# Ensembles and Random Forests

Ryan Miller

Grinnell College
Statistics

# Introduction

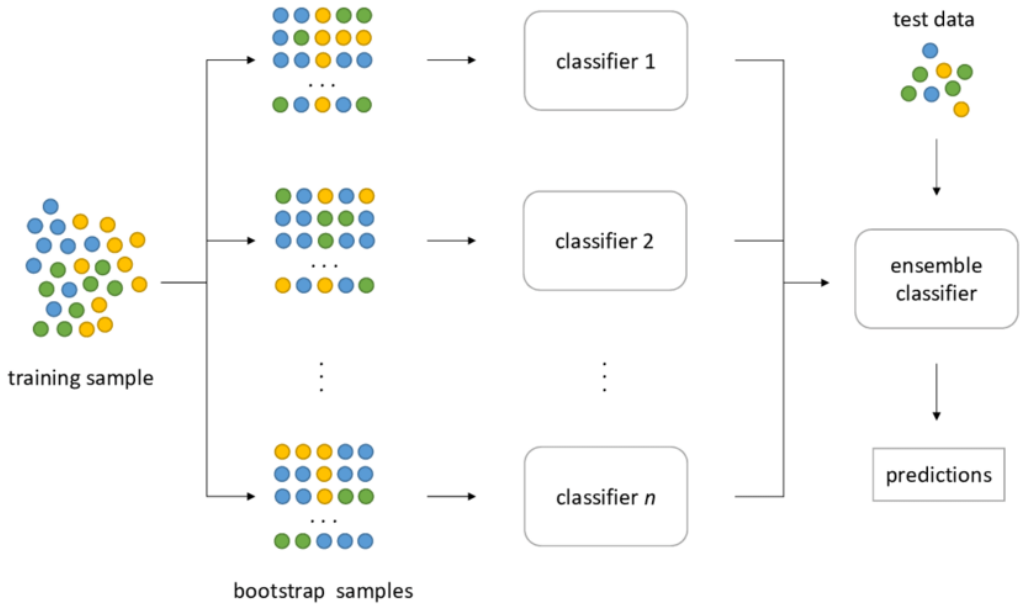
- Earlier in the semester, we learned about the decision tree algorithm, which recursively partitioned the data in search of "purity"
  - Despite their interpretability, decision trees are usually outperformed by other methods such as SVM or KNN
  - This is because complex relationships require deep trees (lots of splits), but deep trees exhibit high variance (they're prone to overfitting)
- The *random forest** algorithm combines predictions from a large number of decision trees to combat the overfitting that often happens with single decision trees

# Ensembles

- An **ensemble model** is a type of model that combines multiple individual models known as **base learners**
  - In the random forest algorithm, base learners are individual decision trees that are trained separately from one another
- The final prediction of an ensemble model is determined by aggregating the predictions from each of its base learners
  - This could be simple majority or weighted voting (classification tasks) or simple or weighted averaging (regression) tasks

# Bagging

Random forests rely upon *bagging*, or "bootstrap aggregation", a procedure where each base learner is trained on a bootstrapped sample of training data:




Why might bagging be necessary for an ensemble of decision trees?

Image credit: https://hudsonthames.org/bagging-in-financial-machine-learning-sequential-bootstrapping-python/

# Random Forest

- Bagging allows for some diversity among the base learners used in a random forest
  - However, because bootstrap samples are similar the base learners will still be very similar unless other steps are taken
- Thus, the random forest algorithm also relies upon **feature subsampling** to further enhance the diversity of the base learners
  - Subsampling can be done at the level of the tree, within specific depths, or at individual nodes (splits)
  - Subsampling randomly selects a set of features and the decision tree splits within that level use only those features
- Combining bagging and subsampling produces trees that are approximately independent, thereby allowing the random forest to benefit from the "wisdom of the crowd"

