COMPUTATIONAL APPLICATIONS TO POLICY AND STRATEGY (**CAPS**)

Session 3 – Decision Trees, Random Forests, and Gradient Boosting

Leo Klenner, Henry Fung, Cory Combs

Outline

1. Model Evaluation
2. Case Study
3. Decision Tree models
4. Bootstrap Aggregation (Bagging)
5. Bagged Trees
6. Random Forests
7. Gradient Boosting

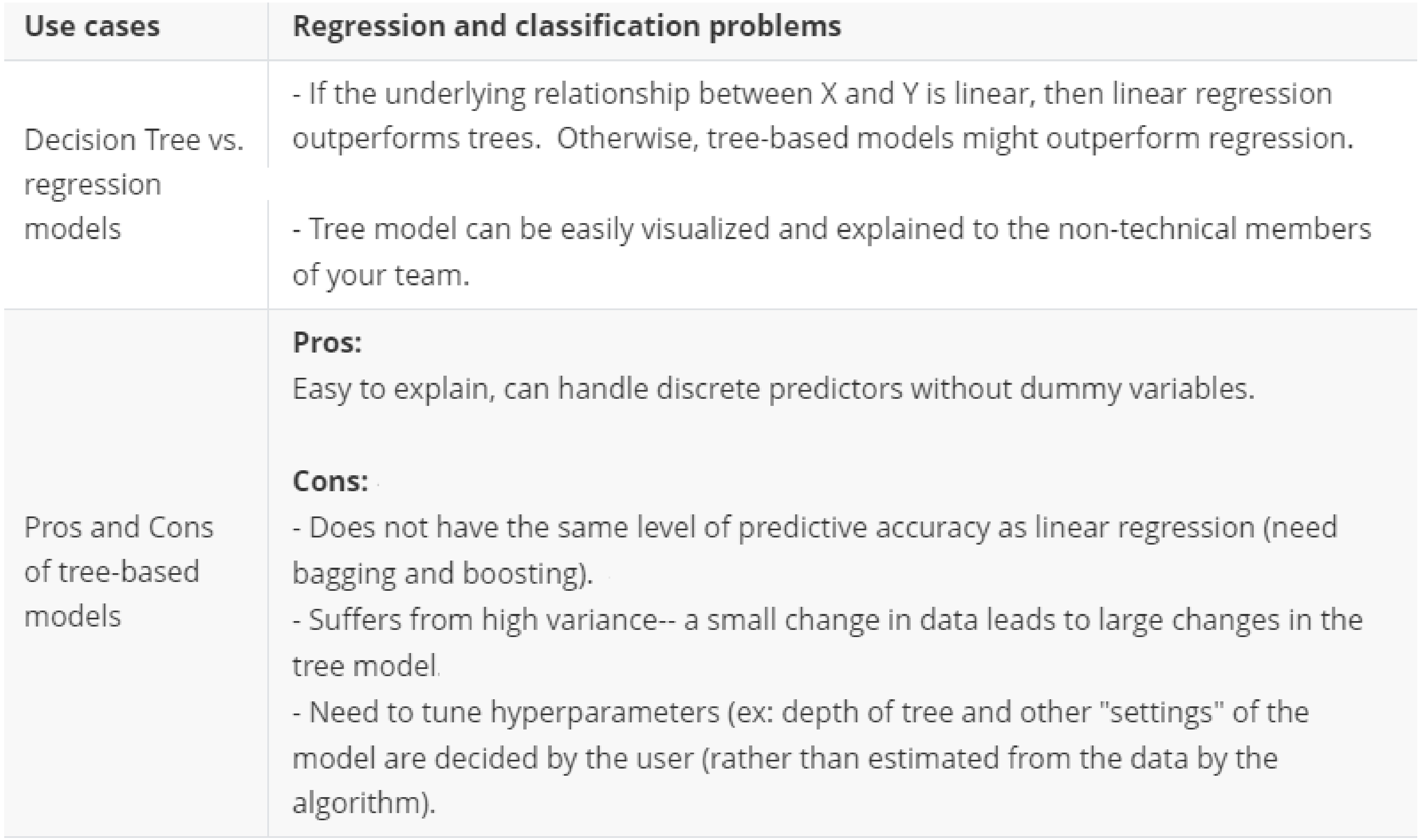
Big-picture Goal:

Look at the details of several tree-based models and discuss the pros and cons of each. Introduce several considerations for model selection.

