What is the KNN Algorithm?

KNN(K-nearest neighbours) is a **supervised** learning and **non-parametric** algorithm that can be used to solve both classification and regression problem statements.

2. Why is KNN a non-parametric Algorithm?

The term “**non-parametric”** refers to not making any assumptions on the underlying data distribution. These methods do not have any fixed numbers of parameters in the model.

What is “K” in the KNN Algorithm?

K represents the number of nearest neighbours you want to select to predict the class of a given item, which is coming as an unseen dataset for the model.

Why is the odd value of “K” preferred over even values in the KNN Algorithm?

The odd value of K should be preferred over even values in order to ensure that there are no ties in the voting. If the square root of a number of data points is even, then add or subtract 1 to it to make it odd.

Is Feature Scaling required for the KNN Algorithm? Explain with proper justification.

Yes, feature scaling is required to get the better performance of the KNN algorithm.

What is space and time complexity of the KNN Algorithm?

Time complexity:

The distance calculation step requires quadratic time complexity, and the sorting of the calculated distances requires an **O(N log N)** time. Together, we can say that the process is an **$O(N^3 \log N)$** process, which is a monstrously long process.

Can the KNN algorithm be used for regression problem statements?

Yes, KNN can be used for regression problem statements.

In other words, the KNN algorithm can be applied when the dependent variable is continuous. For regression problem statements, the predicted value is given by the average of the values of its k nearest neighbours.

Why is the KNN Algorithm known as Lazy Learner?

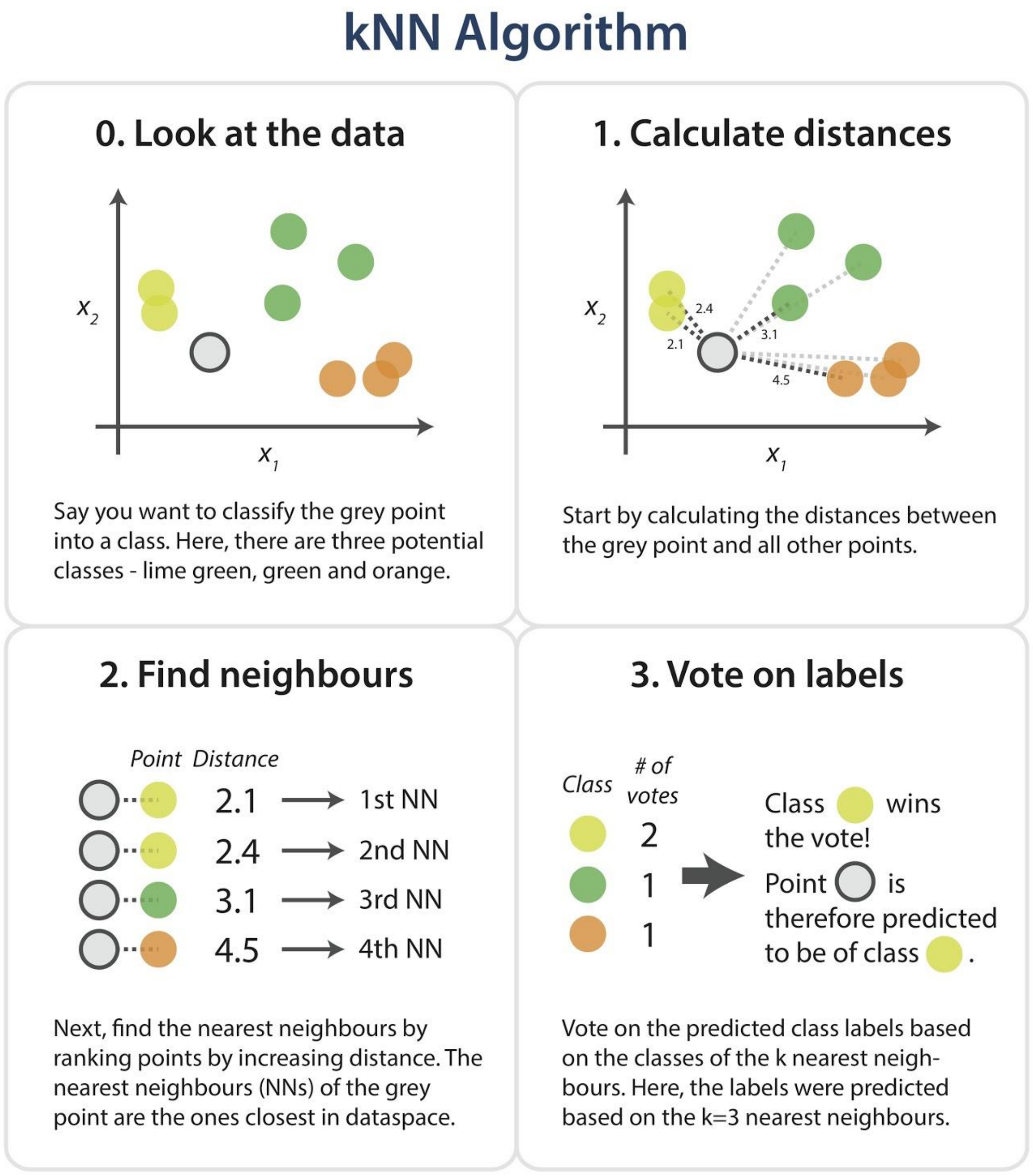
When the KNN algorithm gets the training data, it does not learn and make a model, it just stores the data. Instead of finding any discriminative function with the help of the training data, it follows **instance-based learning** and also uses the training data when it actually needs to do some prediction on the unseen datasets.

5. How does the KNN algorithm make the predictions on the unseen dataset?

The following operations have happened during each iteration of the algorithm. For each of the unseen or test data point, the kNN classifier must:

Step-1: Calculate the distances of test point to all points in the training set and store them

Step-2: Sort the calculated distances in increasing order
Step-3: Store the K nearest points from our training dataset
Step-4: Calculate the proportions of each class
Step-5: Assign the class with the highest proportion


kNN Algorithm
0. Look at the data
x_2
x_1
Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.
1. Calculate distances
2.4
2.1
3.1
4.5
Start by calculating the distances between the grey point and all other points.
2. Find neighbours
Point Distance
2.1 → 1st NN
2.4 → 2nd NN
3.1 → 3rd NN
4.5 → 4th NN
Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.
3. Vote on labels
Class
# of votes
2
1
1
Class wins the vote!
Point is therefore predicted to be of class .
Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

Why is it recommended not to use the KNN Algorithm for large datasets?
The Problem in processing the data:

KNN works well with smaller datasets because it is a lazy learner. It needs to store all the data and then make a decision only at run time. It includes the computation of distances for a given point with all other points. So if the dataset is large, there will be a lot of processing which may adversely impact the performance of the algorithm.

Sensitive to noise:

Another thing in the context of large datasets is that there is more likely a chance of noise in the dataset which adversely affects the performance of the KNN algorithm since the KNN algorithm is sensitive to the noise present in the dataset.

How to handle categorical variables in the KNN Algorithm?

To handle the categorical variables we have to create **dummy variables** out of a categorical variable and include them instead of the original categorical variable. Unlike regression, create k dummies instead of (k-1).

For example, a categorical variable named **"Degree"** has 5 unique levels or categories. So we will create 5 dummy variables. Each dummy variable has 1 against its degree and else 0.

How to choose the optimal value of K in the KNN Algorithm?

There is no straightforward method to find the optimal value of K in the KNN algorithm.

You have to play around with different values to choose which value of K should be optimal for my problem statement. Choosing the right value of K is done through a process known as **Hyperparameter Tuning**.

The optimum value of K for KNN is **highly dependent on the data** itself. In different scenarios, the optimum K may vary. It is more or less a hit and trial method.

As the value of K increases, the error usually goes down after each one-step increase in K, then stabilizes, and then raises again. Finally, pick the optimum K at the beginning of the stable zone. This technique is also known as the **Elbow Method.**

How can you relate KNN Algorithm to the Bias-Variance tradeoff?

Problem with having too small K:

The major concern associated with small values of K lies behind the fact that the smaller value causes noise to have a higher influence on the result which will also lead to a large variance in the predictions.

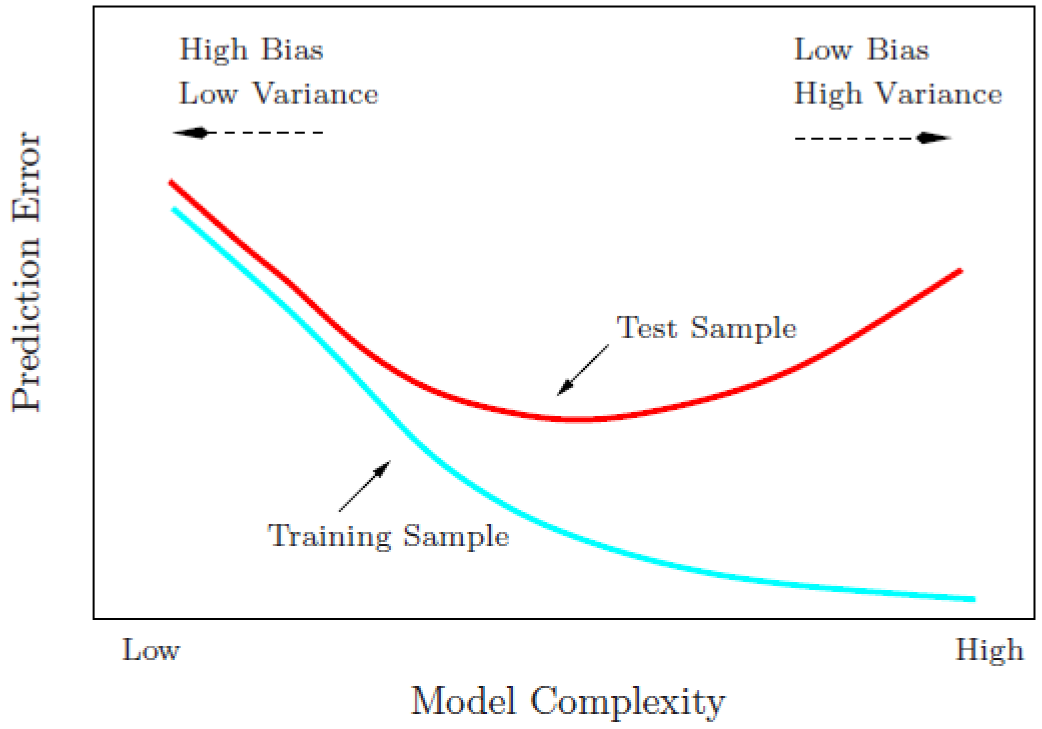
Problem with having too large K:

The larger the value of K, the higher is the accuracy. If K is too large, then our model is under-fitted. As a result, the error will go up again. So, to prevent your model from under-fitting it should retain the generalization capabilities otherwise there are fair chances that your model may perform well in the training data but

drastically fail in the real data. The computational expense of the algorithm also increases if we choose the k very large.
So, choosing k to a large value may lead to a model with a large bias(error).
The effects of k values on the bias and variance is explained below :

- As the value of k increases, the bias will be increases(underfitting)
- As the value of k decreases, the variance will increases(overfitting)
- With the increasing value of K, the boundary becomes smoother

So, there is a tradeoff between **overfitting and underfitting** and you have to maintain a balance while choosing the value of K in KNN. Therefore, **K should not be too small or too large**.



