# A Brief Overview of K-Nearest Neighbour (KNN)

K-NN is a non-parametric supervised learning technique in which we attempt to classify data points using a set of instructions. In basic terms, it records all training and classifies fresh cases on the basis of their similarities.

Predictions for a new instance (x) are made by a search across the complete training set for the K examples that are most similar to one another (neighbours), and calculating a summary of the output variable for those K cases. In terms of categorisation, this is the default (or most often used) class value.

Let us begin by setting certain definitions and notations. We'll use x to represent a feature (also known as a predictor or attribute) and y to represent the target (also known as a label or class) that we're attempting to forecast.

KNN belongs to the supervised learning algorithm family. Informally, we are given a labelled dataset of training observations (x, y) and want to capture the relationship between x and y. In more technical terms, our goal is to learn a function h: X → Y such that, given an unknown observation x, h(x) can definitely predict the matching output y.

The KNN classifier is a non-parametric and instance-based learning algorithm as well.

• Non-parametric implies that it makes no explicit assumptions regarding the functional form of h, eliminating the risks of mismodeling the underlying distribution of the data. Assume our data is highly non-Gaussian, yet the learning model we select expects a Gaussian form. Our algorithm would produce exceedingly bad predictions in that instance.

• Because we use instance-based learning, our system does not explicitly learn a model. Instead, it memorises the training instances, which are then employed as "knowledge" in the prediction phase.

This means that the algorithm will only use the training examples to spit out a response when we query our database (i.e. when we ask it to predict a label given an input).

“KNN is a non-parametric, instance-based learning algorithm that is used in supervised learning.”

It is worth noting that the limited training phase of KNN incurs both a memory cost, as we must store a potentially massive data set, and a computing cost during test time, as classifying a given observation necessitates a run down of the entire data set. In practise, this is undesirable because we normally expect quick responses.

“Minimal training but expensive testing.”

# How the KNN algorithm operates

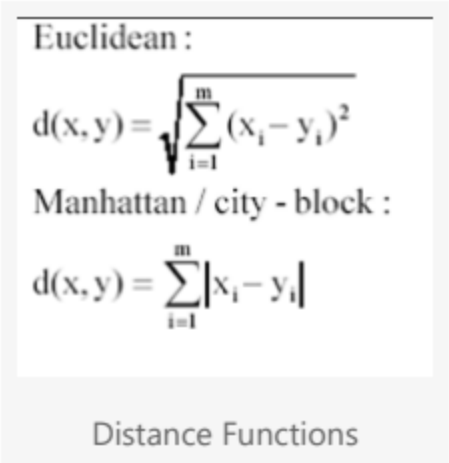
Suppose we have the height, weight and T-shirt size of some customers and we need to predict the T-shirt size of a new customer given only the height and weight information we have. Data including height, weight and T-shirt size information is shown below

| Height (in cms) | Weight (in kgs) | T Shirt Size |
|---|---|---|
| 158 | 58 | M |
| 158 | 59 | M |
| 158 | 63 | M |
| 160 | 59 | M |
| 160 | 60 | M |
| 163 | 60 | M |
| 163 | 61 | M |
| 160 | 64 | L |
| 163 | 64 | L |
| 165 | 61 | L |
| 165 | 62 | L |
| 165 | 65 | L |

| | | |
|---|---|---|
| 168 | 62 | L |
| 168 | 63 | L |
| 168 | 66 | L |
| 170 | 63 | L |
| 170 | 64 | L |
| 170 | 68 | L |

Step 1: Calculate Similarity based on distance function

There are other distance functions, however the Euclidean distance function is the most frequently employed.




Distance Functions

The purpose of the distance measure is to determine the distance (similarity) between the new sample and training cases and then to determine the k-closest customers in terms of height and weight to the new customer.

