Supervised vs Unsupervised Learning

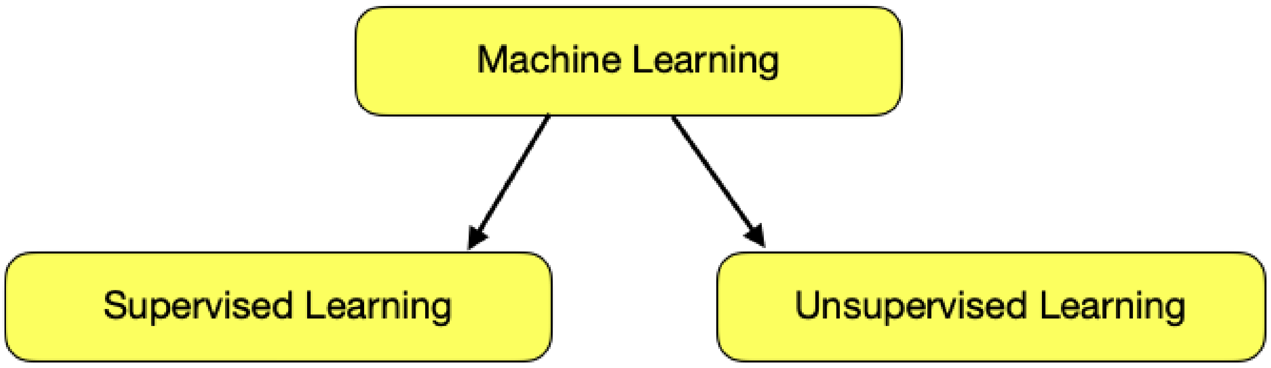
Posted in Machine Learning (https://hackr.io/blog/category/machine-learning-ml)

Simran Kaur Arora
(https://hackr.io/blog/author/simran-arora)
Last Updated 17 Dec, 2019

Share:
(https://twitter.com/intent/tweet?text=Supervised+vs+Unsupervised+Learning+https%3A%2F%2Fhackr.io%2Fblog%2Fsupervised-vs-unsupervised-learning) (https://www.linkedin.com/shareArticle?mini=true&url=https://hackr.io/blog/supervised-vs-unsupervised-learning) (http://www.reddit.com/submit?url=https://hackr.io/blog/supervised-vs-unsupervised-learning) (https://news.ycombinator.com/submitlink?u=https://hackr.io/blog/supervised-vs-unsupervised-learning) (https://api.whatsapp.com/send?text=https%3A%2F%2Fhackr.io%2Fblog%2Fsupervised-vs-unsupervised-learning)

Table of Contents

The most successful kinds of machine learning algorithms are those that automate decision-making processes by generalizing from known examples. In this article, we discuss the two forms of machine learning i.e. Supervised learning, Unsupervised Learning, and comparison between these two.

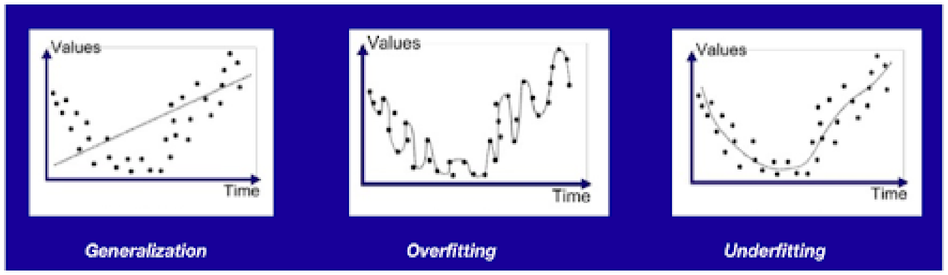


Supervised Learning

In supervised learning, we provide the algorithm with pairs of inputs and desired outputs by the user, to find a way to produce the desired output given an input. This learning is one of the most commonly used and successful types of machine learning (https://hackr.io/blog/what-is-machine-learning-definition-types#types-of-machine-learning) and is used whenever when it is required to predict a specific outcome from a given input, and we have samples of input/output pairs. A machine learning model from these input/output pairs is built, which comprises of the training set. The goal is to make accurate predictions for new, never-before-seen data. Supervised learning often requires human effort to build the training set, but afterward automates and often speeds up an otherwise laborious or infeasible task.

For example, a spam classification uses machine learning. The algorithm is provided with a large number of emails (which are the input) by the user. Information about whether these emails are spam (which is the desired output) is also provided with the input. The algorithm is then given a new email to predict as to whether the new email is spam.

Generalization, Overfitted, and Underfitted



1. Generalization

In supervised learning, we want to build a model on the training data and then be able to make accurate predictions on new, unseen data that has the same characteristics a the training set that we used. If a model is successful in making accurate predictions n the unseen data, we say that it can generalize from the training set to the test set.

2. Overfitted

The only measure of whether an algorithm performs well on new data is the evaluation on the test set. Building a model that is too complex for the amount of information we have is overfitting. Overfitting occurs when a model is fit too closely to the particularities of the training set and obtain a model that works well on the training set but is not able to generalize to new data.

3. Underfitted

If the model is too simple, then it would be challenging to capture all aspects and variability in the data, and thus the model performs poorly on the training set. Choosing a too simple model is underfitting.

>h PLUS Learn programming from the world's best Educators JOIN NOW

(https://hackr.io/hackr-plus?ref=blog)

Relation of Model Complexity to Data Size

Model complexity is tied to the variations of inputs contained in the training data set: the larger variety of data points a data set contains, the more complex a model can be used without overfitting. Collecting more data points yields more variety, so larger datasets allow building more complex models.

Supervised Machine Learning Algorithms

Let us now see the most popular supervised machine learning algorithms and explain how they learn from data and how they make predictions. We examine the strengths, weaknesses of algorithms, and what kind of data they can best be applied.

1. k-Nearest Neighbor

The k-NN algorithm is arguably the most straightforward machine learning algorithm. Building the model comprises only of storing the training dataset. The algorithm finds the closest data points in the training dataset—its "nearest neighbors" to predict a new data point.

Strengths, weaknesses, and parameters

1. Strength

One of the advantages of k-NN is that the model is easy to understand and often gives a reasonable performance with a lot of adjustments.

2. Weakness

When using the k-NN algorithm, it's essential to preprocess the

data. The approach does not perform well on data sets on with many features, and it shows particularly bad with datasets where most features are 0 most of the time.

3. Parameters

There are two critical parameters to the k-NN classifier: the number of neighbors and how to measure the distance between data points.