Which algorithm can be used for value imputation in both categorical and continuous categories of data?
KNN is the only algorithm that can be used for the imputation of both categorical and continuous variables. It can be used as one of many techniques when it comes to handling missing values.
To impute a new sample, we determine the samples in the training set “nearest” to the new sample and averages the nearby points to impute. A **Scikit learn library of Python** provides a quick and convenient way to use this technique.
Note: NaNs are omitted while distances are calculated**. Hence we replace the missing values with the average value of the neighbours.** The missing values will then be replaced by the average value of their “neighbours”.

Explain the statement- “The KNN algorithm does more computation on test time rather than train time”.

The above-given statement is **absolutely true**.

The basic idea behind the kNN algorithm is to determine a k-long list of samples that are close to a sample that we want to classify. Therefore, the training phase is basically storing a training set, whereas during the prediction stage the algorithm looks for k-neighbours using that stored data. Moreover, KNN does not learn anything from the training dataset as well.

What are the things which should be kept in our mind while choosing the value of k in the KNN Algorithm?

If K is small, then results might not be reliable because the noise will have a higher influence on the result. If K is large, then there will be a lot of processing to be done which may adversely impact the performance of the algorithm.

So, the following things must be considered while choosing the value of K:

- K should be the square root of n (number of data points in the training dataset).
- K should be chosen as the odd so that there are no ties. If the square root is even, then add or subtract 1 to it.

What are the advantages of the KNN Algorithm?

Some of the advantages of the KNN algorithm are as follows:

1. No Training Period: It does not learn anything during the training period since it does not find any discriminative function with the help of the training data. In simple words, actually, there is no training period for the KNN algorithm. It stores the training dataset and learns from it only when we use the algorithm for making the real-time predictions on the test dataset.

As a result, the KNN algorithm is much faster than other algorithms which require training. **For Example**, SupportVector Machines(SVMs), Linear Regression, etc.

Moreover, since the KNN algorithm does not require any training before making predictions as a result new data can be added seamlessly without impacting the accuracy of the algorithm.

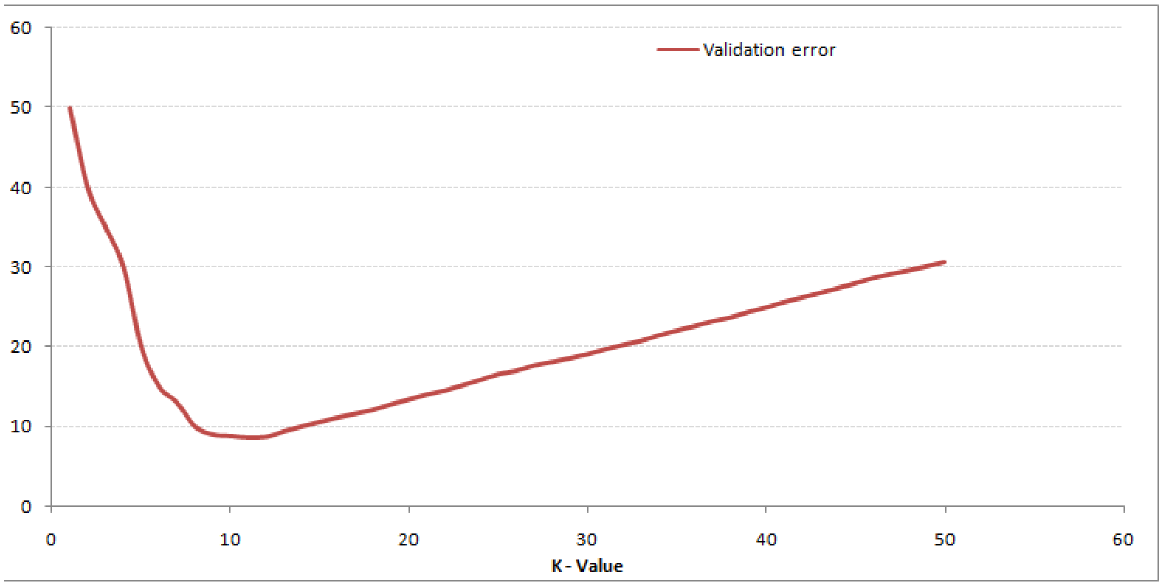
2. Easy to implement and understand: To implement the KNN algorithm, we need only two parameters i.e. the value of K and the distance metric(e.g. **Euclidean or Manhattan**, etc.). Since both the parameters are easily interpretable therefore they are easy to understand.

What are the disadvantages of the KNN Algorithm?

Some of the disadvantages of the KNN algorithm are as follows:

1. Does not work well with large datasets: In large datasets, the cost of calculating the distance between the new point and each existing point is huge which decreases the performance of the algorithm.

2. Does not work well with high dimensions: KNN algorithms generally do not work well with high dimensional data since, with the increasing number of dimensions, it becomes difficult to calculate the distance for each dimension.
3. Need feature scaling: We need to do feature scaling (standardization and normalization) on the dataset before feeding it to the KNN algorithm otherwise it may generate wrong predictions.
4. Sensitive to Noise and Outliers: KNN is highly sensitive to the noise present in the dataset and requires manual imputation of the missing values along with outliers removal.
Is it possible to use the KNN algorithm for Image processing?
Yes, KNN can be used for image processing by converting a 3-dimensional image into a single-dimensional vector and then using it as the input to the KNN algorithm.
What are the real-life applications of KNN Algorithms?
The various real-life applications of the KNN Algorithm includes:
1. KNN allows the calculation of the **credit rating**. By collecting the financial characteristics vs. comparing people having similar financial features to a database we can calculate the same. Moreover, the very nature of a credit rating where people who have similar financial details would be given similar credit ratings also plays an important role. Hence the existing database can then be used to predict a new customer's credit rating, without having to perform all the calculations.
2. In political science: KNN can also be used to predict whether a potential voter "will vote" or "will not vote", or to "vote Democrat" or "vote Republican" in an election.
1) [True or False] k-NN algorithm does more computation on test time rather than train time.
A) TRUE
B) FALSE
Solution: A The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.
In the testing phase, a test point is classified by assigning the label which are most frequent among the *k* training samples nearest to that query point – hence higher computation.
In the image below, which would be the best value for k assuming that the algorithm you are using is k-Nearest Neighbor.



A) 3
B) 10
C) 20
D 50

Solution: B Validation error is the least when the value of k is 10. So it is best to use this value of k

Which of the following distance metric can be used in k-NN?

A) Manhattan
B) Minkowski
C) Tanimoto
D) Jaccard
E) Mahalanobis
F) All can be used

Solution: F All of these distance metric can be used as a distance metric for k-NN.

4) Which of the following option is true about k-NN algorithm?

A) It can be used for classification
B) It can be used for regression
C) It can be used in both classification and regression

Solution: C We can also use k-NN for regression problems. In this case the prediction can be based on the mean or the median of the k-most similar instances.

Which of the following statement is true about k-NN algorithm?

1. k-NN performs much better if all of the data have the same scale
2. k-NN works well with a small number of input variables (p), but struggles when the number of inputs is very large

3. k-NN makes no assumptions about the functional form of the problem being solved

A) 1 and 2
B) 1 and 3
C) Only 1
D) All of the above

Solution: D The above mentioned statements are assumptions of kNN algorithm

Which of the following machine learning algorithm can be used for imputing missing values of both categorical and continuous variables?

A) K-NN
B) Linear Regression
C) Logistic Regression

Which of the following is true about Manhattan distance?

A) It can be used for continuous variables
B) It can be used for categorical variables
C) It can be used for categorical as well as continuous
D) None of these

Solution: A

Manhattan Distance is designed for calculating the distance between real valued features.

Which of the following distance measure do we use in case of categorical variables in k-NN?

1. Hamming Distance
2. Euclidean Distance
3. Manhattan Distance

A) 1
B) 2
C) 3
D) 1 and 2
E) 2 and 3
F) 1,2 and 3

Solution: A

Both Euclidean and Manhattan distances are used in case of continuous variables, whereas hamming distance is used in case of categorical variable.

Which of the following will be Euclidean Distance between the two data point A(1,3) and B(2,3)?

A) 1
B) 2

C) 4
D) 8
Solution: A
sqrt((1-2)^2 + (3-3)^2) = sqrt(1^2 + 0^2) = 1
Which of the following will be Manhattan Distance between the two data point A(1,3) and B(2,3)?
A) 1
B) 2
C) 4
D) 8
Solution: A
sqrt(mod((1-2)) + mod((3-3))) = sqrt(1 + 0) = 1
Which of the following will be true about k in k-NN in terms of Bias?
A) When you increase the k the bias will be increases
B) When you decrease the k the bias will be increases
C) Can’t say
D) None of these
Solution: A
large K means simple model, simple model always condider as high bias
Which of the following will be true about k in k-NN in terms of variance?
A) When you increase the k the variance will increases
B) When you decrease the k the variance will increases
C) Can’t say
D) None of these
Solution: B
Simple model will be consider as less variance model
When you find noise in data which of the following option would you consider in k-NN?
A) I will increase the value of k
B) I will decrease the value of k
C) Noise can not be dependent on value of k
D) None of these
Solution: A
In k-NN it is very likely to overfit due to the curse of dimensionality. Which of the following option would you consider to handle such problem?

1. Dimensionality Reduction
2. Feature selection

A) 1
B) 2

C) 1 and 2
D) None of these

Solution: C

In such case you can use either dimensionality reduction algorithm or the feature selection algorithm

Below are two statements given. Which of the following will be true both statements?

