Bagging and Random Forest

Chris Reger

(Adapted from Erich Wellinger and Brent Lemieux)

galvanize

Objectives

galvanize

Morning:

- Review **Decisions Trees**
- Introduce the concept of **Ensemble Methods**
- Discuss **Bagging** (Bootstrap Aggregation) as a type of an Ensemble
- Discuss **Random Forest** and how it improves upon bagging

Afternoon:

- Discuss **Out-of-Bag (OOB) Score** as a method for evaluating model performance
- Discuss **Feature Importance** for model interpretability

Review

galvanize

Review **Decisions Trees**

- What metrics are used to describe the impurity in a node of categorical variables?
- What about for a continuous variable?
- How is the value of each split determined?
- Are decision trees deterministic? Parametric or nonparametric?
- Describe the bias and variance of a fully-grown tree.
- How do we reduce the variance of a decision tree?
- What are some hyperparameters associated with decision trees?
- How is regression performed with decision trees?

Decision Trees Review

galvanize

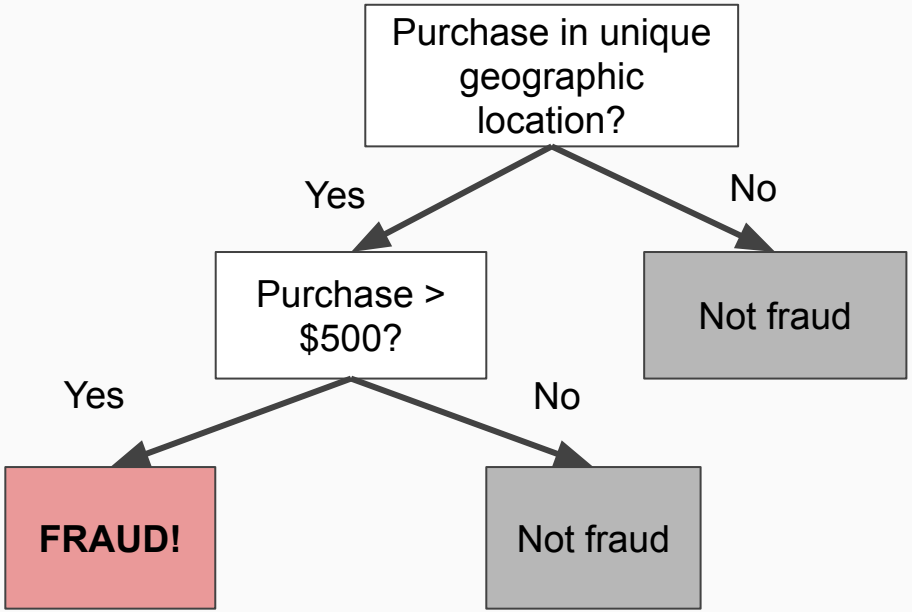
Decision Trees are deterministic:

- Once they are trained, they will arrive at the same conclusion every time given the same X values.

Decision Trees are "nonparametric":

- We don't make any assumptions (mean, variance, etc.) about our data when we create a tree.

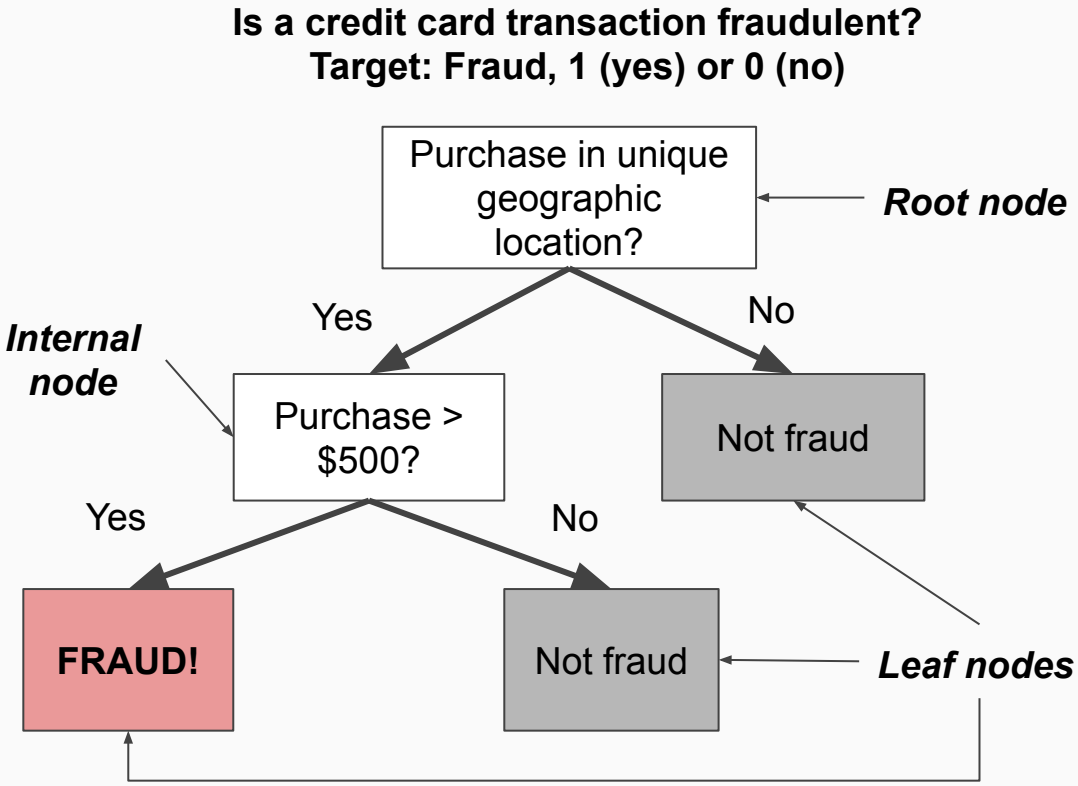
Is a credit card transaction fraudulent?
Target: Fraud, 1 (yes) or 0 (no)



