



Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



Makine Öğrenmesi — KNN (K-Nearest Neighbors) Algoritması Nedir?



Evren Arslan · [Follow](#)

5 min read · May 19, 2020



Share



More

Makine öğrenmesi ile çözülebilen problemlerden biri çok geniş kullanım alanı ile sınıflandırma problemleridir. Günümüzdeki bir çok problem bir şekilde sınıflandırma problemi olarak tasarlanıp çözülebilmektedir. Aşağıdaki problemler sınıflandırma problemlerine örnek olarak verilebilir.

- Müşterinin verilen krediyi ödeyip ödeyemeyeceği (Customer Default Risk)
- Müşterinin aldığı servisi bırakıp bırakmayacağı (Customer Churn)
- Müşteri segmentasyon (Customer Segmentation)
- Kişiye özel ilaç tespiti
- E-posta spam filitrelemesi
- Görüntü tanıma (Image Recognition)
- El yazısı tanıma (Handwriting Recognition)
- Biyometrik tanımlama (Biometric Identification)

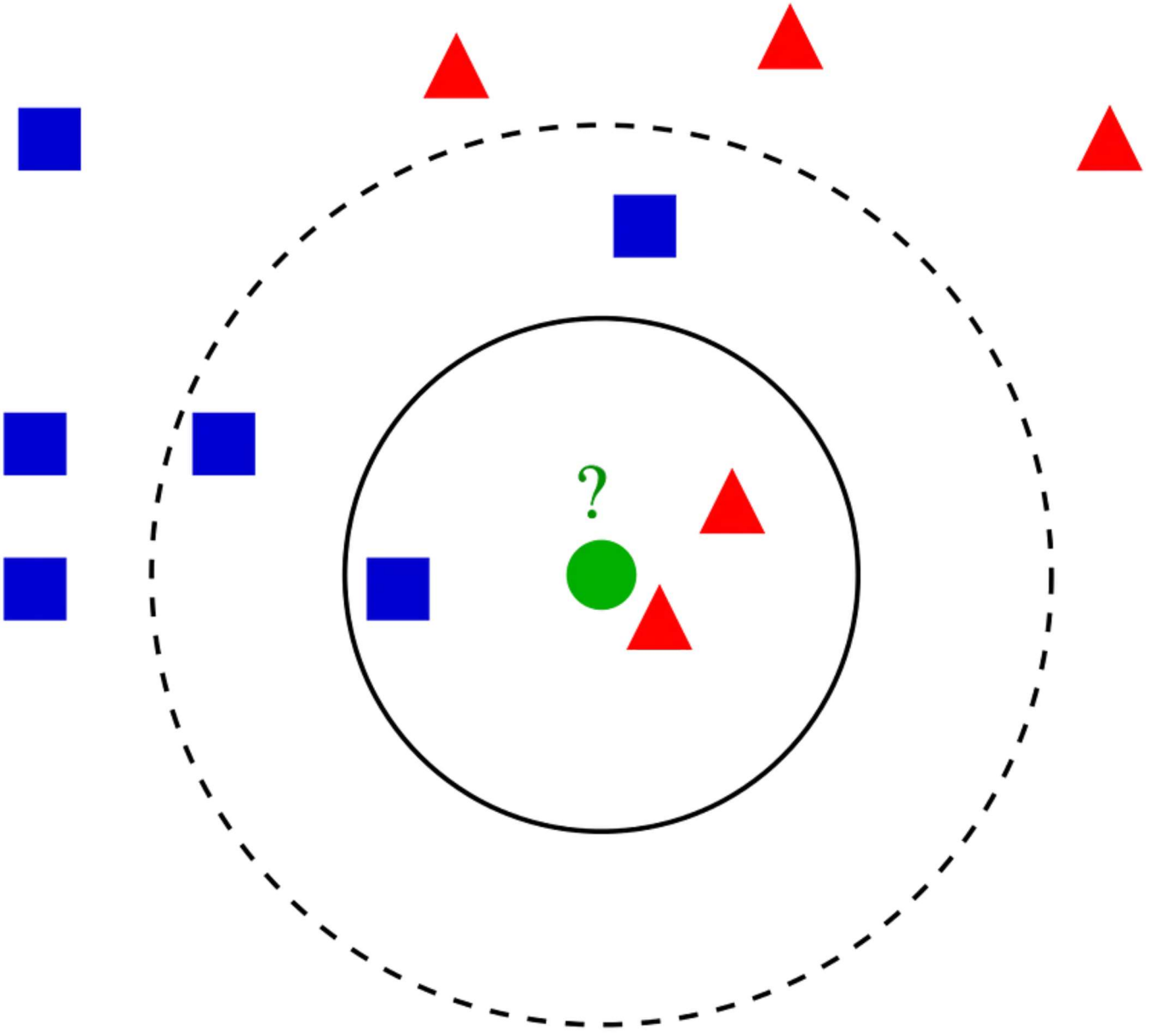
Bu problemleri çözmek üzere üretilmiş bir dolu algoritma bulunmaktadır. bunlar içinden KNN (K-Nearest Neighbors) Algoritmasından bir örnek ile bahsedeceğim.

