Machine Learning Interview Questions & Answers for Data Scientists

Questions

Q18: You are working on a clustering problem, what are different evaluation metrics that can be used, and how to choose between them?

Q19: What is the ROC curve and when should you use it?

Q20: What is the difference between hard and soft voting classifiers in the context of ensemble learners?

Q21: What is boosting in the context of ensemble learners discuss two famous boosting methods

Q22: How can you evaluate the performance of a dimensionality reduction algorithm on your dataset?

Q23: Define the curse of dimensionality and how to solve it.

Q24: In what cases would you use vanilla PCA, Incremental PCA, Randomized PCA, or Kernel PCA?

Q25: Discuss two clustering algorithms that can scale to large datasets

Q26: Do you need to scale your data if you will be using the SVM classifier and discus your answer

Q27: What are Loss Functions and Cost Functions? Explain the key Difference Between them.

Q28: What is the importance of batch in machine learning and explain some batch depend gradient descent algorithm?

Q29: What are the different methods to split a tree in a decision tree algorithm?

Q30: Why boosting is a more stable algorithm as compared to other ensemble algorithms?

Q31: What is active learning and discuss one strategy of it?

Q32: What are the different approaches to implementing recommendation systems?

Q33: What are the evaluation metrics that can be used for multi-label classification?

Q34: What is the difference between concept and data drift and how to overcome each of them?

Q35: Can you explain the ARIMA model and its components?

Q36: What are the assumptions made by the ARIMA model?

Questions & Answers

Q1: Mention three ways to make your model robust to outliers?

Investigating the outliers is always the first step in understanding how to treat them. After you understand the nature of why the outliers occurred you can apply one of the several methods mentioned here.

Q2: Describe the motivation behind random forests and mention two reasons why they are better than individual decision trees?

The motivation behind random forest or ensemble models in general in layman's terms, Let's say we have a question/problem to solve we bring 100 people and ask each of them the question/problem and record their solution. The rest of the answer is here

Q3: What are the differences and similarities between gradient boosting and random forest? and what are the advantage and disadvantages of each when compared to each other?

Similarities:

1. Both these algorithms are decision-tree based algorithms
2. Both these algorithms are ensemble algorithms
3. Both are flexible models and do not need much data preprocessing.

The rest of the answer is here

Q4: What are L1 and L2 regularization? What are the differences between the two?

Answer:

Regularization is a technique used to avoid overfitting by trying to make the model more simple.The rest of the answer is here

Q5: What are the Bias and Variance in a Machine Learning Model and explain the bias-variance trade-off?

Answer:

The goal of any supervised machine learning model is to estimate the mapping function (f) that predicts the target variable (y) given input (x). The prediction error can be broken down into three parts: The rest of the answer is here

Q6: Mention three ways to handle missing or corrupted data in a dataset?

Answer:

In general, real-world data often has a lot of missing values. The cause of missing values can be data corruption or failure to record data. The rest of the answer is here

Q7: Explain briefly the logistic regression model and state an example of when you have used it recently?

Answer:

Logistic regression is used to calculate the probability of occurrence of an event in the form of a dependent output variable based on independent input variables. Logistic regression is commonly used to estimate the probability that an instance belongs to a particular class. If the probability is bigger than 0.5 then it will belong to that class (positive) and if it is below 0.5 it will belong to the other class. This will make it a binary classifier.

It is important to remember that the Logistic regression isn't a classification model, it's an ordinary type of regression algorithm, and it was developed and used before machine learning, but it can be used in classification when we put a threshold to determine specific categories"

There is a lot of classification applications to it:

Classify email as spam or not, To identify whether the patient is healthy or not, and so on.

Q8: Explain briefly batch gradient descent, stochastic gradient descent, and mini-batch gradient descent? and what are the pros and cons for each of them?

Gradient descent is a generic optimization algorithm cable for finding optimal solutions to a wide range of problems. The general idea of gradient descent is to tweak parameters iteratively in order to minimize a cost function.

Batch Gradient Descent: In Batch Gradient descent the whole training data is used to minimize the loss function by taking a step towards the nearest minimum by calculating the gradient (the direction of descent)

Pros: Since the whole data set is used to calculate the gradient it will be stable and reach the minimum of the cost function without bouncing (if the learning rate is chosen cooreclty)

Cons:

Since batch gradient descent uses all the training set to compute the gradient at every step, it will be very slow especially if the size of the training data is large.

Stochastic Gradient Descent:

Stochastic Gradient Descent picks up a random instance in the training data set at every step and computes the gradient-based only on that single instance.

Pros:

1. It makes the training much faster as it only works on one instance at a time.
2. It become easier to train large datasets

Cons:

Due to the stochastic (random) nature of this algorithm, this algorithm is much less regular than the batch gradient descent. Instead of gently decreasing until it reaches the minimum, the cost function will bounce up and down, decreasing only on average. Over time it will end up very close to the minimum, but once it gets there it will continue to bounce around, not settling down there. So once the algorithm stops the final parameter are good but not optimal. For this reason, it is important to use a training schedule to overcome this randomness.

Mini-batch Gradient:

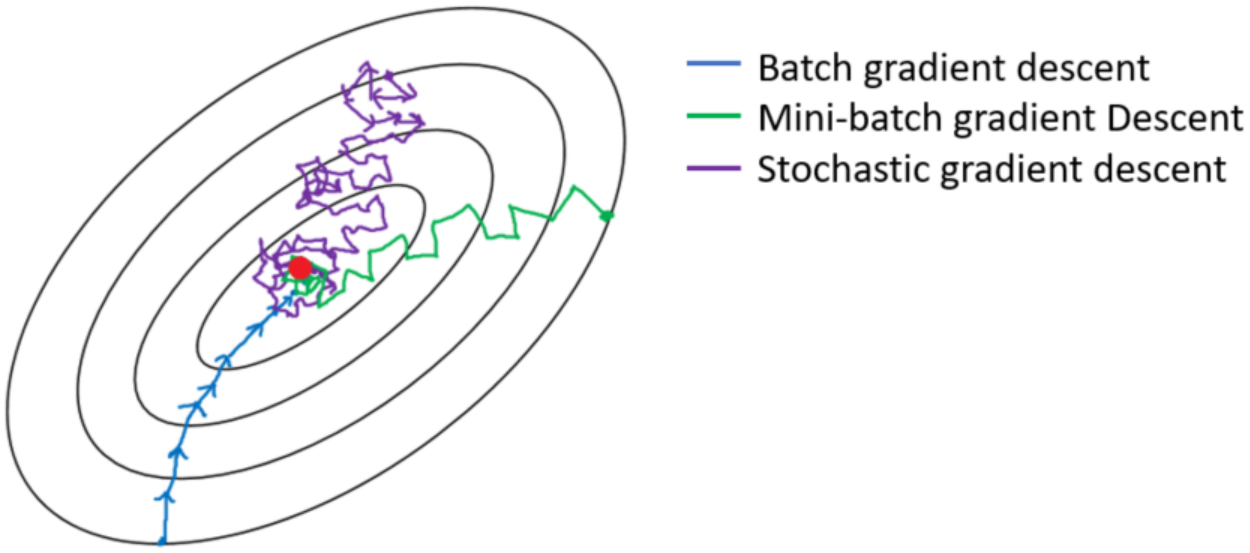
At each step instead of computing the gradients on the whole data set as in the Batch Gradient Descent or using one random instance as in the Stochastic Gradient Descent, this algorithm computes the gradients on small random sets of instances called mini-batches.

Pros:

1. The algorithm's progress space is less erratic than with Stochastic Gradient Descent, especially with large mini-batches.
2. You can get a performance boost from hardware optimization of matrix operations, especially when using GPUs.

Cons:

1. It might be difficult to escape from local minima.



Q9: Explain what is information gain and entropy in the context of decision trees?

Entropy and Information Gain are two key metrics used in determining the relevance of decision making when constructing a decision tree model and to determine the nodes and the best way to split.

The idea of a decision tree is to divide the data set into smaller data sets based on the descriptive features until we reach a small enough set that contains data points that fall under one label.

