Member-only story

PCA vs Autoencoders for a Small Dataset in Dimensionality Reduction

Neural Networks and Deep Learning Course: Part 45

Rukshan Pramoditha · Follow
Published in Towards Data Science
8 min read · Feb 16

Listen Share More

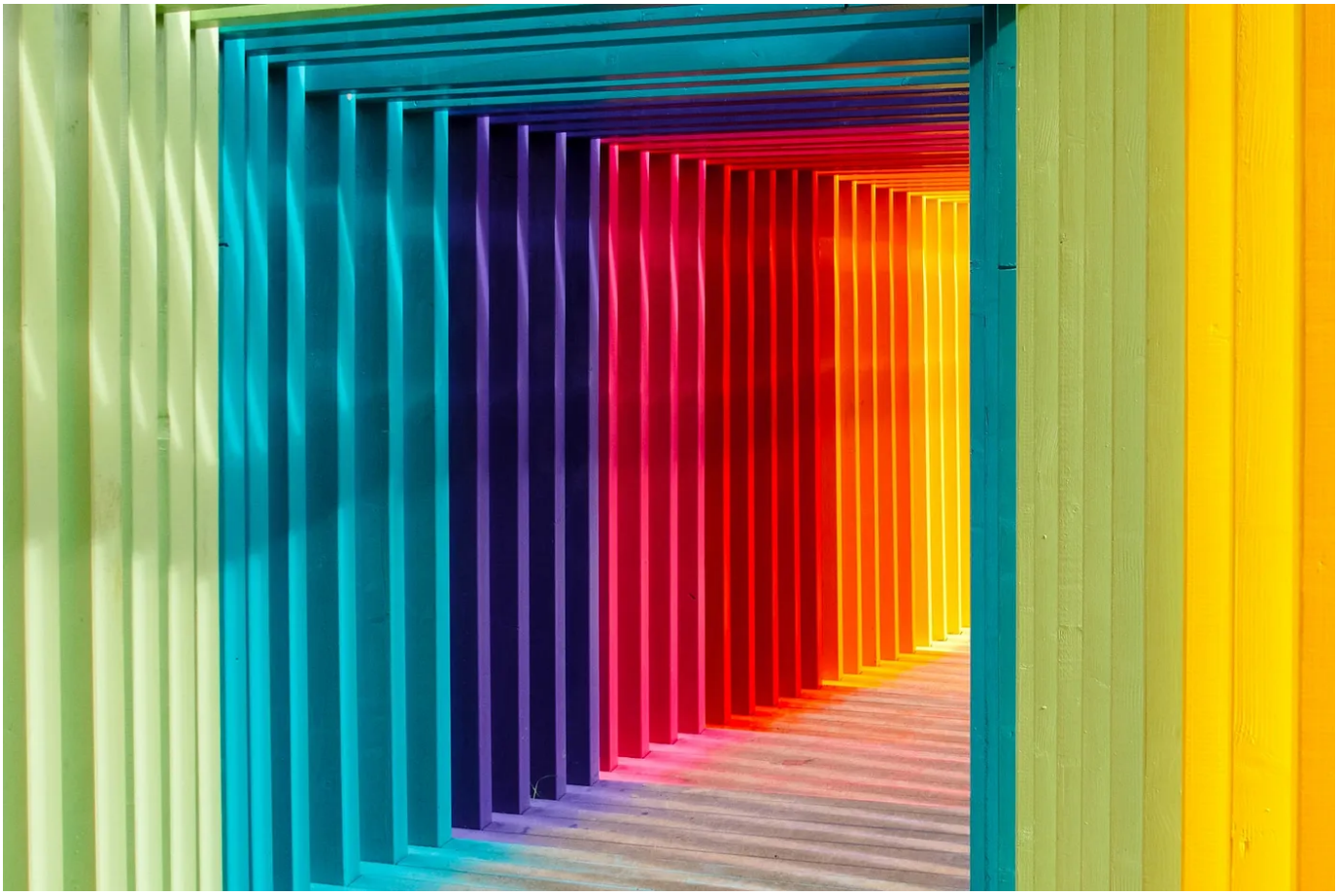


Photo by Robert Katzki on Unsplash

Can general machine learning algorithms outperform neural networks with small datasets?

In general, deep learning algorithms such as neural networks require a massive amount of data to achieve reasonable performance. So, neural networks like autoencoders can benefit from very large datasets that we use to train the models.

Sometimes, general machine learning algorithms can outperform neural network algorithms when they are trained with very small datasets.

Autoencoders can also be used in dimensionality reduction applications, even though they are widely used in other popular applications such as image denoising, image generation, image colorization, image compression, image super-resolution, etc.

Earlier, we compared the performance of autoencoders in dimensionality reduction against PCA by training the models on the *very large* MNIST dataset. There, the autoencoder model easily outperformed the PCA model [ref[1]] because the MNIST data is large and non-linear.

Autoencoders work well with large and non-linear data.

Even though autoencoders are a type of neural network, it is still possible to use them with smaller datasets to achieve some performance with the correct model architecture and hyperparameter values.

> Autoencoders are powerful and flexible enough to capture complex and non-linear patterns in data.

Today, we will compare the performance of an autoencoder (the neural network model) in dimensionality reduction against PCA (the general machine learning algorithm) by training the models on the *very small* Wine dataset.

