Underfitting model

When a model has not learned the patterns in the training data well and is unable to generalize well on the new data, it is known as underfitting. An underfit model has poor performance on the training data and will result in unreliable predictions. Underfitting occurs due to high bias and low variance.
Techniques to reduce underfitting:
Increase model complexity
Increase the number of features, performing feature engineering
Remove noise from the data.
Increase the number of epochs or increase the duration of training to get better results.

Linear discriminant analysis (LDA)

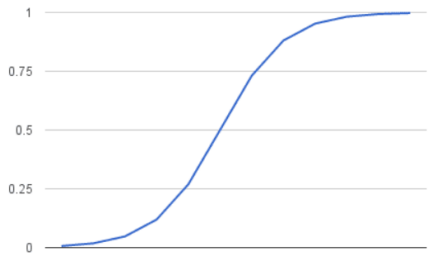
Linear Discriminant Analysis or LDA is a dimensionality reduction technique. It is used as a pre-processing step in Machine Learning and applications of pattern classification.LDA is a supervised classification technique that is considered a part of crafting competitive machine learning models.. It is used for modelling differences in groups i.e. separating two or more classes. It is used to project the features in higher dimension space into a lower dimension space.

Logistic Regression

ogistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis,logistic regression is estimating the parameters of a logistic model(a form of binary regression)Logistic Regression is used when the dependent variable (target) is categorical.

For example:-

- To predict whether an email is spam (1) or (0)
- Whether the tumor is malignant (1) or not (0)



Logistic Function

Candidate Elimination Algorithm Examples

Sky	Temperature	Humid	Wind	Water	Forest	Output
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	change	yes

Initially : G = [[?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?],
[?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?]]
S = [Null, Null, Null, Null, Null, Null]

For instance 1 : <'sunny','warm','normal','strong','warm ','same'> and positive output.
G1 = G
S1 = ['sunny','warm','normal','strong','warm ','same']

For instance 2 : <'sunny','warm','high','strong','warm ','same'> and positive output.
G2 = G
S2 = ['sunny','warm',?,'strong','warm ','same']

For instance 3 : <'rainy','cold','high','strong','warm ','change'> and negative output.
G3 = [['sunny', ?, ?, ?, ?, ?], [?, 'warm', ?, ?, ?, ?], [?, ?, ?, ?, ?, ?],
[?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, 'same']]
S3 = S2

For instance 4 : <'sunny','warm','high','strong','cool','change'> and positive output.
G4 = G3
S4 = ['sunny','warm',?,'strong', ?, ?]

At last, by synchronizing the G4 and S4 algorithm produce the output.
G = [['sunny', ?, ?, ?, ?, ?], [?, 'warm', ?, ?, ?, ?]]
S = ['sunny','warm',?,'strong', ?, ?]

Q2.

Example	Citations	Size	InLibrary	Price	Editions
1	Some	Small	No	Affordable	One
2	Many	Big	No	Expensive	Many
3	Many	Medium	No	Expensive	Few
4	Many	Small	No	Affordable	Many

<u>k-Medoids Clustering Algorithm</u>

K-Medoids (also called Partitioning Around Medoid) algorithm was proposed in 1987 by Kaufman and Rousseeuw. A medoid can be defined as a point in the cluster, whose dissimilarities with all the other points in the cluster are minimum. The dissimilarity of the medoid(Ci) and object(Pi) is calculated by using E = |Pi – Ci|

Initialize: select k random points out of the n data points as the medoids.

Associate each data point to the closest medoid by using any common distance metric methods.

While the cost decreases: For each medoid m, for each data o point which is not a medoid:

Swap m and o, associate each data point to the closest medoid, and recompute the cost.

If the total cost is more than that in the previous step, undo the swap.

Calculating cost. The dissimilarity of each non-medoid point with the medoids is calculated and tabulated:

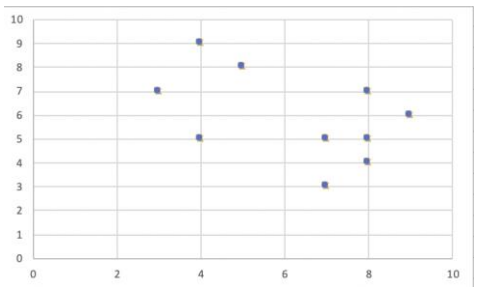
	X	Y	Dissimilarity from C1	Dissimilarity from C2
0	8	7	6	2
1	3	7	3	7
2	4	9	4	8
3	9	6	6	2
4	8	5	-	-
5	5	8	4	6
6	7	3	5	3
7	8	4	5	1
8	7	5	3	1
9	4	5	-	-

Here we have used Manhattan distance formula to calculate the distance matrices between medoid and non-medoid points. That formula tell that Distance = |X1-X2| + |Y1-Y2|.

