Feature importance

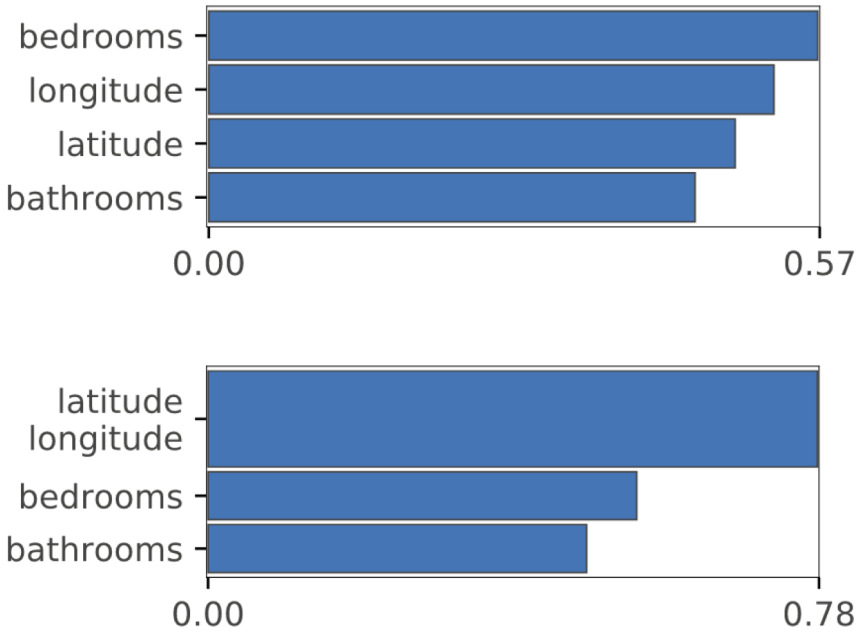
Terence Parr
MSDS program
University of San Francisco

What does the model say about the data?

- A good model is great, but most the time we want to be able to interpret the model; feature importance is most useful interp.
- Feature importance gives a relative ranking of the predictive strength among the features of the model
- Feature importance tells us about the business or application; e.g., what matters to people renting apartments or buying used bulldozers?

Rent example

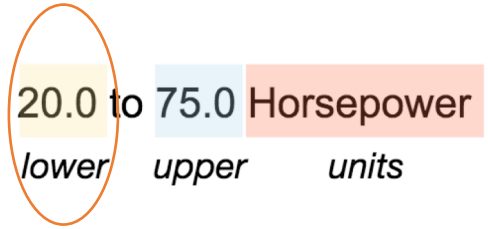
- Number of **bedrooms** appears to be the most important, meaning that it is most predictive of rent price
- That tells us something about the market for New York City rent
- Combined, however, the location matters more by far; note: **longitude** and **latitude** are codependent features



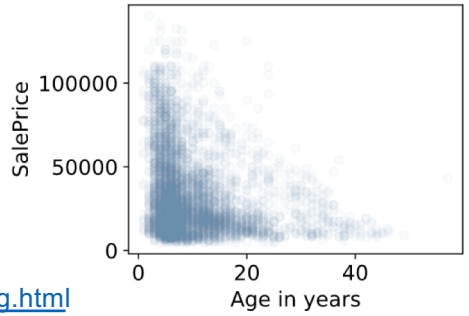
See https://mlbook.explained.ai/first-taste.html

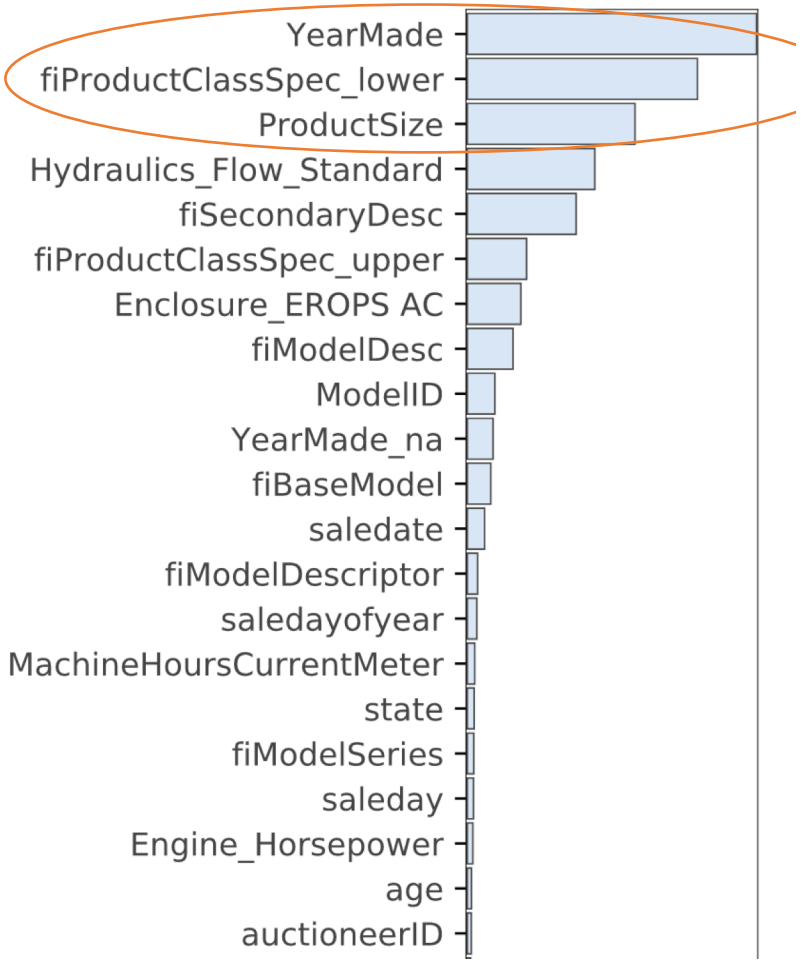
Bulldozer example

- **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:

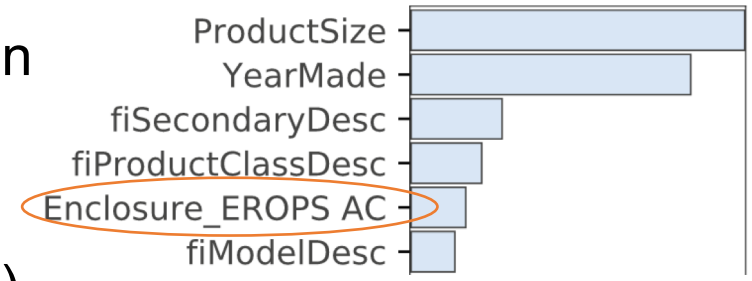




See https://mlbook.explained.ai/bulldozer-feateng.html

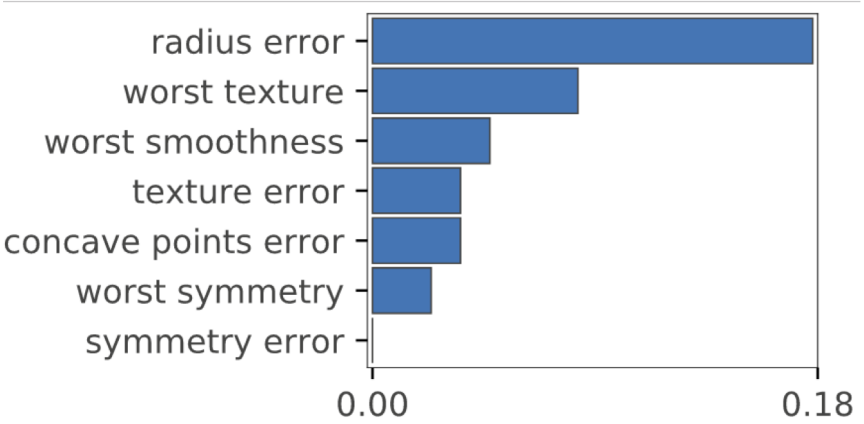
Bulldozer one-hot example

- 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)
- Fewer features increases training and prediction speed

Beware!

- Can't trust feature importances from *weak* models
- Can't trust feature importances from *unstable* models
- Different models give different importances; it's telling you 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 important for generalization
- Even with good model, importances are a clue not gospel

Formal definition of feature importance?

- Not sure there is an agreed-upon definition; scores themselves don't usually mean anything…only relative rank/magnitude matters
- For suitably prepared data and appropriate assumptions, linear model coefficients give the amount of y variance explained; but, not really since R^2 assumes linearity and so on
  (oh, and regularization kills interpretation of coefficients)
- Even if linear model coefficients are what we want, linear models are often too weak in practice
- We still want feature importance to be something like "the amount of y variation explained by feature x_i"
- SHAP lib calls it "average impact on model output magnitude"
- More on difficulties of interpreting regression coefficients by Breiman https://projecteuclid.org/euclid.ss/1009213726

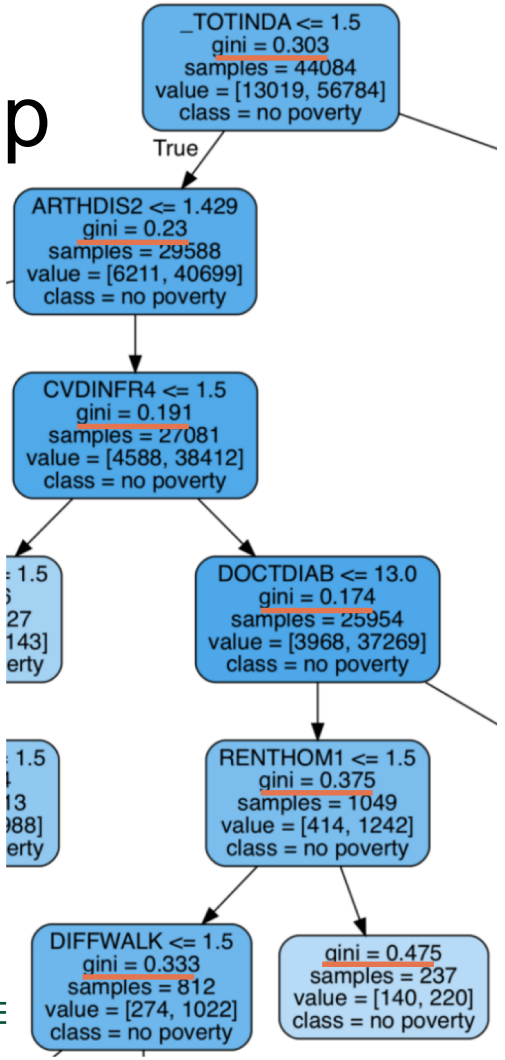
So how do we get feature importance?

- gini/MSE drop (RF specific and sklearn default)
- permutation importance (coming soon to sklearn)
- drop-column importance
- SHAP (https://github.com/slundberg/shap) likely most accurate so far (w/amazing library) but I find it unbearably slow
- James Wilson (MSDS faculty) and I have an idea we hope will be both fast and accurate…coming soon

See **rfpimp** package: https://github.com/parrt/random-forest-importances/blob/master/README.md

Random Forest loss function drop

- Linear models have coefficients to indicate importance and Random Forests also have a built-in mechanism called “gini drop” (for regression, it would be called “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_i across decision nodes for x_i and across all trees gives the feature importance
- Unfortunately, this gives biased feature importances



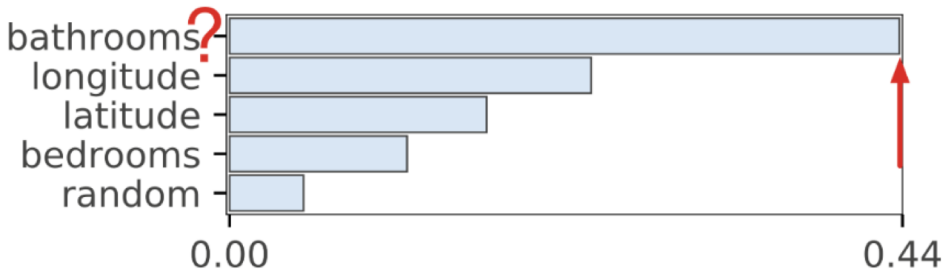
Random Forest gini/MSE drop issues

- Exhaustively testing every unique x_i value when finding decision node splits increases likelihood of finding a x_i value that, purely by chance, happens to predict y well
- That increases the likelihood that variable x_i 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_i) and max(x_i) 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)