Entropy is the measure of impurity, disorder, or uncertainty in a bunch of examples. Entropy controls how a Decision Tree decides to split the data. Information gain calculates the reduction in entropy or surprise from transforming a dataset in some way. It is commonly used in the construction of decision trees from a training dataset, by evaluating the information gain for each variable, and selecting the variable that maximizes the information gain, which in turn minimizes the entropy and best splits the dataset into groups for effective classification.

Q10: Explain the linear regression model and discuss its assumption?

Linear regression is a supervised statistical model to predict dependent variable quantity based on independent variables. Linear regression is a parametric model and the objective of linear regression is that it has to learn coefficients using the training data and predict the target value given only independent values.

Some of the linear regression assumptions and how to validate them:

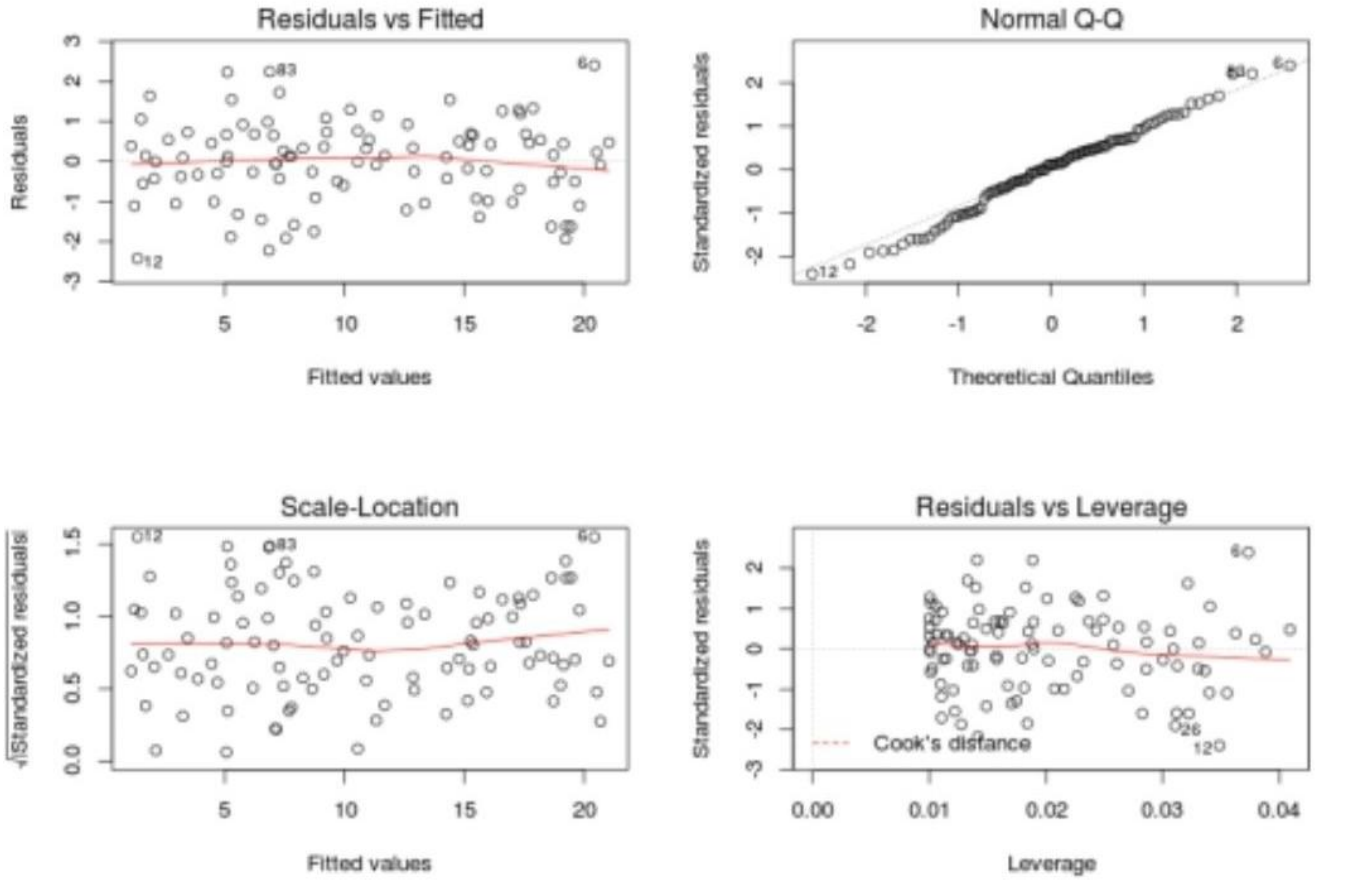
1. Linear relationship between independent and dependent variables
2. Independent residuals and the constant residuals at every x We can check for 1 and 2 by plotting the residuals(error terms) against the fitted values (upper left graph). Generally, we should look for a lack of patterns and a consistent variance across the horizontal line.
3. Normally distributed residuals We can check for this using a couple of methods:

- Q-Q-plot(upper right graph): If data is normally distributed, points should roughly align with the 45-degree line.
- Boxplot: it also helps visualize outliers
- Shapiro–Wilk test: If the p-value is lower than the chosen threshold, then the null hypothesis (Data is normally distributed) is rejected.

4. Low multicollinearity

- you can calculate the VIF (Variable Inflation Factors) using your favorite statistical tool. If the value for each covariate is lower than 10 (some say 5), you're good to go.

The figure below summarizes these assumptions.



Q11: Explain briefly the K-Means clustering and how can we find the best value of K?

K-Means is a well-known clustering algorithm. K-Means clustering is often used because it is easy to interpret and implement. The rest of the answer is here

Q12: Define Precision, recall, and F1 and discuss the trade-off between them?

Precision and recall are two classification evaluation metrics that are used beyond accuracy. The rest of the answer is here

Q13: What are the differences between a model that minimizes squared error and the one that minimizes the absolute error? and in which cases each error metric would be more appropriate?

Both mean square error (MSE) and mean absolute error (MAE) measures the distances between vectors and express average model prediction in units of the target variable. Both can range from 0 to infinity, the lower they are the better the model.

The main difference between them is that in MSE the errors are squared before being averaged while in MAE they are not. This means that a large weight will be given to large errors. MSE is useful when large errors in the model are trying to be avoided. This means that outliers affect MSE more than MAE, that is why MAE is more robust to outliers. Computation-wise MSE is easier to use as the gradient calculation will be more straightforward than MAE, which requires linear programming to calculate it.

Q14: Define and compare parametric and non-parametric models and give two examples for each of them?

Answer:

Parametric models assume that the dataset comes from a certain function with some set of parameters that should be tuned to reach the optimal performance. For such models, the number of parameters is determined prior to training, thus the degree of freedom is limited, and reduces the chances of overfitting.

Ex. Linear Regression, Logistic Regression, LDA

Nonparametric models don't assume anything about the function from which the dataset was sampled. For these models, the number of parameters is not determined prior to training, thus they are free to generalize the model based on the data. Sometimes these models overfit themselves while generalizing. To generalize they need more data in comparison with Parametric Models. They are relatively more difficult to interpret compared to Parametric Models.

Ex. Decision Tree, Random Forest.

Q15: Explain the kernel trick in SVM and why we use it and how to choose what kernel to use?

