Module: ITI104 Machine Learning Algorithms
Assignment 1 – Voting & Stacking Ensembles
Name: Nursharinah Binte Sohaimi
Student ID: 6422706H

Written Assignment

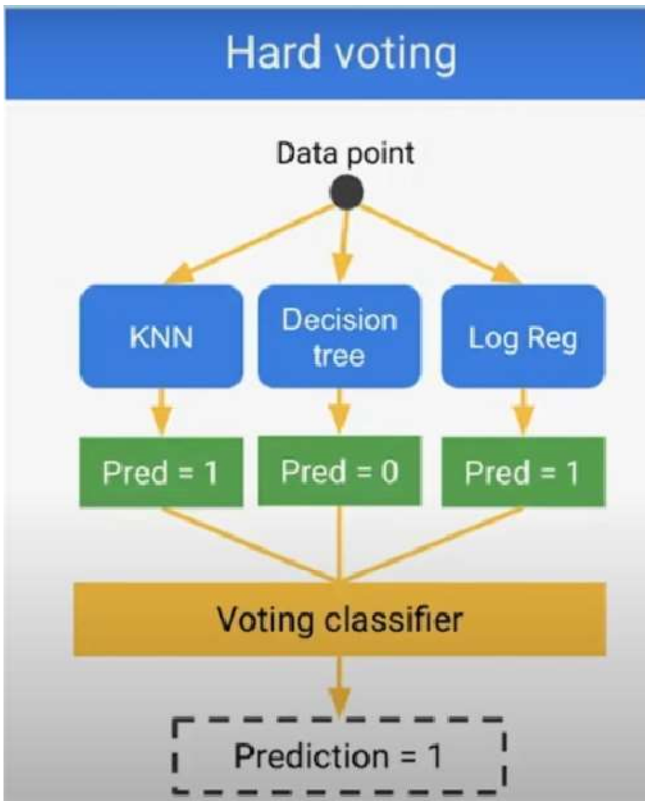
a. Describe how bagging, boosting, voting and stacking ensemble methods work.

1) Voting:

Also known as averaging. It is an ensemble ML algorithm that involves making a prediction that is the average (for regression) or the sum (for classification) of multiple machine learning (ML) models.

- First, we will consider multiple numbers of models (e.g. decision tree regression, ridge regression, lasso regression etc). And each of these models will be trained based on a common (given) dataset. Once the model has been trained, whenever we get a new example (i.e. sample) it will be given as new inputs to the same model. Each of the models will then predict the output. These outputs will later be considered to come out with the final prediction.
- To combine the output predictions, there are multiple methods.
  If it is a classification problem, we will take the majority voting amongst the output (i.e. if ⅔ prediction outcome from 3 models is 1 then the final output from ensemble will be 1).

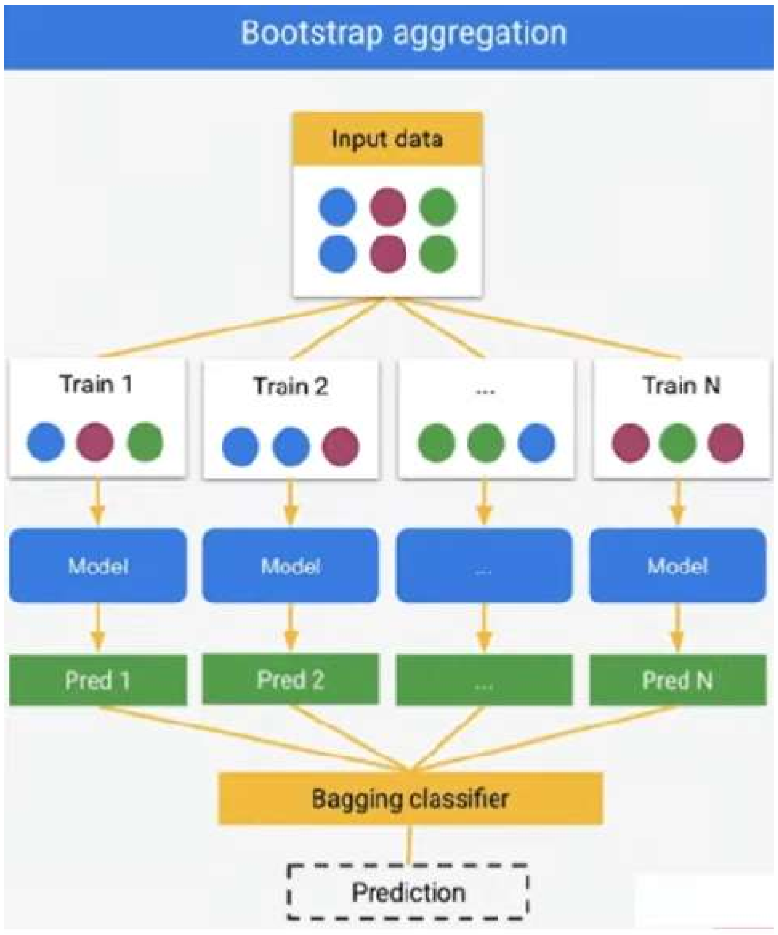
  If it is a regression problem, we will take the average of the probabilities (i.e. values of the outcome) to be the ensemble's final probabilities. Because in regression we get the probabilities as outcomes instead of the class value for classification.