Each point is assigned to the cluster of that medoid whose dissimilarity is less. Points 1, 2, and 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2. The Cost = (3 + 4 + 4) + (3 + 1 + 1 + 2 + 2) = 20

	X	Y
0	8	7
1	3	7
2	4	9
3	9	6
4	8	5
5	5	8
6	7	3
7	8	4
8	7	5
9	4	5

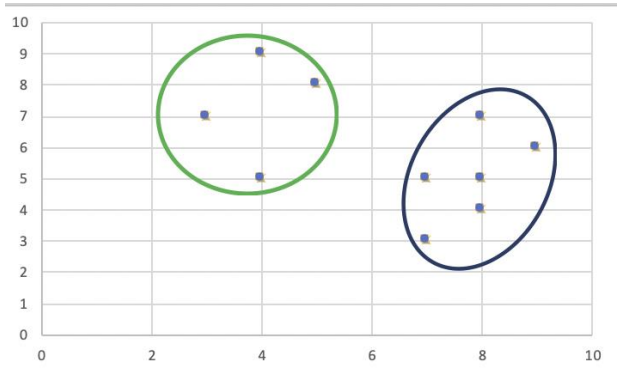
Let's consider the following example: If a graph is drawn using the above data points, we obtain the following:



Step 3: randomly select one non-medoid point and recalculate the cost. Let the randomly selected point be (8, 4). The dissimilarity of each non-medoid point with the medoids – C1 (4, 5) and C2 (8, 4) is calculated and tabulated.

	X	Y	Dissimilarity from C1	Dissimilarity from C2
0	8	7	6	3
1	3	7	3	8
2	4	9	4	9
3	9	6	6	3
4	8	5	4	1
5	5	8	4	7
6	7	3	5	2
7	8	4	-	-
8	7	5	3	2
9	4	5	-	-

Each point is assigned to that cluster whose dissimilarity is less. So, points 1, 2, and 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2. The New cost = (3 + 4 + 4) + (2 + 2 + 1 + 3 + 3) = 22 Swap Cost = New Cost – Previous Cost = 22 – 20 and 2 >0 As the swap cost is not less than zero, we undo the swap. Hence (4, 5) and (8, 5) are the final medoids. The clustering would be in the following way



Different Evaluation Metrics for Regression Problems

There are three error metrics that are commonly used for evaluating and reporting the performance of a regression model; they are:

- Mean Squared Error (MSE).
- Root Mean Squared Error (RMSE).
- Mean Absolute Error (MAE)

Mean Squared Error

Mean Squared Error, or MSE for short, is a popular error metric for regression problems.It is also an important loss function for algorithms fit or optimized using the least squares framing of a regression problem. Here “least squares” refers to minimizing the mean squared error between predictions and expected values.

The MSE is calculated as the mean or average of the squared differences between predicted and expected target values in a dataset.

- MSE = 1 / N * sum for i to N (y_i – yhat_i)^2

Where y_i is the i'th expected value in the dataset and yhat_i is the i'th predicted value. The difference between these two values is squared, which has the effect of removing the sign, resulting in a positive error value.

Root Mean Squared Error

The Root Mean Squared Error, or RMSE, is an extension of the mean squared error.

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

For example, if your target variable has the units "dollars," then the RMSE error score will also have the unit "dollars" and not "squared dollars" like the MSE.

As such, it may be common to use MSE loss to train a regression predictive model, and to use RMSE to evaluate and report its performance.

The RMSE can be calculated as follows:

- RMSE = sqrt(1 / N * sum for i to N (y_i – yhat_i)^2)

Where y_i is the i'th expected value in the dataset, yhat_i is the i'th predicted value, and sqrt() is the square root function.

We can restate the RMSE in terms of the MSE as:

- RMSE = sqrt(MSE)

Mean Absolute Error

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

Unlike the RMSE, the changes in MAE are linear and therefore intuitive.

That is, MSE and RMSE punish larger errors more than smaller errors, inflating or magnifying the mean error score. This is due to the square of the error value. The MAE does not give more or less weight to different types of errors and instead the scores increase linearly with increases in error.

As its name suggests, the MAE score is calculated as the average of the absolute error values. Absolute or abs() is a mathematical function that simply makes a number positive. Therefore, the difference between an expected and predicted value may be positive or negative and is forced to be positive when calculating the MAE.

The MAE can be calculated as follows:

- MAE = 1 / N * sum for i to N abs(y_i – yhat_i)

Where y_i is the i'th expected value in the dataset, yhat_i is the i'th predicted value and abs() is the absolute function.

Issues in Machine Learning.

