Name: Puja Nitin Redij

UTA ID : 1001651089

DM : Project2

K Nearest Neighbors Classifier

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression.[1] In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

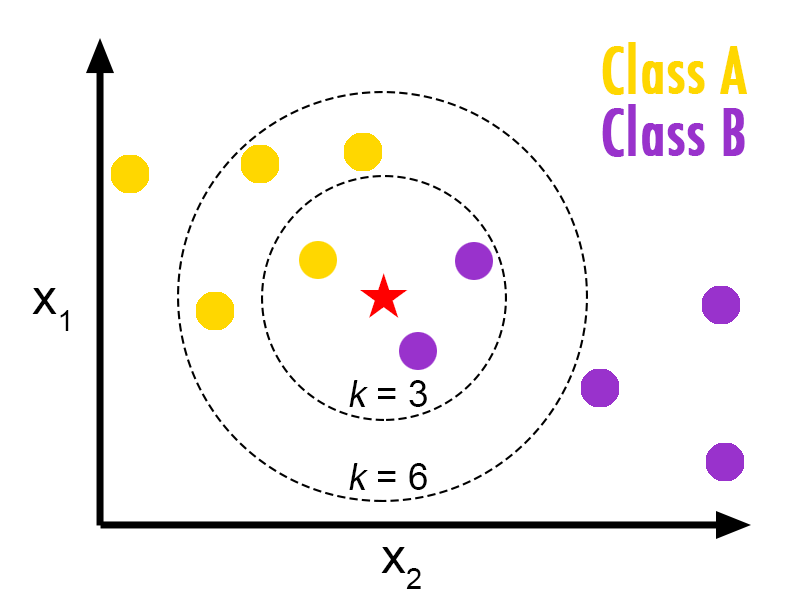
In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification.

Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor.[2]

The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A peculiarity of the k-NN algorithm is that it is sensitive to the local structure of the data.An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive [integer](https://en.wikipedia.org/wiki/Integer), typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.



class of star would be B if k =3 and A if k = 6

Algorithm:

Example of k-NN classification. The test sample (green dot) should be classified either to blue squares or to red triangles. If k = 3 (solid line circle) it is assigned to the red triangles because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the blue squares (3 squares vs. 2 triangles inside the outer circle).

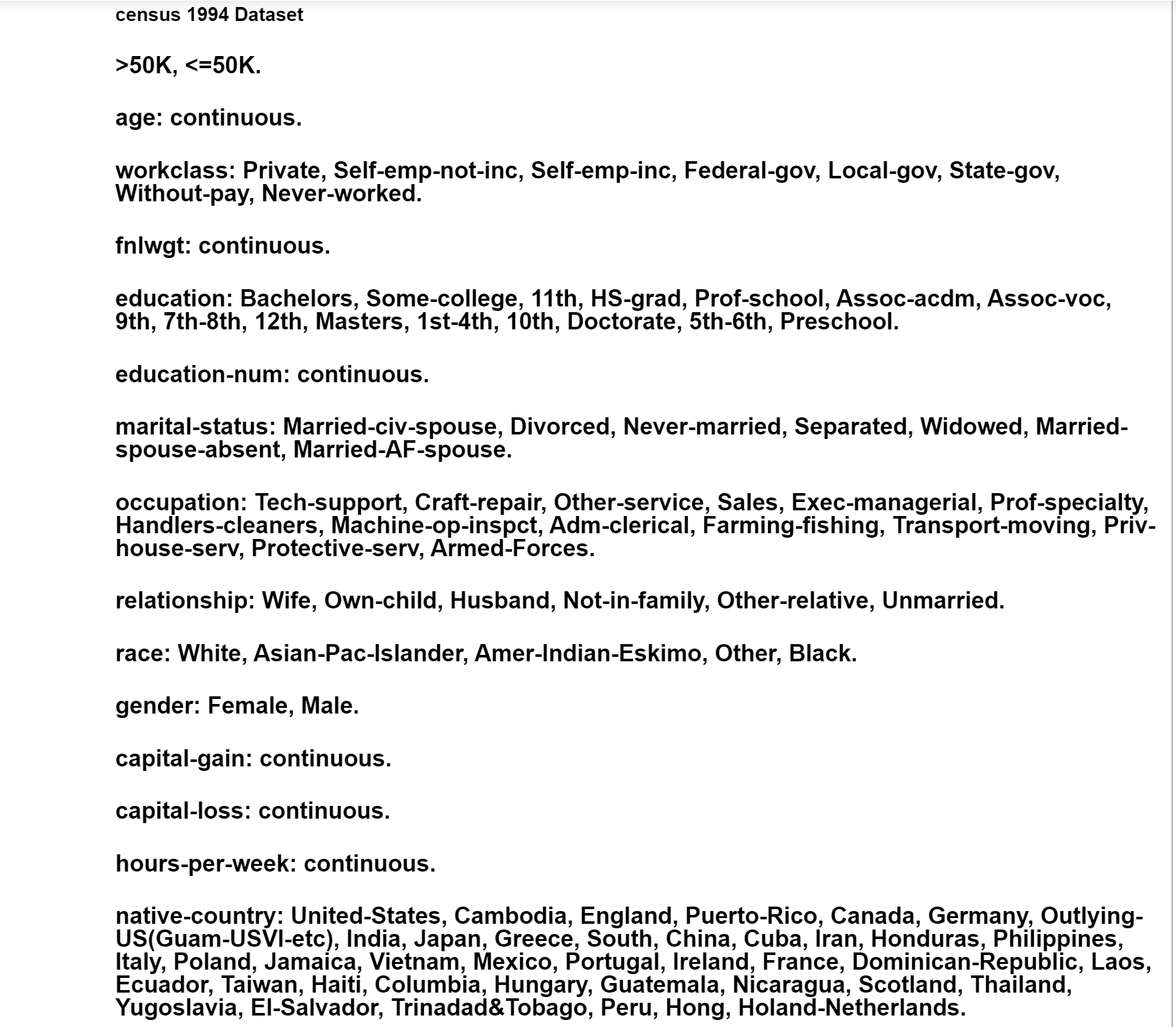
The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance). In the context of gene expression microarray data, for example, k-NN has been employed with correlation coefficients, such as Pearson and Spearman, as a metric.[3] Often, the classification accuracy of k-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighbourhood components analysis.

