Open in app ↗ Sign up Sign In

Search Medium

You have **2 free member-only stories left** this month. Sign up for Medium and get an extra one.

✨ Member-only story

K-Nearest Neighbors (kNN) — Explained

Detailed theorotical explanation and scikit-learn implementation

Soner Yıldırım · Follow
Published in Towards Data Science
6 min read · Feb 29, 2020

Listen Share

Image source

K-nearest neighbors (kNN) is a supervised machine learning algorithm that can be used to solve both classification and regression tasks. I see kNN as an algorithm that comes from *real life*. People tend to be effected by the people around them. Our behaviour is guided by the friends we grew up with. Our parents also shape our personality in some ways. If you grow up with people who love sports, it is higly likely that you will end up loving sports. There are ofcourse exceptions. kNN works

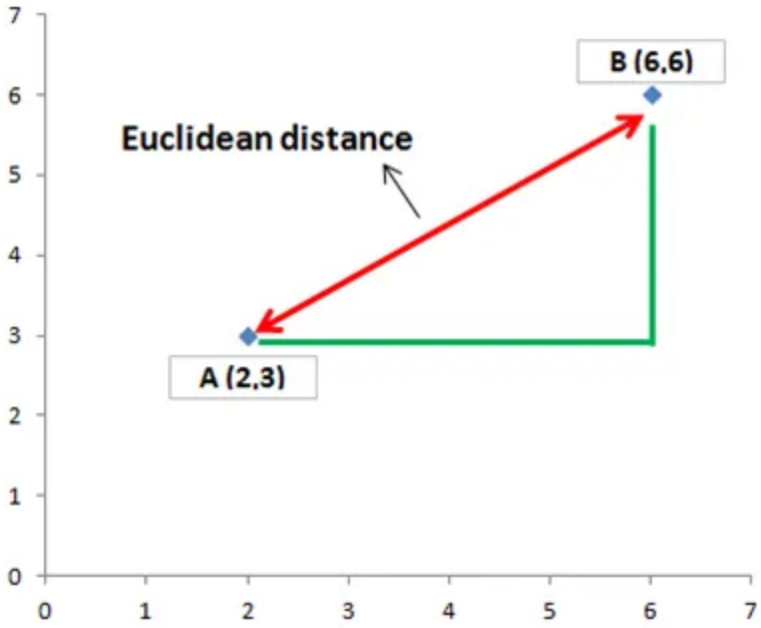
similarly.

> The value of a data point is determined by the data points around it.

- If you have one very close friend and spend most of your time with him/her, you will end up sharing similar interests and enjoying same things. That is kNN with k=1.
- If you always hang out with a group of *5,* each one in the group has an effect on your behavior and you will end up being the average of 5. That is kNN with *k=5*.

kNN classifier determines the class of a data point by majority voting principle. If k is set to 5, the classes of 5 closest points are checked. Prediction is done according to the majority class. Similarly, kNN regression takes the mean value of 5 closest points.

We observe people who are close but how data points are determined to be close? The distance between data points is measured. There are many methods to measure the distance. Euclidean distance (minkowski distance with p=2) is one of most commonly used distance measurement. The figure below shows how to calculate euclidean distance between two points in a 2-dimensional space. It is calculated using the square of the difference between x and y coordinates of the points.



$$\textit{Euclidean distance}\ (a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

In the case above, euclidean distance is the square root of (16 + 9) which is 5. Euclidean distance in two dimensions remind us the famous pythagorean theorem.

It seems very simple for two points in 2-dimensional space. Each dimension represents a feauture in the dataset. We typically have many samples with many features. To be able to explain the concept clearly, I will go over an example in 2-dimensional space (i.e. 2 features).

> *You can access all the code in this github repo. Feel free to use it!*

Let's start with importing libraries:

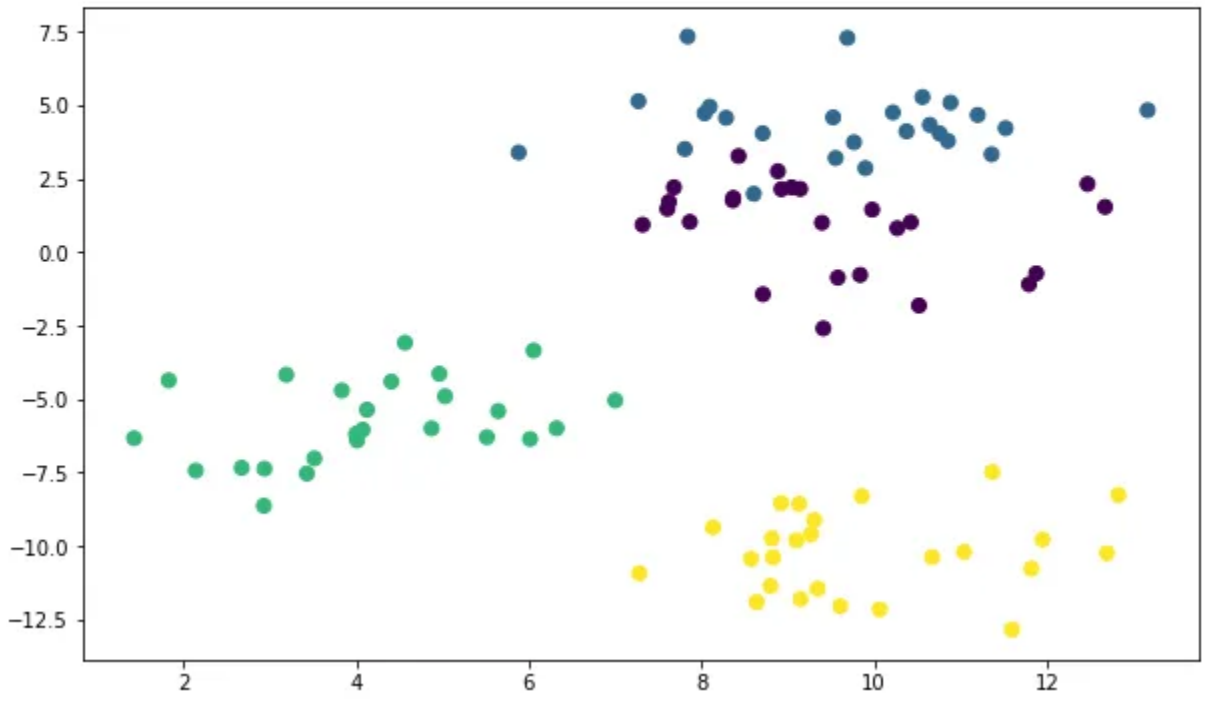
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt #data visualization

from sklearn.datasets import make_blobs #synthetic dataset
from sklearn.neighbors import KNeighborsClassifier #kNN classifier
from sklearn.model_selection import train_test_split #train and test sets
```

> Scikit-learn provides many useful functions to create synthetic datasets which are very helpful for practicing machine learning algorithms. I will use **make_blobs** function.

```
#create synthetic dataset
X, y = make_blobs(n_samples = 100, n_features = 2, centers = 4,
                        cluster_std = 1.5, random_state = 4)
```

This code creates a dataset with 100 samples divided into 4 classes and the number of features is 2. Number of samples, features and classes can easily be adjusted using related parameters. We can also adjust how much each cluster (or class) is spread. Let's visualize this synthetic data set:

```
#scatter plot of dataset
plt.figure(figsize = (10,6))
plt.scatter(X[:,0], X[:,1], c=y, marker= 'o', s=50)
plt.show()
```



For any supervised machine learning algorithm, it is very important to divide dataset into train and test sets. We first train the model and test it using different parts of dataset. If this separation is not done, we basically test the model with some

data it already knows. We can easily do this separation using **train_test_split** function.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
```

We can specify how much of the original data is used for train or test sets using **train_size** or **test_size** parameters, respectively. Default separation is 75% for train set and 25% for test set.

Then we create a kNN classifier object. To show the difference between the importance of k value, I create two classifiers with k values 1 and 5. Then these models are trained using train set. **n_neighbors** parameter is used to select k value. Default value is 5 so it does not have to be explicitly written.

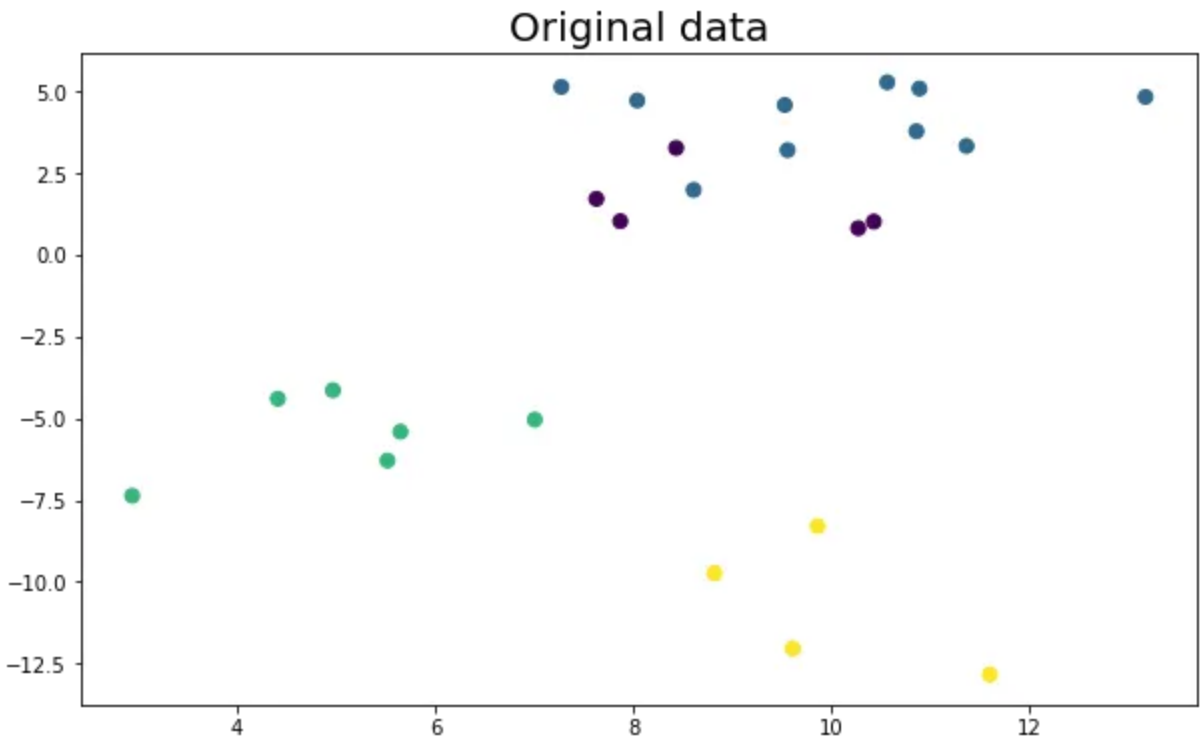
```
knn5 = KNeighborsClassifier() #k=5
knn1 = KNeighborsClassifier(n_neighbors=1) #k=1
```