2. Linear Models

Linear models make a prediction using the linear function of input features.

There are different models for regression, let us have a brief overview below:

1. Linear Regression

Linear regression, or ordinary least squares (OLS), is the simplest and most classic linear method for regression. Linear regression finds the parameters that minimize the mean squared error between predictions and the right regression targets, y, on the training set. The mean squared error is the sum of the squared differences between the predictions and the true values. Linear regression has no parameters, which is a benefit, but it also has no way to control model complexity.

2. Ridge Regression

In ridge regression, the coefficients are chosen to fit an additional constraint and not only to predict well on the training data. We also want the magnitude of coefficients to be as small as possible. Intuitively, this means each feature should have as little effect on the outcome as possible (which translates to having a slight slope), while still predicting well. This constraint is an example of what is called regularization. Regularization means explicitly restricting a model to avoid overfitting. The particular kind used by ridge regression is known as L2 regularization.

3. Lasso

Lasso is an alternative to Ridge for regularizing linear regression. Using lasso restricts coefficients to be close to zero, but in a slightly different way, called L1 regularization. The consequence of L1 regularization is that when using the lasso, some coefficients are precisely zero. It means some features are entirely ignored by the model. Having some coefficients be exactly zero often makes a model easier to interpret, and can reveal the essential features of the model.

Strengths, weaknesses, and parameters

1. Strength

- Linear models are very fast to train, and also fast to predict. They scale to very large datasets and work well with sparse data.
- Another strength of linear models is that they make it relatively easy to understand how a prediction is made, using the formulas we saw earlier for regression and classification.

2. Weakness

It is not clear why coefficients are the way they are. It is particularly true if the dataset has highly correlated features; in these cases, the coefficients might be hard to interpret.

Linear models often perform well when the number of features is large compared to the number of samples. They are also often used on massive datasets, simply because it's not feasible to train other models. However, in lower-dimensional spaces, other models might yield better generalization performance.

3. Parameters

The main parameter of linear models is the regularization parameter, called alpha in the regression models and C in LinearSVC and LogisticRegression. Large values for alpha or small amounts for C mean simple models.

3. Naive Bayes Classifier

Naive Bayes classifiers are quite similar to the linear models. They are even faster in training, but in return, naive Bayes models often provide generalization performance though it is slightly worse than that of linear classifiers like LogisticRegression and LinearSVC.

The reason that naive Bayes models learn parameters by looking at each feature individually and collect simple per-class statistics from each feature, thus making the model efficient. GaussianNB, BernoulliNB, and MultinomialNB are three kinds of naive Bayes classifiers implemented in sci-kit learn. GaussianNB is applied to any continuous data, whereas BernoulliNB assumes binary data, and MultinomialNB assumes count data (like how often a word appears in a sentence). Text data classification uses BernoulliNB and MultinomialNB.

Strengths, weaknesses, and parameters

1. Strength

They are very fast to train and to predict, and the training procedure is easy to understand. The models work very well with high-dimensional sparse data and are relatively robust to the parameters. Naive Bayes models are great baseline models and are often used on massive datasets, where training even a linear model might take too long.

2. Weakness

The weaknesses of the Naive Bayes Classifier share with that of a linear model.

3. Parameters

BernoulliNB and MultinomialNB have a single parameter, alpha, which controls model complexity. The way alpha works are that the algorithm adds to the data alpha, many virtual data points that have positive values for all the features. This results in a “smoothing” of the statistics. A significant alpha means more smoothing, resulting in less sophisticated models. The algorithm’s performance is relatively robust to the setting of alpha, meaning that setting alpha is not critical for an excellent performance. However, tuning it usually improves accuracy somewhat.

GaussianNB is mostly used on very high-dimensional data, while the other two variants of naive Bayes are widely used for sparse count data such as text. MultinomialNB usually performs better than BinaryNB, particularly on datasets with a relatively large number of nonzero features (i.e., large documents).

4. Decision Trees

Decision trees are widely used models for classification and regression tasks. Essentially, they learn a hierarchy of if/else questions, leading to a decision.

These questions are similar to the questions one might ask in a game of 20 Questions. Imagine a user who wants to distinguish between the following four animals: bears, hawks, penguins, and dolphins. The goal is to get to the right answer by asking as few if/else questions as possible. The user might start by asking whether the animal has feathers, a question that narrows down the possible animals to just two. If the answer is “yes,” he can ask another question that could help him distinguish between hawks and penguins. For example, one could ask whether the animal can fly. If the animal doesn’t have feathers, the possible animal choices are dolphins and bears, and he needs to ask a question to distinguish between these two animals—for example, asking whether the animal has fins.

Strengths, weaknesses, and parameters

1. Strength

- The resulting model can easily be visualized and understood by nonexperts (at least for smaller trees), and the algorithms are entirely invariant to scaling of the data.
- No preprocessing like standardization or normalization of features is needed for decision tree algorithms since all features are processed separately, and the possible splits of the data don’t depend on scale.

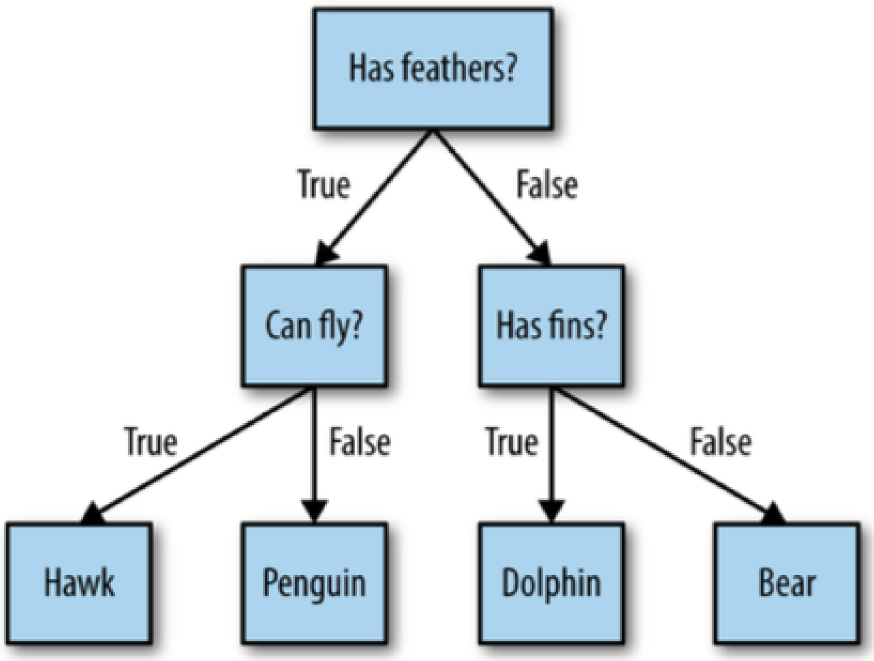
2. Weakness

The main downside of decision trees is that despite the use of pre-pruning, they tend to overfit and provide poor generalization performance. Therefore, in most applications, the ensemble methods we discuss next are usually used in place of a single decision tree.

