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

Course Title

M.Eng Information Technology
Computational Intelligence

Submitted by

Md Maruf Hossain
***ID* 1390272**
MARUF.HOSSAIN@stud.fra-uas.de

Khan Mushfiqur Rahman
ID 1347995
khan.rahman@stud.fra-uas.de

MD FATHIR AHMED SHISHIR
ID 1344477
md.shishir@stud.fra-uas.de

Akash saha
ID 1345829
akash.saha@stud.fra-uas.de

Abstract

Many computational problems in engineering and data science fields rely on the ability to detect and analyze peaks in a time-series noisy periodic signal which is useful in a variety of applications. Identifying peaks can be extremely beneficial to simply comprehend sudden increases or decreases in any kind of data. In this paper, we propose our deep research using two approaches; firstly, using signal processing, we find or detect the first peak of a signal with the help of libraries. Secondly, by using Machine Learning (ML) approaches such as Linear Regression and Support Vector Machine (SVM) to train, predict and detect the peaks of the signal. The effectiveness and comparison of these two algorithms are shown in our research.

Keywords— peak detection, local maxima and minima, Machine Learning, Linear Regression, Support Vector Machine

Introduction

The detection and analysis of peaks in time-series data play a pivotal role across various scientific and engineering domains. These peaks often signify critical events or transitions within the data, making their identification crucial for applications such as anomaly detection, quality control, and pattern recognition. In this context, this research endeavors to explore effective methods for peak detection in noisy time-series signals, combining signal processing and machine learning techniques.

The primary objective of this study is to develop a comprehensive approach to peak detection that harnesses the strengths of both traditional signal processing and modern machine learning methodologies. This fusion of techniques enables us to not only identify peaks accurately but also discern valuable insights from complex and noisy data.

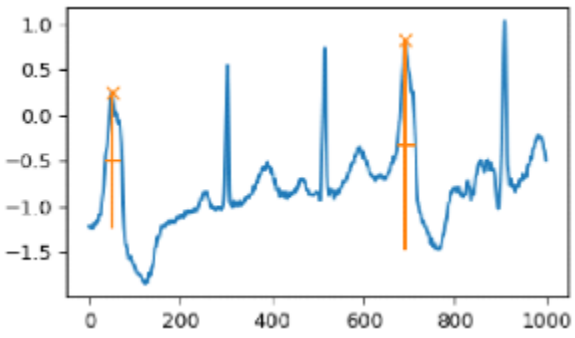
To achieve this goal, we employ well-established libraries for signal processing, specifically leveraging the find_peaks function from the scipy library. Additionally, we delve into the realm of machine learning, utilizing algorithms such as Linear Regression to predict and detect peaks within the signal. The integration of these methods provides a holistic view of peak detection, facilitating a comparative analysis of their effectiveness.

This paper not only demonstrates the practicality of the proposed approach but also highlights the potential for significant advancements in peak detection methods, which can find applications in diverse fields, including data analysis, signal processing, and industrial quality control.

Project Implementation

Background Studies

In many different disciplines, from engineering to finance, it is essential to be able to recognize and use signal peaks. Any functional value's maxima and minima locations are defined by or determined by its peaks. Realizing the position and values of any statistical values is crucial in order to make it simple to forecast the outcomes of any given information.



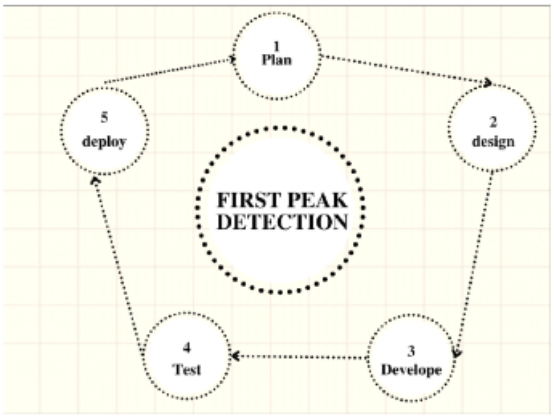
In this function, a sample has two primary neighbors with lesser amplitudes if it is considered to be a peak or local maximum. The middle sample's index is given when there are many samples with equal amplitude width. Even though noise might alter the position of local maxima, peak locations may vary in noisy signals. Prior to looking for peaks in the signal, think about flattening it. In such circumstances, consider softening the signal before peak hunting, or use alternate peak finding and fitting techniques.

