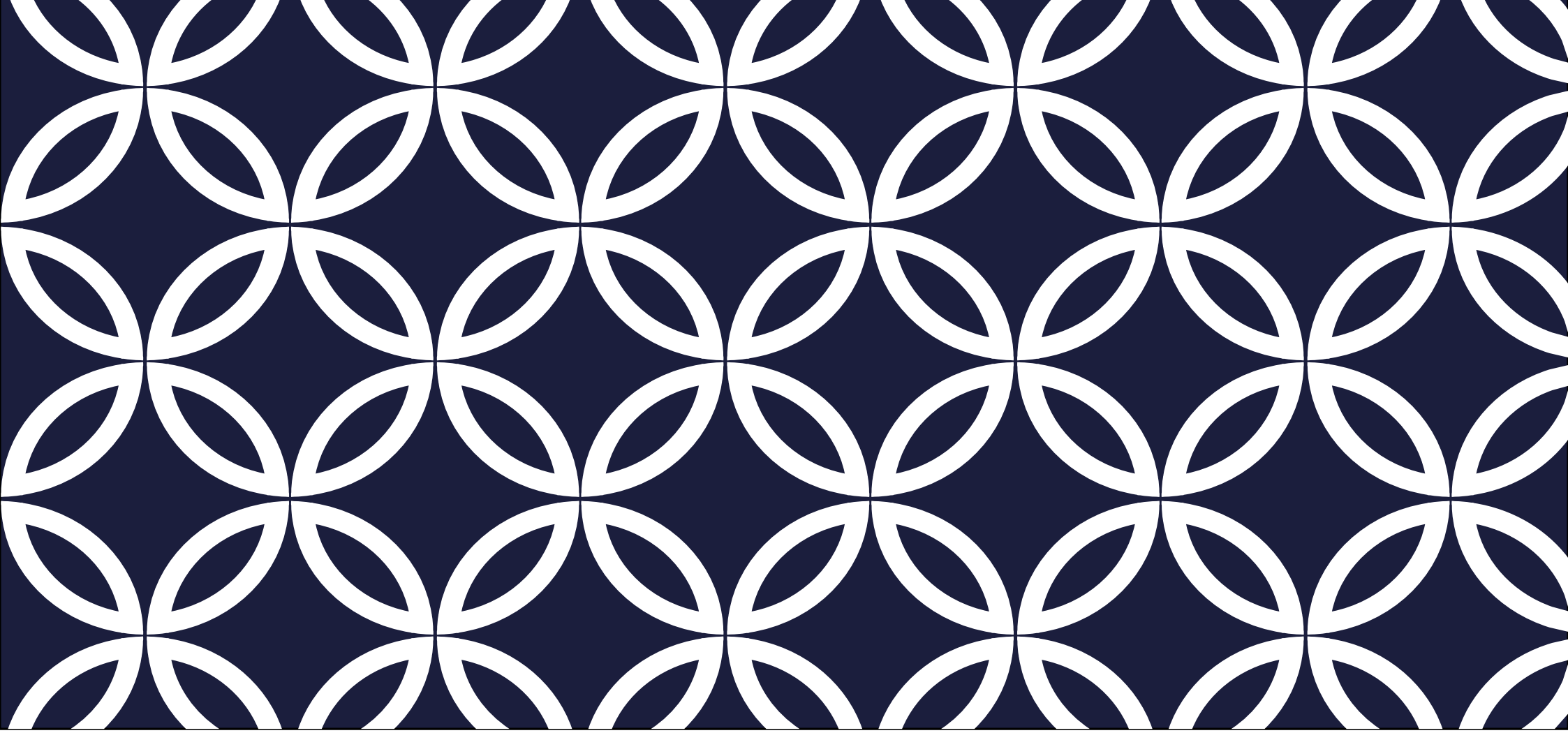


DECISION TREE & RANDOM FOREST

Anshul Verma
Hiba Doudar
Nada El Moghany
Obaid Muhammad
Zoha Sherkat-Masoumi

Outline

Introduction

Decision Trees
Random Forest

Algorithm

Intuition
Pseudo Code

Example #1

Dummy Dataset

Example #2

Temperature
Dataset

Example #3

Diabetes Dataset

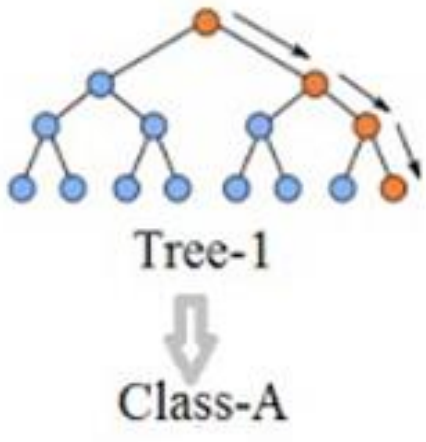
Conclusion

Bagging
K-Fold

Introduction

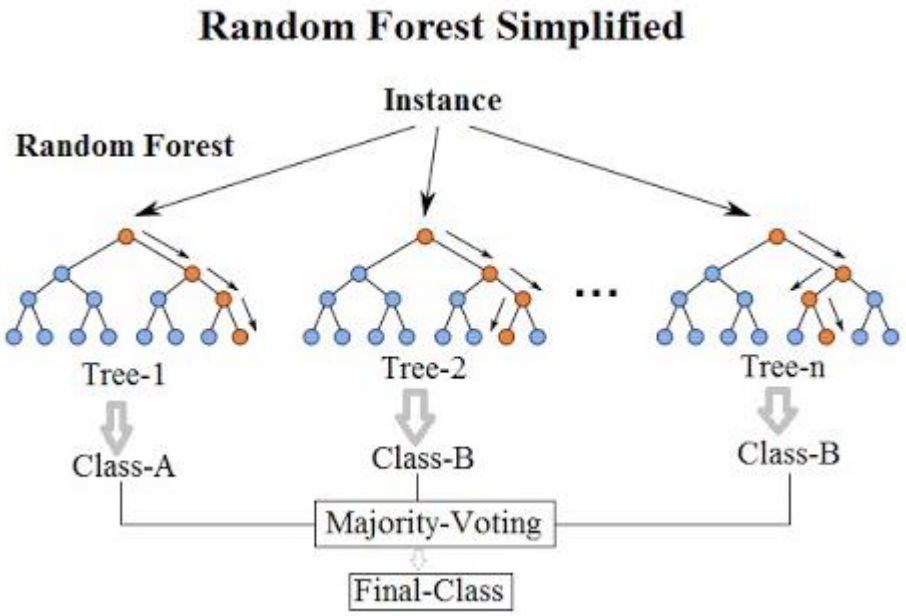
Decision Trees

- Require continuous split of data according to a certain parameter.
  - **Classification trees** (Categorical)
  - **Regression trees** (Continuous)
- Used to visually represent decision making.



Random Forests

- Type of ensemble learning method.
  - **Ensemble learning methods** combine several machine learning techniques.
- Individual decision trees where each tree in the random forest makes a prediction
- Relatively uncorrelated models (trees) operating together are likely to outperform a constituent tree.
  - Low correlation is key!



Algorithm

Idea

- Sample m independent training sets from Dataset (S)
