Automotive Sensor for Object Recognition using RedPitaya and Raspberry Pi

Masters in Information Technology
Automation Lab project task in
cooperation with AIS ultrasonic project
Taimour Yousuf & Anil Kumar Gupta
1276574 & 1266270
yousuf@stud.fra-uas.de &
gupta@stud.fra-uas.de

***Abstract—* The idea of AIS project is to send ultrasonic signals from RedPitaya towards different objects i.e. human, wall and car and receive it back with the help of Raspberry Pi. 4 & python code. After receiving data, display signal and save that acquired signal so that feature extraction can be performed on saved signal values of all three objects. Classification can be done on real time signal through machine learning, which is task of Automation Lab project.**

Keywords—ultrasonic, extraction, classification

I. Overview of Technologies

A. Raspberry Pi 4 Model B

This is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption. For the end user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems. This product's key features include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on). The dual-band wireless LAN and Bluetooth have modular compliance certification, allowing the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market [1].

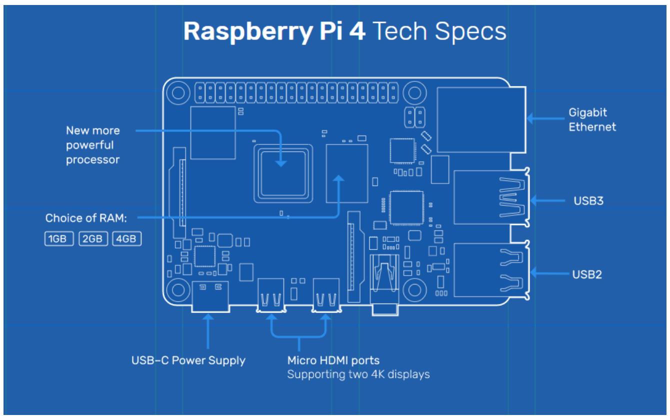


Figure 1: Raspberry Pi 4 Tech Specs

B. STEMlab 125-14 (originally RedPitaya V1.1)

This an open source measurement and control tool in size of a credit card (dimensions: 107 x 60 x 21 mm). It can replace many expensive laboratory measurement instruments. The STEMlab 125-14 unit is a network attached device based on Linux operating system. It includes Radio Frequency signal acquisition and generation technologies, FPGA, Digital Signal Processing and CPU processing. The STEMlab 125-14 unit is equipped with a Micro USB socket used for console connection, a microSD slot for microSD memory card, a RJ45 socket for Ethernet connection and a USB socket used for standard USB devices[2].

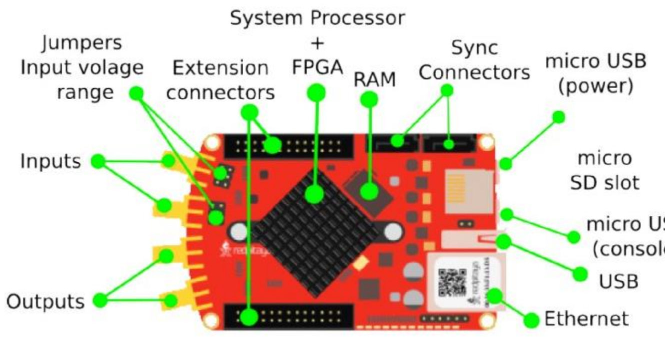


Figure 2: RedPitaya V1.1 Tech Specs

Some of the main features:

- High performance hardware at a surprising price tag.
- Replaces most essential instruments: (Oscilloscope, Spectrum analyzer, Signal generator, LCR meter)
- LAN or wireless access from any WEB browser via tablet or a PC regardless of the OS (MAC, Linux, Windows, Android, iOS).
- LabView and MATLAB® interface.
- Possibility to make your own application and share it with others.
- Open source software.

RF inputs:	RF outputs:
• Number of channels: 2	• Number of channels: 2
• Bandwidth: 50 MHz (3 dB)	• Bandwidth: 50 MHz
• Sample rate: 125 Msps	• Sample rate: 125 Msps
• Input impedance: 1 MΩ / 10 pF	• Input impedance: 50 Ω
• Connector: SMA-F	• Connector: SMA-F

Auxiliary analog input channels:	Auxiliary analog output channels:
• Number of channels: 4	• Number of channels: 4
• Nominal sampling rate: 100 ksps	• Output type: Low pass filtered PWM
• ADC resolution: 14 bits	• PWM time resolution: 4ns (1/250 MHz)
• Input voltage range: 0 to +3.5 V	• Output voltage range: 0 to +1.8 V
• Input coupling: DC	
• Connector: IDC connector	

RedPitaya is further equipped with sonic sensor which sends ultrasonic waves to an object and receives data as a reflection depending upon object.

C. *Python with Raspberry Pi*

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages[3].

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive − You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python with Raspberry Pi lets you connect your project to the real world. Python syntax is very clean, with an emphasis on readability.

D. *Python Libraries & Modules*

In this project we are using different python libraries, packages and modules in order to achieve required functionality, which are listed below.

1) *RedPitaya SCPI*

STEMlab board can be controlled remotely over LAN or wireless interface using Matlab, Labview, Scilab or Python via RedPitaya SCPI (Standard Commands for Programmable Instrumentation) list of commands. SCPI interface/environment is commonly used to control T&M instruments for development, research or test automation purposes. SCPI uses a set of SCPI commands that are recognized by the instruments to enable specific actions to be taken (e.g.: acquiring data from fast analog inputs, generating signals and controlling other periphery of the RedPitaya STEMlab platform). The SCPI commands are extremely useful when complex signal analysis is required where SW environment such as MATLAB provides powerful data analysis tools and SCPI commands simple access to raw data acquired on STEMlab board[4].

