Raspberry Pi

<https://de.rs-online.com/web/generalDisplay.html?id=raspberrypi>

<https://computer.howstuffworks.com/raspberry-pi2.htm>

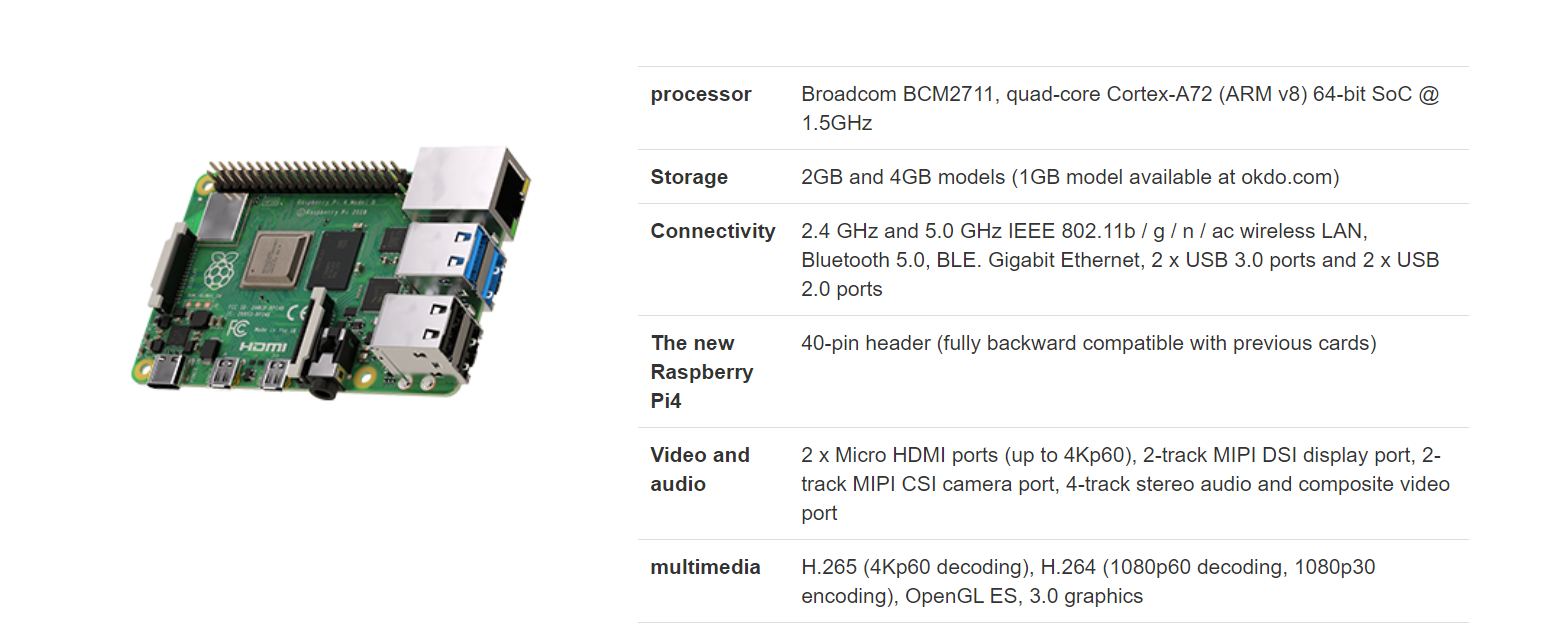
<https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/>

<https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/>

The Raspberry Pi is a low cost computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. We can use it to design own software and we can use program in languages like Scratch and Python. It’s capable of doing everything you’d expect a desktop computer to do, from browsing the internet and coding for software.

The reason to use this is to able to connect it with Red pitaya to show how one device which is connected to the sensor can send data and we can use raspberry as a receiver for that information and code to make it intelligent to recognize different objects precisely.