1. k-NN is a memory-based approach is that the classifier immediately adapts as we collect new training data.
2. The computational complexity for classifying new samples grows linearly with the number of samples in the training dataset in the worst-case scenario.

A) 1
B) 2
C) 1 and 2
D) None of these

Solution: C

Both are true and self explanatory

A company has build a kNN classifier that gets 100% accuracy on training data. When they deployed this model on client side it has been found that the model is not at all accurate. Which of the following thing might gone wrong?
Note: Model has successfully deployed and no technical issues are found at client side except the model performance

A) It is probably a overfitted model
B) It is probably a underfitted model
C) Can’t say
D) None of these

Solution: A

In an overfitted module, it seems to be performing well on training data, but it is not generalized enough to give the same results on a new data.

You have given the following 2 statements, find which of these option is/are true in case of k-NN?

1. In case of very large value of k, we may include points from other classes into the neighborhood.
2. In case of too small value of k the algorithm is very sensitive to noise

A) 1
B) 2

C) 1 and 2
D) None of these
Solution: C
Both the options are true and are self explanatory.
Which of the following statements is true for k-NN classifiers?
A) The classification accuracy is better with larger values of k
B) The decision boundary is smoother with smaller values of k
C) The decision boundary is linear
D) k-NN does not require an explicit training step
Solution: D
Option A: This is not always true. You have to ensure that the value of k is not too high or not too low.
Option B: This statement is not true. The decision boundary can be a bit jagged
Option C: Same as option B
Option D: This statement is true

26) True-False: It is possible to construct a 2-NN classifier by using the 1-
True-False: It is possible to construct a 2-NN classifier by using the 1-NN classifier?
A) TRUE
B) FALSE
Solution: A
You can implement a 2-NN classifier by ensembling 1-NN classifiers
In k-NN what will happen when you increase/decrease the value of k?
A) The boundary becomes smoother with increasing value of K
B) The boundary becomes smoother with decreasing value of K
C) Smoothness of boundary doesn't dependent on value of K
D) None of these
Solution: A
The decision boundary would become smoother by increasing the value of K
Following are the two statements given for k-NN algorthm, which of the statement(s)
is/are true?

1. We can choose optimal value of k with the help of cross validation
2. Euclidean distance treats each feature as equally important

A) 1
B) 2
C) 1 and 2
D) None of these

Solution: C
Both the statements are true
Suppose, you have trained a k-NN model and now you want to get the prediction on test data. Before getting the prediction suppose you want to calculate the time taken by k-NN for predicting the class for test data.
Note: Calculating the distance between 2 observation will take D time.

29) What would be the time taken by 1-NN if there are N(Very large) observations in test data?

A) N*D
B) N*D*2
C) (N*D)/2
D) None of these

Solution: A
The value of N is very large, so option A is correct

30) What would be the relation between the time taken by 1-NN,2-NN,3-NN.

A) 1-NN >2-NN >3-NN
B) 1-NN < 2-NN < 3-NN
C) 1-NN ~ 2-NN ~ 3-NN
D) None of these

Solution: C
The training time for any value of k in kNN algorithm is the same.

Difference between k-means and k-nearest neighbors?

k-Means is a *clustering algorithm* that tries to partition a set of points into k sets such that the points in each cluster tend to be near each other. It is *unsupervised* because the points have no external classification.

- **k-Nearest Neighbors** is a *classification* (or *regression*) algorithm that, in order to determine the classification of a point, combines the classification of the k nearest points. It is *supervised* because it is trying to classify a point based on the known classification of other points.

identify the false statement according to KNN disadavantage_________

- a) The cost of predicting the k nearest neighbours is very high.
- b) Doesn't work as expected when working with big number of features/parameters.
- c) Hard to work with categorical features.
- d) Feature engineering is not possible.

Answer - a) The cost of predicting the k nearest neighbours is very high

1. Choosing _______values for k can be noisy and will have a higher influence on the result.
    - a) Smaller k value.
    - b) Standard k value.
    - c) Larger k value.
    - d) All of the above.

   Answer - a) Smaller k value

1. True/False. Is cross validation another way to choose k?
    - a) True.
    - b) False.

   Answer - a) True

1. Higher dimension problem in KNN Algorithm _________
    - a) Nearest data points.
    - b) Curse of dimensionality.
    - c) Variable curse.

   Answer - a) Nearest data points

1. In KNN, increase in dimension also leads to the problem of ________.
    - a) Underfitting.
    - b) Overfitting.

   Answer - a) Underfitting

1. __________is the KNN function used for KNN algorithm in python programming
    - a) KNN.
    - b) K neighborsclassifier.

   Answer - b) K neighborsclassifier

1. How many metrics are available in KNN algorithm?
    - a) 3.
    - b) 2.
    - c) 4.
    - d) 1.

   Answer - a) 3

What mathematical concept Naive Bayes is based on?

Naive Bayes is based on Bayes theorem in statistics. It calculates probabilities independently for each class based on conditions and without conditions and then predicts outcomes based on that.

What are the different types of Naive Bayes classifiers?

Multinomial Naive Bayes
Bernoulli Naive Bayes
Gaussian Naive Bayes

Is Naive Bias a classification algorithm or regression algorithm?
It is a classification algorithm. Naive Bayes is a supervised learning algorithm but it can also be trained as semi-supervised learning algorithm.
What are some benefits of Naive Bayes?
It works better than simple algorithms like logistic regression etc. It also works well with categorical data and with numerical data as well. Additionally, It is very easy and fast to work with the Naive Bayes classifier. Complex and high dimensional data is well suited for Naive Bayes classifier. It can also be trained using a small labeled dataset with semi-supervised learning.
In other words:
• It performs well with both clean and noisy data.
• Training takes a few samples, but the fundamental assumption is that the training dataset is a genuine representation of the population.
• Obtaining the likelihood of a forecast is simple.

What are the cons of Naive Bayes classifier?
Naive Bayes classifiers suffer from "Zero Frequency" problem. This happens when a category is not present in the training set. It will give it 0 probability.
Its biggest downside is the consideration of features as independent of each other because in real life it is impossible to get independent features. All features are somehow co-related with each other.

What are the applications of Naive Bayes?
Naive Bayes classifier is a very powerful technique. It is applied in various classification techniques which are used for real-time prediction. The algorithm is also widely used in NLP tasks like sentiment analysis of text sentences, applying spam filtering, text classification etc. It is also used to make recommendation systems and for collaborative filtering.
Is Naive Bayes is a discriminative classifier or generative classifier?
Naive Bayes is a generative classifier. It learns from the actual distribution of the dataset by performing operations on it. It does not create a decision boundary to classify data.

What is the formula given by Bayes theorem?

$$P(A|B) = \frac{P(B|A) \,.\, P(A)}{P(B)}$$

What is posterior probability and prior probability in Naïve Bayes?

Probability of event A given event B is true, P(A|B), is called as posterior probability and independent probability of event A, P(A), is called as prior probability i.e.
P(A|B) = posterior probability
P(A) = prior probability

Define likelihood and evidence in Naive Bayes?

The probability of event B given that event A is true is called likelihood and the independent probability of event B is called evidence.
P(B|A) = likelihood
P(B) = evidence

Define Bayes theorem in terms of prior, evidence and likelihood.

Probability of event A given that event B is true (Posterior) can be found using the expression:
Posterior = Likelihood * Prior / Evidence

While calculating the probability of a given situation, what error can we run into in Naïve Bayes and how can we solve it?

We might encounter the zero division error when the probability for a particular scenario in the numerator is zero. To mitigate this, we can use **Laplace Smoothing** which basically adds a number to the numerator and another number to the denominator.

What is the Bernoulli distribution in Naïve Bayes?

This is a distribution that evaluates a particular outcome as binary. For example, in the Bernoulli Naïve Bayes classifier, given a word, that word can either be in a message or not.

How does Naïve Bayes treat numerical and categorical values?

- For the categorical features, we can estimate our probability using a distribution such as multinomial or Bernoulli.
- For the numerical features, we can estimate our probability using a distribution such as Normal or Gaussian.

What is the best dataset scenario for the Naïve Bayes Classifier?
If the training data is smaller or if the dataset has fewer number of observations (samples) and a high number of features. Naïve Bayes works well on this data because of its High bias – Low variance trade off.

What's the difference between Generative Classifiers and Discriminative Classifiers?

A **Generative Model** ,develops classifier based on underlying distribution.It does not create decision boundary.
An example of a generative model would be Naive Bayes.
Where as,a **Discriminative Model** models the decision boundary between the classes An example of a Discriminitive model would be Logistic Regression.

What is a *Naïve Bayes Classifier*?

- ***Naive Bayes Classifiers*** are a family of simple *probabilistic classifiers* based on applying *Bayes' theorem* with strong independence assumptions between the features.

What is *Statistical Significance*?
Statistical significance is a term used by researchers to state that it is unlikely their observations could have occurred under the null hypothesis of a statistical test. Significance is usually denoted by a *p*-value, or probability value.
Statistical significance is arbitrary – it depends on the threshold, or alpha value, chosen by the researcher. The most common threshold is $p < 0.05$, which means that the data is likely to occur less than 5% of the time under the null hypothesis.
When the *p*-value falls below the chosen alpha value, then we say the result of the test is statistically significant.

Why Naive Bayes is called *Naive*?
We call it **naive** because its assumptions (it assumes that all of the features in the dataset are equally important and independent) are really optimistic and rarely true in most real-world applications:

- we consider that these *predictors* are *independent*
- we consider that all the predictors have an *equal effect* on the outcome (like the day being windy does not have more importance in deciding to play golf or not)

Can you choose a *classifier* based on the *size of the training set*?

- If the *availability of data is a constraint*, i.e. if the training data is smaller or if the dataset has a fewer number of observations and a higher number of features, we can choose algorithms with *high bias/low variance* like ***Naïve Bayes*** and ***Linear SVM***.

- If the training data is *sufficiently large* and the number of observations is higher as compared to the number of features, one can go for *low bias/high variance* algorithms like ***K-Nearest Neighbors***, ***Decision trees***, ***Random forests***, and ***kernel SVM***.

What are some *disadvantages* of using *Naive Bayes Algorithm*?

Mid

Naïve Bayes 18

Answer
Some *disadvantages* of using ***Naive Bayes Algorithm*** are:

- **It relies on a very big assumption that the independent variables are not related to each other**.
- It is generally *not suitable for datasets with large numbers of numerical attributes*.
- It has been observed that if particular category is not present in training dataset but is in the *testing* dataset, then it will most **definitely be wrong**.

What are some advantages of using *Naive Bayes Algorithm*?