Features:

- Quickly write control routines and programs using Matlab, Labview, Scilab or Python
- Use powerful data analysis tools of Matlab, Labview, Scilab or Python to analyze raw signals acquired by STEMlab board
- Write testing scripts and routines
- Incorporate your STEMlab and Labview into testing and production lines
- Take quick measurements directly with your PC

2) *Matplotlib*

matplotlib.pyplot is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g. creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that "axes" here and in most places in the documentation refers to the axes part of a figure and not the strict mathematical term for more than one axis)[5].

3) *Numpy*

NumPy is a Python package which stands for ‘Numerical Python’. It is the core library for scientific computing, which contains a powerful n-dimensional array object, provide tools for integrating C, C++ etc. It is also useful in linear algebra, random number capability etc. NumPy array can also be used as an efficient multi-dimensional container for generic data[6]. Now, let me tell you what exactly is a python numpy array.

NumPy Array: Numpy array is a powerful N-dimensional array object which is in the form of rows and columns. We can initialize numpy arrays from nested Python lists and access it elements. In order to perform these numpy operations.

4) Tsfresh

tsfresh is a python package. It automatically calculates a large number of time series characteristics, the so called features. Further the package contains methods to evaluate the explaining power and importance of such characteristics for regression or classification tasks[7].

Why do you need such a module?

tsfresh is used to extract characteristics from time series. Let's assume you recorded the ambient temperature around your computer over one day as the following time series:

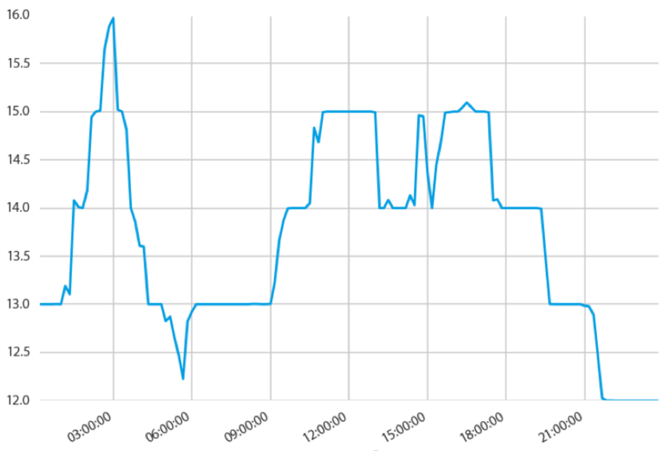


Figure 3: Signal

Now you want to calculate different characteristics such as the maximal or minimal temperature, the average temperature or the number of temporary temperature peaks:

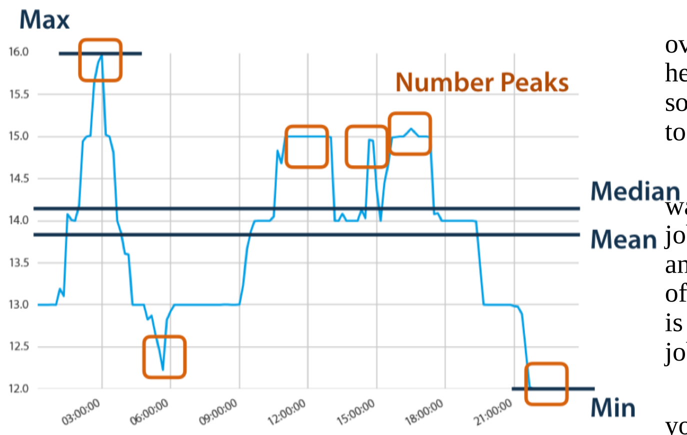


Figure 4: Features of Signal

Without tsfresh, you would have to calculate all those characteristics by hand. With tsfresh this process is automated and all those features can be calculated automatically.

What to do with these features?

The extracted features can be used to describe or cluster time series based on the extracted characteristics. Further, they can be used to build models that perform classification/regression tasks on the time series. Often they and their dynamics.

The tsfresh package has been used successfully in projects involving

- the prediction of the life span of machines
- the prediction of the quality of steel billets during a continuous casting process [8].

5) Sklearn

Also referred as Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy[9].

It aims to provide simple and efficient solutions to learning problems that are accessible to everybody and reusable in various contexts: machine-learning as a versatile tool for science and engineering.

6) Joblib

Joblib is a set of tools to provide lightweight pipelining in Python. In particular, joblib offers:

- transparent disk-caching of the output values and lazy re-evaluation (memorize pattern)
- easy simple parallel computing
- logging and tracing of the execution

Joblib is optimized to be fast and robust on large data and has specific optimizations for numpy arrays. It is BSD-licensed.

The vision is to provide tools to easily achieve better performance and reproducibility when working with long running jobs [10].

Avoid computing twice the same thing: code is rerun over and over, for instance when prototyping computational-heavy jobs (as in scientific development), but hand-crafted solution to alleviate this issue is error-prone and often leads to unreproducible results

Persist to disk transparently: persisting in an efficient way arbitrary objects containing large data is hard. Using joblib's caching mechanism avoids hand-written persistence and implicitly links the file on disk to the execution context of the original Python object. As a result, joblib's persistence is good for resuming an application status or computational job, e.g. after a crash.

Joblib strives to address these problems while leaving your code and your flow control as unmodified as possible (no framework, no new paradigms).

II. Machine learning

Machine Learning is a method of data analysis that automates analytical model building. It is a branch of Artificial Intelligence based on the idea that machines should be able to learn and adapt through experience.

A. *Evolution of Machine Learning*

- Alan Turing – 1950: Alan Turning, a mathematician who initiated Artificial Intelligence during 1940's and created a "Turing Test" to find whether a computer is intelligent or not?
- Computer Learning Program by Arthur Samuel – 1952: Arthur Samuel in collaboration with IBM created a machine learning program which learned to play the game Checkers. By the mid 1970's, this program was capable of beating human players.
- The Perceptron – 1957: Perceptron is a part of Neural Network which was developed by Frank Rosenblatt at the US office of Naval Research for Visual Recognition. In 1969, Marine and Saima published a book on Perceptron which showed the limitations of Perceptron and Neural Networks.
- The Nearest Neighbor Algorithm – 1967: The Nearest Neighbor algorithm allows recognizing basic pattern into given data set.
- The Stanford Cart – 1979: Stanford Cart invented by the students of MIT was capable to navigate obstacles in a room on its own.
- EBL – Explanation Based Learning – 1981: The program analyses the training information and avoids irrelevant information to form a general rule to follow.
- Data-Driven Approach to Machine Learning – 1990: Scientist started making application/program for a large amount of data. Here, it picks up!
- IBM's Deep Blue – 1997: It beats the world champion Garry Kasparov at a game of chess.
- IBM Watson – 2011: Using Natural Language Processing (NLP), IBM Watson beats the human champion in the game of Jeopardy.