3. Parameters

The pre-pruning parameters control model complexity in decision trees that stop the building of the tree before it is fully developed.

Picking one of the pre-pruning strategies—setting either max_depth, max_leaf_nodes, or min_samples_leaf—is sufficient to prevent overfitting.



In machine learning parlance, we built a model to distinguish between four classes of animals (hawks, penguins, dolphins, and bears) using the three features “has feathers,” “can fly,” and “has fins.” Instead of building these models by hand, we can learn them from data using supervised learning.

5. Neural Networks

A family of algorithms known as neural networks has recently seen a revival under the name “deep learning.” While deep learning shows great promise in many machine learning applications, deep learning algorithms are often tailored very carefully to a specific use case. Here, we discuss some relatively simple methods, namely multilayer perceptrons for classification and regression, that can serve as a starting point for more involved deep learning methods.

A family of algorithms known as neural networks has recently seen a revival under the name “deep learning.” While deep learning shows great promise in many machine learning applications, deep learning algorithms are often tailored very carefully to a specific use case. There are relatively simple methods, namely multilayer perceptrons for classification and regression, that serves as a starting point for more involved deep learning methods.

The Neural Network Model

1. Tuning Neural Networks

Strengths, weaknesses, and parameters

1. Strength

The main advantages are that they are able to capture information contained in large amounts of data and build incredibly complex models.

2. Weakness

Neural networks, particularly the large and powerful ones, often take a long time to train. They also require careful preprocessing of the data, as we saw here.

3. Parameters

Tuning neural network parameters is also an art unto itself. Similarly to SVMs, they work best with “homogeneous” data, where all the features have similar meanings. For data that has very different kinds of features, tree-based models might work better.

Unsupervised Learning

Unsupervised learning includes all kinds of machine learning where there is no known output, no teacher to instruct the learning algorithm. In unsupervised learning, the learning algorithm is just shown the input data and asked to extract knowledge from this data.

Types of Unsupervised Learning

1. Transformation of the Data Set

Unsupervised transformations of a dataset create a new representation of the data, which might be easier for humans or other machine learning algorithms to understand compared to the first representation of the data. A typical application of unsupervised transformations is dimensionality reduction, which takes a high-dimensional representation of the data, consisting of many features, and finds a new way to represent this data that summarizes the essential characteristics with fewer features. A typical application for dimensionality reduction is a reduction in two dimensions for visualization purposes.

Another application for unsupervised transformations is finding the parts or components that “make up” the data. An example of this is topic extraction on collections of text documents. The task is to find the unknown topics that are talked about in each document and to learn what topics appear in each document. It is useful for tracking the discussion of themes like elections, gun control, or pop stars on social media.

2. Clustering

Clustering algorithms, on the other hand, partition data into distinct groups of similar items. Consider the example of uploading photos to a social media site. To allow you to organize your pictures, the site might want to group together pictures that show the same person. However, the site doesn’t know which pictures show whom, and it doesn’t know how many different people appear in your photo collection. A sensible approach would be to extract all the faces and divide them into groups of faces that look similar. Hopefully, these correspond to the same person, and the images can be grouped together for you.

Challenges in Unsupervised Learning

A significant challenge in unsupervised learning is evaluating whether the algorithm learned something useful. Unsupervised learning algorithms are applied to data that does not contain any label information, so we don’t know what the correct output should be. Therefore, it is tough to say whether a model “did well.” For example, our hypothetical clustering algorithm could have grouped all the pictures that show faces in profile and all the full-face pictures. It would certainly be a possible way to divide a collection of pictures of people’s faces, but it’s not the one we were looking for. However, there is no way for us to “tell” the algorithm what we are looking for, and often the only way to evaluate the result of an unsupervised algorithm is to inspect it manually.

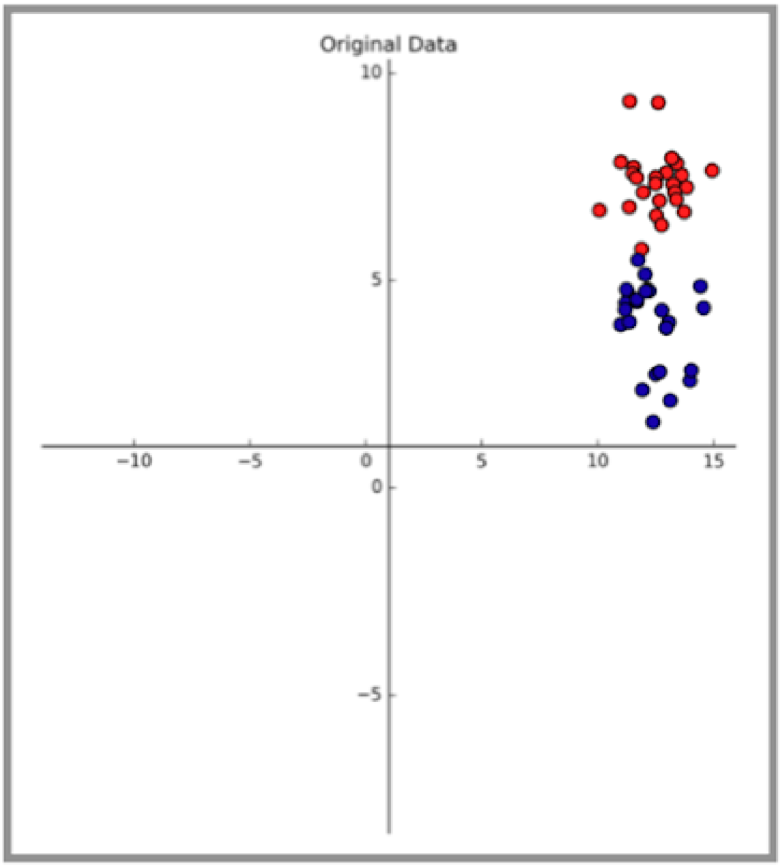
As a consequence, unsupervised algorithms are used often in an exploratory setting, when a data scientist wants to understand the data better, rather than as part of a more extensive automatic system.

