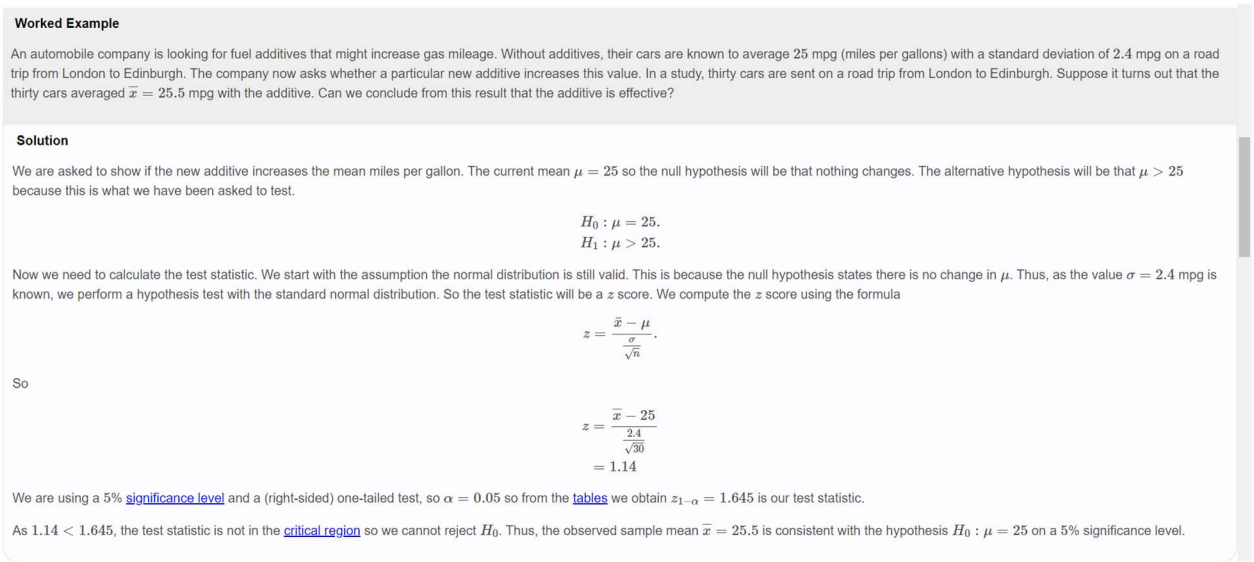
1. Define hypothesis testing and Normal Distribution.

Hypothesis testing is a form of statistical inference that uses data from a sample to draw conclusions about a population parameter or a population probability distribution. First, a tentative assumption is made about the parameter or distribution. This assumption is called the null hypothesis and is denoted by H0. An alternative hypothesis (denoted Ha), which is the opposite of what is stated in the null hypothesis, is then defined. The hypothesis-testing procedure involves using sample data to determine whether or not H0 can be rejected. If H0 is rejected, the statistical conclusion is that the alternative hypothesis Ha is true.



Worked Example

An automobile company is looking for fuel additives that might increase gas mileage. Without additives, their cars are known to average 25 mpg (miles per gallons) with a standard deviation of 2.4 mpg on a road trip from London to Edinburgh. The company now asks whether a particular new additive increases this value. In a study, thirty cars are sent on a road trip from London to Edinburgh. Suppose it turns out that the thirty cars averaged $\overline{x} = 25.5$ mpg with the additive. Can we conclude from this result that the additive is effective?

Solution

We are asked to show if the new additive increases the mean miles per gallon. The current mean $\mu = 25$ so the null hypothesis will be that nothing changes. The alternative hypothesis will be that $\mu > 25$ because this is what we have been asked to test.

$$H_0 : \mu = 25.$$
$$H_1 : \mu > 25.$$

Now we need to calculate the test statistic. We start with the assumption the normal distribution is still valid. This is because the null hypothesis states there is no change in μ. Thus, as the value $\sigma = 2.4$ mpg is known, we perform a hypothesis test with the standard normal distribution. So the test statistic will be a z score. We compute the z score using the formula

$$z = \frac{\overline{x} - \mu}{\frac{\sigma}{\sqrt{n}}}.$$

So

$$z = \frac{\overline{x} - 25}{\frac{2.4}{\sqrt{30}}}$$
$$= 1.14$$

We are using a 5% significance level and a (right-sided) one-tailed test, so $\alpha = 0.05$ so from the tables we obtain $z_{1-\alpha} = 1.645$ is our test statistic.