Machine Learning is developing from a long time. It is so popular now because of BIG DATA and CLOUD STORAGE. Big data helps machine learning in the same way as catalyst does for chemical reaction[11].

B. *How exactly do we train the machine?*

Teaching the machines involve a structural process where every stage builds a better version of the machine. For simplification, process of teaching machines is given in 3 parts:

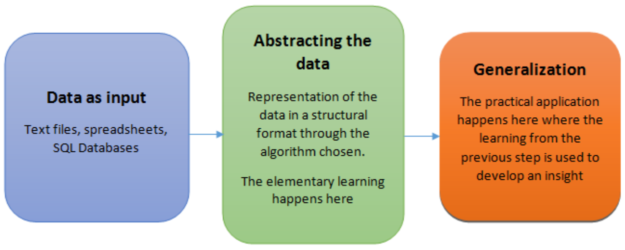

Data as input
Text files, spreadsheets, SQL Databases
Abstracting the data
Representation of the data in a structural format through the algorithm chosen.
The elementary learning happens here
Generalization
The practical application happens here where the learning from the previous step is used to develop an insight

Figure 5: Process of teaching machine

Success of a machine depends on two factors:

- How well the generalization of abstraction data take place.
- How well the machine can put it's learning into practical usage for predicting the future course of action[12].

C. *Machine Learning Algorithms*

- **Supervised Learning:** These algorithms are trained using labeled examples such as an input where the desired output is known. E.g.: a piece of equipment could have data points labeled either "F" (Failed) or "R" (Run). The learning algorithm receives a set of inputs along with the corresponding outputs, the algorithm learns by comparing its actual output with the correct output to find errors, it then modifies the model accordingly. Through methods like classification, regression, prediction and gradient boosting, supervised learning uses patterns to predict the values of the label on additional unlabeled data. Supervised learning is commonly used in applications where historical data predicts likely future events. E.g.: it can anticipate when credit card transactions are likely to be fraudulent or which insurance customer is likely to file a claim.
- **Unsupervised Learning:** it is used against data that has no historical labels. The system is not told the "right answer". The algorithm must figure out what is being shown. The goal is to explore the data and find some structure within. Unsupervised Learning works well on transactional data. E.g.: it can identify segments of customers with similar attributes that separate customer segments from each other. Popular techniques include self-organizing maps, nearest-neighbor mapping, k-means clustering and singular value decomposition. These algorithms are used to segment text topics, recommend items and identify data outliers.
- **Semisupervised Learning:** it is used for same applications as supervised learning. But it uses both labeled and unlabeled data (because unlabeled data is less expensive and takes less effort to acquire). This type of learning can be used with methods like classification, regression and prediction. Semisupervised learning is useful when the cost associated with labeling is too high to allow for a fully labeled training process. Early examples of this include identifying a person's face on web cam.
- **Reinforcement Learning:** it is often used for robotics, gaming and navigation. With this learning,

the algorithm discovers through trial and error which actions yield the greatest rewards. This type of learning has 3 primary components: Agent (learner or decision maker), Environment (everything the agent interacts with) and Actions (what the agent can do). The objective is for the agent to choose actions that maximize the expected reward over a given amount of time. Agent will reach the goal much faster by following a good policy. The goal in this algorithm is to learn the best policy [13].

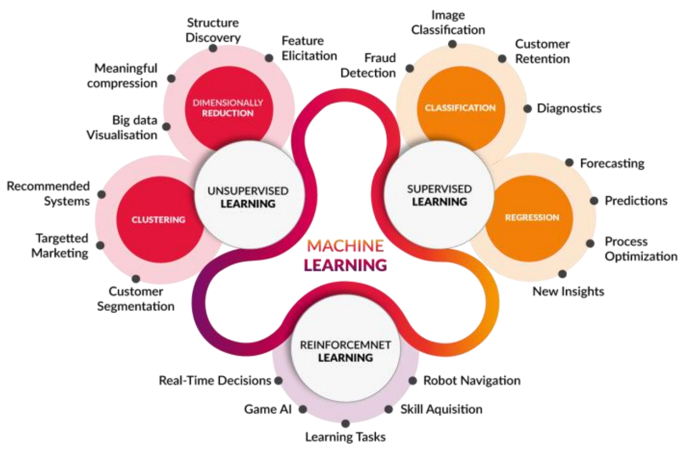


Figure 6: Machine Learning Algorithm Types

D. *Advantages of Machine Learning*

- **Image recognition:** this feature is widely spread among mobile application. E.g.: it is sometimes used for identification purposes or work with photos including filters and editing. Besides, using different types of machine learning algorithms you can define user's sex and age within an app, implement the recognition of an eye retina or fingerprint etc., A good example of machine leaning application is the recognition of license plates o roads in case of violence.

- **Voice recognition:** Apple's Siri and Google Now use a list of machine learning algorithms to recognize the user's command and react to them.

- **Optical character recognition:** This feature recognizes documents, credit cards, translate foreign words on pictures, etc., It is important to consider that a text has a variety of characters such as font, size and more. That is why this model of algorithms for machine learning should be built having these characteristics in mind.

- **Advanced customization:** It is used for creation of personalized content that will be able to take the user's preferences into account.

- **Intelligent data analysis**: for this purpose, machine learning along with Big Data is used because Big Data collects information while machine learning processes as well learns from it. It is used to make further predictions based on certain data. Such combinations are already used in Amazon and Google in some of their services.