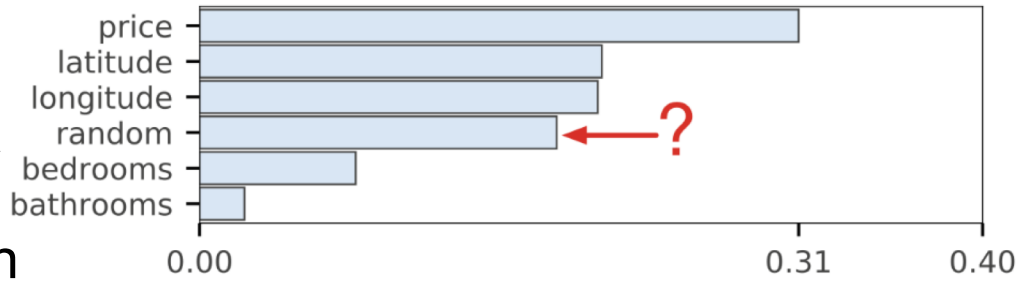
- Don't trust default ("gini drop") importances for RF in sklearn
- Here we see the unlikely idea that New Yorkers care most about bathrooms and much more than location or bedrooms
- New Yorkers can be weird, but they can't be this fixated on bathrooms
- Random noise column is last (which it should be)



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

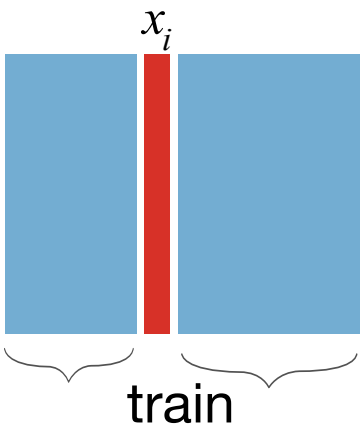
Trouble in paradise
(classification: predicting interest in apt web page)

- 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



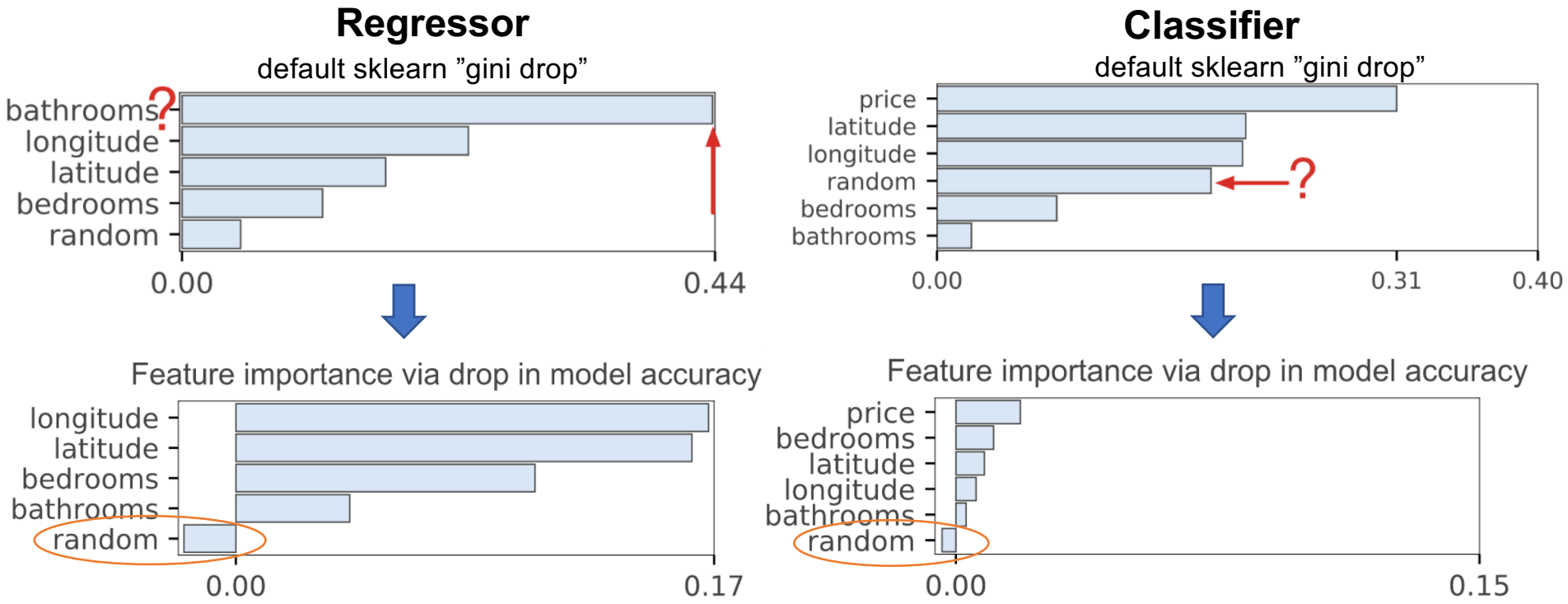
Hmm… what can we do instead?

Drop-column importance



- Brute force mechanism to examine importance of any feature or combination of features
- Procedure:
  1. Compute metric for model trained on all features using validation set
  2. Drop column x_i
  3. Retrain model
  4. Compute metric using validation set
  5. Importance score is the drop 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



What does negative importance mean? Means dropping improves metric!