- It is a simple, fast, and very robust method of classification.
- It does well with both clean and noisy data.
- It requires a few examples for training, but the underlying assumption is that the training dataset is a true representative of the population.

What's the difference between *Generative Classifiers* and *Discriminative Classifiers*? Name some examples of each one
 A **Generative Model** explicitly models the *actual distribution* of each class. It learns the joint probability distribution $p(x,y) = P(y) * P(x|y)$, where both $P(y)$ and $P(x|y)$ can be estimated from the dataset by computing class frequencies. Some examples of generative models are:

- Naïve Bayes

- Bayesian networks
- Markov random fields

 A **Discriminative Model** models the *decision boundary* between the classes by learning the conditional probability distribution P(y|x) from the dataset. Some examples of such kinds of classifiers are:

- Logistic regression
- Scalar Vector Machine
- Traditional neural networks
- KNN.

Are there any problems using *Naive Bayes* for Classification?

Compare *Naive Bayes* vs with *Logistic Regression* to solve classification problems

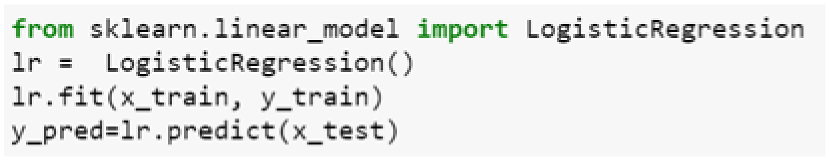
What are the trade-offs between the different types of *Classification Algorithms*? How would do you choose the best one?

2.1 Logistic Regression

Definition: Logistic regression is a machine learning algorithm for classification. In this algorithm, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function.

Advantages: Logistic regression is designed for this purpose (classification), and is most useful for understanding the influence of several independent variables on a single outcome variable.

Disadvantages: Works only when the predicted variable is binary, assumes all predictors are independent of each other and assumes data is free of missing values.



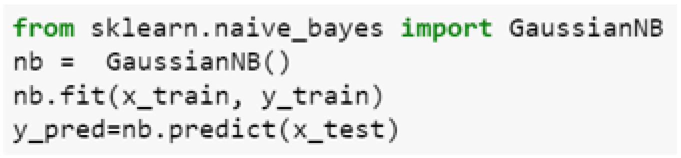
```
from sklearn.linear_model import LogisticRegression
lr =  LogisticRegression()
lr.fit(x_train, y_train)
y_pred=lr.predict(x_test)
```

2.2 Naïve Bayes

Definition: Naive Bayes is based on Bayes theorem in statistics. It calculates probabilities independently for each class based on conditions and without conditions and then predicts outcomes based on that.

Advantages: This algorithm requires a small amount of training data to estimate the necessary parameters. Naive Bayes classifiers are extremely fast compared to more sophisticated methods.

Disadvantages: Naive Bayes is is known to be a bad estimator.



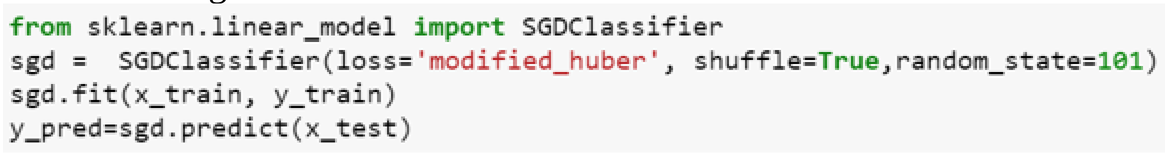
```
from sklearn.naive_bayes import GaussianNB
nb =  GaussianNB()
nb.fit(x_train, y_train)
y_pred=nb.predict(x_test)
```

2.3 Stochastic Gradient Descent

Definition: To minimize the error of predictions, Stochastic techniques is used. It is a simple and very efficient approach to fit linear models. It is particularly useful when the number of samples are very large. It supports different loss functions and penalties for classification.

Advantages: Efficiency and ease of implementation.

Disadvantages: Requires a number of hyper-parameters and it is sensitive to feature scaling.



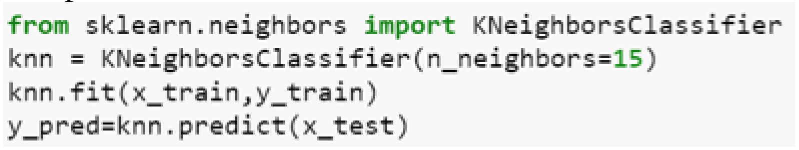
```
from sklearn.linear_model import SGDClassifier
sgd =  SGDClassifier(loss='modified_huber', shuffle=True,random_state=101)
sgd.fit(x_train, y_train)
y_pred=sgd.predict(x_test)
```

2.4 K-Nearest Neighbours

Definition: Neighbours based classification is a type of lazy learning as it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the k nearest neighbours of each point.

Advantages: This algorithm is simple to implement, robust to noisy training data, and effective if training data is large.

Disadvantages: Need to determine the value of K and the computation cost is high as it needs to compute the distance of each instance to all the training samples.



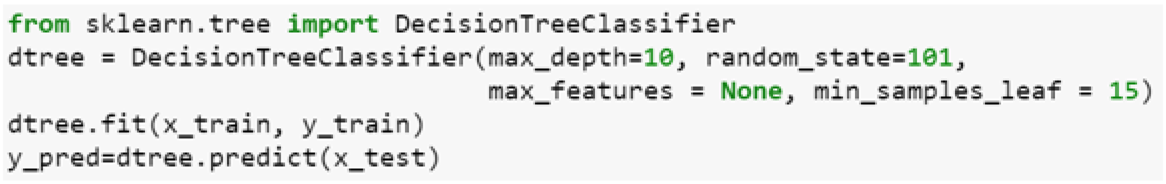
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=15)
knn.fit(x_train,y_train)
y_pred=knn.predict(x_test)
```

2.5 Decision Tree

Definition: Given a data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to classify the data.

Advantages: Decision Tree is simple to understand and visualise, requires little data preparation, and can handle both numerical and categorical data.

Disadvantages: Decision tree can create complex trees that do not generalise well, and decision trees can be unstable because small variations in the data might result in a completely different tree being generated.



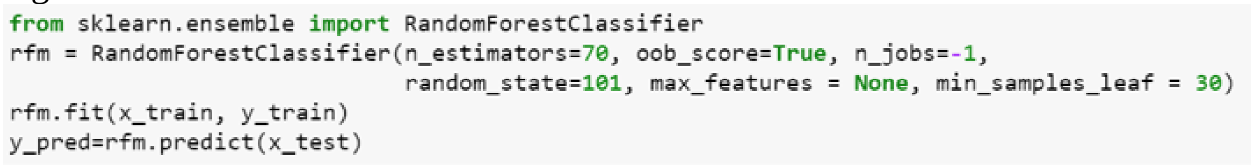
```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(max_depth=10, random_state=101,
                               max_features = None, min_samples_leaf = 15)
dtree.fit(x_train, y_train)
y_pred=dtree.predict(x_test)
```

2.6 Random Forest

Definition: Random forest classifier is a meta-estimator that fits a number of decision trees on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and controls over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

Advantages: Reduction in over-fitting and random forest classifier is more accurate than decision trees in most cases.

Disadvantages: Slow real time prediction, difficult to implement, and complex algorithm.



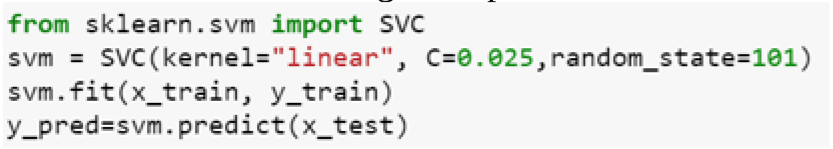
```
from sklearn.ensemble import RandomForestClassifier
rfm = RandomForestClassifier(n_estimators=70, oob_score=True, n_jobs=-1,
                             random_state=101, max_features = None, min_samples_leaf = 30)
rfm.fit(x_train, y_train)
y_pred=rfm.predict(x_test)
```

2.7 Support Vector Machine

Definition: Support vector machine is a representation of the training data as points in space separated into categories by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

Advantages: Effective in high dimensional spaces and uses a subset of training points in the decision function so it is also memory efficient.

Disadvantages: The algorithm does not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.