- **Sensory data analysis:** This technology is used in medicine as well. In modern iOS and Android apps can keep track of person's heartbeat, count steps, etc., using machine learning applications your app will be able to monitor users' activity constantly [14].

E. *Applications of Machine Learning*

- **Financial Services:** Banks and other business in financial industry use machine learning technology for 2 purposes: to identify important insights in data and prevent fraud. The insights can identify investment opportunities or help investors know when to trade. Data mining can also identify clients with high-risk profiles or use cybersurveillance to pinpoint warning signs of fraud.

- **Health care:** Machine Learning is a fast-growing trend in health-care industry, due to the advent of wearable devices and sensors that can use data to assess a patient's health in real-time. The technology can also help medical experts analyse data to identify trends or red flags that may lead to improved diagnoses and treatment.

- **Oil and Gas:** Finding new energy sources, analysing minerals in ground, predicting refinery sensor failure, streamlining oil distribution to make it more efficient and cost effective. The usage of machine learning in this industry is vast and still expanding.

- **Government:** Government agencies such as public safety and utilities have a need for machine learning due to the multiple sources of data that can be mined for insights. Analysing sensor data, i.e., identifies ways to increase efficiency and save money. Machine learning can also help detect fraud and minimize identity theft.

- **Marketing and Sales:** Websites recommending items you might like based on previous purchases are based on machine learning to analyse your buying history and promote others items you'd be interested in. this ability to capture data, analyse it and use it to personalize shopping experience is the future of retail.

- **Transportation:** Analysing data to identify patterns and trends is key to transportation industry, which relies on making routes more efficient and predicting potential problems to increase profitability. The data analysis and modelling aspects of machine learning are important tools to delivery companies, public transportation and other transportation organizations[15].

III. Classification Algoritms

A group of algorithms that could be viewed as a specific application of Decision making is Classification algorithms. Classification algorithms are algorithms that assign a certain label, or classification, to an input.

For Automation lab project we are sending ultrasonic waves towards an object and recognizing this object by the reflected wave by applying machine learning (ML) to the measured data in order to perform the recognition task. For machine learning we are applying two different classifier/algorithms to get the result and comparing result for both algorithms.

A. *Support Vector Machine*

Support Vector Machine is one of the supervised Machine Learning Technique, which was first heard during COLT-92 introduced by Vapnik, Boser, Guyon. Support Vector Machines are used for classification and regression; it belongs to generalized linear classifiers. SVM is a mostly used method in pattern recognition and object recognition. The objective of the support vector machine is to form a hyperplane as the decision surface in such a way that the margin of separation between positive and negative examples is maximized by utilizing optimization approach. Generally linear functions are used as a separating hyperplane in the feature space. For achieving better performance, several kernel functions are used such as polynomial function and radial-bias function, in this paper, polynomial function is used as kernel function. When using kernel functions, the scalar product can be implicitly computed in a kernel feature space[16].

B. *Bayes Classifier*

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the colour, roundness, and diameter features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers. Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests[17].

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

IV. Working of the Project

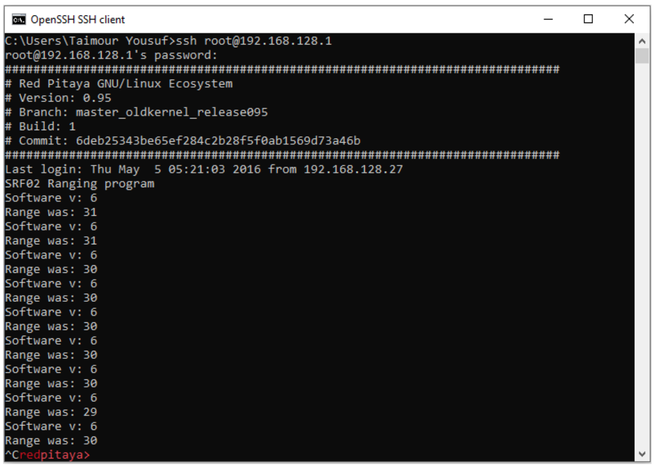
The working of this project is divided into mainly 5 steps through which we can analyse, save signals from Red Pitaya and perform feature extraction, classification.

A. *Connection with RedPitaya*

Power on Red Pitaya and after 30 seconds connect via Wi-Fi of your Raspberry pi, password is 'redpitya' by default. After successful connection ssh your Red pitaya with command 'ssh root@192.168.128.1' it will ask for password which is 'root' by default. Once you run this command and point Red Pitaya sonic sensor to any object it will start giving you reading as per object's nature.

Make sure that position of Red Pitaya sensor should be nearly 1 meter away from an object and It should be on 1-meter height above from ground while getting readings.

RedPitaya is already configured and we are told not to change any configuration in it but for learning purpose you can see the GUI version by opening 192.168.128.1 in your web browser and also through command line, you can generate different waves.



```
C:\Users\Taimour Yousuf>ssh root@192.168.128.1
root@192.168.128.1's password:
##############################################################################
# Red Pitaya GNU/Linux Ecosystem
# Version: 0.95
# Branch: master_oldkernel_release095
# Build: 1
# Commit: 6deb25343be65ef284c2b28f5f0ab1569d73a46b
##############################################################################
Last login: Thu May  5 05:21:03 2016 from 192.168.128.27
SRF02 Ranging program
Software v: 6
Range was: 31
Software v: 6
Range was: 31
Software v: 6
Range was: 30
Software v: 6
Range was: 30
Software v: 6
Range was: 30
Software v: 6
Range was: 30
Software v: 6
Range was: 30
Software v: 6
Range was: 29
Software v: 6
Range was: 30
^Credpitaya>
```

Figure 7: SSH Connection

B. *Signal Generation*

After successful connection with Red Pitaya and analysing readings, we have now converted readings into signal graph with the help of python code. Since through signal we can easily differentiate between objects e.g. Human, Wall & Car. Kindly note that signal strength may vary depending upon distance, height and angle from an object, so this is just to give you rough idea how it looks.

