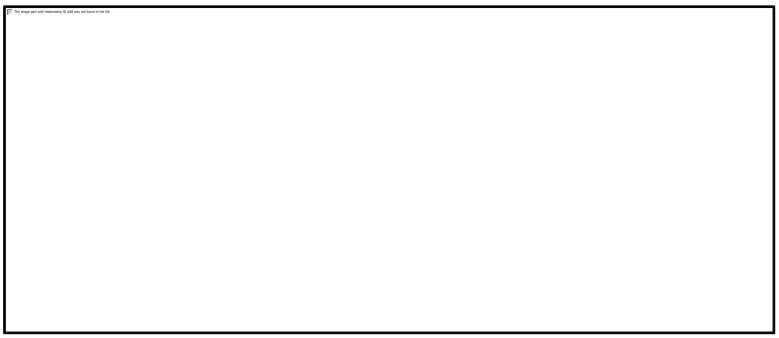
- **Ensemble Models**
  - Ensemble literally means a group. Ensemble modelling is a process where multiple diverse models are created to predict an outcome, either by using many different modelling algorithms or using different training data sets.
  - There are 4 types which are Bagging, Boosting, Stacking, Cascading
  - More different are the models, better we can combine them
- **Bootstrapped Aggregation (Bagging)**



  - Given a standard training set D of size n, bagging generates m new training sets D_i, each of size n, by sampling from D uniformly and with replacement.
  - By sampling with replacement, some observations may be repeated in each D_i,. If $n' = n$, then for large n the set D_i is expected to have the fraction $\left(1 - \frac{1}{e}\right) \approx 63.2\%$ of the unique examples of D, the rest being duplicates. This kind of sample is known as a bootstrap sample.
  - Then, m models are fitted using the above m bootstrap samples and combined by averaging the output (for regression) or voting (for classification).
  - If we change the dataset e.g., we have removed 100 points. As we have implemented bootstrap strategy, only Some D subsets will be changing and ultimately corresponding M models.
  - Since we are aggregating the results i.e., taking average or majority vote, overall results will not be changed.
  - Thus, due to bagging, variance will be reduced i.e., the model doesn't change much with change in data.
  - Suppose, base models M_i has low bias and high variance. With bagging this M_i models will have model with low bias and reduced variance model.
  - A good example of low bias and high variance model will be a decision tree of a reasonable depth
- **Random Forest**

$$RF = Decision\ Tree\ (reasonable\ depth) + row\ sampling\ with\ replacement + column\ sampling + aggregation\ (Majority\ vote\ or\ Mean)$$

  - The random forest is a classification algorithm consisting of many decision trees
  - In RF, we perform column sampling without replacement in addition to row sampling i.e., bootstrap sampling.

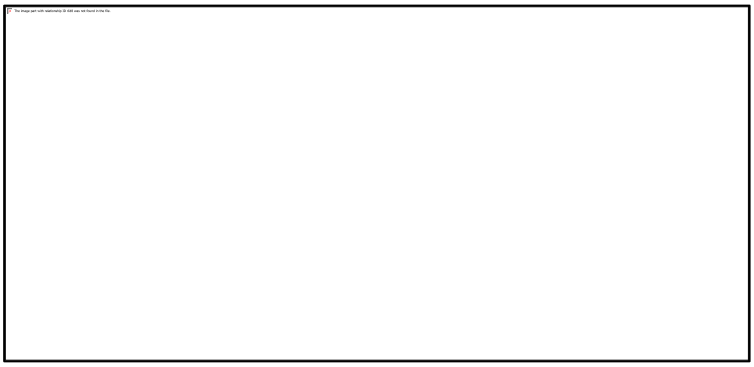
$$\boxed{\left(D_n \middle| {d \atop n}\right) \rightarrow \left(D' \middle| {d' \atop m}\right) \mid {d' < d \atop m < n}}$$

  - Bootstrap sampling = Row sampling with replacement: Create k samples;
  - Column Sampling = select a subset of features randomly
  - For a dataset D_i used for training model M_i.
    - $D_n - D_i$ points are called as out of bag points (OOB points). This OOB points can be used as cross validation dataset for same model M_i
    - Majority of packages when trained by RF gives us OOB error too
- **Bias-Variance Trade off**
  - RF has low bias as each of base models will also be low bias models
  - Bagging is a great way to reduce variance i.e., as no of base models increase, variance will decrease and vice-versa.

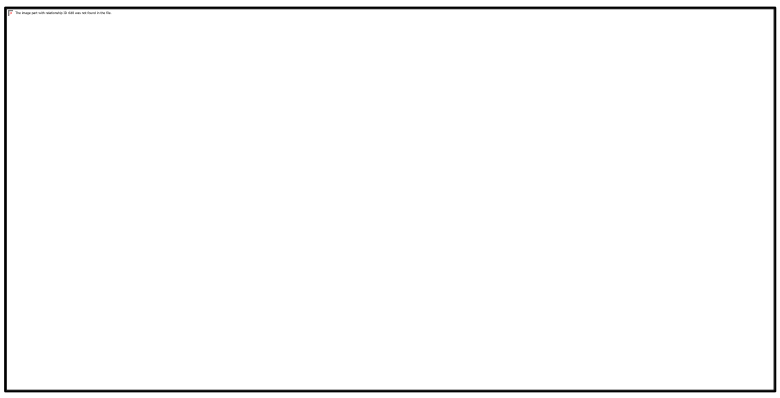
$$\boxed{\begin{array}{l}\left(D_n \middle| {d \atop n}\right) \rightarrow \left(D' \middle| {d' \atop m}\right) \\ col\ samp\ rate = \frac{d'}{d} \\ row\ samp\ rate = \frac{m}{n}\end{array}}$$ as col sampling rate & row sampling rate decreases variance in model also decreases

  - The most important hyperparameter will be no of base models(k)

- **Extremely randomized trees**
  - In Random Forest models, we perform **Row Sampling + Column Sampling + Aggregation.**
  - Whereas in Extremely randomized trees, we perform **Row Sampling + Column Sampling + Aggregation + Randomization in selecting T(tau).**
  - In case of Random Forest models, we are reducing the variance while keeping the bias same using a majority vote on the base learners built using column and row samplings.
  - Whereas in extremely randomized trees, in addition to majority vote, we are also trying to reduce the variance further on each base learner by adding one more level of randomization in splitting the features.
  - In **random forest model**, this splitting is performed by making each and every value as a threshold and computing the information gain. Whichever value gives the maximum information gain, at that particular value the split is performed. All the values less than this threshold will be on the left side and all the values greater than this threshold will be on the right side.
  - In case of **extremely randomized trees,** the same split operation for a feature is performed by picking some values of the feature randomly and computing the information gain only with those randomly selected values, instead of all the values. Due to this randomization, the variance in the model is reduced to some more extent when compared to the Random Forest models.
  - An Intuitive Explanation of Random Forest and Extra Trees Classifiers | by Frank Ceballos | Towards Data Science
- **Boosting**



  - Bagging: High variance and low bias models: with randomization and aggregation to reduce variance
  - Boosting: take low variance and high bias models: use additive combining to reduce bias
  - The N models are trained sequentially, taking into account the success of the previous model and increasing the weights of the data that this previous model has had the highest error on, which makes the subsequent models focus on the most difficult data observations.
  - Also, the individual models that perform the best on the weighted training samples, will become stronger (get a higher weight), and therefore have a higher impact on the final prediction.
    1. All the data samples start with the same weights. These samples are used to train an individual model (a Decision Tree let's say).
    2. The prediction error for each sample is calculated, increasing the weights of those samples which have had a greater error, to make them more important for the training of following individual model.
    3. Depending on how well this individual model did on its predictions, it gets assigned a. A model that outputs very good predictions will have a high amount of say in the final decision.
    4. The weighted data is passed on to the posterior model, and 2) and 3) are repeated.
    5. Number 4) is repeated until we have reached a certain number of models or until the error is below a certain threshold.



  - Given a dataset:

- Stage 0: Train a model using the whole dataset. This model should have high bias and low variance. E.g., DT with shallow depth; Large Train error = $y_i - \hat{y}_i$
- Stage 1: In stage 1, we train a model on error of the previous model

$$\boxed{F_1(x) = \propto_0 h_0(x) + \propto_1 h_1(x)}$$

$\boxed{F_k(x) = SUM[\propto_i h_i(x)]}$ additive weighted model

- Each of the models at each stage is trained to fit on the residual error at the end of the previous stage

- **Residuals, Loss functions and gradients**
  - In some cases, boosting models are trained with a specific fixed weight for each learner (learning rate) and instead of giving each sample an individual weight, the models are trained trying to predict the differences between the previous predictions on the samples and the real values of the objective variable. This difference is residuals.
  - Here we will be finding a function $\boxed{F_k(x) = \sum[\propto_i h_i(x)]}$
  - Let the problem be linear regression. The loss function will be

$$\boxed{\begin{gathered} L(y_i, F_k(x)) = (y_i - F_k(x))^2 \\ let\ F_k(x) = \mathbb{Z}_i \\ L(y_i, \mathbb{Z}_i) = (y_i - \mathbb{Z}_i)^2 \end{gathered}}$$

$$\boxed{\begin{aligned} \frac{\partial L}{\partial \mathbb{Z}_i} &= (y_i - \mathbb{Z}_i)^2 \\ &= -1 * 2(y_i - \mathbb{Z}_i) \end{aligned}}$$

$$\boxed{\underbrace{-\frac{\partial L}{\partial F_k(x)}}_{negative\ derivative} = \underbrace{2(y_i - F_k(x))}_{residual}}$$

  - Negative gradient of loss function at the end of stage k is proportional to residual at that stage; negative gradient $\propto$ pseudo residual (this allows to use any loss functions)
  - So, during boosting, we will replace the error with pseudo-residual i.e., negative residual, as it allows us to minimize with any loss function if that function is differentiable.
  - Random Forest are limited with loss functions, Gradient Boosted DTs permit use of any loss functions
  - https://youtu.be/qEZvOS2caCg
  - https://youtu.be/1o0yd6eMmA0
  - For non-squared loss functions: Is pseudo residual equal to negative gradient?
  - Log loss: negative gradient = probability estimate – actual class
  - But pseudo residuals cannot be interpreted as residual / error always
- **Gradient Boosting**
  - Input: training set $\boxed{(x_i, y_i)_1^n}$ a differentiable loss function $\boxed{L(y, F(x))}$ number of iterations M.
  - Initialize model with a constant value:

$$\boxed{F_0(x) = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma) |\ \gamma\ is\ a\ predicted\ value}$$

  - Here we are trying to reduce the loss between actual y & predicted γ . this loss could be any loss function.
  - Let us consider linear regression loss function.

$$\boxed{Loss = \sum_{i=1}^{n} \frac{1}{2}(y_i - \gamma)^2}$$

Experience	Degree	Salary
2	BE	50k
3	PhD	70k
4	ME	60k

E.g.,

$$\begin{gathered} Loss = \frac{1}{2}(50-\gamma)^2 + \frac{1}{2}(70-\gamma)^2 + \frac{1}{2}(60-\gamma)^2 \\ \downarrow First\ Order\ Derivative \\ = \frac{2}{2}(50-\gamma)(-1) + \frac{2}{2}(70-\gamma)(-1) + \frac{2}{2}(60-\gamma)(-1) \\ = -50 + \gamma - 70 + \gamma - 60 + \gamma \\ = 3\gamma - 180 \\ \gamma = 60\ (Initial\ Value\ for\ base\ model) \end{gathered}$$

  - For m = 1 to M:
    - Compute pseudo-residuals

$$r_{im} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}_{F(x)=F_{\{m-1\}(x)}} \quad for\ i = 1, \ldots, n$$

- Pseudo-residual is derivative of y and loss function from previous iteration with respect to loss function from previous iterations

$$Loss = \frac{1}{2}(y_i - \gamma)^2$$
$$\downarrow First\ Order\ Derivative$$
$$\frac{\partial h}{\partial \gamma} = \frac{2}{2}(y_i - \gamma)(-1)$$
$$= -y_i + \gamma$$
$$= -(y_i - \gamma)$$
$$-\frac{\partial h}{\partial \gamma} = (y_i - \gamma)$$
$$-\frac{\partial h}{\partial \gamma} = pseudo\ residue$$

Calculating pseudo-residuals

Experience	Degree	Salary	Predicted value	r
2	BE	50k	60	-10
3	PhD	70k	60	10
4	ME	60k	60	0

- Fit a base learner (or weak learner, e.g., tree) $h_m(x)$ to pseudo-residuals, i.e., train it using the training set $(x_i, r_i)_1^n$
- Compute multiplier γ_m by solving the following one-dimensional optimization problem

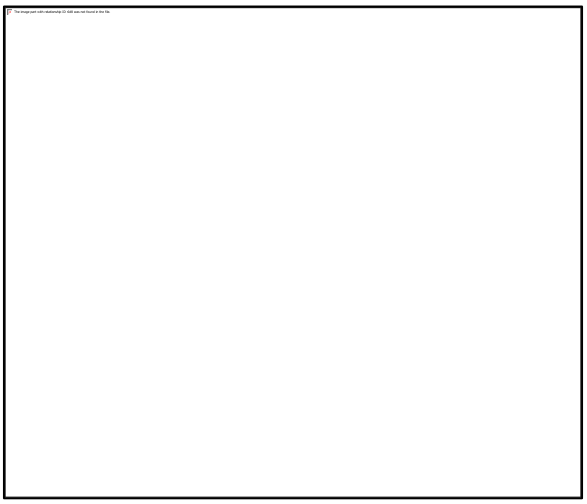
$$\gamma_m = argmin_\gamma \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma)$$

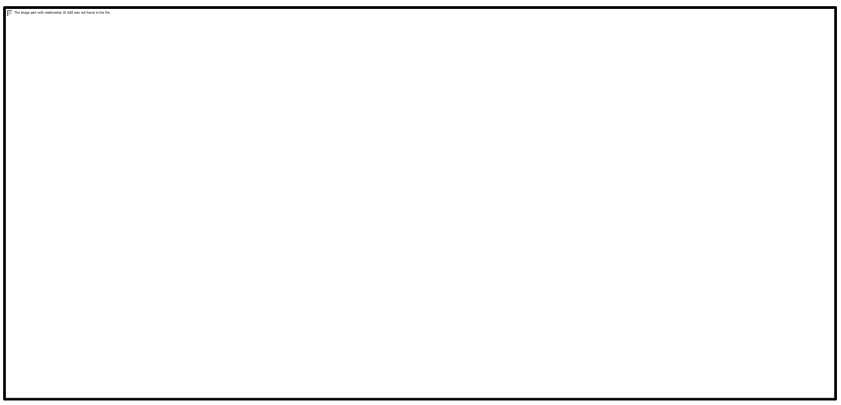
- Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

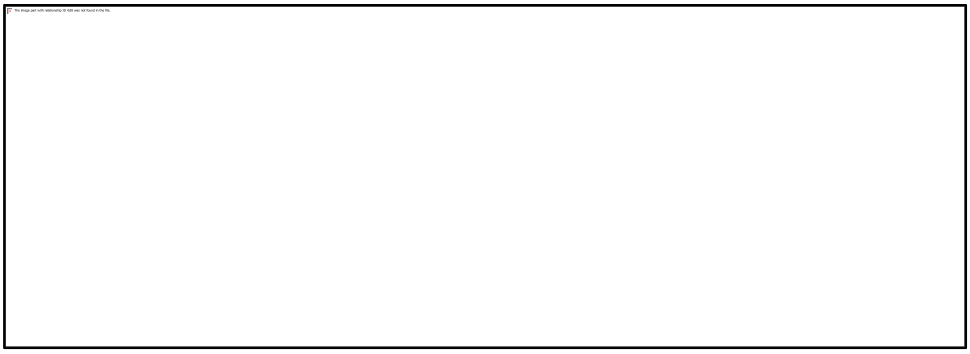
- Repeat the same procedure again

- **Stacking Classifiers**
  - http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/#stackingclassifier
  - Stacking is an ensemble learning technique to combine multiple classification models via a meta-classifier.
  - The individual classification models are trained based on the complete training set; then, the meta-classifier is fitted based on the outputs -- meta-features -- of the individual classification models in the ensemble.
  - The meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble.
  - Stacking is least used on real world problems due to its poor latency performance





- **Cascading Classifiers**



- Cascading is basically “a process whereby something, typically information or knowledge, is successively passed on”.
- Cascading is one of the most powerful ensemble learning algorithms which is used s when they want to be absolutely dead sure about the accuracy of a result, e.g., if a credit card transaction is fraudulent or not.