ENSEMBLE METHODS

A. Sanchez, F. Reverter and E. Vegas

INTRODUCTION TO ENSEMBLES

SOME PROBLEMS OF *WEAK* LEARNERS.

- Decision trees have many good properties but some important drawbacks:
    - Smaller accuracy than competing alternatives
    - Very sensitive to small changes in data
    - Overall it makes the highly variable predictors
- Tree are not the only classifers to suffer from such problems.

ENSEMBLES

- A common strategy to deal with these issues is to build repeated (weak learners) models on the same data and combine them to form a single result.
- These are called *ensemble* or consensus estimators/predictors.
- As a general rule, ensemble learners tend to improve the results obtained with the weak learners they are made of.

ENSEMBLE METHODS

- Ensemble can be built on different learners but we will focus on those built on trees:
    - Bagging,
    - Random Forests,
    - Boosting,
    - Bayesian Trees.

BAGGING: AGGREGATING PREDICTORS

BAGGING: BOOTSTRAP AGGREGATION

- Decision trees suffer from high variance when compared with other methods such as linear regression, especially when n/p is moderately large.
  - *NOTE: Write a small script to check this assertion*
- Given that high variance is intrinsec to the trees a possibility, suggested by Breimann (**Breiman 1996**), is to build multiple trees derived from the same dataset and, somehow, average them.

AVERAGING DECREASES VARIANCE

- Bagging relies, informally, on the idea that:
  - given $X \sim F()$, s.t. $Var_F(X) = \sigma^2$,
  - given a s.r.s. $X_1, \ldots, X_n$ from F then
  - if $\overline{X} = \frac{1}{N} \sum_{i=1}^{n} X_i$ then $var_F(\overline{X}) = \sigma^2/n$.
- That is, *relying on the sample mean instead of on simple observations decreases variance by a factor of* n.

AVERAGING TREES ...

Two questions arise here:

1. How to go from X to $X_1, \ldots, X_n$?

- This will be done using *bootstrap resampling*.

2. What means “averaging” in this context.

- Depending on the type of tree:
  - Average predictions for regression trees.
  - Majority voting for classification trees.

THE BOOTSTRAP

- *Bootstrap* methods were introduced by Bradley Efron in 1979 (**Efron 1979**) to estimate the standard error of a statistic.
- The success of the idea lied in that the procedure was presented as ``automatic”, that is:
  - instead of having to do complex calculations,
  - it allowed to approximate them using computer simulation.
- Some people called it ``the end of mathematical statistics”.

BOOTSTRAP APPLICATIONS

- The bootstrap has been applied to almost any problem in Statistics.
  - Computing standard errors,
  - Bias estimation and adjustment,
  - Confidence intervals,
  - Significance tests, …
- We begin with the easiest and best known case: *estimating the standard error (that is the square root of the variance) of an estimator*.

PRECISION OF AN ESTIMATE (1)

- Assume we want to estimate some parameter θ, that can be expressed as $\theta(F)$, where F is the distribution function of each X_i in $(X_1, X_2, \ldots, X_n)$.
- For example:

$$\theta = E_F(X) = \theta(F)$$
$$\theta = Med(X) = \{m : P_F(X \leq m) = 1/2\} = \theta(F).$$

PLUG-IN ESTIMATES

- To estimate $\theta(F)$ we usually rely on *plug-in estimators*: $\hat{\theta} = \theta(F_n)$:

$$\hat{\theta} = \overline{X} = \int X dF_n(x) = \frac{1}{n}\sum_{i=1}^{n} x_i = \theta(F_n)$$

$$\hat{\theta} = \widehat{Med}(X) = \{m : \frac{\# x_i \leq m}{n} = 1/2\} = \theta(F_n)$$

PRECISION OF AN ESTIMATE (1)

- An important when computing an estimator $\hat{\theta}$ of a parameter θ is *how precise is $\hat{\theta}$ as an estimator of θ?*
  - With the sample mean, $\overline{X}$, the standard error estimation is immediate because the expression of the variance estimator is known: $ _{}= $
  - So, a natural estimator of the standard error of $\overline{X}$ is: $\hat{\sigma}_{\overline{X}} = \frac{\hat{\sigma}(X)}{\sqrt{n}}$

