Fun Fact

- About 37% of the original data does not end up in the bootstrap sample
- You need to have more data points then you have parameters, otherwise you don't have enough degrees of freedom to make any sort of model

Overfit Model

- One that is too complicated for your data set. When this happens, the model becomes tailored to fit the quirks and random noise in your specific sample rather than reflecting the overall population. If you drew another sample, it would have its own quirks, and your original overfit model would not likely fit the new data.
- Instead, we want our model to approximate the true model for the entire population. Our model should not only fit the current sample, but new samples too.

Clusters

- The expected error from randomness is expected to be lower when using cluster sampling when most of the variance in the population is within cluster, opposed to across clusters - that way each cluster is representative of the population.
- When between block variance is high but within block variance low, stratification greatly decreases variance of ATE estimate.
- When ICC is high then variance of estimator is increased because clusters are not representative of the population

Linear models

- Four major assumptions and how to check for them
    - 1. Linearity: look at scatter plots
    - 2. Homoskedacticity - equal variance of errors: look at plot of residuals
    - 3. Independence of residuals
    - 4. Normality of residuals

Logistic regression

- Similar to Multiple regression, except here we are predicting a binary outcome.
- So we are looking for the probability of being either of the two outcomes given the data
- Logistic regression is based on MLE and not least squares, thus no p-values and no R squared values and thus difficult to compare models
- Generally you'd perform some sort of analysis of deviance to select the best logistic model.
- You could also use a likelihood ratio test to determine whether adding a particular coefficient is beneficial
- So it is essentially a normal linear regression if you spread out the y-axis based on its log odds, with an intercept and coefficients and standard errors of those estimates.
- Logistic regression is the same as linear regression except the coefficients are in terms of the log odds
- You can use categorical and numeric data in logistic regression

Regularization in Logistic Regression

- https://www.reddit.com/r/datascience/comments/8kne2r/different_coefficients_scikitlearn_vs_statsmodels/dz90uen/
- Without regularization, logistic regression won't converge if the data is completely separable.
- For the simple case with two classes, the beta value will continuously increase, as this simultaneously increases the log-likelihood of data in class 1 being assigned to class 1 and

data in class 0 being assigned to class 0. Therefore scikit-learn's loss function would find it optimal to continuously increase beta, and you would never converge to a finite solution.

ANOVA Model

- Used when you have multiple categories and trying to determine whether there is a significant difference among them.
- You essentially try to ratio the signal over the noise. So variance <u>between</u> divided by variance <u>within</u>, and under the null, this should be centered at one. This is the F statistic.

Missing Data in Random Forests

- There are many ways of dealing with missing data
    - Essentially you're going to use existing data to fill in the most common values or medians values as placeholders for all the missing data
    - Then you're going to see which data samples the rows with data are most similar to and then correct the placeholder values with more statistically predicted values based on weights and running data through trees to determine similarity

Ada Boost

- A stump is a tree with one root and two leaves
- The goal of Ada Boost is to use the variable information of an observation to predict some unknown response value for that observation
- Ada Boost uses a series of stumps to make that decision. Some stumps have more influence in the final decision than other stumps. Whereas in random forests, each tree has an equal vote
- Order of the stumps matters in Ada Boost. I.e. the stumps are not independent of each other. The errors that the first stump makes, influences how the second stump is made, etc.

<u>Steps</u>

1. You give each observation a weight that indicates how important it is for it to be correctly classified. In the beginning, all weights are equal and sum to 1.
2. You decide which variable to use for the first stump using the normal decision tree methods, i.e. Gini values
3. To decide how much say that stump has in the final classification we use the sum of the weights associated with the incorrectly classified observations
    a. Amount of say for each stump = 0.5*ln(1 - Sum of Incorrect Weights / Sum of incorrect Weights)
    b. Sum of weights will always be between 0 and 1
    c. If sum of weights is close to 1, then amount of say will be a large negative number
    d. If sum of weights is close to 0, then amount of say will be a large positive number
    e. If sum of weights is 0.5, then amount of say will be 0. Intuitively this is because the stump is apparently no better than flipping a coin.
    f. If amount of say is negative, then it's impact in prediction is negative, as in the stump is so bad that if you use it backwards it is better for prediction
4. Now you adjust the weights of the observations, increasing the weights for incorrectly classified observations and correspondingly decreasing the weights for the correctly classified observations
    a. New weight of incorrectly classified observations = observation weight * e^(amount of say)