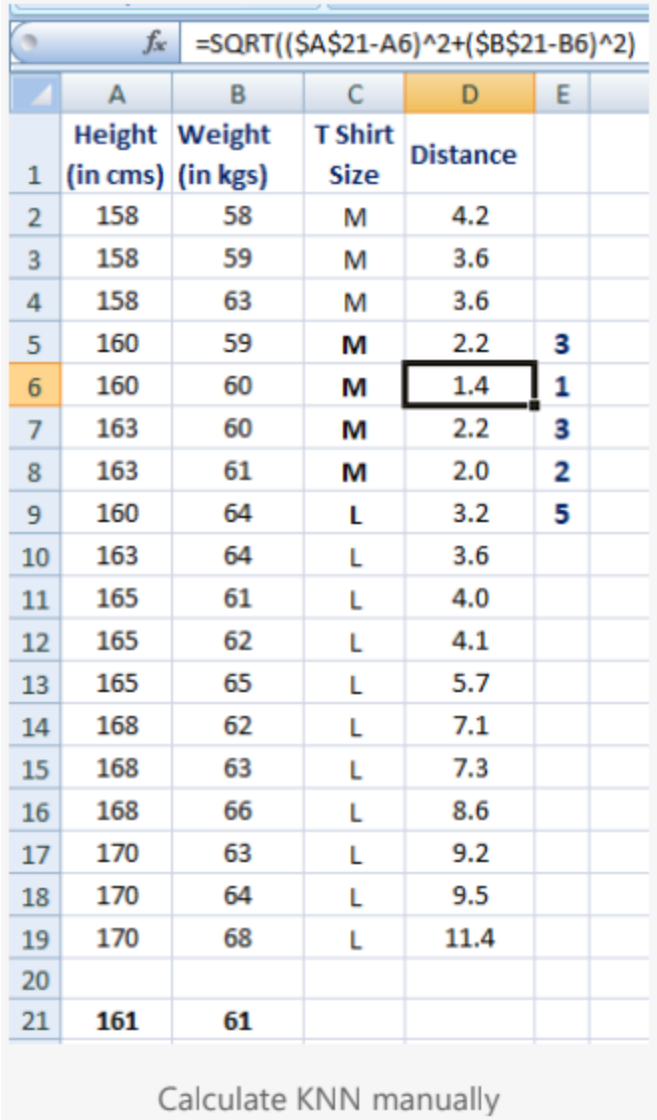
New customer named 'Monica' has a height of 161cm and weighs 61kg.Euclidean distance between first observation and new observation.(Monica) is as follows -

=SQRT((161-158)^2+(61-58)^2)

Similarly, we will calculate the distance between all of the training examples and the new case and rank them according to the distance. The distance value with the least value will be rated first and regarded as the nearest neighbour.

Step 2: Find K-Nearest Neighbours

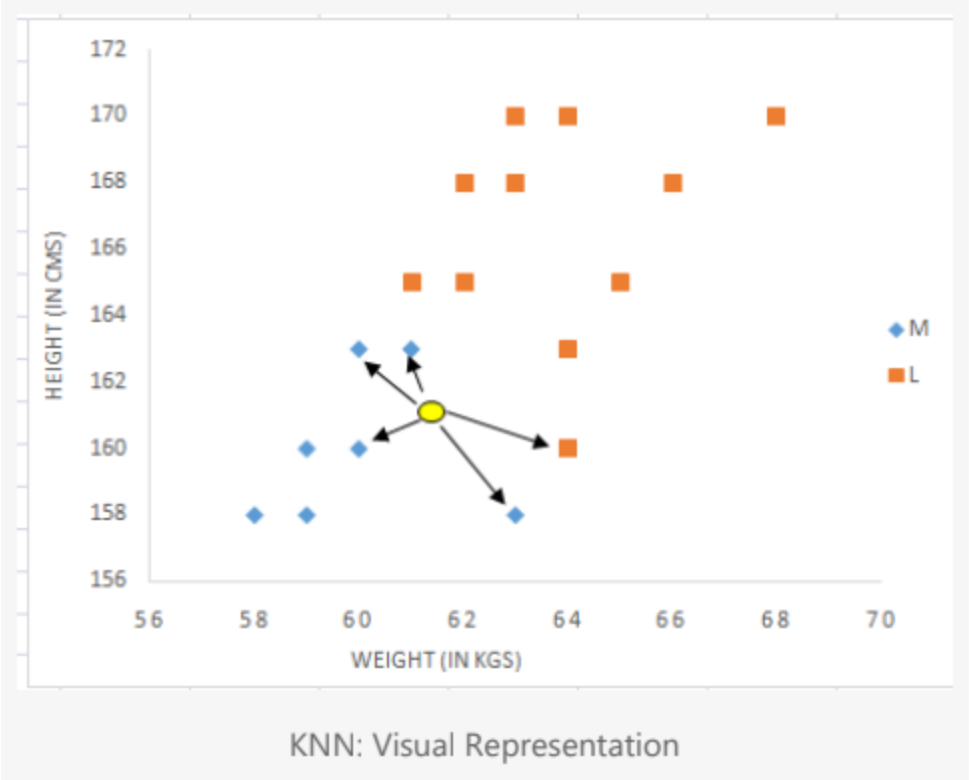
Let k be 5. Then the algorithm searches for the 5 customers closest to Monica, i.e. most similar to Monica in terms of attributes, and see what categories those 5 customers were in. If 4 of them had ‘Medium T shirt sizes’ and 1 had ‘Large T shirt size’ then your best guess for Monica is ‘Medium T shirt. See the calculation shown in the snapshot below -




| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Height (in cms) | Weight (in kgs) | T Shirt Size | Distance | |
| 2 | 158 | 58 | M | 4.2 | |
| 3 | 158 | 59 | M | 3.6 | |
| 4 | 158 | 63 | M | 3.6 | |
| 5 | 160 | 59 | M | 2.2 | 3 |
| 6 | 160 | 60 | M | 1.4 | 1 |
| 7 | 163 | 60 | M | 2.2 | 3 |
| 8 | 163 | 61 | M | 2.0 | 2 |
| 9 | 160 | 64 | L | 3.2 | 5 |
| 10 | 163 | 64 | L | 3.6 | |
| 11 | 165 | 61 | L | 4.0 | |
| 12 | 165 | 62 | L | 4.1 | |
| 13 | 165 | 65 | L | 5.7 | |
| 14 | 168 | 62 | L | 7.1 | |
| 15 | 168 | 63 | L | 7.3 | |
| 16 | 168 | 66 | L | 8.6 | |
| 17 | 170 | 63 | L | 9.2 | |
| 18 | 170 | 64 | L | 9.5 | |
| 19 | 170 | 68 | L | 11.4 | |
| 20 | | | | | |
| 21 | 161 | 61 | | | |

In the graph below, the binary dependent variable (T-shirt size) is displayed in blue and orange color. 'Medium T-shirt size' is in blue color and 'Large T-shirt size' in orange color. New customer

information is exhibited in a yellow circle. Four blue highlighted data points and one orange highlighted data point are close to the yellow circle. so the prediction for the new case is a blue highlighted data point which is Medium T-shirt size.




KNN: Visual Representation

# Assumptions of KNN

## 1. Standardization

When independent variables in training data are measured in different units, it is important to standardize variables before calculating distance. For example, if one variable is based on height in cms, and the other is based on weight in kgs then height will influence more on the distance calculation. In order to make them comparable we need to standardize them which can be done by any of the following methods:

$$Xs = \frac{X - mean}{s.d.}$$

$$Xs = \frac{X - mean}{max - min}$$

$$Xs = \frac{X - min}{\max - min}$$

After standardization, the 5th closest value got changed as height was dominating earlier before standardization. Hence, it is important to standardize predictors before running the K-nearest neighbour algorithm.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Height (in cms) | Weight (in kgs) | T Shirt Size | Distance | |
| 2 | -1.39 | -1.64 | M | 1.3 | |
| 3 | -1.39 | -1.27 | M | 1.0 | |
| 4 | -1.39 | 0.25 | M | 1.0 | |
| 5 | -0.92 | -1.27 | **M** | 0.8 | **4** |
| 6 | -0.92 | -0.89 | **M** | 0.4 | **1** |
| 7 | -0.23 | -0.89 | **M** | 0.6 | **3** |
| 8 | -0.23 | -0.51 | **M** | 0.5 | **2** |
| 9 | -0.92 | 0.63 | L | 1.2 | |
| 10 | -0.23 | 0.63 | L | 1.2 | |
| 11 | 0.23 | -0.51 | **L** | 0.9 | **5** |
| 12 | 0.23 | -0.13 | L | 1.0 | |
| 13 | 0.23 | 1.01 | L | 1.8 | |
| 14 | 0.92 | -0.13 | L | 1.7 | |
| 15 | 0.92 | 0.25 | L | 1.8 | |
| 16 | 0.92 | 1.39 | L | 2.5 | |
| 17 | 1.39 | 0.25 | L | 2.2 | |
| 18 | 1.39 | 0.63 | L | 2.4 | |
| 19 | 1.39 | 2.15 | L | 3.4 | |
| 20 | | | | | |
| 21 | **-0.7** | **-0.5** | | | |

### 2. Outlier

Low k-value is sensitive to outliers and a higher K-value is more resilient to outliers as it considers more voters to decide prediction.