As $1.14 < 1.645$, the test statistic is not in the critical region so we cannot reject H_0. Thus, the observed sample mean $\overline{x} = 25.5$ is consistent with the hypothesis $H_0 : \mu = 25$ on a 5% significance level.

2. Define possibility of conversion for Regression into Classification and vice versa.

Some regression models are already classification models - e.g. logistic regression. One could set the cut point at any particular level to get a classification. Usually one would choose a 50-50 split, but there may be reasons not to (the cost of the two types of classification error might be different).

Regression trees turn into classification trees if the dependent variable changes. In general, it is not a good idea to turn a continuous dependent variable (as for regression trees) into a categorical one - it loses information. But there might be times when it is necessary (e.g. to make certain kinds of decisions).

Similarly, if you categorize the dependent variable, a linear regression is inappropriate and a logistic regression model is better.

Regression models can be very sensitive to outliers. Also, a practical challenge is, a predicted value might be far off from the real value in extreme ranges, however it still may fall in the correct side of the data distribution with respect to the mean. For e.g. you have a device that measures heart rate from your finger tips images (just cooking up an example here), it'll be lot easier from data science standpoint to first try to predict whether the pulse rate is normal or high or below normal.

The definitions of what is “normal” is very clear from medical standpoint. However, you may wish to redefine your response variable segments, first using a clustering technique.

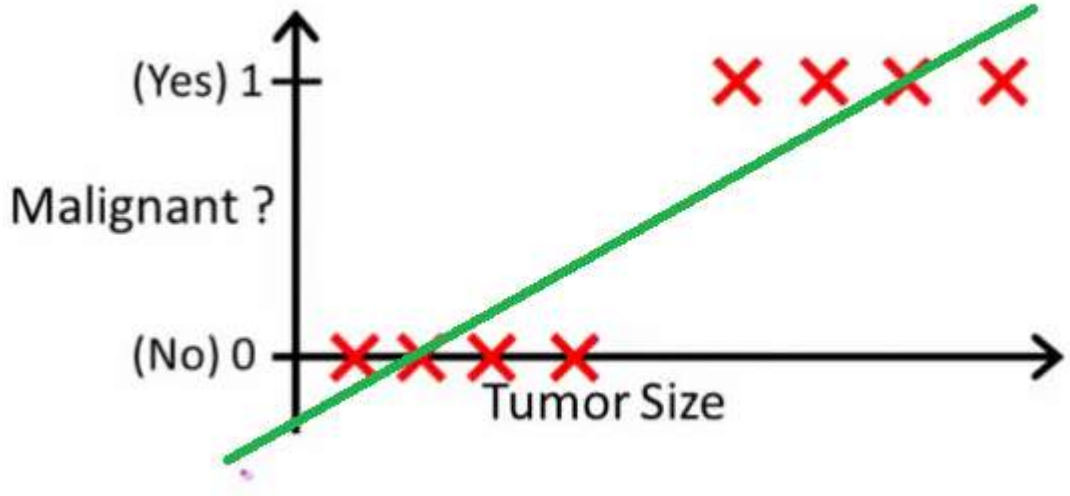
Regression models can be very sensitive to outliers. Also, a practical challenge is, a predicted value might be far off from the real value in extreme ranges, however it still may fall in the correct side of the data distribution with respect to the mean. For e.g. you have a device that measures heart rate from your finger tips images (just cooking up an example here), it’ll be lot easier from data science standpoint to first try to predict whether the pulse rate is normal or high or below normal.

The definitions of what is “normal” is very clear from medical standpoint. However, you may wish to redefine your response variable segments, first using a clustering technique.

Before we do this, it is important to clarify the distinction between regression and classification models. Regression models predict a continuous variable, such as rainfall amount or sunlight intensity. They can also predict probabilities, such as the probability that an image contains a cat. A probability-predicting regression model can be used as part of a classifier by imposing a decision rule - for example, if the probability is 50% or more, decide it's a cat.

Logistic regression predicts probabilities, and is therefore a regression algorithm. However, it is commonly described as a classification method in the machine learning literature, because it can be (and is often) used to make classifiers. There are also "true" classification algorithms, such as SVM, which only predict an outcome and do not provide a probability. We won't discuss this kind of algorithm here.