The first thing we do is build a function that can predict the peaks starting with the test dataset that is provided. We conducted our investigation in the following steps, which are detailed below, for that process.

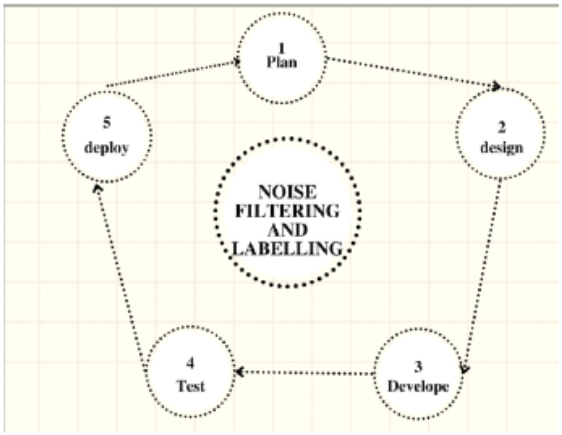
Software Engineering Workflow

Here, we tried to work with an iterative and agile software development process. It happened in three cycles.

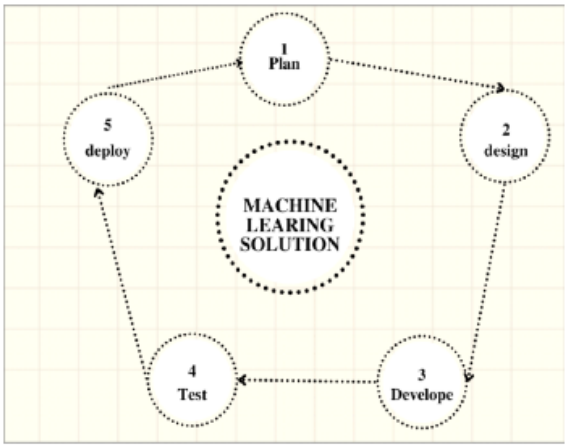
First, for signal processing for first peak detection.



Second, for noise filtering and labeling.



Third, for machine learning approaches.



Environment Setup

The first step in starting our experiment is to build and execute the project on our PC. We prepare the surroundings. We decided to design our project using the Python scripting language in order to build an algorithm that can produce the desired results. Due to its rich technology, Python can offer a wide range of computational intelligence libraries. Numerous programming paradigms, such as object-oriented and functional programming, are compatible with Python.

First, Python 3.8 or a more recent version must be installed on the computer. The system then needs to have the Anaconda environment or perhaps PyCharm community edition installed. The deployment and package management are made simpler by this console.

User Interface

A GUI (graphical user interface) toolkit called Tkinter or Tk is installed and used to develop graphical interaction. Python and Tk are linked. A specific command run from the prompt might display a basic Tk interface, showing that TKinter is correctly installed on your computer [16]. We imported "file dialog" modules from Tkinter, which enable users to quickly specify a file that has to be opened or uploaded.

```
python -m tkinter
```

Installed Packages or Libraries

We can begin installing the specific packages needed to run our project once the prior setup is complete. By executing those commands, the packages listed below can be installed immediately. These programs include more advanced algorithms and are more versatile to use because working with AI is more difficult. To make our model generate reliable signal peaks with noise filtration using an AI technique, we need to integrate programs 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.

We are importing the function "find_peaks" to allow it to locate the initial local maxima of the given data when visualizing the signal in a graphical framework using libraries like matplotlib. Second, after data pre-processing, the provided test datasheets are translated into readable form using the pandas package and may be accessed in excel format. For use in signal processing applications, the Scipy package is being installed.