The Wine dataset has 178 samples and 13 features. This dataset is very small when we compare it with the MNIST dataset which has 60,000 samples and 784 features!

Load the Wine dataset

The Wine dataset comes preloaded with Sciki-learn and can be loaded as follows.

```
# Load the Wine dataset
from sklearn.datasets import load_wine

wine = load_wine()
X = wine.data
y = wine.target

print("Wine dataset size:", X.shape)
```

```
Wine dataset size: (178, 13)
```

The shape of the Wine dataset (Image by author)

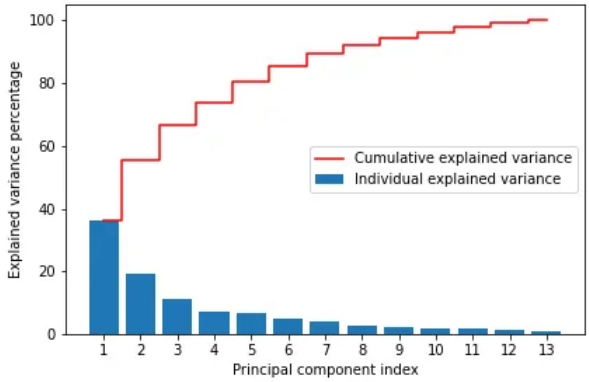
Run PCA with Wine data

First, we will need to select the best number of components by running the PCA with all components [ref[2]].

ref[2]: *3 Easy Steps to Perform Dimensionality Reduction Using Principal Component Analysis (PCA)*

```
# Feature scaling
from sklearn.preprocessing import StandardScaler

X_scaled = StandardScaler().fit_transform(X)

import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# Select the best number of components
# by running PCA with all components
pca = PCA(n_components=None)
pca.fit(X_scaled)

# Plot cumulative explained variances
exp_var = pca.explained_variance_ratio_ * 100
cum_exp_var = np.cumsum(exp_var)
```

```
plt.bar(range(1, 14), exp_var, align='center',
        label='Individual explained variance')

plt.step(range(1, 14), cum_exp_var, where='mid',
         label='Cumulative explained variance', color='red')

plt.ylabel('Explained variance percentage')
plt.xlabel('Principal component index')
plt.xticks(ticks=list(range(1, 14)))
plt.legend(loc='best')
plt.tight_layout()

plt.savefig("cumulative_explained_variance_plot.png")
```



Cumulative explained variance plot (Image by author)

By analyzing the above cumulative explained variance plot, I've decided to keep the first seven components which capture about 90% variance in the data. So, those seven components will accurately represent the original Wine dataset.

Run PCA again with selected components

To apply dimensionality reduction to the Wine dataset, we need to run PCA again with the selected components and apply the transformation.

```
# Run PCA again with selected (7) components
pca = PCA(n_components=7)
X_pca = pca.fit_transform(X_scaled)
print("PCA reduced wine dataset size:", X_pca.shape)
```

```
PCA reduced wine dataset size: (178, 7)
```

The shape of the reduced Wine dataset after applying PCA (Image by author)

Now, there are only seven components (features) in the dataset. So, the dimensionality of the data has been reduced!

Perform dimensionality reduction with an autoencoder

The following code defines an autoencoder in which the encoder part can be used to obtain the lower dimensional (encoded) representation of the data.

```
import numpy as np
from tensorflow.keras import Model, Input
from tensorflow.keras.layers import Dense

# Build the autoencoder
input_dim = X.shape[1]
latent_vec_dim = 7
```

```
input_layer = Input(shape=(input_dim,))

# Define encoder
x = Dense(8, activation='relu')(input_layer)
x = Dense(4, activation='relu')(x)
encoder = Dense(latent_vec_dim, activation="tanh")(x)

# Define decoder
x = Dense(4, activation='relu')(encoder)
x = Dense(8, activation='relu')(x)
decoder = Dense(input_dim, activation="sigmoid")(x)

autoencoder = Model(inputs=input_layer, outputs=decoder)

# Compile the model with optimizer and loss function
autoencoder.compile(optimizer="adam", loss="mse")

# Train the model with 100 epochs
autoencoder.fit(X_scaled, X_scaled, epochs=100, verbose=0,
                batch_size=16, shuffle=True)

# Use the encoder part to obtain the lower dimensional,
# encoded representation of the data
encoder_model = Model(inputs=input_layer, outputs=encoder)
X_encoded = encoder_model.predict(X_scaled)

# Print the shape of the encoded data
print("Autoencoder reduced wine dataset size:", X_encoded.shape)
```

```
6/6 [==============================] - 0s 3ms/step
Autoencoder reduced wine dataset size: (178, 7)
```

The shape of the reduced Wine dataset after auto-encoding data (Image by author)

The autoencoder model has an input layer, encoding layers and decoding layers. The input dimension is the number of input features in the Wine dataset, which is 13. The latent vector dimension is 7 which is equal to the number of components that we selected earlier in PCA.

All layers are connected using the Keras Functional API method [ref[3]]. Then, the whole autoencoder model is created by connecting the input layer and decoder part.

ref[3]: *Two Different Ways to Build Keras Models: Sequential API and Functional API*

Then, we compile the whole autoencoder with the Adam optimizer and the mean squared error (mse) loss function.

