

55. What is the effect of sampling on KNN?

Ans: Sampling does several things from the perspective of a single data point since kNN works on a point-by-point basis.

1. The average distance to the k nearest neighbours increases due to increased sparsity in the dataset.
2. Consequently, the area covered by k-nearest neighbours increases in size and covers a larger area of the feature space.
3. The sample variance increases.

A consequence of this change in input is an increase in variance. When we talk of variance, we refer to the variability in the predictions given different samples from the population. Why would the immediate effects of sampling lead to the increased variance of the model?

Notice that now a larger area of the feature space is represented by the same k data points. While our sample size has not grown, the population space that it represents has increased in size. This will result in higher variance in the proportion of classes in the k nearest data points, and consequently a higher variance in the classification of each data point.

56. What happens when we change the value of K in KNN?

Ans: Short Answer: The class boundaries of the predictions become more smooth as k increases.

Long Answer: What really is the significance of these effects? First, it gives hints that a lower k value makes the KNN model more "sensitive." That is, it is more sensitive to the local changes in the dataset. The "sensitivity" of the model directly translates to its variance.

All of these examples point to an inverse relationship between variance and k. Additionally, consider how KNN operates when k reaches its maximum value, $k=n$, where n is the number of points in the training set. In this case, the majority class in the training set will always dominate the predictions. It will simply pick the most abundant class in the data, and never deviate, effectively resulting in zero variance. Therefore, it seems to reduce variance, k must be increased.

Final Verdict: In order to offset the increased variance due to sampling, k can be increased to decrease model variance.

57. What is the thumb rule to approach the KNN problem?

Ans:

1. Load the data
2. Initialize the value of k
 - Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
 - Sort the calculated distances in ascending order based on distance values
 - Get top k rows from the sorted array
 - Get the most frequent class of these rows
 - Return the predicted class for getting the predicted class, iterate from 1 to the total number of training data points.

KNN Code Snippet:

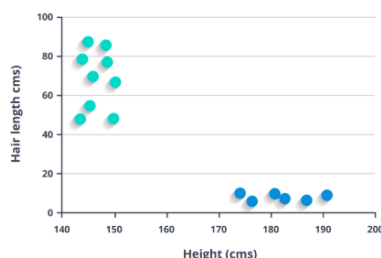
```
from sklearn.datasets import load_digits
# K-NN Algorithm for Classification | Author: Shubh Satyam
# Creating our own dataset
# Assigning features and label variables
# First feature
weather = ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast', 'Sunny', 'Sunny',
           'Overcast', 'Overcast', 'Overcast', 'Rainy']
# Second feature
temp = ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild', 'Cool', 'Mild', 'Mild', 'Hot', 'Mild']
# Label or target variable
play = ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
# Import LabelEncoder
from sklearn import preprocessing
# Creating LabelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers
weather_encoded = le.fit_transform(weather)
print(weather_encoded)
# Converting string labels into numbers
temp_encoded = le.fit_transform(temp)
label = le.fit_transform(play)
# Combining weather and temp into single list of features
features = list(zip(weather_encoded, temp_encoded))
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
# Train the model using the training sets
model.fit(features, label)
# Predict Output
predicted = model.predict([[0,2]]) # 0:Overcast, 2:Mild
print(predicted)
```

(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/pehla.png>)

58. What is SVM Algorithm?

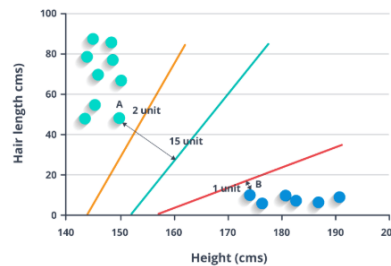
Ans: SVM stands for support vector machine, it is a supervised machine learning algorithm that can be used for both Regression and Classification. In this algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate.

For example, if we only had two features like Height and Hair length of an individual, we'd first plot these two variables in two-dimensional space where each point has two coordinates (these co-ordinates are known as Support Vectors)



(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/svm.png>)

Now, we will find some line that splits the data between the two differently classified groups of data. This will be the line such that the distances from the closest point in each of the two groups will be farthest away.



(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/svm2.png>)

In the example shown above, the line which splits the data into two differently classified groups is the black line, since the two closest points are the farthest apart from the line. This line is our classifier. Then, depending on where the testing data lands on either side of the line, that's what class we can classify the new data as.

59. What are support Vectors?

Ans: A support vector machine attempts to find the line that “best” separates two classes of points. By “best”, we mean the line that results in the largest margin between the two classes. The points that lie on this margin are the support vectors.

The vectors that define the hyperplane are the support vectors.

60. What is the purpose of the Support Vector in SVM?

Ans: A Support Vector Machine (SVM) performs classification by finding the hyperplane that maximizes the distance margin between the two classes. The extreme points in the data sets that define the hyperplane are the support vectors.

61. What are kernels?

Ans: SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of the kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be of different types.

There are four types of kernels in SVM.

1. Linear Kernel
2. Polynomial kernel
3. Radial basis kernel
4. Sigmoid kernel

62. What is Kernel Trick?

Ans: Short Answer: It allows us to operate in the original feature space without computing the coordinates of the data in a higher-dimensional space.

Long Answer:

1. For a dataset with n features (n -dimensional), SVMs find an $n-1$ -dimensional hyperplane to separate it (let us say for classification)
2. Thus, SVMs perform very badly with datasets that are not linearly separable
3. SVM can now do well with datasets that are not linearly separable
4. But, quite often, it's possible to transform our not-linearly-separable dataset into a higher-dimensional dataset where it becomes linearly separable, so that SVMs can do a good job
5. Unfortunately, quite often, the number of dimensions you have to add (via transformations) depends on the number of dimensions you already have (and not linearly)
 1. For datasets with a lot of features, it becomes next to impossible to try out all the interesting transformations
6. Enter the Kernel Trick
 - Thankfully, the only thing SVMs need to do in the (higher-dimensional) feature space (while training) is computing the pair-wise dot products
 - For a given pair of vectors (in lower-dimensional feature space) and a transformation into a higher-dimensional space, there exists a function (The Kernel Function) which can compute the dot product in the higher-dimensional space without explicitly transforming the vectors into the higher-dimensional space first
 - We are saved!