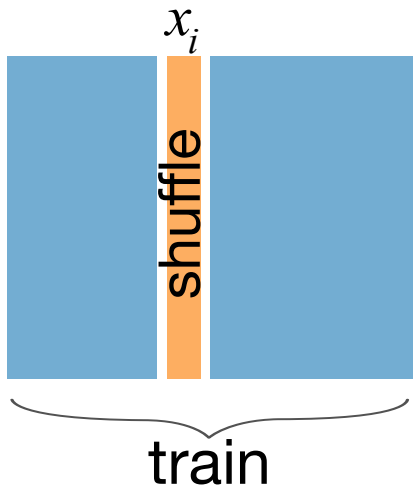
Drop-column implementation

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

Drop-column pros/cons

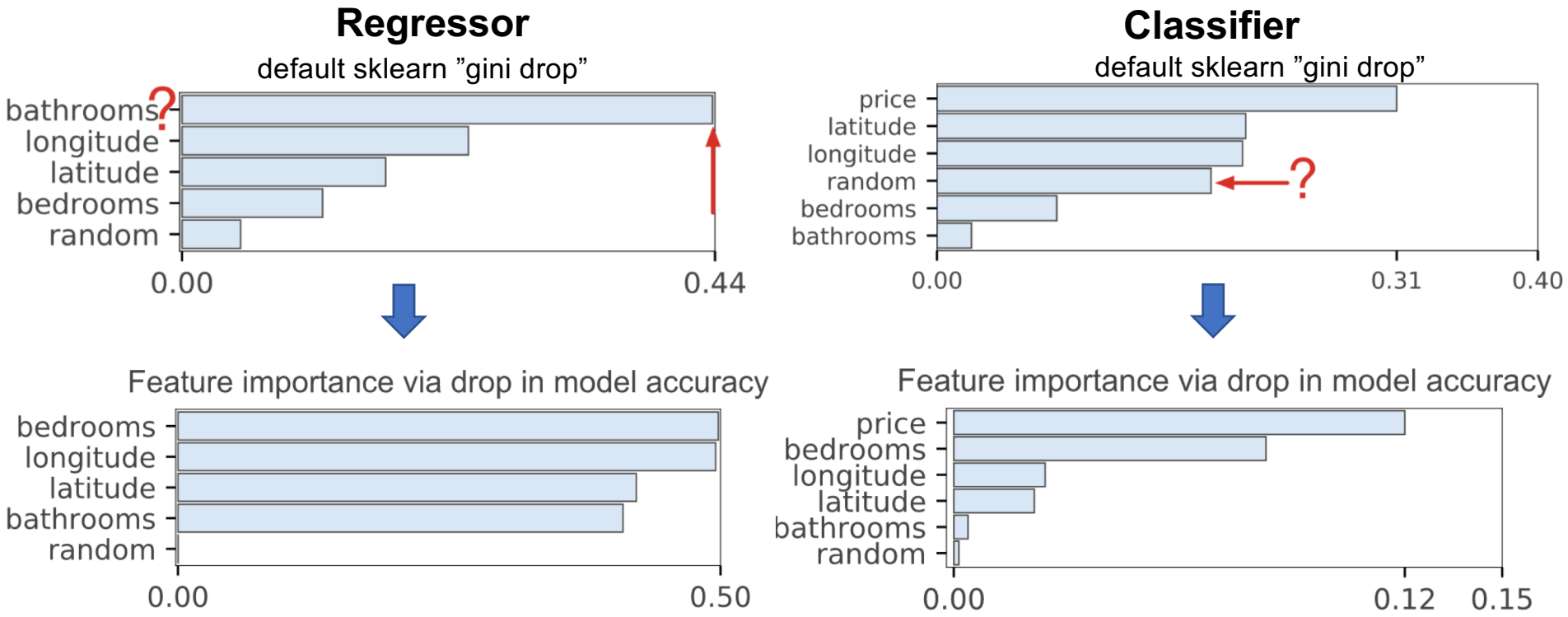
- Easy to understand
- Simple to implement
- Very direct means of measuring importance
- Works for any machine learning model
- BUT, very expensive because it means retraining the model p times for p features; try on a subset of the data for speed
- Codependent features often result in 0 or very low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

Permutation importance



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

Compare permutation to gini/MSE drop



Permutation implementation

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

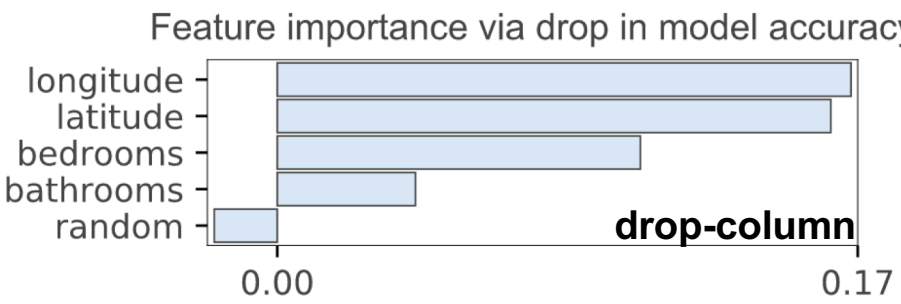
(Should really be computing metric on validation set)