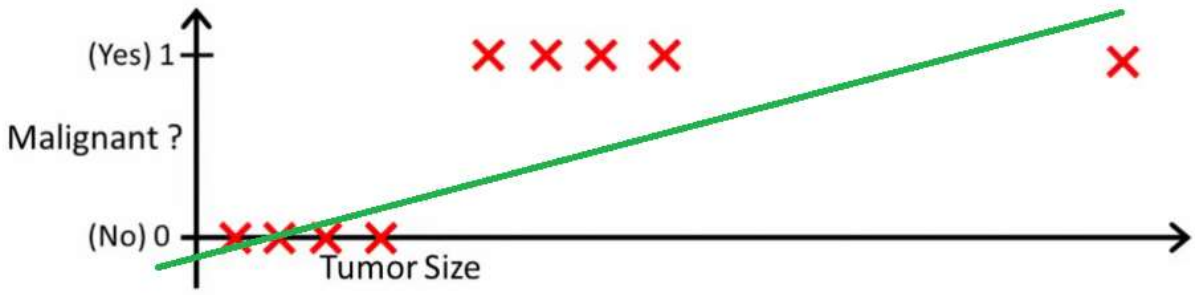
with linear regression you fit a polynomial through the data - say, like on the example below we're fitting a straight line through *{tumor size, tumor type}* sample set:



Above, malignant tumors get $1$1 and non-malignant ones get $0$0, and the green line is our hypothesis $h(x)$h(x). To make predictions we may say that for any given tumor size xx, if $h(x)$h(x) gets bigger than $0.5$0.5 we predict malignant tumor, otherwise we predict benign.

Looks like this way we could correctly predict every single training set sample, but now let's change the task a bit.

Intuitively it's clear that all tumors larger certain threshold are malignant. So let's add another sample with a huge tumor size, and run linear regression again:



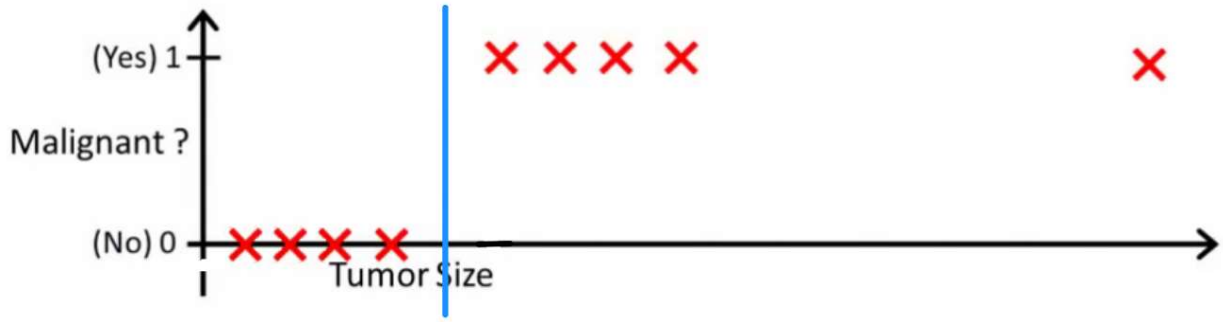
Now our $h(x)>0.5\rightarrow\text{malignant}$h(x)>0.5→malignant doesn't work anymore. To keep making correct predictions we need to change it to $h(x)>0.2$h(x)>0.2 or something - but that not how the algorithm should work.

We cannot change the hypothesis each time a new sample arrives. Instead, we should learn it off the training set data, and then (using the hypothesis we've learned) make correct predictions for the data we haven't seen before.

Hope this explains why linear regression is not the best fit for classification problems! Also, you might want to watch **VI. Logistic Regression. Classification** video on ml-class.org which explains the idea in more detail.

EDIT

probabilityislogic asked what a good classifier would do. In this particular example you would probably use logistic regression which might learn a hypothesis like this (I'm just making this up):



Note that both linear regression and logistic regression give you a straight line (or a higher order polynomial) but those lines have different meaning:

- $h(x)$ for linear regression interpolates, or extrapolates, the output and predicts the value for x we haven't seen. It's simply like plugging a new x and getting a raw number, and is more suitable for tasks like predicting, say car price based on *{car size, car age}* etc.
- $h(x)$ for logistic regression tells you the **probability** that x belongs to the "positive" class. This is why it is called a regression algorithm - it estimates a continuous quantity, the probability. However, if you set a threshold on the probability, such as $h(x)>0.5$, you obtain a classifier, and in many cases this is what is done with the output from a logistic regression model. This is equivalent to putting a line on the plot: all points sitting above the classifier line belong to one class while the points below belong to the other class.

