Bagging and Random Forest

Brent Lemieux

(Adapted from Erich Wellinger)

Objectives

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

Decision Trees Review

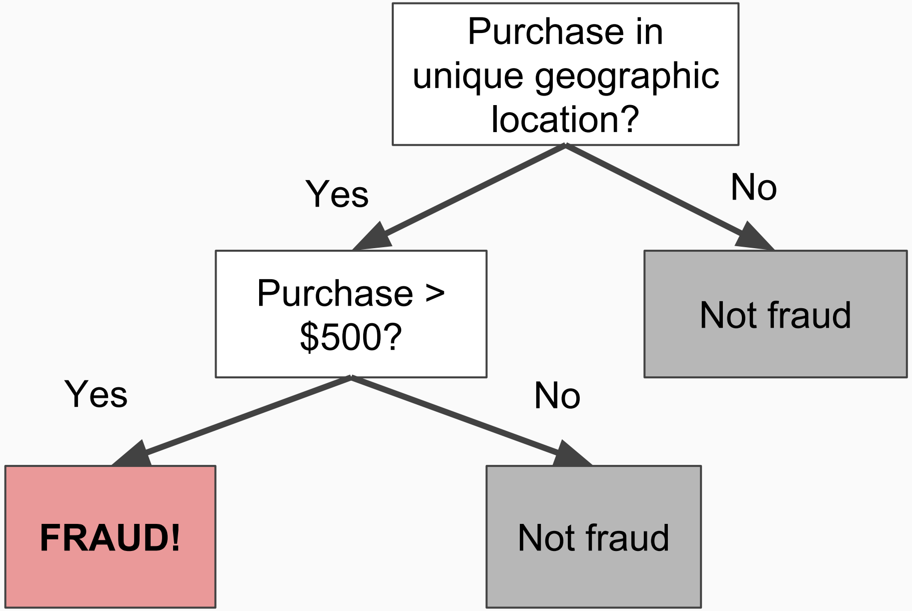
Decision Trees are deterministic:

- 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.

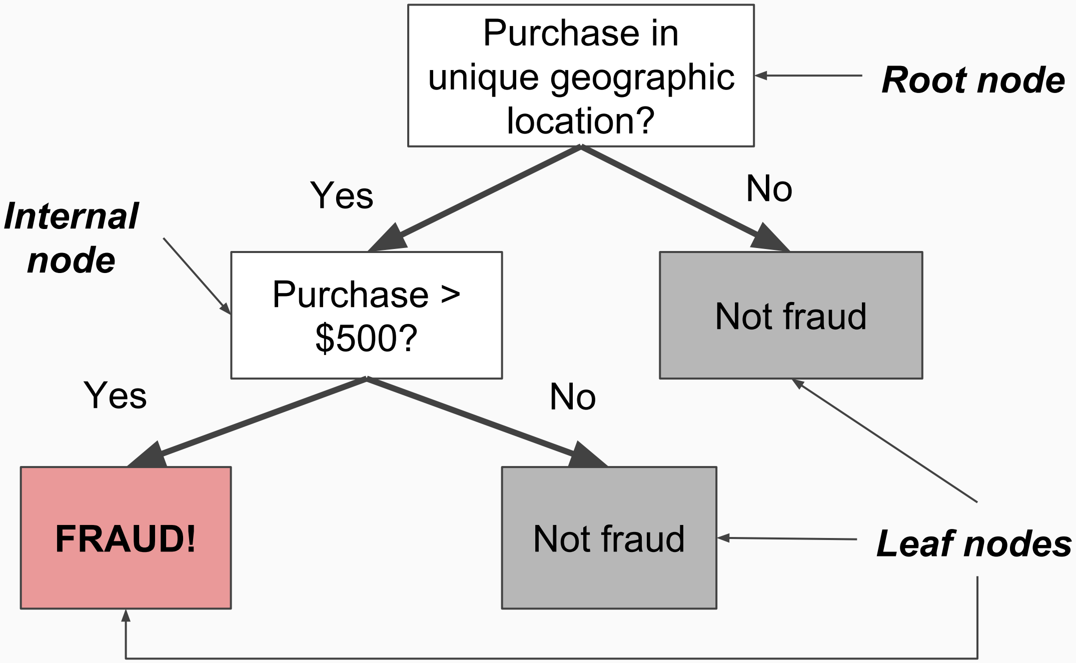
Is a credit card transaction fraudulent?
Target: Fraud, 1 (yes) or 0 (no)