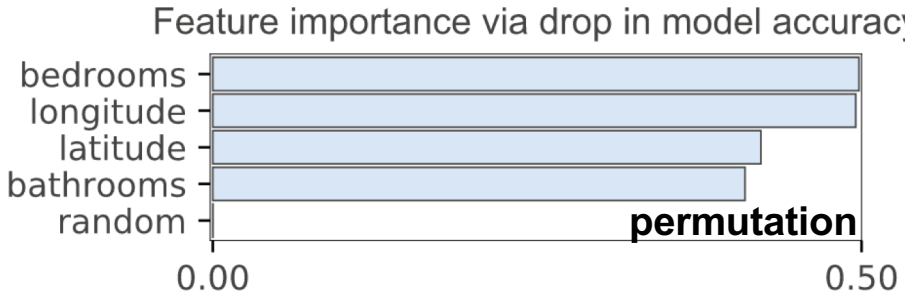
Permutation importance pros/cons

- 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

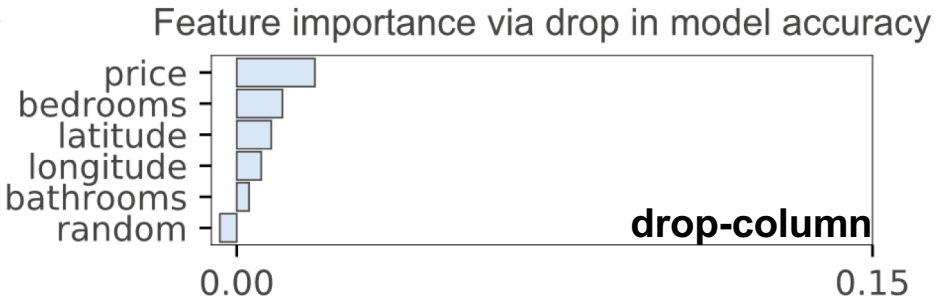
Regressor

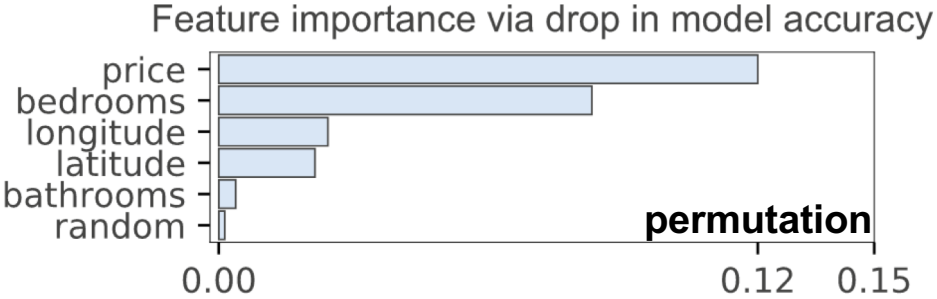




(fairly different due to codependence)

Classifier





(very similar)

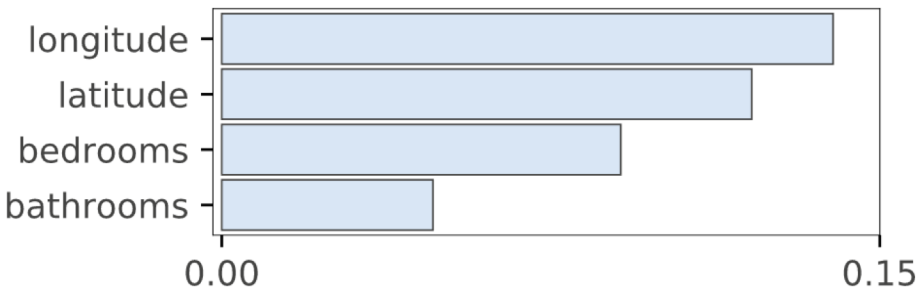
Interpreting importance results

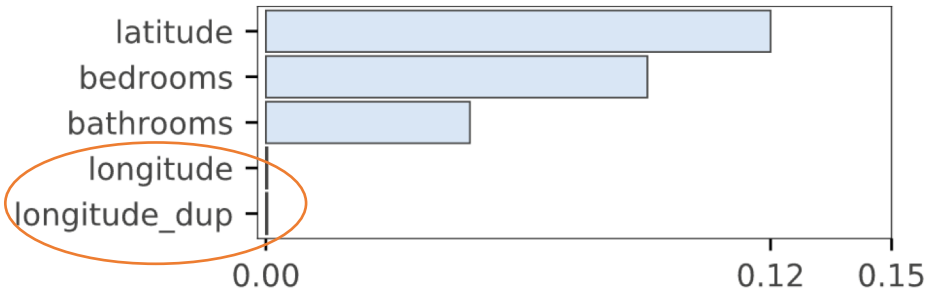
Codependent features

- 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

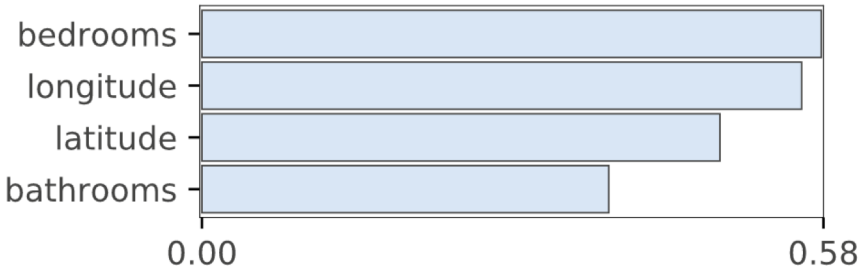
- Compare feature importances for original regressor model and another with duplicated longitude
- Shocking to see BOTH longitude and duplicated longitude both go to zero importance but we measure as drop in accuracy
- Dropping one doesn't affect accuracy; other column covers for it