- Compute the prediction **yi**, and take the average $y = \frac{1}{m}\sum_{i=1}^{m} y_i$

$$\mathbb{E}[y] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \mathbb{E}[y_i] \qquad \text{Var}[y] = \text{Var}\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \frac{1}{m^2}\sum_{i=1}^{m}\text{Var}[y_i] = \frac{1}{m}\text{Var}[y_i]$$

- Problem: Sets aren't independent, so 1/m reduction in variance is unlikely.

Algorithm

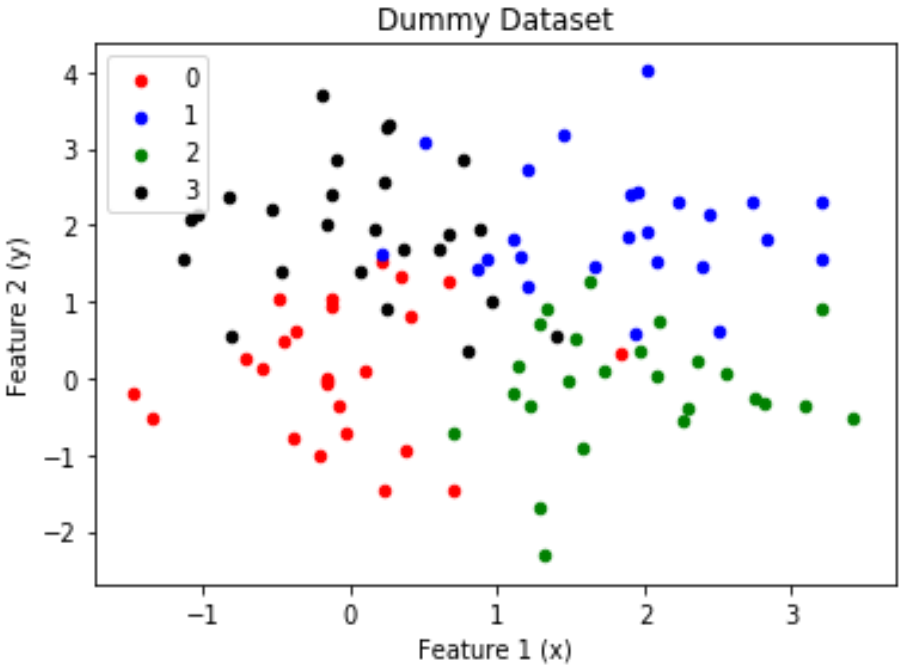
- **Inputs:**
  - Dataset (S), Features (F)
  - Number of Trees (B)
- **Algorithm:**
  - Sub sample from the Dataset
  - Build Decision Tree
    - choose random set (f) at each node of the decision tree
  - Repeat until all trees are created

```
1  function RandomForest(S, F)
2      H ← ∅
3      for i ∈ 1, ..., B do
4          S^(i) ← A bootstrap sample from S
5          h_i ← RandomizedTreeLearn(S^(i), F)
6          H ← H ∪ {h_i}
7      end for
8      return H
9  end function
10 function RandomizedTreeLearn(S, F)
11     At each node:
12         f ← very small subset of F
13         Split on best feature in f
14     return The learned tree
15 end function
```