Decision Trees Review

- **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**

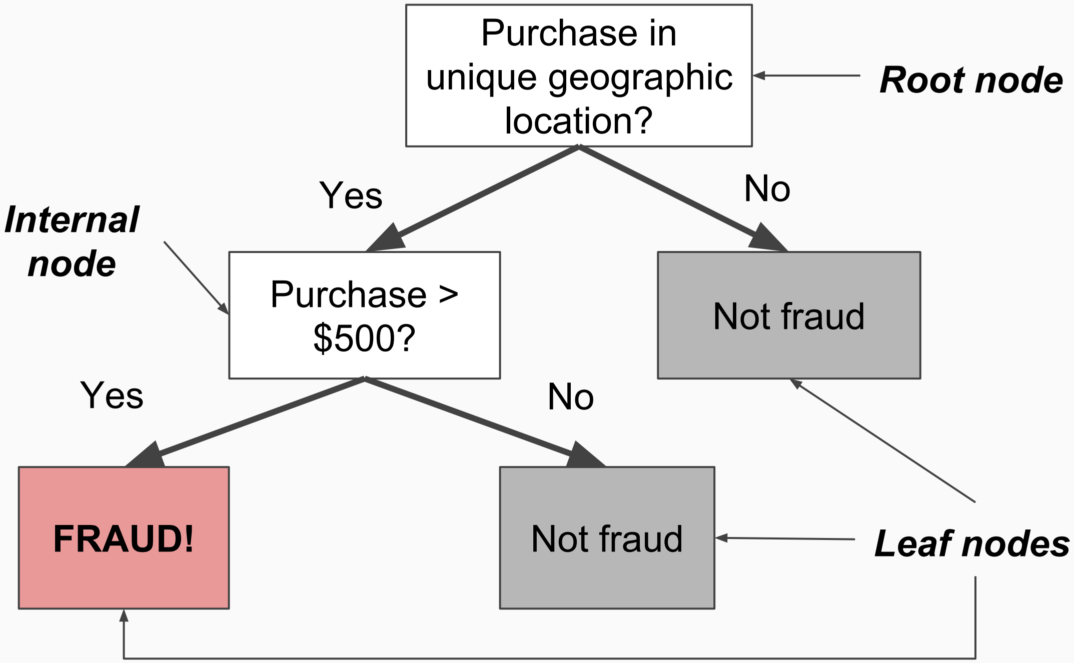
Is a credit card transaction fraudulent?
Target: Fraud, 1 (yes) or 0 (no)