So, the bottom line is that in classification scenario we use a *completely different* reasoning and a *completely different* algorithm than in regression scenario.

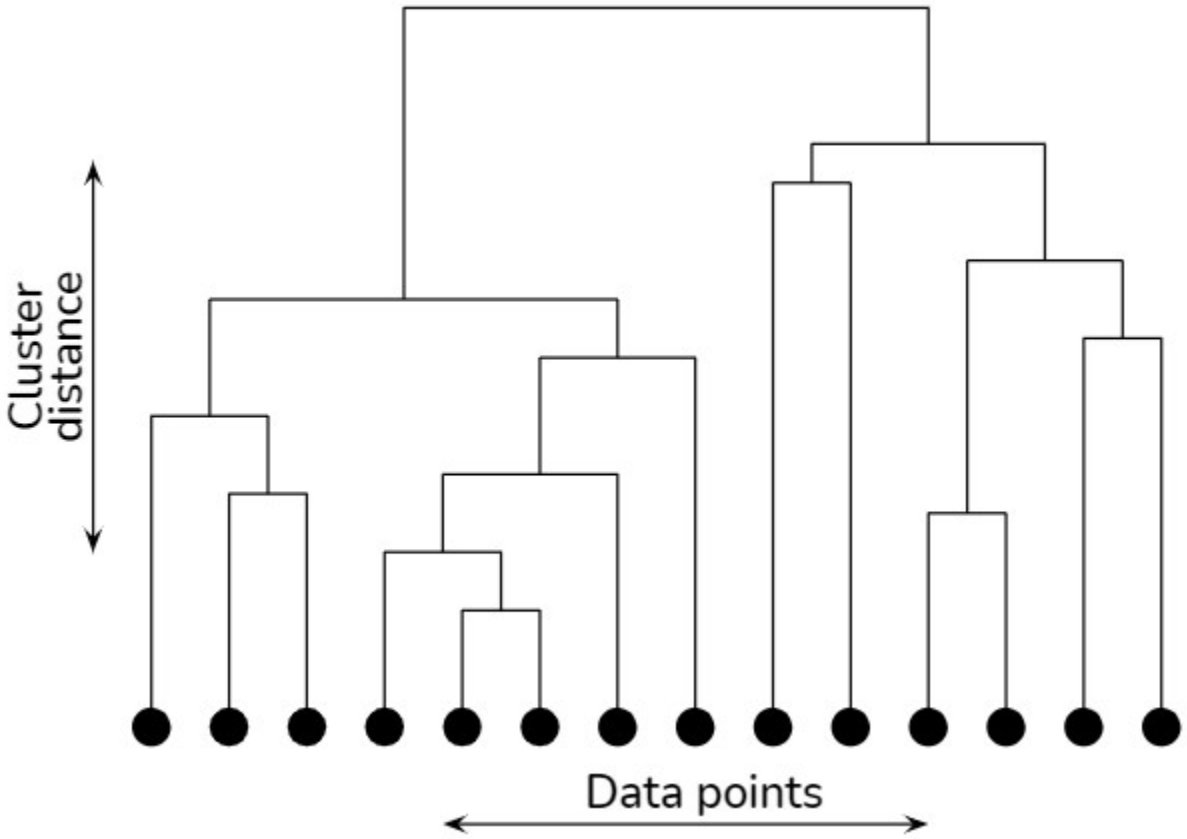
3. Differentiate KNN and K-means Clustering.

	k-NN	k-Means
Type	Supervised	Unsupervised
Meaning of k	Number of closest neighbors to look at	Number of centroids
Calculation of prediction error	Yes	No
Optimization done using	Cross validation, and confusion matrix	Elbow method, silhoutte method
Convergence	When all observations classified at the desired accuracy	When cluster memberships don't change anymore
Complexity	Train: O(d) Test: O(nd) Where: d: Dimensions/features n: Number of observations	O(nkId) Where: n: Number of points k: Number of clusters I: Number of iterations d: Number of attributes

4. Define dendrogram in Hierarchical Clustering Algorithm.

A *dendrogram* is a diagram that shows the hierarchical relationship between objects. It is most commonly created as an output from *hierarchical clustering.* The main use of a dendrogram is to work out the best way to allocate objects to clusters.

