EE220 PROJECT

Analysis of ECG Signals and detection of Disease

BY

220121065 – **Srepadmashiny K**

From the department of **Engineering Physics**

Under the supervision of Professor

Dr. Manish Bhatt

****

***Acknowledgement***

We wish to express our heartfelt gratitude to the Electronics and Electrical Engineering Department and our EE220 Signals and Systems professor Dr. Manish Bhatt Sir for their unwavering guidance , support throughout the duration of the course and also for providing us the opportunity to do this project. Their constant motivation played a pivotal role in our successful completion of the project.

I would like to express our sincere thanks to our professor, Dr. Manish Bhatt for his encouragement and support to work on this o complete this project

This page is intentionally left blank

Table of Contents

[Acknowledgement 3](#_Toc82987787)

[1. Introduction 5](#_Toc82987788)

[2.Project Description](#_Toc82987789) [5](#_Toc82987790)

3. Signals Overview…………………………………………………………………………………………………………….15

4. Systems Overview………………………………………………………………………………………………………….20

ECG Signal: Analyzing Cardiac Activity through Signals and Systems…………24

[4. Libraries Used 28](#_Toc82987794)

[5. Variables Used 30](#_Toc82987795)

[6. Source Code 3](#_Toc82987796)3

Analysis of ECG Signals and detection of disease

***1.Introduction:***

Cardiovascular diseases (CVDs) stand as the leading global cause of mortality, responsible for approximately 17.9 million deaths, constituting 31% of all worldwide fatalities. The majority of CVD-related deaths, about four out of five, result from heart attacks and strokes, with one-third of these occurring prematurely in individuals under the age of 70. An electrocardiogram (ECG) can continuously monitor a patient's cardiac electrical activities by recording voltage variations through electrodes placed on the chest, arms, and legs. ECGs offer a rapid, safe, and painless means to assess heart rate, rhythm, and potential signs of cardiac issues.

Presently, a twelve-lead ECG is the standard tool employed by cardiologists to identify various cardiovascular irregularities. Nevertheless, conventional 10-second recordings in hospitals or clinics may not always detect heart problems comprehensively. To address this limitation, innovative sensing technologies have made long-term ECG monitoring feasible, ensuring continuous heart condition tracking in various circumstances. Portable ECG recording devices like the Apple Watch, AliveCor, Omron HeartScan, QardioMD, and more recently, the Astroskin Smart Shirt, have revolutionized cardiac diagnostics. They capture a patient's cardiac activities around the clock and transmit this data to remote cloud services for storage and analysis.

However, the extensive data generated by this continuous monitoring is challenging for the medical community to handle effectively, given the limited time and resources available to review lengthy ECG recordings spanning two to three weeks. To make this technology truly beneficial, it is imperative to develop new, automated, and dependable algorithms for detecting cardiac anomalies. These algorithms are essential for aiding medical professionals in managing and interpreting this substantial dataset.

The core objective of this project is to proactively deduce diagnostic steps well in advance, ultimately enhancing human health and well-being.

***2.Project Description:***

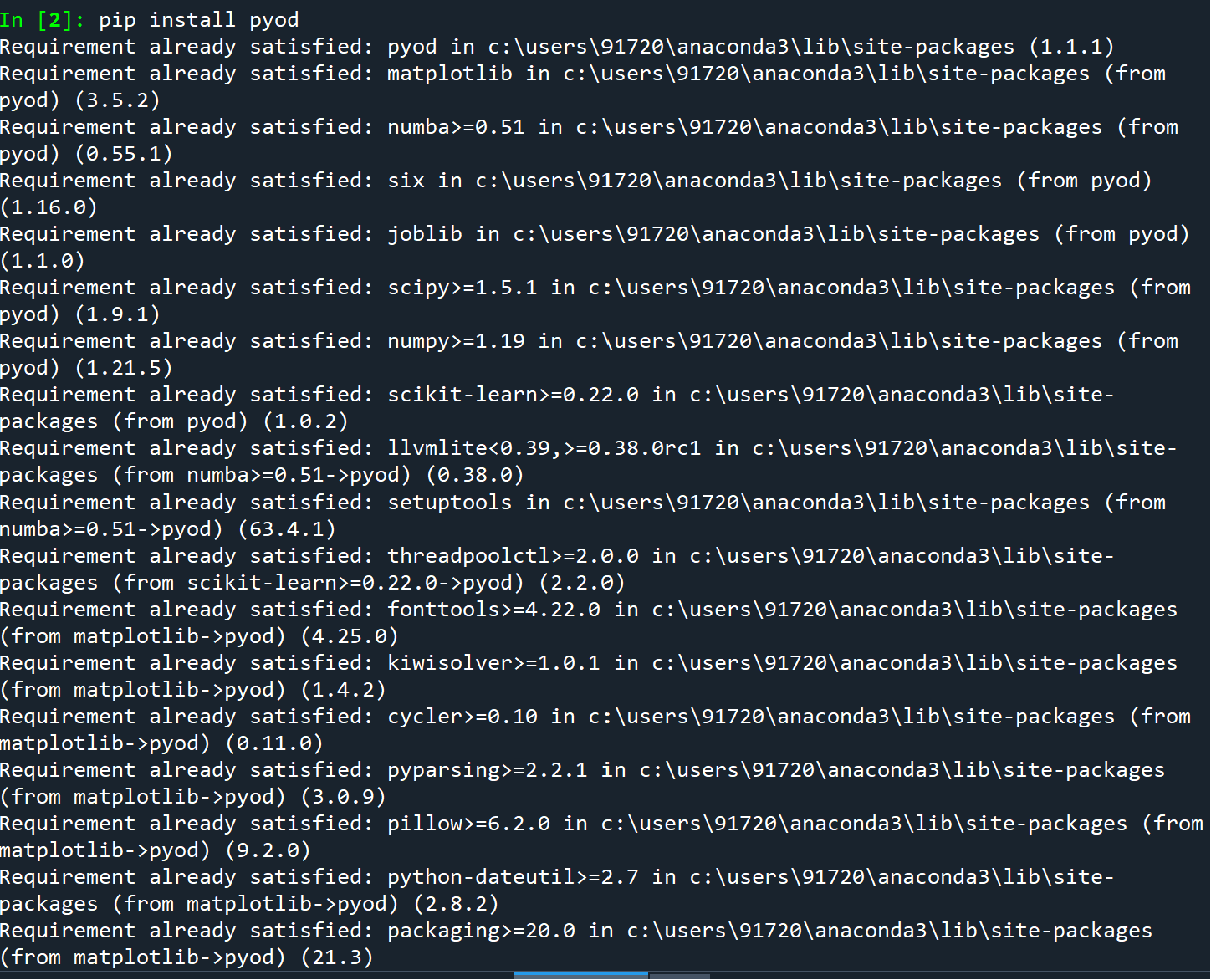
In our project we have utilized multivariate anomaly detection using the histogram based outlier detection. The terms of significance have been explained below:

**Anomaly Detection :** Anomaly detection is a technique that identifies abnormal patterns or outliers within a dataset. In the context of this project, it is used to detect irregularities in ECG (Electrocardiogram) signals, which can indicate potential health issues.

**Multivariate Anomaly Detection :** Multivariate anomaly detection is a method that assesses anomalies by considering multiple variables simultaneously, making it suitable for complex datasets with interconnected features. In this project, it has been applied to ECG signals for enhanced accuracy.

**Python PyOD Library :** Python PyOD is a relatively new Python library introduced approximately three years ago, specifically designed for anomaly detection tasks. It offers a wide range of techniques and algorithms to identify outliers in various data domains.

This project makes use of Python PyOD to implement multivariate anomaly detection in ECG signals. One notable advantage of PyOD is its versatility, making it a valuable resource for researchers and developers in the anomaly detection field.

**PyOD Installation:** ****

**Histogram-Based Outlier Detection:** The project employs histogram-based outlier detection as a core technique. This method is favored for its effectiveness in identifying anomalies in ECG data. By utilizing histograms, it can capture irregular patterns and fluctuations within the signals, contributing to the accuracy of anomaly detection.

The primary goal of this project is to not only detect anomalies in ECG signals but also predict the type of disease or health condition associated with these anomalies. The presence or absence of one anomaly is considered independently from other anomalies, ensuring that all potential issues are thoroughly assessed. This project holds promise in improving healthcare diagnostics by automating the process of detecting abnormalities in ECG signals and providing valuable insights into potential health conditions.

***3.Data Description:***

1. ***Data Source:*** The ECG signal data used in this project was obtained from Kaggle, a popular platform for datasets and data science resources.

2. ***Data Size:*** The dataset comprises more than 4900 records, each containing over 140 variables. These variables represent various aspects of the ECG signals.

3. ***Data Split:***To develop and evaluate the models, 70% of the dataset was used for training purposes, ensuring a substantial portion of the data was dedicated to model development.

***Implementation:***

The project involves the following key components and implementation steps:

***1. Models:***

- ***Trainer Model:*** A Trainer model has been created, which is responsible for training the anomaly detection model. In this case, a Histogram-based outlier detection model is utilized and subsequently saved as a pickle file object.

- ***Scorer Model (using Kafka Consumer):*** The Scorer model is designed to assess incoming ECG signals in real-time. It uses Kafka Consumer to consume data from a Kafka topic and determine whether the signals are anomalies.

***2. Kafka Setup:***

- Kafka, a distributed streaming platform, is set up to simulate the behavior of an ECG machine.

- **A Kafka Producer** is run in the command prompt to feed ECG data into the Kafka system. The data is sent over a specific port (9092) to ensure it reaches the Kafka topic.

3. ***Scoring Program:***

- The Scoring program is responsible for evaluating the ECG signals as they are received.

- Anomalies detected by the model are labeled as '1,' indicating they are anomalies according to the Python PyOD library.

In summary, this project utilizes ECG signal data from Kaggle, employs a Histogram-based outlier detection model trained by the Trainer model, and uses Kafka for simulating the ECG machine and scoring the incoming signals in real-time. Anomalies are indicated by a label of '1' based on the Python PyOD library's output.

***Implementation of Program(Steps):***

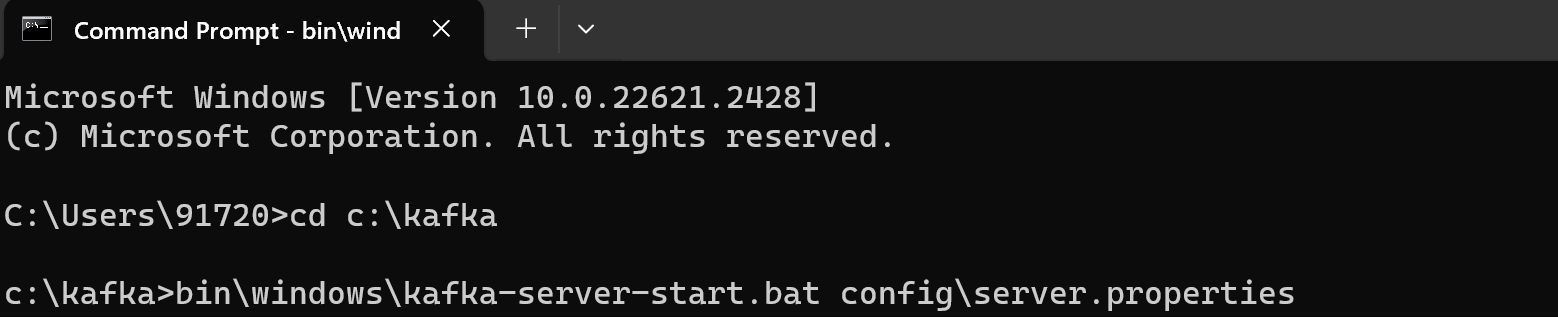
Firstly we have to ***run Zookeeper*** in command prompt



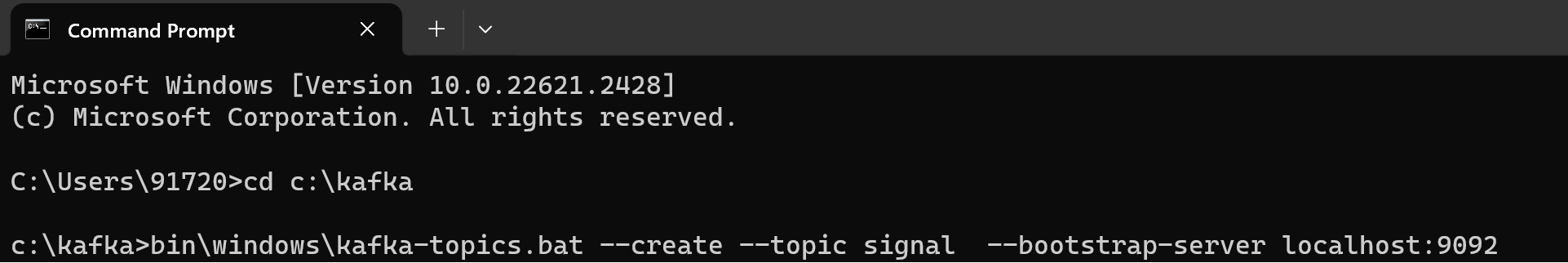
If Zookeeper is successfully run we get the above screen:

Now,

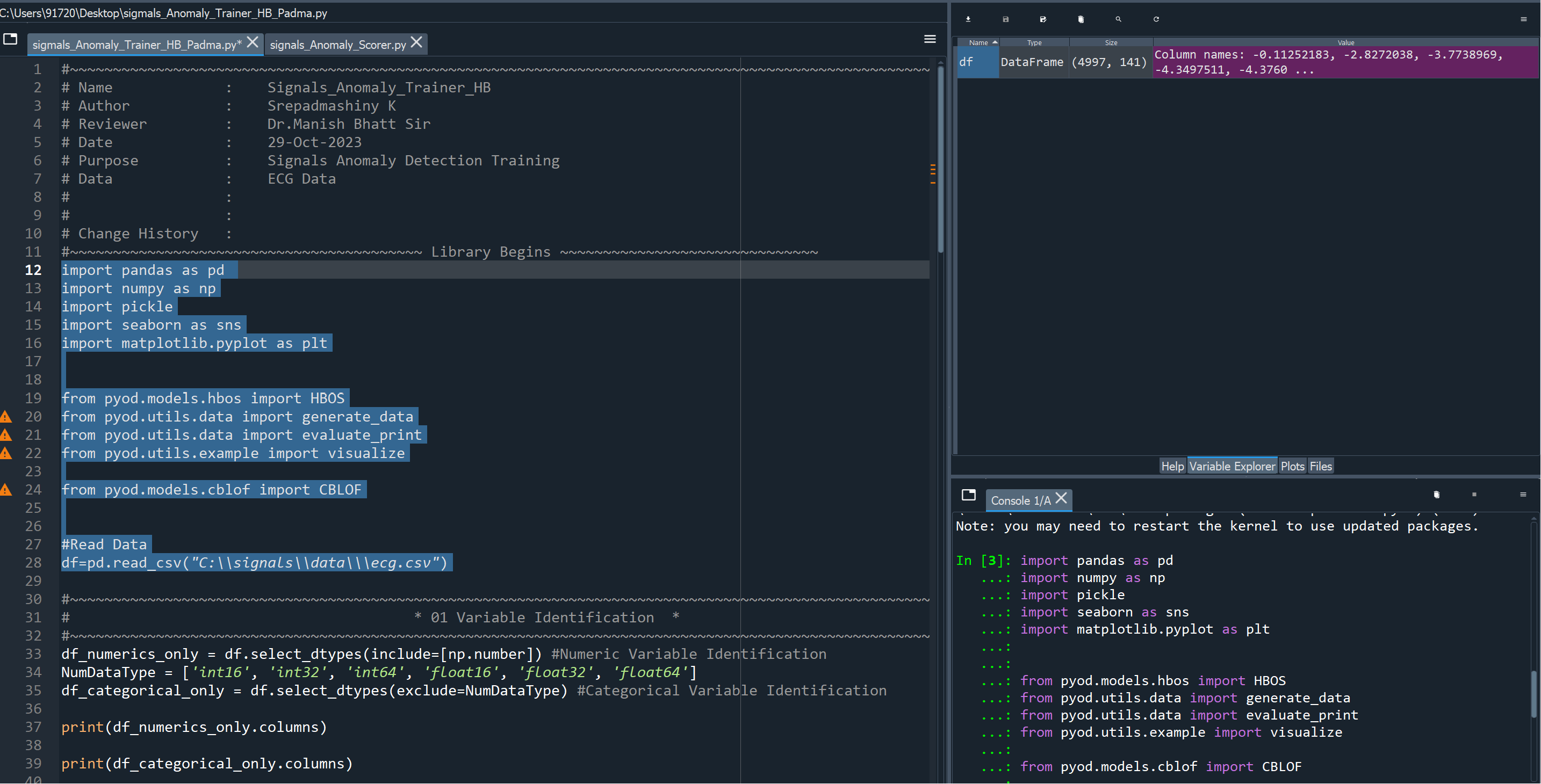
Don’t close the command prompt where zookeeper is running , open a new command prompt and type the following as shown in the screen: ***To start Kafka Broker***



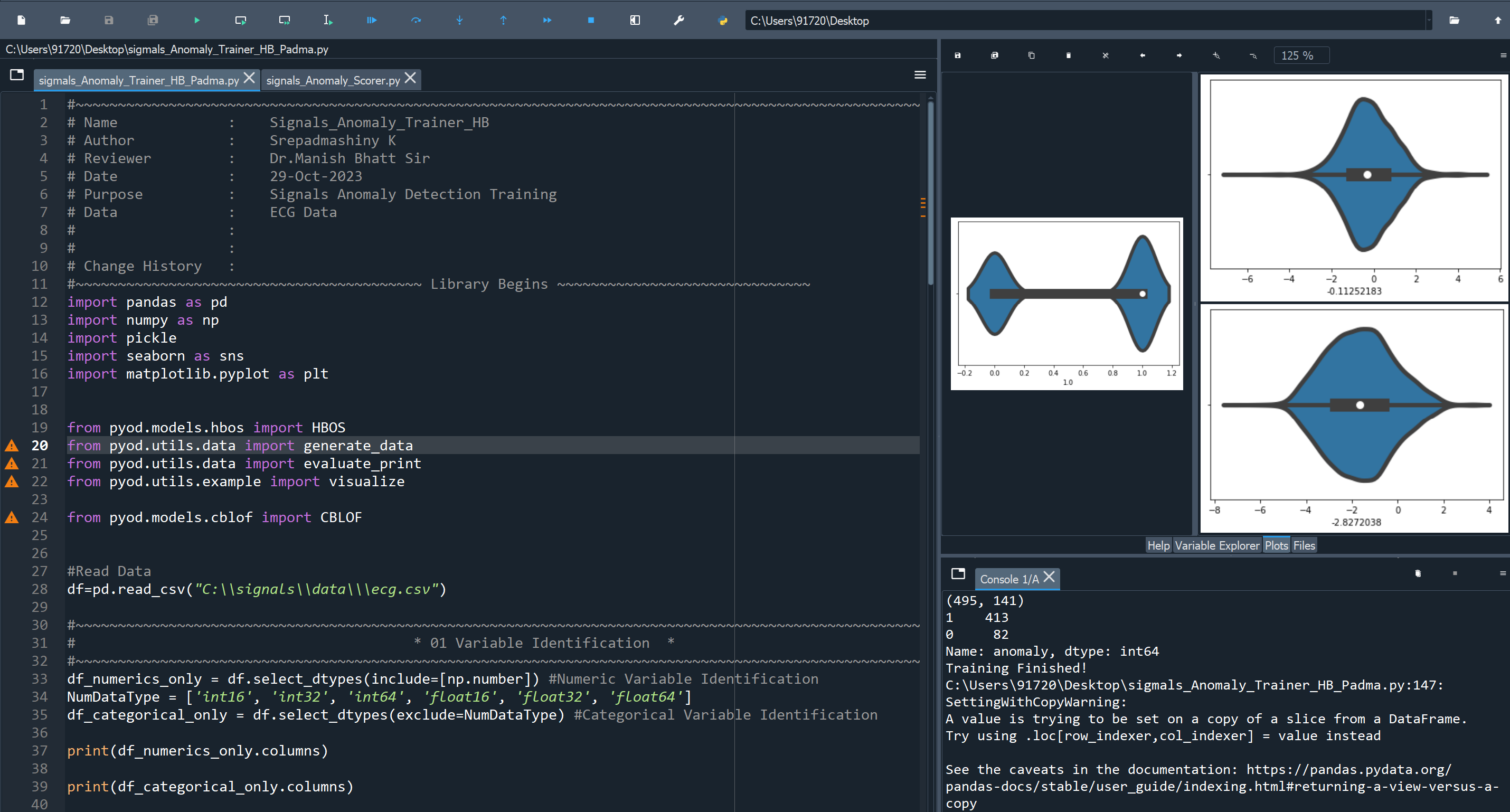
Now we are going to start a new topic called signals in a new command prompt window.



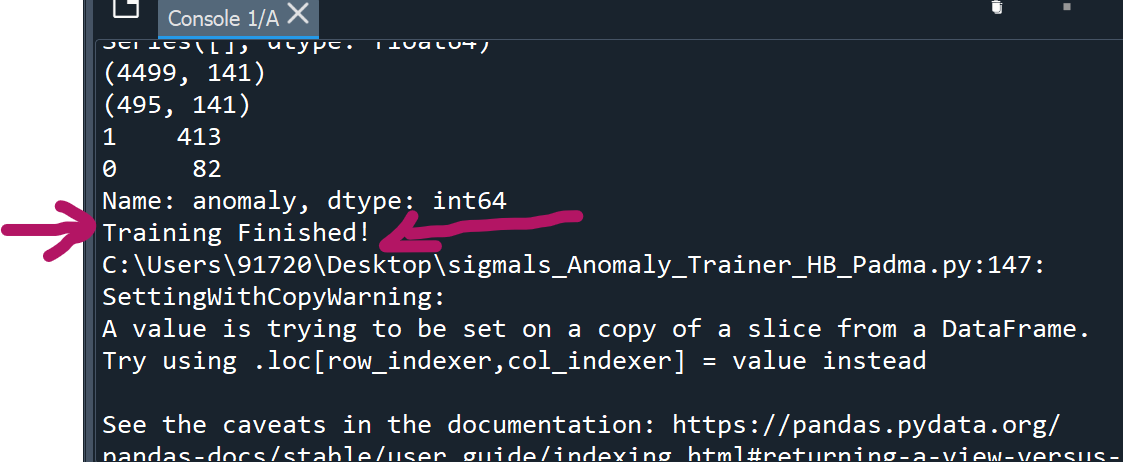
Now in the Trainer program run the selected block of text first to ensure all required libraries are installed .



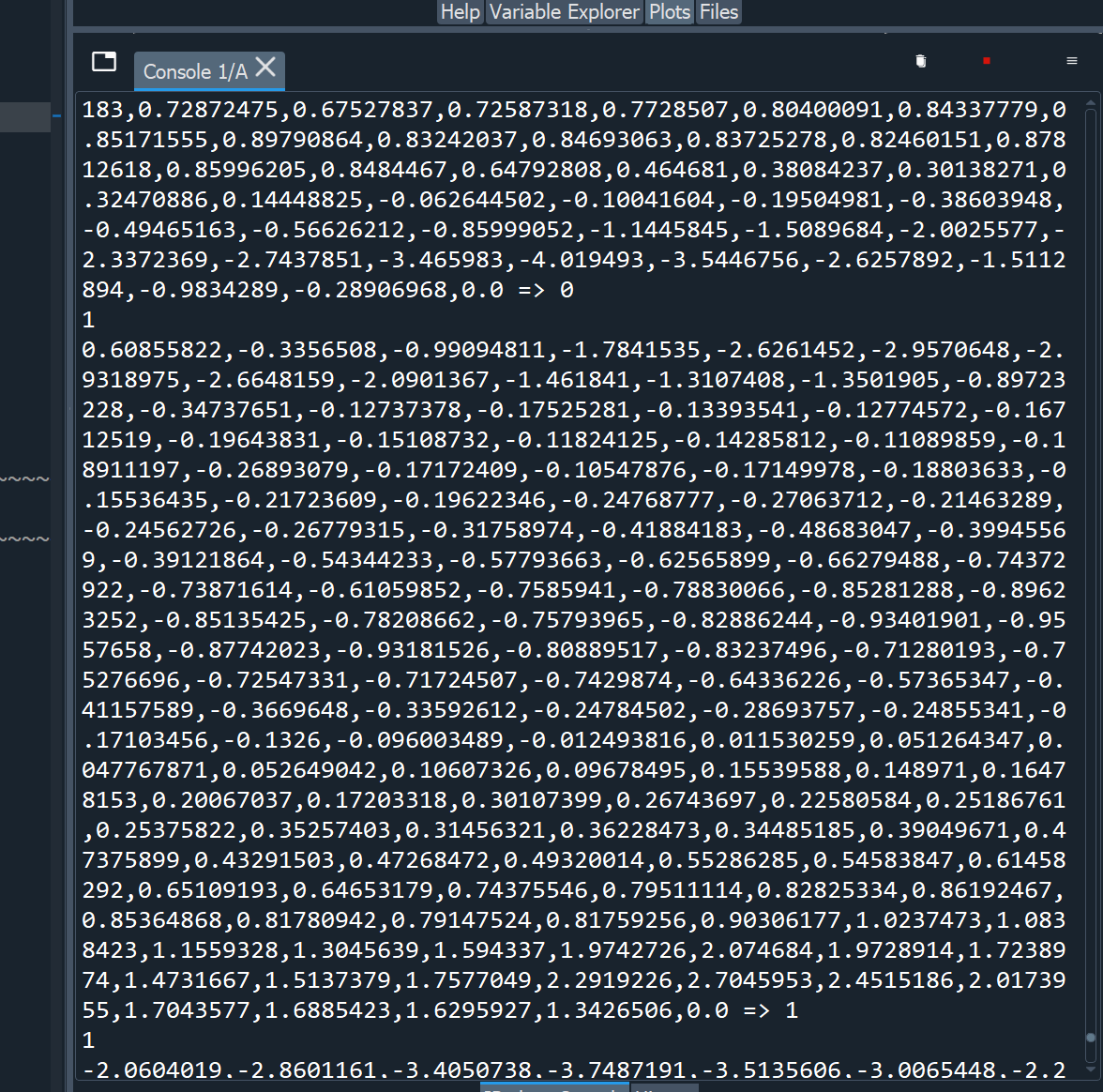
Now we have to run the Trainer program all together and we obtain a screen like this:



When we finish the training of the model we check the console panel of spyder



Next we have to run the scorer program all together and then open a new command prompt and ensure that all the existing command prompts are running:



Here 0 indicates normal distribution without ‘outliers’

Here 1 indicates the presence of anomaly, which means person has heart disease



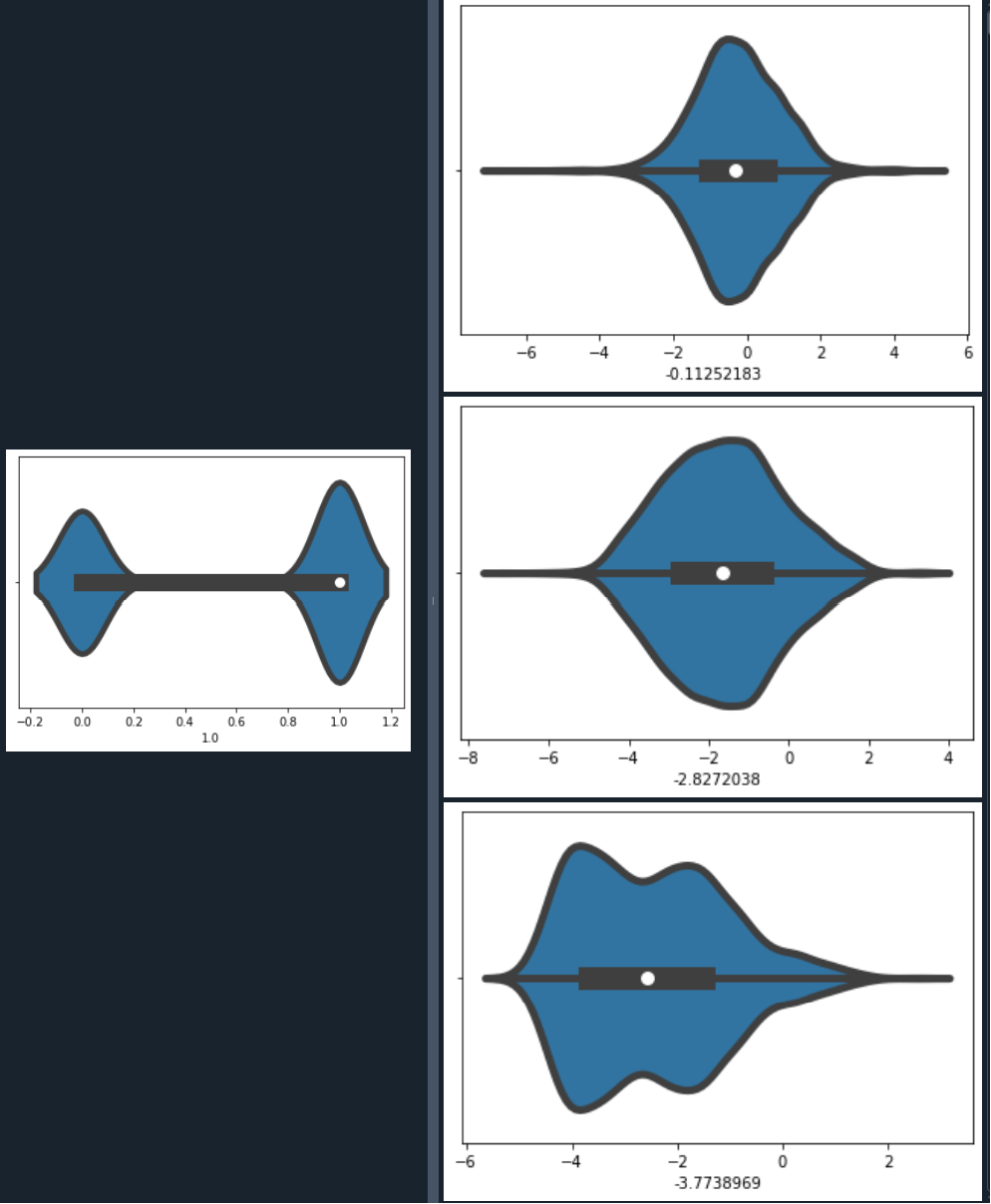
The violin plot for some of the data points have been attached below:

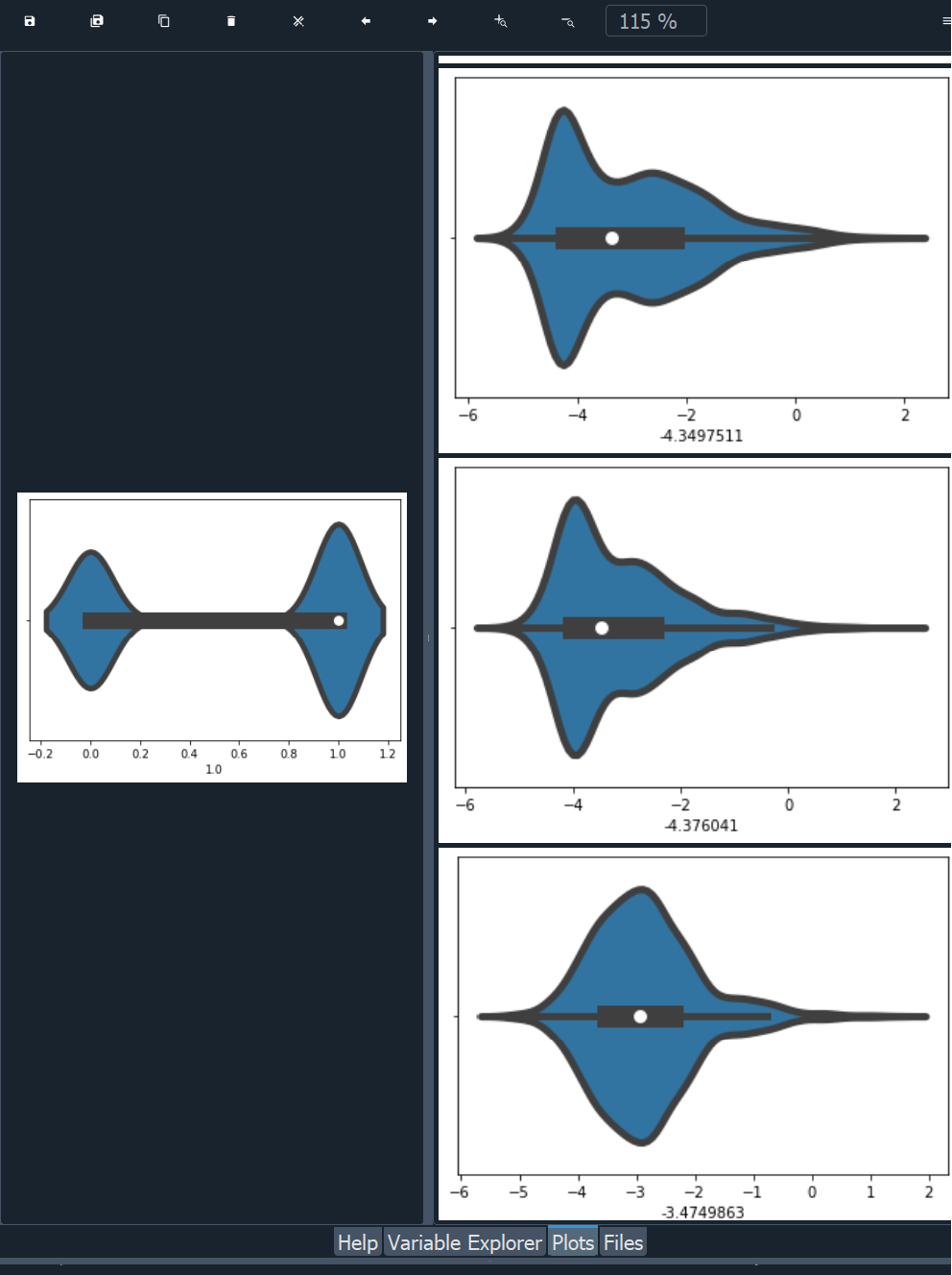
***Violin Plot:***

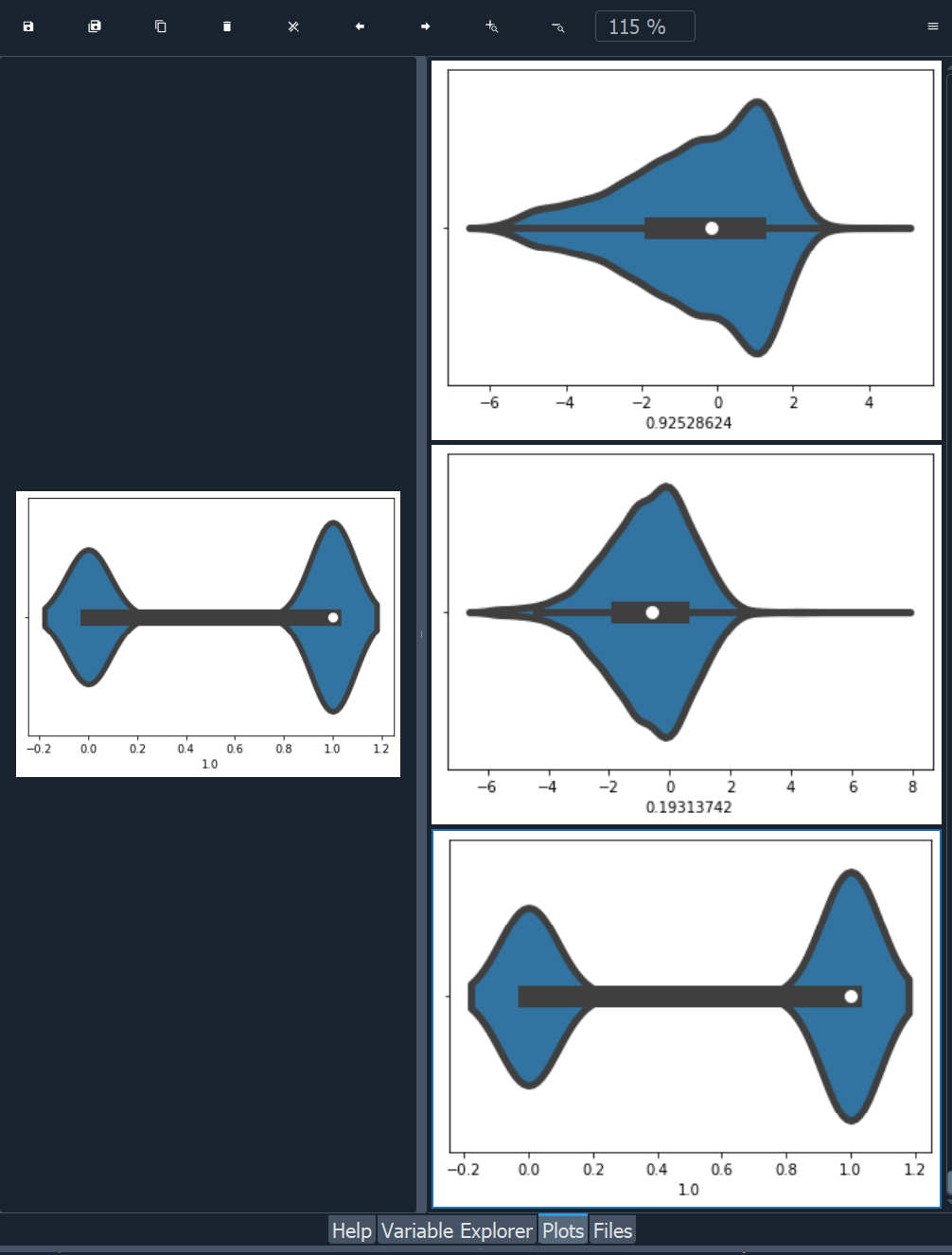
A violin plot is a data visualization that combines elements of a box plot and a kernel density plot. It is used to depict the distribution of data across different categories or groups.

A violin plot is a statistical chart that displays the probability density of numeric data at different values, providing insights into the data's distribution, central tendency, and variability. It consists of a mirrored pair of vertical or horizontal 'violins,' which represent data distributions for various categories, while the width of each 'violin' illustrates the density of data at different values. Violin plots are useful for comparing data distributions between groups or visualizing the spread of data.

Violin plots are particularly effective for understanding the shape of data distributions and identifying potential outliers, making them a valuable tool in data analysis and visualization.







For the final step open a new command prompt window and run the ECG producer and type the following code in command prompt after typing cd c:\kafka:

bin\windows\kafka-console-consumer.bat --topic signal --bootstrap-server localhost:9092 --from-beginning



This data is crucial to predict the kind of heart disease the person has.

***Overview of Signals:***

In the realm of signal processing and analysis, a signal is a fundamental concept representing variations in physical phenomena, data, or information over time or space. Signals can be categorized based on various properties, including periodicity, directionality, energy, and power. In this comprehensive overview, we delve into the key characteristics and elementary operations associated with signals.

***1.Periodicity:***

A continuous signal, denoted as x(t), is deemed periodic if it exhibits the property:

x(t) = x(t + T), for all t.

Here, T is referred to as the period of the signal, and it should be a strictly positive real constant. In discrete sequences, a similar periodicity condition is expressed as:

x[n] = x[n + N] for all n (where n is an integer).

A signal that does not possess periodicity is termed aperiodic or non-periodic.

Some examples of periodic signals:

***Frequency and Angular Frequency:***

For periodic signals, the concept of frequency becomes relevant. If the time period of a signal is denoted as T, then its frequency is 1/T, and the angular frequency is ω = 2πf = 2π/T. Similarly, for N-periodic functions, the frequency is 1/N, and the angular frequency is Ω = 2πf = 2π/N.

***Fundamental Period and Frequency:***

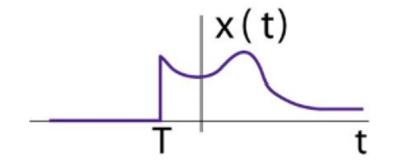
The fundamental period is the smallest period with which a signal exhibits periodicity, and its corresponding frequency is known as the fundamental frequency.

***Combining Periodic Signals:***

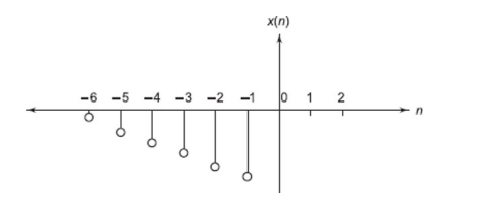
When combining two periodic functions, x1 and x2, each with fundamental periods T1 and T2, the resulting sum, y = x1 + x2, is periodic if and only if the ratio T1/T2 is a rational number (q/r). In this case, the fundamental period of y is T0 = rT1 = qT2.

***2.Right and Left Sided Signals:***

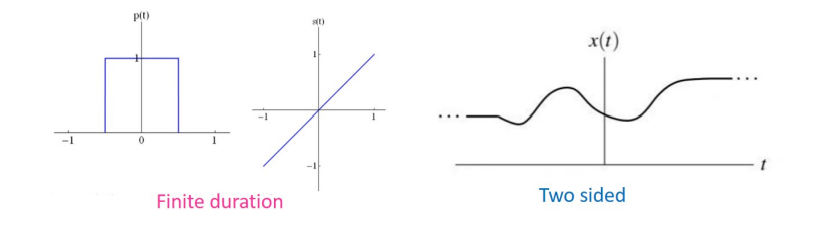
- ***Right-Sided Signals:*** A signal x is considered right-sided if there exists a finite real constant T such that x(t) equals zero for all t < T. A special case of right-sided signals is causal signals, which are only present for t ≥ 0.



- ***Left-Sided Signals:*** A signal x is left-sided if there is a finite real constant t0 for which x(t) equals zero for all t > t0. A special case of left-sided signals is anticausal signals, which have zero values for t < 0.



- ***Finite Duration and Two-Sided Signals:*** A signal that is both right-sided and left-sided is categorized as a finite duration or time-limited signal. Signals that do not fall into either the right or left-sided category are referred to as two-sided signals.



***3.Elementary Operations on the Independent Variable (t):***

- ***Time Scaling:*** Modifying the signal by a scaling factor, expressed as y(t) = x(at), where 'a' is a strictly positive real number.

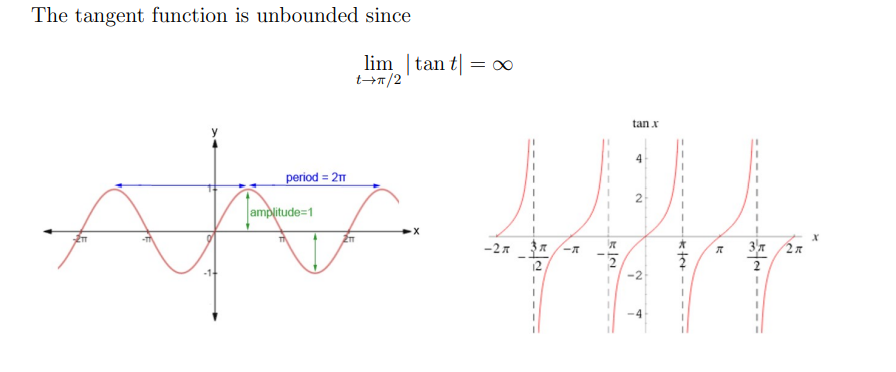
- ***Time Shift:*** Shifting the signal in time, such as y(t) = x(t - b), where 'b' is a real constant. This results in either a right shift (time delay) if 'b' is positive or a left shift (time advance) if 'b' is negative.

- ***Time Reversal***: Inverting the signal with respect to the vertical axis, achieved through y(t) = x(-t).

- ***Combination of Scale and Shift:*** Combining time scaling and time shifting, resulting in complex transformations. The order in which these transformations are applied matters, and it should be approached with caution.

***4.Bounded Functions:***

A function x is considered bounded if there exists a finite positive real constant A such that |x(t)| ≤ A for all t, indicating that x(t) remains finite for all t.



***5.Absolute Integrability:***

An absolutely integrable function is one whose absolute value is integrable, meaning that the integral of |x(t)| over its domain is finite.

***6.Energy and Power Signals:***

- \*\*Total Energy:\*\* The total energy of a continuous-time signal x(t) is given by E = ∫x^2(t)dt over the entire domain. An energy signal is a signal with finite energy.

- \*\*Average Power:\*\* The average power of a signal is determined by Pavg = lim (T → ∞) [1 / (2T) ∫x^2(t)dt]. A power signal is a signal with finite, nonzero average power. An important distinction is that energy signals have zero average power.

***7.Discrete Time Energy and Power Signals:***

Similar to continuous time, energy and power signals are defined for discrete time signals, where the sums replace integrals.

***8.Popular Elementary Signals:***

Several elementary signals commonly used in signal processing include:

- ***Sinusoidal Signals:*** These signals, characterized by A cos(ωt + ϕ), are periodic with a fundamental period of T = 2π/ω.

- ***Step Signals:*** A unit step function, denoted as u(t), is 1 for t ≥ 0 and 0 otherwise.

- ***Impulse Signals:*** In discrete time, the delta function (δ[n]) is defined as δ[n] = u[n] - u[n - 1].

This overview provides a foundational understanding of signals and their essential properties, periodicity, directionality, energy, and power, all of which play a crucial role in signal analysis and processing.

***Systems in Signal Processing: An Overview***

Systems play a crucial role in the field of signal processing, facilitating the transformation of input signals into output signals through a series of operations. These operations are often represented by mathematical operators. This overview will delve into the fundamental concepts and properties of systems, shedding light on their key characteristics.

***1. Signals as Interconnections of Operations:***

A system can be viewed as a series of interconnections of operations that process input signals to produce output signals. In continuous-time, this relationship is expressed as y(t) = H{x(t)}, while in discrete-time, it is denoted as y[n] = H{x[n]}. This mathematical representation highlights how systems operate on signals to generate new signals.

Consider an example of a discrete-time system that calculates the mean of the three most recent input values:

y[n] = 1/3(x[n] + x[n - 1] + x[n - 2])

In this case, the operator H can be described as H = 1/3(S0 + S1 + S2), where S0, S1, and S2 represent the shift operators for delays of 0, 1, and 2 time units, respectively.

***2. Properties of Systems:***

Systems possess several important properties that help in understanding and characterizing their behavior.

***2.1 Linearity:***

A system exhibits linearity if it adheres to two key principles: superposition and homogeneity.

- ***Superposition:*** This principle states that if two different inputs, x1(t) and x2(t), produce corresponding outputs y1(t) and y2(t), then the combined input x1(t) + x2(t) results in the output y1(t) + y2(t). In simpler terms, the system's response to a sum of inputs equals the sum of the system's responses to individual inputs.

- ***Homogeneity:*** A linear system satisfies the principle of homogeneity, which means that if an input is multiplied by a constant, such as ax(t), the output becomes ay(t). This property ensures that the system's behavior scales proportionally with input changes.

***Example:*** Consider a system with the input-output relationship y(t) = tx(t). This system is linear since it adheres to the principles of superposition and homogeneity.

***2.2 Stability:***

Stability in a system signifies that for any bounded input, the system produces a bounded output. A bounded input signal is one that remains finite for all time instances. Stability is a critical property in engineering as it ensures that the system does not exhibit uncontrolled or excessive responses.

***Example:*** The system y[n] = rn\*x[n] is unstable because the term rn can result in unbounded output. For stability, the term rn must be finite.

***2.3 Memory:***

A system is classified as memoryless if its output signal depends solely on the present value of the input signal. Conversely, a system is considered to have memory if its output relies on past or future values of the input signal. Memoryless systems are also known as instantaneous systems.

Example: In a system with the input-output relationship y[n] = (1/3)(x[n] + x[n - 1] + x[n - 2]), the system has memory because its output depends on past input values.

***2.4 Causality:***

Causality is a property of a system where the output response depends only on present and past values of the input signal and not on future values. Causal systems are suitable for real-time applications, as they do not rely on future information.

***Example:*** A system described by the equation y[n] = x[n] + x[n - 1] is causal, as its output depends on the present and past values of the input.

***2.5 Invertibility:***

Invertibility refers to the existence of an inverse system that can undo the effects of the original system. An invertible system can be represented as H−1, and it satisfies H−1H = I, where I is the identity operator.

***Example***: An inductor in an electrical circuit is invertible, and its inverse operation is represented by the equation v(t) = L \* (di(t)/dt).

***2.6 Time Invariance*:**

A system is time invariant if a time shift in the input signal corresponds to an identical time shift in the output signal. Time-invariant systems retain their behavior over time.

***Example:*** A system that shifts an input signal x[n] to the right by n0 units and produces an output y[n] = x[n - n0] is time invariant because the shift in input results in an equivalent shift in the output.

***3. Types of Discrete-Time Systems:***

Discrete-time systems share similarities with continuous-time systems but have their own distinct characteristics:

***3.1 Linear Systems:***

Linear systems obey the principles of superposition and homogeneity. They demonstrate that combining different inputs and scaling inputs lead to corresponding outputs.

***3.2 Shift Invariance Systems:***

A shift-invariant system remains consistent over time, and shifting the input signal results in an equivalent shift in the output signal. Time-invariant systems are also referred to as shift-invariant systems.

***3.3 Causality:***

Causal systems provide outputs based on the present and past values of the input signal, ensuring that they do not rely on future information. Causality is crucial for real-time applications.

***3.4 Stability:***

BIBO (Bounded Input, Bounded Output) stability is a fundamental property of discrete-time systems. It guarantees that the output remains bounded when the input is a bounded sequence. Stability is essential to prevent uncontrolled responses.

***In summary***, understanding the properties of systems is vital in signal processing and engineering. Linear, time-invariant, causal, and stable systems are typically preferred in practical applications due to their predictability and reliability. These properties ensure that systems can effectively process signals and produce meaningful results, making them valuable tools in various fields.

***ECG Signal: Analyzing Cardiac Activity through Signals and Systems:***

The electrocardiogram (ECG or EKG) signal is a quintessential example of real-world data that can be comprehensively analyzed and understood using concepts from signals and systems. This diagnostic tool plays a fundamental role in monitoring and assessing cardiac activity, providing valuable insights into the health of the human heart.

***Signal Representation***:

In the context of signals and systems, the ECG signal can be seen as a continuous-time signal. It is typically acquired by attaching electrodes to specific points on the body, primarily the chest, limbs, and wrist. These electrodes capture the electrical activity of the heart over time, creating a continuous waveform.

***ECG Signal Characteristics:***

***1. Periodicity and Cyclic Behavior:*** The ECG signal exhibits periodic behavior, which is fundamental to its analysis. It consists of a sequence of repetitive patterns that correspond to various phases of the cardiac cycle, namely the P-wave, QRS complex, and T-wave. This cyclic nature is akin to the periodic signals discussed in signals and systems.

***2. Linearity:*** The ECG signal is linear with respect to superposition. In clinical settings, multiple electrodes are used to record the ECG, and the resulting signals from each electrode can be linearly combined to analyze cardiac activity comprehensively. This linearity principle aligns with the linearity property discussed earlier.

***3. Stability:*** The ECG signal reflects the stability of the heart's electrical activity. A stable ECG indicates a consistent and healthy heartbeat, while instability may indicate arrhythmias or other cardiac abnormalities.

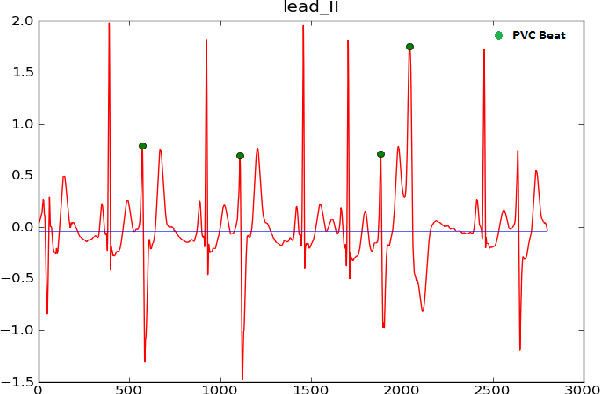
***System Characteristics in ECG Analysis:***

***1. Causality:*** The causality of an ECG analysis system is vital for real-time monitoring of cardiac activity. Causal systems ensure that the assessment of the heart's electrical activity depends only on past and present ECG signal values. This enables healthcare professionals to monitor and diagnose cardiac conditions in real-time.

***2. Time-Invariance:*** Time-invariance plays a significant role in ECG analysis systems. In practice, the delay in signal acquisition and processing is typically negligible, allowing for real-time assessments. Ensuring time-invariance in the ECG analysis process is critical for the timely detection of cardiac issues.

***Applications of ECG Signal Analysis:***

***1.Arrhythmia Detection:*** ECG signals are analyzed to detect arrhythmias, which are irregular heart rhythms. The system examines the cyclic patterns in the ECG signal to identify deviations that may indicate arrhythmias, thus fulfilling the causality and time-invariance criteria.

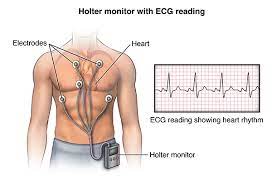


***2. Myocardial Infarction Diagnosis:*** ECG analysis is instrumental in diagnosing myocardial infarctions (heart attacks). Certain patterns and changes in the ECG signal indicate myocardial ischemia, helping healthcare providers make rapid and accurate diagnoses.

***3. Heart Rate Monitoring:*** ECG analysis provides a real-time evaluation of heart rate and rhythm. Systems analyze the intervals between successive R-waves (QRS complex) to determine heart rate and assess whether it is within the normal range.



***4.Long-Term Holter Monitoring:*** Holter monitors record ECG signals continuously over extended periods. These systems analyze the data to identify transient arrhythmias, further highlighting the importance of causality and time-invariance in detecting subtle cardiac irregularities.



***Final Remarks:***

The ECG signal is a compelling real-world application of signals and systems principles. By interpreting this continuous-time signal using concepts such as linearity, causality, time-invariance, and stability, healthcare professionals can effectively monitor and diagnose cardiac conditions. The periodic nature of the ECG signal aligns with the periodic signals discussed in signals and systems, allowing for the systematic analysis of heart rhythms and the detection of abnormalities. This real-world application showcases how theoretical concepts from signals and systems are employed to save lives and maintain cardiovascular health.

***Libraries used:***

***Libraries used in the Scorer model:***

***1. pickle:*** The `pickle` module is used to deserialize the pre-trained machine learning model that was previously saved. It loads the model from a file and prepares it for use in predicting anomalies.

***2. kafka:*** The `kafka` library is used to interact with Apache Kafka, a distributed streaming platform. In this code, it connects to a Kafka topic named 'signal' and retrieves streaming data from it to perform streaming analytics.

These libraries serve the following functions in the code:

- ***pickle:*** Loads the pre-trained anomaly detection model from a file stored at the path "C:\\signals\\data\\ecg.pickle."

- ***kafka***: Connects to a Kafka topic named 'signal' using the Kafka Consumer. It then processes streaming messages from the Kafka topic, which are expected to be ECG data. The data is decoded and split, and the anomaly detection model is used to predict whether the incoming data represents an anomaly or not. The results of the prediction are printed to the console.

The code effectively uses these libraries to perform real-time anomaly scoring on incoming ECG data from a Kafka producer. ***Libraries used in Trainer model:***

***1. pandas (imported as pd):*** Pandas is a powerful data manipulation and analysis library. It is used for reading and manipulating data from CSV files and for data analysis, such as selecting specific columns or rows.

***2. numpy (imported as np):*** NumPy is a fundamental package for scientific computing with Python. It provides support for arrays and matrices and mathematical functions to operate on them. In this code, it is used for various numerical operations and data processing.

***3. pickle:*** The `pickle` module is used for serializing and deserializing Python objects. In this code, it is used to save the trained model to a file.

***4. seaborn (imported as sns):*** Seaborn is a data visualization library based on Matplotlib. It is used to create various types of plots and visualizations to analyze data.

***5. matplotlib.pyplot (import imported as plt):*** Matplotlib is a plotting library for creating static, animated, and interactive visualizations. In this code, it is used to create and display various plots and charts.

***6. pyod.models.hbos:*** This is part of the PyOD (Python Outlier Detection) library. It is used to import the HBOS (Histogram-based Outlier Score) model, which is a method for detecting outliers in data.

***7. pyod.utils.data:*** This is part of the PyOD library and is used to generate data, evaluate model performance, and preprocess data for anomaly detection.

***8. pyod.utils.example:*** This is also part of the PyOD library and provides utility functions for examples and visualizations in PyOD.

***9. pyod.models.cblof:*** This is part of the PyOD library and is used to import the CBLOF (Cluster-Based Local Outlier Factor) model for anomaly detection.

These libraries serve different purposes, including data manipulation, numerical operations, data visualization, and anomaly detection, in the code snippet.

***Variables Used:***

***Variables used in the Scorer model:***

***1. topic:***

- Use: This variable stores the Kafka topic name, which is set to 'signal.' It specifies the Kafka topic from which the code consumes streaming data.

***2. AnamolyModel:***

- Use: This variable is used to load the pre-trained anomaly detection model. It loads the model from a file using the `pickle.load` function. The model is used to make predictions on incoming data from the Kafka topic.

***3. consumer:***

- Use: This variable represents the Kafka Consumer instance and is used to consume streaming data from the Kafka topic. It is initialized with the specified Kafka topic and server address ('localhost:9092').

***4. message:***

- Use: This variable stores each incoming message or data record from the Kafka topic. The code iterates through the messages received from the Kafka topic to process and predict anomalies.

***5. val:***

- Use: This variable is used to store the decoded and split version of the incoming message data. It splits the data by commas (',') and stores it as a list.

***6. val1:***

- Use: This variable stores the data as a list of floating-point numbers. It converts the split data elements from `val` into floating-point numbers for anomaly prediction.

***7. predicted:***

- Use: This variable stores the anomaly prediction result obtained by applying the loaded `AnamolyModel` to the input data stored in `val1`. It represents whether the incoming data is classified as an anomaly (anomalous) or not (normal).

These variables collectively enable the code to consume streaming ECG data from Kafka, process the data, make real-time anomaly predictions using the pre-trained model, and print the results to the console.

***Variables used in Trainer model:***

***1. df:***

- Use: This variable stores the ECG data loaded from a CSV file using the pandas library. It represents the dataset for the anomaly detection training.

***2. df\_numerics\_only:***

- Use: This variable contains only the numeric variables from the dataset. It is used to isolate and identify the numerical features for analysis.

***3. NumDataType:***

- Use: This list stores data types that are considered numeric (e.g., integers and floats). It is used in the process of identifying numeric variables.

***4. df\_categorical\_only:***

- Use: This variable contains only the categorical variables from the dataset. It is used to isolate and identify the categorical features for analysis.

***5. Corr\_Num\_Matix:***

- Use: This variable stores the correlation matrix of numeric variables. It is used for bivariate analysis to understand the relationships between numeric features.

***6. outliers:***

- Use: This variable stores the data points identified as anomalies by the SignalAnomolyModel. It contains the records classified as anomalies based on the model's predictions.

***7. outlier\_index:***

- Use: This variable is a list that stores the indices of the data points classified as anomalies by the SignalAnomolyModel.

***8. SignalAnomolyModel:***

- Use: This variable represents the HBOS (Histogram-Based Outlier Score) model for anomaly detection. It is initialized with a contamination parameter of 0.35 to specify the expected fraction of anomalies in the dataset.

These variables are essential for data preparation, analysis, and anomaly detection within the code snippet. They help in loading and processing the ECG data, performing univariate and bivariate analysis, building the anomaly detection model, and identifying anomalous data points in the dataset.

***Source Code:***

***source code for trainer model:***

|  |
| --- |
| #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ {  # Name : signals\_Anomaly\_Trainer\_HB  # Reviewer : Dr.Manish Bhatt Sir  # Date : 29-Oct-2023  # Purpose : Signals Anomaly Detection Training  # Data : ECG Data  # :  # :  # Change History :  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Library s Begins ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  **import** **pandas** **as** **pd**  **import** **numpy** **as** **np**  **import** **pickle**  **import** **seaborn** **as** **sns**  **import** **matplotlib.pyplot** **as** **plt**  **from** **pyod.models.hbos** **import** HBOS  **from** **pyod.utils.data** **import** generate\_data  **from** **pyod.utils.data** **import** evaluate\_print  **from** **pyod.utils.example** **import** visualize  **from** **pyod.models.cblof** **import** CBLOF  #Read Data  df=pd.read\_csv("C:**\\**signals**\\**data**\\**\ecg.csv")  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#  # \* 01 Variable Identification \*  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#  df\_numerics\_only = df.select\_dtypes(include=[np.number]) #Numeric Variable Identification  NumDataType = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']  df\_categorical\_only = df.select\_dtypes(exclude=NumDataType) #Categorical Variable Identification  **print**(df\_numerics\_only.columns)  **print**(df\_categorical\_only.columns)  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#  # \* 02 Data Approach \*  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#  **print**()  **print**("Data Set (Rows, Cols)")  **print**(". . . . . . . . . . . ")  **print**(df.shape)  **print**()  **print**("df\_numerics\_only (Rows, Cols)")  **print**(". . . . . . . . . . . . . . .")  **print**(df\_numerics\_only.shape)  **print**()  **print**("df\_categorical\_only (Rows, Cols)")  **print**(". . . . . . . . . . . . . . .")  **print**(df\_categorical\_only.shape)  **print**()  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#  # \* 04 Bi-variate Analysis \*  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#  **def** **get\_redundant\_pairs**(df):  '''Get diagonal and lower triangular pairs of correlation matrix'''  pairs\_to\_drop = set()  cols = df.columns  **for** i **in** range(**0**, df.shape[**1**]):  **for** j **in** range(**0**, i+**1**):  pairs\_to\_drop.add((cols[i], cols[j]))  **return** pairs\_to\_drop  **def** **get\_top\_abs\_correlations**(df, n=**5**):  au\_corr = df.corr().abs().unstack()  labels\_to\_drop = get\_redundant\_pairs(df)  au\_corr = au\_corr.drop(labels=labels\_to\_drop).sort\_values(ascending=False)  **return** au\_corr[**0**:n]  #Continous and Continous  Corr\_Num\_Matix = df\_numerics\_only.corr().round(**3**)  **print**("~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~~ ~ ~ ~ ~ ")  **print**(Corr\_Num\_Matix)  **print**("~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~~ ~ ~ ~ ~ ")  **print**(". . . . . . . . . . . . . . . . . . Top Most Correlations . . . . . . . . . . . . . . . . . ")  **print**(get\_top\_abs\_correlations(df\_numerics\_only, **150**))  **print**()  # #Categorical and Target(Categorical)  # for column in list(df\_categorical\_only):  # print(column)  # grouped = dataset.groupby(['SFPA\_color', column])  # print(grouped.size())  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#  # \* 03 Univariate Analysis \*  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#  **print**(". . . . . . . . . . . . . . . . . . Central Tendancy Summary . . . . . . . . . . . . . . . . . ")  **print**(df\_numerics\_only.describe().transpose())  tmp= df\_numerics\_only.describe().transpose()  **print**()  **print**(". . . . . . . . . . . . . . . . . . Numerical Missing Values . . . . . . . . . . . . . . . . . .")  **print**(len(df\_numerics\_only) - df\_numerics\_only.count())  **print**()  **print**(". . . . . . . . . . . . . . . . . . Categorical Missing Values . . . . . . . . . . . . . . . . . ")  **print**(len(df\_categorical\_only) - df\_categorical\_only.count())  **print**()  # Numerical Variable Analysis` \*  **for** column **in** list(df\_numerics\_only):  **print**(column)  sns.violinplot(df\_numerics\_only[column], linewidth=**5**, orient='h')  #sns.distplot(df\_numerics\_only[column])  plt.show(**6**)  # Categorical Variable Analysis` \*  df\_categorical\_only.fillna("NaN") #correct Missing Values for Distribution view  **print**(". . . . . . . . . . . . . . . . . . Frequency Table . . . . . . . . . . . . . . . . . ")  **print**(df\_categorical\_only.apply(**lambda** x: x.value\_counts()).T.stack())  # Distribution Plot  **for** column **in** list(df\_categorical\_only):  sns.countplot(x=column, data=df\_categorical\_only)  plt.show(**10**)  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#  # \* Train / Test Split \*  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#  train,test = df[**1**:**4500**], df[**4501**:**4996**]  **print**(train.shape)  **print**(test.shape)  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#  # \* Model Building \*  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#  SignalAnomolyModel = HBOS( contamination =.**35**)  SignalAnomolyModel.fit(train)  pred = SignalAnomolyModel.predict(test)  test['anomaly']=pred  outliers=test.loc[test['anomaly']==-**1**]  outlier\_index=list(outliers.index)  #Find the number of anomalies and normal points here points classified -1 are anomalous  **print**(test['anomaly'].value\_counts())  pickle.dump(SignalAnomolyModel,open("C:**\\**signals**\\**data**\\**\ecg.pickle", 'wb'))  **print**("Training Finished!") |

***Source code for scorer:***

|  |
| --- |
| *#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ {*  *# Name : signals\_Anomaly\_Scorer*  *# Authour : K. SrePadmashiny*  *# Reviewer : Signals Professor*  *# Date : 31-Oct-2023*  *# Purpose : Signals Anomoly Detection Scoring*  *# Data : ECG Data from ecg kafka producer*  *# :*  *# :*  *# Change History :*  *#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Library s Begins ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*  *# import required libraries*  import pickle  from kafka import KafkaConsumer  *# Kafka topics*  topic = 'signal'  *#Loading model*  print("Loading pre-trained model")  AnamolyModel = pickle.load(open("C:\\signals\\data\\\ecg.pickle", 'rb'))  *#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#*  *# \* predicting the streaming kafka messages \**  *#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#*  consumer = KafkaConsumer('signal',bootstrap\_servers=['localhost:9092'])  print("Starting ML predictions.")  for message in consumer:  val = ((message.value).decode("utf-8")).split (",")  val1 = [float(i) for i in val]  predicted = AnamolyModel.predict([val1])  print(predicted[0])  print(message.value.decode("utf-8") +" => " + str(predicted[0])) |