KNN en basit anlamı ile içerisinde tahmin edilecek değerin bağımsız değişkenlerinin oluşturduğu vektörün en yakın komşularının hangi sınıfta yoğun

olduğu bilgisi üzerinden sınıfını tahmin etmeye dayanır.

KNN (K-Nearest Neighbors) Algoritması iki temel değer üzerinden tahmin yapar;

- **Distance (Uzaklık):** Tahmin edilecek noktanın diğer noktalara uzaklığı hesaplanır. Bunun için Minkowski uzaklık hesaplama fonksiyonu kullanılır.
- **K (komuşuluk sayısı):** En yakın kaç komşu üzerinden hesaplama yapılacağını söyleriz. K değeri sonucu direkt etkileyecektir. K 1 olursa overfit etme olasılığı çok yüksek olacaktır. Çok büyük olursa da çok genel sonuçlar verecektir. Bu sebeple optimum K değerini tahmin etmek problemin asıl konusu olarak karşımızda durmaktadır. K değerinin önemini aşağıdaki grafik çok güzel bir şekilde göstermektedir. Eğer $K=3$ (düz çizginin olduğu yer) seçersek sınıflandırma algoritması ? işareti ile gösterilen noktayı, kırmızı üçgen sınıfı olarak tanımlayacaktır. Fakat $K=5$ (kesikli çizginin olduğu alan) seçersek sınıflandırma algoritması, aynı noktayı mavi kare sınıfı olarak tanımlayacaktır.



KNN (K-Nearest Neighbors) Algoritması ile üretilmiş bir modelin başarımını ölçmek için genel olarak kullanılan 3 adet indikatör vardır.

- **Jaccard Index:** Doğru tahmin kümesi ile gerçek değer kümesinin kesişim kümesinin bunların birleşim kümesine oranıdır. 1 ile 0 arası değer alır. 1 en iyi başarımlar anlamına gelir.
- **F1-Score:** Confusion Matrisi üzerinden hesaplanan Precision ve Recall değerlerinden hesaplanır. $Pre = TP / (TP + FP)$ $Rec = TP / (TP + FN)$ $F1-Score = 2((Pre \cdot Rec) / (Pre + Rec))$ 1 ile 0 arası değer alır. 1 en iyi başarımlar anlamına gelir.
- **LogLoss:** Logistic Regresyon sonunda tahminlerin olasılıkları üzerinden LogLoss değeri hesaplanır. 1 ile 0 arası değer alır. Yukarıdaki iki değerden farklı olarak 0 en iyi başarımlar anlamına gelir.