The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs. In the dendrogram plot, the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.



5. List out different algorithms to solve a problem in Reinforcement Learning.

There are two important learning models in reinforcement learning:

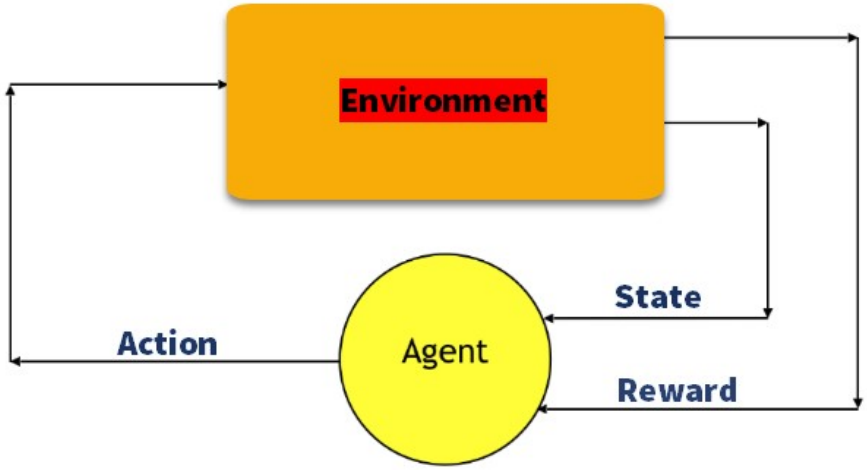
- Markov Decision Process
- Q learning

Markov Decision Process

The following parameters are used to get a solution:

- Set of actions- A
- Set of states -S
- Reward- R
- Policy- n
- Value- V

The mathematical approach for mapping a solution in reinforcement Learning is recon as a Markov Decision Process or (MDP).

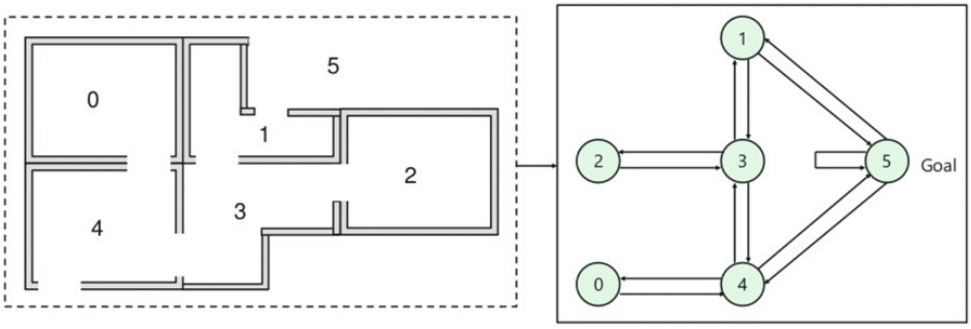


Q-Learning

Q learning is a value-based method of supplying information to inform which action an agent should take.

Let's understand this method by the following example:

- There are five rooms in a building which are connected by doors.
- Each room is numbered 0 to 4
- The outside of the building can be one big outside area (5)
- Doors number 1 and 4 lead into the building from room 5



Next, you need to associate a reward value to each door:

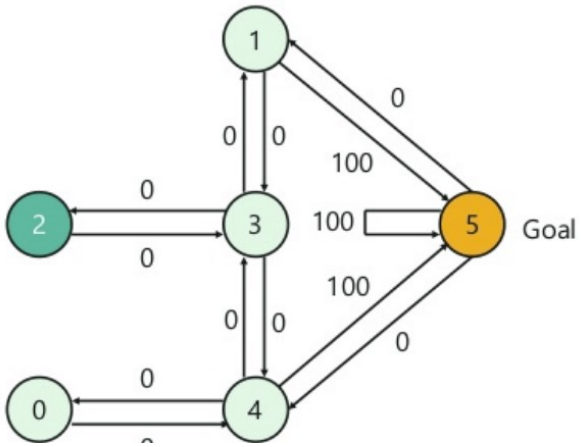
- Doors which lead directly to the goal have a reward of 100
- Doors which is not directly connected to the target room gives zero reward
- As doors are two-way, and two arrows are assigned for each room
- Every arrow in the above image contains an instant reward value

Explanation:

In this image, you can view that room represents a state

Agent's movement from one room to another represents an action

In the below-given image, a state is described as a node, while the arrows show the action.



