**Table of Contents**

[Bias vs variance: 3](#_Toc199227806)

[1. Feature Engineering: 5](#_Toc199227807)

[**1.1. Feature transformation:** 5](#_Toc199227808)

[1.1.1. Outlier Detection: 5](#_Toc199227809)

[1.1.1.1. Z-Score method: 5](#_Toc199227810)

[1.1.2. Missing value Imputation: 6](#_Toc199227811)

[1.1.2.1. KNN imputation: 6](#_Toc199227812)

[1.1.2.2. Multivariate Imputer: 7](#_Toc199227813)

[Column Transformer and Pipelines: 8](#_Toc199227814)

[1.1.3. Encoding: 9](#_Toc199227815)

[1.1.3.1. OHE: 9](#_Toc199227816)

[1.1.3.2. Categorical encoding 9](#_Toc199227817)

[1.1.3.3. Label encoding 9](#_Toc199227818)

[1.1.3.4. Label encoding 9](#_Toc199227819)

[1.1.3.5. Target encoding: 9](#_Toc199227820)

[Data Leakage 13](#_Toc199227821)

[2. Metrics: 14](#_Toc199227822)

[**2.1. MAE and RMSE:** 14](#_Toc199227823)

[3. Supervised Learning: 19](#_Toc199227824)

[**3.1. Linear Regression:** 19](#_Toc199227825)

[**3.2. Logistic Regression** 19](#_Toc199227826)

[**3.3. SVM: Support Vector Machines.** 21](#_Toc199227827)

[**3.4. KNN.** 21](#_Toc199227828)

[**3.5. Naïve Bayes:** 21](#_Toc199227829)

[**3.6. Decision Tree:** 23](#_Toc199227830)

[**3.7. Ensemble Learning:** 31](#_Toc199227831)

[3.7.1. Voting: 31](#_Toc199227832)

[3.7.2. Stacking: 32](#_Toc199227833)

[3.7.3. Bagging. (Bootstrap + Aggregation) 33](#_Toc199227834)

[3.7.4. Boosting: 35](#_Toc199227835)

[How boosting works: 36](#_Toc199227836)

[Gradient boosting algorithms: 36](#_Toc199227837)

[For classification (Difference specific to classification): 39](#_Toc199227838)

[Why gradient boosting is a gradient descent in a function space? 39](#_Toc199227839)

[3.7.5. XGBOOST: 40](#_Toc199227840)

[Core idea of XGBOOST: 40](#_Toc199227841)

[Mathematical foundation: 40](#_Toc199227842)

[3.7.6. LIGHTGBM 42](#_Toc199227843)

[Leaf base split: 42](#_Toc199227844)

[Histogram base learning: 43](#_Toc199227845)

[Categorical data handling: 43](#_Toc199227846)

[Exclusive feature bundling: 43](#_Toc199227847)

[GOSS (Gradient base one side sampling): 44](#_Toc199227848)

[What is Parametric and non-parametric models? 45](#_Toc199227849)

[4. Unsupervised machine learning: 46](#_Toc199227850)

[4.1.3. Agglomerative clustering: 46](#_Toc199227851)

[4.1.4. Silhouette score: 46](#_Toc199227852)

[4.2.1. PCA 47](#_Toc199227853)

[4.2.2. T-SNE 47](#_Toc199227854)

[5. NLP 47](#_Toc199227855)

[DL 52](#_Toc199227856)

[Vanishing Gradient 57](#_Toc199227857)

[How to recognize 57](#_Toc199227858)

[**Reduce Vanishing Gradient** 57](#_Toc199227859)

# **Bias vs variance:**

Learn it

from a book and lecture in way to depth with numerical calculation.

**Bias:** Bias is the error introduced in the model’s prediction due to over simplistic nature of the model. It reflects / measures how much the models’ predicted outputs deviate from the true values, often because model is unable to capture the full complexity and patterns of the data.

Bias the words

**Why is bias called bias?**

Bias indicates the bias nature of the model to predict values towards its ideal simplistic nature, like linear regression is bias towards linear relationships irrespective of the real nature of the data. Naïve bayes is bias towards the feature independency. This is the start of the origin of the concept, bias indicates how good model can fit on intricate patterns with no restrictive assumption. High bias means high inclination of the model towards the restrictive assumption, very simplistic model like linear regression. High bias means models simple model which makes the mistakes on the training data itself. Whereas the low bias means model does not incline towards the specific restrictive assumption.

Sources:

Choice of the algorithms, like selecting linear regression for the highly non-linear data.

Important relevant features are omitted.

Non-representative training data,

The best geometrical intuition is you fit a linear model on the non-linear data.

Quantify Bias?

* Quantifying bias is based on the performance of the model on the testing set.
* The overall error is bias \*\*2 + variance + irreducible error.
* Quantifying bias here?

Why do we need overfitting and underfitting if we have bias and variance concept.

* Bias is the mathematical statistical concept which calculates the actual error/deviation in the model prediction due to oversimplistic nature of the model.
* High bias causes underfitting.
* High variance causes overfitting.

Example: for house prediction you only select location as one input variable and then make the prediction base off the location, which makes the prediction very biased towards the location variable only, though the actual dependency is on many variables.

**E[MSE] = (Bias)\*\*2 + Variance + Irreducible error**

**Bias = (E[f’(x)] – f(x)) \*\*2**

**Variance = E[(f ‘(x)- E[f(x)])\*\*2]**

Why do we need bias and variance.

Intuition: so, the best model is a model where the sum of this error is minimum i.e. we want to capture most patterns in the data at the same time don't want to make the model too specific to that dataset only, so while capturing the most detailed complex patterns from the data, we want to make sure that we don't make the model very specific to that dataset only, and have an ideal trade-off between the bias and variance.

Bias refers to the error that arises from a model’s tendency to make overly simplistic assumptions about the underlying data, leading it to miss important patterns or relationships. In other words, bias occurs when a model is not flexible enough to capture the true structure of the data, resulting in predictions that are systematically off-target. High bias causes underfitting, where the model performs poorly on both the training and test data because it fails to learn the relevant features and relationships

**Watch bias and variance part from campus in details.**

Variance measures how much a model’s predictions change across different training samples.  
If a sample mostly contains the true underlying pattern and little noise, the model’s predictions will be similar (low variance)  
If a sample has lots of noise, the model may fit to that noise, causing predictions to vary more between samples (high variance).

Variance: error generated when input to the model changes.

Trade off

Overfitting

Underfitting

Overfitting is like low bias.

What is the difference between Loss function and error metric on test set.

# **Feature Engineering:**

## **Feature transformation:**

Transforming the raw features into a format which better represents the data pattern. It includes changing the existing features.

### Outlier Detection:

You either removes the outlier values based on your detection techniques, or you treat them as missing and then try to fill them or you can cap those values to the extreme allowable values based on your model requirements. Or you can do the power transformation before that might just handle the outlier. Most importantly the tree-based algorithms do not really get impacted by presence of the outliers.

#### **Z-Score method:**

Mainly used on the normal distributed data where, we calculate the Z score of each point which indicates how far away the point is from mean in terms of the standard deviations. Or rather how many standard deviations away the point is from the mean. And then standard deviation in our case is the unit of the variability in the data, so basically indicates how much is the variability of the point around the mean. i.e. you calculate the z-score and basically the variability of the points around the mean and if this variability is beyond a certain threshold like 3 or 3.5 which basically contains most of the data, i.e. value of variability beyond these points means it lies beyond majority of the data and can be considered as outlier.

So just calculate the z-score and take only the values which are within the threshold. S.D. = 3 is general approach.

Data needs to be gaussian to use Z-score method of outlier detection. And if it is not then in that case a **modified a Z-score method** is used which uses median and IQR with the scaling multiplier to reduce the value of this z score to make it more comparable with mean z-score as median z-score values come up as bit bloated, so to have same comparable threshold we reduce those z-score values from the median modified z-score. Uses MAD, median absolute deviation. Modified Z-score method is generally a better approach than IQR. How to calculate the modified Z-score?

0.6745 \* (x-median) / (MAD) where MAD is median absolute difference,

So, the main concept is same which is calculate the degree of deviation/ variance of a point from central measure/ tendency with respect to the actual dispersion / variability of the entire data, i.e. calculate

Dispersion / distance from mean per unit SD (overall dispersion/variability).

How single point is dispersed compared to the overall dispersion of the dataset, however the main difference in modified z-score the variability of datapoint is calculated by comparing with median and the overall variability of the dataset is calculated by MAD(median absolute deviations) meaning the measure of the dispersion here is MAD instead of SD. Concept is similar, just the way we calculate the numerator and denominator are with different metrics but the same thing.

Both try to quantify the z-score, and modified z-score tries to quantify the relative deviation of a data point from central measure. (relative with the SD of the entire dataset).

Just the difference is factor of 0.6745, cause

What happens in modified z-score the numerator comes out to be smaller usually for skewed / non-normal distributions (for perfect normal it is anyways mean=median) as median gets affected less by the outliers and stays closer to the dataset overall. And the MAD is smaller as well compared to the SD,

Which is almost equals to MAD = 0.6745 \* sigma for perfect normal distribution. i.e. why you multiply by the same factor as the values without multiplying comes out to be bit higher (bloated in modified Z-score). You try to take the similar comparative variability measure i.e. why take MAD/0.6745 which is increasing the MAD eventually decreasing the entire quantity. Now the question is why 0.6745 specifically?

Mean, mode, median values remain same for normal distribution data, whereas for skewed it changes, mode obviously is to the extreme and it remains towards the extreme. For left skewed data i.e. tail on the left sequence goes like this, mean, median, mode. And for the right it goes like mode, median, mean.

The way to understand it is, first assume the normal distribution and then add the outliers at the ends and try and see how it changes the mean and median, adding the outliers towards the right making it right skew, this will increase the mean by very significant amount but won’t change or increase the median by significant amount as however much could be the extent of this

* + - 1. **IQR method:**

In this method a IQR is used to detect the outliers in the non-normal data which uses IQR percentile approach, the lower and upper limits are decided by the Q1-1.5(IQR) and Q3 + 1.5(IQR) respectively.

so what you do is, you take the middle 50% of the data, which lies between percentile 25-75 which also gives you the spread/dispersion measure IQR as the difference between (75-25), now to set the limit so that you can have datapoints beyond the middle 50% datapoints, you extend the boundaries of this middle 50% datapoints by adding 1.5\*IQR (extending the upper limit by 1.5 times the spread) whereas reducing the lower limit by 1.5\*IQR ( reducing the limit by 1.5 times the dispersion in the dataset). is it correct intuition of the IQR method?

There is a practical and statistical rationale for using **1.5** it balances sensitivity and specificity, aligns well with expectations for normally distributed data, and has become a standard convention due to its effectiveness and simplicity. However, it is not the only possible choice and can be adjusted for specific needs.

* + - 1. **DBSCAN:**

Implement in notebook and understand. Main thing here is selecting the parameters like epsilon and n, which decides the outliers, low epsilon with high n is stricter compared to the vice-versa, try and run multiples values, plot the required graphs, investigate this from the DBSCAN working.

### Missing value Imputation:

For numerical feature, make sure you try to keep more than 95% data if you are going for CCA.

And on imputation the distribution of the numerical variables should not change too much, Check for KDE distribution. o

For categorical with regression problem, see the distribution of the output target variable with respect to that categorical variable before and after the imputation and make sure that distribution of those all the categorical variables is not completely changed.

Dig deep into this thing, like how it changes and if it changes what should you change.

#### **KNN imputation:**

Based on the K nearest neighbours, but the nearest value is calculated based on the variables/values which are available, just ignore the dimensions where the values are null. And weight is used in calculating the output. Distance as a weight or any custom weight or just normal without any weight.

A weight is reciprocal of the distance is used as the distance is inversely proportionate to the impact that point have on the imputation value.

By ignoring the dimensions which are not available you basically do not calculate the actual distance, but you calculate the planer distance as you do not have the exact value of that dimension. But as you do the same for all the points it evens out the error as ultimately you calculate the imputed value based on the points aggregation not the distance aggregation.

Not specifically used for categorical data. So, does the iterative imputer uses categorical features for calculating the distance?

Yes, encoded categorical features are included in the distance calculation, but standard distance metrics may not always be appropriate for categorical data. For best results with mixed data types, consider using a specialized distance metric like Gower distance.

**Missing completely at random, missing at random, missing not at random.**

* **MCAR: Missingness is random and unrelated to data:** Data is missing because a participant forgot to submit a form by accident
* **MAR:** Missingness is related to observed data: Older people skip the last question because it’s on the next page, but not due to content
* **MNAR:** Missingness is related to unobserved, missing data itself: People with worse health drop out of a study because they are unwell

**After the encoding of the categorical features, scaling is not done on those features.**

**How algorithm works?**

For each missing values, take this point (row) as the centre and calculate the distance of all the point

from this point, while calculating the distance the features which are missing are ignored means lower dimensional distance is calculated. And the based on the top specified points the value is filled.

Weighted distance works like, summation of (1/ distance) \* value / summation of (1/distance)

#### **Multivariate Imputer:**

The columns which are filled at the last are much better filled than the ones which are filled initially.

* Initially the values are initialized with mean etc.
* For the first iteration the first column null value rows are treated as test data and the remaining data is treated as train data which is trained with first column treated as target column. The prediction is done for the test part, this prediction is nothing but the imputation of the first column which based on the regression model.
* For each subsequent column you use the newly updated values and not the values from the earlier iteration.
* Likewise, you do for the all the subsequent columns, for each current column you use the better and better data for the model as the previous values are filled based on the regression and not just the mean initialization which is the case for the earlier columns.
* The intuition is that we try to capture the numerical patterns between the features and use those patterns to fill the imputed data. But pattern capturing is done iteratively as model starts with the initial mean values which with each iteration improves because it gets more and more accurate data for next feature values prediction / filling.
* Max iterations are set based on the subsequent changes in the next iteration.

Main question is how do you used it on the mixed data? i.e. data with numerical and categorical features as well.

* By repeating the imputation process in an iterative fashion, the algorithm enables the imputed values to gradually stabilize and converge toward more accurate estimates, as each round leverages the most recent imputed values from the previous iteration, thereby progressively refining the predictions and improving overall imputation quality
* **How to impute for the categorical, and which measure to check after?**

#### **Column Transformer and Pipelines:**

Use to streamline the preprocessing and prediction steps on the training and testing data. Very easy to optimize way of preprocessing and prediction with very much ease of automation, particularly useful for production purpose. Although pipelines work seamlessly with cross-validation as well, so very useful in model building stage as well and not just the production automation.

Column transformers transform the columns one by one as it given to it in its steps and joins those columns alongside and returns a NumPy array. Column Transformers works parallelly combining all the data points.

When you specify a column twice in a distinct transformation then even the second time it receives the initial column only and adds it along with the other column and now, we’ve two transformed columns out of single initial column. The natural intuition of yours is that for the second transformation you’ll received transformed column after first transformation is not true.

You cannot apply two distinct column transformers on the data in series as the first one returns the NumPy array, so the second one receives the NumPy array and fails because of it. The solution is you use pipeline in column transformers, which allows the serial transformation on a single column in transformer. For a one transformation within a column transformer use pipeline and specify multiple serial operations.

The intuition is you pick one by one column and with the help of the pipelines specify all the serial transformation for that column in one transformation of the column transformer and likewise do it for all the columns and eventually you’ll get the NumPy array which you can fed to predictor without any issue.

Whereas the **pipeline** works serially on the different transformation’s steps put into to the pipeline, i.e. the first step will be processed first on entire data and then the second and then the third obviously how and what is depends on which step and what methods is used.

The last step in pipeline is by default treated as the predictor and works slightly different than all the previous steps.

All the transformer steps on given fit which is one of the most basics of the method does **fit\_transform** on the data whereas it does only fit on the predictor. And we do fit on the train data, i.e. for training by fit method of pipeline we do fit\_transform on transformation and only fit on the predictor which is the last step.

Likewise for the test, we do predict, which does the only transform on the data, which is test data here, and predict for the predictor step on the test data usually.

Pipeline does not do any transformation, column transformers and predictor do the transformation, but pipelines just combine those transformation in serial manner, outside of column transformers or within the column transformers.

Just like pipeline there is one more function make\_pipeline which is simpler version of the pipeline which receives the list as a parameter with each element being the estimator object directly unlike the Pipeline which receives the list of the tuples where first element of the tuple is name of the estimator and second element being the object of the estimator. The name to the estimator step is taken as the lowercase name of the estimator itself.

Major advantage of the pipeline over the make\_pipeline is its ability give custom meaningful names. Though it provides more flexibility and control in certain complex scenarios.

### Encoding:

**Except from the LIGHTGBM every other ML algo requires encoding, types of encoding are:**

#### **OHE**

#### **Categorical encoding**

#### **Label encoding**

#### **Target encoding**

Encoding the categorical feature based of the target value, it replaces each category based off the aggregated statistic of the target for that category, mean for the regression and probability (likelihood) for the classification. So, the main aim here is, instead of allocating random numbers to categories it’s very logical thing to assign this category based off the target.

How does it work?

Group by category

Compute target statistic

Replace the categories.

For regression: Categories are replaced by the mean of the target.

For classification, the same is done by probability or likelihood. i.e. calculate the local (category) probability of the category with respect to the positive class. i.e. for category A, positive classes / total no. of category A entries.

**But if not regularize it has few drawbacks like,**

1. **Overfitting on the training data:** if the category is directly represented by mean/likelihood of the target then, model would remember the training data as there is direct coupling between the target and feature as the feature categories are replaced by the direct and exact mean of the target, which leads to overfitting by leaking target stat in the categorical feature. Regularization is done with target’s global stat then how come it reduces this **data leakage**? By having the weighted involvement of global statistic, it reduces the impact of the leakage by controlling the weight of the global stat by regularization parameter.
2. **Instability with small sizes**: statistics calculated with **small samples** is highly unreliable and is not the true/exact representation of the category, hence it the statistic is highly unstable and sensitive to outliers and noise. E.g. if the rare category has a single category observation with an extreme value, then the category representation would have very high share of this rare outlier.
3. **Handling rare categories**: high cardinality features have high number of categories, sometimes even single occurrences like zip code, in this case this rare category gets very extreme, noisy encodings.
4. **Handling unseen categories**: For unseen categories in the test set with this we would have some value due to presence of the global and its regularizing coefficient.

how smoothing or regularization is done.

