Regression:

In a regression problem our target variable is continuous and there is a linear relationship between independent and dependent variables.
Ex- predicting the housing prices using house size.

1] Linear Regression

Linear regression is a statistical method of finding the relationship between independent and dependent variables.

A] Linear Regression with one variable

Linear regression with one variable is called univariate linear regression.
In univariate linear regression we only have a single input variable and single target variable.

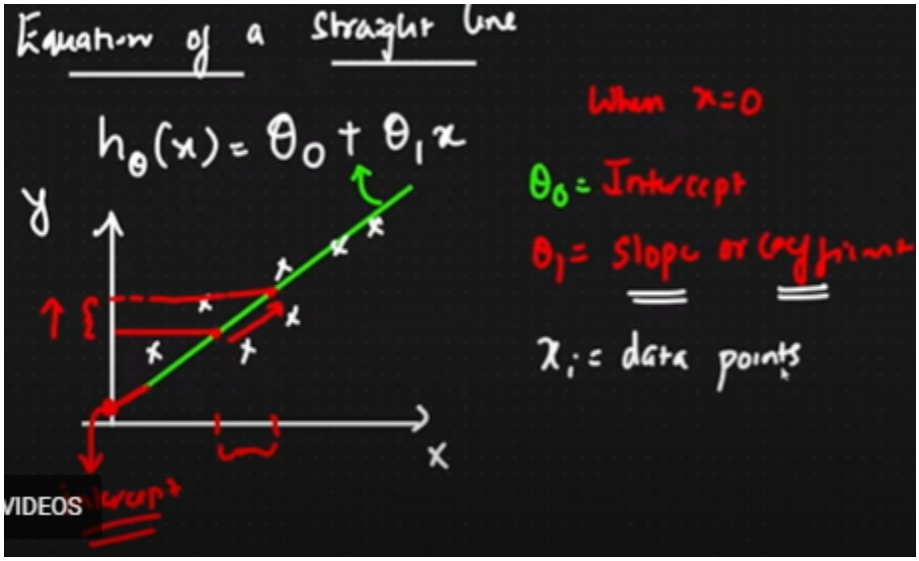
Hypothesis Function

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x$$

Hypothesis function takes input data and converts it into output data.

y^ is predicted value.

Theta zero - This is intercept.it indicates that when x=0 then $\hat{y} = h_\theta(x) = \theta_0$ at what point the line is crossing the y-axis.
Theta One - This is a slope or coefficient.it tells you how much Y changes as X increases by one unit.
when theta one=0 then it indicates there is no relationship between x and y.
When theta one is positive then there is a positive relationship between x and y else negative relationship between x and y.



Interpretation of intercept and slope:

ex-1 - Suppose an independent variable is area in square feet and target variable is price of house.after applying linear regression on this dataset you get the following equation.

$\hat{y}$ = 47588.70+ 93.57x

Intercept - 47588.70

When the value for square feet is zero, the average expected value for price is $47,588.70. (In this case, it doesn't really make sense to interpret the intercept, since a house can never have zero square feet)

Slope - 93.57

For each additional square foot, the average expected increase in price is $93.57.

Example -2 for interpretation of intercept and slope

A school determines that students' marks can be predicted using no of hours they study using regression model y = 50 + 2x where x-is the no of hours student study and y- is their marks. The y-intercept value indicates that a student will get on average 50 marks without studying. The value slope(2) indicates that for each hour of study marks will increase on average by 2 marks.

So, now we know that for each additional square foot, the average expected increase in price is $93.57. To find out if this increase is statistically significant, we need to conduct a hypothesis test for B1 or construct a confidence interval for B1.

Constructing a Confidence Interval for a Regression Slope:

Formula to calculate confidence interval

b1 +/- t_value * (standard error of b1)

b1 - Slope

T- critical value

Standard error - MSE

93.57 +/- (2.228) * (11.45) = (68.06 , 119.08)

This means we are 95% confident that the true average increase in price for each additional square foot is between $68.06 and $119.08.

Cost Function:

The average squared difference between actual and predicted value.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

M - is no of training example.(it is used for calculating average)
½ - is used for simplification while calculating derivatives in gradient descent algorithm.

We can measure the accuracy of our hypothesis using cost function.It will give us the error between actual and predicted value.It is nothing but the average squared difference between actual and predicted value.

This function is also called the "Squared error function", or "Mean squared error".

So we have our hypothesis function and we have a way(cost function) of measuring how well it fits into the data. Now we need to estimate the parameters in hypothesis function. That's where gradient descent comes in.

Gradient Descent algorithm:
The gradient descent algorithm tell us to repeat until convergence or till getting the best parameter.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

Gradient descent algorithm for linear regression:

$$\text{repeat until convergence: } \{$$
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i) x_i)$$
$$\}$$

we are multiplying xi at the end due to the derivative.

Alpha - The Learning Rate is a hyper-parameter that can determine the speed or step size at each iteration while moving towards a minimal point in Gradient Descent. This value should not be too small or too high because if it's too small then it takes too much time to converge and if it's too large then the step size will increase and it moves quickly and never reaches a global minima point even after repeated iterations.

Speed up gradient descent:

We can speed up gradient descent by having each of our input values in the same range using feature scaling.

Standardization(z-normalization)

In Standardization we scale values where mean is 0 and standard deviation is 1 i.e we try to make feature should follow standard normal distribution

$$x_i := \frac{x_i - \mu_i}{s_i}$$

Where,

μi is average of all values of feature
Si is standard deviation

Normalization(min-max scaler)

In normalization we scale values between the range of -1 to 1 or 0 to 1

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Standardization vs Normalization:

We can use normalization when data is not following normal distribution or don't know the distribution and when data is following normal distribution use standardization.
When we have outliers we need to use standardization because we are scaling values using mean and standard deviation but in normalization for scaling we use min and max values.

Ordinary Least square(OLS):

OLS is an alternative method for gradient descent to find the best parameter minimizing loss function.so in the OLS method we use normal equations and this is close form solution.

OLS is an analytical or statistical method and using this method directly u will get theta values after putting all values in the formula.In scikit-learn LinearRegression() uses OLS method.

To use gradient descent in scikit learn will have to use `SGDRegressor()`

Normal Equation:-

It is an alternative for gradient descent.

OLS formula for univariate regression:

The slope b_1 and the intercept b_0 of the sample regression equation are calculated as

$$b_1 = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\Sigma(x_i - \bar{x})^2} \text{ and}$$

$$b_0 = \bar{y} - b_1\bar{x}.$$

OLS formula for multivariate linear regression:Normal equation

$$\theta = (X^T X)^{-1} X^T y$$

Normal equations can be used when we have less than 10000 features.When we have a large features, the normal equation will be slow.It is used to find the best possible parameters without iterations.There is no need to do feature scaling with the normal equation.

where,

$X^T X$ this may be Non-Invertible (non-symmetric) i.e determinant is zero.

Main Causes of Non-Invertible:

1] Redundant features:-add no relevant information to your other features.because they are correlated

2] Too many features.

Solution to the above causes is to delete dependent features(correlation).

Difference between Gradient descent and Normal equation:

Gradient Descent	Normal Equation
Need to choose alpha	No need to choose alpha
Needs many iterations	No need to iterate
O (kn^2)	O (n^3), need to calculate inverse of $X^T X$
Works well when n is large	Slow if n is very large

B] Linear Regression with multiple variable

Linear regression with multiple variables is also known as "multivariate linear regression".

Hypothesis function:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

Note: The sum of the residuals in a linear regression model is 0 since it assumes that the errors (residuals) are normally distributed with an expected value or mean 0, i.e.

Summary of Linear Regression algorithm:

1] Our ultimate goal of the linear regression algorithm is to find the best fit line so that it can do prediction correctly and the difference between actual and predicted value should be very small.
2] Cost function is used to calculate error between actual and predicted value.
3] To get the best fit line we need the best parameter(Theta zero and Theta one) and to calculate the best parameter we use gradient descent algorithm by minimizing the error.

R-Squared and Adjusted R-Square:

R-squared and Adjusted R-squared are two such evaluation metrics both are extremely important to evaluate regression problems.