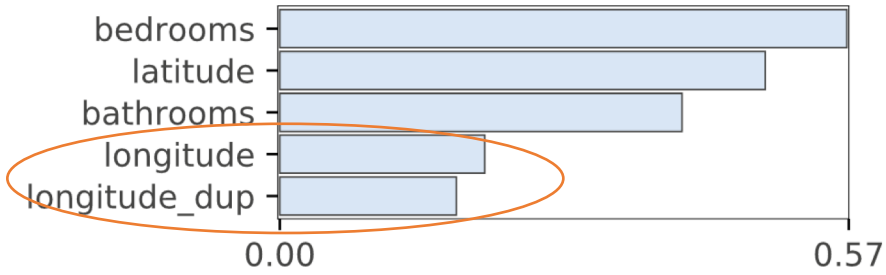




Effect of duplicated columns on permutation importance

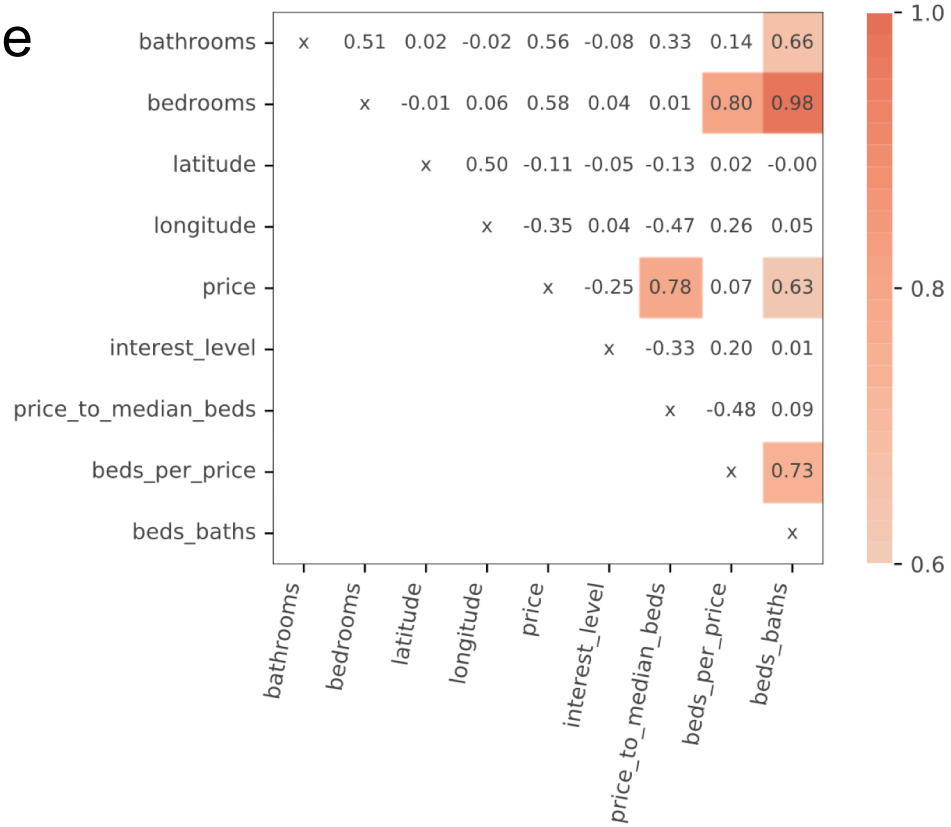
- Consider RFs; during training, node splitting should choose equally important variables roughly 50-50
- Permuting a duplicated column should still allow prediction to be half supported by the other identical column
- That's what we see in practice for duplicated columns; has the effect of pulling down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





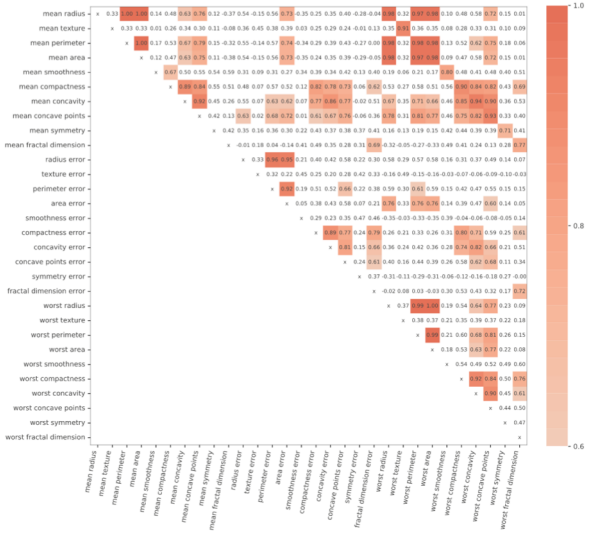
Feature spearman's R heat map

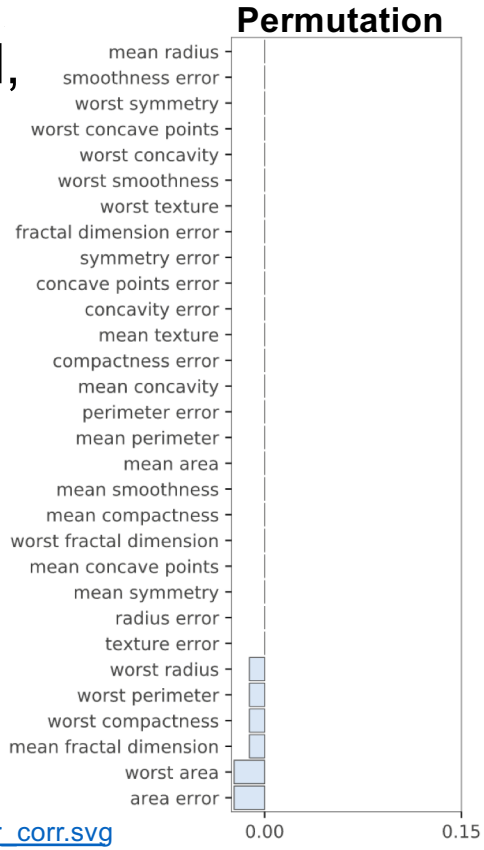
- Spearman's correlation is same as converting two variables to rank values and running standard correlation
- Highly-correlated features should be dropped/permuted together, such as **bedrooms**, **beds_baths**, and **beds_per_price**