For regression, encoded\_value = (n \* category\_mean + m \* global mean) / (n+ m)

Where m is smoothing parameter (controls the weight of the global mean)

N for categorical local smoothing which is equal to the number of rows of the categories.

Same for classification, encoded\_value = (n \* category probability + m \* global probability) / (n+ m)

Higher m leads higher smoothing i.e. higher impact of the global mean, values tend towards the global mean, extreme values move towards the global mean. Whereas,

Lower m leads to the lower smoothing with i.e. having higher impact the category statistics.

Factors which affect M size:

Category frequency (size of n): If the frequencies of the category values is high then the lower smoothing is required, and vice-versa.

Dataset size: Larger dataset would have large number of samples to even out the extremities, so low m is needed.

Target variance: if the target variance is high then same would be reflected in the category stat (encoded value) so the high m might smooth out the extreme category means.

Model type: Few models are far more robust to the extremities in the category stats, so smaller m might be okay.

Ranges of m = 1-10, start with 1 then go towards the, or grid search can be used.

**Why use target encoding:**

Capture target relationship: unlike the label encoding, target encoding uses the target stat to represent the categories, which creates meaningful numerical encoding based of the to predict variable, as the encoding provides the information about the predictive variable.

Reduces dimensionality: Unlike OHE, which is the go-to option, it does not increase the dimensionality of the input.

Works very well with tree base models: tree base model and ensemble works well with target encoding as they thrive on numerical splits, so what better than splitting based off the meaningful, ordinal features.

**Cons of the target encoding:** Target encoding can lead to the overfitting, target leakage, and loss of information if not managed well. Requires extra step like smoothing, depends on the relationship of categorical with target, not ideal for the new categories, imbalanced data and all model types(linear). Which makes it a trade-off.

1. **Target leakage: (explained above)**
2. **Overfitting**
3. **Requires regularization (smoothing)**
4. **Loss of categorical information**: a category is represented into a numerical value, which leads to loss of other information by oversimplifying. E.g. two categories with same mean but with different spread can exist, which now has identical encoding.
5. Dependency on the target: encoding quality depends on the correlation of the feature with target.
6. **Not ideal for all models** works best for tree base algorithms but does not work well with linear models, where the encoded category values could be interpreted as the ordinal which may not always be the case and even the scale could be totally different. Compared to the other feature values.
7. **Handling new categories in the data:** if the category appears in the test set but not in the train set then you do not have target mean to encode for that category, so the global stat is used which may not be the accurate or informative representation.
8. **Sensitivity to data imbalance:** if the data is imbalance, then in this the global mean dominates, and rare categories may not reflect their true positive class probability, even with smoothing.

**When to use and when to avoided?**

* High cardinality
* Strong relationship with target
* Tree-Based model
* Dimensionality constraints
  + 1. **Feature scaling:**
* Feature scaling is a technique is used to represent the numerical data into a common scale so that larger values do not dominate the model predictions. Makes sure all the features contribute equally. Models which depend on distance metric particularly, a feature with higher value dominates the distance and eventually model’s prediction, to avoid the same we get those values in similar scale keeping the distance/distribution within the feature same.
* Very useful for distance-based algorithms like KNN, Linear models, KMeans, SVM and the models which are based gradient descent like NN which converge better with scaled features as the normalized features coefficients are kind of similar unlike very much difference in them cause much smooth transverse towards the optima rather than messy zig-zag movement towards the minima.
* Not necessary for tree base models, as these techniques does not change the variance within the features and variance within features is not dependent on the scales of features as well.
* There are two methods to do it:
  1. Standard Scaler: Use Z-Score method to change the scale of the data. What Z-Score does, it basically gets the data into a scale where the mean of the data is 0 with S.D. is 1, keeping the distribution same, i.e. **shape remains the same, just the scale changes such a that it shrinks in a way where all values get distributed around zero mean with S.D. as 1. It is a linear transformation which does not change the shape just shrinks the shape, to change the shape it must do non-linear transformation like log, power, yea-Johnson transformations.**

More robust to outliers compared to the normalization as normalization depends on the range which very much depends on the max and min values.

Circles the values around 0 as mean.

* 1. Normalization: changes the values in a fixed range unlike the standardization. (0-1, -1-1) etc.

Min-Max

Check from notes and fill it next time when you revise.

* + 1. **Power transformations:**

First get clear with homoscedasticity and then come to this topic.

Let's go little deeper into the thing, let me tell my understanding of the constant variance or homoscedastic, let's say you have 100 datapoints and you calculate the variance for it, it comes out to be some number, now obviously this number is average of the squared distances of those points from the mean, so it basically average out the individual datapoint distance / variance with respect to the mean. now these individual distances could be very high for some datapoints and could be very very low for some datapoints, and this will eventually be balanced out and we'll get a single value, which represents the average sort differences/distances/variation of values with respect to the mean. now if the if the data does not have this variation of some datapoints distances high and some's low, i.e. almost every datapoint has near about same distance/squared difference then we can it is constant variance of homoscedasticity is there and vice versa for heteroscedastic, is this understanding, correct?

### Data Leakage

Data leakage is a broad term, whenever a model has access to the information which it should not have. Model learns from this information which leaked into the training part and gives model an unfair advantage, because it learns from the data which it should have hidden from the model in training. Data leakage happens

due to following reasons:

If info about the test set is known to the model, or an unknown helpful feature is available to the model only at the time of training or the future data is known to the model while training (mainly in case of the time series data) then it the case of the data leakage.

Causes, feature only available during training something like

**Reasons of data leakage:**

* **Improper data splitting:** If the info from the test set or validation set leaks into the training set or training process, then model gets unfair advantage, leading to overly optimistic performance, cause model now knows what kind of test/validation set is there to certain degree. Which gives good result on the validation set or whichever set from which the information is leaked into the training set.In improper splitting the datapoints could be completely overlap or partially overlap which might leads to leakage.
* **Target leakage:** All the individual reasons of the target leakage led to the data leakage only, as the target leakage is the reason for the data leakage only.
* **Time base leakage:** when the future data is use for past prediction or future outcomes are used for current or past prediction.
* **Data preprocessing mistakes:** If the data preprocessing is done on entire data i.e. train-test set combinedly, then info of the validation/test is leaked into the training, preprocessing like imputation, scaling, feature engineering, outlier detection and all.
* **Access to a feature won’t be available at the time of the prediction:**
* **Overlapping data sources.**
* **Feature engineering with hindsight**
* **Synthetic data or simulation error.**

**Target Leakage:**

One special case of this leakage would be information from the target(Y) or output is leaked into the input(X). i.e. information from the target is used to train the model input variables.

Though Target leakage is the specific kind of leakage on wholistically it is one of the reasons data leakages.

**Reasons for target leakage:**

* **Inclusion of target variable:** if the target variable is as it is the part of the input variable. Exact feature is included or some part of it is include.
* **Features derived from the target:** If the feature is derived directly from the target.
* **Temporal misalignment:** If the future data is used as a part of the input, very typical case would be the prediction of patient admit is yes or no based on the discharge date. Its case of the future temporal data of patient is used if is admitted, if discharge data is available then prediction would be 1. Any such data whether it is temporal, or static does not matter if a single input feature represent target this perfectly then it is the case of the target leakage.

How to check? If there are discrepancies in performance on the validation set and test set, then there is chance that some data might have leaked into the train set from validation. Very high correlation between any of the input feature and the target.

# **Metrics:**

## **MAE and RMSE:**

what is the difference between them on intuitive level, just imagine the bars for each data point bars are just to one side or some part to left and some part to the right, then averaged them if the bars were an absolute difference which will give you and another bar which is nothing but the crux and aggregation of all the bars.

In case of the MSE and RMSE the bars are elongated by quadratic proportionate, which gives more weightage to the outlier points with the degree of two. To get the bars in same unit as the base data we take root of the MSE which is the average of the squared differences. i.e. MSE is the average of the squared bars whereas the RMSE is the root of the mean of the squared differences and not the mean of the rooted differences.

The RMSE can be calculated by two methods, first the correct approach and other one is rooting the sum and then dividing by the N, which is wrong but why?

The reasoning is intuitive level, Imagine the bars and on introduction of the square terms, the bars are elongated by the square proportionate, now our error is this value of each point, and we want to average it for all the points so that we can have its representation for each point. The average of this point is calculated by only one method which is sum the difference, which is the sum of the squared distances, which in this case represents the total error and for average we need to have the total distance or difference and just take partial value of it for each datapoint by dividing by the no. of datapoints. Which represents the squared error for each datapoint and then we take the square root of it to get the error representation for each datapoint in actual units. And if do the square root of the total error first then it does not remain the total error to distribute it to the each datapoint by dividing N. It gives a smaller value which is the underrepresentation of the actual average error.

MAE is more robust to the outliers whereas the MSE AND RMSE penalizes them much more.

If the data is suspected to have the outliers, then use RMSE or MSE depending on the context obviously.

Theory in details with every little nuance including all the hyperparameters along with the code implementation of the everything.

**R2 SCORE (Coefficient of Determination):**

So, basically the R2 score the competitive model which calculates the how well the current regression model fits with respect the mean model, so it checks the relative sum of squares i.e. sum of squares of all the datapoints by regression model with respect to the sum of square of all the datapoints by mean model. just to get the values in the specific range for all the comparative model it checks the error with respect to sort of one absolute model which is mean model, so that we can get a standard value between the 0-1 and for the models across different model.

The R² score is a comparative measure that evaluates how well your regression model performs relative to a simple mean model.

R2 scores allows the comparisons between the models based on how much variance in the data is explained by each model. If the r2 score is 0.7 then the 70 percent variance in the dependent variable is explained by that specific model which obviously translates to the other errors as well.

R2 score does not necessarily inversely proportional to the MAE

If you develop **Model B** that fits well to the non-outlier data points, it can reduce the sum of squares for those points and improve the R² score from 0.88 to 0.90. However, this model might move further away from the outliers, leading to larger absolute errors for those outliers and thus increasing the MAE from 0.50 to 0.60. This illustrates that while R² measures the proportion of variance explained by the model, MAE is more sensitive to individual large errors, especially those from outliers. Both metrics provide valuable but different insights into model performance. In this scenario, **Model B** shows an improved R² score by better capturing the variance for non-outlier data points, but its MAE increases due to the larger errors associated with the outliers.

R2 score is not the correct metrics in the scenarios where:

1. Significant outliers
2. High no. of features, particularly the non-useful
3. Small sample size
4. Non-linear datasets are not that straightforward to interpret.

The adjusted R2 score is used cause on increasing the no. of feature the model complexity is increase which increases the R2 score but to make sure the increase R2 score is justifiable that is due to the actual feature which add the understanding of the variance in the dependent variable and not just the increased in overfitting we use adjusted R2 score which increase or decreases the R2 score with respect to the no. of independent variables, how does it balances, increases, decrease or changes in general R2 score. It balances the R2 score with respect to the number of the independent feature.

MAPE (Mean absolute Percentage Error): same as MAE but just the relative of the MAE. Should be as low as possible.

Mean deviation Bias (MDB): summation of the errors without the mod, tells you about the over, under or exact prediction tendency of the model.

MDB > 0 = Model under predicts

MDB < 0 = Model over predicts

MDB = 0 Model is not biased towards any side.

Classification metrics:

1. Accuracy: total right to the total datapoints.

Accuracy is very vague way to judge a metric, and not the best metric in lot of scenarios:

* Imbalanced data: high accuracy score in the imbalanced data can be due to the correct classification major class and minor class accuracy could be very low.
* It does not distinguish between the type of the errors: In some scenarios few errors could be very costly compared to the others. Like in medical data where you are predicting cancer, False negatives are very costly compared to the False positives.
* Types of the errors in the Classification Metrics which are a bit different compared to the Type of errors in hypothesis testing.

Type 1 error: False Positive (Precision, critical in email spam)

Type 2 error: False Negative (Recall, critical in Cancer)

Type of errors in the hypothesis testing:

* Type 1 Error: reject the null hypothesis when it’s true.
* Type 2 error: accepts the null hypothesis when it is False.
* One true thing about the metrics is that they are not the absolute evaluation criterions for the models, you mainly try to check if the model is correctly doing the aspect which you want the model to do correctly. Most evidently it is seen in the Recall, precision in below metrics.

1. Recall / (Sensitivity) / True positive Rate (TPR):

When the cost of predicting the Positive class as negative is much more detrimental compared to the other way around then in that case we recall.

It mainly focuses on the fact that make sure you predict all the positive classes as positive and the other class’s correct prediction is not that important.

In recall you try to calculate the true positive with respect to the all the positive classes in the dataset.

i.e. Recall = TP / (TP + FN) 🡪TP / horizontal (Where horizontal is actual)

in Recall you do not give any fuck to the negative class. If only the positive classes are classified correctly out of the entire data, then also we can Recall as 100% or

rather if we classify all the datapoint as the positive the recall still would be 100 but that would be bad model.

Only using the Recall is probably not the correct strategy for the model evaluation, as recall value can be misleading in lot of sense. It is very easy to have high recall with classifying all the points as positive. As well it does not consider the FPR.

1. Precision:

When the cost of the predicting the negative class significant then in that case it important check the how many of the negative classes are predicted incorrectly with the correct prediction of the positive class.

Predicting the positive class correct is always the goal but with that you also want to make sure the negative class is not misclassified, then in that case you use the Precision. We want to make sure, out of the predicted positive values the falsely predicted values very less as the cost of those misclassification is much more, though the goal is predicting positive class which is not achieved by the precision for the same we can use recall. And that is why the Precision and Recall are used together or in combination.

It’s just that we want to predict the correct positive class, but we do not want to compensate for the negative class’s false prediction which was not that important in the initial context of precision.

Recall is the indicator of the correct prediction of the entire positive class, and tells me nothing about the other's class's misclassification, whereas the intuition for the

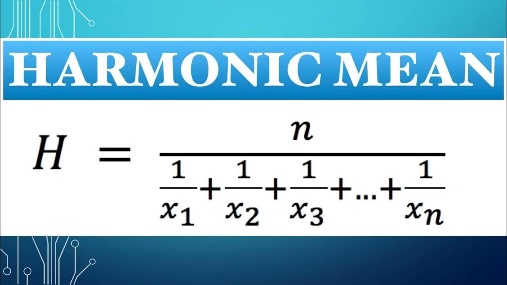
Precision is, when model predicts high precision it indicates, out of the positively predicted classes by the model most of them are true positive only and the misclassification of the negative class is not that much which is more detrimental in the above scenarios.

Recall focuses on the correctly predicting the positive class whereas the precision focuses on the not including the negative class in the positive prediction. Not it mainly focuses on or ensures that the model does not predict the negative class as the positive. It has no sense of how many of the positive classes are predicted correctly all it tries to check that it does not have any negative predicted as positive.

Positive class: The class of the primary interest.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Predicted** | |
| **1** | **0** |
| **Actual** | **1** | **TP** | **FN** |
| **0** | **FP** | **TN** |

Harmonic Mean:



**F1 score** aggregates both the scores based on harmonic mean where the beta values are 1.

The more underlying metric is F-Beta score.

Beta gives you the ability to adjust the impact precision and recall in calculation of the F-Beta score.

β < 1: More weight to precision

β > 1: More weight to recall

β = 1: Equal weight (equivalent to F1 score)

chipake khichta hai neeche.

Weighted F1 score is better at handling imbalance than the normal F1 score, which is the weightage average of the F1 score with respect to positive class and F1 score with respect to the negative class.

ROC curve (Receiver operating curve):

A curve plotted between the TPR and FPR for a model with different threshold.

First, what is TPR and FPR?

TPR is nothing but the Recall/Sensitivity.

FPR is (FP / FP + TN) i.e. with respect to the negative class how many of those are misclassified, means more the misclassification of the negative class more will be the FPR, so you want it less whereas the TPR is you want it as high as possible as it tells you how many of your positive classes are classified as positive.

FPR and TPR are noting but the indicators that your model is good at predicting the positive and negative values.

Plotting a ROC curve for a model, you take different threshold and calculates FPR and TPR for each threshold for that specific model and plot those values for each threshold you get a curve.

High AUC-ROC value indicates the model generally across the thresholds have better ability to distinguishes between the positive and negative classes, cause the high AUC means high TPR which means better ability to classify the positive classes.

At the same time high AUC indicates the low FPR which implies the low FP’s which implies better ability of the model to identify negative class which also implies the better ability precision.

High AUC-ROC value, better ability of the model to distinguish between the positive and negative classes with high Recall and High Precision.

AUC curve which travels closes the top left corner is selected means that model as it has more area (Usually).

# **Supervised Learning:**

## **Linear Regression:**

## **Logistic Regression**

Logistic Regression: The basic intuition which is to separate the classes by a linear plane based on the distance (probabilities which are proportional distance) to the i.e. Furthest the point is from the model plane better predicted that point is. But how do you get the model. Random model improved by the gradient descent.

Once you’ve a model with respect to that linear model, you calculate the distance of the datapoints from the linear model, furthest the point, better classified it is. In general, the model which keeps the most distance from the model and datapoints is the best model. So, once calculating the distance of all the points we just product the distances of all the datapoints irrespective of the classes and whichever model gives you the most product of the distance is selected. (Obviously you can use step function which would just tell you which point is classified correctly, and which one is not but it does not value the higher chance of point to be classified as certain class). But generally, these probabilities are very small and product of this is also very small so, you basically add log into it which makes everything negative you instead of wanting it higher you want the value of the product minimum.

This log addition of the probabilities we call as likelihood. From the same likelihood we get the concept of the log loss error which is binary cross entropy which is smart logical way to calculate the probability of each datapoint with respect to each class.