Another common application for unsupervised algorithms is as a preprocessing step for supervised algorithms. Learning a new representation of the data can sometimes improve the accuracy of supervised algorithms, or can lead to reduced memory and time consumption.

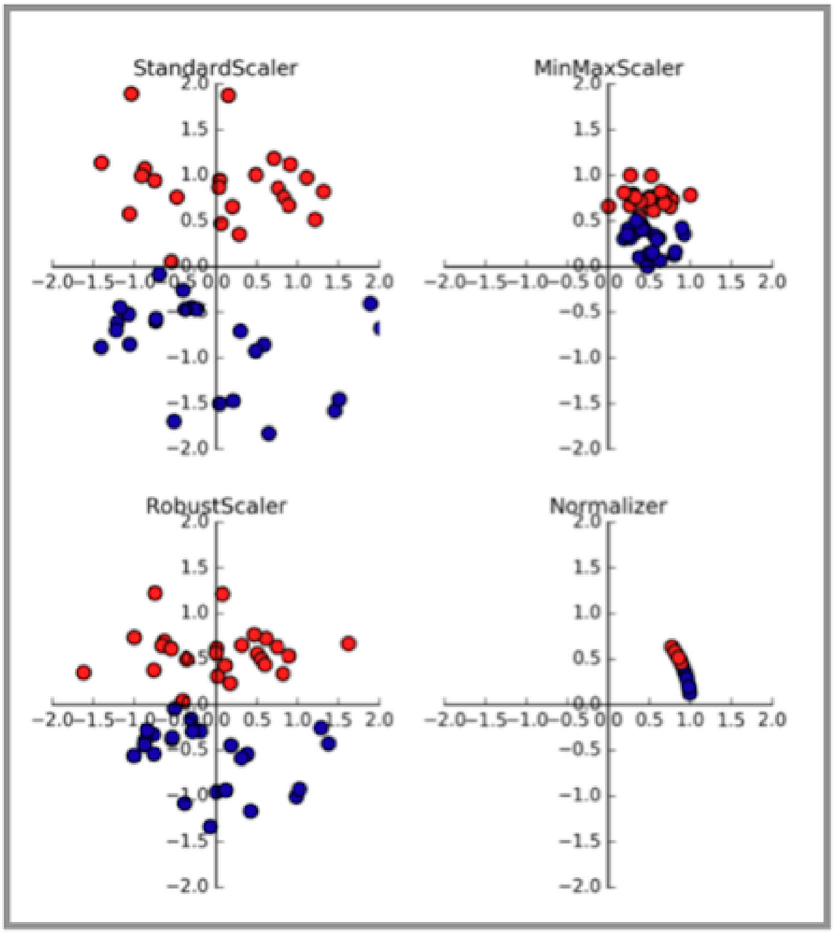
Before we start with “real” unsupervised algorithms, we briefly discuss some simple preprocessing methods that often come in handy. Even though preprocessing and scaling are often used in tandem with supervised learning algorithms, scaling methods don’t make use of the supervised information, making them unsupervised.

Preprocessing and Scaling

The first plot in Figure shows a synthetic two-class classification dataset with two features. The first feature (the x-axis value) is between 10 and 15. The second feature (the y-axis value) is between around 1 and 9.



The following four plots show four different ways to transform the data that yield more standard ranges.



The **StandardScaler** ensures that for each feature, the mean is 0, and the variance is 1 by bringing all features to the same magnitude. However, this scaling does not ensure any particular minimum and maximum values for the features.

The **RobustScaler** works similarly to the StandardScaler in that it ensures statistical properties for each feature that guarantees that they are on the same scale. However, the RobustScaler uses the median and quartiles,1 instead of mean and variance. It makes the RobustScaler ignore data points that are very different from the rest. These unique data points are also called outliers and can lead to trouble for other scaling techniques.

The **MinMaxScaler** shifts the data such that all features are precisely between 0 and 1 for the two-dimensional dataset. It means all the data is contained within the rectangle created by the x-axis between 0 and 1 and the y-axis between 0 and 1.

Finally, the **Normalizer** scales each data point such that the feature vector has a Euclidean length of 1. In other words, it projects a data point on the circle or sphere with a radius of 1. It means a different number of scales every data point. This normalization is used when only the direction (or angle) of the data matters, not the length of the feature vector.

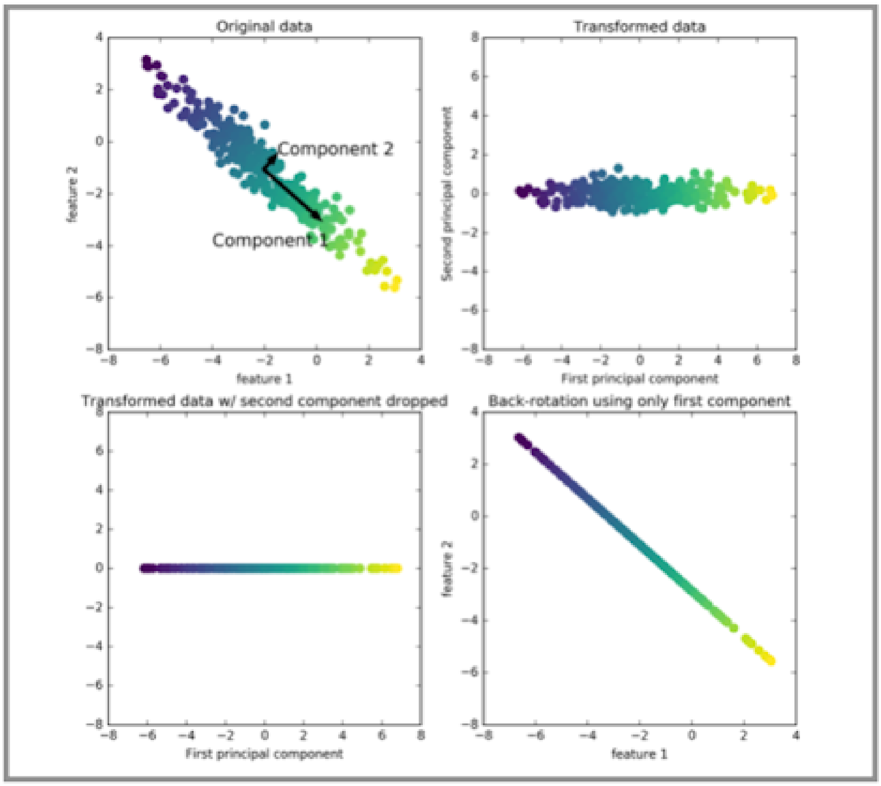
Unsupervised Algorithms

Transforming data using unsupervised learning can have many motivations. The most common motivations are visualization, compressing the data, and finding a representation that is more informative for further processing.

