Video-2:

Notebook-link:

00:03 Diving deep into random forests and interpreting results

Get the best predictions -> deeply understand the data as well

Read Data

Understand Evaluation metrics
Replacing values with log values

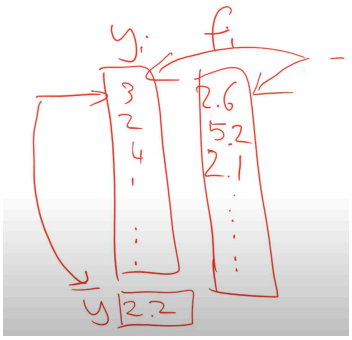
Parse_dates -> converting date columns into different columns

Convert categories into numbers

Missing values

11:00 R-square

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$



- The total sum of squares (proportional to the variance of the data):

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

- The regression sum of squares, also called the explained sum of squares:

$$SS_{\text{reg}} = \sum_i (f_i - \bar{y})^2,$$

- The sum of squares of residuals, also called the residual sum of squares:

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

fi -> predictions, yi -> actual values from the dataset, yi_bar -> mean

13:08 Understanding the intuition

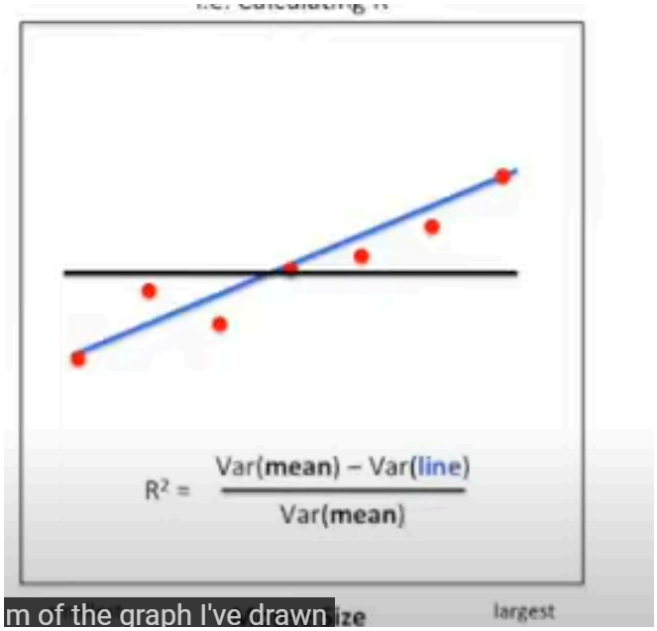
R-square values -> Anything less than one

If we predict mean exactly, SS_res = SS_tot, hence, R_square=0
If we predict exactly the values from the training set -> yi-fi = 0, R_square = 1-0=1

How good is the model compared to predicting its mean?

Creating some synthetic dataset with two dimensional data points to see what different values of R-square correlations look like?

The idea of figuring out the best machine learning model is to zero in on the model that best understands the distribution in the training dataset. Here, SS_tot is the variance of the distribution of the data. So, it represents how the data lies in comparison to its mean value.



18:35 Creating representative validation sets for machine learning models is crucial for production success.

Prevents overfitting. If the dataset has something to do with time with it, create a model trained on data with a dates range in the training dataset and validation set with a new range of dates. It is important that our validation dataset is representative.

Submit too often in kaggle-> we might overfit the leaderboard

20:47 Understanding and setting hyper parameters that work best?

If we just have one hold_out set, we might tune the hyperparameters that work on the specific hold out set. They just accidentally work on the given holdout_set, but doesn't actually understand the distribution.

2nd hold_out set -> only to use in the end to understand how good is the model actually

25:02 Proper separation of test, training, and validation sets is crucial to avoid overfitting

Taking a random validation and test sets to test our model against -> makes it much easier as it just cross references values to find the distribution

It is really important to have a representative validation and test sets -> In time series -> data from the new dates that the model did not see during training

31:51 Interactively analyze the best way possible

```
df_trn, y_trn = proc_df(df_raw, 'SalePrice', subset=30000)
X_train, _ = split_vals(df_trn, 20000)
y_train, _ = split_vals(y_trn, 20000)
```

```
m = RandomForestRegressor(n_jobs=-1)
%time m.fit(X_train, y_train)
print_score(m)
```