Decision Trees Review

galvanize

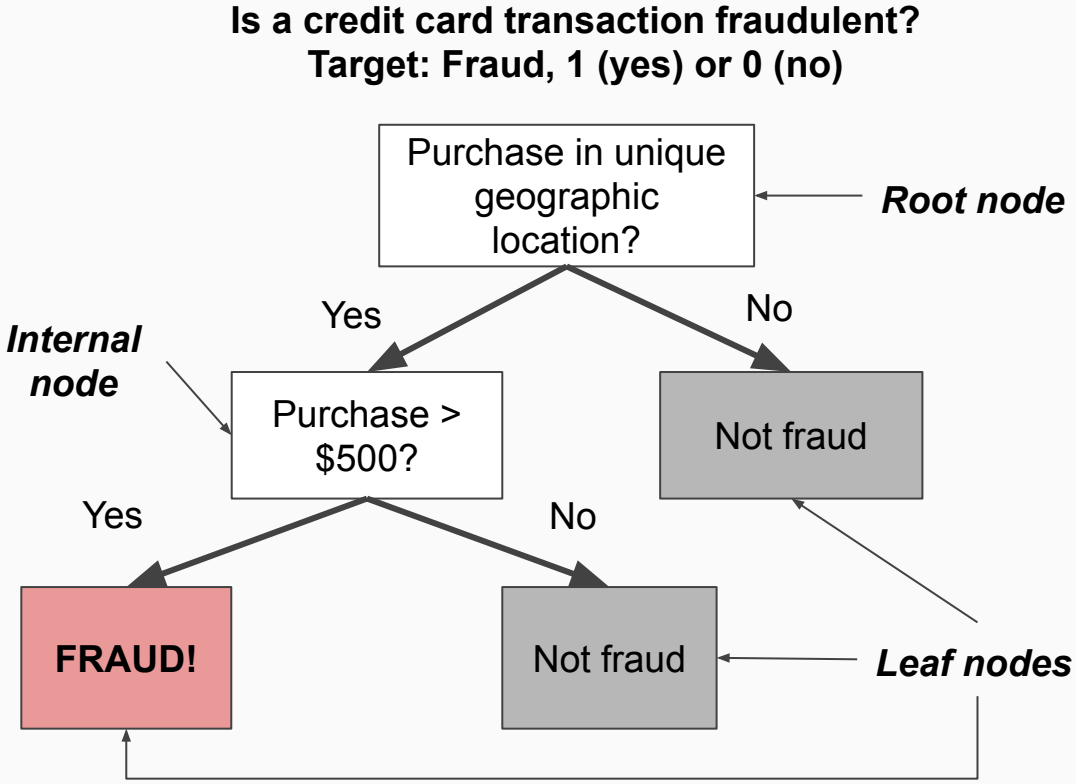
- **Specify a target** (fraud) **and features** (purchase geotag, $ amount, known vendor, etc.).
- Model looks at each of the features and and splits on the feature that **maximizes information gain** about the target variable at each step.
- Each group of data is called a **node**
- The final decision points are called **leaf nodes**