Signal representation are shown below.

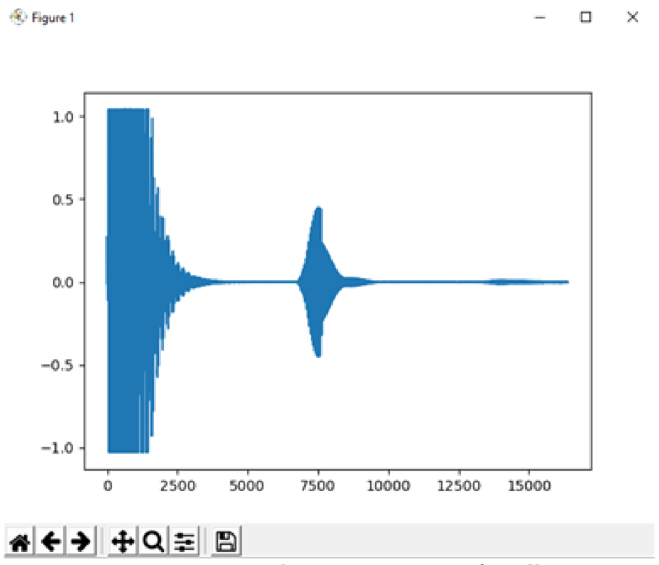


Figure 8: Signal Representation of Wall

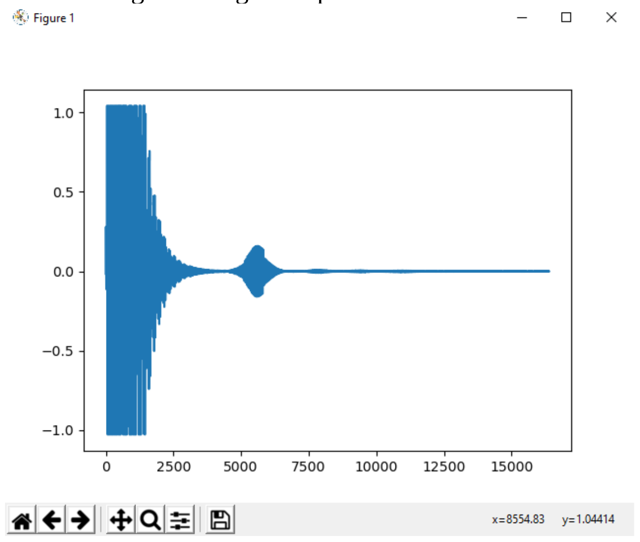


Figure 9: Signal Representation of Human

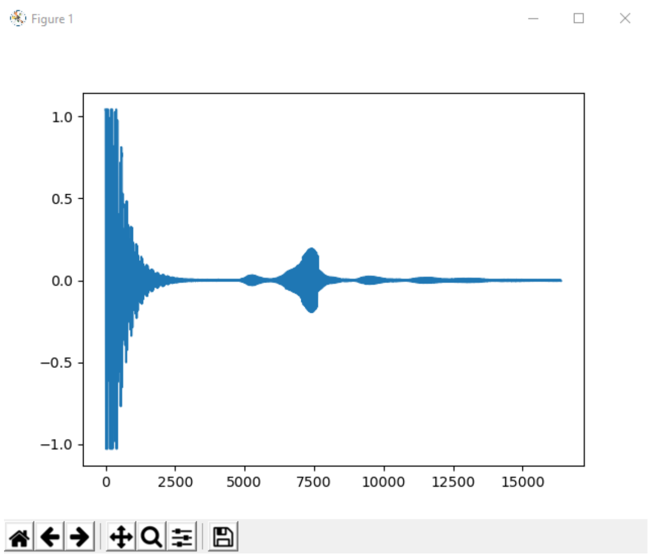


Figure 10: Signal Representation of Car

Python code for signal representation

```
import redpitaya_scpi as scpi
import matplotlib.pyplot as plt
import numpy as np
rp = scpi.scpi('192.168.128.1')
while 1:
    rp.tx_txt('ACQ:DEC 64')
    rp.tx_txt('ACQ:TRIG EXT_PE')
    rp.tx_txt('ACQ:TRIG:DLY 8192')
    rp.tx_txt('ACQ:START')
    while 1:
        rp.tx_txt('ACQ:TRIG:STAT?')
        if rp.rx_txt() == 'TD':
            break
        rp.tx_txt('ACQ:SOUR1:DATA?')
        str = rp.rx_txt()[1:-1]
        measRes = np.fromstring(str,dtype=float,sep=',')
        plt.plot(measRes)
        plt.pause(1)
        plt.clf()
```

C. *Saving signal samples*

Now we have saved those signals reading in a .csv file through python code and we have collected roughly 100+ samples of each object and each sample have 16000 values e.g. for car we have collected 25 samples from front, 25 from back, 25 from each sides. All this data is stored in .csv file for further evaluation like feature extraction.

Make sure through signal graph that collected data is accurate otherwise it will create issue for feature extraction and mainly at the time of classification.

Python code for saving data.