Like a normal computer raspberry pi contains processor, storage,connectivity, video audio support, multimedia



Objective:

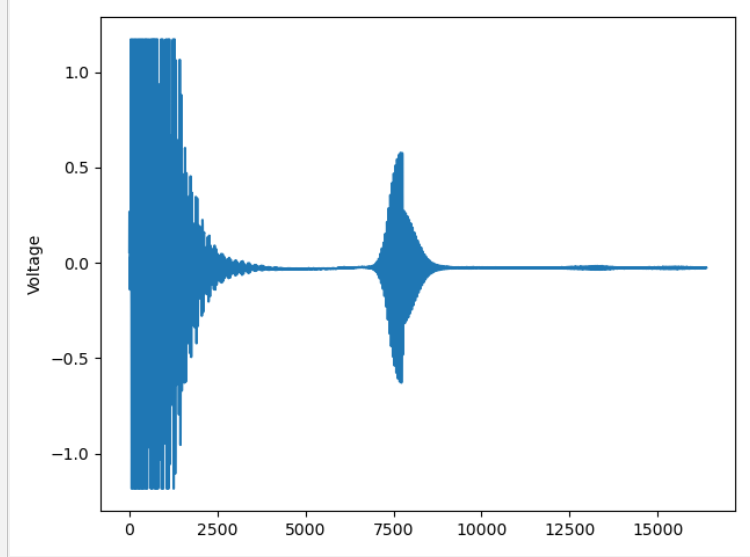
Automotive Sensor for Object Recognition using RedPitaya and Raspberry Pi

Objective is to design software for receiving, displaying and saving the acquired ultrasonic signals from Red Pitaya with Raspberry Pi. The objective is to make the system stable. First is to acquire the data from the ultrasonic sensor with the range of 1 meter. taking the data with least noise ratio by capturing the time when object data is available using external trigger condition.

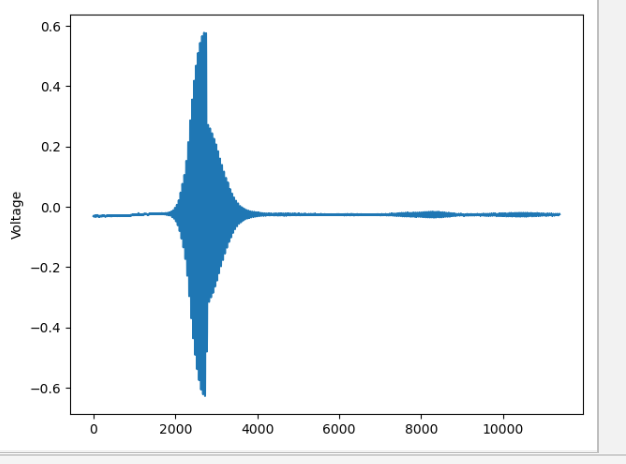
When we have desired object before working on feature part we have cut down the input part to make it more stable and accurate data for each object.

then after analyzing the data, develop feature extraction and classification. We have chosen Bayes Classifier for Machine learning part and trained our model. We have used frequency domain for this purpose.

Signal input with object reflection:



After cutting input signal:



In order to do this which give us more precise look to object to extract features.

We have applied Fast Fourier transform fft in order to have frequency domain for captured signal,

After that we have done power spectrum in order to have

A Power Spectral Density (PSD) is the measure of signal's power content versus frequency. A PSD is typically used to characterize broadband random signals. It can be looked upon as a frequency-domain plot of power per unit Hz vs. frequency.

Therefore, while the power spectrum calculates the area under the signal plot using the discrete Fourier Transform, the power spectrum density assigns unit of frequency and thus, enhances periodicities. By doing this it will give us more detailed signal to extract features.

By this data we have taken peaks, shape, mean and variance as features to proceed with chosen Bayes Classification.

Machine Learning:

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.

Naive Bayes comes under the class of generative models for classification. It models the posterior probability from the class conditional densities. So the output is a probability of belonging to a class.

SVM on the other hand is based on a discriminant function given by y = w.x+b. Here the weights w and bias parameter b are estimated from the training data. It tries to find a hyperplane that maximizes the margin and there is optimization function in this regard.