2) Bagging:

Also known as bootstrap aggregation. It is an ensemble ML meta-algorithm that is designed to improve the stability & accuracy of ML algorithms like classifications & regressions. What it does is, it decreases the variance & helps to avoid overfitting. This is usually applied to decision tree methods. Note that bagging is a special case of the model averaging approach.

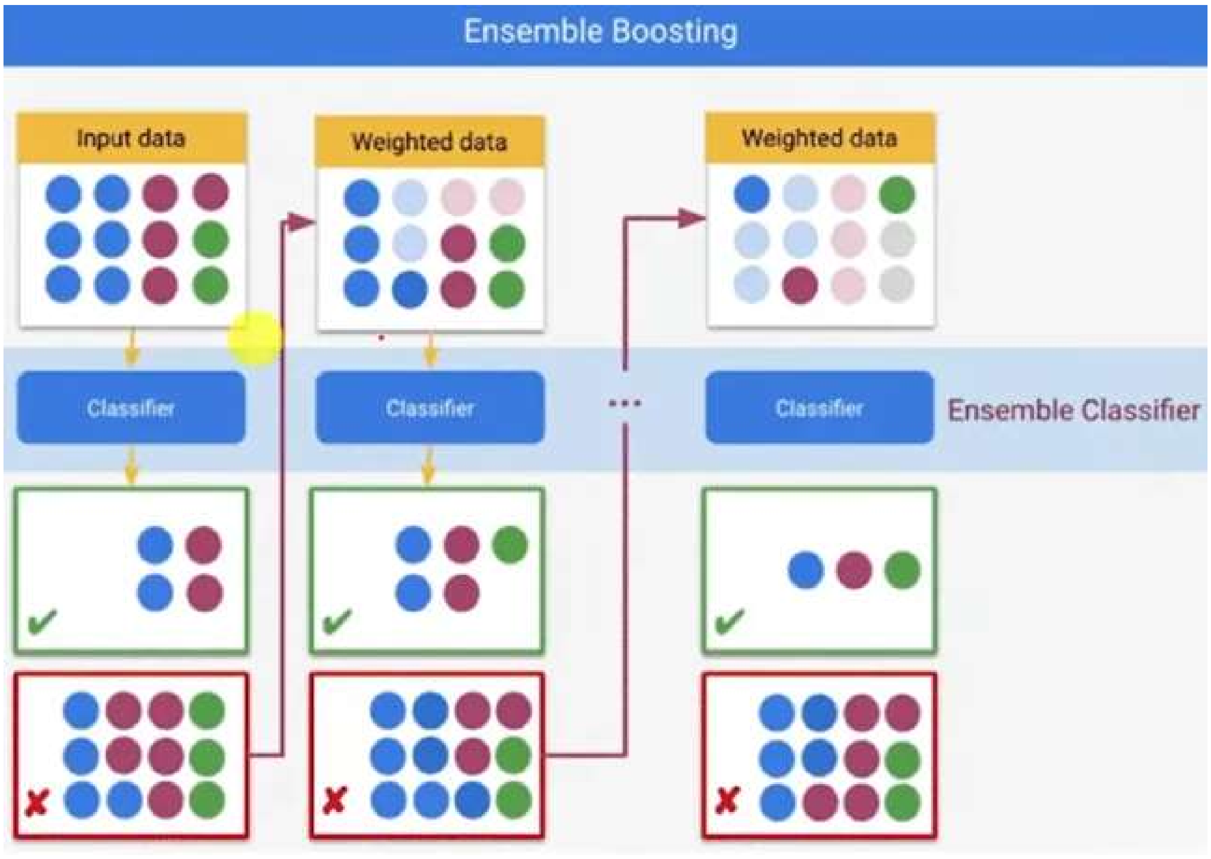
- Rather than considering the multiple number of models, the input data (from dataset) will be divided into multiple number of groups for training. Each particular n groups will be given as input to the same model.
- If for example in classification problem, we will get a prediction as an output. These prediction outcomes will be combined in a voting way (highest vote wins).
- If for example in regression problem, we will get a probability as an output. These probability outcomes will be combined in an averaging way.