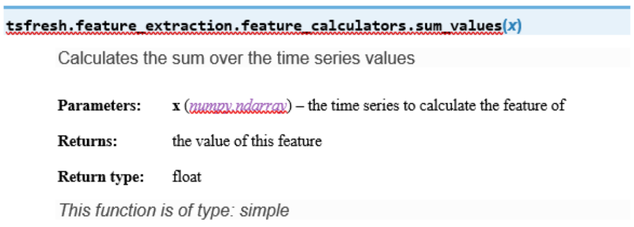
```
import redpitaya_scpi as scpi
import matplotlib.pyplot as plt
import numpy as np
import time
rp = scpi.scpi('192.168.128.1')
while 1:
    rp.tx_txt('ACQ:DEC 64')
    rp.tx_txt('ACQ:TRIG EXT_PE')
    rp.tx_txt('ACQ:TRIG:DLY 8192')
    rp.tx_txt('ACQ:START')
    while 1:
        rp.tx_txt('ACQ:TRIG:STAT?')
        if rp.rx_txt() == 'TD':
            break
    rp.tx_txt('ACQ:SOUR1:DATA?')
    str = rp.rx_txt()[1:-1]
    measRes = np.fromstring(str,dtype=float,sep=',')
    plt.plot(measRes)
    plt.show()
    signal = np.array([measRes])
    f = open("Wall.csv", "a")
    np.savetxt(f,signal,fmt='%3.8f',delimiter=',')
    f.close()
    time.sleep(3)
    plt.close()
```

D. *Feature Extraction*

On collected data we have performed feature extraction, there are a lot of features present but we are using 3 of them which are enough in order to differentiate between 3 objects you can increase number of feature extractions in order to get more accurate results but make sure that feature has only parameter to extract data from.

Python code for feature extraction.

```
import numpy as np
from numpy import genfromtxt
import tsfresh

data1 =
np.array(genfromtxt('CollectedData/Wall.csv',delimiter=','))

for i in range(data1.shape[0]):
    shortData1 = np.array([data1[i][3500:]])
feature1 = tsfresh.feature_extraction.feature_calculators
.abs_energy(shortData1[0])
feature2 = tsfresh.feature_extraction.feature_calculators
.sum_values(shortData1[0])
feature3 = tsfresh.feature_extraction.feature_calculators
.skewness(shortData1[0])
    features = np.array([[feature1, feature2, feature3]])
    f = open("FeaturedData/WallFeatures.csv", "a")
    np.savetxt(f,features,fmt='%3.8f',delimiter=',')
    f.close()

data2 =
np.array(genfromtxt('CollectedData/Human.csv',delimiter=','))
for i in range(data2.shape[0]):
    shortData2 = np.array([data2[i][3500:]])
feature1 = tsfresh.feature_extraction.feature_calculators
.abs_energy(shortData2[0])
feature2 = tsfresh.feature_extraction.feature_calculators
.sum_values(shortData2[0])
feature3 = tsfresh.feature_extraction.feature_calculators
.skewness(shortData2[0])
    features = np.array([[feature1, feature2, feature3]])
    f = open("FeaturedData/HumanFeatures.csv", "a")
    np.savetxt(f,features,fmt='%3.8f',delimiter=',')
    f.close()

data3 =
np.array(genfromtxt('CollectedData/Car.csv',delimiter=','))
for i in range(data3.shape[0]):
    shortData3 = np.array([data3[i][3500:]])
feature1 = tsfresh.feature_extraction.feature_calculators
.abs_energy(shortData3[0])
feature2 = tsfresh.feature_extraction.feature_calculators
.sum_values(shortData3[0])
feature3 = tsfresh.feature_extraction.feature_calculators
.skewness(shortData3[0])
    features = np.array([[feature1, feature2, feature3]])
    f = open("FeaturedData/CarFeatures.csv", "a")
    np.savetxt(f,features,fmt='%3.8f',delimiter=',')
    f.close()
```



`tsfresh.feature_extraction.feature_calculators.sum_values(x)`

Calculates the sum over the time series values

Parameters:	x (numpy.ndarray) – the time series to calculate the feature of
Returns:	the value of this feature
Return type:	float

This function is of type: simple

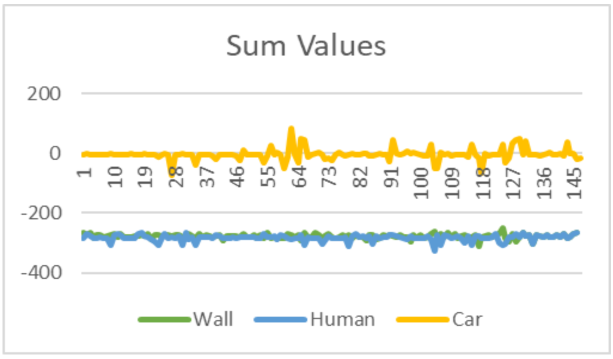
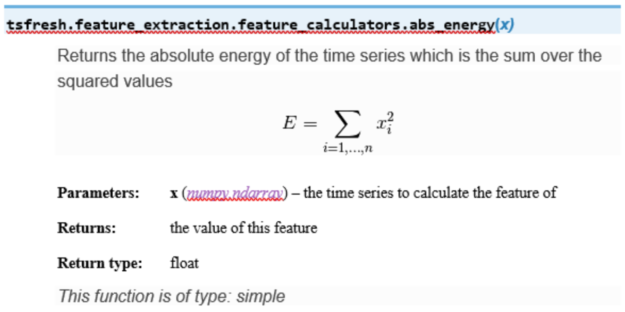


Figure 9: Graph of sum values for all objects



`tsfresh.feature_extraction.feature_calculators.abs_energy(x)`

Returns the absolute energy of the time series which is the sum over the squared values

$$E = \sum_{i=1,\ldots,n} x_i^2$$

Parameters:	x (numpy.ndarray) – the time series to calculate the feature of
Returns:	the value of this feature
Return type:	float

This function is of type: simple

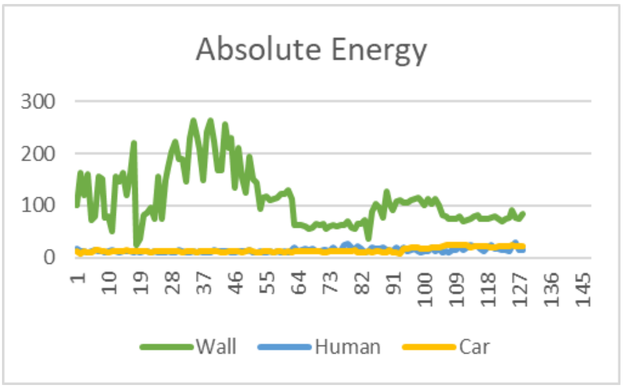
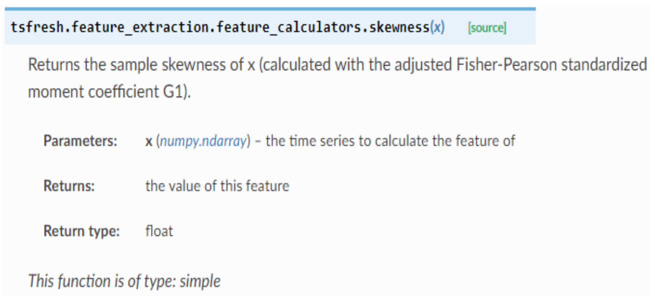


