@LEARNEVERYTHINGAI

6 ESSESNTIAL MACHINE LEARNING ALGORITHMS

SHIVAM MODI

@learneverythingai

Linear Regression

Linear Regression, a fundamental algorithm in Machine Learning, enables us to predict numerical values based on the relationship between input features and the target variable. Its equation "y = mx + b" represents a line that best fits the data points, allowing us to make predictions for new data. This algorithm proves handy when forecasting, trend analysis, and understanding relationships between variables.

```
# Loading dataset and preparing data
data = pd.read_csv('dataset.csv')
X = data[['Feature1', 'Feature2']].values
y = data['Target'].values

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Applying feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

SHIVAM MODI
@learneverythingai

Decision Trees

Imagine making decisions by following a tree-like structure, where each branch represents a criterion leading to different outcomes. That's exactly how Decision Trees work in Machine Learning! These intuitive algorithms excel in classification and regression tasks, as they split the data into subsets based on features, ultimately arriving at decisions or predictions.

```
# Importing DecisionTreeClassifier

from sklearn.tree import DecisionTreeClassifier

# Creating the classifier

classifier = DecisionTreeClassifier()

# Training the model

classifier.fit(X_train, y_train)
```

SHIVAM MODI
@learneverythingai

Support Vector Machines (SVM)

Support Vector Machines (SVM) are remarkable algorithms for classification tasks. They create decision boundaries that separate different classes, maximizing the margin between data points of different classes. SVM is a powerful tool for both linear and non-linear classification tasks, and it has found applications in various domains, including image recognition and sentiment analysis.

```
# Importing SVM Classifier
from sklearn.svm import SVC

# Creating the classifier
classifier = SVC(kernel='linear', C=1.0)

# Training the model
classifier.fit(X_train, y_train)
```

SHIVAM MODI
@learneverythingai

k-Nearest Neighbors (KNN)

Intrigued by the saying "birds of a feather flock together"? That's the essence of k-Nearest Neighbors (KNN) algorithm! KNN works on the principle that similar data points are often located near each other in the feature space. When presented with a new data point, KNN looks at its 'k' nearest neighbors and assigns it the most common class among them. This simple yet effective algorithm finds applications in recommendation systems and pattern recognition.

```python
# Importing KNN Classifier
from sklearn.neighbors import KNeighborsClassifier

# Creating the classifier
classifier = KNeighborsClassifier(n_neighbors=5)

# Training the model
classifier.fit(X_train, y_train)
```

SHIVAM MODI
@learneverythingai

Neural Networks

Neural Networks, inspired by the human brain, are the backbone of Deep Learning. They consist of interconnected nodes (neurons) organized in layers. Each layer processes information and passes it on to the next until the final output is generated. Neural Networks excel in complex tasks like image recognition, natural language processing, and more. These networks are trained through a process called backpropagation, which optimizes the model to make accurate predictions.

```python
# Importing libraries for neural networks
import tensorflow as tf
from tensorflow.keras import layers, models

# Creating the neural network model
model = models.Sequential()
model.add(layers.Dense(128, activation='relu', input_shape=(input_dim,)))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(output_dim, activation='softmax'))

# Compiling the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Training the model
model.fit(X_train, y_train, epochs=10, batch_size=32)
```

Model Evaluation

Building and training models is exciting, but how do we know if they perform well? Model evaluation is the key! We use metrics like accuracy, precision, recall, and F1-score to assess the model's performance. By comparing the predicted outcomes with the actual ones, we gauge how well the model generalizes to new, unseen data. Understanding model evaluation is crucial in making informed decisions about which algorithms to use for specific tasks.

```
# Evaluating the model's performance
accuracy = classifier.score(X_test, y_test)
print(f"Accuracy: {accuracy:.2f}")
```

@learneverythingai

Like this Post?

- Follow Me
- Share with your friends
- Check out my previous posts

SHIVAM MODI
@learneverythingai

Follow

SHARE

SAVE THIS

www.learneverythingai.com