1] Residual Sum of Squares

Residual is nothing but the difference between the actual and predicted value.
RSS as a whole gives us the variation in the target variable that is not explained by our model.

Residual plots tell us whether the regression model is the right fit for the data or not. It is actually an assumption of the regression model that there is no trend in residual plots.

Using the residual values, we can determine the sum of squares of the residuals also known as Residual sum of squares or RSS.The lower the value of RSS, the better is the model predictions.

$$RSS = \sum (y_i - \hat{y}_i)^2$$

2] Total Sum of Squares

Total Sum of Squares Indicates total variation in target variable
TSS is the sum of squares of the difference between the actual values and their mean.

$$TSS = \sum (y_i - \bar{y})^2$$

3] Explained variation in the target variable:

Σ(ŷi −y¯) ^2 . this is way we can measure the explained variation in y

R-Squared:

R square is an evaluation metric which tell us how much variation of the target variable is explained by the input variable or model.

Now, if TSS gives us the total variation in Y, and RSS gives us the variation in Y not explained by X, then TSS - RSS gives us the variation in Y that is explained by our model! We can simply divide this value by TSS to get the proportion of variation in Y that is explained by the model. And this is our R-squared statistic!

= 1 – Unexplained variation/ Total variation

$$R^2 = 1 - \frac{RSS}{TSS}$$

Interpretation of r-square : if R-square is 0.7, it means 70% of the variation in the output variable is explained by the input variables

Ex- RSS(variance unexplained by model) = 0.3 TSS(total variance in target) = 1
R^2 = 1-0.3/1=0.7 70% variance explained by model

Problem with R-squared:
The problem with r-square is that even if you add unnecessary feature r-square value will be same or increase
This clearly does not make sense because some of the independent variables might not be useful in determining the target variable. Adjusted R-squared deals with this issue.

Adjusted R-Squared:
Adjusted R-squared takes into account the number of independent variables used for predicting the target variable. In doing so, we can determine whether adding new variables to the model actually increases the model fit.

$$Adjusted\ R^2 = \{1 - [\frac{(1 - R^2)(n-1)}{(n-k-1)}]\}$$

n - represents the number of data points in our dataset
k - represents the number of independent variables, and
R - represents the R-squared values determined by the model.

Summary of R-squared & Adjusted R-squared:

R-squared or R2 explains the degree to which your input variables explain the variation of your output / predicted variable. So, if R-square is 0.8, it means 80% of the variation in the output

variable is explained by the input variables. So, in simple terms, higher the R squared, the more variation is explained by your input variables and hence better is your model.

However, the problem with R-squared is that it will either stay the same or increase with addition of more variables, even if they do not have any relationship with the output variables. This is where “Adjusted R square” comes to help. Adjusted R-square penalizes you for adding variables which do not improve your existing model.

Hence, if you are building Linear regression on multiple variables, it is always suggested that you use Adjusted R-squared to judge the goodness of the model. In case you only have one input variable, R-squared and Adjusted R squared would be exactly the same.

Typically, the more non-significant variables you add into the model, the gap in R-squared and Adjusted R-squared increases.

Bias and Variance:

Bias:

Bias is the difference between the average predicted value of our model and the actual value which we are trying to predict.

Bias tell us how well the model fits the data.

"Low " bias occurs when the model' s predicted values are close to real values i.e. model is doing well on the training set and is good enough to mimic the training data distribution. Conversely, "high" bias(underfitting) means the model is not fitting well on the training set.

Variance:

It basically describes how much the model changes based on the changes in the inputs.

If we have "low " variance, we are performing well on the validation set. Conversely, if we have "high" variance(overfitting), we are not performing good enough on the validation set.

Overfitting: Model performs well on training data but fails to perform on test data.
In overfitting model has low bias and high variance.
Underfitting: Model does not perform well on both training and test data.
In underfitting model has high bias and low variance.
Generalized model has low bias and low variance

Solution to overfitting:

If your model is suffering from low bias and high variance, it means your model is performing poorly on unseen data. In such cases, you can try using regularization methods and can try selecting top ' n ' features by looking at feature importance in predicting target variables.

Conversely, if your model is suffering from a high bias and low variance problem, you should try using a more complex model so that underlying patterns from the data can be detected.

Regularization:

Regularization techniques are used to prevent overfitting.

1] Ridge Regularization (L2)

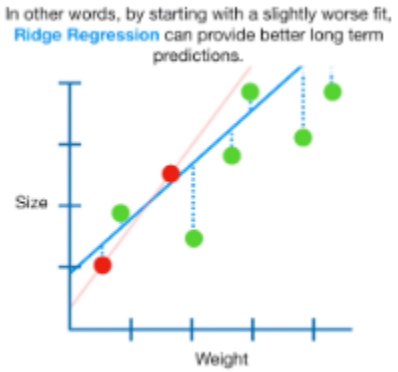
The main idea of Ridge Regression is to fit a new line that doesn't fit the training data. In other words, we introduce a certain Amount of Bias into the new trend line.so the line will be smooth.

$$min_\theta \frac{1}{2m} \left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$$

We add bias because when the model is overfitting our loss is zero so the best fit line will pass through all the data points so if we add some bias into it the line will be steep(won't fit data) and won't pass through all the data points.

In Ridge we add bias with the help of the lambda parameter and also penalize the coefficient with lambda taking the square of the slope.
If lambda is chosen to be too large, it may smooth out the function too much and cause underfitting.

Lambda is the Tuning Parameter that controls the bias-variance tradeoff and we estimate its best value via cross-validation.



2] L1 Lasso Regression (least absolute shrinkage and selection operator)

It is similar to RIDGE REGRESSION except to a very important difference: the Penalty Function now is: lambda*|slope|.

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Lasso penalize unnecessary coefficient to zero

The big difference between Ridge and Lasso starts to be clear when we Increase the value of Lambda. In fact, Ridge can only shrink the slope almost close to zero, while Lasso can shrink the slope all the way to zero. The advantage of this is clear when we have lots of parameters in the model.

In Ridge, when we increase the value of Lambda, the most important parameters might shrink a little bit and the less important parameters stay at high value.
In contrast, with Lasso when we increase the value of Lambda the most important parameters shrink a little bit and the less important parameters go close to zero. So, Lasso is able to exclude silly parameters from the model.

How to choose the value of Lambda?

For a very low value of lambda, an overfitting curve is obtained and for a very high value of lambda, an underfitting or highly biased model is obtained. How then can an optimal value of lambda be achieved?

The answer to this is cross validation. Typically a 10 fold cross validation can help us identify the optimal lambda value

Limitations of L1 and L2:

Ridge does not do feature selection.
In Lasso if no features are more than data points then it selects only no features equal to no of data points.
If two features are correlated then it will randomly select one and remove another one.

Assumptions of Linear Regression:

1] Linear relationship

There should be a linear relationship between dependent variable and independent variable.

How to determine if this assumption is met:

Plot a scatter plot of residual vs predicted(fited) value.This allows you to visually see if there is linear relationship between the two variables

What to do if this assumption is violated

1. Apply a nonlinear transformation(log,square-root) to the independent and/or dependent variable. Common examples include taking the log, the square root, or the reciprocal of the independent and/or dependent variable.
2. Add another independent variable to the model. For example, if the plot of x vs. y has a parabolic shape then it might make sense to add X2 as an additional independent variable in the model.

2] Independence of error

There is not a relationship between the residuals and the variable; in other words, is independent of errors.
There should be no correlation between the residual (error) terms. Absence of this phenomenon is known as Autocorrelation.

How to determine if this assumption is met:

The simplest way to test if this assumption is met is to look at a residual time series plot, which is a plot of residuals vs. time.You can also formally test if this assumption is met using the Durbin-Watson test.

What to do if this assumption is violated

For positive serial correlation, consider adding lags of the dependent and/or independent variable to the model.
For negative serial correlation, check to make sure that none of your variables are overdifferenced.

3] Homoscedasticity

The error terms must have constant variance. This phenomenon is known as homoscedasticity. The presence of non-constant variance is referred to as heteroskedasticity.