For example, an agent traverse from room number 2 to 5

- Initial state = state 2
- State 2-> state 3
- State 3 -> state (2,1,4)
- State 4-> state (0,5,3)
- State 1-> state (5,3)
- State 0-> state 4

6. Differentiate Root Mean Squared Error (RMSE) with Mean Squared Error (MSE) for Linear Regression?

The Mean Squared Error (MSE) is a measure of how close a fitted line is to data points. For every data point, you take the distance vertically from the point to the corresponding y value on the curve fit (the error), and square the value. Then you add up all those values for all data points, and, in the case of a fit with two parameters such as a linear fit, divide by the number of points minus two. The squaring is done so negative values do not cancel positive values. The smaller the Mean Squared Error, the closer the fit is to the data. The MSE has the units squared of whatever is plotted on the vertical axis.

RMSE stands for root mean square error and MSE stands for mean square error. It is just the square root of the mean square error. That is probably the most easily interpreted statistic, since it has the same units as the quantity plotted on the vertical axis.
They are the most common measures of accuracy for a linear regression model. The formulas are below.

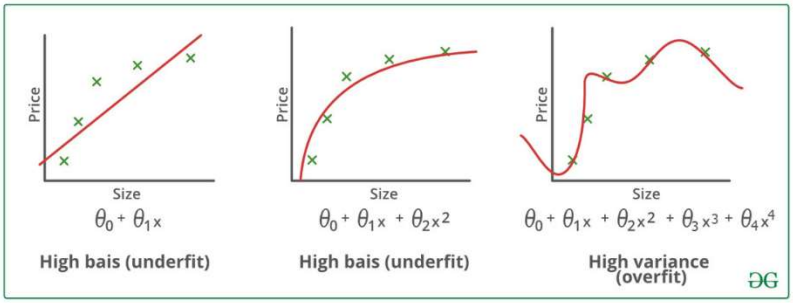
$$\text{MSE} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{test set}} \underbrace{(y_i}_{\text{predicted vaue}} - \underbrace{\hat{y}_i)^2}_{\text{actual value}}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}$$

RMSE is the square root of MSE. MSE is measured in units that are the square of the target variable, while RMSE is measured in the same units as the target variable. Due to its formulation, MSE, just like the squared loss function that it derives from, effectively penalizes larger errors more severely.

7. Describe over fitting with comparison to underfitting? Give any one method to avoid over fitting.

1. Train with more data.
2. Data augmentation.
3. Addition of noise to the input data.
4. Feature selection.
5. Cross-validation.
6. Simplify data.
7. Regularization.
8. Ensembling



Techniques to fight underfitting and overfitting		
	Underfitting	**Overfitting**
Complexity of the model	More complex model Try a more powerful model with a larger number of parameters Ensemble learning More layers / number of neurons per layer	More simple model Try a less powerful model with a fewer number of parameters Less layers / number of neurons per layer
Regularization	Less regularization Decrease regularization	More regularization Increase regularization impact Early stopping, L1 / L2 regularization, dropout
Quantity of features	A larger quantity of features Get additional features, feature engineering, polynomial features, etc.	A smaller quantity of features Remove all additional features, feature selection
Data	Data cleaning, hold-out validation or cross validation. ~~Getting more data most likely will not help~~	Data cleaning, hold-out validation or cross validation. Getting more data most likely will help (data augmentation)

8. Elaborate Apriori algorithm using confidence, support, and lift with an appropriate example.

The Apriori algorithm uses frequent itemsets to generate association rules, and it is designed to work on the databases that contain transactions. With the help of these association rule, it determines how strongly or how weakly two objects are connected. This algorithm uses a **breadth-first search** and **Hash Tree** to calculate the itemset associations efficiently. It is the iterative process for finding the frequent itemsets from the large dataset.

Consider the following dataset and we will find frequent itemsets and generate association rules for them.

TID	items
T1	I1, I2 , I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

minimum support count is 2
minimum confidence is 60%

Step-1: K=1
(I) Create a table containing support count of each item present in dataset – Called **C1(candidate set)**

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

(II) compare candidate set item's support count with minimum support count(here min_support=2 if support_count of candidate set items is less than min_support then remove those items). This gives us itemset L1.