Example #1: Decision Tree Model

DataSet

- Dummy dataset using make_blobs function in **python.**
- The dataset has two features ‘x’ and ‘y’ so that it’s visualizable and there are four different classes in the dataset.



Decision Tree Mechanisms

- Split the nodes recursively by using features and value which maximize the information gain.
- Then splitting mechanism is continued till a maximum-height condition is reached or there is no information gain from any of the possible splits. The terminating nodes are labelled as leafs in the end.

$$I_G = 1 - \sum_{i \in \ classes} p_i^2$$

$$IG(split) = I_G(node) - \sum_{c \in \ children} p(c) * I_G(c)$$

Example #1: Decision Tree Model

Decision Tree & Boundary

- Decision-tree was obtained without using any built-in functions.

Fig: Decision Tree

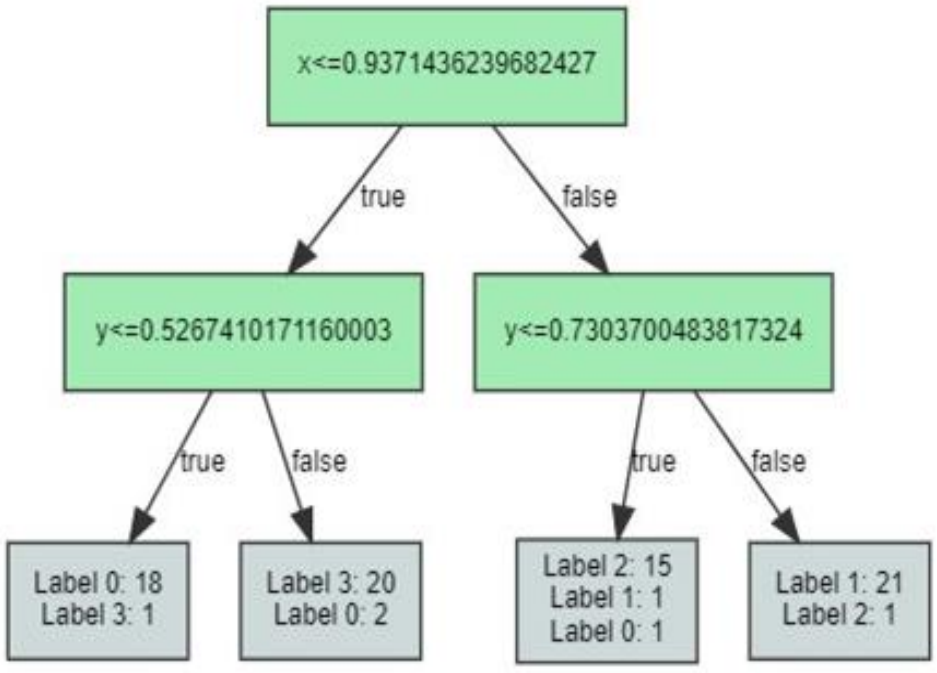
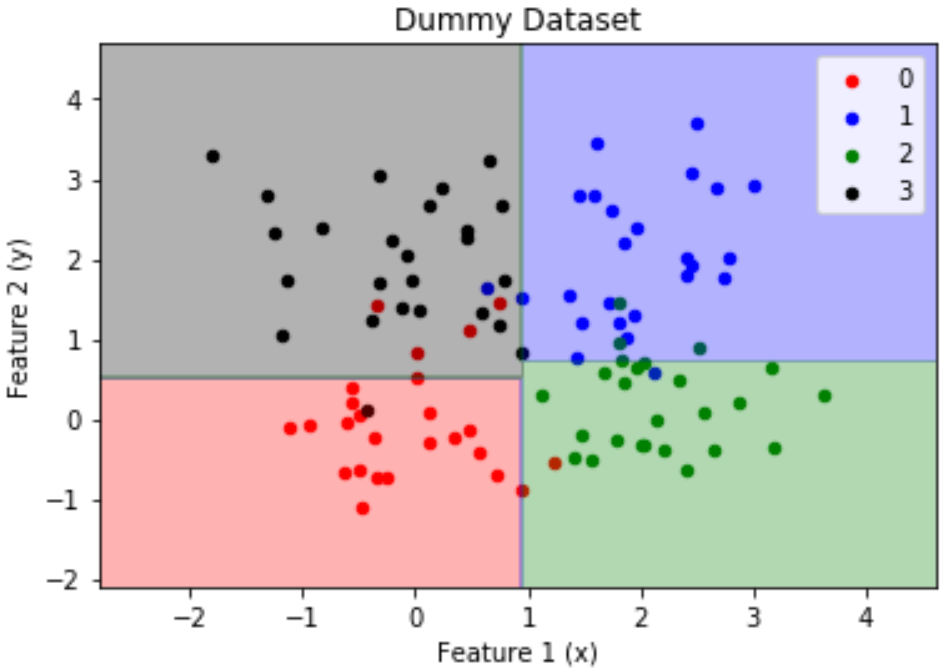