How to determine if this assumption is met:

The simplest way to detect heteroscedasticity is by creating a predicted(fited) value vs. residual plot.
Once you fit a regression line to a set of data, you can then create a scatterplot that shows the predicted values of the model vs. the residuals of those predicted values.
Notice how the residuals become much more spread out as the predicted values get larger. This “cone” shape is a classic sign of heteroscedasticity:

What to do if this assumption is violated

Transform(log,square-root) the dependent variable. One common transformation is to simply take the log of the dependent variable.

4] Normality of errors

The residuals of the models are normally distributed.

How to determine if this assumption is met:

Check the assumption visually using Q-Q plots.If residuals are normally distributed all the values will fall on the diagonal line of q-q plot. Data quantile vs Normal quantile

Using the density plot also we can visually check normality.

What to do if this assumption is violated

First, verify that any outliers aren't having a huge impact on the distribution. If there are outliers present, make sure that they are real values and that they aren't data entry errors.

Next, you can apply a nonlinear transformation to the independent and/or dependent variable. Common examples include taking the log, the square root, or the reciprocal of the independent and/or dependent variable.

5] Multicollinearity

The independent variables should not be correlated. Absence of this phenomenon is known as multicollinearity.

How to determine if this assumption is met:

You can use scatter plots to visualize correlation effects among variables. Also, you can also use the VIF factor. VIF value <= 4 suggests no multicollinearity whereas a value of >= 10 implies serious multicollinearity.

What to do if this assumption is violated

We can apply feature engineering like feature1 + feature2 or f1*f2 or f1-f2.

In some cases when two features are highly correlated we can drop one of the features which has less correlation with the target variable.

Evolution metrics in linear regression:

1] Mean Squared Error (MSE):

MSE is calculated as the average squared difference between actual and predicted values.

We can use this metric if we want to give bigger penalization to outliers and apply optimizers who require differentiation. MSE is a differentiable function that makes it easy to perform mathematical operations in comparison to a non-differentiable function like MAE.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Interpretation of MSE: Suppose we have a model that predicts the salary of 4 employees which are $988, $1943, $1239, $2124 where the actual salaries were $1000, $1500, $2000. $2500 respectively.then we got 398 as MSE
This value tells us that the on average model is predicting $398 more or less than the actual value.
On average the error between actual and predicted value will be 398.
Note:Predicted value + or -- MSE
If we have 10 as MSE and got 30 as predicted value now we can say that actual values will be between 20-40

Advantages:
The squaring also has the effect of *punishing the models'* more for larger errors.
The MSE is mostly used as a cost function to find the best parameter by minimizing MSE.

Disadvantage:
The MSE unit is squared.
MSE is squared value so it is not in the same unit as of target values that's why it is not a good practice to use MSE to interpret the evaluation metric result instead we can use RMSE which has the same unit as actual values.

2] Root Mean squared error(RMSE):
RMSE is square root of MSE

$$RMSE = \sqrt{MSE}$$

Importantly, the square root of the error is calculated, which means that the units of the RMSE are the same as the original units of the target value that is being predicted.

For example, if your target variable has the units "dollars," then the RMSE error score will also have the unit "dollars" and not "squared dollars" like the MSE.
As such, it may be common to use MSE loss to train a regression predictive model, and to use RMSE to evaluate and report its performance.

RMSE is highly sensitive to outlier values(an outlier is a data point that differs significantly from other observations). Hence, prior to using this metric, you must remove the outliers from your data set.

Note that the RMSE cannot be calculated as the average of the square root of the mean squared error values.

3] Mean absolute error(MAE)

MAE is a popular metric because, like RMSE, the units of the error score match the units of the target value that is being predicted.

It can be used when we want our model to be robust to outliers, but this metric has the disadvantage of not being differentiable so we can't use it if we want to apply optimizers like Gradient descent.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - x|$$

The MAE does not give more or less weight to different types of errors and instead the scores increase linearly with increases in error.

We can use MAE when we have few or no outliers.

It is a good idea to first establish a baseline MAE for the models.

MAE vs MSE vs RMSE:

MAE: you can use this when you have very few or no outliers in your data or in a better way when you want to ignore the outliers while fitting your model to your data.

MSE/RMSE: you use this when you don't have outliers in your data.

MSE is better when it comes to optimization in gradient descent.

MSE evaluates with squared units which is clearly unjustified so RMSE is better for this.

RMSE is widely used For model evaluation.

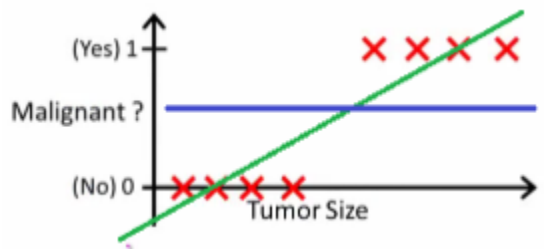
IN housing prices example MAE makes more sense.

Logistic Regression:

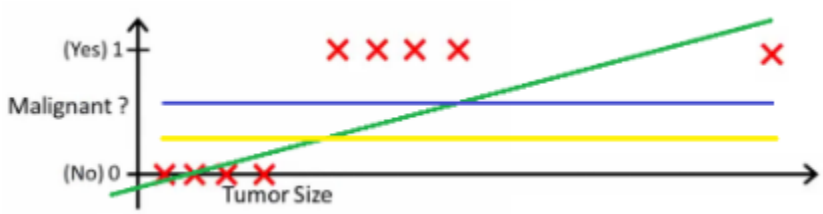
logistic regression is a classification algorithm which is used for binary classification problems. In logistic regression there won't be any linear relationship between IV and DV.

Issue with linear regression:

Now, if we use linear regression to find the best fit line which aims at minimizing the distance between the predicted value and actual value, the line will be like this:



Here the threshold value is 0.5, which means if the value of h(x) is greater than 0.5 then we predict malignant tumor (1) and if it is less than 0.5 then we predict benign tumor (0). Everything seems okay here but now let's change it a bit, we add some outliers in our dataset, now this best fit line will shift to that point. Hence the line will be somewhat like this:



Another problem with linear regression is that the predicted values may be out of range. We know that probability can be between 0 and 1, but if we use linear regression this probability may exceed 1 or go below 0.

To overcome these problems we use Logistic Regression, which converts this straight best fit line in linear regression to an S-curve using the sigmoid function, which will always give values between 0 and 1.

Sigmoid function squeezes a straight line into an S-curve and converts the output into a range of 0 to 1.

We can't use MSE as cost function bcz the logistic function isn't always convex. The logarithm of the likelihood function is however always convex.

Note:

Odds = P / 1 - P P-probability of success range:- 0 to infinity

Log odds or logit function = log(1/ 1- P) range:- - infinity to + infinity

Sigmoid function = sigmoid(logit(p)) = 1/ 1 + e^-z range:- 0 to 1
Sigmoid function maps real values from 0 to 1.

Hypothesis:

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$
$$g(z) = \frac{1}{1+e^{-z}}$$

Cost function(Maximum likelihood function):

If we use linear regression cost function(MSE) then it will give a non-convex graph with many local minima.
In order to solve this problem, we derive a different cost function for logistic regression called log loss which is also derived from the maximum likelihood estimation method.

The main aim of MLE is to find the value of our parameters for which the likelihood function is maximized.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

Notice that when y is equal to 1, then the second term will be zero and will not affect the result.
If y is equal to 0, then the first term will be zero and will not affect the result.
Note: in formula probability of success is p and probability of failure is 1-p

In machine learning we always try to minimize a loss(error) function via gradient descent, rather than maximize an objective function via gradient ascent. If we maximize this above function then we'll have to deal with gradient ascent to avoid this we take negative of this log so that we use gradient descent.

Gradient descent for logistic regression:

$$\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Assumptions of logistic regression:

1] The Response Variable is Binary

Logistic regression assumes that the response variable only takes on two possible outcomes. Some examples include:

2] The Observations are Independent

Logistic regression assumes that the observations in the dataset are independent of each other.

The easiest way to check this assumption is to create a plot of residuals against time (i.e. the order of the observations) and observe whether or not there is a random pattern. If there is *not* a random pattern, then this assumption may be violated.