PRECISION OF AN ESTIMATE (2)

- If, as in this case, the variance of X (and, here, that of $\overline{X}$) is a functional of F:

$$\sigma_{\overline{X}} = \frac{\sigma(X)}{\sqrt{n}} = \frac{\sqrt{\int [x - \int x\, dF(x)] \backslash \text{sp} 2 dF(x)}}{\sqrt{n}} = \sigma_{\overline{X}}(F)$$

then, the standard error estimator is the same functional applied on F_n, that is:

$$\hat{\sigma}_{\overline{X}} = \frac{\hat{\sigma}(X)}{\sqrt{n}} = \frac{\sqrt{1/n \sum_{i=1}^{n} (x_i - \overline{x})^2}}{\sqrt{n}} = \sigma_{\overline{X}}(F_n).$$

STANDARD ERROR ESTIMATION

- Thus, a way to obtain a standard error estimator $\widehat{\sigma}_{\hat{\theta}}$ of an estimator $\hat{\theta}$ consists on replacing F with F_n in the ``population" standard error expression of $\hat{\theta}$, $\sigma_{\hat{\theta}} = \sigma_{\hat{\theta}}(F)$, **whenever it is known**.
- In a schematic form:
  $$\boldsymbol{\sigma_{\hat{\theta}} = \sigma_{\hat{\theta}}(F) \Longrightarrow \sigma_{\hat{\theta}}(F_n) = \widehat{\sigma}_{\hat{\theta}}}.$$
  That is, *the process consists of "plugging-in" F_n in the (known) functional form, $\sigma_{\hat{\theta}}(F)$ that defines $\sigma_{\hat{\theta}}$}.*

THE BOOTSTRAP (1)

- The previous approach, $F \simeq F_n \Longrightarrow \sigma_{\hat{\theta}}(F) \simeq \sigma_{\hat{\theta}}(F_n)$ presents the obvious drawback that, when the functional form $\sigma_{\hat{\theta}}(F)$ is unknown, it is not possible to carry out the substitution of F by F_n.
- This is, for example, the case of standard error of the median or **that of the correlation coefficient**.

THE BOOTSTRAP (2)

- The *bootstrap* method makes it possible to do the desired approximation:
  $\hat{\sigma}_{\hat{\theta}} \simeq \sigma_{\hat{\theta}}(F_n)$
  without having to to know the form of $\sigma_{\hat{\theta}}(F)$.
- To do this,*the bootstrap estimates, or directly approaches* $\sigma_{\hat{\theta}}(F_n)$ *over the sample*.

BOOTSTRAP SAMPLING (*RESAMPLING*)

- The *bootstrap* allows to estimate the standard error from samples of F_n, that is,
- Substituting F_n by F carried out in the *sampling step*.

Instead of:

$$F \xrightarrow{s.r.s} \mathbf{X} = (X_1, X_2, \ldots, X_n) \quad (\hat{\sigma}_{\hat{\theta}} = \underbrace{\sigma_\theta(F_n)}_{unknown})$$

It is done:

$$F_n \xrightarrow{s.r.s} \quad \mathbf{X}^* = (X_1^*, X_2^*, \ldots, X_n^*) \quad (\hat{\sigma}_{\hat{\theta}} = \hat{\sigma}^*_{\hat{\theta}} \simeq \sigma^*_{\hat{\theta}}).$$

BOOTSTRAP RESAMPLING (2)

- Here, $\sigma^*_{\hat{\theta}}$ is the bootstrap standard error of $\hat{\theta}$ and
- $\hat{\sigma}^*_{\hat{\theta}}$ the bootstrap estimate of the standard error of $\hat{\theta}$.
- That is, the new (re-)sampling process consists of *extracting samples of size n of F_n*:
  $\mathbf{X}^* = (X_1^*, X_2^*, \ldots, X_n^*)$ is a random sample of size n obtained *with replacement* from the original sample $(X_1, X_2, \ldots, X_n)$.
- Samples $\mathbf{X}^*$, obtained through this procedure are called *bootstrap* samples or *re-samples*.