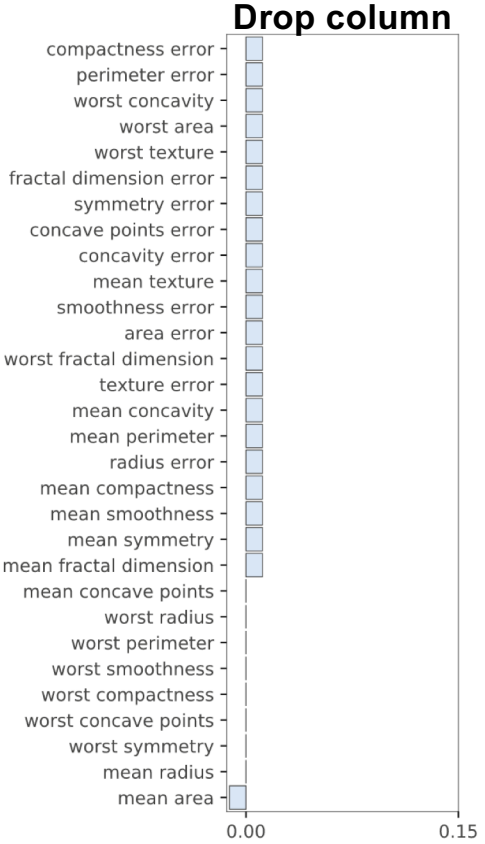


Breast cancer correlation matrix

- Features are super correlated, yielding useless feature importances



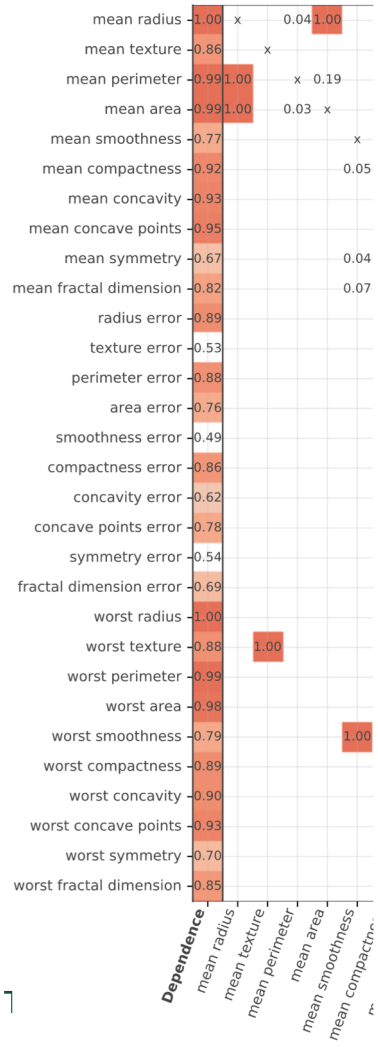




Larger correlation image https://explained.ai/rf-importance/images/cancer_corr.svg

Feature codependence

- To identify if x_i is dependent on other features, train model using x_i 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_i using the other features
- Feature importances of x_i targeted model identify which features are strongly-codependent of x_i
- The higher the score, the more codependent feature x_i 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



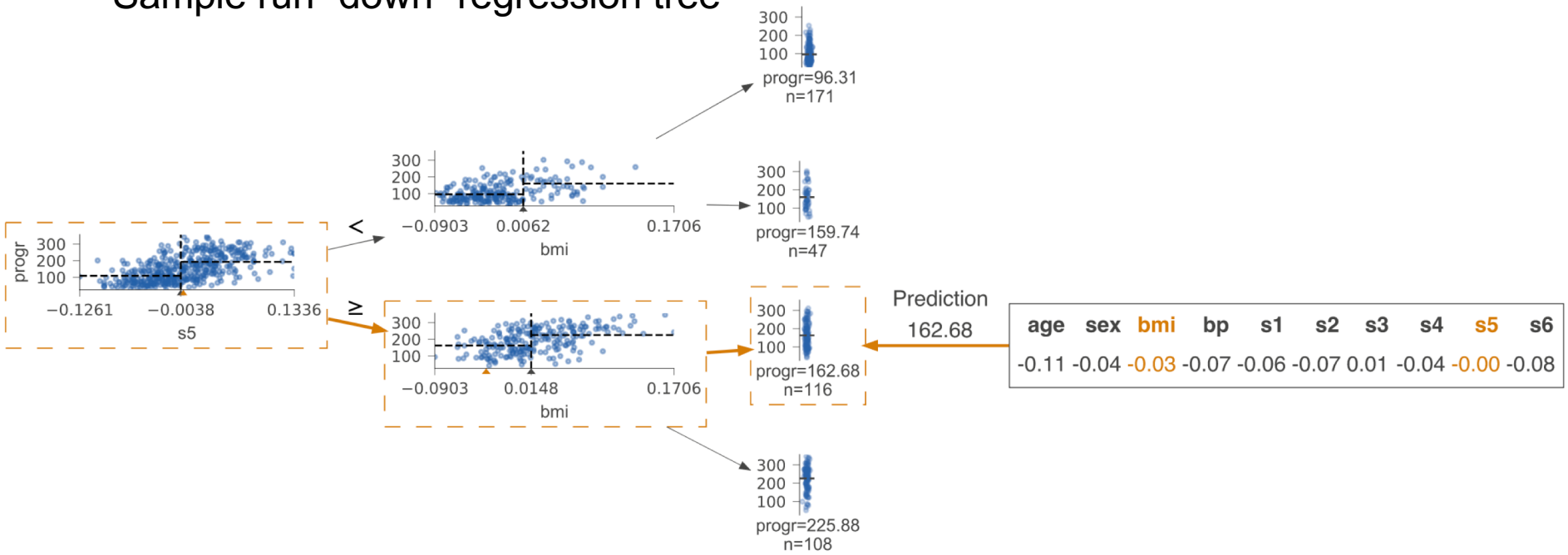
Bigger image https://explained.ai/rf-importance/images/cancer_dep.svg

UNIVERSITY

Interpreting individual record results

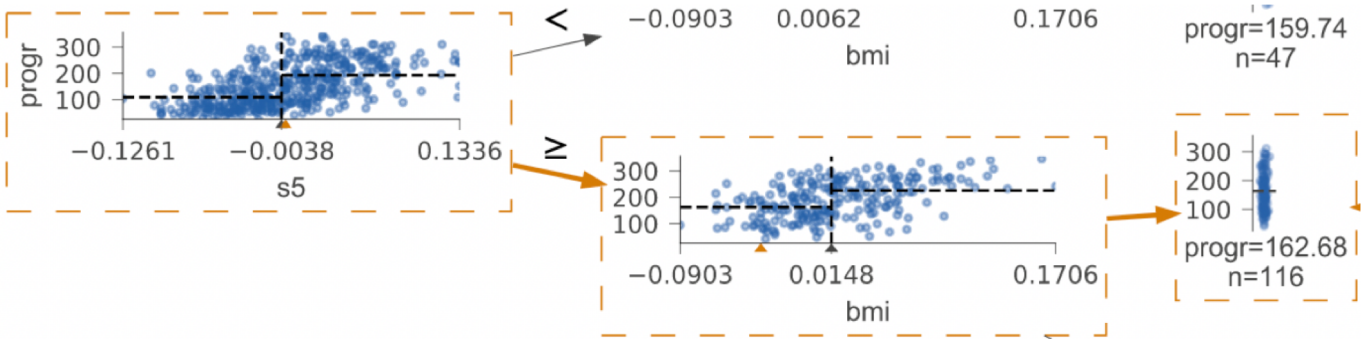
Viz how a test vector reaches leaf

Sample run ”down” regression tree



Path from root to leaf has useful info!