KNN (K-Nearest Neighbors) Algoritması İçin Basit Bir Örnek

Örnek olarak bir telefon şirketinin örnek verileri üzerinden müşteri kategorisini tahmin eden bir model inşa etmeye çalışacağız. Tahmin etmeye çalışacağımız veri “custcat” kolonunda bulunuyor. Bu kolondaki her bir sayısal değerin karşılığı aşağıdaki gibidir;

- 1- Basic Service
- 2- E-Service
- 3- Plus Service
- 4- Total Service

Öncelikli olarak verinin pandas ile yüklenmesi ve incelenmesi adımlarını yapacağız.

```
import itertools
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter
import pandas as pd
import numpy as np
import matplotlib.ticker as ticker
from sklearn import preprocessing
%matplotlib inline
```

Import edilecek modüller.

```
df = pd.read_csv('teleCust1000t.csv')
df.head()
```

	region	tenure	age	marital	address	income	ed	employ	retire	gender	reside	custcat
0	2	13	44	1	9	64.0	4	5	0.0	0	2	1
1	3	11	33	1	7	136.0	5	5	0.0	0	6	4
2	3	68	52	1	24	116.0	1	29	0.0	1	2	3
3	2	33	33	0	12	33.0	2	0	0.0	1	1	1
4	2	23	30	1	9	30.0	1	2	0.0	0	4	3

Yüklenen Dataframe içerisindeki data

Hangi sınıfta kaç tane müşteri olduğuna bakalım. Bu inceleme ele aldığımız verinin model oluşturmak için uygunluğu konusunda bize fikir verecektir. Eğer sınıflara göre dağılımı düzgün olmayan bir veri setimiz var ise doğru bir model geliştirmemiz mümkün olmayacaktır.

```
df['custcat'].value_counts()

3    281
1    266
4    236
2    217
Name: custcat, dtype: int64
```

Hedef veri dağılımı

Öncelikli olarak verimizi analiz için hazırlamak gerekiyor. Scikit-learn ile çalışabilmek için bağımlı değişken ve bağımsız değişkenlerimizi ayrı ayrı numpy array tipine dönüştürmemiz gerekiyor.

```
df.columns # Bir sonraki kod satırında kullanmak için kolon isimlerini yazdırıyorum.
```

```
Index(['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed',
       'employ', 'retire', 'gender', 'reside', 'custcat'],
      dtype='object')
```

```
# custcat kolonu hariç diğer kolonlardan bağımsız değişkenlerden oluşan bir numpy array yarattım.
```

```
X = df[['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed', 'employ', 'retire', 'gender', 'reside']].values
X[0:5]
```

```
array([[ 2., 13., 44., 1., 9., 64., 4., 5., 0., 0., 2.],
       [ 3., 11., 33., 1., 7., 136., 5., 5., 0., 0., 6.],
       [ 3., 68., 52., 1., 24., 116., 1., 29., 0., 1., 2.],
       [ 2., 33., 33., 0., 12., 33., 2., 0., 0., 1., 1.],
       [ 2., 23., 30., 1., 9., 30., 1., 2., 0., 0., 4.]])
```

```
# custcat feature kullanarak numpy array yarattım.
```

```
y = df['custcat'].values
y[0:5]
```

```
array([1, 4, 3, 1, 3], dtype=int64)
```

Verinin array haline dönüştürülmesi

KNN gibi noktalar arası mesafe temelli algoritmalarından daha başarılı sonuç elde etmek için veriler normalize edilir.

```
X=preprocessing.StandardScaler().fit(X).transform(X.astype(float))
X[0:5]
```

```
array([[ -0.02696767, -1.055125 ,  0.18450456,  1.0100505 , -0.25303431,
        -0.12650641,  1.0877526 , -0.5941226 , -0.22207644, -1.03459817,
        -0.23065004],
       [ 1.19883553, -1.14880563, -0.69181243,  1.0100505 , -0.4514148 ,
         0.54644972,  1.9062271 , -0.5941226 , -0.22207644, -1.03459817,
         2.55666158],
       [ 1.19883553,  1.52109247,  0.82182601,  1.0100505 ,  1.23481934,
         0.35951747, -1.36767088,  1.78752803, -0.22207644,  0.96655883,
        -0.23065004],
       [ -0.02696767, -0.11831864, -0.69181243, -0.9900495 ,  0.04453642,
        -0.41625141, -0.54919639, -1.09029981, -0.22207644,  0.96655883,
        -0.92747794],
       [ -0.02696767, -0.58672182, -0.93080797,  1.0100505 , -0.25303431,
        -0.44429125, -1.36767088, -0.89182893, -0.22207644, -1.03459817,
         1.16300577]])
```

Verinin Normalizasyonu

Bir model geliştirdikten sonra sonuçlarını bildiğimiz veriler ile modelin test edilmesi oldukça önemlidir. Bu testin ise eğitim yaparken kullanılan veri seti dışında bir veri seti olması, bize gerçek hayatta modelimizin nasıl bir performans göstereceğini söyleyecektir. Bu amaçla elimizdeki toplam veri setini train/test (eğitim/test) olarak ikiye ayıracağız.

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2, random_state=4)
print("Train set:", X_train.shape, y_train.shape)
print("Test set:",X_test.shape,y_test.shape)
```

```
Train set: (800, 11) (800,)
Test set: (200, 11) (200,)
```

Eğitim/Test veri setinin ayrılması

KNN (K-Nearest Neighbors) Algoritmasını kullanarak sınıflandırma işlemine başlayacağız. İlk olarak $k=4$ (*En yakın 4 komşu içerisindeki yoğunluğa göre karar verecek*) olarak modelimizi eğitelim.

```
from sklearn.neighbors import KNeighborsClassifier

k=4
neigh=KNeighborsClassifier(n_neighbors=k).fit(X_train,y_train)
neigh

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                     weights='uniform')
```

KNN Algoritması ile modelin eğitilmesi

neigh ismini verdiğimiz modelimizi eğittik. Şimdi bu model ile prediction yapalım ve sonuçların performansına bakalım.

```
y_hat=neigh.predict(X_test)
y_hat[0:5]

array([1, 1, 3, 2, 4], dtype=int64)
```

KNN Algoritması ile tahmin

Multilabel (Çoklu sonuç) tahmin yapılması durumunda bunun accuracy (doğruluğu) değerlendirmesi için *accuracy classification score* fonksiyonu kullanılabilir. Bu

fonksiyon *jaccard index* benzeri bir sonuç verir.

```
from sklearn import metrics
print("Eğitim verisi doğruluğu:", metrics.accuracy_score(y_train,neigh.predict(X_train)))
print("Test verisi doğruluğu:", metrics.accuracy_score(y_test,y_hat))
```

Eğitim verisi doğruluğu: 0.5475
Test verisi doğruluğu: 0.32

KNN Algoritması Performansı

Eğer K değerini 4 yerine 6 seçseydik sonuç nasıl değişirdi. Bunu görmek için modelimizi yeniden eğitip sonuçlarını karşılaştıralım.

```
k=6
neigh_6=KNeighborsClassifier(n_neighbors=k).fit(X_train,y_train)
print("Eğitim verisi doğruluğu:", metrics.accuracy_score(y_train,neigh_6.predict(X_train)))
print("Test verisi doğruluğu:", metrics.accuracy_score(y_test,neigh_6.predict(X_test)))
```

Eğitim verisi doğruluğu: 0.51625
Test verisi doğruluğu: 0.31

K=6 için performans

Sonuç daha iyi olmadı. Peki doğru k değerini seçtiğimizden nasıl emin olabiliriz. Daha iyi bir soru ile optimum k değerini nasıl bulabiliriz. Bunun için bir aralık için tüm k değerleri için modelimizi eğitim accuracy score üzerinden bir karşılaştırma yapıp olası en iyi k değeri bulunabilir.

Örnek olsun diye k değerinin 1–10 arasındaki durumlarını inceleyelim.

```
Ks = 10
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
ConfustionMx = [];
for n in range(1,Ks):

    #Train Model and Predict
    neigh = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)
    yhat=neigh.predict(X_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)

    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])

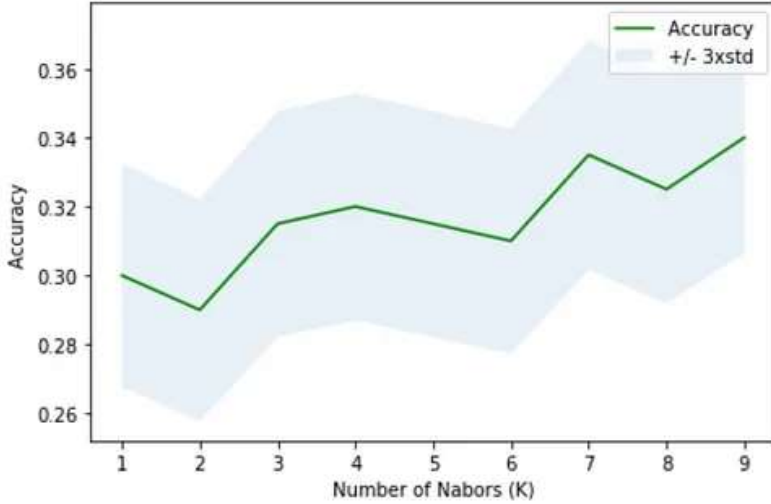
mean_acc

array([0.3 , 0.29 , 0.315, 0.32 , 0.315, 0.31 , 0.335, 0.325, 0.34 ])
```

Farklı K değerleri için performans

Elde ettiğimiz sonuçlara grafik üzerinden bakarak optimum K değerini görebiliriz.

```
plt.plot(range(1,Ks),mean_acc,'g')
plt.fill_between(range(1,Ks),mean_acc - 1 * std_acc,mean_acc + 1 * std_acc, alpha=0.10)
plt.legend(('Accuracy ', '+/- 3xstd'))
plt.ylabel('Accuracy ')
plt.xlabel('Number of Nabors (K)')
plt.tight_layout()
plt.show()
```



Grafikten görebileceğimiz gibi en iyi değer K=9 olduğunda gerçekleşmiş. Bunu aşağıdaki şekilde grafik çizdirmeden de görebiliriz.