The model is trained on the standardized (scaled) Wine data for 100 epochs with a batch size of 16.

The latent vector represents the most important features of the input data in a lower-dimensional form [ref[4]]. Therefore, after training the whole autoencoder, we can use its encoder part to obtain the lower dimensional (encoded) representation of the data.

ref[4]: *An Introduction to Autoencoders in Deep Learning*

The `encoder_model` is created by connecting the input layer and the encoder part. Then, we can call its predict() method on the scaled Wine data to obtain the lower dimensional (encoded) representation of the Wine dataset which is represented by the variable, `X_encoded`. Since the **"latent vector dimension"** is set to 7, the encoded representation has 7 features and the dimensionality of the data has been reduced!

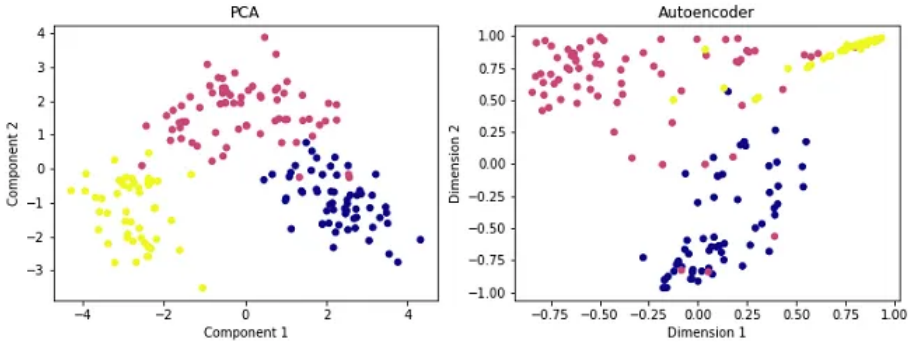
Compare both PCA and autoencoder models by visualizing data

Visualization of high-dimensional data can be achieved through dimensionality reduction [ref[5]]. So, dimensionality reduction is extremely useful for data visualization.

ref[5]: *11 Different Uses of Dimensionality Reduction*

By using only two components (dimensions), we plot the Wine dataset outputs returned by both PCA and autoencoder models.

```
import matplotlib.pyplot as plt

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))

# Plot PCA output
ax1.scatter(X_pca[:, 0], X_pca[:, 1], c=y, s=25, cmap='plasma')
ax1.set_title('PCA')
ax1.set_xlabel('Component 1')
ax1.set_ylabel('Component 2')

# Plot autoencoder output
ax2.scatter(X_encoded[:, 0], X_encoded[:, 1], c=y, s=25, cmap='plasma')
ax2.set_title('Autoencoder')
ax2.set_xlabel('Dimension 1')
ax2.set_ylabel('Dimension 2')

plt.savefig("Output.png")
```



Two-dimensional representations of Wine data: PCA vs Autoencoder (Image by author)

As you can see, the two-dimensional representation of Wine data obtained using PCA shows a clear separation between the three classes of wine, but the separation is not good enough in autoencoder output.

PCA works well with small datasets like the Wine dataset.

> ***Note:*** *The output of the autoencoder (right plot) may vary significantly due to the stochastic nature of the algorithm and the values of hyperparameters such as the number of hidden layers and hidden units, type of activation function used in each layer, type of loss function, type of optimizer, number of epochs and the batch size! But, the separation of the three classes may not be as good as the PCA output.*

PCA vs Autoencoder: Which is better in dimensionality reduction?

The choice of PCA and autoencoder for dimensionality reduction depends on the following factors.

- Size of the dataset
- The complexity of the dataset (linear or non-linear, image or numerical data)
- Goals of the analysis
- Availability of computational resources
- Interpretability

PCA works well with small datasets. It can also be used with larger datasets. However, autoencoders work really well with very large datasets.

PCA works well with linear data as it is a linear dimensionality reduction technique. It is not effective with non-linear data. In contrast, autoencoders can easily capture complex and non-linear patterns in the data. So, they work well with non-linear data.

Autoecnder works well with image data. PCA works well with numerical data.

There is no way to determine the importance of each component (feature) in the latent vector of an autoencoder model. But in PCA, we can create the cumulative explained variance plot for that.

Since an autoencoder is a neural network, its architecture may become complex. In addition to that, it requires a massive amount of data. Therefore, autoencoders require more computational resources than PCA.

Autoecndoers are black-box models. So, they are hard to interpret. We don't know how they select important features from the data we provide. So, the interpretation of those models is very difficult.

Conclusions

Both PCA and autoencoders can be used to perform dimensionality reduction. PCA is a general machine-learning algorithm while autoencoders are a type of neural network architecture that require large datasets and a lot of computational resources.

Autoencoders work well with large and non-linear data. They are powerful and flexible enough to capture complex and non-linear patterns in data, but may fail to outperform general machine learning algorithms like PCA with smaller datasets!

