K-Nearest Neighbors(KNN)

*In this article we will understand what is K-nearest neighbors, how does this algorithm work, what are the pros and cons of KNN. Difference between K-means and KNN and finally an example to use KNN using Python*

Prerequisites: [Machine Learning](https://medium.com/datadriveninvestor/machine-learning-demystified-4b41c3a55c99), [K-Means](https://medium.com/datadriveninvestor/k-means-clustering-6f2dc458cce8) (good to know)

***What is K- Nearest neighbors?***

K- Nearest Neighbors is a

* **Supervised machine learning algorithm**as target variable is known
* **Non parametric**as it does not make an assumption about the underlying data distribution pattern
* **Lazy algorithm** as KNN does not have a training step. All data points will be used only at the time of prediction. With no training step, prediction step is costly. An eager learner algorithm eagerly learns during the training step.
* Used for both **Classification and Regression**
* Uses **feature similarity** to predict the cluster that the new point will fall into.

let’s take a real life example and understand.

You moved to a new neighborhood and want to be friends with your neighbors. You start to socialize with your neighbors. Yo decide to pick neighbors that match your thinking, interests and hobbies. Here thinking, interest and hobby are features. You decide your neighborhood friend circle based on interest, hobby and thinking similarity. This is analogous to how KNN works

***What is K is K nearest neighbors?***

K is a number used to identify similar neighbors for the new data point.

Referring to our example of friend circle in our new neighborhood. We select 3 neighbors that we want to be very close friends based on common thinking or hobbies. In this case K is 3.

KNN takes K nearest neighbors to decide where the new data point with belong to. This decision is based on feature similarity.

***How do we chose the value of K?***

Choice of K has a drastic impact on the results we obtain from KNN.

we can take the test set and plot the accuracy rate or F1 score against different values of K.

We see a high error rate for test set when K=1. Hence we can conclude that model overfits when k=1.

For a high value of K, we see that the F1 score starts to drop. The test set reaches a minimum error rate when k=5. This is very similar to the elbow method used in [K-means](https://medium.com/datadriveninvestor/k-means-clustering-6f2dc458cce8).

Value of K at the elbow of test error rate gives us the optimal value of K.



Plot of K against F1 score for cars database used in python example

We can evaluate accuracy of KNN classifier using [K fold cross validation.](https://medium.com/datadriveninvestor/k-fold-and-other-cross-validation-techniques-6c03a2563f1e)

***How does KNN work?***

We have age and experience in an organization along with the salaries. We want to predict the salary of a new candidate whose age and experience is available.

Step 1: **Choose a value for K**. K should be an odd number.

Step2: **Find the distance of the new point to each of the training data**.

Step 3:**Find the K nearest neighbors to the new data point**.

Step 4: For classification, count the number of data points in each category among the k neighbors. **New data point will belong to class that has the most neighbors**.

For regression, value for the **new data point will be the average of the k neighbors**.



K =5. We will average salary of the 5 nearest neighbors to predict the salary of the new data point

***How is the distance calculated?***

Distance can be calculated using

* **Euclidean distance**
* **Manhattan distance**
* **Hamming Distance**
* **Minkowski Distance**

Euclidean distance is the square root of the sum of squared distance between two points. It is also known as L2 norm.



Manhattan distance is the sum of the absolute values of the differences between two points



Hamming distance is used for categorical variables. In simple terms it tells us if the two categorical variables are same or not.



Minkowski distance is the used to find distance similarity between two points. When p=1, it becomes Manhattan distance and when p=2, it becomes Euclidean distance



***What are the Pros and Cons of KNN?***

**Pros of K Nearest Neighbors**

* Simple algorithm and hence easy to interpret the prediction
* Non parametric, so makes no assumption about the underlying data pattern
* used for both classification and Regression
* Training step is much faster for nearest neighbor compared to other machine learning algorithms

**Cons of K Nearest Neighbors**

* KNN is computationally expensive as it searches the nearest neighbors for the new point at the prediction stage
* High memory requirement as KNN has to store all the data points
* Prediction stage is very costly
* Sensitive to outliers, accuracy is impacted by noise or irrelevant data.