```
print( "En yüksek Doğruluk=", mean_acc.max(), "K=", mean_acc.argmax()+1,"olduğunda gerçekleşti.")
```

En yüksek Doğruluk= 0.34 K= 9 olduğunda gerçekleşti.

Basit bir örnek üzerinden KNN (K-Nearest Neighbors) Algoritmasını anlatmaya çalıştım. Ekran görüntülerini gördüğünüz kodlara ve daha detaylı bilgilere aşağıdaki linklerden ulaşabilirsiniz.

evrenarslan's gists

Instantly share code, notes, and snippets.

gist.github.com

KNN Classification using Scikit-learn

K Nearest Neighbor(KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning...

www.datacamp.com

1.6. Nearest Neighbors - scikit-learn 0.23.1 documentation

provides functionality for unsupervised and supervised neighbors-based learning methods. Unsupervised nearest neighbors...

scikit-learn.org

Knn

Knn Algorithm

Knn Sınıflandırma

Sınıflandırma

Makine Öğrenmesi



Follow

Written by Evren Arslan

76 Followers

I'm Ece and Ali's father and Ozgul's husband. I'm a graduate of physics. I love reading books about mathematics and history also learning new things online.

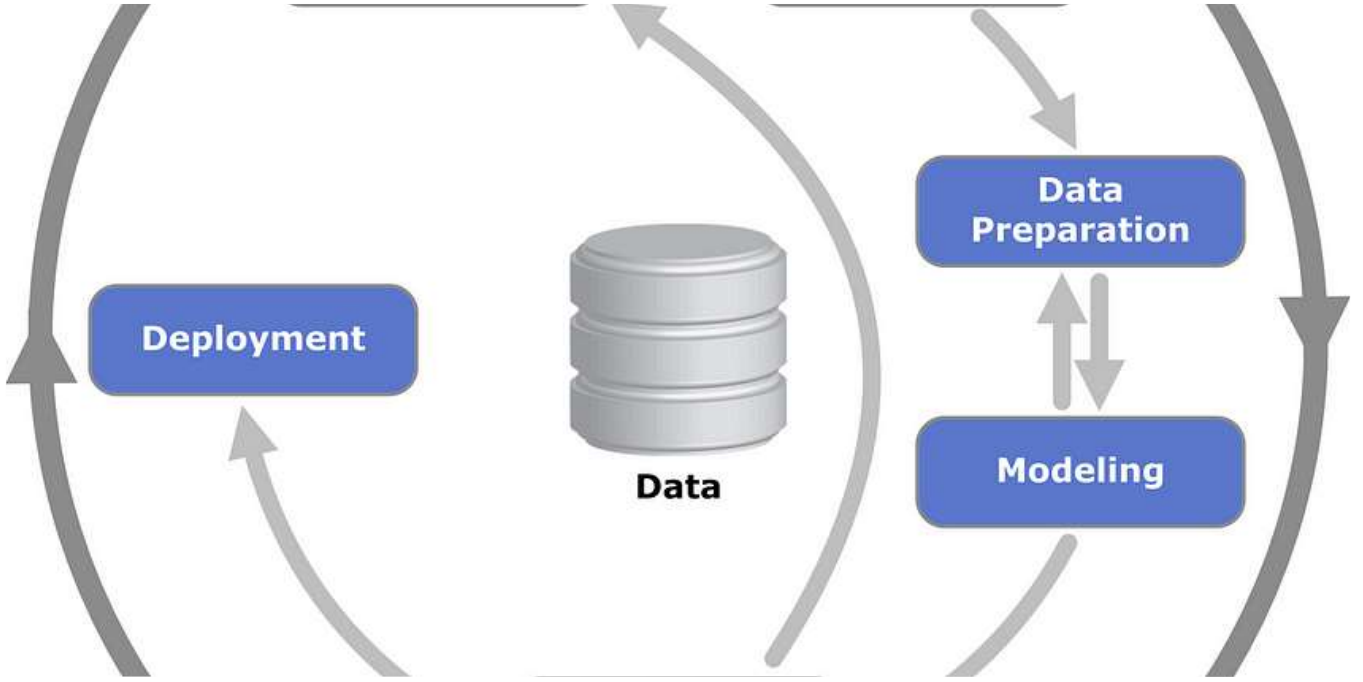
More from Evren Arslan

Zamana bağlı olarak değişen verilerin analizinde kullanılan yöntemlerden birisi de Ortalamaya Dönüş yöntemidir. Yöntem kısaca zamana bağlı...

3 min read · Dec 24, 2021



101



Evren Arslan

CRISP-DM nedir?

CRISP-DM metodolojisi, çok çeşitli iş uygulamaları ve endüstrilerde veri madenciliğinin kullanımını artırmayı ve doğru sonuçları elde...

3 min read · Jan 12, 2020



11





Evren Arslan

Gerçekte rastlantı var mı?