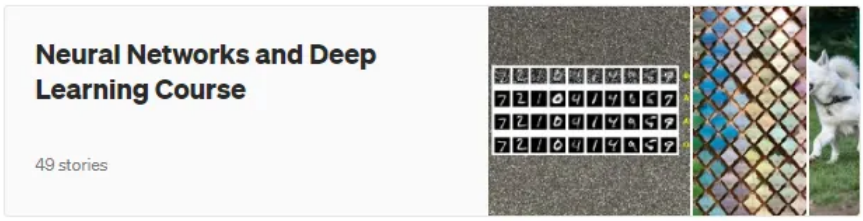
If you really want to consider class separability while performing dimensionality reduction, LDA (Linear Discriminant Analysis) is the best choice. Read the complete guide below.

LDA Is More Effective than PCA for Dimensionality Reduction in Classification Datasets

This is the end of today's article.

Please let me know if you've any questions or feedback.

How about an AI course?



Join my Neural Networks and Deep Learning Course, the first ever on Medium (Screenshot by author)

Support me as a writer

*I hope you enjoyed reading this article. If you'd like to support me as a writer, kindly consider **signing up for a membership** to get unlimited access to Medium. It only costs $5 per month and I will receive a portion of your membership fee.*

Join Medium with my referral link - Rukshan Pramoditha

Read every story from Rukshan Pramoditha (and thousands of other writers on Medium). Your membership fee directly...

rukshanpramoditha.medium.com

Join my private list of emails

*Never miss a great story from me again. By **subscribing to my email list**, you will directly receive my stories as soon as I publish them.*

Thank you so much for your continuous support! See you in the next article. Happy learning to everyone!

Wine dataset info

- **Dataset source:** You can download the original dataset here.
- **Dataset license:** This dataset is available under the *CC BY 4.0* (*Creative Commons Attribution 4.0*) license.
- **Citation:** Lichman, M. (2013). UCI Machine Learning Repository [https://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Rukshan Pramoditha

2023–02–16

Artificial Intelligence | Data Science | Machine Learning | Dimensionality Reduction | Autoencoder

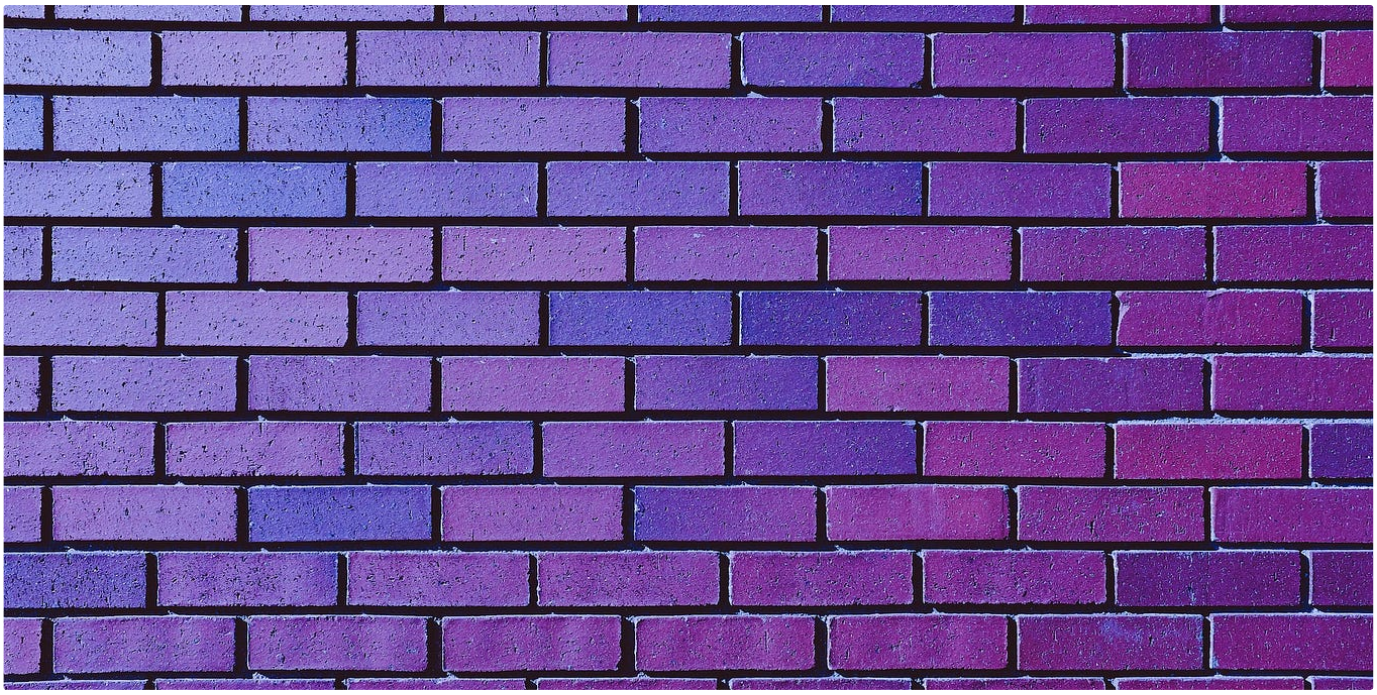
Follow

Written by Rukshan Pramoditha

6.2K Followers · Writer for Towards Data Science

2,000,000+ Views | BSc in Stats | Top 50 Data Science, AI/ML Technical Writer on Medium | Data Science Masterclass: https://datasciencemasterclass.substack.com/