63. Why is SVM called as Large Margin Classifier?



Ans: Short Answer: Because it places the decision boundary such that it maximizes the distance between two clusters.

Long Answer: choosing the best hyperplane is to choose one in which the distance from the training points is the maximum. This is formalized by the geometric margin. Without getting into the details of the derivation, the geometric margin is given by:

$$\gamma^{(i)} = \frac{y^{(i)}(w^T x^{(i)} + b)}{\|w\|}$$

(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/11.png>)

Which is simply the functional margin normalized. So, these intuitions lead to the maximum margin classifier which is a precursor to the SVM.

64. What is the difference between Logistics Regression and SVM? When to use which model?

Ans:

1. SVM tries to find the "best" margin (distance between the line and the support vectors) that separates the classes and this reduces the risk of error on the data, while logistic regression does not, instead it can have different decision boundaries with different weights that are near the optimal point.
2. SVM works well with unstructured and semi-structured data like text and images while logistic regression works with already identified independent variables.
3. SVM is based on the geometrical properties of the data while logistic regression is based on statistical approaches.
4. Logistic Regression can't be applied to a nonlinearly separable dataset whereas SVM can be applied.
5. The risk of overfitting is less in SVM, while Logistic regression is vulnerable to overfitting.

65. When to Use Logistic Regression vs Support Vector Machine?

Ans: Depending on the number of training sets (data)/features that you have, you can choose to use either logistic regression or support vector machine.

Let's take these as an example where:

n = number of features,

m = number of training examples

1. If n is large (1–10,000) and m is small (10–1000): use logistic regression or SVM with a linear kernel.
2. If n is small (1–1000) and m is intermediate (10–10,000): use SVM with (Gaussian, polynomial, etc) kernel
3. If n is small (1–100), m is large (50,000–1,000,000+): first, manually add more features and then use logistic regression or SVM with a linear kernel

66. What does c and gamma parameter in SVM signify?

Ans: Short Answer:

Cost and Gamma are the hyper-parameters that decide the performance of an SVM model. There should be a fine balance between Variance and Bias for any ML model. (this is a science and an art – as we call it in empirical studies)

For SVM, a High value of Gamma leads to more accuracy but biased results and vice-versa. Similarly, a large value of Cost parameter (C) indicates poor accuracy but low bias and vice-versa.

Following table summarizes the above explanation –

The art is to choose a model with optimum variance and bias. Therefore, you need to choose the values of C and Gamma accordingly.

Optimum values of C and Gamma can be found by using methods like Grid search.

Long Answer:

The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'.

The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. If gamma is too large, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C will be able to prevent overfitting.



When gamma is very small, the model is too constrained and cannot capture the complexity or "shape" of the data. The region of influence of any selected support vector would include the whole training set. The resulting model will behave similarly to a linear model with a set of hyperplanes that separate the centers of the high density of any pair of two classes.

67. What are the Advantages and Disadvantages of SVM?

Ans: SVM Advantages

- SVM's are very good when we have no idea about the data.
- Works well with even unstructured and semi-structured data like text, Images, and trees.
- The kernel trick is a real strength of SVM. With an appropriate kernel function, we can solve any complex problem.
- Unlike in neural networks, SVM is not solved for local optima.
- It scales relatively well to high dimensional data.
- SVM models have generalization in practice, the risk of over-fitting is less in SVM.
- SVM is always compared with ANN. When compared to ANN models, SVMs give better results.

SVM Disadvantages

- Choosing a "good" kernel function is not easy.
- Long training time for large datasets.
- Difficult to understand and interpret the final model, variable weights, and individual impact.
- Since the final model is not so easy to see, we can not do small calibrations to the model hence its tough to incorporate our business logic.
- The SVM hyperparameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact.

SVM code snippet:

```
1 # Import the necessary libraries
2 import numpy as np
3 from sklearn.datasets import load_breast_cancer
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import accuracy_score
6 from sklearn.svm import SVC
7
8 # Load the dataset
9 cancer = load_breast_cancer()
10
11 # Split the dataset into training and testing sets
12 X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3, random_state=0)
13
14 # Create a new classifier
15 clf = SVC(kernel='linear')
16
17 # Train the model using the training sets
18 clf.fit(X_train, y_train)
19
20 # Predict the labels for the test dataset
21 y_pred = clf.predict(X_test)
22
23 # Model Accuracy: how often is the classifier correct?
24 print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
25
26 # Model Precision: what percentage of positive labels are labeled as such?
27 print("Precision:", metrics.precision_score(y_test, y_pred))
28
29 # Model Recall: what percentage of positive labels are labeled as such?
30 print("Recall:", metrics.recall_score(y_test, y_pred))
```

(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/dusra.png>)

68. What is Naïve Bayes Algorithm?

Ans: It is a classification algorithm that predicts the probability of each data point belonging to a class and then classifies the point as the class with the highest probability.

Discuss Bayes Theorem.

Bayes' Theorem gives us the probability of an event actually happening by combining the conditional probability given some result and the prior knowledge of an event happening.

Conditional probability is the probability that something will happen, given that something has occurred. In other words, the conditional probability is the probability of X given a test result or $P(X|Test)$. For example, what is the probability an e-mail is spam given that my spam filter classified it as spam.

The prior probability is based on previous experience or the percentage of previous samples. For example, what is the probability that any email is spam?

Formally

- $P(A|B)$ = Posterior probability = Probability of A given B happened
- $P(B|A)$ = Conditional probability = Probability of B happening if A is true
- $P(A)$ = Prior probability = Probability of A happening in general
- $P(B)$ = Evidence probability = Probability of getting a positive test

69. Why is Naïve Bayes Naïve?

Ans: In Layman's Term: The simple meaning of Naive is willing to believe that that life is simple and fair, which is not true. Naive Bayes is naive because it assumes that the features that are going into the model are not related to each other anyhow Change in one variable will not affect the other variable directly.