Kimi zaman bir çoğumuzun aklına gelen soru; raslantı gibi görünen durumlar gerçektende bir düzen içermiyor mu? Bir süre önce gördüğüm...

3 min read · Feb 2, 2020



2



See all from Evren Arslan

Recommended from Medium



Rohan Kumar Bohara

How to find the optimal value of K in KNN?

Introduction

7 min read · Oct 29, 2023



7



Vishvaasswaminathan

Weighted KNN Algorithm

Weighted k-NN is a modified version of k nearest neighbors. One of the many issues that affect the performance of the k-NN algorithm is the...

2 min read · Sep 28, 2023

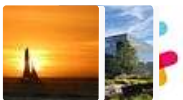


Lists



Staff Picks

550 stories · 616 saves



Stories to Help You Level-Up at Work

19 stories · 405 saves



Self-Improvement 101

20 stories · 1164 saves



Productivity 101

20 stories · 1066 saves



Can Benli

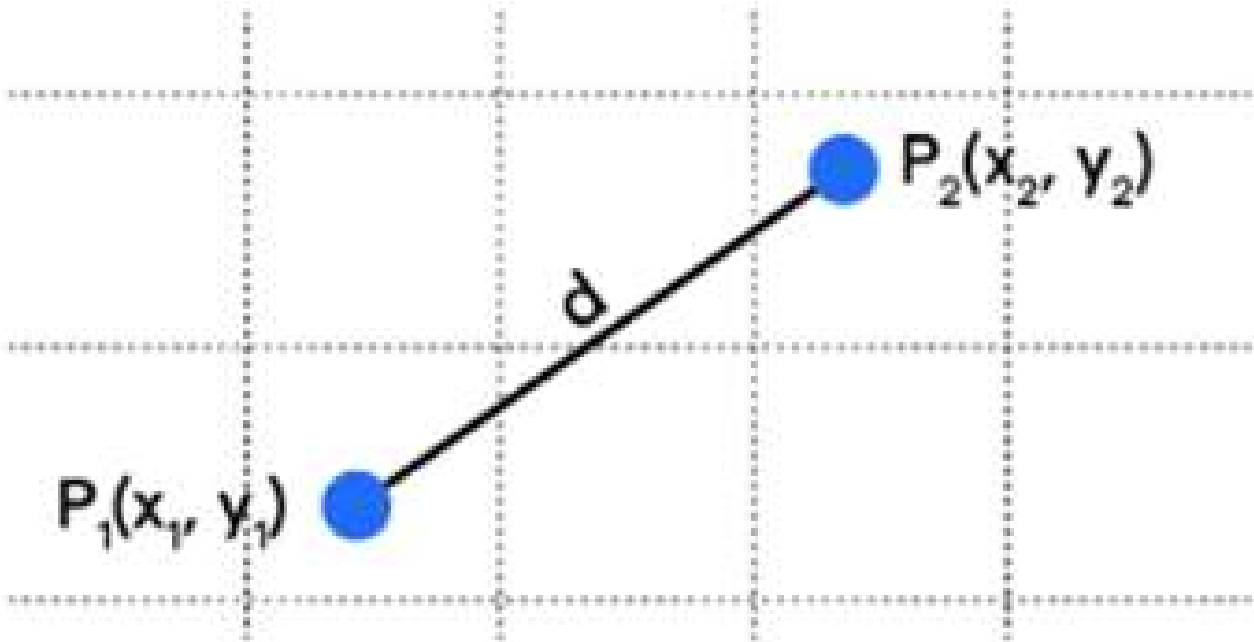
K-Nearest Neighbors (KNN)


K-Nearest Neighbors (KNN) algorithm is an approach used in regression and classification problems in supervised machine learning models...

6 min read · Oct 18, 2023

 249  1



 Shivagoud

K-Nearest Neighbors (KNN) Algorithm

Introduction

5 min read · Dec 26, 2023



 Robertbrown

A Complete Guide To Scrape Instagram Post Data, Photos, Likes & Comments

The official API of Instagram lets you access your posts and comments on the platform. But the Instagram scraping API doesn't permit you to...

6 min read · Jul 14, 2023





Anthony Baum in Towards Data Science

Example Applications of K-Nearest-Neighbors

Why the simple algorithm is more practical than you think

★ • 6 min read • Aug 4, 2023



119



See more recommendations