```
from sklearn.svm import SVC
svm = SVC(kernel="linear", C=0.025,random_state=101)
svm.fit(x_train, y_train)
y_pred=svm.predict(x_test)
```

Here are some important considerations while choosing an algorithm

1. **Size of the training data**

It is usually recommended to gather a good amount of data to get reliable predictions. However, many a time the availability of data is a constraint. So, if the training data is smaller or if the dataset has a fewer number of observations and a higher number of features like genetics or textual data, choose algorithms with high bias/low variance like **Linear regression, Naïve Bayes, Linear SVM**. If the training data is sufficiently large and the number of observations is higher as compared to the number of features, one can go for low bias/high variance algorithms like **KNN, Decision trees, kernel SVM.**

1. **Accuracy and/or Interpretability of the output**

Accuracy of a model means that the function predicts a response value for a given observation which is close to the true response value for that observation. Highly interpretable algorithm (restrictive models like Linear Regression) means that one can easily understand how any individual predictor is associated with the response while the flexible models give higher accuracy at the cost of low interpretability.

A representation of trade-off between Accuracy and Interpretability, using different statistical learning methods. (source)

Some algorithms are called Restrictive because they produce small range of shapes of the mapping function. For example, linear regression is a restrictive approach because it can only generate linear functions such as the lines. Some algorithms are called flexible because they can generate a wider range of possible shapes of the mapping function. For example, KNN with k=1 is highly flexible as it will consider every input data point to generate the mapping output function. The below picture displays the tradeoff between flexible and restrictive algorithms.

A representation of trade-off between Flexibility and Interpretability, using different statistical learning methods. (source)

Now, to use which algorithm depends on the objective of the business problem. **If inference is the goal, then restrictive models are better as they are much more interpretable. Flexible models are better if higher accuracy is the goal. In general, as the flexibility of a method** increases, its interpretability decreases.

1. **Speed or Training time**

Higher accuracy typically means higher training time. Also, algorithms require more time to train on large training data. In real-world applications, the choice of algorithm is driven by these two factors predominantly. Algorithms like **Naïve Bayes, Linear and Logistic regression** are easy to implement and quick to run. Algorithms like SVM which involve tuning of parameters, Neural networks with high convergence time, random forests need a lot of time to train the data.

1. **Linearity**

Many algorithms work on the assumption that classes can be separated by a straight line (or its higher-dimensional analog). Examples include logistic regression and support vector machines. Linear regression algorithms assume that data trends follow a straight line. If the data is linear then these algorithms perform quite good. However, not always the data is linear, so we require other algorithms which can handle high dimensional and complex data structures. Examples include kernel SVM, random forest, neural nets. Best way to find out the linearity is to either fit a linear line or run a logistic regression or SVM and check for residual errors. Higher error means the data is not linear and would need complex algorithms to fit.

1. **Number of features**

The dataset may have a large number of features which may not all be relevant and significant. For certain type of data such as genetics or textual, the number of features can be very large compared to the number of data points. A large number of features can bog down some learning algorithms, making training time unfeasibly long. SVM is better suited in case of data with large feature space and lesser observations. PCA and feature selection techniques should be used to reduce dimensionality and select important features.   Here is a handy

Whether Feature Scaling is required in Naïve bayes?
No
Impact of Missing Values?
Naive Bayes can handle missing data. Attributes are handled separately by the algorithm at both model construction time and prediction time. As such, if a data instance has a missing value for an attribute, it can be ignored while preparing the model, and ignored when a probability is calculated for a class value tutorial.
Impact of outliers?
It is usually robust to outliers
Different Problem statement you can solve using Naive Baye's

1. Sentiment Analysis
2. Spam classification
3. twitter sentiment analysis
4. document categorization

There are three naïve Bayes classifiers:

1. The ***Multinomial*** classifier uses multinomial distribution on each word of a sentence. Every word is treated independently rather than being treated as a part of the sentence.
2. The ***Gaussian*** classifier is utilized with continuous data. It assumes that each data class is distributed as a Gaussian distribution.
3. The ***Bernoulli*** classifier assumes that every feature present is binary, which means it can only take either of the two values.

Explain The Strength Of Bayesian Statistics
It is sometimes preferred over other methods, here's why:

1. Bayesian gives intuitive and direct inferences. It meaningfully tells the probability of a hypothesis being true.

2. Introduction to Bayesian statistics enhanced the power of answering complicated questions easily and clearly.
3. Bayesian uses every available information to find the probability. This indicates that apart from data, the method uses prior information as well.
4. The method enhances decision-making. When there is a lack of parameters and facts, Bayesian quantify uncertainties using available evidence.

Explain The Difference Between Maximum Likelihood Estimation (MLE) And Bayesian Statistics

Suppose, you tossed a coin 10 times, the result was 7 heads and 3 tails. This means that the probability of getting a “head” is 70% and “tail” is 30% in the 11th toss. **This is one of the many possible arrangements and related estimation. The method is called maximum likelihood estimation.**

Now, since we know that the possibility of heads and tails is 50%, we can consider this prior information. Then, calculate what will be the outcome in the 11th toss. This method is Bayesian statistics.

Explain The Difference Between Bayes Theorem And Conditional Probability

Conditional Probability:

Conditional probability is defined as the probability that an event A occurs, given that an event B already occurred. This is formally written as P(A|B)P(A|B), which reads as: the probability of A given B. Conditional probability P(A|B)P(A|B) can be calculated with the following formula:

P(A|B)=P(A∩B)/P(B)(1)(1)P(A|B)=P(A∩B)/P(B)

where P(A∩B)P(A∩B) is the probability that both event A and event B occur, and P(B)P(B) is just the probability that event B occurs.

Now, if we want to calculate the probability that event B occurs, given that event A has already occurred, we can use the same formula, only this time changing out the denominator as follows:

P(B|A)=P(A∩B)P(A)(2)(2)P(B|A)=P(A∩B)P(A)

where it is important to note that P(A∩B)=P(B∩A)P(A∩B)=P(B∩A), as both represent the probability that events A and B occur.

Answer and Explanation:

Bayes' theorem is simply derived from the definition of conditional probability (as described above), and is formally written as:

P(A|B)=P(B|A)∗P(A)P(B)(3)(3)P(A|B)=P(B|A)∗P(A)P(B)

Note that the only difference between the equation that defines conditional probability (equation (1) above) and our definition of Bayes' theorem (equation

(3)), is the numerator on the right-hand side of the equation. In conditional probability, we see $P(A\cap B)P(A\cap B)$ in the numerator, whereas in Bayes' theorem, we see $P(B|A)\ast P(A)P(B|A)\ast P(A)$ in the numerator.
To see where this equation comes from, we simply need to rearrange equation (2). Multiplying both sides of equation (2) by $P(A)P(A)$ gives us:
$P(B|A)\ast P(A)=P(A\cap B)P(B|A)\ast P(A)=P(A\cap B)$
We can also multiply both sides of equation (1) above by $P(B)P(B)$ to show that
$P(A|B)\ast P(B)=P(A\cap B)P(A|B)\ast P(B)=P(A\cap B)$
Thus, **Bayes' theorem is simply an extension of the definition of conditional probability, making use of the fact that**$P(A\cap B)=P(B|A)\ast P(A)=P(A|B)\ast P(B)$

What is the difference between Causation and Correlation? Explain with example.
Answer: Causation is the relationship between two variables such that occurrences of the other cause one of them.
Correlation is the relationship between two variables that are related to each other but not caused by each other.
For example, Inflation causes the price fluctuations in petrol and groceries, so inflation has a causation relationship with both of them. Between gasoline and groceries, there is a correlation that both of them can increase or decrease due to changes in inflation, but neither of them causes or impacts the other one.

Answer: *Conditional probability is the measure of the likelihood of one event, given that one event has occurred. Let us consider two events are given A and B, then the conditional probability of A, given B has already occurred, is provided as:*

Bayes theorem

Answer: *The theorem is used to describe the probability of an event based on the prior knowledge of other events related to it. For example, the probability of a person having a particular disease would be found on the symptoms shown.*

Question: What is stratified sampling?
Answer: Stratified sampling is a probability sampling technique wherein the entire population is divided into different subgroups called strata than a probability sample is drawn proportionally from each layer. For instance, in the case of binary classification, if the ratio of positive and negative labeled data were 9:1, then in stratified sampling, you would randomly select subsample from each of the positive and negative labeled datasets such that after sampling the ratio remains 9:1

Define F-test. Where would you use it?
Answer: The F Test Formula is a Statistical Formula used to test the significance of differences between two groups of Data. It is often used in research studies to determine whether the difference in the means of two populations is Statistically significant.Question: What is a chi-squared test?
Answer: Chi-squared is any statistical hypothesis test where the test statistic follows a chi-squared distribution (distribution of the sum of the squared standard normal deviates) under the null hypothesis. It measures how well the observed distribution of data fits the expected distribution if the variables are independent.

Explain how a ROC curve works.
Answer: ROC curve or Receiver Operating Characteristic Curve is the graphical representation of the performance of a classification model for all the classification thresholds. The graph shows two parameters, i.e. True Positive Rate (TPR) and False Positive Rate (FPR) at different classification thresholds. A typical ROC curveis as follows

Define precision and recall.
Answer: Precision and recall are measures used to evaluate the performance of a classification algorithm. In a perfect classifier, precision and recall are equal to 1. Precision is the fraction of relevant instances amongst the retrieved instances, whereas recall is the fraction of retrieved instances amongst the relevant instances.
Precision = true positive/(true positive + false positive)
Recall = true positive/(true positive + false negative)

Question: What is the difference between L1 and L2 regularization?