More from Rukshan Pramoditha and Towards Data Science

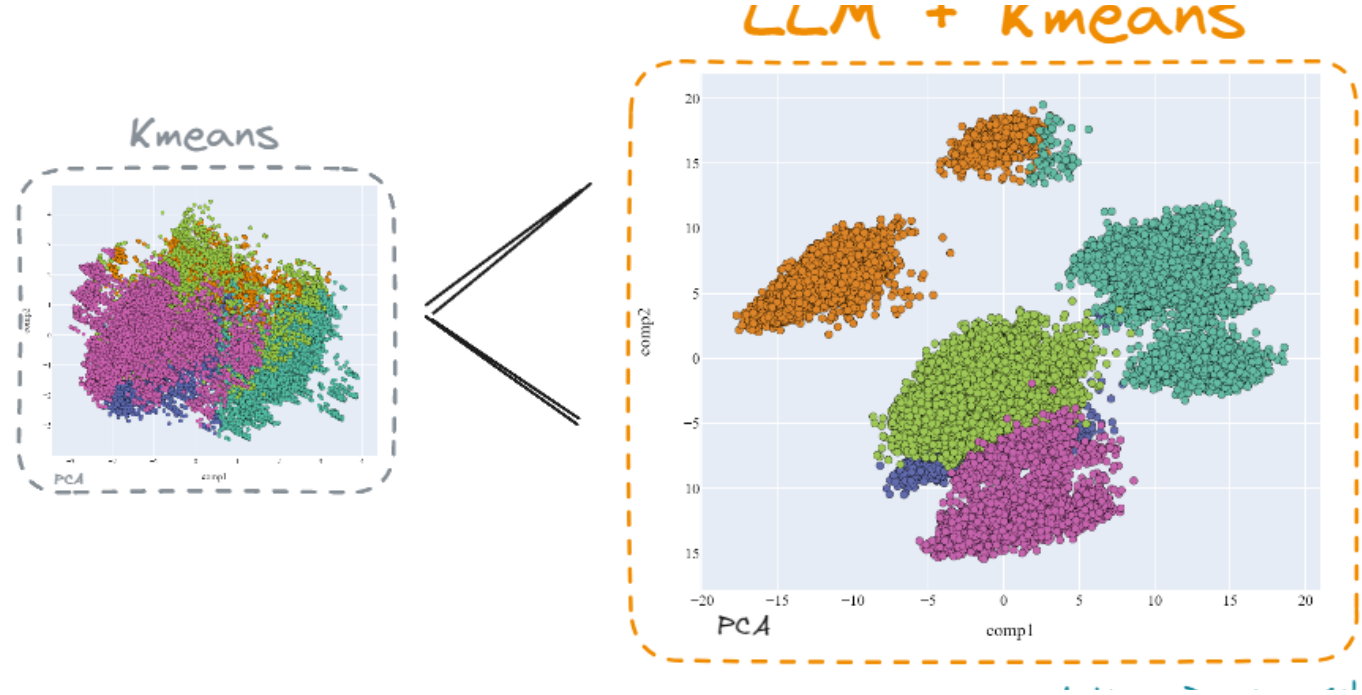


Rukshan Pramoditha

7 Types of Cross-Validation (CV) Techniques You Should Know as a Data Scientist in 2023

With their Python implementations graphical visualizations

· 8 min read · Oct 9



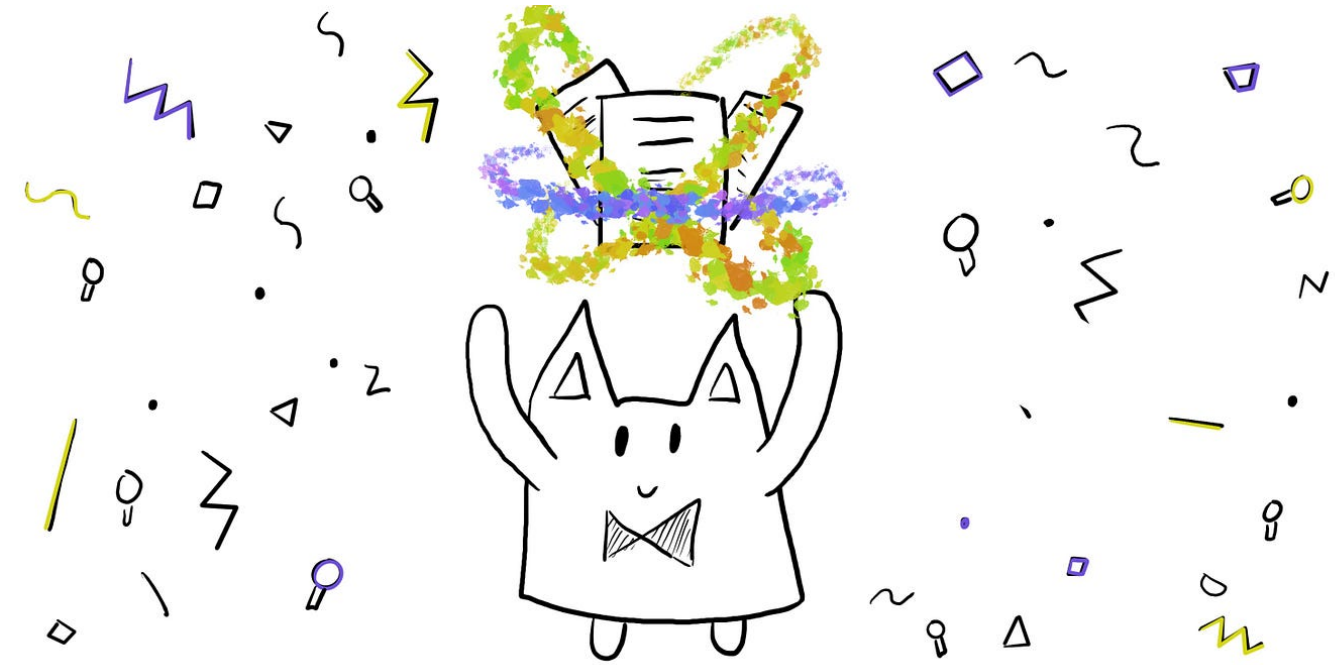
Damian Gil in Towards Data Science

Mastering Customer Segmentation with LLM

Unlock advanced customer segmentation techniques using LLMs, and improve your clustering models with advanced techniques

24 min read · Sep 27

3.3K 26



Adrian H. Raudaschl in Towards Data Science

Forget RAG, the Future is RAG-Fusion

The Next Frontier of Search: Retrieval Augmented Generation meets Reciprocal Rank Fusion and Generated Queries