THE BOOTSTRAP DISTRIBUTION

- The distribution of a statistic computed from re-samples is called the *bootstrap distribution*,

$$\mathcal{L}(\hat{\theta}) \simeq P_F(\hat{\theta} \leq t) : \text{Sampling distribution of } \hat{\theta},$$
$$\mathcal{L}(\hat{\theta}^*) \simeq P_{F_n}(\hat{\theta}^* \leq t) : \text{Bootstrap distribution of } \hat{\theta},$$

- This distribution is usually not known.
- However the sampling process and the calculation of the statistics can be approximated using a Monte Carlo Algorithm.

BOOTSTRAP MONTE CARLO ALGORITHM

1. Draw a bootstrap sample, $\mathbf{x}_1^*$ from F_n and compute $\hat{\theta}(\mathbf{x}_1^*)$.
2. Repeat (1) B times yielding $\hat{\theta}(\mathbf{x}_2^*)$, …, $\hat{\theta}(\mathbf{x}_B^*)$ estimates.
3. Compute:

$$\hat{\sigma}_B(\hat{\theta}) = \sqrt{\frac{\sum_{b=1}^{B}\left(\hat{\theta}(\mathbf{x_i^*}) - \overline{\hat{\theta}^*}\right)^2}{(B-1)}}, \quad \overline{\hat{\theta}^*} \equiv \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}\left(\mathbf{x}_b^*\right)$$

BOOTSTRAP ESTIMATES OF SE

- Main idea is that the *bootstrap* standard error of $\hat{\theta}$, $\sigma_B(\hat{\theta})$ can be *approximated* by $\hat{\sigma}_B(\hat{\theta})$.

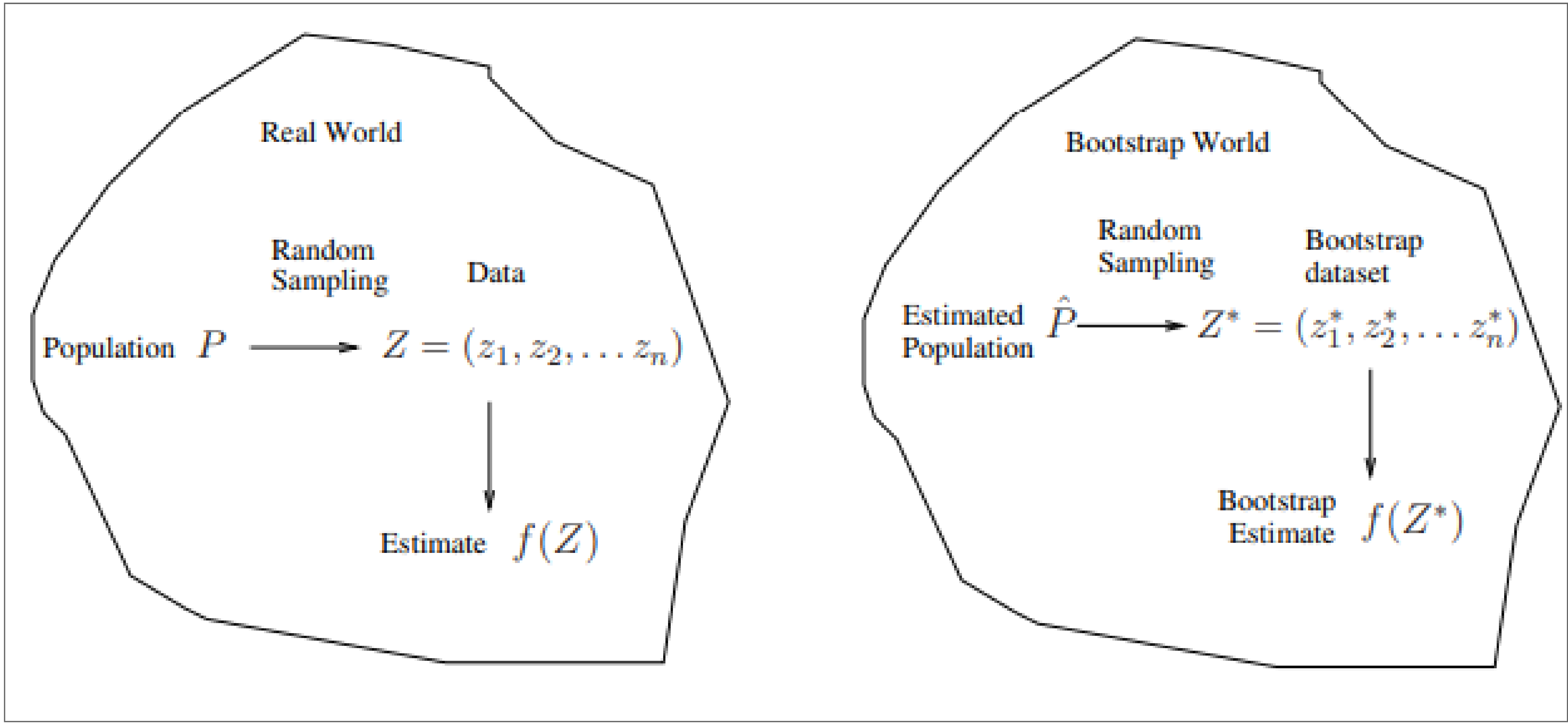
$$\textbf{if } B \to \infty \textbf{ then } \hat{\sigma}_B(\hat{\theta}) \to \hat{\sigma}_\infty(\hat{\theta}) = \sigma_B(\hat{\theta}) = \sigma_{\hat{\theta}}(F_n).$$

