**K-Nearest Neighbors (KNN)**

**KN Classifier parameters**: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

**KN Regressor parameters**:

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html#sklearn.neighbors.KNeighborsRegressor

**KNN for clustering (unlabeled data):** <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

As you can see from the chart above, k-Nearest Neighbors belongs to the **supervised** branch of Machine Learning algorithms, which means that it requires labeled data for training.

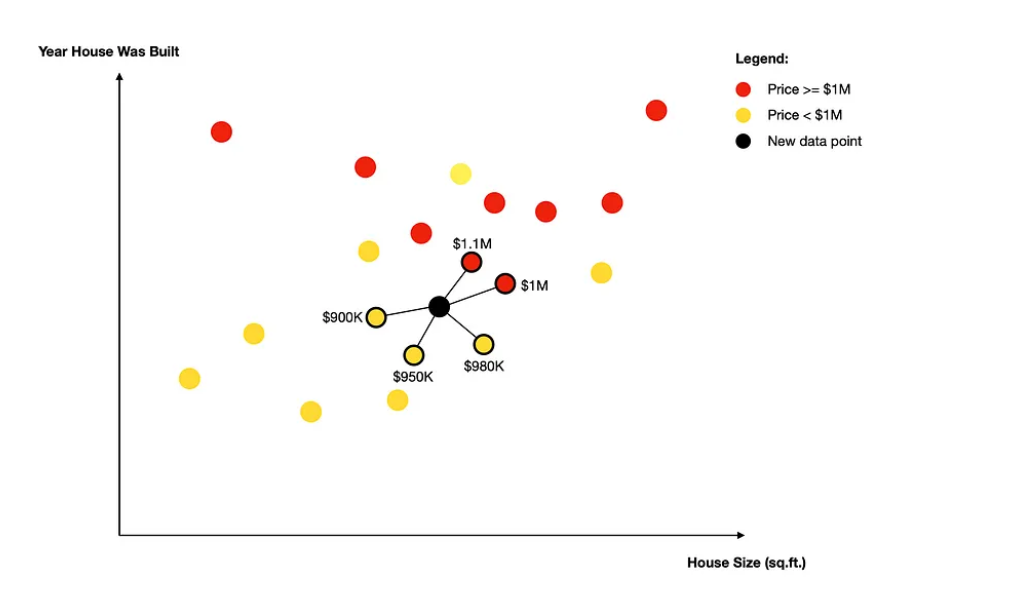
However, suppose you only want to find similar data points (i.e., find neighbors) instead of making predictions. In that case, it is possible to use kNN in an unsupervised manner (see sklearn’s [NearestNeighbors](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html" \l "sklearn.neighbors.NearestNeighbors" \t "_blank) implementation of such unsupervised learner).

It is worth noting that kNN is a very flexible algorithm and can be used to solve different types of problems. Hence, in this article, I will take you through its use for **classification** and **regression**.

# ****How does kNN work?****

Let’s start by looking at “k” in the kNN. Since the algorithm makes its predictions based on the nearest neighbors, we need to tell the algorithm the exact number of neighbors we want to consider. Hence, “k” represents the number of neighbors and is simply a hyperparameter that we can tune.

Now let’s assume that we have picked k=5, and we have a dataset containing house size (sq.ft.), the year house was built, and house price. We want to train kNN on this data and then use it to predict the price of another house that we are interested in.



In the graph above, the black circle represents a new data point (the house we are interested in). Since we have set k=5, the algorithm finds five nearest neighbors of this new point.

*Note, typically, Euclidean distance is used, but some implementations allow alternative distance measures (e.g., Manhattan).*

Once the neighbors are found, one of the two things will happen depending on whether you are performing classification or regression analysis.

* **Classification:** the algorithm uses simple majority voting to assign the label to the new data point. In our example, the majority consists of 3 neighbors with a price<$1M. Hence, the predicted label for the new data point is <$1M.
* **Regression:**the algorithm calculates the average value of all the neighbors. In our example this would be: ($900K + $950K + $980K + $1M + $1.1M) / 5 = $986K. So, the predicted price of a house (new data point) is $986K.

As you can see from this example, kNN is a very intuitive algorithm, making it easy to explain how the predictions were made. Thus, it is in contrast to other classification and regression algorithms such as RandomForest or XGBoost.

One final thing to add, the explanation above showed what happens when uniform weights are being used. I.e., each neighbor carries the same weight in the calculation. However, in some cases (e.g., when you have sparse data), it may be beneficial to use distance-based weights.

Sparse data refers to datasets with many features with zero values. It can cause problems in different fields, especially in machine learning.

Distance weighting assigns weights proportional to the inverse of the distance from the query point, which means that neighbors closer to your data point will carry proportionately more weight than neighbors that are further away.