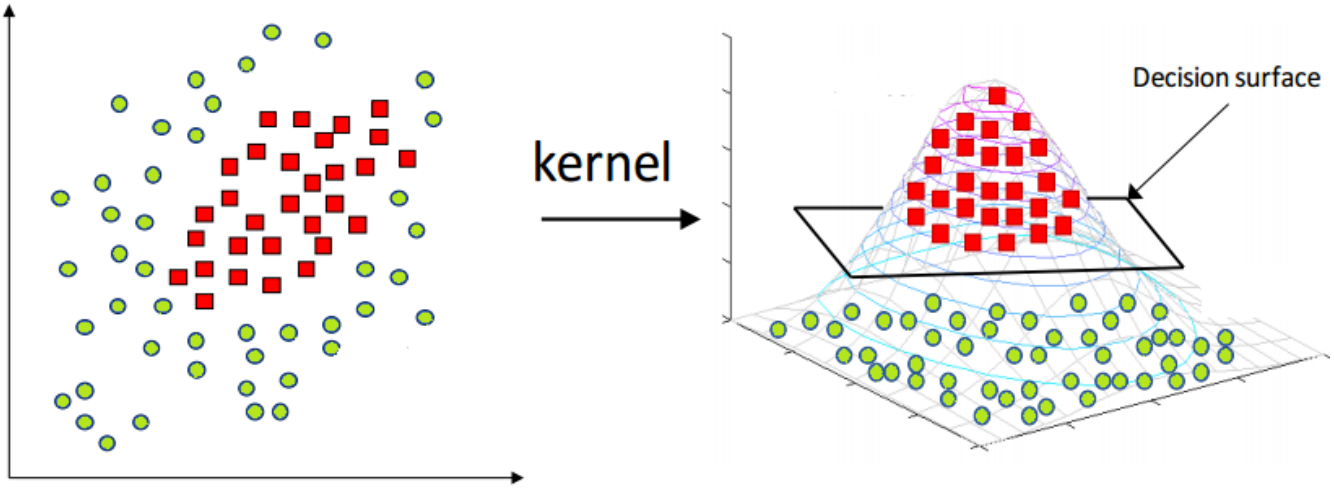
Answer: Kernels are used in SVM to map the original input data into a particular higher dimensional space where it will be easier to find patterns in the data and train the model with better performance.

For eg.: If we have binary class data which form a ring-like pattern (inner and outer rings representing two different class instances) when plotted in 2D space, a linear SVM kernel will not be able to differentiate the two classes well when compared to a RBF (radial basis function) kernel, mapping the data into a particular higher dimensional space where the two classes are clearly separable.

Typically without the kernel trick, in order to calculate support vectors and support vector classifiers, we need first to transform data points one by one to the higher dimensional space, and do the calculations based on SVM equations in the higher dimensional space, then return the results. The ‘trick’ in the kernel trick is that we design the kernels based on some conditions as mathematical functions that are equivalent to a dot product in the higher dimensional space without even having to transform data points to the higher dimensional space. i.e we can

calculate support vectors and support vector classifiers in the same space where the data is provided which saves a lot of time and calculations.

Having domain knowledge can be very helpful in choosing the optimal kernel for your problem, however in the absence of such knowledge following this default rule can be helpful: For linear problems, we can try linear or logistic kernels and for nonlinear problems, we can use RBF or Gaussian kernels.



Q16: Define the cross-validation process and the motivation behind using it?

Cross-validation is a technique used to assess the performance of a learning model in several subsamples of training data. In general, we split the data into train and test sets where we use the training data to train our model and the test data to evaluate the performance of the model on unseen data and validation set for choosing the best hyperparameters. Now, a random split in most cases(for large datasets) is fine. But for smaller datasets, it is susceptible to loss of important information present in the data in which it was not trained. Hence, cross-validation though computationally bit expensive combats this issue.

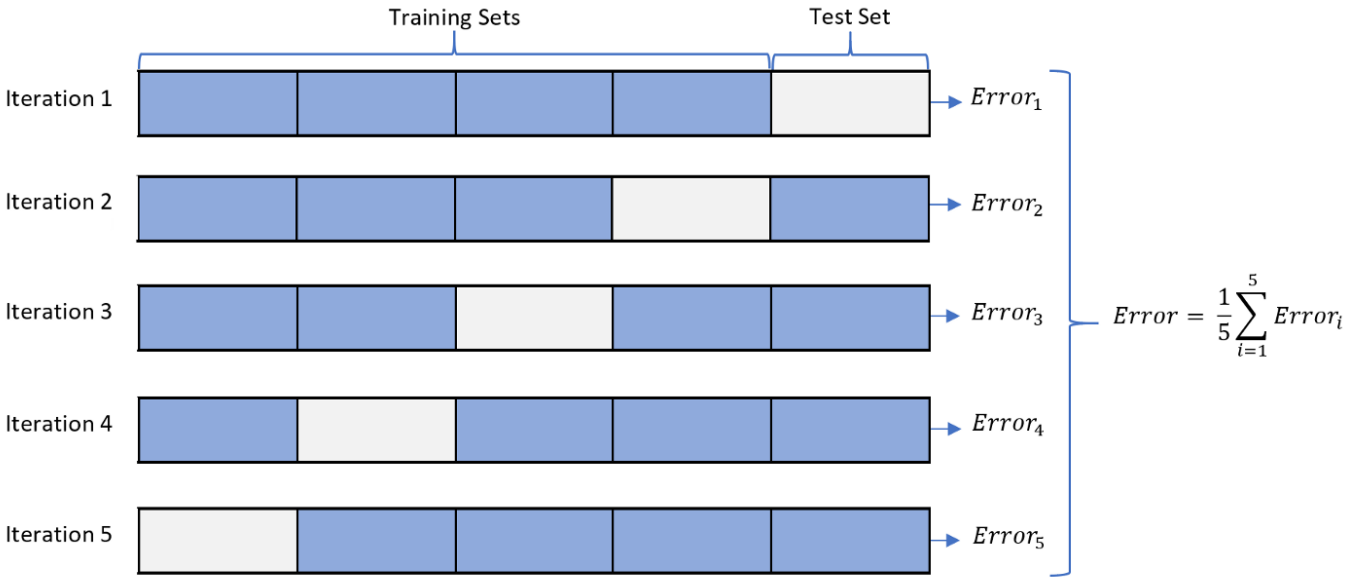
The process of cross-validation is as the following:

1. Define k or the number of folds
2. Randomly shuffle the data into K equally-sized blocks (folds)
3. For each i in fold 1 to k train the data using all the folds except for fold i and test on the fold i.
4. Average the K validation/test error from the previous step to get an estimate of the error.

This process aims in accomplishing the following: 1- Prevent overfitting during training by avoiding training and testing on the same subset of the data points

2- Avoid information loss by using a certain subset of the data for validation only. This is important for small datasets.

Cross-validation is always good to be used for small datasets, and if used for large datasets the computational complexity will increase depending on the number of folds.



Q17: You are building a binary classifier and you found that the data is imbalanced, what should you do to handle this situation?

Answer: If there is a data imbalance there are several measures we can take to train a fairer binary classifier:

1. Pre-Processing:

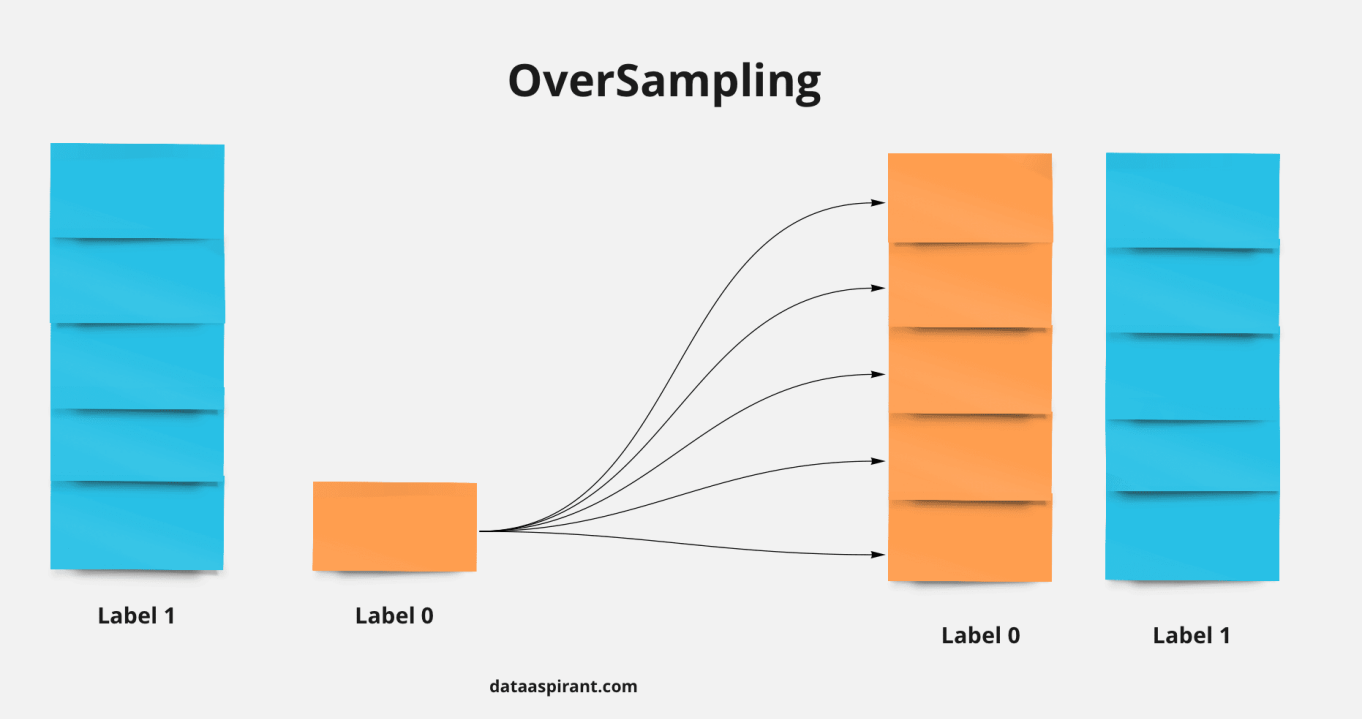
- Check whether you can get more data or not.
- Use sampling techniques (Up Sample minority class, Downsample majority class, can take the hybrid approach as well). We can also use data augmentation to add more data points for the minority class but with little deviations/changes leading to new data points which are similar to the ones they are derived from. The most common/popular technique is SMOTE (Synthetic Minority Oversampling technique)
- Suppression: Though not recommended, we can drop off some features directly responsible for the imbalance.
- Learning Fair Representation: Projecting the training examples to a subspace or plane minimizes the data imbalance.
- Re-Weighting: We can assign some weights to each training example to reduce the imbalance in the data.

2. In-Processing:

- Regularisation: We can add score terms that measure the data imbalance in the loss function and therefore minimizing the loss function will also minimize the degree of imbalance with respect to the score chosen which also indirectly minimizes other metrics which measure the degree of data imbalance.
- Adversarial Debiasing: Here we use the adversarial notion to train the model where the discriminator tries to detect if there are signs of data imbalance in the predicted data by the generator and hence the generator learns to generate data that is less prone to imbalance.

3. Post-Processing:

- Odds-Equalization: Here we try to equalize the odds for the classes wrt the data is imbalanced for correct imbalance in the trained model. Usually, the F1 score is a good choice, if both precision and recall scores are important
- Choose appropriate performance metrics. For example, accuracy is not a correct metric to use when classes are imbalanced. Instead, use precision, recall, F1 score, and ROC curve.



Q18: You are working on a clustering problem, what are different evaluation metrics that can be used, and how to choose between them?

Answer:

Clusters are evaluated based on some similarity or dissimilarity measure such as the distance between cluster points. If the clustering algorithm separates dissimilar observations apart and similar observations together, then it has performed well. The two most popular metrics evaluation metrics for clustering algorithms are the **Silhouette coefficient** and **Dunn's Index**.

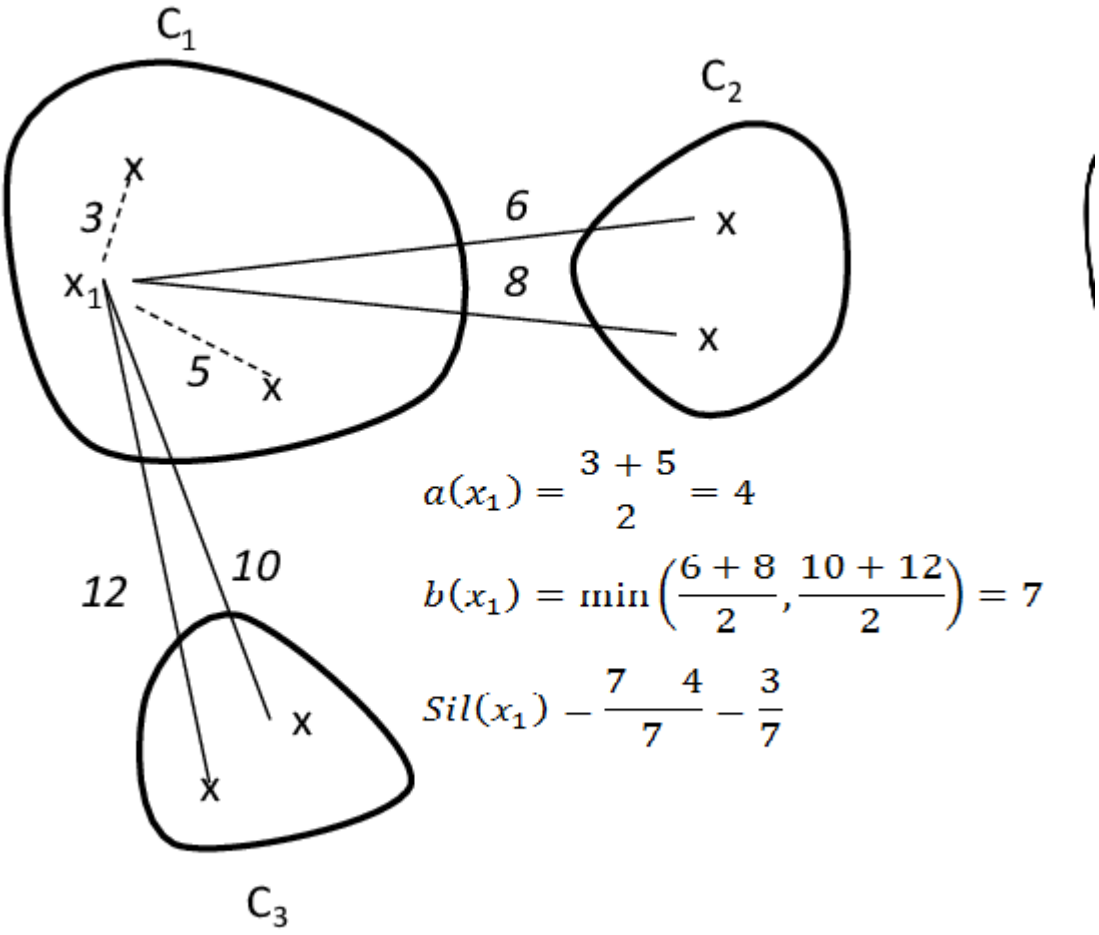
Silhouette coefficient The Silhouette Coefficient is defined for each sample and is composed of two scores: a: The mean distance between a sample and all other points in the same cluster. b: The mean distance between a sample and all other points in the next nearest cluster.

S = (b-a) / max(a,b)

The **Silhouette coefficient** for a set of samples is given as the mean of the Silhouette Coefficient for each sample. The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters. The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.

Dunn’s Index

Dunn’s Index (DI) is another metric for evaluating a clustering algorithm. Dunn’s Index is equal to the minimum inter-cluster distance divided by the maximum cluster size. Note that large inter-cluster distances (better separation) and smaller cluster sizes (more compact clusters) lead to a higher DI value. A higher DI implies better clustering. It assumes that better clustering means that clusters are compact and well-separated from other clusters.



Q19: What is the ROC curve and when should you use it?

Answer:

ROC curve, Receiver Operating Characteristic curve, is a graphical representation of the model's performance where we plot the True Positive Rate (TPR) against the False Positive Rate (FPR) for different threshold values, for hard classification, between 0 to 1 based on model output.

This ROC curve is mainly used to compare two or more models as shown in the figure below. Now, it is easy to see that a reasonable model will always give FPR less (since it's an error) than TPR so, the curve hugs the upper left corner of the square box 0 to 1 on the TPR axis and 0 to 1 on the FPR axis.

The more the AUC(area under the curve) for a model's ROC curve, the better the model in terms of prediction accuracy in terms of TPR and FPR.