2.1 How to use decision tree models to make predictions

- **Problem:** Predict student test scores from the student's family income and the his/her home-school distance.
- **Response (dependent variable):** Test score
- **Predictors (independent variables):** Family income, Distance between home and school
- **Data:** I have 10,000 student records that I use as the training dataset, and I have 1000 student records (with unknown test scores) that I use as the test dataset.
- Using the training dataset, we trained a decision tree model: essentially a series of rules that splits the predictor space into smaller regions.
- Suppose we have a test observation (a student called "Cory" that do not belong in the training dataset). Using the trained decision tree model, we want to predict Cory's test score, given has family income and home-school distance.
- What is the predicted test score of Cory? What is the residual? What is the test error of the 1000 test dataset (MSE)?

2.2 Fact sheet on the Decision Tree Model



Use cases	**Regression and classification problems**
Decision Tree vs. regression models	- If the underlying relationship between X and Y is linear, then linear regression outperforms trees. Otherwise, tree-based models might outperform regression. - Tree model can be easily visualized and explained to the non-technical members of your team.
Pros and Cons of tree-based models	**Pros:** Easy to explain, can handle discrete predictors without dummy variables. **Cons:** - Does not have the same level of predictive accuracy as linear regression (need bagging and boosting). - Suffers from high variance-- a small change in data leads to large changes in the tree model. - Need to tune hyperparameters (ex: depth of tree and other "settings" of the model are decided by the user (rather than estimated from the data by the algorithm).

2.3 Bias-Variance Tradeoff

- **Key:** We are not interested in building a model that make accurate predictions (have low training error) on the training data (we have the actual response values already for the training data!). Instead we want the model to make accurate predictions (low test error) on the test observations.
- Suppose we have a highly nonlinear model (a 5th order model) that fits the training data really well. This model will have low training error (low bias). However, it has high variance since a small change in the training dataset will result in a completely different model (a different curve). This model also tends to have a large test error since it cannot be generalized to test data.
- In contrast, a linear model has high bias, but low variance– if a different training dataset is used, the change in the linear model will be relatively small (the slope will only flatten or steepen).
- **Goal:** we want the best of both worlds: a model that has low bias and low variance so that its test error will be low (it will be useful in predicting the response of test observations).

2.4 Bagging I: Problem

- **Fact:** the variance of the average of a large set of independent random variables is smaller than each individual random variable.
- **Problem:** I want to predict test score from students' family income and IQ score.
- **Ideal case:**
  - I have 10,000 student records (test score, family income, IQ score) from 400 schools.
  - I build 400 decision tree models using the 400 independent sets of student record. I allow the trees to grow deeply to get a low bias. I end up with 400 low-bias but high variance decision trees.
  - Given the family income and IQ score of a student called "Leo". I make 400 predictions of his test score using 400 trained decision tree models.
  - To reduce variance of the prediction, I take the average of the 400 predicted test scores to get a final low bias and low variance predicted test score of Leo.
- **Question:** what happens if I don't have 400 datasets? Due to time/cost, I can only collect 10,000 student records from a single school.

2.5 Bagging II: Bootstrapping

- **Answer:** We use a technique called bootstrap.
- I have a single dataset of 10,000 student records. I can artificially generate 399 additional datasets using "sampling with replacement".
- We can think of my 10,000 student records as a bag of balls. I take a ball out of the bag and record its test score, family income, and IQ score. Then I put the ball back into the bag.
- I pick another ball out of the bag and record its information. I repeat this process until I have 399 "bootstrapped" samples. The sample will NOT contain unique student records (some balls will get picked multiple times). The idea is that the bootstrapped samples can approximate student records that I would have gotten had I collected data from 399 other schools.