· 10 min read · Oct 6

1.6K 22

Rukshan Pramoditha in Towards Data Science

20 Necessary Requirements of a Perfect Laptop for Data Science and Machine Learning Tasks

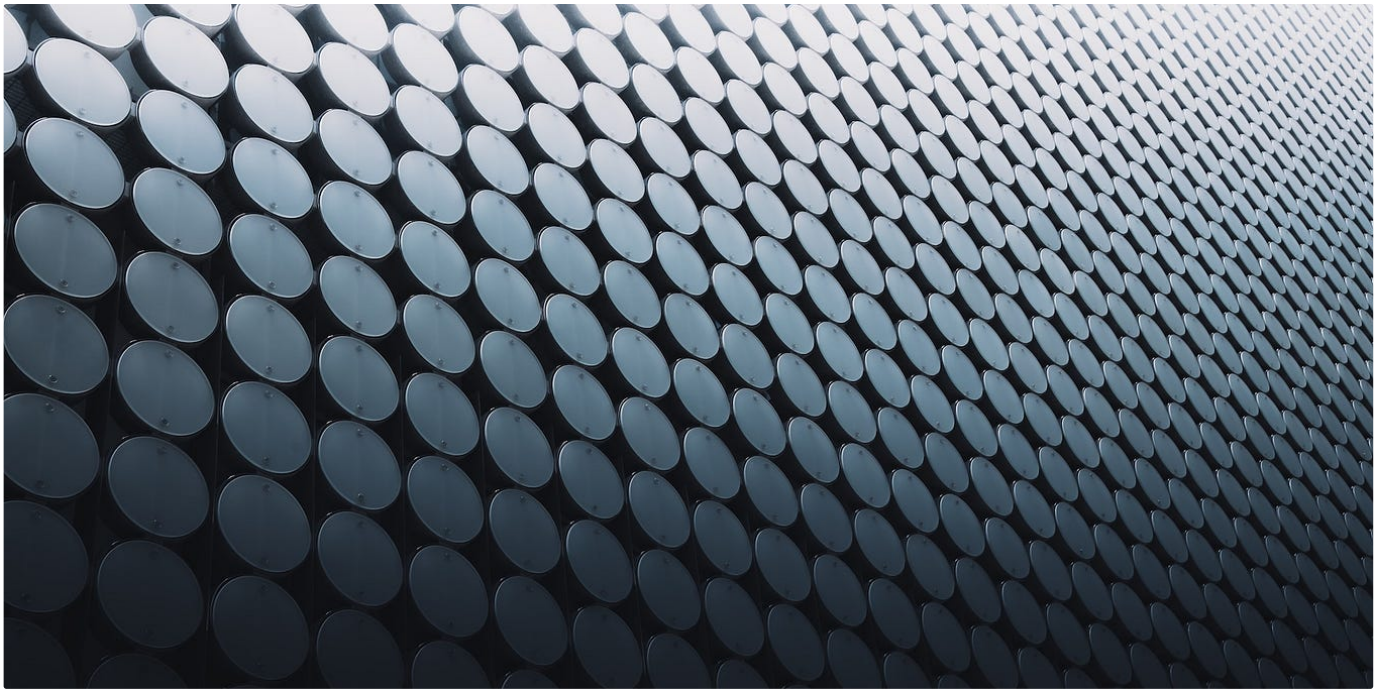
· 7 min read · Jun 6, 2021

809 19

See all from Rukshan Pramoditha

See all from Towards Data Science

Recommended from Medium



Rukshan Pramoditha in Towards Data Science

How t-SNE Outperforms PCA in Dimensionality Reduction

PCA vs t-SNE for visualizing high-dimensional data in a lower-dimensional space

✦ · 15 min read · May 23

89

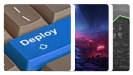
Ujang Riswanto

What Is An Autoencoder? Why It's So Important For Dimensionality Reduction

A Comprehensive Guide to Dimensionality Reduction and its Importance in Data Analysis