1. Lack Of Quality Data
2. Non-representative training data
3. Overfitting and Underfitting
4. Monitoring and maintenance
5. Getting bad recommendations
6. Lack of skilled resources
7. Customer Segmentation
8. Process Complexity of Machine Learning
9. Data Bias
10. Irrelevant features

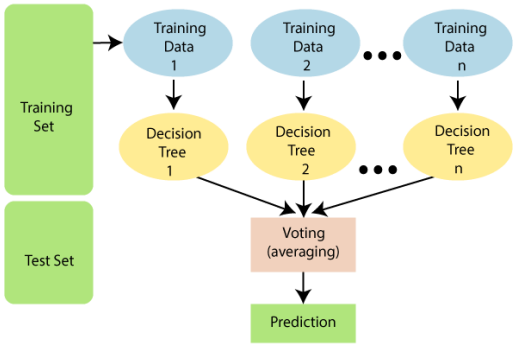
Reinforcement Learning

Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.Since there is no labeled data, so the agent is bound to learn by its experience only.RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as game-playing, robotics, etc.

Random Forests

The Forest-based Classification and Regression tool trains a model based on known values provided as part of a training dataset. This prediction model can then be used to predict unknown values in a prediction dataset that has the same associated explanatory variables.
Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:
Step-1: Select random K data points from the training set.
Step-2: Build the decision trees associated with the selected data points (Subsets).
Step-3: Choose the number N for decision trees that you want to build.
Step-4: Repeat Step 1 & 2.



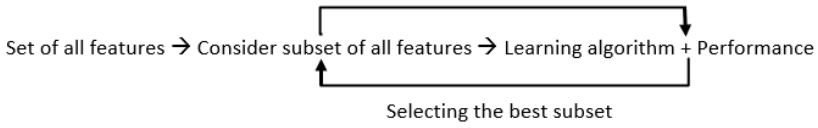
Various ways of Feature Selection

There are mainly two types of Feature Selection techniques, which are:
Supervised Feature Selection technique
Supervised Feature selection techniques consider the target variable and can be used for the labelled dataset.
Unsupervised Feature Selection technique
Unsupervised Feature selection techniques ignore the target variable and can be used for the unlabelled dataset.
There are mainly three techniques under supervised feature Selection:
1. Wrapper Methods
Wrapper methods, also referred to as greedy algorithms train the algorithm by using a subset of features in an iterative manner. Based on the conclusions made from training in prior to the model, addition and removal of features takes place. Stopping criteria for selecting the best subset are usually pre-defined by the person training the model such as when the performance of the model decreases or a specific number of features has been achieved. The main advantage of wrapper methods over the filter methods is that they provide an optimal set of features for training the model, thus resulting in better accuracy than the filter methods but are computationally more expensive.

2. Filter Methods
These methods are generally used while doing the pre-processing step. These methods select features from the dataset irrespective of the use of any machine learning algorithm. In terms of computation, they are very fast and inexpensive and are very good for removing duplicated, correlated, redundant features but these methods do not remove multicollinearity. Selection of feature is evaluated individually which can sometimes help when features are in isolation (don't have a dependency on other features) but will lag when a combination of features can lead to increase in the overall performance of the model.

Set of all features → Selecting the best subset → Learning algorithm → Performance

3. Embedded Methods
In embedded methods, the feature selection algorithm is blended as part of the learning algorithm, thus having its own built-in feature selection methods. Embedded methods encounter the drawbacks of filter and wrapper methods and merge their advantages. These methods are faster like those of filter methods and more accurate than the filter methods and take into consideration a combination of features as well.



Principal Component Analysis(PCA)algorithm

Some properties of these principal components are given below:

- The principal component must be the linear combination of the original features.
- These components are orthogonal, i.e., the correlation between a pair of variables is zero.
- The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

Steps for PCA algorithm

1. **Getting the dataset**
   Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.
2. **Representing data into a structure**
   Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.

3. **Standardizing the data**
   In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.
   If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.
4. **Calculating the Covariance of Z**
   To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.
5. **Calculating the Eigen Values and Eigen Vectors**
   Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.
6. **Sorting the Eigen Vectors**
   In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.
7. **Calculating the new features Or Principal Components**
   Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other.
8. **Remove less or unimportant features from the new dataset.**
   The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

Naive Bayesian Classification Example

Q1.