The bootstrap approximation, $\hat{\sigma}_B(\hat{\theta})$, to the bootstrap SE, $\sigma_B(\hat{\theta})$, provides an estimate of $\sigma_{\hat{\theta}}(F_n)$:

$$\hat{\sigma}_B(\hat{\theta})(\simeq \sigma_B(\hat{\theta}) = \sigma_{\hat{\theta}}(F_n)) \simeq \hat{\sigma}_{\hat{\theta}}(F_n).$$

SUMMARY

From real world to *bootstrap* world:


Real World
Random Sampling
Data
Population P
$Z = (z_1, z_2, \dots z_n)$
Estimate $f(Z)$
Bootstrap World
Random Sampling
Bootstrap dataset
Estimated Population $\hat{P}$
$Z^* = (z_1^*, z_2^*, \dots z_n^*)$
Bootstrap Estimate $f(Z^*)$

BACK TO BAGGING

- Breiman (**Breiman 1996**) combined the ideas of:
  - Averaging provides decreased variance estimates,
  - Bootstrap provides multiple (re)samples.
- He suggested: **b**ootstrap **agg**regat**ing** :
  - Take resamples from the original training dataset
  - Learn the model on each bootstrapped training set to get a prediction $\hat{f}^{*b}(x)$.
  - Use the boostrap estimates to obtain improved prediction/classification.

BAGGING PREDICTION/CLASSIFIER

- For regression (trees) the **bagged estimate** is the average prediction at x from these B trees.