            i. So when amount of say of stump is high, then observation weight will increase substantially
                1. Intuitively this is because this is a “good” stump, so any mistakes it made ought to be highlighted significantly for the next stump to be better
            ii. When amount of say of stump is low, then observation weight will increase less substantially
    b. New weight of correctly classified observations = observation weight * e^-(amount of say)
        i. When Amount of Say of the stump is large then new observation weight is substantially smaller
        ii. When Amount of Say of the stump is small then new observation weight is only a little bit smaller
    c. Then you normalize the observation weights of the new observations, both correctly classified and incorrectly classified, so they add up to 1
5. Now you make the second stump using the new weights and thus weighted gini indexes. The stump can be based on the same variable as the first stump, if that variable is again the one that best predicts the response (lowest weighted gini)
    a. Basically, if a new stump does a poor job of compensating for the previous stump’s errors, then it will have a reduced amount of say
6. You can make as many stumps as you want, say 100, or stop when you get a perfect fit
7. Once you’ve made all the stumps then you add up the Amount of Say for all the stumps that classify an observation as “yes” and the Amount of Say for all the stumps that classify an observation as “no”, and pick the greater of those two values as the ultimate classification.

Gradient Boosting

- Gradient Boost starts by making a single leaf, instead of a tree or stump. It represents an initial guess of the response variable for the observations. The leaf value starts with the average value of the response for continuous data, or the log(odds) for binary data.
- Then gradient boost builds a tree, which, similar to ada boost, is based on the errors made by the previous tree. But it is usually larger than a stump. Gradient Boost scales all trees by the same “learning rate”.
- Training successive trees on the gradient of the error with respect to the loss predictions of the previous tree, teaches the tree to correct the mistakes of the previous tree. This is the core of gradient boosting, and what allows many simple trees to compensate for each other’s weaknesses to better fit the data.
- Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. A popular example is the AdaBoost algorithm that weights data points that are hard to predict. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models. This approach supports both regression and classification predictive modeling problems.

Gradient Boosting for Regression (Continuous Response)

Steps

1. Calculate the average response value for the entire dataset

    a. This is the first attempt at predicting the response for all observations
2. Calculate the residuals of each observation in the dataset to this average
    a. These are called pseudo-residuals
3. Now we build a tree the usual way using the variables to predict the <u>pseudo-residual</u> values for each observation.
    a. Generally, you limit the tree to be only 8 to 32 leaves
4. When you get to the leaves, instead of classification status, as you do with a categorical response, you'll put the predicted pseudo-residual number, since this is numeric data.
    a. There will be multiple values in some leaves, since there should be more observations than leaves, so take the average of these values as the prediction value for those leaves.
5. Now, to make a prediction, you run the observation characteristics through the tree and find its predicted residual value. Then you multiply this value by the "learning rate", say 10%.
6. Then you <u>add</u> this value to the initial leaf value (the dataset-wide response average) to get the new predicted value for that observation
7. Then you do this for all other observations in the dataset and get a new predicted value for each observation
8. Then you calculate the new pseudo-residual values for each observation
9. Then you build a new tree trying to model these new residual values
10. Then you run the observations through this new tree and get a new predicted residual value for each observation
11. Then you multiply these values by the same "learning rate" of 10% and add it to the previous iteration's "new residual".
12. And you keep doing this until eventually you converge towards better predicted value than just having one overfit tree

Gradient Boosting for Categorical Response

Steps:
1. We are assuming a binary response
2. Similar to Gradient Boosting for Regression, we start with an initial leaf, which is an initial prediction for each observation
    a. We use the initial Log(Odds) of "success" given the entire data set reponses to get the initial value for this leaf
3. Then we apply the inverse logit function to get the probability value (the probability of "success"). For the initial value this should just be a simple proportion of "successes" from the dataset
4. If this probability value is above 0.5, then the initial prediction is that everyone in the dataset will display the "Success" outcome.
5. Similar to how we started with the average response and found the residuals, we will start with this predicted probability and take the residuals.
    a. Since the observed values are either 0 or 1 the residuals will either be one of two numbers
6. Now you build a tree to predict the residuals
7. The predictions in your leaves are in terms of probability of "success", not in terms of logg(Odds), as is our initial leaf. So we have to transform the values in these leaves.
8. There is a semi-complicated formula for this transformation, but essentially you add the residual values in the leaf, if there are multiple values there, and then divide by the bernouli variance of the previous iteration's predicted probability for each of those observations, if there are multiple observations

9. Then you run your observations through the tree and get your predictions for each observation on the log(Odds) scale
10. Then you multiply these values by some “learning rate”
11. Then you add these values to the initial leaf value
12. Then you transform these values back into probability form
13. Then you find the new residuals
14. And you repeat the process
15. And you continue until you have your desired number of trees
16. To make predictions you run your observation through your final tree

XGBoost:

- e**X**treme **G**radient **B**oosting
- XGBoosting is a lot faster than normal Boosting methods
    - Systematically
        - It uses parallelization to speed up loops
        - Pruning criterion is optimized
        - More efficient use of Cache
    - Algorithmically
        - It penalizes models through Ridge and Lasso regularization to prevent over fitting
        - Better equipped to handle sparsity in data
        - Cross-validates automatically?
- XGBoot dominates structured or tabular datasets on classification and regression predictive modeling problems.
- Whereas Neural Nets still dominates in the more unstructured space of prediction

Pros vs. Cons of different Tree Methods

Tree:

- Easy to visualize
- Easy to interpret
- Over-fitting can be addressed with pruning

Forest:

- Strong Predictive power but lower interpretability than a tree
- Generally 2 Parameters to tune
    - Number of TRees
    - Number of features to be selected at each node
- More hyperparameters
    - Number of trees
    - Sampling rate
    - Number of variables to try

Boosted Trees:

- Trees Depend on each other
- Generally 3 parameters
    - Number of Trees
    - Depth of Trees
    - Learning Rate
- Even harder to interpret
- Follows a Boosted model, vs. a bagged model

    - Uses residuals of previous tree
- Even more hyper-parameters than forests
    - Learning Rate
    - Ridge
    - Lasso

Principal Component Analysis
- Reduction of high dimensional analysis to two axes
- Starts by calculating the correlations amoung the samples
- Then does eigen decomposition
    - Out of this we get coordinates for a graph
    - Percent of variation each axis accounts for
    - And loading scores to determine which variables have the largest effect
- Those data clustered together are more similar

Multi-Dimensional Scaling
- Does the same thing as above except starts by calculating distances amoung samples
- Those data clustered together are more similar

Euclidean Distance
- Distance from one data point to another in dimensional space
- There are limitations to calculating Euclidean distance when the data have covariates
- So we can try to re-scale the data such that the variables no longer had any covariance through Mahalanobis Distance

Mahalanobis Distance
- Essentially, you use the eigenvectors of the data as the new axes, and re-scale the data such that the covariances are removed and you can now use Euclidean distance to analyze the data
- This method gives you distance between points after accounting for covariance of the data

Cross Validation
- Allows us to compare various machine learning methods and get a sense of how well they will work in practice
- 10-Fold cross validation
    - You use the first 90% of data to train and the last 10% to test
    - Then you use the first 80% and last 10% ot train and the second to last 10% to test
    - Etc.
    - You aggregate how well each algorithm performed given the 10 test sets for each, and pick the best performing one
- Cross-validation doesn't work in situations where you can't shuffle your data, most notably in time-series.

Training
- Estimating the parameters of the model/algorithm
- Use like 60-75% of the data to train
    - Bootstrapping tend to give us models which are trained with 63% of the data

Testing

- Evaluating the algorithm/model
- Need to use new data to test the algorithm to test for overfitting
- Reusing the same data for both training and testing is a bad idea because we need to know how the model will work for data it wasn't trained on

Validation
- Set of data that is used during training to tune our hyperparameters

Bias-Variance Tradeoff
- Bias
    - Inability for a machine learning algorithm to capture the true relationship in the data
- Variance
    - Difference in fits between data-sets
- So with low bias but high variance, then your model could be really accurate, but it will have high variance and be really bad
- But with high bias and low variance, you'll have an okay model but will be more likely to perform okay more often opposed to poorly.
- Ideally your model has low bias and low variability, you'd have to find the sweet spot in the tradeoff through regularization, boosting, and bagging.

Ridge Regression
- The main idea is to introduce a new line that doesn't necessarily fit the data as well (so we add a little bias) but has a lower variance.
    - I.e. the line does a better job at predicting new data but fits the training data "sub-optimally"
- When Ridge-regression determines parameters for intercept and coefficient betas, it minimizes the sum of squares + lambda*(slope^2)
- For multivariable it become sum of squares + lambda*(coefficient1^2 + coefficient2^2 + coefficient3^2 …+ etc.) (note: y-intercept not included)
- As lambda approaches infinity, the line approaches y bar
- To choose a value for lambda, you use 10-fold cross validation
- Ridge regression essentially makes the dependent variable less influenced by the independent variables
- The whole point of doing ridge regression is because small sample sizes can lead to poor least squares estimates, which then lead to terrible machine learning predictions
- If you don't have enough data points for all the parameters, then you can use ridge regression

Lasso Regression
- Really similar to ridge regression except it does a little better when there are some useless variables in the model. It drives them to zero faster.
- Ridge does better when most of the variables are actually useful
- When Lasso-regression determines parameters for intercept and coefficient betas, it minimizes the sum of squares + lambda*|slope|

False Positive
- Type I error
    - Reject the null when null is true

False Negative

- Type II error
    - Fail to accept alternative when alternative is true
- Depends on the situation and your constraints but a false negative is worse if the stakes are higher. If you fail to predict something bad or consequential then that's worse than predicting something bad or consequential but doesn't actually materialize

Regularization

- What you do to prevent overfitting

- **Feature Selection**
    - Forward selection
    - Backward selection
    - Stepwise regression

- **Dimensionality**
    - improves performance by reducing number of variables to be considered

- **How to deal with unsymmetrical data**
    - You can take the log for right skewed data
    - You could use ranks
    - You could use aggregate statistics which in the long run can be analyzed using the CLT

- **KNN Model**
    - This is a classification which classifies the "type" of new data by determining the most common type of a particular number (k) of the closest data points. Generally you have to determine the optimal value for k by looking at the accuracy of the prediction for a particular train/test split (using cross validation).

- **Recursion**
    - When a function calls itself to complete an action. Common example is the factorial function.

- **Machine Learning**
    - Giving computers the ability to make decisions from data without being explicitly programmed to.

- **Supervised Learning**
    - When there are labels present, we call that supervised learning. If outcome variable is categorical, then it's a classification problem. If it's a numerical outcome variable, then it's a regression problem.

- **Unsupervised Learning**
    - When there are no labels present, we call that unsupervised learning. Clustering. And reinforcement learning: how to optimize behaviour given system of rewards and punishment.

- **Central Limit Theorem**
    - The mean of some metric (X-bar) for a reasonably sized sample of a population will be normally distributed with mean unbiased for the population mean, and variance equal to the population variance divided by n, the size of the sample.

- **Z-score**
    - When we know the standard deviation of the population (sigma), we can compute a Z-score. This is where we take a sample, find the average (X-bar), subtract the population mean, and divide by (sigma^2 / n). This value can then be compared to the z-table to determine whether the sample average of a population is significantly different from the population mean. Typically this means the z-score was > 1.96 in magnitude for a 2-sided test.
    - The Z-score is compared to a Z-distribution, which is a standard normal distribution.