***Is K-means and KNN related or is there a difference between KNN and K-Means?***

* KNN is supervised machine learning algorithm whereas K-means is unsupervised machine learning algorithm
* KNN is used for classification as well as regression whereas K-means is used for clustering
* K in KNN is no. of nearest neighbors whereas K in K-means in the no. of clusters we are trying to identify in the data

***Using cars dataset, we write the Python code step by step for KNN classifier***

We will use cars data available at [https://archive.ics.uci.edu/ml/datasets/car+evaluation](https://archive.ics.uci.edu/ml/datasets/car%2Bevaluation)

It has 6 categorical input variables and one categorical output variable

we will run KNN classifier and check the F1 score to [evaluate the model’s performance](https://medium.com/datadriveninvestor/how-to-evaluate-the-performance-of-a-machine-learning-model-45063a7a38a7).

First import the required libraries. we will add more libraries as we build KNN Classifier

import numpy as np
import pandas as pd
from sklearn import preprocessing

I have saved the data in default Jupyter folder as cars.csv.

Reading the data into dataset\_1

dataset\_1 = pd.read\_csv(‘cars.csv’)

Exploring the data in the dataset and describing all input categorical variables

dataset\_1.describe(include=[‘O’])



describing Cars categorical input features

Handling missing data, if any by dropping the null values

dataset\_1 = dataset\_1.dropna()

We cannot run the classifier on text attributes. we need to convert categorical input features using LabelEncoder. LabelEncoder will converts categorical variables into numbers.

le = preprocessing.LabelEncoder()
dataset\_1['Buying\_1'] = le.fit\_transform(dataset\_1['Buying'])
dataset\_1['Maint\_1'] = le.fit\_transform(dataset\_1['Maint'])
dataset\_1['doors\_1'] = le.fit\_transform(dataset\_1['doors'])
dataset\_1['persons\_1'] = le.fit\_transform(dataset\_1['persons'])
dataset\_1['lug\_boot\_1'] = le.fit\_transform(dataset\_1['lug\_boot'])
dataset\_1['safety\_1'] = le.fit\_transform(dataset\_1['safety'])

printing the first two rows after converting categorical variables to numbers.

dataset\_1.head(2)



dataset\_1 after applying LabelEncoder

Above output shows how the categorical variables are converted to numbers

Now we will create the input features X and output feature Y from the dataset\_1

X= dataset\_1.iloc[:,[7,8,9,10,11,12]]
Y=dataset\_1.iloc[:,6]

Splitting the dataset\_1 to training and test sets. Test set will be 40% and training set will 60% of the dataset\_1

from sklearn.cross\_validation import train\_test\_split
X\_train, X\_test,Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.4)

we will now create KNN classifier.

we need to import ***KNeighborsClassifier***as our problem is a classification problem. For regression problem we use ***KNeighborsRegressor***.

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n\_neighbors=5, metric='minkowski', p=2)
classifier.fit(X\_train, Y\_train)

Here we have specified three parameters n\_neighbors=5. This implies that we will take vore from 5 nearest neighbors for feature similarity.

Distance between the new point and the point in the dataset will be calculated using minkowski and p=2, so the distance is euclidean distance.

Now that we have trained the dataset, we will fit our test set

Y\_pred= classifier.predict(X\_test)

Let’s check the accuracy and F1\_score

from sklearn import metrics
print(metrics.accuracy\_score(Y\_test, Y\_pred))
print(metrics.f1\_score(Y\_test, Y\_pred, average=’weighted’)))0.8930635838150289
0.8840592386040468

we can try with different K values and check how it impacts the accuracy and F1 score

Source Link: <https://medium.com/datadriveninvestor/k-nearest-neighbors-knn-7b4bd0128da7>

Further reading: <https://medium.com/machinelearningalgorithms/k-nearest-neighbors-c9823dca611b>