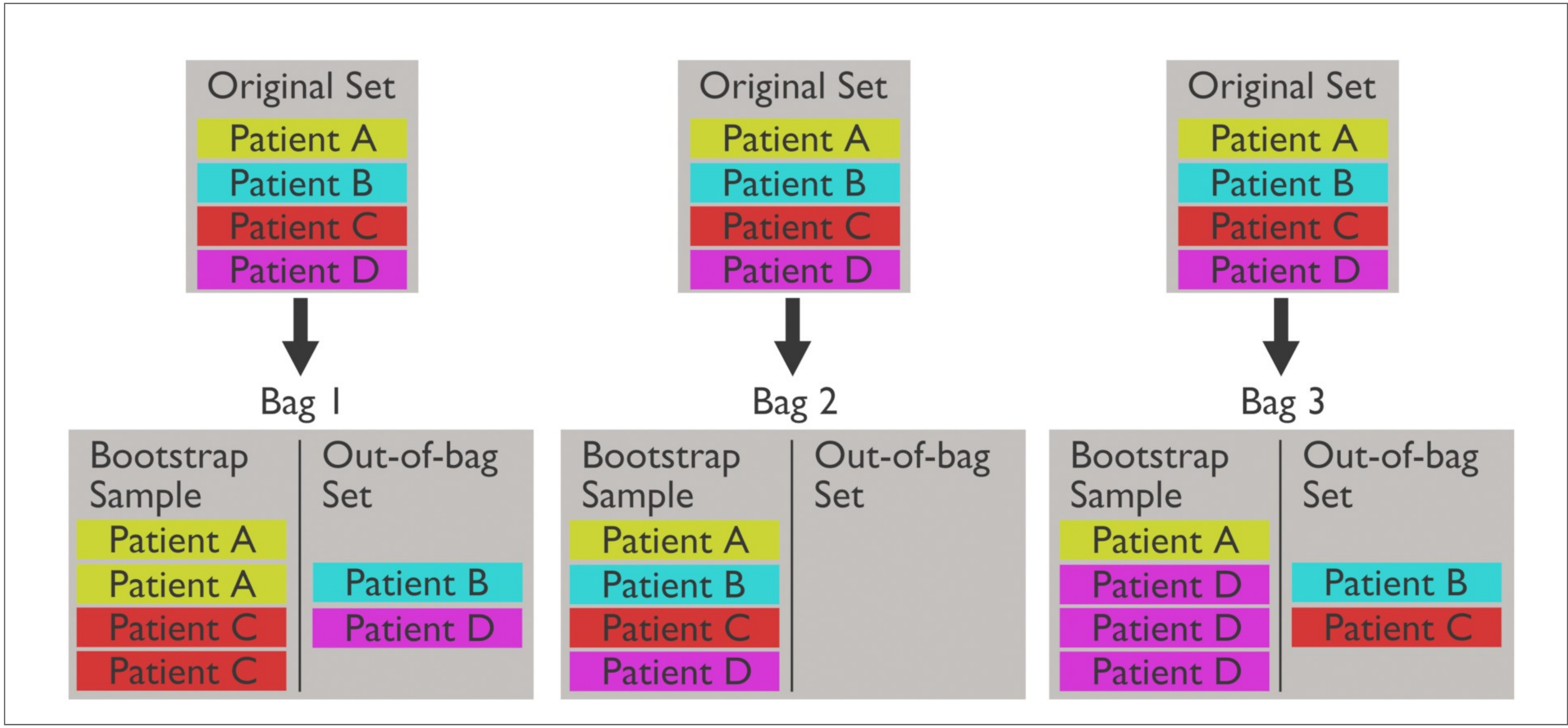
$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

- For classification (trees) the **bagged classifier** selects the class with the most “votes” from the B trees:

$$\hat{G}_{bag}(x) = \arg\max_{k} \hat{f}_{bag}(x).$$

OUT-OF-BAG OBSERVATIONS

- Every time a resample is taken *with replacement*, some observations are ommitted, due to the multiple occurring of others.



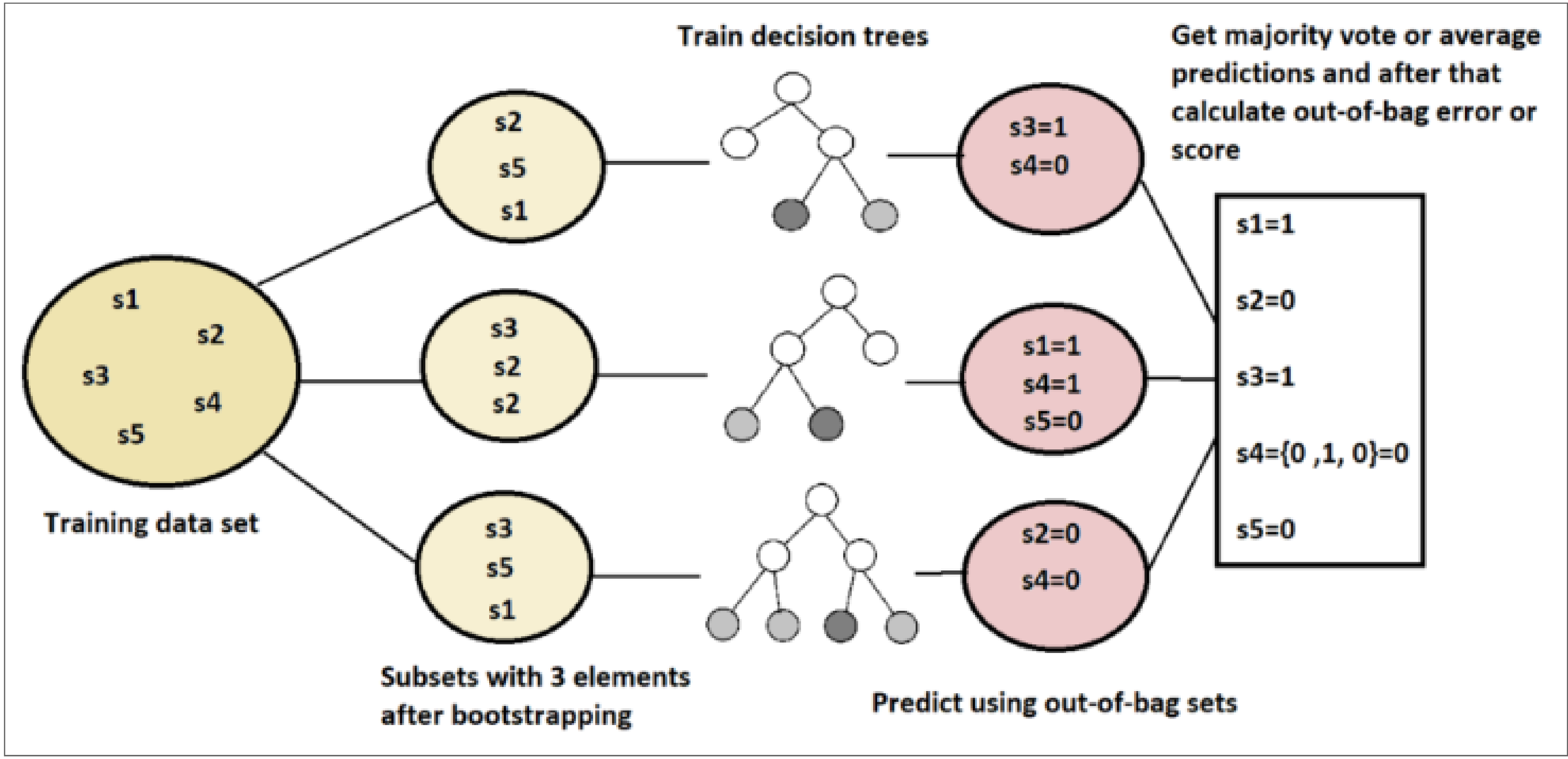
- These *out-of-bag* (OOB) observations can be used to build an estimate of prediction error.

OUT-OF-BAG ERROR ESTIMATES

Since each out-of-bag set is not used to train the model, it can be used to evaluate performance.

1. Find all trees that are not trained by the OOB instance.
2. Take the majority vote of these trees for the OOB instance, compared to the true value of the OOB instance.
3. Compile OOB error for all instances in the OOB dataset.

ILLUSTRATION OF OOB EE


Train decision trees
Get majority vote or average predictions and after that calculate out-of-bag error or score
s1
s2
s3
s4
s5
Training data set
s2
s5
s1
s3
s2
s2
s3
s5
s1
Subsets with 3 elements after bootstrapping
s3=1
s4=0
s1=1
s4=1
s5=0
s2=0
s4=0
Predict using out-of-bag sets
s1=1
s2=0
s3=1
s4={0 ,1, 0}=0
s5=0

Source: https://www.baeldung.com/cs/random-forests-out-of-bag-error

BAGGING IN R (1.1)

- This exampe relies on the well-known `AmesHousing` dataset on house prices in Ames, IA.
- We use libraries:
  - `rpart` for stratified resampling
  - `ipred` for bagging.

```r
# Prepare "clean" dataset from raw data
ames <- AmesHousing::make_ames()

# Split in test/training
set.seed(123)
split <- rsample::initial_split(ames, prop = 0.7,
                                strata = "Sale_Price")
ames_train  <- rsample::training(split)
ames_test   <- rsample::testing(split)
```