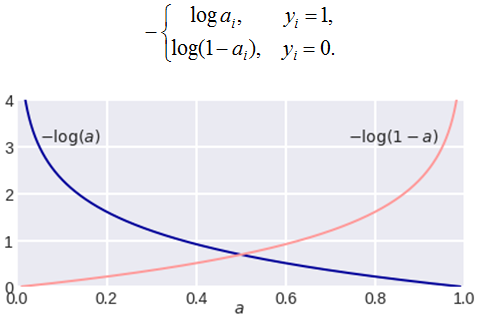
Actual step:

* Calculate the distance.
* Probabilities from the distance by the Sigmoid / SoftMax.
* Error from the probabilities by log loss error.
* Based on the error, by the gradient descent calculate the model and do it iteratively, obviously the error has the feature variables because it based on the probabilities, as probabilities has Z which is the distance marker which has the feature variable which ultimately represent the model. i.e. means with gradient descent, derivative of the loss with respect to the feature variable is sort of very indirectly dependent on the loss.

Loss function: Log loss Error / Binary Cross Entropy:

It is designed such a way that for the positive points only the positive values error is calculated and vice-versa.

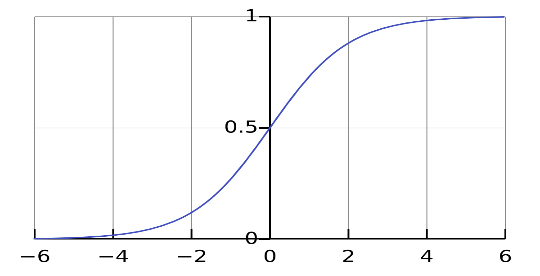
And obviously the log is for the handling the small values and is derived from the



Function in the logistic regression:

**Sigmoid:** The functions “S” shaped curve which is convert the minus infinity to positive infinity values into the range of the 0-1. The higher the X higher the output is more leaning towards the 1. So there definitely the positive correlation between x and y here, just the exponential one. Symmetrical along the y axis. And at the x= 0 the output is 0.5.

Specifically used to convert the distances in the logistic regression into the probabilities.

****

**SoftMax:** a function which returns an exponential normalization of an array given. Which in out context basically represents a relative probability of classes. So called as uncalibrated probabilities as they do not have any absolute sense, just the relative sense which gives more weightage to the higher values.

How it handles the multiclass problem?

In general approach in multiclass problem is to create as many hyperplanes as the number of the classes. And now there are two different ways to create this hyperplane first OVR and the second one is SoftMax.

There are two methods.

* + 1. **OVR (one vs rest).**
* You one hot encodes the target variable and for N number of classes N number of the models are created with respect to each class. And the normalized probabilities are calculated for each datapoints of the all the probabilities given by each model as each model return the probabilities in the scale of the 1, so we need to convert those into scale of 1. And whichever probability comes the highest, the datapoint belongs to that class.
* So, basically multiple models are created and trained each has its hyperplane for binary classification unlike SoftMax which handles multiple planes in a single model.
* In OVR, for each model/hyperplane a probability of the 1/positive of that class (a probability of point belonging to that class) is calculated which is then normalized.
  + 1. **SoftMax:**
* In SoftMax multiple hyperplanes created and handle within a single model with the inclusion of the SoftMax function which handles the multiple probabilities in a single model.
* A distance (Z) for a single datapoint is calculated for all the N classes with respect to their hyperplanes. And a SoftMax return the relative probabilities of the classes out of which the highest one is selected. What SoftMax does is? it calculates the normalized exponential of the distances (Z) of each class with respect to their hyperplanes. And this normalized exponential value is nothing but the relative probability of the classes (or rather relative probabilities of the distances which are great indicator of class classification or geometrically a region indicator of that point).
* And this max value decides to which class this point belongs.

Just the main difference is in the way the SoftMax and OVR integrates the multiple hyperplanes, OVR creates the multiple models and select the highest normalized class out of all whereas the SoftMax can handle multiple hyperplanes in single model returns the highest relative probability of a point to belong to a class or region.

Number of hyperplanes:

Softmax: Uses K hyperplanes for K classes.

OvR: Also uses K hyperplanes for K classes.

Training approach:

Softmax: Trains all hyperplanes simultaneously

OvR: Trains each hyperplane independently

Decision making:

Softmax: Uses all hyperplanes together, applying SoftMax function.

OvR: Considers each hyperplane separately, often using sigmoid function.

Output interpretation:

Softmax: Directly outputs probabilities that sum to 1

OvR: Outputs independent probabilities for each class

Computational efficiency:

Softmax: Generally, more efficient, especially for many classes

OvR: Can be less efficient, especially with many classes.

So, while both methods use multiple hyperplanes, their approach to training and using these hyperplanes differs significantly.

Interview Questions on the logistic regression:

Why logistic regression is called logistic regression?

First the logistic part comes from the logistic function which is nothing but the sigmoid function. Logistic probably comes from the logistic number which probably comes from the logarithms.

The regression part is because it uses linear combinations of the features to calculate the probability of the datapoint and ultimately calculates the linear plane which distinguishes the classes. Cause the regression was used in linear Regression and due to their historical relevance, they have similarities in a way they represent the features.

## **SVM: Support Vector Machines.**

Support vector machines

At the end make sure to add the similarities and differences between the SVM and Logistic Regression.

## **KNN.**

## **Naïve Bayes:**

Uses the probabilistic approach by bayes theorem to decide the highest probably of a target class given the input conditions which is input features combinations. So, technically based on the input features combinations the probability of the all the classes is calculated whichever has the highest probability of occurrence for the given condition or given combination of the features is chosen as the correct classification.

The point is how do you calculate these probabilities. Bayes theorem is used to calculate the conditional probabilities, what is the probability of the given class given the condition.

The proof is as such, to calculate the probabilities of classes given the condition, with the bayes theorem for the above probabilities we calculate the probability of the conditions given the class and multiply by the prior probability of that class. So, the main issue lies in the fact how do you calculate this first term of probabilities of the feature occurrence given the class, it turns out that this terms simplifies to the individual probabilities of feature values for that datapoint for that class, which is fairly easy to calculate now, as we just sum this category value for the given class and divide by the total number of the class occurrence, basically calculate for a given class and given category how many of this class occurrence has this probability and you do the same for all the classes and do it for all the categories of a feature then repeat for all the features.

Learning in algorithm is basically just the probability calculations and stored in dictionary and on prediction, you just call the value puts it into a model calculations whichever class is more likely is our output.

Naïve bayes works good with the small datasets.

For the numerical features you use probability distribution, mostly preferably normal distribution and if it’s not normal make sure you make it normal distribution. Or even the other distributions are used as in when needed or if more suitable. And obviously the distribution is supposed to be normally distributed for a given class not he entire feature, if the N classes are there then all numerical feature values corresponding to those N classes should follow the distribution separately.

Obviously, there is another method of creating the categories by binning and KDE can be used and so on.

Assumes the independence of the features needed for the proof and that is why called as the naïve and gives equal weightage to all the features.

There are different types of the naïve bayes based on the distribution of the variables:

For numerical as well as categorical features together, we do not have a specific naïve bayes in sklearn

Better use

from mixed\_naive\_bayes import MixedNB

other implementations in the SkLearn are

based on the distributions:

Bernoulli

## **Decision Tree:**

* No preprocessing needed like scaling, distribution skewness etc unlike other algo’s.
* white box model, result is very much interpretable.
* But it has huge tendency to overfit.
* Not stable nor continuous output prediction which makes it difficult to use for extrapolation.
* DTs are very unstable, even the slight change in the input data would make entirely different DT which cause high change in the output for very small change in the input.

**What do DT’s do?**

Let’s take an example, we’ve a data of personal showing their expenditure characteristics and personal demographics based on which we would like to predict whether the personal is Extravagant spender or careful spender. Data includes personal info like Name, Age, Gender, Address, Urbanicity, spending chars etc. Now let’s suppose we decides to separate the target variable based on single variable, like urbanicity in above case, we’ll have Extravagant spender in urban and careful spender in rural, so even based on this single column we can very much predict the outcome if all the Extravagant gets into rural split and the careful ones into rural split. Obviously, we won’t have pure nodes after this split and then on next splits we’ll try to split this individual split on the next best features and ultimately reach the pure nodes. i.e. DT’s.

**Intuition:**

Viz this data and try to see which feature is split the data so that we have bast segregation of the target variable. i.e. each split is pure or as close to pure as it can be. And the vanilla/ traditional DTs tend to split till each node/split is pure.

The goal in the DTs is to find all the possible branches, pathways to reach pure node with homogenous datapoints. This branches which reach pure node are created by splitting the data into more similar section (tends towards homogenous). These branches are created based on the feature values, and while prediction based on the feature values branch is traversed and corresponding node helps predict the output.

Geometric Intuition: Geometrically Decision trees splits the region with plane orthogonal to the dimension/axis with each branch creation and regions are formed in a such way that homogenous points are clubbed together, and more branches you create/ deeper you go in DTs more and more subregions are created where each deeper region is more and more homogenous, at the deepest level of DT with max branches you have regions which are purely homogenous which are pure nodes itself. This recursive partition makes rectangular regions in planer region whereas hyperrectangular regions in multi-dimensional decision space.

Actual Algorithms: so, we basically want to create this branches / region based on feature values which are all the possible unique ways to reach a pure node.

Classification: The basic underline rule is you need to split the data into more homogenous subsets. The way to do it is, you start with feature with a feature value that splits the data in best possible homogenous subsets i.e. you split based on the feature and feature value which lets you make the splits with least entropy / variance in the subsets. i.e. more homogenous subsets.

How do you judge the entropy, variability, randomness in the subsets based on the feature and feature value?

In classification problem there are mainly tow ways measure the entropy,

1. **Gini impurity:**

Calculates the probability of misclassification of the randomly selected element if it were randomly labelled based on the class distribution in a subset. It indicates the degree of split, or the quality of the subsets created. It indicates the mixture in the subset, all the classes are same then the Gini impurity would be lowest. Value of the Gini impurity varies between the 0 and 0.5 for the binary classification, where 0.5 being the worst value with maximum mixture and creating this subset is not helpful irrespective root Gini impurity score.

On fundamental level it tries to answer the similar information like entropy which is the degree of randomness and mixture in subset. Though there is slight change in which both do it.

The range of the Gini impurity value is based on the number of classes and given by:

1-(1/n) i.e. for 2 🡪 1-0.5 = 0.5

* (1-1/3) = 0.66
* And 0.75 for 4
* has option to select entropy.

Gini impurity (Ho) = Gini impurity indicates the probability of misclassification of randomly selected element if the elements are labelled randomly (randomly chosen from pool/dataset i.e. proportional probability).

It indicates the similar thing as indicated by the entropy, which is randomness, mixture, inability to predict correctly.

Gini impurity = 1 **-**

Gini impurity = 1 - (p1^2 + p2^2)

Formula breakdown:

First let’s understand what is randomly labelled. Randomly labelled means randomly a single element is picked, i.e. proportional probability of each element. i.e. if there are 100 balls of Red, Pink, Yellow with 60, 30 and 10 samples respectively. Then on labelled randomly or one is picked randomly then what is the probability, for red = 0.6, pink = 0.3 and for yellow = 0.1.

Now the correctly classifying red as red, pink as pink and yellow as yellow with this proportional char is given as p(r) \* p(r) + p(p) \* p(p) + p(y) \* p(y)

i.e. 0.6 \* 0.6 + 0.3\*0.3 + 0.1\*0.1

and now the probability of misclassifying the elements is 1- this above value.

This is how we get the metric of randomness, inability to predict, entropy, mixture is obtained.

Used for CART

1. **Entropy / information Gain:**

LOG: what are logs? The basic intuition of the log is that they describe what power of the base is the required value. i.e. LOG (100) i.e. logarithmic value of 100 with base 10 is 2, because it tells what power of base it needed to get the 100, which is 2, 10^2. Same for all the logs, for ln it finds what power of e i.e. 3.71 is needed to get the required specified value. Log for 1 with any base is always 0 cause of the same and 1 is logarithmic of value same as base.

Make sure to watch intuitive content of log.

Entropy: Randomness / mixture/disorder/information/surprise in the system is generally referred as entropy, system could be any statistical, physical (thermodynamics). Entropy indicates how uncertain the system is or how unpredictable it is. Most important definition: In information theory, i.e. in our context entropy **indicates the average amount of information or uncertainty in random variables**. And basically, tells how unpredictable the outcome of the process is. This entropy is given by Shannon entropy formula which is:

In the above formula the p(x) indicates the probability of the outcome whereas the log2(p(x)) measures the information or surprise element of the outcome, lower the prob higher will be the surprise element of that outcome.

Entropy for 2 outcomes, i.e. tossing a coin,

= 0.5(log2(0.5) + 0.5(log2(0.5) = -1 = 1

This is max entropy of the system in binary output problem, same for the 3 output it is log2(3) = 1.58 and dice it would be 2.58, log2(6) and 4 is 2.

Why is entropy important? Because it determines the amount of information needed to describe the data.

How come the entropy is average of the information or uncertainty or surprise? Information term is weighted by the probability, means only the respective required proportion of the information is considered.

**Information is surprise is uncertainty. Which is always the higher when we have equal likelihood and no bias. i.e. system is most random for no bias and equal likelihood. And information, uncertainty, surprise must be lower as low as possible.**

Information tells how much the uncertainty of this event is, calculated with the help of the probability of the event and weighted by probability itself. Smaller the probability, rarer the event higher is the information,

Information gain is reduction is entropy or average information after the split about the dataset.

Used in ID3 and C.45

1. **Mean squared errors (variance): For regression.**

There are few DT algorithms based on how they split the data and creates branches:

* CART: is more advanced version which uses Gini impurity and do only two splits at each node, has inherent ability to work with both categorical as well as numerical values. In sklearn the implementation of the DT is though CART only. Though CART has option to select entropy as optimization metric.
* ID3 used entropy which as metric to decide best split where the number of splits for categorical are as many as the number of categories in a feature.

Let’s start with CART. (Classification and regression trees):

Let's go through the detailed calculation again, considering the binary splits for Feature C as described. We'll use the same dataset:

Check the tree viz. in notebook for better understanding.

| **ID** | **Feature A** | **Feature B** | **Feature C** | **Class** |
| --- | --- | --- | --- | --- |
| 1 | Cat1 | B1 | C1 | Yes |
| 2 | Cat1 | B1 | C2 | No |
| 3 | Cat2 | B2 | C1 | Yes |
| 4 | Cat1 | B2 | C1 | No |
| 5 | Cat2 | B1 | C3 | Yes |
| 6 | Cat2 | B2 | C2 | No |
| 7 | Cat1 | B1 | C1 | Yes |
| 8 | Cat2 | B2 | C3 | Yes |

Step 1: Root Node (Level 0)

Calculate Gini Impurity for the whole dataset:

Total instances: 8

Classes: Yes (5), No (3)

Gini = 1 - (5/8) ^2 - (3/8)^2 = 1 - 0.3906 - 0.1406 = 0.46875

Evaluate each feature for splitting:

Feature A:

Cat1: Yes (2), No (2) -> Gini = 1 - (2/4) ^2 - (2/4)^2 = 0.5

Cat2: Yes (3), No (1) -> Gini = 1 - (3/4) ^2 - (1/4)^2 = 0.375

Weighted Gini = (4/8) \*0.5 + (4/8)\*0.375 = 0.4375

Feature B:

B1: Yes (3), No (1) -> Gini = 0.375

B2: Yes (2), No (2) -> Gini = 0.5

Weighted Gini = (4/8) \*0.375 + (4/8)\*0.5 = 0.4375

Feature C:

C1 vs. not C1:

C1: Yes (3), No (1) -> Gini = 0.375

Not C1: Yes (2), No (2) -> Gini = 0.5

Weighted Gini = (4/8) \*0.375 + (4/8)\*0.5 = 0.437

C2 vs. not C2:

C2: Yes (0), No (2) -> Gini = 0

Not C2: Yes (5), No (1) -> Gini = 0.277

Weighted Gini = (2/8) \*0 + (6/8)\*0.2778 = 0.208

C3 vs. not C3:

C3: Yes (2), No (0) -> Gini = 0

Not C3: Yes (3), No (3) -> Gini = 0.5

Weighted Gini = (2/8) \*0 + (6/8)\*0.5 = 0.375

Best Split: Feature C with the split 'C2 vs. not C2' gives the lowest weighted Gini impurity (0.2083).

Resulting Tree (Level 1):

Root splits on Feature C:

C2: {2, 6} (Yes: 0, No: 2)

Not C2: {1, 3, 4, 5, 7, 8} (Yes: 5, No: 1)

Step 2: Split on Not C2 (Level 2)

Not C2 Group:

Gini = 0.2778 (from above)

Split on Feature A:

Cat1: Yes (2), No (1) -> Gini = 0.4444

Cat2: Yes (3), No (0) -> Gini = 0

Weighted Gini = (3/6) \*0.4444 + (3/6)\*0 = 0.2222

Split on Feature B:

B1: Yes (3), No (0) -> Gini = 0

B2: Yes (2), No (1) -> Gini = 0.4444

Weighted Gini = (3/6) \*0 + (3/6)\*0.4444 = 0.2222

Best Split for Not C2: Both Feature A and B give the same result; choose Feature A for simplicity.

Resulting Tree (Level 2):

C2: {2, 6} (No)

Not C2 -> Cat1: {1, 4, 7} (Yes: 2, No: 1)

Not C2 -> Cat2: {3, 5, 8} (Yes: 3, No: 0)

Step 3: Split on Cat1 under Not C2 (Level 3

Cat1 Group under Not C2:

Gini = 0.4444

Split on Feature B:

B1: Yes (2), No (0) -> Gini = 0

B2: Yes (0), No (1) -> Gini =

Weighted Gini = (2/3)\*0 + (1/3)\*0 = 0

Best Split for Cat1 under Not C2: Feature B.

Resulting Tree (Level 3):

C2: {2, 6} (No)

Not C2 -> Cat1 -> B1: {1, 7} (Yes)

Not C2 -> Cat1 -> B2: {4} (No)

Not C2 -> Cat2: {3, 5, 8} (Yes)

Step 4: No further splits needed for this example since we've reached leaves where no further beneficial split can be made given our small dataset.

Summary:

Here, we've gone through four levels of splitting, selecting the best binary split at each step based on Gini impurity.