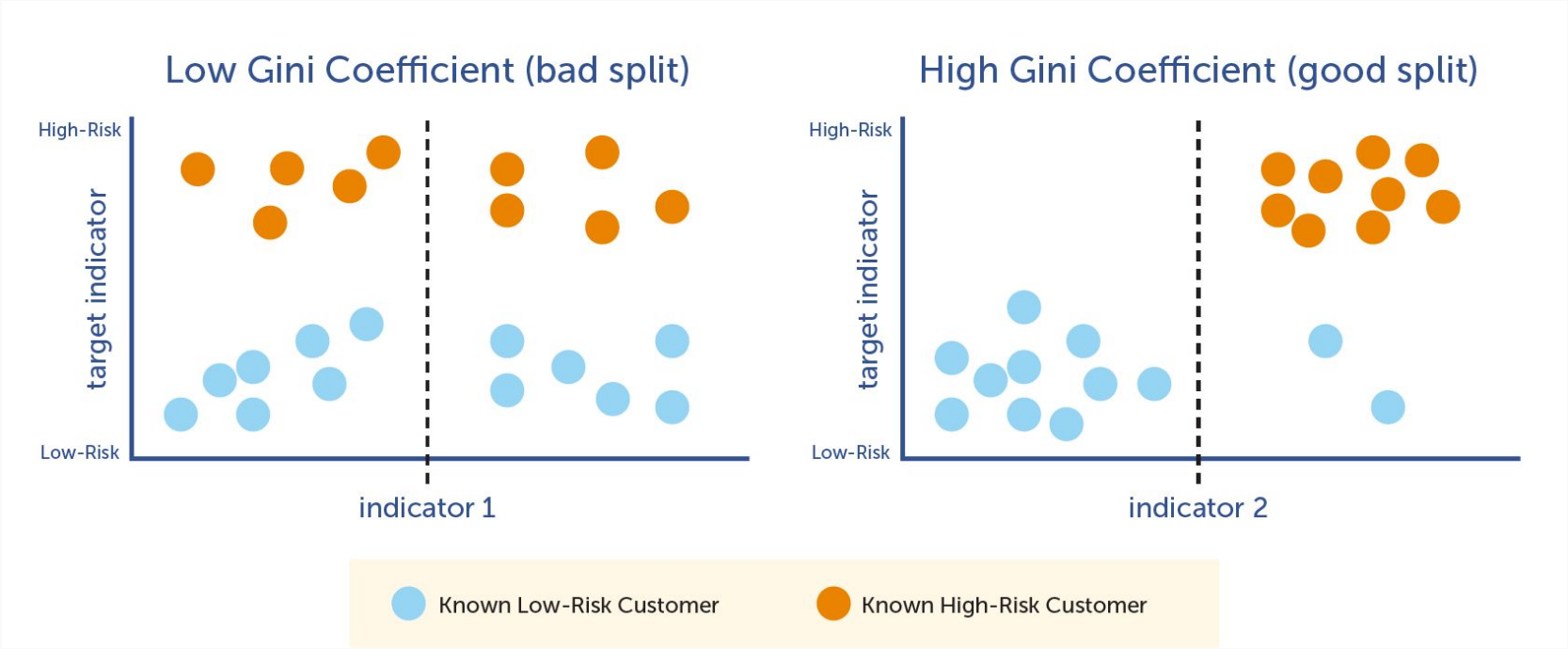
Decision Trees: Classification

galvanize

- Impurity of each node is measured on *Shannon Entropy* or *Gini Index*
- Information gain is the weighted reduced impurity from the split
- Model iterates through all possible splits to find the best one
- Model predicts based off most common class in leaf node



Decision Trees: Classification



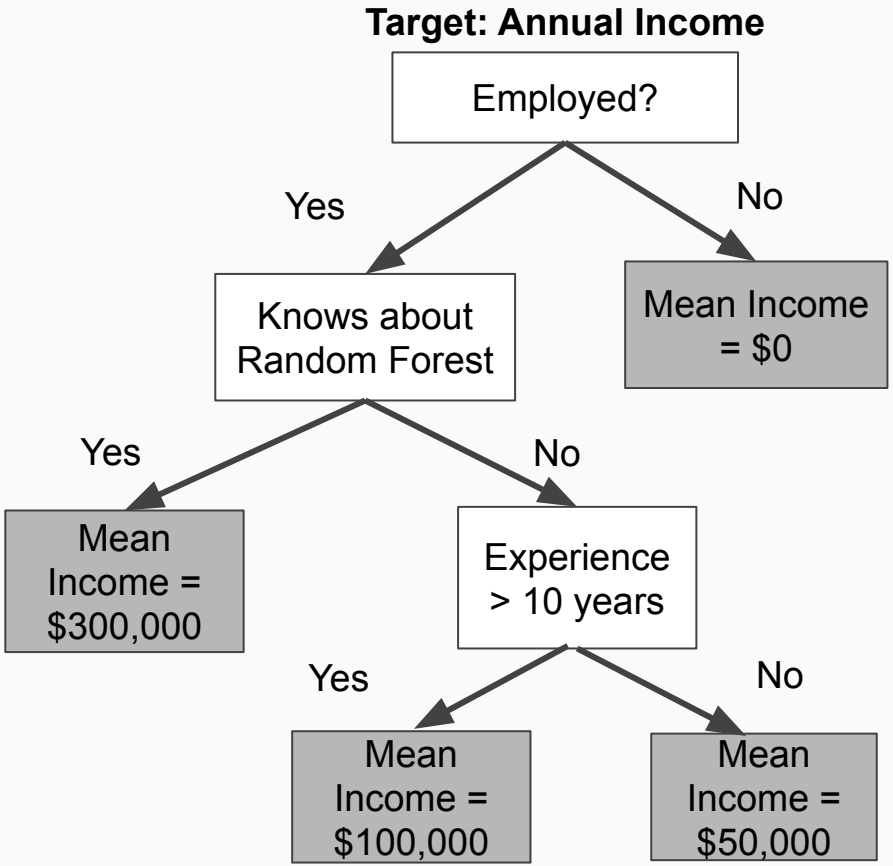
Decision Trees: Regression

- Each split tries to minimize the Total Squared Error

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Where J is the number of child nodes and R is each child node

- The model output is typically the mean within each leaf node. Or... train a new model within each leaf! For instance, we could have several linear models within our tree.