One of the simplest and most widely used algorithms for all of these is principal component analysis. We'll also look at two other algorithms: non-negative matrix factorization (NMF), used for feature extraction, and t-SNE, used for visualization using two-dimensional scatter plots.

1. Principal Component Analysis (PCA)

The principal component analysis is a method that rotates the dataset in a way such that the rotating features are statistically uncorrelated. This rotation followed by selecting only a subset of the new features, according to how important they are for explaining the data. The following example illustrates the effect of PCA on a synthetic two-dimensional dataset:



The first plot (top left) shows the original data points, colored to distinguish among them. The algorithm proceeds by first finding the direction of maximum variance, labeled "Component 1." It is the direction (or vector) in the data that contains most of the information, or the direction along which the features are most correlated with each other. Then, the algorithm finds the direction that contains the most information while being orthogonal (at a right angle) to the first direction. In two dimensions, there is only one possible orientation that is at a right angle, but in higher-dimensional spaces, there would be (infinitely) many orthogonal directions. Although the two components are drawn as arrows, it doesn't matter where the head and the tail are; we could have drawn the first component from the center up to the top left instead of down to the bottom right. The directions found using this process are called principal components, as they are the main directions of variance in the data. In general, there are as many principal components as original features.

The second plot (top right) shows the same data but now rotated so that the first principal component aligns with the x-axis, and the second principal component aligns with the y-axis. Before the rotation, the mean was subtracted from the data, so that the transformed data is centered around zero. In the rotated representation found by PCA, the two axes are uncorrelated, meaning that the correlation matrix of the data in this representation is zero except for the diagonal.

We use PCA for dimensionality reduction by retaining only some of the principal components. In this example, we might keep only the first principal component, as shown in the third panel in Figure (bottom left). It reduces the data from a two-dimensional dataset to a one-dimensional dataset. Note, however, that instead of keeping only one of the original features, we found the most exciting direction (top left to bottom right in the first panel) and kept this direction, the first principal component.

Finally, we can undo the rotation and add the mean back to the data. It results in the data shown in the last panel in Figure. These points are in the original feature space, but we kept only the information contained in the first principal component. This transformation is sometimes used to remove noise effects from the data or visualize what part of the information is retained using the principal components.

2. Non-Negative Matrix Factorization

Non-negative matrix factorization is another unsupervised learning algorithm that aims to extract useful features. It works similarly to PCA and can also be used for dimensionality reduction. As in PCA, we are trying to write each data point as a weighted sum of some components, as illustrated above. But whereas in PCA we wanted orthogonal components and that explained as much variance of the data as possible, in NMF, we want the components and the coefficients to be non-negative; that is, we want both the components and the coefficients to be greater than or equal to zero. Consequently, this method can be applied to data where each feature is non-negative, as a non-negative sum of non-negative components cannot become negative.

The process of decomposing data into a non-negative weighted sum is particularly helpful for data that is created as the addition (or overlay) of several independent sources, such as an audio track of multiple people speaking, or music with many instruments. In these situations, NMF can identify the original components that make up the combined data. Overall, NMF leads to more interpretable components than PCA, as negative components and coefficients can lead to hard-to-interpret cancellation effects. The eigenfaces in Figure, for example, contain both positive and negative parts, and as we mentioned in the description of PCA, the sign is arbitrary. Before we apply NMF to the face dataset, let's briefly revisit the synthetic data.

3. t-SNE

While PCA is often an excellent first approach for transforming your data so that you might be able to visualize it using a scatter plot, the nature of the method (applying rotation and then dropping directions) limits its usefulness, as we saw with the scatter plot of the Labeled Faces in the Wild dataset. There is a class of algorithms for visualization called manifold learning algorithms that allow for much more complex mappings and often provide better visualizations. A particularly useful one is the t-SNE algorithm.

Manifold learning algorithms aim at visualization, and so are rarely used to generate more than two new features. Some of them, including t-SNE, compute a new representation of the training data but don't allow transformations of new data. It means these algorithms can only transform the data they were trained for and cannot apply to a test set. Manifold learning can be useful for exploratory data analysis but is rarely used if the final goal is supervised learning. The idea behind t-SNE is to find a two-dimensional representation of the data that preserves the distances between points in the best possible way. t-SNE starts with a random two- dimensional representation for each data point and then tries to make points that are close in the original feature space closer and points that are far apart in the original feature spaced farther apart. t-SNE puts more emphasis on points that are close by, rather than preserving distances between far-apart points. In other words, it tries to preserve the information indicating which points are neighbors to each other.

4. Clustering

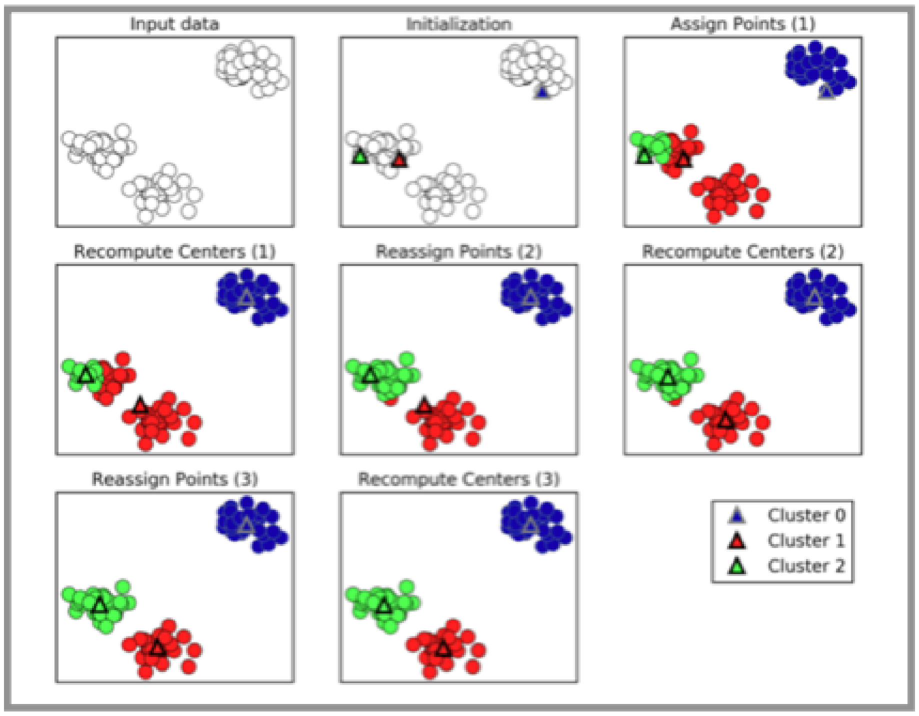
As described earlier, clustering is the task of partitioning the dataset into groups, called clusters. The goal is to split up the data in such a way that points within a single cluster are very similar and points in different clusters are different. Similarly to classification algorithms, clustering algorithms assign (or predict) a number to each data point, indicating which cluster a particular point belongs to.