ID	Homeowner	Status	Income	Defaulted
1	YES	Employed	High	No
2	NO	Business	Average	NO
3	NO	Employed	Low	NO
4	YES	Business	High	NO
5	NO	Unemployed	Average	Yes
6	NO	Business	Low	No
7	YES	Unemployed	High	NO
8	NO	Employed	Average	Yes
9	NO	Business	Low	No
10	NO	Employed	Average	Yes

Q2. the probability of dangerous fires is rare (1%)

but smoke is fairly common (10%) due to barbecues,

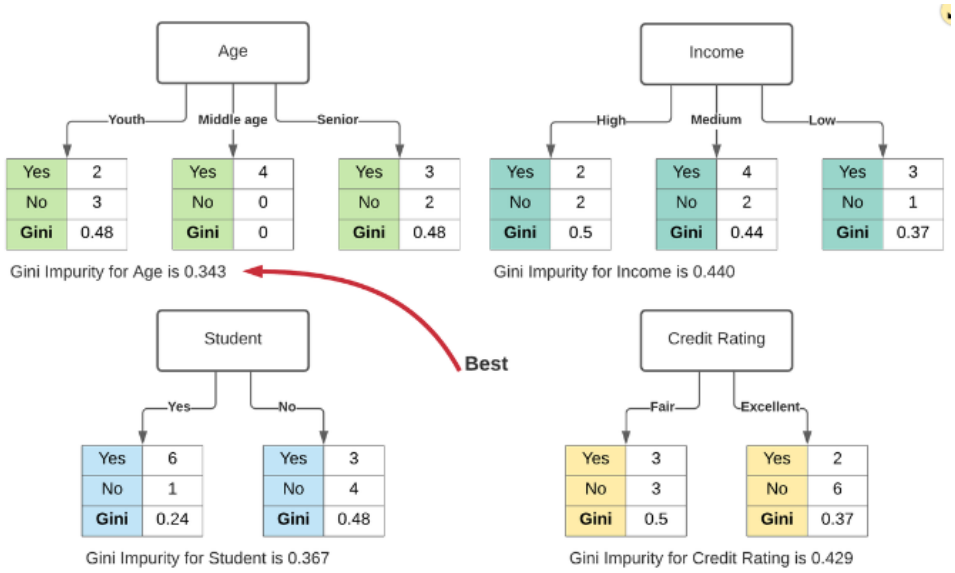
and 90% of dangerous fires make smoke

Find the probability of dangerous Fire when there is Smoke?

Gini Impurity

Gini Impurity is a measurement of the likelihood of an **incorrect classification** of a new instance of a random variable, if that new instance were **randomly classified according to the distribution of class labels** from the data set.

Gini impurity is **lower bounded by 0**, with 0 occurring if the data set contains only one class.



Generative models are models where the focus is the distribution of individual classes in a dataset and the learning algorithms tend to model the underlying patterns/distribution of the data points. These models use the intuition of joint probability in theory, creating instances where a given feature (x)/input and the desired output/label (y) exist at the same time.

Generative models use probability estimates and likelihood to model data points and distinguish between different class labels in a dataset. These models are capable of generating new data instances. However, they also have a major drawback. The presence of outliers affects these models to a significant extent.

Discriminative models, also called conditional models, tend to learn the boundary between classes/labels in a dataset. Unlike generative models, the goal here is to find the decision boundary separating one class from another.

So while a generative model will tend to model the joint probability of data points and is capable of creating new instances using probability estimates and maximum likelihood, discriminative models (just as in the literal meaning) separate classes by rather modeling the conditional probability and do not make any assumptions about the data point. They are also not capable of generating new data instances.

Discriminative models have the advantage of being more robust to outliers, unlike the generative models.

The following criteria between Discriminative and Generative Models:

- Performance
- Missing Data
- Accuracy Score
- Applications

Based on Performance

Generative models need fewer data to train compared with discriminative models since generative models are more biased as they make stronger assumptions i.e, assumption of conditional independence.

Based on Missing Data

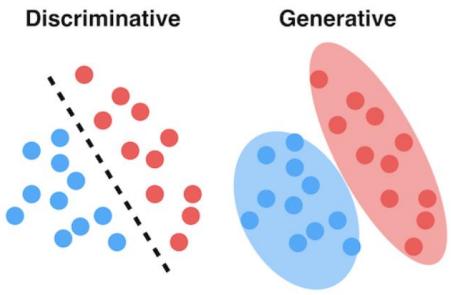
In general, if we have missing data in our dataset, then Generative models can work with these missing data, while on the contrary discriminative models can't. This is because, in generative models, still we can estimate the posterior by marginalizing over the unseen variables. However, for discriminative models, we usually require all the features X to be observed.

Based on Accuracy Score

If the assumption of conditional independence violates, then at that time generative models are less accurate than discriminative models.

Based on Applications

Discriminative models are called "discriminative" since they are useful for discriminating Y's label i.e, target outcome, so they can only solve classification problems while Generative models have more applications besides classification .



Examples of machine learning generative models

- Naive Bayes (and generally Bayesian networks)
- Hidden Markov model
- Linear discriminant analysis (LDA), a dimensionality reduction technique

Examples of discriminative models in machine learning are:

- Logistic regression
- Support vector machine
- Decision tree
- Random forest