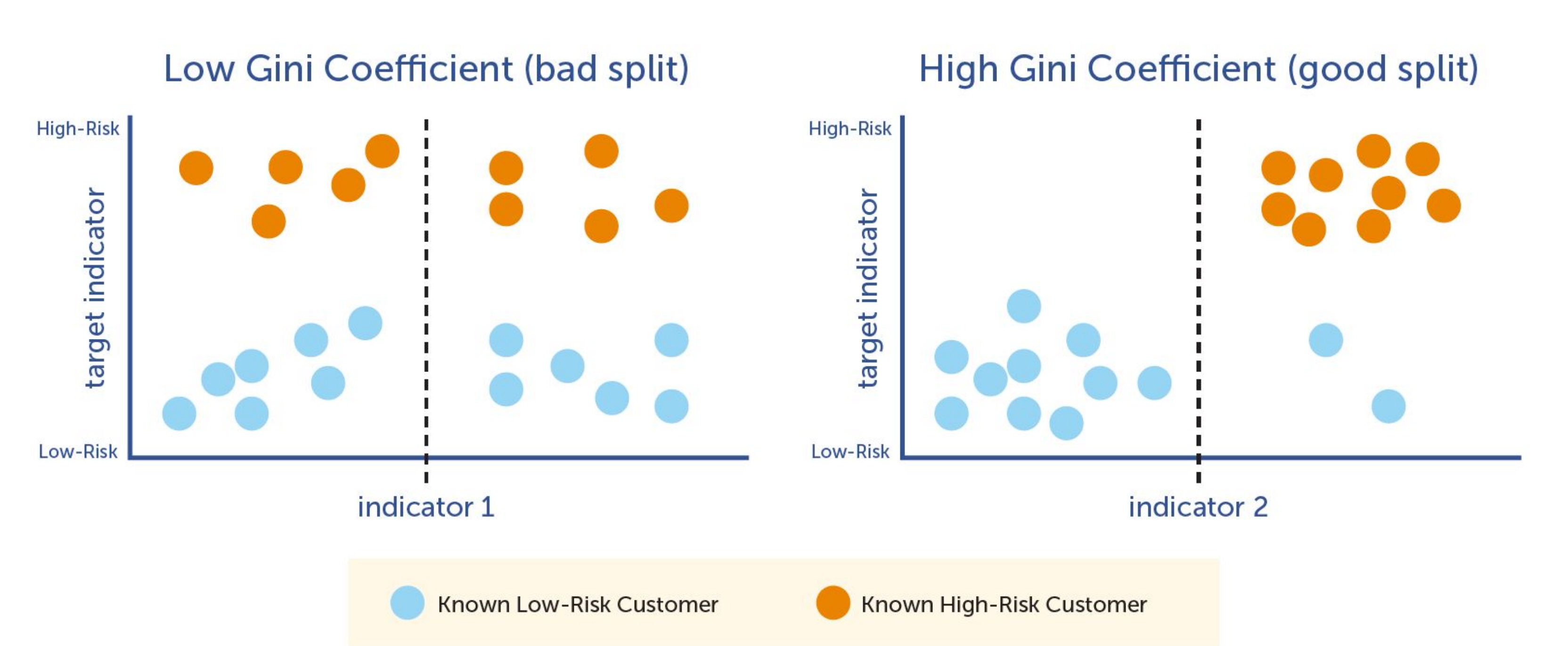
Decision Trees: Classification

- 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

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



Decision Trees: Classification



Decision Trees: Regression

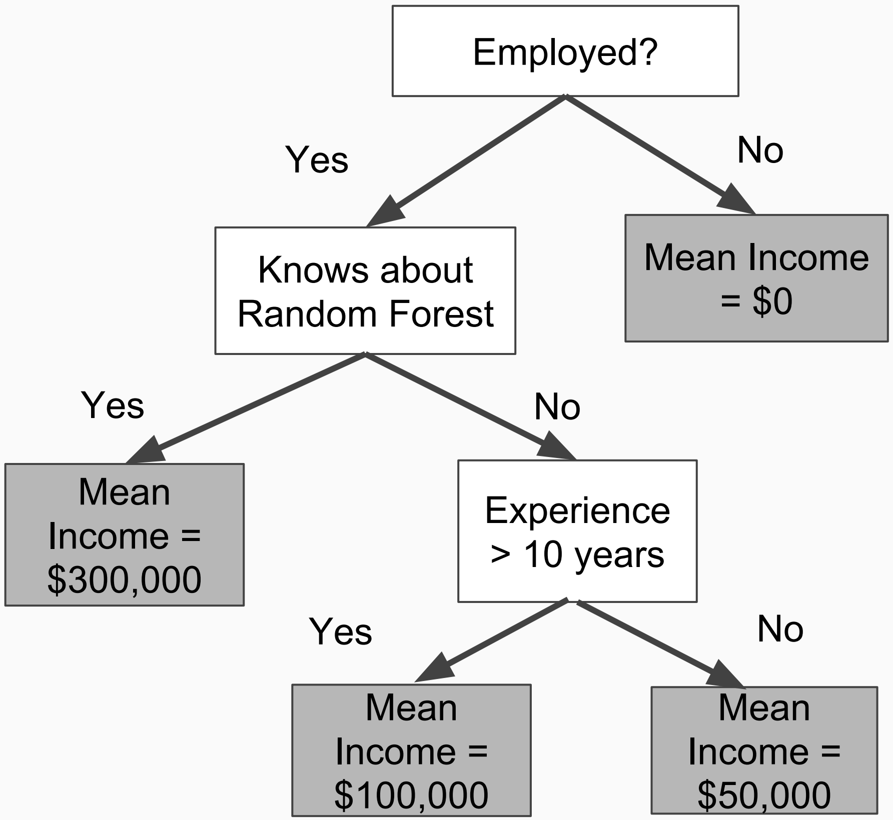
- Each split tries to minimize the Total Squared Error

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

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.

Target: Annual Income



Employed?

Yes

No

Knows about Random Forest

Mean Income = $0

Yes

No

Mean Income = $300,000

Experience > 10 years

Yes

No

Mean Income = $100,000

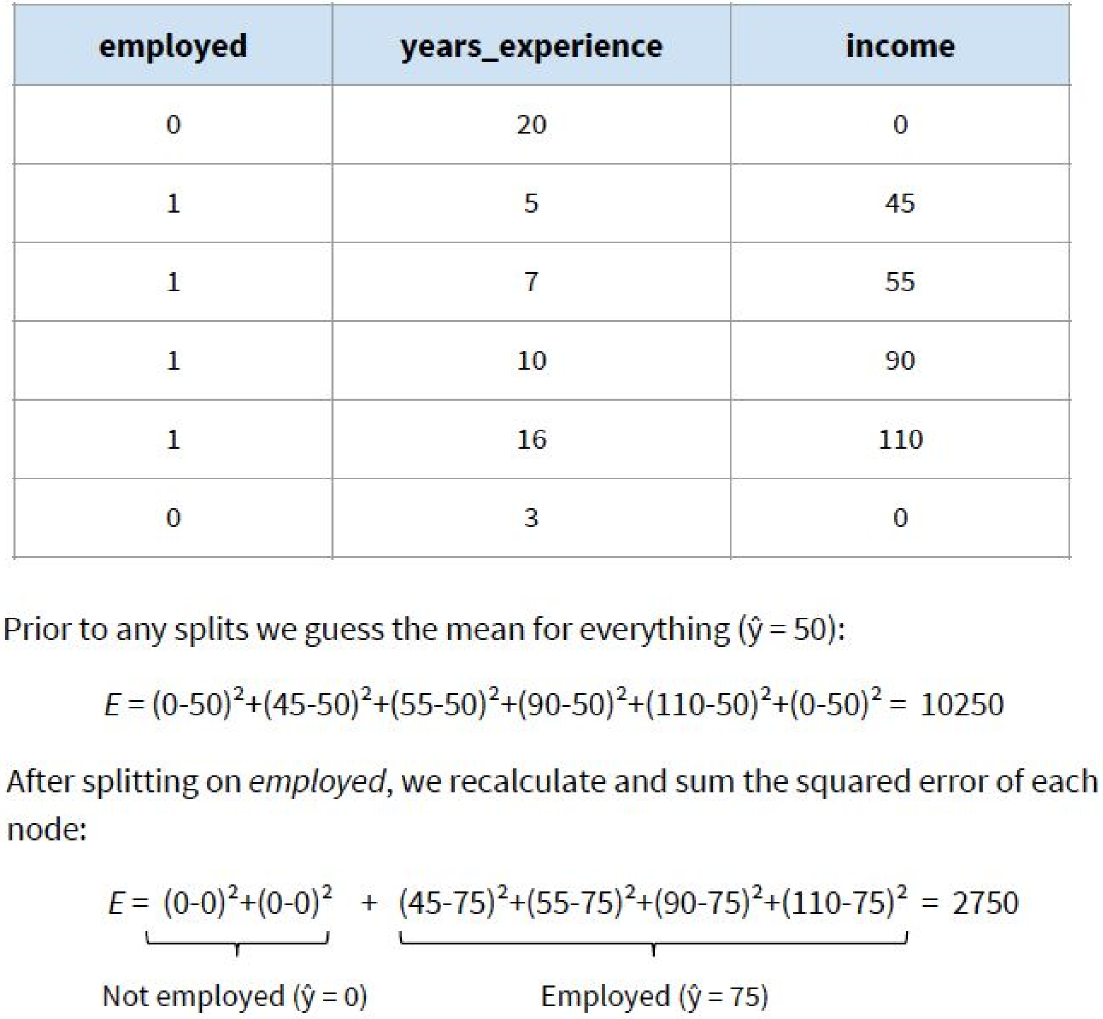
Mean Income = $50,000

Decision Trees: Regression

***

Same example as before, we are just ignoring the “Knows about Random Forest” variable

***



employed	years_experience	income
0	20	0
1	5	45
1	7	55
1	10	90
1	16	110
0	3	0

Prior to any splits we guess the mean for everything ($\hat{y} = 50$):

$$E = (0\text{-}50)^2+(45\text{-}50)^2+(55\text{-}50)^2+(90\text{-}50)^2+(110\text{-}50)^2+(0\text{-}50)^2 = 10250$$

After splitting on *employed*, we recalculate and sum the squared error of each node:

$$E = \underbrace{(0\text{-}0)^2+(0\text{-}0)^2}_{\text{Not employed } (\hat{y} = 0)} + \underbrace{(45\text{-}75)^2+(55\text{-}75)^2+(90\text{-}75)^2+(110\text{-}75)^2}_{\text{Employed } (\hat{y} = 75)} = 2750$$

What are some pros/cons associated with Decision Trees?

Discuss with the people at your table!

Pros and Cons of Decision Trees

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

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

Ensemble methods combine **multiple weak learners** to make **one strong learner** -- (think of “Ask the Audience” lifeline on Who Wants to Be a Millionaire)

1. Train multiple models (Decision Trees for today, but you can use others) on the data
2. Predict
    a. For regressors, take the average of the different models (doesn’t have to be a simple average)
    b. For classifiers, take the plurality choice or average the percentages of each class

Ensemble Methods

How might Ensemble Methods help with bias-variance tradeoff?

Ensemble Methods

How might Ensemble Methods help with bias-variance tradeoff?

Individual Decision Trees tend to:

- Overfit
- Exhibit high variance

When we aggregate their predictions, they tend to center around the true mean.

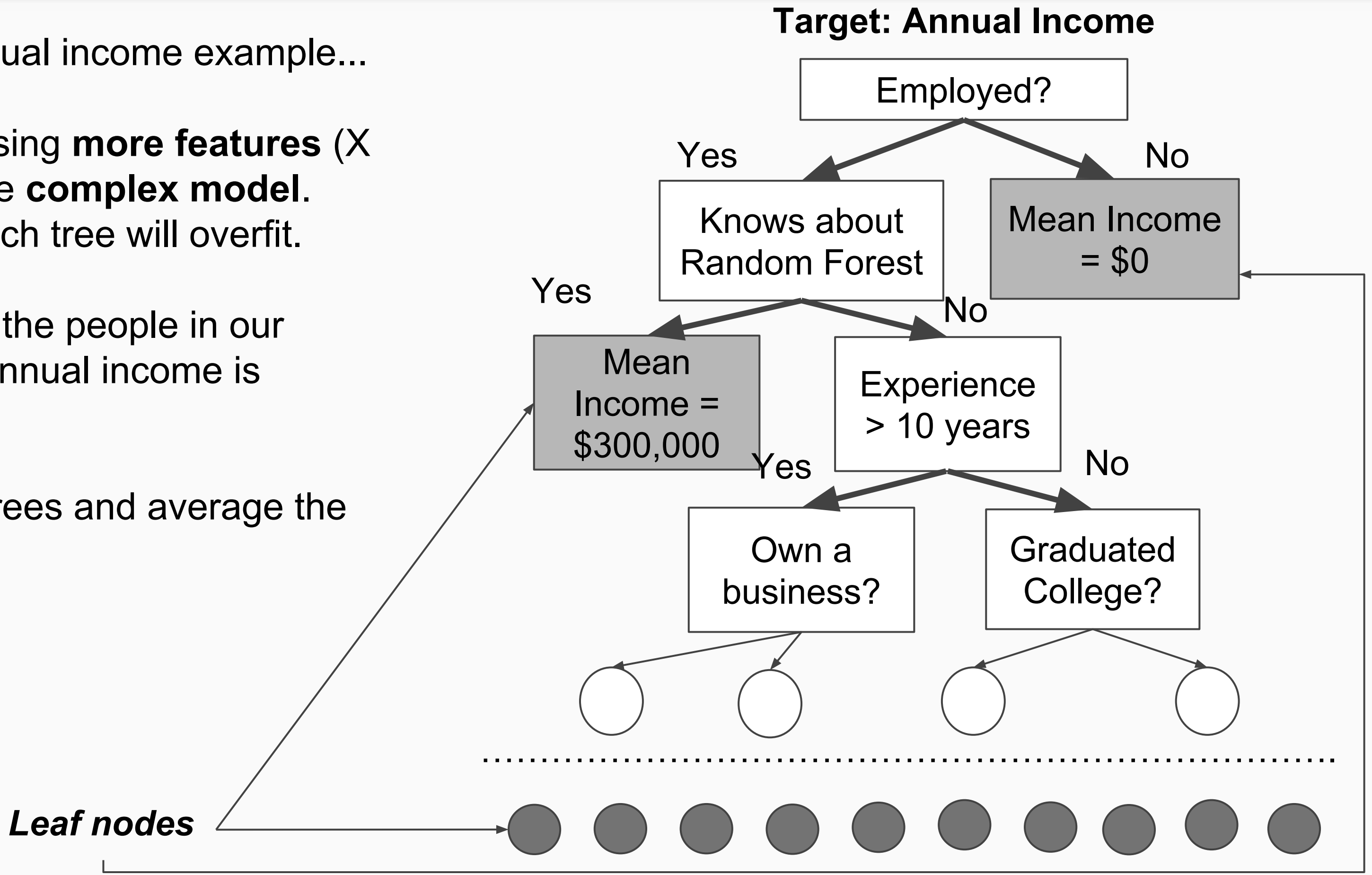
Ensembled Trees - Intuition

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.

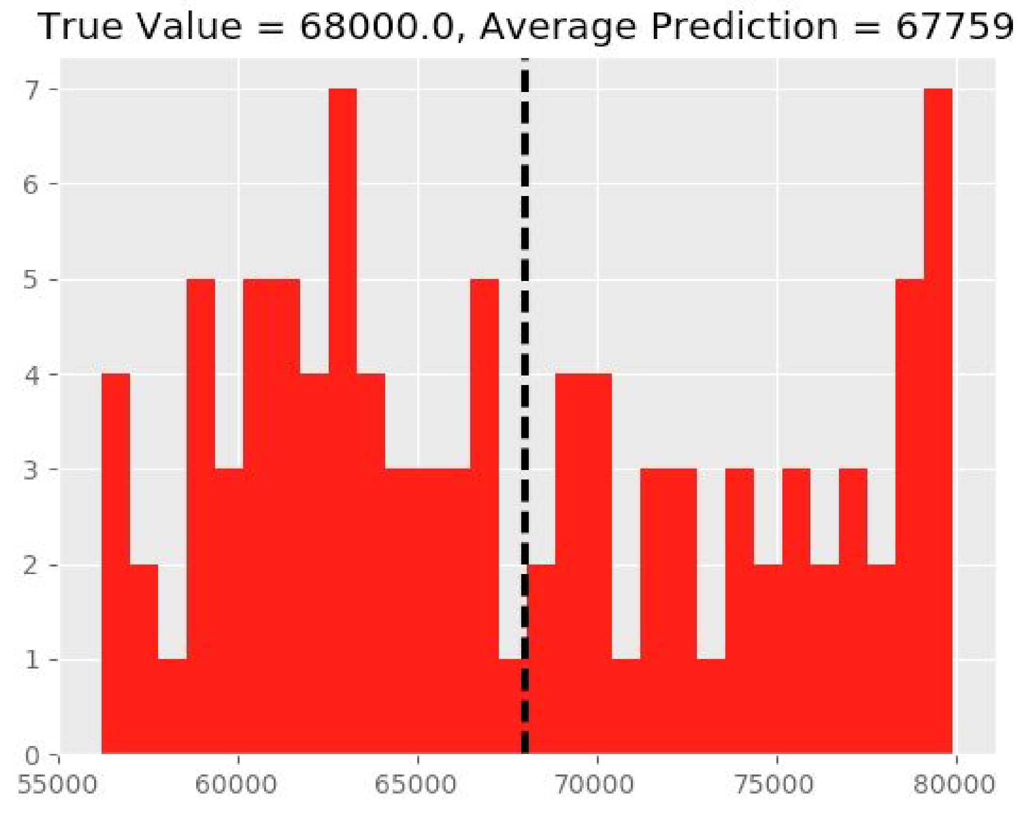


Ensembled Trees - Intuition

galvanize

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



This seems too easy, right?

Why won't this work? ***- take a minute to discuss***

(Hint: decision trees are deterministic)

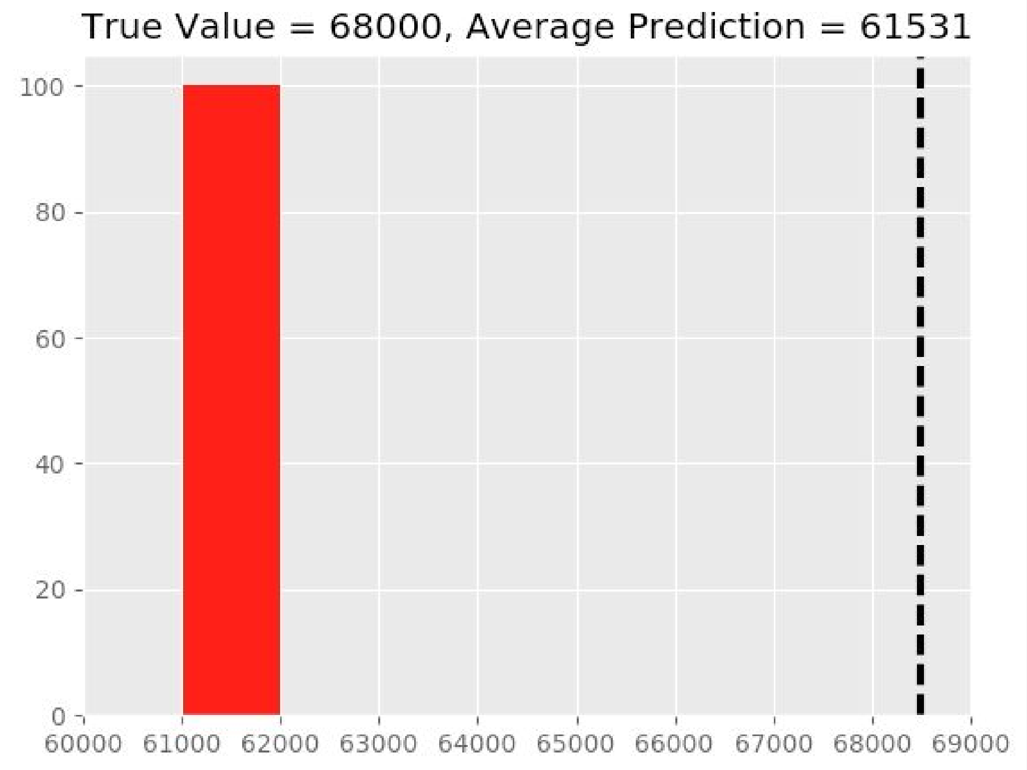
Ensembled Trees - Intuition

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

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

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)

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 -> Fit a model -> Repeat -> Aggregate

- Bagging is a general purpose method for reducing the variance of a statistical learning method.
- 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...

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!

Objectives

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

Random Forest

What is it?

Why is it better than simply bagging with decision trees?

Random Forest

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!**	▨	▨

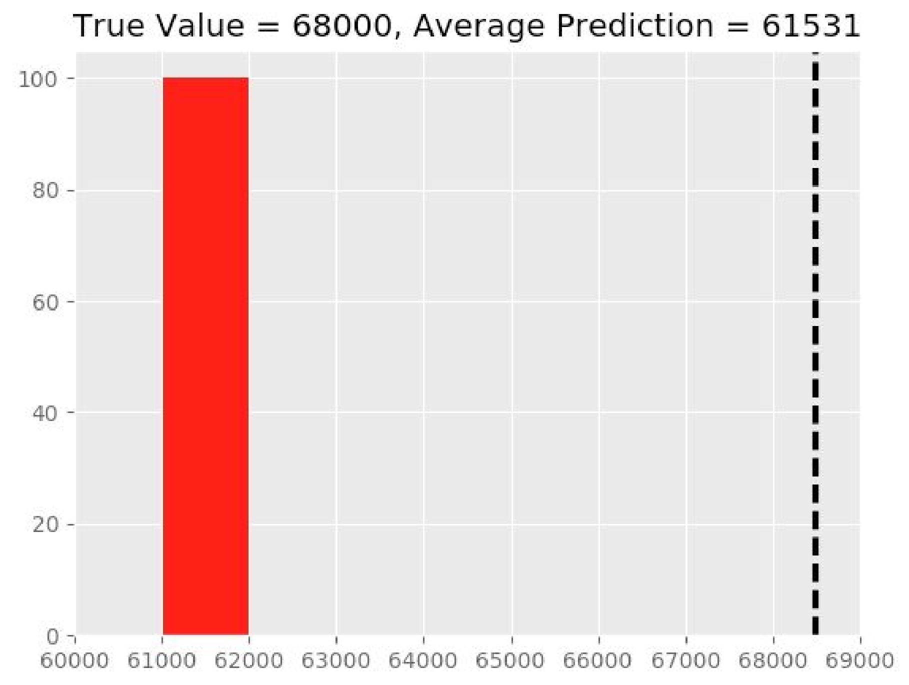
🟩 **Considered for split**	▨ Not considered for split	⬛ Can't be considered

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

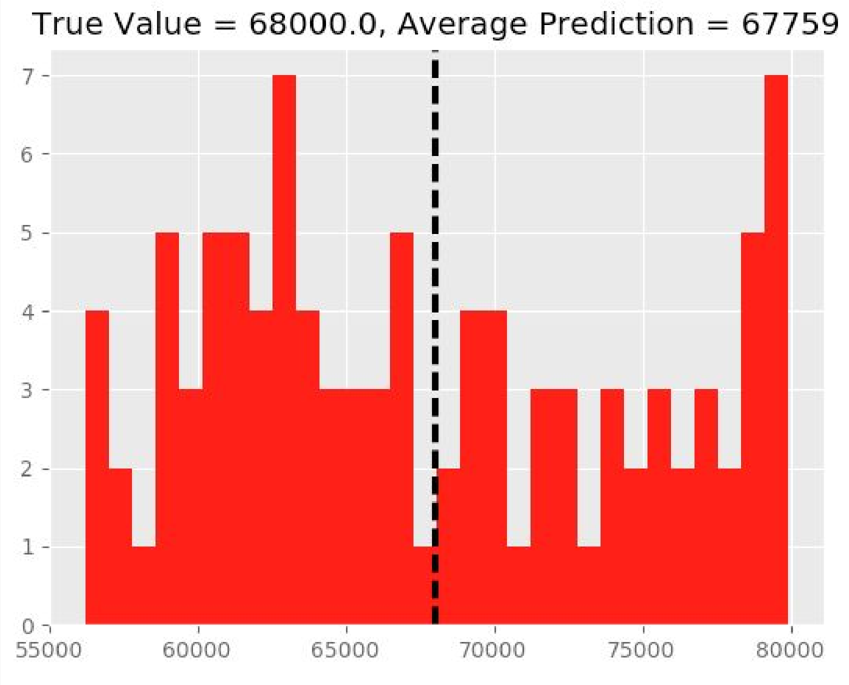
Why is decorrelating trees so important?

Recall from our previous example…

High Correlation Between Trees



Low Correlation Between Trees



Random Forest Hyperparameters

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

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)

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?

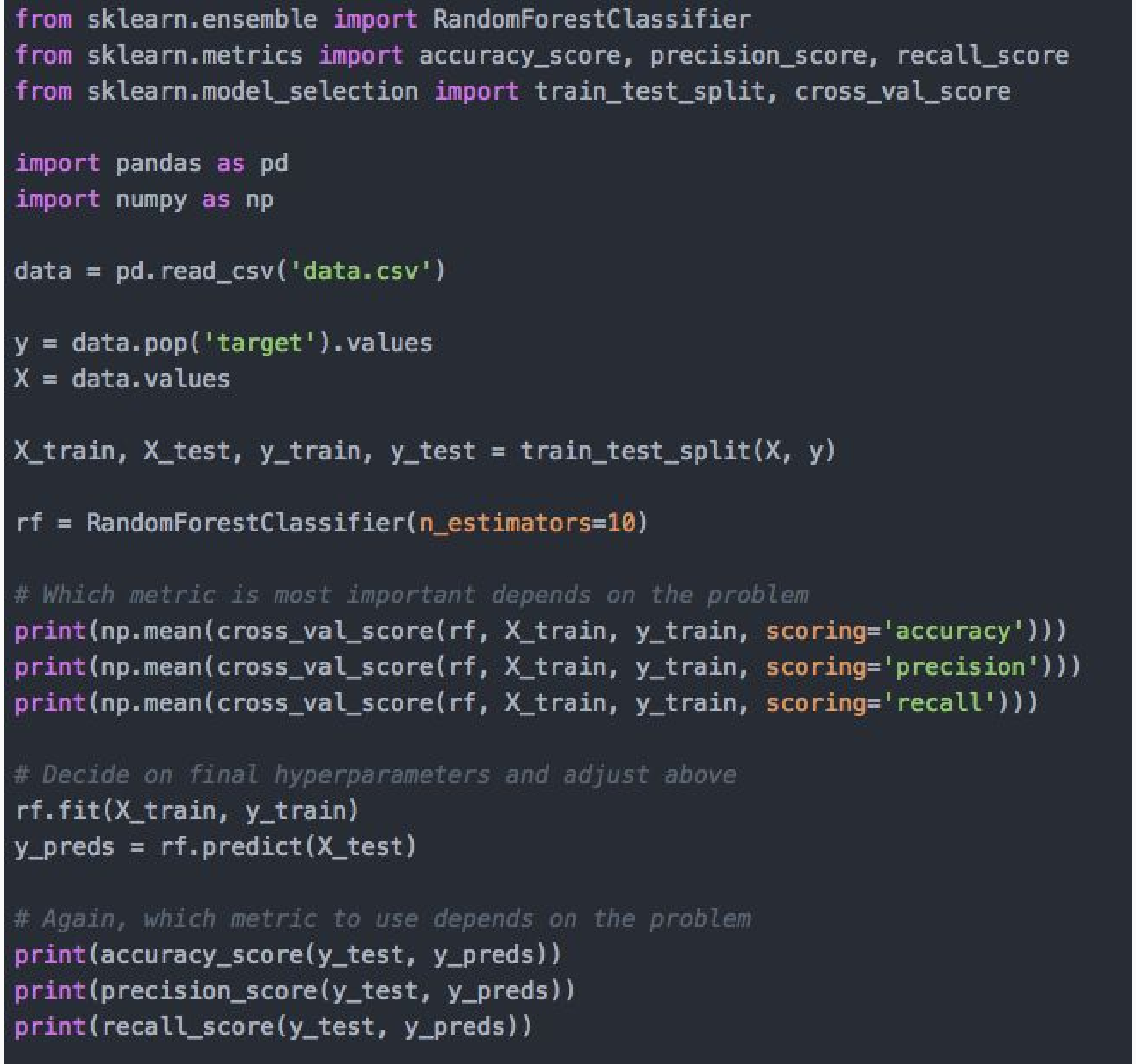
Our information gain criterion… Entropy, Gini for classification and RSS for regression.