1. k-Means Clustering

k-means clustering is the simplest and most commonly used clustering algorithms. It tries to find cluster centers that are representative of certain regions of the data. The algorithm alternates between two steps:

- Assigning each data point to the closest cluster center and
- Then setting each cluster center as the mean of the data points that are assigned to it.

The algorithm is finished when the assignment of instances to clusters no longer changes. The following example illustrates the algorithm on a synthetic dataset:

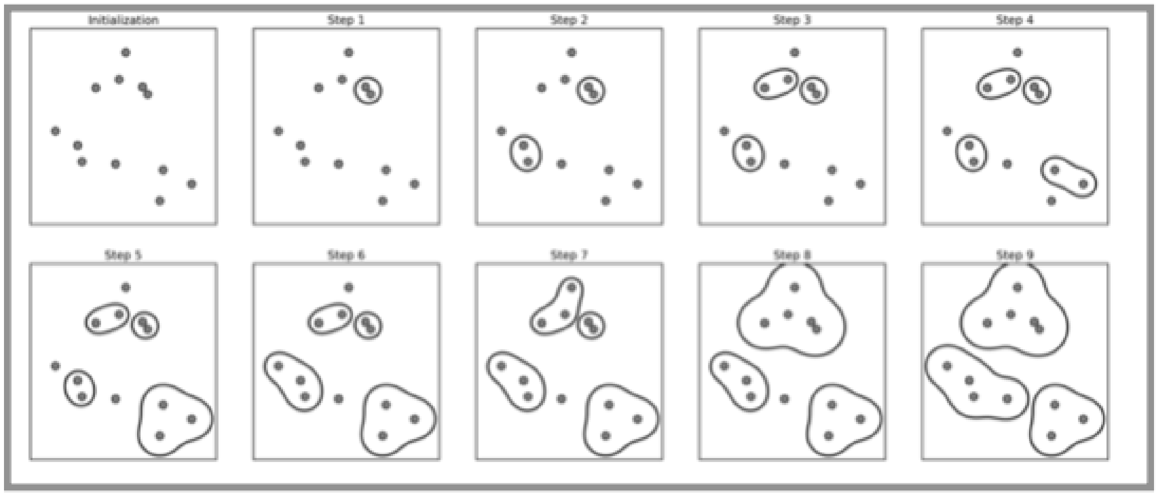


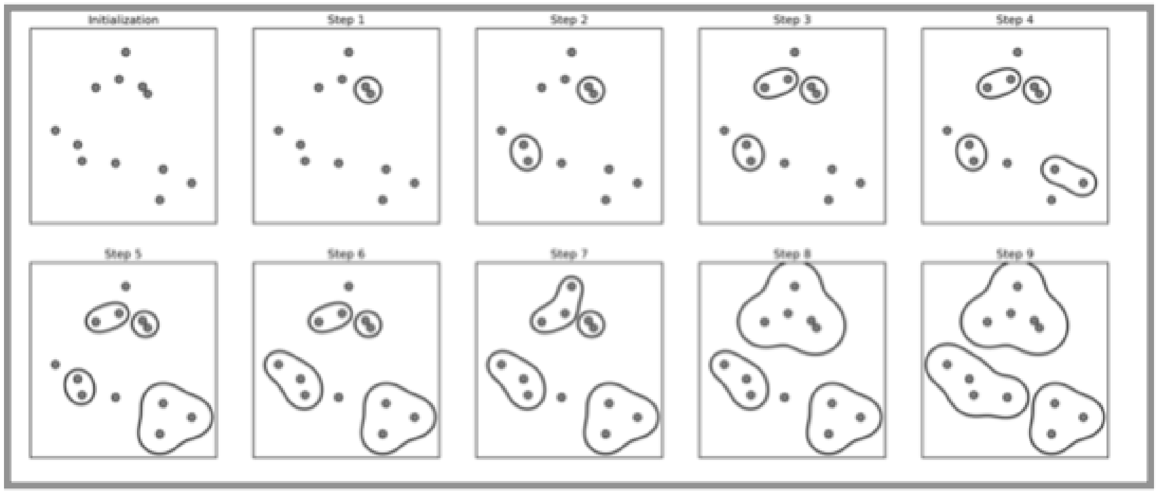
Cluster centers are shown as triangles, while data points are shown as circles. Colors indicate cluster membership. It is specified that the goal is to look for three clusters, so the algorithm is initialized by declaring three data points randomly as cluster centers. Then the iterative algorithm starts. First, each data point is assigned to the cluster center it is closest to (see "Assign Points (1)"). Next, the cluster centers are updated to be the mean of the assigned points (see "Recompute Centers (1)"). Then the process is repeated two more times. After the third iteration, the assignment of points to cluster centers remained unchanged, so the algorithm stops.

2. Agglomerative Clustering

Agglomerative clustering refers to a collection of clustering algorithms that all build upon the same principles: the algorithm starts by declaring each point its cluster and then merges the two most similar clusters until some stopping criterion is satisfied. The stopping criterion implemented in scikit-learn is the number of clusters, so similar clusters are merged until only the specified number of clusters are left. Several linkage criteria specify how exactly the "most similar cluster" is measured. This measure is always defined between two existing clusters.

The following plot illustrates the progression of agglomerative cluster- ing on a two-dimensional dataset, looking for three clusters:





Initially, each point is its cluster. Then, in each step, the two closest clusters are merged. In the first four steps, two single-point clusters are picked, and these are joined into two-point clusters. In step 5, one of the two-point clusters is extended to a third point, and so on. In step 9, only three clusters are remaining. As we specified that we are looking for three clusters, the algorithm then stops.

3. DBSCAN

Another advantageous clustering algorithm DBSCAN (Density-Based Spatial Clustering of Applications with Noise").

DBSCAN works by identifying points that are in "crowded" regions of the feature space, where many data points are close together. These regions are referred to as dense regions in feature space. The idea behind DBSCAN is that clusters form dense regions of data, separated by relatively empty regions.