BAGGING IN R (1.2)

```r
system.time(
ames_bag1 <- ipred::bagging(
  formula = Sale_Price ~ .,
  data = ames_train,
  nbagg = 100,  coob = TRUE,
  control = rpart::rpart.control(minsplit = 2, cp = 0)
)
)
#    user  system elapsed
#   40.16    0.15   40.34
```

```r
show(ames_bag1)
# Bagging regression trees with 100 bootstrap replications
#
# Call: bagging.data.frame(formula = Sale_Price ~ ., data = ames_train,
#     nbagg = 100, coob = TRUE, control = rpart.control(minsplit = 2,
#         cp = 0))
```

```
#
# Out-of-bag estimate of root mean squared error:  26350.91
```

INTERPETABILITY: THE “ACHILES HEEL”

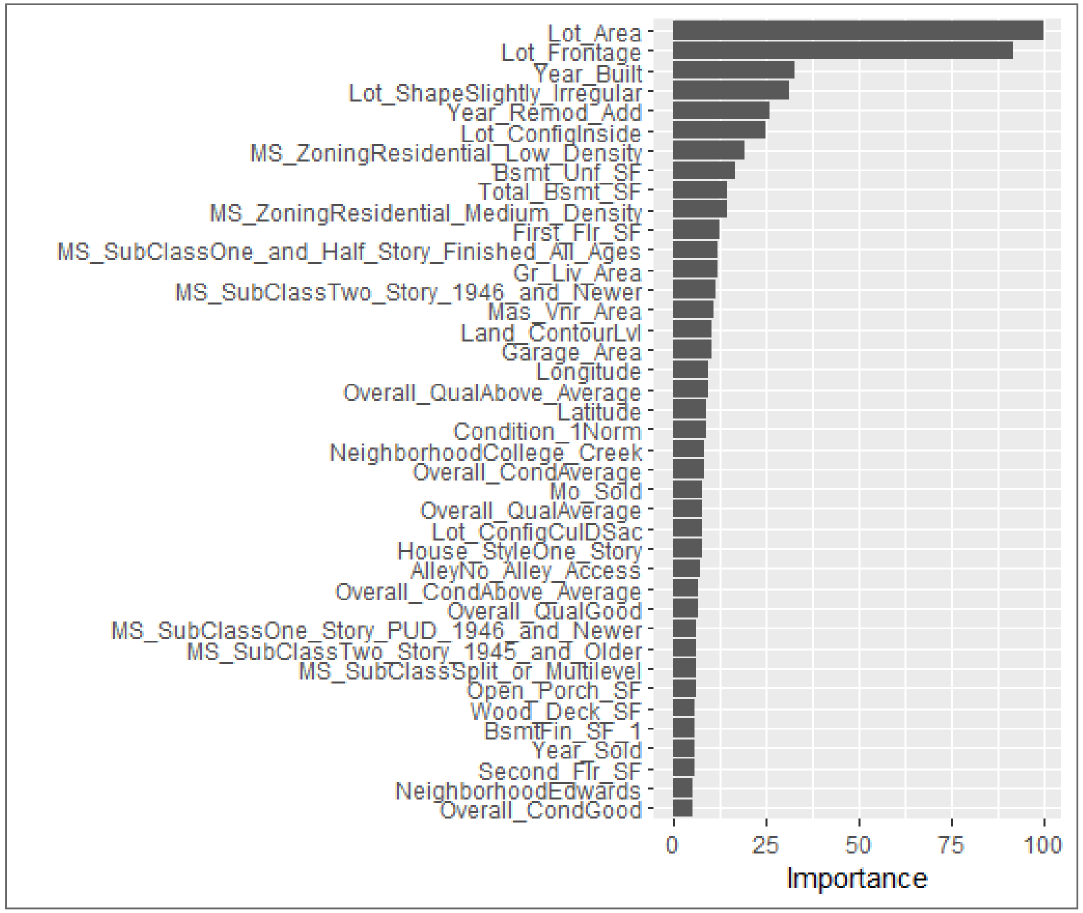
- Trees may have a straightforward interpretation,
  - Plotting the tree provides information about
    - which variables are important
    - how they act on the prediction
- Ensembles are less intuitive because
  - there is no consensus tree.
  - not clear which variables are most important

VARIABLE IMPORTANCE

- A complementary way to interpret a tree is by quantifying how *important* is each feature.
- Done measuring the total reduction in loss function associated with each variable across all splits.
- This measure can be extended to an ensemble simply by adding up variable importance over all trees built.