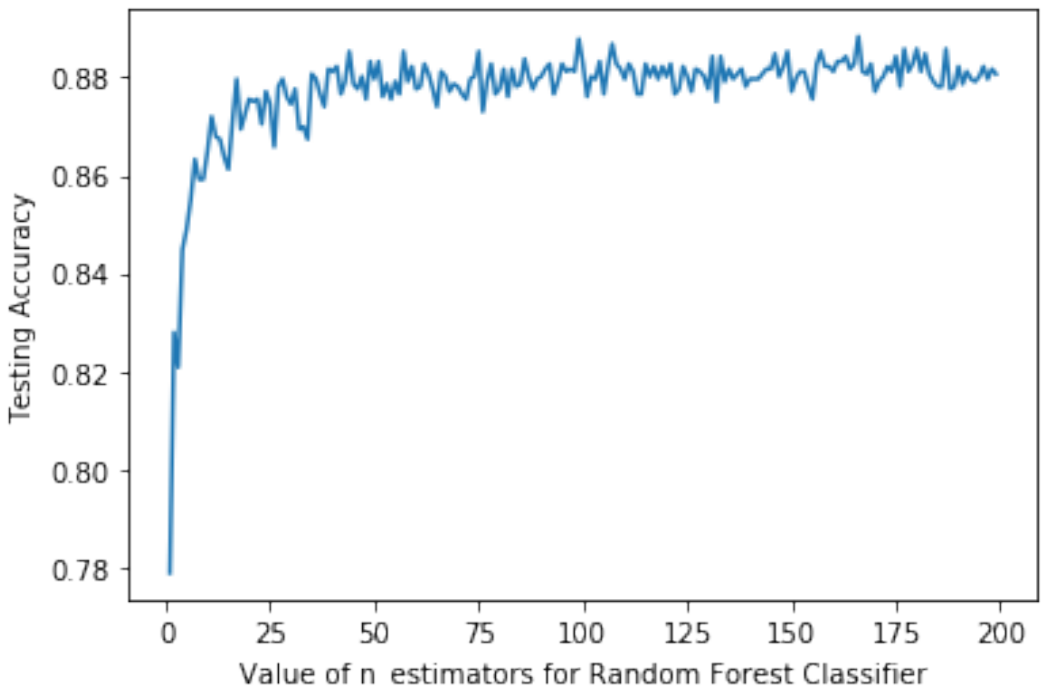
# Random Forest Hyperparameters

The performance of a random forest model depends upon hyperparameters that impact two different aspects of the algorithm

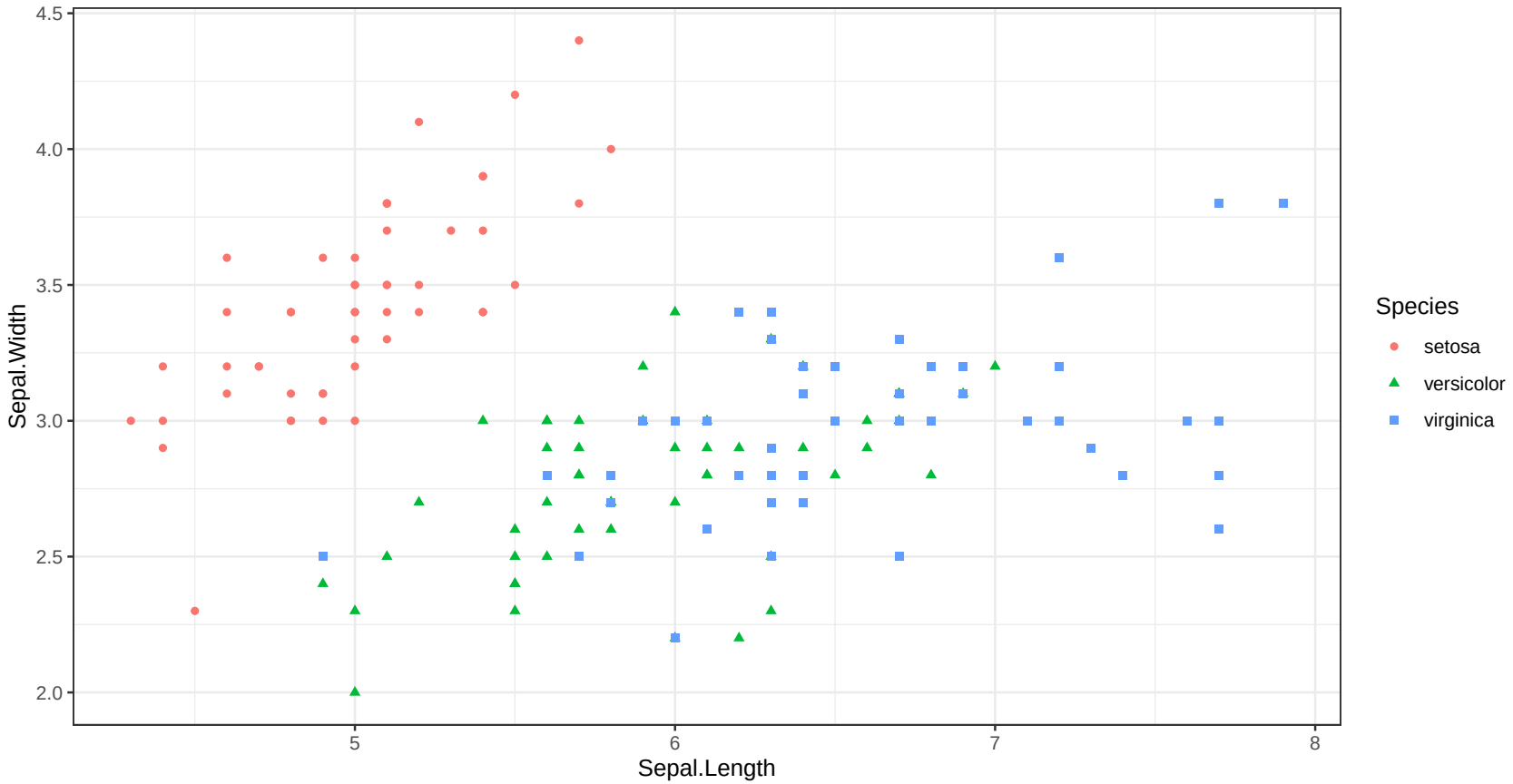
1. Behavior of the base learners:
   - `max_depth`
   - `min_samples_split`
   - `min_impurity_decrease`
2. How many/what kind of base learners:
   - `n_estimators`
   - `max_features`