Performance wise SVMs using the radial basis function kernel are more likely to perform better as they can handle non-linearity’s in the data. Naive Bayes performs best when the features are independent of each other which often does not happen in real. Having said that it still performs good even when the features are not independent.

We have chosen naïve Naive Bayes as our features are independent of each other and we have three objects to identify (class) Class 1 Wall, Class 2 Human , Class 3 Car.

By using Naive Bayes, we will train and create these three type of classes and train our model.

How Naïve works:

We can describe this in 5 part:

* Step 1: Separate By Class.
* Step 2: Summarize Dataset.
* Step 3: Summarize Data By Class.
* Step 4: Gaussian Probability Density Function.
* Step 5: Class Probabilities

1:

We will need to calculate the probability of data by the class they belong to, the so-called base rate.This means that we will first need to separate our training data by class. A relatively straightforward operation.We can create a dictionary object where each key is the class value and then add a list of all the records as the value in the dictionary.Below is a function named separate\_by\_class() that implements this approach. It assumes that the last column in each row is the class value.

2:

We need two statistics from a given set of data.

We’ll see how these statistics are used in the calculation of probabilities in a few steps. The two statistics we require from a given dataset are the mean and the standard deviation (average deviation from the mean).The mean is the average value and can be calculated as:

* mean = sum(x)/n \* count(x)

Where *x* is the list of values or a column we are looking.

Below is a small function named *mean()* that calculates the mean of a list of numbers.

We require the mean and standard deviation statistics to be calculated for each input attribute or each column of our data.

We can do that by gathering all of the values for each column into a list and calculating the mean and standard deviation on that list. Once calculated, we can gather the statistics together into a list or tuple of statistics. Then, repeat this operation for each column in the dataset and return a list of tuples of statistics.

3:

We require statistics from our training dataset organized by class.Above, we have developed the separate\_by\_class() function to separate a dataset into rows by class. And we have developed summarize\_dataset() function to calculate summary statistics for each column.We can put all of this together and summarize the columns in the dataset organized by class values.Below is a function named summarize\_by\_class() that implements this operation. The dataset is first split by class, then statistics are calculated on each subset. The results in the form of a list of tuples of statistics are then stored in a dictionary by their class value.

4.

Calculating the probability or likelihood of observing a given real-value like X1 is difficult.One way we can do this is to assume that X1 values are drawn from a distribution, such as a bell curve or Gaussian distribution.A [Gaussian distribution](https://machinelearningmastery.com/continuous-probability-distributions-for-machine-learning/) can be summarized using only two numbers: the mean and the standard deviation. Therefore, with a little math, we can estimate the probability of a given value. This piece of math is called a Gaussian [Probability Distribution Function](https://en.wikipedia.org/wiki/Gaussian_function) (or Gaussian PDF) and can be calculated as:

* f(x) = (1 / sqrt(2 \* PI) \* sigma) \* exp(-((x-mean)^2 / (2 \* sigma^2)))

Where *sigma* is the standard deviation for *x*, *mean* is the mean for *x* and *PI* is the value of pi.

Below is a function that implements this. I tried to split it up to make it more readable.

5.

Now it is time to use the statistics calculated from our training data to calculate probabilities for new data.

Probabilities are calculated separately for each class. This means that we first calculate the probability that a new piece of data belongs to the first class, then calculate probabilities that it belongs to the second class, and so on for all the classes.

The probability that a piece of data belongs to a class is calculated as follows:

* P(class|data) = P(X|class) \* P(class)

You may note that this is different from the Bayes Theorem described above.

The division has been removed to simplify the calculation.

This means that the result is no longer strictly a probability of the data belonging to a class. The value is still maximized, meaning that the calculation for the class that results in the largest value is taken as the prediction. This is a common implementation simplification as we are often more interested in the class prediction rather than the probability.

The input variables are treated separately, giving the technique it’s name “*naive*“. For the above example where we have 2 input variables, the calculation of the probability that a row belongs to the first class 0 can be calculated as:

* P(class=0|X1,X2) = P(X1|class=0) \* P(X2|class=0) \* P(class=0)

Now you can see why we need to separate the data by class value. The Gaussian Probability Density function in the previous step is how we calculate the probability of a real value like X1 and the statistics we prepared are used in this calculation.