Core samples are defined as the points that are within a dense region are called, and they are defined as follows. DBSCAN has two parameters: eps and min_samples. If there are at min_samples many data points within a distance of eps to a given data point, that data point is classified as a core sample. Core samples closer to each other than the distance eps are put into the same cluster by DBSCAN.

The algorithm works by picking an arbitrary point. It then finds all points with distance eps or less from that point. If there are less than min_samples points within distance eps of the starting point, this point is labeled as noise, meaning that it doesn't belong to any cluster. If there are more than min_samples points within a distance of eps, the point is labeled a core sample and assigned a new cluster label. Then, all neighbors (within eps) of the point are visited. If they have not been assigned a cluster yet, they are assigned the new cluster label that was just created. If they are core samples, their neighbors are visited in turn, and so on. The cluster grows until there are no more core samples within distance eps of the cluster. Then another point that hasn't yet been visited is picked, and the same procedure is repeated.

In the end, there are three kinds of points: core points, points that are within distance eps of core points (called boundary points), and noise. When the DBSCAN algorithm is run on a particular dataset multiple times, the clustering of the core points is always the same, and the same points always are labeled as noise. However, a boundary point might be neighbor to core samples of more than one cluster. Therefore, the cluster membership of boundary points depends on the order in which points are visited. Usually, there are only a few boundary points, and this slight dependence on the order of points is not essential.

The main benefits of DBSCAN are that it does not require the user to set the number of clusters a priori, it can capture clusters of complex shapes, and it can identify points that are not part of any cluster. DBSCAN is somewhat slower than agglomerative clustering and k-means but still scales to relatively large datasets.

Supervised vs Unsupervised Learning: Head to Head Comparison

Parameters	**Supervised Learning**	**Unsupervised Learning**
Definition	Method in which the machine is taught using labelled data.	The machine is trained on unlabelled data without any guidance. Machine should discover hidden patterns in the data.

Problem Type	• Regression: Predicts continuous quantity. • Classification: Predicting a label or class.	• Association: Involves discovering patterns and finding co-occurrences. • Clustering: Used in targeted marketing.
Type of Data	Provided with a labelled set of input and output data.	The machine is only has given the input data(unlabelled).
Training Phase	The training phase is well defined and explicit.	The training phase is vague.
Aim	Forecast outcomes- give predicted outcomes.	Discover patterns and extract useful insights.
Approach	Map the known input to the known output.	Explore and find patterns and extract useful insights.
Output Feedback	Direct feedback mechanism as the machine is trained with labeled data.	No feedback mechanism because the machine is unaware of output during the training phase.
Popular Algorithm	Linear Regression k-Nearest Neighbor	K-Means C-Means
Applications	The business sector for forecasting risks	Recommendation system

Summary

A supervised learning algorithm is trained using labeled data and helps to predict outcomes for unforeseen data.

Successfully building, scaling, and deploying accurate supervised machine learning models takes time and technical expertise from a team of highly skilled data scientists. Moreover, Data scientist (https://hackr.io/blog/how-to-become-a-data-scientist) must rebuild models to make sure the insights given remains true until its data changes. Popular algorithms include linear regression and k-nearest neighbors with applications in the business sector to predict useful insights for the growth of the organization.

On the other hand, Unsupervised learning algorithms use unlabeled data and allow them to perform more complex processing tasks compared to supervised learning, although unsupervised learning can be more unpredictable compared with other natural learning methods. Popular algorithms are K-means and C-means with applications such as recommendation systems.

People are also reading:

- What is Machine Learning? (https://hackr.io/blog/what-is-machine-learning-definition-types)
- Top 10 Machine Learning Certifications (https://hackr.io/blog/machine-learning-certifications)
- What is Data Analysis? Methos, Techniques, and Tools (https://hackr.io/blog/what-is-data-analysis-methods-techniques-tools)
- Top 10 Artificial Intelligence Books (https://hackr.io/blog/artificial-intelligence-books)
- Difference between Pytorch vs Tensorflow (https://hackr.io/blog/pytorch-vs-tensorflow)
- How to become a data engineer? (https://hackr.io/blog/how-to-become-a-data-engineer)
- What is Hadoop Architecture? (https://hackr.io/blog/hadoop-architecture)
- Top 10 Data Analytics Courses (https://hackr.io/blog/data-analytics-courses)
- Difference between r vs Python (https://hackr.io/blog/r-vs-python)
- Top Apache Spark Interview Questions (https://hackr.io/blog/apache-spark-interview-questions)
- Top Deep Learning Books (https://hackr.io/blog/deep-learning-books)

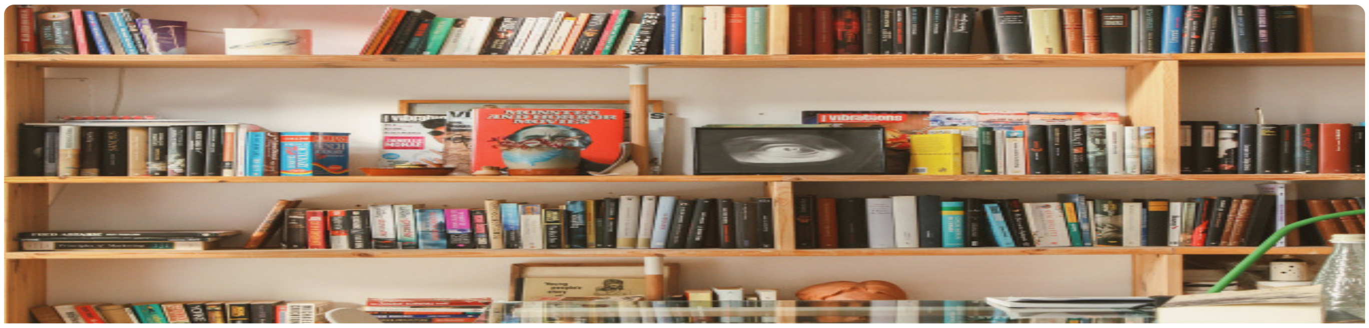
Machine Learning (https://hackr.io/blog/tag/machine-learning)
Supervised Learning (https://hackr.io/blog/tag/supervised-learning)
Unsupervised Learning (https://hackr.io/blog/tag/unsupervised-learning)

Share:
(https://twitter.com/intent/tweet?text=Supervised+vs+Unsupervised+Learning+https%3A%2F%2Fhackr.io%2Fblog%2Fsupervised-vs-unsupervised-learning) (https://www.linkedin.com/shareArticle?mini=true&url=https://hackr.io/blog/supervised-vs-unsupervised-learning) (http://www.reddit.com/submit?url=https://hackr.io/blog/supervised-vs-unsupervised-learning) (https://news.ycombinator.com/submitlink?u=https://hackr.io/blog/supervised-vs-unsupervised-learning) (https://api.whatsapp.com/send?text=https%3A%2F%2Fhackr.io%2Fblog%2Fsupervised-vs-unsupervised-learning)

Simran Kaur Arora

(https://hackr.io/blog/author/simran-arora)
Simran, born in Delhi, did her schooling and graduation from India in Computer Science. Curious and passionate about technology urged her to study for an MS in the same from the renowned Silicon Valley, California, USA. Graduated in 2017, she flew back to India and now works for hackr.io as a freelance technical writer. View all posts by the Author (https://hackr.io/blog/author/simran-arora)

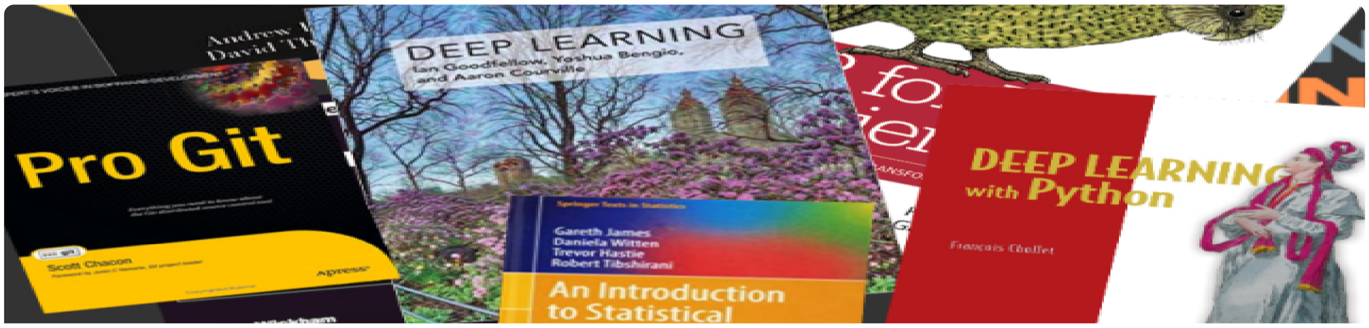
Related Posts


(https://hackr.io/blog/best-machine-learning-books)

20 Best Machine Learning Books for Beginner & Experts in 2020

(https://hackr.io/blog/best-machine-learning-books)

Read More (https://hackr.io/blog/best-machine-learning-books)


Pro Git
DEEP LEARNING
Ian Goodfellow, Yoshua Bengio, and Aaron Courville
DEEP LEARNING with Python
An Introduction to Statistical
(https://hackr.io/blog/deep-learning-books)

10 Best Deep Learning Books

(https://hackr.io/blog/deep-learning-books) **Read More (https://hackr.io/blog/deep-learning-books)**

(https://hackr.io/blog/machine-learning-certifications)

10 Best Machine Learning Certification for 2020 [Updated]

(https://hackr.io/blog/machine-learning-certifications)

Read More (https://hackr.io/blog/machine-learning-certifications)

Leave a comment

Email address*

Enter email

Your email will not be published

Name*

Name

Comment*

SUBMIT

Related Tutorials

Artificial Intelligence

(https://hackr.io/tutorials/learn-artificial-intelligence-ai)

Data Science

(https://hackr.io/tutorials/learn-data-science)

Hadoop

(https://hackr.io/tutorials/learn-hadoop-big-data)

Machine Learning

(https://hackr.io/tutorials/learn-machine-learning-ml)

Python

(https://hackr.io/tutorials/learn-python)

Recommended Learning

Machine Learning (Stanford University) (www.coursera.org)

(https://hackr.io/tutorial/machine-learning-stanford-university)

Machine Learning A-Z: Hands-On Python & R In Data Science (www.udemy.com)

(https://hackr.io/tutorial/machine-learning-a-z-hands-on-python-r-in-data-science)

Machine Learning with Python (www.youtube.com)

(https://hackr.io/tutorial/machine-learning-with-python)

VIEW MORE (https://hackr.io/tutorials/learn-machine-learning-ml)

Top Blogs

What is Data Science? (https://hackr.io/blog/what-is-data-science)
Programming Languages 2020 (https://hackr.io/blog/best-programming-languages-to-learn-2020-jobs-future)
Web Development Frameworks (https://hackr.io/blog/top-10-web-development-frameworks-in-2020)
Python Books (https://hackr.io/blog/best-python-books-for-beginners-and-advanced-programmers)
Numpy Matrix Multiplication (https://hackr.io/blog/numpy-matrix-multiplication)
Java IDE (https://hackr.io/blog/best-java-ides)
IoT Applications (https://hackr.io/blog/top-10-iot-applications)
Javascript Map (https://hackr.io/blog/javascript-map)
Features of Java (https://hackr.io/blog/features-of-java)
Normalization in DBMS (https://hackr.io/blog/dbms-normalization)

Interview Questions

Angular Interview Questions (https://hackr.io/blog/angular-interview-questions)
Python Interview Questions (https://hackr.io/blog/python-interview-questions)
C++ Interview Questions (https://hackr.io/blog/cpp-interview-questions)

Selenium Interview Questions (https://hackr.io/blog/selenium-interview-questions)
SQL Interview Questions (https://hackr.io/blog/top-sql-interview-questions)
Java Interview Questions (https://hackr.io/blog/java-interview-questions)
Javascript interview Questions (https://hackr.io/blog/javascript-interview-questions)
PHP Interview Questions (https://hackr.io/blog/php-interview-questions)
Spring Interview Questions (https://hackr.io/blog/spring-interview-questions)
AWS Interview Questions (https://hackr.io/blog/aws-interview-questions)

Top Courses/Certifications

Python Course (https://hackr.io/blog/best-python-courses)
Data Analytics Course (https://hackr.io/blog/data-analytics-courses)
Javascript Course (https://hackr.io/blog/best-javascript-courses)
Web Developer Course (https://hackr.io/blog/best-web-developer-courses-for-beginners)
C Course (https://hackr.io/blog/best-c-courses)
SQL Certification (https://hackr.io/blog/sql-server-certifications)
Java Certification (https://hackr.io/blog/java-certification-courses)
Python Certification (https://hackr.io/blog/python-certification)
Machine Learning Certification (https://hackr.io/blog/machine-learning-certifications)
AWS Certification (https://hackr.io/blog/aws-certifications)