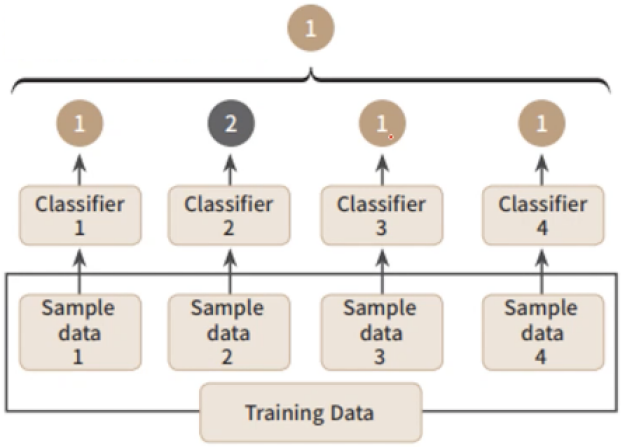
Here are some benefits of using the ROC Curve :

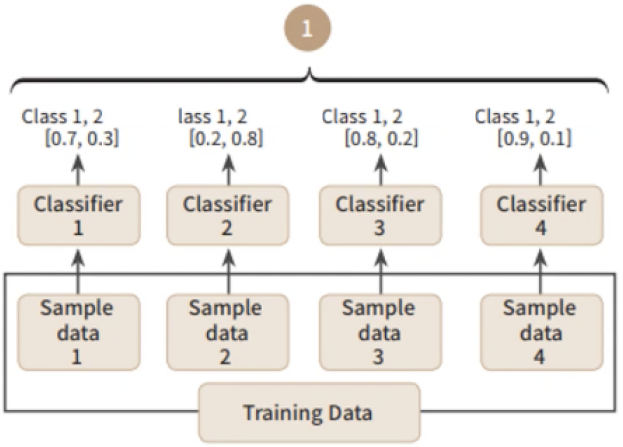
- Can help prioritize either true positives or true negatives depending on your case study (Helps you visually choose the best hyperparameters for your case)
- Can be very insightful when we have unbalanced datasets
- Can be used to compare different ML models by calculating the area under the ROC curve (AUC)

Q20: What is the difference between hard and soft voting classifiers in the context of ensemble learners?

Answer:

- Hard Voting: We take into account the class predictions for each classifier and then classify an input based on the maximum votes to a particular class.
- Soft Voting: We take into account the probability predictions for each class by each classifier and then classify an input to the class with maximum probability based on the average probability (averaged over the classifier's probabilities) for that class.





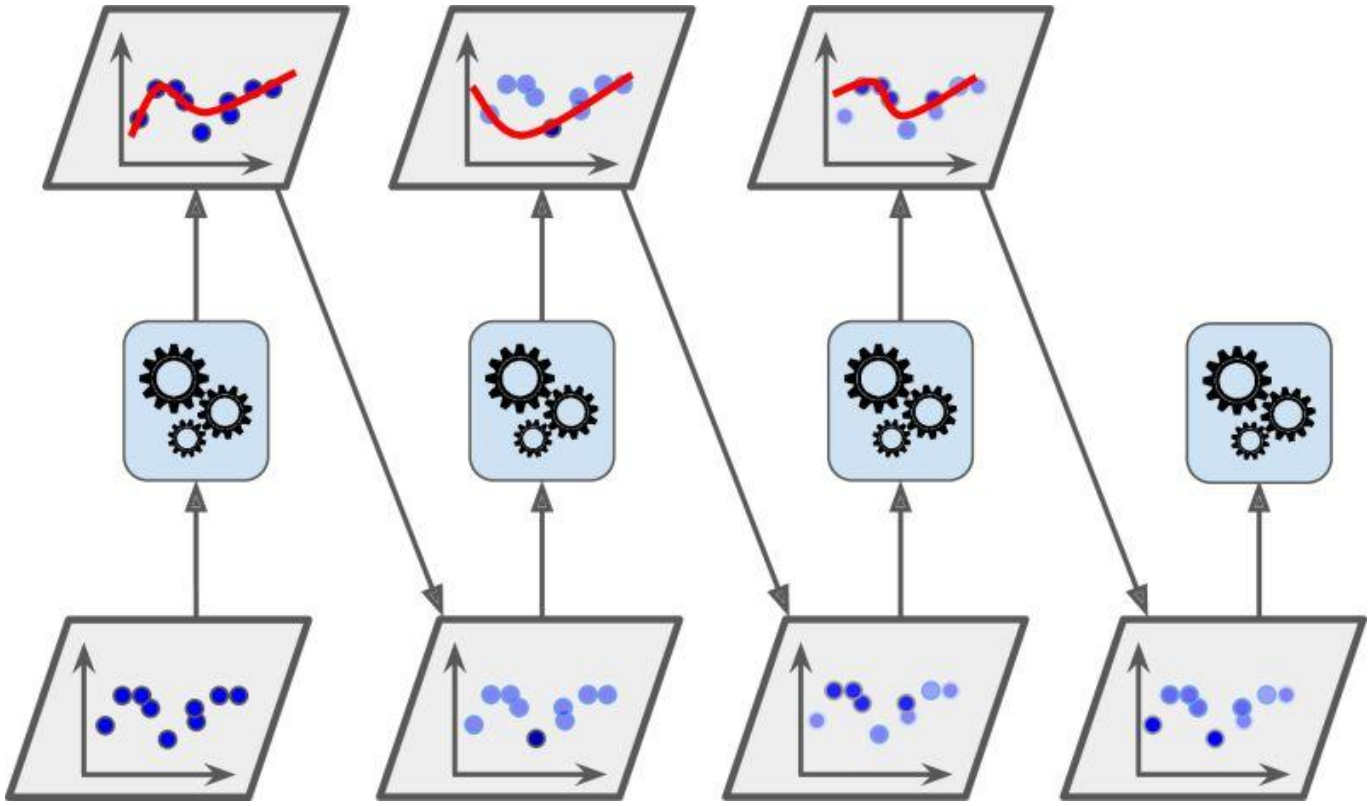
Q21: What is boosting in the context of ensemble learners discuss two famous boosting methods

Answer:

Boosting refers to any Ensemble method that can combine several weak learners into a strong learner. The general idea of most boosting methods is to train predictors sequentially, each trying to correct its predecessor.

There are many boosting methods available, but by far the most popular are:

- Adaptive Boosting: One way for a new predictor to correct its predecessor is to pay a bit more attention to the training instances that the predecessor under-fitted. This results in new predictors focusing more and more on the hard cases.
- Gradient Boosting: Another very popular Boosting algorithm is Gradient Boosting. Just like AdaBoost, Gradient Boosting works by sequentially adding predictors to an ensemble, each one correcting its predecessor. However, instead of tweaking the instance weights at every iteration as AdaBoost does, this method tries to fit the new predictor to the residual errors made by the previous predictor.



Q22: How can you evaluate the performance of a dimensionality reduction algorithm on your dataset?

Answer:

Intuitively, a dimensionality reduction algorithm performs well if it eliminates a lot of dimensions from the dataset without losing too much information. One way to measure this is to apply the

reverse transformation and measure the reconstruction error. However, not all dimensionality reduction algorithms provide a reverse transformation.