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest neighbors due to their large number.[4] One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its k nearest neighbors. The class (or value, in regression problems) of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. For example, in a self-organizing map (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. K-NN can then be applied to the SOM.

Dataset Details



Univariate Selection

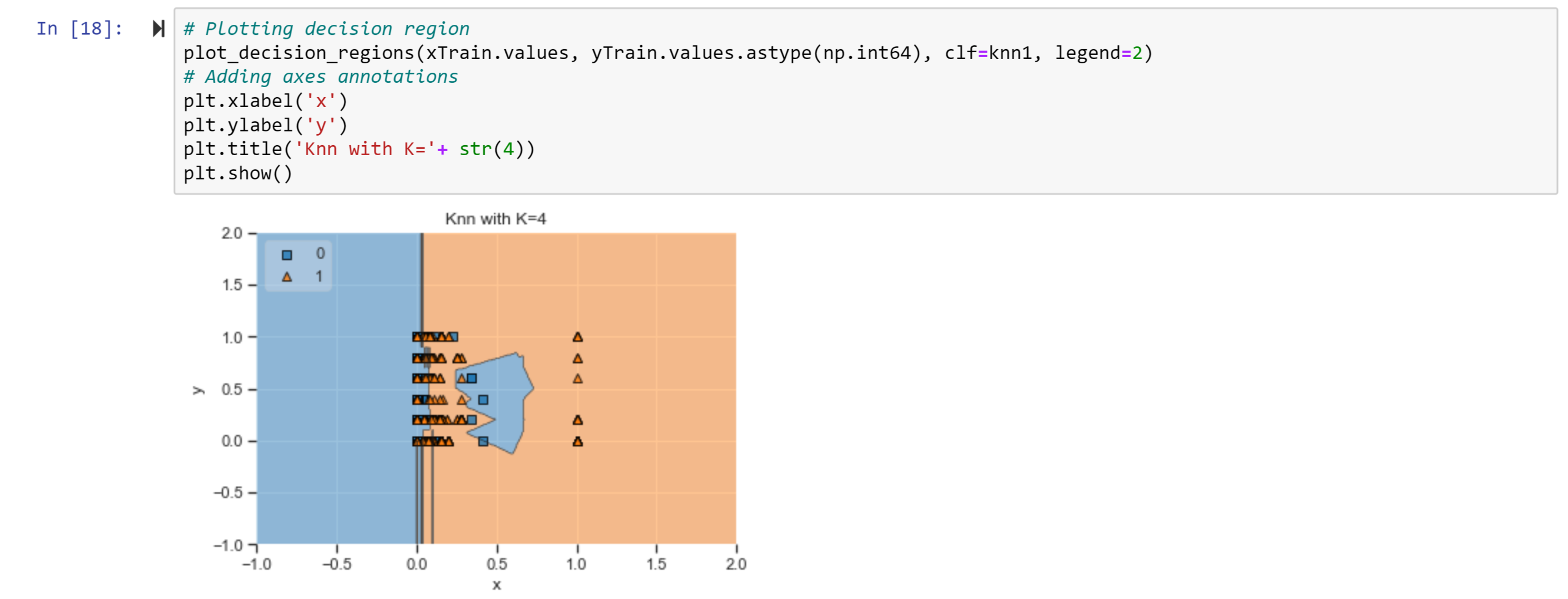
Statistical tests can be used to select those features that have the strongest relationship with the output variable.