VARIABLE IMPORTANCE EXAMPLE

- If bagging is performed with `caret`
- the `vip` function from the `vip` package can be used (see lab examples).



RANDOM FORESTS

RANDOM FORESTS: DECORRELATING PREDICTORS

- Bagged trees, based on re-samples (of the same sample) tend to be highly correlated.
- To get away from this Breimann introduced Random forests, that use a “clever trick” that decorrelates trees:
  - When growing a tree from one bootstrap sample,
  - At each split use only a randomly selected *subset of predictors*.

RANDOM FORESTS

HOW MANY VARIABLES PER SPLIT?

- The usual recommendation for random selection of variables at each split has been studied by simulation:
  - For regression default value is $m = p/3$
  - For classification default value is $m = \sqrt{p}$.
- Alternatively the number m can be chosen using cross-validation.

RANDOM FOREST ALGORITHM

Random Forests Algorithm, from chapter 17 in (**Hastie and Efron 2016**)

OUT-OF-THE BOX PERFORMANCE

- Random forests have become popular because they tend to provide very good out-of-the-box performance, that is:
  - Although they have several hyperparameters that can be tuned,
  - the default values tend to produce good results.
- Moreover, among the more popular machine learning algorithms, random forests have the least variability in their prediction accuracy when tuning (**Probst, Wright, and Boulesteix 2019**).

OUT OF THE BOX PERFORMANCE

- Training a random forest model with all hyperparameters set to their default values, we get an OOB RMSE that is better than many other classifiers, with or without tuning.
- This combined with good stability and ease-of-use has made it the option of choice for many problems

OUT OF THE BOX PERFORMANCE EXAMPLE

```r
# number of features
n_features <- length(setdiff(names(ames_train), "Sale_Price"))

# train a default random forest model
ames_rf1 <- ranger(
  Sale_Price ~ .,
  data = ames_train,
  mtry = floor(n_features / 3),
  respect.unordered.factors = "order",
  seed = 123
)

# get OOB RMSE
(default_rmse <- sqrt(ames_rf1$prediction.error))
## [1] 24859.27
```

TUNING HYPERPARAMETERS

There are several parametres that, appropriately tuned, can improve RF performance.

1. The number of trees in the forest.
2. The number of features to consider at any given split (m_{try}).
3. The complexity of each tree.
4. The sampling scheme.
5. The splitting rule to use during tree construction.

1 and 2 tend to have the largest impact on predictive accuracy.

RANDOM FORESTS IN BIOINFORMATICS

- Random forests have been thoroughly used in Bioinformatics. See (**Boulesteix et al. 2012**).
- Bioinformatics data are often high dimensional with
  - dozens or (less often) hundreds of samples/individuals
  - thousands (or hundreds of thousands) of variables.

APPLICATION OF RANDOM FORESTS

- Random forests provide robust classifers for instance for
  - Distinguishing cancer from non cancer
  - Predicting tumor type in cancer of unknown origin
  - Selecting variables (SNPs) in Genome Wide Association Studies
- Some variation of Random forests are used only for variable selection

REFERENCES

Boulesteix, Anne Laure, Silke Janitza, Jochen Kruppa, and Inke R. König. 2012. “Overview of Random Forest Methodology and Practical Guidance with Emphasis on Computational Biology and Bioinformatics.” *Undefined* 2 (November): 493–507. **https://doi.org/10.1002/WIDM.1072**.

Breiman, Leo. 1996. “Bagging Predictors.” *Machine Learning* 24: 123–40. **https://doi.org/10.1007/BF00058655/METRICS**.

Efron, B. 1979. “Bootstrap Methods: Another Look at the Jackknife.” *The Annals of Statistics* 7 (1): 1–26. **https://doi.org/10.1214/aos/1176344552**.

Hastie, T., and B. Efron. 2016. *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. Cambridge University Press.

Probst, Philipp, Marvin N. Wright, and Anne-Laure Boulesteix. 2019. “Hyperparameters and Tuning Strategies for Random Forest.” *WIREs Data Mining and Knowledge Discovery* 9 (3): e1301. https://doi.org/**https://doi.org/10.1002/widm.1301**.