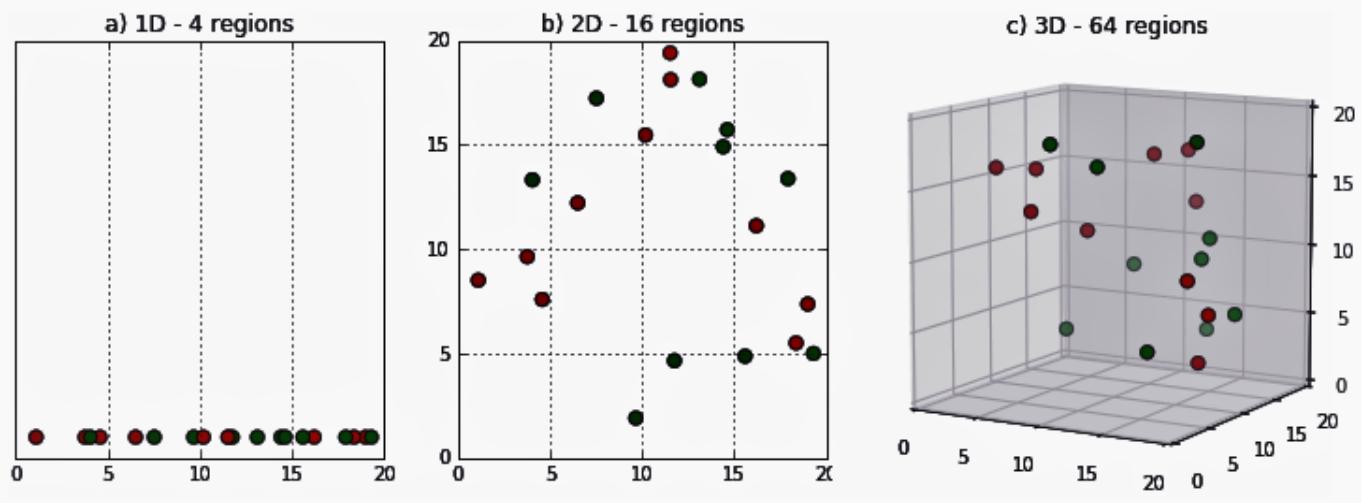
Alternatively, if you are using dimensionality reduction as a preprocessing step before another Machine Learning algorithm (e.g., a Random Forest classifier), then you can simply measure the performance of that second algorithm; if dimensionality reduction did not lose too much information, then the algorithm should perform just as well as when using the original dataset.

Q23: Define the curse of dimensionality and how to solve it.

Answer: Curse of dimensionality represents the situation when the amount of data is too few to be represented in a high-dimensional space, as it will be highly scattered in that high-dimensional space and it becomes more probable that we overfit this data. If we increase the number of features, we are implicitly increasing model complexity and if we increase model complexity we need more data.

Possible solutions are: Remove irrelevant features not discriminating classes correlated or features not resulting in much improvement, we can use:

- Feature selection(select the most important ones).
- Feature extraction(transform current feature dimensionality into a lower dimension preserving the most possible amount of information like PCA).



Q24: In what cases would you use vanilla PCA, Incremental PCA, Randomized PCA, or Kernel PCA?

Answer:

Regular PCA is the default, but it works only if the dataset fits in memory. Incremental PCA is useful for large datasets that don't fit in memory, but it is slower than regular PCA, so if the dataset fits in memory you should prefer regular PCA. Incremental PCA is also useful for online tasks when you need to apply PCA on the fly, every time a new instance arrives. Randomized PCA is useful when you want to considerably reduce dimensionality and the dataset fits in memory; in this case, it is much faster than regular PCA. Finally, Kernel PCA is useful for nonlinear datasets.

Q25: Discuss two clustering algorithms that can scale to large datasets

Answer:

Minibatch Kmeans: Instead of using the full dataset at each iteration, the algorithm is capable of using mini-batches, moving the centroids just slightly at each iteration. This speeds up the algorithm typically by a factor of 3 or 4 and makes it possible to cluster huge datasets that do not fit in memory. Scikit-Learn implements this algorithm in the MiniBatchKMeans class.

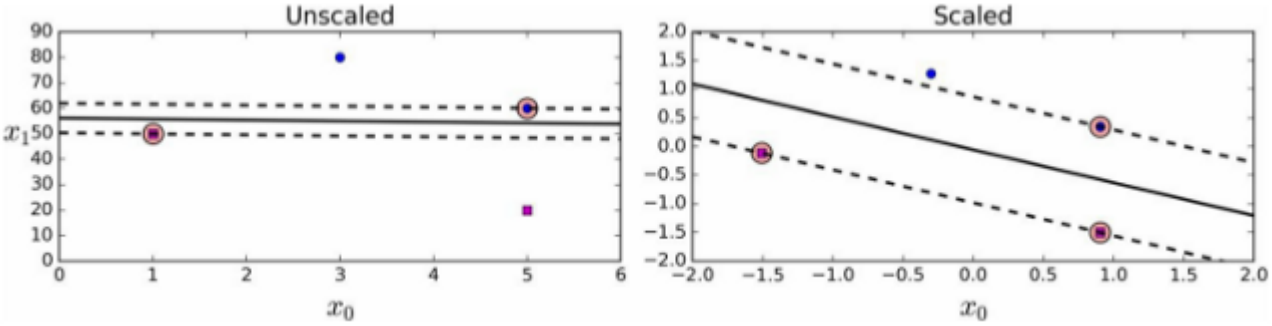
Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is a clustering algorithm that can cluster large datasets by first generating a small and compact summary of the large dataset that retains as much information as possible. This smaller summary is then clustered instead of clustering the larger dataset.

Q26: Do you need to scale your data if you will be using the SVM classifier and discus your answer

Answer: Yes, feature scaling is required for SVM and all margin-based classifiers since the optimal hyperplane (the decision boundary) is dependent on the scale of the input features. In other words, the distance between two observations will differ for scaled and non-scaled cases, leading to different models being generated.

This can be seen in the figure below, when the features have different scales, we can see that the decision boundary and the support vectors are only classifying the X1 features without taking into consideration the X0 feature, however after scaling the data to the same scale the decision boundaries and support vectors are looking much better and the model is taking into account both features.

To scale the data, normalization and standardization are the most popular approaches.



Q27: What are Loss Functions and Cost Functions? Explain the key Difference Between them.

Answer: The loss function is the measure of performance of the model on a single training example, whereas the cost function is the average loss function over all training examples or across the batch in the case of mini-batch gradient descent.

Some examples of loss functions are Mean Squared Error, Binary Cross Entropy, etc.

Whereas, the cost function is the average of the above loss functions over training examples.

Q28: What is the importance of batch in machine learning and explain some batch depend gradient descent algorithm?

Answer: In the memory, the dataset can load either completely at once or in a form of a set. If we have a huge size of the dataset, then loading the whole data into memory will reduce the training speed, hence batch term introduce.

Example: image data contains 1,00,000 images, we can load this into 3125 batches where 1 batch = 32 images. So instead of loading the whole 1,00,000 images in memory, we can load 32 images 3125 times which requires less memory.

In summary, a batch is important in two ways: (1) Efficient memory consumption. (2) Improve training speed.

There are 3 types of gradient descent algorithms based on batch size: (1) Stochastic gradient descent (2) Batch gradient descent (3) Mini Batch gradient descent

If the whole data is in a single batch, it is called batch gradient descent. If the single data points are equal to one batch i.e. number of batches = number of data instances, it is called stochastic gradient descent. If the number of batches is less than the number of data points or greater than 1, it is known as mini-batch gradient descent.

Q29: What are the different methods to split a tree in a decision tree algorithm?

Answer:

Decision trees can be of two types regression and classification. For classification, classification accuracy created a lot of instability. So the following loss functions are used:

- Gini's Index Gini impurity is used to predict the likelihood of a randomly chosen example being incorrectly classified by a particular node. It’s referred to as an “impurity” measure because it demonstrates how the model departs from a simple division.
- Cross-Entropy or Information Gain Information gain refers to the process of identifying the most important features/attributes that convey the most information about a class. The entropy principle is followed with the goal of reducing entropy from the root node to the leaf nodes. Information gain is the difference in entropy before and after splitting, which describes the impurity of in-class items.

For regression, the good old mean squared error serves as a good loss function which is minimized by splits of the input features and predicting the mean value of the target feature on the subspaces resulting from the split. But finding the split that results in the minimum possible residual sum of squares is computationally infeasible, so a greedy top-down approach is taken i.e. the splits are made at a level from top to down which results in maximum reduction of RSS. We continue this until some max depth or number of leaves is attained.

Q30: Why boosting is a more stable algorithm as compared to other ensemble algorithms?