Fig: Decision Boundary of Decision Tree



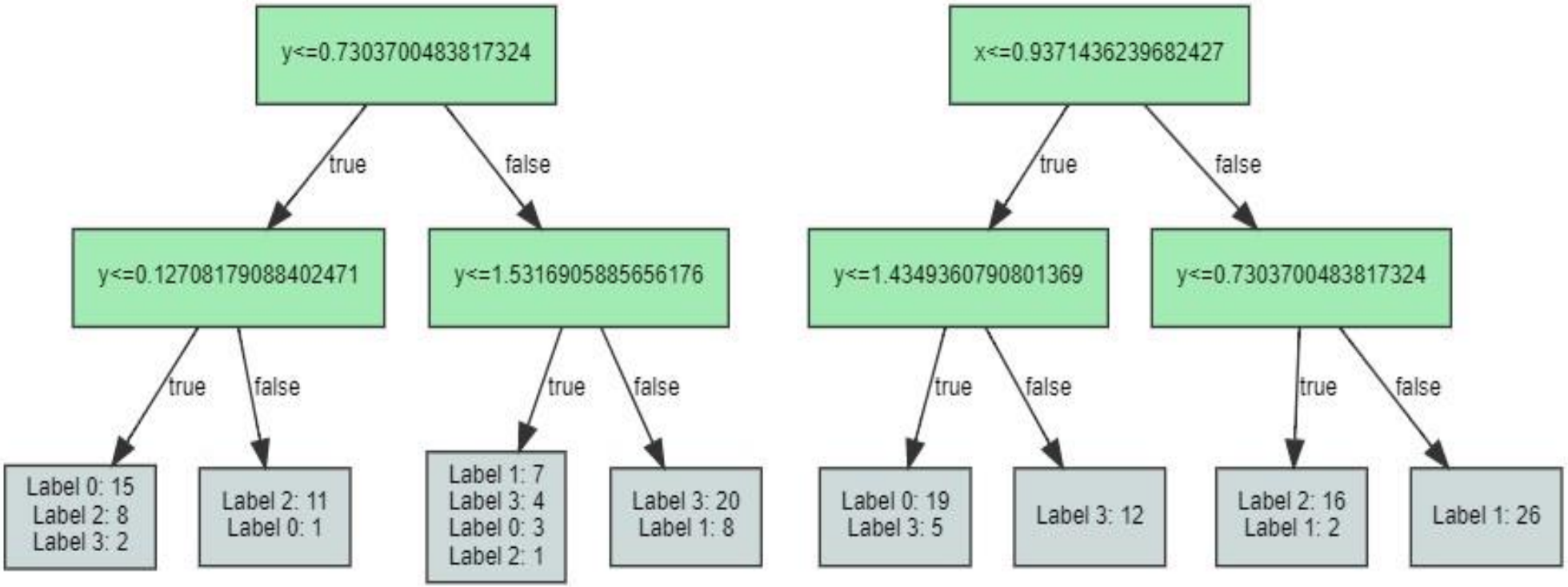
- Hyperparameters of a decision tree model are selection criterion and max-depth which can be tuned using cross validation.
- The dummy dataset is small and easily separable thus tuning is not needed.

Example #1: Random Forest Model

Random Forest

- Using the algorithm defined above an ensemble of decision trees was made.
- The decision trees in the forest (ensemble) were different as different subsets of data was used to train each tree and used a different random subset of features to split a node.

Fig: Two trees of the random forest model



Example #1: Random Forest Model

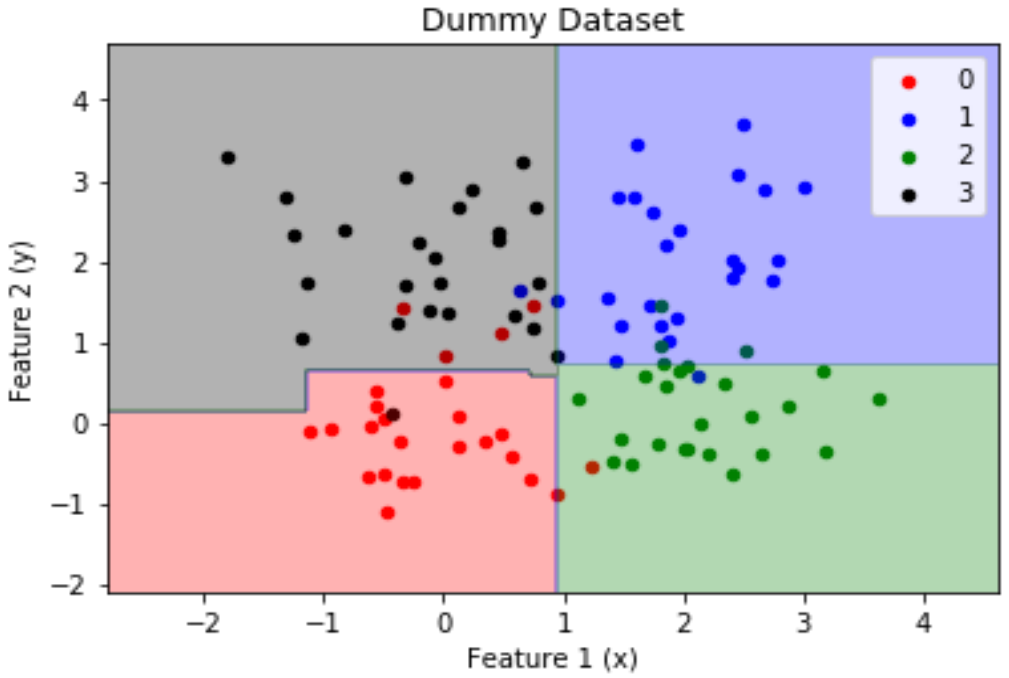
Random Forest Hyperparameters

- The hyperparameters for the random forest model is number of trees, maximum depth of trees and selection criterion for the split.
- All these hyperparameter can be tuned but since the data-set was small and easily separable it was not needed.

Predictions

- Final random forest model uses for final prediction had 10 trees, each with a maximum height of 2.
- The prediction from forest model was by using a voting scheme where maximum votes of all the trees in the forest was decided to be the predicted class of a test point.

Fig: Decision Boundary of Random Forest



Example #2: Decision Trees (Regression)

- **Dataset:**
  - 8 features
  - **Target:** Maximum temperature in Seattle
- **Function:**
  - DecisionTreeRegressor() from sklearn
- **Parameter:**
  - max_depth
- **Results:**
  - Bias remains low, variance increases **-> Overfitting**
  - Loss Function decreases initially; increases when max_depth > 5
  - After Tuning:
    - Training Accuracy = 87 %
    - Testing Accuracy = 82 %