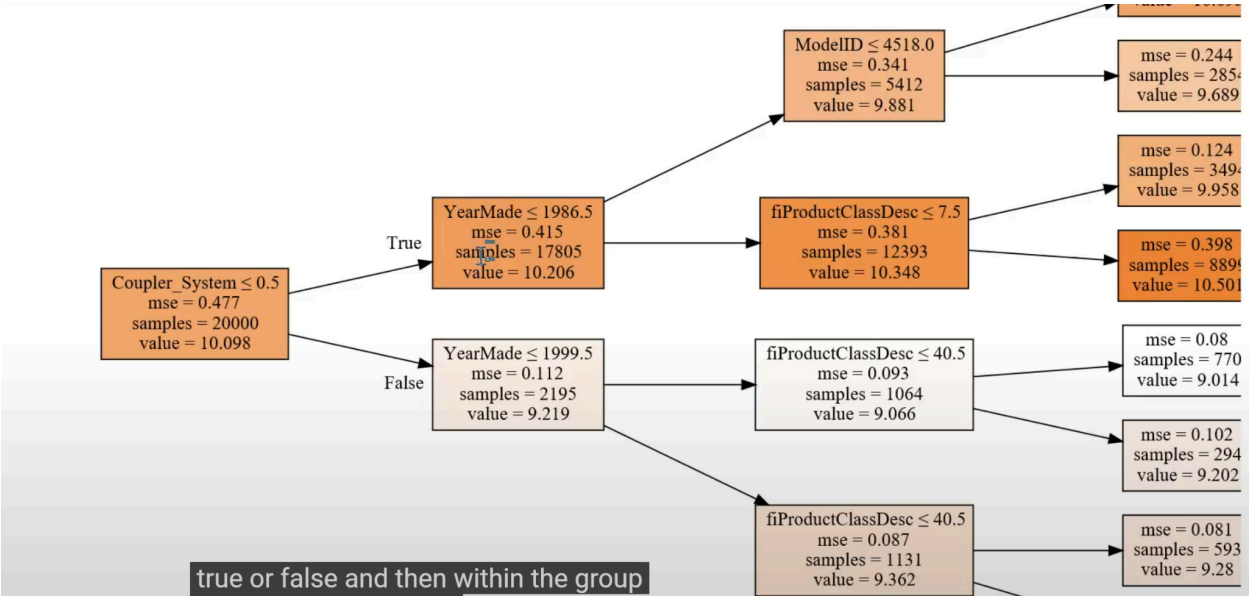
For interactively analyzing, it is important that we are working with small data. But, we need to get the results that work for and represent the entire dataset.

From a subset of 30000 randomly, first 20k values are training dataset

31:48 Building a simple model with a single tree

```
In [28]: m = RandomForestRegressor(n_estimators=1, max_depth=3, bootstrap=False, n_jobs=-1)
         m.fit(X_train, y_train)
         print_score(m)

[0.5376887554738916, 0.5782664994291248, 0.39338655904062148, 0.40382166952650589]
```



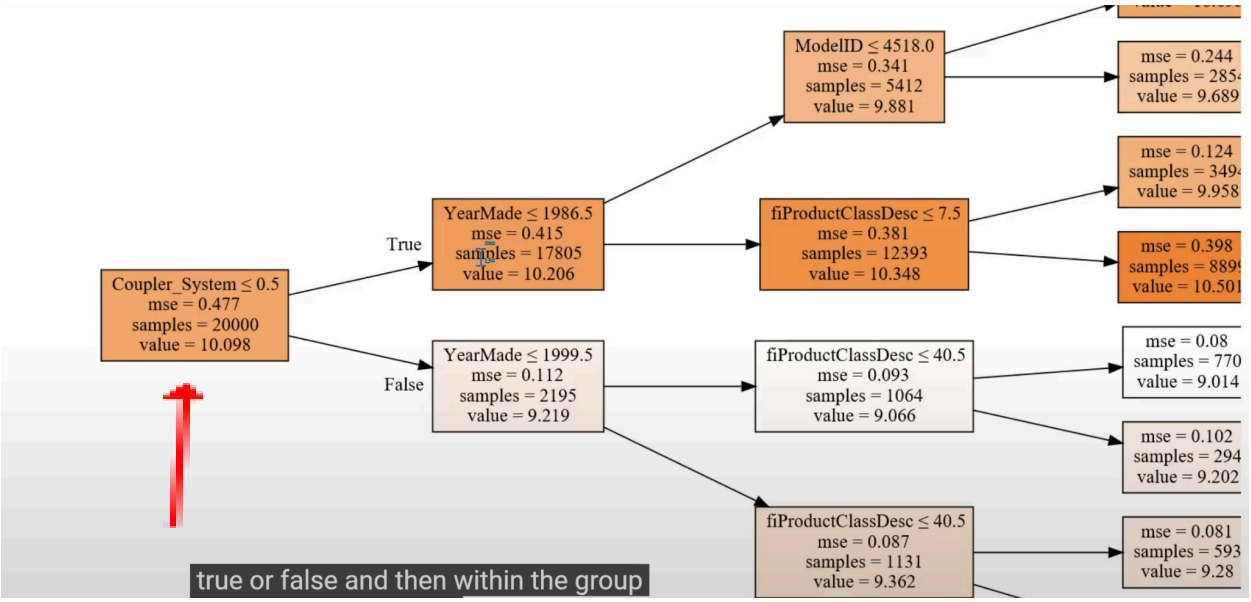
34:08 Discussing the impact of coloration true/false on error

Darker -> Higher value

38:55 Testing all possible splits and comparing RMSE for each variable

What is the best possible variable and split point that divides the entire dataset into two different groups?

Best way to create two completely different datasets?



What does mse and value mean? This means that, initially there is a variable **Coupler_system** which is boolean. There are 17805 samples with this value as true in the dataset. 2195 samples have it as False. The average of log of prices(target variable) is the value -> 10.098 and if we

build a model that just predicts this average all the time, mse=0.477, variance of the values with respect to the mean as seen in the denominator in the R-squared formula.

After we split using Coupler system, first grp -> mse=0.415, second_grp ->0.112
Observe the number of samples in both the groups too
Value -> the average log error value of all the samples we talked before

We can see that the second group has much less mse. But, it also has relatively very few values among the whole training dataset.

41:20 Explaining the concept of weighted average in machine learning

Let us just assume that, we decide the split point such that both the groups have the mse's as different as possible. The problem is, maybe we will single out a point that is very different from the mean value that is from a very old time and divide it into two values based on that value. The second group only has one single sample. This is a bad split as this nowhere represents the distribution of the data.

So, we can use the weighted average from the second stage.

1. **Calculate Group Errors**:
   - For samples $X \leq 5$: Compute $\text{MSE}(L)$.
   - For samples $X > 5$: Compute $\text{MSE}(R)$.
2. **Determine Group Sizes**:
   - Let $N_L = 6$ (number of samples where $X \leq 5$).
   - Let $N_R = 4$ (number of samples where $X > 5$).
3. **Compute Weighted Average MSE**:
   - Suppose $\text{MSE}(L) = 2.0$ and $\text{MSE}(R) = 3.0$.
   - Weighted average MSE:

$$E(S) = \frac{6}{10} \cdot 2.0 + \frac{4}{10} \cdot 3.0 = 0.6 \cdot 2.0 + 0.4 \cdot 3.0 = 1.2 + 1.2 = 2.4$$

So, even if a value is singled out, very less of the respective value's worth is actually considered on the formation of the tree. We use the variable and the splitting point combination that gives us the minimum overall weighted average MSE after going through all the sets of combinations. We do this iterative comparison of variables(we will also include the variables from the previous split) at every stage.

Why only binary split? - As we still have these variables later while iteratively testing out the variables and splitting points all the other possible combinations of the split are still considered somewhere along the branch. So, binary splitting is all we need!!

We either stop when we hit the limit of the maximum depth along the tree or when our leaf node has only one single sample.

Now, when we are checking the prediction value for a new set of features, these features are compared to the remaining splits in the tree and a value is assigned according to the branch it is assigned to.

