Automated Robust First Peak Detection in a Time Signal using Computational Intelligence

M.Eng Information Technology
Computational Intelligence

Syed Mostain Ahmed
1390214
syed.ahmed@stud.fra-uas.de

Farjana Akter
1344037
farjana.akter@stud.fra-uas.de

Al Rahat Hossain
1345285
al.hossain@stud.fra-uas.de

***Abstract*— In numerous computational challenges within engineering and data science, the crucial task of identifying and analyzing peaks in noisy periodic time-series signals plays a pivotal role. This capability finds relevance across a wide spectrum of applications, allowing for a better understanding of abrupt fluctuations in diverse datasets. In this research paper, we present a comprehensive investigation employing two distinct methodologies. The first approach centers on signal processing techniques, where we employ specialized libraries to pinpoint the initial peak within a signal. Subsequently, we delve into the realm of Machine Learning (ML) methodologies, specifically Linear Regression and Support Vector Machine (SVM) algorithms. These ML techniques are utilized for training, prediction, and peak detection within the signal. Our research provides a detailed examination of the efficacy of these two approaches, facilitating a comparative analysis.**

Keywords*— *Signal Processing, Peak Detection, Machine Learning, Linear Regression, Support Vector Machine

I. INTRODUCTION

Peak detection within signals holds a pivotal role across various signal processing applications. The identification of peaks within data carries substantial societal significance, encompassing scenarios such as sudden fluctuations in stock markets, heart rate monitoring, traffic analysis, fuel price monitoring, and more. While the identification of peaks in single-variable time series data is relatively straightforward, it is imperative to establish a standardized definition of a peak to eliminate subjectivity. This standardization enables the development of algorithms capable of identifying peaks within diverse time series datasets. Presently, existing peak detection algorithms exhibit limitations in terms of seamless operation, underscoring the need for a straightforward and effective peak detection algorithm tailored for noisy signal analysis.

To maintain the broad applicability of such algorithms, several essential steps can be employed. These steps include the detection of all local maxima points, the application of intelligent techniques for manual peak verification, and the establishment of a peak detection methodology resilient to high and low-frequency noise [1].

The task of detecting peaks within a signal and precisely characterizing their positions, thresholds, magnitudes, widths, or locations stands as a fundamental function within data processing applications. One approach to tackle this challenge is to leverage the mathematical insight that the first derivative of a peak exhibits a zero-crossing precisely at the peak's maximum point. To detect the first peak in a time-series signal, a practical strategy involves the utilization of scientific computation libraries such as SciPy, which is part of the Scientific Python ecosystem and builds upon NumPy functionalities.

There are many practiced techniques for detecting peak in the Scientific industry. Among the notable methods that have been devised are the Hilbert transform [2], Kalman filtering [3], wavelet transforms [4], as well as hybrid approaches that combine Hilbert transforms, adaptive thresholding, and wavelet transforms [5]. Additionally, non-linear filtering techniques [6-9], K-Means Clustering [10], Gabor filtering [11], Gaussian second derivative filtering [12], histogram or cumulative distribution function-based approaches [13], and linear prediction analysis [14] have been harnessed to address the intricate challenge of peak detection.

However, beyond these established signal processing techniques, there exist two promising methods that not only excel in noise filtering but also leverage Machine Learning (ML) principles to detect the first peak within a time signal. These methodologies, namely Linear Regression and Support Vector Machine, have demonstrated remarkable efficacy in the realm of peak detection when employed intelligently.

In our research model, we harness the capabilities of Linear Regression and Support Vector Machine to perform noise reduction, ultimately enabling the accurate prediction of the initial peak within a given time signal. By integrating these ML-driven approaches into the domain of peak detection, our research seeks to provide a novel and effective solution to the challenges posed by noisy signals. Through this innovative approach, we aim to enhance the reliability

and precision of peak detection, opening new avenues for its application in a multitude of fields and scenarios.

This paper presents a comprehensive exploration of our model's workflow, which encompasses two distinct approaches for detecting the first peak in given test data files. Our journey commences with the creation of a proprietary algorithm rooted in signal processing techniques. Subsequently, we delve into an alternative path, leveraging the power of Machine Learning (ML) to develop an automated and robust peak detection methodology. Employing scientific computational libraries, we implement and evaluate both approaches for peak detection across a range of test data files.

In Section II of this paper, we delve into the intricate details of our project's implementation. We elaborate on the workflow adopted for peak detection within signal formations, elucidate the necessary steps for setting up the project environment, delve into the design and functionality of the Graphical User Interface (GUI), and provide insights into the essential packages that require installation to facilitate seamless project operation.

Section III constitutes a deep dive into the heart of our project's implementation. Here, we offer comprehensive insights into the methodologies adopted for data pre-processing, encompassing both signal processing and Machine Learning (ML) algorithms that have been integrated into our model. Additionally, we provide a concise overview of the results obtained and the outputs generated through our experimentation and analysis.

In conclusion, this study offers a profound understanding of our efforts in the domain of peak detection within noisy signals. By exploring two distinct approaches, signal processing, and ML, we have endeavored to enhance the accuracy and reliability of peak detection, ultimately contributing to the broader understanding of this critical field of research.

II. PROJECT IMPLEMENTATION

A. *Background Studies*

The ability to identify and effectively work with signal peaks holds immense importance across a wide spectrum of fields, ranging from engineering to economics. These peaks serve as critical markers, delineating the highest and lowest points within a dataset of functional values. This capability is instrumental in determining the positions and magnitudes of key statistical values, thereby facilitating predictive analysis and informed decision-making.

In this context, a peak, or a local maximum, is defined as a data point within the signal that boasts higher amplitude than its immediate neighbors. Specifically, it is a sample with two adjacent data points of smaller amplitudes. In cases where multiple data points exhibit equivalent amplitudes, the index of the central data point is considered the peak position. However, it is important to note that peak locations within noisy signals can exhibit variability due to the influence of noise, which may cause shifts in the positions of local maxima.

To address this challenge in noisy signals, it is advisable to consider signal smoothing techniques before embarking on peak detection. Alternatively, one may explore other advanced peak-finding and fitting methodologies [15] to enhance the accuracy and robustness of peak identification within such datasets.

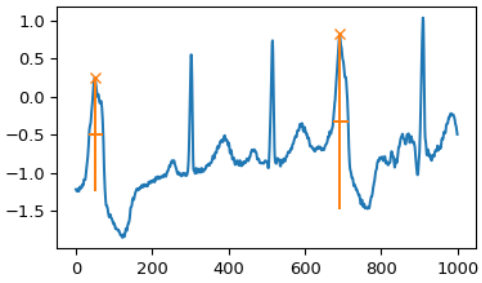

Figure. 1. Finding Peak using SciPy function [15].

Beginning with the provided test dataset, the first thing we do is to create a function which can anticipate the peaks. For that process, we did our research in some steps which are described in detail as below.

B. *Software Engineering Workflow*

We have implemented the whole process using Agile methodologies, where we first plan the whole process consisting of different approaches, design the system, develop our algorithm and then evaluate and test the built system.

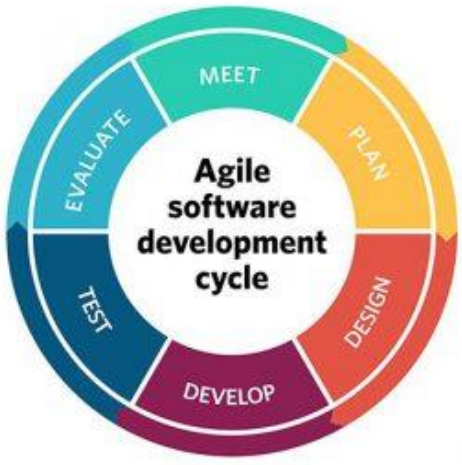

Figure 2: Agile methodology process

C. *Environment Setup*

To commence our experimental work, the initial step involves setting up and executing the project on our local machine. This entails configuring the necessary environment. In opting for the Python scripting language, we aimed to develop an algorithm capable of generating the desired output. Python's extensive technological capabilities afford a rich array of libraries conducive to computational intelligence. Furthermore, Python's versatility is highlighted by its compatibility with various programming paradigms, encompassing object-oriented and functional approaches.

For proper execution, the machine necessitates an installation of Python 3.10 or a more recent version. Additionally, the installation of the Anaconda environment or the PyCharm community edition is recommended. These tools streamline package management and facilitate seamless deployment.

D. User Interface

Tkinter, also referred to as Tk, stands as a pivotal GUI (graphical user interface) toolkit, seamlessly integrated with Python to facilitate graphical interactions. Executing a designated command in the command prompt effectively unveils a rudimentary Tk interface, affirming the successful installation of Tkinter on the system [16]. Within Tkinter, we leveraged the "file dialog" modules to effortlessly enable users to designate a file for opening or uploading.

```
python -m tkinter
```

E. Installed Packages or Libraries

Once we've set up everything initially, we move on to installing specific software packages needed to run our project. These packages are like toolkits with smart tools that make it easier to work with the complexity of artificial intelligence (AI). We need to install packages like Matplotlib (v 3.6.0), NumPy (v 1.23.3), openpyxl (v 3.0.10), Pandas (v 1.5.0), SciPy (v 1.9.1), and Tkinter to help our model generate clear signal peaks while filtering out unwanted noise using AI.

Matplotlib is a helpful tool for showing signals in a graph, and we're using a specific function called "find_peaks" to spot the highest points in our data. Pandas helps us read and organize the test data into an easy-to-read format, sort of like arranging information neatly in a table. SciPy does extra work for processing our signals in a useful way.

Now, for the machine learning part of our project, we're using a library called Scikit-learn. Think of it like a toolbox filled with different tools for sorting things. In our project, we tried two different tools from this toolbox: linear regression (imagine drawing a straight line to predict stuff) and SVM (Support Vector Machine, a bit like a smarter way to draw a line) to teach the computer and test how well it learned.

```
import csv
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg
import FigureCanvasTkAgg
from scipy.signal import find_peaks
import pandas as pd
import os
import tkinter as tk
from tkinter import filedialog
from tkinter import messagebox
```

F. Python IDE

Python scripts are written using an IDE (Integrated Development Environment). The python script for this project was written in a Pycharm tool. Also, under Anaconda, we can install Pycharm.

Pycharm is an extremely simple to use web-based or machine application tool. Once installed, run the jupyter notebook by typing jupyter notebook at the anaconda command prompt. This IDE offers a variety of critical Python developer tools that are deeply intertwined to offer a pleasant environment for efficient Python, web, and data science research.

III. IMPLEMENTATION

A. Signal Processing Approach

1. Pre-Processing

Noise removal is necessary as data was collected in a noisy environment. To do that we use a process called Hanning Window and numpy.convolve() with value 50. But we as this didn't actually help us in the long run for solving the problem and only enlarge the execution time we chose to go with only reducing noise in the found peaks.

```
def remove_noisy_peaks(pulses,
found_peaks):
    peak_heights =
found_peaks[1]['peak_heights']
    peak_positions =
pulses[found_peaks[0]]

    new_positions = []
    new_heights = []

    peak_mean =
np.average(peak_heights)

    for i in range(0,
len(peak_heights) - 1):
        if peak_heights[i] >
peak_mean:

new_heights.append(peak_heights[i])

new_positions.append(peak_positions[
i])

    return new_positions,
new_heights
```

2. Peak Detection

To identify peaks in our signal, we opted to utilize the "scipy.signal.find_peaks()" function. However, this function identifies all peaks in a signal, whereas we only require the initial peak. Additionally, the function's definition of a "peak" doesn't align

perfectly with our specific problem. Consequently, we implemented a minor post-processing step to pinpoint the first peak accurately.

```
peaks = find_all_peaks(y)
        peak_positions,
peak_heights =
remove_noisy_peaks(x, peaks)
        fig = draw_peaks(x, y,
peak_positions, peak_heights)

        # Display the graph in
the GUI
        canvas =
FigureCanvasTkAgg(fig,
master=root)
        canvas.draw()

canvas.get_tk_widget().pack(expa
nd=True, fill=tk.BOTH)

def find_all_peaks(y_axis):
    return find_peaks(y_axis,
height=y_axis, distance=1000)

def remove_noisy_peaks(pulses,
found_peaks):
    peak_heights =
found_peaks[1]['peak_heights']
    peak_positions =
pulses[found_peaks[0]]

    new_positions = []
    new_heights = []

    peak_mean =
np.average(peak_heights)

    for i in range(0,
len(peak_heights) - 1):
        if peak_heights[i] >
peak_mean:

new_heights.append(peak_heights[
i])

new_positions.append(peak_positi
ons[i])

    return new_positions,
new_heights
```

B. *Machine Learning Approach*

1. *Pre-Processing*

As mentioned earlier, the data sheets initially lack the necessary processing for machine learning purposes. Therefore, in order to proceed with our machine learning tasks, we needed to label the data. To achieve this, we marked the desired peak and assigned this peak value as a "Class" label, incorporating it into the respective datasheet. This allowed us to consider the signal (X) as our input data and classify it as the output (y), essential for training and implementing machine learning algorithms.

```
data =
pd.read_excel('2023_1_1.xlsx',
header=None, index_col=None)

    row, col = data.shape

    mod_list = np.zeros((row, col))

    for r in range(row):
        for c in range(col):
            val = data.iloc[r, c]
            val = val.astype(float)
            new_list[r][c] = val

    generated_dataframe =
pd.DataFrame(mod_list)


generated_dataframe.to_excel("2023_1
_modified.xlsx")
```

2. *Finding Peaks*

In our project, we harnessed the widely-used machine learning library for Python, "Scikit-learn," to facilitate data training and processing. Given our objective of obtaining a continuous stream of peak values, we opted to employ regression algorithms for both training and testing purposes. More specifically, we utilized the "Sklearn.linearregression" and "sklearn.SVM" algorithms.

During our comparative analysis of error scores, we determined that "SVR" from the "sklearn.SVM" module best suited our requirements for continuous peak value prediction. To further enhance our model's performance, we leveraged the "GridSearchCV()" function to systematically search for the optimal parameter settings, aiding in fine-tuning and optimizing our model.

```
y_train = train.Class
train.drop([Peak], axis=1,
inplace=True)
X_train = train

y_test = test.Peak
test.drop(['Peak'], axis=1,
inplace=True)
X_test= test
```

```
RA = SVR()
cross_val =
GridSearchCV(estimator=RA,

param_grid={'max_iter':[-1, 100,
500, 750, 1000]},
                             cv=4)
cross_val.fit(X_train, y_train)
pred = cross_val.predict(X_test)

print("First 5 prediction
{}".format(pred[525:]))
print("First 5 original labels
{}".format(y_test[525:]))

print(mean_absolute_error(y_test,
pred))
```

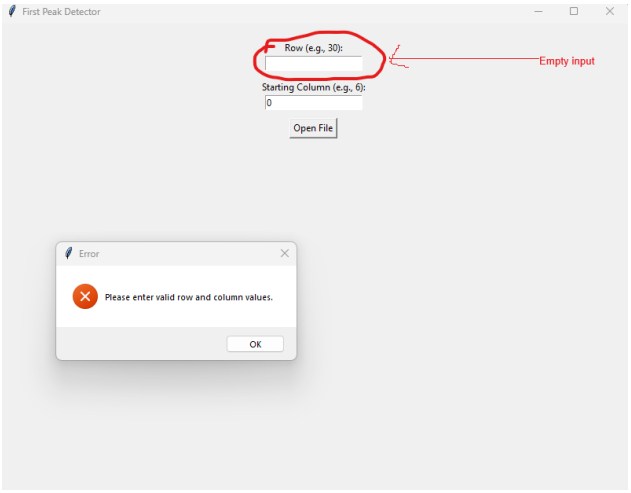

Figure 5: Showing error for empty fields.

C. User Interface:

1) Data Input: We made a user friendly interface. Where user can easily input row and column number and then select the datasheet from file explorer. We also add some checks for invalid data types or null data before letting the user choose their file.

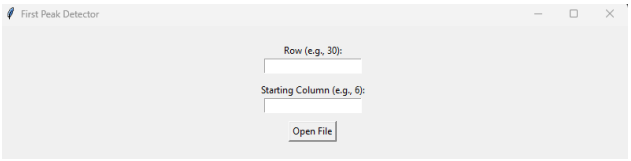

Figure:3 User input screen

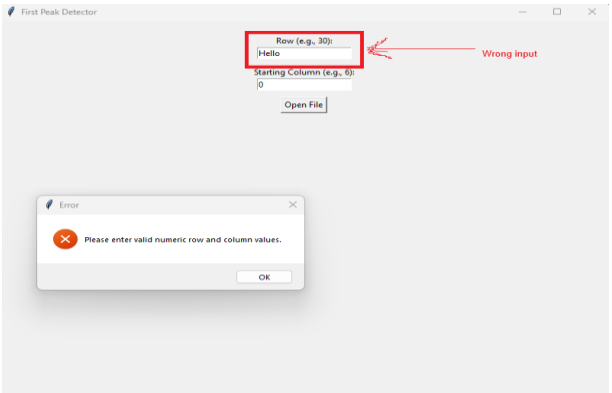

Figure 4: Showing error for string input

2) Data loading: After inputting valid data user can choose their desired file from the explorer as .csv or .xlsx from a the application will process the data

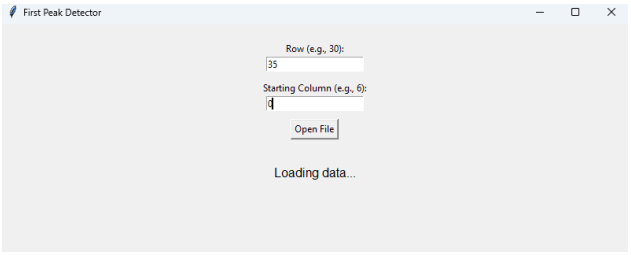

Figure 6: Loading data screen.

D. *Results:*

1. Signal processing:

The first datasheet provided has a data of 530 signals. After processing a large chunk of data, we got the below result.

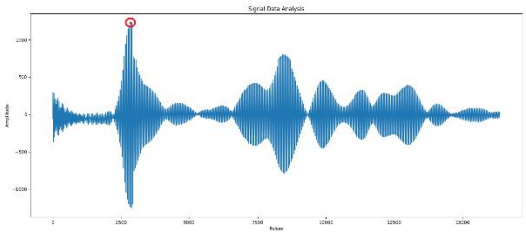

Fig. 7. First Peak find from large dataset

After giving a little less data the processed first peak looks like this: .

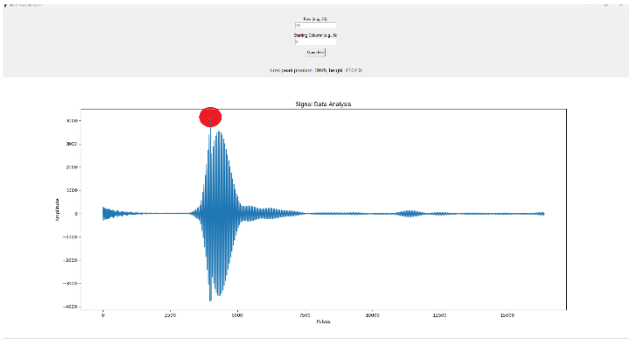

Figure. 8. Signal after small data processing using First Dataset

In the above figure, the large red dot on the signal is the first calculated peak. Regarding second dataset.

Before noise filtering(signal from the second data sheet)-

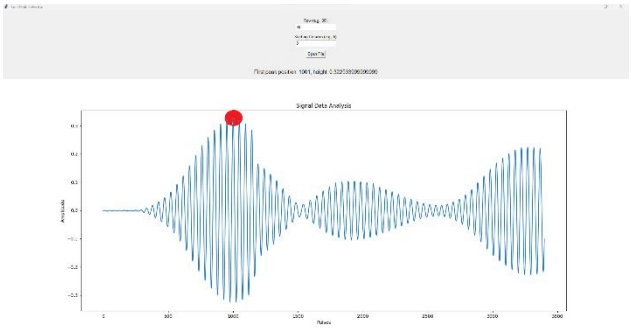

Figure. 9. Signal generated from second Dataset

And for providing small data from second dataset-

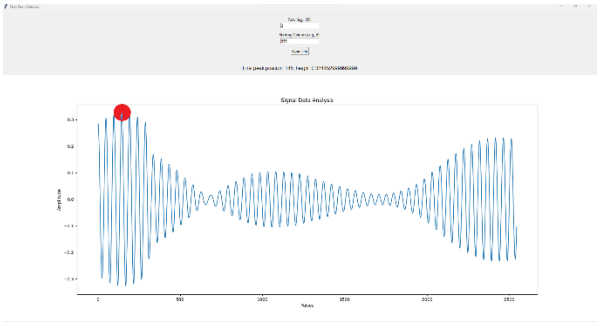

Figure 10: Filtered Signal of second Dataset

2. Machine learning solution:

In our machine learning solution, when we trained the data using "sklearn.Linear_Model.linearRegression()," we observed a Mean Absolute Error (MAE) of 1385.3128. However, when we opted for the SVR() model from Support Vector Machine and trained the data, the resulting Mean Absolute Error (MAE) decreased significantly to 954.1279. This indicates that the SVR() model performed better in terms of minimizing the absolute errors when compared to the linear regression model.

Through the use of grid search with cross-validation, we were able to determine the optimal configuration for our model, specifically the SVR() model from Support Vector Machine. This approach involved systematically exploring various parameter combinations within a defined range to find the settings that resulted in the best performance for our particular task. The selected parameters were those that led to the lowest mean absolute error, providing us with valuable insight into when the output was at its best and indicating the most effective model configuration for our project.

E. Problems encountered and solutions:

Initially, we attempted to develop a custom algorithm tailored to our project's needs. However, we encountered challenges in achieving the desired outcome with this approach. Subsequently, we transitioned to utilizing Python's default libraries to streamline our development process.

As we delved into noise filtering, we initially faced issues due to a coding error - inadvertently selecting the wrong function. To address this, we introduced the Hamming window and attempted cross-correlation with the signal. Unfortunately, this approach proved ineffective. We pivoted to leveraging the numpy.convolve() function to perform convolution between the signals and the windows. Later on, we skipped this because of the increased execution time and for simplicity and effectiveness in signal processing.

In the context of machine learning, we encountered a hurdle due to the absence of labels in the datasheets. To overcome this, we manually identified the desired outcomes and created a secondary labeled datasheet. This approach enabled us to proceed with a more intelligent and structured approach to machine learning.

IV. Discussions & Conclusion

In this paper, we have used the methodology for identifying the initial peak in a noisy periodic signal using both signal processing and machine learning techniques. We have outlined two distinct approaches, each with its respective limitations when applied to simulated or real-time data.

In the signal processing approach, we leveraged the scipy package and its functions to pinpoint the foremost and precise peak, returning the index positions of local maxima in a periodic signal. However, despite successfully identifying the first peak of the noisy signal, its robustness was compromised. Achieving robustness in peak detection necessitated incorporating a multitude of conditions to accommodate signal variations. Unfortunately, this introduced a trade-off, diminishing the efficacy of handling both high and low-frequency signals. Consequently, this represents a limitation inherent to signal processing methodologies.

On the other hand, in the machine learning approach, we initiated with data pre-processing to mitigate the impact of noisy signals. This involved incorporating the first peak value obtained from the signal processing approach into a designated Class label. Subsequently, we proceeded to data labeling, followed by training and processing employing Linear Regression and SVM algorithms. Notably, during training on extensive signal datasets, the detected or predicted

peak closely aligned with the first peak, yet did not precisely match the position encountered in the signal processing approach.

In summary, our findings reveal that while utilizing a signal processing algorithm, the initial peak can be detected accurately but may lack precision across diverse signal types. Conversely, in the machine learning approach, the first peak can be identified across various data types, albeit with potential discrepancies in positioning, sometimes not aligning precisely with the actual first peak.

ACKNOWLEDGMENT

We would like to take this moment to extend our sincere appreciation to our instructor, Dr. Andreas Pech, for his invaluable support, guidance, and encouragement throughout the duration of this project. His expertise and mentorship have been instrumental in enabling us to undertake and successfully complete this endeavor. Without his unwavering assistance, this project would not have been possible.

Furthermore, we wish to gratefully acknowledge that the research and findings presented in this report are the result of our collective efforts as a group and the individual contributions of each author. While we received guidance and support from our subject-matter teachers, the content and conclusions of this work are entirely attributed to the authors.

We also acknowledge the extensive use of open-source platforms and the wealth of information available on the internet, which greatly aided in our research and study. In an earnest attempt to uphold academic integrity, we have made diligent efforts to provide appropriate citations for relevant literature and research articles referenced in this report.

REFERENCES

[1] Felix Scholkmann, Jens Boss, Martin Wolf.; "An Efficient Algorithm for Automatic Detection in Noisy Periodic and Quasi-Periodic signal.", Algorithms, 2012, 5, www.mdpi.com/journal/algorithms

[2] Benitez, D.; Gaydecki, P.A.; Zaidi, A.; Fitzpatrick, A.P. "The use of the Hilbert transform in ECG signal analysis." Comput. Biol. Med. 2001, 31, 399–406.

[3] Tzallas, A.T.; Oikonomou, V.P.; Fotiadis, D.I. "Epileptic spike detection using a Kalman filter based approach." In Proceedings of the 28th IEEE EMBS Annual International Conference, New York, NY, USA, 2006; pp. 501–504.

[4] Singh, O.; Sunkaria, R.K. "A robust R-peak detection algorithm using wavelet packets." Int. J. Comput. Appl. 2011, 36, 37–43.

[5] Rabbani, H.; Mahjoob, M.P.; Farahabadi, E.; Farahabadi, "A. R peak detection in electrocardiogram signal based on an optimal combination of wavelet transform, Hilbert transform, and adaptive thresholding." J. Med. Signals Sens. 2011, 1, 91–98.

[6] Sun, Y.; Suppappola, S.; Wrublewski, T.A. "Microcontroller-Based real-time QRS detection." Biomed. Instrum. & Technol. 1992, 26, 477–484.

[7] 24. Ferdi, Y.; Herbeuval, J.P.; Charaf, A.; Boucheham, B. "R wave detection using fractional digital differentiation." ITBM-RBM 2003, 24, 273–280.

[8] 25. Aboy, M.; McNames, J.; Thong, T.; Tsunami, D.; Ellenby, M.S.; Goldstein, B. "An automatic beat detection algorithm for pressure signals." IEEE Trans. Biomed. Eng. 2005, 52, 1662–1670.

[9] 26. Shim, B.; Min, H.; Yoon, S. "Nonlinear preprocessing method for detecting peaks from gas chromatograms." BMC Bioinf. 2009, doi:10.1186/1471-2105-10-378.

[10] Mehta, S.S.; Sheta, D.A.; Lingayat, N.S.; Chouhan, V.S. "K-Means algorithm for the detection and delineation of QRS-complexes in electrocardiogram." IRBM 2010, 31, 48–54.

[11] Nguyen, N.; Huang, H.; Oraintara, S.; Vo, "A. Peak detection in mass spectrometry by Gabor filters and envelope analysis." J. Bioinf. Comput. Biol. 2009, 7, 547–569.

[12] Fredriksson, M.J.; Petersson, P.; Axelsson, B.O.; Bylund, D. "An automatic peak finding method for LC-MS data using Gaussian second derivative filtering." J. Seperation Sci. 2009, 32, 3906–3918.

[13] Sezan, M.I. "A peak detection algorithm and its application to histogram-based image data reduction." Comput Vis. Graph. Image Proc. 1990, 49, 36–51.

[14] Lin, K.-P. "QRS feature extraction using linear prediction." IEEE Trans. Biomed. Eng. 1989, 36, 1050–1055.

[15] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

[16] https://docs.python.org/3/library/tkinter.html

TEAM MEMBER	TASK PERFORMED
Syed Mostain Ahmed	CODE: ● First Peak detection ● Machine learning (trying different approaches and finding best possible outcome) ● GUI design
	REPORT: ● Implementation (Signal Processing and machine learning approach) ● Results & Output Section ● Conclusion & Discussion
Farjana Akter	CODE: ● Data labeling ● Machine learning(SVM) ● Handling errors in code
	REPORT ● Abstract & Conclusion ● Project Implementation Section ● SE Workflow
Al Rahat Hossain	CODE: ● GUI check ● Planning outcome
	REPORT: ● Environment setup ● Problems & solution ● Presentation PPT ● Background studies