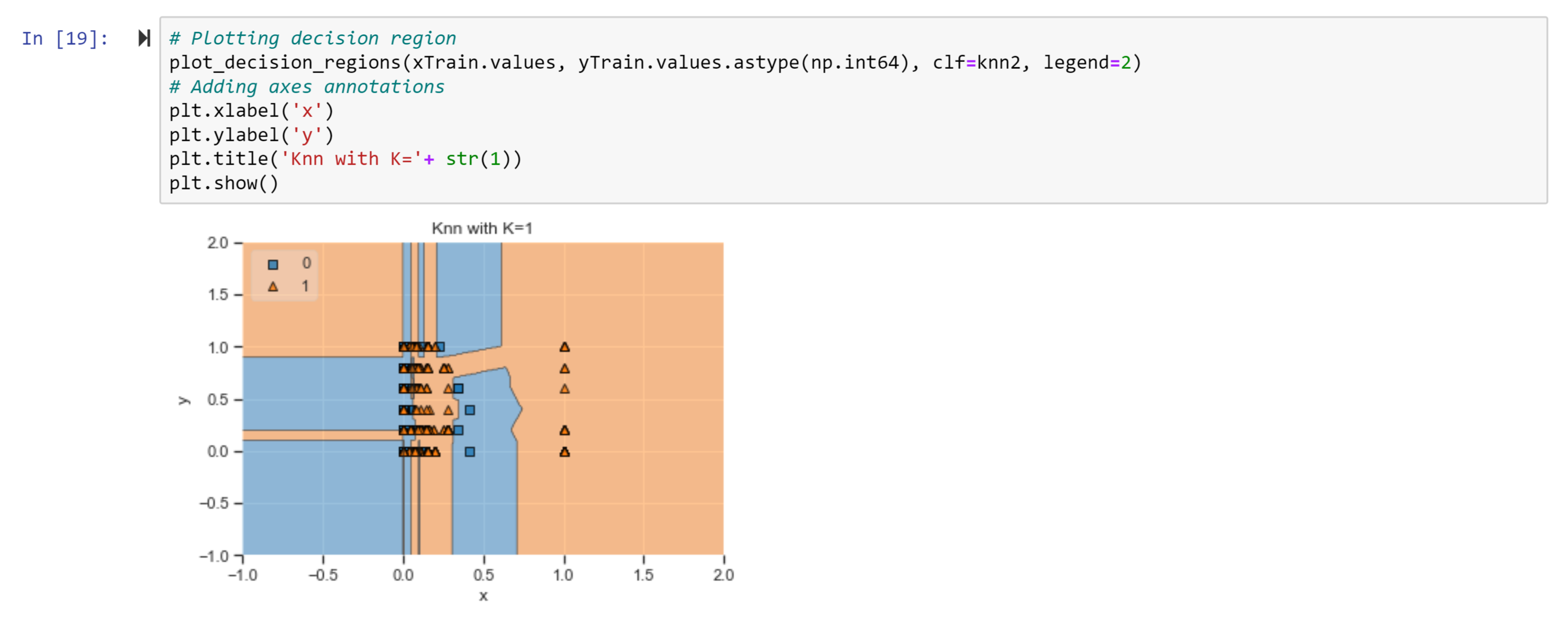
The scikit-learn library provides the [SelectKBest](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html" \l "sklearn.feature_selection.SelectKBest" \t "_blank) class that can be used with a suite of different statistical tests to select a specific number of features.