The tree stops growing when no more splits would decrease impurity or when all instances in a node belong to the same class.

For numerical feature a middle feature is ordered in ascending order and a split is made based on every middle value between the two subsequent values.

Complete this CART’s rest info after viz in splits in notebook.

Feature importances.

**In actual implementation of the CART in normal decision tree, random forest, gradient boosting and in standard part of the XGBOOST there is nothing as categorical handling like shown above, This vanilla CART knows only the numerical understanding of the features, as encoded in preprocessing only and it process everything in similar manner of how it does the split on numerical column, so even if you label encode or category encode these features it treats then the same.**

**Only in LIGHTGBM and in XGBOOST’s with its experimental parameter we can have categorical features treated as categorical with not equal to operator. This is one of the biggest advantages of the LIGHTGBM.**

**Regression:**

for regression, the metric to judge the impurity is variance (MSE in this case, as the prediction for entire split/leaf is same which is mean). Even MAE can be used to judge the split.

**Feature importances in Decision Trees:**

Compute the impurity reduction like you calculate during forming decision trees. Do the normalized sum of this delta impurities for each node where the split is made by the feature for which you are calculating the importance. Total sum of this values will give you the feature importance of that value. Then this feature importances for each feature are normalized with respect to the total sum of the features importances, so that we have feature importances of all features which sums to 1.

For normalization in delta impurities for each node, do this with respect to all the datapoints (i.e. root node and not the immediate previous local root).

i.e. why first split after root node will have weight of 1. Cause Nnode and Ntotal is same. Which leads to a point that this feature will have the highest feature importance most of the time. But not always, mostly in complex datasets it’s not the case. If the first split has very high delta impurity and when the trees are shallow i.e. other feature do not get that much chance to get split on.

But in deeper trees where lot of feature get’s split on there might the case that weighted delta impurity in total might be highest for some other feature which is not used for fist split.

Formula can be referred in the notes and which is for each node, you sum it for the all the node splits for each feature.

Hyperparameters of the Decision Trees:

1. **Depth:**

Decide the maximum depth of the search or decides the max path of leaf node from the root node. Limits how deep trees can grow. Gives you the ability to make either simpler or complex tress depending on the depth. Which eventually controls the underfitting or overfitting. Obviously shallow trees will be simpler with underfitting whereas the deeper trees are more complex and captures more complex patterns and hence are leads to overfitting.

* If the one of the branches reaches the leaf node prior this does not mean we stop, branches are kept on forming where they can be formed till the specified depth.
* Default value is None i.e. max depth is till the leaf nodes are reached. For simpler trees i.e. underfitting trees value of depth is 3-10 and for deeper trees the same value is between 10-30, but obviously this entirely depends on the dataset.
* Tuning can be done with cross validation with hyperparameter tuning for different possible values and the graph between the accuracy (any metric) and the possible van be seen to how the metric is getting affected.

1. **Min samples split:**

* If the no. of samples in any of the nodes are greater than certain threshold limit then, no split is done in this case.
* Intuition is pretty much the same as above, about how it effects, higher the value of the threshold lower will be the depth.
* Absolute min value which can be set is 2, which is the default in the sklearn’s CART implementation.
* Common thumb rules, if the dataset is larger the min samples in a node allowed are bigger compared to the smaller dataset for obvious reasons.
* Actual range for smaller dataset is 2-5 or at max 5-10 for slightly medium dataset, whereas for larger dataset it is 10-20 and 20-50.
* This min samples can also be specified in terms of the percentages of the dataset size, for smaller datasets it is 0.5%, then you go on towards 1% and then to 5% for medium datasets. And for larger dataset value up to 10% can be tried. There is no special parameter defined, you do this calculation prior and then specified the inter value.

**[2, 10, 50, 100, 500] or [0.01, 0.05, 0.1])**

1. **Min samples leaf:**

* Like the min sample split, jus the difference is you put the criterion on the leaves which are getting after the split rather than putting criterion on nodes itself. So, potential leaves sizes are constrained rather than simply constraining the node itself.
* The min value here is 1 which is also the default value in CART implementation, the same value was 2 in case of the min sample split.
* Values are 1-5 for small dataset, 5-10 for medium and 10-20 and 20-50 for larger,
* In terms of percentages values are 0.5%-1% for smaller dataset, 1% - 2% for medium and 2%-5%, 2%-10% for larger dataset.
* Difference between the min samples split and min samples leaves, first one i.e. min sample split tells that do I have min samples in a node to even consider split whereas min samples leaves say will be potential new leaves will have enough samples, node in this case should have at leas samples equal to 2 \* min sample leaves.

1. **Max number of features:**

Max number of features which are used for splitting each split is restricted to this number and these features are randomly selected, if the unique features remained or max features specified are higher than available features for split then in that case all the available features are used. Why is it helpful, it adds that level of randomness which makes the model more normalized and reducing the overfitting.

Max features are specified as [None, "sqrt", "log2", 0.1, 0.5, 0.8].

1. **Max leaf nodes** Counts the max number of leaf nodes in the tree and based on this criterion decides the growth/pruning of the tree. But it only concerns the leaf nodes and not the internal nodes. Values possibly set are None, 5-10,20-50 depending on the data.
2. **Min impurity reduction:** If the impurity reduction in next split is not above the specific threshold, then in that case the split is not done.

* Possible values are: [0.0, 0.001, 0.01, 0.05, 0.1]. Where 0.0 mean no stopping criterion and the 0.001 to 0.01 is for moderate criterion whereas the 0.05 to 0.1 strong pruning with very high normalization.

1. **Criterion:** For classification, Entropy/Gini Impurity. Gini is bit fast as no log whereas entropy favours the balanced dataset. Though Hadley any changes for classification whereas for the regression, we have variance mainly but MAE and poissions can be used.
2. CCP alpha: How does CCP alpha works? How does it prune the trees, how does it do the post pruning?

**Make sure you understand the difference between this different hyperparameters.**

## **Ensemble Learning:**

Ensemble techniques use principle of wisdom of crowd to correctly identify patterns from the crowd. It involves predicting the same value by multiple models i.e. by crowd.

There are multiple ways in which you create this crowd or multiple models. The fundamental assumption while creating this model is every model should be totally independent from each other.

**Why do ensemble work?**

Ensemble techniques work because they exploit the statistical, computational, and representational principles to improve predictive performance.

1. Error reduction through Aggregation and Diversity.

Let’s answer it later.

Based on how it creates different models, there are four types of ensemble techniques:

Different models are created such that they do not have correlation between them.

### Voting:

Most intuitive of the ensemble techniques where different models are trained on same data. That is how it creates the decorrelated models. There is also option of sampling the data, just to create additional layer of the independency between the models.

Very basic and democratic way to include the different voters to predict the output by aggregation.

**Basic concepts:**

Usually the base learners (different models, or same with different parameters) are trained on the same data usually. And these outputs are combined by aggregation or majority count.

Types of the Voting classifiers:

**Hard Voting:** Each model makes the single definitive prediction, and output is the major count of one of this prediction.

**Soft Voting (relevant to Classification):** Each model returns the probabilities, and these probabilities are aggregated for prediction, this is the main advantages of the soft margin as hard margin is not able to capture the essence of higher likelihood of its prediction. 51% chance of prediction is same as 99%. But the disadvantage is each model should be able to return the probability.

Even the weights could be assigned to each of the model before aggregation it has higher impact of the specific base learner, though it is not stacking cause stacking has one more layer of the learning.

Even sampling is possible in some cases.

Assumptions and requirements:

**Diversity:** All the voters should be highly diverse; cause voting works best when each model makes different kind of mistake on data.

**Independence:** The errors made by the models should be highly uncorrelated.

**Competency:** Each model should be at least slightly better than random guessing, then only combining lot of them.

**Why does it work?**

**Errors are cancelled out,** cause all the models do not make same mistake,

**Strength amplifies,** Different models capture different patterns in the data.

**Implementation, Voting Classifier ().**

### Stacking:

Multiple models are combined such a way that their output is given to another model which weights the output of all the models and then those weighted outputs are combined/aggregated.

Stacking is an advanced ensemble learning method where the models (base learners) are combined by training the meta model to learn how to merge this different model together.

The only difference between the voting and stacking is how you combine this base learner’s prediction.

**How is the meta model developed or works?**

* Train your base model.
* Generate the OOF prediction for each meta model with cross validation for each base model.
* Now this OOF predictions of each base learners becomes the training data, and the target is the original target feature.
* So, what happens is for each base learner you get as prediction for each, and every training data points through CV OOF. And now you as many predictions for each training data point as number of base learners used. And the problem to solve now is while aggregative this prediction which of these prediction to weight what. So basically, you want to find the relation/impact between the prediction (level 1 prediction) by each model and actual target.
* So, for the same a new model is trained called as meta model, which finds how relate this prediction of each base learner and the actual target. Which one to what weight for what kind of original data.
* Suppose the meta model is linear then meta model learns, how to weigh base learners or how to combine them unlike voting where you directly aggregate, in linear model specifically finding the weight of aggregation means finding the features importances of the features in meta model training. For non-liner models like random forest or GD, non-linear patterns are found out like, for certain kind of data certain base learner works best then it weighs that base learner higher importances and makes prediction likewise, then more certain ranges where one model could be doing fine but the other may not be doing any good.
* So, the actual process is you passed a datapoint though each base learner then passed those prediction of the same point though meta model and finally meta models’ prediction is the real model.

How does training data looks like for meta model:

* Let’s suppose we have 3 base learners and 100 datapoints, for 3 base learners with hard predictions, data looks like this for 20% test set.

80 X 3 – input data

80 X 1 – Target variable, original training set.

* For soft margin with multiple probabilities, its columns increase based on which probabilities you are increasing.

Advantages:

* Captures complex relations between the base learners, i.e. non-linear weights unlike voting.
* Outperforms voting usually.
* Adaptability to various scenarios.

Disadvantages:

* Risk of overfitting.
* Complexity is high.

Practical tips:

* Base models: use wide variety, or diverse models.
* Use simple meta model to avoid overfitting.
* StackingClassifer () is the implementation in sklearn.

### Bagging. (Bootstrap + Aggregation)

Ensemble learning techniques introduced by the Leo Breiman in 1996. The core idea is to reduce the variance of the model’s prediction by training the same model on multiple subsets of the data and then aggregating it.

How does it work?

Bootstrap sampling: sample the data with replacement, one sample for each model.

Train base learner: train single kind of base learner on this model.

Aggregate predictions: aggregate the prediction of each base learner.

**Intuition:**

Very high variance models, like DT’s are used as base learner which overfit on the sample data, mining all the intricate patterns in those samples, but the prediction/trees for each sample could be totally different as this overfit model are very sensitive to variance. Means, each base learner with its sample data digs deep and finds new patterns but this DTs are different for each sample.

Hence on averaging this common pattern gets highlighted whereas the errors get overshadowed.

Viz a one chunk of the data creates one decision, and another chunk creates another one. Like this each tree zigzag the noise in the data.

**Why does bagging works:**

* High variance models.
* Averaging reduces the variance.
* Bias stays roughly the same.

**Key characteristics of the bagging:**

* Parallelizable
* No overfitting Penalty
* Best with unstable learners.

**Advantages:**

* Reduces variance.
* Robust, improves stability.

**Disadvantages:**

* Does not reduce bias.
* Computationally heavy
* Single DT has better interpretability.
* Correlation between the points, should not be high correlation between the samples, could be the issue in smaller datasets.

**Why is bias called bias?**

Bias indicates the bias nature of the model to predict values towards its ideal simplistic nature, like linear regression is bias towards linear relationships irrespective of the real nature of the data. Naïve bayes is bias towards the feature independency. This is the start of the origin of the concept, bias indicates how good model can fit on intricate patterns with no restrictive assumption. High bias means high inclination of the model towards the restrictive assumption, very simplistic model like linear regression. High bias means models simple model which makes the mistakes on the training data itself. Whereas the low bias means model does not incline towards the specific restrictive assumption.

**Bootstrapping:**

Sampling with replacement is done in bagging. Usually each sample (goes to each estimator) is sample with replacement and usually the size of original dataset only. With replacement creates bit more of the randomness though it has repeated rows.

With replacement and without replacement?

With replacement means on drawing a sample you replace the original pool with same sample, so the next picking would have same probability of picking of each datapoint as for first drawing of the sample.

Without replacement after each draw, you do not replace the removed datapoint with anything. So, for each subsequent draw the probabilities changes.

For N datapoints if you draw a sample of N datapoint with replacement then out of the 100 unique datapoints there would 63.4 % unique datapoints would be there and remaining would be repeated.

Max\_samples in the sklearn decides the sample size, default value is ‘None’ means equal to pool size means equal to the original dataset.

Bootstrap is another parameter which decides the sampling with and without replacement. Default is True which means sampling with replacement. And make false, you change the max\_samples as well, this is the general way to use it else it will just give same data to all the trees.

**Types of bootstrapping:**

1. **Bootstrap bagging:**

Sampling with replacement, very typical case bagging with sample size same as initial data.

1. **Subsampling Bagging:**

Without replacement bagging with sample size lower than initial data. i.e. if the dataset has 100 datapoints then out of this 100 datapoints, each sample will have 80 datapoints only but without replacements.

1. **Pasting (disjoint sampling):** like above subsampling bagging just the sample size is divided such that there will be no overlap of the datapoints across all the samples. Totally unique and disjoint samples.
2. **Random subspace:** sampling on the columns, always without replacement.
3. **Random patches**: combining bootstrapping (sampling with replacement) with random subspace (without replacements)

**What is the difference between the bagging with DT and Random Forest?**

For subset sampling random forest uses, normal bootstrap bagging (Row sampling with replacement), but it does column sampling on each node for split. i.e. column sampling happens at the tree formation stage. Th

No. of columns selected, for classification it is square root of no. of columns whereas for regression it is no. of columns / 3.

This number is higher for regression cause the regression can handle higher correlation between the models, as the final output gets averaged. But the same is not possible to an extent as it is possible on regression for classification cause outputs do not get averaged, here value count wins.

Technically higher number of features cause lower decorrelation, so we desire lower no. of features, yet we want as much info as possible with lower noise means max features, so we want max decorrelation with min features.

In classification the outputs are aggregated by majority count means the extra decorrelation caused by having more features for extra info and lower noise cannot be adjusted by aggregating the outputs, which is very much possible by the aggregating the regression outputs, that’s why we allow higher features (M/3) cause regression can handle bit for the decorrelation, because of its ability to handle the decorrelation in the output aggregation.

**OOB score:** sampling with replacement leaves about 33.2 points out of the sample, so the prediction is done on this non-used datapoints, to get the evaluation idea.

How it is done, take each of the datapoint one by one from the original sample and then check in which of the sample it was not selected, then use those respective tresses to predict the output of this datapoint and aggregated those outputs.

It helps you not use validation set. It is like test set, i.e. in terms of data leakage and fundamental calculation method, just the trees used to predict the point are not all, only the subset of the trees is used, which obviously may not be the best estimate of the datapoint.

**Why it is useful?**

* No need of validation set.
* Hyperparameter tuning.
* Efficiency (no extra calculation)

When you do the oob\_score = True then the OOB score during the training is shown.

Hyperparameters of the random forest

Extra-Tree Classifier: splits do not use impurity metric to judge which metric is split is better, random splits for better decorrelation.

### Boosting:

Why boosting is most of the time better than bagging:

* Bagging relies heavily on the base learner ability to capture the patterns, i.e. it relies on base learners being extremely strong, or their inherent ability to be extremely strong. If the base learners are not strong then bagging averages out this weak prediction for variance reduction (Generalization).
* Boosting focus on the extreme/outlier, different/difficult to predict points whereas as the bagging tries to average them out in aggregation stage. Whereas boosting explicitly focuses on these points either with weights or in residual fitting parts itself (next models training). Boosting focuses on the difficult problems(points/patterns) specifically whereas the bagging does not.
* Boosting can optimize any complex differential function, same cannot done in bagging.
* Though bagging is better when there is a risk of overfitting whereas the boosting overfits very normally as it tries to add the sequential models to reduce the any errors, where bagging does reduce the bias by aggregation significantly.

Baggins is faster as it can be parallelly performed whereas the boosting is sequential adding of the models and hence slow.

#### **How boosting works:**

It works completely in opposite way than how does bagging works, in boosting you start with very simple model with very high bias and low variance model, and then tries to improve the bias by adding the weighted sequential models, where each new added model trains on the residuals of the previous model and tries to reduce this residual as much as it can.

These models are weighted based on the relative performance in reducing the residuals (relative error reduction with respect to the whatever residual is left to it, cause the absolute residual reductions are always higher for earlier models.

Weights are lesser than 1 means model required to converge the solution are more than what required without the weights as with weights you only take/add certain proportion of the residual reduced.

#### **Gradient boosting algorithms:**

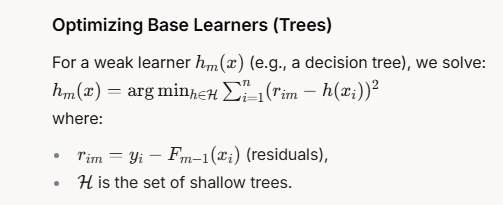
Now we know how boosting works? Let’s see how everything is implemented.