12 min read · May 18

Lists

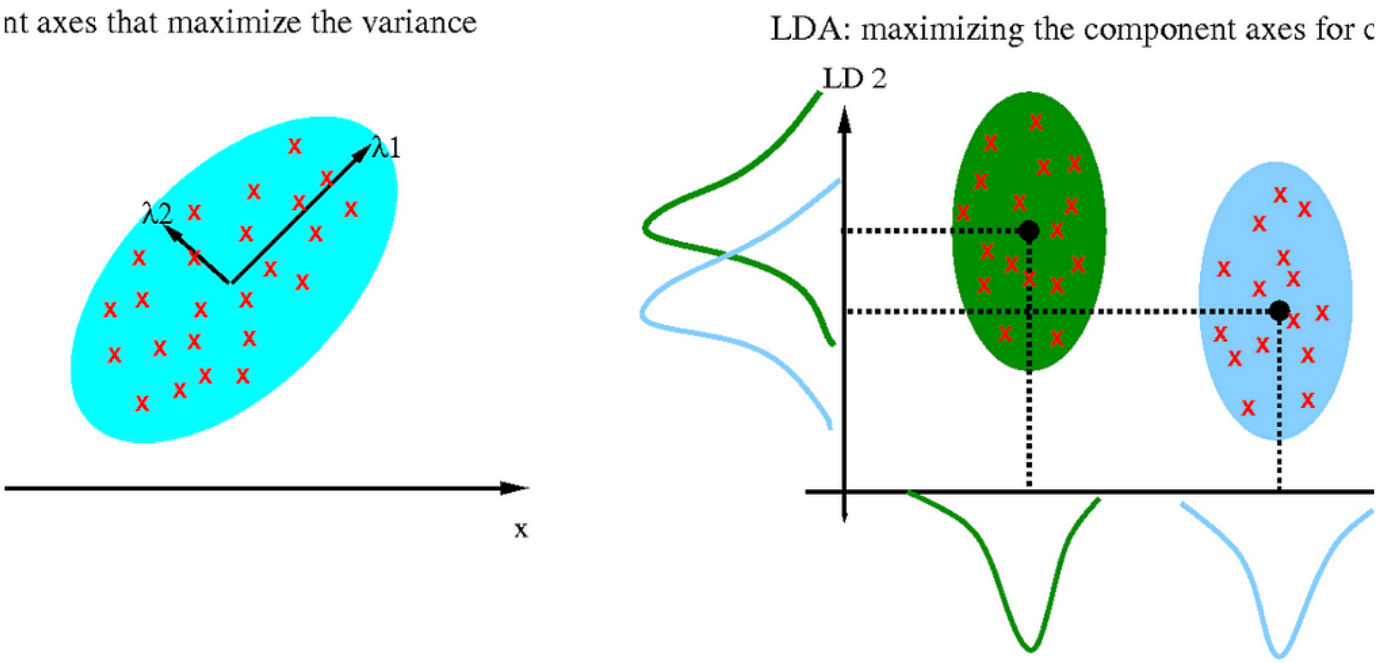


Predictive Modeling w/ Python
20 stories · 524 saves

Practical Guides to Machine Learning
10 stories · 603 saves

Natural Language Processing
746 stories · 336 saves

ChatGPT prompts
27 stories · 541 saves



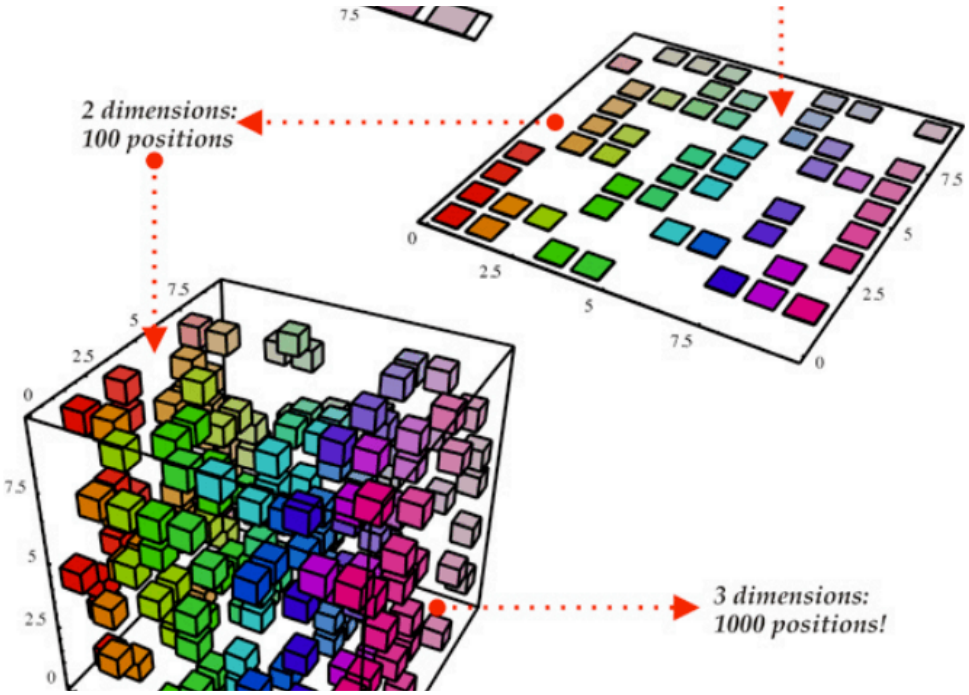
Seshu Kumar Vungarala

PCA vs LDA—No more confusion!