3] There is No Multicollinearity Among Independent Variables

Multicollinearity occurs when two or more explanatory variables are highly correlated to each other.
The most common way to detect multicollinearity is by using the variance inflation factor (VIF), which measures the correlation and strength of correlation between the predictor variables in a regression model.

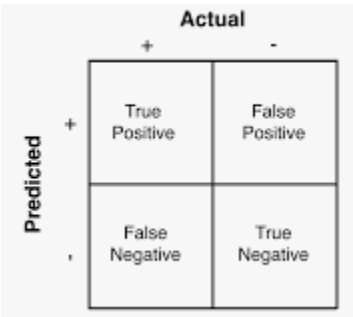
4] There are No Extreme Outliers

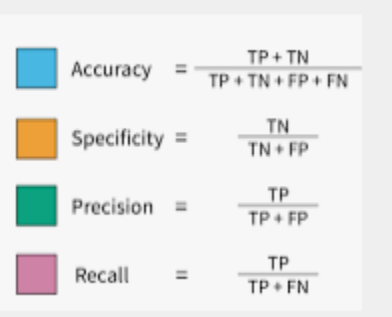
Logistic regression assumes that there are no extreme outliers or influential observations in the dataset.

5] The Sample Size is Sufficiently Large

Classification evaluation metrics:

Confusion matrix:





Precision:it tells us what proportion of predicted positive classes are correct.
It tells us that out of total positive predicted samples how many are actually positive.

Ex- Court Example: Guilty & Innocent

In this example the positive class is Guilty and the Negative class is Innocent.If model predicts a guilty person as Innocent it doesn't matter but if it predicts Innocent person as Guilty that will be the problem so here model falsely predicting positive class. In this case False positives are important and precision is important.

Recall:tells us what proportion of the positive class got correctly classified.
Recall tells us that out of total actual positive samples how many samples the model predicted correctly.
Ex- Cancer example: Cancer & Not cancer.
In this example, the positive class is that the patient has Cancer and the negative class is that the patient doesn't have cancer. Here if the model predicts a person doesn't have cancer but actually he has cancer then that will be a problem.But there is no problem if the model predicts a person has cancer but he doesn't have cancer.
In this case the model is falsely predicting negative class.So False negatives are important and recall should be taken seriously.

Specificity :tells us what proportion of the negative class got correctly classified.

False negative rate:False Negative Rate (FNR) tells us what proportion of the positive class got incorrectly classified by the classifier.

False positive rate:FPR tells us what proportion of the negative class got incorrectly classified by the classifier.

1] ROC AUC curve
(https://mlwhiz.com/blog/2021/02/03/roc-auc-curves-explained/)
ROC curves, or receiver operating characteristic curves, are one of the most common evaluation metrics for checking a classification model's performance

What is the problem with accuracy?
The answer is that accuracy doesn't capture the whole essence of a probabilistic classifier, i.e., it is neither a threshold-invariant metric nor a scale-invariant metric.
Note: Accuracy is sensitive to threshold.

1] Why is accuracy not threshold invariant?

Assuming a threshold of 0.5 for a logistic regression classifier, what do you think the accuracy of this classifier is?

Classifier A	
y_pred	y_true
0.6	0
0.7	0
0.8	1
0.9	1

Ex- Threshold=0.5 then result = 1 1 1 1 but actually we have 0 0 1 1 so it means 50% accuracy. Thres

Assuming 0.5 threshold we are misclassifying two zeros so the accuracy is 50%.
So is our model that bad?now if we assume 0.75 as threshold then our accuracy is 100%
So this is why accuracy is totally dependent on threshold even though the model is predicting the correct result.so we need to find a way where metric should not be sensitive to threshold.

Actually what happens when we use accuracy to evaluate the performance is that
We need to change the threshold multiple times to check how our model is classifying the classes because in some use cases we keep the threshold large to minimize false negatives and vice-versa.
So instead of analyzing many confusion metrics for all the thresholds(0.5,0.6,0.7) we use the ROC curve to evaluate performance of the model.

ROC(receiver operating characteristic) curves:

We plot the ROC curve using true positive rate(TPR) on the y-axis and false positive rate(FPR) on the x-axis.

True positive rate:(TPR or Recall or sensitivity)

True positive rate tells us what proportion of positive class got correctly classified.

$$\text{TPR (sensitivity)} = \frac{TP}{TP + FN}$$

False positive rate:(1-specificity)

False positive rates tell us what proportion of the negative class got incorrectly classified as positive.

$$\text{FPR (1-specificity)} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

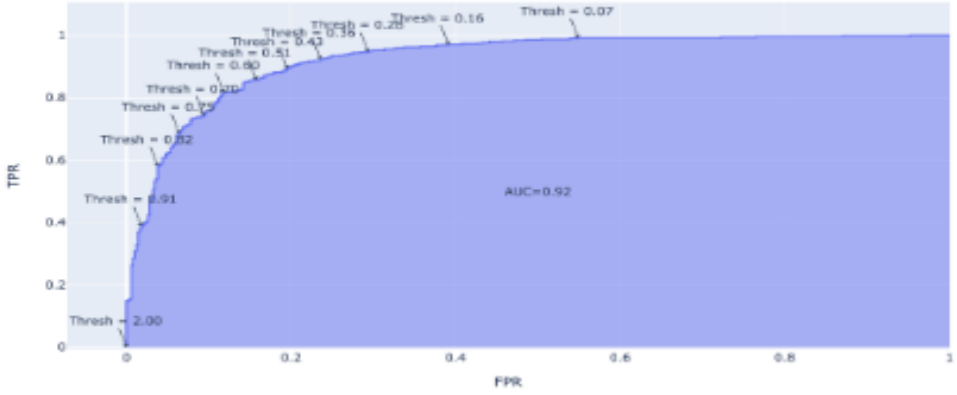
Now we want to evaluate how good our model is using ROC curves. To do this, we need to find FPR and TPR for various threshold values.
In most of the cases we need high TPR and low FPR

How to use the ROC Curve?

We can generally use ROC curves to decide on a threshold value. The choice of threshold value will also depend on how the classifier is intended to be used.
In the cancer example we can select 0.7 threshold because we really don't want to predict "no cancer" for a person who actually has cancer.

Now, what is AUC?

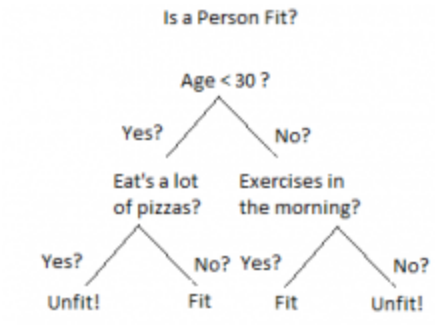
The AUC is the area under the ROC Curve. This area is always represented as a value between 0 to 1 (just as both TPR and FPR can range from 0 to 1), and we essentially want to maximize this area so that we can have the highest TPR and lowest FPR for some threshold.



Decision Tree:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.



Why use Decision Trees?

Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
The logic behind the decision tree can be easily understood because it shows a tree-like structure.

In decision tree the important thing this is how to select which feature should be root node
For this problem we have two methods.
1] Gini impurity
2] Information gain and Entropy

1] Gini impurity(solved example with gini - https://www.youtube.com/watch?v=zNYdkpAcP-g)
Gini impurity is a measure of impurity.lower the gini impurity more the information gain for the feature.The feature which has lower gini impurity we select that feature for split.
CART(classification regression algorithm uses gini impurity)

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Note: A node is pure when all samples belong to one class for example if node has 4 yes & 0 no then this node is pure and if node has 4 yes and 3 no then node is impure.

2] Information gain and Entropy:
(https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html)

Information gain calculates how much information a feature provides us about a class.
Entropy is a measure of impurity.Information gain uses entropy and when information gain is more for the feature we select that feature for split.

A] Entropy:

It is the measure of the amount of uncertainty or randomness in data.
If entropy is 0 means all samples belong to one class and if entropy is 1 then 50% sample belong to one class and another half belongs to other class.If entropy is 1 then it is perfect impure or randomness or uncertainty.

$$H(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

B] Information gain

$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

ID3 Algorithm will perform following tasks recursively

1. Create root node for the tree
2. If all examples are positive, return leaf node 'positive'
3. Else if all examples are negative, return leaf node 'negative'
4. Calculate the entropy of current state H(S)
5. For each attribute, calculate the entropy with respect to the attribute 'x' denoted by H(S, x)
6. Select the attribute which has maximum value of IG(S, x)
7. Remove the attribute that offers highest IG from the set of attributes
8. Repeat until we run out of all attributes, or the decision tree has all leaf nodes.