- Which features tested?
- Partitioning of feature space: *s5 > -0.0038 and bmi < 0.0148*

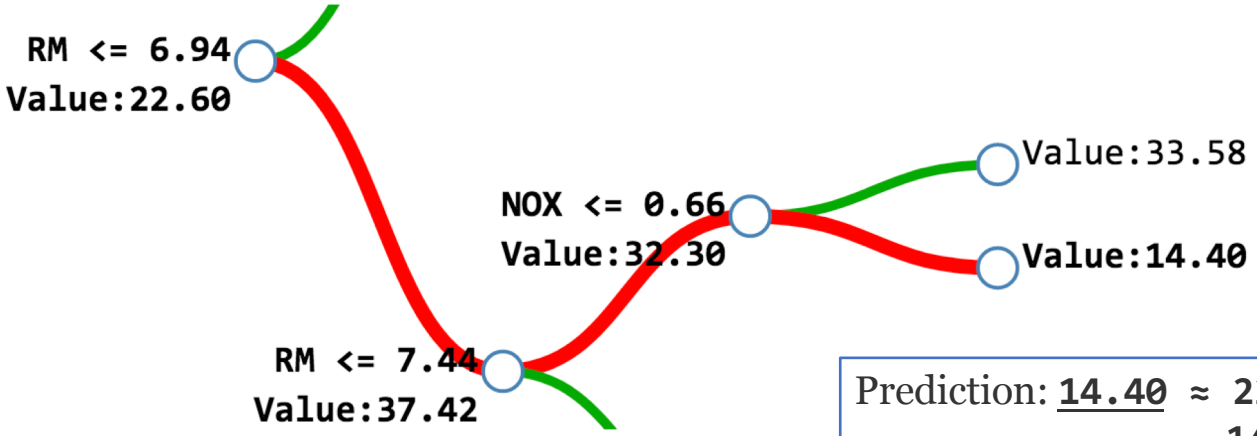


- Explain prediction using vars/values:

  Predict 162.68 because s5 > -0.0038 and bmi < 0.0148

Regression as sum of contributions along path

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



Prediction: **14.40** ≈ **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/

Extend sum of contributions to Random Forest regression

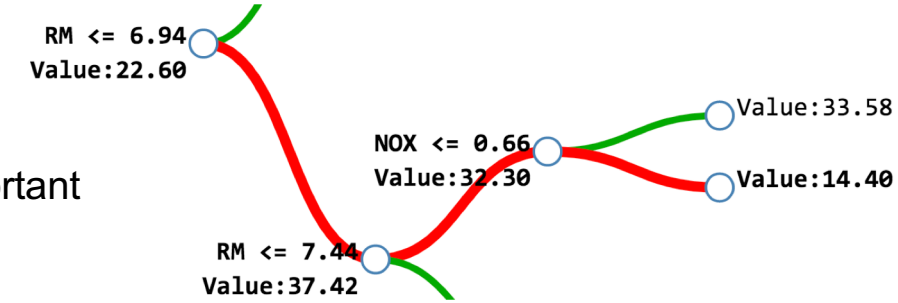
- Compute the average mean(y_b) of all bootstrapped y as the base contribution
- Compute the average drop or gain (delta) for each move through each tree and through all trees in forest
- Some variables will have more deltas than others; some will have no contributions
- The prediction is roughly base + average contribution for each x_i

Test vector feature importances (regressor example)

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

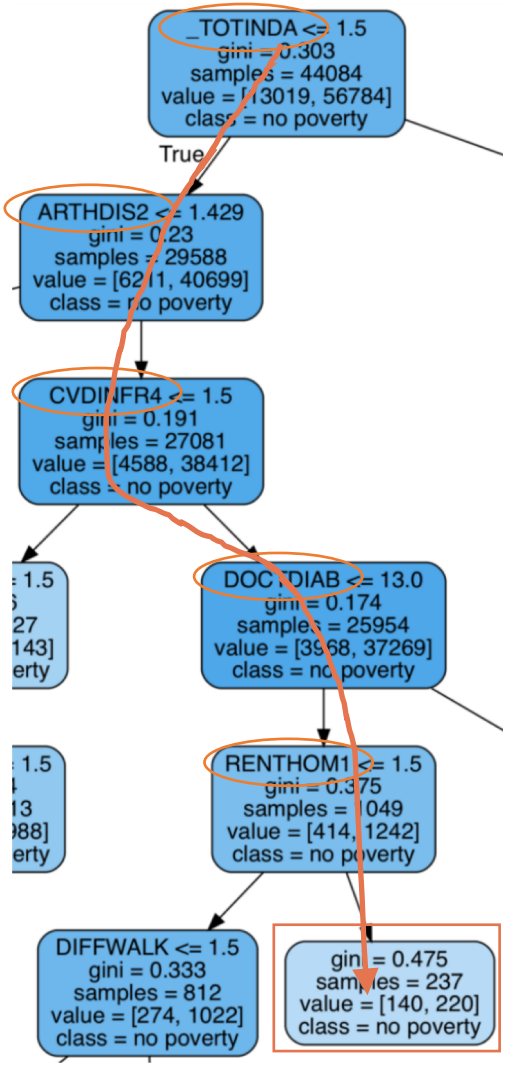
Prediction: **14.40** ≈ **22.60** (trainset mean) **+**
14.82(gain from RM) **-**
5.12(loss from RM) **-**
17.9(loss from NOX)

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 MSE or gini (mimicking ginidrop importance), but for a single record
  - _TOTINDA drop = .303 - .23 = 0.073 ⟵ important
  - ARTHDIS2 drop = .23 - .191 = 0.039
  - CVDINFR4 drop = .191 - .174 = 0.017
  - 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 on validation not training set
- Feature importances are clues not gospel
- Useful for simplifying model
- Can tell us something about the business application / market