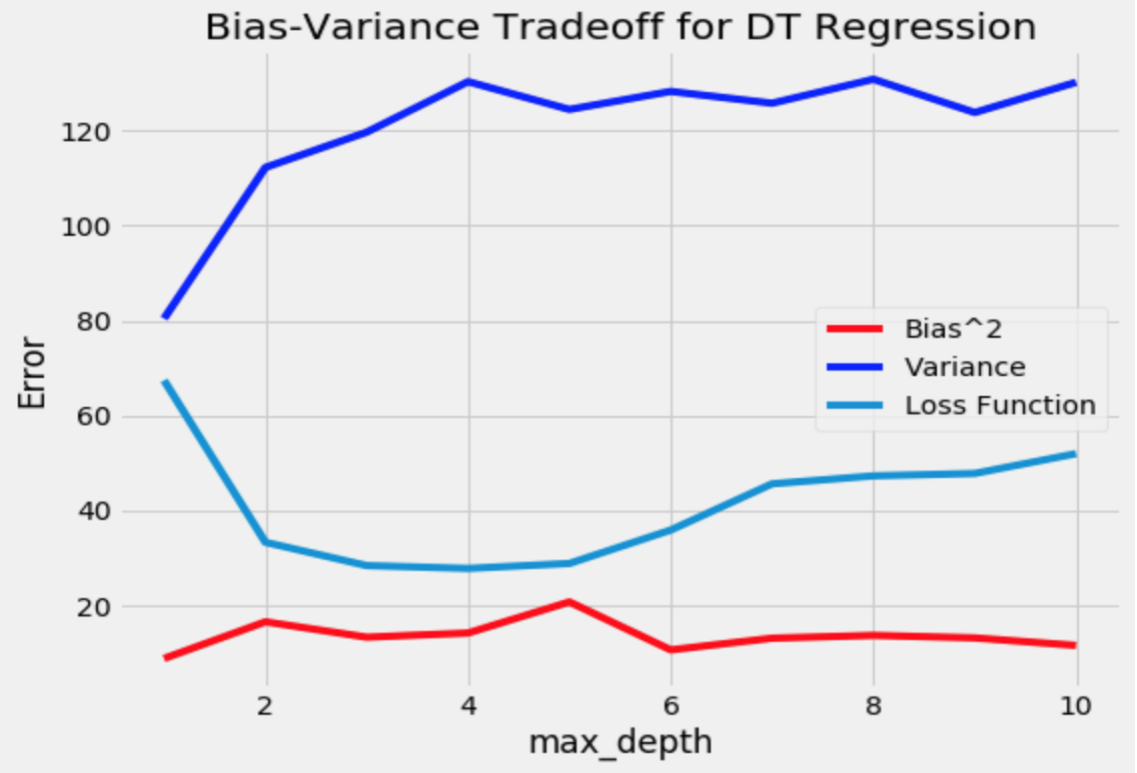
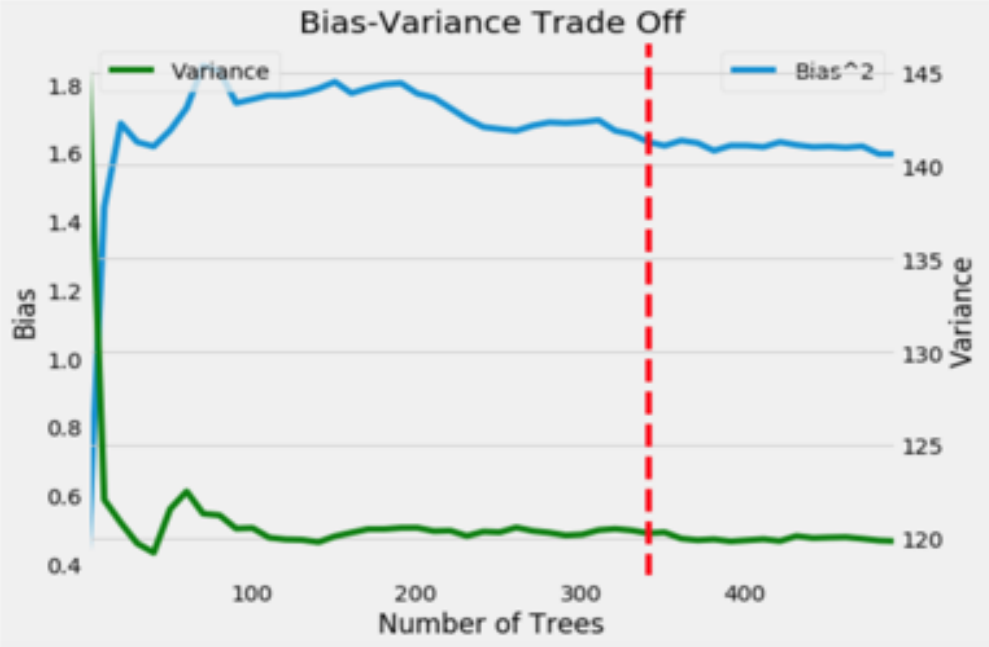
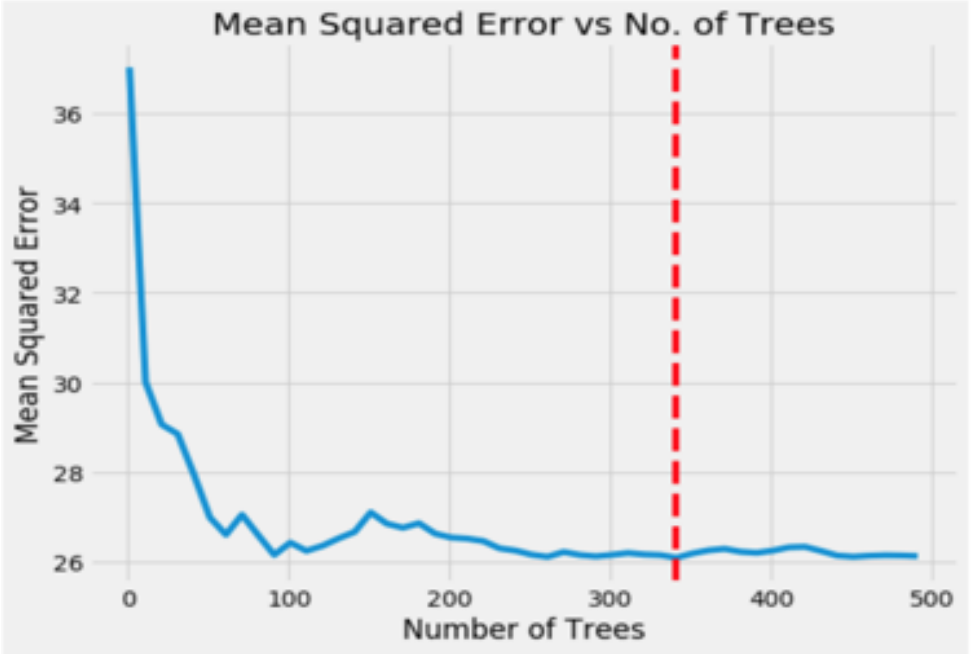