2.6 Bagging III: Bagged Trees

- **Armed with 400 bootstrapped datasets,** we can build 400 “deep” decision tree models (we call these “bagged trees”).
- As before, using the bagged trees, we make 400 predictions of the test score of Leo. Finally, we get a final low-bias, low-variance predicted test score by taking the average of the 400 predicted test scores.
- **Problem with bagged trees:** if we have n independent random variables, we can get a small variance if we take their average. **What happens if the random variables are not independent?** Let’s look at the case of n = 2.
- **Question:** are the bagged trees independent?
- **Answer:** no, since the bagged trees are generated from bootstrapped data from the original dataset, I expect these datasets to be similar and therefore, the trained decision tree models should also be similar.
- **Example:** if IQ score is an important predictor of test score, then it will be selected frequently by the algorithm when we do splitting (since it reduces the sum of square error by a lot). In fact, since IQ is so important, it will probably be the first predictor (first node) selected in ALL 400 bagged trees. Thus, the 400 bagged trees are highly correlated.

2.7 Random Forests

- We can reduce the variance of our model if we can somehow **decorrelate** the bagged trees. This is exactly what **Random Forests do.**
- To build a Random Forest model, we follow *almost* the exact process as the bagged trees, with the following exception:
  - We ONLY consider *m* out of *p* predictors whenever we make a splitting decision.
- For example, let's say my predictors are: Family income, parent's education, home-school distance, student-teacher ratio, IQ score.
- Whenever I make a splitting decision, I only randomly consider 3 out of 5 predictors (ex: family income, parent's education, and student-teacher ratio). Therefore, even though IQ score is an important predictor, it will not be considered at every split decisions. For this reason, each bagged tree will look slightly different and be less correlated.
- Since the bagged trees are less correlated, my final predicted test score will have even less variance and less test error.