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

Step-2: K=2

- Generate candidate set C2 using L1 (this is called join step). Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common.
- Check all subsets of an itemset are frequent or not and if not frequent remove that itemset.(Example subset of{I1, I2} are {I1}, {I2} they are frequent.Check for each itemset)
- Now find support count of these itemsets by searching in dataset.

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

(II) compare candidate (C2) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L2.

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I2,I5	2

Step-3:

- Generate candidate set C3 using L2 (join step). Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common. So here, for L2, first element should match.
  So itemset generated by joining L2 is {I1, I2, I3}{I1, I2, I5}{I1, I3, i5}{I2, I3, I4}{I2, I4, I5}{I2, I3, I5}
- Check if all subsets of these itemsets are frequent or not and if not, then remove that itemset.(Here subset of {I1, I2, I3} are {I1, I2},{I2, I3},{I1, I3} which are frequent. For {I2, I3, I4}, subset {I3, I4} is not frequent so remove it. Similarly check for every itemset)
- find support count of these remaining itemset by searching in dataset.

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

(II) Compare candidate (C3) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L3.

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

Step-4:

- Generate candidate set C4 using L3 (join step). Condition of joining L_{k-1} and L_{k-1} (K=4) is that, they should have (K-2) elements in common. So here, for L3, first 2 elements (items) should match.
- Check all subsets of these itemsets are frequent or not (Here itemset formed by joining L3 is {I1, I2, I3, I5} so its subset contains {I1, I3, I5}, which is not frequent). So no itemset in C4
- We stop here because no frequent itemsets are found further

Thus, we have discovered all the frequent item-sets. Now generation of strong association rule comes into picture. For that we need to calculate confidence of each rule.

Confidence –
A confidence of 60% means that 60% of the customers, who purchased milk and bread also bought butter.

```
Confidence(A->B)=Support_count(AUB)/Support_count(A)
```

So here, by taking an example of any frequent itemset, we will show the rule generation.
Itemset {I1, I2, I3} //from L3
SO rules can be
[I1^I2]=>[I3] //confidence = sup(I1^I2^I3)/sup(I1^I2) = 2/4*100=50%
[I1^I3]=>[I2] //confidence = sup(I1^I2^I3)/sup(I1^I3) = 2/4*100=50%
[I2^I3]=>[I1] //confidence = sup(I1^I2^I3)/sup(I2^I3) = 2/4*100=50%
[I1]=>[I2^I3] //confidence = sup(I1^I2^I3)/sup(I1) = 2/6*100=33%
[I2]=>[I1^I3] //confidence = sup(I1^I2^I3)/sup(I2) = 2/7*100=28%
[I3]=>[I1^I2] //confidence = sup(I1^I2^I3)/sup(I3) = 2/6*100=33%

So if minimum confidence is 50%, then first 3 rules can be considered as strong association rules.

9. Compare Naive Bayes with Logistic Regression to solve classification problems.

Naïve Bayes is a classification method based on Bayes' theorem that derives the probability of the given feature vector being associated with a label. Naïve Bayes has a naive assumption of conditional independence for every feature, which means that the algorithm expects the features to be independent which not always is the case.
Logistic regression is a linear classification method that learns the probability of a sample belonging to a certain class. Logistic regression tries to find the optimal decision boundary that best separates the classes.

1. Both algorithms are used for classification problems
The first similarity is the classification use case, where both Naive Bayes and Logistic regression are used to determine if a sample belongs to a certain class, for example, if an e-mail is spam or ham.

2. Algorithm's Learning mechanism
The learning mechanism is a bit different between the two models, where Naive Bayes is a generative model and Logistic regression is a discriminative model. What does this mean?

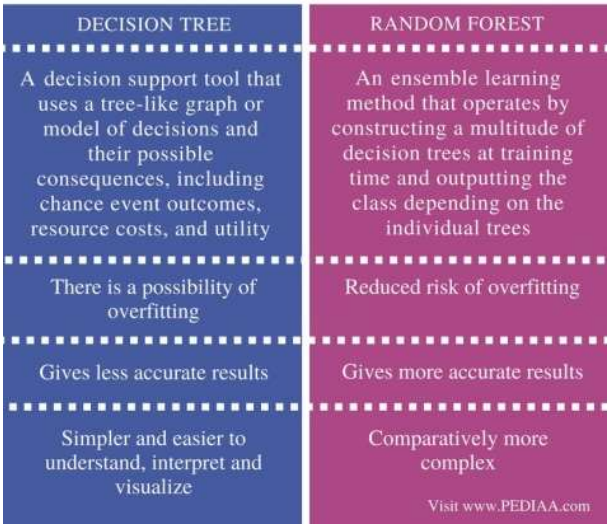
Generative model: Naive Bayes models the joint distribution of the feature X and target Y, and then predicts the posterior probability given as P(y|x)
Discriminative model: Logistic regression directly models the posterior probability of P(y|x) by learning the input to output mapping by minimising the error.