# Random Forest Hyperparameters (cont.)

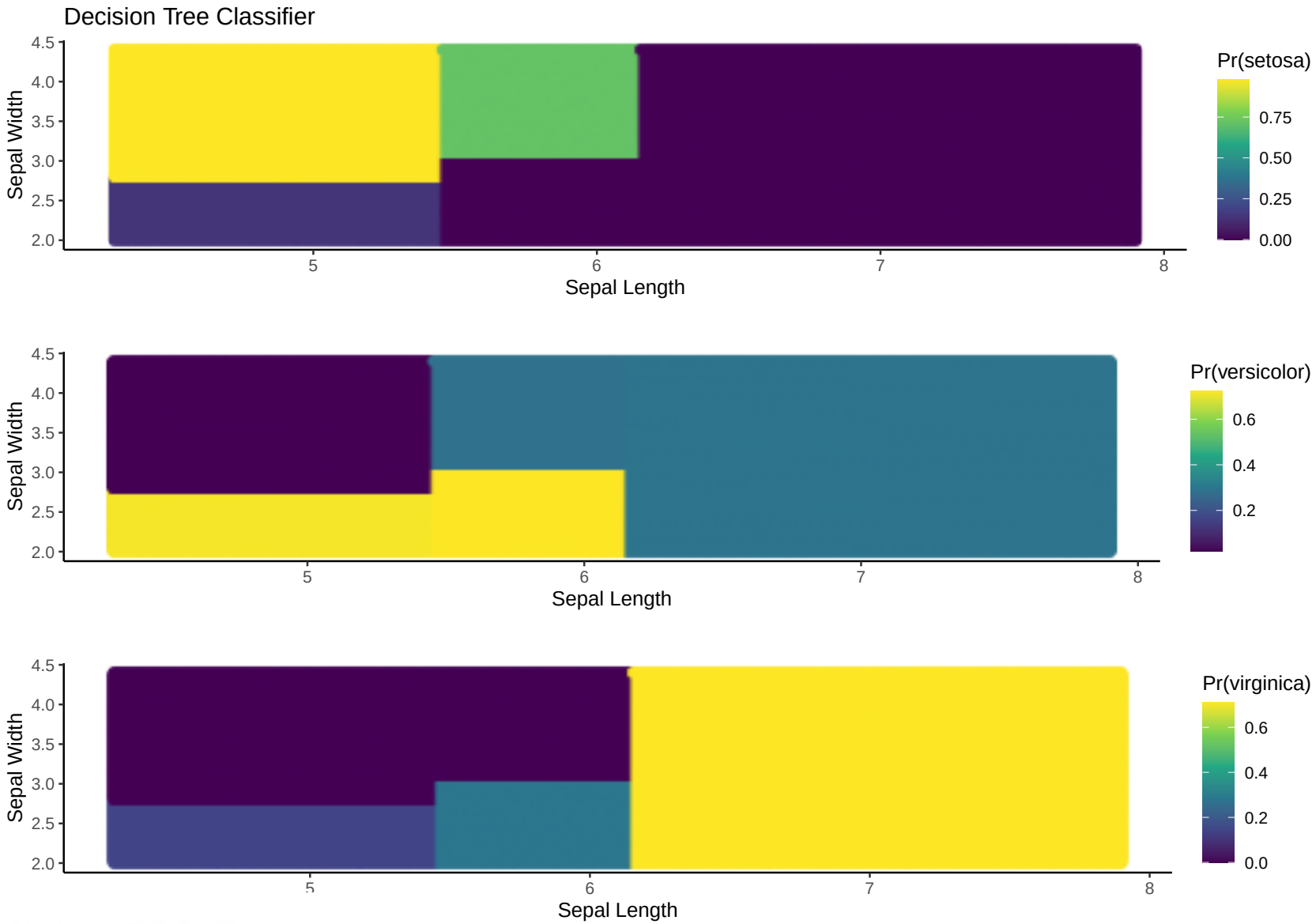
The `n_estimators` parameter is unique, as performance stabilizes once enough base learners are included in the forest, and more adding estimators *does not* lead to overfitting:



# Fisher's Iris Data

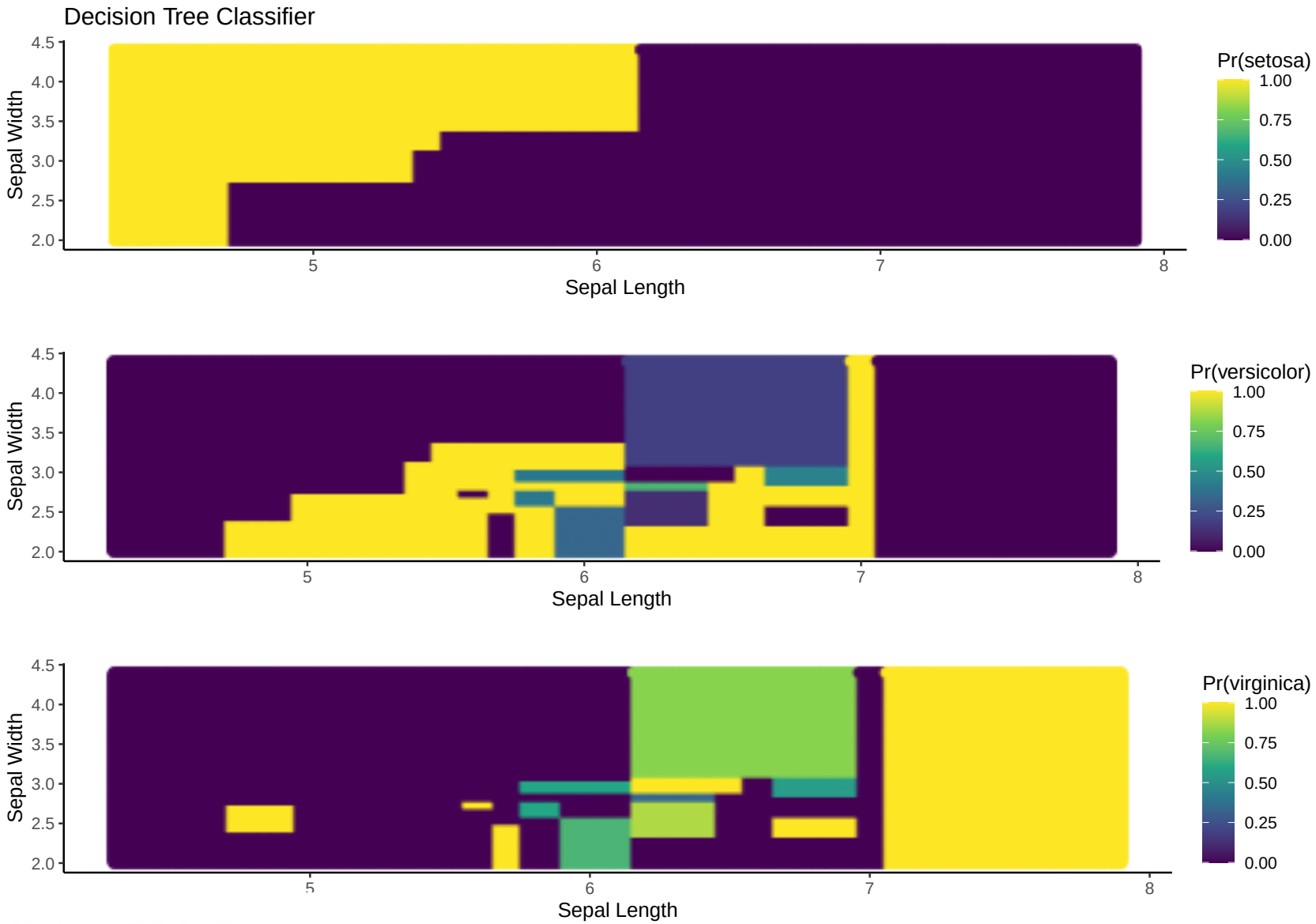


# Single Decision Tree (depth = 3)



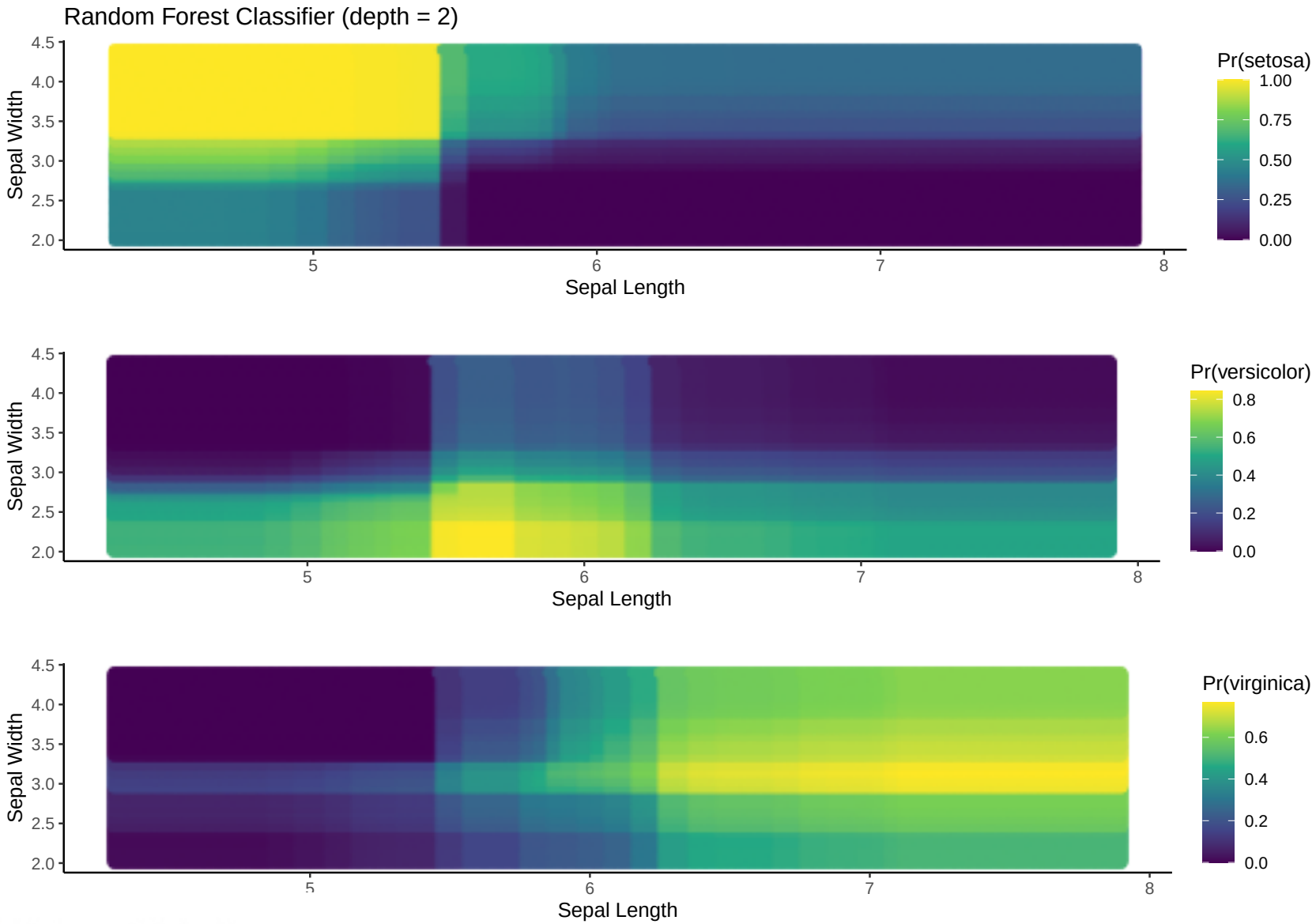

Grinnell College
Statistics

# Single Decision Tree (depth = 10)



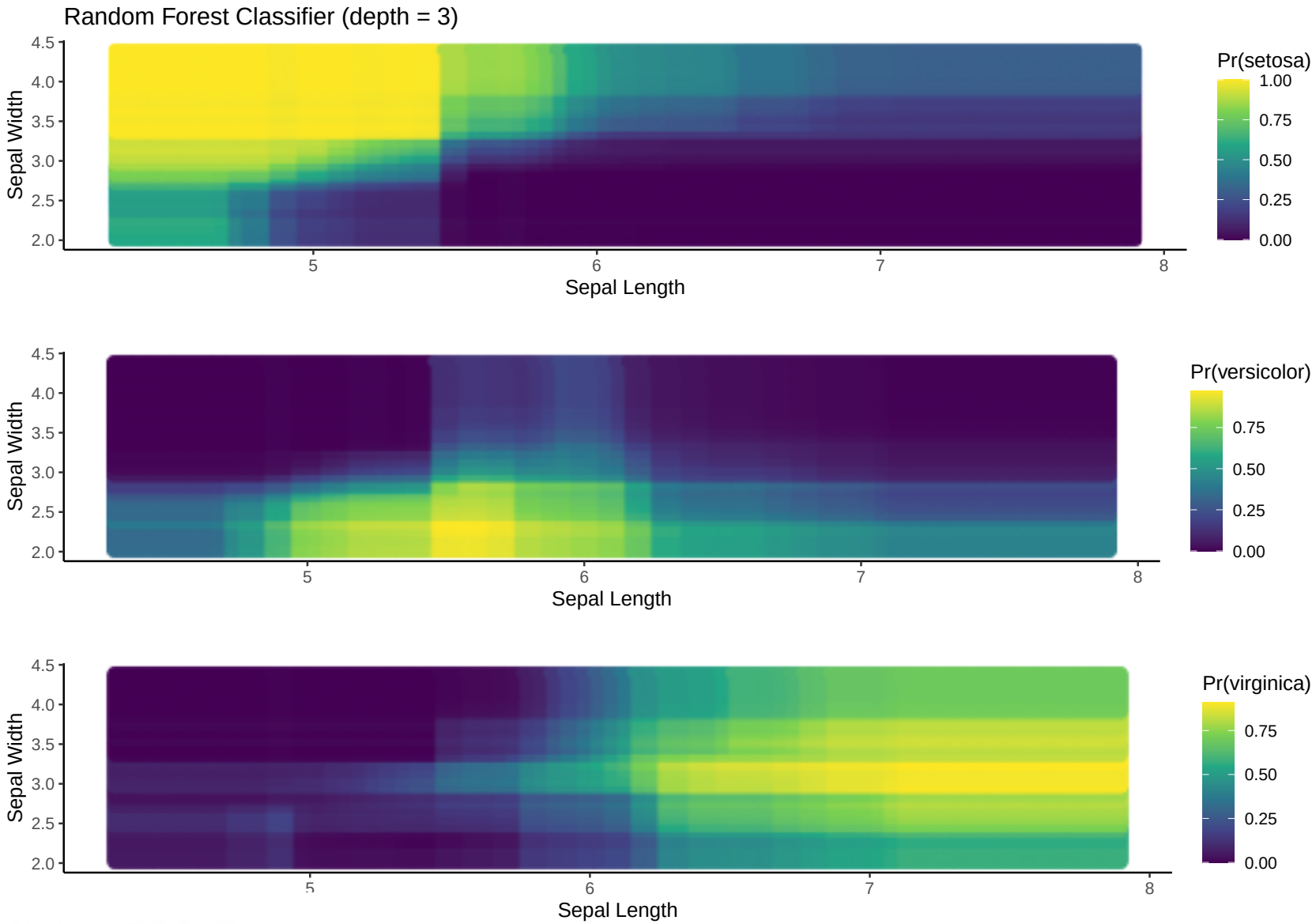

Grinnell College
Statistics

# Random Forest (depth = 2)




Grinnell College
Statistics

# Random Forest (depth = 3)



# What to Know for the Next Quiz

- Understand the core steps of the random forest algorithm:
  - Create bootstrapped samples from the training data
  - Train a decision tree using a subsample of features on each bootstrapped sample
  - Get final predictions by aggregating the predictions of the individual decision trees (base learners)
- Have a basic understanding of the role of each major hyperparameter (`max_depth`, `n_estimators`, and `max_features` in particular)