We created a function named *calculate\_class\_probabilities()* that ties all of this together.

It takes a set of prepared summaries and a new row as input arguments.

First the total number of training records is calculated from the counts stored in the summary statistics. This is used in the calculation of the probability of a given class or *P(class)* as the ratio of rows with a given class of all rows in the training data.

Next, probabilities are calculated for each input value in the row using the Gaussian probability density function and the statistics for that column and of that class. Probabilities are multiplied together as they accumulated.

This process is repeated for each class in the dataset.

Finally a dictionary of probabilities is returned with one entry for each class.

Code Explanation:

Step 2:

Feature extraction:



Here above code explain how to get feature for each file which we save after data acquisition. Read the data from “Car csv file” then for each data we calculated peaks, mean and variance We have set the label for each object here for Car label is 3.

Similarly, we have done for data file human and wall respectively.

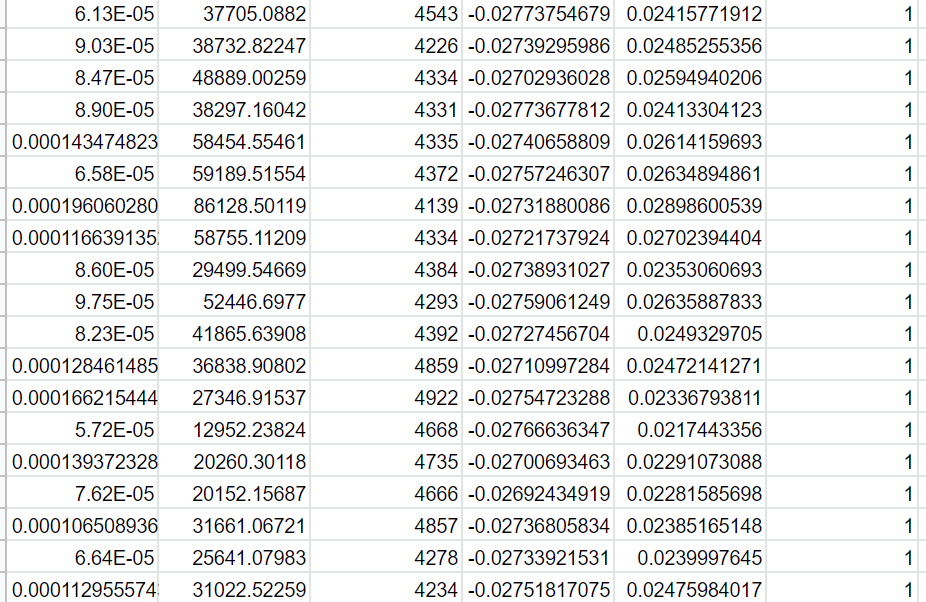




We have save data in csv file, here is separate file which calculate peaks.



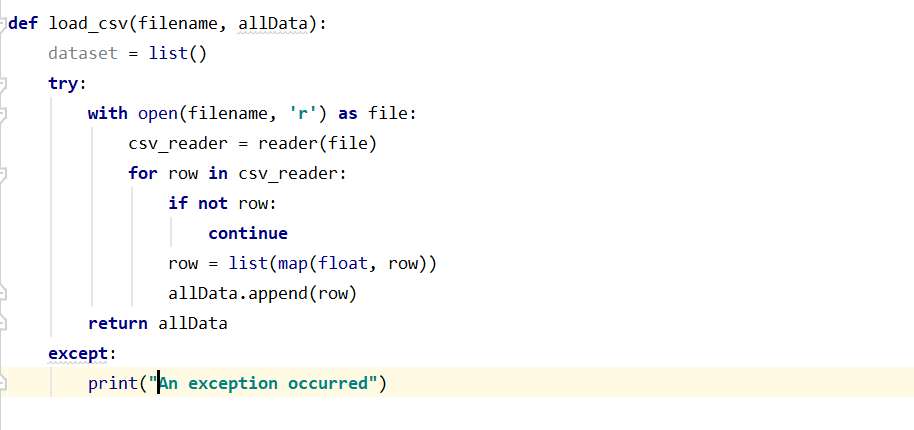
Below are the features extracted file Following are the features, Min peak, max peak,shape,mean,variance and label