What are some pros/cons associated with Decision Trees?

Pros and Cons of Decision Trees

galvanize

Pros

- Handle non-linear relationships well
- Highly interpretable (unless we have a ton of features)
- Useful for classification and & regression
- No feature scaling required

Cons

- Expensive to train (has to iterate through each of the options to split on)
    - For continuous X-variables, the model also has to iterate through different threshold values
- Often poor predictors (high variance)

Remember, we can prune these trees by changing our hyper-parameters to limit some of the cons… What happens if we prune too much?

Objectives

galvanize

Morning:


Review **Decisions Trees**

- Introduce the concept of **Ensemble Methods**
- Discuss **Bagging** (Bootstrap Aggregation) as a type of an Ensemble
- Discuss **Random Forest** and how it improves upon bagging

Afternoon:

- Discus **Out-of-Bag (OOB) Score** as a method for evaluating model performance
- Discuss **Feature Importance** for model interpretability

Ensemble Methods

E**nsemble methods** involve combining **multiple weak learners** to make **one strong learner**

Two general categories of ensemble methods: Aggregators (think of “Ask the Audience” lifeline on Who Wants to Be a Millionaire) and iterators.

Aggregators:

1. Train multiple models on the data
2. Predict using average (regressors) or plurality choice/percentages (classifiers)

Ensemble Methods - Decision Trees

- Overfitting is a common pitfall of decision trees
- Let's try using a large number of trees to **reduce variance**
- What if we just build a bunch of decision trees and average the results?
- Do you foresee any issues?

Ensembled Trees - Intuition

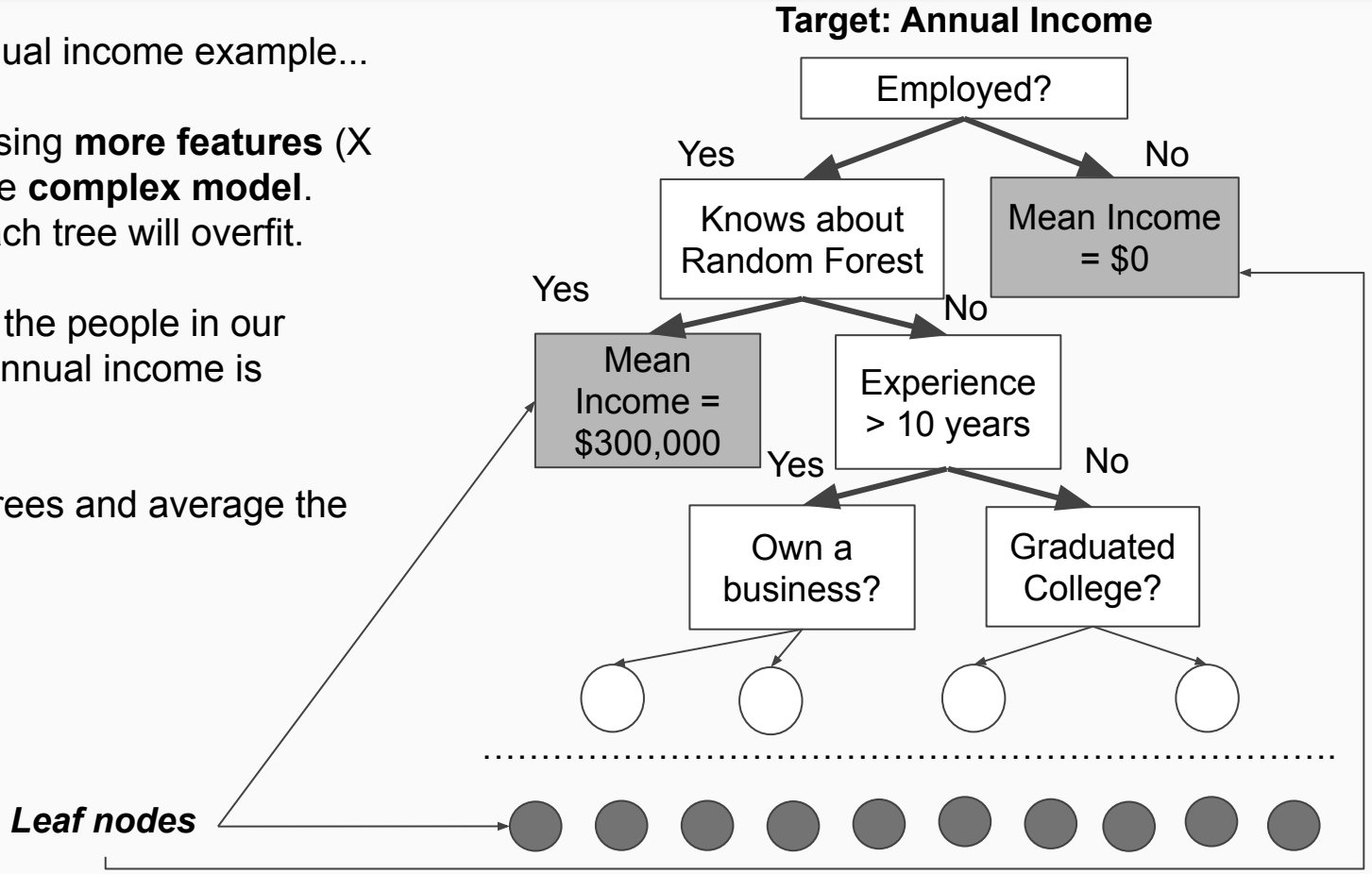
galvanize

Let's revisit our annual income example...

This time, we are using **more features** (X values), thus a more **complex model**. Without pruning, each tree will overfit.

Let's say for one of the people in our dataset, their true annual income is $68,000...

Let's grow a lot of trees and average the predictions.

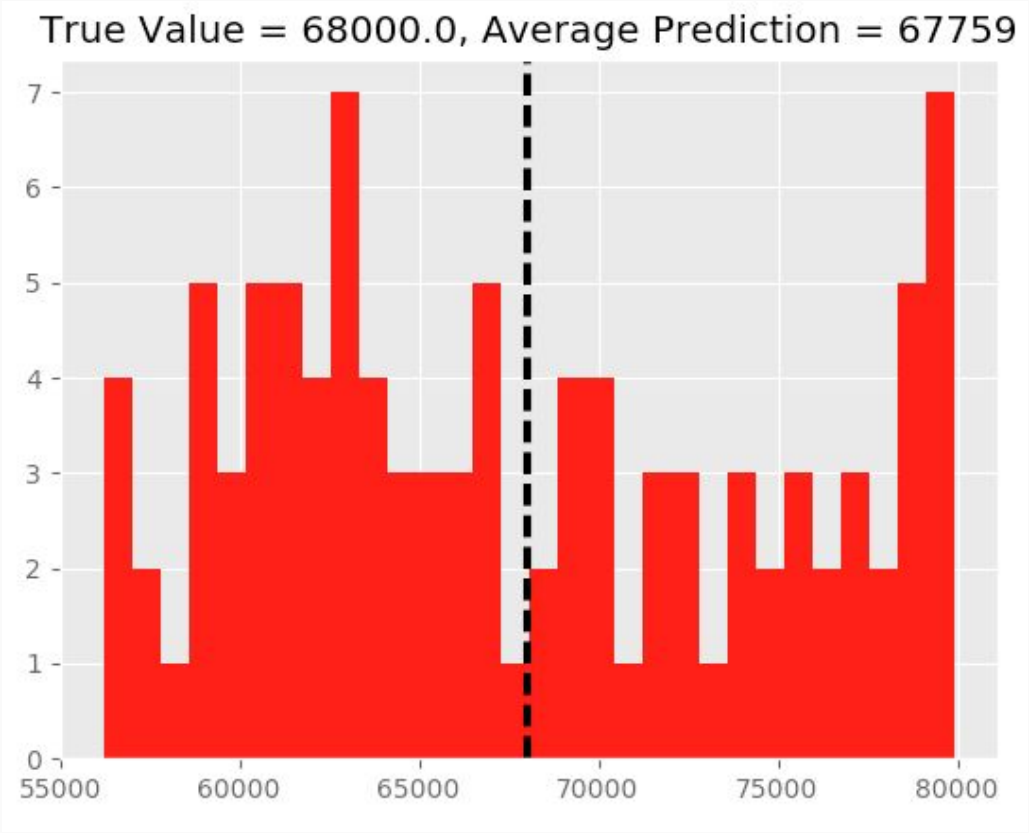


Ensembled Trees - Intuition

If we grow a lot of trees (100 in this case), the average prediction will be a much stronger predictor -- at least in theory...

Ensembled Model →

(Decision trees are deterministic)