Introduction

3 min read · Apr 30

15



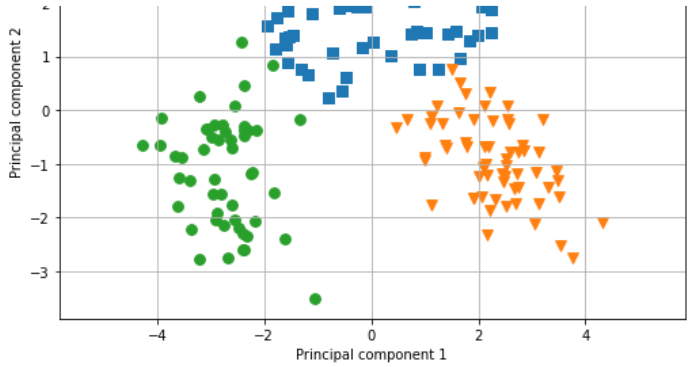
Dhanoop Karunakaran in Intro to Artificial Intelligence

Principal Component Analysis(PCA)

Dimensionality reduction is the process of transforming the data from high-dimensional space to low-dimensional space[5]. In other words...

7 min read · Jul 10

43




```
Optimal number of clusters: 3

- Cluster 0
  36.5169% of total patterns
  Number of patterns with real class A: 0
  Number of patterns with real class B: 65
  Number of patterns with real class C: 0

- Cluster 1
  34.8315% of total patterns
  Number of patterns with real class A: 59
  Number of patterns with real class B: 3
  Number of patterns with real class C: 0
```

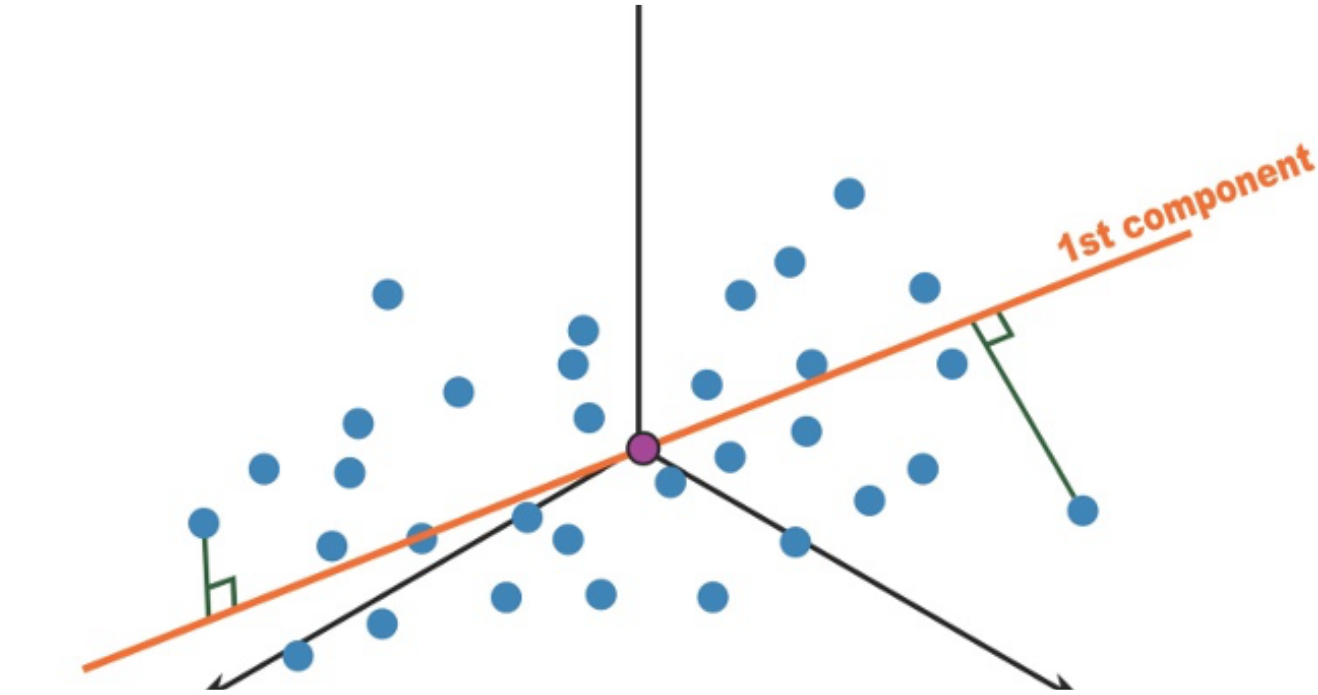
Ana Belén Manjavacas

Dimensionality Reduction and PCA

As a Data Scientist, one of the most significant challenges in a project is dealing with large datasets that contains numerous dimensions...

6 min read · Aug 9

7



Huda Swati

Understanding Principal Component Analysis (PCA)

What is PCA?

8 min read · Sep 26