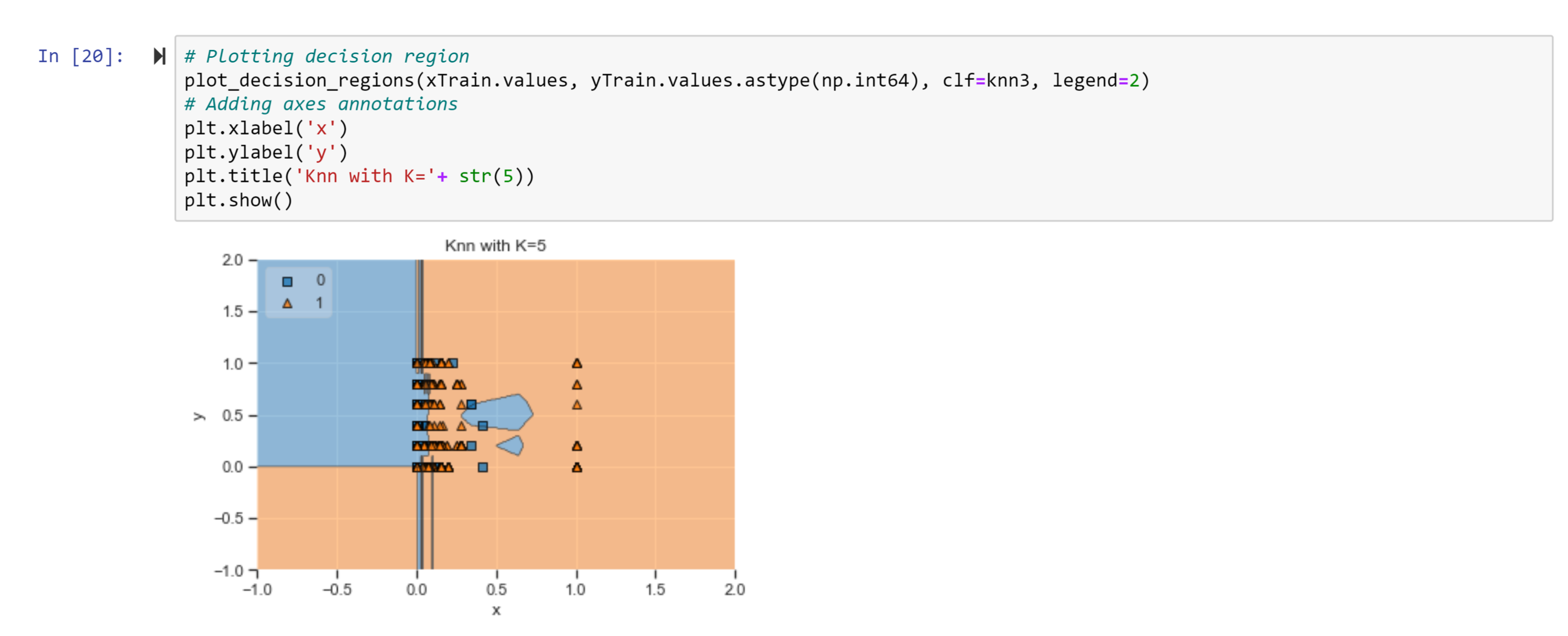
The example below uses the chi-squared (chi²) statistical test for non-negative features to select 10 of the best features from the Mobile Price Range Prediction Dataset.



Visualize the classifier in a 2D projection, for all three different number of neighbors.







Decision Tree for GINI and Entropy

The Decision tree method involves the upside-down splitting in which the root/internal node is the basis for the splitting of the entire tree into branches/edges. The end node after which no further splitting occurs is called the leaf/decision node.

**ADVANTAGES:**

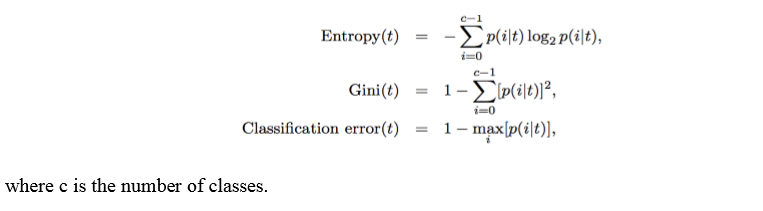
• Decision trees are simple and easy to understand.

• Decision trees are less influenced by outliers and missing values as compared to the other models.

**DISADVANTAGES:**

• Decision trees face overfitting, and this is solved using pruning.

• Decision trees loose information while working with Continuous variables.



Comparing the results of Gini and Entropy:

* The second-best split attributes for X8<=0.5 vary for Gini and entropy
* The third-best split attributes for X5<=12.5 and X11<=7073.5 also vary for Gini and entropy and it follows for the entire tree.
* The accuracy of both Gini and Entropy is 0.8442 but the Recall for Gini in Class 1 is 0.51 whereas Entropy has 0.50
* Gini is used for continuous attributes and entropy is used for attributes that occur in classes.
* Gini will find the largest class, and entropy tends to find groups of classes that make up ~50% of the data. Gini minimizes the misclassification.
* Entropy may be a little slower to compute.

Sources :

* <https://www.kaggle.com/jepsds/feature-selection-using-selectkbest?utm_campaign=News&utm_medium=Community&utm_source=DataCamp.com>
* <https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e>
* <https://medium.com/machine-learning-101/k-nearest-neighbors-classifier-1c1ff404d265>
* <https://www.edureka.co/blog/k-nearest-neighbors-algorithm/>
* <https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/>
* <https://stackoverflow.com/questions/42088336/how-to-get-the-most-contributing-feature-in-any-classifier-sklearn-for-example-d>