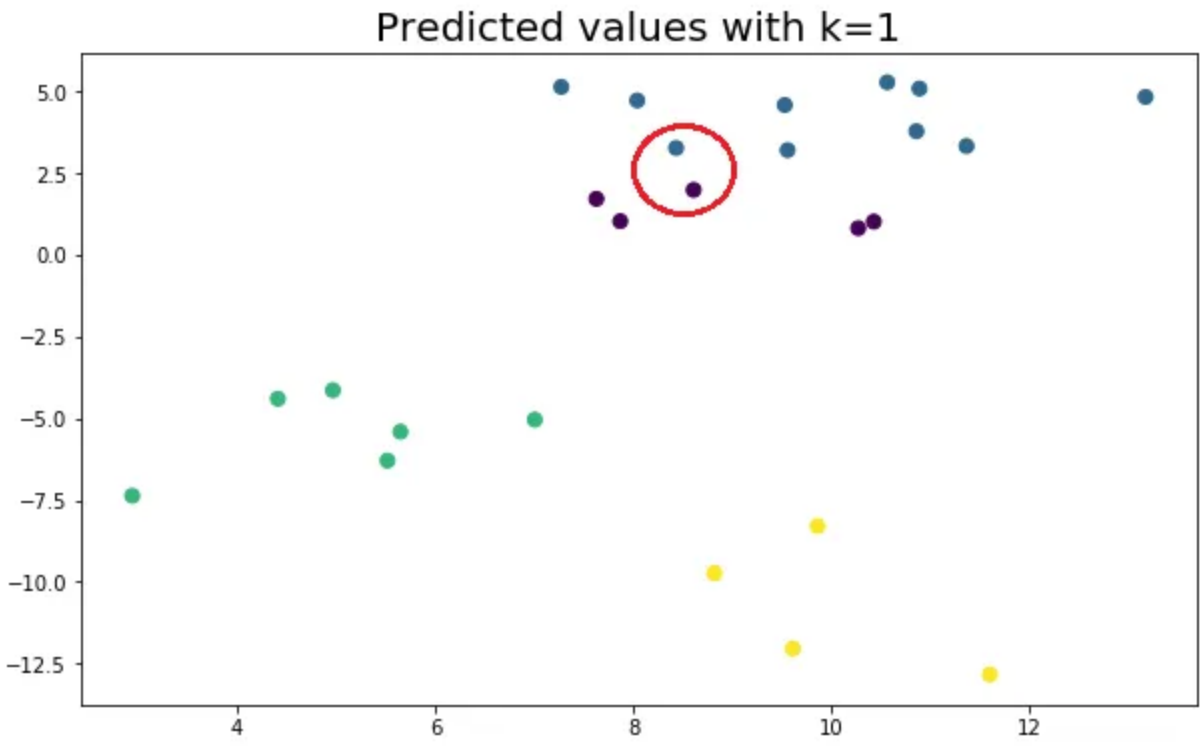
```
knn5.fit(X_train, y_train)
knn1.fit(X_train, y_train)
```