Answer: Both L1 and L2 regularization are done to avoid overfitting. L1 tries to calculate the median, whereas L2 calculates the mean of the data for the same. L1 is also called Lasso and L2, Ridge regularization technique.
In L1 regularization, features that are not important are eliminated, thus selecting only the most relevant features. In L2, the loss function tries to minimize loss by subtracting it from the average (mean) of the distribution of data.

What is the Decision Tree Algorithm?
A Decision Tree is a supervised machine learning algorithm that can be used for both Regression and Classification problem statements. It divides the complete dataset into smaller subsets while at the same time an associated Decision Tree is incrementally developed.
The final output of the Decision Trees is a Tree having Decision nodes and leaf nodes. A Decision Tree can operate on both categorical and numerical data.
List down some popular algorithms used for deriving Decision Trees along with their attribute selection measures.

Some of the popular algorithms used for constructing decision trees are:
1. ID3 (Iterative Dichotomiser): Uses Information Gain as attribute selection measure.
2. C4.5 (Successor of ID3): Uses Gain Ratio as attribute selection measure.
3. CART (Classification and Regression Trees) – Uses Gini Index as attribute selection measure.

Explain the CART Algorithm for Decision Trees.

The CART stands for **Classification and Regression Trees** is a greedy algorithm that greedily searches for an optimum split at the top level, then repeats the same process at each of the subsequent levels.

Moreover, it does verify whether the split will lead to the lowest impurity or not as well as the solution provided by the greedy algorithm is not guaranteed to be optimal, it often produces a solution that's reasonably good since finding the optimal Tree is an **NP-Complete problem** that requires **exponential time complexity**.

List down the attribute selection measures used by the ID3 algorithm to construct a Decision Tree.

The most widely used algorithm for building a Decision Tree is called ID3. ID3 uses Entropy and Information Gain as attribute selection measures to construct a Decision Tree.

1. Entropy: A Decision Tree is built top-down from a root node and involves the partitioning of data into homogeneous subsets. To check the homogeneity of a sample, ID3 uses entropy. Therefore, entropy is zero when the sample is completely homogeneous, and entropy of one when the sample is equally divided between different classes.

Information Gain: Information Gain is based on the decrease in entropy after splitting a dataset based on an attribute. The meaning of constructing a Decision Tree is all about finding the attributes having the highest information gain.

Do we require Feature Scaling for Decision Trees? Explain.

Decision Trees are mainly intuitive, easy to interpret as well as require less data preparation. In fact, they don't require feature scaling or centering(standardization) at all. Such models are often called **white-box models**.

What are the disadvantages of Information Gain?

Information gain is defined as the reduction in entropy due to the selection of a particular attribute. Information gain biases the Decision Tree against considering attributes with a large number of distinct values which might lead to overfitting.

In order to solve this problem, the **Information Gain Ratio** is used.

List down the problem domains in which Decision Trees are most suitable.

Decision Trees are suitable for the following cases:
1. Decision Trees are most suitable for **tabular data**.

2. The outputs are **discrete**.
3. Explanations for Decisions are required.
4. The training data may contain errors, noisy data(outliers).
5. The training data may contain **missing feature** values.

If it takes one hour to train a Decision Tree on a training set containing 1 million instances, roughly how much time will it take to train another Decision Tree on a training set containing 10 million instances?

As we know that the computational complexity of training a Decision Tree is given by **O(n × m log(m))**. So, when we multiplied the size of the training set by 10, then the training time will be multiplied by some factor, say K.

Now, we have to determine the value of K. To finds K, divide the complexity of both:

K = (n × 10m × log(10m)) / (n × m × log(m)) = 10 × log(10m) / log(m)

For 10 million instances i.e., m = 10^6, then we get the value of K ≈ 11.7.

Therefore, we can expect the training time to be roughly 11.7 hours.

How does a Decision Tree handle missing attribute values?

Decision Trees handle missing values in the following ways:

- Fill the missing attribute value by the most common value of that attribute.
- Fill the missing value by assigning a probability to each of the possible values of the attribute based on other samples.

How does a Decision Tree handle continuous(numerical) features?

Decision Trees handle continuous features by converting these continuous features to a **threshold-based boolean** feature.

To decide The threshold value, we use the concept of Information Gain, choosing that threshold that maximizes the information gain.

What is the Inductive Bias of Decision Trees?

The ID3 algorithm preferred Shorter Trees over longer Trees. In Decision Trees, attributes having high information gain are placed close to the root are preferred over those that do not.

Explain Feature Selection using the Information Gain/Entropy Technique.

The goal of the feature selection while building a Decision Tree is to select features or attributes (Decision nodes) which lead to a split in children nodes whose combined entropy adds up to lower entropy than the entropy value of the data segment before the split. This implies higher information gain.

Compare the different attribute selection measures.

The three measures, in general, returns good results, but:

1. Information Gain: It is biased towards multivalued attributes

2. Gain ratio: It prefers unbalanced splits in which one data segment is much smaller than the other segment.

3. Gini Index: It is biased to multivalued attributes, has difficulty when the number of classes is large, tends to favor tests that result in equal-sized partitions and purity in both partitions.

Why do we require Pruning in Decision Trees? Explain.

After we create a Decision Tree we observe that most of the time the leaf nodes have very high homogeneity i.e., properly classified data. However, this also leads to overfitting. Moreover, if enough partitioning is not carried out then it would lead to underfitting.

Hence the major challenge that arises is to find the optimal trees which result in the appropriate classification having acceptable accuracy. So to cater to those problems we first make the decision tree and then use the error rates to appropriately prune the trees.

Are Decision Trees affected by the outliers? Explain.

Decision Trees are not sensitive to noisy data or outliers since, extreme values or outliers, never cause much reduction in **Residual Sum of Squares(RSS),** because they are never involved in the split.

What do you understand by Pruning in a Decision Tree?

When we remove sub-nodes of a Decision node, this process is called pruning or the opposite process of splitting. The two techniques which are widely used for pruning are- Post and Pre Pruning.

Post Pruning:

- This type of pruning is used after the construction of the Decision Tree.
- This technique is used when the Decision Tree will have a very large depth and will show the overfitting of the model.
- It is also known as backward pruning.
- This technique is used when we have an infinitely grown Decision Tree.

Pre Pruning:

- This technique is used before the construction of the Decision Tree.
- Pre-Pruning can be done using Hyperparameter tuning.
- Overcome the overfitting issue.

List down the advantages of the Decision Trees.

1. Clear Visualization: This algorithm is simple to understand, interpret and visualize as the idea is mostly used in our daily lives. The output of a Decision Tree can be easily interpreted by humans.

2. Simple and easy to understand: Decision Tree works in the same manner as simple if-else statements which are very easy to understand.

3. This can be used for both classification and regression problems.

4. Decision Trees can handle both continuous and categorical variables.

5. No feature scaling required: There is no requirement of feature scaling techniques such as standardization and normalization in the case of Decision Tree as it uses a rule-based approach instead of calculation of distances.
6. Handles nonlinear parameters efficiently: Unlike curve-based algorithms, the performance of decision trees can't be affected by the Non-linear parameters. So, if there is high non-linearity present between the independent variables, Decision Trees may outperform as compared to other curve-based algorithms.
7. Decision Tree can automatically handle missing values.
8. Decision Tree handles the outliers automatically, hence they are usually robust to outliers.
9. Less Training Period: The training period of decision trees is less as compared to ensemble techniques like Random Forest because it generates only one Tree unlike the forest of trees in the Random Forest.
List out the disadvantages of the Decision Trees.
1. Overfitting: This is the major problem associated with the Decision Trees. It generally leads to overfitting of the data which ultimately leads to wrong predictions for testing data points. it keeps generating new nodes in order to fit the data including even noisy data and ultimately the Tree becomes too complex to interpret. In this way, it loses its generalization capabilities. Therefore, it performs well on the training dataset but starts making a lot of mistakes on the test dataset.
2. High variance: As mentioned, a Decision Tree generally leads to the overfitting of data. Due to the overfitting, there is more likely a chance of high variance in the output which leads to many errors in the final predictions and shows high inaccuracy in the results. So, in order to achieve zero bias (overfitting), it leads to high variance due to the bias-variance tradeoff.
3. Unstable: When we add new data points it can lead to regeneration of the overall Tree. Therefore, all nodes need to be recalculated and reconstructed.
Not suitable for large datasets: If the data size is large, then one single Tree may grow complex and lead to overfitting. So in this case, we should use Random Forest instead, an ensemble technique of a single Decision Tree.

- ***Random forest*** is an ***ensemble learning*** method that works by constructing a multitude of ***decision trees***. A random forest can be constructed for both classification and regression tasks.
- Random forest **outperforms** decision trees, and it also does not have the habit of *overfitting* the data as decision trees do.
- A decision tree trained on a specific dataset will become very deep and cause overfitting. To create a random forest, decision trees can be trained on different subsets of the training dataset, and then the different decision trees can be averaged with the goal of *decreasing the variance*.

The gini index, or gini coefficient, or gini impurity computes the degree of probability of a specific variable that is wrongly being classified when chosen randomly and a variation of gini coefficient. It works on categorical variables, provides outcomes either be “successful” or “failure” and hence conducts binary splitting only.

The degree of gini index varies from 0 to 1,

- Where 0 depicts that all the elements be allied to a certain class, or only one class exists there.
- The gini index of value as 1 signifies that all the elements are randomly distributed across various classes, and
- A value of 0.5 denotes the elements are uniformly distributed into some classes.

- **Ensemble learning** is a *machine learning* method that uses *multiple learning algorithms* to obtain a better predictive performance than could be obtained from any of the constituent learning algorithms alone.
- *Ensembles* tend to yield better results when there is significant diversity among the models. The ensemble methods, therefore, seek to promote diversity among the models they combine.
  In ensemble learning theory, we call **weak learners** (or **base models**) models that can be used as building blocks for designing more complex models by combining several of them. Most of the time, these basics models perform not so well by themselves either because they have a high bias (low degree of freedom models, for example) or because they have too much variance to be robust (high degree of freedom models, for example).

- ***Random forest*** is an ***ensemble learning*** method that works by constructing a multitude of ***decision trees***. A random forest can be constructed for both classification and regression tasks.
- Random forest **outperforms** decision trees, and it also does not have the habit of *overfitting* the data as decision trees do.
- A decision tree trained on a specific dataset will become very deep and cause overfitting. To create a random forest, decision trees can be trained on different subsets of the training dataset, and then the different decision trees can be averaged with the goal of *decreasing the variance*.

The idea of **stacking** is to learn several different weak learners and combine them by training a *meta-model* to output predictions based on the multiple predictions returned by these weak models.
If a stacking ensemble is composed of L weak learners, then to fit the model the following steps are followed:

- Split the training data into two folds.
- Choose L weak learners and fit them to the data of the first fold.
- For each of the L weak learners, make predictions for observations in the second fold.
- Fit the meta-model on the second fold, using predictions made by the weak learners as inputs.

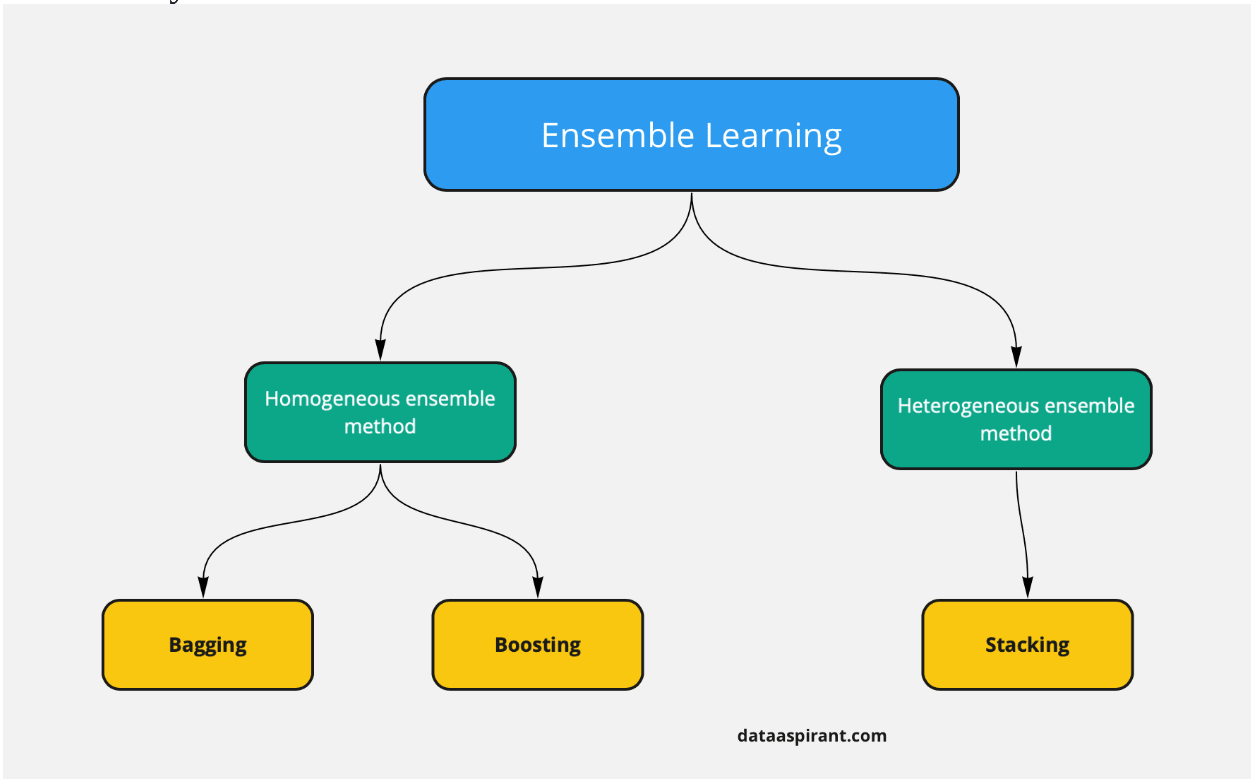
The process of *stacking* is shown in the figure below:
Since *Ensemble Learning* provides better output most of the time, why do you not use it all the time?

- Although it provides a better outcome many times, it is not true that it will always perform better.
- There are several ensemble methods, each with its own *advantages/disadvantages*, and choosing one to use depends on the problem at hand.
- If there are models with **high variance**, then it will benefit from **bagging**. If the model is **biased**, it is better to use **boosting**.
- If the work is in **probabilistic setting**, the ensemble methods may not work because it is known that **boosting** delivers poor probability estimates.

- **Homogeneous** ensemble consists of members having a single-type base learning algorithm. Popular methods like *bagging* and *boosting* generate diversity by sampling from or assigning weights to training examples but generally utilize a single type of base classifier to build the ensemble.
- **Heterogeneous** ensemble consists of members having different base learning algorithms such as *SVM*, *ANN*, and *Decision Trees*. A popular heterogeneous ensemble method is stacking, which is similar to boosting.

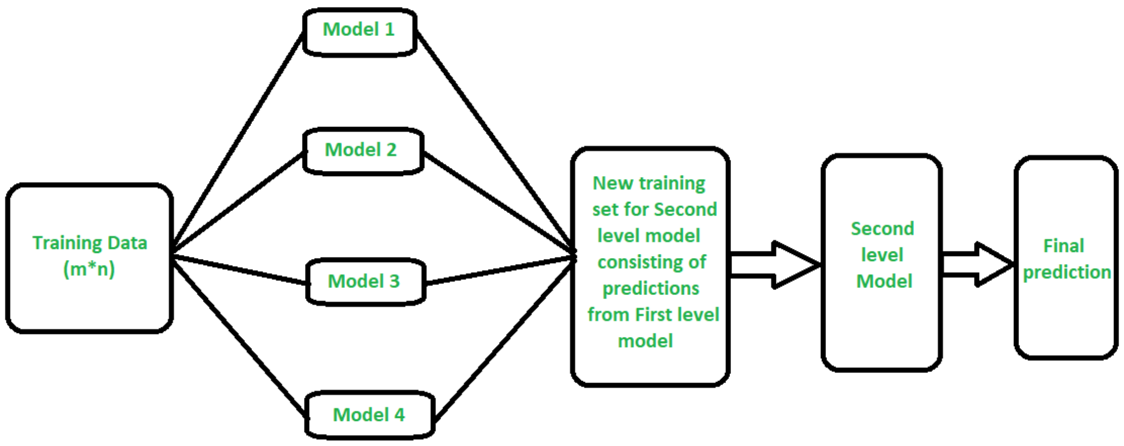
- **Homogeneous** ensemble work by applying the same algorithm on all the estimators. These algorithms should not be fine-tuned. In contrast to *heterogeneous ensembles*, a large number of estimators are used.

- **Heterogeneous** ensembles use different fine-tuned algorithms. They work well with *small* amount of estimators. The number of algorithms should always be odd to avoid ties.



super learner algorithm

- When a problem has already been fit with many different algorithms on the dataset, and some algorithms have been evaluated many times with different configurations, using all the models instead of just the best model from the group is the intuition behind the **super learner** ensemble algorithm.
- The **super learner algorithm** involves first pre-defining the k-fold split of your data, then evaluating all different algorithms and algorithm configurations on the same split of the data. All out-of-fold predictions are then kept and used to train an algorithm that learns how to best combine the predictions.
- The super learner technique is an example of the general method called *stacking*.


Model 1
Model 2
Model 3
Model 4
Training Data (m*n)
New training set for Second level model consisting of predictions from First level model
Second level Model
Final prediction

Q1. Which of the following algorithm is not an example of an ensemble method?

- A. Extra Tree Regressor
  B. Random Forest
  C. Gradient Boosting
  D. Decision Tree
- **Solution: (D)**
- Option D is correct. In case of decision tree, we build a single tree and no ensembling is required.
- **What is true about an ensembled classifier?**
- **1. Classifiers that are more "sure" can vote with more conviction**
  2. Classifiers can be more "sure" about a particular part of the space
  3. Most of the times, it performs better than a single classifier
- A. 1 and 2
  B. 1 and 3
  C. 2 and 3
  D. All of the above
- **Solution: (D)**
- In an ensemble model, we give higher weights to classifiers which have higher accuracies. In other words, these classifiers are voting with higher conviction.
- On the other hand, weak learners are sure about specific areas of the problem. By ensembling these weak learners, we can aggregate the results of their sure parts of each of them.
- The final result would have better results than the individual weak models.

- **Q3. Which of the following option is / are correct regarding benefits of ensemble model?**
- **1. Better performance**
  2. Generalized models
  3. Better interpretability
- A. 1 and 3
  B. 2 and 3
  C. 1 and 2
  D. 1, 2 and 3
- **Solution: (C)**
- 1 and 2 are the benefits of ensemble modeling. Option 3 is incorrect because when we ensemble multiple models, we lose interpretability of the models.
- **Q4) Which of the following can be true for selecting base learners for an ensemble?**
- **1. Different learners can come from same algorithm with different hyper parameters**
  2. Different learners can come from different algorithms
  3. Different learners can come from different training spaces
- A. 1
  B. 2
  C. 1 and 3
  D. 1, 2 and 3
- **Solution: (D)**
- We can create an ensemble by following any / all of the options mentioned above. So option D is correct.
- **Q5. True or False: Ensemble learning can only be applied to supervised learning methods.**
- A. True
  B. False
- **Solution: (B)**
- Generally, we use ensemble technique for supervised learning algorithms. But, you can use an ensemble for unsupervised learning algorithms also.
- **Q6. True or False: Ensembles will yield bad results when there is significant diversity among the models.**
- **Note: All individual models have meaningful and good predictions.**
- A. True
  B. False
- **Solution: (B)**
- An ensemble is an art of combining a diverse set of learners (individual models) together to improvise on the stability and predictive power of the

model. So, creating an ensemble of diverse models is a very important factor to achieve better results.

- **Which of the following is / are true about weak learners used in ensemble model?**
- **1. They have low variance and they don't usually overfit**
  2. They have high bias, so they can not solve hard learning problems
  3. They have high variance and they don't usually overfit
- A. 1 and 2
  B. 1 and 3
  C. 2 and 3
  D. None of these
- **Solution: (A)**
  Weak learners are sure about particular part of a problem. So they usually don't overfit which means that weak learners have low variance and high bias.
- **True or False: Ensemble of classifiers may or may not be more accurate than any of its individual model.**
- A. True
- B. False
- **Solution: (A)**
- Usually, ensemble would improve the model, but it is not necessary. Hence, option A is correct.
- **If you use an ensemble of different base models, is it necessary to tune the hyper parameters of all base models to improve the ensemble performance?**
- A. Yes
  B. No
  C. can't say
- **Solution: (B)**
- It is not necessary. Ensemble of weak learners can also yield a good model.
- **Generally, an ensemble method works better, if the individual base models have ____________?**
- Note: Suppose each individual base models have accuracy greater than 50%.
- A. Less correlation among predictions
  B. High correlation among predictions
  C. Correlation does not have any impact on ensemble output
  D. None of the above
- **Solution: (A)**

- A lower correlation among ensemble model members will increase the error-correcting capability of the model. So it is preferred to use models with low correlations when creating ensembles.

Which of the following ensemble method works similar to above-discussed election procedure?

- **Hint: Persons are like base models of ensemble method.**
- A. Bagging
  B. Boosting
  C. A Or B
  D. None of these
- **Solution: (A)**
- In bagged ensemble, the predictions of the individual models won't depend on each other. So option A is correct.
- **Suppose you are given 'n' predictions on test data by 'n' different models (M1, M2, …. Mn) respectively. Which of the following method(s) can be used to combine the predictions of these models?**
- **Note: We are working on a regression problem**
- **1. Median**
  2. Product
  3. Average
  4. Weighted sum
  5. Minimum and Maximum
  6. Generalized mean rule
- A. 1, 3 and 4
  B. 1,3 and 6
  C. 1,3, 4 and 6
  D. All of above
- **Solution: (D)**
- All of the above options are valid methods for aggregating results of different models (in case of a regression model).
-
- **Suppose, you are working on a binary classification problem. And there are 3 models each with 70% accuracy.**
- **Q13. If you want to ensemble these models using majority voting method. What will be the maximum accuracy you can get?**
- A. 100%
  B. 78.38 %
  C. 44%
  D. 70
- **Solution: (A)**

- Refer below table for models M1, M2 and M3.

Actual output	M1	M2	M3	Output
1	1	0	1	1
1	1	0	1	1
1	1	0	1	1
1	0	1	1	1
1	0	1	1	1
1	0	1	1	1
1	1	1	1	1
1	1	1	0	1
1	1	1	0	1
1	1	1	0	1

If you want to ensemble these models using majority voting. What will be the minimum accuracy you can get?
A. Always greater than 70%
B. Always greater than and equal to 70%
C. It can be less than 70%
D. None of these
Solution: (C)
Refer below table for models M1, M2 and M3.

Actual Output	M1	M2	M3	Output
1	1	0	0	0
1	1	1	1	1
1	1	0	0	0
1	0	1	0	0
1	0	1	1	1
1	0	0	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

How can we assign the weights to output of different models in an ensemble?
1. Use an algorithm to return the optimal weights
2. Choose the weights using cross validation
3. Give high weights to more accurate models
A. 1 and 2
B. 1 and 3
C. 2 and 3
D. All of above
Solution: (D)
All of the options are correct to decide weights of individual models in an ensemble.
Which of the following is true about averaging ensemble?
A. It can only be used in classification problem
B. It can only be used in regression problem
C. It can be used in both classification as well as regression
D. None of these
Solution: (C)
You can use average ensemble on classification as well as regression. In classification, you can apply averaging on prediction probabilities whereas in regression you can directly average the prediction of different models.

- **Meta-learning** refers to *learning about learning*. In machine learning, it most commonly refers to machine learning algorithms that learn from the output of other machine learning algorithms.
- Meta-learning can be used in ensemble learning, e.g., fitting a meta-model on the out-of-fold predictions from the k-fold cross-validation.

Is random forest is ensemble technique?

- Yes, **Random forest** is a **tree-based** *ensemble algorithm* with each tree depending on a collection of random variables. Random forest is using **bagging** as the ensemble method and **decision tree** as the individual model.
- *Random forests* can be used for **classification**, or **regression** and other tasks that operate by constructing a multitude of *decision trees* at a training time.

- For **classification** tasks, the output of the random forest is the class selected by most trees.
- For **regression** tasks, the mean or average prediction of the individual tress is returned.
- *Random forests* correct for decision trees' habit of *overfitting* to their *training set*.

Real world applications of ensembles techniques

Person recognition

- Person recognition is the problem of verifying the identity of a person using the characteristics of that person, typically for security applications. It includes iris recognition, fingerprint recognition, face recognition, and behavior recognition.
- It has problems involving multiple types of features, difficulty in collecting good data, and different misclassification costs (e.g. denying system access to a legitimate user vs. allowing access to an illegitimate user).

Medicine

- Medical applications include analyzing X-ray images, human genome analysis, and examining sets of medical test data to look for anomalies.
- It has problems of having limited training and test examples, imbalanced datasets, too many attributes, and different misclassification costs (i.e., false negatives significantly worse than false positives).
-

  What is incremental learning

- **Incremental learning** refers to the ability of an algorithm to learn from new data that may become available after a *classifier* has already been generated from a previously available dataset.
- An algorithm is said to be an **incremental learning algorithm** if, for a sequence of training datasets, it produces a sequence of hypotheses where the current hypothesis describes all data that have been seen thus far but depends only on previous hypotheses and the current training data.
- **Ensemble-based systems** can be used for such problems by training an additional *classifier* (or an additional ensemble of classifiers) on each dataset that becomes available.

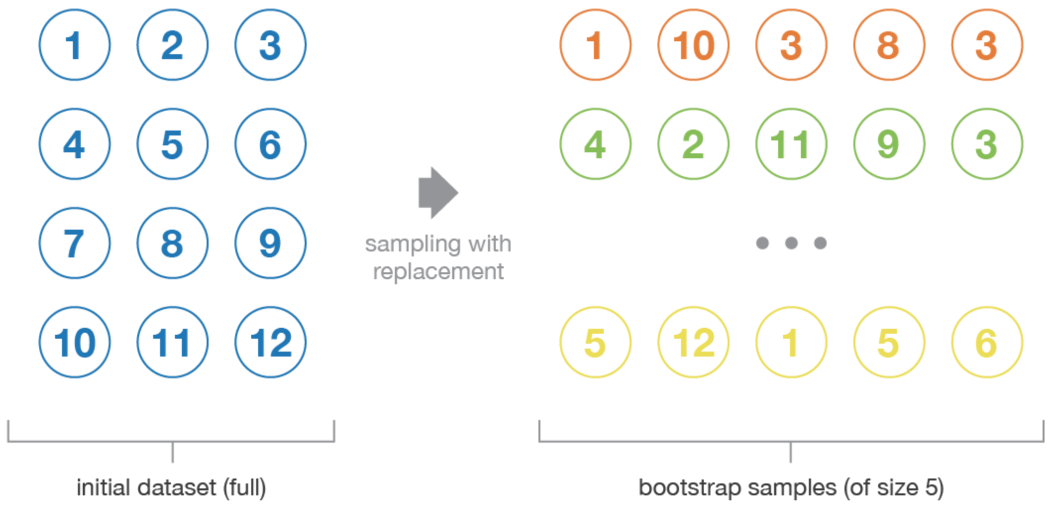
Difference between bagging ,boosting and stacking?

Very roughly, we can say that bagging will mainly focus at getting an ensemble model with less variance than its components whereas boosting and stacking will mainly try to produce strong models less biased than their components (even if variance can also be reduced).

What is bootstrapping?

Bootstrapping

Let’s begin by defining bootstrapping. This statistical technique consists in generating samples of size B (called bootstrap samples) from an initial dataset of size N by randomly drawing with replacement B observations.



What is bagging?

First, we create multiple bootstrap samples so that each new bootstrap sample will act as another (almost) independent dataset drawn from true distribution. Then, we can **fit a weak learner for each of these samples and finally aggregate them such that we kind of “average” their outputs**

What is hard voting and soft voting?

There are several possible ways to aggregate the multiple models fitted in parallel. For a regression problem, the outputs of individual models can literally be averaged to obtain the output of the ensemble model. For classification problem the class outputted by each model can be seen as a vote and the class that receives the majority of the votes is returned by the ensemble model (this is called **hard-voting**). Still for a classification problem, we can also consider the probabilities of each classes returned by all the models, average these probabilities and keep the class with the highest average probability (this is called **soft-voting**). Averages or votes can either be simple or weighted if any relevant weights can be used.

Random forests

Learning trees are very popular base models for ensemble methods. Strong learners composed of multiple trees can be called "forests". Trees that compose a forest can be chosen to be either shallow (few depths) or deep (lot of depths, if not fully grown). **Shallow trees have less variance but higher bias** and then will be better choice for sequential methods that we will described thereafter. **Deep trees, on the other side, have low bias but high variance** and, so, are relevant choices for bagging method that is mainly focused at reducing variance.

In other words shallow trees are overfitted and deep trees are underfitted.

Boosting

Boosting methods work in the same spirit as bagging methods: we build a family of models that are aggregated to obtain a strong learner that performs better. However, unlike bagging that mainly aims at reducing variance, boosting is a technique that consists in fitting sequentially multiple weak learners in a very adaptative way: each model in the sequence is fitted giving more importance to observations in the dataset that were badly handled by the previous models in the sequence. Intuitively, each new model **focus its efforts on the most difficult observations** to fit up to now, so that we obtain, at the end of the process, a strong learner with lower bias (even if we can notice that boosting can also have the effect of reducing variance). Boosting, like bagging, can be used for regression as well as for classification problems.

Bagging is used to reduce variance and boosting is used to reduce bias

Adaptive boosting updates the weights attached to each of the training dataset observations whereas gradient boosting updates the value of these observations.

In adaptative boosting (often called "adaboost"), we try to define our ensemble model as a weighted sum of L weak learners instead of trying to solve it in one single shot (finding all the coefficients and weak learners that give the best overall additive model), we make use of an **iterative optimisation process** that is much more tractable, even if it can lead to a sub-optimal solution. More especially, we add the weak learners one by one, looking at each iteration for the best possible pair (coefficient, weak learner) to add to the current ensemble model.

The main difference with adaptative boosting is in the definition of the sequential optimisation process. Indeed, gradient boosting **casts the problem into a gradient descent one**: at each iteration we fit a weak learner to the opposite of the gradient of the current fitting error with respect to the current ensemble model.

Difference between bagging,boosting and Stacking

Stacking mainly differ from bagging and boosting on two points. First stacking often considers **heterogeneous weak learners** (different learning algorithms are combined) whereas bagging and boosting consider mainly homogeneous weak learners. Second, stacking learns to combine the base models using a meta-model whereas bagging and boosting combine weak learners following deterministic algorithms.

- in bagging methods, several instance of the same base model are trained in parallel (independently from each others) on different bootstrap samples and then aggregated in some kind of “averaging” process
- the kind of averaging operation done over the (almost) i.i.d fitted models in bagging methods mainly allows us to obtain an ensemble model with a lower variance than its components: that is why base models with low bias but high variance are well adapted for bagging
- in boosting methods, several instance of the same base model are trained sequentially such that, at each iteration, the way to train the current weak learner depends on the previous weak learners and more especially on how they are performing on the data
- this iterative strategy of learning used in boosting methods, that adapts to the weaknesses of the previous models to train the current one, mainly allows us to get an ensemble model with a lower bias than its

components: that is why weak learners with low variance but high bias are well adapted for boosting

- in stacking methods, different weak learners are fitted independently from each others and a meta-model is trained on top of that to predict outputs based on the outputs returned by the base models

What is OOB score and validation score?
This is basically used to check the performance of random forest tree algorithms.As like r values,OOB score and validation score do not have standard value with which you can compare and evaluate best performance or poor performance.
OOB score is being used when we do not have a big dataset and splitting into training and validation set is taking away useful data that can be used to train the models. So we basically decide to use the training data as the validation set by using those samples that were not used for training particular trees.

For each tree we have some rows that were not used to train that particular tree. So when evaluating training data for prediction , for each sample we only consider trees that did not use that sample to train themselves.

Let’s say we have 100 trees and 3k samples in the training set. We’re going to evaluate the model. For evaluation we need the model’s prediction for each sample. We start iterating through the samples.

In general we use all 100 trees to make their prediction and average their predictions, but in this OOB case we only use those trees which did not use this sample for training, so it’s very likely this number is less than 100 as some tree probably have used this sample to train itself.

On average OOB score is showing less generalization than the validation score because we are using less trees to get the predictions for each sample. Recall as the number of trees grow, in general we get better predictive power even if it flattens out in the end, so if we are using fewer trees than available we’re getting slightly less accurate models. But since we’re using all of the training data for training instead of keeping them for validation and getting a validation score at the same time, this trade off is not bad.And the more trees we add the less serious this underestimation is.
There's no such thing as good oob_score, its the difference between valid_score and oob_score that matters.

What is difference between weak learners and strong learners?

- Weak learners are models that perform slightly better than random guessing.
- Strong learners are models that have arbitrarily good accuracy.
- Weak and strong learners are tools from computational learning theory and provide the basis for the development of the boosting class of ensemble methods.

What is decision stump?

- **Decision Stump**: A decision tree with a single node operating on one input variable, the output of which makes a prediction directly.

What are advantages of decision tree?

1. Compared to other algorithms decision trees requires less effort for data preparation during pre-processing.
2. A decision tree does not require normalization of data.
3. A decision tree does not require scaling of data as well.
4. Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent.
5. A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders.

What are disadvantages of decision tree:

1. A small change in the data can cause a large change in the structure of the decision tree causing instability.
2. For a Decision tree sometimes calculation can go far more complex compared to other algorithms.
3. Decision tree often involves higher time to train the model.

4. Decision tree training is relatively expensive as the complexity and time has taken are more.

5. **The Decision Tree algorithm is inadequate for applying regression and predicting continuous values.**

How gradient boosting is going to help in supervised learning

In Machine Learning, we use gradient boosting **to solve classification and regression problems**. It is a sequential ensemble learning technique where the performance of the model improves over iterations. This method creates the model in a stage-wise fashion.