Ensembled Trees - Intuition

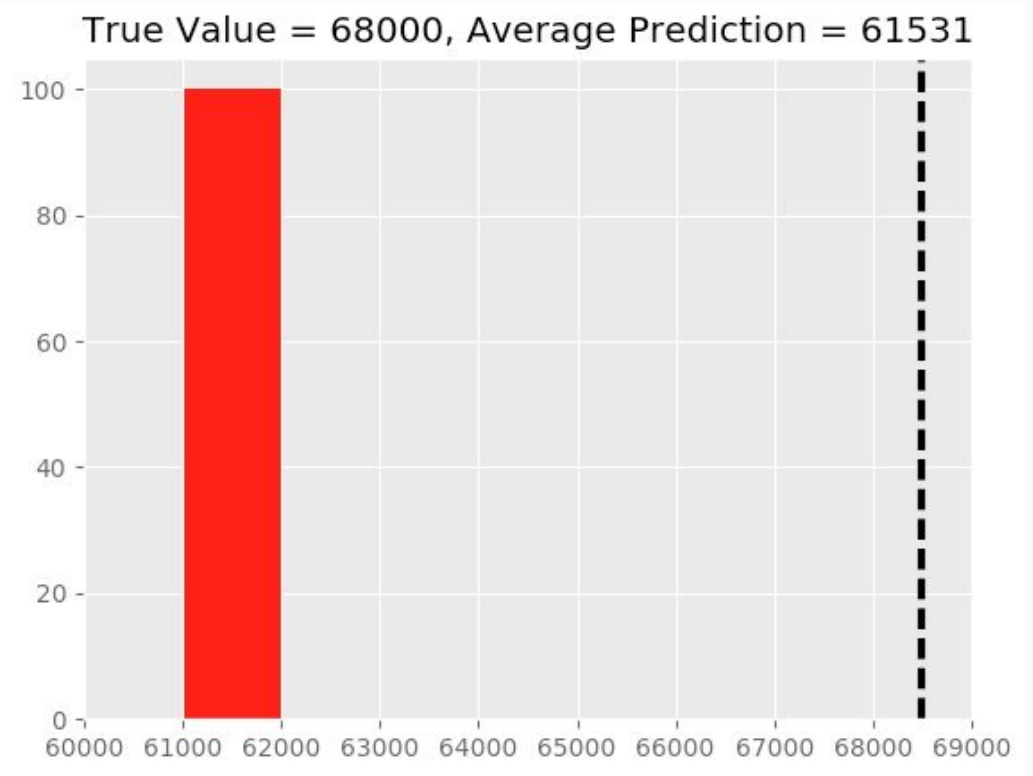
galvanize

Why can't we just grow a bunch of trees with our data and aggregate them to get a strong learner???

If our **trees all get the same data**, they will all **give the exact same prediction**. *We are polling the same audience member repeatedly.*

How do we get around this if we only have one set of data?

Hint:



Objectives

galvanize

Morning:

- ✓ Review **Decisions Trees**
- ✓ Introduce the concept of **Ensemble Methods**
- Discuss **Bagging** (Bootstrap Aggregation) as a type of an Ensemble
- Discuss **Random Forest** and how it improves upon bagging

Afternoon:

- Discuss **Out-of-Bag (OOB) Score** as a method for evaluating model performance
- Discuss **Feature Importance** for model interpretability

galvanize

Bootstrapping

What is a bootstrap?

- Given a sample of *n* data points, we take *B* bootstrapped samples of our data **with replacement**
- We typically use the bootstrap to get confidence intervals around a statistic/parameter

Bagging (Bootstrap Aggregation)

galvanize

Decision trees have high variance.

To reduce our models variance, we need *independent samples** to train our models on…

- We can draw new samples from our population (or bootstrap them)
- Train a model on each new sample
- Average the predictions from each

* Bootstrapping *does not give us independent samples*, **but it is an improvement**

Why does this reduce the variance of our model? From ISLR:

- Recall that given a set of n independent observations $Z_1,\ldots,Z_n$, each with variance σ^2, the variance of the mean Z-bar of the observations is given by σ^2/n. In other words, averaging a set of observations reduces variance.

Bagging (Bootstrap Aggregation)

galvanize

- What did we assume about our estimators?
  - Independence
  - Bootstrapped samples are not independent

$$Var\left(\hat{f}_{bag}(x_0)\right) = \rho\sigma^2 + \frac{(1-\rho)\sigma^2}{B}$$

- ρ = rho, the correlation coefficient for the different samples
- As rho goes towards one, the effect of multiple estimators on decreasing variance weakens
- Only the uncorrelated parts of the sample decrease variance

Bagging = Bootstrap -> Fit a model -> Repeat -> Aggregate

galvanize

- Trees are the most common model we use with bagging, but we can use others!
- We typically grow the trees deep (no pruning) which gives each individual high variance, but low bias.
- Aggregating the outputs of our *B* trees serves to give us a large reduction in variance at the cost of a relatively small increase in bias.

Bagging Continued...

galvanize

A note on the number of trees...

- The number of trees *B* is not a critical parameter. Higher *B* won't lead to overfitting, because adding additional different trees will only reduce overfitting.
- In practice, we want to increase *B* until our test error stabilizes. At a certain point, increasing *B* won't help our model generalize to unseen data much more and increasing *B* is computationally expensive!

Bagging Review

galvanize

- What is an ensemble method?
- Why do we use to bootstrap?
- The general idea of bagging decision trees is to start with ___ bias, ___ variance trees and aggregate across multiple models to decrease _____
- Why are bootstrapped samples correlated?

Decorrelating Trees

galvanize

- Bagged trees are correlated because:
  - Bootstrap samples are about the same (approximately $^2/_3$)
    - Can’t do much about this
  - Influential features tend to be the same
    - We can fix this

Objectives

galvanize

Morning:

- ✓ Review **Decisions Trees**
- ✓ Introduce the concept of **Ensemble Methods**
- ✓ Discuss **Bagging** (Bootstrap Aggregation) as a type of an Ensemble
- Discuss **Random Forest** and how it improves upon bagging

Afternoon:

- Discuss **Out-of-Bag (OOB) Score** as a method for evaluating model performance
- Discuss **Feature Importance** for model interpretability

galvanize

Random Forest

What is it?

Why is it better than simply bagging with decision trees?

Random Forest

galvanize

Random Forest improves over bagging with a small tweak that **decorrelates** the trees -- in other words, *it makes the individual trees more independent of each other.*

At each split in each tree, Random Forest randomly selects *m* features to consider for the split -- oftentimes the default ***m = sqrt(p)*** where *p* is the number of features we are using to train our model

- This is called ***subset sampling***
- Subset sampling ensures our trees don't always choose the same splits at each level!

For this illustration we are only following one branch after our initial split...

Features	Employment Status	Years Experience	Education Level	Knows Random Forest or not	Owns a business or not
Split 1				**SPLIT!**	
Split 2	**SPLIT!**				
Split n		**SPLIT!**			

Random Forest - Predict Income

galvanize

Tree 1

Features	Employment (0 or 1)	Experience (# years)	Education (# years)	Know RF? (0 or 1)	Own biz? (0 or 1)
Split 1				**SPLIT!**	
Split 2	**SPLIT!**				
Split n			**SPLIT!**		

Tree 2

Features	Employment (0 or 1)	Experience (# years)	Education (# years)	Know RF? (0 or 1)	Own biz? (0 or 1)
Split 1	**SPLIT!**				
Split 2					**SPLIT!**
Split n			**SPLIT!**		

Tree B

Features	Employment (0 or 1)	Experience (# years)	Education (# years)	Know RF? (0 or 1)	Own biz? (0 or 1)
Split 1			**SPLIT!**		
Split 2	**SPLIT!**				
Split n			**SPLIT!****		

Considered for split	Not considered for split	Can't be considered

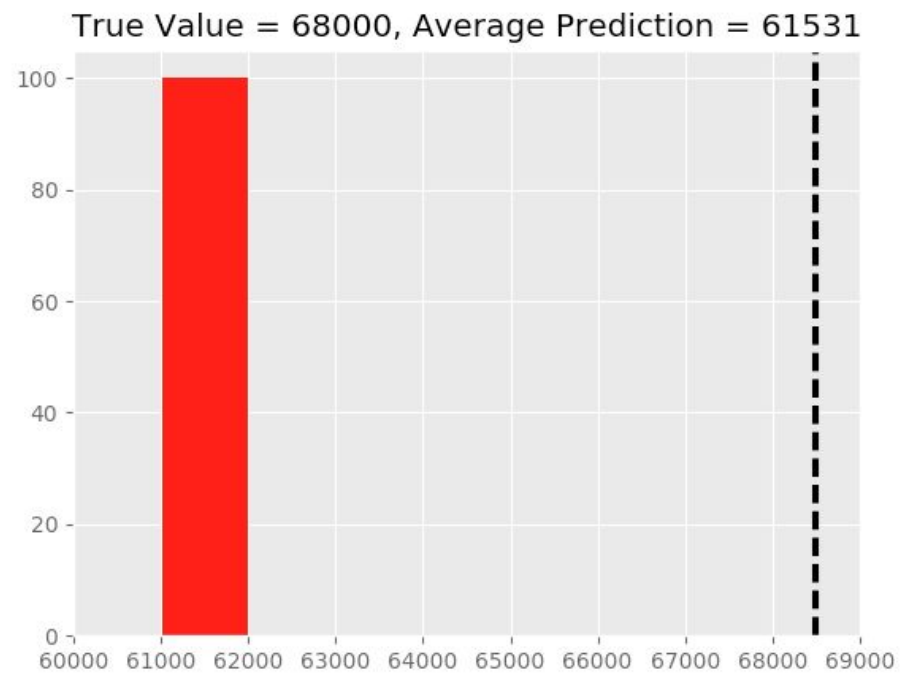
***Note: we can split on continuous variables more than once -- even within the same branch.*

Why is decorrelating trees so important?

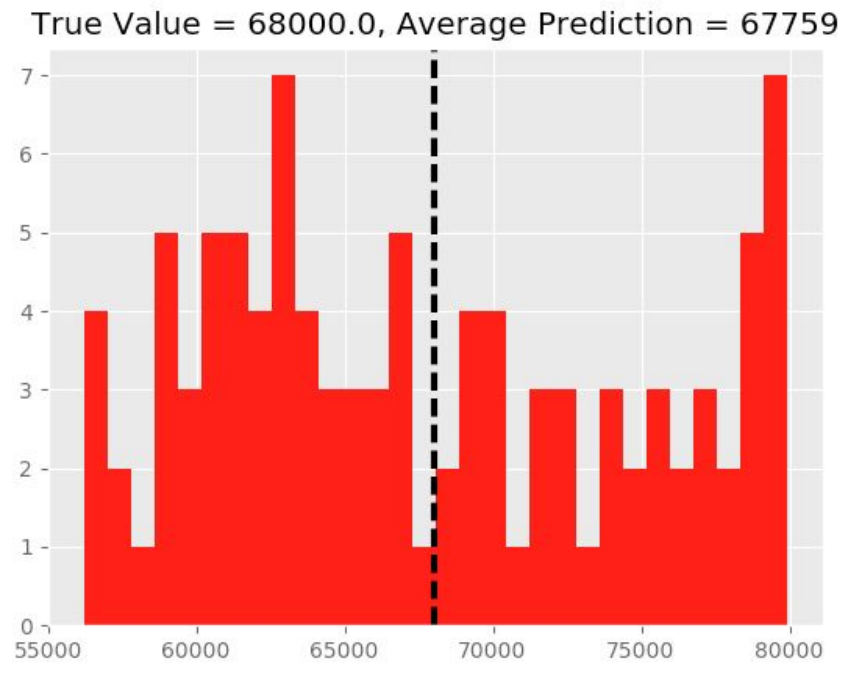
galvanize

Recall from our previous example…

High Correlation Between Trees



Low Correlation Between Trees



Random Forest Hyperparameters

galvanize

Review: Parameters vs. Hyperparameters

- Parameters are attributes of our data
- Hyperparameters are attributes of the machine learning algorithms we use to model our data

class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=10*, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_split=1e-07*, *bootstrap=True*, *oob_score=False*, *n_jobs=1*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*)

Random Forest Hyperparameters:

- Number of trees in the forest = ***n_estimators***
- Information gain metric = ***criterion***
- Number of features to consider for split = ***max_features***
- Individual tree hyperparameters -
    - Tree pruning ***(max_depth, max_leaf_nodes, min_samples_split, etc.)***

For the most part, Random Forest is very robust to the choice of hyperparameters and overfitting.

Pros and Cons of Random Forest

galvanize

Pros

- Often gives near state of the art performance
- Good out of the box performance (hyperparameter tuning not as beneficial as some other models)
- No feature scaling needed
- Models non-linear relationships well
- Can be trained in parallel on multiple machines (more on this during big data week)

Cons

- Expensive to train
- Can be very large to store (especially as you increase the number of trees/n_estimators) -- can be several GBs
- Not easily interpretable (although more interpretable than more complex models)

galvanize

Random Forest Trivia

How do we decide on the set of features to consider for a split at each node?

Random Forest Trivia

How do we decide on the set of features to consider for a split at each node?

Randomly… We randomly select m (which we can define) features at each node.

Random Forest Trivia

How do we decide on the set of features to consider for a split at each node?

Randomly… We randomly select m (which we can define) features at each node.

Once we have selected m features, how do we decide which one to split on?

Random Forest Trivia

How do we decide on the set of features to consider for a split at each node?

Randomly… We randomly select m (which we can define) features at each node.

Once we have selected m features, how do we decide which one to split on?

Our information gain criterion… Entropy, Gini for classification and RSS for regression.

Random Forest Trivia

How do we decide on the set of features to consider for a split at each node?

Randomly… We randomly select m (which we can define) features at each node.

Once we have selected m features, how do we decide which one to split on?

Our information gain criterion… Entropy, Gini for classification and RSS for regression.

What are the reasons that make Random Forest a drastically lower variance model than single Decision Trees -- and even simple bagged models?

Random Forest Trivia

How do we decide on the set of features to consider for a split at each node?

Randomly… We randomly select m (which we can define) features at each node.

Once we have selected m features, how do we decide which one to split on?

Our information gain criterion… Entropy, Gini for classification and RSS for regression.

What are the reasons that make Random Forest a drastically lower variance model than single Decision Trees -- and even simple bagged models?

The trees that make up the forest are *more independent*. They don't train on all of the same data and they don't split on the same features. This makes them more robust and helps them generalize to new data.

Random Forest in scikit learn

We'll discuss this afternoon why cross validation isn't always necessary for Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import train_test_split, cross_val_score

import pandas as pd
import numpy as np

data = pd.read_csv('data.csv')

y = data.pop('target').values
X = data.values

X_train, X_test, y_train, y_test = train_test_split(X, y)

rf = RandomForestClassifier(n_estimators=10)

# Which metric is most important depends on the problem
print(np.mean(cross_val_score(rf, X_train, y_train, scoring='accuracy')))
print(np.mean(cross_val_score(rf, X_train, y_train, scoring='precision')))
print(np.mean(cross_val_score(rf, X_train, y_train, scoring='recall')))

# Decide on final hyperparameters and adjust above
rf.fit(X_train, y_train)
y_preds = rf.predict(X_test)

# Again, which metric to use depends on the problem
print(accuracy_score(y_test, y_preds))
print(precision_score(y_test, y_preds))
print(recall_score(y_test, y_preds))
```