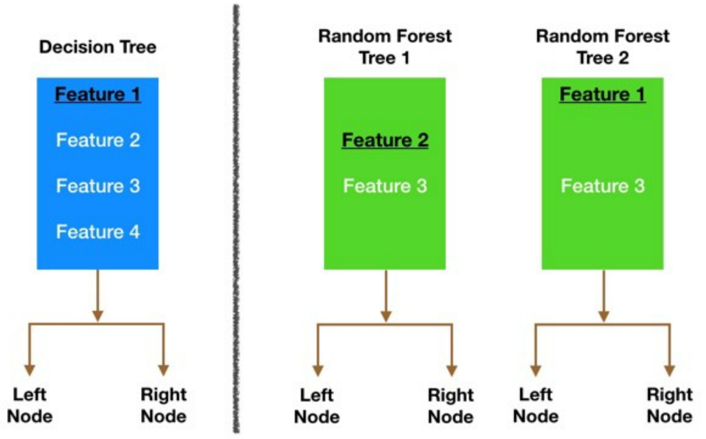
Random Forest Appendix

Group 1: Beth Vander Hoek, Dennis Kelly, Kylee LaPierre, Stephanie Leiva

This document contains the resources our group used for our research product.There are links to blogs, scikit documentation, images, flowcharts, and YouTube videos. We've highlighted some information that we found useful. We encourage you to dig through these resources and further continue your exploration!

1. https://towardsdatascience.com/understanding-random-forest-58381e0602d2
   - Here are some essential quotes to explain Random Forest pulled from the article:
   - "Similarly, with a random forest model, our chances of making correct predictions increase with the number of uncorrelated trees in our model."
   - 'Random forest ensures that the behavior of each individual tree is not too correlated with the behavior of any of the other trees in the model. [How]? It uses the following two methods:
   - 'Bagging (Bootstrap Aggregation) — Decisions trees are very sensitive to the data they are trained on — small changes to the training set can result in significantly different tree structures. Random forest takes advantage of this by allowing each individual tree to randomly sample from the dataset with replacement, resulting in different trees. This process is known as bagging.'



Node splitting in a random forest model is based on a random subset of features for each tree.

   -
   - 'Feature Randomness — In a normal decision tree, when it is time to split a node, we consider every possible feature and pick the one that produces the most separation between

the observations in the left node vs. those in the right node. In contrast, each tree in a random forest can pick only from a random subset of features.'

  - 'The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.'

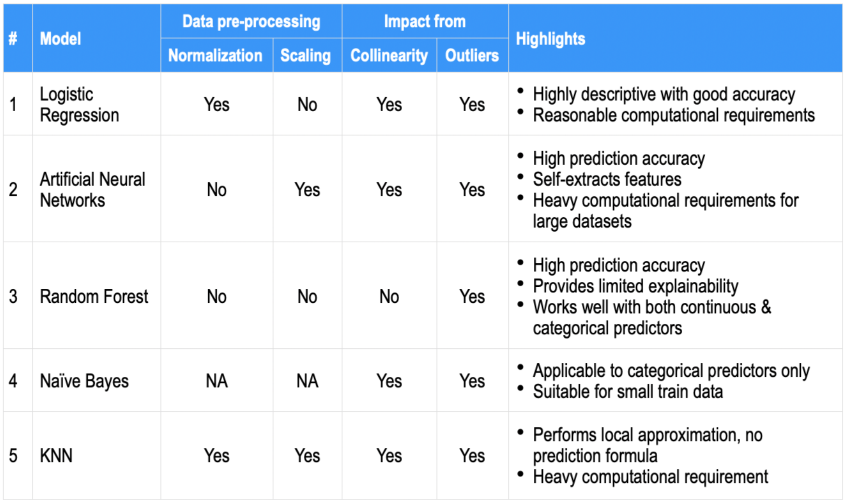
2. https://www.youtube.com/watch?v=ok2s1vV9XW0&ab_channel=codebasics
   - (helpful video to explain random forest + example code + how to change parameters to fine tune it by increasing # random tree models)
   - Random forest is majority rule from random tree models
3. https://www.youtube.com/watch?v=PHxYNGo8NcI&ab_channel=codebasics
   - (video that explains tree decisions + code, the building block of random forest)
4. https://www.youtube.com/watch?v=J4Wdy0Wc_xQ
   - (video series using random forest from start to finish)
5. 1.11. Ensemble methods — scikit-learn 1.1.2 documentation (Scikit Learn documentation)
   - 1.11.2. Random Forest
   - In random forests (see **RandomForestClassifier** and **RandomForestRegressor** classes), each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.
   - Furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size `max_features`. (See the parameter tuning guidelines for more details).
   - The purpose of these two sources of randomness is to decrease the variance of the forest estimator. Indeed, individual decision trees typically exhibit high variance and tend to overfit. The injected randomness in forests yield decision trees with somewhat decoupled prediction errors. By taking an average of those predictions, some errors can cancel out. Random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. In practice the variance reduction is often significant hence yielding an overall better model.