We are use the Scikit-learn library for our machine learning strategy. It displays strategies for classification, regression, and clustering. For example, in our work, we attempted our experiment with linear regression and SVM (Support Vector Machine) models for training and testing the data.

For our ML approach, we are using the Scikit-learn library. It showcases classification, regression, and clustering algorithms, such as for our work, we tried our experiment with linear regression and SVM (Support Vector Machine) models for training and testing the data.

```
from scipy.signal import find_peaks
from matplotlib import pyplot as plt
import numpy as np
import tkinter as tk
import pandas as pd
from tkinter import filedialog
```

Python IDE

Integrated Development Environments (IDEs) are used to create Python scripts. A tool called Pycharm was used to create the project's Python script. We may install Pycharm under Anaconda as well.

A web-based or machine application tool, Pycharm is incredibly easy to use. jupyter notebook should be entered at the anaconda command prompt once it has been installed to start using it. This IDE provides numerous essential Python developer tools that are intricately entwined to provide a comfortable environment for effective Python, web, and data science research.

Implementation

Signal Processing Approach

Pre-Processing

Since the data were gathered in a loud setting, a noise removal filter is required. We tested two distinct procedures to do it. With the "Scipy.signal.butter()" and "Scipy.signal.filtfilt()" routines, the first one is accomplished. We twice ran the later function.

```
b, a = butter(3, 0.1, 'highpass')
filtered2 = filtfilt(b, a, dev_y)
filtered3 = filtfilt(b, a, filtered2)
```

Another method we attempted involved using "numpy.hannig()" to generate a Hanning Window with a size of 50 and then applying "numpy.convolve()" to both the window and the signal. The result is a noise filter. Later, in order to keep things simple, we merely used "Scipy.signal.filtfilt()".

Peak Detection

We decided to utilize the function "scipy.signal.find_peaks()" to find peaks, which locates all of the peaks in a signal. But we only seek the initial peak. The concept of "peak" in this function also differs significantly from the definition in our problem. As a result, we performed a minimal amount of processing to these peaks in order to identify the first peak.

```
indexes, _ = find_peaks(filtered3, distance=1000)
        for vals in range(len(indexes)):
            peaks.append(filtered3[indexes[vals]])
            print(peaks)

        mean_indexes = np.average(peaks)
        print(mean_indexes)

        for ind in range(0, len(indexes)-1):
           if (ind == 0) & (filtered3[indexes[ind]] >
filtered3[indexes[ind+1]]):
                first_peak = filtered3[indexes[ind]]
                location = indexes[ind]
            elif (ind > 0) & (filtered3[indexes[ind]] > mean_indexes) & \
                    (filtered3[indexes[ind]] > filtered3[indexes[ind+1]]) &
(filtered3[indexes[ind]] > filtered3[indexes[ind-1]]):
                first_peak = filtered3[indexes[ind]]
                location = indexes[ind]
        print(first_peak)
        print(location)
```

We created an array of values using the indices that the "find_peaks" function returned because that is what we did. And from there, we calculated an average. Then, all of the peaks that are higher and larger than the peaks before and after are selected. These peaks are what our solution wants to have. The value was set to "First Peak" since we were in need of the first peak.

Machine Learning Approach

Pre-Processing

Data sheets are not processed for machine learning when they are provided, so we must label the data in order to complete the task. Following the required peak's labeling, we added the peak value to the datasheet as a "Class" label. We categorized the output (y) as the signal (X) and treated it as data.