Figure 10: Graph of absolute energy for all objects



`tsfresh.feature_extraction.feature_calculators.skewness(x)` [source]

Returns the sample skewness of x (calculated with the adjusted Fisher-Pearson standardized moment coefficient G1).

Parameters:	x (numpy.ndarray) – the time series to calculate the feature of
Returns:	the value of this feature
Return type:	float

This function is of type: simple

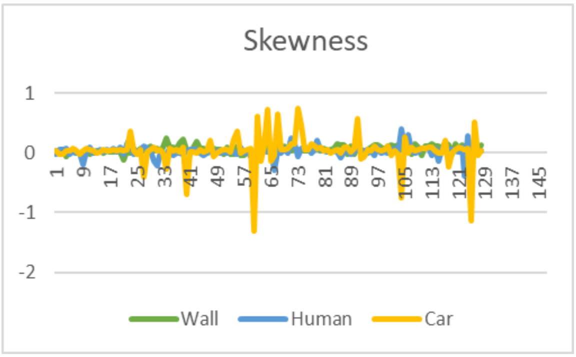


Figure 11: Graph of Skewness for all objects

E. *Classification*

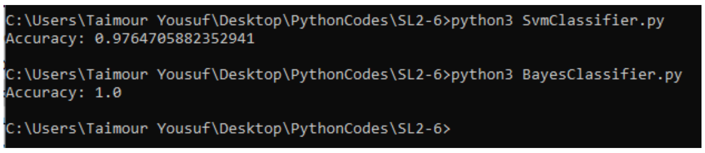
As per given task we have to do classification via two different methods and compared results of both, so we have used SVM and Bayes classifier. On saved data we have applied classification one by one and compared results from both of them.

We are training 80% of data with GaussianNB() and saving it to BayesClassifierModel.joblib so that It can predict result on runtime during classification. For testing we are using 20% of data which are giving accuracy of 1 every time, which indicates that this classifier is working well for collected data.

```
import numpy as np
from numpy import genfromtxt
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn import metrics
from joblib import dump, load

data1 = np.array(genfromtxt('FeaturedData/WallFeatures.csv',delimiter=','))
data2 =
np.array(genfromtxt('FeaturedData/HumanFeatures.csv',delimiter=','))
data3 =
np.array(genfromtxt('FeaturedData/CarFeatures.csv',delimiter=','))

x = np.concatenate((data1,data2,data3))

label1 = np.repeat(-1,data1.shape[0])
label2 = np.repeat(0,data2.shape[0])
label3 = np.repeat(1,data3.shape[0])

y = np.concatenate((label1,label2,label3))

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2)

clf = GaussianNB()
clf.fit(x_train, y_train)
dump(clf,'BayesClassifierModel.joblib')

clf_loaded = load('BayesClassifierModel.joblib')

y_pred = clf_loaded.predict(x_test)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

On this as well we are training 80% data with svm.SVC(kernel='linear') and saving it to SvmClassifierModel.joblib so that It can predict result on runtime during classification. For testing we are using 20% of data which are giving accuracy of 0.97 mostly, which indicates that this classifier is also working fine.

```
import numpy as np
from numpy import genfromtxt
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn import metrics
from joblib import dump, load

data1 = np.array(genfromtxt('FeaturedData/WallFeatures.csv',delimiter=','))
data2 =
np.array(genfromtxt('FeaturedData/HumanFeatures.csv',delimiter=','))
data3 =
np.array(genfromtxt('FeaturedData/CarFeatures.csv',delimiter=','))

x = np.concatenate((data1,data2,data3))

label1 = np.repeat(-1,data1.shape[0])
label2 = np.repeat(0,data2.shape[0])
label3 = np.repeat(1,data3.shape[0])

y = np.concatenate((label1,label2,label3))

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2)

clf = svm.SVC(kernel='linear')
clf.fit(x_train, y_train)
dump(clf,'SvmClassifierModel.joblib')

clf_loaded = load('SvmClassifierModel.joblib')
y_pred = clf_loaded.predict(x_test)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```


```
C:\Users\Taimour Yousuf\Desktop\PythonCodes\SL2-6>python3 SvmClassifier.py
Accuracy: 0.9764705882352941

C:\Users\Taimour Yousuf\Desktop\PythonCodes\SL2-6>python3 BayesClassifier.py
Accuracy: 1.0

C:\Users\Taimour Yousuf\Desktop\PythonCodes\SL2-6>
```
Figure 12: Accuricies from both classifiers

Difference between both approaches and conclusion:

We have trained data with both classifiers and as per result accuracy is slightly better for Bayes than SVM and it totally depends on data that we have collected.

We are using Support Vector Classification for the case of a linear kernel and performance wise SVMs using the radial basis function kernel are more likely to perform better as they can handle non-linearities in the data.

Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution. In general, Naive Bayes performs best when the features are independent of each other which often does not happen in real but in our case, it is independent, and therefore accuracy is better.