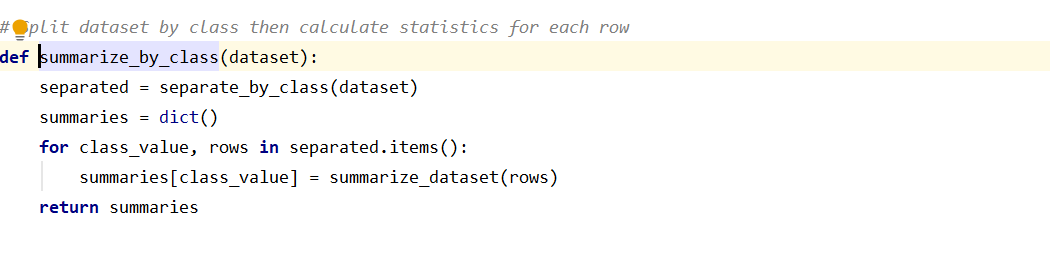
Machine Learning Code explanation:

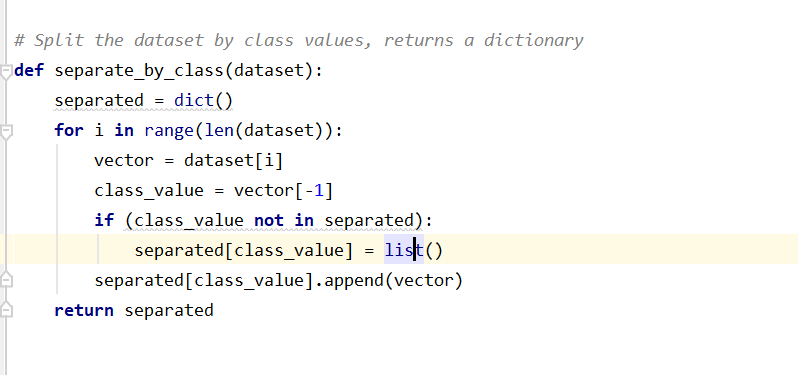
1. Load data:

First we load data from feature file in which we save all features.

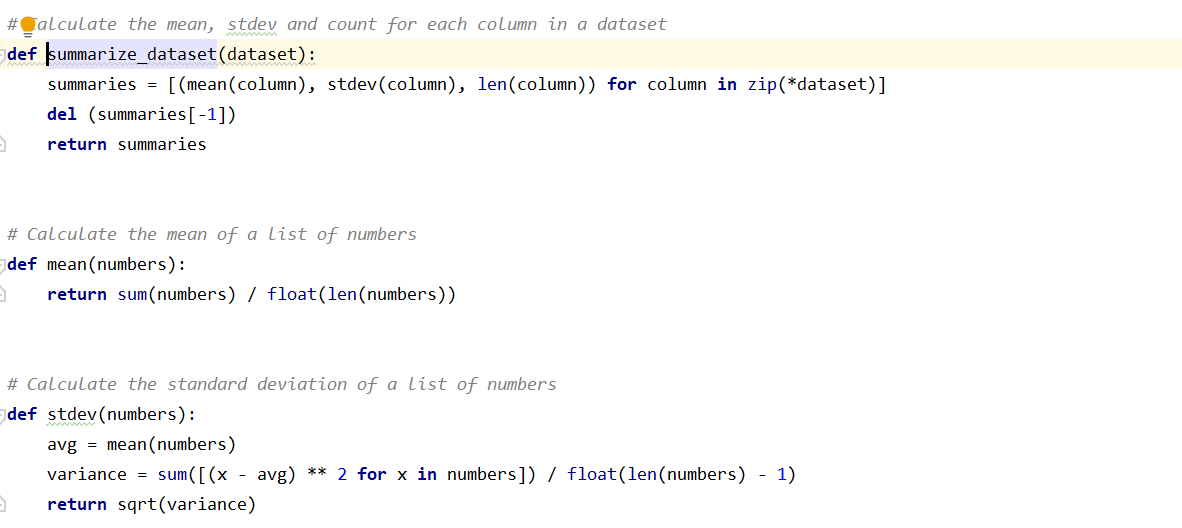


Then we separate data by class and save in dictionary:

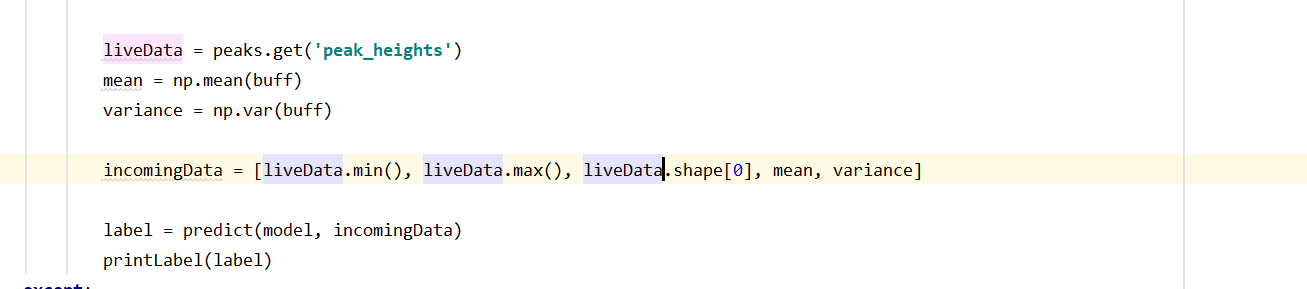




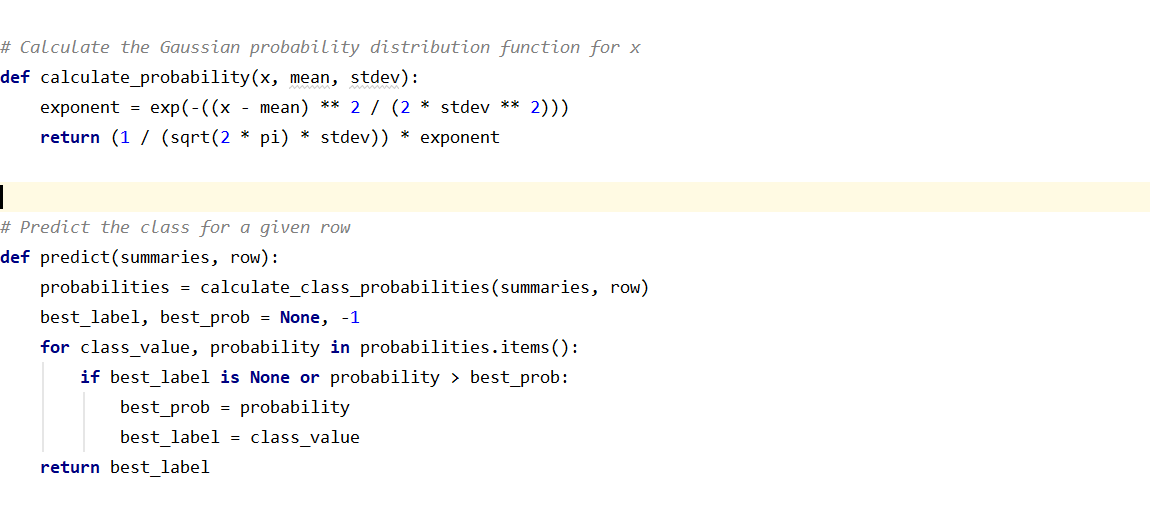
Then for each column we calculated mean, variance and standard deviation:



Till here we have trained out system,Now we will call predict function to predict single data.



In predict function we will calculate Gaussian probability of this input data that how likely it Is belongs to anyone of the trained class.



To show the result We have use two approaches.1 to print label on screen 2 to play audio for object name



Testing: