K-Nearest Neighbors (K-NN)

#100DaysOfMLCode
Day 7
©Avik Jain

K NEAREST NEIGHBOURS

An Intuition to K-NN Classification Algorithm

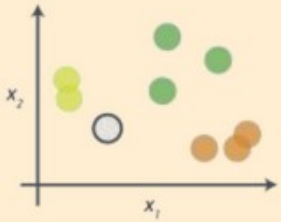
What is k-NN?

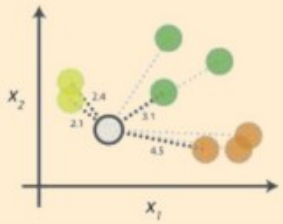
K-Nearest Neighbor algorithm is a simple yet most used classification algorithm. It can also be used for regression.

KNN is non-parametric (means that it does not make any assumptions on the underlying data distribution), instance-based (means that our algorithm doesnt explicitly learn a model. Instead, it chooses to memorize the training instances.) and used in a supervised learning setting.

k-NN is also called a lazy algorithm because it is instance based.

We want to classify the grey point into one of the three classes light green, green and red





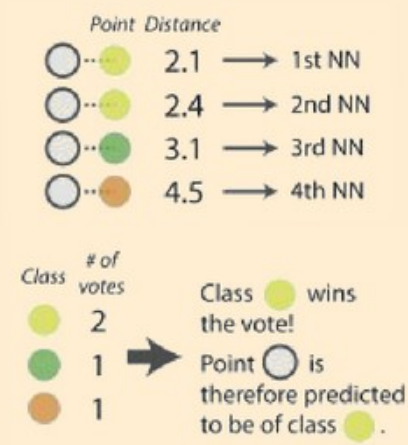
Start by calculating the distance between the grey point and k -nearest points

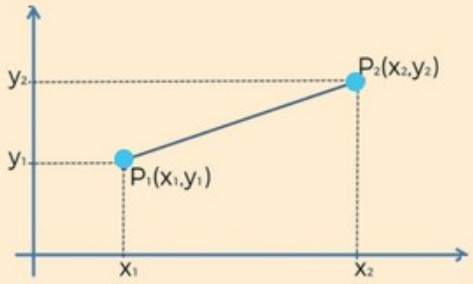
How Does k-NN Algorithm work?

k-NN when used used for classification — the output is a class membership (predicts a class — a discrete value).
There are three key elements of this approach: a set of labeled objects, e.g., a set of stored records, a distance between objects, and the value of k, the number of nearest neighbors.

Making Predections

To classify an unlabeled object, the distance of this object to the labeled objects is computed, its k-nearest neighbors are identified, and the class label of the majority of nearest neighbors is then used to determine the class label of the object. For real-valued input variables, the most popular distance measure is Euclidean distance.





Euclidean Distance between P_1 and $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

The Distance

Euclidean distance is calculated as the square root of the sum of the squared differences between a new point and an existing point across all input attributes .

Other popular distance measures include:

- Hamming Distance
- Manhattan Distance
- Minkowski Distance

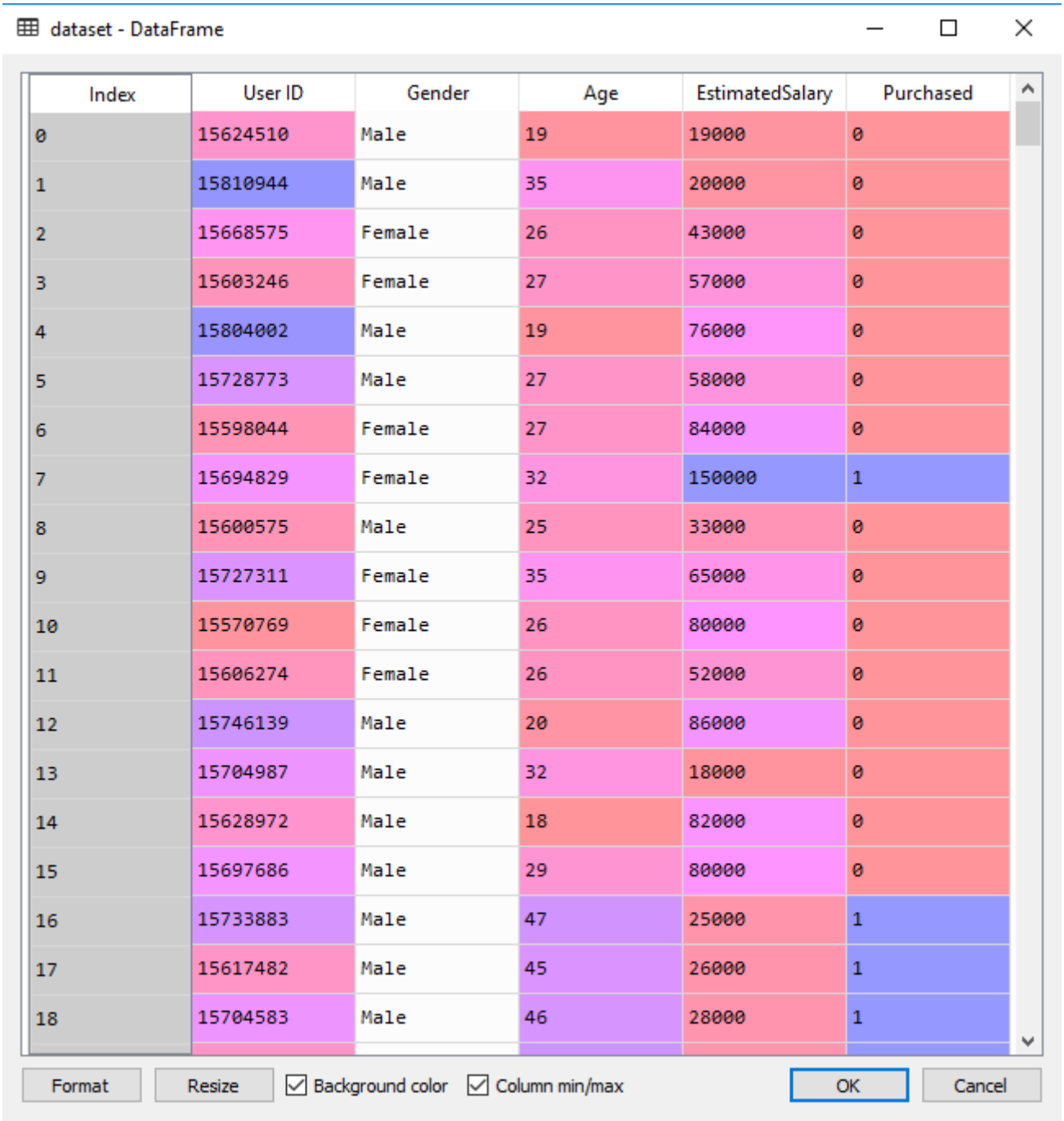
Value of k

Finding the value of k is not easy. A small value of k means that noise will have a higher influence on the result and a large value make it computationally expensive. It depend a lot on your individual cases, sometimes it is best to run through each possible value for k and decide for yourself.

Check out the Repository at: github.com/Avik-Jain/100-Days-Of-ML-Code

Follow Me For More Updates

The DataSet | Social Network



Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0
15	15697686	Male	29	80000	0
16	15733883	Male	47	25000	1
17	15617482	Male	45	26000	1
18	15704583	Male	46	28000	1

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

Splitting the dataset into the Training set and Test set

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state = 0)
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Fitting K-NN to the Training set

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p =
2)
classifier.fit(X_train, y_train)
```

Predicting the Test set results

```
y_pred = classifier.predict(X_test)
```

Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```