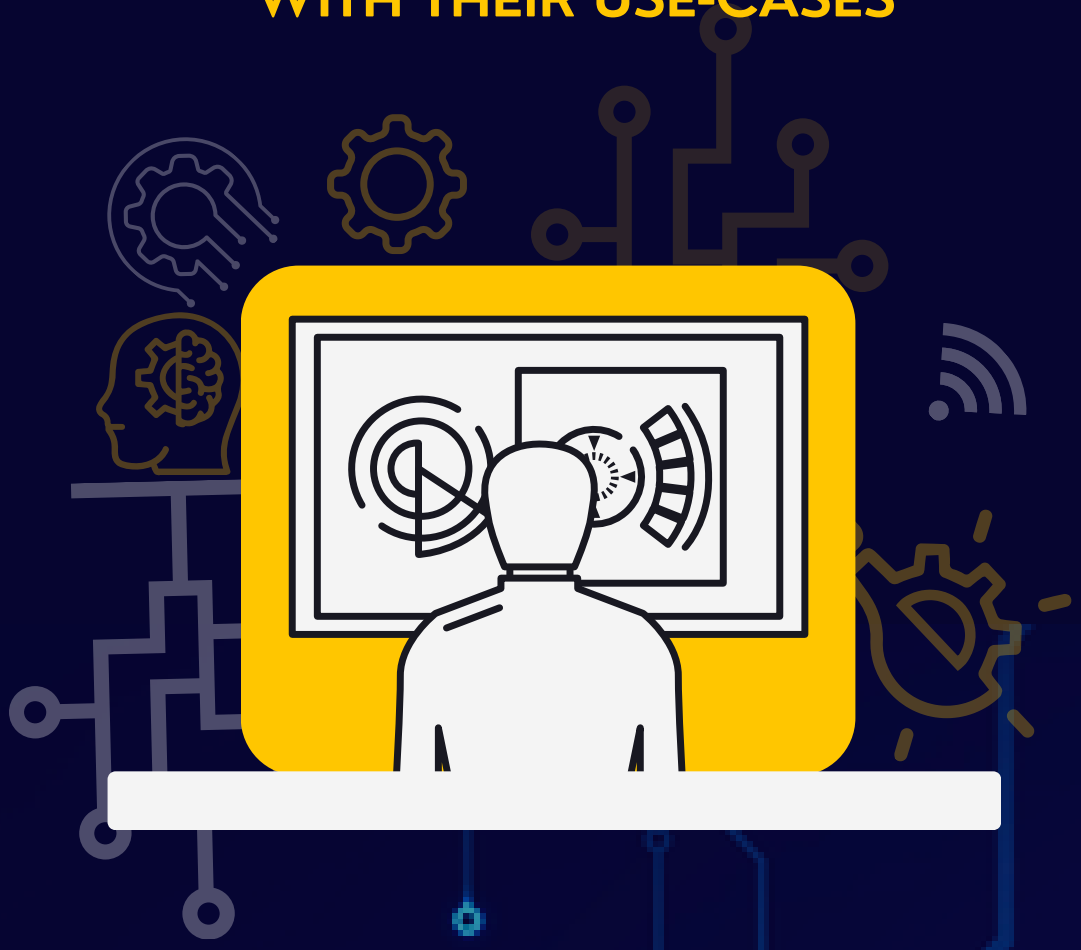
ramchandrapadwal

TOP 10
MACHINE LEARNING ALGORITHMS
WITH THEIR USE-CASES



INTRODUCTION

Machine learning is one of the most exciting fields in the current technological landscape. It's changing the way we live, works, and think about problem-solving. With the help of machine learning algorithms, we can now tackle complex real-world problems with ease and efficiency.

In this Document, we'll be exploring the top 10 most used machine learning algorithms, along with their code snippets and real-world use cases. Whether you're a beginner or a seasoned professional, this Document will give you a comprehensive understanding of these algorithms and help you choose the right one for your next project. So, let's dive in and discover how these algorithms are changing the world.

LINEAR REGRESSION

Linear regression is one of the most commonly used machine learning algorithms for solving regression problems. It is a statistical method that is used to model the relationship between a dependent variable and one or more independent variables. The goal of linear regression is to find the best-fitting line that represents the relationship between the variables.

USE-CASES:

1. House-price estimations using various variables like the area of the property, location, number of bedrooms, etc.
2. Stock price prediction models

ramchandrapadwal

Here's the code snippet to implement the Linear regression algorithm using the sci-kit learn library:

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Load the data into a Pandas dataframe
data = pd.read_csv("data.csv")

# Split the data into training and testing sets
X = data.drop("Dependent Variable", axis=1)
y = data["Dependent Variable"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=0)

# Train the model using the training data
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predict the dependent variable using the test data
y_pred = regressor.predict(X_test)
```

LOGISTICS REGRESSION

Logistic regression is a type of regression analysis that is used for solving classification problems. It is a statistical method that is used to model the relationship between a dependent variable and one or more independent variables. It used the 'logit' function to classify the outcome of input into two categories. Unlike linear regression, logistic regression is used to predict a binary outcome, such as yes/no or true/false.

USE-CASES:

1. Credit risk classification
2. Fraud detection
3. Medical diagnosis classification

ramchandrapadwal

Let's look at the code implementation of the Logistics regression algorithm using the sklearn library.

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Load the data into a Pandas dataframe
data = pd.read_csv("data.csv")

# Split the data into training and testing sets
X = data.drop("Dependent Variable", axis=1)
y = data["Dependent Variable"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=0)

# Train the model using the training data
classifier = LogisticRegression()
classifier.fit(X_train, y_train)

# Predict the dependent variable using the test data
y_pred = classifier.predict(X_test)
```

SUPPORT VECTOR MACHINES

Support Vector Machine (SVM) is a machine learning algorithm that represents data as points in a high-dimensional space, called a hyperplane. The hyperplane is found that maximizes the margin between the training data and the margin of misclassification on it. The algorithm compares this margin with a threshold called the support vector. This threshold determines how accurately each point will be classified as belonging to one of two classes.

SVM has been widely used in many different applications, especially in computer vision and text classification. Some of them are as below:

USE-CASES:

1. Image understanding
2. Speech recognition
3. Natural language processing

ramchandrapadwal

Let's look at the code implementation of the Support Vector Machines algorithm using the sklearn library.

```
import numpy as np
from sklearn.svm import SVC

# Sample data
X = np.array([[1, 2], [2, 3], [3, 4], [4, 5], [5, 6]])
y = np.array([0, 0, 1, 1, 1])

# Create and fit the model
model = SVC(kernel='linear')
model.fit(X, y)

# Predict using the model
y_pred = model.predict(X)
```

DECISION TREES

Decision Trees are one of the most popular machine-learning algorithms. They are used for classification, regression, and anomaly detection. Decision trees set up a hierarchy of decisions based on the outcome of the test data. Each decision is made by choosing a split at some point in the tree.

The decision tree algorithm is useful because it can be easily visualized as a series of splits and leaf nodes, which helps understand how to make a decision in an ambiguous situation.

Decision trees are widely used because they are interpretable as opposed to black box algorithms like Neural Networks, gradient boosting trees, etc.

USE-CASES:

1. Loan approval classification
2. Student graduation rate classification
3. Medical expenses prediction
4. Customer churn prediction

ramchandrapadwal

Let's look at the code implementation of the Decision Trees algorithm using the sklearn library.

```
import numpy as np
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor

# Sample data
X = np.array([[1], [2], [3], [4], [5]])
y_classification = np.array([0, 0, 1, 1, 1])
y_regression = np.array([2, 4, 5, 4, 5])

# Create and fit the classification model
clf_model = DecisionTreeClassifier()
clf_model.fit(X, y_classification)

# Create and fit the regression model
reg_model = DecisionTreeRegressor()
reg_model.fit(X, y_regression)

# Predict using the models
y_pred_clf = clf_model.predict(X)
y_pred_reg = reg_model.predict(X)
```

NAIVE BAYES

Naive Bayes is a probabilistic inference algorithm for continuous (rather than discrete) data. It's also known as Bayes' theorem, Bayesian inference, and Bayes' rule.
In its simplest form, Naive Bayes assumes that the conditional probability of an event given evidence A is proportional to the product of two terms:

P(A|B) = (P(A) * P(B|A))/P(B)

The first term represents the probability of A given B, while the second term represents the probability of B given A, multiplied by the probability of A whole divided by the probability of B.
The Naive Bayes algorithm is used widely in text data classification given the amount of data available in a text corpus. The algorithm assumes all the input variables are independent of each other which is the reason it is called a Naive Bayes algorithm. let's look at some of its use cases.

USE-CASES:

1. Document classification (e.g. newspaper article category classification)
2. Email spam classification
3. Fraud detection

ramchandrapadwal

Let's look at the code implementation of the Naive Bayes algorithm.

```
import numpy as np
from sklearn.naive_bayes import GaussianNB

# Sample data
X = np.array([[1, 1], [1, 2], [2, 2], [2, 3], [3, 3]])
y = np.array([0, 0, 1, 1, 1])

# Create and fit the model
model = GaussianNB()
model.fit(X, y)

# Predict using the model
y_pred = model.predict(X)
```

K-NEAREST NEIGHBORS

K-Nearest Neighbors (KNN) is a supervised learning algorithm that is used for classification and regression tasks. It works by finding the k-closest data points to a given data point and then using the labels of those data points to classify the given data point.
KNN is commonly used for image classification, text classification, and predicting the value of a given data point. Some of the use cases are as below:

USE-CASES:

1. Product recommendation system
2. Fraud prevention

ramchandrapadwal

Let's look at the code implementation of the K-Nearest Neighbors algorithm using the sklearn library.

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor

# Sample data
X = np.array([[1], [2], [3], [4], [5]])
y_classification = np.array([0, 0, 1, 1, 1])
y_regression = np.array([2, 4, 5, 4, 5])

# Create and fit the classification model
clf_model = KNeighborsClassifier(n_neighbors=3)
clf_model.fit(X, y_classification)

# Create and fit the regression model
reg_model = KNeighborsRegressor(n_neighbors=3)
reg_model.fit(X, y_regression)

# Predict using the models
y_pred_clf = clf_model.predict(X)
y_pred_reg = reg_model.predict(X)
```

ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANNs) are a type of supervised learning algorithm that is inspired by the biological neurons in the human brain. They are used for complex tasks such as image recognition, natural language processing, and speech recognition.

ANNs are composed of multiple interconnected neurons which are organized into layers, with each neuron in a layer having a weight and a bias associated with it. When given an input, the neurons process the information and output a prediction.

There are types of neural networks used in a variety of applications. Convolutional Neural Networks are used in image classification, object detection, and segmentation tasks while Recurrent Neural Networks are used in language modeling tasks. Let's look at some of the use cases of ANNs

USE-CASES:

1. Image classification tasks
2. Text classification
3. Language Translation
4. Language detection

ramchandrapadwal

Let's look at the code implementation of the Artificial Neural Networks algorithm.

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

# Sample data
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([0, 0, 1, 1, 1])

# Create the model
model = Sequential()
model.add(Dense(10, input_dim=1, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile and fit the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, y, epochs=100, batch_size=1)

# Predict using the model
y_pred = model.predict_classes(X)
```

RANDOM FORESTS

Random forest is a type of machine learning algorithm that is used for solving classification and regression problems. It is an ensemble method that combines multiple decision trees to create a more accurate and stable model. Random forest is particularly useful for handling large datasets with complex features, as it is able to select the most important features and reduce overfitting.

Random forest algorithms can be expensive to train and are really hard to interpret model performance as opposed to decision trees. let's look at some of the use cases of random forests.

USE-CASES:

1. Credit scoring models
2. Medical diagnosis prediction
3. Predictive maintenance

ramchandrapadwal

Let's look at the code implementation of the Random Forests algorithm.

```
import numpy as np
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor

# Sample data
X = np.array([[1], [2], [3], [4], [5]])
y_classification = np.array([0, 0, 1, 1, 1])
y_regression = np.array([2, 4, 5, 4, 5])

# Create and fit the classification model
clf_model = RandomForestClassifier()
clf_model.fit(X, y_classification)

# Create and fit the regression model
reg_model = RandomForestRegressor()
reg_model.fit(X, y_regression)

# Predict using the models
y_pred_clf = clf_model.predict(X)
y_pred_reg = reg_model.predict(X)
```

K-MEANS CLUSTERING

K-means is a popular unsupervised machine-learning algorithm that is used for clustering data. It works by dividing a set of data points into a specified number of clusters, where each data point belongs to the cluster with the nearest mean. K-means is an iterative algorithm that repeats the clustering process until convergence is achieved.

The k-means algorithm is easier to train compared to other clustering algorithms. It is scalable on large datasets for clustering samples. It is simple to implement and interpret. let's look at some of the use cases of the K-means algorithm.

USE-CASES:

1. Customer segmentation
2. Anomaly detection
3. Medical image segmentation

ramchandrapadwal

Let's look at the code implementation of the K-Means Clustering algorithm.

```python
import numpy as np
from sklearn.cluster import KMeans

# Sample data
X = np.array([[1, 2], [1, 3], [2, 2], [2, 3], [5, 6], [6, 5], [7, 7], [8, 6]])

# Create and fit the KMeans model with k=2 (2 clusters)
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

# Get cluster centroids and labels
centroids = kmeans.cluster_centers_
labels = kmeans.labels_

# Predict cluster membership for new data points
new_data = np.array([[3, 4], [6, 6]])
new_labels = kmeans.predict(new_data)
```

GRADIENT BOOSTING

Gradient boosting trees (GBT) is a popular machine learning algorithm that is used for classification and regression tasks. It is an ensemble method that combines multiple decision trees to create a more accurate and stable model. GBT works by sequentially adding decision trees, where each new tree is trained to correct the errors of the previous trees. The model combines the predictions of all trees to make a final prediction.

The gradient boosting algorithm is better compared to other models for regression tasks. It can handle multicollinearity and non-linear relationships between variables. It is sensitive to an outlier, therefore can cause overfitting. Now let's look at some of its use cases.

USE-CASES:

1. Fraud detection
2. Customer Churn Prediction

ramchandrapadwal

Let's look at the code implementation of the Gradient Boosting algorithm.

```python
import numpy as np
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor

# Sample data
X = np.array([[1], [2], [3], [4], [5]])
y_classification = np.array([0, 0, 1, 1, 1])
y_regression = np.array([2, 4, 5, 4, 5])

# Create and fit the classification model
clf_model = GradientBoostingClassifier()
clf_model.fit(X, y_classification)

# Create and fit the regression model
reg_model = GradientBoostingRegressor()
reg_model.fit(X, y_regression)

# Predict using the models
y_pred_clf = clf_model.predict(X)
y_pred_reg = reg_model.predict(X)
```