   - In contrast to the original publication [B2001], the scikit-learn implementation combines classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class.
   - 1.11.2.3. Parameters
   - The main parameters to adjust when using these methods is n_estimators and max_features. The former is the number of trees in the forest. The larger the better, but also the longer it will take to compute. In addition, note that results will stop getting significantly better beyond a critical number of trees. The latter is the size of the random subsets of features to consider when splitting a node. The lower the greater the reduction of variance, but also the greater the increase in bias. Empirical good default values are max_features=1.0 or equivalently max_features=None (always considering all features instead of a random subset) for regression problems, and max_features="sqrt" (using a random subset of size sqrt(n_features)) for classification tasks (where n_features is the number of features in the data). The default value of max_features=1.0 is equivalent to bagged trees and more randomness can be achieved by setting smaller values (e.g. 0.3 is a typical default in the literature). Good results are often achieved when setting max_depth=None in combination with min_samples_split=2 (i.e., when fully developing the trees). Bear in mind though that these values are usually not optimal, and might result in models that consume a lot of RAM. The best parameter values should always be cross-validated. In addition, note that in random forests, bootstrap samples are used by default (bootstrap=True) while the default strategy for extra-trees is to use the whole dataset (bootstrap=False). When using bootstrap sampling the generalization error can be estimated on the left out or out-of-bag samples. This can be enabled by setting oob_score=True.

6. Hyperparameters of Random Forest Classifier - GeeksforGeeks (Hyperparameters that can be tuned to increase accuracy of the training model)
   - The scikit-learn documentation states that n_estimators and max_features are the main parameters to adjust, but they are *not* the only hyperparameters. This article lists Random Forests' hyperparameters and how they can increase the accuracy of the model.
7. Random Forest Classifier Tutorial: How to Use Tree-Based Algorithms for Machine Learning (freecodecamp.org)
   - "Step 1: The algorithm selects random samples from the dataset provided."

   - "Step 2: The algorithm will create a decision tree for each sample selected. Then it will get a prediction result from each decision tree created."
   - "Step 3: Voting will then be performed for every predicted result. For a classification problem, it will use mode, and for a regression problem, it will use mean."
   - "Step 4: And finally, the algorithm will select the most voted prediction result as the final prediction."

8. ADAfaEPoV.pdf (cmu.edu) (Chapter 13 of this book is on Decision Trees, thorough but mathy)
9. Random Forest Algorithm with Python and Scikit-Learn (stackabuse.com) (Advantages and disadvantages of using Random Forest)
10. Classification Models in Machine Learning | Classification Models (analyticsvidhya.com) (comparison of different classification models)



#	Model	Data pre-processing		Impact from		Highlights
		Normalization	Scaling	Collinearity	Outliers	
1	Logistic Regression	Yes	No	Yes	Yes	• Highly descriptive with good accuracy • Reasonable computational requirements
2	Artificial Neural Networks	No	Yes	Yes	Yes	• High prediction accuracy • Self-extracts features • Heavy computational requirements for large datasets
3	Random Forest	No	No	No	Yes	• High prediction accuracy • Provides limited explainability • Works well with both continuous & categorical predictors
4	Naïve Bayes	NA	NA	Yes	Yes	• Applicable to categorical predictors only • Suitable for small train data
5	KNN	Yes	Yes	Yes	Yes	• Performs local approximation, no prediction formula • Heavy computational requirement

11. Random Forest Regression in Python - GeeksforGeeks (Tutorial with sample code using Random Forest)
12. Example code can be found in GitHub repository as sample-random-forest-code.ipynb (uses diabetes.csv dataset from M09 Logistic Regression exercise).