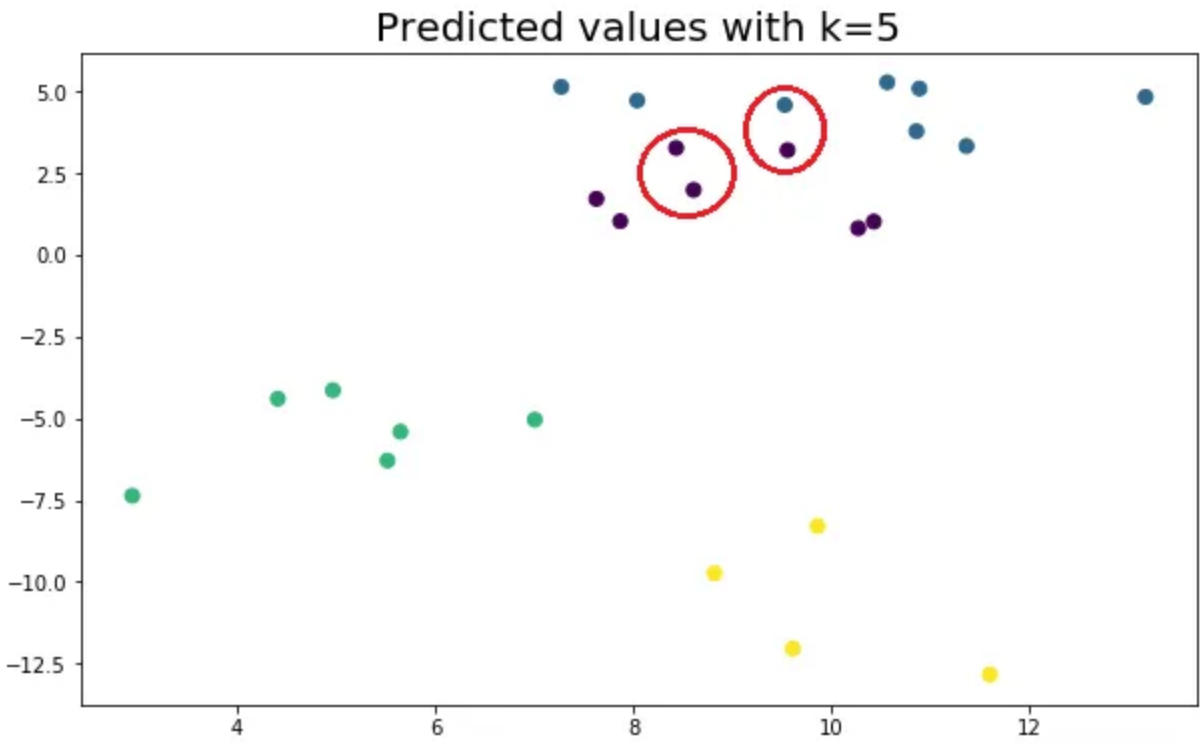
Then we predict the target values in the test set and compare with actual values.

```
y_pred_5 = knn5.predict(X_test)
y_pred_1 = knn1.predict(X_test)
```

In order to see the effect of k values, let's visualize test set and predicted values with k=5 and k=1.


Original data
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
4
6
8
10
12


Predicted values with k=1
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
4
6
8
10
12



The result seems to be very similar because we used a substantially small dataset. However, even on small datasets, different k values predict some points differently.

How to find the best k value

- **k=1**: The model is too specific and not generalized well. It also tends to be sensitive to noise. The model accomplishes a high accuracy on train set but will be a poor predictor on new, previously unseen data points. Therefore, we are likely to end up with an overfit model.

- **k=100**: The model is too generalized and not a good predictor on both train and test sets. This situation is known as underfitting.

How do we find the optimum k value? Scikit-learn provides **GridSearchCV** function that allows us to easily check multiple values for k. Let's go over an example using a dataset available under scikit-learn datasets module.

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import GridSearchCV
```

```
cancer = load_breast_cancer()
(X_cancer, y_cancer) = load_breast_cancer(return_X_y = True)
```

After we import the required libraries and load the dataset, we can create a GridSearchCV object.

```
knn_grid = GridSearchCV(estimator = KNeighborsClassifier(),
                        param_grid={'n_neighbors': np.arange(1,20)}, cv=5)
```

```
knn_grid.fit(X_cancer, y_cancer)
```

We do not need to split the the datasets because **cv** parameter splits the dataset. The default value for cv parameter is 5 but I explicitly wrote it to emphasize why we don't need to use train_test_split.

cv=5 basically splits the dataset into 5 subsets. GridSearchCV does 5 iterations and use 4 subsets for training and 1 subset for testing at each time. In this way, we are able to use all data points for both training and testing.

We can check which parameters give us best results using **best_params_** method:

```
knn_grid.best_params_
```

```
{'n_neighbors': 12}
```

In this case, the optimum value for k is 12.

Pros and cons of k-Nearest-Neigbors

Pros

- Simple and easy to interpret

Thank you for reading. Please let me know if you have any feedback.

My other posts on machine learning algorithms

- Naive Bayes Classifier — Explained
- Logistic Regression — Explained
- Support Vector Machine — Explained
- Decision Trees and Random Forests — Explained
- Gradient Boosted Decision Trees — Explained
- Principal Component Analysis — Explained
- Predicting Used Car Prices with Machine Learning

Machine Learning Data Analysis Data Science Predictive Analytics

Supervised Learning

near tasks.

assification with multiple classes

- Works on both classification and regression tasks

Cons

- Becomes very slow as the number of data points increases because the model needs to store all data points.

emory efficient

- ensitive to outliers. Outliers also have a vote!

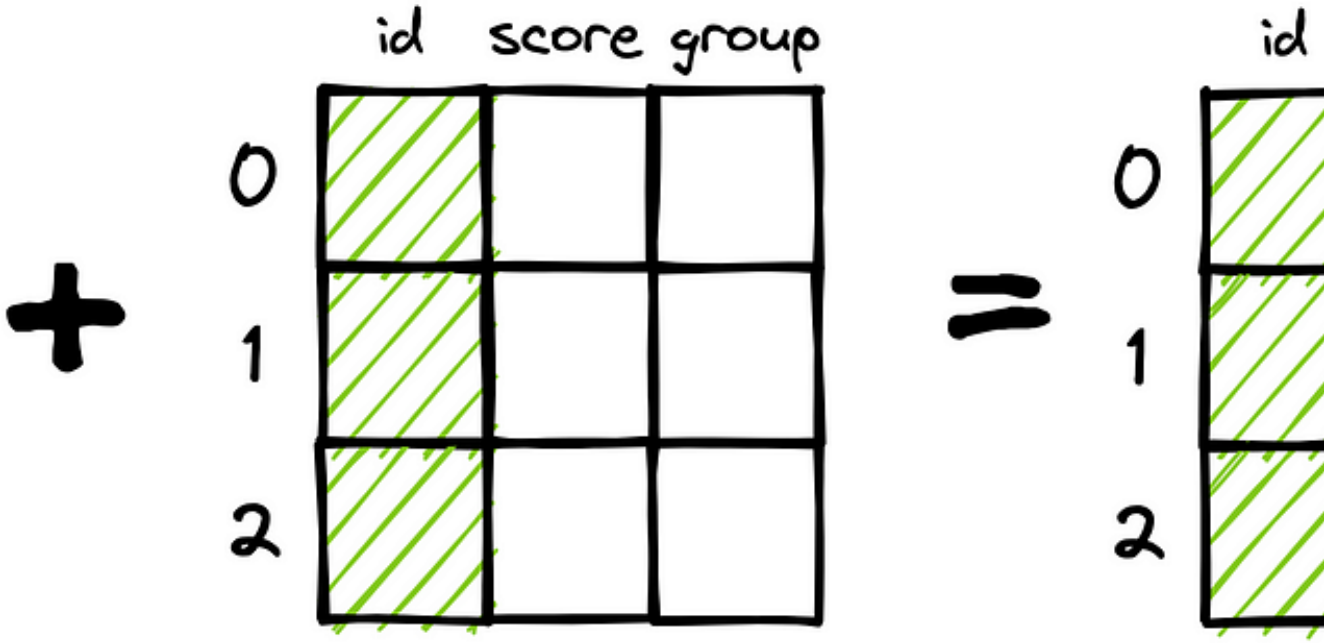
Follow

Written by Soner Yıldırım

23K Followers · Writer for Towards Data Science

Data Scientist | Top 10 Writer in AI and Data Science | linkedin.com/in/soneryildirim/ | twitter.com/snr14

More from Soner Yıldırım and Towards Data Science



Soner Yıldırım in Towards Data Science

20 Examples to Master Merging DataFrames in Python Pandas

A comprehensive practical guide.

✨ · 10 min read · May 30

390 3

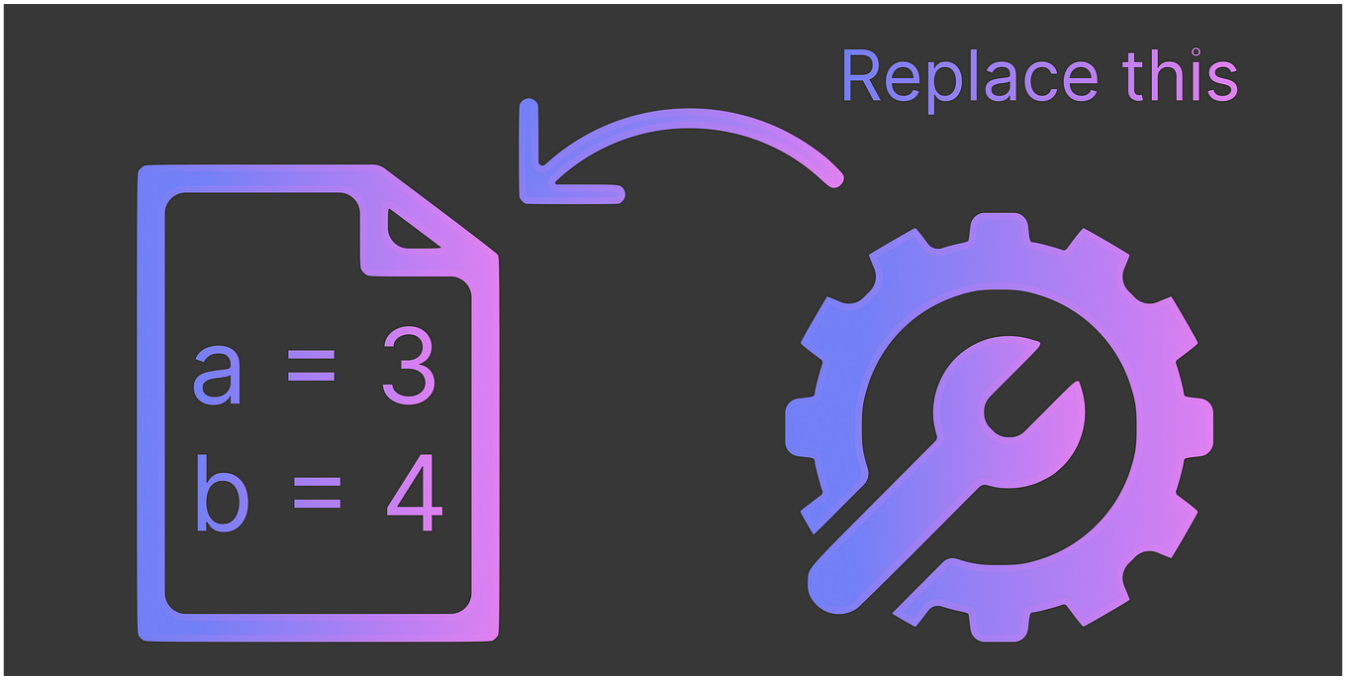
Jacob Marks, Ph.D. in Towards Data Science

How I Turned My Company's Docs into a Searchable Database with OpenAI

And how you can do the same with your docs

15 min read · Apr 25

4.1K 49



Khuyen Tran in Towards Data Science

Stop Hard Coding in a Data Science Project — Use Config Files Instead

And How to Efficiently Interact with Config Files in Python

· 6 min read · May 26

1.6K 20

Soner Yıldırım in Towards Data Science

Use LangChain's Output Parser with ChatGPT for Structured Outputs

Explained with an example use case.

✦ · 6 min read · Jun 6

131 2

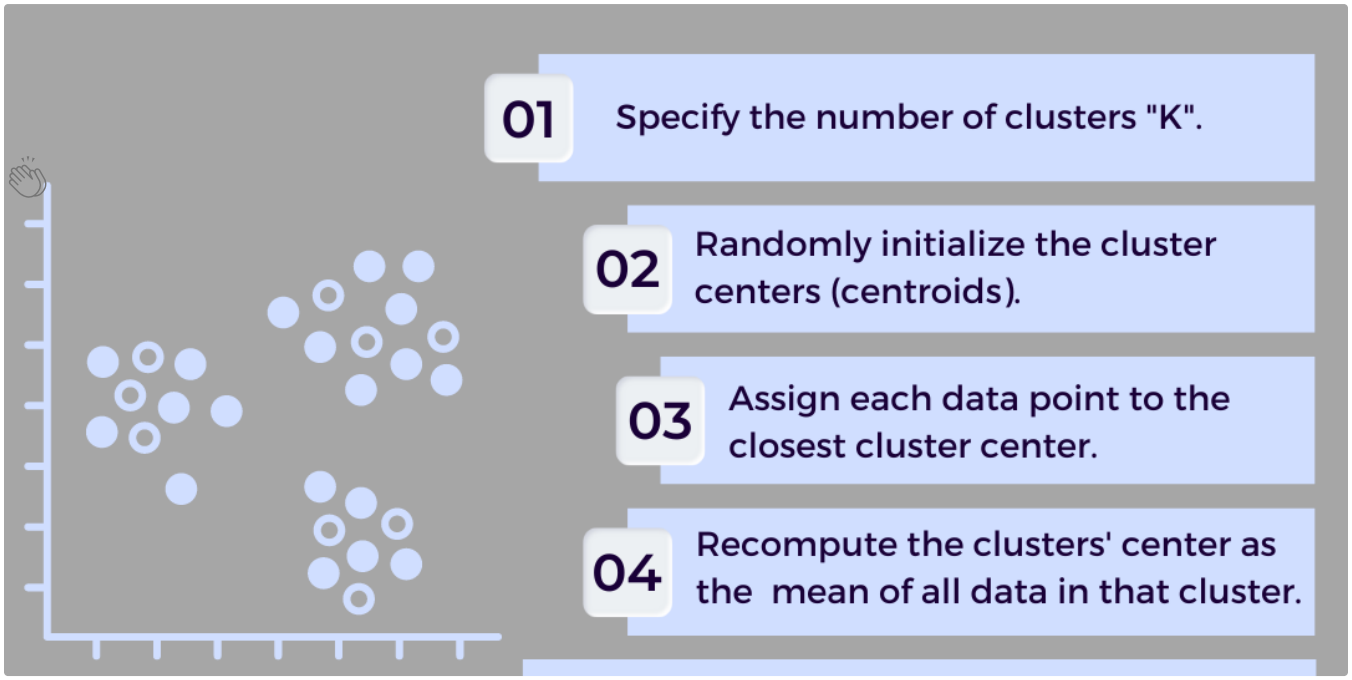
See all from Soner Yıldırım

See all from Towards Data Science

Recommended from Medium

Piero Paialunga in Towards Data Science

Naive Bayes Classifier from Scratch, with Python



Zoumana Keita in Towards Data Science

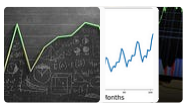
How to Perform KMeans Clustering Using Python

A complete overview of the KMeans clustering and implementation with Python

· 7 min read · Jan 17

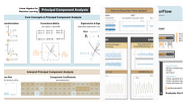
73

Lists



Predictive Modeling w/ Python

18 stories · 11 saves

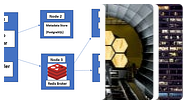


Practical Guides to Machine Learning

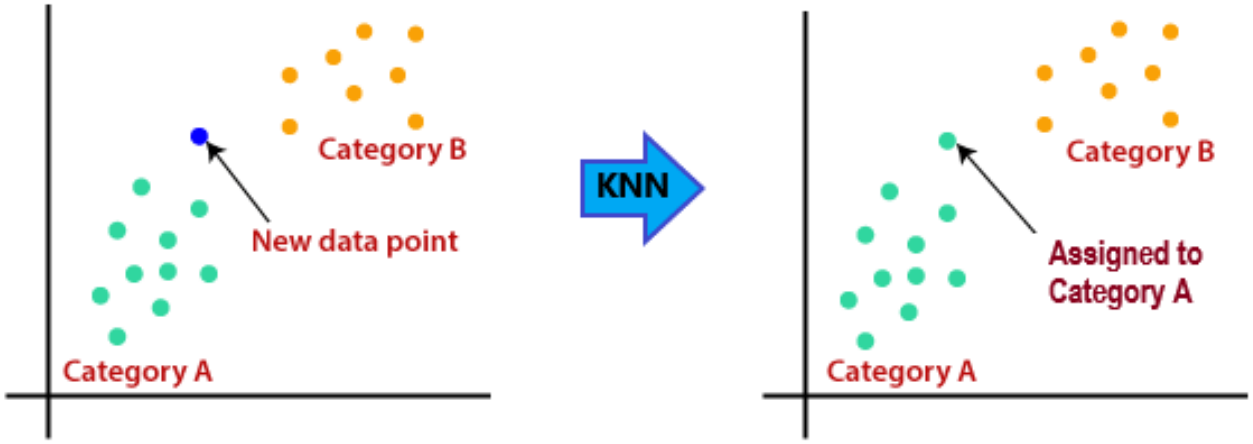
10 stories · 25 saves

Natural Language Processing

346 stories · 7 saves



New_Reading_List
173 stories · 1 save



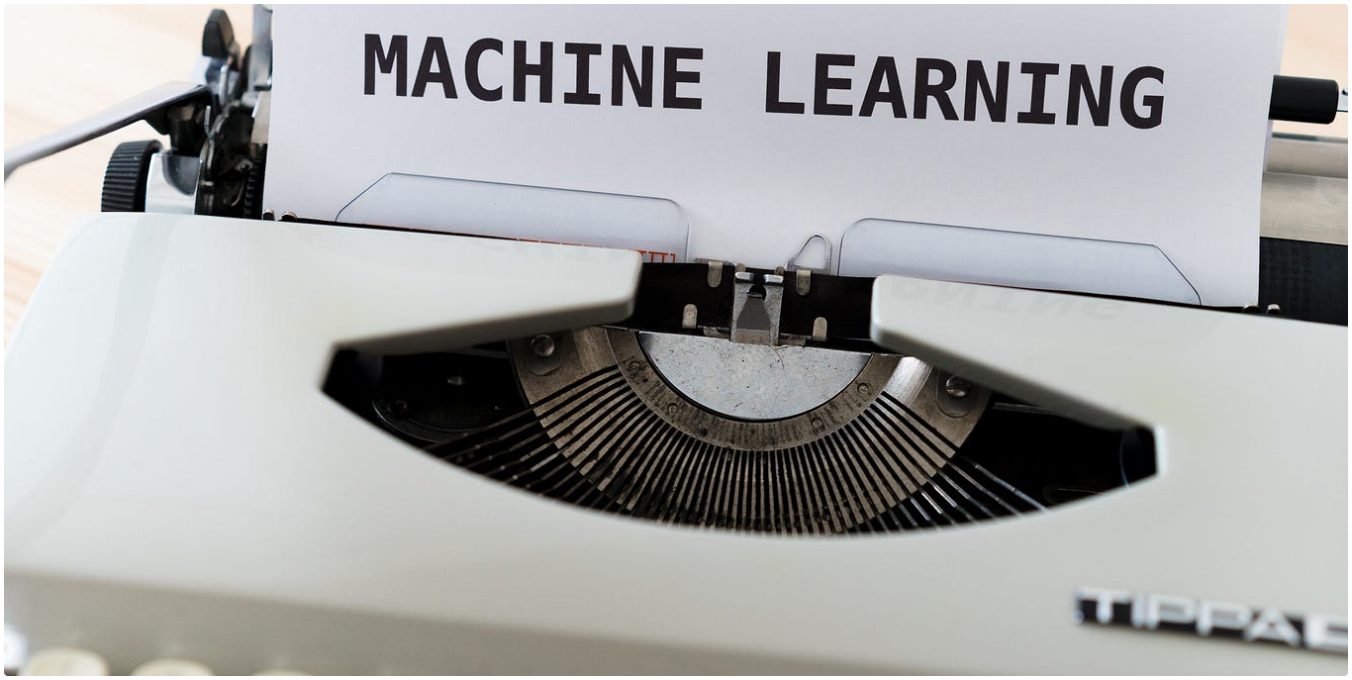
Peter Karas in Artificial Intelligence in Plain English

K-Nearest Neighbors (KNN) in Depth

K-NN, machine learning, classification

· 4 min read · Feb 5

98



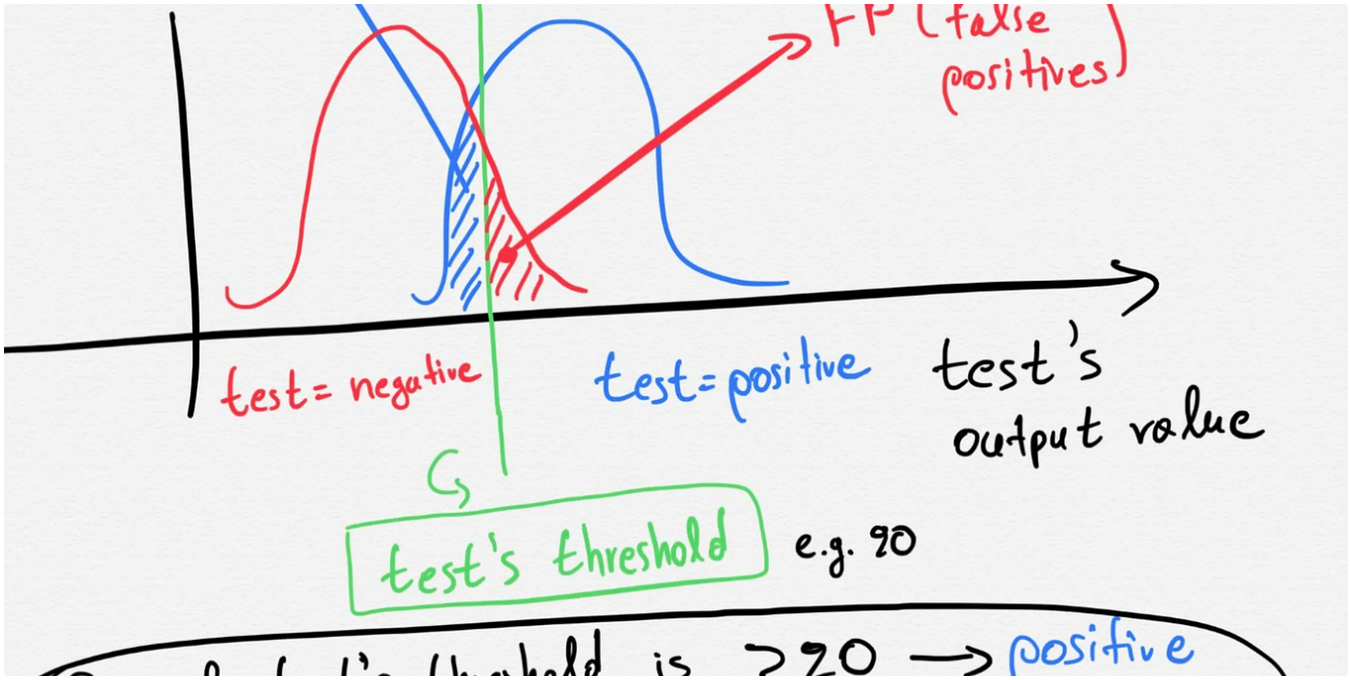
Aaron Zhu in Towards Data Science

Essential Evaluation Metrics for Classification Problems in Machine Learning

A Comprehensive Guide to Understanding and Evaluating the Performance of Classification Models

· 10 min read · Mar 10

21



Serafeim Loukas, PhD in Towards AI

How To Estimate FP, FN, TP, TN, TPR, TNR, FPR, FNR & Accuracy for Multi-Class Data in Python in 5...

In this post, I explain how someone can read a confusion matrix and how to extract several performance metrics for a multi-class...

· 6 min read · Feb 5

64

 Matt Chapman in Towards Data Science

The Portfolio that Got Me a Data Scientist Job

Spoiler alert: It was surprisingly easy (and free) to make

✦ · 10 min read · Mar 24

3.1K 46

See more recommendations