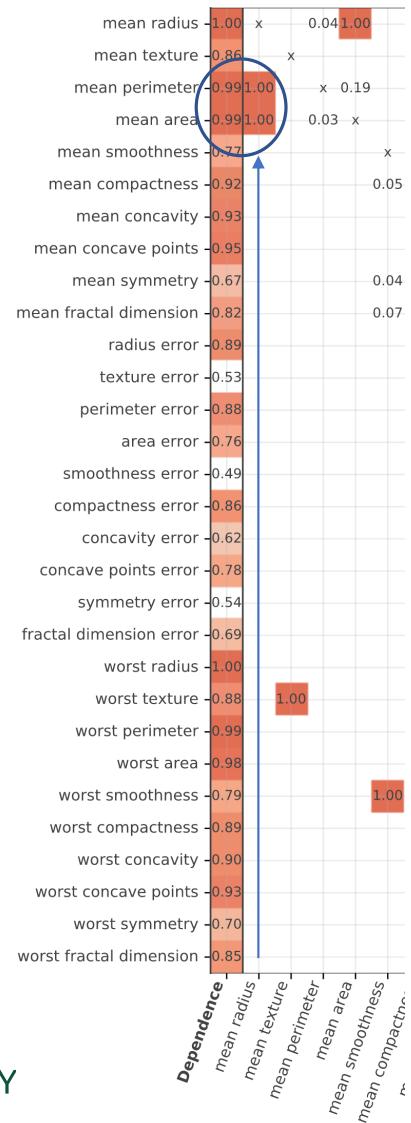


# Feature codependence

(stronger measure than spearman's)

- To identify if  $x_j$  is codependent with other features, train model using  $x_j$  as the target variable and all other features as explanatory variables (multicollinearity)
- $R^2$  prediction error indicates how easy it is to predict feature  $x_j$  using the other features
- Feature importances of  $x_j$  targeted model identify which features are strongly-codependents of  $x_j$
- The higher the score, the more codependent feature  $x_j$  is with other features; can drop all but one in highly-codependent feature group to simplify model
- E.g., **mean radius** is important to predict **mean perimeter** and **mean area**; can probably drop those two
- E.g., could tell you which virus mutations develop together

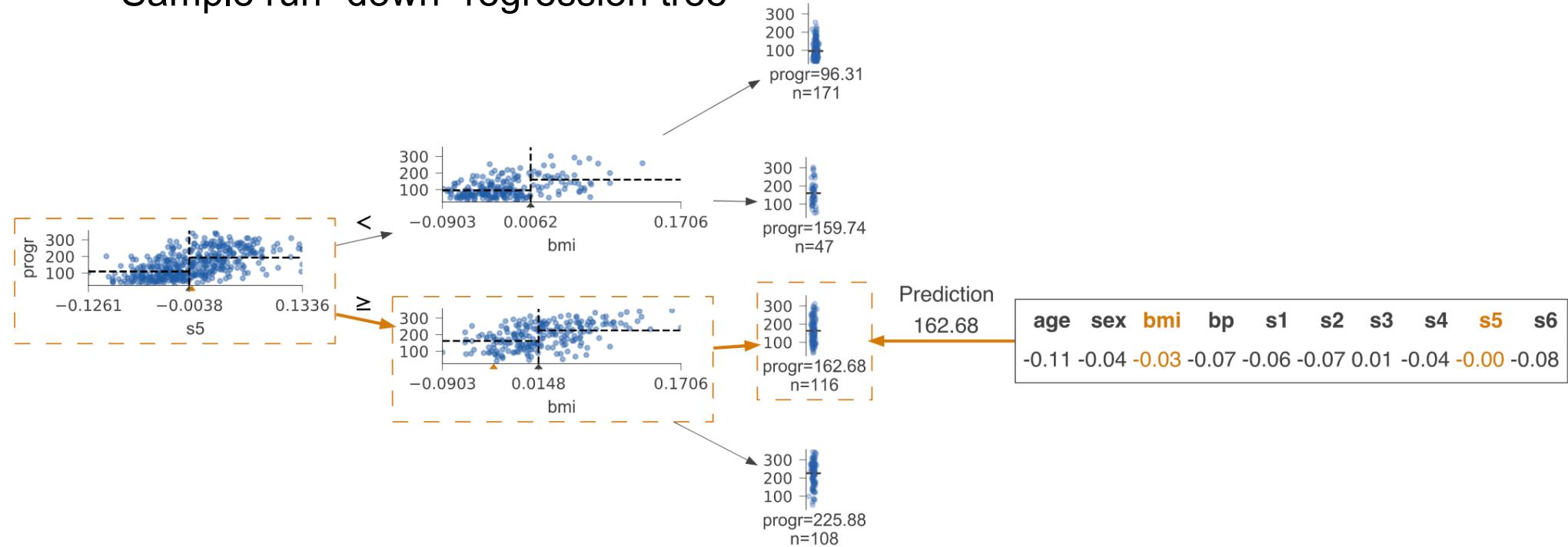
Bigger image [https://explained.ai/rf-importance/images/cancer\\_dep.svg](https://explained.ai/rf-importance/images/cancer_dep.svg)



# Interpreting individual record results

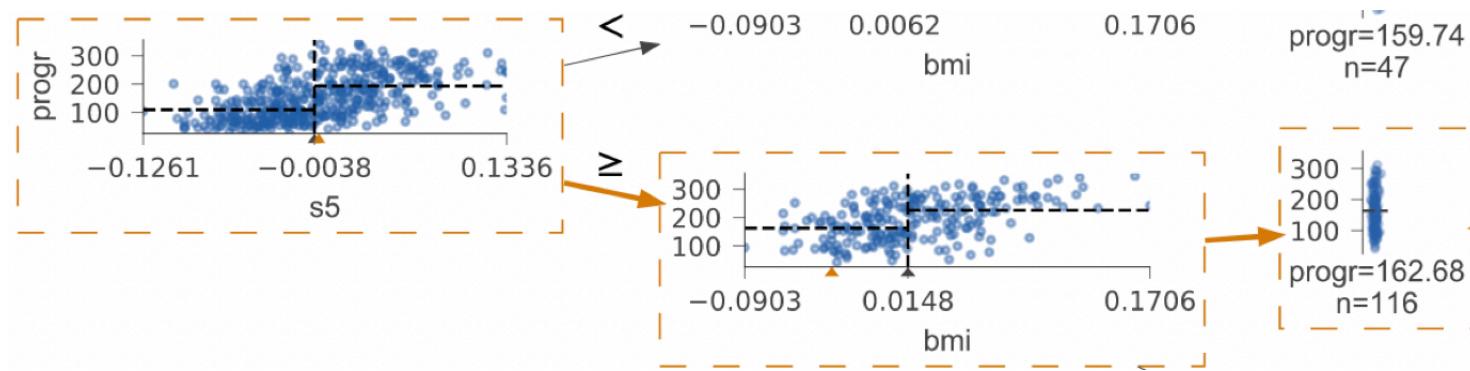
# Viz how a test vector reaches leaf

Sample run "down" regression tree



# Path from root to leaf has useful info!

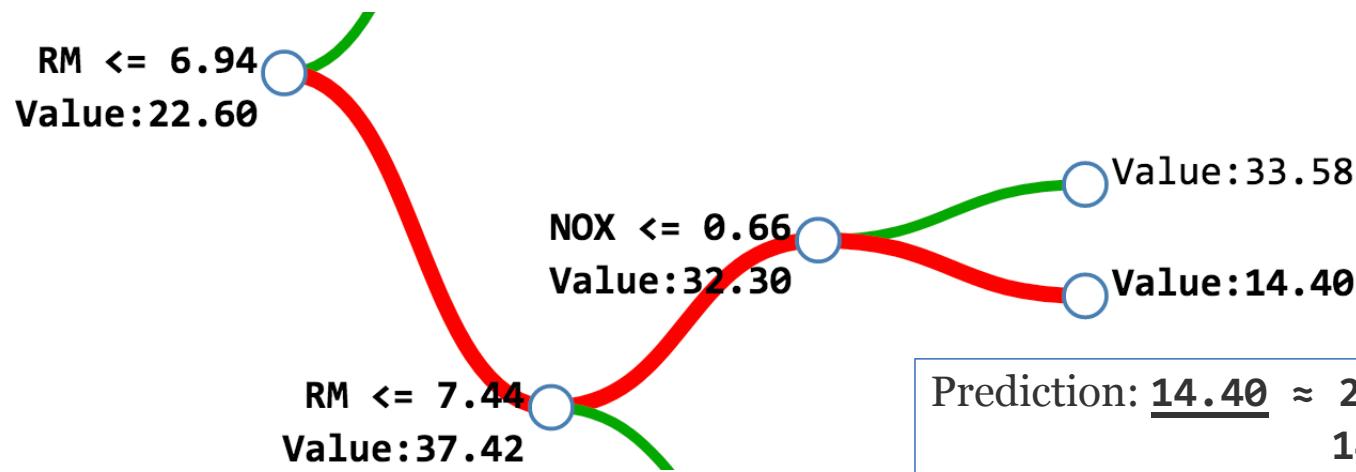
- Which features tested?
- Partitioning of feature space:



- Explain prediction using vars/values:  
*Predict 162.68 because  $s5 > -0.0038$  and  $bmi < 0.0148$*

# Explain regression prediction as sum of contributions along path

- Using change in sample mean from each node
- Start with  $\text{mean}(y)$ , value of root node
- Compute sample mean deltas



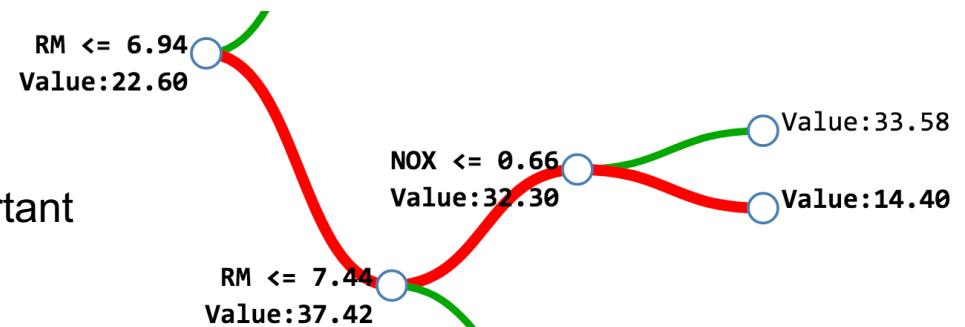
Prediction: 14.40  $\approx$  22.60 (trainset mean) +  
14.82(gain from RM) -  
5.12(loss from RM) -  
17.9(loss from NOX)

Image/example from <http://blog.datadive.net/interpreting-random-forests/>

# Test vector feature importances (regressor example)

- Revisit earlier slide; magnitude and frequency of variable contribution acts like the importance
- Could also use MSE drop similar to gini drop, but drop/gain in value seems more likely to be accurate

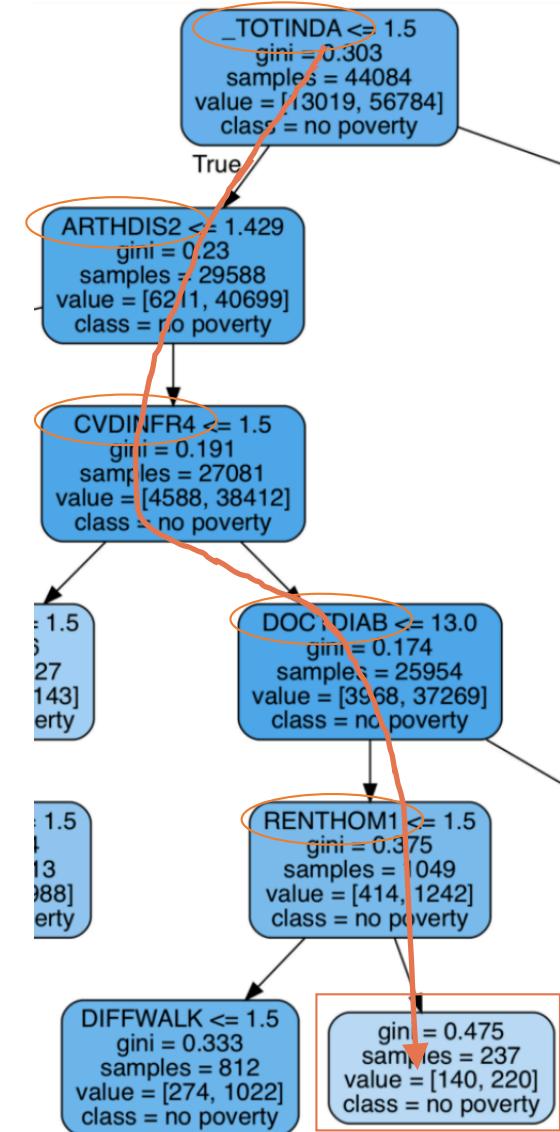
Prediction: 14.40  $\approx$  22.60 (trainset mean) +  
14.82(gain from RM) -  
5.12(loss from RM) -  
17.9(loss from NOX) ← important



# Test vector feature importances (classifier example)

- First approximation: count the number of times each variable referenced along the path from root to the leaf
- Improve by weighting vars by average drop in gini (mimicking ginidrop importance), but for a single record
  - $\text{_TOTINDA drop} = .303 - .23 = 0.073$  ← important
  - $\text{ARTHDIS2 drop} = .23 - .191 = 0.039$
  - $\text{CVDINFR4 drop} = .191 - .174 = 0.017$
  - $\text{DOCTDIAB gain} = .174 - .375 = -.201$  ← negatively important

data set <https://www.kaggle.com/cdc/behavioral-risk-factor-surveillance-system>



# Summary

- Use permutation importance, but check drop-column too
- Use only on stable, accurate model
- We only get relative importances, not proportion of total variance
- A 0 drop-column importance doesn't mean useless; might also be a codependent feature
- Add a noise column; can ignore any vars at or below
- RF specific: can interpret single test record as drop in value or MSE/gini on path from root to leaf
- Compute metric changes on validation not training set
- Feature importances are clues not gospel
- Useful for simplifying model
- Can tell us something about the business application / market