The real algorithm stepwise:

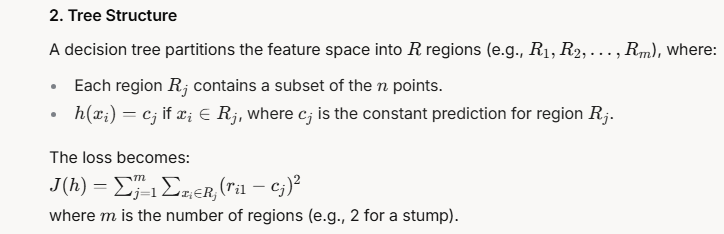
* + First initial model. The initial model is always the constant model (function) which tries to optimize the loss between the actual target values and the model prediction.
  + For the actual formula and calculation with MSE as a loss function see the notes, which involves the defining the loss function for entire data with respect to the initial function(model) and optimizing this loss function with respect to this initial function which is to be found (decided by solving this equation only) you get the initial function. How do you optimize it, by gradient descent i.e. derivative minima optimization. For the MSE as loss function, we get initial model is mean. We directly add the mean or initial model as it is not the improvement step in correction of the loss as it is the first general prediction.
  + Once the first model is calculated then we add the weak learners whose aim is to predict the residual (Ri) between the initial model and the target variables. Weak learners DTs takes the same Input X and target as the residual (Ri) and try to predict best residual (remaining part of the prediction) by finding the most optimized function/model/weak learner. But unfortunately, the constraint on the weak learners is that we mostly have DTs as weak learners and this multiparameter models are not differential, so we cannot directly use the gradient descent as we used in the case of the initial model.
  + There is as such requirement of only having trees as the weak learners, there are only two requirements as such, the weak learners should be weak, slightly better than random guessing and second criterion is that these learners should be calculated from the x (i.e. input) so that they can be input/data specific. But why do we select decision trees, due to specific advantages:
    - Natural fit to gradient approximation: Trees partitions the data into different regions and the prediction is done based off the region mean, this piece wise constant fit aligns with the approximation of the gradient direction simple interpretable way.
    - Non-linearity and flexibility: can efficiently capture non-linear pattern yet can be made simple(weak) without feature engineering
    - Weakness control: High and direct easy weakness control.
    - Scalability:
* Other options of the base learners:
  + Linear model
  + NN
  + Splines
  + KNN etc.

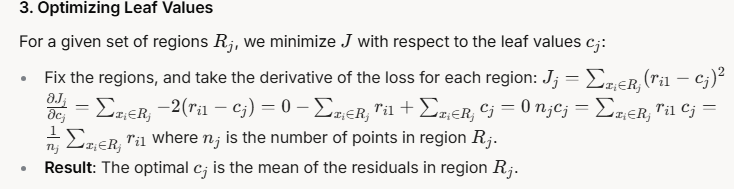
But we do select the DTs, let’s see how we find out the best DTs or most optimized DTs in terms of right direction and right magnitude to reduce the residual given by the previous step. Anyways that’s the job of the DT (base learner). If they were differentiable, we could have been able to use gradient descent as it was used for initial model, but here we do the optimization in a different manner, which is explained like this.

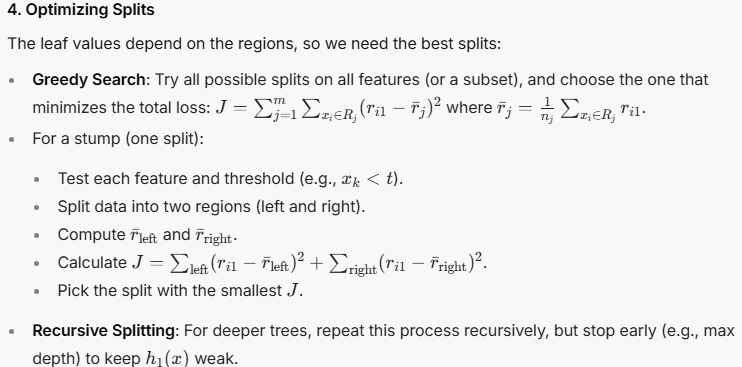
Optimization function with loss is given as this, where base learner is selected from the pool of base learner H, where the best learner which minimizes the loss (residual).



Finding the best “h” is a two-step process first finding way to aggregate the regions points and second the optimum number of regions or leaves.







This is how the best most optimum DT (base learner is selected) with mean as aggregation (regression) and best no. of split for regions are selected.

* So, what we do is we find the most optimum h1 and then take only the weighted proportion of it, which is analogous to taking step in the right direction but only the limited proportionate step?
* Weak learner is already a shallow tree and then we again add the weighted portion of it?

**Weakness Alone Isn’t Enough**:

* A shallow base learner is intentionally simple to avoid overfitting—it captures only major patterns in residual​. But even this simple correction might be too aggressive if added fully.
* Example: If base learner overcorrects a noisy pattern (due to its fit-on residual), adding it fully could harm generalization.
* **η Adds Fine Control**:
* Iterative **Refinement**:
* The learning rate η \eta η acts like a “dimmer switch” on h1(x) h\_1(x) h1​(x)’s contribution. Even though h1(x) h\_1(x) h1​(x) is weak, scaling it down ensures we don’t trust it too much in one go. Analogy: h1(x) h\_1(x) h1​(x) is a rough sketch of the correction direction; η \eta η decides how much of that sketch we paint onto the canvas at once. Gradient boosting builds a strong model by summing many weak learners: Shallow trees ensure each hm(x) h\_m(x) hm​(x) is simple; η \eta η ensures each step is small, allowing the ensemble to refine gradually.
  + Balancing Bias and Variance: Shallow model High bias (underfits residual​), low variance (stable).
  + Full step (η=1 \eta = 1 η=1): Risks increasing variance by relying too much on one weak fit.
  + Small η \eta η: Keeps variance low, reduces bias slowly over many iterations.

so, what we do is we find the most optimum h1 and then take only the weighted proportion of it, which is analogous to taking step in the right direction but only the limited proportionate step.

Trees are kept shallow in learning phase by constraining the overfitting parameters.

* **Numerical calculations in the notes.**

#### **For classification (Difference specific to classification):**

Gradient boosting algorithm for classification works just fine, like it works for regression except loss function it uses. In classification the loss function used is log loss error and then onwards the subsequent steps are similar, means the first constant model is developed with log loss error, and it comes out to be log of odds. Log of odds, log (probability of yes/ probability of no), this yes and no are prior probabilities, i.e. count of yes/total and count of no/total

Then for the base learner, optimized base learners are calculated, the method to get the optimized DT is same, just the loss function is different so, the way to calculate the log of odds for a node or rather way to aggregate in classification to find the log of odd of the node (cause DT gives log of odd, and not the probability. Log of odds is calculated as sum of residuals to the sum of variances (or sum of their uncertainties).

And this log of odds then changed to probabilities and then compared to get the residual,

The model in gradient boosting for classification does not directly predict the residual as the difference of probabilities. Instead, it predicts adjustments to the log-odds, which are then transformed into probability changes, though it takes difference of the probabilities as the input.

#### **Why gradient boosting is a gradient descent in a function space?**

Gradient boosting mirrors gradient descent by iteratively stepping in the direction that most reduces the loss, but it does so by adding functions rather than tweaking a finite set of parameters. The weak learners act as approximations to the steepest descent direction in this abstract function space, making the process both powerful and flexible for a wide range of problems.

### XGBOOST:

Curvature: slope is the rate of change of the function whereas the curvature is the rate of change of the slope itself. Tells if the slope is changing at the point or no and the magnitude of it.

High curvature (large (x)): The function bends sharply, so small steps are safer. Low curvature (small (x)): The function is flatter, allowing larger steps. Cause for higher curvatures the values change rapidly, and could be rapidly overshoot, so for higher curvature the steps are little.

This curvature is represented by hessian matrix, and the xgboosts optimization is based off this second order function optimization unlike the gradient boosting.

#### **Core idea of XGBOOST:**

* **Speed and efficiency:** use parallelization, tree pruning and hardware optimization.
* **Regularization:** uses L1 and L2 regularization penalties to prevent overfitting.
* **Flexibility:** supports custom functions and handles missing data natively.
* **Scaling:** Can be used on large dataset.

#### **Mathematical foundation:**

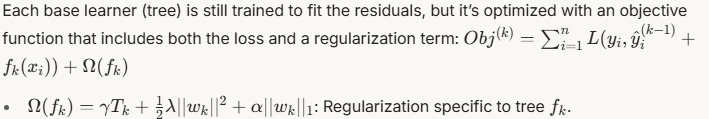
1. **Regularization in each base learner:**

The main mathematical difference in gradient boosting and XGboost is that XGBOOST add regularization parameter along with loss function and the combine optimization is performed.

Optimization function for the base learner to add is combination of the loss function + Regularization parameter. Means the optimization for the base learner (i.e. finding most optimum base learner) is based of loss function + regularization parameter, means regularized base learner is added.

So, that’s the main difference between the GD and XGB, that regularized base learners are added in the XBG whereas in GB non regularized base learners are added.

Objective function given below with regularization function



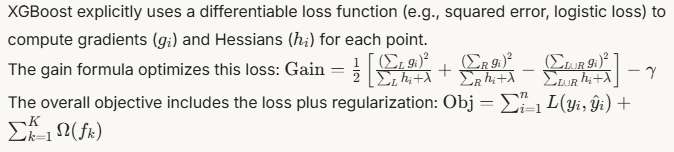
1. **Second- Order Gradient (Hessian) in optimization:** It also uses with hessian derivatives (curvature) for finding the optimum solution with Taylor series expansion (for faster approximation of the solution).
2. **Customizable loss function:**

Unlike the traditional GD, XGBOOST can handle any customize loss function which is differentiable of first and second order. This flexibility is due to the second order Tylor approximation method.

1. **Weighted quantile sketch for split finding:**

Unlike traditional DT which finds the best split after calculating all the possible splits of the numerical values of all the features, XGBOOST uses the smart method of dividing the features values into a quantile buckets, and these buckets are formed on the cumulative sum of the hessian values. The number of the buckets are dependent on the sketch eps, where the number of buckets is given as 1/sketch\_eps. Sketch\_eps determine the allowable absolute error in terms of the cumulative values in quantile approximation. Then the splits are done only for these buckets and the best one with the max gain is selected. The hessian is calculated for each value of each feature by hessian formula of Pi(1-Pi) where Pi is by sigmoid (1/1+e\*i). numerical process involves calculating he total hessian then dividing the total hessian by number of buckets to get the bucket limits, then grouped the points falling into each bucket and find the gain for each bucket. Finding gain here is nothing but finding the reduction in loss. Which does not involve the proportionate taking difference between the root entropy or loss and sum of left + right after split, the weight proportionate is already there in the gain of right and in hessian values itself. The formula for the gain is bit different with ½ at the front.

The **main idea** behind the concept is XGBoost splits by grouping points into buckets based on cumulative Hessian weights, which highlight regions of significant curvature. It then selects the split between these groups that maximizes gain, ensuring the greatest reduction in the loss function.



1. **Sparsity-aware split finding:**

Ability to handle nan values and sparse in sparse matrix, so what happens is that nan values are treated such a that these values are put once to the left and then to the right whichever gets a better score is considered. In normal dense matrix an only nan values are considered for the above approach unless specifically specified about the zeros (converted to nan earlier only). For the sparse matrix the sparse as well as the nan values are treated similar and the score for both left and right with sending all sparse values (which includes sparse zeros as well as nan) is calculated.

1. **Tree pruning with gain threshold**
2. **Customizable Tree construction:**
3. **Hardware differences: use of GPU, Multi- core CPU’s, catch aware access, out of core learning,**
4. **Software differences: Built in Cross-validation, Early stopping.**

**Greedy search:** local optimum at each step and that is the global optimum is achieved no backtracking to explore all the combinations. NLP or LLM’s in general.

Brute force (exhaustive search): finds the global optimum with the trying all the combinations. Like hyperparameter with grid search CV

### LIGHTGBM

Five differences

* **Leaf split and not the stage level split.**
* **Histogram based learning.**
* **Categorical data handling based of importance to the accuracy.**
* **EFB (feature bundling).**
* **GOSS (Gradient base one side sampling)**

#### **Leaf base split:**

Unlike the normal DT split which is the case in XGBOOST, in LGB the split is not level base but rather leaf base. In leaf base split not each node of the level is split rather a node with which reduces the most entropy/variance is use for split, likewise all the leaves till the point are used irrespective of the level to judge which single leave reduces the impurity and this single leave is split.

Very unbalanced, non-symmetric tree compared to balance symmetric tree made by the level base split.

* E.g. of Level base split

Root

├── Left (Level 1)

│ ├── Left (Level 2)

│ └── Right (Level 2)

└── Right (Level 1)

├── Left (Level 2)

└── Right (Level 2)

Leaf wise split:

**Root**

**├── Left**

**│ ├── Left-Left**

**│ └── Left-Right**

**└── Right**

**Now for next split there are three leave are available left-left, left-right and right chose one to split.**

**Pros:** The main advantage is, it converges faster compared to the level base split, though there is not inherent advantage in term of split performance.

Though it can be overfit on the noise, so there is overfitting chance.

#### **Histogram base learning:**

The numerical variables are used divided into histograms, and the boundaries of those histograms are used for spit. i.e. only check the splits for the histogram boundaries.

#### **Categorical data handling:**

categorical data in the LIGHTGBM is handled without any encoding, categorical data splits are made just like the splits of simple explanation of the for categorical features in CART explanation. i.e. treat each category vs other with not equal to operator for all the possible combinations (not all technically) and then check the gain for each split. This is not what happens in the usual vanilla CART, which is used in the DT, ensemble like RF, GB, XGB etc. Each category is use for histogram like categorical feature treatment just the difference is it creates no bins rather uses each category as a bin and then computes the gradient and hessian for each category or bar calculates the gain based of these values. (just the difference from numerical is it uses categories for bins rather than step values of continuous variable). There is something called as **max\_cat\_to\_onehot** which decides whether the LGBs special technique is use for categorical feature handling or one hot encoding is to be used. This above parameter decides the max categories up to which the OHE is used, above this number of categories special handling technique of the LGB is used which tries all the possible combinations of the feature for split. For lower number of categories like below 4 (which is the default value of the parameter) OHE is used because it works better and is very simple and intuitive, higher number of OHE cause complexities and are far more computationally and memory wise difficult which is not the case for the lower number of the OHE.

Though again even for the higher cardinality it’s not like all the combinations are tried, cause for n=32 i.e. 32 categories in a feature (2\*31) -1 which is 2.1 billion combinations of the splits, not possible to do so there are few heuristic techniques it used for split. A parameter **max\_cat\_threshold** is use which decides the number of splits to decide int this case, default value is 32, which means only the 32 splits are made from the millions and billions of possible combinations based of the n. One more parameter here is after which value of N, does LGM starts to reduce the no. of splits out of all the possible ones, when the no. of possible split combinations gets beyond the 32 like in case of the 6 (63 only the 32 of these 63 will be check, obviously the best 63 would be selected by heuristic methods. For n=5, the no. of possible combinations are 31 so all these would be checked.

how do you select this best 32 split.

* One approach is one vs rest
* Top1 vs rest, (Top1, Top2 vs rest), (Top1, Top2, Top3 vs rest) this approach can be used.

#### **Exclusive feature bundling:**

The feature is bundled together which are mutually exclusive with their values. Like the mutually exclusive OHE values of the two independent features. Suppose we have feature called as the colours with 3 colour categories (Red, Blue, Green) and this feature is encoded with OHE then in that case we have 3 features (without dropping first) which perfectly mutually exclusive and these features can be bundles together. So, you process only one feature onwards in the data which is obviously computationally and memory wise efficient.

How bundling happens: For each datapoint the there will be only non-zero value out of these 3 features. That value is the value of these new feature for that row, if you just put the values as it is, then in that case the entire feature as single value 1, so that is why offset is used for each feature before putting into the bundled feature, or the index of column can also be put into there which keep track of the feature form this value arise which helps to track back to the original value with corresponding row and feature and that’s the ultimate goal, cause ultimately there values are tracked to the original feature and then then based on the corresponding target column values gain values are calculated. Offsetting is done such away,

Suppose features looks like

Red [**1**,0,0]

Blue [0,**1**,0]

Green [0,0,**1**]

Then offset for the blue would be 10 and for the Green would be 20 then after offsetting features looks like,

Red [**1**,0,0]

Blue [0,**11,**0]

Green [0,0,**21**]

The offset calculation is as such that it covers the entire ranges of the individual features.

And so does histogram bin calculation is based on this offset values so that single feature points get into one bin.

In case of indexing just the index of the column is mentioned.

How are this BF gets processed onward, there is only one histogram made for the bundled feature, based on this histogram and corresponding aggregates of the target, gain is calculated just like the normal XGBOOST, and the splitting is done. The main advantage of using the EFB is, it creates a single histogram which initially would have required 3. So EFB makes LIGHTGBM computationally efficient on the expense of the little accuracy if the conflict happens i.e. features are not totally mutually exclusive.

How does it handle the conflict, there is a parameter which decides the degree of the conflict, which is allowable, how rate of conflict is measured? it is nothing but the proportionate of the non-mutually exclusive rows to the total rows. Which feature values gets preference? the exact algorithm is not known but most likely the feature values with higher offset get’s preference, then the question arise how you decide the order of the features, ordering could be based off the

There are few nuances to cover about it,

First the feature preprocessing it does to identify the potential feature for bundling, it checks the sparsity ratios of the feature then then runs the patterns to check the mutually exclusive nature of the features.

#### **GOSS (Gradient base one side sampling):**

**It is** a technique in LGB used to improve the efficiency of the training especially on the large datasets. It makes the training on large dataset in terms of the computational and memory aspect for more efficient on small compensation of the accuracy.

**Core Idea:**  The core ideas are you select most of the points with high gradient but only the certain proportionate of the remaining low gradient with proportionate weightage is selected, the training is done on this small dataset which makes the training more efficient with all exact gradient of top gradient points and weighted aggregated datapoints of this sample datapoints.

How do you do it, sort the data base of the gradient, select the top 20% of the data points and from the remaining 80% only select the certain percentage(10%).i.e. if you have 1000 datapoints then you would have (200 + 80) i.e. 280 datapoints to train on, just the 80 datapoints would have weightage of (1-b)/a, 8 in above case, as to have the representation of the missed out points. So, basically for each split if which could be a mixture of points from any of the set, just while doing node calculation (gain etc.) if the datapoint belongs to the sampled set then it would get multiplied with the weight. High gradient points are prioritised and that is why they not sampled.

**Nan value treatment is same as XGBOOST.**

**Which one to use when gradient boosting, XGBOOST and LIGHTGBM?**

LightGBM is very handy on large data as it is faster and memory wise with good accuracy with better categorical feature support. Whereas the XGBOOST is slower with high memory usage but often with highest accuracy especially on smaller datasets when properly tuned.

#### **What is Parametric and non-parametric models?**

Parametric models assume fixed functional form defined by a finite set of parameters, regardless data size, these models simplify learning by imposing structure but may underfit if assumptions are incorrect.

**Fixed complexity,** fixed number of parameters with defined structure makes it far less complex.

**Efficiency** as resources is lower, i.e. data requires is less with fewer and smarter computation.

**Interpretability** if high as the assumed structure is fixed and defined with fix number of parameters.

**Nonflexible with underfitting risk (as the structure is very rigidly defined).**

Whereas the **non-parametric models** adapt to the structure of the data, with flexibility increasing as data grows. They make minimal assumptions but requires more resources (data, computation).

**Data driven**

**Flexible**

**Robustness.**

**Computational cost with overfitting risk and less interpretable.**

# **Unsupervised machine learning:**

* 1. **Clustering algorithms.**
     1. KMEANS:

Working you can see in the book, the important thing is parameters,

How to decide number of clusters with WCSS: By elbow curve.

Init: **How to assign the initial points,**

**‘K-means++’:** first onwards the centroids are assigned as function of the square of the distance from the previous centroid. The intuition is to pick the point far away in different cluster. For next centroid, calculate the distance of each point from the centroid, and calculate the probability of each point as the square of the distance / (total sum of square distance) then based of these individual probabilities select the points, which are mostly likely to be the points which are furthest away. Using these techniques lead to better initialization which leads to the faster convergence. And avoids local minima by starting away from the original point.

**‘Random’**: centroids are assigned randomly.

**Specify manually**

**N\_init:** Number of times K-Means algorithm runs with the whichever centroid initialization technique specified. And the iteration with lowest WCSS (inertia) is selected.

Max\_iter: max iteration for convergence.

* + 1. **DBSCAN:** working is quite bit simple and explain clearly in the notes, how to select the best values of the min point and eps, for min points the best starting point based of the heuristic approach is min point = 2 \* (dimensions)

For the eps: calculate the Kth distance value for each datapoint, i.e. the Kth nearest element distance to each datapoint, arrange this distance in ascending order, and the location of elbow is the eps.

### Agglomerative clustering:

Look in the notes mentioned properly, just see how are dendrograms are used for the best cluster selection.

### Silhouette score:

From notes.

Where and how to use it, for K-Means clustering try all the K/N whichever K gives highest silhouette is chosen.

Silhouette is used instead of WCSS, cause silhouette is based on the cohesion as well as separation.

WCSS which uses only uses cohesion

* 1. Dimensionality reduction.

### PCA

### T-SNE

# **NLP**

**Things to cover:**

* **CATBOOST.**

**NLP**

1. GIT:

Version control software which is used to keep the different versions of the software, it takes the snapshots of the software and maintains it.

There are two types:

1. Distributed
2. Centralized

Advantages:

* Bug fixing
* Version control
* Non-linear development
* Collaborative development

Example to understand is google sheet which takes a snapshot after updates and gives you the ability to go back to any of the version.

Git-Bash – command line tool.

Git – GUI tool for the same thing.

Main Hierarchy:

1. Python.
2. VS Code, PyCharm, GIT, Streamlit. (More practical)
3. Stats
4. ML
5. DL
6. GEN AI

Page settings:

Margines: 2 X 2

Indentations

0.25-0.50-0.75-0.75

Spacing vertical: before after is 3-6, Encase of both specifying higher one is taken.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**On the GO learning:**

**ML Learnings:**

* 1. Label encoder cannot be used along with the column transformer for non-target columns, one hot and other encoding techniques are designed to work with column transformer. Better to use loops for the same.
  2. Column transformer changes the order of the columns with columns which are passthrough as added at the end, but pipeline handles it on its own.
  3. Feature importances of the columns where we use One Hot Encoding is calculated by aggregating the scores for the one hot encoded column.

OHE without dropping the first variable cause the dummy variable trap leading to the multicollinearity.

For classification problems metrics the worst model is model with accuracy lower than 50% and for imbalanced dataset accuracy should be higher than the proportionate of higher-class cause of the obvious reason of selecting just one class for all.

While cross validation model selection always gets a comparative table with required metrics as well as the standard deviation between the samples for the metric.

**HistGradientBoosting** handle the lot of missing value data, uses histogram bins which Ables it to handle missing values and allows missing values.

So, you can use it for a data with lot of null data even without filling null values.

Numerical data transformations.

Scaling and normalization in detail intuition of the data. How to change the skew and what happens when you do so.

Removing the skew from the data improves the performance of the parametric models which relies on the distribution of the data. Even tree-based model helps as find out why?

Just learn the entire topic in detail why do we skewness helps and how to remove it?

* Learn about the Data validation set how is it different from the test set and why do we need it, even after we have the test set. Why judging the accuracy on the dynamic test set generated in the cross validation is not enough and why do we need validation set?
* Make sure you understand how validation set is used.
* On regression problem check the correlation between the features as well , if many of the features have high correlation means there is high scope for the PCA.

**Data Frame:**

Mixed variable column is object in the data frame, where each element is stored as its own dtype which can be seen as apply type ()

There are both the function and methods available for the many of the panda’s data frame/series methods. Where each has their difference significance. Functions can be used even for the non-data frame objects like individual values as well, which comes handy in functions and all.

But not all the methods have functions.

Gradient Descent: For batch gradient descent the values are updated by the mean of derivatives with respect to that features for the entire batch.

Things to look:

Abhishek thakur ML channel,

The list of 10 Kaggle notebooks.

Learning from the notebooks:

Every notebook you read cites the learning here and save the notebook with specific finder to that notebook. Then You can choose to put those learnings particularly in those topics.

* If the correlation between the independent features is high then there is high chance that PCA will help, so make sure to always change it.

Kaggle notebooks:

1. Backpack price prediction:

* Description(include= all ) for non-numerical values as well.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*888888888888888888888888888888888888888888888888888888888888888\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Statistics:**

Variance of the sample is lower compared to the variance of the population in most of the cases, cause the chances of having outliers in the sample is low as the chances of having the general most occurring element is high.

Why use n-1 instead of n while calculating the sample variance,

1. **Biased sum of squared difference:**

The sum of square distances from sample mean are always lower (underestimated) compared to the sum of square distances from the population mean, cause the mean calculated in the samples is far closer to the points (major chunk of the points) than the actual mean. Meaning the sample mean is not the exact correct mean it is the biased

most probable (of similar kind) in the dataset as the most probable points will have higher probability to be in the sample in the first place, cause the extreme outlier points will have lower probability to be in the sample. So, point is sample mean is more tends towards the this highly occurring points and won’t have any outlier extreme essence in it. So, if you calculate the squared sum of the differences by this sample mean it will be smaller compared to if we calculate by population mean. Let’s take an example for calculation, suppose we have sample mean as well the population mean, now let’s pick two random points form the population and calculate the sum of squared differences from the both the means, the value of sum of squared differences from the sample mean is always lower compared, compared to the value form the population mean. So, the squared sum of the differences is biased cause of the mean in accuracies.

1. **Degree of freedom:**

So, the above reason tells you why the sample variance does is not same as the population variances and tells you it is biased and underestimated, so to reduce the bias, the denominator is reduced, DOF tells you by what amount the denominator to reduce by.

Calculating the variance with sample mean, reduces the DOF by 1 as, calculating the mean already makes the use of one value redundant as it can be calculated by mean and rest available values, reducing the independent information value to n-1.

**Understand why is it that while calculating the percentiles we do use n+1 rather than using the “n” only.**

Where to use the cumulative probability function, in cases where the probability is compared against the known threshold, like financial risk management, calculating the probability of the following below or above a certain threshold.

**Percentile:**

Percentile location/value (index form percentile number) is calculated as that proportionate position after ordering the numbers, i.e. 75th proportionate position out of all based on the order. 0.75\* N, this the index, get the number belonging to this index.

Percentile number (i.e. what percentile from the location/index or the value), number values below this value to the total number of values. The proportionate is the percentile. In case of the repetition of the value use mid rank approach i.e. add the 0.5 X (no. of times the value is repeated). Using 0.5 which means in a way that half the values are below the current value which we are counting and that is why add this second term indicating this many values below the current value while other 0.5 X n indicates the value above the current value which we are skipping.

**Covariance: C**ovariance tell you the kind of relationship between the two variables. It does not tell you the degree of relationship.

Intuition is as such.

Check for each point of variable whether it is above or below the variable average then do the same for the other variable, get the summation of the product of these two quantities. The sign indicates the kind of relationship positive or negative. Take the average like you do for the variance, with n-1 for samples and n for the population.

Each difference represents if the corresponding datapoint is above or below the average of the variable and if the two above average datapoints multiplied together they give positive value and same for the two negative values. Indicating they have correlation that is increasing one increase other, and similarly for decreasing one decreases the other. And you take the summed average to see the same thing for entire dataset.

**Covariance** does not give you the strength of the relationships because it not normalized value. The units of both variables could be changed, and the covariance could be changed. So, basically what you do is, normalized the covariance with it’s per unit variability, and that is why it is between (0-1) and (0-(-1)).

So, correlation depends on variability (SD) of the variables. If the variability is less even small positive spread across both variables (i.e. COV between variables) will tend to have high degree of overall CORR strength and vice versa).

Why use multiplication instead of summation in covariance, cause the summation won’t able to preserve the sign of each deviation (summation of differences from mean is zero for both), cause the most important thing that needs to be measured is the direction (sign) of each datapoints which describes or gives you the indication of how each datapoint moves (i.e. in positive direction or negative), as in summation the sign or direction preservation is based on the magnitude of the sum of differences of datapoint from their means respectively. E.g. suppose the first difference is 10 and second one is -3, the summation would be 7 but for the entire it would be zero and won’t make sense at all, it does not preserve the sign or direction though the relation for a single point is completely negative as one is higher than mean correspondingly the other one is decreasing with ref to its mean. But the same with multiplication can be preserved.

Correlation, which normalizes the covariance to their variability (variability of each variable) by dividing the covariance by the product of the SDs of two variables. I still do not understand the intuition of the dividing by the SDs.

Coefficient of variation: why needed and what is the intuition of it?

Variance/SD gives the idea about the dispersion in the dataset, but it is the absolute variation in the dataset. The meaning/interpretation of variation is specific or dependent on the magnitude of values. So, to have the measure of dispersion with respect to the magnitude of values or central tendency of the dataset coefficient of variation is calculated which gives the measure of the dispersion with respect to the measure of central tendency. Useful, mainly in the cases where we need to compare the variability of two datasets.

**Why the y-axis i.e. probability density in the probability density function** is not the exact probability itself, but it is the probability density or rather a probability per unit (whatever could be the unit of the x-axis) and hence the actual probability is calculated for two points only.

**Density estimation,**

**Parametric and non-parametric:**

Fundamentally the parametric means it assumes standard fixed form globally throughout with fixed parameters. (like linear regression in ml models, like normal distribution in density estimation).

Parametric model are models with parametric distribution assumption about the data, whose density function is standard formula for that function with parameters calculated from the data. For normal distribution assumption of the data parameters would be (sigma and mean).

Non-parametric estimation when the standard form is not directly assumes for the distribution of the data globally, because they cannot be assumed to be following the standard distribution, the density function for these is calculated with the help of the kernels which are standard parametric forms of the distributions only. Unlike the parametric which are defined globally for the entire dataset, in non-parametric with the help of the kernels local density is established.

The main key difference is parametric models does not grow with data, which is not the case for the non-parametric model which grows with data, like in case of the DT, KNN, SVM with RBF etc.

How to calculate the non-parametric?

Probability distributions?

* Probability density function (for continuous numerical data).
* Probability Mass Function.

Skewed and non-skewed distribution mean, mode and median distribution?

* How does it affect, very important image to remember, so firstly we know that for perfect normal distribution mean, mode and median all are overlapping. Now, supposedly we add few bigger extreme values in the distribution what happens then, distribution becomes right skewed and the mean shifts towards the right as the higher values are added, and so does the median cause now the range is increase towards higher side. And why does median is lower than mean, the answer lies in using median over mean in the presence of the outliers.
* Same thing for left skewed.
* Thing to remember, do not compare the left and right skewed straight away, compare the left skewed with the perfectly normal and same for the right skewed.

What is **kurtosis** and why it is there? first let’s understand how distributions are described.

Various shapes of the data are mapped by various distributions with its own parameters, like normal distribution is by mean and sigma, log by almost similar parameters and many more. So once mapped, with the help of the distribution’s probability density function, probability can be calculated. Whereas the kurtosis and skewness can be calculated for each distribution.

**Skewness for the perfect normal distribution is 0 and kurtosis for the same is 3.** If any of these two changes the distribution is not normal (perfect) normal distribution. To make the distribution non-normal change either of the above two. Pdf formula does not have both values included in it. Cause the condition where the pdf formula valid i.e. for perfect normal distribution these values are absolute constant. If the kurtosis is not 3 for the distribution, then the distribution becomes non-normal and the probability for the same can be estimated by any method of density estimation of non-normal data like KDE.

It’s not like each distribution/shape, or its probability density function is given by mean, sigma, skewness, kurtosis. Each distribution has its own parameters though this one can be calculated for each one.

Skewed data can be model by log normal, gamma and many other distributions,

Each distribution has different value of the kurtosis, for normal distribution this value is 3. Reason is rooted in the mathematics of it.

How to calculate the kurtosis?

* Calculate the fourth central moment just like variance (second central moment) just instead of degree 2 calculate with degree 4 or 4th power of the difference.
* Then the kurtosis would be (forth central moment) / (sigma\*\*4).
* Why central word in moment, cause the deviation is calculate from mean.
* Excess kurtosis is kurtosis - 3, but why?
* **Kurtosis for the normal distribution is 3** always, so compare relatively with normal distribution excess kurtosis is calculated which shows the relative of the kurtosis indicating whether the tail is fatter or narrower.
* Excess kurtosis greater than 0 is leptokurtic, thicker tails
* Excess kurtosis lower than 0 is platykurtic, thinner tails (plateau, i.e. platykurtic)
* Excess kurtosis equal to 0 is mesokurtic. (middle way to remember naming)

**How to know if the data distribution is normal or not?**

* QQ plot
* Statistical test like, Shapiro Wilson test, Anderson-darling test.

**Normal distribution:** e.g. height distribution, weight etc. Z represents how far the point is form the mean in terms of SD i.e. how many SDs away the point is from the mean.

**Uniform distribution:** uniform probability of all X’s e.g. drawing random number from the 1-100, drawing a card form the well shuffled pack of cards, rolling a dice and getting a number. Etc.

Used in many cases like random initialization, data augmentation, sampling, and hyperparameter tuning by defining the range of hyperparameter (or already known a range and then selecting the arbitrary random values within that range)

**Log normal distribution** will always be right skewed, the log normal distribution is exponential of the normal distribution, and the percentage increment by exp with respect to the original data point is more for higher datapoints, so the higher points would be placed far towards the right than the point compared to the lower points, which won’t shift as much towards the right.

**CLT:**

There is something called sampling distribution, where the distribution of the sample stat is plotted. Sampling distribution with mean and sum stats CLT comes into picture.

CLT is the fundamental basis because normal distribution works.

Properties of the CLT, mean of the distribution of the samples means is the equal to the mean of the population, whereas the SD of the sample means is SD (population)/sample size. (not the number of samples).

Basically, now we have two distribution one is population distribution and other one is sample means/sums distribution the mean is same for both (i.e. mean of the sample means is equal to the population mean) and SD for the sample distribution is population SD/ sample size.

This is the property use in population parameter estimation by confidence interval, statistical testing, etc.

If the sample size is not 30 then sampling distribution resembles the underlying population distribution. Meaning, the sampling distribution is normal only for normally distributed population data. There is no idea of student’s t distribution here, based of sample size and all, it only exists sample SD is used to resemble/estimate the population SD.

**Standard Error:** It is the standard deviation of the sampling distribution, i.e. SD of the sample statistic (mean) meaning how far away this sampling mean/stat from the distribution mean or population mean. It is called standard cause error because it is the SD of the sampling distribution and error because it represents the error in estimating the population parameter from the sample, i.e. error/difference of our sample mean and actual population mean which is the mean of the sampling distribution, basically standard error represents the distance of point from the mean(population). Higher the sampling size lower is the SE, cause the uncertainty associated is reduced with higher number of samples.

SE = SD/root of N. SD is the population SD when population SD is known, else it is samples SD, for multi sample test the SE is calculated accordingly with certain of aggregation.

**Confidence intervals:** Estimating the population parameter from a single sample to lie in certain range of values with certain degree of confidence.

**we cannot be confident that this very first sample’s confidence interval contains the true population parameter**-there is **no guarantee** for any individual sample’s interval.

How is CLT use in the case of the CI? You know what CLT states which tells that mean of samples mean distribution is equal to the mean of the population and SD sample mean distribution is SE which is population SD/ root of sample size.

Now is this useful in estimating CI, to make CLT we need multiple samples, and you cannot create multiple samples from data which is available then they won’t be independent samples even the very large data is available. So, **from single sample we predict the CI for population mean (parameter). How do you do so?**

* We know the sampling distribution follows the normal distribution, and we have single sample mean i.e. a single datapoint form this normal distribution of the sample means if there were multiple samples drawn obviously.
* Now from this single datapoint of the normal distribution whose mean is equal to population mean and SD is SD of the population / root of sample is size, we need to estimate the mean given the sample size and SD of the population.
* We know the sampling distribution follows the normal distribution, and we have single sample mean i.e. a single datapoint form this normal distribution of the sample means if there were multiple samples drawn obviously.
* Now from this single datapoint of the normal distribution whose mean is equal to population mean and SD is SD of the population / root of sample is size, we need to estimate the mean given the sample size and SD of the population.
* point is if the sample size is more than 30 or large in general then the underlying distribution of the population does not matter, but if the sample size is small then in that case it handy to have underlying distribution as normal.
* A 95% confidence interval is centred at the sample mean, not the population mean, so it does not always cover the central 95% of the sampling distribution. Instead, 95% refers to the long-run proportion of such intervals that will contain the true population mean
* A confidence interval (CI) estimates a range around the sample mean that, based on the chosen confidence level (like 95%), is likely to contain the true population mean-not a certain proportion of the sample data, but a certain proportion of possible sample means if the sampling were repeated many times.

**Where and when to use Z-Procedure?**

I say you want to estimate the mean of the national level test score for this year based of sample score of 100 students where 100000 students gave the exam, though the SD of the exam for last 50 year is available, can I say this is the example where Z procedure is used

* If you have a **known population standard deviation ($\sigma$)**-either from historical data or from a very large, reliable sample-you can use the **Z-procedure** for confidence interval estimation, provided your sample size is reasonably large (usually n≥30*n*≥30).

**Where and when t-procedure is use:**

* When the SD of the population is not known, the SD of the samples is used, which increases the uncertainty which ultimately changes the distribution making the distribution as students t-distribution with fatter tails to accommodate the uncertainty involved due to use of sample SD instead of population SD. When the sample size
* the point is if the SD of the population is not known then you always use t-procedure just the fact that if the sample size is high, this means the t test is more reliable cause t distribution for large sample gets closer to normal distribution, is it the case
* **Student’s t distribution**, symmetrical bell-shaped curve with thicker tells which accounts for more extreme scenarios, used in case when the population SD is unknown, and sample size is small. Students-t distribution is constructed as distribution of the sample means of the samples when the SD of the population is not known. It is the pdf of the t-statistics for DOF.
* Cause the SE of the sample distribution is calculated from the sample SD instead of population SD that is why the distribution does not remain normal and becomes the students T distribution, which accounts for the more uncertainty which is involved due to use of SD of the sample.
* For higher sample sizes, the t-distribution becomes increasingly like the normal distribution.
* The t-distribution is a normal-like distribution with higher kurtosis (heavier tails), especially noticeable with small sample sizes.
* So, theoretically where we use t distribution, i.e. we assume the sample statistic follows student’s t distribution instead ideally it should follow perfect normal in if the sample size is high. So, the reason for using the t distribution is only to compensate the biased or randomness involved because we are not able to correctly get the SD of the sampling distribution?

Things to do in details after,

* PNC
* CONVOLUTION
* CLT and NORMAL INTUITION.

Certainly! Here’s a slightly more detailed and nuanced explanation of A/B testing, while keeping it clear and organized:

**A/B testing** is a widely used experimental methodology for comparing two versions of something—such as a webpage, app feature, or marketing message—to determine which one performs better according to a specific goal, like increasing clicks, sign-ups, or sales. The process begins with identifying a clear objective and forming a hypothesis about how a particular change might improve performance. You then create two versions: version A (the control, or current version) and version B (the variant, with your proposed change). Users are randomly split into two groups, each experiencing only one version, ensuring that differences in results are due to the change itself and not external factors.

As users interact with the versions, their behaviour is tracked, and key metrics are collected. After gathering enough data, statistical hypothesis testing is used to analyse the results. This involves setting up a null hypothesis (there’s no difference between A and B) and an alternative hypothesis (there is a difference). Statistical tests—like the t-test or z-test—are applied to determine if the observed differences are statistically significant, meaning they’re unlikely to have occurred by chance. The p-value from these tests helps decide whether to accept or reject the null hypothesis.

A/B testing is not a statistical test itself, but rather a structured way to apply statistical hypothesis testing in real-world decision making. It allows organizations to make confident, data-driven choices about design, content, or features by directly measuring user responses. This approach minimizes guesswork, reduces the risk of making changes that don’t work, and supports continuous improvement based on actual evidence from real users.

**Hypothesis Testing:**

So, basically, all we do is, we consider the null hypothesis, i.e. state of no change as true and then try to find the evidence whether it is false or not, if we are not able to find evidence against it, then we stick with our presume opinion of null hypothesis being True and reject the proposed change i.e. alternate hypothesis, and vice versa. There is nothing as proving something It’s all about accepting null or not based off the evidence from the data.

You use the location of your sample statistic in the sampling distribution to determine how likely it is to occur by chance. If it’s very unlikely, you reject the null hypothesis. This is the essence of statistical hypothesis testing.

The core intuition of the hypothesis testing is, you try not accepting the alternate hypothesis without evidence, so try to kind of reject it with the data from the sample while finding evidence against null hypothesis, only if we you find evidence against the null hypothesis you accept the alternate hypothesis.

Why there are two hypotheses? usually the alternate hypothesis and null hypothesis are kind of complementary to each other, they are fully complementary sometimes, but not all the time. Wherever they are totally complementary then in that case only one is required.

**so, whichever we accept or reject we can't surely say the one is guaranteed correct or not?**

Certainly! Here’s your paragraph, revised for clarity and accuracy:

In hypothesis testing, we start with a null hypothesis, which we want to test for evidence against. We do this by checking where the sample statistic (such as the sample mean) falls within the sampling distribution under the assumption that the null hypothesis is true. If the sample statistic falls beyond a certain critical region (determined by our chosen significance level, such as 95%), this is considered evidence against the null hypothesis. For example, in a two-sided z-test where the null hypothesis is that the population mean equals 50, and the alternative hypothesis is that the population mean is not equal to 50, we calculate the z-score of the sample mean. We then determine the boundaries (critical values) that contain 95% of the sampling distribution around the mean (typically z = ±1.96 for a 95% confidence level). If the sample mean falls within this central region, we do not reject the null hypothesis. If it falls outside this region (in the outer 5%), we reject the null hypothesis, concluding that there is significant evidence that the population mean is not equal to 50.

e.g. in insurance

**Testing Average Claim Amounts**

* **Scenario:** An insurance company originally set policy rates assuming the average claim amount is $1,800. They are concerned the true mean may now be higher, which could impact profitability.
* **Hypotheses:**
  + Null hypothesis (H₀): Mean claim amount ≤ $1,800
  + Alternative hypothesis (H₁): Mean claim amount > $1,800
* **Process:** The company samples 40 claims and finds a sample mean of $1,950. Using a t-test, they check if this increase is statistically significant. In the example, the test statistic did not exceed the critical value, so they failed to reject H₀, meaning there was not enough evidence to conclude the mean claim amount had increased. In this case the SD of the sample is used.

**Comparing Claim Amounts Between Groups**

**Example:**  
An insurer wants to know if smokers file higher medical claims than non-smokers.

* **Null hypothesis (H₀):** Mean claim amount for smokers = mean claim amount for non-smokers
* **Alternative hypothesis (H₁):** Mean claim amount for smokers > mean claim amount for non-smokers

A t-test is used to compare the average claims of the two groups. If the p-value is below the significance threshold (e.g., 0.05), the insurer concludes that smokers have higher claims

In a **Type I error,** the null hypothesis is rejected more often than it should be (so the alternative hypothesis is incorrectly selected more frequently), whereas in a **Type II error**, the alternative hypothesis is not selected as often as it should be (because the null hypothesis is incorrectly retained.

**Generally, Type 1 errors are considered more serious in many fields.**

If a new cancer medicine is falsely found effective due to a Type 1 error, it gets approved and prescribed despite not actually working. Patients may rely on this ineffective drug, risking their health and missing out on real treatments. This makes a high Type 1 error rate extremely dangerous in medical research. and in case of type 2 error, it mainly represent the missed opportunity, but this is very specific the problem how test is designed.

case where accepting the wrong alternate is highly costly then type 1 is more detrimental and where accepting the wrong is null is more costly then type 2 is more severe and detrimental.

One side and two side tests.

The **p-value** is the probability of obtaining a result as extreme as, or more extreme than, what you observed in your data, **assuming the null hypothesis is true.**

Not the probability of rejecting the null hypothesis. The p-value is the probability that measures the strength of the evidence against the null hypothesis

The **significance level**, often denoted by the Greek letter alpha (α), is the threshold set by the researcher before conducting a statistical test to determine how strong the evidence must be to reject the null hypothesis. It represents the **maximum probability of making a Type I error**-that is, the risk of incorrectly rejecting the null hypothesis when it is true.2

For example, a significance level of 0.05 means you are willing to accept a 5% chance of concluding that an effect exists when it does not. If the p-value from your statistical test is less than or equal to the significance level, you reject the null hypothesis and consider the result statistically significant.

**The test stat follows the null hypothesis distribution, and if it lies far away i.e. with lower probabilities meaning chances of it lying in the null distribution is less.**

Different types of the hypothesis test?

1. **What is a Z-Test?**

A **z-test** is a statistical hypothesis test used to determine whether there is a significant difference between sample and population means, or between the means of two independent samples, **when the population variance (or standard deviation) is known,** and the sample size is large (typically n ≥ 30).

**Z-tests are appropriate when:**

* The data are approximately normally distributed (or the sample size is large enough for the Central Limit Theorem to apply).
* The population standard deviation is known.
* The data points are independent and randomly selected.

**How Does a Z-Test Work?**

State the Hypotheses:

**Null hypothesis (H₀):** No difference or effect (e.g., the sample mean equals the population mean).

**Alternative hypothesis (H₁):** There is a difference (e.g., the sample mean does not equal the population mean).

**Set the Significance Level (α):**

Commonly 0.05 (5%).

**Calculate the Z-Statistic (Z-Score):**  
The z-score measures how many standard deviations the sample mean is from the population mean:

* Z=Xˉ−μ0σ/n*Z*=*σ*/*nX*ˉ−*μ*0
* Where:
* Xˉ*X*ˉ = sample mean
* μ0*μ*0 = population mean under the null hypothesis
* σ*σ* = population standard deviation
* n= sample size

**Compare to the Critical Value or Find the p-value:**

* For a two-tailed test at α = 0.05, the critical z-values are ±1.9
* If the calculated z-score is more extreme than the critical value, **reject the null hypothesis**.
* Alternatively, calculate the p-value and compare it to α.

**Draw a Conclusion:**

* If |z| > critical value or p-value < α, reject H₀.
* Otherwise, do not reject H₀
* **Types of Z-Tests**
* **One-sample z-test:** Compares a sample mean to a known population mean.
* **Two-sample z-test:** Compares means from two independent samples, same as the two sample test, just the difference is it uses population SD instead of sample SD. E.g. comparing the mean IQs of the two cites when the SD of the two cities population is known.

**Example**

* Suppose a company claims the average salary of its employees is $50,000 (σ = $5,000). You take a random sample of 40 employees and find an average salary of $52,000. You can use a z-test to determine if this difference is statistically significant
* **In summary:**  
  A z-test is a hypothesis test for comparing means (or proportions) when the population standard deviation is known, and the sample size is large. It uses the z-score to measure how far your sample statistic is from the hypothesized population value, helping you decide whether to reject the null hypothesis

1. **T -tests:**

**One-sample t-test:** Compare the mean of a single group to a known value or population mean

**Independent two-sample t-test:** Compare the means of two independent groups

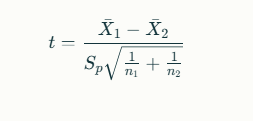
**Paired (within-subjects) t-test:** Compare means from the same group at two different times or under two conditions (e.g., before and after an intervention).

One sample t-test: The intuition of the single sample test is same as the z-test,

e.g. A nutritionist claims that the average daily protein intake for adults in a certain city is 50 grams. To test this claim, a researcher randomly selects 15 adults and finds their average protein intake is 54 grams with a standard deviation of 6 grams. T statistic (T score) is calculated by the given data as (54-50)/SE, where SE is SD of sample upon root of sample size.

e.g. 2. An insurance company historically processes claims at an average cost of $60, but with rising expenses, they want to test if this has changed. They sample 20 recent claims, finding a mean cost of $65 and a standard deviation of $8. Using a one-sample t-test, they calculate a t-value of approximately 2.80 (t=65−608/20*t*=8/2065−60), which exceeds the critical value of 2.093 for 19 degrees of freedom at a 0.05 significance level. This result leads them to reject the null hypothesis and conclude that the average claim processing cost has significantly increased from $60.

**Two sample tests:**



*df* =*n*1+*n*2−2

**The purpose of the independent two-sample t-test is to check whether two samples, drawn from two independent populations that are assumed to have equal standard deviations (SD), have the same mean or not.**

Intuition:

*The independent two-sample t-test helps you answer a simple but important question:****Are the average values (means) of two separate groups different, or could the observed difference be due to random chance?***

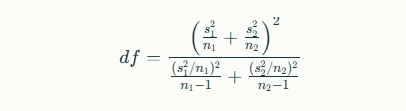
*The t-test measures how many “standard errors” apart the two means are. The bigger this number, the more confident you can be that the difference is not just due to random variation.*

**You’re checking whether the difference between the two-group means is large relative to the variability (standard deviation) within the groups.**

**e.g.** The company calculates the average claim amount for each group and applies a two-sample t-test to compare the means. If the t-test shows that new drivers have a statistically significantly higher average claim amount (with a low p-value), the insurer can confidently conclude that new drivers represent a higher risk. This evidence justifies charging higher premiums to new drivers compared to experienced ones, ensuring that the pricing structure accurately reflects the underlying risk

**Welch’s t-test**

t-stat is calculated in similar manner just the difference is how is df calculated and it is mentioned below.



Intuition of CLT in Two-Sample Tests

* **As sample sizes grow, the means of each sample (from their respective populations) become approximately normally distributed, regardless of the original population distributions.**
* **Because both sample means are (approximately) normal, the difference between the two sample means also follows a normal distribution when the samples are large.**
* This is true even if the underlying populations are not normal, if the samples are sufficiently large and independent.

Why This Matters

* The normality of the difference in sample means (thanks to the CLT) allows us to use t-tests and calculate p-values, making statistical inference possible and reliable in real-world scenarios.

What You’re Doing in a Two-Sample Test

* You **calculate the difference between your two-sample means**.
* The **Central Limit Theorem** tells you that, for large enough samples, the **sampling distribution of the difference between means** will be (approximately) normal.
* You then **ask: “Where does my observed difference fall within this theoretical distribution?”**

Interpreting the Result

* If your observed difference is **close to the centre** of the sampling distribution (near zero, assuming the null hypothesis that the means are equal), it’s likely just due to random chance.
* If your observed difference is **far in the tails** (very unlikely under the null hypothesis), you have evidence that the population means are different.

e.g. You measure the grams of protein in two different brands of energy bars. Each brand represents an independent group. A two-sample t-test helps you assess whether the average protein content differs between the two brands

**paired sample t – test:**

Students take a practice test and then, after a study session, take a final test. You use a paired t-test to see if the study session significantly improved their scores, since each student’s scores are paired (pre- and post-study)

Students take a practice test and then, after a study session, take a final test. You use a paired t-test to see if the study session significantly improved their scores, since each student’s scores are paired (pre- and post-study)

The key **difference between a two-sample t-test and a paired sample t-test** is that the two-sample t-test compares the means of two independent, unrelated groups (such as different people in each group), while the paired sample t-test compares the means of two related groups where each data point in one group is matched to a specific data point in the other group (such as measurements from the same person before and after a treatment); essentially, the two-sample t-test assesses differences between groups, whereas the paired t-test evaluates changes within matched pairs.

**Chi square test:**

**Chi square distribution:**

So, chi square is just the naturally occurring phenomenon or distribution occurs when the random variable is taken from the standard normal distribution and square them then sum them and you do it for n time, then you get the chi square distribution, as DOF increases this distribution becomes normal.

In chi square test location of sample chi square stat is found out and checked where does it lie in sampling chi square distribution?

Why chi square is non-parametric?

*O follows a binomial distribution, and for large n, the binomial is well-approximated by a normal distribution.  
This is a key reason why the chi-square test works.*

*So, the O is the normally distributed for large sample size.*

same fundamental is use while changing the normal distribution to standard normal as well, where the mean is E and variance is also E cause for high n in binomial with low probability the variance becomes E, and mean is Expected or average of outcome which is also E.

and this is how we get the normal sample stat, which is follows chi square distribution, cause this sample stat is calculated by squaring the independent random variables (categories value in actual test) which are from the standard normal distribution and how this one value which is squared is standard normal distribution is explained above.

and this is how we get the normal sample stat, which is follows chi square distribution, cause this sample stat is calculated by squaring the independent random variables (categories value in actual test) which are from the standard normal distribution and how this one value which is squared is standard normal distribution is explained above.

**Why do we need CLT?**

For large sample sizes, observed counts (O*O*) from a binomial distribution can be approximated by a normal distribution with mean and variance E=np, thanks to the Central Limit Theorem (CLT). This allows the standardized difference to be treated as approximately standard normal. When these standardized differences are squared and summed across categories, the result follows a chi-square distribution, which forms the basis of the chi-square test. The CLT is thus essential for justifying the use of normal and chi-square approximations in categorical data analysis.

In one accept of CLT is, means of samples follow normal distribution, which is handy in hypothesis testing and CI estimation, similarly other aspect is, if the squares of the **standard normal distributed** random variables are taken, they follow chi quare which becomes normal for large n or DOF.

* **CLT:** Means (or sums) become normal as sample size increases.
* **Chi-square:** Sums of squared standard normal become chi-square distributed, and for large k*k*, this distribution itself becomes approximately normal.

expected distribution meaning, what are the expected counts of each category

**Goodness of fit:** The goodness of fit test is a statistical hypothesis test used to determine how well your observed data match the expected distribution specified by a theoretical model. Most commonly, the chi-square goodness of fit test compares the observed frequencies in each category of a categorical variable to the frequencies you would expect if the data followed a particular distribution (such as equal proportions or a known probability distribution). If the observed and expected values are close, the model is considered a good fit; if they differ significantly, the model may not fit the data well.

Check notes for more revision.

**Example**

**Expected distribution:** Each number (1–6) should appear 10 times (if the die is fair).

Observed data: You count how many times each number appears.

The test checks if the observed counts are close enough to 10 for each number to conclude the die is fair

**Claim severity categories:**

An insurer expects that, based on historical data, 20% of claims will be low severity, 30% moderate, and 50% high severity. After a year, the actual observed proportions are different. The chi-square goodness of fit test can be used to check if these observed frequencies significantly diverge from expectations. If so, it may indicate a change in risk profile or require adjustment of reserves and pricing strategies

**Claim frequency distribution:**

Suppose an insurance company models the number of auto accident per policyholder per year using a Poisson distribution. After collecting data from 700 drivers, the company compares the observed number of claims per driver to what the Poisson model predicts. The chi-square goodness of fit test helps determine if the observed distribution of claim frequencies fits the theoretical Poisson distribution, validating or questioning the suitability of the model.

**Fraud detection:**

Insurers may use the chi-square test to detect fraud by comparing the distribution of certain claim attributes (such as claim amounts, types, or timing) to what is expected under normal, non-fraudulent circumstances. Significant deviations flagged by the test can prompt more detailed investigations into potentially fraudulent claims.

**Test for independence:** The chi-square test for independence is a statistical test used to determine whether there is a significant association (relationship) between two categorical variables. It assesses whether the distribution of one variable is independent of the other by comparing the observed frequencies in a contingency table to the frequencies expected if the variables were independent.

The statistical intuition is same as above for the tests, whereas the actual working with the formulas is in the notes.

e.g. association between the education and exercise.

Suppose an insurance company wants to know if the **type of claim filed** (e.g., accident, theft, natural disaster) is **independent of the policyholder’s age group** (e.g., under 30, 30–60, over 60).

* **Data:** The company collects a contingency table showing the number of each claim type filed by policyholders in each age group.
* **Null Hypothesis (H0):** Claim type and age group are independent (no association).

in above case if the test stat falls out of null region, then the age group and claim filled are dependent

**ANNOVA:**

**F distribution:**

* Ratio of two independent chi-square distributed variables, each divided by its respective degrees of freedom, represents the ratio of two variance estimates, and it follows f distribution.
* What do mean by variance estimate in this case, variance estimates meaning the variance of the sample from to which is this chi square random variable corresponds to, meaning it is the ratio of the variance of the two samples whose sum of squared of all samples represents the RV in chi square distribution.
* The term "chi-square independent variable per degree of freedom" refers to the average value of the sum of squared independent standard normal variables per degree of freedom, just like the mean of the sum of RV’s.
* For the F-distribution and F-test, samples must come from two independent, normally distributed populations. Sample variances from these samples, when appropriately scaled, each follow a chi-square distribution. The ratio of these two independent chi-square variables (divided by their degrees of freedom) forms the F-statistic. The F-statistic is used to compare the variances of the two populations-if the populations are normal, this ratio follows the F-distribution.
* Parameters are dof1, dof2, a curve is defined for combinations of dof1, dof2. Then based on f stat and this DOFs combination in cumulative f distribution finds out the probability.
* F distribution is used in:
* Comparing means of three or more variables (Analysis of variance)
* Comparing variance(F-test):
* Regression analysis to test the overall significance.

**One-way ANOVA:** One-way ANOVA (one-way analysis of variance) is a statistical method used to test whether there are significant differences between the means of three or more independent groups.

e.g. Suppose a researcher wants to know if three different teaching methods (Lecture, Workshop, Online) result in different average exam scores for students

Null hypothesis (*H*0): The mean exam scores are equal across all teaching methods.

Alternative hypothesis (*Ha*): At least one teaching method has a different mean exam score.

E.g. An insurance company wants to determine whether the average claim amounts differ significantly across four geographic regions (North, South, East, West).

How is this f distribution used in the Anova?

* F\_stat in Anova is (SSB/ ) /DF (SSW / DF),
* Where the numerator SSB does not follow chi square, but the SSB/ SD (population) follows the chi square distribution, likewise for the SSW and population SD (population is cancelled out in here.
* So, the aim is to find out the ratio of variance ratio of SSB (how much group mean differs from individual mean) and SSW (How much data points differs within each group with respect to group mean) i.e. ratio of variance between groups and variance within groups.
* The intuition of the SSB is how far away the individual mean lies compared to the population mean, if these individual means are close to population meaning these itself would be away from each other.  larger SSB suggests that the group means are more spread out from the grand mean, indicating potential significant differences among groups. Whereas SSW indicates the variability within each group, indicating the spread of each groups data. if the variance within a group is low, the mean of that group is generally a more reliable and representative estimate of the group's true value.
* So, in Anova what happens is, you try to check the variability of each group mean with population mean with respect to individual average variability of each mean.

**F- test for variance check:**

The **F-test** is a statistical test used to compare variances. It is most applied to determine whether the variances of two samples (or populations) are significantly different by calculating a statistic (the F value) as the ratio of the two sample variances. This F value is then compared to a critical value from the F-distribution, considering the degrees of freedom and significance level, to decide whether to reject the null hypothesis that the variances are equal.

The F-test assumes that samples are independent, drawn from normally distributed populations, and (for some tests) have equal variances.

The further above 1 your F-ratio is (for given degrees of freedom), the more likely it is to be considered “extreme,” and the stronger the evidence that the two population variances are different.

Usually while calculating the F stat, the lower variance is kept at the bottom to get the f stat greater than 1 and how big is this ratio than 1 for given DOFs determine the probability which indicates probability for existence of this difference in variance when sample are randomly chosen from respective population.

**E.g.**

- Manufacturing company compares the samples for variance, if the two batches have same variance or not? though the mean could be same, but the variance is very important to understand the consistency of the sample quality.

- Two investment portfolios have the same average return, but one is much more volatile than the other.

- An insurance company is evaluating two types of auto insurance policies. Both policy types have the same average (mean) claim amount per year-say, $2,000. However, for Policy A, the claim amounts are tightly clustered around the mean (low variance), while for Policy B, claim amounts are highly variable-some years have very high claims, others have very low (high variance).

**How to check the normality?**

* Shapiro wilk test.
* Kolmogorov - Smirnov Test.
* Anderson-Darling Test.
* D’Agostino-Pearson Omnibus Test.

Visualization methods.

* Hist plot
* QQ plot
* Probability-Probability plot.

**Shapiro wilk test:** So, Shapiro wilk test tells you if the underlying distribution from which the sample is drawn is it the normal distribution or not?

* The test stat here does not follow any distribution (sampling distribution) instead a test stat is calculated by analytical formula and the corresponding probability is found by monte Carlo simulation.
* Once the Shapiro-Wilk test statistic is calculated analytically, the corresponding probability (p-value) is determined using results from Monte Carlo simulations, since W does not follow a standard named distribution.

**Monte Carlo simulation** uses random sampling and repeated calculations to estimate the range and probability of possible outcomes in uncertain or complex systems.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Python

General python:

For binary: try to calculate with 2’s power then sees the sum of it, how much you are getting to that decides the number of bits, and likewise you can then decide the 1 and 0’s.

Bin of (111) = 1 \* 2^2 + 1 \* 2^1 + 1 \* 2\*0

Hashable: objects which can be hashed, int, str, tuple (with unique values)

What is hash= unique and fixed (no change), immutable objects are hashable, whereas mutable are not hashable.

Mutable in python: List, dict, sets etc.

Immutable: int, float, str, tuple, frozen set etc.

Comprehensions: Runs the code from left to right, i.e. the left one is outer one and right one is inner one and likewise the hierarchy in the loops. Left to right, same is the case for the conditions, left to right condition after the loop is condition for that loop. To simply the evaluation just tries the logic with left loop as outer loop and first immediate condition as outer condition right loop as in inner one and right condition after that is the inner loops condition.

What is hexadecimal representation?

* It is the representation with 0-9 and then A-F, use int(‘FF’,16) for int conversion like you do for the int(binary,2)

Lexicographical sorting: sorting based of the ascii values.

Sorted ([‘Harshal’, ‘Nayan’]), sorting start based off comparing from left, if unmatched found then it stops there itself. Same for the numbers.

Same for the comparison, i.e. “Harshal” < “Nayan”, same lexicographical comparison, start form the left then stops as soon as the inequality is matched.

**Module, package and library?**

A **module** is a single file containing functions, classes, and variables; a **package** is a directory with multiple related modules and an \_\_init\_\_.py file to organize them; and a **library** is a collection of packages and modules bundled together to provide broader reusable functionality across projects. Modules form the basic building blocks, packages group these modules, and libraries encompass multiple packages for larger-scale use.

**Modules:** math, random, os, sys, re, time, Json, shutil, queue, pprint

**Packages:**  
flask, django, numpy, pandas, scipy, matplotlib, sklearn, urllib, xml, tkinter

**Libraries:**  
NumPy, Pandas, Matplotlib, Scikit-learn, TensorFlow, Requests, Flask, Django, BeautifulSoup, Pygame, SciPy, Seaborn

e.g. from itertools import combinations

from itertools import groupby

or itertools.combinations() or itertools.groupby()

both ways works, meaning you can access the classes and function in the module just like the methods of any object instance.

**OOPS:**

Special methods (Dunder methods):

Methods which have special ability, which gets externally called based off the conditions specified mostly externally, the logic is written for the special condition in terms of new method, and as the condition is mate the same method is called.

Example, add, write the special method inside the class with \_\_add\_\_ where it takes one set of input from the current (object) from the future, and the other input from the other object where the other object must be passed to the special method like the self. The special condition here would be the use of + in the object1 and object2. Likewise other methods are defined many are specific to single object whereas the other are interactive. E.g. \_\_len\_\_, \_\_str\_\_ etc. here \_\_str\_\_ get called on the condition of the using str or print on the object.

Whenever you create attributes from outside those just get sorted as it is, without any self and not in the constructor or anything.

NumPy:

Np.round (a,2) where second parameter describes up to what digit to round off to.

Masking and indexing, both are used for accessing the specific elements of the array, indexing is based off the index values whereas the masking is based off the Boolean conditions (True/False) of the similar shape.

Np.tile – repeats the array in the specific directions.

Np.repeats- repeats the individual values multiple times.

MLOPS

**MLOPS (Andrew Ng)**

Manages the lifecycle of the machine learning model. In entire lifecycle, model gets build in to far fewer lines compared to entire code of the ml lifecycle of the model.

Edge system: local system which processes the data locally, interact with deployed models and get the predicted response.

ML project lifecycle:

Scoping: Define project

Data: Define data and established baseline, label and organize

Modelling: Select and train model, perform error analysis

Deployment: Deploy in production, monitor and maintain system.

e.g. Voice recognition in Lifecycle: How does it works?

Few issues in data level step, first is the data correct labeling, volume normalization etc.

Issues in modeling, in modeling there are two approaches of improving models,

First is kind of research base approach to change the,

Change the code(algorithm/model)

Change the Hyperparameters

Data Constant

Whereas the second approach which is more product team approach is to

Keep the code/model/algorithm same

But change or tune the hyperparameters

And improve the data, (based on the data improve quality of the data to the model,)

**Data drift:** Data distribution changes, i.e. statistical properties of the input variable (features changes), with same relationship between X and Y.

e.g. A retail model trained on customer data from one region is deployed in another region with different age groups and income levels, causing the input distributions to shift, though the relationship between the input and output remained the same

**Concept Drift:** Relationship between the input variable (X) with label target variable (Y) changes.

e.g. in spam detection, suppose now the spammers start using different techniques meaning the relationship between the input changes, what was spam earlier is not spam now.

Software engineering issues:

* Realtime or Batch
* Cloud vs edge/browser
* Computer resources (CPU’s / GPU’s)
* Latency, throughput
* Logging
* Security and Privacy

**Deployment Patterns:**

Common deployment cases:

1. New product/capability: deploying a new product.
2. Automate/ assist with manual task: Add on to already deployed product/ running product, like smart phone monitoring product.
3. Replace previous ML system: replacing existing ml system

Key Ideas:

* Gradual ramp up with monitoring: Have small traffic initially then monitor and ramp up the traffic gradually.

Rollback:

* Revert to the previous version if the current one gets you errors.

**Types of the deployment:**

* Shadow mode deployment: a new model is deployed but does not handle queries directly, existing model only process the queries, and new model is tested for the same queries. Facilitate the testing of the model on the live data before handling actual queries.
* Canary deployment: New version is released to small user first while other users still use the old one and then gradually if no errors and performance drop in new model then the new one is rolled out to more users.
* Blue green deployment: In Blue green deployment, blue as in old model with green as in new model both maintained in two environments, only one is live at a time, and when testing is done the entire is switched entirely. Easy to rollback as both are maintained in separate environments.

**How to monitor ML production model:**

Dashboard metrics are monitored, now which metrics to monitor?

* Software metrics, like Memory, compute, latency, throughput, server load etc.
* Input metrics: certain metrics about the input data can be monitored which describes the statistics of the input data (like data proportionate). E.g. for speech recognition model input avg input length, avg input volume, num missing etc. can be checked with pre-decided thresholds
* Output thresholds: Output quality measuring metrics, like in case of the speech recognition null return, timer users redo the search (indicating the inefficiency in detecting the speech first time). This metrics are finalized as you run the model, initially you can start with lot of these metrics.
* Like model development, deployment is iterative process, you develop the model, deploys it, test it as the traffic gets on the model, do the performance analysis makes the modifications and its loop now. Retraining is either manual or automatic.

ML pipeline monitoring: entire project is based on the multiple steps which are combined in a pipeline, each step could be a ML algorithm itself where the input of one step is fed to the next step. So, pipeline could have multiple elements combined which could be ML elements or non-ml elements combined, it’s important to monitor the metrics at multiple stages as the next stage element is dependent on the first one.

Software metrics, drifts could be measured at each stage’s ML element.

Best practices for building for building the ML model:

* Model(code) centric approach
* Data Centric approach

Key challenges:

* Doing well on the train set.
* Doing well on test set.
* Doing well on business metrics/project goals.

Mainly the ML projects are driven by the test set accuracy, but there could be discrepancy between the test set and business logic. How?

**How low-test error is not good enough?**

* Fairness, bias and legal risk, if the model performs poorly on the specific groups like gender, age, ethnicity then in that case, business application won’t allow the model, though it would be giving the good result on the test set. (sort of like imbalanced set).
* Cost of mistake varies, like the mistake made on certain classes is far expensive than then other one (e.g. mistake on recommending wrong shirt is not that but approving fraudulent transaction is very high)

**How to establish the baseline model for the data? (structured and unstructured data)**

* Human level performance (comparing with human performance)
* Literature search (open-source model or work indicating the performance)
* Performance on older system
* Quick and implementation (like Harshal builds the quick baseline model and then start improving the model).

Tips for getting started.

Good data is better than good algorithm. Always takes deployment/compute constraint into account while developing the model.

Sanity check: before start training a model, check with very few datapoints and try to overfit the model with very smaller data and see if the accuracy is good or not, this gives you the idea whether is it worth going deeper and worth taking the efforts to train and optimized this model on this data.

**Error analysis example:**

Analyzing the error, error patterns, quantify, understand the errors made. E.g. for speech recognition is,

For the misclassified speech texts, creates the different features stating the reasons of the noise, like car noise, human noise, low bandwidth and mark for each of the text, the columns don’t have to be mutually exclusive. This indicates the speech with its unique reason of error and help you understand which kind of error is what and which one to prioritize and how? the column can be added as you move in the data. This column I’m referring are tags,

So, this error analysis or tagging process is the iterative process like the other process like model building as well as the deployment process.

**Error Analysis for Skewed dataset (imbalanced):**

Use metrics like Precision, recall and F1 scores, F1 scores harmonic mean penalized the score far more for smaller value of any of the recall or F1 score.

**Performance Auditing:** even after the error analysis, one last time before deploying, check for fairness and bias,

1. Brainstorm the ways the system might go wrong.

* Performance on subset of data, (e.g. ethnicity, gender)
* How common are certain error.
* Performance on rare classes.

1. Established metrics to assess performance on appropriate **slices of the data**.
2. Get business/product owner buy-in.

Speech recognition example.

1. Brainstorm the ways the system might go wrong:

* Accuracy on different genders and ethnicities.
* Accuracy on different devices.
* Prevalence of the rude mis-transcriptions. (Like GAN could be interpreted as GUN or GANG, could mean the data would have lot of content related to gun violence and could be misleading.

1. Established metrics to assess performance against these issues on appropriate slices of data.

* Mean accuracy for different genders and major accents.
* Check for prevalence of offensive words in output.

Now just, imagine your error analysis has pushed you to improve your model on certain slice (category or tag etc.)

Model centric approach – have better model (most of the research development in the field of AI/ML is based on the same approach).

Data centric approach- Improves the quality of the model (Fixed code, change the quality of the data). Best approach to improve the quality of the data is by data augmentation.

**Conceptual picture of thinking about the data augmentation:**

Let’s take an example of the speech recognition.

Different types of the speech input noise:

* Car noise
* Plane noise
* Train noise
* Machine noise
* Caffe noise
* Library noise
* Food court noise

**Graphical intuition,** just imagine graph, with y-axis being the performance with X having different noise kinds, now suppose some noise kinds have lower performance on them and some have high, compare the performance with human level performance, so now we have two curves one indicating the actual model performance on the different noises and human level performance curve as well, the max height or difference between them gives you the highest chance or scope of the improvement, so you tend to improve that one, improve as in the data can be augmented for this noise, by improving the data for this noise does not reduces the performance on the far extreme/ good noises but does improves performances on the similar/nearby noise datapoints as well.

Just like a rubber strings, you tend to stretch the rubber string of the model towards the rubber string of the human level performance from the point where there is max scope of the improvement. By doing the nearby points/ noises also improves or gets stretches, nearby ones get stretched the most compared to the far ones.

**How to do data augmentation?**

Let’s take a similar voice detection example.

Voice signal + Caffe noise = synthetic training example

Voice signal + library noise = synthetic training example

Goal:

Create a realistic example, that algorithm does poorly on, but humans or another baseline does well on.

Checklist:

* Does it sound realistic?
* Is the X🡪 Y mapping clear (e.g. can the humans recognize speech?)
* Is the algorithm currently doing poorly on it?

e.g. for the scratch detection in phone manufacturing?

- ways to augment would be contrast/ color change, dark, light obviously not too dark not too light.

- Even GANs can be used.

You fit the data augmentation in the data iterative loop, i.e. you do augmentation then you train, error analysis and depending on the error analysis repeat the process.

**Can Adding data by data augmentation hurt?**

mostly not.

* If the model is large enough
* The mapping X🡪 is clear (given only the X humans can accurately predict).

Adding data rarely hurts.

**Structured data:** represented in the structured format like tables, databases etc.

Whereas unstructured data does not follow a structured format. E.g. Images, audios, NLP etc.

MLOPS(CampusX)

L