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

new_dataframe = pd.DataFrame(new_list)

new_dataframe.to_excel("Wand_000_changed.xlsx")
```

Predicting Peaks

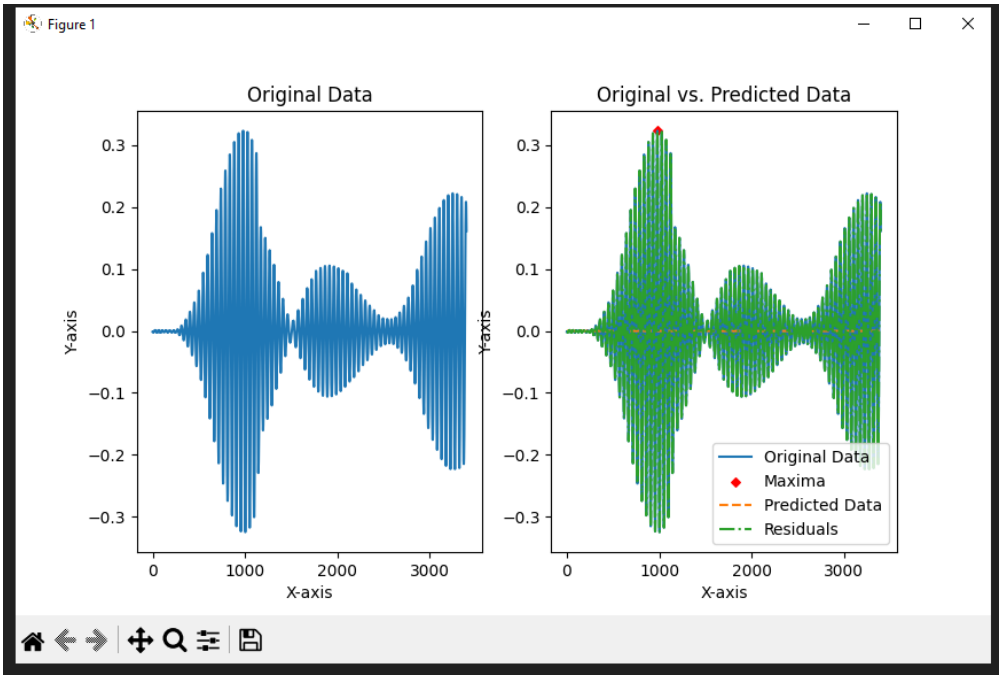
We trained and processed the data using the well-known Python machine learning toolkit "Scikit_learn". Regression techniques were employed to train and test since we needed to obtain continuous peak values. We used "Sklearn.linearregression" and "sklearn.SVM" specifically. Following a comparison of the error scores, we decided to use "SVR" from "sklearn.SVM". In addition, "GridSearchCV()" was utilized to find the ideal parameter choice.

```
y_train = train.Class  # if y is only one column
train.drop(['Class'], axis=1, inplace=True)
X_train = train

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

DT = SVR()
cross_val = GridSearchCV(estimator=DT,
                         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("The first five prediction {}".format(pred[525:]))
print("The real first five labels {}".format(y_test[525:]))

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

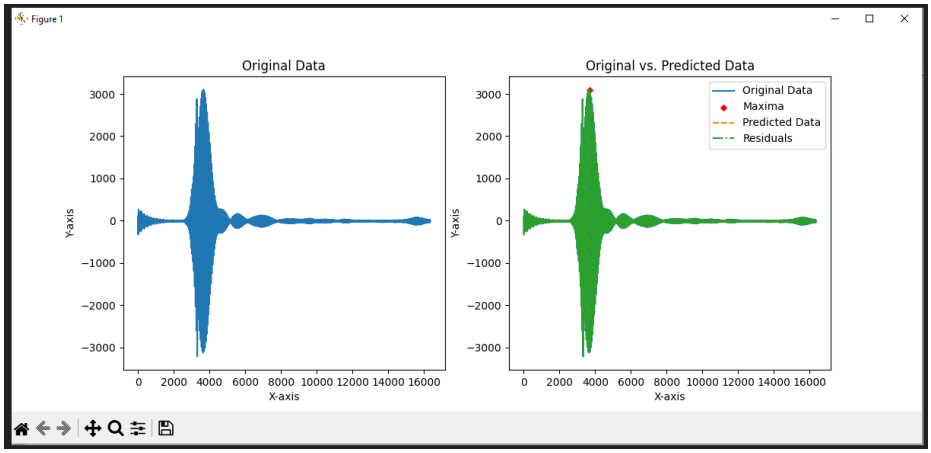
Results

1. Generic signal processing:

From the datasheet, we get 50 signals. Before noise filtering, it looked like the below figure.