## Why is KNN a non-parametric model?

Non-parametric means not making any assumptions on the underlying data distribution. Non-parametric methods do not have fixed numbers of parameters in the model. Similarly in KNN, model parameters actually grow with the training data set - you can imagine each training case as a "parameter" in the model.

## K-means vs. KNN

Many people get confused between these two statistical techniques- K-means and K-nearest neighbour. See some of the difference below -

1. K-means is an unsupervised learning technique (no dependent variable) whereas KNN is a supervised learning algorithm (dependent variable exists)
2. K-means is a clustering technique which tries to split data points into K-clusters such that the points in each cluster tend to be near each other whereas K-nearest neighbour tries to determine the classification of a point, combines the classification of the K nearest points

# KNN's Advantages and Disadvantages

## Advantages

1. Simple to comprehend

2. No inferences regarding data

3. Can be used for classification as well as regression

4. Easily solves problems involving multiple classes

## Disadvantages

1. Memory Consuming / Expensive in terms of computation

2. Sensitive to the data's scale

3. Does not perform well when the target variable is a rare occurrence (skewed).

4. Difficulty when there are a large number of independent factors

A modest value of k results in a considerable variance in any given task predictions. Alternatively, increasing k may result in a big bias in the model.

# How does KNN handle categorical variables?

Create dummy variables out of a categorical variable and include them instead of the original categorical variable. Unlike regression,create k dummies instead of (k-1). For example, a

categorical variable named "Department" has 5 unique levels / categories. So we will create 5 dummy variables. Each dummy variable has 1 against its department and else 0.

## How to determine the optimal K value?

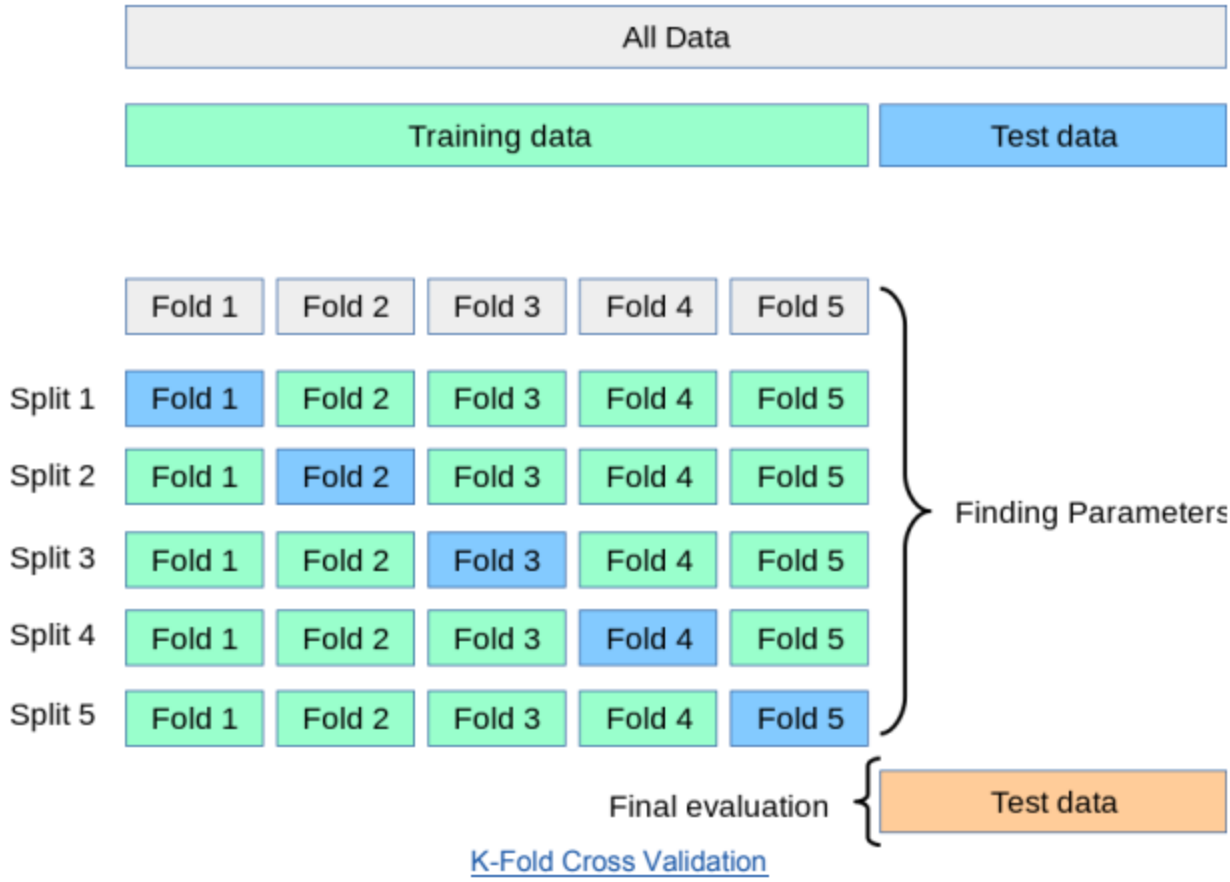
Cross-validation is a smart way to find out the optimal K value. It estimates the validation error rate by holding out a subset of the training set from the model building process.Cross-validation (let's say 10-fold validation) involves randomly dividing the training set into 10 groups, or folds, of approximately equal size. 90% data is used to train the model and remaining 10% to validate it. The misclassification rate is then computed on the 10% validation data. This procedure repeats 10 times. Different groups of observations are treated as a validation set each of the 10 times. It results to 10 estimates of the validation error which are then averaged out.