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



```python
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))
```

Objectives

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:

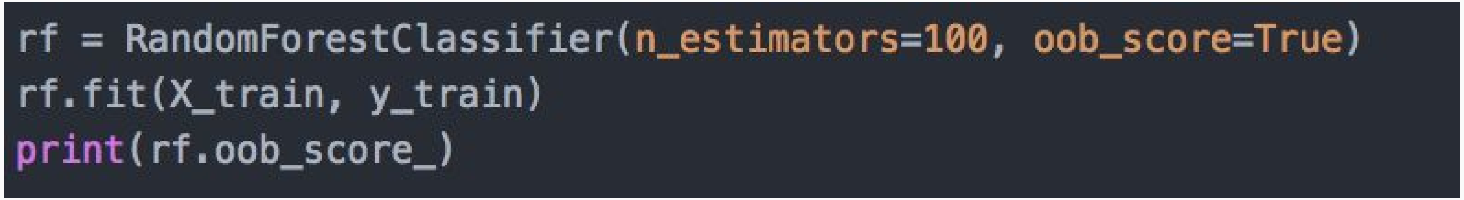
- Discuss **Out-of-Bag (OOB) Score** as a method for evaluating model performance
- Discuss **Feature Importance** for model interpretability

Out of Bag Score (OOB Score)

OOB Score is a quick and dirty “replacement” for cross validation

- We already have data that each tree has not seen yet -- each bootstrapped sample only includes about ⅔ of the data.
- We can feed the data that wasn’t used in a tree as a **test set** for that tree.
- We can then aggregate the *accuracy score** for each of our points (each test data point tested on ~⅓ of our trees)

The downside is that oob_score_ in sklearn only computes accuracy or R^2, so if we want precision, recall, or other metrics we will still need to cross validate :(



```
rf = RandomForestClassifier(n_estimators=100, oob_score=True)
rf.fit(X_train, y_train)
print(rf.oob_score_)
```

Objectives

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

Interpreting Trees, Bagged Trees and Forests

Recall, one of the main strengths of Decision Trees is interpretability.

However, when we aggregate our trees with simple Bagging or Random Forests, it’s not so easy…

- We can no longer simply rank our features in the order in which they were split on

But… We can look at Feature Importances (Note: nowhere near as reliable as coefficients for a linear regression)

Measuring Feature Importance

There are a several ways we can go about identifying important features:

1. Measure the total amount the information gain increases due to splits over a given feature
2. Record what portion of points pass through a single split -- the higher in a tree a feature is split on, the more important
3. **Combine 1 & 2 with rf.feature_importances_ (where rf is your fit RandomForestClassifier / RandomForestRegressor)**

More Ways to Feature Importance

1. **When tree B_i is grown score it with OOB, then remove that feature and score it again to measure the change in your validation metric(s) -- Leave One Out Feature Importances**
2. **Iterate through features dropping m_i out and plotting feature importances -- will identify “multicollinearity”**

Measuring Feature Importances

Let’s get some intuition for how we calculate feature importances…

1. For each feature m_j, we calculate the decrease in our impurity criterion (MSE, Gini, etc.) for the node(s) that split on m_j
2. We then weight it by how many points passed through the nodes that split on m_j
3. And finally, we average the calculations for steps 1 and 2 across our entire forest

Feature Importance Continued...

What does feature importance not tell us?

It is difficult to learn the effect size and direction of a feature… But this is the *price we pay* for a model that can handle *nonlinear relationships*.

For most real world problems, *features don't have a constant effect size* across all X-values, and sometimes the *effect direction can even reverse* at different levels of X.

Feature Importances

Let’s check out random_forest_baseball.ipynb for some ways to visualize feature importances...

Objectives

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