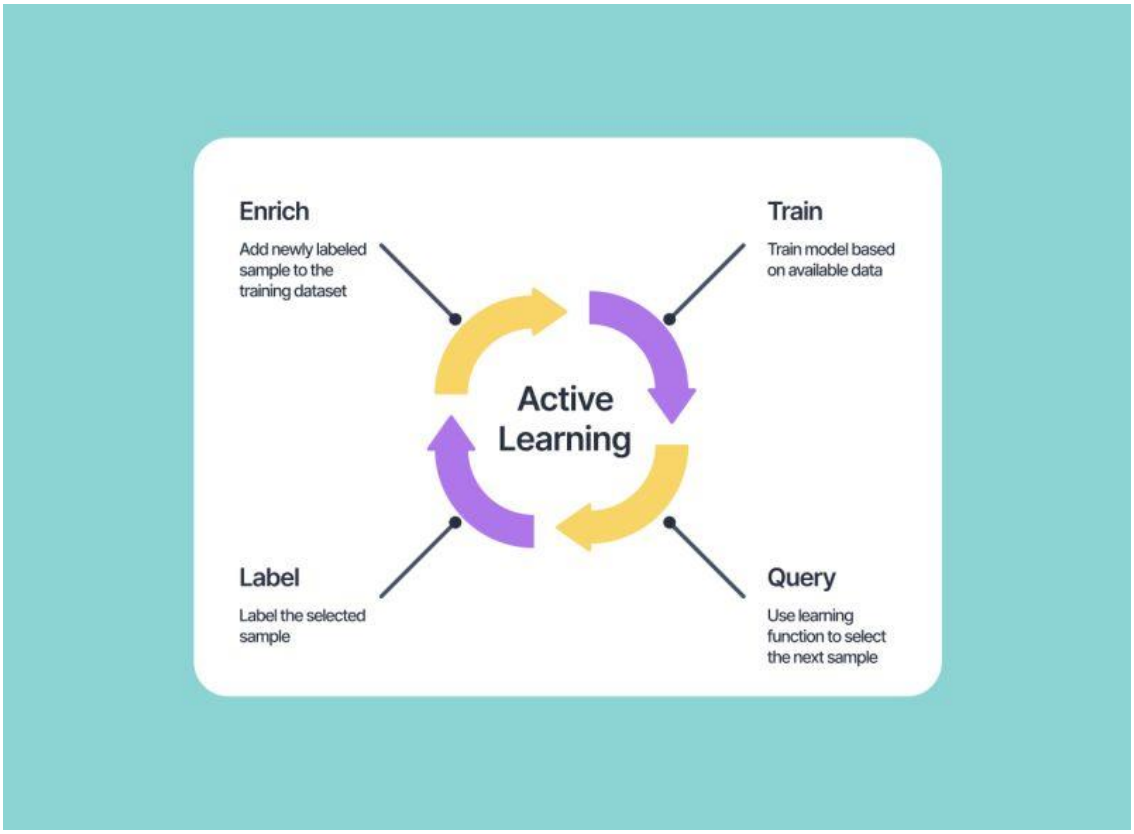
Answer:

Since Boosting algorithms focus on errors found in previous iterations until they become obsolete. Whereas in bagging there is no corrective loop. That's why boosting is a more stable algorithm compared to other ensemble algorithms.

Q31: What is active learning and discuss one strategy of it?

Answer: Active learning is a special case of machine learning in which a learning algorithm can interactively query a user (or some other information source) to label new data points with the desired outputs. In statistics literature, it is sometimes referred to as optimal experimental design.

1. Stream-based sampling In stream-based selective sampling, unlabelled data is continuously fed to an active learning system, where the learner decides whether to send the same to a human oracle or not based on a predefined learning strategy. This method is apt in scenarios where the model is in production and the data sources/distributions vary over time.
2. Pool-based sampling In this case, the data samples are chosen from a pool of unlabelled data based on the informative value scores and sent for manual labeling. Unlike stream-based sampling, oftentimes, the entire unlabelled dataset is scrutinized for the selection of the best instances.

- 

Q32: What are the different approaches to implementing recommendation systems?

Answer:

1. **Content-Based Filtering**: Content-Based Filtering depends on similarities of items and users' past activities on the website to recommend any product or service.

This filter helps in avoiding a cold start for any new products as it doesn't rely on other users' feedback, it can recommend products based on similarity factors. However, content-based filtering needs a lot of domain knowledge so that the recommendations made are 100 percent accurate.

2. **Collaborative-Based Filtering**: The primary job of a collaborative filtering system is to overcome the shortcomings of content-based filtering.

So, instead of focusing on just one user, the collaborative filtering system focuses on all the users and clusters them according to their interests.

Basically, it recommends a product 'x' to user 'a' based on the interest of user 'b'; users 'a' and 'b' must have had similar interests in the past, which is why they are clustered together.

The domain knowledge that is required for collaborative filtering is less, recommendations made are more accurate and it can adapt to the changing tastes of users over time. However, collaborative filtering faces the problem of a cold start as it heavily relies on feedback or activity from other users.

3. **Hybrid filtering**: A mixture of content and collaborative methods. Uses descriptors and interactions.

More modern approaches typically fall into the hybrid filtering category and tend to work in two stages:

1). A candidate generation phase where we coarsely generate candidates from a corpus of hundreds of thousands, millions, or billions of items down to a few hundred or thousand

2. A ranking phase where we re-rank the candidates into a final top-n set to be shown to the user. Some systems employ multiple candidate generation methods and rankers.

Q33: What are the evaluation metrics that can be used for multi-label classification?

Answer:

Multi-label classification is a type of classification problem where each instance can be assigned to multiple classes or labels simultaneously.

The evaluation metrics for multi-label classification are designed to measure the performance of a multi-label classifier in predicting the correct set of labels for each instance. Some commonly used evaluation metrics for multi-label classification are:

1. Hamming Loss: Hamming Loss is the fraction of labels that are incorrectly predicted. It is defined as the average number of labels that are predicted incorrectly per instance.
2. Accuracy: Accuracy is the fraction of instances that are correctly predicted. In multi-label classification, accuracy is calculated as the percentage of instances for which all labels are predicted correctly.
3. Precision, Recall, F1-Score: These metrics can be applied to each label separately, treating the classification of each label as a separate binary classification problem. Precision measures the proportion of predicted positive labels that are correct, recall measures the proportion of actual positive labels that are correctly predicted, and F1-score is the harmonic mean of precision and recall.
4. Macro-F1, Micro-F1: Macro-F1 and Micro-F1 are two types of F1-score metrics that take into account the label imbalance in the dataset. Macro-F1 calculates the F1-score for each label and then averages them, while Micro-F1 calculates the overall F1-score by aggregating the true positive, false positive, and false negative counts across all labels.

There are other metrics that can be used such as:

- Precision at k (P@k)
- Average precision at k (AP@k)
- Mean average precision at k (MAP@k)

Q34: What is the difference between concept and data drift and how to overcome each of them?

Answer:

Concept drift and data drift are two different types of problems that can occur in machine learning systems.