## Parameter Tuning with Cross Validation

In this section, we'll explore a method that can be used to tune the hyperparameter K. Obviously, the best K is the one that corresponds to the lowest test error rate, so let's suppose we carry out repeated measurements of the test error for different values of K. Inadvertently, what we are doing is using the test set as a training set! This means that we are underestimating the true error rate since our model has been forced to fit the test set in the best possible manner.

Our model is then incapable of generalizing to newer observations,a process known as overfitting. Hence, touching the test set is out of the question and must only be done at the very end of our pipeline.

Using the test set for hyperparameter tuning can lead to overfitting. An alternative and smarter approach involves estimating the test error rate by holding out a subset of the training set from the fitting process. This subset, called the validation set, can be used to select the appropriate level of flexibility of our algorithm! There are different validation approaches that are used in practice, and we will be exploring one of the more popular ones called k-fold cross validation.




K-Fold Cross Validation

As seen in the image, k-fold cross validation (the k is totally unrelated to K) involves randomly dividing the training set into k groups, or folds, of approximately equal size.The first fold is treated as a validation set, and the method is fit on the remaining k−1k−1 folds. The misclassification rate is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error which are then averaged out.

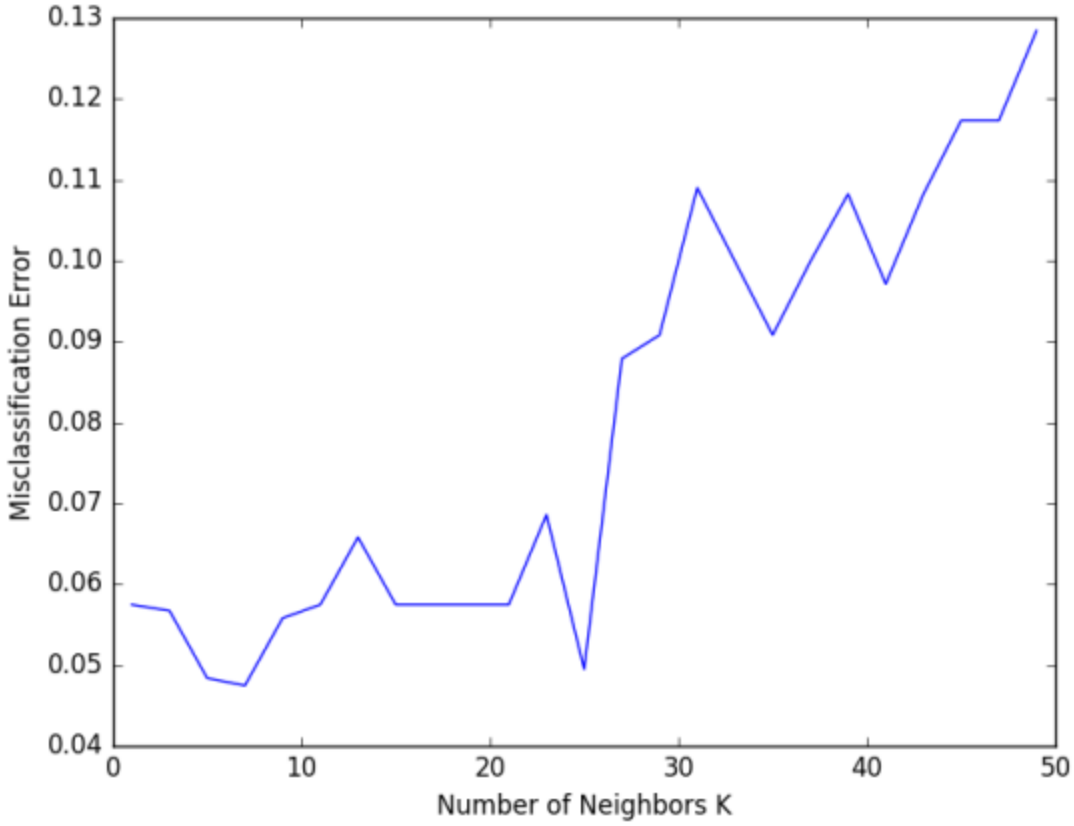
Cross-validation can be used to estimate the test error associated with a learning method in order to evaluate its performance, or to select the appropriate level of flexibility.
If that is a bit overwhelming for you, don't worry about it. We're gonna make it clearer by performing a 10-fold cross validation on our dataset using a generated list of odd K's ranging from 1 to 50.

```
# creating odd list of K for KNN
neighbors = list(range(1, 50, 2))
# empty list that will hold cv scores
cv_scores = []
```

```
# perform 10-fold cross validation
for k in neighbors:
knn = KNeighborsClassifier(n_neighbors=k)
scores = cross_val_score(knn, X_train, y_train, cv=10,
scoring='accuracy')
cv_scores.append(scores.mean())
```

Again, scikit-learn comes in handy with its cross_val_score method. We specify that we are performing 10 folds with the cv = 10 parameter and that our scoring metric should be accuracy since we are in a classification setting.Finally, we plot the misclassification error versus K.

```
# changing to misclassification error
mse = [1 - x for x in cv_scores]
# determining best k
optimal_k = neighbors[mse.index(min(mse))]
print("The optimal number of neighbors is {}".format(optimal_k))
# plot misclassification error vs k
plt.plot(neighbors, mse)
plt.xlabel("Number of Neighbors K")
plt.ylabel("Misclassification Error")
plt.show()
```




Cross validation tenfold indicates that K=7 produces the lowest validation error.

## Why is KNN referred to as the Lazy Learner?

It is often called “lazy learning”. It does not build any “model”. Instead, all the work (i.e., finding the nearest neighbours) is done at prediction time.

## Snippet of KNN Code:

Users > satyam > Desktop > ML_Algos > KNNclassification.py

```python
# KNN Algorithm for Classification | Author: Dhrub Satyam
# Creating our own features
# Assigning features and label variables
# First Feature
weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny',
'Rainy','Sunny','Overcast','Overcast','Rainy']
# Second Feature
temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild']

# Label or target varible
play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']

# Import LabelEncoder
from sklearn import preprocessing
#creating labelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers.
weather_encoded=le.fit_transform(weather)
print(weather_encoded)

# converting string labels into numbers
temp_encoded=le.fit_transform(temp)
label=le.fit_transform(play)

#combinig weather and temp into single listof tuples
features=list(zip(weather_encoded,temp_encoded))

from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)

# Train the model using the training sets
model.fit(features,label)

#Predict Output
predicted= model.predict([[0,2]]) # 0:Overcast, 2:Mild
print(predicted)
```