2.8 Test error of Bagged Trees vs Random Forest

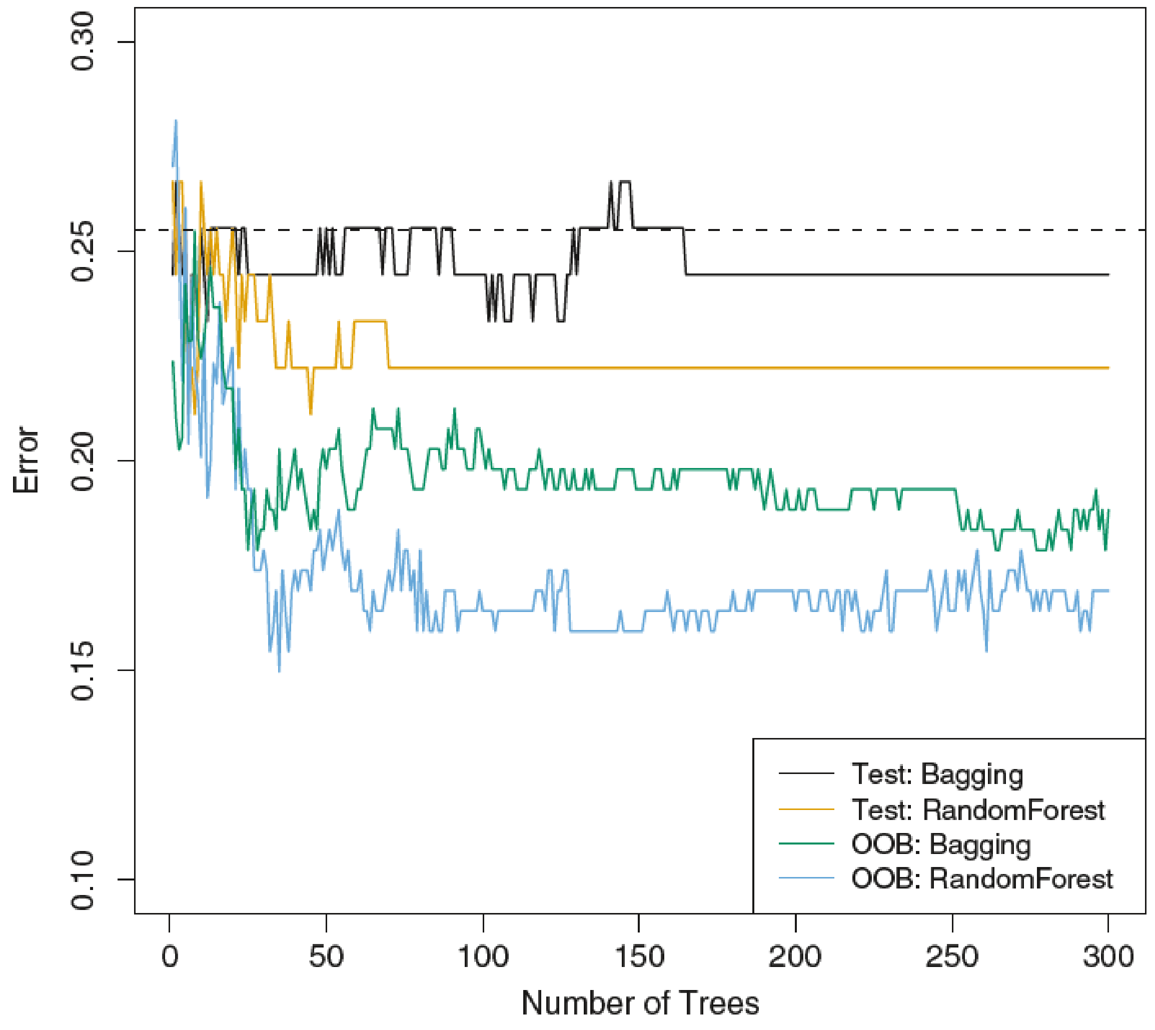
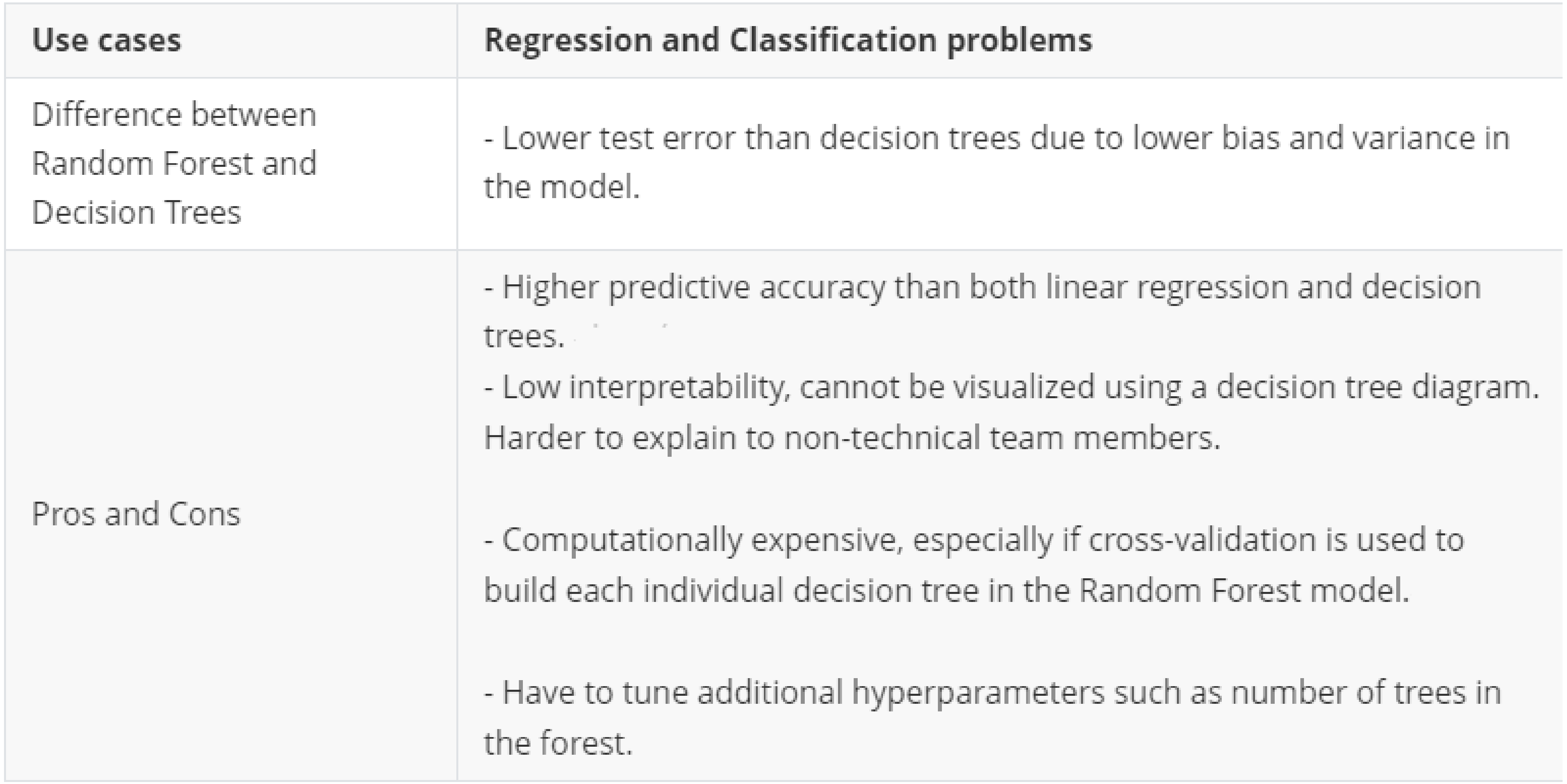