Long Answer: Naive Bayes (NB) is 'naive' because it makes the assumption that features of measurement are independent of each other. This is naive because it is (almost) never true. Here is how it works even then – NB is a very intuitive classification algorithm. It asks the question, "Given these features, does this measurement belong to class A or B?", and answers it by taking the proportion of all previous measurements with the same features belonging to class A multiplied by the proportion of all measurements in class A. If this number is bigger than the corresponding calculation for class B then we say the measurement belongs in class A.



70. What are feature matrix and response vectors?

Ans: Feature matrix:- The feature matrix contains all the vectors(rows) of the dataset in which each vector consists of the value of dependent features.

Response vectors:- The response vector contains the value of the class variable (prediction or output) for each row of the feature matrix.

71 Applications of Naïve Bayes Classification Algorithms?

Ans: Some of the real-world examples are as given below

- To mark an email as spam, or not spam?
- Classify a news article about technology, politics, or sports?
- Check a piece of text expressing positive emotions, or negative emotions?
- Also used for face recognition software.

72. What are the Advantages and Disadvantages of using the Naïve Bayes Algorithm?

Ans: Advantages

1. Fast
2. Highly scalable.
3. Used for binary and Multiclass Classification.
4. Great Choice for text classification.
5. It can easily train smaller data sets.

Disadvantages

Naive Bayes considers that the features are independent of each other. However, in the real-world, features depend on each other.

Naïve Bayes Code Snippet:

```
1 # Import the Naive Bayes
2 # Naive Bayes class
3 # Import the sklearn library
4 from sklearn import datasets
5 from sklearn.cross_validation import train_test_split
6 from sklearn.naive_bayes import GaussianNB
7 from sklearn import metrics
8
9 # Load dataset
10 wine = datasets.load_wine()
11
12 # Split dataset into training set and test set
13 X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, test_size=0.3, random_state=100)
14
15 # Create a Gaussian classifier
16 gnb = GaussianNB()
17
18 # Train the model using the training sets
19 gnb.fit(X_train, y_train)
20
21 # Predict the response for test dataset
22 y_pred = gnb.predict(X_test)
23
24 # Model accuracy, how often is the classifier correct?
25 print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/teesra.png>)

73. What is K-Means Clustering? What are the steps for it?

Ans: K-means (Macqueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. K-means clustering is a method of vector quantization, original from signal processing, that is popular for cluster analysis in data mining.

If k is given, the K-means algorithm can be executed in the following steps:

- Partition of objects into k non-empty subsets
- Identifying the cluster centroids (mean point) of the current partition.
- Assigning each point to a specific cluster
- Compute the distances from each point and allot points to the cluster where the distance from the centroid is minimum.
- After re-allotting the points, find the centroid of the new cluster formed.

74. Why is the word “means” associated with the name of the K-Means algorithm?

Ans: The 'means' in the K-means refers to **averaging of the data**; that is, finding the centroid.

There are k-medoids and k-medians algorithms as well.

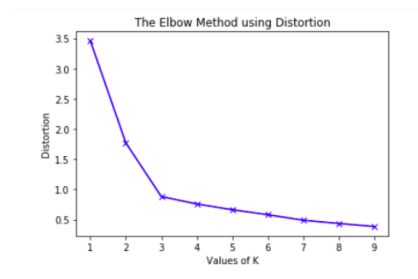
k-medoids minimizes the sum of dissimilarities between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the k-means algorithm, k-medoids choose datapoints as centers (medoids or exemplars).

k-medians is a variation of k-means clustering where instead of calculating the mean for each cluster to determine its centroid, one instead calculates the median.

75. How to find the optimum number of clusters in K-Means? Discuss the elbow curve/elbow method?

Ans: The basic idea behind partitioning methods, such as k-means clustering, is to define clusters such that the total intra-cluster variation [or total within-cluster sum of square (WSS)] is minimized. The total WSS measures the compactness of the clustering and we want it to be as small as possible.

The Elbow method looks at the total WSS as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't improve much better the total WSS.



(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/12.png>)

Notice the elbow at $k=3$.

The optimal number of clusters can be defined as follow:

1. Compute clustering algorithm (e.g., k-means clustering) for different values of k . For instance, by varying k from 1 to 10 clusters.
2. For each k , calculate the total within-cluster sum of square (WSS).
3. Plot the curve of WSS according to the number of clusters k .
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

76. What is the difference between K-Means and Hierarchical Clustering? When to use which?

Ans: Hierarchical Clustering and k-means clustering complement each other. In hierarchical clustering, the researcher is not aware of the number of clusters to be made whereas, in k-means clustering, the number of clusters to be made is specified before-hand.

Advice- If unaware of the number of clusters to be formed, use hierarchical clustering to determine the number and then use k-means clustering to make more stable clusters as hierarchical clustering is a single-pass exercise whereas k-means is an iterative process.

77. What are the advantages and disadvantages of using K-Means Algorithms?

Ans: K-Means Advantages :

- 1) If variables are huge, then K-Means most of the times computationally faster than hierarchical clustering, if we keep k smalls.
- 2) K-Means produce tighter clusters than hierarchical clustering, especially if the clusters are globular.

K-Means Disadvantages:

- 1) Difficult to predict K-Value.
- 2) With a global cluster, it didn't work well.
- 3) Different initial partitions can result in different final clusters.
- 4) It does not work well with clusters (in the original data) of Different sizes and Different density.

KNN code snippet:

```
1 # Author: Souvik Dey
2 # K-Means Clustering
3 # Importing Libraries
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6 import numpy as np
7 from sklearn.cluster import KMeans
8
9 #Prepare Data
10
11 X = np.array([[5,1],
12               [10,15],
13               [15,12],
14               [10,10],
15               [18,5],
16               [12,7],
17               [17,8],
18               [18,10],
19               [10,12],
20               [15,10]])
21
22 #Visualize Data
23 plt.scatter(X[:,0],X[:,1], label='True Position')
24
25 #KMeans Clusters
26 kmeans = KMeans(n_clusters=2)
27 kmeans.fit(X)
28
29 #Let's see what centroid values the algorithm generated for the final clusters.
30 print(kmeans.cluster_centers_)
31
32 #If we are able to see the data point, execute the following script.
33 print(kmeans.labels_)
34
35 #Let's plot the data points again on the graph and visualize how the data has been clustered.
36 #Plotting along with the labels
37 plt.scatter(X[:,0],X[:,1], c=kmeans.labels_, cmap='rainbow')
38 #Setting axis
39 plt.scatter(X[:,0], X[:,1], c=kmeans.labels_, cmap='rainbow')
40 plt.scatter(kmeans.cluster_centers_[0,0], kmeans.cluster_centers_[0,1], color='black')
```

(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/choutha.png>)

78. What is Hierarchical Clustering?

Ans: Hierarchical clustering is another unsupervised learning algorithm that is used to group together the unlabelled data points having similar characteristics. Hierarchical clustering algorithms fall into the following two categories.

Agglomerative hierarchical algorithms – In agglomerative hierarchical algorithms, each data point is treated as a single cluster and then successively merge or agglomerate (bottom-up approach) the pairs of clusters. The hierarchy of the clusters is represented as a dendrogram or tree structure.

Divisive hierarchical algorithms – On the other hand, in divisive hierarchical algorithms, all the data points are treated as one big cluster and the process of clustering involves dividing (Top-down approach) the one big cluster into various small clusters

79. What are the steps to perform Agglomerative Hierarchical Clustering?

Ans: Most used and important Hierarchical clustering i.e. agglomerative. The steps to perform the same is as follows –

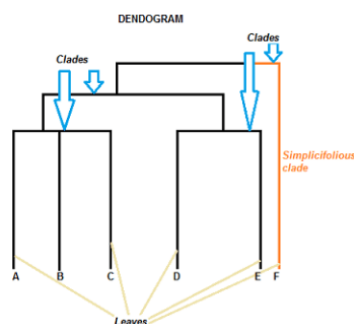
- **Step 1** – Treat each data point as a single cluster. Hence, we will be having, say K clusters at the start. The number of data points will also be K at the start.
- **Step 2** – Now, in this step we need to form a big cluster by joining two closet datapoints. This will result in a total of K-1 clusters.
- **Step 3** – Now, to form more clusters we need to join two closet clusters. This will result in a total of K-2 clusters.
- **Step 4** – Now, to form one big cluster repeat the above three steps until K would become 0 i.e. no more data points left to join.
- **Step 5** – At last, after making one single big cluster, dendrograms will be used to divide into multiple clusters depending upon the problem.

80. What is Dendrogram and what is its importance in Hierarchical Clustering?

Ans: A dendrogram is a type of Tree Diagram showing hierarchical clustering — relationships between similar sets of data. They are frequently used in biology to show clustering between genes or samples, but they can represent any type of grouped data.

The role of the dendrogram starts once the big cluster is formed. Dendrogram will be used to split the clusters into multiple clusters of related data points depending upon our problem.

Parts of Dendrogram:



(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/13.png>) **Hierarchical Clustering Code Snippet:**

```
Users > subyan > Desktop > ML_Algos > HierarchicalClustering.py
1 # author: Shreyas Sanyal (shreyas@learnbay.com)
2 # Hierarchical Clustering Example (with Dendrogram)
3 # Import Libraries
4 import matplotlib.pyplot as plt
5 import pandas as pd
6 %matplotlib inline
7 import numpy as np
8
9 # Import Data
10 customer_data = pd.read_csv('path/to/file.csv')
11
12 # To view the results in two-dimensional feature space,
13 # we will retain only two of these first columns. (If more than 2 available)
14 # data = customer_data.iloc[:, 3:5].values
15
16 """we need to know the clusters that we want our data to be split to.
17 We will again use the scipy library to create the dendrogram for our dataset.
18 Execute the following script to do so"""
19
20 import scipy.cluster.hierarchy as shc
21
22 plt.figure(figsize=(10, 7))
23 plt.title('Customer Dendrogram')
24 dend = shc.dendrogram(shc.linkage(data, method='ward'))
25
26 # Now we know the number of clusters for our dataset,
27 # the next step is to group the data points into these five clusters.
28 from sklearn.cluster import AgglomerativeClustering
29
30 cluster = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
31 cluster.fit_predict(data)
32
33 # As a final step, let's plot the clusters to see how actually our data has been clustered
34 plt.figure(figsize=(10, 7))
35 plt.scatter(data[:,0], data[:,1], c=cluster.labels_, cmap='rainbow')
```

(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/Panchwa.png>)

81. What is Boosting?

Ans: Boosting is a method of converting weak learners into strong learners. In boosting, each new tree is a fit on a modified version of the original data set.

Purpose of Boosting: It helps the weak learner to be modified to become better.

How it evolved: The first Boosting Algorithm gained popularity was AdaBoost or Adaptive Boosting. Further it evolved and generalized as Gradient Boosting.

82. What is Adaboost?

Ans: Adaboost combines multiple weak learners into a single strong learner. The weak learners in AdaBoost are decision trees with a single split, called decision stumps. When AdaBoost creates its first decision stump, all observations are weighted equally. To correct the previous error, the observations that were incorrectly classified now carry more weight than the observations that were correctly classified. AdaBoost algorithms can be used for both classification and regression problems.

Adaboost Code Snippet:




```

1 # Author: Simon Hurley
2 # AdaBoost Classification
3 # Importing Libraries
4 import pandas
5 from sklearn import model_selection
6 from sklearn.ensemble import AdaBoostClassifier
7
8 # Loading data from url
9 url = "https://raw.githubusercontent.com/bruneliot/datasets/master/pima_indian_diabetes_data.csv"
10 names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
11 dataframe = pandas.read_csv(url, names=names)
12 array = dataframe.values
13
14 # Preprocessing
15 x = array[:,0:8]
16 y = array[:,8]
17
18 # Train-Test Split and AdaBoost
19
20 seed = 7
21 train_test_size = 0.33
22 # Split = model_selection.shuffle(x, y, train_test_size, random_state=seed)
23 model = AdaBoostClassifier(x, y, train_test_size, random_state=seed)
24 results = model_selection.cross_val_score(model, x, y, cv=10)
25 print(results.mean())

```

(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/chatwaa.png>)

83. What is Gradient Boosting Method (GBM)?

Ans: Gradient Boosting works by sequentially adding predictors to an ensemble, each one correcting its predecessor. However, instead of changing the weights for every incorrect classified observation at every iteration like AdaBoost, the Gradient Boosting method tries to fit the new predictor to the residual errors made by the previous predictor.

GBM uses Gradient Descent to find the shortcomings in the previous learner's predictions. The GBM algorithm can be given in the following steps.

Fit a model to the data, $F_1(x) = y$

Create a new model, $F_2(x) = F_1(x) + h_1(x)$

By combining weak learners after weak learners, our final model is able to account for a lot of the error from the original model and reduces this error over time.

Gradient Boosting Code Snippet:

```

1 # Gradient Boosting
2 # Importing Libraries
3 import pandas as pd
4 from sklearn.preprocessing import RobustScaler
5 from sklearn.model_selection import train_test_split
6 from sklearn.metrics import (classification_report, confusion_matrix)
7 from sklearn.ensemble import GradientBoostingClassifier
8
9 # Preprocessing
10 # Load data values[0:8]
11 X_train = X_train.values[0:8]
12 y_train = y_train.values[0:8]
13
14 # RobustScaler
15 X_train = RobustScaler().fit_transform(X_train)
16 X_test = RobustScaler().fit_transform(X_test)
17
18 # Train-Test Split
19 train_test_size = 0.33
20 test_size = 0.33
21
22 # Split = model_selection.shuffle(x, y, train_test_size, random_state=seed)
23 X_train, X_test, y_train, y_test = train_test_split(X_train, y_train,
24                                                    test_size=test_size, random_state=seed)
25
26 # Training and Testing
27 # The following code will evaluate the performance of the classifier's performance at different learning rates
28 lr_list = [0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 1]
29
30 for learning_rate in lr_list:
31     gb = GradientBoostingClassifier(loss='deviance', learning_rate=learning_rate, max_depth=5, max_features='sqrt', min_samples_split=10)
32     gb.fit(X_train, y_train)
33
34     # Predictions
35     predicted_training = gb.predict(X_train)
36     predicted_test = gb.predict(X_test)
37
38     # Classification Report
39     print('Learning rate: ', learning_rate)
40     print('Training score: ', accuracy_score(y_train, predicted_training))
41     print('Test score: ', accuracy_score(y_test, predicted_test))
42
43 # Performance Metrics
44 # The following code will evaluate the performance of the classifier's performance at different learning rates
45 # The following code will evaluate the performance of the classifier's performance at different learning rates
46 # The following code will evaluate the performance of the classifier's performance at different learning rates
47 # The following code will evaluate the performance of the classifier's performance at different learning rates
48 # The following code will evaluate the performance of the classifier's performance at different learning rates
49 # The following code will evaluate the performance of the classifier's performance at different learning rates
50 # The following code will evaluate the performance of the classifier's performance at different learning rates
51 # The following code will evaluate the performance of the classifier's performance at different learning rates
52 # The following code will evaluate the performance of the classifier's performance at different learning rates
53 # The following code will evaluate the performance of the classifier's performance at different learning rates
54 # The following code will evaluate the performance of the classifier's performance at different learning rates
55 # The following code will evaluate the performance of the classifier's performance at different learning rates
56 # The following code will evaluate the performance of the classifier's performance at different learning rates
57 # The following code will evaluate the performance of the classifier's performance at different learning rates
58 # The following code will evaluate the performance of the classifier's performance at different learning rates
59 # The following code will evaluate the performance of the classifier's performance at different learning rates
60 # The following code will evaluate the performance of the classifier's performance at different learning rates
61 # The following code will evaluate the performance of the classifier's performance at different learning rates
62 # The following code will evaluate the performance of the classifier's performance at different learning rates
63 # The following code will evaluate the performance of the classifier's performance at different learning rates
64 # The following code will evaluate the performance of the classifier's performance at different learning rates
65 # The following code will evaluate the performance of the classifier's performance at different learning rates
66 # The following code will evaluate the performance of the classifier's performance at different learning rates
67 # The following code will evaluate the performance of the classifier's performance at different learning rates
68 # The following code will evaluate the performance of the classifier's performance at different learning rates
69 # The following code will evaluate the performance of the classifier's performance at different learning rates
70 # The following code will evaluate the performance of the classifier's performance at different learning rates
71 # The following code will evaluate the performance of the classifier's performance at different learning rates
72 # The following code will evaluate the performance of the classifier's performance at different learning rates
73 # The following code will evaluate the performance of the classifier's performance at different learning rates
74 # The following code will evaluate the performance of the classifier's performance at different learning rates
75 # The following code will evaluate the performance of the classifier's performance at different learning rates
76 # The following code will evaluate the performance of the classifier's performance at different learning rates
77 # The following code will evaluate the performance of the classifier's performance at different learning rates
78 # The following code will evaluate the performance of the classifier's performance at different learning rates
79 # The following code will evaluate the performance of the classifier's performance at different learning rates
80 # The following code will evaluate the performance of the classifier's performance at different learning rates
81 # The following code will evaluate the performance of the classifier's performance at different learning rates
82 # The following code will evaluate the performance of the classifier's performance at different learning rates
83 # The following code will evaluate the performance of the classifier's performance at different learning rates
84 # The following code will evaluate the performance of the classifier's performance at different learning rates
85 # The following code will evaluate the performance of the classifier's performance at different learning rates
86 # The following code will evaluate the performance of the classifier's performance at different learning rates
87 # The following code will evaluate the performance of the classifier's performance at different learning rates
88 # The following code will evaluate the performance of the classifier's performance at different learning rates
89 # The following code will evaluate the performance of the classifier's performance at different learning rates
90 # The following code will evaluate the performance of the classifier's performance at different learning rates
91 # The following code will evaluate the performance of the classifier's performance at different learning rates
92 # The following code will evaluate the performance of the classifier's performance at different learning rates
93 # The following code will evaluate the performance of the classifier's performance at different learning rates
94 # The following code will evaluate the performance of the classifier's performance at different learning rates
95 # The following code will evaluate the performance of the classifier's performance at different learning rates
96 # The following code will evaluate the performance of the classifier's performance at different learning rates
97 # The following code will evaluate the performance of the classifier's performance at different learning rates
98 # The following code will evaluate the performance of the classifier's performance at different learning rates
99 # The following code will evaluate the performance of the classifier's performance at different learning rates
100 # The following code will evaluate the performance of the classifier's performance at different learning rates

```

(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/satwa.png>)

84. What is XGBoost?

Ans: XGBoost stands for eXtreme Gradient Boosting. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. Gradient boosting machines are generally very slow in implementation because of sequential model training. Hence, they are not very scalable. Thus, XGBoost is focused on computational speed and model performance. XGBoost provides:

- **Parallelization** of tree construction using all of your CPU cores during training.
- **Distributed Computing** for training very large models using a cluster of machines.
- **Out-of-Core Computing** for very large datasets that don't fit into memory.
- **Cache Optimization** of data structures and algorithm to make the best use of hardware.

XGBoost Code Snippet:

```

1 # Author: Simon Hurley
2 # XGBoost Code
3 # Importing Libraries
4 from numpy import loadtxt
5 from xgboost import XGBClassifier
6 from sklearn.model_selection import train_test_split
7 from sklearn.metrics import accuracy_score
8
9 # Load data
10 dataset = loadtxt('pima_indian_diabetes.csv', delimiter=',')
11
12 # Split data into x and y
13 X = dataset[:,0:8]
14 y = dataset[:,8]
15
16 # Split data into train and test sets
17 seed = 7
18 test_size = 0.33
19 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=seed)
20
21 # Fit model on training data
22 model = XGBClassifier()
23 model.fit(X_train, y_train)
24
25 # Make predictions for test data
26 y_pred = model.predict(X_test)
27 predictions = [round(value) for value in y_pred]
28
29 # Evaluate predictions
30 accuracy = accuracy_score(y_test, predictions)
31 print('Accuracy: %.2f%%' % (accuracy * 100.0))

```

(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/aathwa.png>)

85. What are the basic enhancements done to Gradient Boosting?

Ans: Gradient boosting is a greedy algorithm and can overfit a training dataset quickly. It can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting.

We will look at 4 enhancements to basic gradient boosting:

1. Tree Constraints
2. Shrinkage
3. Random sampling
4. Penalized Learning

1. **Tree Constraints:** A good general heuristic is that the more constrained tree creation is, the more trees you will need in the model, and the reverse, where less constrained individual trees, the fewer trees that will be required.

Below are some constraints that can be imposed on the construction of decision trees:

- **The number of trees**, generally adding more trees to the model can be very slow to overfit. The advice is to keep adding trees until no further improvement is observed.
 - **Tree depth**, deeper trees are more complex trees, and shorter trees are preferred. Generally, better results are seen with 4-8 levels.
 - **The number of nodes** or number of leaves, like depth, can constrain the size of the tree but is not constrained to a symmetrical structure if other constraints are used.
 - **Number of observations per split** imposes a minimum constraint on the amount of training data at a training node before a split can be considered
 - Minimum improvement to loss is a constraint on the improvement of any split added to a tree.
2. **Penalized Gradient Boosting:** Additional constraints can be imposed on the parameterized trees in addition to their structure. Classical decision trees like CART are not used as weak learners, instead, a modified form called a regression tree is used that has numeric values in the leaf nodes (also called terminal nodes). The values in the leaves of the trees can be called weights in some literature. As such, the leaf weight values of the trees can be regularized using popular regularization functions, such as **L1 regularization of weights** and **L2 regularization of weights**. The additional regularization term helps to smooth the final learned weights to avoid over-fitting. Intuitively, the regularized objective will tend to select a model employing simple and predictive functions.
3. **Weighted Updates:** The predictions of each tree are added together sequentially. The contribution of each tree to this sum can be weighted to slow down the learning by the algorithm. This weighting is called a shrinkage or a learning rate.
4. **Stochastic Gradient Boosting:** A big insight into bagging ensembles and the random forest was allowing trees to be greedily created from subsamples of the training dataset. This same benefit can be used to reduce the correlation between the trees in the sequence in gradient boosting models. This variation of boosting is called stochastic gradient boosting. At each iteration a subsample of the training data is drawn at random (without replacement) from the full training dataset. The randomly selected subsample is then used, instead of the full sample, to fit the base learner.

86. What is Dimensionality Reduction? Why is it used?

Ans: Dimensionality reduction refers to the process of converting a set of data. That data needs to having vast dimensions into data with lesser dimensions. Also, it needs to ensure that it conveys similar information concisely.

Although, we use these techniques to solve machine learning problems. And the problem is to obtain better features for a classification or regression task.

87. What are the commonly used Dimensionality Reduction Techniques?

Ans: The various methods used for dimensionality reduction include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)

88. How does PCA work? When to use?

Ans: Short Answer: Principal Component Analysis (PCA) is an unsupervised, non-parametric statistical technique primarily used for dimensionality reduction in machine learning.

High dimensionality means that the dataset has a large number of features. The primary problem associated with high dimensionality in the machine learning field is model overfitting, which reduces the ability to generalize beyond the examples in the training set.

PCA in Layman's Term: Consider the 2D XY plane.

For the sake of intuition, let us consider variance as the spread of data – the distance between the two farthest points.

Assumption:

Typically, it is believed, that if the variance of data is large, it offers more information, than data that has a small variance. (This may or may not be true). This is the assumption which PCA intends to exploit.

I give you 4 points – {(1,1), (2,2), (3,3), (4,4)}
(all lie on the line X=Y)

What is the variance on X-axis?

Variance(X) = $4-1 = 3$

What is the variance on Y-axis?

Variance(Y) = $4-1 = 3$

Can we obtain new data with higher variance in some manner?

Rotate your XY system by 45 degrees anticlockwise. What happens? The line X=Y has now become the X(new)-axis. And, X = -Y is now the Y(new)-axis. Let's compute the variance again (in the form of distance)



Variance(X(new)) = distance ((4,4), (1,1)) = $\sqrt{18}$ = 4.24

Variance(Y(new)) = requires some calculations.

89. What did we get by doing this rotation?

Ans: Original data – had the highest variance on any axis as 3. This rotation gave us a variance of 4.24

That was the intuitive explanation of what PCA does. Just for further clarification

Eigenvalues = variance of the data along a particular axis in the new coordinate system. In above example, Eigenvalue(X(new)) = 4.24.

Eigenvectors = the vectors which represent the new coordinate system. In above example, vector [1,1], would be an eigenvector for X(new), and [1,-1] eigenvector for Y(new). Since they are just directions – solvers typically give us unit vectors.

Getting transformed data

Once you have the eigenvectors, a dot product of the eigenvector with the original point will give you the new point in the new coordinate system.

Diagonalization: This is the part where you equate covariance to λI . This is basically trying to find an eigenvector, such that all points would lie on the same line, and thus it will have only elements of variance, and covariance terms would be zero.

Steps of PCA:

1. Calculate the covariance matrix X of data points.
2. Calculate eigenvectors and correspond eigenvalues.
3. Sort eigenvectors accordingly to their given value in decrease order.
4. Choose first k eigenvectors and that will be the new k dimensions.
5. Transform the original n-dimensional data points into k-dimensions

PCA code snippet:

```
1 # Importing Libraries
2 # Principal Component Analysis Code
3 # Importing Libraries
4 import numpy as np
5 import pandas as pd
6 from sklearn.model_selection import train_test_split
7 from sklearn.preprocessing import StandardScaler
8 from sklearn.decomposition import PCA
9 from sklearn.metrics import confusion_matrix
10 from sklearn.metrics import classification_report
11 from sklearn.metrics import accuracy_score
12
13 # Loading the Dataset
14 dataset = pd.read_csv('path/to/data/file.csv')
15
16 # Preprocessing
17 X = dataset.drop('Class', 1)
18 Y = dataset['Class']
19
20 # Splitting the Data
21 X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
22
23 # Feature Scaling
24 sc = StandardScaler()
25 X_train = sc.fit_transform(X_train)
26 X_test = sc.transform(X_test)
27
28 # Initializing PCA
29 pca = PCA()
30 X_train = pca.fit_transform(X_train)
31 X_test = pca.transform(X_test)
32
33 # Finding the Explained Variance Ratio
34 explained_variance = pca.explained_variance_ratio_
35
36 # Using Principal Component to train the algorithm
37 # For simplicity, let's use only 1st component (i.e. first principal component)
38 pca = PCA(n_components=1)
39 X_train = pca.fit_transform(X_train)
40 X_test = pca.transform(X_test)
41
42 # Training the Logistic Regression Classifier
43 classifier = LogisticRegression()
44 classifier.fit(X_train, y_train)
45 y_pred = classifier.predict(X_test)
46
47 # Calculating Performance
48 cm = confusion_matrix(y_test, y_pred)
49 print(cm)
50 print('Accuracy :', accuracy_score(y_test, y_pred))
```

(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/gth.png>)

90. How does LDA work? When to use?

Ans: LDA is a way to reduce 'dimensionality' while at the same time preserving as much of the class discrimination information as possible.

How does it work?

Basically, LDA helps you find the 'boundaries' around clusters of classes. It projects your data points on a line so that your clusters 'are as separated as possible', with each cluster having a relative (close) distance to a centroid.

What was that stuff about dimensionality?

Let's say you have a group of data points in 2 dimensions, and you want to group them into 2 groups. LDA reduces the dimensionality of your settings like so:

$K(\text{Groups}) = 2, 2-1 = 1$.

Why? Because "The K centroids lie in an at most K-1-dimensional affine subspace". What is the affine subspace? It's a geometric concept or 'structure' that says, "I am going to generalize the affine properties of Euclidean space". What are those affine properties of the Euclidean space? Basically, it's the fact that we can represent a point with 3 coordinates in a 3-dimensional space (with a nod toward the fact that there may be more than 3 dimensions that we are ultimately dealing with).

So, we should be able to represent a point with 2 coordinates in 2-dimensional space and represent a point with 1 coordinate in a 1-dimensional space. LDA reduced the dimensionality of our 2-dimension problem down to one dimension. So now we can get down to the serious business of listening to the data. We now have 2 groups, and 2 points in any dimension can be joined by a line. How many dimensions does a line have? 1! Now we are cooking with Crisco!



So we get a bunch of these data points, represented by their 2d representation (x,y). We are going to use LDA to group these points into either group 1 or group 2.

91. What are the Steps for LDA?

Ans: Steps of LDA:

1. Compute the d-dimensional mean vector for the different classes from the dataset.
2. Compute the Scatter matrix (in between class and within the class scatter matrix)
3. Sort the Eigen Vector by decrease Eigen Value and choose k eigenvector with the largest eigenvalue to form a d x k dimensional matrix w (where every column represents an eigenvector)
4. Used $d \times k$ eigenvector matrix to transform the sample onto the new subspace.

This can be summarized by the matrix multiplication.

$Y = X \times W$ (where X is an $n \times d$ dimension matrix representing the n samples and **you** are transformed $n \times k$ dimensional samples in the new subspace.

LDA code snippet:

```
1 # Author: Shree Rajan
2 # Linear Discriminant Analysis Code
3 # Importing Libraries
4 import numpy as np
5 import pandas as pd
6 from sklearn.model_selection import train_test_split
7 from sklearn.preprocessing import StandardScaler
8 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
9 from sklearn.ensemble import RandomForestClassifier
10 from sklearn.metrics import confusion_matrix
11 from sklearn.metrics import accuracy_score
12
13 # Importing Iris Dataset
14 dataset = pd.read_csv('path/to/data/file.csv')
15
16 # Dividing into features and labels
17 X = dataset.iloc[:, 0:4].values
18 y = dataset.iloc[:, 4].values
19
20 # Train Test Split
21 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
22
23 # Feature Scaling
24 sc = StandardScaler()
25 X_train = sc.fit_transform(X_train)
26 X_test = sc.transform(X_test)
27
28 # Performing LDA
29 lda = LDA(n_components=1)
30 X_train = lda.fit_transform(X_train, y_train)
31 X_test = lda.transform(X_test)
32
33 # Training and making predictions
34 classifier = RandomForestClassifier(max_depth=2, random_state=0)
35 classifier.fit(X_train, y_train)
36 y_pred = classifier.predict(X_test)
37
38 # Evaluating Performance
39 cm = confusion_matrix(y_test, y_pred)
40 print(cm)
41 print('Accuracy :', str(accuracy_score(y_test, y_pred)))
```

(<https://www.learnbay.co/data-science-course/wp-content/uploads/2020/05/10th-akriti.png>)

92. What is GDA?

Ans: When we have a classification problem in which the input features are continuous random variable, we can use GDA, it's a generative learning algorithm in which we assume $p(x|y)$ is distributed according to a **multivariate normal distribution** and $p(y)$ is distributed according to **Bernoulli**.

Gaussian discriminant analysis (GDA) is a generative model for classification where the distribution of each class is modeled as a multivariate Gaussian.

93. What are the advantages and disadvantages of Dimensionality Reduction?

Ans: Advantages:

- Dimensionality Reduction helps in data compression, and hence reduced storage space.
- It reduces computation time.
- It also helps remove redundant features, if any.
- Dimensionality Reduction helps in data compressing and reducing the storage space required
- It fastens the time required for performing the same computations.
- If there present fewer dimensions then it leads to less computing. Also, dimensions can allow the usage of algorithms unfit for a large number of dimensions.
- It takes care of multicollinearity that improves model performance. It removes redundant features. For example, there is no point in storing a value in two different units (meters and inches).
- Reducing the dimensions of data to 2D or 3D may allow us to plot and visualize it precisely. You can then observe patterns more clearly.

Disadvantages:

- Basically, it may lead to some amount of data loss.
- Although, PCA tends to find linear correlations between variables, which is sometimes undesirable.
- Also, PCA fails in cases where mean and covariance are not enough to define datasets.
- Further, we may not know how many principal components to keep- in practice, some thumb rules are applied.





Learnbay (<https://www.learnbay.co/data-science-course/>) provides industry accredited data science courses

(<https://www.learnbay.co/data-science-course/>) in Bangalore. We understand the conjugation of technology in the field of Data science hence we offer significant courses like Machine learning, Tensor Flow, IBM Watson, Google Cloud platform, Tableau, Hadoop, time series, R, and Python. With authentic real-time industry projects. Students will be efficient by being certified by IBM. Around hundreds of students are placed in promising companies for data science roles. Choosing Learnbay you will reach the most aspiring job of present and future.

Learnbay data science course covers Data Science with Python, Artificial Intelligence with Python, Deep Learning using Tensor-Flow. These topics are covered and co-developed with IBM.



CATEGORIES:

SHARE THIS POST  (HTTPS://PINTEREST.COM/PIN/CREATE/BUTTON/?
URL=HTTPS://WWW.LEARNBAY.CO/DATA-SCIENCE-COURSE/BLOG-
POST/TOP-50-INTERVIEW-QUESTIONS-OF-MACHINE-
LEARNING/&MEDIA=HTTPS://WWW.LEARNBAY.CO/DATA-SCIENCE-
COURSE/WP-CONTENT/UPLOADS/2020/05/FIRST-
BLOG.JPG&DESCRIPTION=TOP%2050%20INTERVIEW%20QUESTIONS%20OF%20M
 (HTTPS://TWITTER.COM/SHARE?
URL=HTTPS://WWW.LEARNBAY.CO/DATA-SCIENCE-COURSE/BLOG-
POST/TOP-50-INTERVIEW-QUESTIONS-OF-MACHINE-
LEARNING/&TEXT=TOP%2050%20INTERVIEW%20QUESTIONS%20OF%20MACHINE?
 (HTTPS://WWW.FACEBOOK.COM/SHARER.PHP?
U=HTTPS://WWW.LEARNBAY.CO/DATA-SCIENCE-COURSE/BLOG-
POST/TOP-50-INTERVIEW-QUESTIONS-OF-MACHINE-
LEARNING/&IMAGES=HTTPS://WWW.LEARNBAY.CO/DATA-SCIENCE-
COURSE/WP-CONTENT/UPLOADS/2020/05/FIRST-BLOG.JPG) 
(HTTPS://PLUS.GOOGLE.COM/SHARE?
URL=HTTPS://WWW.LEARNBAY.CO/DATA-SCIENCE-COURSE/BLOG-
POST/TOP-50-INTERVIEW-QUESTIONS-OF-MACHINE-
LEARNING/&TITLE=TOP%2050%20INTERVIEW%20QUESTIONS%20OF%20MACHINE'

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Name *

Email *

Website *

Comment

POST COMMENT



COPYRIGHT © 2019 LEARNBAY . ALL RIGHTS RESERVED