46:14 Creating a forest using bagging technique to enhance decision tree model

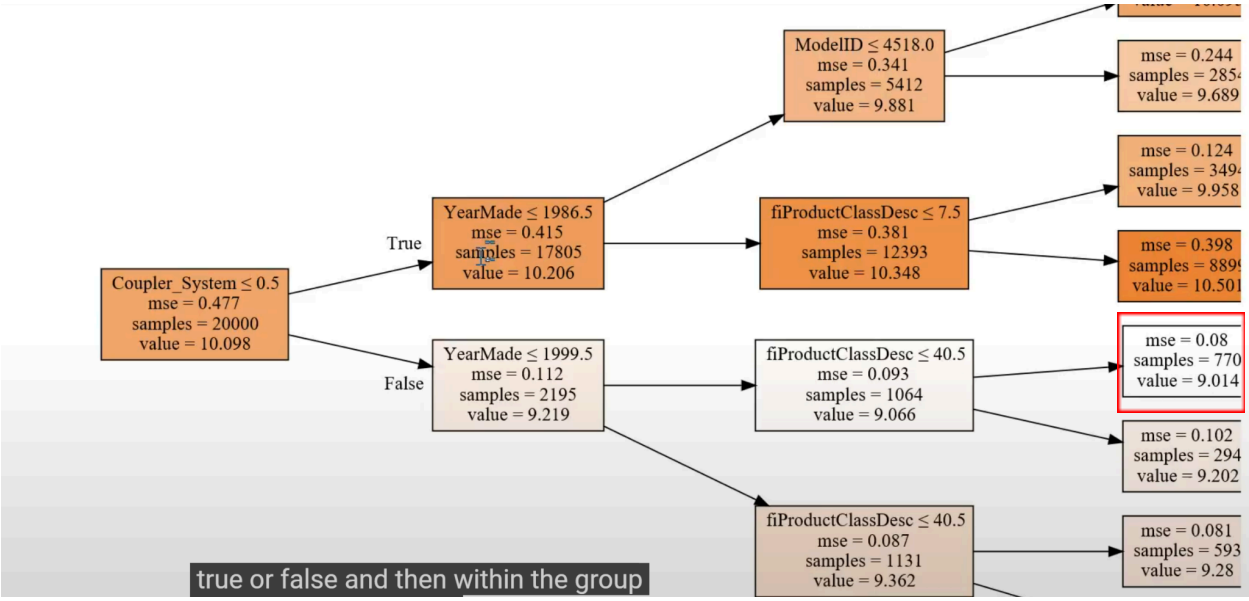
Bag of Little Bootstraps -> Using bagging for any kind of models to bring best and robust models

Five different models with no correlation with each other. This means that all the five models have different insights about the data. Averaging would give us much robust -> Ensembling

48:36 Using random subsets to build overfit trees and averaging their errors gives us the true relationship - random forest.

Create massive random trees that easily overfit and train them on a random subset of data and make them overfit that specific subset of data. -> Understands the properties of that specific subset, but not the rest. Have more trees with more random subsets of data. More trees are overfit on different samples of data in different ways. Average of these trees, we get a much closer approximation to the true relationship. The random subsets generation with replacement is called BOOTSTRAPPING. So, this means that many of those rows appear multiple times.

53:24 Decision trees identify similar data points



So, here we can see that the tree is grouping similar points that are closer to the respective values. For instance, in the above red box, it contains 770 values that are close and hence attributed the average value 9.014.It is important to understand that this way of understanding similarity between different data points operates across tree space and this offers us various advantages compared to operating on Euclidean space as these provide more insights about how different variables are interacting with each other to produce the aforesaid results.

55:24 Key to effective machine learning models is accuracy and generalization to new data.

In bagging, it is important that our individual trees in the random forest must be really predictive, but poorly correlated with each other. Recent results show that it is okay sometimes to have individual trees that are less predictive(less accurate), but should not have any correlation with each other.

Instead of every individual tree trying every split of every variable, trees just work with random splits of random variables. We just have to have more trees.

1:06:55 Adding more trees improves R-squared up to a point

One plausible hyperparameter. It does not hurt us to have more trees. But, after a time, it doesn't help anymore. But, it actually slows it down.

Jeremy tries his projects with 20 to 30 initially. Maybe, overnight leaves to train with 1000 trees.

1:10:00 Out-Of-Bag score

Different validation sets for different trees - For the first tree, as we are not using certain rows to train. We can use them as the validation set for the tree. So, for validating one sample of such a datapoint, we can use the average prediction of all the trees that did not use that specific sample and have that as prediction against the validation.

1:13:54 Grid search helps in finding the best hyperparameters.

1:16:12 Using subsets of data in Random Forest helps improve efficiency and accuracy

We have taken 30000 values and from the 30000 values, our random forests use different subsets of these 30000 values. We can take different subsets of 30000 each time instead. This gives the trees access to the entire dataset. If we have enough trees, eventually our model is going to see everything. Now increasing the estimators will make a bigger difference. So, NO DATASET IS TOO BIG FOR A RANDOM FOREST !!!

1:20:53 Focus on reasonable accuracy and quick training time for machine learning models.

1:23:03 Optimizing individual decision trees for better generalization

Min_samples leaf -> Stop training the tree when the leaf node has 3 samples instead of one. So, this means that when we are talking about an individual decision tree, instead of taking one value, we take the average of three data points. This decreases the accuracy of the individual tree a little, But, it is going to train faster with better generalization. Min_samples leaf values that worked better generally are 1,3,5,7,9,11

1:27:33 Using only certain features?

Imagine that there is an important column, then every tree always includes this column. So, individual trees are not uncorrelated and identical to an extent. So, we can take a random subset of columns. Max_features = 0.5 -> half of features. Values that worked good are: 1,0.5, log2, square root

1:29:48 Parameter tuning improves model accuracy significantly

1:34:36 Creating custom functions to analyze data -> next lesson