Fig: Bias-Variance tradeoff for DT regressor (temperature dataset)

Example #2: Random Forest (Regression)

- **Function:**
  - Using the RandomForestRegressor from sklearn()
- **Parameters:**
  - Criterion – the error the model uses to evaluate each split (MSE)
  - n_estimator – No. of Trees
- **Results:**
  - Decrease in variance (from 145 to 122)
  - Minimal increase in bias (from 0.47 to 1.59)
  - Training Accuracy = 97%, Testing Accuracy = 83%





Example #3: Decision Trees (Classification)

- **Dataset:**
  - 8 features
  - **Binary Target:** 1 (Patient has diabetes), 0 (Patient does not have diabetes)
- **Function:**
  - DecisionTreeClassifier() from sklearn
- **Parameter:**
  - max_depth
- **Results:**
  - Bias remains low, variance increases **-> Overfitting**
  - Loss Function increases
  - After Tuning:
    - Training Accuracy = 80%
    - Testing Accuracy = 77%

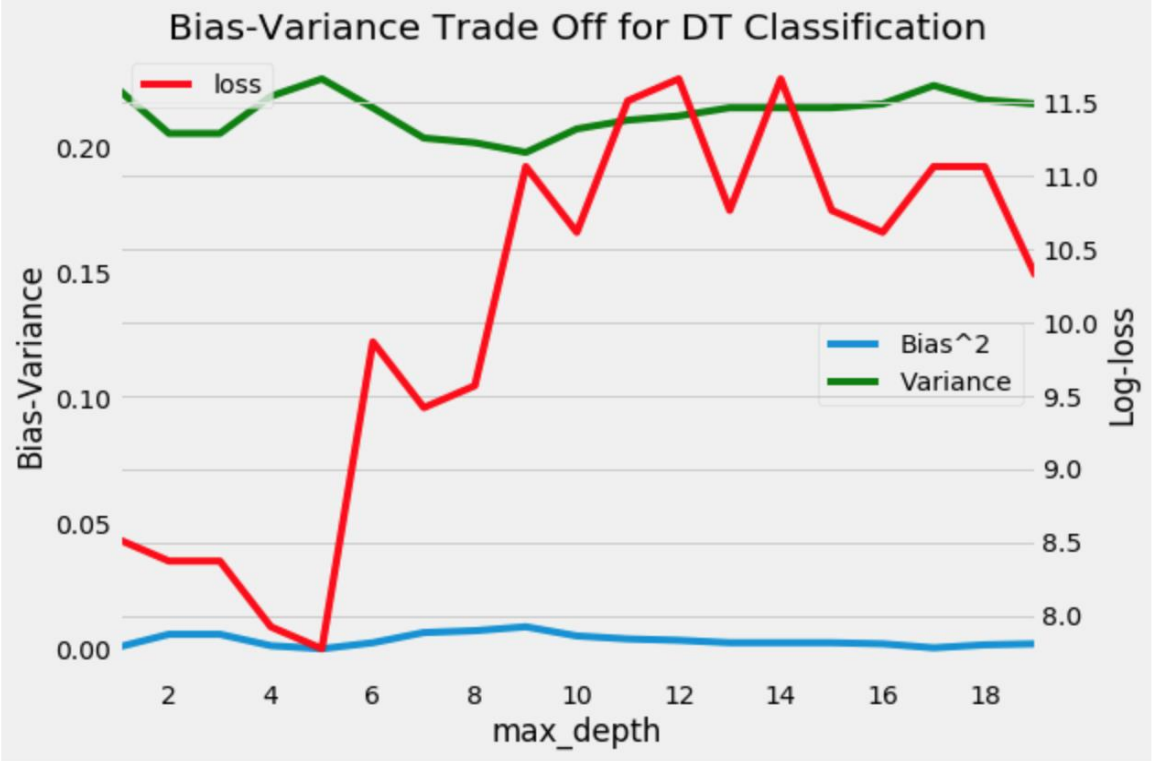
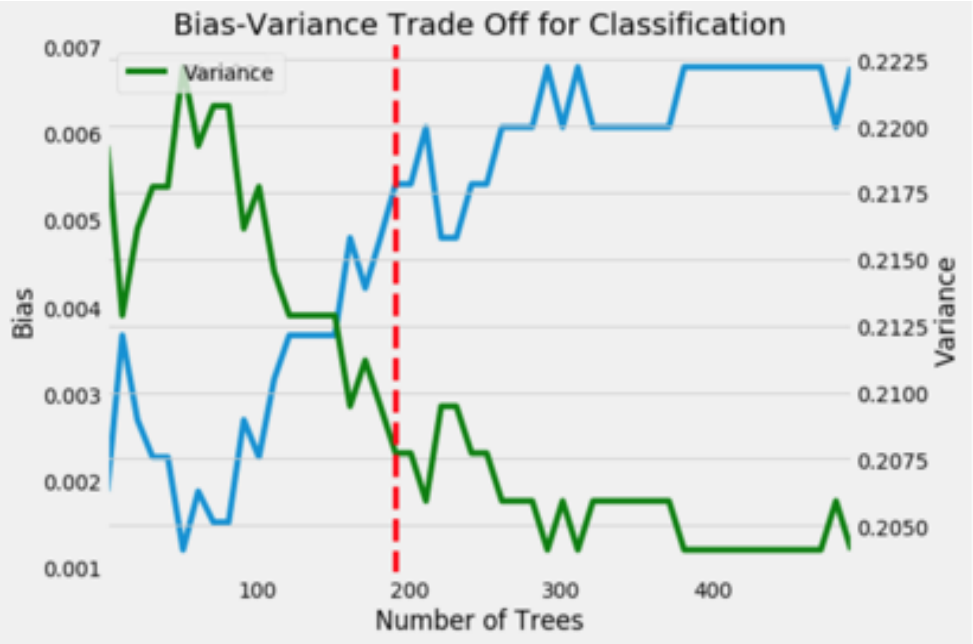
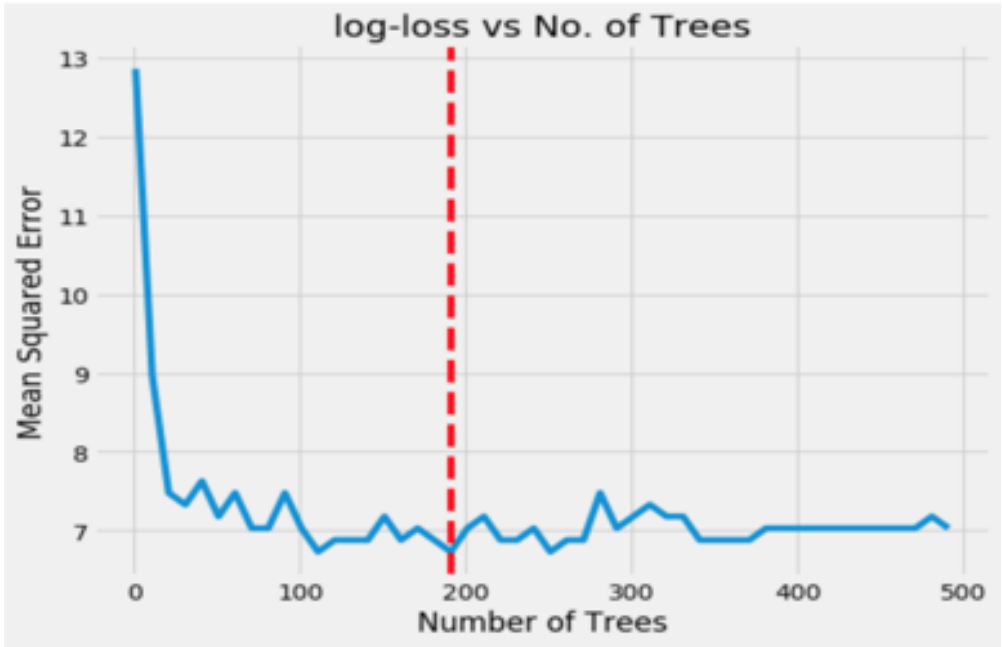


Fig: Bias-Variance tradeoff for DT regressor (diabetes dataset)

Example #3: Random Forest (Classification)

- **Function:**
  - Using the RandomForestClassifier from sklearn()
- **Parameters:**
  - Criterion – the error the model uses to evaluate each split (gini)
  - n_estimator – No. of Trees
- **Results:**
  - Decrease in variance, minimal increase in bias
  - Decrease in loss error
  - Training Accuracy = 100%, Testing Accuracy = 80%





Conclusion

Random Forest vs Bagging

- Both use bootstrapping method by taking independent samples from existing data with replacement and use each sample to train learners.
- Random Forest makes a small tweak to Bagging which results in a great improvement and creates a strong classifier.
- Random forest randomly selects a set of features which are used to decide the best split at each node of the decision tree.
- Bagging uses all the features

Cross validation for Random Forest

- Cross validation is computationally intensive for random forest
- Out of Bag Error is computed as alternative.
- The bootstrap sample does not contain all the data points in the original data.
- These out of bag points can be used as test case to get an unbiased estimate of the error.

thanks!