3) Boosting:

It is an ensemble ML algorithm that attempts to build a strong classifier from the number of weak classifiers. The working principle of boosting is done by building a final model by using weak models in series.

- Firstly, given a training dataset we will consider any basic ML classifier to be trained. After training the ML model, there is a possibility that some part of the training data will be classified correctly. While some parts will be classified incorrectly.
- In boosting, it will consider the incorrectly classified data & assign more weightage to it. This is accomplished by building the second model where it tries to correct those particular errors present in the first model.
- This procedure is iterated & models are added until either the complete training dataset is predicted/classified correctly or the maximum number (i.e. threshold) of models to be added is achieved.

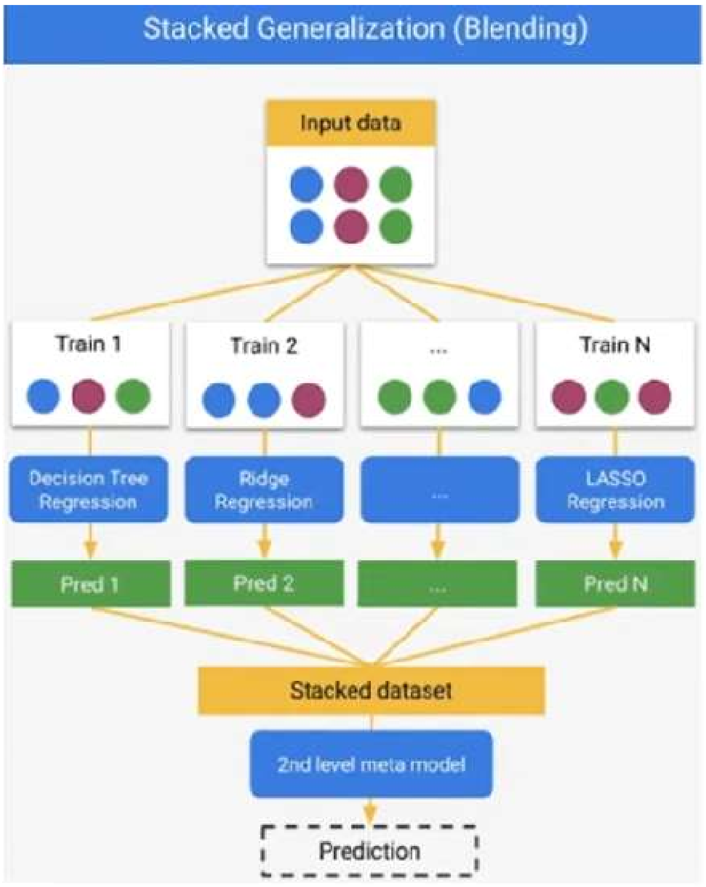


4) Stacking:

Also known as Stacked Generalization or Blending. In a traditional ensemble learning, we have multiple classifiers trying to fit a training dataset to approximate the target function. Or, the dataset is divided into multiple number of groups and train them with a particular model. Once these model(s) are trained or on different training dataset, we will get multiple outputs. Hence, we will need to find a combining mechanism to combine the results (i.e. final predictions). This can be through voting (majority wins), weighted voting (some classifier has more authority then the others), averaging the results (for regression problems), etc.

In stacking, the ensemble ML algorithm has a combining mechanism that takes the output of classifiers (Level 0 classifiers) to be used as training data for another classifier (Level 1 classifier) to approximate the same target function. Basically, it lets the Level 1 classifier to figure out the combining mechanism.

- First the input training dataset will be divided into multiple number of training groups. Each of these particular training groups will be given as an input to different types of models (e.g. for regression problems: decision tree regression, ridge regression, lasso regression etc | for classification problems: decision tree classifier, SVM classifier etc).
- Each of these models will make some predictions with an output of classes (for classification problem) or probability (for regression problem).
- Rather than combining the results using either majority voting (for classification) or averaging (for regression), the outputs are considered as an input to the 2nd level meta-model. The combination of outputs (from 1st level meta-model) is known as a “stacked dataset”.
- Based on the output of the 2nd level model, it will come up with the final prediction.



b. Choose two classical machine learning models that you have learnt in the course such as linear regression, logistic regression, K-nearest neighbours, Naïve Bayes, decision trees, SVM, and explain how each model works.

Naive Bayes:

Naive Bayes classifiers are a collection of classification algorithms (not a single algorithm) based on Bayes' Theorem. These family of algorithms share a common principle where every pair of features being classified is independent of each other. Thus the term "Naive" in the algorithm name. The classifier assumes that the features used to describe an observation are conditionally independent, given the class label.

Naive Bayes algorithm is used for classification problems, being one of the most simple & effective classification algorithms. It is highly used in text classification where data contains high dimensionality. Other uses include spam filtering, sentiment detection, rating classification etc. The main advantage of this classifier is its speed, where it has the ability to make fast predictions easily with high dimensions of data.

A Naive Bayes model uses a probabilistic classifier. Where it predicts the probability of an instance belonging to a class with a given set of feature values. This is because it assumes feature independence where each feature contributes to the predictions with no relation between each other. This classifier makes use of Bayes theorem for training & prediction.

Fundamentally, Naive Bayes makes an assumption that each features has the following characteristics:

1. Feature independence: each data feature is conditionally independent of each other, given the class label.
2. Continuous features are normally distributed: features of continuous type are to be normally distributed within each class.
3. Discrete features are multinomial distributed: features of discrete type are to have multinomial distribution within each class.
4. Features have the same weight (importance): all features contribute equally to prediction of the class label.
5. No missing data: dataset shall not contain missing values.

Workings of Naive Bayes Classifier:

- Convert given dataset into frequency tables.
- Generate a Likelihood table. Accomplished by finding the probabilities of given features.
- Use Bayes theorem to calculate the posterior probability.

Bayes Theorem (i.e. Bayes' Rule or Bayes' Law) determines the probability of a hypothesis with prior knowledge. It depends on the conditional probability given below:

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

- Posterior probability, P(A|B), is a probability of hypothesis A on the observed event B.
- Likelihood probability, P(B|A), is a probability of the evidence given that the probability of a hypothesis is true.
- Prior Probability, P(A), is a probability of hypothesis before observing the evidence.
- Marginal Probability, P(B), is a probability of evidence where it does not equal to zero.
- Note: events A & B do not equal zero. Basically, we're trying to find the probability of event A, given that event B is true. Event B is termed as evidence.

Applying Bayes' theorem into dataset:	Input naïve assumption into Bayes' theorem :	To create a classifier, need to find the probability of given set inputs for all possible values of class variable y and pick up the output with maximum probability:
$P(y\|X) = \frac{P(X\|y)P(y)}{P(X)}$	Split evidence into the individual parts. Where, if any two events A & B are independent, then: `P(A,B) = P(A)P(B)`	$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i\|y)$
Where y is class variable, X is dependent feature vector (of size n).	$P(y\|x_1, \ldots, x_n) = \frac{P(x_1\|y)P(x_2\|y)\ldots P(x_n\|y)P(y)}{P(x_1)P(x_2)\ldots P(x_n)}$	Finally left with calculating $P(y)$ & $P(x_i\|y)$.

Note that P(y) is known as class probability. There are different Naive Bayes classifiers that can be implemented in a model. They differs mainly in the assumptions to make regarding the distribution of conditional probability, P(x_i|y).

(1) <u>Multinomial Naive Bayes:</u>
Each feature vector is represented in a histogram manner to show the frequencies with which certain events happened. This is generated by a multinomial distribution. This event model is typically used for document classification.

(2) <u>Gaussian Naive Bayes:</u>
Each feature associated with continuous values is assumed to be distributed according to Gaussian distribution. Also known as normal distribution, it has a characteristic of a bell shaped curve when plotted. Its symmetric point is the mean of the feature values. As the likelihood of the features is assumed to be Gaussian, the conditional probability is given by:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

(3) <u>Bernoulli Naive Bayes:</u>
Is a multivariate Bernoulli event model, where features are independent booleans (binary variables) describing input. This model is popularly used for document classification, where binary term occurrence features are used rather than term frequencies in Natural Language Processing problems.

Advantages of Naive Bayes Classifier:

- Computationally efficient and easy to implement.
- Effective for cases that have a large number of features.
- Performs well in cases with limited training data.
- Performs well in presence of categorical features.
- For data that has numerical features, they are assumed to be of normal distributions.

Disadvantages of Naive Bayes Classifier:

- Assumes each feature is independent, which might not be true in real-world data context.
- As all features are given the same importance, irrelevant attributes can influence the outcomes.
- May assign zero probability to unseen events, leading to poor generalization.

Applications of Naive Bayes Classifier:

- Spam email filtering: based on features, classify emails as spam or not.
- Text classification: typically used in sentiment analysis, document categorization, topic classification.
- Medical diagnosis: based on symptoms, predict the likelihood of a disease.
- Credit scoring: help in evaluating an individual creditworthiness for loan approval.
- Weather prediction: based on various environment factors, classify weather conditions.

Decision Trees:

Decision trees are used in ML models for predictive modeling based on input data. It is a type of supervised learning algorithm. Decision trees are great for classification & regression problems because of its algorithm's decision structures. Fundamentally, the tree-like structure has each internal node tests on attribute, each branch corresponds to attribute value, and each leaf node represents the final prediction or decision.

To form a decision tree, there needs to be a repetitive process of partitioning the data based on values of different attributes. To select the best attribute for splitting the data at each internal node, the algorithm selects based on certain criteria. For example, information gain or Gini impurity. The splitting is an iterative process until a stopping criterion is met. For example, reaching maximum depth or having a minimum number of instances in a leaf node.

A decision tree approach is to use a tree representation to solve a problem. Whereby each leaf node corresponds to a class label, and attributes are represented on the internal node of the tree. Any boolean function can be represented on discrete attributes using a decision tree.

At the beginning, the whole training dataset is considered & set as the root. However, there are some criteria to be met:

1. Categorical feature values are preferred. Else if its continuous, they are converted to discrete before building the model.
2. For attribute values, records are distributed recursively.
3. Statistical methods are used to order attributes as root or the internal node.

The inner workings of a decision tree is based on the Sum of Product form (i.e. Disjunctive Normal Form). The major challenge in decision trees is the identification of the attribute for the root node at each level, this process is known as attribute selection. The popular attribute selection methods are Information Gain & Gini Index:

(1) <u>Information Gain</u>

Whenever the training set is partitioned into smaller subsets by the decision tree node, the entropy changes. The measure of this change in entropy is known as Information Gain. Entropy is the measure of uncertainty of a random variable. This measure characterizes the impurity of an arbitrary collection of samples. The higher the entropy, the more the information content.

$$Gain(S, A) = Entropy(S) - \sum_{v}^{A} \frac{|S_v|}{|S|} . Entropy(S_v)$$

(2) <u>Gini Index</u>

Gini Index measures how often a randomly chosen element would be incorrectly identified (i.e. the inequality or impurity of a distribution). Its value ranges from 0 to 0.5. A lower Gini Index is preferable. The value 0 indicates a pure distribution (homogeneous), where all instances belong to the same class. The value 0.5 indicates a maximally impure distribution (heterogeneous), where instances are evenly distributed across classes. In Scikit Learn library, it supports "Gini" criteria for Gini Index, and by default takes the "gini" value.

$$\text{Gini}(S) = 1 - \sum_{i=1}^{c} p_i^2$$

Gini Index is calculated by summing the squared probabilities of each outcome in a distribution and subtracting this result from 1. Gini Index is used in decision trees to evaluate the quality of a split by measuring the differences between impurity of the parent node, and the weighted impurity of the child nodes.

Compared to entropy (which measures impurity), Gini Index is much faster to compute and more sensitive to changes in class probabilities. However, Gini Index tends to favor splits that create equally sized child nodes. Despite them not being optimal for classification accuracy. In practice, whether to choose Gini Index or Information Gain (entropy), depends on the specific problem & dataset. Often, it also requires multiple experimentation & tuning.

Building Decision Tree using Information Gain:

- In the beginning, root node is associated with all training instances.
- To label each node based on the chosen attribute, use the Information Gain method.
- A root-to-leaf path should not contain the same discrete attribute twice.
- In each iteration, construct each subtree on a subset of the training set that would be classified down that path in the tree.
- If the remaining training set are either all positive or all negative, label that node "yes" or "no" accordingly.
- If no attribute remains, label with a majority vote of training instances left at that node.
- If no instances remain, label with a majority vote of the parent's training instances.
- Note: to choose the best-suited feature as root node to make a split on the dataset, the algorithm chooses the option with the maximum Information Gain value.

Advantages of Decision Trees:

- Can easily be interpreted and understood by anyone, even non-experts.
- Able to handle both data types (categorical & numerical) without need for extensive preprocessing.
- Provides insights into feature importance for decision making.
- Can handle missing values & outliers without significant impact to output.
- Can provide solutions for classification & regression problems.

Disadvantages of Decision Trees:

- Possibility that the trained model will be overfitted.
- Limited ability to generalize due to sensitivity to small changes in data. Especially in situations where training data is not representative.
- In situations where data is imbalanced, there's potential for bias to occur.

Applications of Decision Trees algorithm:

- Provides simple visualization to help interpret the underlying decision processes in a model.
- Needs for versatility to adapt to a variety of datasets. Decision trees are proficient with both numerical & categorical data due to their autonomous feature selection capability.
- Provide solutions to either categorisation or regression tasks. A comprehensible insight into the decision logic can be displayed.
- Flexibility to portray complex choice scenarios. Decision trees can consider a variety of causes & outcomes due to their hierarchical structure.

c. Describe the advantages and benefits of Voting vs Stacking methods.

In ensemble learning, its final prediction is made by combining the predictions of multiple individual models. Ensemble learning is commonly used for classification problems, but can also be applied to regression problems. In ensemble technique, each individual model makes a prediction independently based on input data. To determine the final prediction, it then combines the predictions of these models through multiple choices of algorithms which two of them are:

Voting Methods:
There are three types of ensembles voting techniques most commonly used: Hard (Majority) Voting, Soft Voting & Weighted Voting.

1. **Hard (Majority) Voting:**
   In ensemble learning, each model predicts the class label for given input. The selection of the final prediction is dictated by the class label that receives the majority of votes. In situations where there are equal number of votes for different classes, tie-breaking rules can be applied. This is where the algorithm will select the class with highest probability or confidence that was predicted by the model.

2. **Soft Voting:**
   Instead of predicting discrete classes, the individual models provide probability estimates or confident scores for each class label. This approach combines the predicted probabilities, either by averaging or weighted averaging, from each model. The selected final prediction shall be the class with highest average probability.

3. **Weighted Voting:**
   In this approach, different weights are assigned to the predictions of individual models. This is based on their performance and reliability. Each of the model's predictions are then multiplied by its corresponding weight. To obtain the final prediction, the weighted predictions are either summed or averaged. Note that weights can be manually assigned or learned through techniques like cross-validation or optimization algorithms.

Advantages of Voting Methods:

1. **Improved Accuracy:**
   It leverages diversity of multiple models and the collective knowledge to achieve more accurate predictions than any single model by itself. It can reduce bias & variance, and hence achieve better overall performance.

2. **Robustness:**
   Compared to individual models, this technique is more robust to noise or any incorrect predictions. By considering multiple viewpoints, it reduces the impact of individual model biases or errors.

3. **Model Combination:**
   As each different type of model has its own strength and weaknesses, this technique allows the combination of them. This is achieved by aggregating their predictions. An ensemble voting can then benefit from the complementary aspects of different models.

4. **Interpretability:**
   This technique can provide insights into the relative importance & agreement among the different models in ensemble learning. It allows analyzing of patterns & consistency in predictions across the ensemble. Hence, aiding in model's interpretability.

Stacking Methods:

Is an ensemble strategy in ML that integrates multiple models to improve the model's overall performance. This is achieved by combining the predictions of numerous first-level (base) models (learners) to obtain a final prediction. The multiple base models are trained with the same training dataset, which then feeds their predictions into a higher level (second-level) model known as meta-model. In stacking, the predictions of different base models are combined and fed to the meta-model to make the final prediction. Thus, the predictive performance will be much better than utilizing a single model.

Advantages of Stacking Methods:

1. **Improved Predictive Performance:**
   Can reduce bias & variance in the final prediction. This is achieved by merging multiple base models, which resulted in improved overall predictive performance. This technique allows the meta-model to learn from the capabilities of the base models & potentially capture intricate patterns that individual ML algorithm models may not be able to, alone.

2. **Model Diversity:**
   Promotes the varied base models usage that can be trained using a variety of algorithms, architecture & hyperparameter settings. This technique can lessen the chances of overfitting while also making sure that the ensemble model is more robust to different types of data.

3. **Flexibility:**
   This technique is versatile in providing solutions to a variety of ML problems which includes classification, regression & time series forecasting problems. Stacking can be implemented with a variety of basic models like decision trees, support vector machines, neural networks etc.

4. **Interpretability:**
   This technique is able to provide significant insights on the multiple base models used, as well as the reasoning behind the final prediction. The relative importance and interpretability can be acquired by analyzing the weights or contributions of each base model in the meta-model.

Voting vs Stacking:

Fundamentally, their differences are based on how the final aggregation works for each strategy. The decision of the voting algorithm's classifier can either be from taking the class that appears in most cases or by weights of the predictions assigned. The outcome of the stacking algorithm comes from performing the final aggregation by using a blender/meta classifier.

In the voting algorithm, there are some constant methods in generating the outcome. In the stacking algorithm, there is some variability. As it takes advantage of the predictions (from other multiple base models) as a new representation of the problem. This creates another abstraction layer for the meta-model to learn how to predict the correct label.

d. Describe the advantages and benefits of ensemble methods over single classifier/regressor.

In traditional ML, a single model is used to solve a given problem. Though it may happen that this classical model may or may not perform better on the given problem definition.

Advantages of single classifier/regressor:

- For simple problems that have small and clean datasets. (a single high-performing model is enough in this situation).
- Able to provide model interpretability and decent performance.
- Suitable for situations when faced with very limited computational resources.
- Can be used for real-time prediction needs where there's a requirement for low latency.

Benefits of ensemble methods:
Ensemble learning is a supervised learning technique used in ML to improve overall performance by combining the predictions from multiple classical models. In other words, ensemble learning utilizes the advantages (i.e. leverage the strengths) of multiple base models, called "weak learners", to compensate (i.e. minimize) for each model's weaknesses. As generally, "weak learner" models do not perform so well by themselves either due to high bias or high variance. These ensemble techniques use different logics to then improve model validation. Thus, enhances the predictive capability compared to a single model.

There are two types of ensemble models: homogeneous & heterogeneous.

(1) Homogeneous ensemble model uses a single base learning algorithm.
(2) Heterogeneous ensemble model uses different base learning algorithms.

Note that choice of "weak learners" should be coherent with the way the models are combined. In other words, when choosing base models with high variance & low bias, the combination method should aim to reduce the variance and vice versa.

Has three popular combinations methods:

1. **<u>Bagging</u>**
    a. considers homogeneous weak learners, focuses on reducing the variance.
    b. Is achieved by training base models on slightly different data in parallel (bootstrap sampling).

2. **<u>Boosting</u>**
    a. considers homogeneous weak learners, focuses on reducing the bias.
    b. Is achieved with weighted average from different models whereby the models are trained sequentially. Each new model (from previous) focuses more on previously misclassified instances to boost the chances of a correct classification.
3. **<u>Stacking</u>**
    a. considers heterogeneous weak learners, focuses on reducing bias & variance.
    b. Is achieved by training a meta-model to combine predictions from multiple base models, where it learns how to best combine them. Thus, minimizing total error.

<u>Advantages of ensemble learning:</u>

- Improved model performance with multiple perspectives combined from multiple models.
- Has better generalizability due to reduced variance.
- Combines various expertise to handle concept drift.
- Is robust against noise & outliers in data.
- Can boost predictive capability & is less prone to overfitting compared to single models.

<u>Ensemble learning VS single classical ML algorithm:</u>

1. **<u>Enhanced generalization & robustness:</u>**
    a. By evaluating multiple hypotheses, a combined model allows for improved generalization by being better at capturing true signals & ignoring false patterns.
    b. This robustness comes from its ability to aggregate predictions to smooth out variance.
    c. Compared to single models which are highly prone to overfit due to noise.
2. **<u>Flexibility to handle complex data environments:</u>**
    a. In the real-world, data tends to grow exponentially in terms of volume, variety & velocity which increases the complexity of problems.

b. To overcome this, ensemble allows for distributed training on separate data chunks from a single dataset.
c. Single model, however, struggles to learn from sparse & noisy big data.