Now this is now our main code for performing classification and since we have decided to go with the Bayes Classifier due to accuracy, so we have loaded trained data from BayesClassifierModel.joblib and predicting results from runtime data.

```
import redpitaya_scpi as scpi
import numpy as np
import tsfresh
from joblib import load
import time

rp = scpi.scpi('192.168.128.1')
while 1:
  rp.tx_txt('ACQ:DEC 64')
  rp.tx_txt('ACQ:TRIG EXT_PE')
  rp.tx_txt('ACQ:TRIG:DLY 8192')
  rp.tx_txt('ACQ:START')
  while 1:
    rp.tx_txt('ACQ:TRIG:STAT?')
    if rp.rx_txt() == 'TD':
      break
  rp.tx_txt('ACQ:SOUR1:DATA?')
  str = rp.rx_txt()[1:-1]
  measRes = np.fromstring(str,dtype=float,sep=',')
  data = np.array(measRes[3500:])
  feature1=
tsfresh.feature_extraction.feature_calculators.abs_energy(data)
  feature2=
tsfresh.feature_extraction.feature_calculators.sum_values(data)
  feature3=
tsfresh.feature_extraction.feature_calculators.skewness(data)
  features = np.array([[feature1, feature2, feature3]])
  clf_loaded = load(BayesClassifierModel.joblib ')
  class_pred = clf_loaded.predict(features)
  if class_pred == -1:
    rp.tx_txt('DIG:PIN LED1,1')
    rp.tx_txt('DIG:PIN LED2,0')
    rp.tx_txt('DIG:PIN LED3,0')
  elif class_pred == 0:
    rp.tx_txt('DIG:PIN LED1,0')
    rp.tx_txt('DIG:PIN LED2,1')
    rp.tx_txt('DIG:PIN LED3,0')
  elif class_pred == 1:
    rp.tx_txt('DIG:PIN LED1,0')
    rp.tx_txt('DIG:PIN LED2,0')
    rp.tx_txt('DIG:PIN LED3,1')
    time.sleep(1)
```

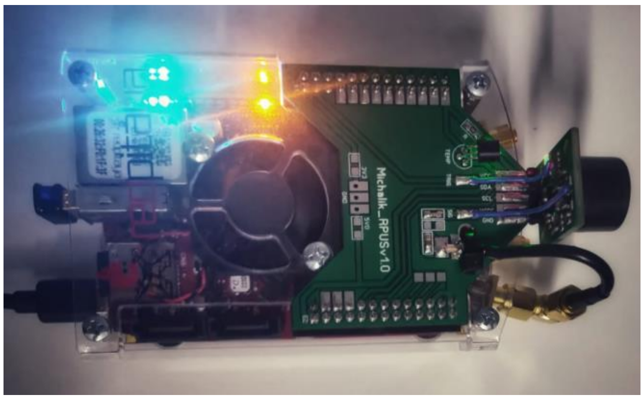


Figure 13: Wall recognization Led1 is on

Led1 on Red Pitaya will turn on for Wall, Led2 for Human and Led3 for Car.

We can also load trained data from SvmClassifierModel.joblib which is also working fine with accuracy 0.97.

F. *Plug and Play*

To make 'plug and play' the whole project, we have done some adjustment in cron job which is a scheduler in Raspbian and very easy to use. Open command line and type 'sudo crontab -e' it will show you 3 option for different editor, select 1 and write below commands and save it. It will execute these commands in background after every 10 and 5 minutes respectively.

```
*/10 * * * *  sudo sshpass -p “root” ssh root@192.168.128.1 &
*/5 * * * * sudo python3 /home/pi/Desktop/SL2-6/Classification.py &
```

Install and update all required packages e.g. sshpass, tsfresh, sklearn etc before forgetting your internet Wi-Fi and make sure that Raspberry pi is only connected with Red Pitaya Wi-Fi.

Keep your Red Pitaya on while rebooting Raspberry Pi so that it can automatically connect with Wi-Fi.

Alternatively, we can also make some changes in the script (etc/rc.local) file of our raspberry pi, this file is traditionally executed after all the normal system services are started. Open command line and with chmod (change mode) make rc.local script executable and through vi editor edit it with given text.

```
sudo chmod +x /etc/rc.local
sudo vi /etc/rc.local
```

```
sshpass -p “root” ssh root@192.168.128.1 &
python3 /home/pi/Desktop/SL2-6/Classification.py &
exit 0
```

Acknowledgment

We have taken efforts in this project. However, it would not have been possible without the kind support and help of our professors and organizations who extend the time for this project during this COVID-19 pandemic situation, which we all are facing. I would like to extend my sincere thanks to all of them.

Our thanks and appreciations also go to my classmates in developing the project and people who have willingly helped me out with their abilities.

References

[1] https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

[2] https://www.elektor.com/stemlab-125-14-diagnostic-kit

[3] https://www.tutorialspoint.com/python/python_overview.htm

[4] https://github.com/RedPitaya/RedPitaya/blob/master/doc/appsFeatures/remoteControl/remoteControl.rst

[5] https://matplotlib.org/tutorials/introductory/pyplot.html

[6] https://www.edureka.co/blog/python-numpy-tutorial/

[7] https://tsfresh.readthedocs.io/en/latest/text/introduction.html

[8] Christ, M., Kempa-Liehr, A.W. and Feindt, M. (2016). Distributed and parallel time series feature extraction for industrial big data applications. ArXiv e-prints: 1610.07717 http://adsabs.harvard.edu/abs/2016arXiv161007717C

[9] https://www.dataquest.io/blog/sci-kit-learn-tutorial/

[10] https://medium.com/@measurespace/use-joblib-to-run-your-python-code-in-parallel-ad82abb26954

[11] https://bitsdroid.com/history-of-machine-learning/

[12] https://www.analyticsvidhya.com/blog/2015/06/machine-learning-basics/

[13] https://www.sas.com/en_us/insights/analytics/machine-learning.html#machine-learning-importance

[14] https://www.cleveroad.com/blog/importance-of-machine-learning-applications-in-various

[15] https://www.sas.com/en_us/insights/analytics/machine-learning.html#machine-learning-importance

[16] https://www.researchgate.net/publication/267808304_Object_Recognition_Using_Support_Vector_Machine_Augmented_by_RST_Invariants

[17] https://en.wikipedia.org/wiki/Naive_Bayes_classifier