Ensembling:

Ensemble simply means combining multiple models. Thus a collection of models is used to make predictions rather than an individual model.so,they are less prone to error, and therefore tend to perform better.

Two types of ensembling:

1] **Bagging**– It creates a different training subset(Bootstrapped data) from sample training data with replacement(meaning that the individual data points can be chosen more than once) & the final output is based on majority voting. For example, Random Forest.

2] **Boosting**– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, AdaBOOST, XGBOOST

Random Forest:

(https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/)

What is the bootstrap method?

This method involves drawing of sample data repeatedly with replacement from the data.

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees by creating a bootstrap dataset and considering a random subset of features at each node or step and then takes a majority vote for classification and average in case of regression.

Deep-dive into Bagging:

Bagging:

Bagging, also known as Bootstrap Aggregation, is the ensemble technique used by random forests. In this method we choose bootstrapped samples and select random features to build multiple decision trees then aggregate results based on majority voting.

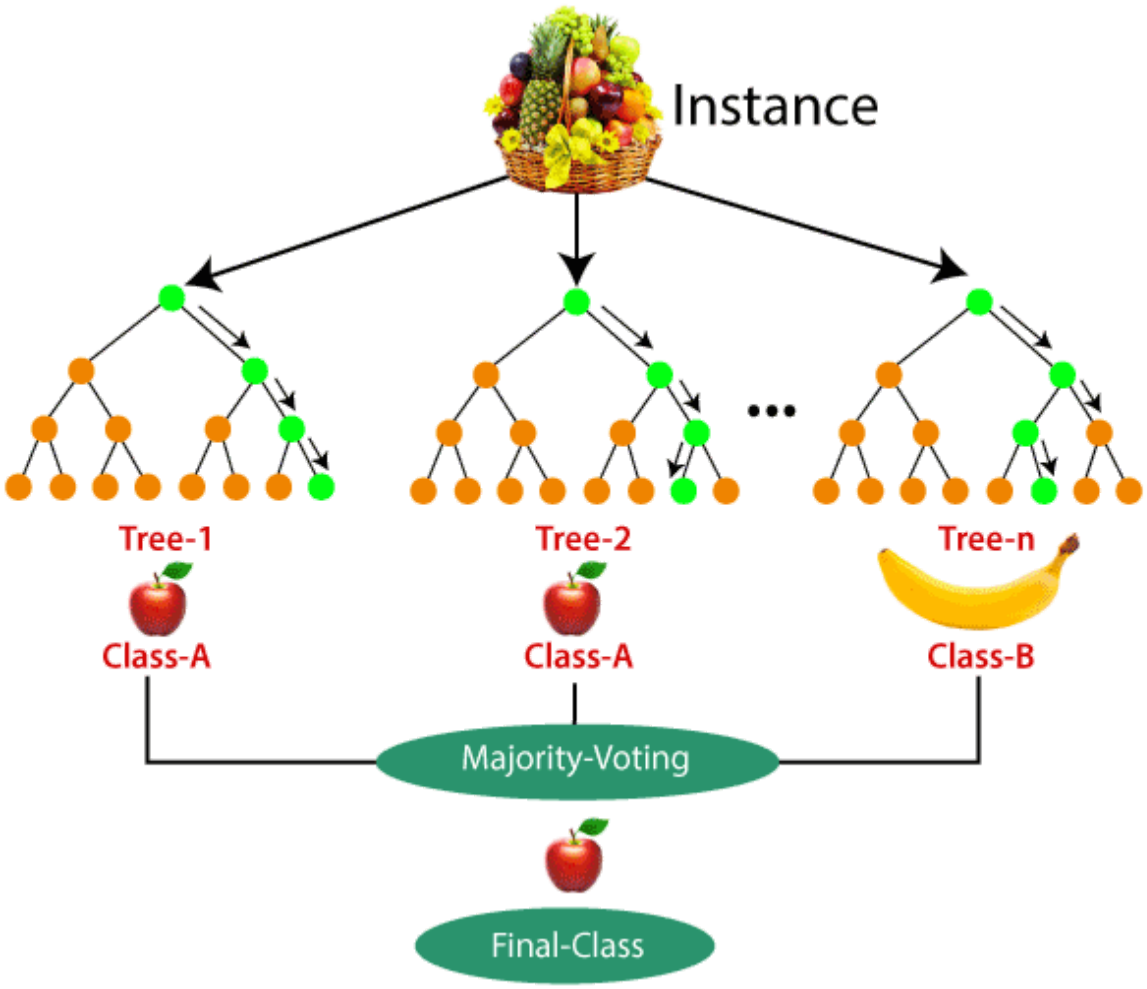
This step of randomly selecting data samples with replacement is called bootstrap. The bootstrapped data size is also the same as original data but in Bootstrapped data samples are repeated.

In summary, a random forest aggregates the predictions of bootstrapped decision trees. This general ensemble method is known in machine learning as bagging.

Note: While selecting random samples By default, random forest classifiers select from the square root of the total number of features when looking for a split. So, if there are 100 features (columns), each decision tree will only consider 10 features when choosing a split.

What is random in Random Forest?

Random forest is called random because we randomly select features while creating a decision tree.



Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:

Very important point:
In a random forest while creating a decision tree we consider a random subset of features at each node or step of the decision tree.
Ex- In the first tree at the first node we might take 3 features randomly out of 6 and in the next node of the same tree we might take other 3 features randomly excluding the feature one which is used at node one.This is how decision trees are made up in random forest.

Steps involved in random forest algorithm:

Step 1:Create a bootstrap dataset:
To create a bootstrap dataset that is the same size as the original we just randomly select samples from the original dataset.The important thing is we are allowed to take one sample more than once that is why it is called bootstrapping.

Step 2: Create a decision tree using the bootstrapped dataset but only use a random subset of features at each node or step.

Step 3: repeat the step1 and step 2
Make a new bootstrapped dataset and build a tree considering a subset of variables at each step.
Now if we get a new sample for prediction then what we do is we iterate through all the trees and note down all the trees outcome then decide the outcome based on the majority of the outcomes.

Very important point regarding out of bag samples:
If you remember while creating bootstrap dataset we were selecting samples repeatedly
So there are few samples which we don't consider while building a decision tree.
So these samples are called out of bag samples and these samples are used to evaluate random forest or to calculate accuracy of the model.
Ex- correctly classified out of bag samples / total out of bag samples

Advantages:
1. Robust to outlier(least important)
2. It solves the problem of overfitting as output is based on majority voting or averaging.
3. It performs well even if the data contains null/missing values.
4. Each decision tree created is independent of the other thus it shows the property of parallelization.
5. It is highly stable as the average answers given by a large number of trees are taken.
6. It maintains diversity as all the features are not considered while making each decision tree though it is not true in all cases.
7. It is immune to the curse of dimensionality. Since each tree does not consider all the features, so feature space is reduced.
8. We don't have to segregate data into train and test as there will always be 30% of the data which is not seen by the decision tree made out of bootstrap.

Disadvantages:
1. Random forest is highly complex when compared to decision trees where decisions can be made by following the path of the tree.

2. Training time is more compared to other models due to its complexity. Whenever it has to make a prediction each decision tree has to generate output for the given input data

Boosting:

It combines weak learners into strong learners by minimizing residuals and creating sequential models such that the final model has the highest accuracy. For example, ADABOOST, XGBOOST Boosting, by contrast, learns from the mistakes of individual trees. The general idea is to adjust new trees based on the errors of previous trees.

Bagging vs Boosting:

In boosting, correcting errors for each new tree is a unique approach from bagging.In a bagging model, new trees pay no attention to previous trees.
Also, In bagging new trees are built from scratch using bootstrapping, and the final model aggregates all individual trees. In boosting, however, each new tree is built from the previous tree

The principle behind the boosting algorithm is that first we train the model and whichever samples are classified incorrectly we pass them to the next model and we do this until and unless we correctly classify all the samples.

Adaboost algorithm(Adaptive boosting):

Adaboost uses a decision tree with 1 depth which is called decision stumps.

Summary of Adaboost:

Adaboost builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lower error is received.

Working of Adaboost:

Consider the following binary classification problem where we are predicting illness.

Row No.	Gender	Age	Income	Illness	Sample Weights
1	Male	41	40000	Yes	1/5
2	Male	54	30000	No	1/5
3	Female	42	25000	No	1/5
4	Female	40	60000	Yes	1/5
5	Male	46	50000	Yes	1/5

Step 1: Initially we are assigning the weights to all the sample i.e 1/ no samples

Step 2: Now we calculate gini impurity of all the features and make a first decision stump selecting the feature which has low gini impurity.
Now we pass all the samples through this decision stump and see how many samples are classified correctly and incorrectly.

Step 3: Now we calculate the total error which is nothing but summation of all the sample weights of misclassified data points. **Note:** Total error will always be between 0 and 1.
Here in our dataset let's assume there is 1 wrong output, so our total error will be ⅕

Step 4: calculate performance of stump

$$Performance\ of\ the\ stump\ =\ \frac{1}{2}\log_e(\frac{1-Total\ Error}{Total\ Error})$$

Performance of stump can denoted by alpha:In our example alpha is:

$$\alpha\ =\ 0.69$$

Total error = 0 then there is no misclassification
Total error = 0.5 then hals samples are correct and half samples are incorrect.
Total error = 1 then all samples are misclassified.

We are calculating the TE and performance of a stump to update the weights because if the same weights are applied to the next model, then the output received will be the same as what was received in the first model.

So after updating the weights the wrong predictions will be given more weight whereas the correct predictions weights will be decreased. Now when we build our next model after updating the weights, more preference will be given to the points with higher weights.

Step 5 : updating the weights
- The amount of say (alpha) will be negative when the sample is correctly classified.
- The amount of say (alpha) will be positive when the sample is miss-classified.

$$New\ sample\ weight\ =\ old\ weight\ *\ e^{\pm Amount\ of\ say\ (\alpha)}$$

New weights for *correctly classified* samples are:

$$New\ sample\ weight\ =\ \frac{1}{5}\ *\ \exp(-0.69)$$

$$New\ sample\ weight\ =\ 0.2\ *\ 0.502\ =\ 0.1004$$

For *wrongly classified* samples the updated weights will be:

$$New\ sample\ weight\ =\ \frac{1}{5}\ *\ \exp(0.69)$$

$$New\ sample\ weight\ =\ 0.2\ *\ 1.994\ =\ 0.3988$$

Note: Now we have updated the weights of the samples and if you notice the mis-classified sample got high weight.

Row No.	Gender	Age	Income	Illness	Sample Weights	New Sample Weights
1	Male	41	40000	Yes	1/5	0.1004
2	Male	54	30000	No	1/5	0.1004
3	Female	42	25000	No	1/5	0.1004
4	Female	40	60000	Yes	1/5	0.3988
5	Male	46	50000	Yes	1/5	0.1004

We know that the total sum of the sample weights must be equal to 1 but here it is 0.8004.To bring this sum equal to 1 we will normalize these weights by dividing the total sum of updated weights that is 0.8004. So, after normalizing the sample weights now we get the sum is equal to 1.

Step 6: Now we need to make a new dataset to see if the errors decreased or not. For this we will remove the “sample weights” and “new sample weights” column and then based on the “new sample weights” we will divide our data points into buckets.

Row No.	Gender	Age	Income	Illness	New Sample Weights	Buckets
1	Male	41	40000	Yes	0.1004/0.8004= 0.1254	0 to 0.1254
2	Male	54	30000	No	0.1004/0.8004= 0.1254	0.1254 to 0.2508
3	Female	42	25000	No	0.1004/0.8004= 0.1254	0.2508 to 0.3762
4	Female	40	60000	Yes	0.3988/0.8004= 0.4982	0.3762 to 0.8744
5	Male	46	50000	Yes	0.1004/0.8004= 0.1254	0.8744 to 0.9998

Step 7 : Now what the algorithm does is selects random numbers from 0-1.
Since incorrectly classified records have higher sample weights and larger bucket size so the probability to select those records is very high.
Suppose the 5 random numbers our algorithm takes are 0.38,0.26,0.98,0.40,0.55.

Now we will see where these random numbers fall in the bucket and according to it, we'll make our new dataset shown below.

Row No.	Gender	Age	Income	Illness
1	Female	40	60000	Yes
2	Male	54	30000	No
3	Female	42	25000	No
4	Female	40	60000	Yes
5	Female	40	60000	Yes

This comes out to be our new dataset and we see the datapoint which was wrongly classified has been selected 3 times because it has a higher weight.

Step 9 – Now this act as our new dataset and we need to repeat all the above steps i.e.

1. Assign equal weights to all the data points
2. Find the stump that does the best job classifying the new collection of samples by finding their Gini Index and selecting the one with the lowest Gini index
3. Calculate the "Total error" to update the previous sample weights.
4. Normalize the new sample weights.

Iterate through these steps until and unless a low training error is achieved.
Suppose with respect to our dataset we have constructed 3 decision trees (DT1, DT2, DT3) in a sequential manner. If we send our test data now it will pass through all the decision trees and finally, we will see which class has the majority, and based on that we will do predictions for our test dataset.

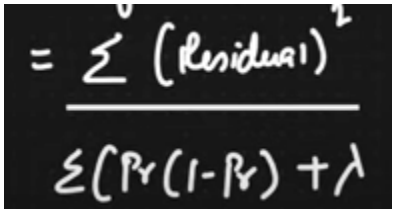
XGBoost:Extreme gradient boosting
(https://www.geeksforgeeks.org/xgboost/)

XGBoost is short for Extreme Gradient Boosting. The Extreme part refers to pushing the limits of computation to achieve gains in accuracy and speed.
Note: The key idea behind gradient boosting is to use gradient descent to minimize the errors of the residuals.
XGBoost is an implementation of Gradient Boosted decision trees.

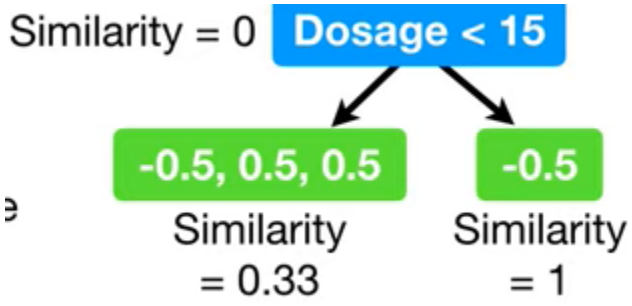
Steps in XGBoost Classification:

Step 1 : Considering base models prediction(0.5) calculate residuals.
Ex- 0-0.5=-0.5 or 1-0.5=0.5
Step 1: Select random features and create a decision tree then assign the residuals to each node.

Step 2: calculate similarity weights for each node using following formula:
Lambda is a regularization parameter which is used to reduce overfitting.



After three steps we can see tre like following:

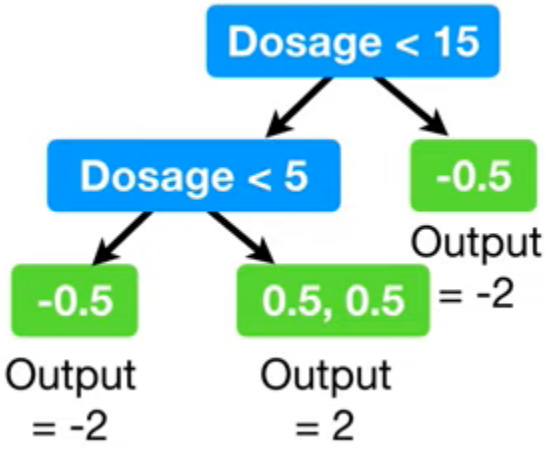


Step 3: Calculate the information gain at each node and use it for the next split.
Whichever node has more information gain we use that node for the next splitting.
Information Gain = left_node simalrity_weight + right_node sw - root_node sw

Note: For Tree pruning we use parameter gamma and calculate IG-gamma (55-130)and if this calculation comes as negative we prune that branch and if the calculation comes as positive we don't prune.

Note: While doing the next split the threshold Dosage<5 has high IG so we selected that for splitting.

After performing the above three steps we got the below tree and the final output at each node is nothing but the similarity weight of the node.



Step 5:while doing prediction we do that in the following way.

$$\log(\text{odds}) \text{ Prediction} = 0 + (0.3 \times -2) = -0.6$$

0- is base prediction
0.3 is learning rate
-2 - is output from the above tree
0.6 is our output but we apply the sigmoid function to this output because sometimes this value can be above 1 or below 0.we do that in the following way.

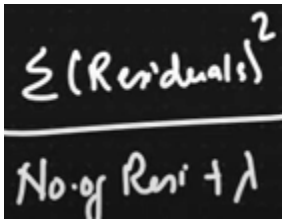
$$\text{Probability} = \frac{e^{0.6}}{1 + e^{0.6}} = 0.65$$

So now we get new predictions and using these calculations we calculate new residuals and then create new trees and assign the new residual to each node and then calculate similarity weights at each node but here the probability value will be different which is for ex-0.65.
Will have to do these things until and unless we get zero or less residuals.

XGBoost Regressor:

Xgboost for regression also works the same as classification but there are few changes.

1] The base prediction is the mean of the target column instead 0.5
2] Similarity weight formula is:



3] In the final step we don't use the sigmoid function in regression.
All remaining steps are the same like xgboost classification.

Difference between Random Forest and XGBoost:
(Interview questions:https://www.mlstack.cafe/blog/random-forest-interview-questions)

1] Random forests builds each tree independently while gradient boosting builds one tree at a time

2] Random forests combine results at the end of the process (by averaging or “majority rules”) while gradient boosting combines results along the way.

Unsupervised Machine Learning

K-Means Clustering:
(https://neptune.ai/blog/k-means-clustering)

K-Means is an unsupervised ml algo where the target variable is not present. It groups similar kinds of data into different groups called clusters and this whole process is called clustering.

K-means is a centroid-based clustering algorithm, where we calculate the distance between each data point and a centroid to assign it to a cluster. The goal is to identify the K number of groups in the dataset.

It is an iterative process of assigning each data point to the groups and slowly data points get clustered based on similar features. The objective is to minimize the sum of distances between the data points and the cluster centroid, to identify the correct group each data point should belong to.

How does K-means work?

Step 1: Choosing the number of clusters

The first step is to define the K number of clusters in which we will group the data.

Step 2: Initializing centroids

Centroid is the center of a cluster but initially, the exact center of data points will be unknown so, we select random data points and define them as centroids for each cluster.

Step 3: Assign data points to the nearest cluster

Now that centroids are initialized, the next step is to assign data points Xn to their closest cluster centroid Ck

In this step, we will first calculate the distance between data point X and centroid C using the Euclidean Distance metric.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

And then choose the cluster for data points where the distance between the data point and the centroid is minimum.

Step 4: Re-initialize centroids .

we will re-initialize the centroids by calculating the average of all data points of that cluster.

Step 5: Repeat steps 3 and 4

We will keep repeating steps 3 and 4 until we have optimal centroids and the assignments of data points to correct clusters are not changing anymore.

K-means follows the Expectation-Maximization approach to solve the problem. The Expectation-step is used for assigning the data points to the closest cluster and the Maximization-step is used for computing the centroid of each cluster.

While working with K-means algorithm we need to take care of the following things –

1] It is recommended to standardize the data because this algorithm uses distance-based measurement to determine the similarity between data points.

2] Due to the iterative nature of K-Means and random initialization of centroids, K-Means may stick in a local optimum and may not converge to global optimum. That is why it is recommended to use different initializations of centroids.

How to choose K?

Selecting a lower number of clusters will result in underfitting while specifying a higher number of clusters can result in overfitting. Unfortunately, there is no definitive way to find the optimal number.

Elbow method:

When the number of clusters, K is increased, the distance from centroid to data points will be decreased and will reach a point where K is the same as the number of data points. This is the reason we have been using the mean of the distance to the centroids.

In the elbow method we plot within cluster sum of squares(WCSS) on the y-axis and k values on the x-axis.
When we consider k=1 then WCSS will be high because the distance from all the points to that single cluster will be more and as we increase k values wcss will decrease because when more centroids come the distance between data points will be small.

Evaluation metric for clustering:

The average Silhouette score is also used as an evaluation measure in clustering. The best silhouette score is 1 and the worst is -1. Values close to zero indicate that data points are on the boundary i.e overlapping the clusters. A negative score [-1, 0] indicates that the samples might have got assigned to the wrong clusters.

Mean distance between the observation and all other data points in the same cluster. This distance can also be called the mean intra-cluster distance. The mean distance is denoted by a

Mean distance between the observation and all other data points of the next nearest cluster. This distance can also be called the mean nearest-cluster distance. The mean distance is denoted by b.

Silhouette score, S, for each sample is calculated using the following formula:

$$S = \frac{(b-a)}{max(a,b)}$$

Hierarchical clustering:

(https://www.javatpoint.com/hierarchical-clustering-in-machine-learning)

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster.

In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.

The hierarchical clustering technique has two approaches:

1] **Agglomerative:** It is a bottom-up approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
2] **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a top-down approach

Why hierarchical clustering?

In K-means algorithm predetermined number of clusters, and it always tries to create the clusters of the same size. To solve these two challenges,we can opt for the hierarchical clustering algorithm because, in this algorithm, we don't need to have knowledge about the predefined number of clusters.

Agglomerative Hierarchical clustering

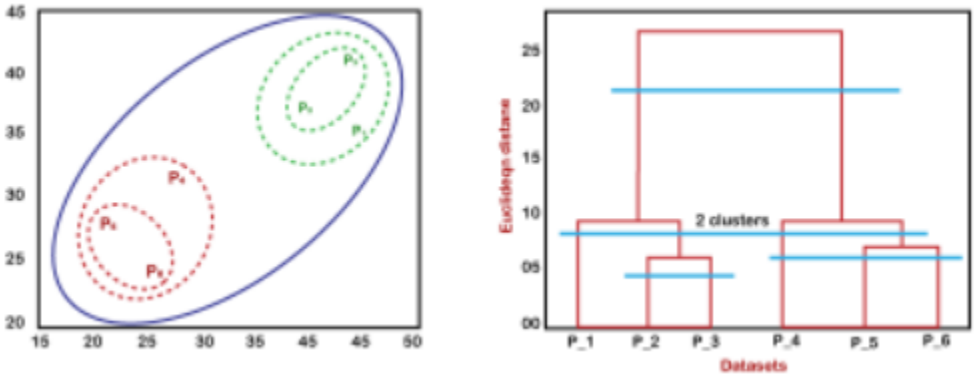
Step-1: Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.
Step-2: Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.
Step-3: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.
Step-4: Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:

Step-5: Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

Measure for the distance between two clusters:
The **closest distance** between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering.

Single Linkage: It is the Shortest Distance between the closest points of the clusters.
Complete Linkage: It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.
Centroid Linkage: It is the linkage method in which the distance between the centroid of the clusters is calculated.

Dendrograms:
In the dendrogram plot, the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.



At last, the final dendrogram is created that combines all the data points together.
Firstly, the data points P2 and P3 combine together and form a cluster, correspondingly a dendrogram is created.
Now to decide how many clusters to create we need to find the longest vertical line that has no horizontal line passed through it.

Note: When there is a large dataset HC takes more time than K-means.

Disadvantage of HC:
They don't handle outliers well. Whenever outliers are found, they will end up as a new cluster, or sometimes result in merging with other clusters.

DBSCAN Clustering(Density based spatial clustering of application with noise):

(https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/#h2_4)
(**example-**https://www.coursera.org/lecture/machine-learning-with-python/dbscan-B8ctK)

Density based clustering is an unsupervised ML algorithm for doing clustering.

It groups 'densely grouped' data points into a single cluster. It can identify clusters in large spatial datasets by looking at the local density of the data points

The most exciting feature of DBSCAN clustering is that it is robust to outliers. It also does not require the number of clusters to be told beforehand, unlike K-Means, where we have to specify the number of centroids.

Why do we need DBSCAN clustering?

K-Means and Hierarchical Clustering both **fail** in creating clusters of arbitrary shapes(easily not separable data). They are not able to form clusters based on varying densities. That's why we need DBSCAN clustering.

Working of DBSCAN:

DBSCAN requires only two parameters: epsilon and minPoints.

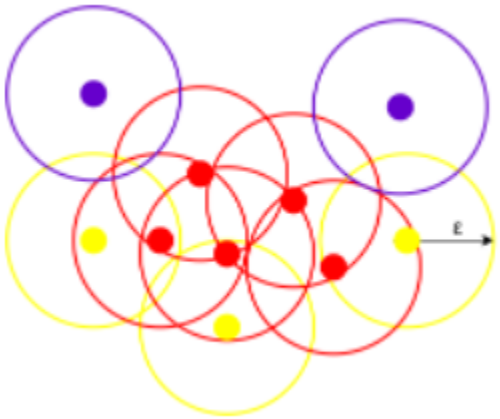
Epsilon: It is the radius of the circle to be created around each data point to check the density
minPoints : it is the minimum number of data points required inside that circle for that data point to be classified as a Core point.

Let's understand it with the help of an example.

we have some data points represented by gray color. Let's see how DBSCAN clusters these data points.

DBSCAN creates a circle of epsilon radius around every data point and classifies them into Core point, Border point, and Noise.

Core point: A data point is a Core point if the circle around it contains at least 'minPoints' number of points.
Border point: If the number of points is less than minPoints, then it is classified as Border Point
Noise: if there are no other data points around any data point within epsilon radius, then it is treated as Noise.



The above figure shows us a cluster created by DBCAN with minPoints = 3. Here, we draw a circle of equal radius epsilon around every data point. These two parameters help in creating spatial clusters.

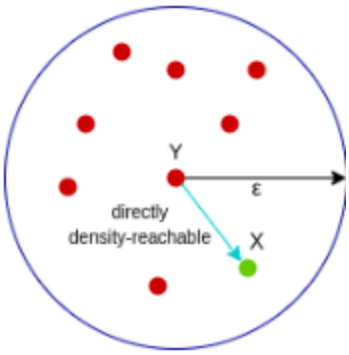
All the data points with at least 3 points in the circle including itself are considered as Core points represented by red color.
All the data points with less than 3 but greater than 1 point in the circle including itself are considered as Border points. They are represented by yellow.
Finally, data points with no point other than itself present inside the circle are considered as Noise represented by the purple color.

Reachability and Connectivity

These are the two concepts that you need to understand before moving further. Reachability states if a data point can be accessed from another data point directly or indirectly, whereas Connectivity states whether two data points belong to the same cluster or not.

A point X is directly density-reachable from point Y w.r.t epsilon, minPoints if,

1. X belongs to the neighborhood of Y, i.e, dist(X, Y) <= epsilon
2. Y is a core point



Note: For locating data points in space, DBSCAN uses Euclidean distance.

Parameter Selection in DBSCAN Clustering:

The value of minPoints should be at least one greater than the number of dimensions of the dataset.

$$minPoints >= Dimensions + 1.$$

It does not make sense to take minPoints as 1 because it will result in each point being a separate cluster. Therefore, it must be at least 3. Generally, it is twice the dimensions. But domain knowledge also decides its value

The value of epsilon can be decided from the elbow method.

Summary:

In DBSCAN it randomly selects one point and using an epsilon value it creates a circle if no of samples inside this circle equal to min_sample then this point is called core point likewise again it will select another random point and create a circle with radius of epsilon value and likewise it mark this point as core point or border point or noise.It will do this process iteratively and at the end it will join all the core points and select all border point also to create cluster.so based on joining of core points it may create 2,3,4,5,n clusters.

Dimensionality Reduction:

Motivation behind dimensionality reduction:

Motivation I: Data Compression

1] We may want to reduce the dimension of our features if we have a lot of redundant data.

2] To do this, we find two highly correlated features, plot them, and make a new line that seems to describe both features accurately. We place all the new features on this single line.
Doing dimensionality reduction will reduce the total data we have to store in computer memory and will speed up our learning algorithm.

Note: in dimensionality reduction, we are reducing our features rather than our number of examples. Our variable m will stay the same size; n, the number of features each example

Motivation II: Visualization

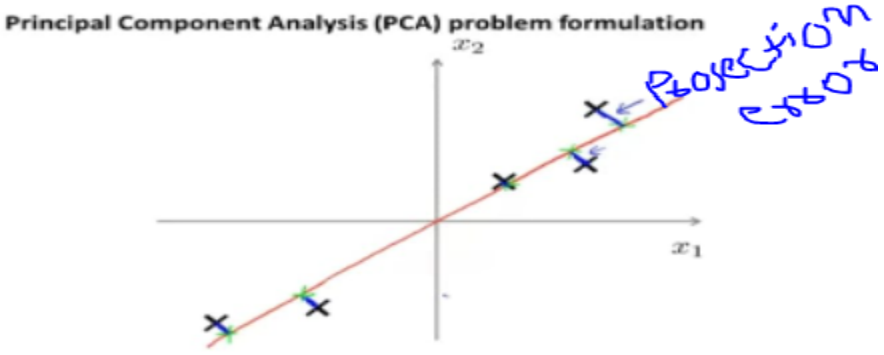
It is not easy to visualize data that is more than three dimensions. We can reduce the dimensions of our data to 3 or less in order to plot it.

Principal component analysis(PCA):

The most popular dimensionality reduction algorithm is Principal Component Analysis (PCA)

Goal of PCA:

Given two features x1 and x2 we want to find a single line that effectively describes both features at once. We then map our old features onto this new line to get a new single feature.



Red line is the projected line and black points are the original features and after projecting features onto the new line the green points will be our new feature.Blue line is the projection error.

The same can be done with three features, where we map them to a plane.

The goal of PCA is to reduce the average of all the distances of every feature to the projection line. This is the projection error

Reduce from n-dimension to k-dimension: Find k vectors(u1,u2, - - - - un) onto which to project the data so as to minimize the projection error.

Note: The Red projection line line is nothing but a vector. There might be another vector but we have chosen this vector because we are getting low projection error.This is the 2 dimensional data so we drown one projection line likewise in multidimensional data we will have more projection lines.

Steps in PCA:
Step 1: feature scaling using mean normalization(x-u/s)
Step 2: Compute covariance matrix
Step 3: Compute eigenvectors from covariance matrix using SVD(single value decomposition). These vectors are our projection line direction.
Step 4: compute the projected data points using eigenvectors and original feature values by using the following formula.

$$z^{(i)} = Ureduce^T \cdot x^{(i)}$$

Steps summary:

```
Sigma = (1/m) * X' * X; % compute the covariance matrix
[U,S,V] = svd(Sigma);   % compute our projected directions
Ureduce = U(:,1:k);     % take the first k directions
Z = X * Ureduce;        % compute the projected data points
```

Choosing the Number of Principal Components:
One way to choose k is by using the following formula:

- Given the average squared projection error: $\frac{1}{m}\sum_{i=1}^{m}||x^{(i)} - x_{approx}^{(i)}||^2$
- Also given the total variation in the data: $\frac{1}{m}\sum_{i=1}^{m}||x^{(i)}||^2$
- Choose k to be the smallest value such that: $\frac{\frac{1}{m}\sum_{i=1}^{m}||x^{(i)} - x_{approx}^{(i)}||^2}{\frac{1}{m}\sum_{i=1}^{m}||x^{(i)}||^2} \leq 0.01$

In other words, the squared projection error divided by the total variation should be less than one percent, so that 99% of the variance is retained.

Bad use of PCA:

Trying to prevent overfitting. We might think that reducing the features with PCA would be an effective way to address overfitting. It might work, but is not recommended because it does not consider the values of our results y. Using just regularization will be at least as effective.

Don't assume you need to do PCA. Try your full machine learning algorithm without PCA first. Then use PCA if you find that you need it.