Figure 1: Comparison of the test error between bagging and Random Forest [1]

2.9 Fact sheet of Random Forests

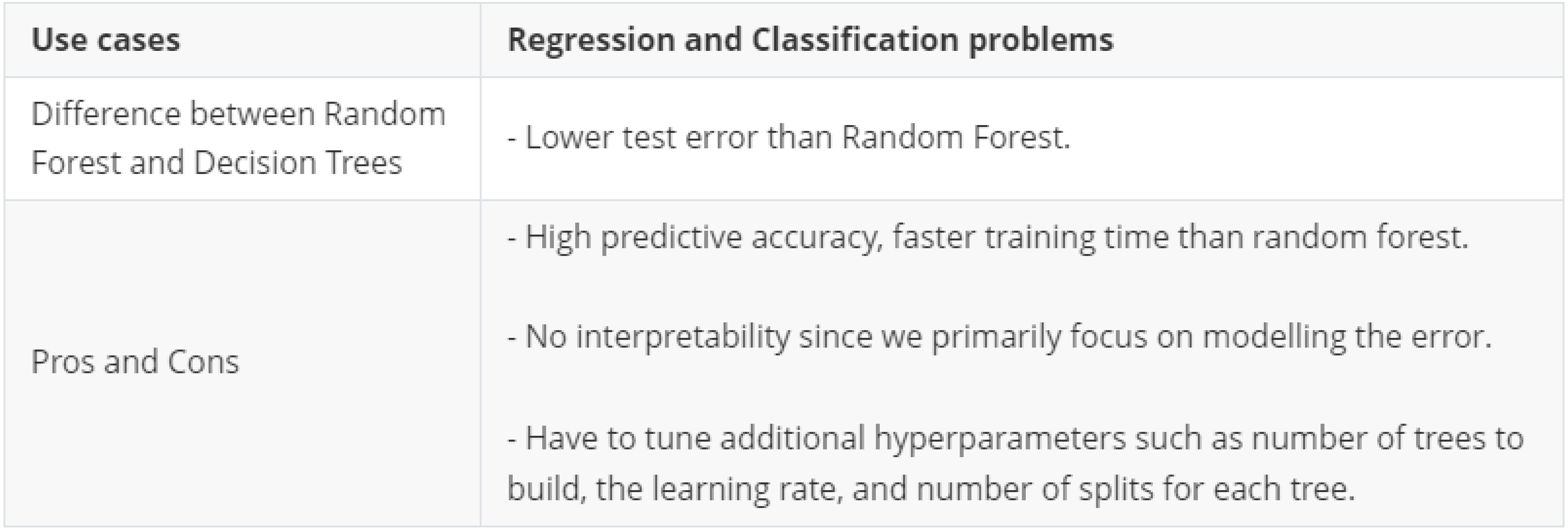


Use cases	Regression and Classification problems
Difference between Random Forest and Decision Trees	- Lower test error than decision trees due to lower bias and variance in the model.
Pros and Cons	- Higher predictive accuracy than both linear regression and decision trees. - Low interpretability, cannot be visualized using a decision tree diagram. Harder to explain to non-technical team members. - Computationally expensive, especially if cross-validation is used to build each individual decision tree in the Random Forest model. - Have to tune additional hyperparameters such as number of trees in the forest.

2.10 Gradient Boosting

- Like bagging, boosting is an approach that can be applied to many statistical learning methods to improve their performance. The main idea is that boosting allow the models to "learn from its past errors".
- Like bagging, a series of models are built. However, in boosting, each model takes into account the error from the previous model. Thus, error is reduced sequentially.
- Boosting is an example of "Ensemble Modelling" in which a set of "weak learners" are combined to form a "strong learner".
- This is similar to a rope: it is compose of many "weak threads" that becomes "strong" when combined together.

2.11 Fact sheet of Gradient Boosting



Use cases	Regression and Classification problems
Difference between Random Forest and Decision Trees	- Lower test error than Random Forest.
Pros and Cons	- High predictive accuracy, faster training time than random forest. - No interpretability since we primarily focus on modelling the error. - Have to tune additional hyperparameters such as number of trees to build, the learning rate, and number of splits for each tree.

2.12 Details of Gradient Boosting I

- Given a dataset (test score, family income, home-school distance) of 5 students, I fit a decision tree model and get the following results:

Student ID	Actual Test Score	Model # 1 Predicted Test Score	Model # 1 Residual
1	87	85	2
2	75	85	-10
3	50	40	10
4	70	65	5
5	80	85	-5

- Next, we build another decision model (Model # 2), using family income as the predictor and **the residual of Model # 1 (instead of test score) as the response variable.**

2.13 Details of Gradient Boosting II

- **Why?** Suppose Model # 2 exactly predicts the residuals. In this case, the predicted response values of Model #2 equals to the residuals of Model #1.
- If that's the case, then I can aggregate Model #1 and Model #2 by summing their respective response values to get a final predicted test score.
- Since I model the residuals of Model # 1 exactly, my final predicted test score will be equal to the actual test score (and I will have zero residual).

Student ID	Actual Test Score	Model # 1 Predicted Test Score	Model # 1 Residual	Model # 2 Predicted Response Value	Model 2 Residual
1	87	85	2	2	0
2	75	85	-10	-10	0
3	50	40	10	10	0
4	70	65	5	5	0
5	80	85	-5	-5	0

2.14 Details of Gradient Boosting III

- In reality, this is rarely the case, I will have differences between the residuals of Model #1 and the predicted response values of Model #2. In this case, I need to do two things:
  - Update the total predicted response value.
  - Update the residuals.
- I will build another decision tree model (Model # 3) to model the residuals of Model #2.
- I will continue this process k times (k is specified by the user).
- In the end, my final predicted test score would be the sum of the predicted response variables of models 1… k, and the residuals will be reduced incrementally.

ID	Actual Test Score	Model # 1 Predicted Test Score	Model # 1 Residual	Model # 2 Predicted Response Value	Model # 2 Residual	Total Predicted Value
1	87	85	2	4	-2	89
2	75	85	-10	-5	-5	80
3	50	40	10	8	2	48
4	70	65	5	4	1	69
5	80	85	-5	-7	2	78

set as response variable for Model # 3

References

- [1] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshiranai, *An introduction to Statistical Learning*, Springer Science + Media, 2013.