Lecture 1:
==========
While importing the dataset, parse date columns as good practice if there are any
parse_dates=["saledate"]

Get all the relevant parts from the date
add_datepart(df_raw, 'saledate')

Convert all string values to category for category related operations
train_cats(df_raw)
apply_cats(df_raw)

Many operations can be done using category features. Such as reordering in case of ordinal values.
df_raw.UsageBand.cat.categories
df_raw.UsageBand.cat.set_categories(['High', 'Medium', 'Low'], ordered=True, inplace=True)
df_raw.UsageBand = df_raw.UsageBand.cat.codes (pandas will continue displaying the text categories, while treating them as numerical data internally. Optionally, we can replace the text categories with numbers, which will make this variable non-categorical)

New feature. Store the dataframe on the disk for faster processing.
os.makedirs('tmp', exist_ok=True)
df_raw.to_feather('tmp/bulldozers-raw')
df_raw = pd.read_feather('tmp/bulldozers-raw')

Convert categorical to numeric, handle missing values
df, y = proc_df(df_raw, 'SalePrice')

Lecture 2
==========
R sqaure => carries the variance in the orginal and predicted results. Can have value anything less than 1.
Apply subset while using proc_df for faster processing
n_estimators = number of trees
bootstrap = randomization turn on/off for subsampling with replacement (sample size same as input size with duplicates)
A tree to split: The model tries to make two splits at each of the values in the dataset and tries to find the least error considering the weighted average of errors of both split groups to make that split actual one
Trees or estimators need to be least correlated to each other
All the trees need to be highly overfitted so that each of that understands the patten the best for the corresponding given input set
Later each of the trees are asked to predict for a new set (validation) of data and each of the trees make the best prediction as per their understood pattern.
Average of each of the trees are considered as a pedicted result.
out of bags: If bootstrap is true, there are some rows left out for each tree. Those are considered as validation set for the corresponding tree in random forest
If we want the base tree to use a sample of size less than that of the input data, then scikitlearn does not provide such option at present. set_rf_samples(n) provides an option to do it though. However with the oob_score does not work as of now. They are trying to make changes for that as well.

min_samples_leaf => number of samples at the leaf of which average is taken as predicted value. Good range (3,5, 10, 25)
max_features => denotes the % of features to be randomly selected for each base tree. Good range (1, 1.5, log2, sqrt). The main purpose is to make the trees less correlated.

Lecture 3
==========
Importing huge dataset in pandas has been slow. However there is a trick. Read first a few lines from the file and create a dictionary of data types of each of the columns. Then parse that dictionary while reading the whole file in pandas. No need to specify all the columns.
Use shuf on the command prompt to have a sample of the dataset to start with.
Internally RF converts the data type to float32. We can conver the dataframe to do the same beforehand to save sometime.
%prun will show the which step is taking how much time
Kaggle: Scatter plot between public leader board score and validation score for multiple models created. If that are linearly correlated with points close to the straight line then validation is a good set.
For each category in a column, check the mean and confidence interval (using standard deviation). According dicision to be taken which ones to be grouped if less confident.
rf_feat_importance is fast.ai function to return feature importance
After deleting less important features, run the feature importance score again to see the updated importance graph that matters more.
How feature importance is calculated: For each column the model suffuls the values of that column and check the magnitude of change in the rmse. Those magnitudes for each column are considered for feature importance calculation.

Lecture 4
==========
max_features: It considers random 50% of columns at each split in a tree. This will make the individual trees less correlated.
If the data points at the leaf nodes are identical in terms of the dependent variable then that node does not split.
Presence of correlated features will not end up with making them important for feature selection. The reason of when of those features is suffled, the presence of the other correlated feature does not make much changes in the dependent variable, hence the model does not designate the considered feature to be important enough.
Random forest is hard to overfit if not modeled using extremely bad hyperparameters.
In one hot encoding, we normally delete one of the encoded columns to break the linearity because that can be infered from the other columns. This is must in case of linear models. However this is not needed in RF because if it finds all the columns have zeros and does not know about the deleted column, then it cannot take a decision there for a split.
max_n_cat -> Any column having number of unique values less than this parameter will be converted to one-hot encoding
After the one-hot encoding, it is necessary to check the feature importance again with introduction of newly added columns.
One-hot encoding might destroy the properties of ordinal features hence it will necessary to to change those back to label encoding to see the model performance and accordingly make a decision.

Dendograms are highly useful to find which columns are similary hence it helps in deciding to delete redundant columns.
As an intuition, correlation is almost same as R square but in stead of between two variables it is between variables and predictions.
The problem with normal correlation is that it assumes linearity among the data which is not a good idea. The data of a feature might be random and the line cannot represent all the data points.
If we plot the ranks of values of a variable, that takes a shape of a line, hence it is great to have the rank correlation mapping between two variables to understand their correlation just in the same way as the random forest does. This rank correlation is called as spearmanr correlation.
For each of the redundant/higly correlated features, check the oob score to understand which are to keep or delete.
Partial dependence analysis can be done to identify seasonality, if present in the data.
Partial dependence can be very well used to derive insights and interpretations.
Tree enterpreter can be used to explain the predicted result to the audience. It can show which feature contributed how much to the predicted value.
Bias is the average value of the entire dataset.

Lecture 5
==========
Why OOB score is less than validation score: Each of OOB scores is the result of the corresponding base tree while the validation score is the result from the prediction of entire forest model
In case the data has temporal data, after the model is validated, the model needs to be trained again including the validation data to make predictions on the test data.
Why not create validation set with random sampling with temporal data -> Predicting into past does not make sense
A trick to try if that might work: Giving weights to each row that most of the ML algorithms support -> With times things change. Hence the most recent data (corresponding to timelines) has more credible information against the past data. Hence we might want to give higher probability weight to the recent data and lower weights to past data.
Validation scores vs test scores -> Create multiple models. For each of them plot the validation score vs teset score. If the relation is not linear then we will not get good feedback from the validation set to tune the model. In this case keep changing the validation set until we get the plot linear so that validation set is indicative of the test set.
Downside of cross-validation: As we understand, validation set has to be as much indicative of test set as possible, hence choosing a random validation set might not match the expectation. In cross-validation, the validation sets are pulled randomly hence if we tune our model to gain better accuracy in CV and ignore the relation between the predictions of CV and test sets then we might end up with bad result.
Tree interpretation: We can exploit the tree or even RF model to analyze the split points and observe how with the binary decision the average value of the output changes and accordingly create a waterfall model to explain.
Extrapolation: RF is not good at extrapolation unlike linear models. However there is a trick. Avoid using time as one of the predictors. Add a new column stating if that row is in validation set of not and try to predict the same. If can be predicted, then the validation sample is not random. Same trick goes for test set as well. After fitting the model to predict if validation set of not, find the strong varaibles. One by one try removing each of those time dependent variables from the data and predit the oginally considered dependent values. Depending which feature removals are actually contributing to the better model accuracy make the decision accordingly to remove those for the final prediction.

Lecture 6
==========
Confidence based on tree variance:

Lecture 7
==========
Figuring out most important interactions - There is a project/package for grandient boosting (XGBOOST) however in nothing as such available for random forest at present.
Number of observations in validation set - Number of observations for each category should be higher than 22. 22 is a magic number for T distribution to convert to normal distribution. It is not only for validation set, the rule applies to training set as well.
Depending upon the use cases: One shot prediction (For the model to see once and remember and predict... like in case of facial recognition) and zero shot prediction (model never saw the data and makes the prediction on)
Training data should be equal to number of observations for each class. If not then duplicate the entries to match each class of same size. RF also has a class weight parameter that decides to pick the rows of that class with higher probability during bootstraping.
For the piece of python code that is taking time, it might be a good idea to implement that in Cython.
For notebook, having nbextension is very helpful.

Lecture 8
==========
With very high cardinality and timeseries data, RF may not give the best result.
"Pickle" is a built in function to store any python object on the disk and retrive it anytime.
Random forest only cares about the order about the values.. not about the magnitude.. Hence it is immune to outliers in many cases and does not care about the scale of the variables. Similarly Area under curve, pearson correlation only care about the sort order of the values, not on the magnitude.
But in deep learning is very much parameterized (not necessarily but mattens in tree based algos). Hence we need to normalize the data.

Lecture 9
==========
http://structuringtheunstructured.blogspot.com/
https://medium.com/mlreview/parfit-hyper-parameter-optimization-77253e7e175e
https://towardsdatascience.com/how-to-make-sgd-classifier-perform-as-well-as-logistic-regression-using-parfit-cc10bca2d3c4
https://www.kaggle.com/devm2024/keras-model-for-beginners-0-210-on-lb-eda-r-d
explore the library tensorly
matrixmultiplication.xyz
In neural network, what backpropogation does is.. it multiplies the delivatives of each of the functions in the chain to get the final derivative at that step

Lecture 10

==========

If all the weight parameters are considered (not being equal to zero) that might lead to overfitting. The idea is to use only those weights that are useful. Hence regularization is used to penalize the non-zero weights. In l2 regularization the square of the average weights is added in the loss function that would in turn try to minimize that towards zero. As a matter of fact, it is not about the loss function rather the gradient of it that matters the most, hence the gradient is calculated and that gives the updated weights as the previous weights minus learning rate times the gradient descent. That is called weight decay.