- **T-distribution**
    - Also called the Student T's distribution, this distribution approaches the Z-distribution as the degrees of freedom increase. Basically, when you know the population variance, then you can use the Z-distribution. When you're using 's' as a proxy for the population standard deviation, then you use the t-distribution to account for that increased uncertainty.
    - It's used to estimate population parameters when the population variance is unknown or when the sample size is small.

- **Test for difference in mean**
    - So you have some population, and you're trying to figure out if a sample of people have a different average height than the population.
    - The null is that the mean of the sample is the same as the mean of the population. The alternative is that it is different.
    - Now we know that if the means are the same (the null hypothesis), then the sample's average minus the population average will be centered at 0, and that the sample average's standard deviation will be sigma/sqrt(n) because we're measuring the sample average (X-bar) not just one random data point.
    - So when we divide that difference by that expected standard deviation under the null, we are basically trying to figure out how many standard deviations away that difference is from where it is supposed to be.
    - If you know the population standard deviation you use the standard normal distribution as the null distribution. So if the difference is > 1.96 in magnitude, then we reject the null and conclude that there is a significant difference in height between the sample group and the population.
    - If you don't know the population standard deviation, you use the t-distribution as the null distribution. Thus the magnitude values will be greater than 1.96 for small degrees of freedom, but will converge towards 1.96 with larger n.

- **T-statistic**
    - This

- **T-test**
    - Assumptions:
        - Data must be continuous
        - Data must be normally distributed
        - Samples independent, of equal variance, and randomly smaples
        - If not equal variance then we diminish number of degrees of freedom of the t-distribution (so higher threshold needed for significance.
    - Purpose

- **One sample test**

- **Two sample test**

- **Paired t-test**
    - So you equivalently you could just use linear regression with an indicator for being in one or the other group. More usefull when variances of the two groups are super different?

- **Statistical Power**
    - d

- **Naive Bayes Classifyer**
    - This is an algorithm by which you classify data as belonging to a particular category. It is called naive because it assumes that features are independent of one another. For example, if a fruit is red, round, and 3 inches in diameter it is classified as an apple. However naive Bayes assumes that these features are independent of one another, even though they might depend on one another or on other external features. Thus it is a very simplistic algorithm. The actual algorithm itself is just Bayes rule. It assigns the posterior probability of being in a certain group given certain feature values via bayes theorum using probabilities in a simple frequency table. But assuming independence of features is almost never a fair assumption without data to back it up.

DS General: Add PCA youtube lessons, gini splitting criterion, tree pruning from 149, regression trees from 149, and https://stats.stackexchange.com/questions/69442/linear-regression-and-non-invertibility and https://www.sagepub.com/sites/default/files/upm-binaries/21121_Chapter_15.pdf, http://cs229.stanford.edu/notes/cs229-notes-all/cs229-notes10.pdf, bagging, Random forests, ensemble methods: combining multiple models.

http://setosa.io/ev/principal-component-analysis/

- **PCA**
- **In detail how to pick the penalization term value in ridge/lasso regression through cross validation**
- **Gradient Descent**
- **How does Logistic regression determine p-values?**
- **Tradeoffs between Tree Based Models and regression models**
- **Tradeoffs among different types of tree based models**
- **Cross validation in minute detail and its function**
- **Building logistic regression from scratch**
- **Polynomial model**
- **Cubic function**
- **Mixed effects model**
- **How does the TukeyHSD calculate these confidence intervals?**
- **How to control for regression to the mean in experiment design?**
- **You'd think you can compare models just by running models and then seeing which one gives you the best R squared but ultimately it's p-hacking**
    - So put aside another validation set so you can make sure the best model wasn't a type one error

- **Within a block/strata, is it more efficient in terms of variance to split the strata down the middle in terms of treatment?**
- **Is matched pairs design better than general stratification?**
- **Neural Networks**
- **Akaike's Information Criterion**
- **Bayes Information Criterion**
- **Adjusted R-Squared**
    - It's like R-Squared except it is adjusted to reflect the number of parameters used in the model
- **Support Vector Machines**
- **Dealing with unbalanced datasets**

https://shapeofdata.wordpress.com/introduction/

In machine learning, a hyperparameter is a parameter whose value is set before the learning process begins.