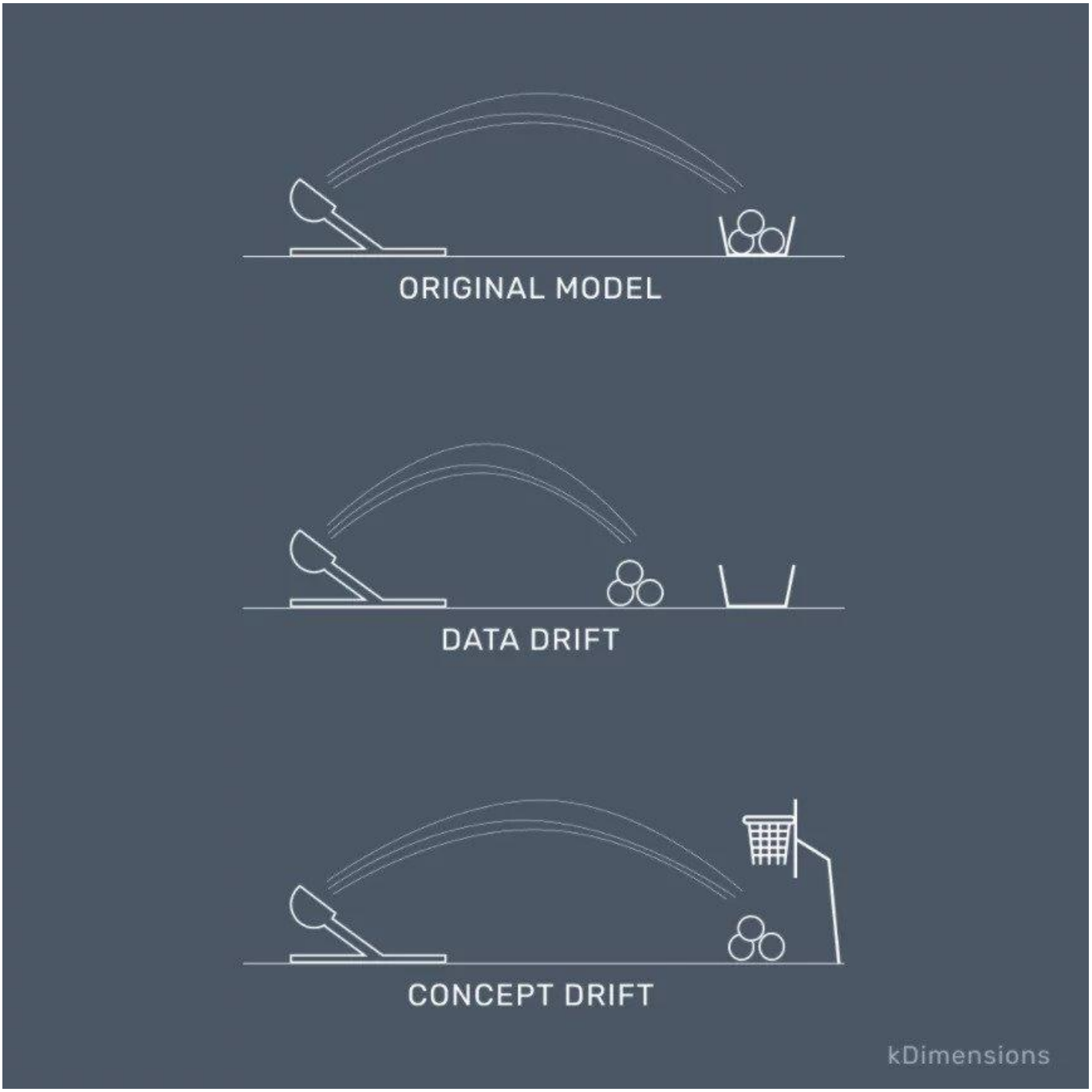
Concept drift refers to changes in the underlying relationships between the input data and the target variable over time. This means that the distribution of the data that the model was trained on no longer matches the distribution of the data it is being tested on. For example, a spam filter model that was trained on emails from several years ago may not be as effective at identifying spam emails from today because the language and tactics used in spam emails may have changed.

Data drift, on the other hand, refers to changes in the input data itself over time. This means that the values of the input feature that the model was trained on no longer match the values of the input features in the data it is being tested on. For example, a model that was trained on data from a particular geographical region may not be as effective at predicting outcomes for data from a different region.

To overcome concept drift, one approach is to use online learning methods that allow the model to adapt to new data as it arrives. This involves continually training the model on the most recent data while using historical data to maintain context. Another approach is to periodically retrain the model using a representative sample of the most recent data.

To overcome data drift, one approach is to monitor the input data for changes and retrain the model when significant changes are detected. This may involve setting up a monitoring system that alerts the user when the data distribution changes beyond a certain threshold.

Another approach is to preprocess the input data to remove or mitigate the effects of the features changing over time so that the model can continue learning from the remaining features.



Q35: Can you explain the ARIMA model and its components?

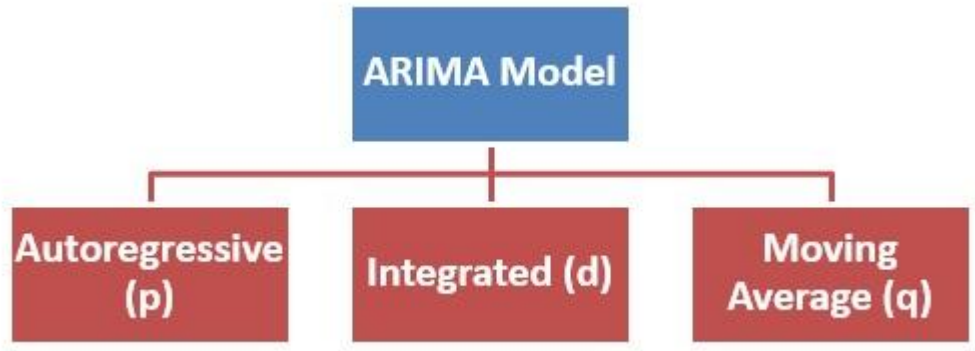
Answer: The ARIMA model, which stands for Autoregressive Integrated Moving Average, is a widely used time series forecasting model. It combines three key components: Autoregression (AR), Differencing (I), and Moving Average (MA).

- Autoregression (AR): The autoregressive component captures the relationship between an observation in a time series and a certain number of lagged observations. It assumes that the value at a given time depends linearly on its own previous values. The "p" parameter in ARIMA(p, d, q) represents the order of autoregressive terms. For example, ARIMA(1, 0, 0) refers to a model with one autoregressive term.
- Differencing (I): Differencing is used to make a time series stationary by removing trends or seasonality. It calculates the difference between consecutive observations to eliminate any non-stationary behavior. The "d" parameter in ARIMA(p, d, q) represents the order of differencing. For instance, ARIMA(0, 1, 0) indicates that differencing is applied once.
- Moving Average (MA): The moving average component takes into account the dependency between an observation and a residual error from a moving average model applied to lagged observations. It assumes that the value at a given time depends linearly on the error terms from previous time steps. The "q" parameter in ARIMA(p, d, q) represents the order of the moving average terms. For example, ARIMA(0, 0, 1) signifies a model with one moving average term.

By combining these three components, the ARIMA model can capture both autoregressive patterns, temporal dependencies, and stationary behavior in a time series. The parameters p, d, and q are typically determined through techniques like the Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC).

It's worth noting that there are variations of the ARIMA model, such as SARIMA (Seasonal ARIMA), which incorporates additional seasonal components for modeling seasonal patterns in the data.

ARIMA models are widely used in forecasting applications, but they do make certain assumptions about the underlying data, such as linearity and stationarity. It's important to validate these assumptions and adjust the model accordingly if they are not met.



Q36: What are the assumptions made by the ARIMA model?

Answer:

The ARIMA model makes several assumptions about the underlying time series data. These assumptions are important to ensure the validity and accuracy of the model's results. Here are the key assumptions:

Stationarity: The ARIMA model assumes that the time series is stationary. Stationarity means that the statistical properties of the data, such as the mean and variance, remain constant over

time. This assumption is crucial for the autoregressive and moving average components to hold. If the time series is non-stationary, differencing (the "I" component) is applied to transform it into a stationary series.

Linearity: The ARIMA model assumes that the relationship between the observations and the lagged values is linear. It assumes that the future values of the time series can be modeled as a linear combination of past values and error terms.

No Autocorrelation in Residuals: The ARIMA model assumes that the residuals (the differences between the predicted values and the actual values) do not exhibit any autocorrelation. In other words, the errors are not correlated with each other.

Normally Distributed Residuals: The ARIMA model assumes that the residuals follow a normal distribution with a mean of zero. This assumption is necessary for statistical inference, parameter estimation, and hypothesis testing.

It's important to note that while these assumptions are commonly made in ARIMA modeling, they may not always hold in real-world scenarios. It's essential to assess the data and, if needed, apply transformations or consider alternative models that relax some of these assumptions. Additionally, diagnostics tools, such as residual analysis and statistical tests, can help evaluate the adequacy of the assumptions and the model's fit to the data.

资料引用自：https://github.com/youssefHosni/Data-Science-Interview-Questions-Answers/blob/main/Machine%20Learning%20Interview%20Questions%20%26%20Answers%20for%20Data%20Scientists.md