After noise filtering, the output signals are generated as shown below.



Here, the red dot on the signal is the first calculated peak. For test data two, these are like,

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

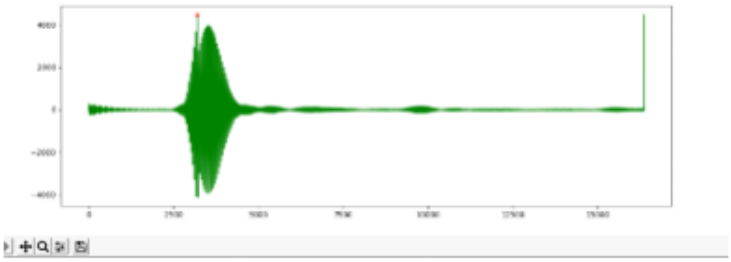


Fig. 4. Noisy Signal generated from second Dataset

After noise filtering-

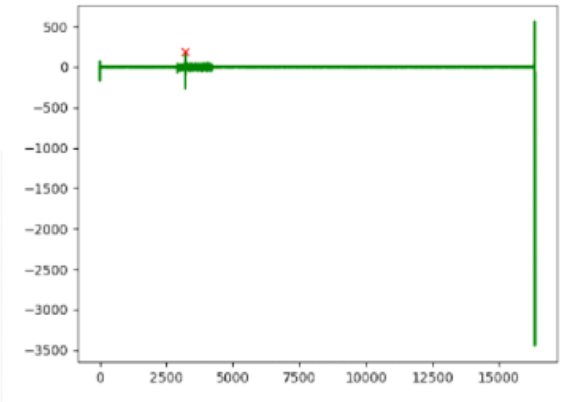


Fig. Filtered Signal of second Dataset

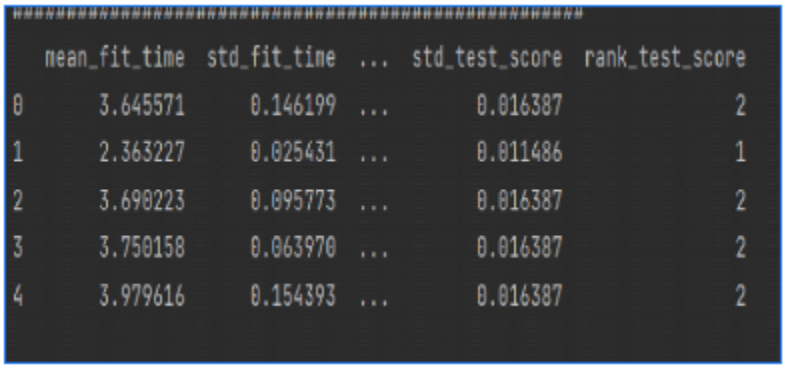
The green line at the end is for added 0.00. It is added while testing with “SVR()”.

2. Machine learning solution:

In machine learning solution, when data is trained with “sklearn.Linear_Model.linearRegression()” it shows Mean_absolute_error 1675.0146

But when we used model SVR() from Support Vector Machine it showed “MEAN-ABSOLUTE_ERROR - 918.2239”

```
The last five prediction
[3508.73619569 3479.73645961 3495.67847291 3505.65778555 3482.81882914]
The real last five labels
166    4948
365    2727
176    3614
179    4936
180    3125
Name: Class, dtype: int64
918.2239334974915
##################################################################
```



```
##################################################
   mean_fit_time  std_fit_time  ...  std_test_score  rank_test_score
0       3.645571      0.146199  ...        0.016387                2
1       2.363227      0.025431  ...        0.011486                1
2       3.698223      0.095773  ...        0.016387                2
3       3.750158      0.063970  ...        0.016387                2
4       3.979616      0.154393  ...        0.016387                2
```

Also the grid search cross validation shows when the output is best.

Problems encountered and solutions:

Initially, we attempted to develop a custom algorithm tailored to our specific needs. Unfortunately, this approach did not yield the desired results. Consequently, we transitioned to utilizing Python's default libraries.During the initial stages of noise filtering, we encountered issues due to coding errors, including the inadvertent selection of an incorrect function. To address these challenges, we introduced a Hamming window, followed by attempting cross-correlation with the signal. However, this approach did not produce the expected results. To overcome this, we turned to the NumPy convolve() function for signal convolution. Later, we opted for simplicity by adopting the scipy.filtfilt() function.

When venturing into machine learning, we encountered a significant challenge. The absence of labeled data sheets posed difficulties in our initial attempts. Ultimately, we resolved this issue by manually identifying the desired outcomes and obtaining a second labeled datasheet, enabling us to pursue a more intelligent approach.

Discussions & Conclusion

This paper offers a comprehensive exploration of the process of identifying the initial peak within a noisy periodic signal, employing both signal processing and machine learning methodologies. We present two distinct approaches, each with inherent limitations when applied to simulated or real-time data.

In the signal processing method, we initiate data preprocessing by applying two noise-filtering functions. Subsequently, we employ additional packages and functions to pinpoint the first peak of the signal, yielding index positions for local maxima in the periodic signal. Although we successfully identify the initial peak in the noisy signal, its robustness is lacking. Achieving robust peak detection requires the inclusion of numerous conditions for various signal variations, which can negatively impact the performance of both high and low-frequency signals. This limitation is an inherent drawback of signal processing techniques.

In the machine learning approach, we once again commence with data preprocessing to eliminate noise signals. We then incorporate the first peak value obtained from the signal processing approach as a designated class label. After preparing the labeled data, we proceed with training and processing, employing Linear Regression and SVM algorithms. Our findings indicate that while training on large signal datasets, the detected or predicted peak closely approximates the initial peak but may not precisely match the one identified in the signal processing approach.

In summary, our research reveals that signal processing algorithms excel in accurately pinpointing the initial peak, but this precision may vary with different signal types. Conversely, the machine learning approach offers versatility in identifying the first peak across various data types, although the identified peak may not align precisely with the actual initial peak.

References

- 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
- 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.
- 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.
- Singh, O.; Sunkaria, R.K. "A robust R-peak detection algorithm using wavelet packets." Int. J. Comput. Appl. 2011, 36, 37–43.
- 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.
- Sun, Y.; Suppappola, S.; Wrublewski, T.A. "Microcontroller-Based real-time QRS detection." Biomed. Instrum. & Technol. 1992, 26, 477–484.
- 24. Ferdi, Y.; Herbeuval, J.P.; Charaf, A.; Boucheham, B. "R wave detection using fractional digital differentiation." ITBM-RBM 2003, 24, 273–280.
- 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.
- 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.
- 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.
- 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.
- 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.
- 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.
- Lin, K.-P. "QRS feature extraction using linear prediction." IEEE Trans. Biomed. Eng. 1989, 36, 1050–1055.
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html
- https://docs.python.org/3/library/tkinter.html

TEAM MEMBER	TASK PERFORMED
MD FATHIR AHMED SHISHIR	CODE: • First Peak detection • Machine learning(Linear Regressing)
	REPORT: • Implementation (Signal Processing and Regression approach) • Results & Output Section • Conclusion & Discussion
Md Maruf Hossain	CODE: • Data labeling • Machine learning(SVM)
	REPORT • Abstract & Conclusion • Project Implementation Section • SE Workflow
Khan Mushfiqur Rahman & Akash saha	CODE: • GUI Creation • Noise filtering • Plotting the output REPORT: • Environment setup • Problems & solution • Presentation PPT • Background studies