You might wonder what posterior probability is, let me give you a hint. Posterior probability can be defined as the probability of event A happening given that event B has occurred, in more layman terms this means that the previous belief can be updated when we have new information. For example, let's say we think the stock market will go up by 50% next year, this prediction can be updated when we get new information such as updated GPD numbers, interest rates etc.

3. Model assumptions
Naïve Bayes assumes all the features to be conditionally independent. So, if some of the features are in fact dependent on each other (in case of a large feature space), the prediction might be poor.
Logistic regression splits feature space linearly, and typically works reasonably well even when some of the variables are correlated.

4. Approach to be followed to improve model results
Naïve Bayes: When the training data size is small relative to the number of features, the information/data on prior probabilities help in improving the results
Logistic regression: When the training data size is small relative to the number of features, including regularisation such as Lasso and Ridge regression can help reduce overfitting and result in a more generalised model.

10. Differentiate Random Forest with Decision Tree and Explain how is it possible to perform Unsupervised Learning with Random Forest?



As stated above, many unsupervised learning methods require the inclusion of an input dissimilarity measure among the observations. Hence, if a dissimilarity matrix can be produced using Random Forest, we can successfully implement unsupervised learning. The patterns found in the process will be used to make clusters.

An artificial class label is created that distinguishes the 'observed' data from suitably generated 'synthetic' data. The observed data is the original unlabeled data, while the synthetic data is drawn from a reference distribution. Supervised learning methods, which distinguish observed data from synthetic data, yield a dissimilarity measure that can be used as input in subsequent unsupervised learning methods.

11. Can PCA be used for regression-based problem statements? If yes, then explain the scenario where we can use it.

Yes, we can use Principal Components for regression problem statements.

PCA would perform well in cases when the first few Principal Components are sufficient to capture most of the variation in the independent variables as well as the relationship with the dependent variable.

The only problem with this approach is that the new reduced set of features would be modeled by ignoring the dependent variable Y when applying a PCA and while these features may do a good overall job of explaining the variation in X, the model will perform poorly if these variables don't explain the variation in Y.

Yes, we can use Principal Components for regression problem statements.

PCA would perform well in cases when the first few Principal Components are sufficient to capture most of the variation in the independent variables as well as the relationship with the dependent variable.

The only problem with this approach is that the new reduced set of features would be modeled by ignoring the dependent variable Y when applying a PCA and while these features may do a good overall job of explaining the variation in X, the model will perform poorly if these variables don't explain the variation in Y.

12. Compare Feature Extraction and Feature Selection techniques. Explain how dimensionality can be reduced using subset selection procedure.

The main difference:- Feature Extraction transforms an arbitrary data, such as text or images, into numerical features that is understood by machine learning algorithms. Feature

Selection on the other hand is a machine learning technique applied on these (numerical) features.

Feature selection is the process of choosing precise features, from a features pool. This helps in simplification, regularization and shortening training time. This can be done with various techniques: e.g. Linear Regression, Decision Trees.

Feature extraction is the process of converting the raw data into some other data type, with which the algorithm works is called Feature Extraction. Feature extraction creates a new, smaller set of features that captures most of the useful information in the data.

The main difference between them is Feature selection keeps a subset of the original features while feature extraction creates new ones.

Feature Selection:- This module is used for feature selection/dimensionality reduction on given datasets. This is done either to improve estimators' accuracy scores or to boost their performance on very high-dimensional datasets.

Feature Extraction:- This module is used to extract features in a format supported by machine learning algorithms from the given datasets consisting of formats such as text and image.

The main difference:- Feature Extraction transforms an arbitrary data, such as text or images, into numerical features that is understood by machine learning algorithms. Feature Selection on the other hand is a machine learning technique applied on these (numerical) features.